cadence®

NAND Flash Memory Controller

Internal Name: HPNFC
Part Number: IP6018

NDA # 0
User Guide
Revision: 1.08

Confidentiality Notice

Cadence Design Systems, Inc. San Jose, CA 95134

Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA.

Trademarks

Restricted Permission

Disclaimer

Restricted Rights

Table of Contents

List of Figures

List of Tables

Preface

This chapter provides a general introduction to this manual, and contains the following sections:

1. About This Document

This manual is intended for chip designers who will use the NAND Flash Memory Controller in their application. Readers of this document should be familiar with NAND Flash Devices specifications.

2. Organization of This Document

This NAND Flash Design Specification was divided into two parts. The first part contains controller behavior description. It can be treated as a User Guide for the NAND Flash Controller.

This manual contains the following chapters:

Preface	Describes the manual and lists the symbols used and provides information on how to get technical assistance.
Chapter 1, *General Information*	This chapter provides general information about controller architecture and features.
Chapter 2, *Command Engine*	This chapter describes the controller application interface as viewed by software engineers. It provides detailed description of two controller work modes: PIO and CMD_DMA.
Chapter 3, *ECC Engine*	Explains how to use the embedded ECC engine.
Chapter 4, *Booting Mechanism*	This chapter describes a booting mechanism implemented in the NAND Flash Controller.
Chapter 5, *Programming Specification*	Provides guidelines on how to program/monitor Cadence NAND Flash Memory Controller to achieve the best performance.
Chapter 6, *Hardware Implementation Requirements*	This chapter is a guideline about hardware requirements for a client application that uses the NAND Flash Memory Controller IP.
Chapter 7, *Special Function Registers*	Lists the registers available to software. This chapter contains a detailed register layout.

3. Acronyms and Abbreviations

The following acronyms and abbreviations are used in this manual:

Table 1. Acronyms and Abbreviations

Term	Meaning
SLC	Single Level Cell
pSLC	pseudo-SLC
MLC	Multiple Level Cell
ECC	Error Correction Code
DMA	Direct Memory Access
MSB	Most Significant Byte
LSB	Least Significant Byte
R/W	Read / Write (i.e. register access type)
W	Write Only (i.e. register access type)
R	Read Only (i.e. register access type)
W1C	Write logic 1 to clear (i.e. register access type)
CRC	Cycle Redundancy Check
FIFO	First In First Out
AMBA	Advanced Microcontroller Bus Architecture
AXI	Advanced eXtensible Interface
ONFi	Open NAND Flash Interface
PHY	Physical Layer of the OSI Model
RTL	Register Transfer Level
PIO	Processor Input Output
CT	Command Type. Field of command word that identifies command type: 00 - CDMA command, 01 - PIO command, 10 - special command. Other values are reserved.
BANK	Single CE line on the NAND Flash interface
LUN	The minimum unit that can independently execute commands and report status. There are one or more LUN-s per NAND Target.
Target	A set of LUN that shares one CE signal within one NAND package. When CE pin reduction is enabled then it will be set of LUN that share the same volume id.
Volume	A Volume is an appointed address to a NAND Target.

4. Getting Help

If you have any problems with using this product or understanding the documentation, you can submit a Service Request (SR) to Cadence Support. When doing so, you must provide enough information about the problem so that it can be investigated efficiently. Describe the problem in detail, provide the version of the software you are using, and state the exact circumstances in which the problem occurs.

If you have a question about using Cadence products, please refer to product documentation which is installed on your network. You can also access documentation for NAND Flash Memory Controller products on the Web:

- Product documentation
- Data sheets

- License information

4.1. Service Requests

Service Requests are your way of giving feedback, asking questions, getting solutions, and reporting problems. Unless told otherwise, Cadence support staff will respond to your service request. If Cadence Support cannot answer your question, Cadence Research and Development personnel will get involved. It is important to specify the severity level of the service request as accurately as possible.

There are three levels of severity:

- Critical — You cannot proceed without a solution to the issue.
- Important — You can proceed, but you need a solution to the issue.
- Minor — You prefer to have a solution, but you can wait for it.

Note

You can request support to increase the severity level of an issue. Therefore, do not use Critical unless immediate resolution of an issue is absolutely necessary and urgently required.

4.2. Using Cadence On-line Support

Cadence encourages you to submit requests using Cadence On-line Support. With Cadence On-line Support you can also track your open service requests.

To use Cadence On-line Support please go to http://support.cadence.com and proceed with instructions found on this web page.

5. Additional information

For more information on Cadence and its products, please visit: http://www.cadence.com

1. General Information

The following topics are discussed in this chapter:

1.1. Features

The NAND Flash Memory Controller provides the features described below to maximize the system-level performance and provide the most feature-rich and most flexible NAND Flash solution that enables enterprise-class storage and embedded memory application.

The main features of Cadence NAND Flash Memory Controller (for the configuration described in this document):

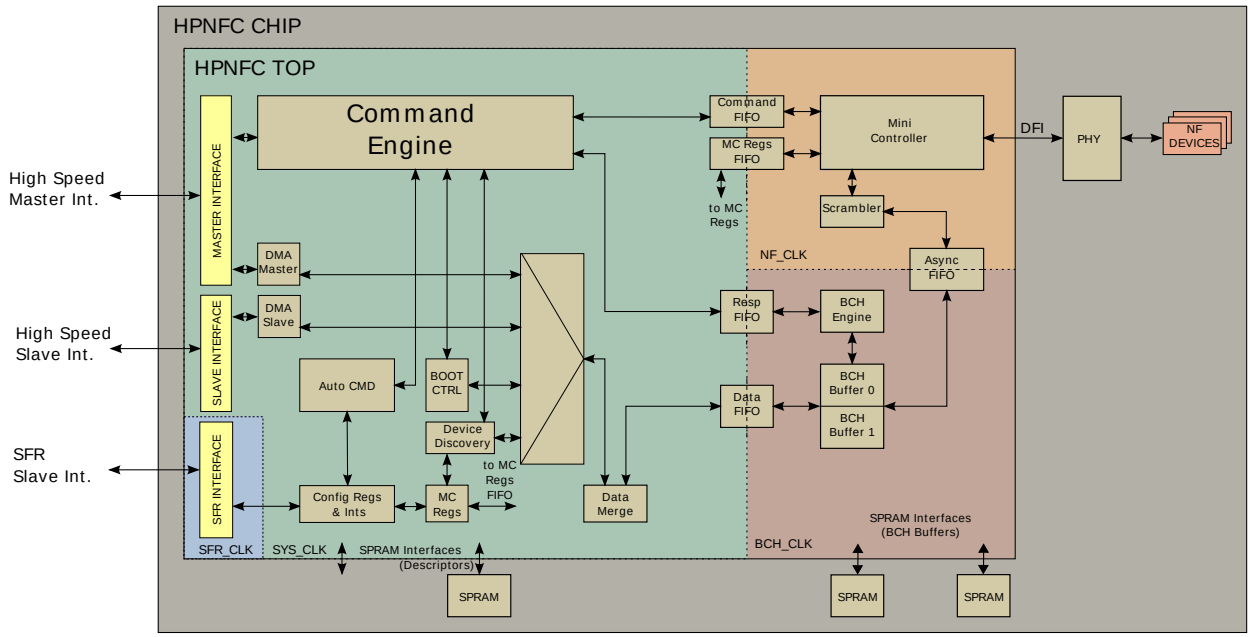
- Supported system interfaces:
  - Controller master DMA interface uses an AXI4 system bus
  - Controller slave DMA interface uses an AXI4 system bus
  - Controller status/control interface uses an AXI4 Lite system bus
- Compatibility:
  - Is compatible with the ONFI 1.x specification
  - Controller support only legacy devices that uses five cycle addressing scheme (3 row address bytes and 2 column address bytes).The flash devices with 2 row address cycles supported only when are able to ignores third row address cycle
- Interrupt Controller:
  - Each interrupt can be masked
  - Each interrupt has its own status flag
  - The status flags are also valid when the given interrupt is masked, and can be checked by the software polling mechanism
  - Common interrupt port is provided for all interrupt sources
- Standard interface pin labeling
- Full duplex asynchronous FIFO is used to synchronize clock domain and adjust data path. This mechanism is used for clock domain synchronization between system clock domain (SYS_CLK) and NAND Flash memory clock domain (NF_CLK) and between NAND Flash memory clock domain (NF_CLK) and BCH engine clock domain (BCH_CLK)
- Four phase synchronization mechanism used between Low Speed Slave interface clock domain (SFR_CLK) and System clock domain (SYS_CLK)
- Implementation of the boot sequence
- Separate port interface for data and control/status registers

- Support for up to two banks with up to eight targets per bank
- Support for volume addressing. Up to sixteen volumes supported.
- Support for pipeline read and write commands for maximum data throughput
- Programmable access timing
- Intelligent hardware abstraction layer to off-load the processor as well as to provide direct data and control paths to the device
- Controller support only devices that has three bytes row address.
- Controller has debug interface that allow to access predefined set of internal signals. Values of those signal can be used to determine controller internal state outside the functional simulation.
- For the legacy devices controller provide basic interface to provide the read/program/erase operation. If device uses different interface for the cache, multiplane or multi-LUN operation then one implemented in the controller and present in current devices implementation then this interface will not be supported.

1.2. Controller Block Diagram

Figure 1.1 shows a block diagram of the NAND Flash Memory Controller.

Figure 1.1. Cadence NAND Flash Memory Controller Architecture



Brief submodules description:

- The low-speed slave interface - It provides access to controllers status and configuration registers. This module translates transactions in AXI format into the transactions in the OCP format. The low-speed slave interface implements AXI Lite specification. Implementation details are described in Section 6.1, *"System Interface"* of the Chapter 6, *Hardware Implementation Requirements*.

- The Controllers high-speed slave interface - The interface logic receives the incoming data transactions from the host interface and passes them onto the Slave DMA module. This path is optimized for data throughput. Implementation details are described in Section 6.1, "*System Interface*" of the Chapter 6, *Hardware Implementation Requirements*.
- The high-speed master interface - It is used for two purposes. The Command Engine unit uses this interface to handle descriptors stored in the system memory and the DMA master uses this interface to transfer data between core internal buffer and system memory. The DMA Master data path is optimized for data throughput. Implementation details are described in Section 6.1, "*System Interface*" of the Chapter 6, *Hardware Implementation Requirements*.
- Command Engine - This module implements high-level ONFI protocols functionality and multi thread logic. It translates high-level commands from Auto CMD module into the series of low-level commands that will be sent to the NAND Flash interface. This module has embedded protocol engine that selects the most appropriate set of features of the used NAND Flash device relieving software engineer from knowing NAND Flash protocol details. More information on how to use this module can be found in Chapter 2, *Command Engine*.
- DMA Master - This module is used to automatically transfer data from the NAND Flash controller internal buffer to the system memory. It supports outstanding transactions (up to 16) and incremental bursts (up to 256 beats). More information on how to use this module can be found in the Chapter 2, *Command Engine*.
- DMA Slave - This module provides high performance slave interface to the controller's internal data buffer. It supports outstanding transactions (up to 16) and incremental bursts (up to 256 beats). More information on how to use this module can be found in the Chapter 2, *Command Engine*.
- Boot CTRL module - This module implements boot from the NAND Flash memory functionality. It allows the use of single non volatile memory in the system. More information on how to use this functionality can be found in the Chapter 4, *Booting Mechanism*.
- Configuration Registers and Interrupts - This module stores control and status registers. Additionally it is responsible for interrupt handling. The register map is described in the Chapter 7, *Special Function Registers*.
- Auto CMD - This module translates write operation to the Command registers (C0 - C3) into the record write operations to the context memory of the Command Engine. Each controller's work mode uses command registers.
- MC Regs - This module maintains accessing registers stored in NF_CLK clock domain.
- Asynchronous FIFO-s - The FIFO-s are used as clock domain crossing mechanism between the system clock domain (SYS_CLK) and the Flash controller clock domain (NF_CLK), as well as between the Flash controller clock domain (NF_CLK) and BCH engine clock domain (BCH_CLK).
- BCH Engine and BCH Buffer - This module provides error detection and correction mechanism in the data path. During write operation, the BCH calculates ECC checksums and places them into the data stream that is written to the NAND Flash memory. During read operation, ECC previously written ECC checksums are extracted and BCH engine checks read data integrity and corrects errors if detected. The BCH engine configuration and usage is described in the Chapter 3, *ECC Engine*.
- Mini Controller module - This module gets a command stream from the Command Engine and translates it into the physical operation on the PHY/NAND Flash interface. Additionally, the Mini Controller module regulates data flow between the PHY/NAND interface and the system interface.
- PHY module - This module provides additional logic which operates as a bridge between DFI and standard Nand Flash interface.

1.3. Pin Index

The NAND Flash Memory Controller supports the following signals:

- Internal host interface

- External PHY interface
- IP interface port sideband and miscellaneous signals
- IP SRAM interface signals

1.3.1. Internal Host Interface

Table 1.1. AXI4 Master DMA Interface

Signal Name	Size	Type	Description	Reset Value	Associated clock
mACLK	1	Input	Global clock signal. All signals are sampled on the rising edge of the global clock	N/A	sys_clk
mARESETn	1	Input	Global reset signal. This signal is active LOW	N/A	sys_clk
mAWADDR	64	Output	Write address. The write address bus gives the address of the first transfer in a write burst transaction. The associated control signals are used to determine the addresses of the remaining transfers in the burst.	64'h0	sys_clk
mAWLEN	8	Output	Burst length. The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address.	8'hff	sys_clk
mAWSIZE	3	Output	Bytes in transfer. Only transfer size equal to the complete data width is supported.	3'h3	sys_clk
mAWBURST	2	Output	Write burst type. Only incremental (2'b01) is supported.	2'h1	sys_clk
mAWPROT	3	Output	Protection type. Controller does not use this signal internally.	3'h2	sys_clk
mAWVALID	1	Output	Write address valid. This signal indicates that valid write address and control information are available: 1 = address and control information available, 0 = address and control information not available. The address and control information remain stable until the address acknowledge signal, AWREADY, goes HIGH.	1'h0	sys_clk
mAWREADY	1	Input	Write address ready. This signal indicates that the slave is ready to accept an address and associated control signals: 1 = slave ready, 0 = slave not ready.	N/A	sys_clk
mWDATA	64	Output	Write data bus.	64'h0	sys_clk
mWSTRB	8	Output	Write strobes. Byte lane selection is not supported. So only all 1's is allowed.	8'hff	sys_clk
mWLAST	1	Output	Write last. This signal indicates the last transfer in a write burst	1'h0	sys_clk
mWVALID	1	Output	Write valid. This signal indicates that valid write data and strobes are available: 1 = write data and strobes available, 0 = write data and strobes not available.	1'h0	sys_clk
mWREADY	1	Input	Write ready. This signal indicates that the slave can accept the write data: 1 = slave ready, 0 = slave not ready.	N/A	sys_clk
mBRESP	2	Input	Write response.	N/A	sys_clk

Signal Name	Size	Type	Description	Reset Value	Associated clock
mBVALID	1	Input	Write response valid. This signal indicates that a valid write response is available: 1 = write response available, 0 = write response not available.	N/A	sys_clk
mBREADY	1	Output	Response ready. This signal indicates that the master can accept the response information. 1 = master ready, 0 = master not ready.	1'h0	sys_clk
mARADDR	64	Output	Read address. The read address bus gives the initial address of a read burst transaction. Only the start address of the burst is provided and the control signals that are issued alongside the address detail how the address is calculated for the remaining transfers in the burst.	64'h0	sys_clk
mARLEN	8	Output	Burst length. The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address.	8'hff	sys_clk
mARSIZE	3	Output	Bytes in transfer. Only transfer size equal to the complete data width is supported.	3'h3	sys_clk
mARBURST	2	Output	Read burst type. Only incremental (2'b01) is supported.	2'h1	sys_clk
mARPROT	3	Output	Protection type. Controller does not use this signal internally.	3'h2	sys_clk
mARVALID	1	Output	Read address valid. This signal indicates, when HIGH, that the read address and control information is valid and will remain stable until the address acknowledge signal, ARREADY, is high. 1 = address and control information valid, 0 = address and control information not valid.	1'h0	sys_clk
mARREADY	1	Input	Read address ready. This signal indicates that the slave is ready to accept an address and associated control signals: 1 = slave ready, 0 = slave not ready.	N/A	sys_clk
mRDATA	64	Input	Read data bus.	N/A	sys_clk
mRRESP	2	Input	Read response.	N/A	sys_clk
mRLAST	1	Input	Read last. This signal indicates the last transfer in a read burst	N/A	sys_clk
mRVALID	1	Input	Read valid. This signal indicates that the required read data is available and the read transfer can complete: 1 = read data available, 0 = read data not available.	N/A	sys_clk
mRREADY	1	Output	Read ready. This signal indicates that the master can accept the read data and response information: 1= master ready, 0 = master not ready.	1'h1	sys_clk

Table 1.2. AXI4 Slave DMA Interface

Signal Name	Size	Type	Description	Reset Value	Associated clock
sAWID	20	Input	Write address ID. This signal is the identification tag for the write address group of signals.	N/A	sys_clk
sAWADDR	64	Input	Write address. Not used by the controller.	N/A	sys_clk

Signal Name	Size	Type	Description	Reset Value	Associated clock
sAWLEN	8	Input	Burst length. The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address.	N/A	sys_clk
sAWSIZE	3	Input	Bytes in transfer. Only transfer size equal to the complete data width is supported.	N/A	sys_clk
sAWBURST	2	Input	Write burst type. Only incremental (2'b01) is supported.	N/A	sys_clk
sAWVALID	1	Input	Write address valid. This signal indicates that valid write address and control information are available: 1 = address and control information available, 0 = address and control information not available. The address and control information remain stable until the address acknowledge signal, AWREADY, goes HIGH.	N/A	sys_clk
sAWREADY	1	Output	Write address ready. This signal indicates that the slave is ready to accept an address and associated control signals: 1 = slave ready, 0 = slave not ready.	1'h0	sys_clk
sWDATA	64	Input	Write data bus.	N/A	sys_clk
sWSTRB	8	Input	Write strobes. Byte lane selection is not supported. So only all 1's is allowed.	N/A	sys_clk
sWLAST	1	Input	Write last. This signal indicates the last transfer in a write burst	N/A	sys_clk
sWVALID	1	Input	Write valid. This signal indicates that valid write data and strobes are available: 1 = write data and strobes available, 0 = write data and strobes not available.	N/A	sys_clk
sWREADY	1	Output	Write ready. This signal indicates that the slave can accept the write data: 1 = slave ready, 0 = slave not ready.	1'h1	sys_clk
sBID	20	Output	Response ID. The identification tag of the write response. The BID value must match the AWID value of the write transaction to which the slave is responding.	20'h0	sys_clk
sBRESP	2	Output	Write response.	2'h0	sys_clk
sBVALID	1	Output	Write response valid. This signal indicates that a valid write response is available: 1 = write response available, 0 = write response not available.	1'h0	sys_clk
sBREADY	1	Input	Response ready. This signal indicates that the master can accept the response information. 1 = master ready, 0 = master not ready.	N/A	sys_clk
sARID	20	Input	Read address ID. This signal is the identification tag for the read address group of signals.	N/A	sys_clk
sARADDR	64	Input	Read address. Not used by the controller.	N/A	sys_clk
sARLEN	8	Input	Burst length. The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address.	N/A	sys_clk
sARSIZE	3	Input	Bytes in transfer. Only transfer size equal to the complete data width is supported.	N/A	sys_clk

Signal Name	Size	Type	Description	Reset Value	Associated clock
sARBURST	2	Input	Read burst type. Only incremental (2'b01) is supported.	N/A	sys_clk
sARVALID	1	Input	Read address valid. This signal indicates, when HIGH, that the read address and control information is valid and will remain stable until the address acknowledge signal, ARREADY, is high. 1 = address and control information valid, 0 = address and control information not valid.	N/A	sys_clk
sARREADY	1	Output	Read address ready. This signal indicates that the slave is ready to accept an address and associated control signals: 1 = slave ready, 0 = slave not ready.	1'h0	sys_clk
sRID	20	Output	Read ID tag. This signal is the ID tag of the read data group of signals. The RID value is generated by the slave and must match the ARID value of the read transaction to which it is responding.	20'h0	sys_clk
sRDATA	64	Output	Read data bus.	64'h0	sys_clk
sRRESP	2	Output	Read response.	2'h0	sys_clk
sRLAST	1	Output	Read last. This signal indicates the last transfer in a read burst	1'h0	sys_clk
sRVALID	1	Output	Read valid. This signal indicates that the required read data is available and the read transfer can complete: 1 = read data available, 0 = read data not available.	1'h0	sys_clk
sRREADY	1	Input	Read ready. This signal indicates that the master can accept the read data and response information: 1= master ready, 0 = master not ready.	N/A	sys_clk

Table 1.3. AXI4 Lite Register Slave Interface

Signal Name	Size	Type	Description	Reset Value	Associated clock
regACLK	1	Input	Global clock signal. All signals are sampled on the rising edge of the global clock	N/A	sfr_clk
regARESETn	1	Input	Global reset signal. This signal is active LOW	N/A	sfr_clk
regAWADDR	32	Input	Write address. The write address bus gives the address of the first transfer in a write burst transaction. The associated control signals are used to determine the addresses of the remaining transfers in the burst.	N/A	sfr_clk
regAWVALID	1	Input	Write address valid. This signal indicates that valid write address and control information are available: 1 = address and control information available, 0 = address and control information not available. The address and control information remain stable until the address acknowledge signal, AWREADY, goes HIGH.	N/A	sfr_clk
regAWREADY	1	Output	Write address ready. This signal indicates that the slave is ready to accept an address and associated control signals: 1 = slave ready, 0 = slave not ready.	1'h0	sfr_clk
regWDATA	32	Input	Write data bus.	N/A	sfr_clk

Signal Name	Size	Type	Description	Reset Value	Associated clock
regWSTRB	4	Input	Write strobes. Byte lane selection is not supported. So only 4'b1111 is allowed.	N/A	sfr_clk
regWVALID	1	Input	Write valid. This signal indicates that valid write data and strobes are available: 1 = write data and strobes available, 0 = write data and strobes not available.	N/A	sfr_clk
regWREADY	1	Output	Write ready. This signal indicates that the slave can accept the write data: 1 = slave ready, 0 = slave not ready.	1'h0	sfr_clk
regBRESP	2	Output	Write response.	2'h0	sfr_clk
regBVALID	1	Output	Write response valid. This signal indicates that a valid write response is available: 1 = write response available, 0 = write response not available.	1'h0	sfr_clk
regBREADY	1	Input	Response ready. This signal indicates that the master can accept the response information. 1 = master ready, 0 = master not ready.	N/A	sfr_clk
regARADDR	32	Input	Read address. The read address bus gives the initial address of a read burst transaction. Only the start address of the burst is provided and the control signals that are issued alongside the address detail how the address is calculated for the remaining transfers in the burst.	N/A	sfr_clk
regARVALID	1	Input	Read address valid. This signal indicates, when HIGH, that the read address and control information is valid and will remain stable until the address acknowledge signal, ARREADY, is high. 1 = address and control information valid, 0 = address and control information not valid.	N/A	sfr_clk
regARREADY	1	Output	Read address ready. This signal indicates that the slave is ready to accept an address and associated control signals: 1 = slave ready, 0 = slave not ready.	1'h0	sfr_clk
regRDATA	32	Output	Read data bus.	32'h0	sfr_clk
regRRESP	2	Output	Read response.	2'h0	sfr_clk
regRVALID	1	Output	Read valid. This signal indicates that the required read data is available and the read transfer can complete: 1 = read data available, 0 = read data not available.	1'h0	sfr_clk
regRREADY	1	Input	Read ready. This signal indicates that the master can accept the read data and response information: 1= master ready, 0 = master not ready.	N/A	sfr_clk

1.3.2. DFI Interface

Cadence NAND Flash Controller interfaces to the PHY for NAND Flash devices.

Table 1.4. DFI Interface

Signal Name	Size	Type	Description	Reset Value	Associated clock
dfi_cebar	2	Output	Chip enable	2'h3	nf_clk

Signal Name	Size	Type	Description	Reset Value	Associated clock
dfi_rbn	2	Input	Ready/Busy# signal	N/A	nf_clk
dfi_ale	1	Output	Address latch enable	1'h0	nf_clk
dfi_cle	1	Output	Command latch enable	1'h0	nf_clk
dfi_rebar	1	Output	Read enable	1'h1	nf_clk
dfi_webar	1	Output	Write enable	1'h1	nf_clk
dfi_wpbar	1	Output	Write protect	1'h0	nf_clk
dfi_wrdata_en	1	Output	Write data-in valid	1'h0	nf_clk
dfi_wrdata	16	Output	Write data to the PHY	16'h0	nf_clk
dfi_rddata	16	Input	Read data from the PHY.	N/A	nf_clk
dfi_rddata_valid	1	Input	Read data from the PHY valid indicator.	N/A	nf_clk

1.3.3. Boot Interface

Cadence NAND Flash Controller interface utilized to perform automatic boot operation.

Table 1.5. Boot Interface

Signal Name	Size	Type	Description	Reset Value	Associated clock
boot_en	1	Input	If set, this pin will trigger automatic boot process. This signal must be stable and have proper value by the time the Controller comes out of reset.	N/A	sys_clk
boot_comp	1	Output	This pin will be set after automatic boot process sequence is finished.	1'h0	sys_clk
boot_error	1	Output	This pin will be set when the boot sequence will be interrupted by errors.	1'h0	sys_clk
boot_ecc_enable	1	Input	Controller initialization parameter for the boot process. Specifies if boot image contains ECC metadata.	N/A	sys_clk
boot_ecc_corr_str	3	Input	Controller initialization parameter for the boot process. Specifies the ECC correction strength if ECC is enabled for boot operation.	N/A	sys_clk
boot_sec_size	16	Input	Controller initialization parameter for the boot process. Specifies the ECC sector size if ECC is enabled for boot operation.	N/A	sys_clk

1.3.4. IP Interface Port Sideband and Miscellaneous Signals

The following table *IP Interface Port Sideband Signals* lists additional signals that are required to connect to Cadence NAND Flash Memory Controller.

Table 1.6.

Signal Name	Size	Type	Description	Reset Value	Associated clock
nf_clk	1	Input	The clock signal for NAND Flash clock domain.	N/A	nf_clk

Signal Name	Size	Type	Description	Reset Value	Associated clock
nf_rst_n	1	Input	The reset signal for NAND Flash clock domain.	N/A	nf_clk
nf_reg_rst_n	1	Input	The register reset signal for NAND Flash clock domain.	N/A	nf_clk
bch_clk	1	Input	The clock signal for BCH engine clock domain.	N/A	bch_clk
bch_rst_n	1	Input	The reset signal for BCH engine clock domain.	N/A	bch_clk
reg_rst_n	1	Input	The reset signal for controller registers in sys_clk clock domain.	N/A	sys_clk
prot_rst_n	1	Input	The reset signal for registers responsible for holding settings of the protect mechanism.	N/A	sys_clk
interrupt	1	Output	External interrupt pin.	1'h0	sys_clk
discovery_inhibit	1	Input	Bootstrap port to inhibit Controller from any initialization. Controller will not make device discovery process. This signal must be stable and have proper value by the time the Controller comes out of reset.	N/A	sys_clk
discovery_ignore_crc	1	Input	When tied to 1, the controller will ignore CRC checking after reading of parameter page during device discovery process.	N/A	sys_clk
rb_valid_time	16	Input	The PHY initialization parameter for device discovery process. Value of this parameter should be calculated as: RB_VALID_TIME = Trb [us] * fsys [MHz], where: Trb - value of the "RB_valid_Vcc" time (from NAND flash device specification) fsys - frequency of the system clock (connected to the mCLK pin).	N/A	sys_clk
dd_req	1	Output	Device Discovery external parameters request.	1'h0	sys_clk
dd_id_value	40	Output	ReadID value read from NandFlash device.	40'h0	sys_clk
dd_ack	1	Input	Device Discovery external parameters acknowledge signal.	N/A	sys_clk
dd_page_size	16	Input	Device Discovery external parameters - number of bytes in single NandFlash page.	N/A	sys_clk
dd_pages_per_block	16	Input	Device Discovery external parameters - number of pages in single block.	N/A	sys_clk
dd_lun_number	4	Input	Device Discovery external parameters - number of LUNs in single NandFlash device.	N/A	sys_clk
dd_four_addr_cycles_en	1	Input	Device Discovery external parameters - device with 4 address cycles connected.	N/A	sys_clk
dd_support_16_bit	1	Input	Device Discovery external parameters - 16b NF device connected.	N/A	sys_clk
init_comp	1	Output	Initialization complete. When device discovery process is finished this bit will be set.	1'h0	sys_clk
init_fail	1	Output	Initialization fail (valid when init_comp = 1). This bit will be set if device discovery process failed to recognize device connected to bank 0.	1'h0	sys_clk
ctrl_busy	1	Output	Signal indicating if the Flash Controller is idle or not.	1'h0	sys_clk

Signal Name	Size	Type	Description	Reset Value	Associated clock
			1 - Controller is busy 0 - Controller is idle This signal can be used by host for active clock management to Controller. Before clock can be switched off, the automatic ZQ calibration needs to be disabled.		
wre_prot_en_0	1	Input	Write protect enable signal for registers. Setting 1 on this pin will enable blocking access to the Write Protect registers with suffix _0.	N/A	sys_clk
wre_prot_en_1	1	Input	Write protect enable signal for registers. Setting 1 on this pin will enable blocking access to the Write Protect registers with suffix _1.	N/A	sys_clk

1.3.5. Debug interface

This interface allow to access set of controller internal signals grouped together to help debuging issues that can be encountered during integration in final application at stage when functional simulation is no longer possible. The describe set of signals allow accessing debuging information using controllers ports.

Table 1.7. Debug Interface

Signal Name	Size	Type	Description	Reset Value	Associated clock
dbg_n_latch_req	1	Input	Request signal for debug data from the NF_CLK clock domain. It is only valid in the debug interface work mode that provide clock domain synchronization for debug data. In work mode without synchronization should be strapped to logical zero.	N/A	sys_clk
dbg_n_latch_ack	1	Output	Acknowledge signal that strobe debug data from the NF_CLK returned as response to the request driven on the dbg_n_latch_req controller port. When inactive it will be cleared.	1'h0	sys_clk
dbg_b_latch_req	1	Input	Request signal for debug data from the BCH_CLK clock domain. It is only valid in the debug interface work mode that provide clock domain synchronization for debug data. In work mode without synchronization should be strapped to logical zero.	N/A	sys_clk
dbg_b_latch_ack	1	Output	Acknowledge signal that strobe debug data from the BCH_CLK clock domain returned as response to the request driven on the dbg_b_latch_req controller port. When inactive it will be cleared.	1'h0	sys_clk
dbg_word_sel	6	Input	This port will select word from the debug vector to be displayed on the dbg_data port.	N/A	sys_clk
dbg_data	32	Output	Word being part of the debug vector selected by the dbg_word_sel port	32'h0	sys_clk

1.3.6. IP SRAM Interface Signals

Cadence NAND Flash Memory Controller IP requires the following external SRAM(s):

- ECC sector SPRAM buffer
- Descriptor and thread context storage SPRAM.

The ECC sector buffer is used in both read and write direction. For read direction it is used to buffer data for applying error correction when the BCH logic computes the error locations and mask. For the write direction it is used to buffer data to allow steady data stream flow between the system and the NAND Flash interfaces.

For details on the size of the ECC sector buffer required, refer to Memory implementation requirements section. This section provides implementation details for each memory used.

The below tables shows interfaces to the SRAM memory used as storage element for the ECC buffer. There are a few identical SRAM-s required for buffer implementation distinguish with numeric suffix. Each buffer can hold whole ECC data sector.

Table 1.8. BCH0 SPRAM Interface

Signal Name	Size	Type	Description	Reset Value	Associated clock
mem_ecc_ren_0	1	Output	Read enable for reading from SRAM. Read address is valid on the same cycle when read enable is asserted. This is an active high signal.	1'h0	bch_clk
mem_ecc_wen_0	1	Output	Write enable for writing into SRAM. Write address and write data is valid on the same cycle when write enable is asserted. This is an active high signal.	1'h0	bch_clk
mem_ecc_addr_0	7	Output	Memory address for read and write.	7'h0	bch_clk
mem_ecc_wdata_0	64	Output	Write data to be written into SRAM.	64'h0	bch_clk
mem_ecc_rdata_0	64	Input	Retrieved read data from SRAM. Read data is expected to be valid next cycle from the cycle read address is presented to SRAM.	N/A	bch_clk

Table 1.9. BCH1 SPRAM Interface

Signal Name	Size	Type	Description	Reset Value	Associated clock
mem_ecc_ren_1	1	Output	Read enable for reading from SRAM. Read address is valid on the same cycle when read enable is asserted. This is an active high signal.	1'h0	bch_clk
mem_ecc_wen_1	1	Output	Write enable for writing into SRAM. Write address and write data is valid on the same cycle when write enable is asserted. This is an active high signal.	1'h0	bch_clk
mem_ecc_addr_1	7	Output	Memory address for read and write.	7'h0	bch_clk
mem_ecc_wdata_1	64	Output	Write data to be written into SRAM.	64'h0	bch_clk
mem_ecc_rdata_1	64	Input	Retrieved read data from SRAM. Read data is expected to be valid next cycle from the cycle read address is presented to SRAM.	N/A	bch_clk

Table 1.10 shows interface to the SRAM memory used as context memory for the Command Engine. The context memory is accessed independently by more than one sub-module of the command engine, so de-multiplexers are used on this interface. Outputs from the sub-modules/inputs to de-multiplexing logic are all registered.

Table 1.10. Descriptors SPRAM Interface

Signal Name	Size	Type	Description	Reset Value	Associated clock
mem_desc_ren	1	Output	Read enable for reading from SRAM. Read address is valid on the same cycle when read enable is asserted. This is an active high signal.	1'h0	sys_clk
mem_desc_wen	1	Output	Write enable for writing into SRAM. Write address and write data is valid on the same cycle when write enable is asserted. This is an active high signal.	1'h0	sys_clk
mem_desc_addr	8	Output	Memory address for read and write.	8'h0	sys_clk
mem_desc_wdata	64	Output	Write data to be written into SRAM.	64'h0	sys_clk
mem_desc_rdata	64	Input	Retrieved read data from SRAM. Read data is expected to be valid next cycle from the cycle read address is presented to SRAM.	N/A	sys_clk

1.4. Initialization protocol

Cadence NAND Flash Memory Controller needs to be configured according to the type of the connected NAND flash device after all power sources are delivered to the ASIC and the memory devices are stable. To help this task automatized initialization protocol (Device Discovery process) is introduced. The Device Discovery process automatically detects what kind of NandFlash memory is connected to the controller and configure minimal set of registers required to perform simple operations. More precisely, after successfully finishing initialization protocol, the controller will be ready to perform simple data transfers (with disabled cache, multi-plane, Multi-lun and ECC) on device connected to bank 0 (Chip Select 0), in the timing mode 0 and with all timings set to the maximum value.

Whole initialization process is performed only basing on NandFlash device connected to bank 0. Controller require connections of the ready/busy pin (*dfi_rbn[0]*) for this bank during integration into system. For other banks this connection is optional.

During initialization process, the Controller classifies NAND devices into the following broad categories:

- ONFI devices - support for READ ID (with address 0x20 and 0x00) READ PARAMETER PAGE, and CHANGE READ COLUMN with 2 address bytes commands is required.
- Legacy devices - support for READ ID @0x00 command is required. Only a few predefined memory types are supported (see: Table 1.11). Additionally the *dd_** pins are used to fetch informations missing in READ ID command.
- Unrecognized devices - in case of unrecognized devices controller will utilize the *dd_** pins to read required configuration. Controller outputs the value returned by NandFlash device on READ ID @0x00 command on the *dd_id_value* port together with asserting the *dd_req* pin. Then it expects logical 1 on the *dd_ack* pin asserted together with values of Page Size (*dd_page_size* port), number of pages per block (*dd_pages_per_block*), flag enabling short address sequence where row address has only two bytess (*dd_four_addr_cycles_en*), flag selecting the 16 bit DQ bus on the flash interface (*dd_support_16_bit_en*), number of LUNs in single NandFlash device (*dd_lun_number*).

Table 1.11. Legacy Memories recognized by the controller.

Vendor	Part Number	Read ID Value
Toshiba	TC58NYG0S3HBAI6	98h A1H 80h 15h 72h

Vendor	Part Number	Read ID Value
Toshiba	TC58NYG1S3HBAI6	
Toshiba	TC58NVG2S0HBAI6	

Work flow of the Device Discovery process can be divided into the following phases:

- Phase 1: Waiting for NandFlash device ready after power-on.

  First, the controller waits for RB_valid_Vcc time. It is determined by the value of input signal *rb_valid_time* which should be greater than:

  RB_VALID_TIME > Trb [us] * fsys [MHz], where:

  Trb - value (in us) of the "RB_valid_Vcc" time (from Nand Flash device specification);

  fsys - frequency (in MHz) of the system clock (connected to the mACLK pin).

  After RB_valid_Vcc time controller checks *dfi_rbn[0]* pin and waits until it will be set to 1. This indicates that NandFlash memory device is ready for next phase of initialization process.

- Phase 2: Sending RESET command (0xFF) to the NandFlash memory device and waiting for finishing this commend. If the *discovery_inhibit* pin is set to 1 after this step controller is configured with values provided on the *dd_** ports and initialization process is finished. Otherwise next phase is executed.

  ## Note

  The Controller issues a RESET command to the devices connected only to bank 0. All other banks must be reset by software by sending a reset command to each device target after initialization is completed.

- Phase 3: Detecting memory type with READ ID command.

  Controller first sends the READ ID with address 0x20 and tries to read the ONFI signature. If signature is incorrect controller sends the READ ID with 0x00 address to detect legacy device. Again, if signature is unknown controller classifies connected memory device as "unrecognized device". Between the READ ID commands the RESET command is send each time to ensure that NandFlash memory device is not in unknown state.

- Phase 4: Reading NandFlash device parameters and setting control registers of the controller.

  Basing on memory classification determined in phase 3, controller reads NandFlash memory parameters value basing on:

  - value of READ ID command with 0x00 address (for all kind of devices)
  - value of Parameter Page (for ONFI devices only)
  - values provided on pins *dd_** (for unrecognized devices)

  ## Note

  If *discovery_ignore_crc* is cleared controller will check CRC value for Parameter Page. In this case if CRC value is incorrect controller will read next copy of Parameter Page (up to 3 copies).

  ## Note

  For Legacy devices controller will utilize the *dd_** interface to get informations about LUN number (*dd_lun_number*) and support for 4 address cycles (*dd_four_addr_cycles_en*) in connected device.

Finishing of the initialization phase will be signalled by setting the init_comp pin and *init_comp* bit in the *ctrl_status (0x0118)* register.

Minimum set of registers required by controller to perform any transfer operation contains:

- *transfer_cfg_1 (0x0404)* - setting of *sector_size* and *last_sector_size* fields with NF page size (main area size) value
- *nf_dev_layout (0x0424)* - setting value of *PPB* (number of pages per block) according to connected NF memory device
- field *four_addr_seq_en* in *device_ctrl (0x0430)* - if device with 4 address cycles is connected
- field *device_16bit* in *common_settings (0x1008)* - if 16-bit NF memory device is connected

If Device Discovery finishes with fail or if it is disabled (*discovery_inhibit* set to 1) host must program those registers before executing any command.

Additionally if Discovery process succeeds controller may fill rest of registers specified in Table 1.12. This table also contains information about how the value of each register is provided for different NandFlash memory type/category:

- ID - value of the register is provided by the READ ID command
- PP - value of the register is provided by the READ PARAMETER PAGE command
- - - information is not available for this memory type

Table 1.12. Registers filled during initialization

Register Name	ONFI	Toggle	Legacy	Unrecognized
transfer_cfg_1 (0x0404)	PP	PP	ID	PINS
nf_dev_layout (0x0424)	PP, only *PPB* and *LN* fields	PP, only *PPB* and *LN* fields	ID, PINS, only *PPB* and *LN* fields	PINS, only *PPB* and *LN* fields
device_ctrl (0x0430)	PP, only *four_addr_seq_en* field	PP, only *four_addr_seq_en* field	PINS, only *four_addr_seq_en* field	PINS, only *four_addr_seq_en* field
common_settings (0x1008)	PP, only *device_16bit* field	PP, only *device_16bit* field	ID, only *device_16bit* field	PINS, only *device_16bit* field
manufacturer_id (0x0808)	ID	ID	ID	ID
nf_device_areas (0x080c)	PP	PP	ID, *spare_area_size* not available	ID, *spare_area_size* not available
device_params_0 (0x0810)	ID and PP	ID and PP	ID, only *device_type* available	ID and PINS *plane_addr_bits* and *bits_per_cell* not available
device_params_1 (0x0814)	ID	ID	-	-
device_features (0x0818)	PP	PP	-	-
device_blocks_per_lun (0x081c)	PP	PP	-	-
device_revision (0x0820)	PP	PP	-	-
onfi_timing_modes_0 (0x0824)	PP	-	-	-
onfi_timing_modes_1 (0x0828)	PP	-	-	-

Register Name	ONFI	Toggle	Legacy	Unrecognized
onfi_iterlv_op_attr (0x082c)	PP	-	-	-
onfi_sync_opt_0 (0x0830)	PP	-	-	-
onfi_sync_opt_1 (0x0834)	PP	-	-	-

Those information are not critical for the controller but can be used by firmware to quickly obtain more information about connected memory. Depending on memory types not all registers will be filled. For detailed register descriptions, refer to Chapter 7, *Special Function Registers*.

1.5. Debug Interface

1.5.1. Introduction

Debug interface is intended to assist in debugging process of the NAND Flash controller after when project reached hardware verification phase. The debug interface signals can be probed inside application that uses controller IP and can be analyzed to narrow down the problem in hand.

The debug interface probe given module FSM states and selected internal signals values. The debug information can be accessed using controller's ports and register software interface.

Debug interface can work in two modes:

- Probing mode - in this mode debug information is probed using given clock domain clock signal and is transfered to the system clock domain without any synchronization logic.
- Probing and synchronization mode - in this mode debug information is probed using given clock domain clock signal and is transfered to the system clock four phase synchronization logic.

1.5.2. Hardware interface

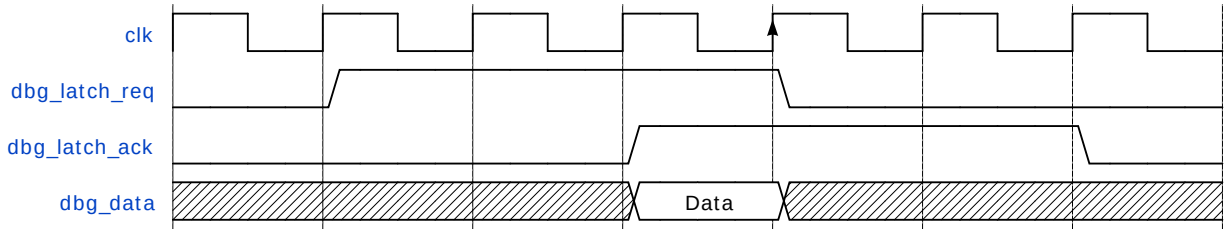
Signal of the debug interface are listed and described in the Table 1.7.

In the probing mode the *dbg_n_latch_req* and/or the *dbg_b_latch_req* signals should be driven low. In this mode controller constantly will transfer summary debug vector to the controller interface. Debug data can be read from the *dbg_data* port. Part of the debug vector that will be displayed is selected by the *dbg_word_sel* port.

In the Probing and synchronization mode the *dbg_n_latch_req* and/or the *dbg_b_latch_req* signals should be driven high and keep high as long as *dbg_n_latch_ack* and/or *dbg_b_latch_ack* signals aren't driven high by synchronization logic in the destination clock domain. When acknowledge signals are high then debug vector is stable and can be safely read. Each clock domain has separate pair of latch request/acknowledge signals. After debug data is read then the latch request signals need to be driven low, and host side need to wait for acknowledge signal to be cleared. When both request and acknowledge signals are clear then probing process can be repeated.

Figure 1.2 shows access sequence through the debug hardware interface. Data should be probed when *dbg_latch_ack* signal driven high as response to the *dbg_latch_req* driven high. Outside this window data shouldn't be probed.

Figure 1.2. Access to the debug hardware interface.



1.5.3. Register interface

Access to the debug information through register interface is implemented using three control registers: The *dbg_ctrl (0x0740)* register, the *dbg_stat (0x0744)* and the *dbg_data (0x0748)* register.

The *dbg_ctrl (0x0740)* register contain latch request signals used to request data in the Probing and synchronization work mode, the word select signal used to select part of debug vector and the *int_sel* bit used to select with interface will be used to access debug data.

The *dbg_stat (0x0744)* register contains latch acknowledge signals used to strobe data in the Probing and synchronization work mode.

The *dbg_data (0x0748)* register is used to transfer debug data when register interface is selected.

In the probing mode the latch request signal for given clock domain should be driven low. In this mode controller constantly will transfer summary debug vector to the system clock domain.

In the Probing and synchronization mode the latch request signals should be driven high and keep high as long as latch acknowledge signal isn't driven high by synchronization logic in the destination clock domain. When acknowledge signal is high then debug vector is stable and can be safely read. Each clock domain has separate pair of latch request/acknowledge signals. To continue work host need to clear acknowledge bits in the control register.

The debug vector is divided into the words that width is equal to the register interface word size. Requested vector part is selected using the *reg_dbg_wrd_sel* port. Debug signals from single clock domain are aligned to the word boundary.

2. Command Engine

This chapter describes the controller application interface and is divided into the following sections:

2.1. Introduction

The NAND flash controller can work in three different modes:

- The Command DMA work mode.
- The PIO work mode.
- The Generic mode.

The Command DMA work mode is dedicated for high-performance application where very low software overhead is required. In this mode the Command Engine is programmed by the series of linked descriptors stored in system memory. These descriptors provide commands to execute and store status information for finished commands.

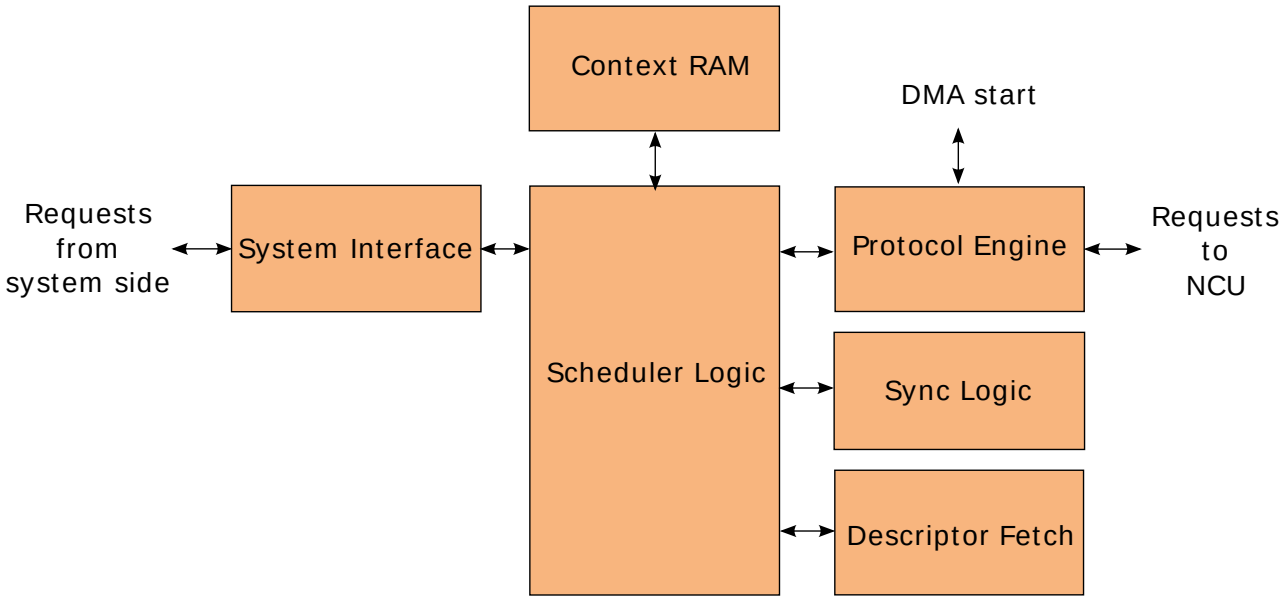
The PIO mode work mode is dedicated for single operation where constructing a linked list of descriptors would require too much effort. Additionally this work mode is used for special commands like set NAND flash device work mode.

The Generic work mode is a special work mode that can be used if the first two mode are not sufficient. In this work mode the Command Engine is transparent for software commands, they are directly passed to the NAND Flash interface. This allows software to control what exactly is sent to the NAND Flash interface.

Software must check if all previous operations were finished before changing work mode.

The following figure shows the Command Engine Architecture.

Figure 2.1. Command Engine Architecture



The scheduler is at the heart of the CMD_DMA module intelligently scheduling and sequencing the execution of descriptors on each execution thread. The scheduler efficiently time slices different operation thread stages, connects and passes control to the other modules like descriptor fetch, protocol engine, etc.

The sync module does the SYNC check and other sync functionality associated with each descriptor. This module is active only in CMD DMA work mode.

The system Interface module handles transmission between the Command Engine and the rest of the controller. It buffers transfers and controls the data flow.

The descriptor fetch module is used in the CMD DMA work mode only. It is responsible for fetching a descriptor from the system memory and placing it in the selected row of the context ram.

The context ram module stores operation record for selected thread. Each record occupies a single memory row.

Protocol engine decodes the operation directed at the flash device and associated information from the descriptor, issues the command to the controller core, captures interrupts and updates the status of the descriptor after completion of the operation. Protocol engine, all this while, also takes care of the device protocol and restrictions.

2.2. NAND Flash Address Layout

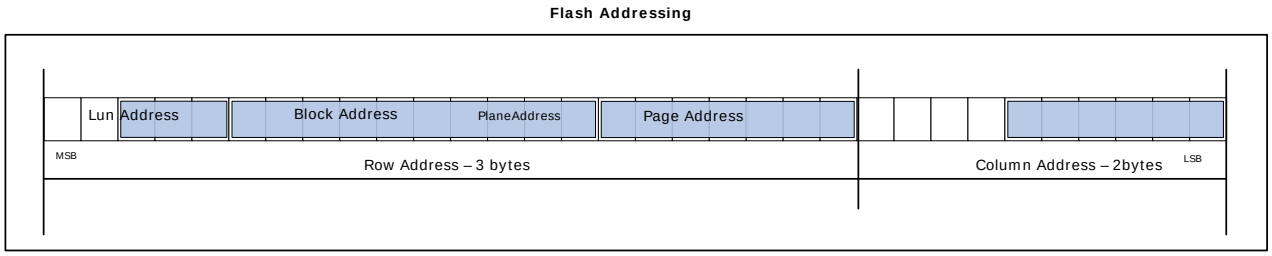
The Figure 2.2 shows NAND flash memory address layout. Two main parts of address are column address and row address. The column address is used to access bytes or words inside a page, it is byte aligned. The row address layout is composed from following parts:

- Page Address - this address part is used to select single page inside of block of pages. Page address doesn't need to be aligned to a power of two.
- Block Address - this address part is used to select single block of the NAND flash device.

  To raise data throughput, some NAND flash devices divide data blocks in to separate groups called planes and assign separate data register to each plane. This allows it to make parallel operations on different planes. Planes are selected by the low significant bits of the block address and are optional.

- LUN address - this address part is used to select the logical units inside the NAND flash memory. Logical unit is the separate target device that can independently execute commands and report status. Logical units share NAND flash device IO-s and are optional.

Figure 2.2. Row Address Layout



The row address is provided as part of command. In the CDMA work mode it is stored in the Flash Pointer descriptor field. In the PIO mode it is stored in the Command 1 register.

The field *offset* of the *transfer_cfg_0 (0x0400)* register allows to configure initial offset for the each transfered page. Value of this field will be used to set column address of each transfered page. The host software needs to take into consideration offset value when it configures transfered data block size, because programed data block will be written starting from the programed offset.

The host software shouldn't trigger transfers that reach outside physical boundaries of the NAND flash target (Device/LUN). In this case controller will try to access a non existing memory location.

2.3. Data Layout

2.3.1. Page Layout

Page size is parametrized by following parameters:

- *sector_cnt* - Number of sectors per single NAND flash page.
- *sector_size* - Size of all sectors except the last one. Sector size needs to be aligned to 16-bit word boundaries when 16-bit device is used.
- *last_sector_size* - Size of the last sector. Sector size needs to be aligned to 16-bit word boundaries when 16-bit device is used.

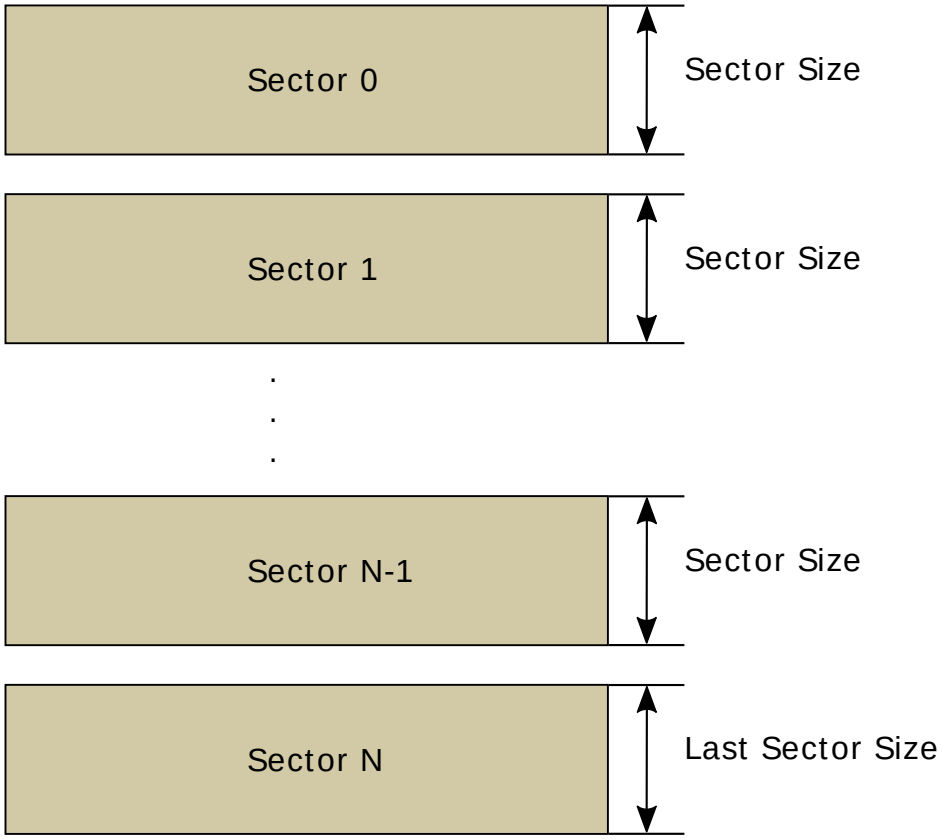
Page size is calculated using following formula:

page_size = (sector_cnt - 1) * sector size + last_sector_size (2.1)

This page size configuration schema was selected because of constraints on the transfered data blocks size introduced by the ECC engine. The ECC engine requires that transfered data needs to be divided into the sectors that can be handled by the ECC engine. Information how to configure this parameters can be found in section ECC Sector sizes.

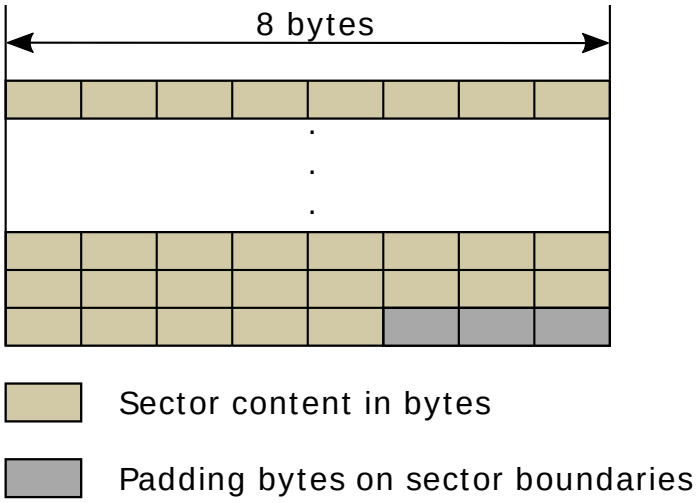
The Figure 2.3 shows how single page is aligned in the host memory.

Figure 2.3. NAND Flash Page Alignment in Host Memory



Each sector need to be aligned to the 64-bit word boundaries, so in case when number of transmitted data for a single sector doesn't meet this requirement then padding bytes need to be added. The Figure 2.4 shows sector layout in this case.

Figure 2.4. Sector Layout - Byte Alignment



2.3.2. Layout Between Pages

Data layout in the NAND flash device is different when multiple planes support is enabled or disabled:

- If this feature is enabled then transfered pages are placed in the following blocks selected by the plane addresses. In this case number of transfered pages must be multiplicity of planes number. When Number of transfered pages is higher than number of planes then after data is transfered to each plane the page address is incremented and another set of pages is transfered to the all planes.
- If this feature is disabled then page address is incremented for each transfered page.

On the system side data is transfered in ascending order starting from address programed in the controllers descriptor or command (for PIO mode).

For the write direction DMA engine will be triggered to transfer each page separately. For the read direction DMA will be triggered to transfer data block composed from pages for all planes.

2.4. CMD DMA Work Mode

This section describes the CMD DMA work mode, its usage and restrictions. The following topics are covered in this chapter.

- Overview
- Command DMA Operation
- Command Descriptor and Fields Description
- Sync functionality
- NOP descriptor

2.4.1. Overview

CMD DMA work mode uses descriptor-based command processing functionality of the Command Engine module. Host software will be able to chain command descriptors in system memory and initiate the Command Engine module to operate on the descriptors chain. Each command descriptor will perform one distinct operation on the controller. Host in the system memory can chain multiple controller operations sequentially as a linked-list of command descriptors.

The Command Engine module will fetch descriptors one-by-one, carry out the operation as described in the descriptor, write status of the operation completed, generate interrupt if required, and then fetch a next descriptor. The next descriptor fetch request to the host will always follow a status write request of the current descriptor. The command issuance would be in order of a status write followed by a descriptor read request if the current descriptor is not the last descriptor. But the status write completion might not happen before the read request though.

Host software will be absolved of monitoring status of issued commands, as the status will be updated as part of the descriptor fields upon command completion by the Command Engine module.

The host can instruct the Command Engine module to issue an interrupt on per descriptor basis on the completion of a particular descriptor.

The Command Engine module is capable of handling a configurable number of threads simultaneously. Each thread can be mapped to a chip select/Volume/LUN of the controller expanded group of devices. The Command Engine will process threads independent of each other.

The host will be able to issue following commands with descriptors:

- Page Read/Write
- Erase
- Copyback
- Reset

2.4.2. Operation in CMD DMA work mode

To start the Command Engine in the CMD DMA work mode, two pieces of information are required: descriptors chain start address and the thread number associated with the descriptors chain. The Command0, Command2 and Command3 registers are used for this purpose. Table 2.1 to Table 2.3 show the command registers layout for the CMD DMA work mode.

Table 2.1. Command 0 register layout

31:30	26:24	23:0
CT=2'b00	TRD_NUM (thread number)	Reserved

Table 2.2. Command 2 register layout

31:0
Descriptor address low

Table 2.3. Command 3 register layout

31:0
Descriptor address high

The host needs to write logical zero to the CT field of the Command 0 register and provide linked descriptors list head address using Command 3 and Command 2 registers.

2.4.3. Command Descriptor and Fields Description

The complete descriptor is fetched using a burst transaction by the Controller. Table 2.4 presents a descriptor layout.

Note

The address pointers should be aligned to the natural boundary of the initiator interface data width.

Table 2.4. Command Descriptor Structure

No	63:48	47:32	31:16	15:0
0	Next Pointer			
1	Reserved		Flash Pointer / Copyback Source Address	
2	Reserved	Command Flags	Reserved	Command Type
3	Memory Pointer / Copyback Destination Address			
4	Reserved		Status	
5	Sync Flag Pointer			
6	Reserved		Sync Arguments	
7	Reserved			

Table 2.5. Command Descriptor Layout

Field	Description
Next pointer	Next descriptor address. Address needs to follow system bus restrictions.
Flash pointer / Copyback Source Address	Flash address is a 32-bit address comprising of BANK and ROW ADDR. If a multi-plane feature is enabled, this pointer should contain the block address of plane 0 of the NAND Flash device (refer to Table 2.6, “Flash Pointer Field Description”). For copyback operations this field contains source address.
Command flags	This contains different control flags for operation of this command (refer to Table 2.7, “Command Flags Field Description”).
Command type	This field identifies what kind of operation the Controller needs to perform. The encoding is: • 0x10PP - Block Erase of 'PP+1' number of sequential blocks. • 0x11PP - bank/vol/LUN Reset. Automation for this command was disabled, so the PP parameter is used to select reset type: PP = 0x00 - asynchronous reset, 0x02 - LUN reset. • 0x12PP - Copyback operation of 'PP+1' number of sequential pages. Both source and destination addresses need to point on the same BANK, LUN and Plane. The hosts software needs to follow device addressing restrictions when it selects source and destination flash pointer. When host set both addresses it must additionally take into account the number of pages to be sure that Controller will not cross LUN/Plane boundary if multi-plane operation in controller are disabled.

Field	Description
	• 0x21PP - Command for Program Operation of 'PP+1' number of sequential pages. • 0x22PP - Command for Read Operation of 'PP+1' number of sequential pages. • 0xFFFF - NOP descriptor. • Other values reserved.
Memory Pointer / Copyback Destination Address	System/host memory address required for Data DMA commands. For copyback operations this field contains destination address.
Status	Controller will update this field with command status once the command operation is complete. Refer to table with Status Field Description below for explanation of the bits in Status field (refer to Table 2.8, "Status Field Description").
Sync Flag pointer	Address pointer to sync buffer location.
Sync arguments	Controls the buffer sync mechanism.

Table 2.6. Flash Pointer Field Description

Bits	Name	Description
31:27	-	Reserved
26:24	BANK	This field contains operation bank number. Only banks with physical devices connected should be selected.
23:0	ROW_ADDRESS	This field contains operation Row Address

On receipt of a complete descriptor (as illustrated above), the Controller will construct a low level command or series of low level commands to be issued to the Mini Controller module. Burst Length information is configured by the *burst_sel* field of the *dma_settings (0x043c)* register.

Table 2.7. Command Flags Field Description

Bits	Name	Description
15:11	Reserved	Reserved
10	DMA Sel	Selects DMA Slave (0) or DMA Master (1) data interface. If selected command does not require data transfer, this bit will be ignored.
9	Cont	The Next descriptor address field is valid and descriptor processing should continue. This bit should be zero only for the last descriptor in a descriptors chain.
8	Int	An interrupt should be issued after the completion of descriptor processing. The triggered interrupt will be *trd_comp* field of the *trd_comp_intr_status (0x0138)* register. Where interrupt bit will be selected by the thread number selected by the TRD_NUM field.
7:4	Target Volume	This field informs the controller the volume this current command is targeting to. If the device does not support volumes or if does not intend on using different volumes, this field needs to be set to 0.
3:2	Reserved	Reserved
1	Flash_Ptr_Cont	If this bit is set then the "Flash pointer" field of current descriptor is ignored and command engine will use existing flash pointer value. This bit allows to continue transfer from the address on the flash interface where previously completed descriptor finished.

Bits	Name	Description
		This feature will work correctly only when this bit is set for the descriptor that follows other descriptor which has this bit cleared and will initialize flash pointer value. It should be used for command of the same type, for example series of read descriptors or series of write descriptors.
0	Mem_Ptr_Cont	If this bit is set then the "Memory pointer" field of current descriptor is ignored and command engine will use existing memory pointer value. This bit allow to continue transfer from the address on the host interface where previously completed descriptor finished. This feature will work correctly only when this bit is set for the descriptor that follows other descriptor which has this bit cleared and will initialize memory pointer value. It should be used for command of the same type, for example series of read descriptors or series of write descriptors.

On completion of command operation in the controller core, the status will be written back into the descriptor placeholder. This field value is ignored during descriptor read operation. The status word description is illustrated in table below.

Note

If the *Int* bit is not set in the current descriptor and the following descriptor chain is drooped from execution due to an error, then controller will not set the *trd[N]_comp* interrupt flag in the *trd_comp_intr_status (0x0138)* register. The error condition is always reported by setting the *trd[N]_error_stat* bit in the *trd_error_intr_status (0x0128)* register and can be used to detect that descriptor chain execution was interrupted due to an error. The N in the field names, represent the thread number for which error or complete condition occur.

Table 2.8. Status Field Description

Bits	Name	Description
31:24	Error Index	If bit the *Fail* is set, this field indicates the operation index number where the first error was detected. Operation are numbered from 0. This field isn't updated when source of error is transfer on the system bus - in this case only the *Bus Error* bit is set.
23:17	Reserved	Reserved
16	Bus error	When set it informs that the controller got an error response on the system bus.
15	Complete	When set, denotes that the controller has updated status information and the operation is complete. This bit shall be set even if the operation ended as a failure. This bit should be in cleared state while descriptor is constructed. If this bit is set when the controller reads the descriptor for execution, the controller will skip execution of the descriptor and continue with execution of further descriptors in the chain depending on the state of the 'continue' bit.
14	Fail	When set, denotes that operation failed to complete successfully.
13	Reserved	Reserved
12	device_error	Device error was detected during read status operation in any of the device planes.
11	Erased Page	When set, denotes that the controller detected an erased page in the read transaction. The detection of erased page is based on the number of 0's in a page. If the number of 0's in a page being read is less than the value on the *erase_det_lvl* field of the *ecc_config_1 (0x042c)* register, an erased page is inferred and no uncorrectable error will be flagged for that page. If ECC is disabled, the erased_page interrupt shall be set as explained above. If ECC is enabled, in addition to the above condition, only when the ECC logic detects no errors or correctable error pattern for that page will the

Bits	Name	Description
		erased_page interrupt be flagged. If the ECC logic detects a uncorrectable error page, this erased page interrupt will not be set. This flag doesn't contribute to the fail flag.
10	prot_err	If this bit is set then it mean that programed operation tried to modify protected area.
9:2	Max Error	For a Flash read command, this field indicates the maximum amount of correction applied to one ECC sector. This field is of significance only if the read transaction resulted in correctable errors. If no errors were found, this field will read zero.
1	ecc_err	Uncorrectable ECC error was detected for the any of the device planes.
0	Descriptor Error	This bit denotes that an invalid descriptor sequence has been detected.

2.4.4. Sync Functionality

Sync structure is present in the Command Engine descriptors to allow data transfers in different Command Engine execution threads to be synchronized based upon sync buffer states. This structure enables all threads to synchronize data transfer and command activity based upon any number of sync buffers and command DMA execution threads. Each sync buffer flag is addressed by a pointer provided in the *Sync Flag Pointer* field of a descriptor and must be placed in memory which can be accessed by the AXI master interface. Sync buffer is 1 byte long (the controller will access full data word but will ignore the most significant bytes). Sync buffer flags are logically associated with CMD DMA descriptors. The sync buffer flags are the global data transfer communication structure.

When descriptors *valid* bit is set in the *Sync arguments* field, each command of the command engine execution thread list requires a sync buffer flag to be at a specific condition at the start of the command before proceeding with the command. When the descriptor operation is complete, it sets the sync buffer flag to a new condition.

The synchronization condition will be checked and updated only for commands that will be accepted for execution. If command cannot be executed for any reason then synchronization process will not be executed as well.

The following steps illustrate the Command Engine execution thread. operations:

1. The Command Engine fetches a new descriptor and tests for the *valid* flag in the *Sync Arguments* field. If *valid* bit is clear, continue with the flash operation present in the descriptor and continue normal processing.

2. If *valid* bit is set in the *Sync Argument* field, sync buffer flag must be equal to or greater than a specified value before starting the command. This start condition is specified in the *Sync Argument* field of the descriptor. The Command Engine will read sync buffer flag from memory. If start condition is met, the Command Engine moves to step Execute flash operation in the selected thread.

3. If the sync buffer flag does not meet the start condition, the Command Engine waits for an update on the sync buffer flag pointer from operation in other execution thread or for the polling counter to expire. The Command Engine then proceeds to read the sync buffer flag as in the above step.

4. Execute flash operation contained in the descriptor.

5. Perform status update of the descriptor.

6. If the *valid* bit is set in the *Sync argument* field, set the buffer flag to a new value or an increment of the current value of the flag as indicated in the *Sync argument* field of the descriptor.

7. Command Engine will issue an interrupt if *Int* bit is set in *Command Flags* field. This concludes the processing of the current descriptor and the next descriptor is fetched.

Using and sharing sync buffers, operations in two independent execution threads can synchronize their data transfers or command activity.

If the *Greater* bit in the *Sync argument* field is set then the *Start Value* defines the beginning of the window that has 128 elements and wrap inside 256 values set. If the *buffer flag* value is inside this window then sync triggering condition is meet. The Figure 2.5 shows the sync functionality defines valid data window when the *Greater* bit in the *Sync argument* field is set.

Figure 2.5. Sync behavior when the Greater flag is set

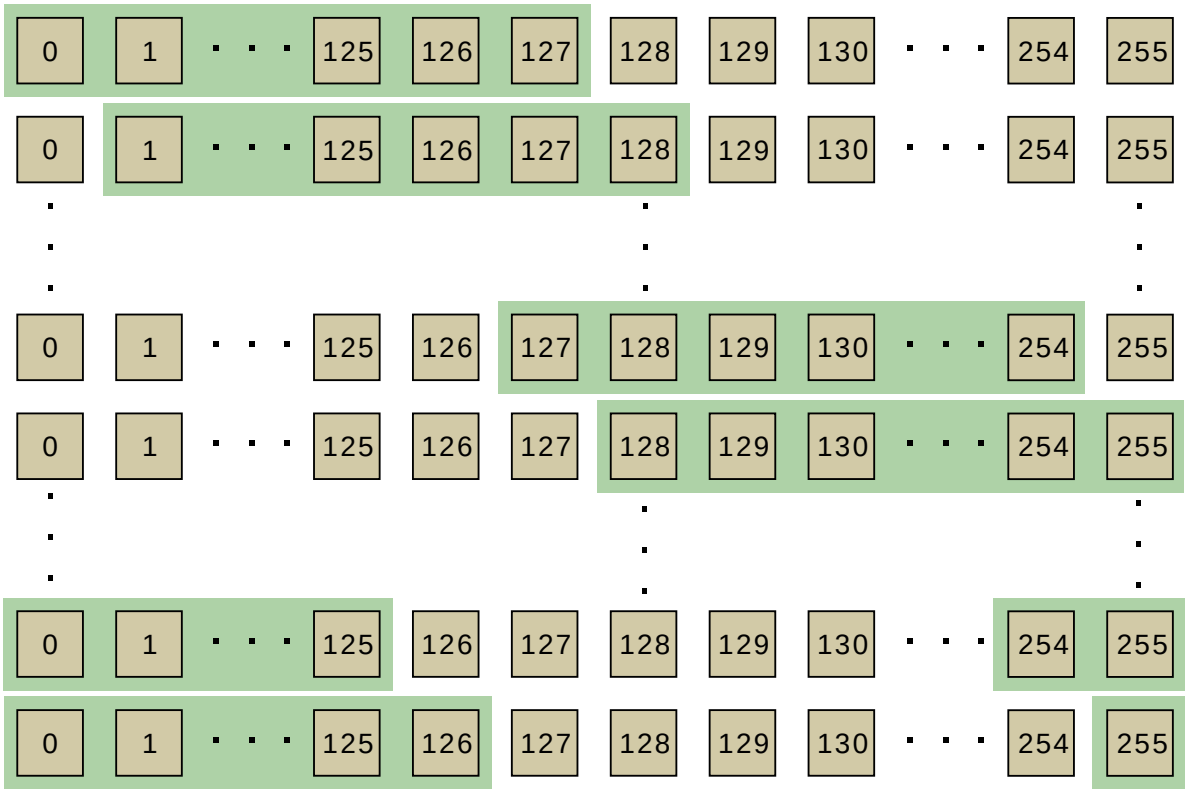


Table 2.9. Sync Argument Field Description

Bits	Name	Description
31:24	Start Value	Start value of the sync buffer flag. The value required to start an operation.
23:16	End Value	Value to be stored to the sync buffer flag upon the successful completion of the operation unless the type is set to Increment.
15:3	Reserved	Must be set to zero by firmware.
2	Valid	Sync information is valid in this descriptor.
1	Greater	When set to zero, the start condition is the sync buffer flag must be equal to the value in bits 31:24. When set to one, the start condition is equal to or greater than the value in bits 31:24. 8-bit Modulo arithmetic is used. The sync buffer flag is considered "equal to or greater than" bits 31:24 when sync buffer flag minus bits 31:24 is between 0 and 127.
0	Increment	When set to zero, the type is to store the value in bits 23:16 to the sync buffer flag. When set to one, the type is to increment the sync buffer flag and the value 23:16 is

Bits	Name	Description
		ignored. The Increment is to read the current value of the flag, increment it by one, and store the sync buffer flag.

2.4.5. NOP Descriptor

No-operation descriptor does not execute any Flash Controller commands. It can be used as a dummy descriptor to create sync points or to generate interrupts. This descriptor can be inserted at the end of a descriptor chain to interrupt the host Controller.

Table 2.10. NOP Descriptor Format

Descriptor field	Value
Next Pointer	Next descriptor address. Address need to follow the system bus restrictions.
Flash Pointer	For the NOP descriptor this field is ignored.
Command Type	0xFFFF
Memory Pointer	System/host memory address required for Data DMA commands. Controller will update this field with command status once command operation is complete.
Status	Controller will update this field to indicate pass/fail status of the operation.

2.5. PIO Work Mode

This section describes the PIO work mode, its usage and restrictions. The following topics are covered.

- Overview
- Commands Description
- Last Operation Status

2.5.1. Overview

In the PIO work mode, dedicated command registers are used to trigger operation in selected execution thread of the Command Engine. In this mode it is possible to trigger single command in each of the Command Engine execution threads. This mode is dedicated to a situation when host software wants to trigger a single operation. In this case it is much simpler and faster to program a few controllers registers than prepare descriptors table in the system memory.

The Command Engine will execute the programmed command and return status word separately for each execution thread. Status words are mapped into the controller register address space. Status for a selected execution thread is valid until a new command is triggered on this thread.

Functionality of this work mode is similar to the CMD DMA work model when descriptors list contains only a single descriptor.

Following commands are allowed in this work mode:

- Page Read/Write
- Erase
- Copyback

- Reset
- Set Features

2.5.2. Commands Description

2.5.2.1. General rules

Depending on selected operation, the number of the used command registers will change. Registers described as unused for given operation do not need to be set. If current register configuration is sufficient for a new operation, it can be left unchanged. The only exception is command register 0. This register is always used on each operation. Write access to this register will trigger operation execution. Host software needs to initialize all required command registers before it writes to command register 0.

2.5.2.2. Page Read Command

This command allows to read a data block that is composed from specified number of NAND Flash memory pages. This command will be translated into a series of low level page read, multi-plane page read or cache read operations, depending on core configuration. The Section 2.7, "*Extended commands support*" section shows how valid transfer mode is selected.

Table 2.11 shows command layout and the field descriptions for Command 0 register. Operation will be triggered after this register is written.

Table 2.11. Page Read Operation - Command 0

Bits	Name	Description
31:30	CT	This field needs to have the 2'b01 value for the PIO work mode
29:27	-	Reserved
26:24	TRD_NUM	This field selects destination thread number for command. Software can select any available thread. Commands can be issued in parallel to all threads
23:22	-	Reserved
21	DMA_SEL	This bit selects DMA engine. Possible options are 0 - DMA Slave and 1 - DMA Master
20	INT	If this bit is set then an interrupt should be issued after this operation will be finished. The triggered interrupt will be bit of the *trd_comp* field in the *trd_comp_intr_status (0x0138)* register, where interrupt bit will be selected by the thread number selected by the TRD_NUM field.
19:16	VOL_ID	This field identifies a volume that is a target for this command. If *ce_pin_reduct* bit of the *device_ctrl (0x0430)* register is cleared, this field is ignored.
15:0	CMD_TYPE	This field identifies the kind of operation that will be executed by the controller. For the Page Read Command this field encoding is as follows: 0x22PP, where "PP" is a number of sequential pages to transfer decoded as "PP+1" (0 means that one page will be transfered).

Table 2.12 shows layout and fields description for Command 1 register.

Table 2.12. Page Read Operation - Command 1

Bits	Name	Description
31:27	-	Reserved

Bits	Name	Description
26:24	BANK	This field contains operation bank number. Only banks with physical devices connected should be selected.
23:0	ROW_ADDRESS	This field contains operation Row Address

Table 2.13 and Table 2.14 show layout and field descriptions for the Command 2 and the Command 3 registers.

Table 2.13. Page Read Operation - Command 2

Bits	Name	Description
31:0	MEM_ADDR_PTR_L	Host memory address required for DMA transfers. It is lower address part.

Table 2.14. Page Read Operation - Command 3

Bits	Name	Description
31:0	MEM_ADDR_PTR_H	Host memory address required for DMA transfers. It is higher address part.

2.5.2.3. Page Program Command

This command allows to program a data block that is composed of specified number of NAND Flash memory pages. This command will be translated into a series of low level page program, multi-plane page program, or cache program operations depending on core configuration. The Section 2.7, "*Extended commands support*" section shows how valid transfer mode is selected.

Table 2.15 shows command layout and fields descriptions for Command 0 register. Operation will be triggered after this register is written.

Table 2.15. Page Program Operation - Command 0

Bits	Name	Description
31:30	CT	This field needs to have the 2'b01 value for the PIO work mode
29:27	-	Reserved
26:24	TRD_NUM	This field selects destination thread number for command. Software can select any available thread. Commands can be issued in parallel to all threads
23:22	-	Reserved
21	DMA_SEL	This bit selects DMA engine. Possible options are 0 - DMA Slave and 1 - DMA Master
20	INT	If this bit is set, an interrupt should be issued after this operation is finished. The triggered interrupt will be *trd_comp* field of the *trd_comp_intr_status (0x0138)* register. Interrupt bit will selected by the thread number selected by the TRD_NUM field.
19:16	VOL_ID	This field identifies a volume that is a target for this command. If *ce_pin_red uct* bit of the *device_ctrl (0x0430)* register is cleared then this field is ignored.
15:0	CMD_TYPE	This field identifies kind of operation that will be executed by the controller. For the Page Program Command this field encoding is as follows: 0x21PP, where "PP" is a number of sequential pages to transfer decoded as "PP+1" (0 means that one page will be transfered).

Table 2.16 shows layout and field descriptions for Command 1 register.

Table 2.16. Page Program Operation - Command 1

Bits	Name	Description
31:27	-	Reserved
26:24	BANK	This field contains operation bank number. Only banks with physical devices connected should be selected.
23:0	ROW_ADDRESS	This field contains operation Row Address

Table 2.17 and Table 2.18 show layout and field descriptions for the Command 2 and the Command 3 registers.

Table 2.17. Page Program Operation - Command 2

Bits	Name	Description
31:0	MEM_ADDR_PTR_L	Host memory address required for DMA transfers. Lower part.

Table 2.18. Page Program Operation - Command 3

Bits	Name	Description
31:0	MEM_ADDR_PTR_H	Host memory address required for DMA transfers. Higher part.

2.5.2.4. Copyback Command

This command allows to read and write back a data block that is composed of specified number of NAND Flash memory pages. This command will be translated into a series of low level Copyback commands. The Section 2.7, *"Extended commands support"* section shows how valid transfer mode is selected.

Table 2.19 shows command layout and fields description for Command 0 register. Operation will be triggered after this register is be written.

Both source and destination addresses need to point on the same BANK, LUN and Plane. The host's software needs to follow device addressing restrictions when it selects source and destination flash pointers.

Table 2.19. Copyback Operation - Command 0

Bits	Name	Description
31:30	CT	This field need to have the 2'b01 value for the PIO work mode
29:27	-	Reserved
26:24	TRD_NUM	This field selects destination thread number for command. Software can select any available thread. Commands can be issued in parallel to all threads.
23:21	-	Reserved
20	INT	If this bit is set, an interrupt should be issued after this operation is finished. The triggered interrupt will be *trd_comp* field of the *trd_comp_intr_status (0x0138)* register, where interrupt bit will be selected by the thread number selected by the TRD_NUM field.
19:16	VOL_ID	This field identifies a volume that is a target for this command. If *ce_pin_reduct* bit of the *device_ctrl (0x0430)* register is cleared then this field is ignored.

Bits	Name	Description
15:0	CMD_TYPE	This field identifies kind of operation that will be executed by the controller. For the Copyback Command this field encoding is as following: 0x12PP. Where "PP" is a number of sequential pages to transfer decoded as "PP+1" (0 means that one page will be transfered).

Table 2.20 and Table 2.21 show layout and fields description for the Command 1 and the Command 2 registers.

Table 2.20. Copyback Operation - Command 1

Bits	Name	Description
31:27	-	Reserved
26:24	BANK	This field contains operation bank number for source location. Only banks with physical devices connected should be selected.
23:0	ROW_ADDRESS	This field contains operation Row Address for source location.

Table 2.21. Copyback Operation - Command 2

Bits	Name	Description
31:27	-	Reserved
26:24	BANK	This field contains operation bank number for target location. Only banks with physical devices connected should be selected.
23:0	ROW_ADDRESS	This field contains operation Row Address for target location.

2.5.2.5. Block Erase Command

This command allows to erase contents of specified number following NAND Flash memory data blocks. This command will be translated into the series of low level block erase commands.

Table 2.22 shows command layout and fields description for Command 0 register. Operation will be triggered after this register is written.

Table 2.22. Block Erase Operation - Command 0

Bits	Name	Description
31:30	CT	This field needs to have the 2'b01 value for the PIO work mode
29:27	-	Reserved
26:24	TRD_NUM	This field selects destination thread number for command. Software can select any available thread. Commands can be issued in parallel to all threads
23:21	-	Reserved
20	INT	If this bit is set, an interrupt should be issued after this operation is finished. The triggered interrupt will be n bit of the *trd_comp* field in the *trd_comp_intr_status (0x0138)* register, where n will be thread number selected by the TRD_NUM field.
19:16	VOL_ID	This field identifies a volume that is a target for this command. If *ce_pin_reduct* bit of the *device_ctrl (0x0430)* register is cleared, this field is ignored.

Bits	Name	Description
15:0	CMD_TYPE	This field identifies the kind of operation that will be executed by the controller. For the Block Erase Command this field encoding is as follows: 0x10PP, where "PP" is a number of sequential blocks to erase decoded as "PP+1" (0 means that one block will be erased).

Table 2.23 shows layout and fields description for the Command 1 register.

Table 2.23. Block Erase Operation - Command 1

Bits	Name	Description
31:27	-	Reserved
26:24	BANK	This field contains operation bank number for source location. Only banks with physical devices connected should be selected.
23:0	ROW_ADDRESS	This field contains operation Row Address for source location.

2.5.2.6. Reset Command

This command sends the Reset Command to selected BANK/LUN. This command will be translated into a low level reset command. Exact command will be selected basing on the CMD_TYPE field:

- 0x1100 - asynchronous reset.
- 0x1102 - LUN reset.

Table 2.24 shows command layout and fields description for Command 0 register. Operation will be triggered after this register has been written to.

Table 2.24. Reset Operation - Command 0

Bits	Name	Description
31:30	CT	This field needs to have the 2'b01 value for the PIO work mode
29:27	-	Reserved
26:24	TRD_NUM	This field selects destination thread number for command. Software can select any available thread. Commands can be issued in parallel to all threads.
23:21	-	Reserved
20	INT	If this bit is set, an interrupt should be issued after this operation is finished. The triggered interrupt will be n bit of the *trd_comp* field in the *trd_comp_intr_status (0x0138)* register, where n will be thread number selected by the TRD_NUM field.
19:16	VOL_ID	This field identifies a volume that is a target for this command. If *ce_pin_red uct* bit of the *device_ctrl (0x0430)* register is cleared then this field is ignored.
15:0	CMD_TYPE	This field identifies the kind of operation that will be executed by the controller. For the Reset Command this field encoding is as follows: 0x11PP. The PP parameter is used to select reset type: PP = 0x00 - asynchronous reset, 0x02 - LUN reset

Table 2.25 shows layout and fields description for the Command 1 register.

The ROW_ADDR field is valid only for the LUN Reset command. For other reset types it is ignored

Table 2.25. Reset Operation - Command 1

Bits	Name	Description
31:27	-	Reserved
26:24	BANK	This field contains operation bank number for source location. Only banks with physical devices connected should be selected.
23:0	ROW_ADDRESS	This field contains operation Row Address for source location.

2.5.2.7. Set Feature Command Description

This command will be translated into the Set Feature command on the NAND Flash interface. This command needs to be used to switch ONFI device work mode.

Controller selects the same configuration for each LUN on selected bank, so LUN set feature commands will never be used.

Table 2.26 shows layout and fields description for Command 0 register. Operation will be triggered after this register has been written to.

Table 2.26. Set Feature Operation - Command 0

Bits	Name	Description
31:30	CT	This field needs to have the 2'b01 value for the PIO work mode
29:27	-	Reserved
26:24	TRD_NUM	This field selects destination thread number for command. Software can select any available thread. Commands can be issued in parallel to all threads
23:21	-	Reserved
20	INT	If this bit is set, an interrupt should be issued after this operation is finished. The triggered interrupt will be n bit of the *trd_comp* field in the *trd_comp_intr_status (0x0138)* register, where n will be thread number selected by the TRD_NUM field.
19:16	VOL_ID	This field identifies a volume that is a target for this command. If *ce_pin_red uct* bit of the *device_ctrl (0x0430)* register is cleared then this field is ignored.
15:0	CMD_TYPE	This field identifies the kind of operation that will be executed by the controller. For the Set Features Command this field encoding is as follows: 0x0100.

Table 2.27 shows layout and fields description for the Command 1 register.

Table 2.27. Set Features Operation - Command 1

Bits	Name	Description
31:27	-	Reserved
26:24	BANK	This field contains operation bank number for source location. Only banks with physical devices connected should be selected.
23:8	-	Reserved

Bits	Name	Description
7:0	FEATURE_ADDR	This field selects feature address. The address value will be translated into the address that will be sent with SET FEATURE command on the NAND Flash interface. Allowed values can be found in the NAND Flash device data sheet

Table 2.28 shows layout and fields description for the Command 2 register.

Table 2.28. Set Features Operation - Command 2

Bits	Name	Description
31:0	FEATURE_VAL	This field stores feature data value. The FEATURE_VAL will be sent with SET FEATURE command on the NAND Flash interface. Allowed values can be found in the NAND Flash device data sheet

2.5.3. Last Operation Status

The host can check latest operation status associated with the selected thread by reading the *cmd_status (0x0014)* register. This register contains command status from thread selected by value of the *cmd_status_ptr (0x0010)* register.

After a new command is triggered in selected thread the *Complete* bit is cleared and from this moment other register bits are invalid. When the *Complete* bit is set back then other bits in the status register are valid.

Table 2.29 shows layout and fields description for the status register.

Table 2.29. Status Field Description

Bits	Name	Description
31:24	Error Index	If bit the *Fail* is set, this field indicates the operation index number where the first error was detected. Operation are numbered from 0. This field isn't updated when source of error is transfer on the system bus - in this case only the *Bus Error* bit is set.
23:17	Reserved	Reserved
16	Bus error	When set it informs that the controller got an error response on the system bus.
15	Complete	When set, denotes that the controller has updated status information and the operation is complete. This bit shall be set even if the operation ended as a failure. This bit should be in cleared state while descriptor is constructed. If this bit is set when the controller reads the descriptor for execution, the controller will skip execution of the descriptor and continue with execution of further descriptors in the chain depending on the state of the 'continue' bit.
14	Fail	When set, denotes that operation failed to complete successfully.
13	Reserved	Reserved
12	device_error	Device error was detected during read status operation in any of the device planes.
11	Erased Page	When set, denotes that the controller detected an erased page in the read transaction. The detection of erased page is based on the number of 0's in a page. If the number of 0's in a page being read is less than the value on the *erase_det_lvl* field of the *ecc_config_1 (0x042c)* register, an erased page is inferred and no uncorrectable error will be flagged for that page. If ECC is disabled, the erased_page interrupt shall be set as explained above. If ECC is enabled, in addition to the above condition, only when the ECC logic detects no errors or correctable error pattern for that page will the

Bits	Name	Description
		erased_page interrupt be flagged. If the ECC logic detects a uncorrectable error page, this erased page interrupt will not be set. This flag doesn't contribute to the fail flag.
10	Reserved	Reserved
9:2	Max Error	For a Flash read command, this field indicates the maximum amount of correction applied to one ECC sector. This field is of significance only if the read transaction resulted in correctable errors. If no errors were found, this field will read zero.
1	ecc_err	Uncorrectable ECC error was detected for the any of the device planes.
0	Command Error	This bit will be set when programed command sequence can't be executed because incorrect controller or command configuration. This bit will not be set when unknown command is programed to the commands registers. Error sources for this bit are described in the Section 2.8.4, *"Descriptor/Command Error"*.

2.6. Generic Work Mode

This work mode allows to bypass the Command Engine module and allows to send low level commands directly to the Mini Controller unit. The Mini Controller unit implements simple operation on the NAND Flash interface (these are operations described in the NAND Flash devices documentation). In the Generic work mode, the software is responsible for all low level implementation details that normally are handled by the Command Engine. This mode should be used only if CDMA or PIO modes are not sufficient.

2.6.1. Command Layout

Table 2.30 shows command layout and fields description for Command 0 register. Operation will be triggered after this register is written.

The unused field are labeled as reserved. Those fields should be set to 0.

Table 2.30. Generic Sequence - Command 0

Bits	Name	Description
31:30	CT	This field need to have the 2'b11 value for the GENERIC work mode
29:27	-	Reserved
26:24	TRD_NUM	This field selects destination thread number for command. Software can select any available thread. Commands can be issued in parallel to all threads
23:21	-	Reserved
20	INT	If this bit is set, an interrupt should be issued after this operation is finished. The triggered interrupt will be n bit of the *trd_comp* field in the *trd_comp_intr_status (0x0138)* register, where n will be thread number selected by the TRD_NUM field.
19:0	-	Reserved

Table 2.31 shows layout and fields description for the Command 2 register.

Table 2.31. Generic Sequence - Command 2

Bits	Name	Description
31:0	CMD_VAL_L	This field store lower part of the Mini Controller command

Table 2.32 shows layout and fields description for the Command 3 register.

Table 2.32. Generic Sequence - Command 3

Bits	Name	Description
31:0	CMD_VAL_H	This field stores higher part of the Mini Controller command

For Generic work mode the Command 1 register is not used. Allowed commands list can be found in the Table 2.34. Generic mode utilizes only Slave DMA interface for data transfer.

The CMD_VAL_H and CMD_VAL_L fields are concatenated into one 64-bit. The CMD_VAL_L is a least significant word of this command [31:0] and the CMD_VAL_H is a most significant word of this command. The command is send to the module responsible for execution of generic sequence. The following table define the layout of the 64-bit command.

Table 2.33. Command layout

Name	Bits	Description
Input	63:11	Additional values required in command sequence. It can be the command values. address values, number of data bytes to transfer. The required input to each sequence is defined in the Table 2.34 Bits that are not defined in the sequence should be set to zero. Bit 15 is common for all sequences. It is used as a ChipEnable hold indicator (ce_hold). When set high the controller will not turn off the active chip enable signal at the end of the sequence. The active chip enable will be turned off at the end of the sequence when ce_hold will be set low or when incomming sequence is targeted to different device (different CE). ce_hold should only be used for volume assignment process.Some devices may require to keep the CE# low through the whole SetFeature sequence. ce_hold can be used for this purpose also. During normal operation this bit shold be turned off. Some instructions with address phase defines the field No_of_BYTES. This field allow to select how many address bytes will be send to the NAND Flash device and with that supports devices with lower number of cycles of the row address (if sequence sends column address only the number of address cycles is fixed to two).: No_of_BYTES = 0 - one address byte send to device No_of_BYTES = 1 - two address byte send to device No_of_BYTES = 2 - three address byte send to device No_of_BYTES = 3 - four address byte send to device No_of_BYTES = 4 - five address byte send to device No_of_BYTES = 5 - six address byte send to device
Bank num	8	This field informs the minicotroller on which memory device the sequence need to be executed
jedec_supp	7	This bit informs the minicotroller which set of commands use in the sequence. Primary commands complaint with ONFI spec or Secondary commands complaint with Jedec spec. This bit is valid only for a few commands (see table Table 2.34) For commands that do not use this bit it should be set to 0.

Name	Bits	Description
tWB active	6	This bit informs the minicotroller if it has to wait for tWB after sending the last CMD/ADDR/DATA in the sequence. It is relevant for the sequences 0,1,2. For other sequences should be set to 0.
Instruction Type	5:0	This field defines the type of instruction. The values assigned to corresponding can be found in the Table 2.34

Commands like ReadStatus/ReadStatusEnhanced/ReadID must be directly followed by the data transfer. List of sequences available in the generic work mode is presented in the following table:

Table 2.34. Instruction types

Instruction Type	Instruction Name	Input [63:11]	Description
0	CMD Sequence	[23:16] CMD	Execute command sequence on the NF interface. Figure 2.6
1	ADDR Sequence	[63:56] ADDR5 [55:48] ADDR4 [47:40] ADDR3 [39:32] ADDR2 [31:24] ADDR1 [23:16] ADDR0 [13:11] No_of_BYTES	Execute address sequence on the NF interface. Up to 6 bytes of address can be send in one sequence. Maximum value of No_of_BYTES is 5 (six address cycles). Figure 2.7
2	Data Sequence	[58:56] corr_cap [55:40] last_sector_size [39:32] sector_cnt [31:16] sector_size [14] erased_page_detect_enable [13] scrambler_enable [12] ecc_enable [11] direction	Transfer the data over the NF interface. Figure 2.8 corr_cap - correction capability last_sector_size - number of data to transfer in the last sector. This value cannot be zero. If this value is set to zero, the data sequence will not be triggered. sector_cnt - defines the number of sectors to transfer within a single sequence. The overall number of data to transfer equals (sector_size * (sector_cnt-1) + last_sector_size).This value cannot be zero. If this value is set to zero, the data sequence will not be triggered. sector_size - number of data to transfer per each sectors except the last one. If sector counter is bigger than 1 then this value cannot be zero. If this value is set

Instruction Type	Instruction Name	Input [63:11]	Description
			to zero, the data sequence will not be triggered. If ECC engine is enabled the sector sizes must include the additional check-bits generated by the ECC engine erased_page_detect_enable: 1'b0 erased page detection is disabled 1'b1 erased page detection is enabled scrambler_enable: 1'b0 Scrambler is disabled 1'b1 Scrambler is enabled ecc_enable: 1'b0 ECC is disabled (only data transfer occurs) 1'b1 ECC is enabled (data and ECC data transfer occurs) direction: 1'b0 - read from NAND Flash device 1'b1 - write to NAND Flash device
3	Read command	[55:48] ADDR4 [47:40] ADDR3 [39:32] ADDR2 [31:24] ADDR1 [23:16] ADDR0 [13:11] No_of_BYTES	Execute 00-ADDR-30 command sequence +tWB. Allowed No_of_BYTES values are 3 and 4 (four and five address cycles). Figure 2.9
4	Write command	[55:48] ADDR4 [47:40] ADDR3 [39:32] ADDR2 [31:24] ADDR1 [23:16] ADDR0	Execute 80-ADDR command sequence +tADL or 81-ADDR +tADL for jedec_supp = 1. Allowed No_of_BYTES values are 3 and 4 (four and five address cycles). Figure 2.10

Instruction Type	Instruction Name	Input [63:11]	Description
		[13:11] No_of_BYTES	
5	Reset command	N/A	Execute FF command sequence +tFEAT. Figure 2.11
6	Erase command	[39:32] ADDR2 [31:24] ADDR1 [23:16] ADDR0 [13:11] No_of_BYTES	Execute 60-ADDR-d0 command sequence +tWB. Allowed No_of_BYTES values are 1 and 2 (two and three address cycles). Figure 2.12
7	Read status command	[11] F2_enable	Execute 70-sequence +tWHR. Figure 2.13 If jedec_supp = 1 the minicontroller sends the secondary command code F1h or F2h. The field F2_enable distinguish the type of command. F2_enable = 1'b0 - F1h command is sent to the device. F2_enable = 1'b1 - F2h command is sent to the device. This command must be directly followed by the data transfer.
8	Read status enhanced command	[39:32] ADDR2 [31:24] ADDR1 [23:16] ADDR0 [13:11] No_of_BYTES	Execute 78-ADDR sequence +tWHR. Allowed No_of_BYTES values are 1 and 2 (two and three address cycles). Figure 2.14 This command must be directly followed by the data transfer.
9	Read Cache random command	[55:48] ADDR4 [47:40] ADDR3 [39:32] ADDR2 [31:24] ADDR1 [23:16] ADDR0 [13:11] No_of_BYTES	Execute 00-ADDR-31 command sequence +tWB or 60-ADDR-3C +tWB for jedec_supp = 1. Allowed No_of_BYTES values are 3 and 4 (four and five address cycles).Figure 2.15
10	Copyback Read command	[55:48] ADDR4 [47:40] ADDR3 [39:32] ADDR2	Execute 00-ADDR-35 command sequence +tWB or 60-ADDR-35 +tWB for jedec_supp = 1. Allowed No_of_BYTES values are 1,2,3 and 4 (two,three,four and five address cycles) depending on the jedec_supp

Instruction Type	Instruction Name	Input [63:11]	Description
		[31:24] ADDR1 [23:16] ADDR0 [13:11] No_of_BYTES	setting. If jedec_supp = 0 the allowed No_of_BYTES values are 3 and 4 and if jedec_supp is 1 the allowed No_of_BYTES values are 1 and 2. Figure 2.16
11	Copyback Program command	[55:48] ADDR4 [47:40] ADDR3 [39:32] ADDR2 [31:24] ADDR1 [23:16] ADDR0 [13:11] No_of_BYTES	Execute 85-ADDR command sequence +tADL. Allowed No_of_BYTES values are 3 and 4 (four and five address cycles). Figure 2.17
12	Change Read Column command	[55:48] ADDR4 [47:40] ADDR3 [39:32] ADDR2 [31:24] ADDR1 [23:16] ADDR0 [13:11] No_of_BYTES	Execute 05-2xADDR-E0 command sequence +tCCS or 05-ADDR-E0 command sequence +tCCS for jedec_supp = 1. Allowed No_of_BYTES values are 3 and 4 (four and five address cycles). If jedec_supp is 0 the number of address cycles is fixed to 2.Figure 2.18
13	Change Read Column Enhanced command.	[55:48] ADDR4 [47:40] ADDR3 [39:32] ADDR2 [31:24] ADDR1 [23:16] ADDR0 [13:11] No_of_BYTES	Execute 06-ADDR-E0 command sequence +tCCS. Allowed No_of_BYTES values are 3 and 4 (four and five address cycles). Figure 2.19
14	Change Read Column Jedec	[55:48] ADDR4 [47:40] ADDR3 [39:32] ADDR2 [31:24] ADDR1 [23:16] ADDR0 [13:11] No_of_BYTES	Execute 00-ADDR-05-2xADDR-E0 command sequence +tCCS. Allowed No_of_BYTES values are 3 and 4 (four and five address cycles).Figure 2.20
15	Multi-plane read command	[55:48] ADDR4 [47:40] ADDR3	Execute 00-ADDR-32 command sequence +tWB or 60-ADDR for jedec_supp = 1. Allowed No_of_BYTES

Instruction Type	Instruction Name	Input [63:11]	Description
		[39:32] ADDR2 [31:24] ADDR1 [23:16] ADDR0 [13:11] No_of_BYTES	values are 1,2,3 and 4 (two,three,four and five address cycles). If jedec_supp = 0 the allowed No_of_BYTES values are 3 and 4 and if jedec_supp is 1 the allowed No_of_BYTES values are 1 and 2.Figure 2.21
16	Multi-plane block erase command	[39:32] ADDR2 [31:24] ADDR1 [23:16] ADDR0 [13:11] No_of_BYTES	Execute 60-ADDR-D1 command sequence +tWB. Allowed No_of_BYTES values are 1 and 2 (two and three address cycles). Figure 2.22
17	Multi-plane block erase command ONFI®-Jedec	[63:56] ADDR5 [55:48] ADDR4 [47:40] ADDR3 [39:32] ADDR2 [31:24] ADDR1 [23:16] ADDR0 [13:11] No_of_BYTES	Execute 60-ADDR-60-ADDR-D0 command sequence +tWB. Allowed No_of_BYTES values are 1 and 2 (two and three address cycles). Controller sends consecutive address bytes in both address sequences ie. if two address bytes need to be send in each address sequence controller sends ADDR0,ADDR1 in first sequence and ADDR2,ADDR3 in second sequence. For three address bytes case controller sends ADDR0,ADDR1,ADDR2 in first address sequence and ADDR3,ADDR4,ADDR5 in second address sequence. Figure 2.23
18	Change Write Column	[31:24] ADDR1 [23:16] ADDR0	Execute 85-tCWAW-2xADDR command sequence +tCCS. Figure 2.24
19	Change Row Address / Small data move command	[55:48] ADDR4 [47:40] ADDR3 [39:32] ADDR2 [31:24] ADDR1 [23:16] ADDR0 [13:11] No_of_BYTES	Execute 85-ADDR command sequence +tCCS. Allowed No_of_BYTES values are 3 and 4 (four and five address cycles).Figure 2.25
20	Synchronous Reset	N/A	Execute FC command sequence +tWB. Figure 2.26
21	Volume Select	[23:16] ADDR0	Execute E1-1xADDR command sequence +tVDLY. ???

Instruction Type	Instruction Name	Input [63:11]	Description
22	ODT configure	[23:16] ADDR0	Execute E2-1xADDR command sequence +tADL. ???
23	Set features	[23:16] ADDR0	Execute EF-1xADDR command sequence +tADL. Figure 2.27 Some devices may require to keep the CE# low through the whole SetFeature sequence. *ce_hold* can be used for this purpose.
24	Get features	[23:16] ADDR0	Execute EE-1xADDR command sequence +tWB. Figure 2.28
25	LUN get features	[31:24] ADDR1 [23:16] ADDR0	Execute D4-2xADDR command sequence +tWB. Figure 2.29
26	LUN set features	[31:24] ADDR1 [23:16] ADDR0	Execute D5-2xADDR command sequence +tADL. Figure 2.30
27	Read ID	[23:16] ADDR0	Execute 90-1xADDR command sequence +tWHR. Figure 2.31 This command must be directly followed by the data transfer.
28	Read Parameter Page	[23:16] ADDR0	Execute EC-1xADDR command sequence +tWB. Figure 2.32
29	ZQ calibration short	[23:16] ADDR0	Execute D9-1xADDR command sequence +tWB. ???
30	ZQ calibration long	[23:16] ADDR0	Execute F9-1xADDR command sequence +tWB. Figure 2.33
31	LUN Reset	[39:32] ADDR2 [31:24] ADDR1 [23:16] ADDR0 [13:11] No_of_BYTES	Execute FA-ADDR sequence +tWB. Allowed No_of_BYTES values are 1 and 2 (two and four address cycles).Figure 2.34

Figure 2.6. CMD sequence

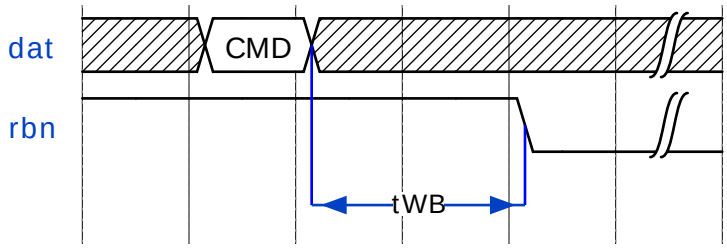
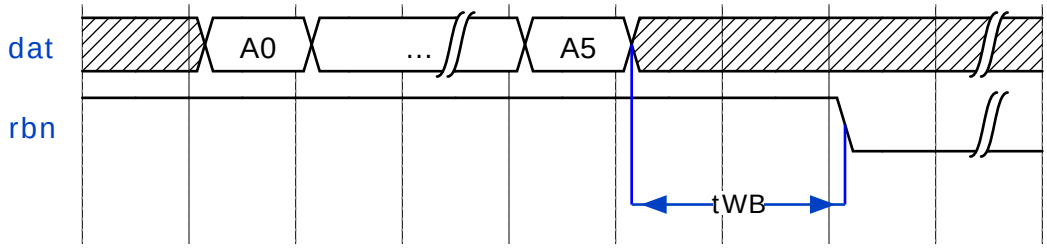


Figure 2.7. ADDR sequence



The controller transfers the number (n) of sectors defined in the *sector_cnt* field. First it transfers n-1 sectors of a size defined by the *sector_size* field. After that the controller transfers a sector of a size defined by the *last_sector_size*. If the *sector_cnt* is 1 then the sector of a size defined by the *last_sector_size* is transfered. If *sector_cnt* is 0 or *last_sector_cnt* is 0, the data sequence will not be triggered. If *sector_cnt* is grater than 1 and *sector_size* is zero, then the data sequence will not be triggered. Every sector starts with a new data word transfered from/to data FIFO (see *Data Layout*).

The controller automatically aligns the number of data written to / read from the NAND Flash device depending on the work mode. In 8bit SDR work mode the number of data is aligned to the 8bit. In 16bit SDR work mode the number of data is aligned to the 16bit.

The controller can skip the number of bytes during the transfer, i.e. the controller starts sending dummy bytes to the NAND Flash device (bytes defined by *marker* field in the *skip_bytes_conf* register). The number of bytes is defined by the *skip_bytes* field in the *skip_bytes_conf* register. The controller sends dummy bytes starting from offset defined by the *skip_bytes_offset*. The offset is counted from the beginning of the transfered data package. The controller does not take into account the column address when calculating offset. This feature is enabled when *skip_byte* field is not zero and *skip_bytes_offset* is not zero. Skip bytes is disabled in Generic Work Mode and in Set Feature command in PIO work mode.

Figure 2.8. Data sequence

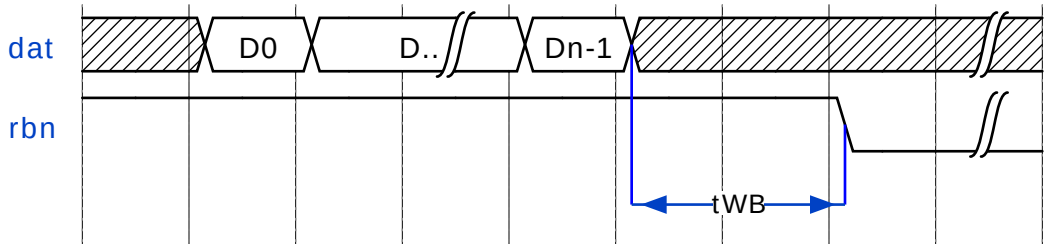


Figure 2.9. Read sequence

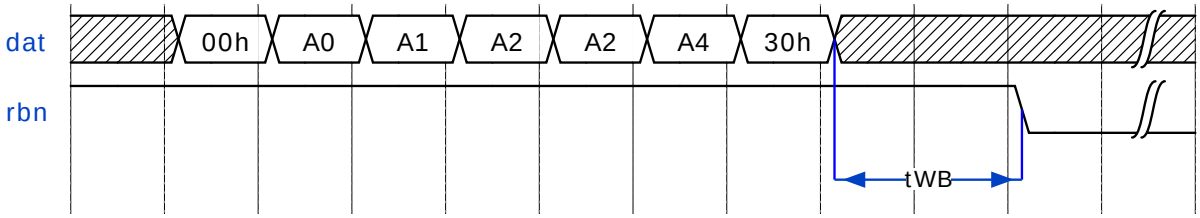


Figure 2.10. Write sequence

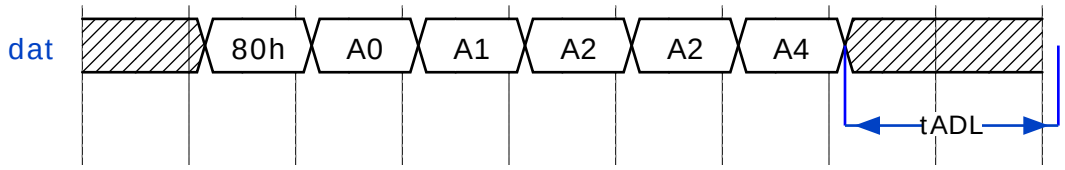


Figure 2.11. Reset sequence

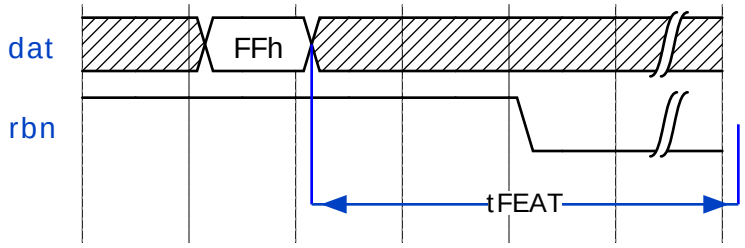


Figure 2.12. Erase sequence

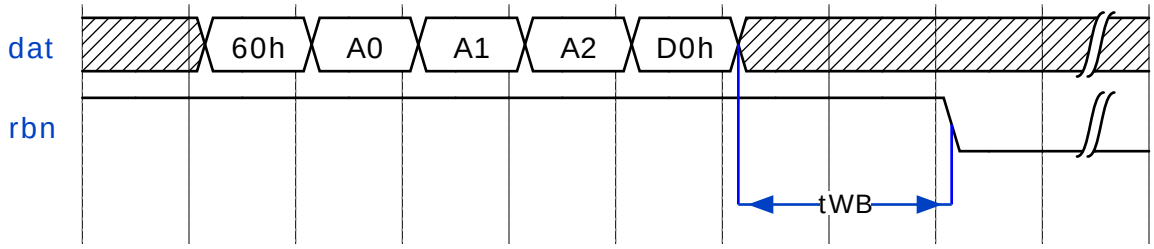


Figure 2.13. Read status sequence

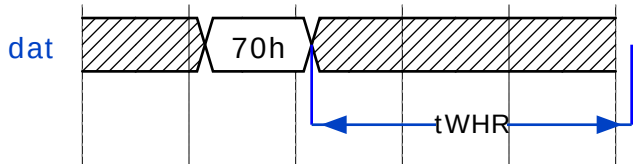


Figure 2.14. Read status enhanced sequence

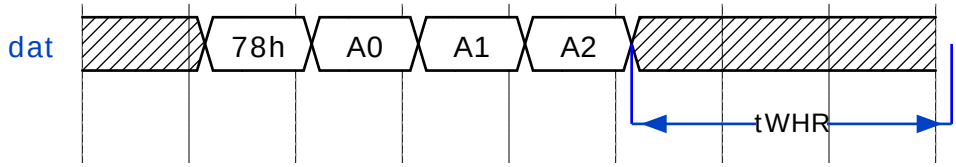


Figure 2.15. Read Cache sequence

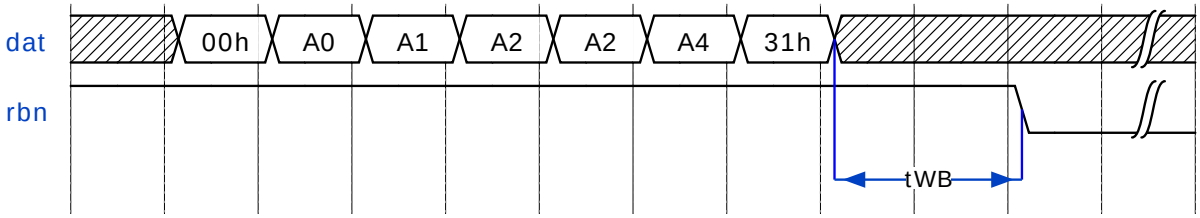


Figure 2.16. Copyback Read sequence

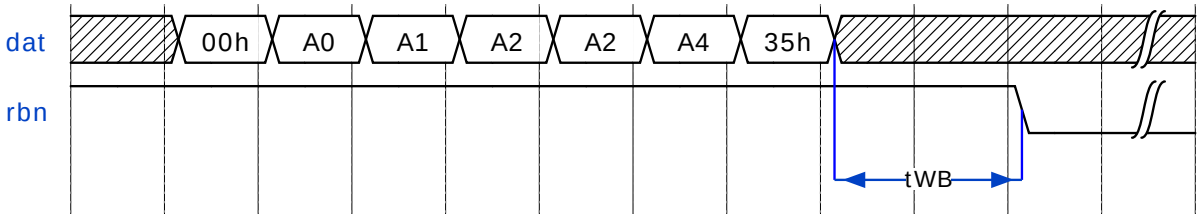


Figure 2.17. Copyback Program sequence

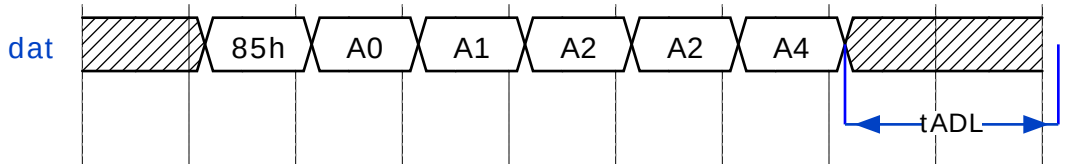


Figure 2.18. Change Read Column sequence

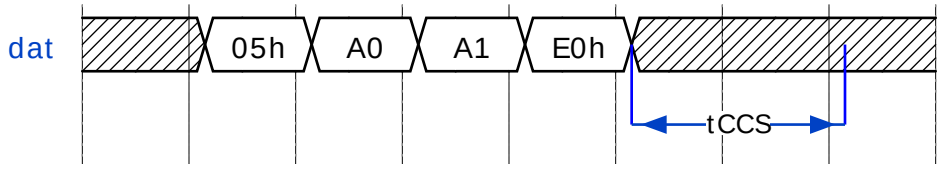


Figure 2.19. Change Read Column Enhanced sequence

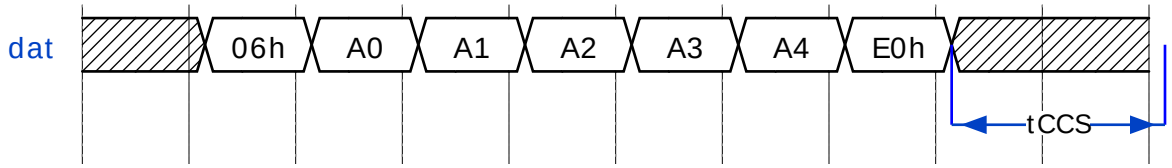


Figure 2.20. Change Read Column Jedec sequence

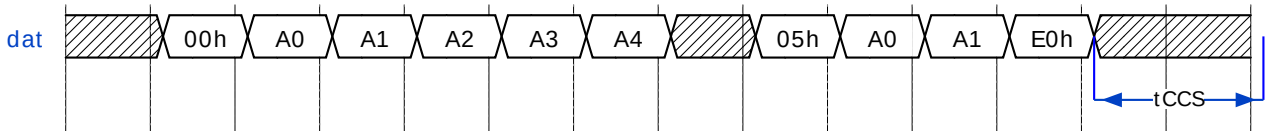


Figure 2.21. Multi-plane Read sequence

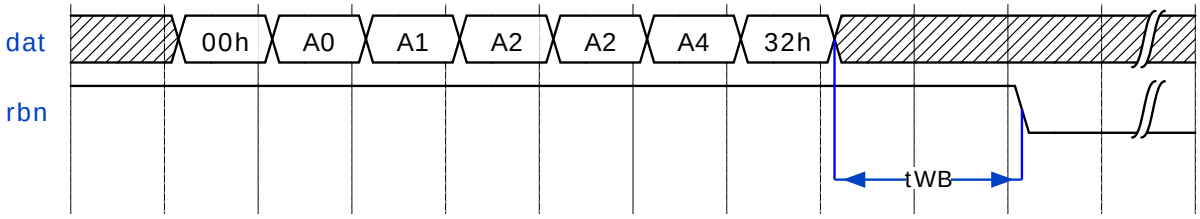


Figure 2.22. Multi-plane block erase sequence

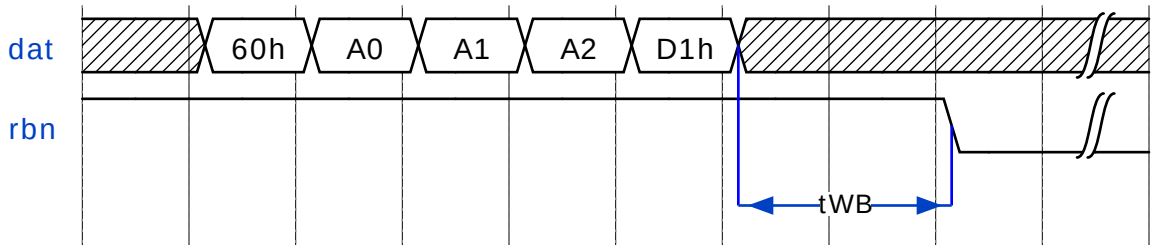


Figure 2.23. Multi-plane block erase sequence enh

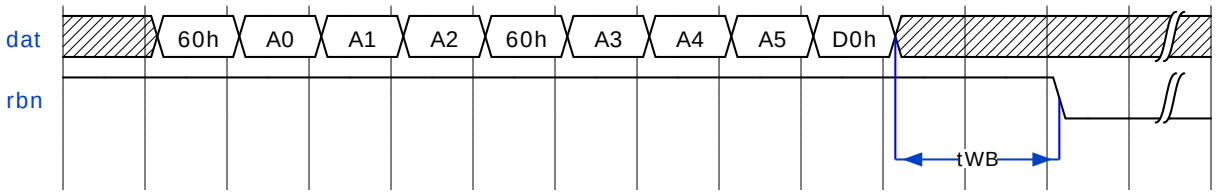


Figure 2.24. Change Write Column sequence

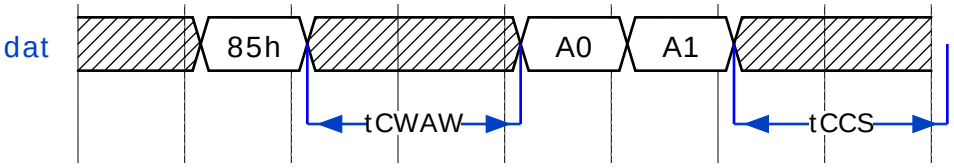


Figure 2.25. Small data move sequence

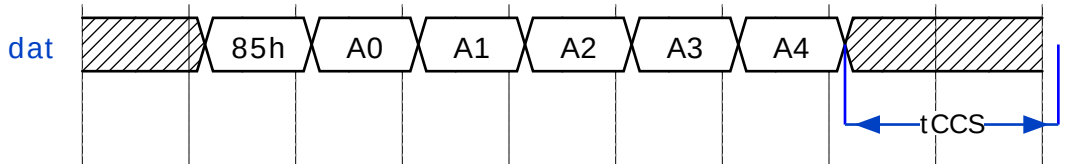


Figure 2.26. Synchronous Reset sequence

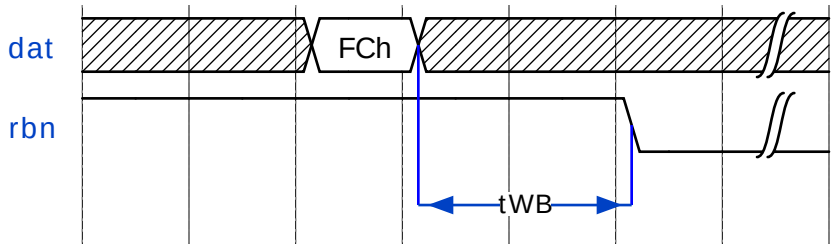


Figure 2.27. Set features sequence

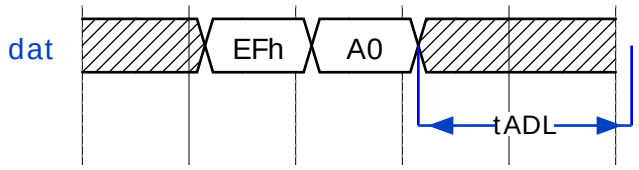


Figure 2.28. Get features sequence

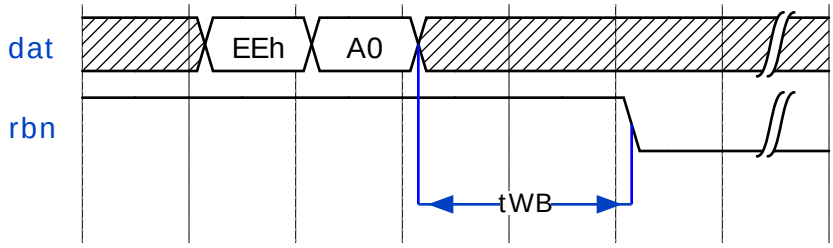


Figure 2.29. LUN Get features sequence

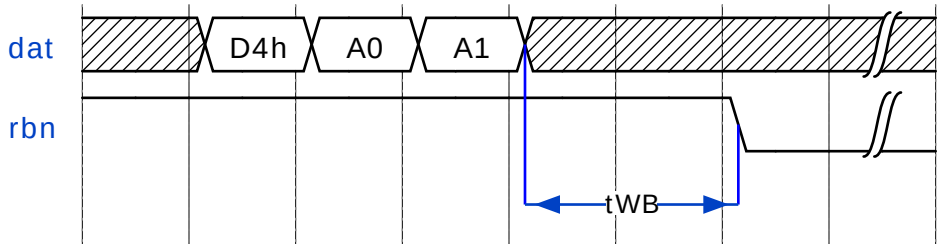


Figure 2.30. LUN set features sequence

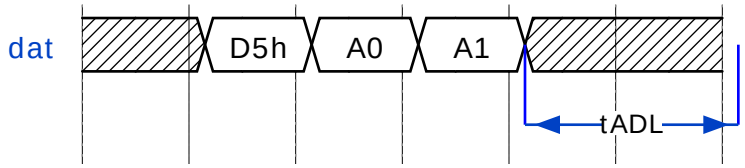


Figure 2.31. Read ID sequence

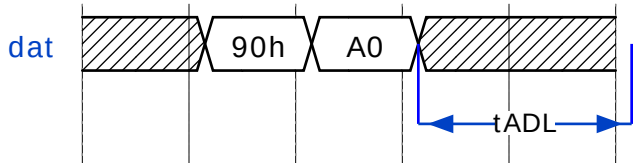


Figure 2.32. Read Parameter Page sequence

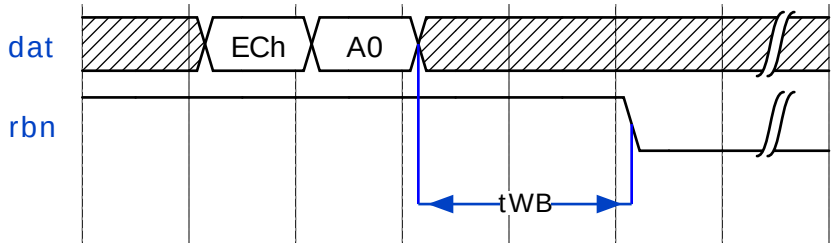


Figure 2.33. ZQ calibration long sequence

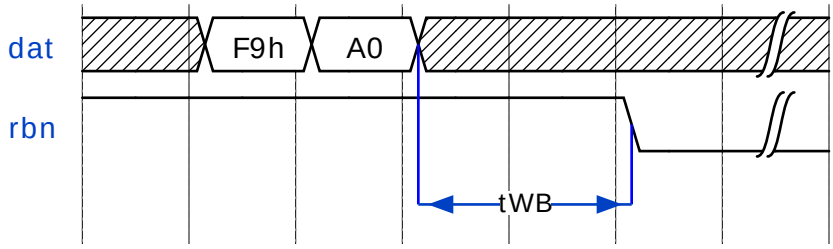
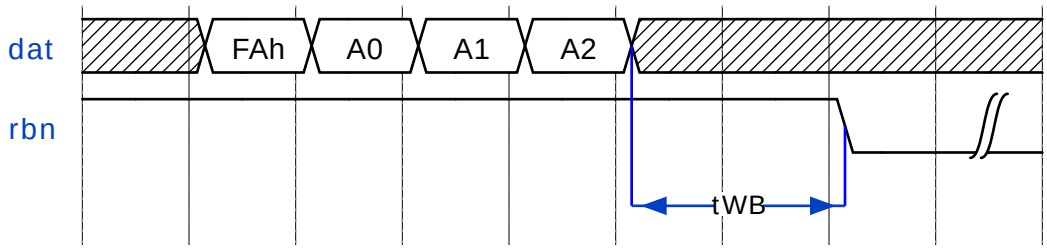


Figure 2.34. LUN Reset



2.6.2. Last Operation Status

The host can check latest operation status associated with the selected thread by reading the *cmd_status (0x0014)* register. This register contains command status from thread selected by value of the *cmd_status_ptr (0x0010)* register.

After a new command is triggered in selected thread the *Complete* bit is cleared and from this moment other register bits are invalid. When the *Complete* bit is set back then other bits in the status register are valid.

Please note that in Generic Work Mode selected thread finishes its work when appropriate command request is sent to the Mini-controller module. It may take some additional time to process the command by Mini-controller and send it to the NF device. If host needs to make sure that all commands were finished on NF interface, it can check controller operation status by polling the *ctrl_busy* bit in the *ctrl_status (0x0118)* register.

Table 2.35 shows layout and fields description for the status register.

Table 2.35. Status Field Description

Bits	Name	Description
31:17	Reserved	Reserved
16	Bus error	When set it informs that the controller got an error response on the system bus.
15	Complete	When set, denotes that the controller has updated status information and the operation is complete. This bit shall be set even if the operation ended as a failure. This bit should be in cleared state while descriptor is constructed. If this bit is set when the controller reads the descriptor for execution, the controller will skip execution of the descriptor and continue with execution of further descriptors in the chain depending on the state of the 'continue' bit.
14	Fail	When set, denotes that operation failed to complete successfully.
13	Reserved	Reserved
12	Reserved	Reserved
11	Erased Page	When set, denotes that the controller detected an erased page in the read transaction. The detection of erased page is based on the number of 0's in a page. If the number

Bits	Name	Description
		of 0's in a page being read is less than the value on the *erase_det_lvl* field of the *ecc_config_1 (0x042c)* register, an erased page is inferred and no uncorrectable error will be flagged for that page. If ECC is disabled, the erased_page interrupt shall be set as explained above. If ECC is enabled, in addition to the above condition, only when the ECC logic detects no errors or correctable error pattern for that page will the erased_page interrupt be flagged. If the ECC logic detects a uncorrectable error page, this erased page interrupt will not be set. This flag doesn't contribute to the fail flag.
10	Reserved	Reserved
9:2	Max Error	For a Flash read command, this field indicates the maximum amount of correction applied to one ECC sector. This field is of significance only if the read transaction resulted in correctable errors. If no errors were found, this field will read zero.
1	ecc_err	Uncorrectable ECC error was detected for the any of the device planes.
0	Descriptor Error	This bit denotes that an invalid descriptor sequence has been detected.

2.6.3. Switching between sequences

Generic work mode provides an interface to execute low level transaction on the NAND flash interface. In this work mode any automatic control sequences will not be executed, so it is the host software's responsibility to enter and leave generic mode when target NAND flash device is in the idle state.

Before host switches to the generic mode it should wait for all operations that address the same NAND flash device to be finished. This will avoid the situation when two operation try to reach the same target.

Before host switches to a mode other then generic it should check if device is in the idle state. If the latest generic command caused the NAND flash device go to the busy state then host should do the *rbn_settings (0x1004)* register polling or execute the read status sequence/read data sequence until device reaches idle state.

2.7. Extended commands support

2.7.1. Chip and LUN Interleaving

Cadence NAND Flash Controller supports Chip select and LUN interleaving to achieve the highest possible performance.

This section describes the Chip select and LUN interleaving feature. The following topics are covered in this section.

- Best case scenarios
- LUN Interleaving and Command DMA

The Chip and LUN interleaving maximizes the activity on the bus interface to the devices. When operation has begun on a particular chip or LUN, the controller can target some of the other free chip or LUN to increase the overall performance of the system.

To get the best performance of the Controller, the host uses command engine multiple threads feature in such a way that triggered commands target different chip selects or LUN-s and maximize the possibility of chip and LUN interleaving.

For example, if the host sends a command to read 'n' pages to chip select 0 or LUN 0 on Thread 0, it is recommended that the host sends another command to a different chip select or LUN on another Thread, so that the Controller can interleave among the chip selects or LUN-s to improvise the performance of the system. If, at all, the host sends multiple commands on different threads targeting the same chip select or LUN, then the execution will happen sequentially and will dramatically reduce the performance of the system.

The interleaving process is essentially handled in the command engine, which looks at the next available free chip select or LUN to send an outstanding command to it.

For example, if the host sends the following four data read commands through multiple threads, the controller on receipt of each of these data commands will send a load command to the first targeted addresses to each of the chip selects or LUN-s.

1. Read of “n” pages to chip-select0/LUN0
2. Read of “m” pages to chip-select0/LUN1
3. Read of “o” pages to chip-select1/LUN0
4. Read of “p” pages to chip-select1/LUN1

On a detection of a load complete on any of the LUN-s, the controller will begin data transfer on that LUN. Once the entire page is read, it will issue the load command for the next page on that LUN and will wait for the load completion on any other LUN. Again, when the controller detects a load completion on any of the LUN-s, it proceeds to transfer data on that LUN. This process continues until there are no more pages to be read.

The same process happens for program commands too. Whenever the controller detects a free LUN to send a page program to, it will initiate a data transfer to that LUN and start the program operation.

Operations such as erase, Copyback, or data transfer are naturally interleaved by the controller. That is, it is possible to have an erase on one chip select, Copyback on the second, and a data transfer into the third for program/read operation.

The controller can handle read/write commands at the same time. The host may, if it chooses to, send a data command to read from a particular chip select/LUN and may also send a data command for write to a different chip select/LUN.

Note

The ONFI Specification places certain restrictions on interleaved LUN transfers. The Specification states that no program operation should be initiated to any LUN in a chip select if a read operation is ongoing in another LUN selected by this chip select. The Controller takes care of this restriction in the command engine. However, to get the best performance from the Controller, it is recommended that the host should not send read/write data commands to the same chip select, different LUN-s at the same time.

It is possible to initiate a program operation when a read operation is ongoing in other LUN if the *program_after_read* bit in the *lun_interleaved_cmd (0x041c)* register is set by software.

When multiple data commands are issued to the same chip-select or LUN, the Controller will execute them in the order they have been received.

2.7.1.1. Best case scenarios

The following are the Best Case Scenarios for achieving high performance using Interleaving:

- Only read data commands from a different chip selects/LUN-s
- Only write data commands to a different chip selects/LUN-s

2.7.1.2. LUN Interleaving

Different threads of the command engine should be binded to individual LUN-s of the NAND Flash devices connected to the controller. This will ensure that the each of command engine threads operates in parallel, which results in the flash operations on different threads occurring in parallel on each LUN. This effectively implements the LUN interleaving functionality.

It should be noted that the interleaving of flash operations across LUN-s will only be performed if permitted under the restrictions mentioned previously.

During LUN interleaving operations, the ready/busy pin is not monitored by the controller. The controller needs to read the status directly from each LUN. To get the best performance from the controller, host should carefully program the *long_polling (0x0408)*, and *short_polling (0x040c)* registers.

The LUN interleaving functionality is based on the following three programmable registers:

- The *lun_addr_offset (0x0420)* register - The Controller is configured to support multiple LUN-s so it can interleave between them. Controller assumes that each LUN is a separate independent communication thread. To address different LUN-s in a device, the controller needs to know the targeted chip enable and the row address that is being targeted. The *lun_addr_offset (0x0420)* informs the Controller about the boundary index between LUN and block/page addresses. After reset, value of this register gets the Controller to consider the device connected as a single LUN device.

- *lun_status_cmd (0x0418)* - NAND devices have different ways to read the status of individual LUN-s when LUN interleaving operations are in progress. In the ONFI devices, the `READ_STATUS_ENHANCED_CMD` is used, where the device expects a 0x78 command for status reads for each of the LUN-s. Alternatively the 0xF1/0xF2 commands could be used to read the status of LUN0/LUN1.

  The Controller after reset sets the *lun_status_cmd (0x0418)* register to 0 (defaulting to ONFI device - command code 0x78). Software can change the value in this register depending on the requirements of the device connected, if necessary.

- *lun_interleaved_cmd (0x041c)* - This register informs the controller what are the valid combinations of command during Multi-LUN operation on the device. The device might have a variety of restrictions as far as LUN interleaved command combinations are concerned. Please refer to the section "Register Description" for detailed information on this register fields and expected programming by software. The software should program these registers as per device command support for LUN/Die interleaving.

  ### Note

  ONFI devices place no restrictions on command combinations for Multi-LUN sequences. So this register does not need to be programmed for ONFI devices and can be left at default values.

2.7.1.3. Notes

The CMD DMA logic implements Device status polling to figure out a device operation completion. Based on single or multi-LUN operations, single/multi-plane commands, appropriate read status commands are sent to the device from the controller. In certain rare traffic scenarios, the controller might send 1 or 2 extra status read to device even if device status received by the controller was a COMPLETED one in the previous status read. This is not related to the device specification but to the internal state machine requirement. While handling interleaving of operations across numerous threads and LUN-s, the CMD DMA state machine, in the aforesaid scenario, sends an extra status read. This is not going to affect performance. Even though an extra status read is taking up device bus time, on the other hand, the internal state machine is saving several cycles of RAM-read and internal computation time in such a scenario. Thus overall performance is not affected.

2.7.2. Multi Page Commands

The host can set up the controller to perform operations on multiple contiguous pages. Command Engine intelligently sequences each page operation on the device starting from the page address provided as command argument or descriptor parameter and moves on to the next page until all pages are read/written. There are no restrictions on the row address boundaries.

It is expected that page address will be aligned to plane boundaries, if multi-plane operations are enabled. Number of transferred pages needs to be a multiple of planes of the NAND Flash device.

For a Flash program operation, data to be fetched from the host memory is expected to be in contiguous memory locations for all the pages starting from the address provided as command argument or descriptor field. Similarly, for reads from Flash device, DMA engine makes the read data for contiguous pages available in contiguous locations in host memory.

For Copyback commands, both the source page address and destination page address are incremented by 1 after each Copyback operation till the specified number of pages has been completed. The odd to even page Copyback operation can be only trigged when connected NAND flash devices support it.

2.7.3. Cache Commands

Controller supports intelligent cache Read and Program commands in both CMD DMA and PIO modes. For devices that support cache read sequences, *cache_rd_en* bit in the *cache_config (0x0438)* register must be set. Cadence NAND Flash Memory Controller will then sequence the multi-page read commands as cache read sequence.

For devices that support cache program sequences, *cache_wr_en* bit in the *cache_config (0x0438)* must be set. Cadence NAND Flash Memory Controller will then sequence the multi-page program commands as cache write sequence.

2.7.4. Multi-Plane Commands

The NAND Flash Controller in Command DMA mode supports interleaved operations across multiple planes if the device has multi-plane operation support. For controller to execute any operation as a multi-plane operation, the following should be confirmed:

- The host should verify or set the *mpl_pl_num* field of the *multiplane_config (0x0434)* register to the correct planes number value.
- The *mpl_rd_en* and/or *mpl_wr_en* bit in the *multiplane_config (0x0434)* register should be set.
- The Flash pointer in the descriptor should be as required by the device.

If the above conditions are all set, the total number of pages in the command should be a multiple of the total number of planes in the device. If the controller detects the number of pages information to be otherwise, then:

- In CMD DMA work mode the *Descriptor Error* bit will be set in the status field and the descriptor will be dropped from execution.
- In the PIO work mode the *Command Error* bit of the *cmd_status (0x0014)* register (for selected thread) will be set and the command will be dropped from execution.

The block address specified (in the Flash pointer) must be for plane 0 block address of the device. During a multi-plane read/program or Copyback operation the block address will be incremented by 1 starting from the address present in the Flash pointer up to to a number of planes declared in the *mpl_pl_num* field of the *multiplane_config (0x0434)* register and start again from Flash pointer block address for every sweep. The pages numbers will be incremented after each sweep of the number of planes present in the device till the total number of pages to be transferred are completed.

This type of multi-plane operation is named a full restriction mode

Certain devices support dedicated multi-plane read command sequences to read data in the same fashion as is written with multi-plane program commands. The *mpl_rd_en* bit in the *multiplane_config (0x0434)* register should be set accordingly for the those devices. When this register is not set but the conditions mentioned in the above section for a multi-plane operations are satisfied, reads in multi-plane mode will still happen in the order of multi-plane writes, though normal read command sequences will be issued to the device.

For erase operations, the controller sends the erase commands to each plane one after the other for all the available planes. This will kick off the erase operations for all the planes simultaneously in the device.

The controller can also execute Multi-plane Cache Reads and Writes on the device. If the total number of pages in the descriptor command are more than the total number of planes (but still a multiple of the number of planes), and the *cache_rd_en* and/or *cache_wr_en* bits in the *cache_config (0x0438)* register are set up accordingly, controller can intelligently achieve Multi-plane cached Read or Write sequences.

2.8. Error and Special Situation Handling

This section will describe how controller handles special and error situations. In the following sections each possible error type will be described. The Table 2.36 summarizes handled error situations and links them with more detailed description.

Table 2.36. Errors - List with links.

Name	Link
System Bus Error	Section 2.8.1, *"Bus Error"*
Device Error	Section 2.8.2, *"Device Error"*
Uncorrectable ECC error	Section 2.8.3, *"ECC errors"*
Descriptor/Command Error	Section 2.8.4, *"Descriptor/Command Error"*

2.8.1. Bus Error

2.8.1.1. Description

Controller can detect incorrect transaction on the system slave and master buses.

For the master interface, incorrect transaction will be detected when host side return error response in the response transfer phase. The master interface is used by both: Master DMA and Command Engine.

> **Note**
>
> If error on system bus will occur during descriptor read process and data bus was compromised then Command Engine can detect incorrect descriptor format and set *Descriptor/Command* error flag.
>
> If error on system bus will be detected during status write then information about this error will be passed only to the controller status registers.

For the slave interface, incorrect transaction will be detected if one of the following conditions is met:

- Host accesses the slave interface before transfer on this interface was triggered.
- Host uses incorrect burst type. The slave DMA will access incremental burst only.
- Host forces burst length equal zero.

2.8.1.2. Controller Actions

If Master DMA drives master interface and detects incorrect transaction then *ddma_terr* bit in the *intr_status (0x0110)* register will be set. Additionally address of transfer that caused error will be written to the *dma_target_error_l (0x0140)* register. If the *ddma_terr_en* bit is set in the *intr_enable (0x0114)* register then additionally the interrupt will be raised.

If slave DMA interface and incorrect transaction occur then the *sdma_err* bit in the *intr_status (0x0110)* register will be set. If the *sdma_err_en* bit is set in the *intr_enable (0x0114)* register then additionally the interrupt will be risen. If an incorrect transaction is detected during data transfer, then the current transfer will be internally aborted.

If Command Engine drives master interface and detects incorrect transaction then *cdma_terr* bit in the *intr_status (0x0110)* register will be set. Additionally address of transfer that caused error will be written to the *dma_target_error_l (0x0140)* register. If the *cdma_err_en* bit is set in the *intr_enable (0x0114)* register then additionally the interrupt will be raised. If error on the system bus is detected during descriptor fetch or sync phases then command execution will be interrupted and the following descriptors in chain will be dropped. It happen independently from setting of the cont_on_err field in the *device_ctrl (0x0430)* register. In this case controller can't expect reliable data from the system side.

Additionally the *Bus Error* and *Fail* flags are set in the last operation status descriptor field or controller register. If error source isn't fetched or sync operations and *cont_on_err* field in the *device_ctrl (0x0430)* register is cleared then for CMD DMA work mode following descriptors will be dropped from execution, in opposite case controller will continue command execution.

2.8.1.3. Host Actions

No special action from the host side is required.

2.8.2. Device Error

2.8.2.1. Description

Common Flash operations like read, write or erase are prone to device errors. When the command engine module issue such operations for execution on the device, the device might report an error. While command engine is waiting for the Flash operation to complete, it can get information that the operation failed on the device.

The NAND Flash controller provides mechanism to configure what part of read status word will be interpreted as NAND Flash memory error information. The mechanism is base on the mask field and value field. The read byte value is logically AND-ed with mask and this operation is compared with value field. If both parts are equal, the memory ready state or memory error are detected. The mask and value for memory error status read operation are provided as fields: *error_mask* and *error_value* of the *rdst_ctrl_1 (0x0414)* register. Controller doesn't apply any implicit checks, so host need to unmask all status bits that need to be checked to detect error condition. Device error will be probed only when command interface is used to detect device status. The command device status checking mechanism is selected by clearing *rb_enable* bit in the *rdst_ctrl_0 (0x0410)* register.

For the multi-plane operation, error index for this error will inform in which set of planes error was detected.

2.8.2.2. Controller Actions

Controller will set *Device Error* and *Fail* flags in the last operation status descriptor field or controller register. If the *cont_on_err* field in the *device_ctrl (0x0430)* is cleared then for CMD DMA work mode following descriptors will be dropped from execution, in opposite case controller will continue command execution.

2.8.2.3. Host Actions

No special action from the host side is required.

2.8.3. ECC errors

2.8.3.1. Description

Only the uncorrectable errors are considered as errors by the controller. An uncorrectable error is reported when the BCH engine detects that number of errors overpass used algorithm correction ability. For correctable errors, controller will not report an ECC error.

2.8.3.2. Controller Actions

Controller will set *ecc_err* and *Fail* flags in the last operation status descriptor field or controller register. If the *cont_on_err* field in the *device_ctrl (0x0430)* is cleared then for CMD DMA work mode following descriptors will be dropped from execution, in opposite case controller will continue command execution.

If correctable errors were detected then controller will update the *Max Error* field of the last operation. This field will hold maximum number of errors corrected per single sector.

2.8.3.3. Host Actions

When uncorrectable error is reported then number of detected errors (field *Max Error*) and Erased Page flag (bit *Erased Page*) need to be ignored.

2.8.4. Descriptor/Command Error

2.8.4.1. Description

Descriptor/Command error occurs when Command Engine detects any issue with command or descriptor that makes its execution impossible. Possible error sources are:

- Multi-planes operation does not start at address of the first plane.
- Number of page per command programmed in the multi-plane work mode is not a multiple of the number of planes
- The Bank field of the Flash Pointer for source and destination address is different.
- For the Copyback operation when multi-plane operations are enabled only for read or write direction. The *mpl_wr_en* and *mpl_rd_en* fields of the *multiplane_config (0x0434)* register need to have the same value or descriptor/command error will be returned.
- When 16-bit SDR mode is selected and transfered data block size isn't aligned to 16-bit word. Both *sector_size* and *last_sector* size fields of the *transfer_cfg_1 (0x0404)* need to be aligned to 16-bit.
- If in the generic mode a instruction type number exceed the last available item.
- Unknown Command - it occurs when the Command Engine cannot identify the command programmed by the host software.

2.8.4.2. Controller Actions

Controller will set *Descriptor Error* and *Fail* flags in the last operation status descriptor field or controller register. If the *cont_on_err* field in the *device_ctrl (0x0430)* is cleared then for CMD DMA work mode following descriptors will be dropped from execution, in opposite case controller will continue command execution.

2.8.4.3. Host Actions

No special action from the host side is required.

2.8.5. Error Interrupt Generation

If an error condition is detected during command execution then the *Fail* bit is set in the last operation status descriptor field or controller register in parallel to the specific bit that identifies what was the error cause. Host can unmask interrupt generation as response for this kind of error. To make it happen the thread error interrupt needs to be unmasked in the *trd_error_intr_en (0x0130)* register. After this if error is detected then bit in the *trd_error_intr_status (0x0128)* register will be set. Each Command Engine execution thread has single bit in both registers.

If the *Fail* bit is set in the last operation status descriptor or register then host side can obtain information which operation in command caused issue by reading the *Error Index* field in the same descriptor field/register. The *Error Index* indicates the operation number where the first error was detected.

Note

The *Error Index* field isn't updated when source of error is a transfer on the system bus - in this case only the *Bus Error* bit is set in the last operation status field/register.

2.8.6. Recommended Software Work Flow

The following is the recommended software flow to interpret the status fields in the descriptor for various operations. Multiple error bits can be set in these cases. If there is a *Descriptor error* no other error bits are set.

1. If: Complete = 1 then: Operation complete. This bit will be set for errors also.
2. All cases are equally important possible options are:
   - If Fail = 1 and Descriptor error = 1 then: Indicates there was an error in the descriptor.
   - Fail = 1 and Bus error = 1 then: Error response on the system bus.
   - Fail = 1 and ECC_error = 1 then: Uncorrectable Error.
   - Fail = 1 and device_error = 1 then: Error from device read status operation.

2.9. Thread Reset Commands

Thread Reset command will abort command processing of a particular thread. In the Command DMA work mode host can use this command to stop a thread from processing the next descriptor in the descriptor chain. After issuing a reset command, the software can poll *trd_busy* field of the *trd_status (0x0120)* register to determine when thread have aborted operations and are idle.

There are two kinds of thread reset commands that can be issued to the command Engine:

1. Thread Reset Type 0 - It is allowed for Command DMA work mode only. For PIO mode it behaves in the same way as Thread Reset Type 1 command.
2. Thread Reset Type 1 - It is allowed in both work modes.

Thread Reset is essentially a PIO command to the Command engine with the following construction:

Table 2.37. Thread Reset Command - Command 0

Bits	Name	Description
31:30	CT	This field needs to have the 2'b10 value.
29:27	-	Reserved
26:24	TRD_NUM	This field selects destination thread number for command. Software can select any available thread. Commands can be issued in parallel to all threads
23:22	-	Reserved
20	INT	If this bit is set, an interrupt should be issued after this operation will be finished. The triggered interrupt will be n bit of the *trd_comp* field in the *trd_comp_intr_status (0x0138)* register. where n will be thread number selected by the TRD_NUM field.

Bits	Name	Description
19:16	-	Reserved
15:0	CMD_TYPE	This field identifies the kind of operation that will executed by the controller. For the Reset command this field encoding is as follows: 0x0200 - for Thread Reset Type 0 and 0x0201 for Thread Reset Type 1.

2.9.1. Thread Reset Type 0

This Command is allowed only in the Command DMA work mode. When a thread Reset of type 0 is received, the present working command in the thread is completed and no further descriptors will be executed by the thread. The status field of the descriptor is updated with the associated status.

Sync associated with a descriptor will not be executed if the thread reset command is received by the thread while waiting for a sync condition to get satisfied. If the thread is waiting for sync condition to be met and receives the thread reset command, it immediately exits the descriptor operation and returns to the IDLE state.

2.9.2. Thread Reset Type 1

This command is allowed in Generic, PIO, and Command DMA work modes. When a thread reset of type 1 is received, the present working command in the thread is aborted. For Command DMA work mode no further descriptors in chain will be fetched and executed.

Data path is initialized automatically, so host side does not need to receive or provide data after reset command was sent. After the thread reset 1 command was sent, then following data transfer to reset thread can cause the sdma_err bit in the intr_status register to be set.

The command/descriptor status and sync flags will be updated normally. After thread back to IDLE state host needs to send RESET command, to put the NAND Flash device in well known state. Additionally software should ignore all status flags reported between sending Thread Reset Type 1 and finishing thread activity.

If a Thread Reset 1 is triggered when a Slave DMA request is outstanding, the address in the sdma_addr registers may be that for any page of the command.

2.10. Time Out Functionality

2.10.1. Introduction

The Command Engine has an embedded watchdog module. This module measures programmed delay after which it will trigger an interrupt that informs the host side that operation started in the Command Engine did not finish in the required time period. It will not trigger any initialization process automatically to initialize command engine, since such an operation should be conducted by the host.

2.10.2. Configuration Flow

The watchdog module is enabled by default. It can be disabled by clearing the *time_out_en* bit of the *device_ctrl (0x0430)* register.

The time out counter is driven by the *sys_clk* clock. The time out period is configured by the *time_out_val* field of the *time_out (0x0448)* register. Its default value is 0xFFFFFFFF. If host configure value 0 to the *time_out_val* then time out feature is disabled.

The *cont_on_err* field in the *device_ctrl (0x0430)* register doesn't affect time out functionality.

2.10.3. Work flow

The time out counter is triggered every time a new command is written to the Command Engine thread. The time out period is counted separately for each thread.

When time out condition for a given thread is met then Command Engine will set bit corresponding to the thread number in the *trd_timeout_intr_status (0x014c)* register.

The host side can enable interrupt generation as response on the error condition in execution thread by setting bit that corresponds to the thread number in the *trd_timeout_intr_en (0x0154)* register. In this case interrupt will be returned when time out condition is met.

The thread for with timeout condition was detected need to be initialized by host to put it in known condition. This can be achieved by sending thread reset 1 command to this thread or hardware reseting whole controller at most appropriate moment.

To resume operation on the target for which the time out condition was detected, the host side needs to put this target in known condition by sending a RESET command to this target.

The watchdog module is disabled after last active thread returns back to the IDLE state.

3. ECC Engine

This chapter describes the ECC engine, its usage and restrictions. The following topics are discussed in detail in this chapter.

- Error correction
- ECC Sector sizes
- Bad Block Marker
- Scrambling Support

Cadence NAND Flash Memory Controller has ECC engine logic that allows detection and correction of the multiple errors in the data stream read from the NAND Flash device.

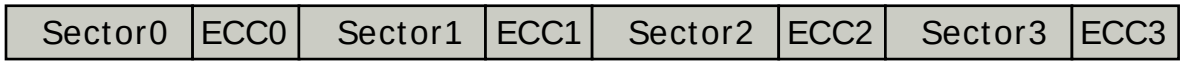
ECC check bits are inserted after each transfered ECC sector. For this ECC engine configuration supported ECC nominal sector size is: 512 bytes.

For the write direction, transferred data goes through the ECC encoder to the NAND flash device. After data block equal to the ECC sector is transferred, then ECC check sums are injected into the data stream and both user data and ECC data are written into the NAND flash device.

For the read direction, transferred data goes through the ECC decoder and it is buffered in case if error correction will be needed. After data block equal to the ECC sector is transferred, then ECC check sums are stripped from the user data and are used to calculate if errors were present in the data stream. If errors were present then the location and correction mask are calculated. Using this information the ECC engine makes error correction on previously buffered data. Corrected data is transferred to the host interface.

The figure below shows the NAND flash device page organization.

Figure 3.1. Page Organization for Flash Devices



The correction capability and the ECC sector size can be configured for different combinations. Customers need to provide the required configuration, based on which delivery RTL is created. Please contact Cadence for any specific required configuration.

It is possible to have a combination of different correction capability and ECC sector size. In this case correction strength can be selected with the *ecc_config_0 (0x0428)* register. Sector size is determined by the value of the *transfer_cfg_1 (0x0404)* register.

The number of check bits written into spare area depends on the required correction capability and the maximum ECC sector size available in controller configuration. The table Table 3.1 shows available settings of the ECC correction strength and possible range of the sector size for selected configuration. It also shows coded value which should be written to corr_str field of the *ecc_config_0 (0x0428)* register to select desired correction strength.

Table 3.1. Available Correction Capability

Correction	Sector Size in Bytes	Check-bit size in bytes	The 'corr_str' field value
4	512 - 544	8	0

Correction	Sector Size in Bytes	Check-bit size in bytes	The 'corr_str' field value
8	512 - 544	14	1
12	512 - 544	20	2
16	512 - 544	26	3
32	512 - 544	52	4

Note

Because of the used algorithm proprieties the ECC engine guaranties the correction ability up to 32 bits.

Note

The total number of check-bytes per page will depend on the page size.

3.1. Error correction

Cadence NAND Flash Memory Controller corrects the read data before transferring the data. There are ECC sector buffers incorporated per device connected to a chip select. These buffers store the read data from the device. After a sector worth of data is read into the buffer, the ECC engine kicks-off an error calculation procedure. The ECC engine searches for error in the sector, when error correction is completed data is transferred out of the Controller. Successive data read from the device are pipe-lined in the ECC sector buffer and ECC error engine to achieve line rate of operation.

Single data sector is marked as error free (if there were no errors detected in sector), correctable (if number of errors found was not greater than selected error correction capability) or as uncorrectable (when there was more errors than selected correction capability). When number of errors in sector exceeds selected correction capability an uncorrectable error pattern should be reported, however it is possible that decoder will behave in random manner (reporting correctable pattern with wrong error locations or reporting no errors in sector). Probability of such random behavior is greater when number of errors exceeds correction capability by more than 1 error and for smaller correction capabilities.

The status field of descriptor - for Command DMA work mode or *cmd_status (0x0014)* register for PIO mode contains ECC correction information for latest operation.

At the end of data correction for the transaction in progress, this field or register will hold the maximum number of corrections applied to any ECC sector in the transaction. In addition, this register will indicate whether the transaction as a whole had correctable errors, uncorrectable errors, or no errors at all. A transaction will have no errors when none of the ECC sectors in the transaction had any errors, transaction will be marked uncorrectable if any one of the sectors is uncorrectable, and transaction is correctable when any one sector had correctable error and none of them is uncorrectable. Transaction will be marked as erased if all blocks in transaction were identified as erased.

At the end of each transaction, it is important for the host to read this field or register. The value in this register will give the host an idea of the health of the block. The host can take corrective action once the number of correctable errors encountered has reached a particular threshold value.

For achieving line rate correction of incoming data, the required size of ECC buffer depends on the max ECC sector size configured and the ECC correction level.

The ECC sector buffer works in a ping-pong function. The following two simultaneous processes occur on the ECC sector buffer:

- From one sector space in the buffer, data is read out and sent to host after applying correction.
- In the other sector space, data is received from the device and staged for error computation.

The above two processes alternate between the sector spaces to achieve line rate operation.

3.2. Sector sizes

The controller provides support for variable sector sizes. The host may choose to program any sector size between the BCH algorithm nominal sector size and max sector size allowed by BCH module - in this release it means possibility of adding up to 32 additional bytes per data sector (even number of bytes only).

The BCH engine adds ECC check sums to each sector. These check sums are always aligned to 16-bit. This is required for data integrity when different work modes (SDR/DDR) are used for data access.

Note

Alignment must be taken into account when calculating the actual amount of data (user data + ECC check sums + padding data) written to the device. Alignment should meet the following rules:

In 8bit SDR work mode the amount of data is aligned to 8-bits.

In 16bit SDR work mode the amount of data is aligned to 16-bits.

The following are the programmable registers available for host to support variable sector sizes:

- The *sector_size* field in the *transfer_cfg_1 (0x0404)* register - If there are "n" sectors to be used in a page, this register specifies in bytes the size of the (n-1) sectors.
- The *last_sector_size* field in the *transfer_cfg_1 (0x0404)* register - If there are "n" sectors to be used in a page, this register specifies in bytes the size of the n-th sector.
- The *sector_cnt* field in the *transfer_cfg_0 (0x0400)* register - This register programs the number of sectors which will be transferred within a single page.

3.3. Bad Block Marker

Manufacturers mark device bad blocks at certain byte location from the starting of the spare area, which may get over-written by user data. It might be necessary for the intended system to preserve the bad block information so that good blocks are not seen as bad blocks later on in the life of the device.

The *skip_bytes* of the *skip_bytes_conf (0x100c)* register configure number of bytes to skip from offset defined by the *skip_bytes_offset (0x1010)* register. If *skip_bytes* or *skip_bytes_offset* value is 0 (default) then skipping is disabled. Skip bytes is disabled also in Generic Work Mode and in Set Feature command in PIO work mode. The controller starts to skip bytes starting from offset counted from the beginning of the transferred data package. The *skip_bytes_offset* should be within the range of data size defined by sector_cnt, sector_size and last_sector_size (from 1 to sector_size*(sector_cnt-1)+last_sector_size-1) The Mini-controller does not take into account the column address when calculating offset. The *skip_bytes* should always be an even number.

Controller skips the number of bytes and replaces it with the *marker* field of the *skip_bytes_conf (0x100c)* register in the skipped bytes. The Host should program *marker* register with a value that corresponds to a good block marker for the device. This way the good blocks are retained as good in the bad block marker.

3.4. Scrambling Support

The controller has support to scramble/de-scramble data being written into and being read from the Flash device. Scrambling the data helps in removing repetitive patterns being written into the flash memory, reducing the chances of read or program disturbs that cause data corruption in the page. To achieve this scrambler module generates pseudorandom bit sequence, which is multiplied with data (through bitwise logical XOR function).

To achieve 64-bit word scrambling in each clock cycle, two 32-bit scramblers are used in parallel inside Scrambler module. They are initiated with different seed at the beginning of data block. Seeds values are:

Polynomial used in every 32-bit scrambler is: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

For every 64-bit data word, scrambler module generates word of equal length. For WRITE transactions bitwise XOR with scrambler module output is applied to data after BCH encoder generates parity bits. For READ transaction bitwise XOR is applied after data are read from Flash and before data are passed to BCH decoder.

Scrambler module is reinitialized at the beginning of data sector, so pseudorandom sequence will be repeated for every sector. Since bitwise XOR is linear function, scrambling does not affect BCH correction capability.

The Scrambler module can be used when the ECC is enabled or disabled. The scrambling works on the ECC sector boundaries. Scrambling must be enabled during write and read of the same pages, otherwise data will be corrupted.

The Scrambler module shall not be used when Erased Page Detection module is enabled.

For Command DMA or PIO work mode the *scrambler_en* bit in the *ecc_config_0 (0x0428)* register enables this feature. For GENERIC work mode scrambler can be enabled directly through the Data Sequence command.

3.5. Erased Page Detection Support

The NAND flash controller can detect erased data pages. This allows for certain data chunks with specific pattern (all bytes of 0xFF value) to be marked as erased even in presence of errors. Such pattern shall not occur in correct data pages with proper ECC check-bits which makes detection possible.

Erased page detection must be enabled during write and read of the same pages, otherwise data will be corrupted. Pattern of 0xFF values must not occur in user data or it will be recognized as erased page. Check-bits will be different for the same page with Erased page detection enabled/disabled.

Erased page detection occurs during data read and is independent from BCH decoder. This feature could be used with BCH enabled/disabled mode. When erased page is detected data correction will not be performed and page will be marked as erased.

If the BCH is enabled then controller uses selected correction ability level as allowed limit of zeros inside ECC sector. If the BCH is disabled then controller uses the *erase_det_lvl* in the *ecc_config_1 (0x042c)* register to configure allowed limit of zeros inside selected sector.

The Erased Page Detection module shall not be used when Scrambler module is enabled.

The *erase_det_en* bit in the *ecc_config_0 (0x0428)* register enables this feature.

The *Erased Page* bit informs if erased page was detected. For the Command DMA work mode bit will be located in the status word of descriptor, for the PIO work mode it will be located in the *cmd_status (0x0014)* register.

4. Booting Mechanism

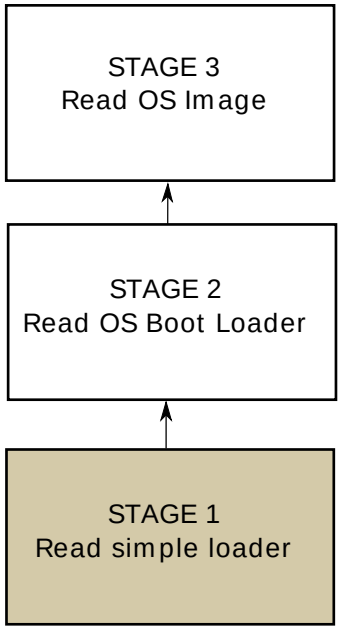
This section describes booting from the NAND Flash memory mechanism. This chapter is divided into the following sections:

Section 4.1, "*Boot Process Overview*"	This section describes typical boot process and shows which part is implemented in the NAND Flash controller.
Section 4.2, "*Required Boot Image Structure*"	This section describes required boot image structure and the NAND Flash memory block layout.
Section 4.3, "*Boot Sequence Algorithm*"	This chapter describes steps of performed booting sequence.
Section 4.4, "*Modified registers list*"	This chapter describes the list of registers, configured by booting sequence.
Section 4.5, "*Boot configuration pins description*"	This chapter provides information about how to set up the boot configuration pins.
Section 4.6, "*Boot status register*"	This chapter describes the boot status register layout.

4.1. Boot Process Overview

Figure 4.1 shows typical booting process flow. The NAND Flash controller implements only the first stage of the booting processes.

Figure 4.1. Typical Boot Process



Tasks executed at each stage are as follows:

- Stage 1 - The simple Loader application is stored in the NAND Flash memory. The booting logic inside the controller copies it to the host SRAM for execution.
- Stage 2 - The simple Loader from previous stage is used to copy operating system loader from the NAND Flash to the host DRAM.

- Stage 3 - The operating system loader is used to copy OS image from the NAND Flash memory to a host DRAM memory, where it is executed. When this process is finished, the operating system takes over control of the host.

4.2. Required Boot Image Structure

The boot mechanism expects that boot image will fit into the single used NAND Flash device data block. It is required to store three copies of boot image in three following NAND Flash device data blocks starting from block 0. Using block zero is essential. The block 0 of the NAND Flash device is guaranteed to be suitable for an extended number of block erase program cycles. The specific number of erase/program cycles depends on the device type and vendor.

Figure 4.2 shows the NAND device block layout used to store the boot image.

The first eight bytes contain pointer to the destination location in the host memory where boot image data need to be copied.

The following two bytes are used to configure image size. The image size needs to be expressed as number of sectors equal to the controller's ECC engine sector size. The *boot_sec_size* controller port is used to inform boot engine what sector size value was used to generate the boot image written to the NAND Flash memory.

Figure 4.2. Block layout

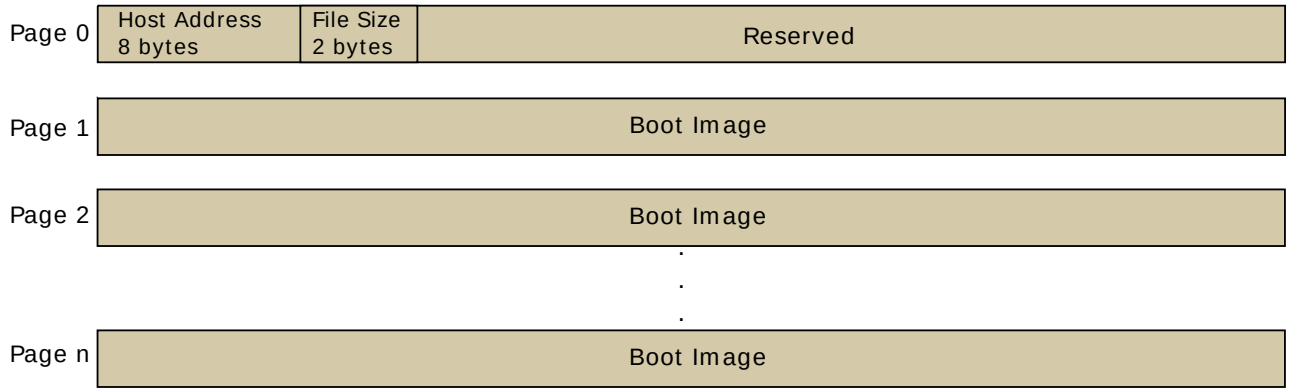


Figure 4.3 shows the boot image data structure for the single page layout. In the situation when the ECC engine is off, the Boot engine does not use the spare area. When the ECC engine is on, the total size of ECC parity bytes, per single page, can not be greater than the page spare area size. The boot image is split across block pages using the following algorithm:

- ECC disabled: sectors per page = page main area size / boot_sec_size,
- ECC enabled: sectors per page = page main area size / boot_sec_size, ECC parity bits per page <= page spare area size.

It is strongly recommended to use ECC engine to protect the boot image against data corruption. If boot image is protected by the ECC, the Figure 4.4 shows the NAND device page layout. In this case application needs to provide ECC data for the programmed boot image data and configuration data. The ECC sector size depends on the ECC option selected for the NAND Flash controller. The ECC initial parameters are provided as values on controller ports described in section *Boot configuration pins description*. Values on these ports need to be strapped to constant values. It is possible to use the controller itself for programming boot blocks contents.

Figure 4.3. Page layout

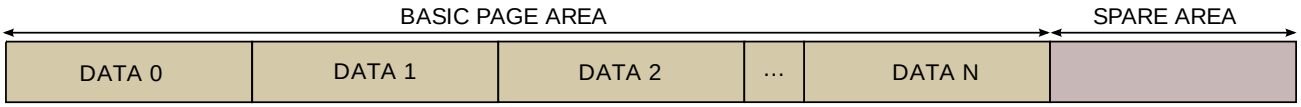
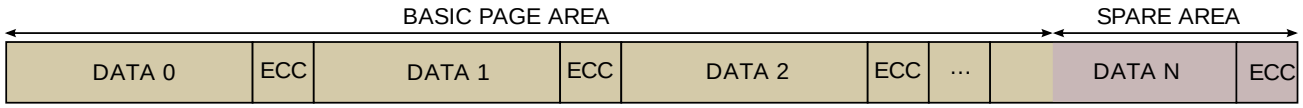


Figure 4.4. Page Layout protected by ECC



4.3. Boot Sequence Algorithm

The boot sequence is enabled when *boot_en* pin is driven high and the Device Discovery is enabled after controller's hardware reset sequence is finished. Disabling device discovery will disable the boot process as well and boot_comp will not be set. The NAND Flash controller drives high the *boot_comp* pin after the boot sequence is finished. Components other than the NAND Flash controller need to be held in a reset state as long as the NAND Flash controller performs boot sequence - it is not allowed to send any transaction to the NAND Flash controller when it performs the booting sequence.

For the boot sequence purposes the device connected to the bank 0 is used.

Before boot sequence is triggered, the controller will perform the device discovery process. For proper operation, the connected device needs to be compatible with ONFI specification to be correctly detected by the device discovery module. After completion information obtained during this process will be used to configure the controller properly before reading boot code stored in the NAND Flash device. The booting process will be disabled if device initialization process fails.

At the first step of boot sequence, the File Size and Host Destination Address are obtained from the NAND Flash device. The File Size needs to be aligned to the multiple of the ECC sector size. This means that the file size is specified in the number of sectors, the byte value of which is defined by *boot_sec_size* pins. When the boot engine uses the ECC mechanism, by setting *boot_ecc_enable* pin high, the value defined on the *boot_sec_size* pins needs to be equal to nominal ECC sector size defined in the ECC specification (Chapter 3, *ECC Engine*). It is not recommended to use booting mechanism without ECC enabled, but when this is done, value of *boot_sec_size* must still be set to available nominal ECC sector size. Ignoring suggested values for the boot_sec_size may cause undefined behavior. The Host Destination Address needs to be aligned to 8B words.

The controller cannot determine if CE pin reduction was enabled when the hardware reset occurs. As the RESET command sent to the NandFlash device doesn't clear Volume Appointment it is required to store a copy of boot image in every NandFlash device in volume chain when using the Boot feature with CE pin reduction. The controller will use last selected device (or device 0 in case when CE pin reduction was disabled) to read boot image after hardware reset.

4.4. Modified registers list

The following Special Function Registers can be modified by the boot module:

- *ecc_config_0 (0x0428)*
- *ecc_config_1 (0x042c)*
- *transfer_cfg_0 (0x0400)*
- *transfer_cfg_1 (0x0404)*
- *boot_status (0x0148)*

During booting sequence the Special Function Registers can not be changed by the system side. After the booting sequence, default values of Special Function Registers are restored, as modified by the Boot Engine and with the exception of Boot Status Register.

4.5. Boot configuration pins description

The boot engine requires the following configuration pins to be set:

- *boot_ecc_corr_str* - This configuration pins selects correction abilities available for the ECC engine used by boot. Available corr_str values are described in Table 3.1
- *boot_ecc_enable* - When this configuration pin is set high, the boot engine uses ECC engine as on.
- *boot_sec_size* - The *boot_sec_size* pins are defined as size of the single sector in bytes, which can be transferred in the boot sequence. When the boot engine works with the ECC on, the value defined on the *boot_sec_size* pins needs to be equal to ecc sector size value. More information about ecc sector size can be found in the chapter 3.

4.6. Boot status register

It is possible to check the latest boot sequence status by reading the *boot_status (0x0148)* register after OS gets control over system. The Table 4.1 shows the boot status register layout.

Table 4.1. Boot status register

Bits	Name	Description
31:8	-	Reserved
7	TIM_OUT_ERR	This field informs the user that during the booting sequence the time out condition was detected.
6	-	Reserved
5:4	CPY_ID	This field identifies a boot block used in the latest boot sequence run. Allowed values are: 00 - block 0 was used, 01 - block 1 was used and 10 - block 2 was used.
2	BUS_ERR	This field describes the data BUS status during boot process. If it is set high, the boot process fails due to data interface receiving an error response from the target.
1:0	BCH_ERR_TYPE	This field identifies the kind of ECC mechanism error detected during boot sequence process. Allowed values are 00 - no errors detected, 01 - correctable error detected, 10 - uncorrectable error detected.

5. Programming Specification

This chapter is a guideline for software developers that shows how to correctly operate controller to achieve best performance in the easiest possible way. This chapter is divided into the following parts:

Programming Control Registers	This section describes how to access control registers.
Programming Command Registers	This section describes how to access command registers.
Status Polling Configuration	This section describes how to configure controllers status polling mechanism.
Device Layout Configuration	This section describes how to configure device layout.
Configure Multi-plane and Cache Operations	This section describes how to enable support for multi-plane and cache operations.
ECC configuration	This section describes how to configure the ECC engine.
Interrupts Configuration	This section describes how to configure the interrupts.
Timing registers	This section describes how to configure the timing registers.
Slave DMA Programming	This section describes how to configure slave DMA.

5.1. Programming Controller Registers

The control registers groups can be programmed only when controller is in the IDLE state otherwise access to these registers will be ignored. Controller status *ctrl_busy* field of the *ctrl_status (0x0118)* register. Following registers are affected:

- Configuration Registers
- Mini controller Registers
- Protect Mechanism registers

Note

Write to the Protect Mechanism registers is additionally secured by external pins: wre_prot_en_0 and wre_prot_en_1. If any of this pins is tied to 1 write access to corresponding protect registers will be blocked.

The following registers groups can be programmed independently from the controller state:

- Command Registers
- Status Registers
- Debug Interface registers

Access to the Mini controller can be delayed because those registers are placed in the outside of the host clock domain.

The Controller and Device Parameters registers group can only be read.

Additionally during initialization and boot process write access to all registers will be ignored.

5.2. Programming Command Registers

All operations in the NAND Flash controller are executed by the Command Engine module. The Command Engine can execute more than one operation in parallel, so it is required to select execution thread that will be used to execute command. This information is passed to the Command Engine using the *TRD_NUM* filed of the *cmd_reg0 (0x0000)* register.

The Command Engine can only accept a new command when thread selected in command is in idle state. The host software should check thread status before it sends any new command. The threads status is available in the *trd_status (0x0120)* register. Commands sent to busy thread are ignored and the *cmd_ignored* flag is set in the *intr_status (0x0110)* register.

To trigger any operation host software needs to program controller's command registers (*cmd_reg0 (0x0000)* - *cmd_reg3 (0x000c)* registers). Allowed command layout is described in the Chapter 2, *Command Engine*. The program sequence provided below should be followed:

1. Check if thread selected in command has idle state. The *trd_status (0x0120)* register needs to be read for this purpose. If bit of the *trd_busy* field corresponding to the thread number is clear then this thread is in the idle state.

2. Write command values to the *cmd_reg0 (0x0000)* - *cmd_reg3 (0x000c)* registers. Unused command registers for given operation do not need to be written. The *cmd_reg0 (0x0000)* register should be written last, write access to this register will trigger programmed operation execution.

5.3. Status Polling Configuration

You need to program the Controller to use the ready/busy pins of the NAND Flash device operation mode or polling by command interface operation mode. Program the following registers to ensure that the polling done on the device interface is done at the correct times and with the correct frequencies:

1. Set the *rb_enable* bit of the *rdst_ctrl_0 (0x0410)* register to the desired mode of operation. The 0x1 value selects R/B pin polling. The 0x0 value selects status polling by command interface. For Multi-LUN operations this bit is ignored and LUN status is checked by Read Status Enhanced command.

2. Set the *long_polling (0x0408)* register to the appropriate value by software depending on the mode of operation of the controller, and the desired wait value.

3. Set the *short_polling (0x040c)* register to the appropriate value by software depending on the mode of operation of the controller, and the desired wait value.

For the status polling mechanism that uses command interface it is possible to configure how the controller will interpret status data read from the NAND Flash memory. This feature is configured by the *rdst_ctrl_0 (0x0410)* register. The following fields need to be configured to enable this feature:

1. The *ready_mask* filed value will be used to mask status data read from the NAND Flash device. The masking uses logical AND operation to select part status data field that will be compared with the expected value.

2. The *ready_value* holds expected status data value. It will be compared with a result of the AND operation between the *ready_mask* field and status data value read from the NAND Flash device.

Some of the legacy devices have the tRW timing which is defined as minimal delay after the RnB line was asserted by device and before the WE line is asserted by controller. This timing can be violated by controller if command interface is selected as the way of checking device status. This will not affect status checking mechanism, because read status command is periodically repeated as long as device ready condition is achieved, so even if first command will get incorrect information then second command will get the correct one.The tRW timing is defined for only small set of legacy devices.

Host should avoid setting either short or long polling to a very low value. Otherwise polling time may increase which will impact overall performance. Value of 20 or greater is strongly advised.

The status polling mechanism configuration can only be changed if controller is in the idle state.

5.4. Device Layout Configuration

The Controller identifies the device features correctly and sets up the following registers for correct operation of the Controller to work with the device. However, for raw NAND devices where there is no universally accepted protocol of defining the identification sequence, it is possible that the device is not identified correctly and the values in the registers may be incorrect. It is the responsibility of the software or test case to ensure that the correct values are programmed in the appropriate registers:

1. The *nf_dev_layout (0x0424)* - to setup correct value of pages per block and LUN-s numbers (or the test case can ignore this step if the device used in simulation is known to be identified correctly by the Controller).

2. Set or clear the *device_16bit* bit in the *common_settings (0x1008)* register to select correct value of device width.

5.5. Configure Multi-plane and Cache Operations

This section provides a list of registers that need to be programmed while choosing to use multi-plane or cache operations on the device. The device should be supporting these types of operations in order to use these registers. If the devices do not support multi-plane operations or cache operations, then these registers can be left at their power-on reset values, and this will not have any impact on the functionality of the Controller. Even if the device supports these sequences, the software may choose not to use these sequences and can leave these registers at their power-on reset values. Placing correct values in the following registers will result in achieving the best performance from a given device.

The steps described below need to be followed to configure multi-plane operations support:

1. If device has support for multi-plane write command sequences, the software should enable these operations by setting the *mpl_wr_en* bit and program the *mpl_prg_seq* field with the appropriate value based on the device type.

   ### Note

   If the Controller is set for multi-plane operations, the number of pages to be accessed is always a multiple of the number of planes in the device.

2. If the device has support for multi-plane read command sequences, then software should enable these operations by setting the *mpl_rd_en* bit and program the *mpl_rd_seq* field with the appropriate value based on the device type.

All of those fields are grouped inside the *multiplane_config (0x0434)* register.

The steps described below need to be followed to configure cache operations support:

1. If the device has support for cache read command sequences, the software should enable those operations by setting the *cache_rd_en* bit.

   If the device has support for cache write command sequences, then software should enable these operations by setting the *cache_wr_en* bit.

Those fields are grouped inside the *cache_config (0x0438)* register.

If both cache and multi-plane operations are enabled the multi-plane cache operations will be enabled.

5.6. ECC enabling

Before you start any data operation on the Flash device, you need to decide whether you want to have the ECC enabled or disabled. If the ECC needs to be enabled, then set the appropriate correction level depending on the page size and the spare area available on the device. The steps described below need to be followed to configure ECC engine correctly:

1. Set the *ecc_enable* bit in the *ecc_config_0 (0x0428)* register to enable ECC engine. If enabled, the following fields need to be programmed accordingly.

2. The *corr_str* field in the *ecc_config_0 (0x0428)* register needs to be programmed with required correction ability value. Value of this field should be coded as stated in Table 3.1. If programmed value is not supported by the ECC engine, the lowest supported correction ability is selected.

3. The *sector_size* field in the *transfer_cfg_1 (0x0404)* register needs to be programmed with the ECC sector size value. Value programmed in this field will be used to parameterize size for all ECC sectors except the last one.

4. The *last_sector_size* field in the *transfer_cfg_1 (0x0404)* register need to be programmed with the ECC sector size value. Value programmed in this field will be used to parameterize size for the last ECC sector.

5. The *scrambler_en* bit in the *ecc_config_0 (0x0428)* register needs to be set if the embedded scrambler must be enabled.

6. Program the *skip_bytes* and *marker* fields in the *skip_bytes_conf (0x100c)* register if the software intends to preserve the spare area marker.

7. Set the *erase_det_en* bit in the *ecc_config_0 (0x0428)* register and program the *erase_det_lvl* in the *ecc_config_1 (0x042c)* register if erased pages detection mechanism will be used.

5.7. Interrupts Configuration

The interrupts mechanism in the NAND Flash controller uses global interrupts enable bit to mask or unmask all available interrupts. Additionally each interrupt has its own enable flag. The status fields are set even when corresponding interrupts are masked. All status bits are cleared by writing logical 0x1 to status bit that need to be cleared.

Following steps need to be followed to enable selected interrupts:

1. To allow interrupts to reach host the *intr_en* bit in the *intr_enable (0x0114)* register need to be set.

2. Individual interrupts are enabled by the single bits of the *intr_enable (0x0114)* register. Individual interrupts statuses are stored on corresponding bits of the *intr_status (0x0110)* register.

3. The *trd_comp* interrupts are special kind of interrupts. These interrupts are enabled by the *INT* bit of the command descriptor in Command DMA work mode or the *INT* bit of the *cmd_reg0 (0x0000)* for the PIO and Generic work modes. These interrupts are reported on the *trd_comp* field of the *trd_comp_intr_status (0x0138)* register. Extended status information for this interrupt is provided in descriptor status field in the Command DMA work mode or in the *cmd_status* register for the PIO and Generic work mode. Status for a given thread need to be selected by the *cmd_status_ptr* register.

External interrupt line will remain high until all interrupt flags will be cleared by the software. If system expects rising edge on *interrupt* pin for every interrupt event software must ensure that appropriate interrupt flags will be cleared before new interrupt event comes. Otherwise some interrupts may be missed. Especially if there is risk that 2 descriptors in the same chain could complete very close together (such as when using NOP command types), then it is advised to just set INT field for a single descriptor (the last one) and not all in the chain.

The interrupt status bits aren't automatically cleared after given interrupt is enabled. If we don't want previous event to trigger interrupt then it is advised to clear interrupt status bit corresponding to the interrupt we want to enable, before we will enable it.

5.8. Timing registers

The following registers need to be optimized depending on the speed of operation:

1. The *async_toggle_timings (0x101c)* - timings characteristic for SDR modes.

2. The *timings0 (0x1024)* - sequence timings common for all work modes.

3. The *timings1 (0x1028)* - sequence timings common for all work modes.

4. The *timings2 (0x102c)* - sequence timings common for all work modes.

The time delay generated by the controller equals the minimum value written into the register, increased by 1. All the timings are generated using the nf_clk clock signal. Once the timing registers are set, the host may change the clock to the Controller. For this, the host needs to ensure that all operations in the Flash Controller have been completed and the Controller is in idle state. This is identified by checking the level of *ctrl_busy* pin or by the *ctrl_busy* bit in the *ctrl_status (0x0118)* register. If this is asserted, it is an indication for the host that the Flash Controller is busy waiting for an operation inside the Controller to complete. If this is de-asserted, it is an indication that the Controller is idle, and clocks may be changed to the Controller.

Now the Controller is ready to accept data commands to the device.

5.9. Slave DMA Programming

To access controllers data path using the Slave DMA interface. Following program sequence needs to be followed:

1. Before host master access the Slave DMA interface it needs to check if data transfer is allowed. This is done by software polling the *sdma_trigg* bit in the *intr_status (0x0110)* register. Instead of software polling, host can unmask interrupt associated with *sdma_trigg* by setting the *sdma_trigg_en* bit in the *intr_enable (0x0114)* register.

   Please note that if *sdma_paused* flag in the *ctrl_status (0x0118)* register is set (i.e. in case of detecting unexpected transaction during earlier data transfer) controller will not request new SDMA transaction. In this case software must first clear the *sdma_paused* flag.

2. After host discovered that data transfer is allowed (using interrupts or software polling) it should read the *sdma_size* and the *sdma_trd_num* registers. The *sdma_size (0x0440)* register provides byte aligned data block size that need to be transfered. The *sdma_trd_num (0x0444)* register identifies command associated with this data transfer and allows to select valid address and direction for data transfer.

3. Before starting data transmission, host must clear the *sdma_trigg* flag by writing 1. After clearing this flag host can execute data transmission on slave DMA interface.

4. If host ignores requirements described above and starts to transfer data when Slave DMA is not ready then the *sdma_err* flag is set in the *intr_status (0x0110)* register. If the *sdma_err_en* bit is set in the *intr_enable (0x0114)* register then interrupt will be triggered. When the *sdma_err_rsp* bit is set in the *dma_settings (0x043c)* the ERROR response is returned, and if it is cleared, the OK response is returned.

5. If host send unsupported transaction to slave interface the Slave DMA ignores this access and both the *sdma_err* flag in the *intr_status (0x0110)* register and the *sdma_paused* flag in the *ctrl_status (0x0118)* register are set.

   Additionally if system bus error is detected after the Slave DMA transfer was triggered, the *Fail* and *bus error* bits will be set in the last operation status.

   Performing transaction in wrong direction (i.e. write transaction to Slave DMA port in case of sending Read Operation command to NF device) is forbidden.

6. Hardware Implementation Requirements

This chapter describes the hardware implementation requirements for application that will use the NAND Flash Controller IP.

- System interface
- Clock
- Reset

6.1. System Interface

6.1.1. System Interface Introduction

Following system interfaces are present on the controller ports:

- The Slave Registers Interface - It is used to access controller's configuration, status, and command registers.
- The Slave Data Interface - It is used to access controller's data path by the host master - DMA or CPU. It is designed for high speed access.
- The Master Data Interface - It is used to transfer data from the controller data path to the hosts memory. It is controlled by embedded Data DMA engine.

Note

Host must ensure that issued transactions are correct with respect to corresponding bus specification. Sending incorrect transactions (i.e. omitting request phase, not finishing data phase, starting transaction during reset) may lead to incorrect behavior of the controller and hangs.

6.1.2. AXI Slave Register Port

This interface is used to access controller's control, status and command registers. This port operates asynchronously and supports only single beat transactions.

All configuration registers can be written only when the controller is in idle state. Otherwise the register operation will be ignored. For more information about rules describing register access please see: Programming Control Registers.

6.1.3. AXI Slave Data Port

The AXI data port functions as an AXI slave to external AXI masters such as CPU, DMA, or other peripherals. The AXI data port interface handles all communication between the AXI bus and the controller data path.

The AXI slave data port accept all incoming transactions independently of the configured number of the outstanding transactions.

6.1.4. AXI Master Data Port

The AXI DMA port functions as an AXI master to external AXI slaves like system memory. The AXI DMA port interface handles all communication between the controller data path and AXI Bus or is used by the Command Engine to fetch command descriptor when it works in the Command DMA mode.

6.1.5. System interface Transactions

The system interface supports a subset of native bus transactions. Burst type of INCR and Outstanding transactions are supported for both DMA Slave and DMA Master interfaces.

Register interface works as low speed interface and supports only single beat transfers and no outstanding transactions.

Table 6.1. Ports Supported Features.

System interface	Standard	ALEN	BURST	WORD SIZE	OUTSTANDING TRANSACTIONS
Slave Registers	AXI4 Lite	Not included	Not included	SIZE = 32b	Not included
Slave Data DMA	AXI4	0 to 255	INCR	SIZE = 64b	16
Master Data DMA	AXI4	0 to 255	INCR	SIZE = 64b	16

Note

Controller allow only the fixed size word transfers on the master and slave data interfaces. Because there is no byte strobing ports in those interfaces, then addresses for all transfers need to be word aligned. Unused address bits need to be connected to a logical zero value.

Command engine access the host memory using single transfer transfers only.

6.1.6. Early Transaction Terminations

The NAND Flash Memory Controller bus protocol does not allow an early burst termination. As a result, controller core requires completion of the whole READ/WRITE transaction of the length specified to/by the controller. The length is a defined quantity that is specified at the beginning of the transaction.

Master Data port will not expect to have it's burst terminated by targets. Burst termination encountered by controller's ports will lead to unpredictable behavior leading to lock-up conditions.

6.1.7. Errors

6.1.7.1. Errors on the Slave Register Port

The Slave Register Port responds to illegal conditions with the ERROR response when the transaction address is not aligned to the size of the transaction.

6.1.7.2. Errors on the Slave Data Port

Slave Data port will return the error response if host master tries to access Slave Data Interface when no transfer transaction is triggered. The correct use procedure for Slave Data Interface is described in section Slave DMA Programming.

The error response generating process can be disabled by clearing the *sdma_err_rsp* bit of the *dma_settings (0x043c)* register. If controller detects unexpected transfer on the Slave Data interface, it raises the *sdma_err* interrupt. This interrupt flag is stored in the *intr_status (0x0110)* register.

6.1.7.3. Errors on the Master Data Port

Any error on the Master Data Port is reported via interrupts. The *ddma_terr* or *cdma_terr* bits of the *intr_status (0x0110)* register indicate DMA master interface received an error response from the target. The transaction address which resulted

in target error will be put in *dma_target_error_l (0x0140)* and *dma_target_error_h (0x0144)* register. This address will not be overwritten by the controller until both *ddma_terr* and *cdma_terr* will be cleared by software.

6.1.8. Clocks and Resets

Controller uses the same clock (*mACLK*) and reset (*mARESETn*) signals at the slave data and master data interfaces. Clock and reset signals at the register interface (*regACLK* and *regARESETn*) can be different.

6.2. Clocking Mechanism

This section describes clock sources required by the HPNFC controller and used clock domains synchronization mechanisms.

6.2.1. Required Clock Sources

Primary clock inputs to the Controller:

- The *nf_clk* - This clock drives the Mini Controller and PHY modules.

  The clock period should be consistent with bigger value of the tRC and tWC timing parameters. The controller has the ability of programming the pulse width and hold times of the RE_N and WE_N signals in the SDR mode which allows to use clock source with higher frequency of operation and program the desired values of delays to meet timing requirements of the connected NAND Flash device. The maximum frequency in this case depends on the timing requirement of the device divided by the maximum delay value possible to program.

- The *bch_clk* - This clock drives controller's BCH engine. The bch_clk frequency depends on the BCH configuration and should be trimmed basing on the BCH engine paralleling factors and the *nf_clk* clock frequency value.

- The *sys_clk* - clock signal connected to the *mACLK* This clock drives most of the controller logic except ECC, Mini Controller, and SFR Interface modules. Its frequency should match the host interface clock frequency. The *sys_clk* clock domain is driven by this signal.

- The *sfr_clk* - clock connected to the *regACLK* - This clock drives controller Slave Register Port. It was added to allow to connect register interface to the system low speed bus. The *sfr_clk* clock domain is driven by this signal.

All clock changes to the Controller should be done only when it is in an IDLE state. IDLE state in the Controller can be determined if the *ctrl_busy* pin on the Controller port has gone low or the *ctrl_busy* bit in the *ctrl_status (0x0118)* register has gone low.

Estimated clock frequencies are listed in the Estimated Clock Frequencies table. All clock frequencies are defined for the TSMC 28HPM technology.

Table 6.2. Estimated Clock Frequencies

Clock Name	Min Freq	Max Freq	Description
SYS_CLK	20 MHz	200 MHz	Frequency defined by the client application. The SYS_CLK can be completely asynchronous to the the NF_CLK, but to achieve full performance SYS_CLK >= NF_CLK/4 (for DATA WIDTH = 64 bit)
SFR_CLK	1 MHz	200 MHz	The SLAVE_CLK needs to be lower or equal to the SYS_CLK

Clock Name	Min Freq	Max Freq	Description
NF_CLK	20 MHz	266 MHz	Frequency needs be adjusted to selected work mode on the NAND flash interface.
BCH_CLK	20 MHz	200 MHz	The BCH_CLK can be completely asynchronous to the the NF_CLK, but to achieve full performance BCH_CLK should be the fastest possible. Number of the processed data depends on the paralleling factors.

6.2.2. Clock Domains Synchronization

Clock domain synchronization logic is placed in separated modules and those are instantiated on the HPNFC top level. The NAND Flash controller uses two clock domains synchronization mechanisms:

- The Asynchronous FIFO - This method is used as clock domains crossing mechanism between the *SYS_CLK* and *BCH_CLK* clock domains - to transfer data, between the *BCH_CLK* and the *NF_CLK* clock domains - to transfer data and between the *SYS_CLK* and the *NF_CLK* clock domains - to transfer commands and statuses.
- The Multi Cycle Path formulation technique. In this method transmitting side in first clock domain is sending unsynchronized data to a receiving side in second clock domain paired with a synchronized control signals. The data and control signals are sent simultaneously allowing data to setup while the control signal is synchronized for two or more receiving side clock cycles before it can strobe data write on receiving side. The receiving side sends back the acknowledge signal back to the transmitting side. This method is used as clock domains crossing mechanism between the *SFR_CLK* and the *SYS_CLK* - to access controller's registers from the host side.

6.3. Reset

It is recommended to use three reset signals in application that utilizes this IP:

- System Reset (*sys_rst_n*) - it is used to initialize controller internal logic except the configuration registers and related modules.
- Registers Reset (*reg_rst_n*) - it is used to initialize controller's configuration registers together with the device initialization and boot module.
- Protect Reset (*prot_rst_n*) - it is used to initialize controller's write/erase protect mechanism registers.

Both reset signals need to be active low. The rising edge of reset signal need to be synchronized to the destination clock domain. Synchronizers used for this purpose need to be placed on the application chip level.

There are five asynchronous reset signals on the IP top level Controller.

The System Reset is connected through rising edge synchronization circuits to the following top level inputs:

- The *mARESETn* input - reset for system clock domain. Its rising edge needs to be synchronous to the *mACLK* clock.
- The *nf_rst_n* - reset for nf_clk clock domain. Its rising edge needs to be synchronous to the *nf_clk* clock.
- The *bch_rst_n* - reset for bch_clk clock domain. Its rising edge needs to be synchronous to the *bch_clk* clock.

The Registers Reset is connected through rising edge synchronization circuits to the following top level inputs:

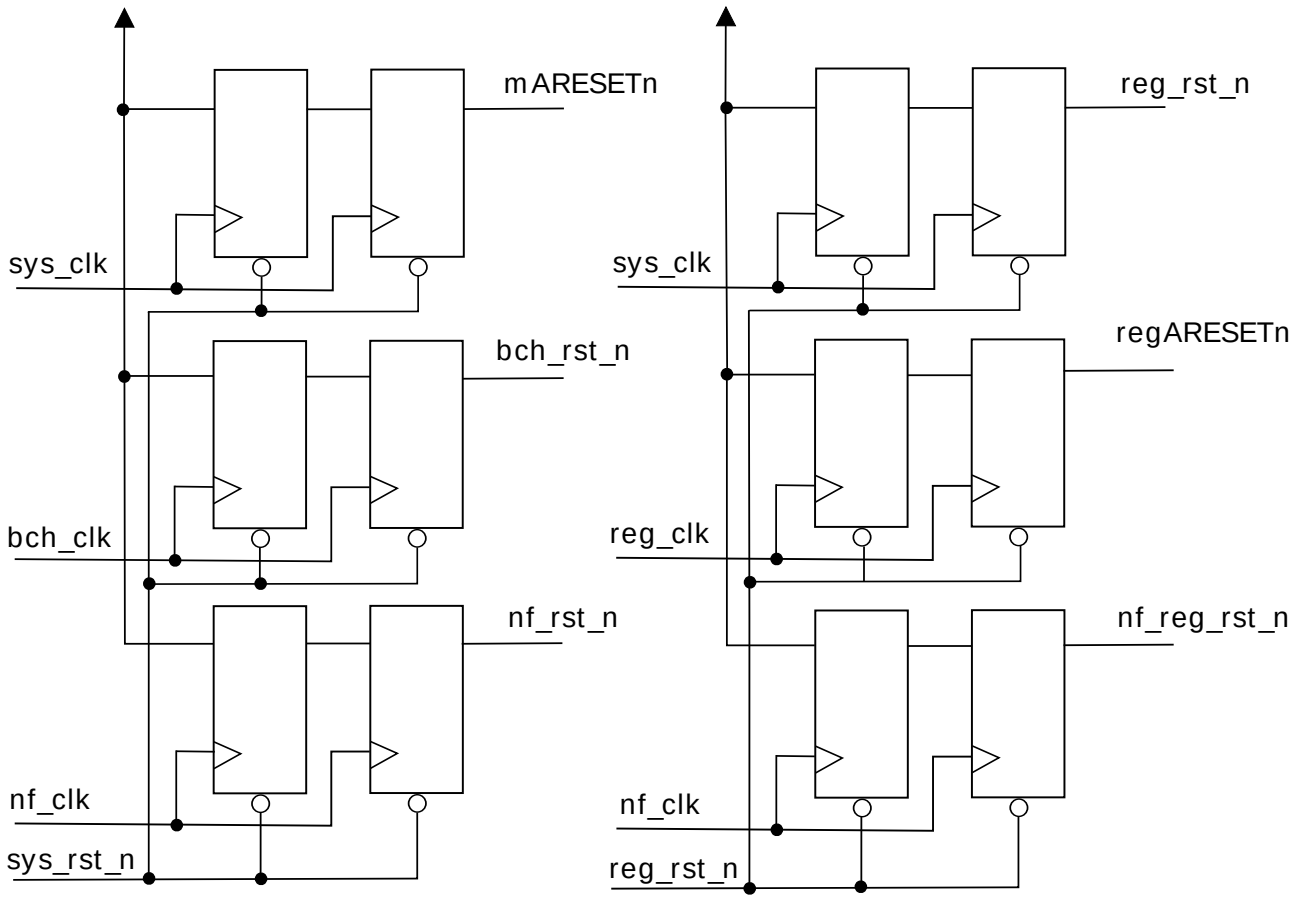
- The *regARESETn* input - reset signal for registers system interface. Its rising edge need to be synchronous to the *regACLK* clock.
- The *reg_rst_n* - reset registers in the storage module. Its rising edge needs to be synchronous to the *mACLK* clock.
- The *nf_reg_rst_n* - reset registers in the storage module. Its rising edge needs to be synchronous to the *nf_clk* clock.

If System Reset is triggered without following Register Reset then interrupt status registers could contain obsolete values and need to be cleared. The following registers need to be cleared:

- The *intr_status (0x0110)* register.
- The *trd_error_intr_status (0x0128)* register.
- The *trd_comp_intr_status (0x0138)* register.
- The *trd_timeout_intr_status (0x014c)* register.

To ensure that reset signal for each clock domain will be de-asserted synchronous to its clock signal, the reset synchronizers should be used. The Figure 6.1 shows a resets synchronization mechanism used in the HPNFC. Used synchronization mechanism causes asynchronous reset enable and synchronous reset disable.

Figure 6.1. Resets Synchronization mechanism



6.4. Memory implementation requirements

The HPNFC uses synchronous static RAM memory implementation as storage element.

Figure 6.2 shows the write operation. The *Addr* and *Wdata* buses are probed only when *Wen* is high. Data should be stored on the first rising clock edge after condition described in previous sentence is true.

Figure 6.2. SRAM Write Transaction

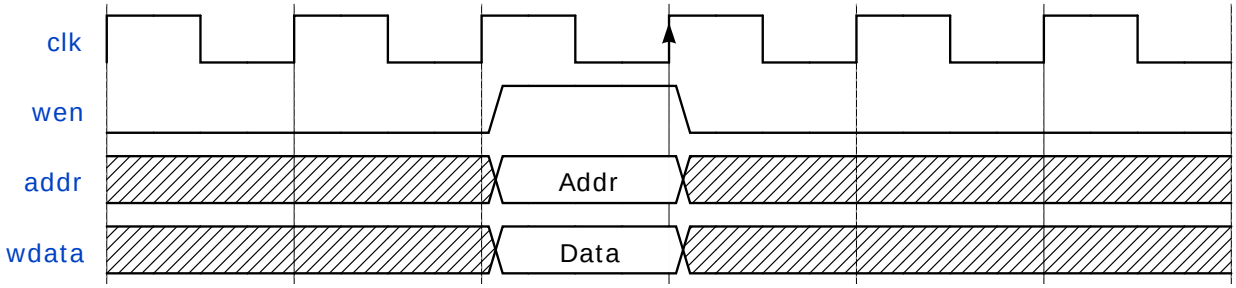
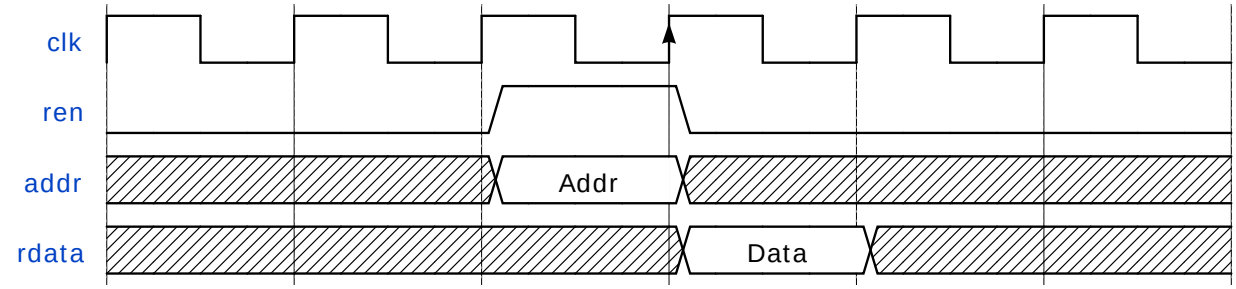


Figure 6.3 shows the read operation. The *Addr* bus is probed only when *Ren* is high. The read data should be returned after first rising clock edge after condition described in previous sentence is true. It needs to be valid for one clock cycle.

Figure 6.3. SRAM Read Transaction.



The Table 6.3 shows the size requirements for specific SPRAM (single-port RAM) memories.

Table 6.3. Size of SPRAM memories

Name	Related interface	Word Width	Words Count	Total size
Context storage RAM	mem_desc_*	64 bit	72	4608 bit
ECC sector buffer RAM 0	mem_ecc_*_0	64 bit	68	4352 bit
ECC sector buffer RAM 1	mem_ecc_*_1	64 bit	68	4352 bit

7. Special Function Registers

7.1. Command and Status registers. (0x0000)

7.1.1. cmd_reg0 (0x0000)

Description: Command register 0. Writing data to this register will initiate a new transaction of the NF controller. Command registers 0-3 are used to trigger controller operation. Fields encoding of those registers depends on selected work mode. Possible work modes are: 1) CMD DMA work mode, 2) PIO Mode, 3) Generic Sequence Mode.

Table 7.1. cmd_reg0

Bits	SW	Name	Description	Reset
31:0	R/W	cmd0	Command 0 register field.	32'h0

7.1.2. cmd_reg1 (0x0004)

Description: Command register 1.

Table 7.2. cmd_reg1

Bits	SW	Name	Description	Reset
31:0	R/W	cmd1	Command 1 register field.	32'h0

7.1.3. cmd_reg2 (0x0008)

Description: Command register 2.

Table 7.3. cmd_reg2

Bits	SW	Name	Description	Reset
31:0	R/W	cmd2	Command 2 register field.	32'h0

7.1.4. cmd_reg3 (0x000c)

Description: Command register 3.

Table 7.4. cmd_reg3

Bits	SW	Name	Description	Reset
31:0	R/W	cmd3	Command 3 register field.	32'h0

7.1.5. cmd_status_ptr (0x0010)

Description: Pointer register to select which thread status will be selected.

Table 7.5. cmd_status_ptr

Bits	SW	Name	Description	Reset
31:3	R	RSVD	Reserved	29'h0

Bits	SW	Name	Description	Reset
2:0	R/W	thrd_status_sel	Number of thread whose status will be available in cmd_status register.	3'h0

7.1.6. cmd_status (0x0014)

Description: Command status register for selected thread.

Table 7.6. cmd_status

Bits	SW	Name	Description	Reset
31:0	R	cmd_status	Command status register field. This field gives the software direct access to selected thread descriptor's status. Number of accessed thread can be selected with cmd_status_ptr register	32'h0

7.1.7. intr_status (0x0110)

Description: Controller status register

Table 7.7. intr_status

Bits	SW	Name	Description	Reset
31:23	R	RSVD	Reserved	9'h0
22	W1C	sdma_err	This bit is set when not allowed access to the Slave DMA interface is detected	1'h0
21	W1C	sdma_trigg	This bit is set when trigger condition for the Slave DMA is met.	1'h0
20	W1C	cmd_ignored	Detected sending of command to busy thread and ignored it.	1'h0
19	R	RSVD	Reserved	1'h0
18	W1C	ddma_terr	Master Data DMA Target error. This bit will be set if Master DMA Data engine module detects system bus error during reading or writing data	1'h0
17	W1C	cdma_terr	Command DMA Target error. This bit will be set if Command Engine module detects system bus error during reading descriptor from system memory or during descriptor status field write operation	1'h0
16	W1C	cdma_idle	Command DMA is in the Idle state	1'h0
15:0	R	RSVD	Reserved	16'h0

7.1.8. intr_enable (0x0114)

Description: Interrupt enable register. If selected bit of this register is set, rising edge of the corresponding bit in intr_status will generate setting of external interrupt line.

Table 7.8. intr_enable

Bits	SW	Name	Description	Reset
31	R/W	intr_en	Global Interrupts enable flag.	1'h0
30:23	R	RSVD	Reserved	8'h0

Bits	SW	Name	Description	Reset
22	R/W	sdma_err_en	Enables interrupt when not allowed access to the Slave DMA interface is detected	1'h0
21	R/W	sdma_trigg_en	Enables interrupt when trigger condition for the Slave DMA is met.	1'h0
20	R/W	cmd_ignored_en	Interrupt enable for detecting of ignored command.	1'h0
19	R	RSVD	Reserved	1'h0
18	R/W	ddma_terr_en	Interrupt enable for detecting Data DMA Master target error.	1'h0
17	R/W	cdma_terr_en	Interrupt enable for detecting CMD Engine target error.	1'h0
16	R/W	cdma_idle_en	Interrupt enable for detecting Command Engine IDLE	1'h0
15:0	R	RSVD	Reserved	16'h0

7.1.9. ctrl_status (0x0118)

Description: Controller internal state.

Table 7.9. ctrl_status

Bits	SW	Name	Description	Reset
31:17	R	RSVD	Reserved	15'h0
16	W1C	sdma_paused	This bit indicates that Slave DMA transaction was aborted due to detected error/incorrect access type on system interface. Controller will not invoke transfer of next data block through the Slave DMA interface if this bit is set.	1'h0
15:11	R	RSVD	Reserved	5'h0
10	R	init_fail	Initialization process failed.	1'h0
9	R	init_comp	The Cadence NAND Flash Memory Controller has completed its reset and initialization process.	1'h0
8	R	ctrl_busy	This bit indicates if controller is in the busy state or not. The 0x1 value informs that controller is in the busy state. This bit is routed to the controller interface as ctrl_busy pin.	1'h0
7:4	R	RSVD	Reserved	4'h0
3	R	mc_busy	If 1 the Mini Controller internal module is busy.	1'h0
2	R	cmd_eng_busy	If 1 the Command Engine internal module is busy.	1'h0
1	R	mdma_busy	If 1 the Master DMA internal module is busy.	1'h0
0	R	sdma_busy	If 1 the Slave DMA internal module is busy.	1'h0

7.1.10. trd_status (0x0120)

Description: Command Engine threads state.

Table 7.10. trd_status

Bits	SW	Name	Description	Reset
31:8	R	RSVD	Reserved	24'h0

Bits	SW	Name	Description	Reset
7:0	R	trd_busy	Indicates Command Engine thread busy status. If 1 corresponding thread is busy.	8'h0

7.1.11. trd_error_intr_status (0x0128)

Description: Thread error indicates that the Command Engine thread detected an error condition. To get more information on the error, s/w needs to read the status field of the descriptor or appropriate status register depending on current work mode.

Table 7.11. trd_error_intr_status

Bits	SW	Name	Description	Reset
31:8	R	RSVD	Reserved	24'h0
7	W1C	trd7_error_stat	Thread 7 error.	1'h0
6	W1C	trd6_error_stat	Thread 6 error.	1'h0
5	W1C	trd5_error_stat	Thread 5 error.	1'h0
4	W1C	trd4_error_stat	Thread 4 error.	1'h0
3	W1C	trd3_error_stat	Thread 3 error.	1'h0
2	W1C	trd2_error_stat	Thread 2 error.	1'h0
1	W1C	trd1_error_stat	Thread 1 error.	1'h0
0	W1C	trd0_error_stat	Thread 0 error.	1'h0

7.1.12. trd_error_intr_en (0x0130)

Description: Interrupt enable register. If selected bit of this register is set, rising edge of corresponding bit in trd_error_intr_status will cause setting of the external interrupt line.

Table 7.12. trd_error_intr_en

Bits	SW	Name	Description	Reset
31:8	R	RSVD	Reserved	24'h0
7:0	R/W	trd_error_intr_en	Interrupt enable for detecting thread error.	8'h0

7.1.13. trd_comp_intr_status (0x0138)

Description: Each bit of this field correspond to the Command Engine thread. Each bit informs about descriptor status for selected thread. It is set only when INT bit of descriptor is set

Table 7.13. trd_comp_intr_status

Bits	SW	Name	Description	Reset
31:8	R	RSVD	Reserved	24'h0
7	W1C	trd7_comp	Thread 7 operation complete flag.	1'h0
6	W1C	trd6_comp	Thread 6 operation complete flag.	1'h0
5	W1C	trd5_comp	Thread 5 operation complete flag.	1'h0
4	W1C	trd4_comp	Thread 4 operation complete flag.	1'h0
3	W1C	trd3_comp	Thread 3 operation complete flag.	1'h0

Bits	SW	Name	Description	Reset
2	W1C	trd2_comp	Thread 2 operation complete flag.	1'h0
1	W1C	trd1_comp	Thread 1 operation complete flag.	1'h0
0	W1C	trd0_comp	Thread 0 operation complete flag.	1'h0

7.1.14. dma_target_error_l (0x0140)

Description: DMA target error address [31:0]. This register can be used to obtain address of transaction which caused setting of cdma_terr or ddma_terr bits in the intr_status register.

Table 7.14. dma_target_error_l

Bits	SW	Name	Description	Reset
31:0	R	target_err_l	Address of the first transaction on the master interface that returned error response.	32'h0

7.1.15. dma_target_error_h (0x0144)

Description: DMA target error address [63:32]. This register can be used to obtain address of transaction which caused setting of cdma_terr or ddma_terr bits in the intr_status register.

Table 7.15. dma_target_error_h

Bits	SW	Name	Description	Reset
31:0	R	target_err_h	Address of the first transaction on the master interface that returned error response.	32'h0

7.1.16. boot_status (0x0148)

Description: This register provides status of the latest boot operation.

Table 7.16. boot_status

Bits	SW	Name	Description	Reset
31:8	R	RSVD	Reserved	24'h0
7	R	tim_out_err	This field (when set) indicates that the time out error occurred during boot sequence.	1'h0
6	R	RSVD	Reserved	1'h0
5:4	R	cpy_id	This field identifies a boot block used in the latest boot sequence run. Allowed values are: • 00 - block 0 was used, • 01 - block 1 was used and • 10 - block 2 was used.	2'h0
3	R	RSVD	Reserved	1'h0
2	R	bus_err	This field describes bus status during boot process. If it is set to high, the boot process failed due to the bus interface receiving an error response from the target. Allowed values are:	1'h0

Bits	SW	Name	Description	Reset
			• 0 - no error detected, • 1 - error detected.	
1:0	R	bch_err_type	This field identifies kind of error detected during boot sequence process. Allowed values are: • 00 - no errors detected, • 01 - correctable error detected, • 10 - uncorrectable error detected.	2'h0

7.1.17. trd_timeout_intr_status (0x014c)

Description: Timeout status register indicates that a timeout condition on the Command Engine thread was detected.

Table 7.17. trd_timeout_intr_status

Bits	SW	Name	Description	Reset
31:8	R	RSVD	Reserved	24'h0
7	W1C	trd7_timeout_stat	Thread 7 timeout.	1'h0
6	W1C	trd6_timeout_stat	Thread 6 timeout.	1'h0
5	W1C	trd5_timeout_stat	Thread 5 timeout.	1'h0
4	W1C	trd4_timeout_stat	Thread 4 timeout.	1'h0
3	W1C	trd3_timeout_stat	Thread 3 timeout.	1'h0
2	W1C	trd2_timeout_stat	Thread 2 timeout.	1'h0
1	W1C	trd1_timeout_stat	Thread 1 timeout.	1'h0
0	W1C	trd0_timeout_stat	Thread 0 timeout.	1'h0

7.1.18. trd_timeout_intr_en (0x0154)

Description: Interrupt enable register. If selected bit of this register is set rising edge of corresponding bit in trd_timeout_intr_status will cause setting of the external interrupt line.

Table 7.18. trd_timeout_intr_en

Bits	SW	Name	Description	Reset
31:8	R	RSVD	Reserved	24'h0
7:0	R/W	trd_timeout_intr_en	Interrupt enable for detecting thread timeout.	8'h0

7.2. Config Registers. (0x0400)

7.2.1. transfer_cfg_0 (0x0400)

Description: Transfer config 0 register. It is utilized to configure data transfer parameters. Value of this register is valid only for command type of program pages and read pages. For other commands transfer size parameters are selected automatically according to chosen command type.

Table 7.19. transfer_cfg_0

Bits	SW	Name	Description	Reset
31:16	R/W	offset	Offset value from the beginning of the page, used for data transfer. If ECC correction is enabled value of this field must be aligned to beginning of the single ECC sector size. Field ignored for the Generic work mode.	16'h0
15:8	R	RSVD	Reserved	8'h0
7:0	R/W	sector_cnt	Number of sectors which will be transferred within single NF device's page. This is one of the fields which determines size of data transfer within one NAND Flash device's page (sector_size * (sector_cnt-1) + last_sector_size). Controller will ignore value 0 for this field and replace it with 1. At least one transfer always will occur. Field ignored for the Generic work mode.	8'h1

7.2.2. transfer_cfg_1 (0x0404)

Description: Transfer config 1 register. It is utilized to configure data transfer parameters. Value of this register is valid only for command type of program pages, read pages and copyback. For other commands transfer size parameters are selected automatically according to chosen command.

Table 7.20. transfer_cfg_1

Bits	SW	Name	Description	Reset
31:16	R/W	last_sector_size	Size of last data sector. This is one of the fields which determines size of data transfer within one NAND Flash device's page (sector_size * (sector_cnt-1) + last_sector_size). If ECC checking is enabled the values of this register should be correct with respect to ECC engine requirements. Field ignored for the Generic work mode.	16'h1000
15:0	R/W	sector_size	Size of not-last data sector. This is one of the fields which determines size of data transfer within one NAND Flash device's page (sector_size * (sector_cnt-1) + last_sector_size). If ECC checking is enabled the values of this register should be correct with respect to ECC engine requirements. Field ignored for the Generic work mode.	16'h1000

7.2.3. long_polling (0x0408)

Description: Wait count value for long polling.

Table 7.21. long_polling

Bits	SW	Name	Description	Reset
31:16	R	RSVD	Reserved	16'h0
15:0	R/W	long_polling	Number of system clock cycles after issue of erase/write/read operation before the controller starts to poll for status. This value is valid only in the status polling mode. First polling will happen after this many number of system clock cycles. Then on polling will happen every short_polling cycles. The long	16'h3e8

Bits	SW	Name	Description	Reset
			polling value should be significantly larger the short polling value.	

7.2.4. short_polling (0x040c)

Description: Status monitor cycle count value.

Table 7.22. short_polling

Bits	SW	Name	Description	Reset
31:16	R	RSVD	Reserved	16'h0
15:0	R/W	short_polling	Number of system clocks after long polling delay before the controller starts to poll for status if first status poll attempt returned information that controller is busy. The long polling value should be significantly larger the short polling value.	16'h1f4

7.2.5. rdst_ctrl_0 (0x0410)

Description: Device ready status control register.

Table 7.23. rdst_ctrl_0

Bits	SW	Name	Description	Reset
31:24	R/W	ready_mask	If rb_enable=0 and then this field determines the mask used to comparison of response from the NF memory to the device status command and ready_value	8'h40
23:16	R/W	ready_value	If rb_enable=0 then this field determines the value which will be compared to response from the NF memory to the device status command	8'h40
15:1	R	RSVD	Reserved	15'h0
0	R/W	rb_enable	Selects R/B pin checks (if 1) or status polling mode (if 0). For Multi-LUN operations this bit is ignored and Read Status Enhanced command is send to check LUN status.	1'h1

7.2.6. rdst_ctrl_1 (0x0414)

Description: Operation status control register. Controller doesn't apply any implicit checks, so host needs to unmask all status bits that need to be checked to detect error condition

Table 7.24. rdst_ctrl_1

Bits	SW	Name	Description	Reset
31:24	R/W	error_mask	This field determines the mask used in comparison of response from the NF memory to the operation status command and status_value	8'h41
23:16	R/W	error_value	This field determines the value which will be compared to response from the NF memory to the operation status command	8'h41

Bits	SW	Name	Description	Reset
15:0	R	RSVD	Reserved	16'h0

7.2.7. lun_status_cmd (0x0418)

Description: Indicates the command to be sent while checking status of the next LUN.

Table 7.25. lun_status_cmd

Bits	SW	Name	Description	Reset
31:1	R	RSVD	Reserved	31'h0
0	R/W	lun_stat_sel	This field allow to select command sequence that will be used to check LUN status. Allowed values are: • 0 - the 78-addr sequence will be use for both first and following LUN-s. • 1 - the F1 command will be used for a first LUN and F2 will be used for second one. This option is allowed only when device has two LUN-s.	1'h0

7.2.8. lun_interleaved_cmd (0x041c)

Description: Interleaved commands support.

Table 7.26. lun_interleaved_cmd

Bits	SW	Name	Description	Reset
31:7	R	RSVD	Reserved	25'h0
6	R/W	program_after_read	This bit informs the controller if the device supports a program operation on a LUN while a Read operation is already ongoing in the other LUN.	1'h0
5:2	R	RSVD	Reserved	4'h0
1:0	R/W	lun_col_cmd	This field allow to select change read column sequence type that will be used for multi-LUN commands: • 2'b00 - use simple 05-addr2/5-e0 sequence. Change read column command type is selected by the chrc_wdth field in the device_ctrl register. • 2'b01 - use the 60-addr5-e0 sequence • 2'b10 - use the 00-addr-05-addr2-e0	2'h0

7.2.9. lun_addr_offset (0x0420)

Description: Indicates the starting address of next LUN.

Table 7.27. lun_addr_offset

Bits	SW	Name	Description	Reset
31:5	R	RSVD	Reserved	27'h0

Bits	SW	Name	Description	Reset
4:0	R/W	lun_addr_offset	Bit in ROW address used for selection of the LUN	5'h0

7.2.10. nf_dev_layout (0x0424)

Description: NF device layout.

Table 7.28. nf_dev_layout

Bits	SW	Name	Description	Reset
31:27	R/W	blk_addr_idx	Block address offset - bit index at which block address starts inside the row address. If the page address and block address are continuous inside the row address then block address offset field value is equal to LOG2(PPB).	5'h0
26:24	R	RSVD	Reserved	3'h0
23:20	R/W	LN	The number of LUN presents in the device. Up to 8 LUN-s are supported.	4'h1
19:17	R	RSVD	Reserved	3'h0
16	R/W	lun_en	Enables Multi LUN operations	1'h0
15:0	R/W	PPB	Pages Per Block - number of pages in a block.	16'h0

7.2.11. ecc_config_0 (0x0428)

Description: ECC engine configuration register 0.

Table 7.29. ecc_config_0

Bits	SW	Name	Description	Reset
31:11	R	RSVD	Reserved	21'h0
10:8	R/W	corr_str	Correction strength. This field selects correction abilities available for the ECC engine. Correction abilities are coded using binary code starting from the lower values to higher values. Only values supported by the ECC engine are allowed. Field ignored for the Generic work mode.	3'h0
7:5	R	RSVD	Reserved	3'h0
4	R/W	scrambler_en	This enables scrambler logic in the controller. Scrambler removes the repeated patterns in the data and decreases the chances of read disturbance and program disturbance. Field ignored for the Generic work mode.	1'h0
3:2	R	RSVD	Reserved	2'h0
1	R/W	erase_det_en	Enable erased pages detection mechanism. Field ignored for the Generic work mode.	1'h0
0	R/W	ecc_enable	Enable controller ECC check bits generation and correction. Field ignored for the Generic work mode.	1'h0

7.2.12. ecc_config_1 (0x042c)

Description: Erase detection config register

Table 7.30. ecc_config_1

Bits	SW	Name	Description	Reset
31:8	R	RSVD	Reserved	24'h0
7:0	R/W	erase_det_lvl	This value informs the ECC logic about the number of zeros inside transfered sector that still allow consider it as Erased. If the number of zeros inside the sector being read is less than the value in this register, an erased sector is inferred. This value is used only when ECC engine is disabled, when ECC engine is enabled selected correction ability is used as number of zeros limit.	8'h0

7.2.13. device_ctrl (0x0430)

Description: Device control register.

Table 7.31. device_ctrl

Bits	SW	Name	Description	Reset
31:8	R	RSVD	Reserved	24'h0
7	R/W	four_addr_seq_en	Bit used to enable/disable the four bytes address sequence for older devices. If set it will cause that controller will send only two row address bytes instead of three as per default.	1'h0
6	R	RSVD	Reserved	1'h0
5	R/W	chrc_wdth	If this bit is cleared then controller will use the change read column sequence with two byte address (05-2xaddr-E0), when it is set then controller will use sequence with five byte address (05-5xaddr-E0).	1'h0
4	R/W	time_out_en	If this bit is set then Command Engine time out mechanism is enabled.	1'h1
3	R/W	cont_on_err	If this bit is cleared and any operation programed by descriptor fails, then controller will drop current descriptors chain execution. When it is set description execution will be continued. Thread Reset command occurrence breaks descriptor chain regardless of this bit.	1'h0
2	R	RSVD	Reserved	1'h0
1	R/W	ce_pin_reduct	Device supports CE pin reduction with volume assignments, volume addressing and volume change command sequence. For any device configured by host to be used in volume addressing mode, this bit must be set to 1. • 1 - supported • 0 - Not supported	1'h0
0	R/W	ce_hold	If this field is set then CE bus state will be preserved between commands. This field is dedicated to the volume assignment process where it is required to hold low CE line through whole process. Some devices may require to keep the CE# low through the whole SetFeature sequence. ce_hold can be used for this purpose also.	1'h0

7.2.14. multiplane_config (0x0434)

Description: Multiplane settings register. The Address part of sequence is described using three symbols: Addr2 - it mean column address only, Addr3 - it mean row address only, Addr5 - it mean both row and column address

Table 7.32. multiplane_config

Bits	SW	Name	Description	Reset
31:27	R	RSVD	Reserved	5'h0
26	R/W	pl_status_en	Field select status probing mechanism: • 1'b0 - summary status for all planes is probed using single basic read status command, • 1'b1 - separate status for each plane is probed using multiple enhanced read status commands.	1'h0
25	R/W	last_wr_cmd	Select sequence for the Copyback Write operation: • 1'b0 - used sequence is 85-Addr5-11 ... 85-Addr5-11 ... 85-Addr5-10 • 1'b1 - used sequence is 85-Addr5-11 ... 81-Addr5-11 ... 81-Addr5-10	1'h0
24	R/W	mpl_erase_seq	Select Erase sequence in multiplane work mode: • 1'b0 - ONFI sequence 60-Addr3-d1 60-Addr3-d0 • 1'b1 - JEDEC sequence 60-Addr3-60-Addr3-d0	1'h0
23:21	R/W	mpl_rd_seq	Selects Multiplane read command sequences. The HPNFC controller can send the following sequences: • 3'b000 - This value informs controller that the sequence send to target device will look like: 00-Addr5_0-32, 00-Addr5_1-32, ... ,00-Addr5_n-30, 06-Addr5_0-E0-Data ... • 3'b001 - This value informs controller that the sequence send to target device will look like: 00-Addr5/2_0-32, 00-Addr5_1-32, ... ,00-Addr5_n-30, 05-Addr5/2_0-E0-Data ..., change read column command type is selected by the chrc_wdth field in the device_ctrl register. For this sequence the chrc_wdth need to be set, because higher three bytes are used to select active plane. • 3'b010 - This value informs controller that the sequence send to target device will look like: 60-Addr3_0-60-Addr3_1 , ..., 60-Addr3_n-30, 00-Addr5_0-05-Addr2-E0-DATA ... • 3'b100 - This value informs controller that the sequence send to target device will look like: 00-Add5_0-32, 00-Addr5_1-32, ... ,00-Addr5_n-30, 00-Addr5_0-05-Addr2-E0 ...	3'h0

Bits	SW	Name	Description	Reset
			• 3'b110 - This value informs controller that the sequence send to target device will look like: 60-Addr3_0-60-Addr3_1 , ..., 60-Addr3_n-C30/C33, 00-Addr5_0-05-Addr2_0-E0-DATA ... . 30 is selected when cache operations are disabled, C33 is selected when cache operations are enabled	
20:18	R	RSVD	Reserved	3'h0
17:16	R/W	mpl_prg_seq	Selects Multiplane program command sequences. If the device has N planes, the values in the field should be as follows based on which sequence the target device expects: • 2'h0 - This value informs the controller that the sequence to follow is 80-Addr5-Data-11 repeated for planes 0 to N-2 ... 80-Addr5-Data-10 for plane N-1 • 2'h1 - This value informs the controller that the sequence to follow is 80-Addr5-Data-11 repeat for planes 0 to N-2... 81-Addr5-Data-10 for plane N-1 • 2'h2 - This value informs the controller that the sequence to follow is 80-Addr5-Data-11 or plane 0 ... 81-Addr5-Data-11 for planes 1 to N-2.... 81-Addr5-Data-10 for plane N-1 • 2'h3 - Reserved.	2'h0
15:10	R	RSVD	Reserved	6'h0
9:8	R/W	mpl_pl_num	Selects number of planes per device. Supported values are: • 2'h0 - single plane • 2'h1 - two planes • 2'h2 - four planes • 2'h3 - eight planes	2'h0
7:5	R	RSVD	Reserved	3'h0
4	R/W	mpl_cpbk_rd_seq	Selects Multiplane copyback read command sequences. The HPNFC controller can send the following sequences: • 1'b0 - This value informs the controller that the sequence to follow is 00-Addr5-32, ..., 00-Addr5-35 • 1'b1 - This value informs the controller that the sequence to follow is 60-Addr3-60-Addr3-35, 00-Addr5-05-Addr2-e0	1'h0
3:2	R	RSVD	Reserved	2'h0
1	R/W	mpl_wr_en	This bit enables multiplane sequences for write and erase operations.	1'h0
0	R/W	mpl_rd_en	This bit enables multiplane sequences for read operations.	1'h0

7.2.15. cache_config (0x0438)

Description: This register contains enable flags for cache operations.

Table 7.33. cache_config

Bits	SW	Name	Description	Reset
31:2	R	RSVD	Reserved	30'h0
1	R/W	cache_wr_en	This bit enables cache write command sequences support.	1'h0
0	R/W	cache_rd_en	This bit enables cache read command sequences support.	1'h0

7.2.16. dma_settings (0x043c)

Description: DMA settings register. It is common register for both Master and Slave interface.

Table 7.34. dma_settings

Bits	SW	Name	Description	Reset
31:18	R	RSVD	Reserved	14'h0
17	R/W	sdma_err_rsp	If this bit is set then ERROR response will be returned if host tries to access unprepared Slave DMA interface. If this bit will be cleared the OK response is returned.	1'h0
16	R/W	OTE	Outstanding transaction enable. It only applies to the master interface, the slave interface will ignore this bit and will accept all incoming transactions.	1'h0
15:8	R	RSVD	Reserved	8'h0
7:0	R/W	burst_sel	Sets the burst used by data DMA for transferring data to/from flash device. The maximum burst size can be calculated as burst_sel+1. This field should be changed only if controller is in IDLE state	8'h0

7.2.17. sdma_size (0x0440)

Description: Transferred data block size for the Slave DMA module.

Table 7.35. sdma_size

Bits	SW	Name	Description	Reset
31:0	R	sdma_size	Transferred data block size in bytes for the Slave DMA module. Data size is rounded up to the data bus word size.	32'h0

7.2.18. sdma_trd_num (0x0444)

Description: Thread number associated with transferred data block for the Slave DMA module.

Table 7.36. sdma_trd_num

Bits	SW	Name	Description	Reset
31:3	R	RSVD	Reserved	29'h0

Bits	SW	Name	Description	Reset
2:0	R	sdma_trd	Thread number associated with transferred data block for the Slave DMA module.	3'h0

7.2.19. time_out (0x0448)

Description: This register configures time out delay

Table 7.37. time_out

Bits	SW	Name	Description	Reset
31:0	R/W	time_out_val	This value will be used to initialize the watchdog module. It configures max allowed time period for command to execute. If value 0 will be programed then time out feature will be disabled.	32'hffffffff

7.2.20. sdma_addr0 (0x044c)

Description: This register stores the buffer address in the host memory that will be used as a sink/source for the SDMA transfer. The SDMA address is based on the Memory Pointer field that was programed by the host as part of the CDMA/PIO command. A single CDMA/PIO command can trigger multiple transfers on the slave interface, so the SDMA address value will be automatically incremented and updated before each SDMA transfer.

Table 7.38. sdma_addr0

Bits	SW	Name	Description	Reset
31:0	R	sdma_addr_l	The SDMA destination/source address - lower part.	32'h0

7.2.21. sdma_addr1 (0x0450)

Description: This register stores the buffer address in the host memory that will be used as a sink/source for the SDMA transfer. The SDMA address is based on the Memory Pointer field that was programed by the host as part of the CDMA/PIO command. A single CDMA/PIO command can trigger multiple transfers on the slave interface, so the SDMA address value will be automatically incremented and updated before each SDMA transfer.

Table 7.39. sdma_addr1

Bits	SW	Name	Description	Reset
31:0	R	sdma_addr_h	The SDMA destination/source address - higher part.	32'h0

7.2.22. control_data_ctrl (0x0494)

Description: Register configures control-data part of transfered data block. Value of this register is valid only for command type of program pages, read pages and copyback. For other commands transfer size parameters are selected automatically according to chosen command.

Table 7.40. control_data_ctrl

Bits	SW	Name	Description	Reset
31:16	R	RSVD	Reserved	16'h0

Bits	SW	Name	Description	Reset
15:0	R/W	control_data_size	Control-data size.	16'h0

7.3. Debug Interface registers. (0x0740)

7.3.1. dbg_ctrl (0x0740)

Description: Debug interface control register.

Table 7.41. dbg_ctrl

Bits	SW	Name	Description	Reset
31:14	R	RSVD	Reserved	18'h0
13:8	R/W	word_sel	Field select with word of the debug vector that will be available through the dbg_data register	6'h0
7:4	R	RSVD	Reserved	4'h0
3	R/W	b_latch_req	Enable debug work mode where data from source clock domain are synchronized when transfered to the SYS_CLK clock domain. Bit control behavior of the BCH_CLK clock domain.	1'h0
2	R	RSVD	Reserved	1'h0
1	R/W	n_latch_req	Enable debug work mode where data from source clock domain are synchronized when transfered to the SYS_CLK clock domain. Bit control behavior of the NF_CLK clock domain.	1'h0
0	R/W	int_sel	Bit selects how the debug interface will be controlled: • 0 - Hardware interface will be used, • 1 - Register interface will be used.	1'h0

7.3.2. dbg_stat (0x0744)

Description: Debug interface status register.

Table 7.42. dbg_stat

Bits	SW	Name	Description	Reset
31:5	R	RSVD	Reserved	27'h0
4	W1C	b_latch_ack	Strobe flag for debug data transfered from the source clock domain when clock domain synchronization is enabled. Signal will be set by hardware after stable data from the source clock domain will be received data remain stable as long as this bit will be set. Bit need to be clear by software. Bit control behavior of the BCH_CLK clock domain.	1'h0
3	R	RSVD	Reserved	1'h0
2	W1C	n_latch_ack	Strobe flag for debug data transfered from the source clock domain when clock domain synchronization is enabled. Signal will be set by hardware after stable data from the source clock	1'h0

Bits	SW	Name	Description	Reset
			domain will be received data remain stable as long as this bit will be set. Bit need to be clear by software. Bit control behavior of the NF_CLK clock domain.	
1:0	R	RSVD	Reserved	2'h0

7.3.3. dbg_data (0x0748)

Description: Debug interface data register.

Table 7.43. dbg_data

Bits	SW	Name	Description	Reset
31:0	R	dbg_data	Part of the debug vector selected by the word_sel field of the dbg_ctrl register	32'h0

7.4. Controller and Device Parameters. (0x0800)

7.4.1. ctrl_version (0x0800)

Description: Register contains release identification number.

Table 7.44. ctrl_version

Bits	SW	Name	Description	Reset
31:16	R	RSVD	Reserved	16'h0
15:8	R	ctrl_fix	Fixed number (minor revision number).	8'h0
7:0	R	ctrl_rev	Controller revision number.	8'ha

7.4.2. ctrl_features_reg (0x0804)

Description: Shows available hardware features of the controller

Table 7.45. ctrl_features_reg

Bits	SW	Name	Description	Reset
31:30	R	RSVD	Reserved	2'h0
29	R	nf_16b_supp	Support for 16b NF interface.	1'h1
28	R	nvddr2_3	Support for NV-DDR2/3 work mode.	1'h0
27	R	nvddr	Support for NV-DDR (source synchronous) work mode.	1'h0
26	R	async_supp	Support for asynchronous work mode.	1'h1
25:24	R	n_banks	Maximum number of banks supported by hardware. This is an encoded value. • 0 - One bank • 1 - Two banks • 2 - Four banks	2'h1

Bits	SW	Name	Description	Reset
			• 3 - Eight banks	
23:22	R	sfr_intf	SFR interface type (0-AXI4 Lite, 1-OCP, 2-APB, other values reserved).	2'h0
21	R	dma_data_width	Slave and Master DMA data width: • 0 - 32bit • 1 - 64bit	1'h1
20	R	dma_addr_width	Slave and Master DMA address width: • 0 - 32bit • 1 - 64bit	1'h1
19:18	R	dma_intf	DMA interface type (0-AXI4, 1-OCP, other values reserved).	2'h0
17	R	ecc_available	Data ECC protection engine present.	1'h1
16	R	boot_available	Boot feature present.	1'h1
15	R	pre_fetch_available	Availability of pre-fetching mechanism.	1'h0
14	R	di_available	Availability of Data Integrity mechanism.	1'h0
13	R	ext_cmd_cnt	Availability Extended Command Feature. Reflects the maximum number of operations in single PIO command/ CDMA descriptor: • 0 - 256 operations • 1 - 65536 operations	1'h0
12	R	rmp_available	Availability of Remap mechanism.	1'h0
11	R	ext_status	Availability Extended Status feature.	1'h0
10	R	control_data	Availability of Control Data feature.	1'h1
9:4	R	RSVD	Reserved	6'h0
3:0	R	n_threads	Number of threads available in the controller. The following decoding is used: • 0 - One thread • 1 - Two threads • 2 - Four threads • 3 - Eight threads • 4 - Sixteen threads • 5-15 - Reserved	4'h3

7.4.3. manufacturer_id (0x0808)

Description: NAND Flash memory device ID information. This register is updated according to ReadID command result during controller initialization phase.

Table 7.46. manufacturer_id

Bits	SW	Name	Description	Reset
31:24	R	RSVD	Reserved	8'h0
23:16	R	dId	Device ID	8'h0
15:8	R	RSVD	Reserved	8'h0
7:0	R	mId	Manufacturer ID	8'h0

7.4.4. nf_device_areas (0x080c)

Description: Device areas settings.

Table 7.47. nf_device_areas

Bits	SW	Name	Description	Reset
31:16	R	spare_area_size	Spare area size in bytes for the NF device page	16'h0
15:0	R	main_area_size	Main area size in bytes for the NF device page	16'h1000

7.4.5. device_params_0 (0x0810)

Description: Indicates the device type and the number of LUN-s present in the device.

Table 7.48. device_params_0

Bits	SW	Name	Description	Reset
31:30	R	device_type	Indicates if the device is an ONFI- or JEDEC-compliant device. • 0 - device type undetected • 1 - ONFI compliant device • 2 - Reserved • 3 - Legacy device	2'h0
29:24	R	RSVD	Reserved	6'h0
23:16	R	bits_per_cell	Number of bits per cell.	8'h0
15:8	R	plane_addr_bits	Number of bits used to addressing planes.	8'h0
7:0	R	no_of_luns	Indicates the number of LUNs present in the NandFlash device.	8'h1

7.4.6. device_params_1 (0x0814)

Description: Device signature register.

Table 7.49. device_params_1

Bits	SW	Name	Description	Reset
31:16	R	RSVD	Reserved	16'h0

Bits	SW	Name	Description	Reset
15:8	R	ReadId_4	4th byte of ReadID command related to Device Signature (ONFI or JEDEC).	8'h0
7:0	R	ReadId_3	3th byte of ReadID command related to Device Signature (ONFI or JEDEC).	8'h0

7.4.7. device_features (0x0818)

Description: Features and optional commands supported by the connected ONFI device.

Table 7.50. device_features

Bits	SW	Name	Description	Reset
31:16	R	optional_commands	This field is a copy of bytes 8-9 from parameter page ('Optional commands supported'). Typically the values in the field should be interpreted as follows:[list] [*]Bit 0 - Supports page cache program command. [*]Bit 1 - Supports read cache commands. [*]Bit 2 - Supports get and set features. [*]Bit 3 - Supports read status enhanced commands. [*]Bit 4 - Supports copyback. [*]Bit 5 - Supports Read Unique Id. [*]Bit 6 - Supports Change Read Column Enhanced for ONFI device. [*]Bit 7 - Supports change row address for ONFI device. [*]Bit 8 - Supports Change small data move. [*]Bit 9 - Supports RESET LUN. [*]Bit 10 - Supports Volume Select for ONFI device. [*]Bit 11 - Supports ODT Configure for ONFI device. [*]Bit 12 - Supports LUN Get and LUN Set Features for ONFI device. [*]Bit 13 - Supports ZQ calibration - Long and Short for ONFI device. [*]Bit 14-15 - Reserved.	16'h0
15:0	R	device_features	This field corresponds to bytes 6-7 of parameter page ('Features supported'). Typically the values in the field should be interpreted as follows: • Bit 0 - Supports 16 bit data bus width. • Bit 1 - Supports multiple LUN operations. • Bit 2 - Supports non-sequential page programming. • Bit 3 - Supports interleaved program and erase operations. • Bit 4 - Supports odd to even page copyback for ONFI device. • Bit 5 - Supports synchronous DDR mode. • Bit 6 - Supports interleaved read operations for ONFI device. • Bit 7 - Supports extended parameter page for ONFI device. • Bit 8 - Supports program page register clear enhancement. • Bit 9 - Supports EZNAND for ONFI devices.	16'h0

Bits	SW	Name	Description	Reset
			• Bit 10 - Supports NV-DDR2 for ONFI devices. • Bit 11 - Supports Volume Addressing for ONFI devices. • Bit 12 - Supports External Vpp for ONFI devices. • Bit 13 - Supports NV-DDR3 for ONFI devices. • Bit 14 - Supports ZQ calibration for ONFI devices. • Bit 15 - Supports Package Electrical Specification for ONFI devices.	

7.4.8. device_blocks_per_lun (0x081c)

Description: Number of blocks per LUN present in the ONFI complaint device.

Table 7.51. device_blocks_per_lun

Bits	SW	Name	Description	Reset
31:0	R	no_of_blocks	Indicates the number of blocks per LUN present in the ONFI complaint device.	32'h0

7.4.9. device_revision (0x0820)

Description: Device revision version (valid for ONFI device).

Table 7.52. device_revision

Bits	SW	Name	Description	Reset
31:16	R	RSVD	Reserved	16'h0
15:0	R	revisions	This field is a copy of bytes 4-5 from parameter page ('Revision number'). The values in the field should be interpreted as follows • Bit 0 - Reserved • Bit 1 - support ONFI version 1.0. • Bit 2 - support ONFI version 2.0. • Bit 3 - support ONFI version 2.1. • Bit 4 - support ONFI version 2.2. • Bit 5 - support ONFI version 2.3. • Bit 6 - support ONFI version 3.0. • Bit 7 - support ONFI version 3.1. • Bit 8 - support ONFI version 3.2.	16'h0

Bits	SW	Name	Description	Reset
			• Bit 9 - support ONFI version 4.0. • Bit 10-15 - Reserved.	

7.4.10. onfi_timing_modes_0 (0x0824)

Description: Device Timing modes supported by the connected ONFI device.

Table 7.53. onfi_timing_modes_0

Bits	SW	Name	Description	Reset
31:24	R	RSVD	Reserved	8'h0
23:16	R	nv_ddr_modes	This field corresponds to byte 141 in parameter page ('NV-DDR timing mode support'). Typically it should be interpreted as follows: • Bit 0 - Supports Timing mode 0. • Bit 1 - Supports Timing mode 1. • Bit 2 - Supports Timing mode 2. • Bit 3 - Supports Timing mode 3. • Bit 4 - Supports Timing mode 4. • Bit 5 - Supports Timing mode 5. • Bit 6-7 - Reserved.	8'h0
15:0	R	sdr_modes	This field reflects value of bytes 129-130 in parameter page (' SDR timing mode support'). Typically the value of this field should be interpreted as follows: • Bit 0 - Supports Timing mode 0. • Bit 1 - Supports Timing mode 1. • Bit 2 - Supports Timing mode 2. • Bit 3 - Supports Timing mode 3. • Bit 4 - Supports Timing mode 4. • Bit 5 - Supports Timing mode 5. • Bit 6-15 - Reserved.	16'h0

7.4.11. onfi_timing_modes_1 (0x0828)

Description: Device Timing modes supported by the connected ONFI device.

Table 7.54. onfi_timing_modes_1

Bits	SW	Name	Description	Reset
31:16	R	nv_ddr3_modes	This field contains value of bytes 160-161 from parameter page ('NV-DDR3 timing mode support'). Typically the values in the field should be interpreted as follows: • Bit 0 - Supports Timing modes 0-3. • Bit 1 - Supports Timing mode 4. • Bit 2 - Supports Timing mode 5. • Bit 3 - Supports Timing mode 6. • Bit 4 - Supports Timing mode 7. • Bit 5 - Supports Timing mode 8. • Bit 6 - Supports Timing mode 9. • Bit 7 - Supports Timing mode 10. • Bit 8-15 - Reserved.	16'h0
15:0	R	nv_ddr2_modes	This field contains concatenated values of bytes 142 and 162 of parameter page ('NV-DDR3 timing mode support'). Typically the values in the field should be interpreted as follows: • Bit 0 - Supports Timing mode 0. • Bit 1 - Supports Timing mode 1. • Bit 2 - Supports Timing mode 2. • Bit 3 - Supports Timing mode 3. • Bit 4 - Supports Timing mode 4. • Bit 5 - Supports Timing mode 5. • Bit 6 - Supports Timing mode 6. • Bit 7 - Supports Timing mode 7. • Bit 8 - Supports Timing mode 8. • Bit 9 - Supports Timing mode 9. • Bit 10 - Supports Timing mode 10. • Bit 11-15 - Reserved.	16'h0

7.4.12. onfi_iterlv_op_attr (0x082c)

Description: Interleaved(Multiplane) operation attributes of the connected ONFI device.

Table 7.55. onfi_iterlv_op_attr

Bits	SW	Name	Description	Reset
31:8	R	RSVD	Reserved	24'h0
7:0	R	iterlv_op	Value of this field reflects the value of byte 114 in parameter page ('Multi-plane operation attributes'). Typically the values in the field should be interpreted as follows: • Bit 0 - Overlapped/concurrent interleaving support. • Bit 1 - no block address restrictions. • Bit 2 - Program cache supported. • Bit 3 - Address restrictions for cache operations. • Bit 4 - Read cache supported. • Bit 5 - Lower bit XNOR block address restriction. • Bit 6-7 - Reserved.	8'h0

7.4.13. onfi_sync_opt_0 (0x0830)

Description: ONFI synchronous device parameters - part 1.

Table 7.56. onfi_sync_opt_0

Bits	SW	Name	Description	Reset
31:16	R	onfi_tccs_min	The minimum value in (ns) that is required between a change column command and the data transfer from the device (tccs). Value of this field is filled based on bytes 139-140 of parameter page.	16'hffff
15:8	R	RSVD	Reserved	8'h0
7:0	R	nvddr_supp_ft	NV-DDR / NV-DDR2 Features. Value of this field is a copy of byte 143 from parameter page ('NV-DDR / NV-DDR2 features'): • Bit 0 - tCAD value to use. • Bit 1 - typical capacitance value present. • Bit 2 - device supports CLK stopped data input. • Bit 3 - device requires Vpp enablement sequence. • Bit 4-7 - Reserved.	8'h0

7.4.14. onfi_sync_opt_1 (0x0834)

Description: ONFI synchronous device parameters - part 2.

Table 7.57. onfi_sync_opt_1

Bits	SW	Name	Description	Reset
31:24	R	warmup_cycles	NV-DDR2/3 warmup cycles: • Bit 3-0 - Data output warmup cycles support. • Bit 7-4 - Data input warmup cycles support.	8'h0
23:16	R	nvddr2_3_features	NV-DDR2/3 features: • Bit 0 - Supports self termination ODT. • Bit 1 - Supports matrix termination ODT. • Bit 2 - Supports ODT value of 30 Ohms. • Bit 3 - Supports differential signaling for RE_n. • Bit 4 - Supports differential signaling for DQS. • Bit 5 - External VREFQ required for >= 200 MT/s. • Bit 6-7 - Reserved.	8'h0
15:8	R	RSVD	Reserved	8'h0
7:0	R	adv_cmd_supp	ONFI-JEDEC JTG primary advanced command support: • Bit 0 - Supports ONFI-JEDEC JTG Random Data Out. • Bit 1 - Supports ONFI-JEDEC JTG Multi-plane Page Program. • Bit 2 - Supports ONFI-JEDEC JTG Multi-plane Copyback Program. • Bit 3 - Supports ONFI-JEDEC JTG Multi-plane Block Erase. • Bit 4-7 - Reserved.	8'h0

7.4.15. bch_cfg_0 (0x0838)

Description: BCH Engine identification register 0 - available correction strengths.

Table 7.58. bch_cfg_0

Bits	SW	Name	Description	Reset
31:24	R	bch_corr_3	BCH correction capability no. 3 (0 value means unavailable).	8'h10
23:16	R	bch_corr_2	BCH correction capability no. 2 (0 value means unavailable).	8'hc
15:8	R	bch_corr_1	BCH correction capability no. 1 (0 value means unavailable).	8'h8
7:0	R	bch_corr_0	BCH correction capability no. 0 (0 value means unavailable).	8'h4

7.4.16. bch_cfg_1 (0x083c)

Description: BCH Engine identification register 1 - available correction strengths.

Table 7.59. bch_cfg_1

Bits	SW	Name	Description	Reset
31:24	R	bch_corr_7	BCH correction capability no. 7 (0 value means unavailable).	8'h0
23:16	R	bch_corr_6	BCH correction capability no. 6 (0 value means unavailable).	8'h0
15:8	R	bch_corr_5	BCH correction capability no. 5 (0 value means unavailable).	8'h0
7:0	R	bch_corr_4	BCH correction capability no. 4 (0 value means unavailable).	8'h20

7.4.17. bch_cfg_2 (0x0840)

Description: BCH Engine identification register 2 - available sector sizes.

Table 7.60. bch_cfg_2

Bits	SW	Name	Description	Reset
31:16	R	bch_sect_1	BCH sector size (in bytes).	16'h0
15:0	R	bch_sect_0	BCH sector size (in bytes).	16'h200

7.4.18. bch_cfg_3 (0x0844)

Description: BCH Engine identification register 3 - additional information.

Table 7.61. bch_cfg_3

Bits	SW	Name	Description	Reset
31:24	R	bch_syndrome_factor	BCH syndrome factor.	8'h10
23:16	R	bch_metadata_size	Max metadata size (in bytes).	8'h28
15:8	R	bch_chien_factor	BCH paralleling factor - Chien.	8'h8
7:0	R	bch_brlk_factor	BCH paralleling factor - Berlekamp.	8'h8

7.5. Protect mechanism registers. (0x0900)

7.5.1. prot_ctrl_0 (0x0900)

Description: Control register for the protect mechanism.

Table 7.62. prot_ctrl_0

Bits	SW	Name	Description	Reset
31:16	R	RSVD	Reserved	16'h0
15:0	R/W	prot_en_0	Each bit of this field enable protection mechanism for corresponding target.	16'h0

7.5.2. prot_down_0 (0x0904)

Description: Register configure protected area address boundaries.

Table 7.63. prot_down_0

Bits	SW	Name	Description	Reset
31:24	R	RSVD	Reserved	8'h0
23:0	R/W	addr_down_0	Define lower row address limit of the protected area. It should be programed to a first address value inside the protected area.	24'h0

7.5.3. prot_up_0 (0x0908)

Description: Register configure protected area address boundaries.

Table 7.64. prot_up_0

Bits	SW	Name	Description	Reset
31:24	R	RSVD	Reserved	8'h0
23:0	R/W	addr_up_0	Define upper row address limit of the protected area. It should be programed to a value one over the last address in protected area.	24'h0

7.5.4. prot_ctrl_1 (0x0910)

Description: Control register for the protect mechanism.

Table 7.65. prot_ctrl_1

Bits	SW	Name	Description	Reset
31:16	R	RSVD	Reserved	16'h0
15:0	R/W	prot_en_1	Each bit of this field enable protection mechanism for corresponding target.	16'h0

7.5.5. prot_down_1 (0x0914)

Description: Register configure protected area address boundaries.

Table 7.66. prot_down_1

Bits	SW	Name	Description	Reset
31:24	R	RSVD	Reserved	8'h0
23:0	R/W	addr_down_1	Define lower row address limit of the protected area. It should be programed to a first address value inside the protected area.	24'h0

7.5.6. prot_up_1 (0x0918)

Description: Register configure protected area address boundaries.

Table 7.67. prot_up_1

Bits	SW	Name	Description	Reset
31:24	R	RSVD	Reserved	8'h0
23:0	R/W	addr_up_1	Define upper row address limit of the protected area. It should be programed to a value one over the last address in protected area.	24'h0

7.6. Minicontroller registers (0x1000)

7.6.1. wp_settings (0x1000)

Description: Write Protect

Table 7.68. wp_settings

Bits	SW	Name	Description	Reset
31:1	R	RSVD	Reserved	31'h0
0	R/W	WP	Write protect signal for all devices. Value of this register is directly routed to the WP output signal. The value can be changed only when NAND Flash interface is in idle state (minicontroller does not perform any sequence). Value of the WP# signal does not influence on the minicontroller sequence execution. The controller does not check the tWW timing - this must be ensured by the host.	1'h1

7.6.2. rbn_settings (0x1004)

Description: Ready/Busy# line status. Represents the value of the Ready/Busy# lines after two stage synchronizers due to the asynchronous nature of R/B#.

Table 7.69. rbn_settings

Bits	SW	Name	Description	Reset
31:2	R	RSVD	Reserved	30'h0
1:0	R	Rbn	RBn status	2'h0

7.6.3. common_settings (0x1008)

Description: Configuration of the Minicontroller.

Table 7.70. common_settings

Bits	SW	Name	Description	Reset
31:9	R	RSVD	Reserved	23'h0
8	R/W	device_16bit	16 bit device connected to the NAND Flash interface. 16bit devices may have sequences at which the data is transfered on LSB of the data bus. For such sequences this field should be set low.	1'h0

Bits	SW	Name	Description	Reset
7:0	R	RSVD	Reserved	8'h0

7.6.4. skip_bytes_conf (0x100c)

Description: Skip bytes settings.

Table 7.71. skip_bytes_conf

Bits	SW	Name	Description	Reset
31:16	R/W	marker	A 16bit value that will be written in the spare area skip bytes. In SDR 8-bit mode the LSB of this field is used only. In DDR modes all bits are used.	16'h0
15:8	R	RSVD	Reserved	8'h0
7:0	R/W	skip_bytes	Number of bytes to skip from offset of block. The bytes will be written with the value programmed in the marker register. This register could be potentially used to preserve the bad block marker in the spare area by marking it good. The default value is zero which means no bytes will be skipped. This value should be an even number.	8'h0

7.6.5. skip_bytes_offset (0x1010)

Description: Skip bytes offset settings.

Table 7.72. skip_bytes_offset

Bits	SW	Name	Description	Reset
31:24	R	RSVD	Reserved	8'h0
23:0	R/W	skip_bytes_offset	Offset after which the Minicontroller starts sending the dummy bytes (defined by marker) to the device. After skip bytes the Minicontroller continues to transfer the data. The offset is counted from the beggining of the data transfer.	24'h0

7.6.6. async_toggle_timings (0x101c)

Description: SDR timings configuration.

Table 7.73. async_toggle_timings

Bits	SW	Name	Description	Reset
31:29	R	RSVD	Reserved	3'h0
28:24	R/W	tRH	The number of clock cycles (nf_clk) the Minicontroller needs to de-assert RE# to meet the tREH (RE# high pulse width) time of the asynchrnonous(SDR) NAND device during command/address/data sequence.	5'h18
23:21	R	RSVD	Reserved	3'h0
20:16	R/W	tRP	The number of clock cycles (nf_clk) the Minicontroller needs to assert RE# to meet the tRP (RE# low pulse width) time	5'h18

Bits	SW	Name	Description	Reset
			of the asynchrnonous(SDR) NAND device during command/ address/data sequence.	
15:13	R	RSVD	Reserved	3'h0
12:8	R/W	tWH	The number of clock cycles (nf_clk) the Minicontroller needs to de-assert WE# to meet the tWH (WE# high pulse width) time of the async(SDR) NAND device during command/ address/data(in SDR only) sequence.	5'h18
7:5	R	RSVD	Reserved	3'h0
4:0	R/W	tWP	The number of clock cycles (nf_clk) the Minicontroller needs to assert WE# to meet the tWP (WE# low pulse width) time of the async NAND device during command/address/data(in SDR only) sequence.	5'h18

7.6.7. timings0 (0x1024)

Description: Global timings configuration - register 0.

Table 7.74. timings0

Bits	SW	Name	Description	Reset
31:24	R/W	tADL	Signifies the number of clock cycles that should be introduced between an address to a data input cycle. The timing value follows tADL. The number programmed in this register should be in terms of Minicontroller clock cycles (nf_clk) that would be required to satisfy the time.	8'hff
23:16	R/W	tCCS	Timing parameter for minimum change column setup time. The timing value follows tCCS. The number programmed in this register should be in terms of Minicontroller clock cycles (nf_clk) that would be required to satisfy the time.	8'hff
15:8	R/W	tWHR	Timing parameter between we high to re low. The timing value follows tWHR. The number programmed in this register should be in terms of Minicontroller clock cycles (nf_clk) that would be required to satisfy the time.	8'hff
7:0	R/W	tRHW	Timing parameter between re high to we low. The timing value follows tRHW. The number programmed in this register should be in terms of Minicontroller clock cycles (nf_clk) that would be required to satisfy the time.	8'hff

7.6.8. timings1 (0x1028)

Description: Global timings configuration - register 1.

Table 7.75. timings1

Bits	SW	Name	Description	Reset
31:24	R/W	tRHZ	Timing parameter between re high to re low for the next bank. The timing value follows tRHZ. The number programmed in	8'hff

Bits	SW	Name	Description	Reset
			this register should be in terms of Minicontroller clock cycles (nf_clk) that would be required to satisfy the time.	
23:16	R/W	tWB	Number of Minicontroller clock cycles (nf_clk) required for meeting the tWB time. The number programmed in this register should be in terms of Minicontroller clock cycles (nf_clk) that would be required to satisfy the time. This value must be increased by two due to the two-stage synchronizers implemented at the top level for dfi_rbn signal	8'hff
15:8	R/W	tCWAW	Signifies the number of Minicontroller clock cycles (nf_clk) that should be introduced between the command cycle of a random data input command to the address cycle of the random data input command. The timing value follows tCWAW.	8'hff
7:0	R/W	tVDLY	Signifies the number of Minicontroller clock cycles (nf_clk) that should be introduced after a volume select command before de-asserting the CE or before sending a command to the new volume. The timing value follows tVDLY.	8'hff

7.6.9. timings2 (0x102c)

Description: Global timings configuration - register 2.

Table 7.76. timings2

Bits	SW	Name	Description	Reset
31:26	R	RSVD	Reserved	6'h0
25:16	R/W	tFEAT	Signifies the number of Minicontroller clock cycles (nf_clk) that should be introduced after a Set Features command and Reset command (since this command change the work mode of the NAND Flash device). For most devices this register refers to device timing parameter tITC. For the devices which do not have this timing ie. does not have timing mode change feature the tWB timing should be defined in this field. It is valid only for Set Features run in PIO mode and Reset command for every controller mode.	10'h3ff
15:14	R	RSVD	Reserved	2'h0
13:8	R/W	CS_hold_time	Number of Minicontroller clock cycles (nf_clk) required for meeting chip select high time. This register refers to device timing parameter tCEH. Some legacy devices may have a tWHC (WE# high to CE# low) timing requirement. To meet this requirement the higher value from the tCEH/tWHC should be written to this field.	6'h3f
7:6	R	RSVD	Reserved	2'h0
5:0	R/W	CS_setup_time	Number of Minicontroller clock cycles (nf_clk) required for meeting chip select setup time. This register refers to device chip enable setup timing parameter tCS. Some devices may define more than one tCS timing parameters. Controller implements one timing for all sequences. It has to be programmed accordingly to the performed sequence.	6'h3f

7.7. Control Timing Block registers (0x2080)

7.7.1. phy_ctrl_reg (0x2080)

Description: This register handles the global control settings for the PHY. Please make sure all the reserved fields are set to 0.

Table 7.77. phy_ctrl_reg

Bits	SW	Name	Description	Reset
31:9	R	RSVD	Reserved	23'h0
8:4	R/W	phony_dqs_timing	The timing of assertion of phony DQS to the read data path. Value of this field determine the read data sampling point. For example to set sampling point at the end of the RE pulse (rising edge of the RE signal), if the RE pulse width is 4 clock cycles, the value of this field should be 3.	5'h18
3:0	R	RSVD	Reserved	4'h0

Appendix A. Special Function Register Map

Table A.1. Special Function Register Map

	+0x00	+0x04	+0x08	+0x0C
...	Command and Status registers.			
0x0000	cmd_reg0	cmd_reg1	cmd_reg2	cmd_reg3
0x0010	cmd_status_ptr	cmd_status	RSVD	RSVD
0x0020	RSVD	RSVD	RSVD	RSVD
0x0030	RSVD	RSVD	RSVD	RSVD
0x0040	RSVD	RSVD	RSVD	RSVD
0x0050	RSVD	RSVD	RSVD	RSVD
0x0060	RSVD	RSVD	RSVD	RSVD
0x0070	RSVD	RSVD	RSVD	RSVD
0x0080	RSVD	RSVD	RSVD	RSVD
0x0090	RSVD	RSVD	RSVD	RSVD
0x00a0	RSVD	RSVD	RSVD	RSVD
0x00b0	RSVD	RSVD	RSVD	RSVD
0x00c0	RSVD	RSVD	RSVD	RSVD
0x00d0	RSVD	RSVD	RSVD	RSVD
0x00e0	RSVD	RSVD	RSVD	RSVD
0x00f0	RSVD	RSVD	RSVD	RSVD
0x0100	RSVD	RSVD	RSVD	RSVD
0x0110	intr_status	intr_enable	ctrl_status	RSVD
0x0120	trd_status	RSVD	trd_error_intr_status	RSVD
0x0130	trd_error_intr_en	RSVD	trd_comp_intr_status	RSVD
0x0140	dma_target_error_l	dma_target_error_h	boot_status	trd_timeout_intr_status
0x0150	RSVD	trd_timeout_intr_en	RSVD	RSVD
...	Config Registers.			
0x0400	transfer_cfg_0	transfer_cfg_1	long_polling	short_polling
0x0410	rdst_ctrl_0	rdst_ctrl_1	lun_status_cmd	lun_interleaved_cmd
0x0420	lun_addr_offset	nf_dev_layout	ecc_config_0	ecc_config_1
0x0430	device_ctrl	multiplane_config	cache_config	dma_settings
0x0440	sdma_size	sdma_trd_num	time_out	sdma_addr0
0x0450	sdma_addr1	RSVD	RSVD	RSVD
0x0460	RSVD	RSVD	RSVD	RSVD
0x0470	RSVD	RSVD	RSVD	RSVD
0x0480	RSVD	RSVD	RSVD	RSVD

	+0x00	+0x04	+0x08	+0x0C
0x0490	RSVD	control_data_ctrl	RSVD	RSVD
...	Debug Interface registers.			
0x0740	dbg_ctrl	dbg_stat	dbg_data	RSVD
...	Controller and Device Parameters.			
0x0800	ctrl_version	ctrl_features_reg	manufacturer_id	nf_device_areas
0x0810	device_params_0	device_params_1	device_features	device_blocks_per_lun
0x0820	device_revision	onfi_timing_modes_0	onfi_timing_modes_1	onfi_iterlv_op_attr
0x0830	onfi_sync_opt_0	onfi_sync_opt_1	bch_cfg_0	bch_cfg_1
0x0840	bch_cfg_2	bch_cfg_3	RSVD	RSVD
...	Protect mechanism registers.			
0x0900	prot_ctrl_0	prot_down_0	prot_up_0	RSVD
0x0910	prot_ctrl_1	prot_down_1	prot_up_1	RSVD
...	Minicontroller registers			
0x1000	wp_settings	rbn_settings	common_settings	skip_bytes_conf
0x1010	skip_bytes_offset	RSVD	RSVD	async_toggle_timings
0x1020	RSVD	timings0	timings1	timings2
...	Control Timing Block registers			
0x2080	phy_ctrl_reg	RSVD	RSVD	RSVD

Appendix B. Document Revision History

Table B.1. Document Revision History

Revision	Modification	Date	Author
0.01	First Draft Release	06 Nov 2014	kszweda@cadence.com
0.02	Grammar review	28 Nov 2014	justyna@cadence.com
0.03	Removed section 'Special Commands'. The 'Set Features' command was moved to PIO commands description.	11 Nov 2014	kszweda@cadence.com
0.04	Updated the 'ctrl_features_reg' register.	12 Nov 2014	kswitala@cadence.com
0.05	Changed default value of selected burst to 0 and width of the burst_sel field (dma_settings register).	02 Jan 2015	kswitala@cadence.com
0.06	Updated section with Data DMA Master module (hardware specification).	02 Jan 2015	kswitala@cadence.com
0.07	Added controller's version register and updated ctrl_features register.	05 Jan 2015	kswitala@cadence.com
0.08	1) Added skip_bytes_offset register into the minictrl SFR address space 2) Erased page detection added 3) async/toggle timing reg - tWP/tWH/tRP/tRH description complemented and widen fields 3) minictrl top level figure added 4) rddata width corrected 5) underrun description added 6) new data transfer control description 7) minictrl FSM description added 8) dfi interfaces errors corrected 9) links to sequences added 10) phony_dqs/rd_del_sel default values upentry	16 Apr 2015	ppietron@cadence.com
0.09	1) warmup* minictrl SFR fields - description complemented; 2) tCR field added to toggle_timings_0 register; 3) CS_setup_time field width reduced in timings2 register	28 Apr 2015	ppietron@cadence.com
0.10	Section "Row Address Layout" was added to a second chapter. The block_address term was changed to the row_address	28 April 2015	kszweda@cadence.com

Revision	Modification	Date	Author
0.11	Set Features special sequence added to minicontroller description. Set Features figure corrected.	07 May 2015	ppietron@cadence.com
0.12	Toggle config register removed. Information about type of the input was added to the pin index table Table with clock frequencies was added Sync Functionality description was updated Column addressing in the controller was updated	29 May 2015	kszweda@cadence.com
0.13	Device discovery description updated in the User Guide part as well as in the modules specification part - PHY initialization step added DDR Mode of Operation description updated - step 8 and 9 swapped. Minicontroller sector size description complemented - zero values forbidden . tWB timing description updated.	11 June 2015	ppietron@cadence.com
0.14	The Flash_Ptr_Cont and the Mem_Ptr_Cont bits were added to the Command Flags field of a descriptor.	16 June 2015	kszweda@cadence.com
0.15	The ECC engine description was updated. The RowAC field from the nf_dev_layout was removed Information about access restriction to the mini controller and PHY was added to the Programming Specification Section. Minor updates to other sections were made	25 June 2015	kszweda@cadence.com
0.16	Note about nf_clk clock frequency for devices that starts working in DDR mode added.	22 July 2015	ppietron@cadence.com
0.17	tCDQSH timing added to the toggle_timings_1 register.	03 August 2015	ppietron@cadence.com
0.18	LUN Reset sequence added. Skip bytes description complemented.	10 September 2015	ppietron@cadence.com
0.19	Added restriction on skip bytes offset value.	23 September 2015	ppietron@cadence.com
0.20	Clarified that in case of disabling CRC checking during discovery process register values can be corrupted.	28 September 2015	kswitala@cadence.com
0.21	dfi_dqs_underrun field removed from dll_phy_ctrl (0x1034) register. DQS underrun was moved to the last operation status	09 October 2015	ppietron@cadence.com
0.22	added busy state flags for internal modules in ctrl_status register.	14 October 2015	kswitala@cadence.com

Revision	Modification	Date	Author
0.23	corrected bit width of mem_desc_addr port.	14 October 2015	kswitala@cadence.com
0.24	changed maximum supported burst size to 256. Updated inconsistent description of number of outstanding transactions.	14 October 2015	kswitala@cadence.com
0.25	changed width of burst_sel field in dma_settings register to 8 bits.	14 October 2015	kswitala@cadence.com
0.26	DDR to SDR mode change description added. Figures with Generic Sequences added.	16 October 2015	ppietron@cadence.com
0.27	The address width was changed to 32-bits The ID field width on the system bus was changes 20-bits The signal names were synchronized between RTL and documentation The address width for the context memory was corrected The sync feature greater flag description was updated The bus error flag was added to the last operation status The trd_err_n bit was removed from implementation The boot engine description was updated The SDR to DDR switching procedure was described The clock naming was fixed in the user guide The busy bits for the DMA, SDMA and Command Engine were added to the ctrl_status register Support for additional multi-plane read sequences were added	16 October 2015	kszweda@cadence.com
0.28	Data sequence description added to the generic mode chapter. Sequence numbers at which tWB_active is valid updated. tCWAW timing moved to Change Write Column sequence to be consistent with sequence figure.	27 October 2015	ppietron@cadence.com
0.29	Updated description of the warmup_cycles field in the onfi_sync_opt_1 register.	28 October 2015	kswitala@cadence.com
0.30	Updated description of the Thread Reset 1 feature. Updated description of device error detection mechanism.	29 October 2015	kszweda@cadence.com

Revision	Modification	Date	Author
0.31	Updated controller and BCH version registers	29 October 2015	kswitala@cadence.com
0.32	Jedec_supp for WRITE_CMD added as a command mnemonic modifier. DQS underrun description complemented in the status field descriptions.	30 October 2015	ppietron@cadence.com
0.33	Added information that bit rb_enable is ignored during multi lun operations. Added information that software should ignore flags after sending thread reset 1 command. Added note that in Generic Work Mode commands can be visible on NF interface after thread complete is raised. Clarified behavior of the controller for thread reset and timeout when cont_on_err is set.	10 November 2015	kswitala@cadence.com
0.34	Clarified using of the Slave DMA interface. Last operation errors generation mechanism description was updated. Register reset pin was added for the flash interface side. More details about register reset usage ware added. Timeout functionality description was updated. Timeout flag was added to the boot status regester.	14 November 2015	kszweda@cadence.com
0.35	Clarified initialization protocol section regarding values stored in transfer registers	15 November 2015	kswitala@cadence.com
0.36	Register access description was updated. Boot description updated.	20 November 2015	kszweda@cadence.com
0.37	The lun_col_cmd field was added to the lun_interleaved_cmd register. Typo in the last_wr_cmd field decription in the multiplane_config register was fixed. Addition allowed value for the mpl_rd_seq field in the multiplane_config register was added. Description of the mpl_wr_en field in the multiplane_config register was updated.	17 December 2015	kszweda@cadence.com
0.38	Read status enhanced figure updated.	30 December 2015	ppietron@cadence.com
0.39	Updated controller's block diagram.	18 January 2016	kswitala@cadence.com
0.40	Generic Command layout table update. Bit 15 (ce_hold) description added.	27 January 2016	kswitala@cadence.com
0.41	Small sector size restriction note added	09 March 2016	ppietron@cadence.com

Revision	Modification	Date	Author
0.42	Configuration-specific documentation	11 March 2016	kswitala@cadence.com
0.43	Added requirement to reset NF memory device in NV-DDR3 mode	11 March 2016	kswitala@cadence.com
0.44	Added advised value for short and long polling	18 March 2016	kswitala@cadence.com
0.45	The ZQ Calibration description was added.	20 March 2016	kszweda@cadence.com
0.46	The Error Index status field description was updated.	05 April 2016	kszweda@cadence.com
0.47	Moved the 'nf_device_areas' to 'Controller and Device Parameters' section.	15 April 2016	kswitala@cadence.com
0.48	Corrected value of metadata size in the bch_cfg_3 register.	27 April 2016	kswitala@cadence.com
0.49	Clarified informations specific for OCP configuration. Renamed nfc_clk to nf_clk on top level diagram. Removed unused ports for ONFI1-only configuration.	27 April 2016	kswitala@cadence.com
0.50	Added description for slave APB configuration.	07 July 2016	kswitala@cadence.com
1.0	Full release.	31 July 2016	kszweda@cadence.com
1.01	Corrected slave DMA pin description.	12 September 2016	kswitala@cadence.com
1.02	The sdma_paused feature was added The ZQ calibration control register gained immediate access right. The data integrity checking for the data path and chip memories was added The debug interface was added The Scrambler module was extracted from the ECC logic and instantiated as separate entity The status preserving logic inside the command engine was updated to use internal storage instead of the context memory The TX-FIFO feature was added	06 December 2016	kszweda@cadence.com
1.03	Added configurable decription of threads number. Possible options are 8 or 16. Added description of seed generator interface - it configurable feature. Added description of the new generic command: seed initialization command. Updated description of the context memory. Added minor updates.	01 January 2017	kszweda@cadence.com

Revision	Modification	Date	Author
1.04	ce_hold description complemented.	24 February 2017	ppietron@cadence.com
1.05	Marked register field unused in the generic work mode. Updated information about register access in the programing specification section. Fixed a few typo in pin names in the boot section. Updated description of the seed control logic	21 March 2017	kszweda@cadence.com
1.06	Updated BCH description generation in configuration with three buffers. Removed redundant description in configuration with data integrity disabled.	07 April 2017	kszweda@cadence.com
1.07	Added remap feature description Added extended command counter feature description Added status reporting per plane description Added 8 word to the descriptor table	28 April 2017	kszweda@cadence.com
1.08	Generic Sequence programmable number of address cycles descritpion added.	12 September 2017	ppietron@cadence.com