Technical Note

Software Device Drivers for Large Page Micron® NAND Flash Memory

Introduction

This technical note explains how to use Micron's large page NAND Flash memory software device drivers, which are the low-level drivers (LLDs) that manage the hardware functionality of the large page NAND Flash memory devices.

This document describes the operation of the devices and provides a basis for understanding and modifying the accompanying source code. The source code driver is written to be as platform independent as possible, and requires minimal changes to compile and run. The technical note also explains how to modify the source code for a target system. The source code is available at www.micron.com or from your Micron distributor. The c2188.c and c2188.h files contain libraries for accessing the devices.

This technical note does not replace the data sheet for any large page NAND device. It refers to the data sheet throughout, and it is necessary to have a copy of it to follow some explanations. Refer to the data sheets for the corresponding device part numbers and densities (see "References" on page 20).

Contact your Micron representative for more information about Micron's large page NAND Flash memory software device drivers.

Large Page NAND Overview

The NAND Flash 2112 byte/1056 word page family of nonvolatile Flash memory devices use NAND cell technology. The devices range from 1Gb to 8Gb, and operate either from a 1.8V or a 3V voltage supply. The size of a page is either 2112 bytes (2048 + 64 spare) or 1056 words (1024 + 32 spare), depending on whether the device has a x8 or x16 bus width.

The address lines are multiplexed with the data input/output signals on a multiplexed x8 or x16 input/output bus. This interface reduces the pin count and makes it possible to migrate to other densities without changing the footprint.

The devices have the following hardware and software security features:

- A WRITE protect pin ($\overline{WP}$) is available to provide hardware protection against PROGRAM and ERASE operations.
- A block locking scheme is available to provide user code and/or data protection.

The devices feature an open-drain ready/busy output that can be used to identify whether the PROGRAM/ERASE/READ (P/E/R) controller is currently active. The use of an open-drain output makes it possible for the ready/busy pins from several memory devices to be connected to a single pull-up resistor.

A COPY BACK PROGRAM command optimizes the management of defective blocks. When a PAGE PROGRAM operation fails, the data can be programmed in another page without resending the data to be programmed.

Cache program and cache read features improve the PROGRAM and READ throughputs for large files. During cache programming, the device loads the data in a cache register while the previous data is transferred to the page buffer and programmed into the memory array. During cache reading, the device loads the data in a cache register while the previous data is transferred to the I/O buffers to be read.

All devices have the “Chip Enable Don’t Care” feature, which enables code to be directly downloaded by a microcontroller, as chip enable transitions during the latency time do not stop the READ operation.

Micron’s 2112 byte/1056 word page NAND Flash memory devices can be delivered with an optional unique identifier (serial number) that makes it possible for each device to be uniquely identified. The unique identifier option is subject to a non-disclosure agreement (NDA), and as a result is not described in the data sheet.

A lock/unlock enable input (PRL) is used to enable and disable the lock mechanism. When PRL is HIGH (V_{IH}), the device is in block lock mode and a BLOCK UNLOCK command is required before programming or erasing a block.

The lock/unlock enable (PRL) pins can be wired in two ways:

- Common wired so that all memory devices are automatically locked/unlocked
- Wired separately so that each memory device can be individually locked/unlocked

Note: The PRL pin may not be supported. Please refer to the NAND data sheet for a specific device to check if this functionality is supported.

In addition, Micron’s M68A/69A/60A NAND Flash memory devices support internal error correction code (ECC). For a full description, please refer to the “Internal ECC and Spare Area Mapping for ECC” section of the device’s data sheet.

Bus Operations

The standard bus operations that control Micron large page NAND Flash memory devices are:

- COMMAND INPUT
- ADDRESS INPUT
- DATA INPUT
- DATA OUTPUT
- WRITE PROTECT
- STANDBY

Table 1: Bus Operations

Bus Operation	Description
COMMAND INPUT	Sends commands to the memory device.
ADDRESS INPUT	Inputs the memory addresses. Four bus cycles are required to input the addresses for 1Gb devices, whereas five bus cycles are required for 2Gb, 4Gb, and 8Gb devices. Refer to the large page NAND Flash memory data sheet for a detailed description of addresses and address input cycles on x8 and x16 devices.
DATA INPUT	Inputs the data to be programmed.
DATA OUTPUT	Reads data from the memory array, status register content, block lock status, electronic signature, or unique identifier.

Table 1: Bus Operations (Continued)

Bus Operation	Description
WRITE PROTECT	Protects the memory against PROGRAM or ERASE operations. When the WRITE PROTECT signal is LOW, the NAND memory device does not accept PROGRAM or ERASE operations. As a result, the contents of the memory array cannot be altered.
STANDBY	When CHIP ENABLE is HIGH, the memory enters standby mode, the device is deselected, outputs are disabled, and power consumption is reduced.

Device Operations

All bus WRITE operations to the device are interpreted by the command interface. The following commands are available on large page Micron NAND Flash memory devices:

- PAGE READ
- RANDOM DATA OUTPUT
- CACHE READ
- PAGE PROGRAM
- MULTIPLANE PAGEPROGRAM
- RANDOM DATA INPUT
- COPYBACK PROGRAM
- MULTI-PLANE COPYBACK PROGRAM
- CACHE PROGRAM
- BLOCK ERASE
- MULTIPLANE BLOCK ERASE
- RESET
- READ STATUS REGISTER
- READ EDC STATUS REGISTER
- READ ELECTRONIC SIGNATURE
- BLOCKS LOCK
- BLOCK UNLOCK
- BLOCKS LOCK-DOWN
- BLOCK LOCK STATUS
- ENABLE INTERNAL ECC
- DISABLE INTERNAL ECC
- VERIFY INTERNAL ECC

Note: Refer to the data sheet for the NAND device for a detailed description of the device operations.

Table 2: Device Operations

Device Operation	Description
PAGE READ	After the first random read access, the page data (2112 bytes or 1056 words) is transferred to the page. Once the transfer is complete, the data can then be read out sequentially (from the selected column address to the last column address).

Table 2: Device Operations (Continued)

Device Operation	Description
RANDOM DATA OUTPUT	The device can output random data in a page (instead of consecutive sequential data) by issuing a RANDOM DATA OUTPUT command. This command can be used to skip some data during a sequential data output.
CACHE READ	Improves the READ throughput by reading data using the cache register. When the user starts to read one page, the device automatically loads the next page into the cache register.
PAGE PROGRAM	This is the standard operation to program data to the memory array. The memory array is programmed by page. However, partial page programming is allowed where any number of bytes (1 to 2112) or words (1 to 1056) can be programmed.
MULTIPLANE PAGEPROGRAM	Issues a MULTIPLANE PAGEPROGRAM command, which enables the programming of two pages in parallel, one in each plane. It can perform multi-plane page program with random data input. A MULTIPLANE PAGEPROGRAM operation requires the following two steps: 1. Serially load up to two pages of data (4224 bytes) into the data buffer. 2. Parallel programming of both pages starts after the issue of the Page Confirm command. If the MULTIPLANE PAGEPROGRAM fails, an error is signaled on bit SR0 of the status register. To determine which page of the two planes failed, the READ STATUS ENHANCED command must be issued twice, once for each plane. Refer to the NAND device's data sheet for a description of the multi-plane operation and the restrictions related to the multi-plane page program sequence.
RANDOM DATA INPUT	During a SEQUENTIAL INPUT operation, the next sequential address to be programmed can be replaced by a random address by issuing a RANDOM DATA INPUT command.
COPY BACK PROGRAM	Copies the data stored in one page and reprograms it in another page. The operation is particularly useful when a portion of a block is updated and the rest of the block must be copied to the newly assigned block.
MULTI-PLANE COPYBACK PROGRAM	This function issues a MULTI-PLANE COPYBACK PROGRAM command as explained in the data sheet of the 2112 byte/1056 word page family. The function permits random data input. In this case, the data inserted are all 1. The multi-plane copy back program requires the same steps as the multi-plane page program command. If the MULTI-PLANE COPYBACK PROGRAM fails, an error is signaled on bit SR0 of the status register. To know which page of the two planes failed, the READ STATUS ENHANCED command must be executed twice, once for each plane. Refer to the NAND device's data sheet for a description of the multi-plane operation and specific information related to the multi-plane copy back program sequence.
CACHE PROGRAM	Improves the programming throughput by programming data using the cache register. The CACHE PROGRAM operation can only be used within one block. The cache register allows new data to be input while the previous data that was transferred to the page buffer is programmed into the memory array.

Table 2: Device Operations (Continued)

Device Operation	Description
BLOCK ERASE	ERASE operations are performed one block at a time. An ERASE operation sets all bits in the addressed block to 1. All previous data in the block is lost.
MULTI-PLANE BLOCK ERASE	Issues a MULTI-PLANE BLOCK ERASE command, which erases two blocks in parallel, one in each plane. To determine which page of the two planes failed, the READ STATUS ENHANCED command must be executed twice, once for each plane. Refer to the NAND device's data sheet for a description of the multi-plane operation and specific information related to the block erase sequence.
RESET	Resets the command interface and status register. If the RESET command is issued during any operation, the operation will be aborted. If it was a PROGRAM or ERASE operation that was aborted, the contents of the memory locations being modified will no longer be valid as the data will be partially programmed or erased.
READ STATUS REGISTER	The device contains a status register, which provides information on the current or previous PROGRAM or ERASE operation. The various bits in the status register convey information and errors on the operation. The status register is read by issuing the READ STATUS REGISTER.
READ EDC STATUS REGISTER	This function is used after a COPYBACK or MULTI-PLANE COPYBACK to check if there is an EDC error inside the page.
READ ELECTRONIC SIGNATURE	Enables device information, such as manufacturer code and device code, to be read. Refer to the NAND device's data sheet for a description of electronic signature values.
BLOCKS LOCK	BLOCKS LOCK is a data protection command that enables all blocks to be locked simultaneously. It is can be issued only if the PRL signal is HIGH (lock/unlock enabled). Refer to the NAND device's data sheet for a detailed description of the block protection scheme.
BLOCK UNLOCK	BLOCK UNLOCK is a data protection command used to unlock a sequence of consecutive locked blocks to enable PROGRAM or ERASE operations. It can be issued only if the PRL signal is HIGH (lock/unlock enabled). Refer to the NAND device's data sheet for a detailed description of the block protection scheme.
BLOCKS LOCK-DOWN	BLOCKS LOCK-DOWN is a data protection command that provides an additional level of protection. When locked-down, a block cannot be unlocked by a software command. Locked-down blocks can only be unlocked by setting the WRITE PROTECT signal to LOW for a minimum of 100ns. This command can be issued only if the PRL signal is HIGH (lock/unlock enabled). Refer to the NAND device's data sheet for a detailed description of the block protection scheme.
BLOCK LOCK STATUS	Checks the block lock status of each block. It can be issued only if the PRL signal is HIGH (lock/unlock enabled). Refer to the NAND device's data sheet for a detailed description of the block protection scheme.
ECC FUNCTIONS	Enables, disables, and verifies the internal ECC status.

EDC Status Register

The devices contain an EDC status register, which provides information on errors that occur during the read cycles of the COPYBACK and MULTI-PLANE COPYBACK operations. When a MULTI-PLANE COPYBACK program occurs, it is not possible to distinguish which of the two READ operations caused the error.

Note: Refer to the NAND device's data sheet for a description of the specific EDC status register bits.

Status Register

The status register provides information on the current or previous PROGRAM or ERASE operation. The various bits in the status register convey information and errors on the operation.

The status register is read by issuing the READ STATUS REGISTER command. The status register information is present on the output data bus (I/O0–I/O7). After the READ STATUS REGISTER command has been issued, the device remains in read status register mode until another command is issued. If a READ STATUS REGISTER command is issued during a random read cycle, a new READ command must be issued to continue with a PAGE READ operation.

Note: Refer to the NAND device's data sheet for a description of the status register bits.

Detailed Example

The Commands table provided in the large page NAND Flash memory data sheet describes the WRITE operation sequences that are recognized as valid commands by the PROGRAM/ERASE controller.

For example, with a NAND01G-B device configured in 8-bit bus width, programming page 19h of block 20h requires the following C sequence to be issued:

1. The first bus write cycle sets up the PAGE PROGRAM command (command code 80h). Data is input sequentially by default. This is done by sending the following C command to the memory:

```
NAND_CommandInput((NMX_uint8)0x80); /* command code */
```

2. Four or five bus cycles are required to input the program address. A four-cycle sequence must be issued to program page 19h of block 20h (Table 3).

```
NAND_AddressInput (0x00);     /* first address byte */
NAND_AddressInput (0x00);     /* second address byte */
NAND_AddressInput (0x19);     /* third address byte */
NAND_AddressInput (0x08);     /* fourth address byte */
```

3. Data is loaded into the data registers:

```
for ( j=0; j<PageSize; j++) {
NAND_DataInput(data[I]);
}
```

4. One bus cycle is required to issue the PAGE PROGRAM CONFIRM command (command code 10h) to start the P/E/R controller. The P/E/R starts only if the data has been loaded in step 3.

```
NAND_CommandInput((ubyte)0x10);
```

5. The P/E/R controller programs the data into the array. Once the PAGE PROGRAM operation has started, the status register can be read using the READ STATUS REGISTER command. During PROGRAM operations, the status register only flags errors for bits set to 1 that have not been successfully programmed to 0. During the PROGRAM operation, only the READ STATUS REGISTER and RESET commands are accepted; all other commands are ignored. Once the PROGRAM operation is complete, the P/E/R controller bit SR6 is set to 1 and the READY/BUSY signal goes HIGH. The device remains in read status register mode until another valid command is written to the command interface.

Table 3: Page and Block Address Setting Sequence

Bus Cycle	I/O7	I/O6	I/O5	I/O4	I/O3	I/O2	I/O1	I/O0
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	1	1	0	0	1
4	0	0	0	0	1	0	0	0

Notes:
1. Grey cells indicate the column address.
2. Green cells indicate the addresses that must be set to V_{IL}.
3. Blue cells indicate the block address.
4. White cells represent the page address.

Software Driver

A low-level driver (LLD) is software that provides a first abstraction layer of the hardware to free the upper software layer from hardware management.

The large page NAND software drivers are easily portable on different hardware platforms. They are based on two layers:

- A *hardware-dependent layer* that uses basic functions to manage NAND memory control signals.
- A *hardware-independent layer* that uses command functions to control device operations such as BLOCK ERASE, PAGE PROGRAM, and READ.

C Library Functions

Basic Data Types

The data types used by the software drivers are described in Table 4.

Table 4: Data Types

typedef unsigned char	NMX_uint8	8 bits unsigned
typedef signed char	NMX_sint8	8 bits signed
typedef unsigned short	NMX_uint16	16 bits unsigned
typedef signed short	NMX_sint16	16 bits signed
typedef unsigned int	NMX_uint32	32 bits unsigned
typedef signed int	NMX_sint32	32 bits signed

Return Codes

The codes that can be returned by the software driver's C functions are:

Table 5: Return Codes

typedef NMX_uint8 NAND_Ret		
#define NAND_PASS	0x00	The operation on the NAND device completed successfully.
#define NAND_FAIL	0x01	The operation on the NAND device failed.
#define NAND_FLASH_SIZE_OVERFLOW	0x02	The address is not within the device.
#define NAND_PAGE_OVERFLOW	0x04	Attempt to access more than one page.
#define NAND_WRONG_ADDRESS	0x08	The address is wrong.
#define NAND_DIFFERENT_PAGES	0x10	The page is different in the data input command.
#define NAND_WRITE_PROTECTED	0x80	The device is write protected.
#define NAND_UNLOCKED_BLOCK	0x09	Block unlocked.
#define NAND_LOCKED_BLOCK	0x06	Block locked.
#define NAND_LOCK_DOWN	0x05	Block lock-down.
#define CACHE_READ_NOT_POSSIBLE	0x07	Required cache read data output with cache read not pending.
NAND_FIRSTPLANE_FAIL	0x11	The operation failed on the first plane.
NAND_SECONDPLANE_FAIL	0x13	The operation failed on the second plane.
NAND_BOTHPLANE_FAIL	0X12	The operation failed on both planes.
NAND_EDC_OK	0x20	No EDC error after copyback or multi-plane copyback.
NAND_EDC_INVALID	0x21	EDC is invalid and cannot be computed.
NAND_EDC_ERROR	0x22	EDC error after copyback or multi-plane copyback.
NAND_ECC_ENABLED	0x23	Internal ECC is enabled.
NAND_ECC_DISABLED	0x24	Internal ECC is disabled.

Hardware-Dependent Layer functions

The basic functions of the hardware-dependent layer are described in Table 6:

Table 6: Basic Functions

Return Value	Function Name	Parameter		Function Description
		Name	Description	
void	**NAND_SetWriteProtect** (void);	–	–	Sets the WRITE PROTECT signal ($\overline{\text{WP}}$) to LOW.
void	**NAND_UnsetWriteProtect** (void);	–	–	Sets the WRITE PROTECT signal ($\overline{\text{WP}}$) to HIGH.
void	**NAND_WaitTime** (NMX_uint8 nanoseconds);	nanoseconds	Time to wait.	Waits a period of time equal to the value of the nanoseconds parameter.
void	**NAND_Open** (void);	–	–	Programs the required settings before issuing a NAND operation.
void	**NAND_CommandInput** (NMX_uint8 ubCommand);	ubCommand	Command to be issued to the NAND device.	Command latch.
void	**NAND_AddressInput** (NMX_uint8 ubAddress);	ubAddress	Address to be issued to the NAND device.	Address latch.
void	**NAND_DataInput** (dataWidth ubData);	ubData	Data to be written.	Input data latch.
dataWidth[1]	**NAND_DataOutput** (void);	ubAddress	Address to be read from.	Output data latch.

Table 6: Basic Functions (Continued)

Return Value	Function Name	Parameter Name	Parameter Description	Function Description
void	**NAND_Close** (void);	–	–	Programs the required settings after issuing a NAND operation.

Notes: 1. The dataWidth format is either NMX_uint8 for x8 bus width or NMX_uint16 for x16 bus width.

Hardware-Independent Layer Command Functions

The NAND_Ret function is used to show whether the operation was successful. When its value is 0, the operation was successful. When its value is 1, the operation failed.

Table 7: Command Functions

Return Value	Function Name	Parameter	Parameter Description	Function Description
NAND_Ret	**NAND_BlockErase** (NMX_uint32 udAddress);	udAddress	Address of the block to be erased.	Performs a BLOCK ERASE operation.
NAND_Ret	**NAND_CacheProgram** (udword udPageAddresses[ ], NMX_uint8 piecesNumber, dataWidth **Buffers, NMX_uint32 udLength[ ], ubyte*errorPage);[1]	udPageAddresses	Addresses of the pages to be programmed.	Performs a CACHE PROGRAM operation.
		PiecesNumber	Number of data pieces to write.	
		Buffers	Buffer of data to write.	
		udLength	Size of each piece of data to write.	
		errorPage	Number of the page where (if applicable) the cache program failed.	
NAND_Ret	**NAND_CopyBack** (NMX_uint32 udSourceAddr, NMX_uint32 udDestinationAddr, NMX_uint16 *offsetInPage, NMX_uint16 *chunkSizes, NMX_uint16 numOfChunks, dataWidth **Buffers);[1]	udSourceAddr	Address of the source page to copy.	Performs a COPYBACK operation.
		udDestinationAddr	Address of the destination page.	
		OffsetInPage	Starting offsets of the pieces of data to rewrite.	
		ChunkSizes	Size of each piece of data to rewrite.	
		numOfChunks	Number of data pieces to rewrite.	
		Buffers	Buffers of data pieces to write.	

Table 7: Command Functions (Continued)

Return Value	Function Name	Parameter	Parameter Description	Function Description
NAND_Ret	**NAND_PageRead** (NMX_uint32 *udAddresses, dataWidth *Buffer, NMX_uint16 numOfChunks, NMX_uint32 *udLength);[1]	udAddresses	Addresses to read from in the Page.	Performs a PAGE READ operation.
		Buffer	Destination buffer to store the read data.	
		numOfChunks	Number of addresses contained in udAddresses.	
		udLength	Lengths of the data pieces to be read.	
NAND_Ret	**NAND_PageProgram** (NMX_uint32 *udAddresses, dataWidth **Buffer, NMX_uint8 numOfChunks, NMX_uint32 *udLength);[1]	udAddresses	Addresses to be programmed in the page.	Performs a PAGE PROGRAM operation.
		Buffer	Source buffers containing the data to be programmed.	
		numOfChunks	Number of addresses contained in udAddresses.	
		udLength	Length of the data pieces to be programmed.	
void	**NAND_ReadElectronicSignature** (dataWidth * Buffer);[1]	Buffer	Buffer to store the read data.	Reads the electronic signature.
void	NAND_Reset (void);	–	–	Resets the NAND device.
NAND_Ret	**NAND_SpareProgram** (NMX_uint32 udAddress, dataWidth *Buffer, NMX_uint32 udLength);[1]	udAddress	Address in the page to be programmed.	Programs in the spare area of the NAND Flash memory array.
		Buffer	Source buffer containing the data to be programmed.	
		udLength	Length of the data piece to be programmed.	
NAND_Ret	**NAND_SpareRead** (NMX_uint32 udAddress, dataWidth *Buffer, NMX_uint32 udLength);[1]	udAddress	Address in the page to read from.	Reads from the spare area of the NAND Flash memory array.
		Buffer	Destination buffer to store the data.	
		udLength	Length of the data pieces to be read.	
NAND_Ret	**NAND_CacheRead** (NMX_uint32 udAddress, dataWidth *Buffer, NMX_uint32 length);[1]	udAddress	Addresses of the page where the cache read starts.	Performs a CACHE READ operation.
		Buffer	Destination buffer to store the data read in the first page.	
		length	Length of the data to be read.	
void	**NAND_Terminate_CacheRead** void)	–	–	Exits from cache read mode.

Table 7: Command Functions (Continued)

Return Value	Function Name	Parameter	Parameter Description	Function Description
NAND_Ret	**NAND_CacheReadDataOutput** (dataWidth *Buffer, NMX_uint32 length)[2]	Buffer	Destination buffer where data is to be stored.	Reads the following pages from the buffer.
		length	Length of data to be read.	
void	**NAND_Lock** (void);[2]	–	–	Issues a BLOCKS LOCK command.
void	**NAND_LockDown** (void);[2]	–	–	Issues a BLOCKS LOCK-DOWN command.
NAND_Ret	**NAND_UnLock** (NMX_uint32 startBlock, NMX_uint32 endBlock)**;**[2]	startBlock	Address of the first block to be unlocked.	Issues an UNLOCK command.
		EndBlock	Address of the last block to be unlocked.	
NAND_Ret	**NAND_UnLockDown** (void);[2]	–	–	UnLocks locked-down blocks.
NAND_Ret	**NAND_ReadBlockLockStatus** (udword address);[2]	address	Address of the selected block.	Issues a READ BLOCK LOCK STATUS command.
ubyte	**NAND_ReadStatusRegister** (void);	–	–	Reads the status register.
NAND_Ret	**NAND_MultiplanePageProgram** (NMX_uint32 FPudAddresses[ ], dataWidth **FPBuffer, NMX_uint8 FPnumOfChunks, NMX_uint16 *FPudlength, NMX_uint32 *SPudAddresses, dataWidth**SPBuffer, NMX_uint8 SPnumOfChunks, NMX_uint16 *SPudlength);	FPudAddresses	Addresses of the first plane to be programmed.	Performs multi-plane page programming with random data input.
		FPBuffer	Source buffer with the first plane data to be programmed.	
		FPnumOfChunks	Numbers of chunks to program in the first plane.	
		FPudlength	Length of data to be programmed to the first plane.	
		SPudAddresses	Addresses of the second plane to be programmed.	
		SPBuffer	Source buffer with the second plane data to be programmed.	
		SPnumOfChunks	Numbers of chunks to program in the second plane.	
		SPudlength	Length of data to be programmed to the first plane.	

Table 7: Command Functions (Continued)

Return Value	Function Name	Parameter	Parameter Description	Function Description
NAND_Ret	**NAND_CopyBack** (NMX_uint32 FPudSourceAddr, NMX_uint32 FPudDestinationAddr, NMX_uint16 *FPoffsetInPage, NMX_uint16 *FPchunkSizes, NMX_uint16 FPnumOfChunks, dataWidth** FPBuffers, NMX_uint32 SPudSourceAddr, NMX_uint32 SPudDestinationAddr, NMX_uint16 *SPoffsetInPage, NMX_uint16 *SPchunkSizes, NMX_uint16 SPnumOfChunks, dataWidth** SPBuffers)	FPudSourceAddr	Address of the source plane to be programmed.	The function performs the MULTI-PLANE COPYBACK function.
		FPudDestinationAddr	Address of the source page to be programmed.	
		FPoffsetInPage	The starting offset of the first page.	
		FPchunkSizes	Length of source data to be programmed.	
		FPnumOfChunks	Numbers of chunks of the source plane to be programmed.	
		FPBuffers	Source buffer with the second plane data to be programmed.	
		SPudSourceAddr	Address of the source plane to be programmed.	
		SPudDestinationAddr	Address of the destination page to be programmed.	
		SPoffsetInPage	The starting offset of the second plane.	
		SPchunkSizes	Length of source data of second plane to be programmed.	
		SPnumOfChunks	Numbers of chunks of the source plane of the second plane to be programmed.	
		SPBuffers	Source buffer with the second plane data to be programmed.	
NAND_Ret	**NAND_MultiplaneBlockErase** NMX_uint32 FPudAddress, NMX_uint32 SPudAddress);	FPudAddresses	Addresses of the first plane to be erased.	The function performs a multi-plane erase.
		SPudAddress	Addresses of the second plane to be erased.	
NMX_uint8	**NAND_ReadEDCStatusRegister**();	void	–	The function is used to check if an EDC error occurred.
void	NAND_EnableECC	void	–	The function is used to enable internal ECC.
void	NAND_DisableECC	void	–	The function is used to disable internal ECC.

Table 7: Command Functions (Continued)

Return Value	Function Name	Parameter	Parameter Description	Function Description
NAND_Ret	NAND_VerifyECC	void		The function is used to check the internal ECC state.

Notes:
1. The dataWidth is either NMX_uint8 for x8 bus width or NMX_uint16 for x16 bus width.
2. Data protection functions are valid only if the PRL signal is HIGH.

Using the Driver

Choosing the Device

Before using the software on the target device, set the device specific #define section located in the c2188.h file.

- The software drivers support all 2112 byte/1056 word page NAND Flash memory devices (see "References" on page 20). All large page NAND Flash memory devices have the same number of pages per block, which is specified by:

```
#define NUM_PAGE_BLOCK
```

- The NAND Flash memory device is defined using the appropriate #define statement (see Table 8 for a definition of the constants):

```
#define NAND01GRW3B
#define NAND01GRW4B
#define NAND02GRW3B
#define NAND02GRW4B
#define NAND04GRW3B
#define NAND04GRW4B
#define NAND08GRW3B
#define NAND08GRW4B
#define MT29F1G08ABADA
```

The target device is automatically defined by a set of constants (Table 8 and Table 9).

Table 8: Device Constants Definition

Constant Name	Description
ECC_ON_DIE	The device supports the internal ECC features.
FLASH_SIZE	NAND size (without spare area).
FLASH_WIDTH	Data width of the NAND Flash memory.
MULTIPLANE	The device supports the multi-plane features.
NUM_BLOCKS	Number of blocks.
PAGE_DATA_SIZE	Size of data area of a page.
PAGE_SIZE	Size of a page.
PAGE_SPARE_SIZE	Size of spare area of a page.
SHIFT_A8	Constant used to calculate the address.

Table 9: Device Constants Values

Device	FLASH_ WIDTH	FLASH_ SIZE	PAGE_ SIZE	PAGE_DATA_ SIZE	PAGE_SPARE _SIZE	NUM_ BLOCKS	SHIFT_A8
1Gb	8 bits	0x8000000 Bytes	2112 Bytes	2048 Bytes	64 Bytes	1024	1
	16 bits	0x4000000 Words	1056 Words	1024 Words	32 Words	1024	0
2Gb	8 bits	0x10000000 Bytes	2112 Bytes	2048 Bytes	64 Bytes	2048	1
	16 bits	0x8000000 Words	1056 Words	1024 Words	32 Words	2048	0
4Gb	8 bits	0x20000000 Bytes	2112 Bytes	2048 Bytes	64 Bytes	4096	1
	16 bits	0x10000000 Words	1056 Words	1024 Words	32 Words	4096	0
8Gb	8 bits	0x40000000 Bytes	2112 Bytes	2048 Bytes	64 Bytes	8192	1
	16 bits	0x20000000 Words	1056 Words	1024 Words	32 Words	8192	0

Enabling Direct Memory Access (DMA)

Enabling direct memory access (DMA) can increase the performance of applications, the READ/WRITE operation from/to the NAND device, and the speed of the data transfer between the NAND page buffer and the RAM.

DMA moves a block of adjacent data from one memory to the other.

The following registers must be set before using the DMA:

- **SRC DMA**: This register contains the start address of the source block.
- **DST DMA**: This register contains the start address of the destination block.
- **CNT DMA**: This register contains the number of memory units of a block.
- **CON DMA**: This is the configuration register.

Set the following #define statement to enable the DMA engine:

```
#define DMA_ENABLE
```

According to the target hardware application, the code to manage the DMA engine must be inserted at every occurrence of the #define. The following example is for a platform based on an ARM7TDMI core:

```
#ifdef DMA_ENABLE
do
i=*(volatile udword*) (GDMACON0);
while ( (i&0x2) != 0);
*((volatile unsigned int*) (GDMASRC0)) = Base_Address;
*((volatile unsigned int*) (GDMADST0)) = (udword)Buffer;
*((volatile unsigned int*) (GDMACNT0)) = udLength[0];
*(volatile unsigned int*) (GDMACON0) = 0x0081;
udIndex+=udLength[0];
#endif
/*read data*/
#ifndef DMA_ENABLE
for(i=0;i<udLength[0];i++)
{
```

```
Buffer[udIndex++] = NAND_DataOutput();
}
#endif
```

Addressing Blocks and Pages

When a READ or PROGRAM operation must be performed, the software device driver receives as input parameters the address (Address) and the number (Length) of bytes/words to read/write from/to the memory.

For a large page NAND memory device configured in 8-bit bus width:

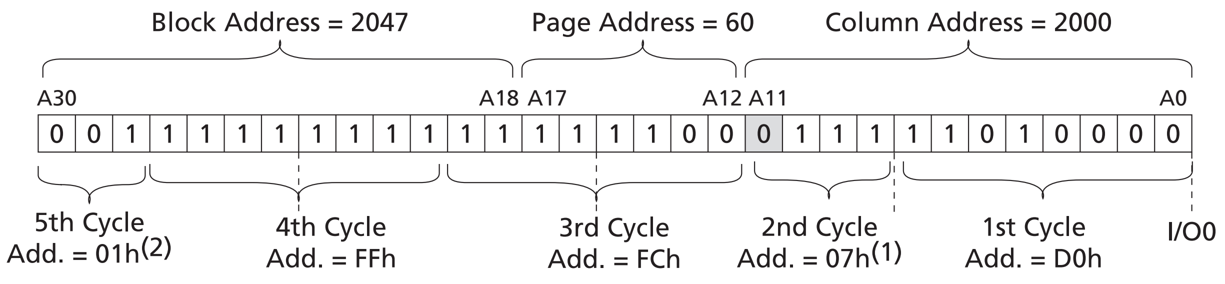
- Address = (Block Address ∗ 2048 ∗ 64) + (Page Address∗ 2048) + Column Address
  - If Column Address < 2048: The main area is addressed.
  - If 0 < Column Address < 64 and length are set accordingly: The spare area is addressed.
- Length = x Bytes. The user must insert the appropriate #define statement before issuing the command:
  - #define MAIN_LENGTH(x) for the main area
  - #define SPARE_LENGTH(x) for the spare area

For a large page NAND device, configured in 16-bit bus width:

- Address = (Block Address ∗ 1024 ∗ 64) + (Page Address∗ 1024) + Column Address
  - If Column Address < 1024: The main area is addressed.
  - If 0 < Column Address < 32 and length are set accordingly: The spare area is addressed.
- Length = x Words. The user must insert the appropriate #define statement before issuing the command:
  - #define MAIN_LENGTH(x) for the main area
  - #define SPARE_LENGTH(x) for the spare area

Note: When issuing functions specific to the spare area, such as NAND_SpareProgram or NAND_SpareRead, the length parameter (udLength) must be set to the number of bytes/words to be read or written. The #define SPARE_LENGTH(x) statement must not be inserted.

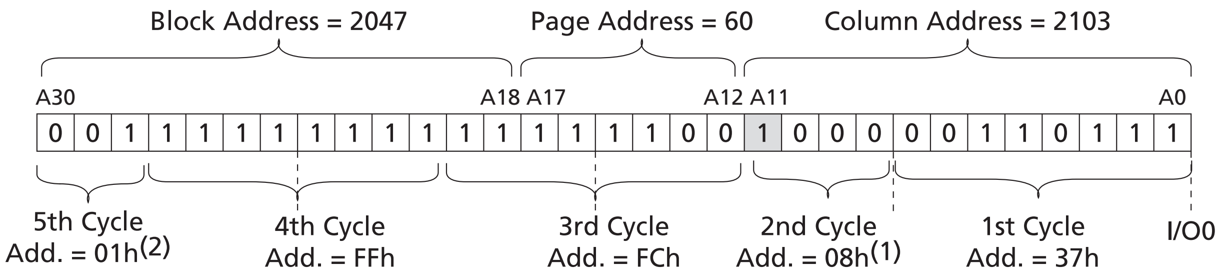
Figure 1: x8 Device: Programming the Main Area of Block 2047, Page 60, Starting from Byte 2001



MAIN AREA: A11 = 0, Column Address = 2000

1. I/O4 to I/O7 are set to Low, VIL.
2. I/O3 to I/O7 are set to Low, VIL.

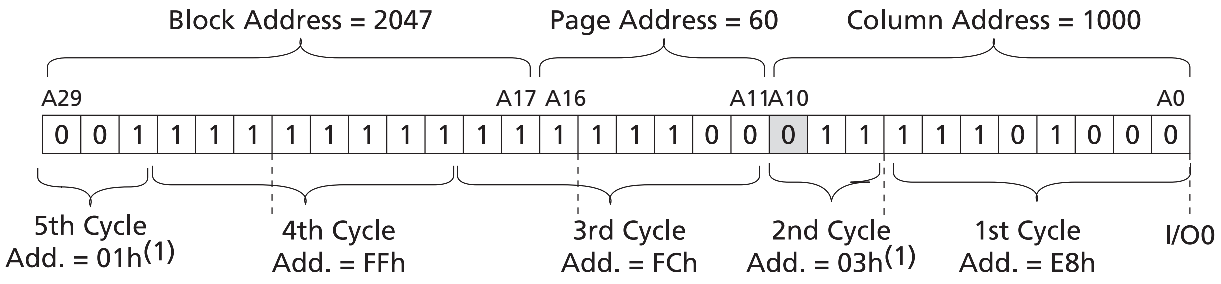
Figure 2: x8 Device: Programming the Spare Area of Block 2047, Page 60, Starting from Byte 56



SPARE AREA: A11 = 1, Column Address = 2048 + 55 = 2103

1. I/O4 to I/O7 are set to Low, VIL.
2. I/O3 to I/O7 are set to Low, VIL.

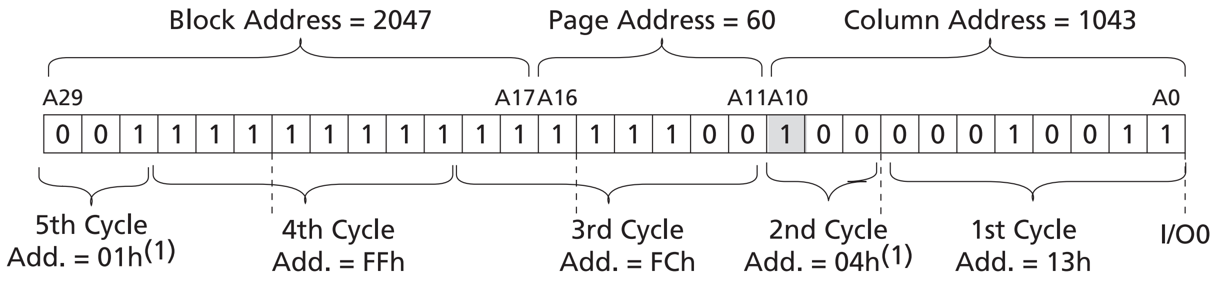
Figure 3: x16 Device: Programming the Main Area of Block 2047, Page 60, Starting from Word 1001



MAIN AREA: A10 = 0, Column Address = 1000

1. I/O3 to IO/7 are set to Low, VIL.

Figure 4: x16 Device: Programming the Spare Area of Block 2047, Page 60, Starting from Word 20



SPARE AREA: A11 = 1, Column Address = 19 + 1024 = 1043

1. I/O3 to IO/7 are set to Low, VIL.

Getting Started

To test the source code in the target system, start by reading from the NANDxxx-B, NANDxxx-C, or NANDxxx-D device. If the device is erased, only 0xFFh data should be read.

Then, read the manufacturer and device codes by issuing a NAND_ReadElectronicSignature and check that they are correct. If these functions work, the other functions are also likely to work. However, all functions should be tested thoroughly.

To start, write a function main() and include the header file of the LLD. All large page Flash memory functions can be called and executed within the main() function. The following example checks the device identifiers (device code, manufacturer code) and performs a BLOCK ERASE command:

```
void main(void) {
    dataWidth buffer[4];
    udword address;
    udword n_block;

    memset(buffer,0xFF,4);
    printf("Read Electronic Signature \n");
    NAND_ReadElectronicSignature(buffer);
    printf("Manufacturer Code: %x\n",buffer[0]);
    printf("Device Code: %x\n",buffer[1]);
    printf("Reserved Byte 3: %x\n",buffer[2]);
    printf("Byte 4: %x\n",buffer[3]);

    n_block = 10;   // block 10 will be erased
    address = n_block * PAGE_DATA_SIZE * NUM_PAGE_BLOCK;
    NAND_BlockErase(address);
} /* EndFunction Main */
```

Porting the Software Driver

Porting the software driver on a particular platform requires that the hardware-dependent layer functions be rewritten to correctly manage the signals of the NAND Flash memory device connected to the platform.

Connecting with GPIO

A NAND Flash memory device can be connected to a ROM/SRAM/Flash bank and at least two GPIO ports and the following signals:

- nOE (not output enable): If a memory access occurs, the nOE output controls the output enable port of the specific memory.
- nWBE (not write byte enable): If a memory access occurs, the nWBE outputs control the write enable port of the specific memory.
- nRCS (not ROM/SRAM/Flash chip select): KS32C50100 can access up to six external Flash banks. By controlling the nRCS signals, the CPU addresses can be mapped into the physical memory banks.

Any bus operation that accesses the NAND device consists of two steps:

1. Setting the GPIO pins that drive the NAND control signals (CL, AL).
2. A WRITE or READ operation to the bank where the NAND Flash memory device is mapped.

The control nOE, nWBE, and nRCS signals of the microcontroller are directly connected to the NAND memory device. This hardware solution is possible only for NAND Flash memory devices with the “Chip Enable Don’t Care” option, which enables the NAND device to share the ROM/SRAM/Flash controller.

The NAND device’s WP signal can be managed via hardware (using a jumper) or via software with a GPI/O. It is possible to monitor the ready/busy state of the NAND device by reading the status register, or by connecting a GPI/O to the ready/busy (RB) pin.

Hardware Interface

The Evaluator7T board used for this example is built around a KS32C50100 microcontroller and based on a 16/32-bit ARM7TDMI RISC processor (refer to the KS32C50100 data sheet). The ARM7TDMI is a low-power, general-purpose microprocessor macrocell developed by Advanced RISC Machines (ARM).

The KS32C50100 can operate at a bus frequency of 50Mhz. It has a system memory controller (SMC) with configurable banks for ROM/SRAM/Flash and DRAM and for external devices.

Example

The following example shows how to configure the GPIO connection by using the hardware-dependent NAND_CommandInput function. Three steps are required:

1. **Set GPIO signals:** Set GPI/O 0 LOW and GPI/O 1 HIGH to drive the NAND control signals latch enable (AL) LOW and command latch enable (CL) HIGH.
2. **Issue a command to the NAND Flash memory:** Perform a WRITE or READ operation to the bank where the memory is mapped to set the CHIP ENABLE ($\overline{\text{E}}$) and WRITE ENABLE ($\overline{\text{W}}$) signals.
3. **UnSet signals:** Reset GPI/O 0 and GPI/O 1 LOW to drive the NAND control signals latch enable (AL) LOW and Command Latch Enable (CL) LOW.

```
void NAND_CommandInput(ubyte ubCommand) {

    ubyte * udAddress = (ubyte*) Base_Address;

    /* Set Op mode Command (AL=0,CL=1) *

     *(GPIODATA)=BASE_IO_DATA|ASSERT_CL;

    /*transfer to NAND ubCommand */

     *(udAddress) = ubCommand;

    /* UnSet Op mode (AL=0,CL=0) */

     *(GPIODATA)=BASE_IO_DATA;

}
```

Connecting With a Field-Programmable Gate Array (FPGA)

The driver was tested on a Mainstone II board equipped with a Rodan card, which consists of all components required to support a configurable Flash interface based on FPG.s interposer/socket design.

With the Flash interposer removed, an FPGA isolates the Flash interposer data and addresses the bus from the Intel® PXA270 XScale processor (Bulverde) processor and SDRAM system bus.

With the Rodan card, all board transceivers are bypassed and the signals are passed through the FPGA to interface with the Flash side of the bus. The interposer data and addresses bus comes from the FPGA and directly connects to the high-density BGA for the Flash interposer site. The control signals are mostly dedicated single point nets between the FPGA and respective interposer site.

The memory management register bank (MMRB) provides a memory-mapped register set to control the base address, size, and type of each chip select driven out of the FPGA.

It is possible to create a function that automatically writes to the MMRB configuration register at boot up. The function is put into the universal bootloader for ease of configuration.

Since the NAND devices cannot be directly memory mapped, a memory-mapped I/O method must be used to communicate with the device. For the Rodan card, it has been set so that the NAND memory will be on chip select 5 of the XScale processor.

Table 10: NAND Controller Register Addresses

Transfer Type	Address	Read/Write
Chip Base (CB)	0x14000000	–
Command	CB Addr. + 0x20000	Write
Address	CB Addr. + 0x10000	Write
Data Write	CB Addr. + 0x0	Write
Data Read	CB Addr. + 0x0	Read

The following example shows how to configure the Mainstone II registers:

```
/*MAINSTONEII BOARD*/
#define BASE_ADDR 0x94000000

NMX_uint16*     p_command;              /*CLE address*/
NMX_uint16*     p_address_common_area;  /*ALE address*/
NMX_uint16*     p_read;                 /*Common memeory read
                                         address*/
NMX_uint16*     p_write;                /*Common memeory write
                                         address*/

p_command = (NMX_uint16*)(BASE_ADDR + 0x20000);
p_address_common_area = (NMX_uint16*)(BASE_ADDR+ 0x10000);
p_read  = (NMX_uint16*)BASE_ADDR;
p_write = (NMX_uint16*)BASE_ADDR;

void NAND_CommandInput(NMX_uint8 ubCommand) {

*p_command = ubCommand;
}
```

References

- NAND01G-B2B NAND02G-B2C – 1Gb, 2Gb, 2112 byte, or 1056 word page, 1.8 or 3 V, SLC Flash memory data sheets
- NAND04G-B2D NAND08G-BxC – 4Gb, 8Gb, 2112 byte, or 1056 word page, multi-plane architecture, 1.8 or 3 V, SLC Flash memory data sheets
- MT29F1GxxABxDAxx – 1Gb, 2112 byte, or 1056 word page, multi-plane architecture, 1.8 or 3 V, SLC Flash memory data sheets

Revision History

Rev. G 10/11

- Added internal ECC support to the “Large Page NAND Overview” section
- Updated the “Device Operations” section
- Added internal ECC to the “Device Operations” table
- Updated the “Page and Block Address Setting Sequence” table
- Updated the “Return Codes” table
- Updated the “Command Functions” table
- Added a #define statement to the “Choosing the Device” section
- Added ECC_ON_DIE to the “Device Constants Definition” table
- Updated the “References” section

Rev. F 05/11

- Rebranded as a technical note.

Rev. E 07/10

- Applied branding and formatting.

Rev. D 01/10

- Edited for spelling, formatting and grammar
- Removed references to the AN1817 and 1758 application notes
- Added the Multiplane Page Program section
- Added the Multiplane Copy Back Program section
- Added the Multiplane Block Erase section
- Added the Read EDC Status Register section
- Added the EDC Status Register section
- Added the Connecting With a Field Programmable Gate Array (FPGA) section
- Updated the Device Operations section
- Updated the A Detailed Example section
- Updated the Basic Data Types section
- Updated the Return Codes section
- Updated the Hardware Dependent Layer: Basic functions section
- Updated the Hardware Independent Layer: Command functions section
- Updated the Choosing the Device section
- Updated the Getting Started section
- Removed the section Connecting with the Glue Logic

Rev. C 02/07

- Description of PRL function added in the Large Page NAND Flash Memory Overview section.
- Reference data sheet updated in the References section.

Rev. B 06/06

- Product list and 512Mb devices removed.
- Device names replaced by densities in the Values of the Device Constants table.
- Reference documents updated in the References section.

Rev. A 09/05

- Initial release of document.