Freescale Semiconductor
Application Note

Document Number: AN4348
Rev. 0, 09/2011

MCF54418 NAND Flash Controller

by: Liew Tsi Chung
Applications Engineer

1 Introduction

The ColdFire® MCF5441x family is the first group of ColdFire microprocessors equipped with a NAND flash memory controller, bringing a cost-effective NAND device into the embedded market with a lower pin count and higher density in size, compared to NOR flash memory.

NAND flash has quickly become a commodity in the embedded systems and personal computer peripherals industry. In the personal computer industry, NAND flash memory devices are used in USB storage drives, solid-state drives (SSDs), and hybrid hard drives. The NAND flash memory is sturdy, silent, and provides fast access compared to traditional rotating disk hard drives. Its current popularity has led to an increased market share.

This application note provides a general discussion of how to use the MCF5441x NAND flash memory controller.

Contents

1 Introduction 1
2 Signal description 2
3 Connection diagrams 2
4 Multi-level cell and single-level cell 3
5 Single-, dual-, and quad-die flash device 3
6 Programming as boot page and boot from universal bootloader (U-Boot) 3
7 NFC ECC status in NFC buffer memory space 7
8 Using the NFC DMA feature 7
9 Sending a sequence of NAND commands 7
10 NFC sector size 8
11 Mapping virtual pages 9
12 Wear leveling 10
13 Conclusion 10

freescale™ semiconductor

2 Signal description

Table 1. NFC Signal Properties

Signal Name	Function	I/O	Reset status
NFC_ALE	Flash address latch enable	Output	1
NFC_CE	Flash chip enable	Output	1
NFC_CLE	Flash command latch enable	Output	1
NFC_R/B	Flash ready/busy	Input	Pullup
NFC_RE	Flash read enable	Output	1
NFC_WE	Flash write enable	Output	1
NFC_IO[15:0]	Flash data bus	Input/Output	—

3 Connection diagrams

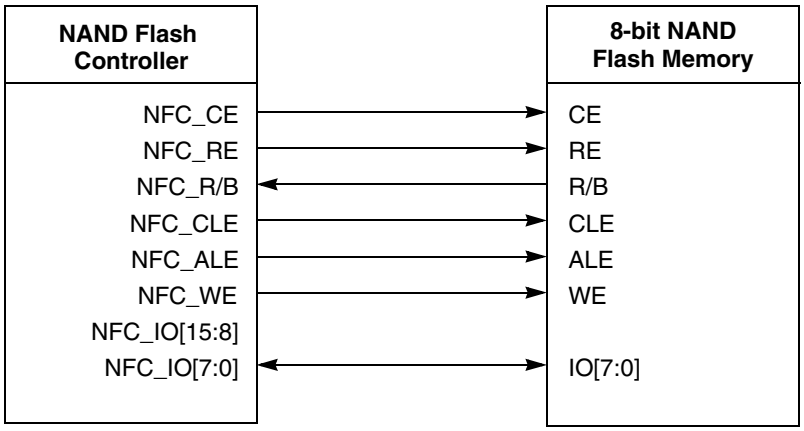


Figure 1. Example diagram for connecting the MCF5441x NFC to an 8-bit NAND memory device

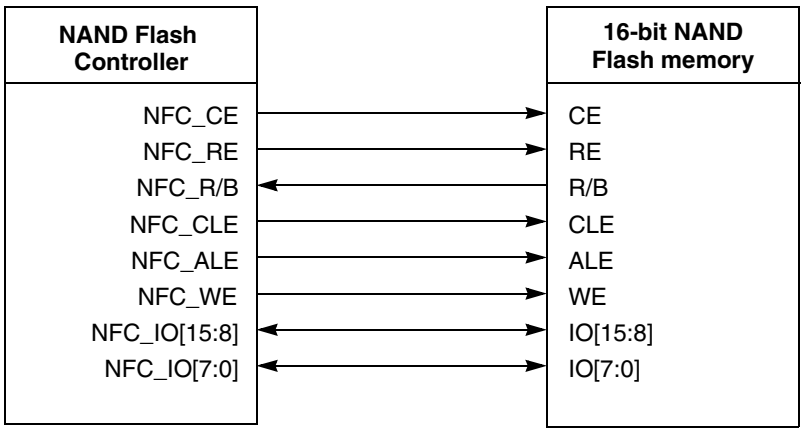


Figure 2. Example diagram for connecting the MCF5441x NFC to a 16-bit NAND memory device (this connection diagram is currently used in TWR-MCF54418)

4 Multi-level cell and single-level cell

A single-level cell (SLC) memory is one in which one bit of data is stored in each cell. A multi-level cell (MLC) memory is one in which more than one bit of data is stored in each cell. The MLC has an advantage of low error rates and higher density. The SLC has an advantage of faster write access, lower power consumption, and higher cell endurance.

The SLC or MLC is the process technology used in the manufacture of the nonvolatile memory device, and it is transparent to the end application. There are no software protocol or signal differences between MLC or SLC devices when connected to a MCF5441x Nand flash controller (NFC).You will only see an increment in page size from 4 KB and up when using the MLC.

5 Single-, dual-, and quad-die flash device

The single-, dual-, and quad-die are methods that the NAND manufacturer uses to package the NAND device. The die may use either SLC or MLC technology. The dual- or quad-die not only saves space on the printed-circuit board, it also reduces the number of I/O pins on the board. The MCF5441x NFC does not support dual- or quad-die NAND flash devices due to the NFC limitation that it only has one CE# and R/B# signal. A dual- or quad-die needs two CE# and two R/B# in order to function.

6 Programming as boot page and boot from universal bootloader (U-Boot)

The MCF5441x NFC can use either an 8- or 16-bit bus NAND device. Programming in 16-bit can be different than programming in 8-bit version. All pages in either 8- or 16-bit NAND devices must be erased before any write can proceed.

In a 16-bit NAND device, the NFC must be set to 8-bit mode for programming or booting the boot page. Using the TWR-MCF54418 NAND flash device as an example, a page is (2048 + 64) bytes (see Figure 3).

When set to 8-bit mode in 16-bit NAND flash, the upper 8-bit bus NFC_IO[15:8] will not be used temporarily. The total size for the usable space will be 2112 ÷ 2 = 1056 bytes per page (see Figure 4). Additionally, the boot mode must have 32-error correction (60 ECC bytes) set in NFC_CFG[ECCMODE] for write. Thus, this will give 996 bytes (1056 – 60 bytes) for data. When changing back to 16-bit after a success boot, the data content in the first four boot pages will alternate between 0xFF and 0xnn.

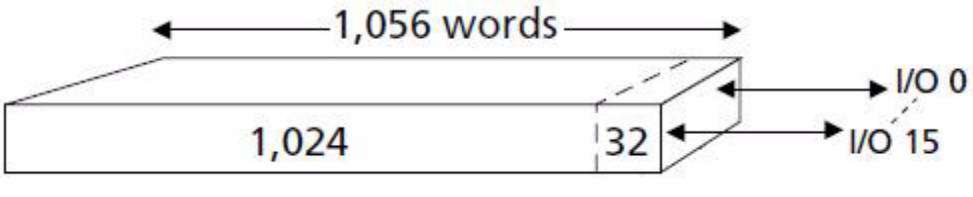


Figure 3. Array organization

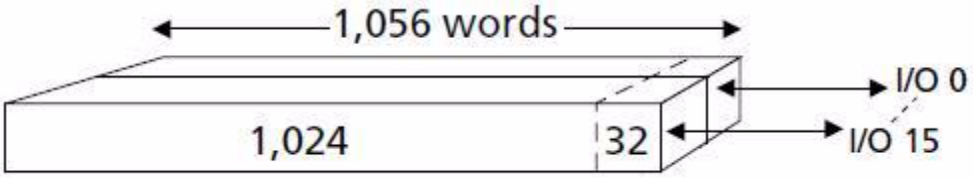


Figure 4. Array organization of 8-bit mode and 16-bit flash

When booting from the universal bootloader, or U-Boot, the bootloader will probe for the first spare data location other than FFh (x8) or FFFFh (x16) of the first two pages of every block to determine if the block is invalid. Two bad block tables will then be created at the end of the last four blocks of the flash. One of the bad block tables is used as mirror backup for the primary table.

Due to the way that the NFC programs or boots in 16-bit NAND, the first spare data in the boot page is represented as FFFFh. It is always represent as FFnnh, and U-Boot will interpret the first block as bad block when booting up in U-Boot. You should ignore the error generated by the U-Boot on the first two blocks.

```
CPU:   Freescale MCF54418 (Mask:a3 Version:1)
       CPU CLK 250 MHz BUS CLK 125 MHz FLB CLK 62.500 MHz
       INP CLK 50 MHz VCO CLK 500 MHz
Board: Freescale M54418 Tower System
SPI:   ready
DRAM:  128 MB
NAND:  256 MiB
Bad block table not found for chip 0
Bad block table not found for chip 0
Bad eraseblock 0 at 0x00000000
Bad eraseblock 1 at 0x00020000
Bad block table written to 0x0ffe0000, version 0x01
Bad block table written to 0x0ffc0000, version 0x01
In:    serial
Out:   serial
Err:   serial
Net:   FEC0, FEC1
->
```

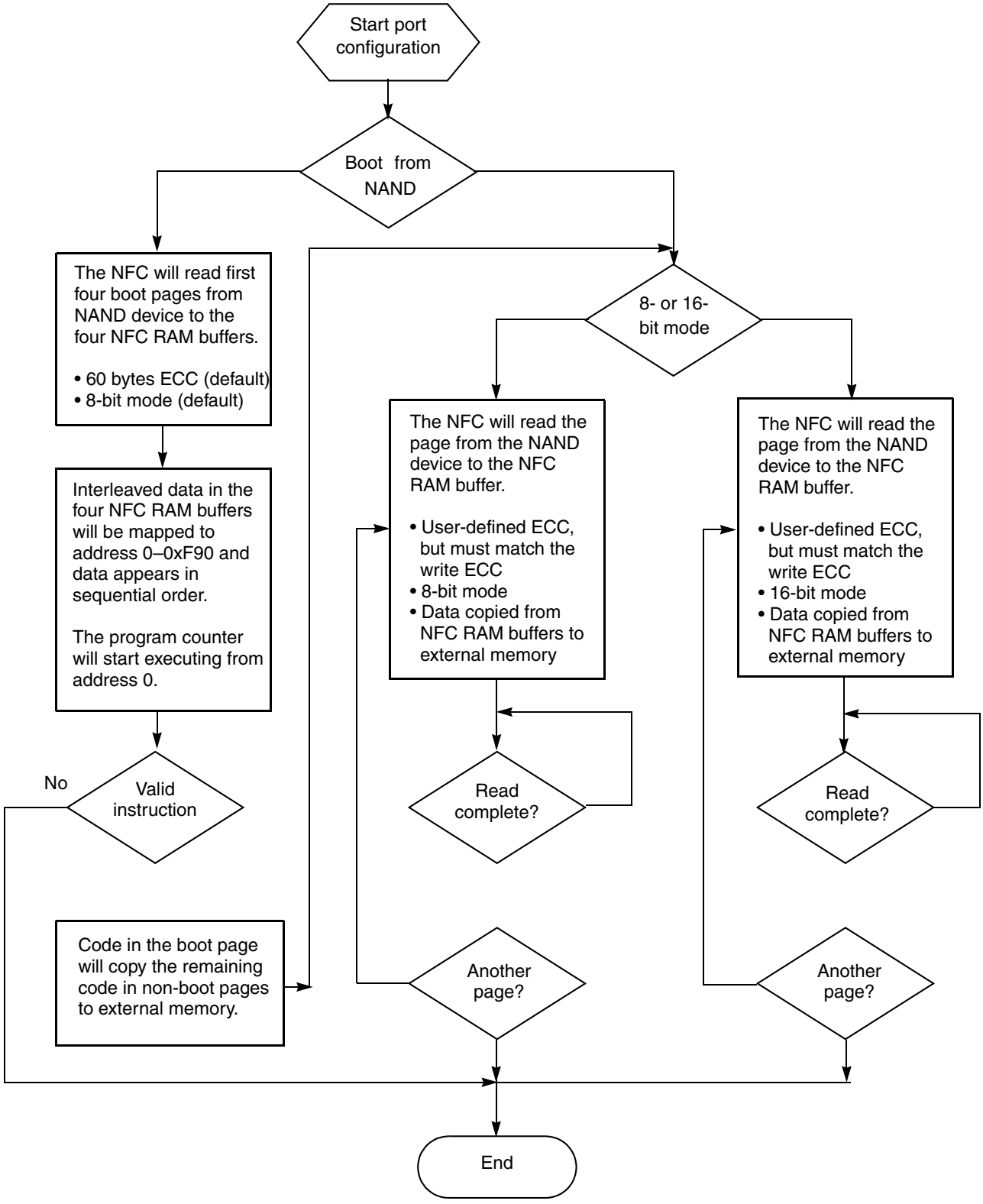


Figure 5. Basic NFC read

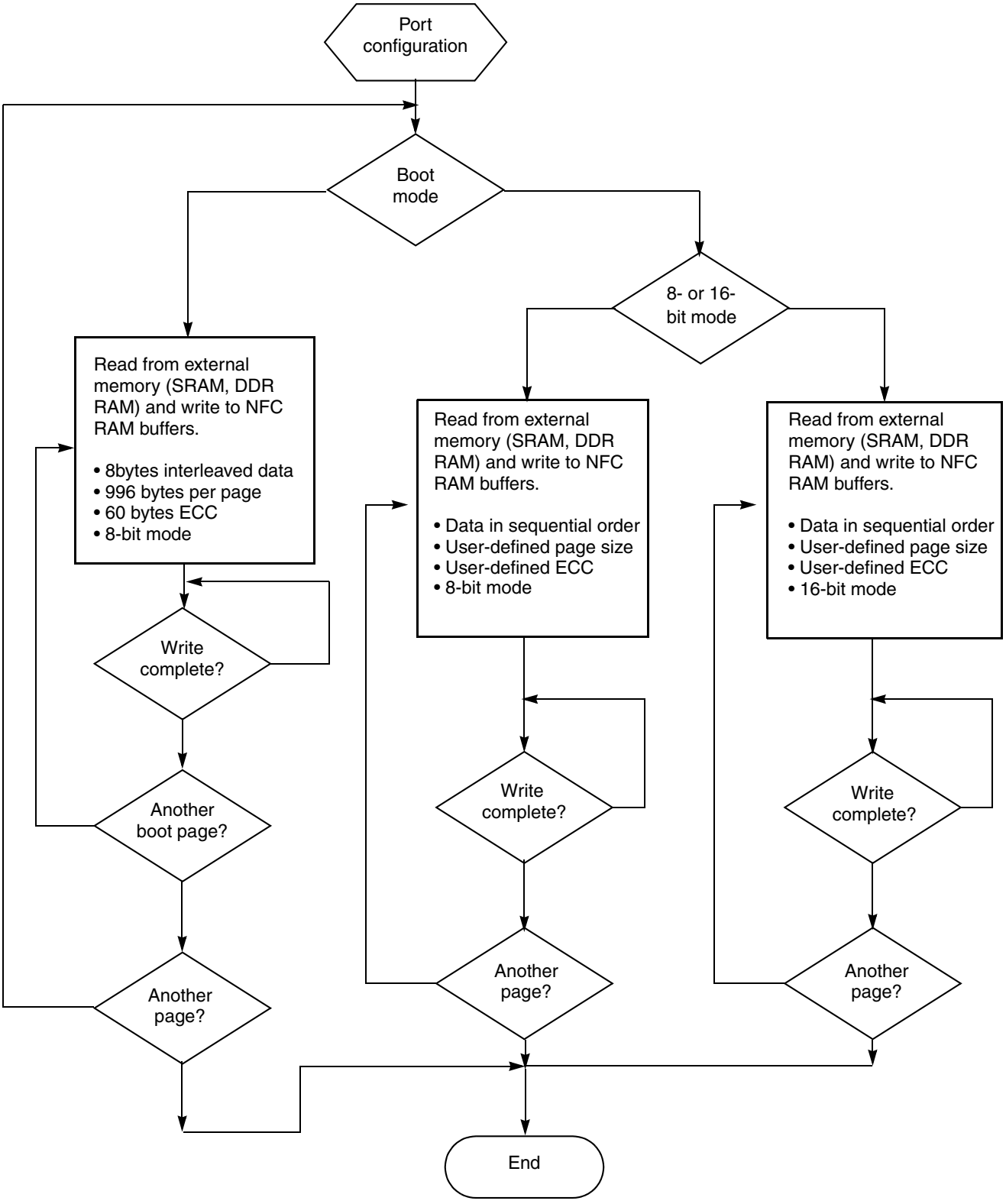
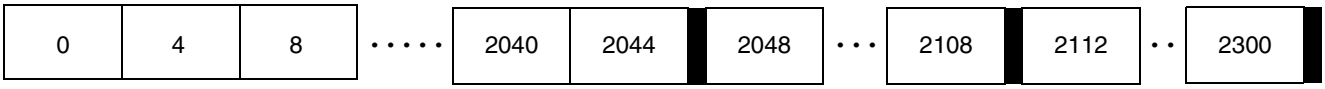


Figure 6. Basic NFC write

7 NFC ECC status in NFC buffer memory space

The NFC ECC status bit allows the NAND controller to report how many error bits are found in a page and to determine whether the page is correctable. Each increment in NFC_CFG[ECCAD] is a long word increment in the NFC buffer memory space. When setting the NFC_CFG[ECCSRAM], the NFC_CFG[ECCAD] should not be left in its default setting or any value lower than the page size. When performing a read from a NAND device, the NFC ECC status will write to the NFC RAM after the data is retrieved from the NAND flash.

There are four buffer memory spaces in the NFC, each with a size of 2304 bytes. Using NAND memory with 2048 + 68 bytes per page as an example, the ECC status address offset should start from a NFC buffer memory offset from 2112 to 2300.



8 Using the NFC DMA feature

The NFC DMA is used for data transfers between external memory (such as SRAM, DDR RAM, and so on) and NAND flash memory devices with minimal intervention from a host processor. The data read from or written to NAND flash memory still has to go through the NFC buffer memory space, though without the CPU interaction to move the data in or out to a external memory space.

DMA channel 1 is used for transferring large chunks of data at offset 0 in a page of the NAND flash memory; the maximum transfer size is up to 2 KB per chunk. DMA channel 2 is used for transferring small chunks of data at any offset up to 2048 bytes; the maximum transfer size can reach up to 128 bytes.

The following example demonstrates an NFC DMA channel 1 transfer between an NFC/NAND flash memory device and internal SRAM:

```
NFC_DMA1 = 0x8000_0000
NFC_DMACFG[COUNT1] = 0x840
NFC_DMACFG[ACT1] = 1
```

The following example demonstrates an NFC DMA channel 2 transfer between an NFC/NAND flash memory device and internal SRAM:

```
NFC_DMA2 = 0x8000_0000
NFC_DMACFG[COUNT2] = 0x40
NFC_DMACFG[OFFSET2] = 0x100
NFC_DMACFG[ACT2] = 1
```

9 Sending a sequence of NAND commands

The MCF5441x NFC can send up to three NAND commands in a row without more CPU and NFC accesses. However, NAND commands, such as NAND READ STATUS, cannot be performed under NFC after a NAND RESET. When a NAND RESET is issued, the NAND flash memory device logic resets

itself to a known state. At this time, any second command issues will be invalid. The NAND command has to always start from BYTE1; you cannot skip BYTE1 and start from BYTE2.

The following example demonstrates how to erase a block with read status.:

```
NFC_CFG[STOPWERR] = 1
NFC_CFG[IDCNT] = 5
NFC_CFG[TIMEOUT] = 6
NFC_CFG[16BIT] = 1


NFC_ISR[WERRCLR] = 1
NFC_ISR[DONECLR] = 1
NFC_ISR[IDLECLR] = 1


NFC_CMD2[BYTE1] = 0x60          // NAND ERASE command 1
NFC_CMD2[CODE] = 0x4ED8         // Refer to MCF54418 RM, Table 22-22
                                // Send command byte 1
                                // Send row address 1, 2 & 3
                                // Send command byte 2
                                // Wait for flash R/B
                                // Send command byte 3
                                // Read flash status


NFC_CMD1[BYTE2] = 0xD0/         / NAND ERASE command 2
NFC_CMD1[BYTE3] = 0x70          // NAND READ STATUS


NFC_RAR[BYTE1] = 0
NFC_RAR[BYTE2] = 0
NFC_RAR[BYTE3] = 0

NFC_RPT[COUNT] = 0

NFC_CMD2[START] = 1

while (!(NFC_ISR[DONE] & 1))

status = NFC_SR2[STATUS1]       // The STATUS1 is obtained from NAND READ STATUS
```

10 NFC sector size

In NAND flash memory, a page consists of data and a spare region. The spare area is usually for ECC, file system markers, bad block markers, and so on. In NFC, sector size is NAND page size. For 16-bit data width flash, you must add a 1 to the sector size. If the NFC sector is 1, no data is written or read.

The ECC in the NFC is based on BCH codes. The NFC ECC is not located at the beginning of the spare region; instead, it starts at the end of the spare offset.When 60 ECC bytes (32-error correction) are used, the starting offset is at 804h on a 2048 + 64 byte page. (Spare region size – ECC bytes = ECC starting offset from the spare region, or page size – ECC bytes = ECC starting offset from a page.)

The following examples demonstrates how the NFC ECC is located on a 2048 + 64 byte page:

Figure 7. A page without ECC enabled

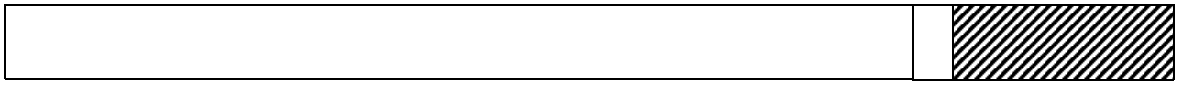


Figure 8. A page with 60 ECC bytes enabled

Figure 9. A page with 8 ECC bytes enabled

11 Mapping virtual pages

The NFC virtual pages are used when a page size is larger than 2 KB, or when certain applications, such as a file allocation table (FAT) file system, require it. The NFC can support a maximum of 15 virtual pages, and the minimum page size is 256 + 8 bytes. Be aware that when NFC is split into virtual pages, the spare region size is also reduced and ECC bytes should be chosen accordingly. When ECC bytes are larger than the space available in the spare area, the NFC ECC logic will write to the data region, causing data corruption.

The following example demonstrates how to apply the eight virtual pages on a 4096 + 208 byte page with a minimum of 8 ECC bytes support.

```
NFC_SECSZ = 538
NFC_CFG[16BIT] = 0
NFC_CFG[PAGECNT] = 8
NFC_CFG[ECCMODE] = 1// 4-error correction (23 ECC bytes)
```

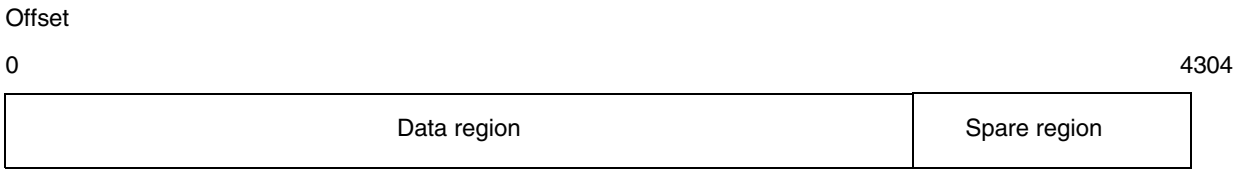


Figure 10. Physical page layout

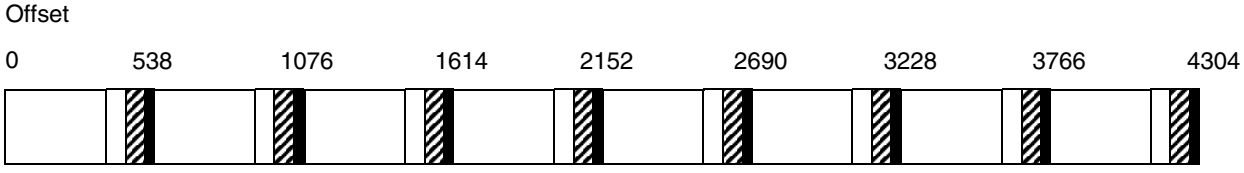


Figure 11. Eight virtual pages mapped with 8 ECC bytes enabled

12 Wear leveling

Today, NAND flash memory can be programmed and erased reliably up to 100,000 cycles. Wear leveling helps to prevent data loss in case of excessive wear. There are two types of wear leveling in NAND technology: error correction code/bad block management and flash translation layer (FTL)/file system leveling (FSL). The ECC logic integrated into NFC reduces the CPU cycles needed for calculation, thus speeding up the read/write process. The code is required to monitor a bad block by reporting and recording it into the bad block management tables. The FTL or FSL helps to distribute the write evenly among the blocks. This will reduce cycle access on certain blocks and prolong the NAND flash memory life.

13 Conclusion

This application note is intended to help you better understand NFC and to make it easier to use. The techniques discussed in this document can also be applied to the MPC5125 and K70 NFC, as well as any future parts based on this NAND flash controller.

THIS PAGE IS INTENTIONALLY BLANK

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: AN4348
Rev. 0
09/2011

freescale™
semiconductor