Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

THIS SPEC IS OBSOLETE

Spec No: 001-61347

Spec Title: AN61347 - NX2LP-FLEX USB TO NAND FLASH FIRMWARE DESIGN NOTE

Replaced by: None

AN61347

NX2LP-Flex USB to NAND Flash Firmware Design Notes

Author: Gayathri Vasudevan
Associated Part Family: CY3686

This application note describes details about the firmware developed by Cypress for NX2LP products. It describes the roles of functions and use of #defines that help determine the right build for a given NAND Flash and NX2LP device combination. This application note also briefly explains the Command Block Wrapper and Command Status Wrapper that are used for USB communication along with how the NX2LP stores the firmware image on the NAND Flash device.

Contents

1 Introduction

The CY3686 is a flexible NAND Flash solution that enables adding features to a USB thumb drive solution. The Cypress EZ-USB® NX2LP-Flex Mass Storage reference design connects the EZ-USB NX2LP-Flex to two to eight NAND Flash chips. Additional devices can be connected through GPIF interfaces or the spare GPIO pins. These devices include MP3 Decoder, Cypress WirelessUSB, pROC, and DVB video capture devices.

You should be familiar with the USB Mass Storage Class specification and general operation of the Cypress

EZ-USB FX2LP and EZ-USB NX2LP-DVK to get the most from this document. For more information, refer to these specifications or the Cypress EZ-USB Technical Reference Manual for EZ-USB FX2LP.

2 Mass Storage Class Specification

The USB Mass Storage Class specification contains two subclasses, the CBI (Command, Bulk, Interrupt), and the newer Bulk Only Transport. This reference design complies with the Bulk Only subclass of the USB Mass Storage Specification. The Bulk Only subclass is supported by Windows XP, 2000, ME, Vista, and 7 drivers as well as MacOS 9 and X. Cypress provides custom drivers for Windows and Macintosh operating systems to add support for security. The latest versions of these drivers are available on the Cypress website.

3 Firmware Overview

The firmware for the device is a straight-forward implementation of a USB Bulk Only mass storage device. After reset, it waits for a CBW packet, checks it, and then executes the data phase of the command (if any). When the data phase is complete, the firmware sends a CSW packet to the host. SETUP commands are handled in an ISR. A timer ISR template is provided in the source code; it is disabled. The firmware gets a SCSI INQUIRY string, USB device descriptors, USB string descriptors, and the NAND Flash configuration information from the NAND Flash configuration pages. The firmware initializes its hardware and software, and then all the SCSI commands from the USB host. For more information, see *ide.c*.

The CY3686 firmware supports both high-speed (480 Mbps) and full-speed (12 Mbps) hosts.

Note CBW, CSW, dataTransferLength, and "Persistent Stall" are defined in the "USB Mass Storage Class, Bulk Only Transport"; see References.

4 Firmware Details

Figure 1 shows the flowchart for the firmware. The firmware has three main sections:

- Initialization
- Command (CBW) processing
- ISRs

The initialization code sets up the hardware, loads the NAND Flash configuration, and detects the number of NAND Flash chips. The initialization routines are: TD_Init, InitNand, and GetNandCfg.

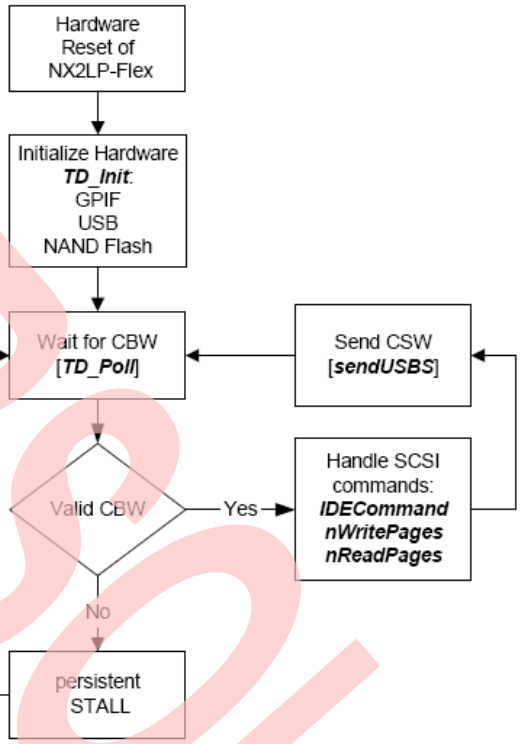
When the initialization code is complete, the hardware is set up, GPIF is configured for an 8-bit NAND bus, NAND Flash chips are fully initialized; and the firmware is ready to accept CBW commands. The CBW processing takes place in the TD_Poll loop. This loop also polls the sleep flag to determine if it is time for a USB suspend.

The ISRs handle SETUP command processing and background polling for events such as USB power management.

The ISR subroutines are:

- `resume_isr`: handles USB Resume
- `ISR_Sudav`: handles Setup Data
- `ISR_Susp`: handles USB Suspend
- `ISR_Ures`: handles USB Reset
- `ISR_Highspeed`: handles USB High Speed

Figure 1. Program Flow



4.1.1 main

The main routine calls the TD_Init routine and then starts the master `while (1)` loop. The `while (1)` loop polls the sleep flag and calls TD_Poll, which is the main command processing routine.

4.1.2 TD_Init

A hard reset calls TD_init; TD_init then initializes the USB hardware and calls the InitNAND to set up the GPIF to work with the 8-bit NAND bus and initialize all the software variables to a known state. To add new hardware, modify this subroutine to configure the hardware.

Notes

- InitNAND is called only once on a hard reset. InitNAND is not called on a soft reset.
- All the device descriptors and GPIF wave data are initialized only on a hardware reset. Any soft reset does not initialize the device descriptors or GPIF wave data.

4.1.3 InitNAND

The InitNAND routine initializes all NAND global variables and calls the GetNandCfg (Get NAND Configuration). This subroutine does the following:

- Updates the device descriptors to the halfKBuffer
- Updates the SCSI string for INQUIRY command
- Loads the LUT (look up table via the gLog2Phy) for the Physical to Logical table

4.1.4 VID and PID (dscr.a51)

Do not modify the VID and PID in the `dscr.a51` file; during NAND manufacturing the default VID and PID is used to download the Cypress manufacturing firmware. The user VID and PID are loaded from the NAND Flash configuration block (See the GetNandCfg subroutine).

4.1.5 halfKBuffer

The halfKBuffer is a dedicated USB buffer storage. It contains the following:

```
// 0xE000 - 0xE100: copy of the dscr.a51
// 0xE100 - 0xE11F: SCSI Inquiry string
// String Descriptors:
// 0xE120 - 0xE14F: String Index 0
// 0xE150 - 0xE17F: String Manufacturer - Index 1
// 0xE180 - 0xE1AF: Product - Index 2
// 0xE1B0 - 0xE1DF: Serial Number - Index 3
// 0xE1E0 - 0xE1FF: Reserved area for debug
```

4.1.6 TD_Poll

As in all Cypress Frameworks–based code, the main code loop is called TD_Poll. This routine is called repeatedly until it detects a packet in the OUT buffer. TD_Poll checks the received packet for a valid CBW signature. If one is found, it processes the CBW. If the packet is not a valid CBW, the device enters a “persistent STALL” condition awaiting a device reset.

When a valid CBW is detected, it checks to see if the gPartialCpy flag is set, which enables the contiguous NAND Flash memory page write. If set, it continues to write the NAND pages without doing a block erase or read modify write. This improves the overall transfer rate for the SCSI Write command. If it is not the SCSI write command, it calls IDECommand to handle all the SCSI commands. The SCSI Read calls nReadPages and SCSI Write calls nWritePages. The IDECommand calls the handleVendorCBW when any special NAND manufacturing command is called by the *NandMfg.exe* (NAND Manufacturing Utility Program in the Windows application).

Note SETUP messages are handled in an ISR, so they may be received and responded to at any time. The entire SETUP message is handled within the ISR; therefore, long SETUP traffic adversely affects USB data transfer performance. This is not an issue because Windows does not use SETUP packets after enumeration except to clear STALL conditions.

4.1.7 handleVendorCBW

This subroutine handles all the Cypress NAND manufacturing interface. Do not modify any commands in this subroutine. You can, however, add more commands. Refer to the NandMfg.pdf/NandMfg.chm that is installed with the utility from the NX2LP DVK to learn how these commands are used.

4.1.8 Resets

The firmware performs a hard reset of the drive when the power is turned on. The firmware performs a soft reset of the 8051 and drives on a USB reset or Mass Storage Class reset.

4.1.9 nEraseBlock

This subroutine erases NAND Flash blocks based on `gSrcAdd`. It also calls the nSearchFreeBlock to look for free available blocks from the gLog2Phy table while waiting for the NAND Flash ready signal.

4.1.10 nGetFreeBlk

This subroutine searches until a free block is found. It simply calls the nSearchFreeBlock.

4.1.11 memset16

This subroutine initializes memory data with the following arguments: address, init-data, and counter.

4.1.12 nReadPages

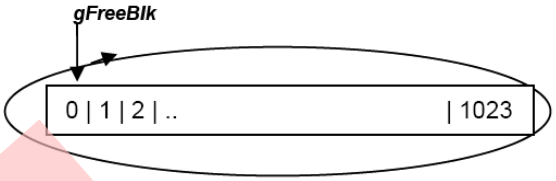
This subroutine handles SCSI Read commands from the CBW. It calls Log2Phy to perform Logical to Physical Block translations and perform the reading of pages from the NAND Flash using GPIF. The NAND reading pages require an address cycle and a data cycle. All address cycle macros are defined in the inand.h file. This subroutine also handles ECC detection. When an ECC compare failure is detected, it sends STALL to the USB host system and expects the USB host system to retry a new SCSI Read command.

4.1.13 nSearchFreeBlock

The firmware finds a free block on 256 iterations. It returns bFreeFound and gFreeBlk as soon as a free NAND block is found. This subroutine supports the wear-leveling algorithm as shown:

- Searching the free block index (gFreeBlk), avoids caching the last erase block

- Searching the free block index cycles through the entire zone. **Note** The zone index range is from 0 to 1023.
- The free block index can either be incremented or decremented.



Wear-leveling is a method to extend the life of a NAND flash. In wear leveling, the logical address that the host accesses on the NAND flash is mapped via a translation table to a random physical address on the NAND flash. Thus, when the host reads, erases, and writes the same logical block address repeatedly, the associated physical block address keeps changing. This way all the physical blocks are used, and only some blocks wear out rapidly. The way these free blocks are handled determines how much wear-leveling is effective in NX2LP. There are various methods of wear leveling. These can be put under two major categories: static and dynamic. The Cypress NX2LP solution uses a dynamic wear-leveling method.

NX2LP firmware divides the NAND into zones of 1024 blocks each. From each zone, it uses only 1000 blocks at a time. Thus, if some blocks are bad in that zone, it would still be acceptable to use that zone for data storage. Every time a block is erased; it is not cached in the search for free blocks. When the host tries to write to the zone, the searchFreeBlock will search for a free block from the rest of the available blocks. This way, all 1024 blocks are used in an orderly fashion without repetition of the last freed block for a full cycle.

Life expectancy of a NAND flash is strictly based on the application usage of the NAND. The typical number of write cycles that a NAND flash can handle is about 100,000. Customers that believe their applications do not follow a typical usage pattern are encouraged to determine how their application will affect the lifetime of the card.

In a typical application, large data is written to the card taking up sequential logical addresses. This results in optimal wear leveling, resulting in the NAND flash exceeding the specification of NAND endurance.

On the opposite side, if the data is written as single sectors to random addresses across the card; these single sector write operations will exercise the wear leveling algorithm extensively. Typically the expectancy can be calculated as follows:

Expected Lifetime (days) =

$$\frac{program\_erase\_cycles}{block} \times number\_of\_blocks \times \frac{block\_size(bytes)}{Bytes\_written/day}$$

The static wear leveling implements usage of the reserved/unused sections of the redundant area (refer to Bad Blocks Management section). The redundant section will carry the information of how many times the block is erased and written to. Every time the host writes a block in a zone, the firmware will have to look through the entire zone to find the least accessed block and write to that block. It is possible to use static wear leveling in NX2LP; however, this will cause performance reduction.

4.1.14 ECCSetup

This subroutine is set to read ECC from the redundant area of the NAND Flash and the GPIF ECC hardware engine. The ecc0[] buffer stores the GPIF ECC hardware engine. The ecc1[] buffer stores the NAND Flash ECC bytes. **Note** The design of this subroutine goal is optimized for the NAND transfer rate.

4.1.15 CheckECC

This subroutine is designed to verify ECC buffers. Error flag (bErr) is set if ECC bytes are not matched. When bErr is set, the USB host receives STALL. This function is used by the original firmware supporting 512-byte page NAND. It is replaced by a combination of ECCSetup_EP4 and CorrectData_EP4. ECCSetup_EP4 reads the spare area and calculates ECC values, compares the values, and calls CorrectData_EP4 if necessary. The function implements a retry mechanism to eliminate transient bit errors. For more information on how the ECC values are calculated, refer the Knowledge Base Article on Hamming Code Explanation.

4.1.16 Log2Phy

This subroutine handles logical to physical translations. For SCSI Read (nReadPages), it returns a physical block address (gPhyAdd). For SCSI Write (nWritePages), it returns a physical block source address (`gSrcAdd`) and a physical block destination address (gPhyAdd).

The Log2Phy subroutine allocates 2K of internal RAM for the logical-to-physical lookup table mapping. Due to the limitation of the NX2LP-Flex internal RAM, 2K of code can map only one Zone table, where the Zone table size is 1K.

Example 1 for zone calculation

128 MByte of 512-NAND type, the zones = 8:

128 MB = 16 KB * 1024 * 8

Where 16 KB = 512 * 32

256 MB of 2K-NAND type, the zones = 2:

256 MB = 128 KB * 1024 * 2

Where 128 KB = 2K * 64

1 GB of 4K-NAND type, the zones = 4:

1 GB = 256 KB * 1024 * 4

Where 256 KB = 4K * 64

The gLog2Phy table reloads from the NAND Flash redundant area when the logical address crosses over zone addressing. The calculation of breaking the LBA (Logical Block Address) to mapping zones + Physical block address mapping is shown in the subroutine Log2Phy.

The data structure of the gLog2Phy is as follows:

```
WORD gLog2Phy[1024];
```

Within a WORD, it maps as follows:

Bits 0-9 = Physical Block Address

Bit10 = Flag indicates initialized bit

Bit15 = Flag indicates use/free flag

Bit14 = Flag indicates configure block

Bit13 = Flag indicates repeat soft ECC error

4.2 Bad Blocks Management

Various applications that use NAND are very write, read, and erase intensive. This causes wearing of memory sections. The NAND device is divided into blocks of memory which allows the storage of data to function even when some of its section wears over a period of time. Managing these bad blocks is necessary and plays a vital role in improving the life expectancy of a NAND device. Cypress NX2LP manages bad blocks by uniquely marking the bad blocks with an identifier.

The bad blocks of a NAND device can be intrinsically bad when they are identified as bad during device manufacturing. Block status byte as shown in Table 2 will be marked as 0x00 for such intrinsic bad blocks. Some blocks that become defective during use are called the acquired bad blocks. The acquired bad blocks are detected by the firmware during read functions, where they fail the ECC test after several retries. These are marked by the firmware with 0xF0 at the block status location.

The firmware partitions the NAND similar to the Smart Media physical format specification as follows:

- Sector size = 512 (1024 for 4K-page NAND firmware)
- Every sector maps to a (logical) page of the NAND Flash where:
  - 512-NAND type has 528 bytes for every page, which partitions into 512-byte for data and 16-byte redundant for bad block management.

- 2K-NAND type has 2112 byte for every page, which partitions into four sectors of 512-byte for data and 16-byte redundant each for bad block management. (In the 2K NAND, refer to this as the subsection when the firmware accesses one page out of four pages.)
- 4K-NAND type has at least 4224 bytes for every page, which partitions to four sectors of 1024-bytes each for data and 32-byte redundant each (spare area) for bad block management (16-byte redundant area for each 512-byte half of data). (In the 4K NAND, refer to each such 1024 byte+32-byte section as a subsection when firmware accesses one page out of four pages.)
- **Note** Refer to Table 1 for NAND page breakup.

- Set the zone size = 1024, where the number of zone computes are based on the NAND type
- Within zone size 1024, it allocates 1000 good blocks and 24 blocks for the bad blocks management.
- **Note** The current firmware can make the good block number become more dynamic to adapt to a NAND that has more bad blocks.
- The configuration block is skipped during a free block search
- Works with the NAND configuration control variables (see GetNandCfg)

Table 1. NAND Page Breakup

(512 byte per page NAND)

Byte	Description
0-0x1FF	NAND data = logical page size (sub section 0)
0x200-0x20F	Redundant Area (Spare Area) (for sub section 0)*

(2K byte per page NAND)

Byte	Description
0-0x1FF	NAND data = logical page size (sub section 0)
0x200-0x20F	Redundant Area (Spare Area) (for sub section 0)*
0x210-0x40F	NAND data = logical page size (sub section 1)
0x40F-0x41F	Redundant Area (Spare Area) (for sub section 1)*
0x420-0x61F	NAND data = logical page size (sub section 2)
0x620-0x62F	Redundant Area (Spare Area) (for sub section 2)*
0x630-0x82F	NAND data = logical page size (sub section 3)
0x830-0x83F	Redundant Area (Spare Area) (for sub section 3)*

(4K byte per page NAND)

Byte	Description
0-0x1FF	NAND data = logical page size (low sub section 0)
0x200-0x3FF	NAND data = logical page size (high sub section 0)
0x400-0x40F	Redundant Area (Spare Area) (low sub section 0)*
0x40F-0x41F	Redundant Area (Spare Area) (high sub section 0)*
0x420-0x61F	NAND data = logical page size (low sub section 1)
0x620-0x81F	NAND data = logical page size (high sub section 1)
0x820-0x82F	Redundant Area (Spare Area) (low sub section 1)*
0x830-0x83F	Redundant Area (Spare Area) (high sub section 1)*

Byte	Description
0x840-0xA3F	NAND data = logical page size (low sub section 2)
0xA40-0xC3F	NAND data = logical page size (high sub section 2)
0xC40-0xC4F	Redundant Area (Spare Area) (low sub section 2)*
0xC50-0xC5F	Redundant Area (Spare Area) (high sub section 2)*
0xC60-0xE5F	NAND data = logical page size (low sub section 3)
0xE60-0x105F	NAND data = logical page size (high sub section 3)
0x1060-0x106F	Redundant Area (Spare Area) (low sub section 3)*
0x1070-0x107F	Redundant Area (Spare Area) (high sub section 3)*

*For redundant area (spare area), see Table 2.

The 16-byte redundant area maps into the following.

Table 2. 16-byte Redundant Area

Byte	Description
0 1	**Block Status:** 0xFF = Good Data block, else Bad block. **Block Type:** 0x01 = configuration page, 0xFF = data block.
2-4	**ECC0:** 3-Byte ECC for the 1st 256-byte
5-7	**ECC1:** 3-Byte ECC for the 2nd 256-byte
8-9	**Addr1** Logical Block Address 1
10-11	**Addr2** Logical Block Address 2 (duplicate field of bytes 8-9)
12-15	Reserved bytes with value 0xFF

The firmware stores the following information: Block Status, Block Type, Addr1, and Addr2 into the gLog2Phy table for handling bad block management. 16 bytes of spare area is used for every 512 bytes of data. In the run-time processing of the SCSI read and SCSI write command, the firmware uses the ECC0 and ECC1 to handle ECC detection and correction.

The manufacturer will store the bad block (intrinsic bad block marking) information on the NAND in the locations 0x200 for 512-byte page NAND, 0x800 for 2K-byte page NAND and 0x1000 for 4K-byte page NAND. This table needs to be relocated in case of 2K-byte and 4K-byte page NAND devices. Hence, it is important for users to issue a cwipe command for first time use of NAND devices with NX2LP. This is performed by using Ctrl+W on the NandMfg utility.

Notes

- Block Status = 0x00 indicates bad block marked by the NAND vendor (intrinsic bad block)
- Block Status = 0xF0 indicates bad block marked by this firmware (acquired bad block)
- The firmware marks the block bad when it detects more than 1-bit ECC error even after several retries during the SCSI read. After marking of such bad blocks, these blocks are not used for data storage unless this information is overwritten or corrupted.

4.2.1 nCopyPages

This subroutine copies pages of the NAND Flash pages. It requires `gSrcAdd` (Source), gPhyAdd (Destination), and page count. It always performs ECC correction while copying data.

4.2.2 nWritePages

This subroutine handles SCSI write commands from the CBW. This subroutine calls Log2Phy to perform Logical-to-Physical Block translations and perform the writing of pages from the NAND Flash using GPIF. The NAND writing pages requires an address cycle and a data cycle.

4.2.3 CorrectData

This subroutine handles data correction using the EP6FIFOBUF. In firmware section for 4K-byte page NAND, another variant of this same function (CorrectDataX) handles correction of data in the higher 512 bytes of EP6FIFOBUF.

4.2.4 nNandMove

This subroutine uses the internal NAND Flash memory move command. It works similarly to the nCopyPages except for the data correction. This subroutine is only used in the 2K NAND type Flash and uses the internal NAND Flash memory move command for a full page 2K-byte move, all others are handled via the nCopyPages.

4.2.5 Fifo6In

This subroutine sets up the FIFO6 as the input to set up data transfers from the NAND to FIFO6 buffer.

4.2.6 NandSendCmd

This subroutine sends NAND commands using a single GPIF write cycle.

4.2.7 NandRead

This general subroutine allows a read from a given endpoint (ep) and the transfer length (len)

4.2.8 GetNandType

This subroutine returns four bytes of NAND ID and two bytes of NAND status.

4.2.9 NandSetAdd

This subroutine is designed to support setting the NAND Flash address for the 2K-NAND and 512-NAND address. In 2K-NAND, it divides 2K into four sub 528-byte page addresses of: 0x000, 0x210, 0x420, and 0x630. An additional address 0x830 is used to access the redundant area of the subpage 4. In 4K-NAND, it divides 4K into four sub 1056-byte page addresses of: 0x000, 0x420, 0x840, and 0xC60. Each sector is 1024 bytes data and 32 bytes spare area. Because the maximum packet size for ECC is 512 bytes, each sector can also be accessed from half the sector size, that is, 512-byte offset.

4.2.10 nCopyBlock

This subroutine is called when an error on ECC is detected during the SCSI Read command. It searches for new free blocks and copy the current block that has a 1-bit ECC error to the new free block.

4.2.11 GetNandCfg

This subroutine gets NAND configuration data for setting up the following information:

1. b30nsCycle = 1: uses 40.8 ns cycles in the GPIF, otherwise use 61.4 ns cycles in the GPIF
2. bInternalMove =1: enable internal move (Only valid for 2K NAND type)
3. bSoftErr = 1: mark the bad block for any 2-bit ECC error and multiple sequences on 1-bit soft ECC error.
4. bWriteProtectEnable = 1: enable the removable device; otherwise this device become unremovable
5. bMaxBlock = 0-0xd: NAND block size expressed as 2^bMaxBlock.

4.2.12 LoadEP2BC

This subroutine controls Endpoint 2. It handles a transfer length larger than the payload, wPacketSize. It breaks the transfer data into multiple chunks of wPacketSize.

4.2.13 ChkErr

This subroutine handles any error for the SCSI Read and SCSI Write commands. When the bErr is set, it sets STALL on the Data phase of the CBW.

4.2.14 stallEP2OUT

This subroutine sets STALL for Endpoint 2 when the dataTransferLen is non-zero.

4.2.15 sendUSBS

This subroutine handles the send USB Status command.

4.2.16 failedIn

This subroutine handles send STALL for Endpoint 4 when dataTransferLen is not zero. The dataTransferLen is extracted from the CBW see TD_Poll. When the host is expecting data from the device, the dataTransferLen is set.

4.2.17 mymemmovexx

This subroutine handles memory moves for the xdata buffer. It handles memory overlap moves.

4.2.18 LoadEP4BC

This subroutine controls Endpoint 4. It handles a transfer length larger than the payload, wPacketSize. It breaks the transfer data into multiple chunks of wPacketSize

4.2.19 cMedia

This subroutine handles the check media status. This subroutine should be called whenever the media changes status. It is called for the following SCSI commands:

- READ_CAPACITY
- TEST_UNIT_READY
- FORMAT_UNIT
- MODE_SELECT

5 Command Block Wrapper (CBW)

The firmware uses the following CBW commands to support the USB Mass Storage device class and NAND Flash programming.

Table 3. CBW Command Block

	Bits							
Offset	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0-3	***DCBWSignature***							
4-7	***DCBWTag***							
8-11	***dCBWDataTransferLength***							
12	***BmCBWFlags***							
13	Dir	***bCBWLUN***						
14	Reserved			***bCBWCBLength***				
15-30	***CBWCB***							

5.1.1 dCBWSignature

This is the signature that helps to identify this data packet as a CBW. The signature field contains the value 0x43425355 (little endian), indicating a CBW.

5.1.2 dCBWTag

A Command Block Tag sent by the host. The device echoes the contents of this field back to the host in the dCSWTag field of the associated CSW. The dCSWTag positively associates a CSW with the corresponding CBW.

5.1.3 dCBWDataTransferLength

The number of bytes of data that the host expects to transfer on the Bulk-In or Bulk-Out endpoint (as indicated by the Direction bit) during the execution of this command. If this field is zero, the device and the host transfer no data between the CBW and the associated CSW. The device ignores the value of the Direction bit in bmCBWFlags.

5.1.4 bmCBWFlags

The bits of this field are defined as follows:

Bit 7 Direction - the device ignores this bit if the dCBWDataTransferLength field is zero, otherwise:

- 0 = Data-Out from host to the device
- 1 = Data-In from the device to the host
- Bit 6 Obsolete. The host sets this bit to zero
- Bits 5..0 Reserved - the host sets these bits to zero

5.1.5 bCBWLUN

This is the device Logical Unit Number (LUN) to which the command block is sent. For devices that support multiple LUNs, the host places into this field the LUN to which this command block is addressed. Otherwise, the host sets this field to zero.

5.1.6 bCBWCBLength

This is the valid length of the CBWCB in bytes. This defines the valid length of the command block. The only legal values are 1 through 16 (01h through 0x10). All other values are reserved.

5.1.7 CBWCB

This is the command block to be executed by the device. The device interprets the first bCBWCBLength bytes in this field as a command block as defined by the command set identified by bInterfaceSubClass.

If the command set supported by the device uses command blocks of fewer than 16 (0x10) bytes in length, the significant bytes are transferred first, beginning with the byte at offset 15 (0xF). The device ignores the content of the CBWCB field past the byte at offset (15 + bCBWCBLength - 1).

Table 4. CSW Status Block

	Bits							
Offset	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0-3	***dCSWSignature***							
4-7	***dCSWTag***							
8-11	***dCSWDataResidue***							
12	***bmCSWStatus***							

The firmware communicates with the *NandMfg.exe* via the CBWCB and the bmCSWStatus for all the command interfaces, which defines as follows:

CBWCB[0] = 0xC8 = Cypress NAND Flash Manufacturing support commands

CBWCB[1] = Opcode

The detail implementation of this support is found in the handleVendorCBW subroutine.

For more information about CSW refer to USB Mass Storage Class – Bulk Only Transport.

5.2 Acronyms Used

The following table lists the acronyms that are used in this document.

Acronym	Description
2K-NAND	NAND has page size = 2112
4K-NAND	NAND has page size ≥ 4224
512-NAND	NAND has page size = 528
CBW	Command Block Wrapper. A packet containing a command block and associated information.
CSW	Command Status Wrapper. A packet containing the status of a command block.
ECC	Error correcting code. Logic designed to detect and correct memory errors
GPIO	General purpose I/O
SCSI	Small Computer System Interface

6 File Descriptions

The following table describes the purpose of files stored in the firmware directory of NX2LP-Flex.

Filename	Purpose
Dscr.a51	Descriptor table containing PID/VID, endpoint descriptions, and other information reported to the host on startup.
reset.a51	Assembly routine used to branch to 0 on USB reset.
Startup.a51	Modified Keil startup file that does not initialize any variables.
USBjmptbl.a51	USB interrupt vector table for FX2LP and NX2LP-Flex. When the NX2LP flag is defined, it selects the NX2LP interrupt vector
inand.c	NAND Flash subroutines that handle SCSI reads and writes. The subroutines are: ❑ InitNand: NAND hardware/software initialization ❑ nGetFreeBlk: Search the gLog2Phy table to get free block ❑ nEraseBlock: Erase the used block ❑ memset16: Fill data ❑ nReadPages: CBW SCSI Read command ❑ ECCSetup: ECC setup to store to internal ram buffers ❑ CheckECC: Verify ECC ❑ Log2Phy: Get Physical block based on the LBA ❑ nCopyPages: Copy NAND data ❑ nSearchFreeBlock: Wear-leveling search for free block from gLog2Phy table ❑ nWritePages: CBW SCSI Write command ❑ CorrectData/CorrectDataX: ECC correction in the EP6 Buffer nNandMove: Internal move command for 2K NAND type
vend_cbw.c	Support for the NAND Flash Manufacturing Utility command interface via the CBW. The subroutines are: ❑ Fifo6In: Setup EP6 for transfer data from NAND to FIFO6 Buffer

Filename	Purpose
	❑ NandSendCmd: General send a command to NAND ❑ NandRead: General read a page data from NAND ❑ GetNandType: Get 4-byte NAND ID and 2-byte of status ❑ NandSetAdd: General send address to the NAND ❑ nReadCfgPage: Reading NAND configuration pages ❑ nCopyBlock: Perform ECC correction while copying NAND data ❑ GetNandCfg: Get NAND configuration information ❑ CheckSignature: Check for special signature for the NAND configuration ❑ handleVendorCBW: *NandMfg.exe* commands interface ❑ loadEP2BC: Enpoint2 Control
fw.c	Frameworks based main routine: ❑ main: main program for this firmware ❑ SetupCommand: handle all chapter 9 commands via interrupt ISR_Sudav ❑ resume_isr: Resume ISR ❑ sendDescriptor: general sending descriptors
Globals.c	Global variable definitions.
ide.c	Handles SCSI commands. The subroutines are: ❑ IDECommand: Handles most of the UFI commands ❑ cMedia: Check Media ❑ waitForInBuffer: general polling endpoint 4 for IN buffer available ❑ loadEP4BC: endpoint 4 control
periph.c	Hooks required to implement the USB peripheral functions ❑ TD_Init: Hardware/Software initialization ❑ TD_Poll: Task Dispatcher hook for USB Mass Storage Class ❑ ChkErr: ECC handler ❑ stallEP2OUT: force STALL on endpoint 2 ❑ sendUSBS: send SCSI CBS ❑ failedIn: force STALL on endpoint 4 ❑ mymemmovexx: general memory move subroutine
Globals.h	Global variable references
gpif.c and gpif.h	Header files containing hardware GPIF wave form.
scsi.h	SCSI command set
CY3686fw.Opt	Options for UV2 project
inand_fw.hex nand_fw512.hex nand_mc2k.hex nand_fw2k.hex nand_mc4k.hex	Output file from the linker ❑ inand_fw.hex: is the firmware file for NX2LP-Flex chip that support 512-NAND type ❑ nand_fw512.hex: is the firmware file for FX2LP chip that support for 512-NAND type

Filename	Purpose
nand_fw4k.hex	❑ nand_mc2k.hex: is the firmware file for NX2LP-Flex chip that support for 2K-NAND type ❑ nand_fw2k.hex: is the firmware file for FX2LP chip that support for 2K-NAND type ❑ nand_mc4k.hex: is the firmware file for NX2LP-Flex chip that support for 4K-NAND type ❑ nand_fw4k.hex: is the firmware file for FX2LP chip that support for 4K-NAND type
CY3686fw.Uv2	UV2 project file
ezusb.lib	FX2LP library file
fx2_intreg.inc	Include the definition for NX2LP-Flex interrupt vectors definitions and FX2LP interrupt vectors definitions

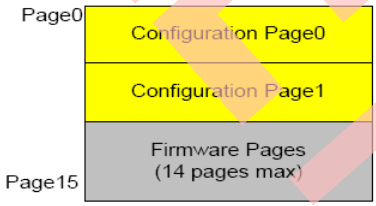
7 NAND Flash Configuration Area

Many of the commonly changed items in the CY3686 configuration are moved to a dedicated NAND Flash configuration area. Locating these items in a dedicated area allows customization of many firmware attributes such as Vendor ID, Product ID, SCSI Device String information, NAND Flash timing, NAND Flash special command features (Read Cache, Program Cache, NAND Flash Internal command move and so on), and the firmware itself.

The NAND Manufacturing Utility program is provided to assist you in creating and programming the configuration file. This NAND Flash Manufacturing utility (*NandMfg.exe*) can be found in the `\Cypress\USB\NX2LP-Flex\MfgTool` directory on your CD. The *NandMfg.exe* can program or modify the NAND Flash configuration block on your CY3686 board.

The NAND Flash configuration block begins at NAND Flash physical block 0 and ends at physical block 23. The *NandMfg.exe* searches for free available blocks within block 0-23 to allocate this configuration block. This block is marked as the configuration block, so the firmware cannot use this block as the data block.

The detailed implementation of the interface between NAND firmware and *NandMfg.exe* is not exposed here, because the interface must not be modified.



The NAND Flash configuration page0 stores the following:

Byte	Field name	Description
0-5	NAND Flash signature bytes	ASCII data “SMTDMG” indicates a good signature for the NAND Flash. This signature is used by the Boot-Loader to validate it as a known good data page
6-7	NextBlock	Next Link Block for the next FW image block
8	Firmware Code Page	Start code address is at 0x400. Number of pages to download FW (0-32). This number should not be bigger than 30
9	Boot image option	N = 0-3 number of 528 per NAND page. For 2KP/4KP NAND this number can be 0-3. For 512-NAND this number should be zero
0xA	NAND devices	Number of NAND device (Mfg tool need to scan the number of NAND in the bus via sending the command via the CBW)
0xB	Page (n)	Number of pages in a block. The expression should be express in the 2^n. For example: n = 9 => 2^n = 512 n = 10 => 2^n = 1024
0xC	Max Block (n)	Maximum number of block. The expression should be express in the 2^n. For example: n = 9 => 2^n = 512 n = 10 => 2^n = 1024
0xD	NAND Configuration byte	Bit0 = 1 = Read Cache enable; 0=disable Bit1 = 1 = Program Cache enable; 0=disable Bit2-3: 00 = 50ns cycle, 01=30ns; 10=100ns 11 = reserved Bit4 = 1 = Internal Data Move enable; 0=disable Bit5-7 are reserved
0xE	Firmware Configuration	Bit0 = 1 = Write Protect Support; 0 = disable Bit1 = 1 = ECC Enable, 0 = disable Bit2 = 1 = Interleave NAND FW, 0=normal Bit4-7 are reserved
0x0F	Firmware Revision	Byte 0-3 = LSB Byte 4-7 = MSB
0x10-0x1F	16-bytes Redundant configuration	Duplicate bytes from 0x00-0x0F

The NAND flash configuration page1 stores the following:

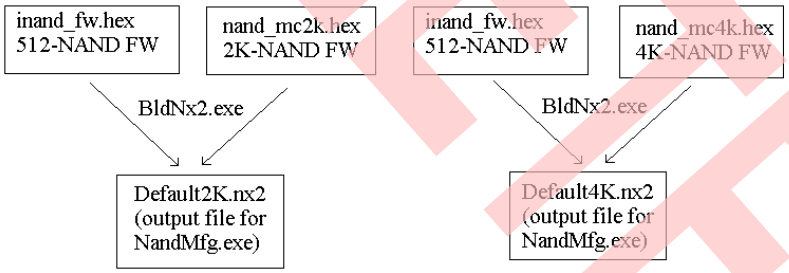
Offset	Field Name	Description
0x00	IdVendorLSB	Vendor ID
0x01	IdVendorMSB	
0x02	IdProductLSB	Product ID
0x03	IdProductMSB	
0x04-0x0F	Reserved	
0x10-0x27	DeviceName	Device Name for SCSI Inquiry (24 ASCII characters, pad with spaces)
0x28-0x2F	Reserved	
USB String Descriptor–Index 0 (LANGID)		
0x30	Blength	String descriptor length in bytes
0x31	BDescriptorType	Descriptor Type (String)

Offset	Field Name	Description
0x32	LANGID (LSB)	Language ID LSB (0x09 = English)
0x33	LANGID (MSB)	Language ID MSB (0x04 = English)
0x34-0x5F	Reserved	
USB String Descriptor–Manufacturer		
0x60	Blength	String descriptor length in bytes
0x61	BdescriptorType	Descriptor Type (String)
0x62-0x8F	Bstring	Manufacturer String (up to 23 UNICODE characters)
USB String Descriptor–Product		
0x90	Blength	String descriptor length in bytes
0x91	BDescriptorType	Descriptor Type (String)
0x92-0xBF	Bstring	Product String (up to 23 UNICODE characters)
USB String Descriptor–Serial Number		
0xC0	Blength	String descriptor length in bytes
0xC1	BDescriptorType	Descriptor Type (String)
0xC2-0xEF	Bstring	Serial Number String (up to 23 UNICODE characters)
0xF0-0x1FF	Reserved	

The *NandMfg.exe* fills this information; the firmware design loads this information during power up. When the NAND Flash is blank, the default VID and PID is used and all the string descriptors are disabled.

7.1 Building Configuration nx2 File

The *BldNx2.exe* program utility is used to build the firmware configuration *.nx2* file. This file is used to program the NAND Flash by the *NandMfg.exe*. The BldNx2.exe requires two separate sets of firmware: 512-NAND firmware *inand_fw.hex* and either 2K-NAND firmware, *nand_mc2k.hex* or 4K-NAND firmware, *nand_mc4k.hex*.



NandMfg.exe is the program that supports the NX2LP-Flex NAND programming. It is an application specifically designed to download vendor configuration and NAND firmware configuration .nx2 parameters to the NAND Flash device and automatically initialize it with FAT32 formatting (formatting is disabled for 4K-NAND type).

This utility can be used to program previously unprogrammed or blank NAND Flash devices. It can also be used to reprogram NAND Flash devices that are already configured and enumerated as Windows Mass Storage Class devices. The application is USB Plug and Play aware, meaning that it automatically detects the presence of a usable NX2LP-Flex device on the USB bus. Simply connect the NX2LP-Flex device to the PC's USB bus and run the NX2LP NAND Programming Utility. The status-bar at the bottom of the application window displays the identified device. The current selections for all parameter fields are stored each time the program closes. These are then restored each time the program runs. More details about this utility is available in the *NandMfg.exe* in **Help** > **User's Guide**.

After creating default.nx2 via the *BldNx2.exe*, use *NandMfg.exe* in **File** > **Select Configuration** to pick up the new firmware configuration. The **Program Device** button can be used to program the NAND Flash.

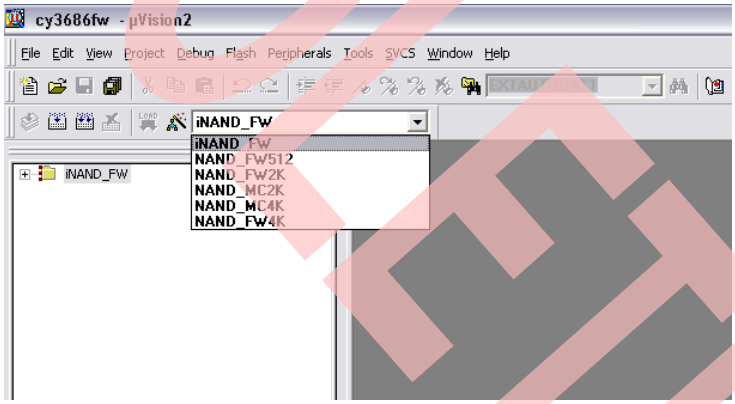
8 Compile Time Configuration Settings

The most common configuration settings are contained in the CY3686fw.Uv2 file available in the kit CD-ROM.

Additional command options and #defines control some of the compile-time settings used by the code. The major ones are explained in this section. To change some of these settings, right-click on the project name in uVision2 and select **Options for target**, then select the C51 tab. “inand.h” contains additional #defines that can be used to further customize the behavior of the firmware. These are located at the top of the “inand.h” file.

The CY3686fw.Uv2 file has six options:

- inand_fw: This option supports NX2LP-Flex silicon for 512-NAND type firmware
- nand_mc2k: This option supports NX2LP-Flex silicon for 2K-NAND type firmware
- nand_fw512: This option supports FX2LP silicon for the 512-NAND type firmware. **Note** The NAND Flash Boot-Loader EEPROM needs to be removed.
- nand_fw2k: This option supports FX2LP silicon for the 2K-NAND type firmware. **Note** The NAND Flash Boot-Loader EEPROM needs to be removed.
- nand_mc4k: This option supports NX2LP-Flex silicon for 4K-NAND type firmware
- nand_fw4K: This option supports FX2LP silicon for the 4K-NAND type firmware. **Note** The NAND Flash Boot-Loader EEPROM needs to be removed.



For more target platforms, you can create defines in the “options for target” tab; this allows you to create multiple targets with different #defines. This is useful if you have multiple targets (such as debug versus production).

Define flags for C51 in target options

If you enable flag NAND_2K, it compiles the firmware to support only 2K-page NAND type. To support 4K-page NAND type, enable both NAND_2K and NAND_4K flags. When you select the inand_fw target, it disables the NAND_2K and NAND_4K flags.

ZONE8

This flag is used to force detection of eight zones. Sometimes the detection of zones for an un-programmed 8-zone NAND device is incorrect. This flag helps solve data integrity issues arising out of such a situation.

Default Setting: disable

NO_WP

If you enable this flag, it removes the write-protect check option in the firmware source code. The firmware in this case only supports the removable drive. Enable this flag, it reduces some code space.

Default Setting: disable

NX2LP

This flag enables the selection of NX2LP-Flex or the FX2LP silicon. When you select inand_fw, nand_mc2k, or nand_mc4k, this flag is enabled. This flag changes the interrupt location from 0x000 to 0x500. It also produces all the output files to have the start-address at 0x500 instead of 0x000.

Default Setting: 0

HID

This flag enables the support for multiple interfaces. The firmware shows up as the Composite device that has the USB Mass Storage device class and HID class.

Default Setting: disable Note: Both C51 and A51 must enable the HID flag

USE_2LUN

This flag enables multiple LUNs support. The firmware supports two LUN devices. The PC desktop shows two "Drive" letters.

Default Setting: disable

Note: Unsupported in 4K-page firmware

USE_2NAND

This flag enables the selection of either the 2-NAND chips select or 8-NAND chips select. When this flag is enabled, it configures Port-E pin 2 to pin 7 as the GPIO pin.

Default Setting: disable

Note: Unsupported in 4K-page firmware

DEBUG

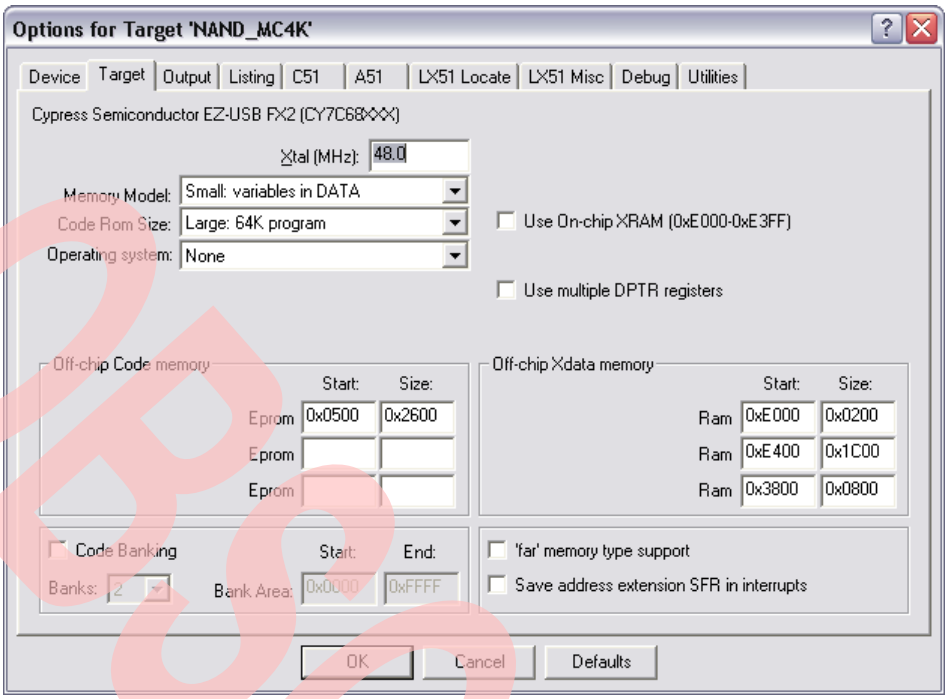
This flag enables run-time debugging and the dumping of data at locations 0xE1E0–0xE1FF for firmware debugging.

Default Setting: disable

8.1 Firmware Code Size

The current firmware default code size is 2K (gLog2Phy) + 8.25K; it is based on the Keil Professional compiler version. Other full versions of the Keil compiler are slightly larger. The memory map of the NX2LP-Flex silicon is divided into the following:

IROM: 0x0000 – 0x04FF	NAND Boot Loader area: 1.25K
IRAM: 0x0500 – 0x25FF	NAND Firmware code area: 8.25K
IRAM: 0x2600 – 0x37FF	Available user's code area: 4.5K
IRAM: 0x3800 – 0x3FFF	gLog2Phy table: 2K



It is possible that your code area is larger than 4.5K bytes. The choices available are:

- If the application can be independently executed, then partition these applications into multiple firmware images. The firmware images can be loaded depending on the mode of operation.
  - For example, in the MP3 player application, the MP3 application itself can be run even if the MP3 player is not connected to the USB host system.
  - When this device connects to the USB host, only the NAND firmware is loaded.
- Code banking support. The Keil compiler supports code banking, which can be found in the following example code in the following directories:
  - \Keil\C51\EXAMPLES\Bank_EX1
  - Detailed documentation for the code banking can be found in:

    - \Keil\C51\HLP\Release_Notes.htm

    - Example for C51: CREATING CODE BANKING PROGRAMS can be found in the following link: http://www.keil.com/support/docs/158.htm

If your code area is larger than 15K bytes, then code banking is required.

9 Building the Software

Make sure that you have installed the FX2LP-DVK and the NX2LP-Flex tools.

This Reference Design is too large to compile with the 4K-demo version of the Keil tools that is shipped with Cypress development kits.

Warnings

When the firmware is linked, it generates three warnings. These warnings are expected. The linker may have to run several iterations to optimize the code and may generate this list two or three times (and report six or nine warnings).

EZUSB_Delay is called from the timer0 ISR and from the background code. This is not an issue because any calls to the EZUSB_Delay function in the ISR are followed by a soft reset.

The EEPROM read and write routines are only used by the ISR during manufacturing and debugging operations. The background code is not active during these operations.

*** WARNING L15: MULTIPLE CALL TO FUNCTION

NAME: _EZUSB_DELAY/DELAY

CALLER1: ?C_C51STARTUP

CALLER2: ISRTIMER0/ATARESET

*** WARNING L15: MULTIPLE CALL TO FUNCTION

NAME: _EEPROMWRITEBLOCK/EEPROM

CALLER1: ?C_C51STARTUP

CALLER2: ISR_SUDAV/PERIPH

*** WARNING L15: MULTIPLE CALL TO FUNCTION

NAME: _EEPROMREAD/EEPROM

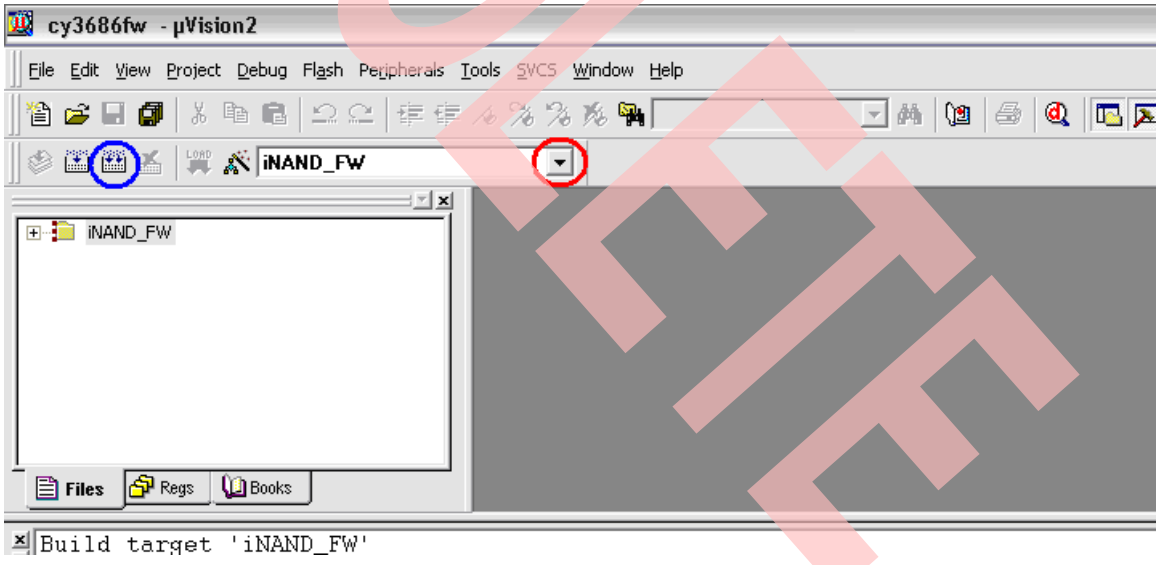
CALLER1: ?C_C51STARTUP

CALLER2: ISR_SUDAV/PERIPH

9.1 Rebuilding the 512-NAND Firmware for NX2LP-Flex Chip

To rebuild the 512-NAND firmware, follow these steps:

Click on *CY3686fw.uv2* to launch the NAND firmware project as follows.



Select **iNAND_FW** target from the drop-down menu (circled red).

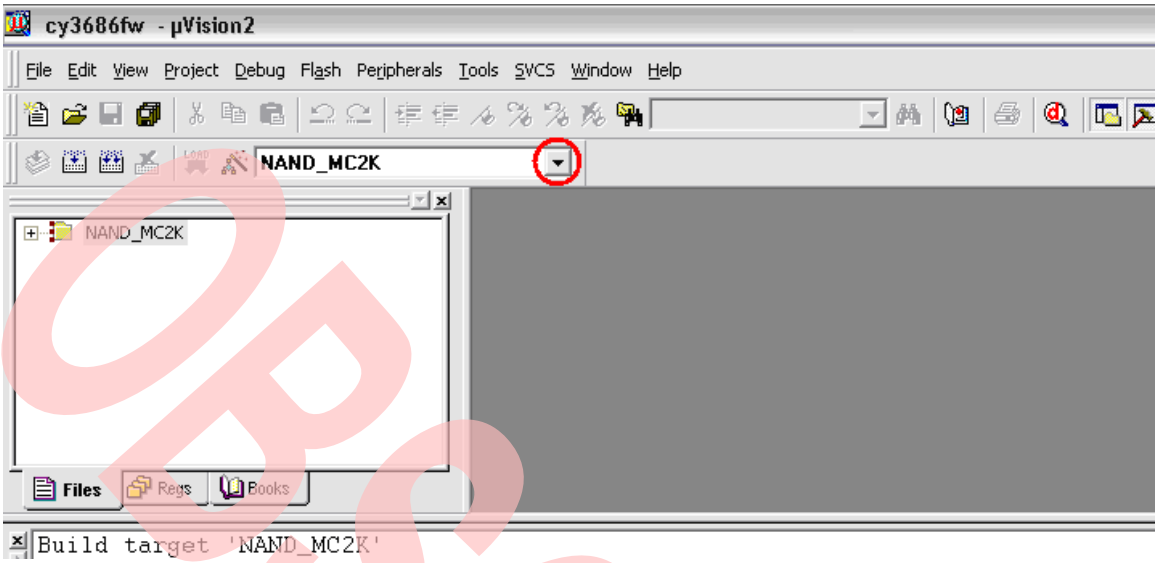
Press **F7** or the **Rebuild** button (circled blue) to rebuild the project, which creates two files.

- inand_fwhex (*BldNx2.exe* - use this file to build the .nx2 file for *NandMfg.exe*)
- inand_fw.iic (The *NandMfg.exe* - use CTRL-d for NAND Flash programming)

9.2 Rebuilding the 2K-NAND Firmware for NX2LP-Flex Chip

To rebuild the 2K-NAND firmware, follow these steps:

- Click on *CY3686fw.uv2* to launch the NAND firmware project

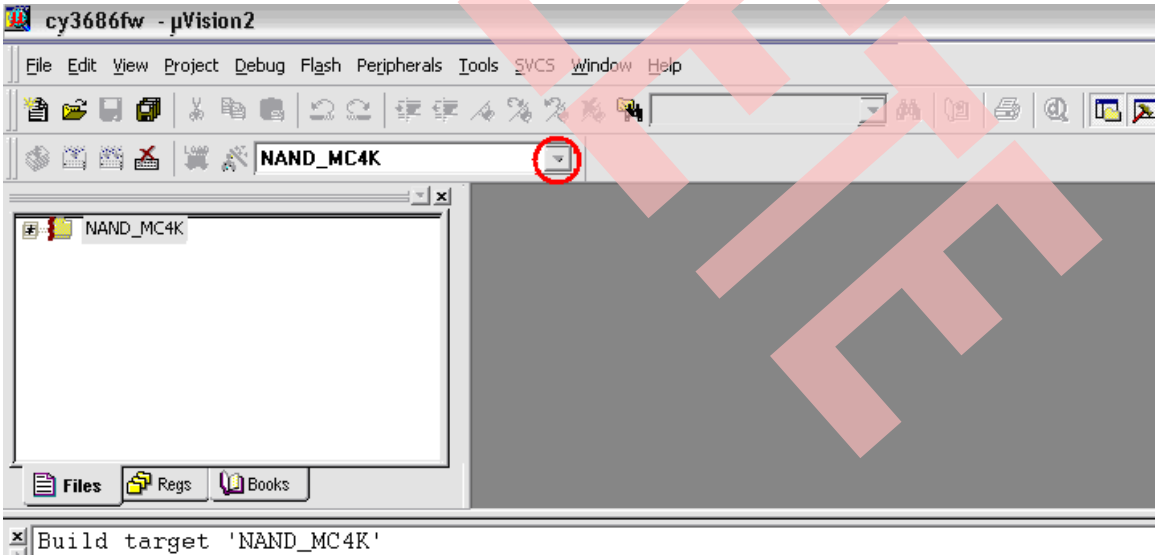


- Select **NAND_MC2K** target from the drop-down menu (circled red).
- Press **F7** to rebuild the project, which creates two files:
  - nand_mc2k.hex (*BldNx2.exe* uses this file to build the .nx2 file for *NandMfg.exe*)
  - nand_mc2k.iic (The *NandMfg.exe* uses CTRL-d for NAND Flash programming)

9.3 Rebuilding the 4K-NAND Firmware for NX2LP-Flex Chip

To rebuild the 4K-NAND firmware, follow these steps:

- Click on *CY3686fw.uv2*, which launches the NAND firmware project.



Select **NAND_MC4K** target from the drop-down menu (circled red).

Press **F7** to rebuild the project, which creates two files:

- nand_mc4k.hex (*BldNx2.exe* use this file to build the .nx2 file for *NandMfg.exe*)

- nand_mc4k.iic (The *NandMfg.exe* uses CTRL-d for NAND Flash programming)

9.4 Program the Firmware with NandMfg.exe CTRL-d Options

The *NandMfg.exe* accepts the program firmware output (.iic) file via CTRL-d. When using CTRL-d, the FAT32 format is disabled. Follow these steps:

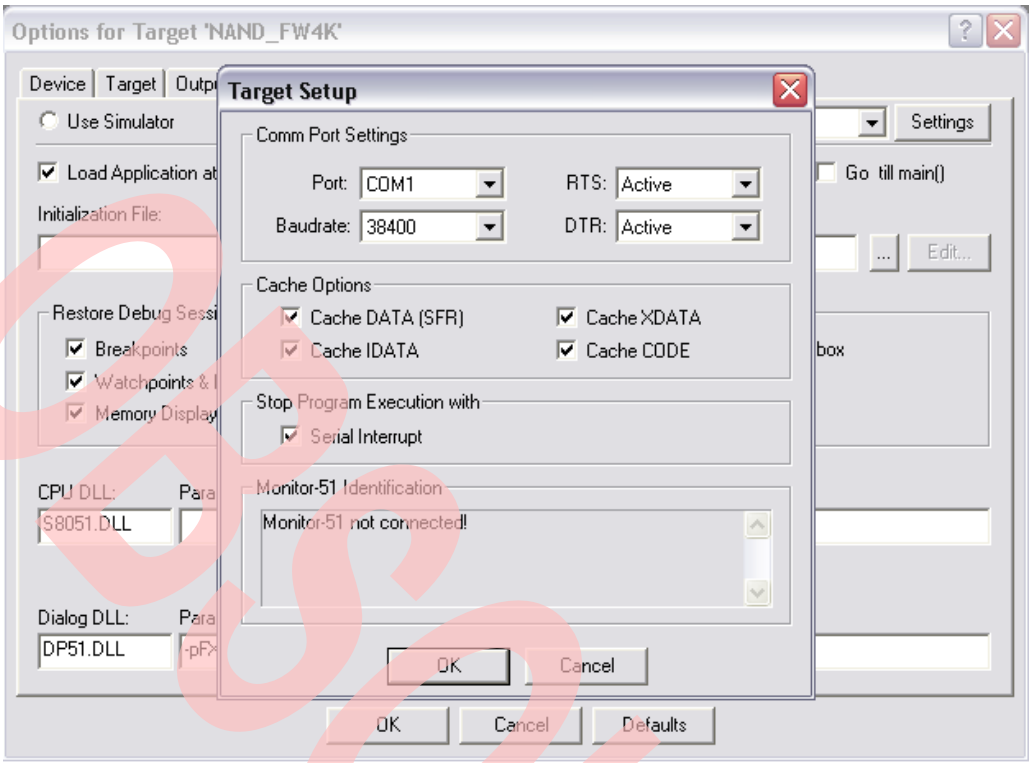
- Execute the *NandMfg.exe*
- Press **CTRL-d**. A popup box (File Explorer) appears.
- Browse to either *inand_fw.iic, nand_mc2k.iic,* or *nand_mc4k.iic* depending on the NAND Flash device you have in the socket.
- When the file is opened, *NandMfg.exe* automatically programs the NAND Flash; if successful, the status message **Programming complete** appears.

9.5 Using a CY3684 Board to Debug Code

The CY3686 software runs with the Keil debugger on the FX2LP development board (CY3684). This is a useful environment to debug startup issues by single stepping the firmware. Note that the following changes to the CY3684 board are necessary:

- SW1: select SMALL EEPROM.
- SW2: Select EEPROM.
- Connect the RS232 cable from SIO-1 to the PC COM port.
- Connect the USB cable from J1 to the PC USB port.
- Click on the CY3686fw.uv2 project file to launch the Keil development software.
- If using 2K-NAND firmware, select NAND_FW2K; for 512-NAND firmware, select NAND_FW512 to debug the firmware using the FX2LP mode.

  For example, if using 4K-NAND firmware, move the mouse to **NAND_FW4K Project Workspace;** right-click and select **Options for the Target 'NAND_FW4K'**; click on **Debug** > **Setting**. The following menu is displayed.



Notes

- The port is set to “COM1”, baud rate: 38400. You may need to change the COM setting to the available port on the PC.
- When connecting the USB cable to the PC, the LED D7 green light is in ON state.

10 Debugging Without the Mass Storage Driver

Debugging specific commands requires a different approach because the Mass Storage driver times out while you are single stepping and it can lock or reboot the host machine. The CY3686 firmware can be bound to the Cyusb generic driver; to do so, follow these steps:

- SW1: select SMALL EEPROM.
- SW2: Select EEPROM.
- Turn on power to your board. The board enumerates and binds to the generic driver (CYUSB.sys).
- Connect the USB cable from J1 to the PC USB port.
- Open CYCONSOLE.EXE and select options/EZ-USB interface.
- Press the Load Mon button.
- Start the Keil debugger and download your firmware via the Keil debugger.
- After the firmware is bound to the generic driver, commands can be sent to the device using the control panel. An easy way to do this is to construct a file containing the command and use the FileTrans button to send it.
- Start the Keil debugger; download your firmware.
- Run the firmware; it enumerates and binds to the general purpose driver.
- Start the control panel.
- Do a “get pipes” on the control panel. This fills in the pipe fields.

- Select the OUT pipe and press the FileTrans button.
- Select your command file.
- Manually transfer the IN or OUT data required by the command.
- Do a final IN to collect the CSW.

10.1 Windows Boot Support

The current level of boot functionality allows you to boot to DOS or Win9x Safe Mode from a hard drive or from CD-ROM. You cannot currently boot to Windows due to issues with the way Windows attempts to access a boot drive directly. Boot functionality has been tested with both Phoenix and AMI BIOS.

10.2 48-Bit LBA Addressing

This reference design supports the 48-bit addressing method. However, the SCSI commands passed by the Mass Storage Class Specification only support 32-bit LBAs, which limits support to 2^41 (2Tera) bytes on a 512-byte sectored device (512 and 2K page firmware) and 2^42(4Tera) bytes on a 1024-byte sectored device (4K page firmware).

11 How This Design Uses GPIF

The NX2LP-Flex design takes advantage of its internal GPIF (General Programmable Interface) to move data from the endpoint buffers to the mass storage device. The NAND waveforms are created using the GPIF Designer tool. The installation application of this program is located in the CDROM at: \GPIF_Designer\Installer_GPIF_Designer.exe. The file “gpif30.gpf” contains the waveform project file that generates the following:

WaveDataPioUDMA: This waveform loads into the GPIF memory at 0xE400 // see gpif30.gpf for timing info and gpif.c

```
const char code WaveDataPioUDMA[128] =
{
// offset e403=2=60ns, 1=40ns (TransferSize need to adjust -1)
// Wave 0
/* LenBr */ 0x08, 0x12, 0x2D, 0x02, 0x33, 0x33, 0x3F, 0x07,
/* Opcode*/ 0x01, 0x01, 0x01, 0x06, 0x01, 0x01, 0x07, 0x00,
/* Output*/ 0x07, 0x01, 0x01, 0x01, 0x07, 0x07, 0x07, 0x07,
/* LFun */ 0x01, 0x00, 0x00, 0x00, 0x6E, 0x6E, 0x00, 0x3F,
// Wave 1
// Write waveform alway support 60ns: offset e423=2=60ns, 1=40ns
/* LenBr */ 0x02, 0x0A, 0x2B, 0x02, 0x0B, 0x3F, 0x01, 0x07,
/* Opcode*/ 0x02, 0x03, 0x03, 0x06, 0x03, 0x07, 0x02, 0x00,
/* Output*/ 0x06, 0x07, 0x07, 0x06, 0x07, 0x07, 0x07, 0x07,
/* LFun */ 0x00, 0xEE, 0x2D, 0x00, 0x6E, 0x40, 0x00, 0x3F,
// Wave 2
/* LenBr */ 0x02, 0x3F, 0x01, 0x01, 0x01, 0x01, 0x01, 0x07,
/* Opcode*/ 0x00, 0x23, 0x22, 0x02, 0x02, 0x02, 0x02, 0x00,
/* Output*/ 0x01, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07,
/* LFun */ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3F,
// Wave 3
/* LenBr */ 0x02, 0x3F, 0x01, 0x01, 0x01, 0x01, 0x01, 0x07,
/* Opcode*/ 0x02, 0x03, 0x02, 0x02, 0x02, 0x02, 0x02, 0x00,
/* Output*/ 0x06, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07,
/* LFun */ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3F,
};
```

There are four wave forms: Wave 0-3:

- Wave 0: This waveform generates the NAND Flash Data Read cycles. The subroutine nReadPages uses this waveform to read NAND Flash data and the GPIF engine sends this data directly to the USB host. In this waveform, the GPIF Counter (GPIF_CB0-1) is used.

- Wave 1: This waveform generates the NAND Flash Data Write cycles. The subroutine nWritePages uses this waveform to write data to the NAND Flash and the GPIF engine receives this data from the USB host and store it into the Endpoint 2 FIFO. In this waveform the GPIF Counter (GPIF_CB0-1) is used.
- Wave 2: This waveform generates the NAND Flash Single Write cycle. The firmware uses this for all NAND commands and NAND Address write cycles.
- Wave 3: This waveform generates the NAND Flash Single Read cycle. The firmware uses this to get NAND status cycle.

11.1 GPIF Waves

Figure 2. Wave 0: (FIFORd)

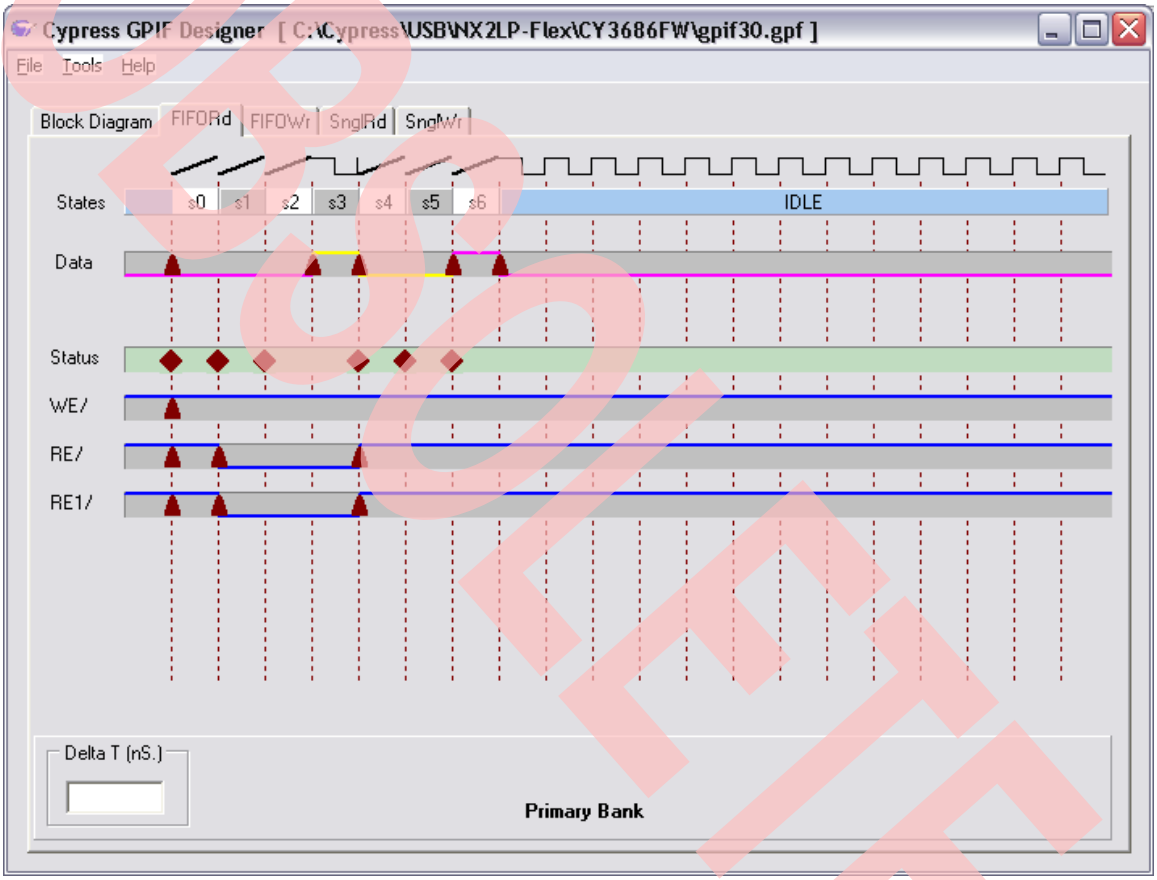


Figure 3. Wave 1: (FIFOWr)

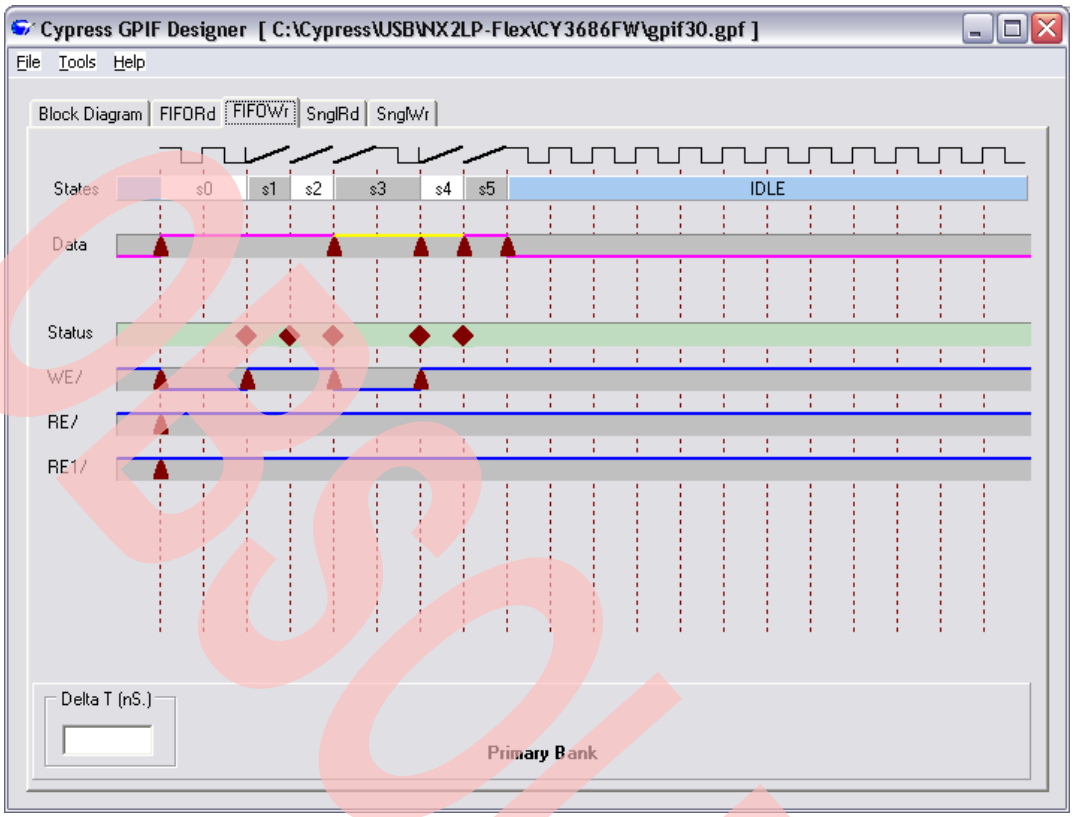
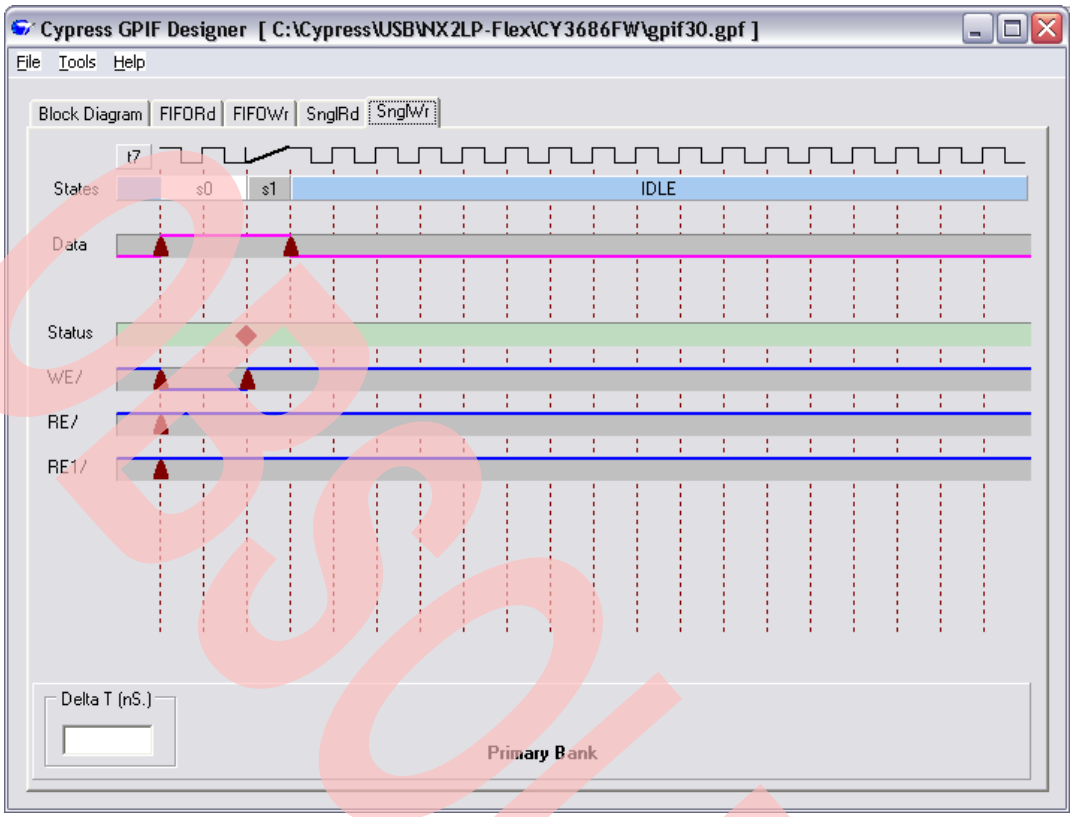


Figure 4. Wave 2: (SngRd)

Figure 5. Wave 3: (SngWr)



Notes

- For more information about the GPIF Designer, see the “Cypress GPIF Designer” application help.
- To learn about internal state, consult EZ-USB Technical Reference Manual for FX2LP, Revision 1.2.
- To view the internal state of these waveforms, click **gpif30.gpf**.

12 References

- USB Mass Storage Class – Bulk Only Transport, USB Mass Storage DWG.
- USB Mass Storage Class – Overview Specification, USB Mass Storage DWG.
- USB Specification – Revision 2.0
- EZ-USB Technical Reference Manual for FX2LP, Revision 1.2
- NX2LP-Flex Data Sheet
- SCSI-3 Specification

13 Additional Resources

- NX2LP Development Kit CY3686
- NX2LP Compatible NAND list
- FX2LP Development Kit CY3684
- USB 2.0 to ATA/CF Reference Design –CY4611B
- Mass Storage + Keyboard with NX2LP
- Interfacing Cypress CY3686 DVK to NAND Flash Memory with Four Chip Selects

About the Author

Name: Gayathri Vasudevan
Title: Applications Engineer

Document History

Document Title: AN61347 - NX2LP-Flex USB to NAND Flash Firmware Design Notes

Document Number: 001-61347

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	2932174	SHAH	05/14/2010	New application note
*A	3183550	SHAH	02/27/2011	Updates from review/audit.
*B	3999840	GAYA	05/14/2013	Updated in new template. Completing Sunset Review.
*C	5314780	GAYA	06/20/2016	Updated template Obsoleting the document

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Lighting & Power Control	cypress.com/powerpsoc
Memory	cypress.com/memory
PSoC	cypress.com/psoc
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless/RF	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

Cypress Developer Community

Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

Phone : 408-943-2600
Fax : 408-943-4730
Website : www.cypress.com