Application note

NAND FLASH Programming with PEEDI

Version 1.0

RONETIX
embedded development tools

Preface

NAND FLASH memory chips are considerably cheaper then their elder brothers – the NOR FLASH chips. So NAND chips started to push away NOR chips in embedded devices and now it is possible to build a device without a NOR FLASH – only using a NAND flash which is used to hold the bootloader and all other executable code that is copied and executed from RAM.

PEEDI is able to program all NAND chips with 8 and 16 bits bus and up to 8Gbits of size.

Programming NAND FLASH chips is easy. The first thing to do is to configure PEEDI to work with the target. This includes making the target initialization section of the PEEDI configuration file and setting the FLASH section.

Hardware requirements

Generally the NAND Flash programming is CPU independent, but the given target must comply with the following hardware requirements:

The CPU must have a direct access to the NAND Flash chip:

1. To be able to write/read data from/to the NAND Flash with one memory transaction command.

2. To be able to write ADDRESSES and COMMANDS (ALE and CLE) to the NAND Flash with one memory transaction command.

3. To be able to ASSERT and RELEASE the chip select of the NAND Flash with one memory write command

PEEDI doesn't use the Read/Busy signal.

PEEDI configuration

The INIT section of the config file must include the initialization for the chip select and the GPIOs, because the Flash Programmer doesn't make any initialization.

Here is an example of an INIT section for a AT91M55800:

```
[INIT_AT91M55800]
memory write 0xFFFF4020 0x004F0002 ; enable main clock (4MHz)
wait 10                            ; wait to stabilize
memory write 0xFFFF4020 0x004F4002 ; switch to main clock
memory write 0xFFFF4020 0x3F006102 ; enable PLL (MUL=33)
wait 10                            ; wait to lock
memory write 0xFFFF4020 0x3F00A122 ; switch to PLL (DIV=4,MUL=33;MCK=33MHz)
memory write 0xFFE00020 0x00000001 ; Cancel reset remapping
memory write 0xFFE00000 0x004020A5 ; csr0 - Flash at 0x400000, 2 Ws
memory write 0xFFE00004 0x00C03029 ; csr1 - SRAM  at 0xC00000, 2 Ws
memory write 0xFFFFF124 0xFFFFFFFF ; disable all interrupts

; setup for the NAND Flash: PB13 - Chip Select
mem write 0xFFFF4010 0x6000        ; enable periph. clock for PIOA and PIOB
mem write 0xFFFF0000 0x2000        ; PIOB_PER - enable PIO13
mem write 0xFFFF0010 0x2000        ; PIOB_OER - configure as output
mem write 0xFFFF0054 0x2000        ; PIOB_MDDR - disable open drain
mem write 0xFFFF0030 0x2000        ; set PB13

; setup CS2 (chip select 2)
mem write 0xFFE00008 0x3002126     ; CSR2 - 8-bis, 2 WS
```

The FLASH section gives information about the base address of the DATA, ADDR and COMMANDS.

It also gives a way to assert and release the NAND FLASH chip select.

```
[NAND_FLASH]
CHIP        = NAND_FLASH
DATA_BASE   = 0x03000000                    ; data
```

```
CMD_BASE    = 0x03400000                    ; commands (CLE)
ADDR_BASE   = 0x03200000                    ; addresses (ALE)
FILE        = "test.bin", 0x0

; address and value for asserting the Nand Flash chip select
; [addr] = value
CS_ASSERT   = 0xFFFF0034, 0x2000            ; PB13 <- 0
; address and value for releasing the Nand Flash Chip select
; [addr] = value
CS_RELEASE  = 0xFFFF0030, 0x2000            ; PB13 <- 1

; list of blocks to be marked as bad
;BAD_BLOCKS = 4, 27, 1002

; CAUTION!!!
; Enable erasing of bad blocks
; DO NOT Enable this if you don't know what you are doing
; For more information see the AN006 (www.ronetix.at/an006.html)
ERASE_BAD_BLOCKS = NO

OOB_INFO = JFFS2  ; how to deal with the oob (spare) info
                  ;   RAW   - program 528/2112 bytes from file
                  ;   JFFS2 - program 512/2048 bytes from file and add ECC
bytes
                  ;   FF    - program 512/2048 bytes from file, the spare
info is 0xFF
```

See at91sam9261_pm9261.cfg and pxa270_custom.cfg from our example configuration:

http://download.ronetix.at/peedi/peedi_config_files.zip

Bad block management

The NAND Flash devices may have blocks that are invalid when they are shipped.

An invalid block is one that contains one or more bad bits. Additional bad blocks may develop with use.

The factory identifies invalid blocks before shipping by programming data other than FFh (x8) or FFFFh (x16) into the first spare location of the first or second page of each bad block.

PEEDI automatically detects the bad blocks and reports them:

```
peedi> flash info

FLASH configuration for core #0:
 Nand Flash: ID = 0x75, 8-bit,  32 MB
   page size            = 512 + 16 bytes
   pages per block      = 32
   number of blocks     = 2048
   number of bad blocks = 2 (use "flash query" to get the list)
   erase bad blocks     = true

peedi> flash query
Total blocks = 2048
Bad   blocks = 2
Bad blocks list:
 1289
 1291

peedi>
```

Once detected, the bad blocks are protected against erasing and programming.

On demand, PEEDI can be forced to try to erase the existing bad blocks.

It is also possible to force blocks as bad.

To erase all blocks including the bad blocks the following parameter must be changed:

```
ERASE_BAD_BLOCKS = YES
```

After PEEDI restart, the command "flash erase" will erase all blocks.



WARNING:

If you erase blocks factory marked as bad, there is now way to detect which were the bad blocks.

Make sure you have saved the output of the “flash query” command so you can mark again the bad blocks as bad.

To force marking of blocks 4, 27 and 1002 as bad:

```
BAD_BLOCKS = 4, 27, 1002
```

After PEEDI restart the "flash info" command will mark the given blocks as bad.

Once marked as bad, the blocks a not marked anymore.

Programming

Now everything is configured and you should be ready to program your NAND chip.

PEEDI supports direct programming of JFFS2 images to the NAND flash. For this, the OOB_INFO parameter must be set to ‘JFFS2’. This way PEEDI will the write the data bytes loading from the image file and will calculate the ECC and program it to the OBB/spare bytes. NAND FLASH chips with 512 bytes page size have 6 ECC bytes for each page, the chips with 2048 bytes page size have 24 ECC bytes. PEEDI supports only BIN images starting from address 0. When programming the image bad blocks will be just skipped and left un-programmed. They will not affect the block count order.

If you use a custom file system, using PEEDI you can program a bootloader to the NAND chip, that will gain the control of the system after it is rebooted and could handle the programming of the left empty NAND FLASH chip, considering the NAND file system you use and the bad block present in the given target.



WARNING:

The PEEDI TFTP client uses fixed 512 bytes transfer block size, which limits the size of the image file to 32MB. If your file is bigger, use HTTP or FTP file server or use MMC/SD card to store the file and put it on PEEDI.