MTD NAND Solution Guide

v1.06

Publication Release Date: Jun, 2015

Contents

1. Introduction

A Memory Technology Device (MTD) is a type of device file in Linux for interacting with flash memory. The MTD subsystem was created to provide an abstraction layer between the hardware-specific device drivers and higher-level applications.

Some resources for you reference:

(1) YAFFS2
http://en.wikipedia.org/wiki/YAFFS

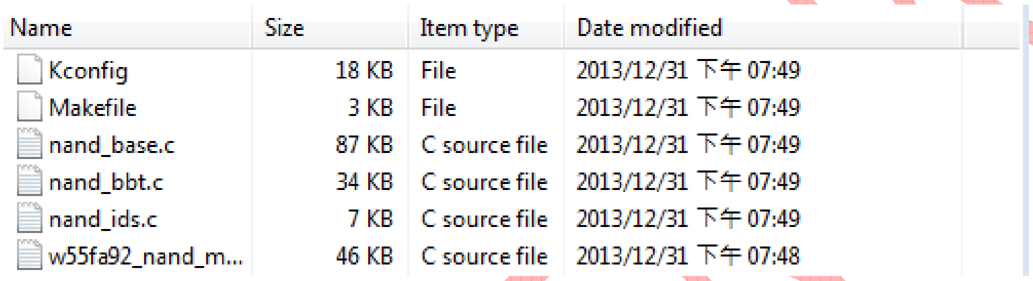
(2) UBIFS
http://en.wikipedia.org/wiki/UBIFS
http://www.linux-mtd.infradead.org/doc/ubifs.html#L_overview

(3) FLASH FILE SYSTEMS FOR MTD DEVICES: JFFS2, YAFFS2, UBIFS AND LOGFS.
http://www.lindusembedded.com/blog/2010/05/20/flash-file-systems-for-mtd-devices-jffs2-yaffs2-ubifs-and-logfs/

2. How to update source files

2.1.To update NAND controller driver

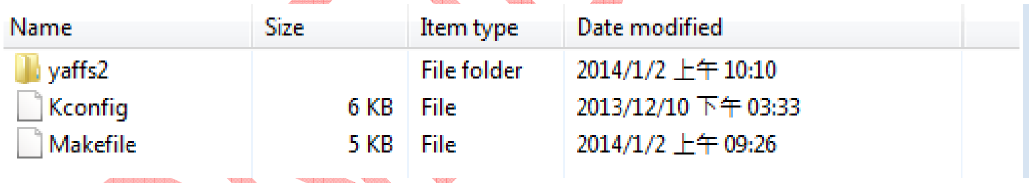
Update these files to the path as <BSP>\linux_2.6.35.4_fa92\drivers\mtd\nand\



Name	Size	Item type	Date modified
Kconfig	18 KB	File	2013/12/31 下午 07:49
Makefile	3 KB	File	2013/12/31 下午 07:49
nand_base.c	87 KB	C source file	2013/12/31 下午 07:49
nand_bbt.c	34 KB	C source file	2013/12/31 下午 07:49
nand_ids.c	7 KB	C source file	2013/12/31 下午 07:49
w55fa92_nand_m...	46 KB	C source file	2013/12/31 下午 07:48

2.2.To update YAFFS2

Copy these files to the path as <BSP>\linux_2.6.35.4_fa92\fs\



Name	Size	Item type	Date modified
yaffs2		File folder	2014/1/2 上午 10:10
Kconfig	6 KB	File	2013/12/10 下午 03:33
Makefile	5 KB	File	2014/1/2 上午 09:26

3. How to apply in kernel configuration

3.1. Built-in MTD NAND Controller driver

In Linux kernel menu configuration, you must to select options as follows in Linux kernel menu configuration.

```
     Device Drivers  --->
          <*> Memory Technology Device (MTD) support  --->
               [*]   MTD partitioning support
               <*>  Direct char device access to MTD devices
               -*-   Common interface to block layer for MTD 'translation layers
               <*>  Caching block device access to MTD devices
               <*>  NAND Device Support  --->
                    <*>   Support for NAND on Nuvoton W55FA92 demo board
```

Notice:

(1) The driver conflicts with NAND driver based on SCSI layer.

OOB layout on NAND flash:

Bad block marker	User available	ECC parity code
Always 4-Bytes	Reserved	The size depends on BCH algorithm and OOB size.

3.2. How to enable UBIFS

If you prefer UBIFS, you can select options as follows in Linux kernel menu configuration.

```
     Device Drivers  -→
        <*> Memory Technology Device (MTD) support  --->
          UBI - Unsorted block images  --->
            <*> Enable UBI
            (4096) UBI wear-leveling threshold
            (5) Percentage of reserved eraseblocks for bad eraseblocks hand
     File systems  --->
          [*] Miscellaneous filesystems  --->
            <*>    UBIFS file system support
            [*]     Extended attributes support
            [*]     Advanced compression options
            [*]       LZO compression support
            [*]       ZLIB compression support
```

3.3.How to enable YAFFS2

If you prefer YAFFS2, you can select as below options in Linux kernel menu configuration.

```
     File systems  --->
          [*] Miscellaneous filesystems  --->
            <*>   yaffs2 file system support
            -*-      512 byte / page devices
            -*-      2048 byte (or larger) / page devices
            [*]        Autoselect yaffs2 format
            [*]      Enable yaffs2 xattr support
```

3.4.How to enable VPOST Init By Driver

If you prefer to enable LCM driver, please also need to set “vpost init by driver” as below options in Linux kernel menu configuration.

```
     Device Drivers  --->
          Graphics support  --->
            --- Support for frame buffer devices
            [*]   W55FA92 LCD framebuffer support
                 Select VPOST init setting (vpost init by driver)  --->
```

4. How to layout MTD partition table

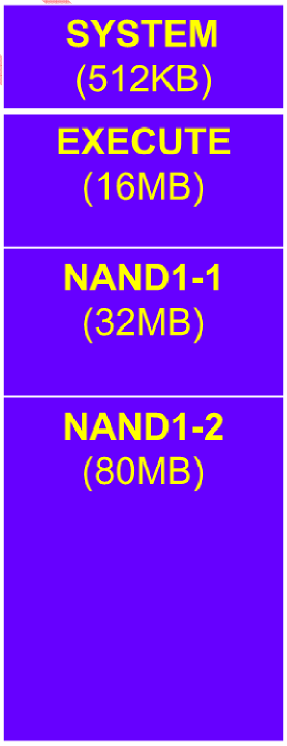
You can modify the MTD partition table in drivers/mtd/nand/w55fa92_nand_mtd.c.
For example, we layout four partitions in the table as:

(1) The **SYSTEM** partition size always is fixed to 4-block by default. The driver will calculate automatically after identifying what flash on board. It is used to save NAND loader and booting information.

(2) The **EXECUTE** partition size is decided by user. It is used to save Linux kernel and logo part.

(3) The **NAND1-1/NAND1-2** is optional. The partition size is decided by user. It is used to save user-data based on file system.

```
/* Define your partitions */
static const struct mtd_partition partitions[]
= {
 {
    .name = "SYSTEM",
    .offset = 0,
    .ecclayout = &w55fa92_nand_SYSTEM_oob
 },
 {
    .name = "EXECUTE",
    .offset = MTDPART_OFS_APPEND,
    .ecclayout = &w55fa92_nand_EXECUTE_oob
    .size = 16 * 1024 * 1024,
 },
 {
    .name = "NAND1-1",
    .offset = MTDPART_OFS_APPEND,
    .size = 32 * 1024 * 1024
 },
 {
    .name = "NAND1-2",
    .offset = MTDPART_OFS_APPEND,
    .size = MTDPART_SIZ_FULL
 }
};
```



Notice: Please keep SYSTEM and EXECUTE partition in the table. They will be used for System-area update.

5. How to make image on PC

5.1. To make YAFFS2 image

We ported YAFFS2 utility from YAFFS2 source pack for making image. The usage is as follows:

```
#mkyaffs2image: image building tool for YAFFS2 built Jan  2 2014
#Missing <dir> and/or <image.yaffs2> argument
#Usage: mkyaffs2image [Options] <dir> <image_file>
#          <dir>         the directory tree to be converted
#          <image.yaffs2> the output file to hold the image
#Options:
#  -c | --convert       produce a big-endian image from a little-endian machine
#  -p | --page-size      Page size of target NAND device [2048]
#  -o | --oob-size       OOB size of target NAND device [64]
#  -b | --block-size     Block size of target NAND device [0x20000]
#  -i | --inabnd-tags    Use Inband Tags
#  -n | --no-pad          Do not pad to end of block
```

If the page size of NAND flash is 2K and oobsize is 64, you can execute the command:

```
# Make an YAFFS2 image with inband-tags (blocksize=128KB, pagesize=2048, oobsize=64)
./mkyaffs2image -i ./foo ./foo_yaffs2.img
```

For 4K+224 NAND example:

```
# Make an YAFFS2 image with inband-tags (blocksize=1MB, pagesize=4096, oobsize=224)
./mkyaffs2image -b 0x100000 -p 4096 -o 224 -i ./foo ./foo_yaffs2.4k.img
```

5.2. To make UBIFS image

If the page size of NAND flash is 2K, you can execute the command:

Step 1: Make a UBIFS file system image from an existing directory tree

```
Usage: mkfs.ubifs [OPTIONS] target
-r, -d, --root=DIR      build file system from directory DIR
-e, --leb-size=SIZE     logical erase block size
-c, --max-leb-cnt=COUNT  maximum logical erase block count
-o, --output=FILE       output to FILE

# Make an UBIFS image 2K
mkfs.ubifs -r /tmp/foo -m 2048 -e 126976 -c 992 -o ./foo_ubifs
```

Step 2: To generate a UBIFS images.

```
Usage: ubinize [-o filename] [-p <bytes>] [-m <bytes>] [-s <bytes>] [-O <num>] [-e
<num>]
            [-x <num>] [-Q <num>] [-v] [-h] [-V] [--output=<filename>]
     [--peb-size=<bytes>]
            [--min-io-size=<bytes>] [--sub-page-size=<bytes>]
     [--vid-hdr-offset=<num>]
            [--erase-counter=<num>] [--ubi-ver=<num>] [--image-seq=<num>]
     [--verbose] [--help]
            [--version] ini-file

echo "[ubifs]" > ./ubifs.cfg
echo "mode=ubi" >> ./ubifs.cfg
echo "vol_id=0" >> ./ubifs.cfg
echo "vol_type=dynamic" >> ./ubifs.cfg
echo "vol_alignment=1" >> ./ubifs.cfg
echo "vol_name=foo" >> ./ubifs.cfg
echo "vol_flags=autoresize" >> ./ubifs.cfg
echo "image=foo_ubifs" >> ./ubifs.cfg

# Notice vol_size value < mtd partition size
# echo "vol_size=60MiB" >> ./ubifs.cfg

# Generate an UBIFS image 2K
ubinize -o ./foo_ubifs.img -m 2048 -p 128KiB -s 2048 ./ubifs.cfg
```

6. Some utilities

6.1.fs_writer

We provide some script files to introduce about “How to erase/burn/mount on N329 series platform”.

For YAFFS2 example,

```
#!/bin/sh
mkdir /mnt/yaffs2
./mtdutils/flash_erase /dev/mtd3 0 0
if ./mtdutils/nandwrite -a -m /dev/mtd3 foo_yaffs2.img > /dev/null; then
    if mount -t yaffs2 -o"inband-tags" /dev/mtdblock3 /mnt/yaffs2; then
        echo "Success"
        cat /mnt/yaffs2/hi.txt
        exit 0
    fi
fi
```

※ You can burn YAFFS2 image with inband-tags to NAND chip by Turbowriter.

For UBIFS example,

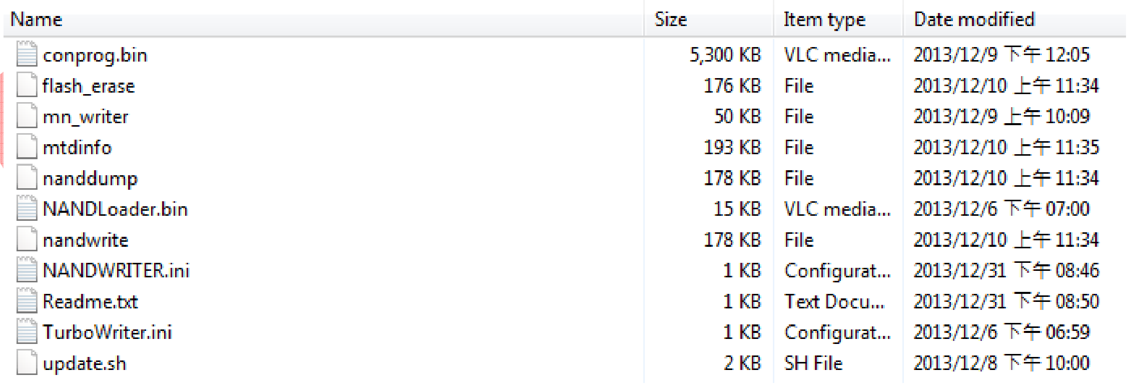
```
#!/bin/sh

PAGESIZE=2048
UBIFSIMGPATH=./ubifs/foo_ubifs.img

echo "mtdutils/flash_erase"
./mtdutils/flash_erase /dev/mtd2 0 0
echo "mtdutils/ubiformat"
if ./mtdutils/ubiformat /dev/mtd2 -O $PAGESIZE -s $PAGESIZE -f $UBIFSIMGPATH >
/dev/null 2>&1; then
    echo "mtdutils/ubiattach"
    if ./mtdutils/ubiattach /dev/ubi_ctrl -m 2 > /dev/null 2>&1; then
        mkdir -p /mnt/ubifs
        echo "mount"
        if mount -t ubifs ubi0_0 /mnt/ubifs > /dev/null 2>&1; then
            echo "Success!!"
            cat /mnt/ubifs/helloworld.txt
            exit 0
        fi
    fi
fi
./mtdutils/ubidetach -m 2
```

For mounting empty partition as UBIFS

```
#!/bin/sh

PAGESIZE=2048
PartitionSize=60MiB

./mtdutils/flash_erase /dev/mtd3 0 0
echo "mtdutils/ubiformat"
if ./mtdutils/ubiformat /dev/mtd3 -O $PAGESIZE -s $PAGESIZE > /dev/null 2>&1; then
    echo "mtdutils/ubiattach"
    if ./mtdutils/ubiattach /dev/ubi_ctrl -m 3 > /dev/null 2>&1; then
        echo "mtdutils/ubimkvol"
        if ./mtdutils/ubimkvol /dev/ubi0 -N test_volume -s $PartitionSize >
/dev/null 2>&1; then
            if [ ! -d "/mnt/ubifs" ]; then
                mkdir /mnt/ubifs
            fi
            if mount -t ubifs ubi0:test_volume /mnt/ubifs > /dev/null 2>&1; then
                echo "Success!!"
                exit 0
            fi
        fi
    fi
fi
./mtdutils/ubidetach -m 3
```

6.2.mn_writer

This tool can program system area parts which include SYSTEM and EXECUTE.



Name	Size	Item type	Date modified
conprog.bin	5,300 KB	VLC media...	2013/12/9 下午 12:05
flash_erase	176 KB	File	2013/12/10 上午 11:34
mn_writer	50 KB	File	2013/12/9 上午 10:09
mtdinfo	193 KB	File	2013/12/10 上午 11:35
nanddump	178 KB	File	2013/12/10 上午 11:34
NANDLoader.bin	15 KB	VLC media...	2013/12/6 下午 07:00
nandwrite	178 KB	File	2013/12/10 上午 11:34
NANDWRITER.ini	1 KB	Configurat...	2013/12/31 下午 08:46
Readme.txt	1 KB	Text Docu...	2013/12/31 下午 08:50
TurboWriter.ini	1 KB	Configurat...	2013/12/6 下午 06:59
update.sh	2 KB	SH File	2013/12/8 下午 10:00

For example:

Please define the binary file name in NANDWRITER.ini file. Include [NandLoader File Name], [Execute File Name], [Execute Image Number] and [Execute Image Address] items.

[NandLoader File Name]: It indicates NAND loader firmware of N329x series.
[Execute File Name]: It indicates executable firmware of N329x series.
[Execute Image Number]: How many backup number of executable firmware you want?
[Execute Image Address]: The Executable firmware will put in system memory. You can define the number. It is usually zero.
[Logo File Name]: (Optional) it indicates Logo image of N329x series. If you want to show Logo image on panel when system booting, you have to program Logo image file to NAND in advance by this option.

```
[NandLoader File Name]
// All file name length MUST <= 511 bytes
NANDLoader.bin

[Logo File Name]
NuvotonLogo_320x240.bin

[Execute File Name]
conprog.bin

[Execute Image Number]
// Backup count for Executing Image
1

[Execute Image Address]
// Execute address with Hex format for Executing Image
0
```

Finally, please run 'update.sh' to start.

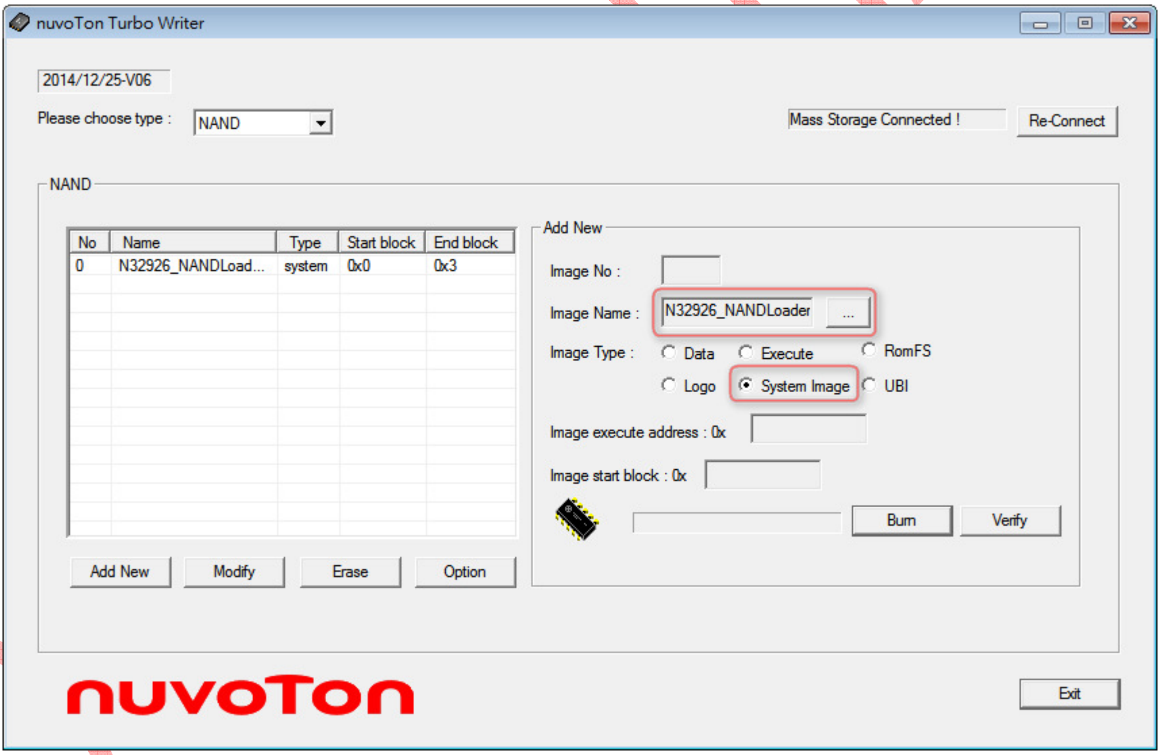
7. Example for MTD Solution

This chapter descripts the ways to program MTD into NAND flash chip step by step.

7.1. TurboWriter

You can program Nand Loader and Linux kernel into all new NAND flash by TurboWriter.

Step 1: Program Nand Loader into NAND flash as "System Image".


nuvoTon Turbo Writer

2014/12/25-V06

Please choose type : NAND

Mass Storage Connected !

No	Name	Type	Start block	End block
0	N32926_NANDLoad...	system	0x0	0x3

Add New

Image No :

Image Name : N32926_NANDLoader

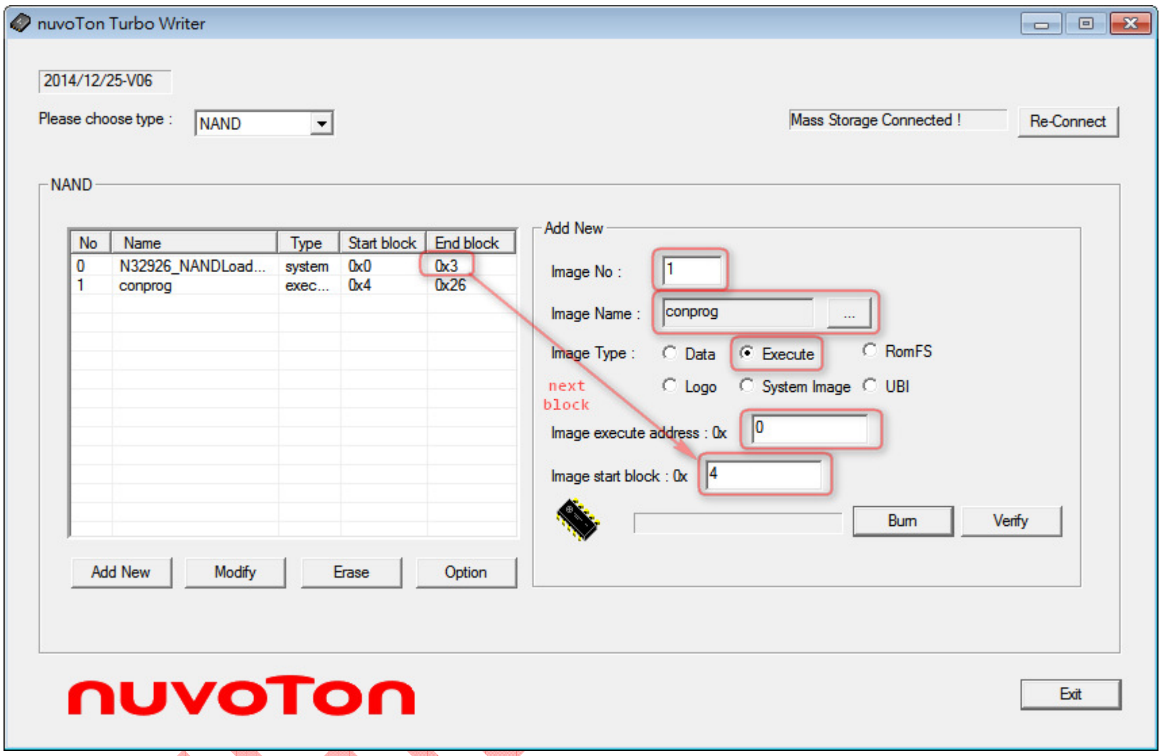
Image Type : ( ) Data ( ) Execute ( ) RomFS ( ) Logo (●) System Image ( ) UBI

Image execute address : 0x

Image start block : 0x

Step 2: Program Linux Kernel into NAND flash as "Execute". Please note that the "Image Execute Address" must be **0x0**.

After step 1 and step 2, you can reboot to run Linux kernel that support MTD device.



nuvoTon Turbo Writer

2014/12/25-V06

Please choose type : NAND

Mass Storage Connected !

NAND

No	Name	Type	Start block	End block
0	N32926_NANDLoad...	system	0x0	0x3
1	conprog	exec...	0x4	0x26

Add New

Image No : 1

Image Name : conprog

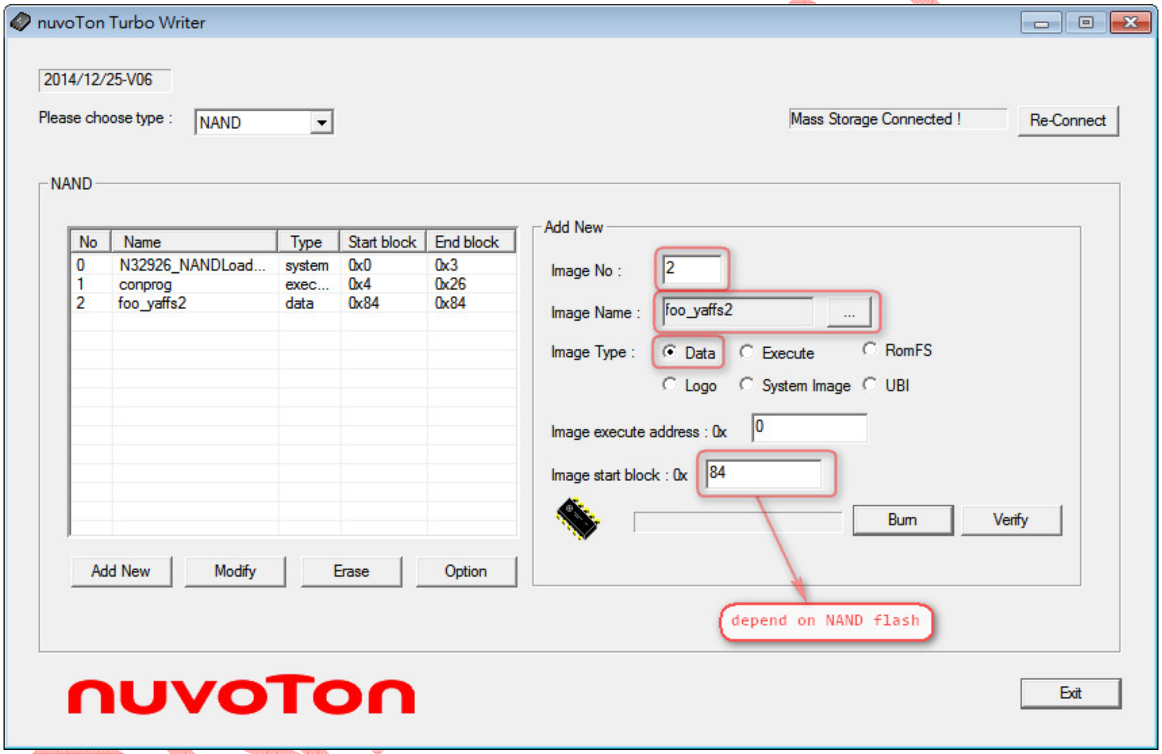
Image Type : Data [✓ Execute] RomFS Logo System Image UBI

next block

Image execute address : 0x 0

Image start block : 0x 4

Step 3: (Optional) Program YAFFS2 file system image into NAND flash as "**Data**". Please note that the "Image Start Block" must be **the start address of MTD partition NAND1-1 or NAND1-2**. For example, if we use the NAND flash with block size *128KB*, the block number of MTD partition SYSTEM is always 4, the size of MTD partition EXECUTE is always 16MB and occupy 16MB / block size = 128 blocks, so the "Image Start Block" is 4 + 128 = 132 = ***0x84***.



No	Name	Type	Start block	End block
0	N32926_NANDLoad...	system	0x0	0x3
1	conprog	exec...	0x4	0x26
2	foo_yaffs2	data	0x84	0x84

Add New

Image No : 2

Image Name : foo_yaffs2

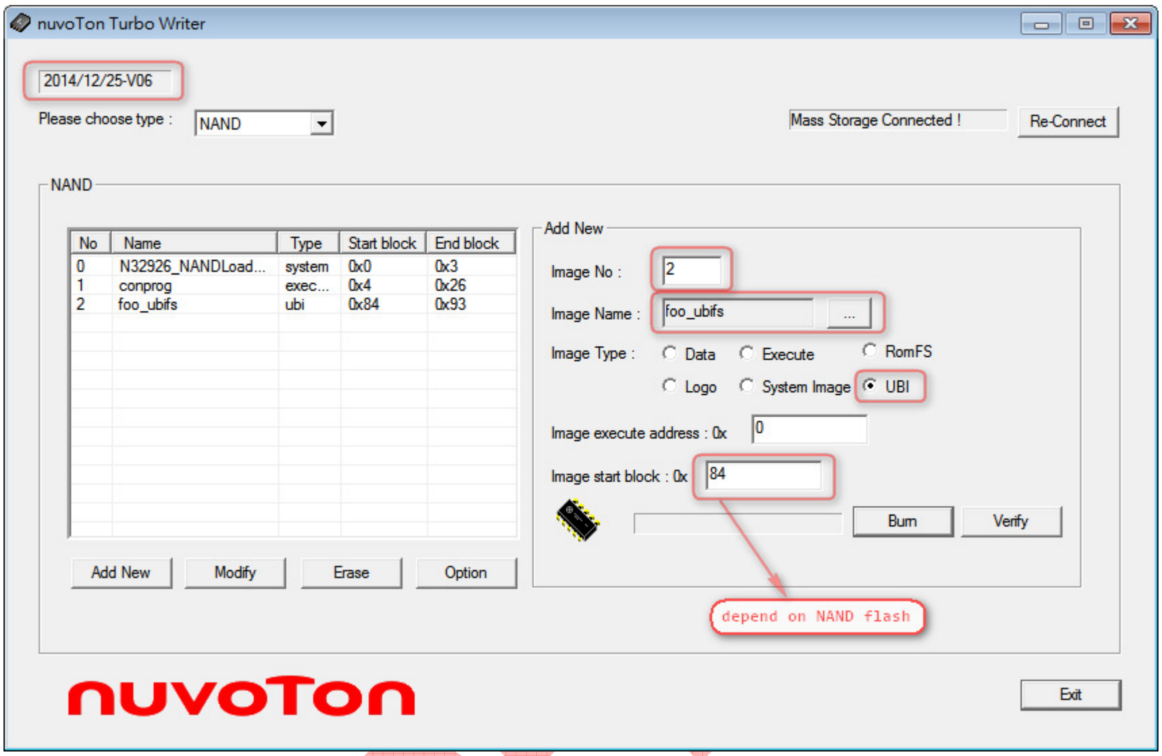
Image Type : [✓ Data] Execute RomFS Logo System Image UBI

Image execute address : 0x 0

Image start block : 0x 84

For NAND1-2, you can program file system image to block ***0x184*** if NAND1-1 disk size is *32MB* since it occupy 32MB / block size = 256 blocks and 4 + 128 + 256 = 388 = 0x184.

If you want to program UBIFS image into NAND flash, please make sure the image type in TurboWriter MUST be "**UBI**". The image type "UBI" only supported in TurboWriter version 2014/12/25-V06 and later. Please update your TurboWriter if you cannot found "UBI" image type.



2014/12/25-V06

Please choose type : NAND

Mass Storage Connected !

NAND

No	Name	Type	Start block	End block
0	N32926_NANDLoad...	system	0x0	0x3
1	conprog	exec...	0x4	0x26
2	foo_ubifs	ubi	0x84	0x93

Image No : 2

Image Name : foo_ubifs

Image Type : Data / Execute / RomFS / Logo / System Image / [✓ UBI]

Image execute address : 0x 0

Image start block : 0x 84

Step 4: (Optional) if you program file system image into NAND flash by TurboWriter, you need to mount it to Linux file system before use it. Try following command to mount YAFFS2 file system on *MTD partition NAND1-1*:

```
mount –t yaffs2 –o"inband-tags" /dev/mtdblock2 /mnt/nand1-1
```

If you want to mount YAFFS2 file system on *MTD partition NAND1-2*, try

```
mount –t yaffs2 –o"inband-tags" /dev/mtdblock3 /mnt/nand1-2
```

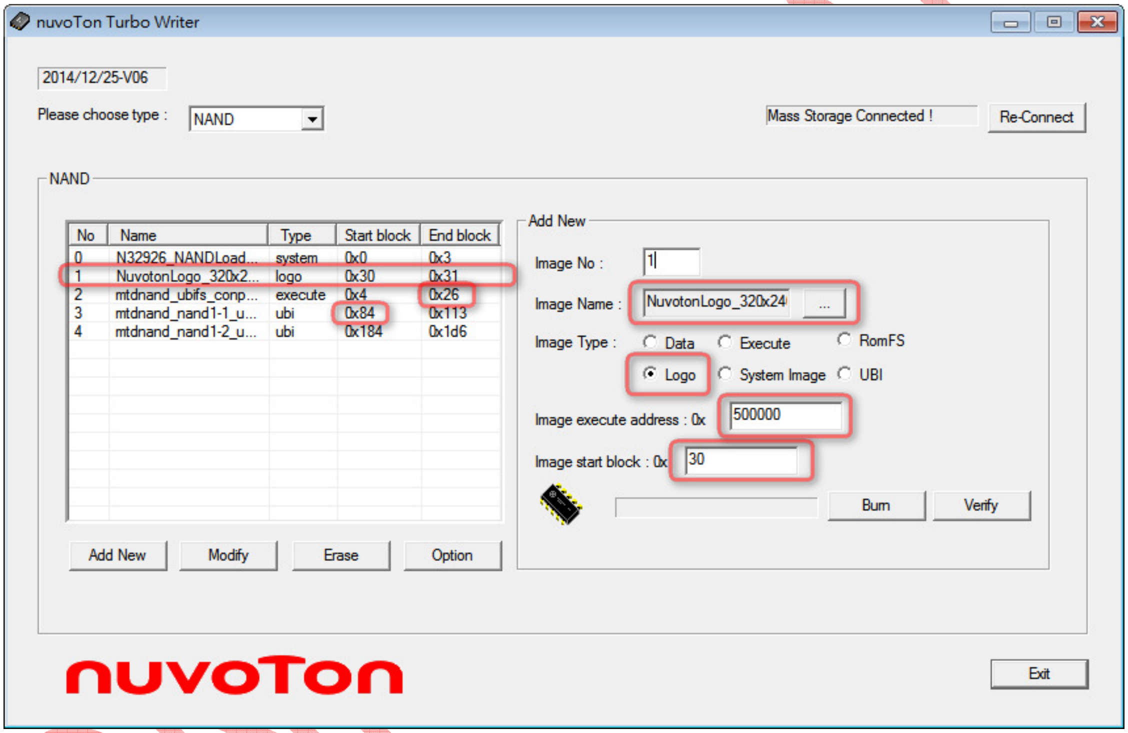
If you want to mount UBIFS file system on *MTD partition NAND1-1*, try

```
/usr/ubiattach /dev/ubi_ctrl -m 2 -O 2048
mount -t ubifs ubi0_0 /mnt/nand1-1
```

If you want to mount UBIFS file system on *MTD partition NAND1-2*, try

```
/usr/ubiattach /dev/ubi_ctrl -m 3 -O 2048
mount -t ubifs ubi1_0 /mnt/nand1-2
```

Step 5: (Optional) if you want to show Logo image on panel when system booting, you have to program Logo image file to NAND in advance by mn_writer or TurboWriter. Please note that the “Image Execute Address” MUST be **0x500000** and “Image start block” MUST locate between execute image and file system image. For example, if execute image located at block 0x4 to 0x26 and file system image located at block 0x84 to 0x113, the Logo image must located between 0x26 and 0x84, such as 0x30 to 0x31 in below case.



No	Name	Type	Start block	End block
0	N32926_NANDLoad...	system	0x0	0x3
1	NuvotonLogo_320x2...	logo	0x30	0x31
2	mtdnand_ubifs_conp...	execute	0x4	0x26
3	mtdnand_nand1-1_u...	ubi	0x84	0x113
4	mtdnand_nand1-2_u...	ubi	0x184	0x1d6

Add New
Image No : 1
Image Name : NuvotonLogo_320x24
Image Type : ( ) Data ( ) Execute ( ) RomFS [✓ Logo] ( ) System Image ( ) UBI
Image execute address : 0x 500000
Image start block : 0x 30

7.2. AutoWriter

You also can program Nand Loader, Linux kernel, Logo (optional), and file system image into all new NAND flash by AutoWriter.

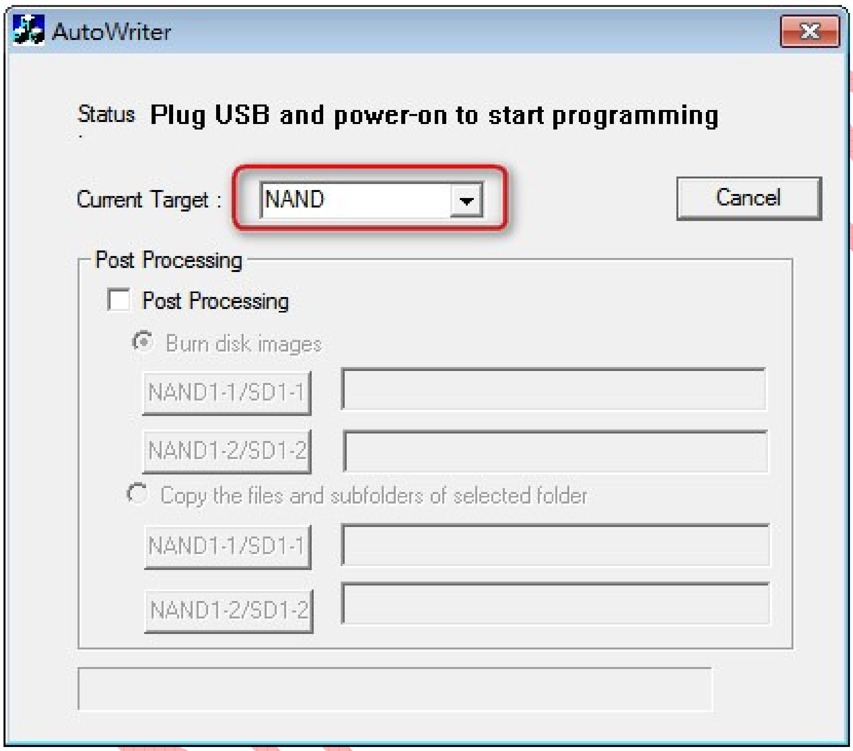
Step 1: Setup the AutoWriter.ini. Please refer to the following example.

```
[Loader File Name]
APP = NANDLoader.bin
ADDRESS= 900000

[Logo File Name]
APP = NuvotonLogo_320x240.bin
ADDRESS= 500000
Start Block = 30

[Execute File Name]
APP = mtdnand_yaffs2_conprog.bin
ADDRESS = 0

[Options]
EraseAll=y

[Data]
No=2
APP1 = mtdnand_rtl8188_nt99141_dev_nand1-1_yaffs2.img
APP2 = mtdnand_rtl8188_nt99141_dev_nand1-2_yaffs2.img
Start Block1 = 84
Start Block2 = 184
```

The [Execute File Name] is the Linux kernel that support MTD device. It will be program into NAND flash and run after reboot.

The [Data] is to burn file system images into the specific block of NAND flash. The key words "Start Block1" and "Start Block2" must be **the start address of MTD partition NAND1-1 or NAND1-2**. It is the hexadecimal number.

Step 2: Run AutoWriter tool to auto program NAND flash according to the setting of AutoWriter.ini.


AutoWriter

Status: Plug USB and power-on to start programming

Current Target : NAND

7.3. fs_writer

If you never program any file system image into NAND flash by TurboWriter or AutoWriter, you can do it by fs_writer utility under Linux kernel.

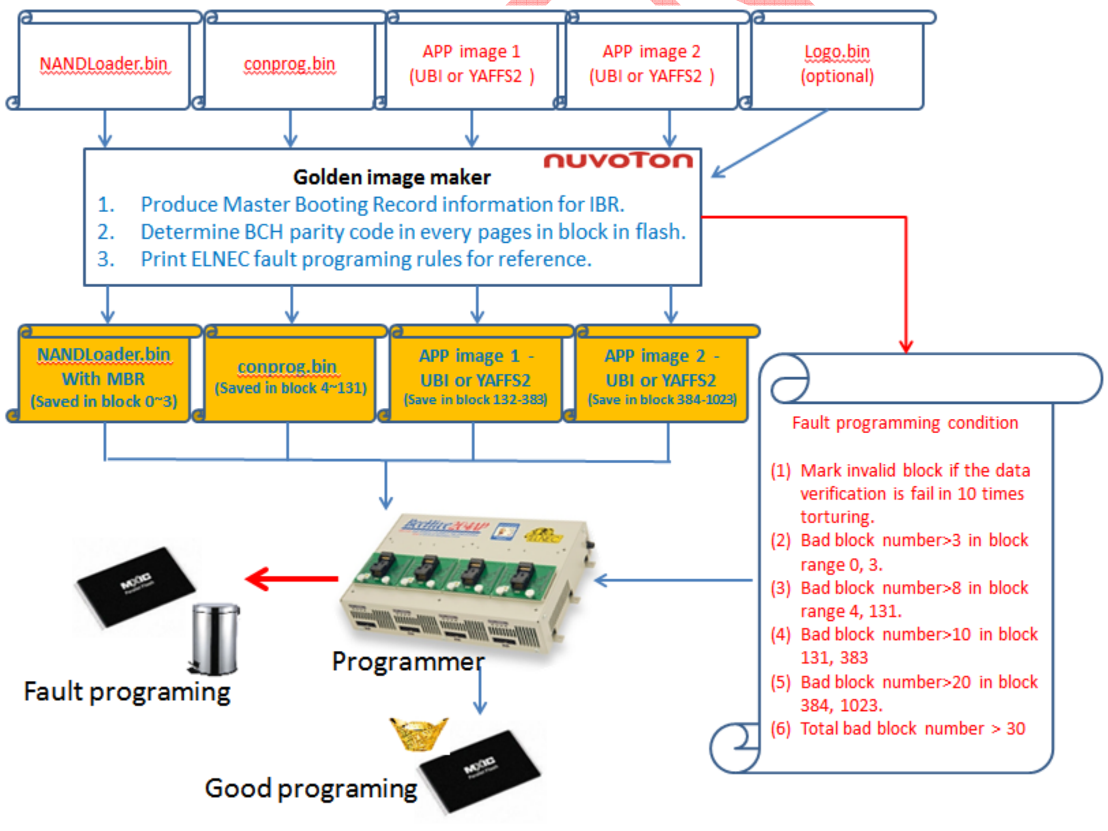
Step 1: Prepare file system image file and fs_writer utility that include shell script and executable file.

Step 2: Run shell script of fs_writer. Refer to chapter 6.1 for detail description.

8. Mass production

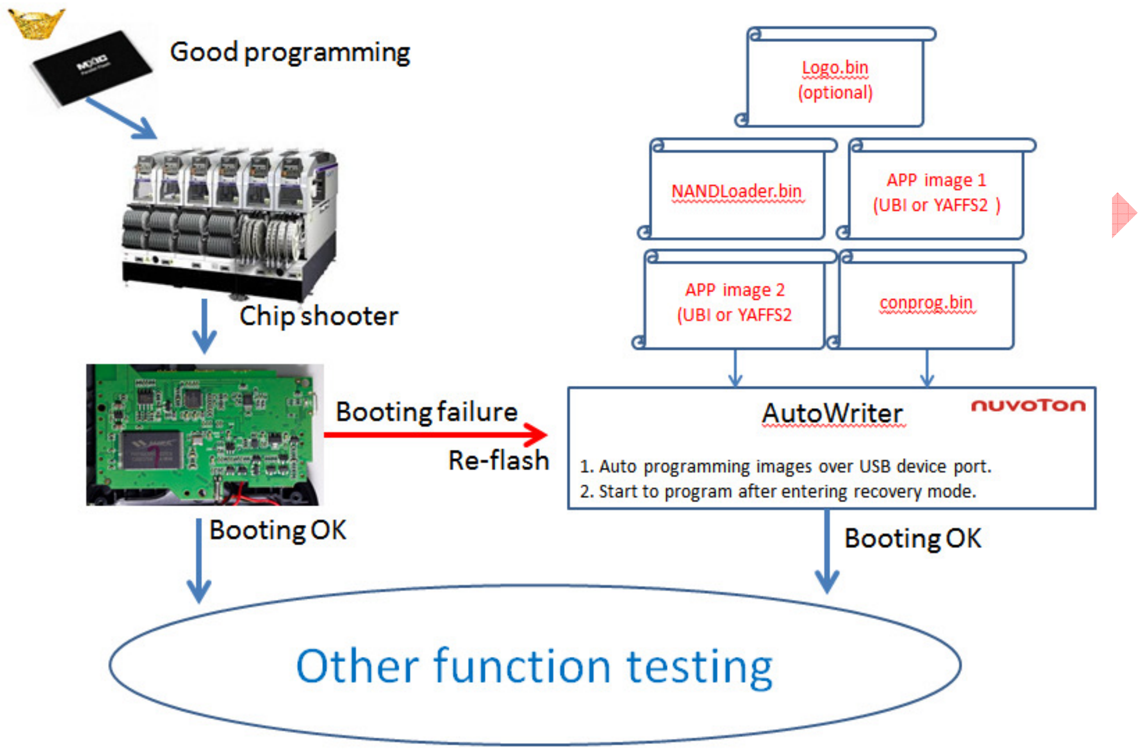
Usually, customer uses flash programmer during mass production stage on assembly line. In this chapter, we will describe some notes.

As below figure, GoldenImgMaker utility produces many golden images as yellow blocks and fault programming rules. The utility can **produce master booting record information for internal booting routine and determines all parity codes in all pages in block in flash**. When customer uses flash programmer, they usually need to **input these golden images and some fault programming rules**. These fault programming rules can help them to find out bad quality flash for increasing production yield. These flashes called good programming are used by chip shooter. Others are called fault programing, aren't used on this project.

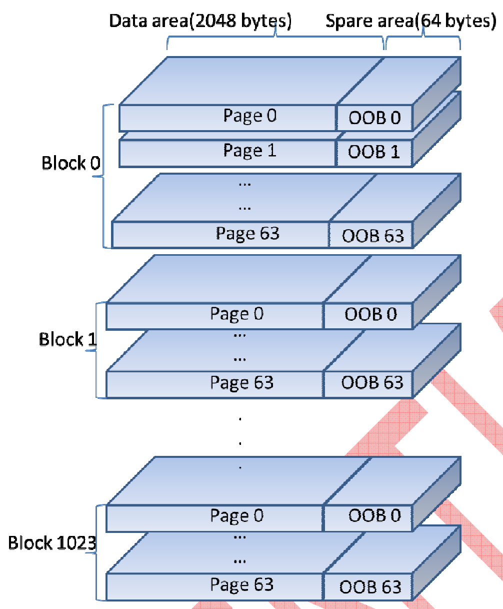

NANDLoader.bin
conprog.bin
APP image 1 (UBI or YAFFS2)
APP image 2 (UBI or YAFFS2)
Logo.bin (optional)
Golden image maker
nuvoTon
1. Produce Master Booting Record information for IBR.
2. Determine BCH parity code in every pages in block in flash.
3. Print ELNEC fault programing rules for reference.
NANDLoader.bin With MBR (Saved in block 0~3)
conprog.bin (Saved in block 4~131)
APP image 1 - UBI or YAFFS2 (Save in block 132-383)
APP image 2 - UBI or YAFFS2 (Save in block 384-1023)
Fault programming condition
(1) Mark invalid block if the data verification is fail in 10 times torturing.
(2) Bad block number>3 in block range 0, 3.
(3) Bad block number>8 in block range 4, 131.
(4) Bad block number>10 in block 131, 383
(5) Bad block number>20 in block 384, 1023.
(6) Total bad block number > 30
Programmer
Fault programing
Good programing

Golden images and fault programing conditions

Sometimes, the flash loss data is caused by high temperature burning so that causes boot failure. At the same time, customer can use AutoWriter to re-program flash in recovery mode.



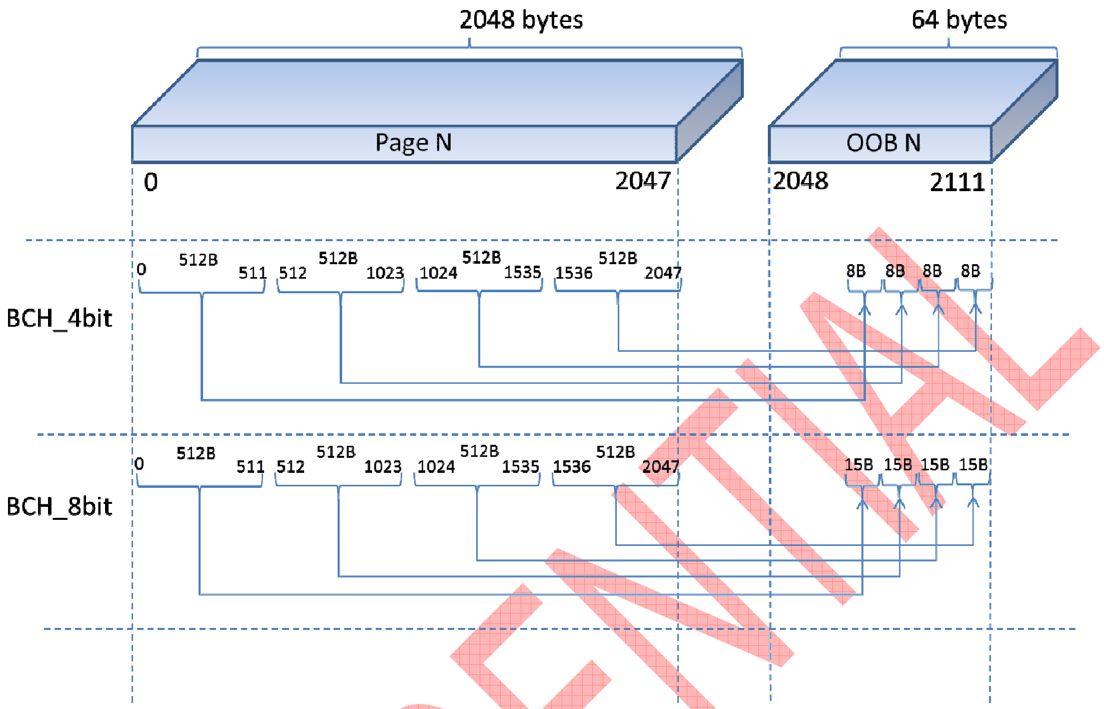
As below figure, it is a NAND flash organization sketch. It includes two parts: data area and spare area. The data area is used to store user data. The spare area is used to store parity codes and marking invalid block.



An example: 2048+64 page NAND Flash organization

As below figure, the BCH codes form a class of cyclic error-correcting codes that are constructed using finite fields. On N329x platform supports BCH error correcting coder. It includes BCH_4, BCH_8, BCH_12, BCH_15, and BCH_24 algorithm. (Please note the BCH_24 algorithm is only supported on N3291 and N3292 platforms).

The BCH_4 algorithm produces 32-bytes parity codes are stored in spare area. Here, the BCH_4 algorithm implies it can correct 4-bit in data area and spare area in a page. If error bits are more than 4 bits, the BCH_4 algorithm won't correct these bits in page successfully. The BCH_8 algorithm produces 60-bytes parity codes are stored in spare area. Here, the BCH_8 algorithm implies the algorithm can correct 8-bit in data area and spare area in a page. If error bits are more than 8 bits, the BCH_8 algorithm won't correct these bits in page successfully.



An example: BCH_4/BCH_8 algorithm

GoldenImgMaker usage:

```
#[Usage] ./GoldenImgMaker
#      -v ....: version
#      -O ....:Output folder path
#      -p ....: page size
#      -o ....: OOB size
#      -s ....: page per block
#      -b ....: block number
#      -t ....: TurboWriter.ini path
#     -c…...: Chip. N3290:0, N3291:1, N3292:2, default: 2
#      -P "<ImgFilePath> [parameters]"
#      [parameters]
#            -t ....: Image type. Data:0, Exec:1, ROMFS:2, Sys: 3, Logo:4
#            -o ....: Redundent area size.(option)\n"
#            -S ....: MTD partition entry size.
#            -O ....: MTD partition entry offset.
#            -j ....: Won't determine parity code if all 0xff in a page.
#            -b ....: BCH error bits(4, 8, 12, 15, 24).
#            -e ....: Execution address.

#[Example]
#./GoldenImgMaker -O <path/to/golden image folder> -t <path/to/turbowriter.ini> \
#                              -p <page size> -o <oob size> -s <page number in block> -b <block number> \
#         -P "<path/to/image1> -t <Image type> -S <partition size, hex> -O <partition offset, hex> -e <Execution address>" \
#         -P "<path/to/image2> -t <Image type> -S <partition size, hex> -O <partition offset, hex> -e <Execution address> -j"
```

An UBI images case study:

MTD partition layout	Image files	Block offset
SYSTEM: BCH4 **Offset: 0x0** **Size: 0x80000 (0.5MB)**	NANDLOADER.bin	Block 0
	NANDLOADER.bin	Block 1
	NANDLOADER.bin	Block 2
	NANDLOADER.bin	Block 3
EXECUTE: BCH8 **Offset: 0x80000** **Size: 0x1000000 (16MB)**	**mtdnand_ubifs_conprog.bin** **and Logo.bin (optional)**	Block 4 ……….. Block 131
NAND1-1: BCH8 **Offset: 0x1080000** **Size: 0x2000000 (32MB)**	**mtdnand_nand1-1_ubi.img**	Block 132 …… Block 387
NAND1-2: BCH8 **Offset: 0x3080000** **Size: 0x4F80000 (79.5MB)**	**mtdnand_nand1-2_ubi.img**	Block 388 …….. Block 1023

1. A NAND flash with 1024 blocks is adopted in the system.
2. Each block has 64 pages.
3. Each page has 2048+64 bytes.
4. (*)The SYSTEM has 4 blocks. The offset address starts at 0x0 and size is 0.5MB. The BCH_4 algorithm is adopted in this region. The execution address is at 0x90000. The image type is **SYS**.
5. The EXECUTE has 128 blocks. The offset address starts at 0x80000 and size is 16MB. The BCH_8 algorithm is adopted in this region. The execution address is at 0x0. The image type is **EXEC**.
   If you want to show Logo image to panel when system booting, you need to program Logo image file to free space in EXECUTE partition. The execution address is at 0x500000. The image type is **LOGO**. This is an optional feature.
6. The NAND1-1 has 256 blocks. The offset address starts at 0x1080000 and size is 32MB. The BCH_8 algorithm is adopted in this region. The image type is DATA. To fill 0xff into spare area if all bytes in a page are 0xff.
7. The NAND1-2 has 636 blocks. The offset address starts at 0x3080000 and size is 79.5MB. The BCH_8 algorithm is adopted in this region. The image type is DATA. To fill 0xff into spare area if all bytes in a page are 0xFF.

For this case, these parameters are shown as below.

```
#!/bin/bash

# 128 MB, 2048+64, page per block: 64, block size :128KB
# MTD NAND Partition layout
# Partition 0: 4 Block        offset: 0,           size: 0x80000
# Partition 1: (16MB)         offset: 0x80000,     size: 0x1000000
# Partition 2: (32MB)         offset: 0x1080000,   size: 0x2000000
# Partition 3: (to FULL)      offset: 0x3080000,   size: 0x4F80000

if ./GoldenImgMaker -O ../Demo_nand/skyeye_golden -t ../Demo_nand/TurboWriter.ini -p 2048 -o 64 -s 64 -b 1024 \
                    -P "../Demo_nand/NANDLoader.bin -t 3 -b 4 -O 0 -S 0x80000 -e 0x900000" \
                    -P "../Demo_nand/mtdnand_ubifs_conprog.bin -t 1 -b 8 -O 0x100000 -S 0x1000000 –e 0x0" \
                    -P "../Demo_nand/NuvotonLogo_320x240.bin -t 4 -b 8 -O 0xF00000 -S 0x100000 -e 0x500000" \
                    -P "../Demo_nand/mtdnand_nand1-1_ubi.img -t 0 -b 8 -O 0x1080000 -S 0x2000000 -j" \
                    -P "../Demo_nand/mtdnand_nand1-2_ubi.img -t 0 -b 8 -O 0x3080000 -S 0x4F80000 -j";
then
     echo "=================================================================="
     echo "All golden images are created."
     echo "All parity code in spare area are produced by GoldenImgMaker."
     echo "Now, Please put these files into memory by your NAND programmer."
     echo "=================================================================="
fi
```

The NANDLoader applies BCH_4 different from others applied BCH_8. It relates with internal booting routine (IBR). The IBR sets SMRA to 64 bytes and applies BCH_4 by default, so we must apply BCH_T4 for 2K page NAND flash.

SMRA size table in IBR:

Page size	SMRA size (Bytes)	Applied BCH algorithm
512	16	BCH_T4
2K	64	BCH_T4
4K	128	BCH_T8
8K	376	BCH_T12

Execution result:

```
Argv[0]=./GoldenImgMaker
argv[1]=-O
argv[2]=../Demo_nand/skyeye_golden
argv[3]=-t
argv[4]=../Demo_nand/TurboWriter.ini
argv[5]=-p
argv[6]=2048
argv[7]=-o
argv[8]=64
argv[9]=-s
argv[10]=64
argv[11]=-b
argv[12]=1024
argv[13]=-P
argv[14]=../Demo_nand/NANDLoader_240MHz_Fast_20140416.bin -t 3 -b 4 -O 0 -S 0x80000 -e 0x900000
argv[15]=-P
argv[16]=../Demo_nand/mtdnand_ubifs_conprog.bin -t 1 -b 8 -O 0x80000 -S 0x1000000
argv[17]=-P
argv[18]=../Demo_nand/mtdnand_nand1-1_ubi.img -t 0 -b 8 -O 0x1080000 -S 0x2000000 -j
argv[19]=-P
argv[20]=../Demo_nand/mtdnand_nand1-2_ubi.img -t 0 -b 8 -O 0x3080000 -S 0x4F80000 -j
GoldenImgMaker Version: 1.0
Compilation Date / Time.....: May 19 2015 / 16:42:27
=====> Parsing ../Demo_nand/TurboWriter.ini as Boot Code Optional Setting INI file <=====
** Process ../Demo_nand/TurboWriter.ini file **
   OptionalMarker = 0xAA55AA55
   Counter           = 13
   Pair 0: Address = 0xB0000004, Value = 0x00032FFF
   Pair 1: Address = 0xB0000208, Value = 0x00008350
   Pair 2: Address = 0x5A5A5A5A, Value = 0x00000000
   Pair 3: Address = 0xB0003054, Value = 0x00000000
   Pair 4: Address = 0xB0003004, Value = 0x00000010
   Pair 5: Address = 0xB0000240, Value = 0x00000009
   Pair 6: Address = 0xB000020C, Value = 0x00000019
   Pair 7: Address = 0x55AA55AA, Value = 0x00000018
   Pair 8: Address = 0xB0003004, Value = 0x00000020
   Pair 9: Address = 0xB000301C, Value = 0x00002403
   Pair 10: Address = 0xB0003054, Value = 0x00000000
   Pair 11: Address = 0xB0003000, Value = 0x00078476
   Pair 12: Address = 0xB0003034, Value = 0x0000FF00
Create NandLoader at block 0 page 0, buffer address at 0x00000000
         Pad 0xFF for NandLoader from block 0 page 8, image file address 0x4200
         Create image information at block 0 page 63, image file address 0x207c0
Write ../Demo_nand/mtdnand_ubifs_conprog.bin to block 4 page 0, memory address at 0x00084000
         Pad 0xFF for ../Demo_nand/mtdnand_ubifs_conprog.bin from block 38 page 34, memory address at 0x4f7880
Write ../Demo_nand/mtdnand_nand1-1_ubi.img to block 132 page 0, memory address at 0x01104000
Write ../Demo_nand/mtdnand_nand1-2_ubi.img to block 388 page 0, memory address at 0x03204000
```

```
Chip: N3292 Platform
Output folder: ../Demo_nand/skyeye_golden
Turbowriter.ini filepath: ../Demo_nand/TurboWriter.ini
All image number: 4
Page size: 2048
OOB size: 64
Page per block: 64
Block number: 1024
[0]: ../Demo_nand/NANDLoader_240MHz_Fast_20140416.bin
        BCH Error bits: 4
        Offset in MTD table: 0(0x0)
        Size in MTD table: 524288(0x80000)
        Skip all 0xff page: no
        Image type: IMG_SYS
        Image name: NANDLoader_240MHz_Fast_20140416
        Image file length: 15192(0x3B58)
        Image start block: 0(0x0)
        Image end block: 3(0x3)
        Golden image information:
        Path: ../Demo_nand/skyeye_golden/NANDLoader_240MHz_Fast_20140416.bin.golden
        At buffer address(Start_block*Page_per_block*(OOB_size+Page_size)): 0x00000000
        Block number(File_length/Page_per_block/(OOB_size+Page_size)): 0x1(1)
[1]: ../Demo_nand/mtdnand_ubifs_conprog.bin
        BCH Error bits: 8
        Offset in MTD table: 524288(0x80000)
        Size in MTD table: 16777216(0x1000000)
        Skip all 0xff page: no
        Image type: IMG_EXEC
        Image name: mtdnand_ubifs_conprog.bin
        Image file length: 4524928(0x450B80)
        Image start block: 4(0x4)
        Image end block: 131(0x83)
        Golden image information:
        Path: ../Demo_nand/skyeye_golden/mtdnand_ubifs_conprog.bin.golden
        At buffer address(Start_block*Page_per_block*(OOB_size+Page_size)): 0x00084000
        Block number(File_length/Page_per_block/(OOB_size+Page_size)): 0x23(35)
[2]: ../Demo_nand/mtdnand_nand1-1_ubi.img
        BCH Error bits: 8
        Offset in MTD table: 17301504(0x1080000)
        Size in MTD table: 33554432(0x2000000)
        Skip all 0xff page: yes
        Image type: IMG_DATA
        Image name: mtdnand_nand1-1_ubi.img
        Image file length: 14155776(0xD80000)
        Image start block: 132(0x84)
        Image end block: 387(0x183)
        Golden image information:
        Path: ../Demo_nand/skyeye_golden/mtdnand_nand1-1_ubi.img.golden
        At buffer address(Start_block*Page_per_block*(OOB_size+Page_size)): 0x01104000
        Block number(File_length/Page_per_block/(OOB_size+Page_size)): 0x6C(108)
[3]: ../Demo_nand/mtdnand_nand1-2_ubi.img
        BCH Error bits: 8
        Offset in MTD table: 50855936(0x3080000)
        Size in MTD table: 83361792(0x4F80000)
        Skip all 0xff page: yes
        Image type: IMG_DATA
        Image name: mtdnand_nand1-2_ubi.img
        Image file length: 13500416(0xCE0000)
        Image start block: 388(0x184)
        Image end block: 1023(0x3FF)
        Golden image information:
        Path: ../Demo_nand/skyeye_golden/mtdnand_nand1-2_ubi.img.golden
        At buffer address(Start_block*Page_per_block*(OOB_size+Page_size)): 0x03204000
        Block number(File_length/Page_per_block/(OOB_size+Page_size)): 0x67(103)
export_elnec_csv: Export CSV.
    0;      3;      1; 0xff7ff003;   NANDLoader_240MHz_Fast_20140416
    4;    131;     35; 0xff7ff006;   mtdnand_ubifs_conprog.bin
  132;    387;    108; 0xff7ff00c;   mtdnand_nand1-1_ubi.img
  388;   1023;    103; 0xff7ff01f;   mtdnand_nand1-2_ubi.img
```

```
To decide maximum allowed invalid block number rule:
          (1) The image type is SYS, the value is 3.
          (2) Others are total block number*0.05 .
=======================================================================
All golden images are created.
All parity code in spare area are produced by GoldenImgMaker.
Now, Please put these files into memory by your NAND programmer.
=======================================================================
```

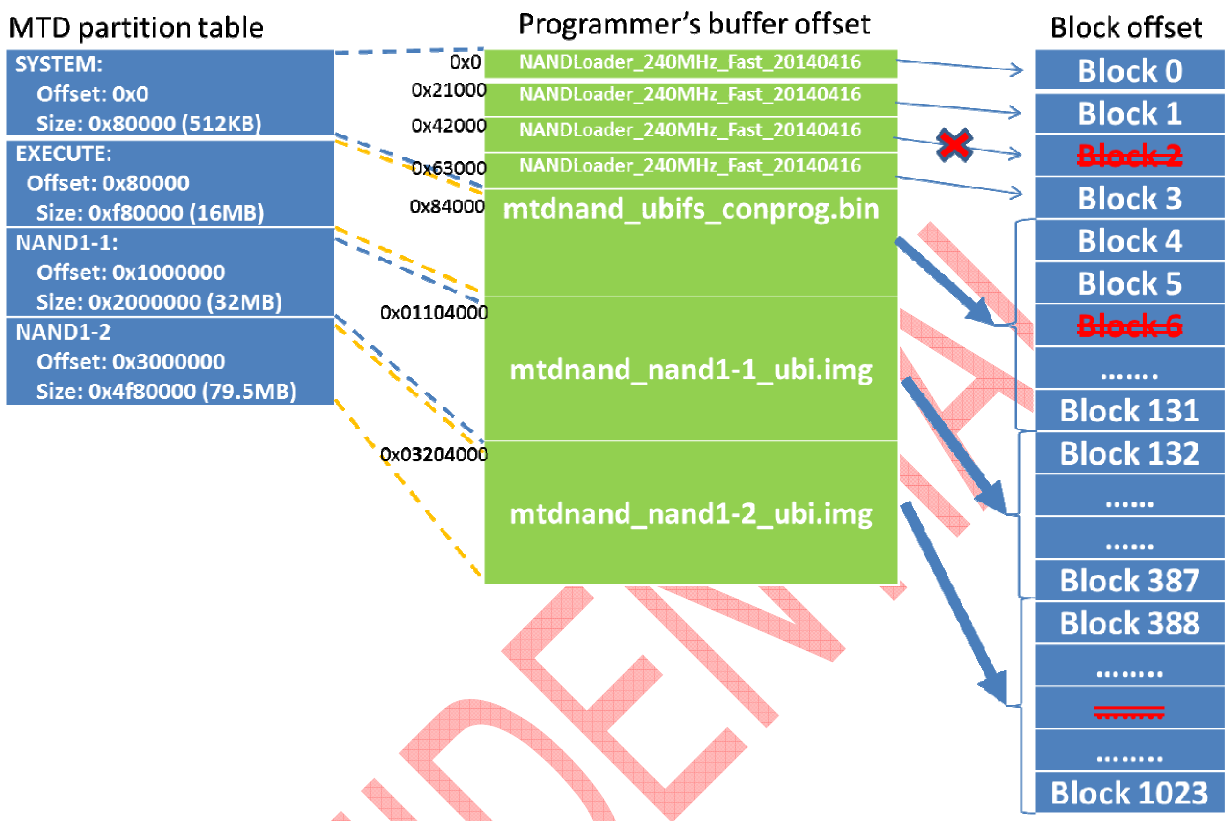
We can get some ELNEC parameters from goldenimgmaker log. e.g.

(1) Each image should put corresponding buffer address.

- I. NANDLoader_240MHz_Fast_20140416.bin.golden at buffer 0x00000000.
  Notice: the binary should put 4 times. (At buffer address 0x00000000, 0x00021000, 0x00042000 and 0x00063000)
- II. mtdnand_ubifs_conprog.bin.golden at buffer address 0x00084000
- III. mtdnand_nand1-1_ubi.img.golden at buffer address 0x01104000
- IV. mtdnand_nand1-2_ubi.img.golden at buffer address 0x03204000

(2) An ELNEC CSV file includes many entries that notes block information, invalid block management information and comment.

```
   0;    3;    1; 0xff7ff003;   NANDLoader_240MHz_Fast_20140416
   4;  131;   35; 0xff7ff006;   mtdnand_ubifs_conprog.bin
 132;  387;  108; 0xff7ff00c;   mtdnand_nand1-1_ubi.img
 388; 1023;  103; 0xff7ff01f;   mtdnand_nand1-2_ubi.img
```


MTD partition table
SYSTEM:
Offset: 0x0
Size: 0x80000 (512KB)
EXECUTE:
Offset: 0x80000
Size: 0xf80000 (16MB)
NAND1-1:
Offset: 0x1000000
Size: 0x2000000 (32MB)
NAND1-2
Offset: 0x3000000
Size: 0x4f80000 (79.5MB)
Programmer's buffer offset
0x0
0x21000
0x42000
0x63000
0x84000
0x01104000
0x03204000
NANDLoader_240MHz_Fast_20140416
NANDLoader_240MHz_Fast_20140416
NANDLoader_240MHz_Fast_20140416
NANDLoader_240MHz_Fast_20140416
mtdnand_ubifs_conprog.bin
mtdnand_nand1-1_ubi.img
mtdnand_nand1-2_ubi.img
Block offset
Block 0
Block 1
~~Block 2~~
Block 3
Block 4
Block 5
~~Block 6~~
…….
Block 131
Block 132
……
……
Block 387
Block 388
……..
~~……..~~
……..
Block 1023

[Q&A]

[Q] How to define fault programming rule to archive maximum flash utilization?

[A] We can give a bit of tolerance. For example, the NANDLoader.bin uses one block size, but we give four block size. The one block fault-tolerant rate is lower than four block size we give.

[Q] How to apply the ELNEC CSV file and execution addresses are showed by goldenimgmaker?

[A] Please refer ELNEC programmer datasheet or demo video we recorded.

[Q] How to determine maximum invalid block number in a partition?

[A] To decide maximum allowed invalid block number rule:

(1) The image type is SYS, the value is 3.

(2) Others are total block number*0.05.

[Q] How to display Logo image to panel as soon as possible when system booting?

[A] First, you need a NandLoader that be able to display Logo image to panel. Nuvoton provides a sample NandLoader that named N329xx_NANDLoader_xxxt_**Logo**.bin (postfix with "Logo") to support Logo display.

Secondly, you have to program Logo image to NAND with image type "Logo" in advance. We suggest you to program Logo image file to free space in EXECUTE partition that don't conflict with Linux kernel image. You can do that by TurboWriter, AutoWriter, mn_writer, or GoldenImgMaker. Please check the detail description in chapter about these tools.

9. Revision history

Version	Date	Chang Log
v1.00	Jan. 2, 2014	● Create
v1.01	Jan. 15, 2014	● Modified MTD layout of example. ● Append YAFFS2 description in section 6.1.
V1.02	Aug. 29, 2014	● Add example to program MTD to NAND flash. ● Modify kernel configure about MTD and VPOST initial.
V1.03	Dec. 19, 2014	● Add notification about TurboWriter configuration for UBIFS image programming
V1.04	Dec. 25, 2014	● Update TurboWriter description about UBIFS file system image supporting.
V1.05	Jun. 1, 2015	● Add mass production case study ● Modify Autowriter.ini example.
V1.06	Jum. 12, 2015	● Support Logo display feature when system booting.

Important Note