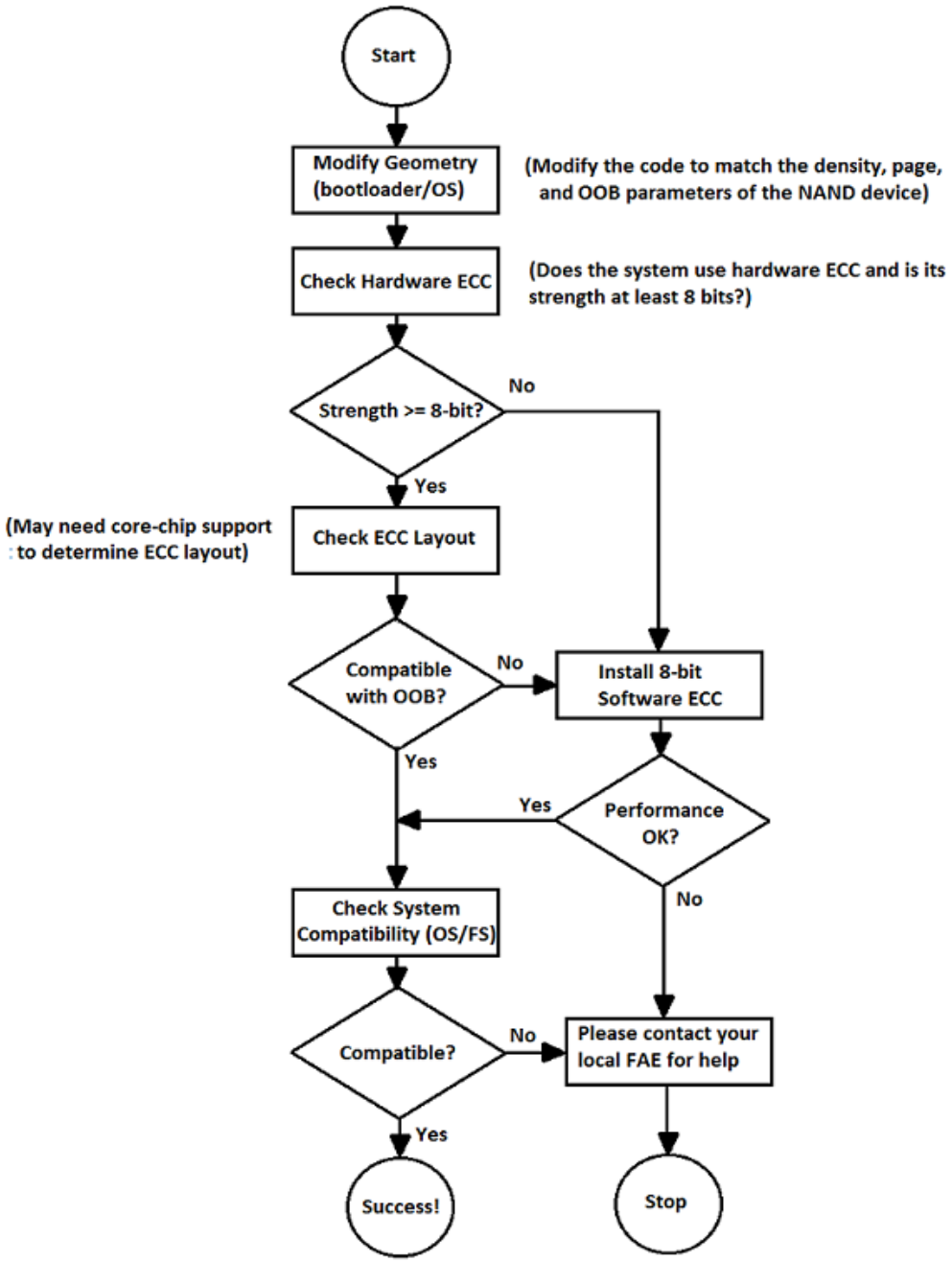
How to handle the spare-byte area of Macronix 36nm NAND Flash

Some NAND Flash come with a non-standard spare area that is larger than what is commonly used by Linux for error correction. MX30LF2G28AB and MX30LF4G28AB NAND Flash are such examples. To take advantage of the larger spare area, the system driver must be modified. The flow chart illustrated in *"Figure 1. Driver Modification Flow"* provides a simple guideline on how the process works.

Figure 1. Driver Modification Flow



The document focuses on issues related to expanding the spare area size. We will also show you how to test ECC (Error Correction Code) strength on a Linux system. Please contact your local Macronix FAE if you have any question. If you have any comments or suggestions, please send them to *flash_model@mxic.com.tw*.

Content

What is OOB?

OOB means **out of band**. A “band” in Flash memory can be viewed as a page, so “out of band” refers to the **spare area** adjacent to a page. Generally, OOB exists in NAND Flash to enable ECC (Error Correction Code) and bad block management. In addition to storing ECC and bad block information, OOB can be used to store information related metadata of the file system.

In the datasheet, the page geometry is generally described as the combination of the data space and the adjacent OOB area. That means that you can access page data and adjacent OOB with a single Read operation.

Example of OOB representation from MX30LF4G28AB datasheet:
Page size: (2048+112) bytes (2048 bytes data + 112 bytes OOB)
Block size: (128K+ 7K) bytes (64 pages)

OOB Usage Considerations

The kinds of changes that are required for different OOB sizes are similar to the changes that would be needed for different page sizes. In addition, we need to modify our boot loader, kernel driver, image maker, and ECC layout. The modifications and tests performed are based on an i.MX28 EVK board running Linux kernel 2.6.35.3. The following bullets are related issues that we should pay special attention to:

Geometry Setting in Driver

Changing OOB size will directly affect the driver setting. For example, MTD (Memory Technology Device) is the first thing in Linux and u-boot we should check. Hence, we should change OOB size definition in *drivers/mtd/nand_base.c*.

Some systems have their own place to define flash geometry that you should also care, such as *nand_device_info.c* file provided by i.MX28EVK. This should be specially take care for.

```
static struct nand_device_info nand_device_info_table_type_2[] __initdata = {
    {
    .end_of_table                       = false,
    .manufacturer_code                  = 0xc2,
    .device_code                        = 0xdc,
    .cell_technology                = NAND_DEVICE_CELL_TECH_SLC,
    .chip_size_in_bytes                 = 512LL*SZ_1M,
    .block_size_in_pages                = 64,
    .page_total_size_in_bytes           = 2*SZ_1K + 112,
    .ecc_strength_in_bits               = 8,
    .ecc_size_in_bytes                  = 512,
```

ECC Layout

The error correction code scheme uses the OOB for storage of ECC data, and the layout of ECC should be checked to make sure that it fits within the OOB. The following is an example of MTD software ECC layout used for 112-byte OOB flash. The struct nand_ecclayout is modified from layout for 64-byte OOB which is 1-bit ECC used for a 2Kbyte page.

```
static struct nand_ecclayout nand_oob_112 = {
          .eccbytes = 24,
          .eccpos = {
                    88,  89,  90,  91,  92,  93,  94,  95,
                    96,  97,  98,  99, 100, 101, 102, 103,
                    104, 105, 106, 107, 108, 109, 110, 111},
          .oobfree = {
                {.offset = 2,
                 .length = 86}}
     };
int nand_scan_tail(struct mtd_info *mtd)
     {
          ......
          if (!chip->ecc.layout && (chip->ecc.mode != NAND_ECC_SOFT_BCH)) {
                switch (mtd->oobsize) {
                ......
                case 112:
                      chip->ecc.layout = &nand_oob_112;
                      break;
                case 128:
                      chip->ecc.layout = &nand_oob_128;
                      break;
```

In Linux kernel 2.6.x version, software ECC support is limited to 1-bit. Since the processing overhead required by software ECC could potentially degrade system performance, we suggest that it not be used. In case software ECC must be used, it is highly recommended that the Linux kernel be updated to a version higher than 3.0, or add *nand_bch.c* file and modify *nand_base.c*. The BCH default in Linux is to support 4-bit ECC as shown in the example. For Macronix 36nm NAND Flash, you should change the parity to 13 bytes for 8-bit ECC.

```
int nand_scan_tail(struct mtd_info *mtd)
   {
   ......
   case NAND_ECC_SOFT_BCH:
          ......
          if (!chip->ecc.size && (mtd->oobsize >= 64)) {
                chip->ecc.size = 512;
                chip->ecc.bytes = 13;
          }
          chip->ecc.priv = nand_bch_init(mtd,
                                    chip->ecc.size,
                                    chip->ecc.bytes,
                                    &chip->ecc.layout);
```

Generally, core-chip vendor provides hardware ECC in their memory controller and the operating system generally adopts hardware ECC by default. Therefore, the hardware ECC layout in each system should be checked. For example, in the Freescale i.MX28 EVK board, the metadata[1] size of the GPMI driver needs to be adjusted.

Note 1. Metadata can be used to manage information such as erase cycle counts, bad block marks, and logical address information.

MTD Utility

MTD utility (Memory Technology Device) is one of the most popular flash tools in the Linux system. However, there are also certain things that need to be modified to take care of the OOB size within Macronix Flash. The most obvious one is the code used for device identification located in the main function of nandwrite.c and nanddump.c. Modify and re-make mtd-utils with a corresponding cross compiler to make sure it works.

```
int main(int argc, char * const argv[]) {
        ......
        /* Make sure device page sizes are valid */
        if (!(meminfo.oobsize == 128 && meminfo.writesize == 4096) &&
                    !(meminfo.oobsize == 112 && meminfo.writesize == 2048) &&
                    !(meminfo.oobsize == 64 && meminfo.writesize == 2048) &&
                    !(meminfo.oobsize == 32 && meminfo.writesize == 1024) &&
                    !(meminfo.oobsize == 16 && meminfo.writesize == 512) &&
                    !(meminfo.oobsize == 8 && meminfo.writesize == 256)) {
```

File System

To boot Linux from NAND flash, the root file system is needed. Normally, we use YAFFS2, JFFS2 or UBIFS for the root file system. We use different image makers to build these types of root file systems. For example, we use mkfs.jffs2 to build a root file system of type JFFS2. And the image maker may need to be modified for different Page and OOB sizes.

Additionally, some file systems may use OOB area for software ECC or storing metadata. We need to make sure that all related options have been turned off during kernel configuration, and then perform tests to check the compatibility of hardware ECC and the selected file system.

How do we test ECC?

Before testing, you should know what type of ECC you are using. Normally, there are three types of ECC support for NAND devices:

On-die ECC

It's difficult for us to test on-die ECC because we are not able to see or modify this type of ECC which is built within the Flash device.

Hardware ECC

This type of ECC is implemented by core-chip vendors and integrated into the memory controller. It's the most popular solution for Flash memory.

Software ECC

Most software ECC solutions reside in the driver or file system. For example, the MTD driver has built-in Hamming code and BCH ECC algorithms for users. However, because of potential performance degradation, software ECC is seldom used. In rare cases, it's used to enhance weak hardware ECC.

You must download and build mtd-utils with a suitable cross compiler. Here we use mtd-utils-2010.06 and arm-linux-gcc-4.3.2. After building, you can copy the mtd-utils commands from arm-linux/ to rootfs/. The commands can be directly used after you reboot. Then you can follow these steps to test ECC:

i. Create a series of ASCII data such as ">" (0x3e) and write it to device.

```
nandwrite -p -a /dev/mtd1 file1
```

ii. Read original data and corresponding parity of a page without ECC.

```
nanddump -o -b -n -l 2048 /dev/mtd1 -f file2
```

iii. Modify part of data from ">" (0x3e) to "<" (0x3c). The number of bytes to be modified depends on your ECC strength. For example if you want to test 8-bit ECC, you need to modify 8 bytes.

iv. Erase all data of the page.

```
flash_erase /dev/mtd1 0 1
```

v. Write back modified data and original parity without ECC.

```
nandwrite -n -o /dev/mtd1 file2
```

vi. Read data with ECC to verify its correction.

```
nanddump -o -b -l 2048 /dev/mtd1 -f file3
```

Sample test result:

```
root@freescale /$ mtd-utils/nanddump -o -b -l 2048 /dev/mtd1 -f eee
ECC failed: 0
ECC corrected: 0
Number of bad blocks: 0
Number of bbt blocks: 0
Block size 131072, page size 2048, OOB size 112
Dumping data starting at 0x00000000 and ending at 0x00000800...
ECC: 8 corrected bitflip(s) at offset 0x00000000
```

Revision History

Table 1. Revision History

Revision No.	Description	Page	Date
REV. 1	Initial Release	ALL	MAY. 13, 2014
REV. 2	Revised document format and content of *"Geometry Setting in Driver"* and Steps to test ECC in *"Software ECC"*	3-6	SEP. 18, 2014
REV. 3	Content (wording) modification	1, 4	DEC. 09, 2015

Except for customized products which have been expressly identified in the applicable agreement, Macronix's products are designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only, and not for use in any applications which may, directly or indirectly, cause death, personal injury, or severe property damages. In the event Macronix products are used in contradicted to their target usage above, the buyer shall take any and all actions to ensure said Macronix's product qualified for its actual use in accordance with the applicable laws and regulations; and Macronix as well as it's suppliers and/or distributors shall be released from any and all liability arisen therefrom.

For the contact and order information, please visit Macronix's Web site at: http://www.macronix.com