Setup Macronix NAND Flash on Freescale i.MX28 EVK

The procedures defined in this document are verified by Linux kernel 2.6.31 and 2.6.35.3 version. You may need to setup hardware environment first. Then install i.MX28 EVK software tool. If you have any question or suggestion, feel free to your local FAE or contact us: flash_model@mxic.com.tw.

Contents

Freescale i.MX28 EVK Environment Guide

We'll show you how to setup Macronix NAND with i.MX28 board step by step. All the modifications are based on Linux-2.6.35.3 kernel source. The following items are related paths in i.MX28 kernel:

Compile environment: `/ltib`
Linux kernel source: `/ltib/rpm/BUILD/linux-2.6.35.3`
Kernel configure file: `/ltib/config/platform/imx/imx28evk_defconfig.dev`
U-boot source: `/ltib/rpm/BUILD/u-boot-2009.08`

Macronix Linux NAND Driver Patching

Add the Macronix manufacture id to the define list in *include/linux/nand.h*

```
548  * NAND Flash Manufacturer ID Codes
549  */
550 #define NAND_MFR_TOSHIBA    0x98
551 #define NAND_MFR_SAMSUNG    0xec
552 #define NAND_MFR_FUJITSU    0x04
553 #define NAND_MFR_NATIONAL   0x8f
554 #define NAND_MFR_RENESAS    0x07
555 #define NAND_MFR_STMICRO    0x20
556 #define NAND_MFR_HYNIX      0xad
557 #define NAND_MFR_MICRON     0x2c
558 #define NAND_MFR_AMD        0x01
559 #define NAND_MFR_MACRONIX   0xc2
```

Insert the “name”, “device id” and “memory density” in the **nand_flash_ids** instance of the nand_flash_dev structure in the file *drivers/mtd/nand/nand_ids.c*.

Because the MX30LF1G08AA’s (1 Gigabit) information already exists, you only need to add the MX30LF1208AA (512 Megabit) information to this table.

```
24  struct nand_flash_dev nand_flash_ids[] = {

76      /*512 Megabit */
77      {"NAND 64MiB 1,8V 8-bit",   0xA2, 0,  64, 0, LP_OPTIONS},
78      {"NAND 64MiB 1,8V 8-bit",   0xA0, 0,  64, 0, LP_OPTIONS},
79      {"NAND 64MiB 3,3V 8-bit",   0xF2, 0,  64, 0, LP_OPTIONS},
80      {"NAND 64MiB 3,3V 8-bit",   0xD0, 0,  64, 0, LP_OPTIONS},
81      {"NAND 64MiB 3,3V 8-bit",   0xF0, 0,  64, 0, LP_OPTIONS},
82      {"NAND 64MiB 1,8V 16-bit",  0xB2, 0,  64, 0, LP_OPTIONS16},
83      {"NAND 64MiB 1,8V 16-bit",  0xB0, 0,  64, 0, LP_OPTIONS16},
84      {"NAND 64MiB 3,3V 16-bit",  0xC2, 0,  64, 0, LP_OPTIONS16},
85      {"NAND 64MiB 3,3V 16-bit",  0xC0, 0,  64, 0, LP_OPTIONS16},
```

And please list Macronix ID definition to struct **nand_manuf_ids** in the file *drivers/mtd/nand/nand_ids.c*.

```
170 struct nand_manufacturers nand_manuf_ids[] = {
171     {NAND_MFR_TOSHIBA, "Toshiba"},
172     {NAND_MFR_SAMSUNG, "Samsung"},
173     {NAND_MFR_FUJITSU, "Fujitsu"},
174     {NAND_MFR_NATIONAL, "National"},
175     {NAND_MFR_RENESAS, "Renesas"},
176     {NAND_MFR_STMICRO, "ST Micro"},
177     {NAND_MFR_HYNIX, "Hynix"},
178     {NAND_MFR_MICRON, "Micron"},
179     {NAND_MFR_AMD, "AMD"},
180     {NAND_MFR_MACRONIX, "Macronix"},
181     {0x0, "Unknown"}
182 };
```

Add information of the Macronix NAND device in the **nand_device_info_table_type_2** instance of the nand_device_info structure in the file *drivers/mtd/nand/nand_device_info.c*. This file only exists in i.MX28 kernel which is provided by Freescale. You can't find it in public release Linux source. It is used for the GPMI NAND controller.

The image below shows the information you must add for the 512Mbit MX30LF1208AA.

```
22 static struct nand_device_info nand_device_info_table_type_2[] __initdata = {
23     {
24     .end_of_table              = false,
25     .manufacturer_code         = 0xc2,
26     .device_code               = 0xf0,
27     .cell_technology           = NAND_DEVICE_CELL_TECH_SLC,
28     .chip_size_in_bytes        = 64LL*SZ_1M,
29     .block_size_in_pages       = 64,
30     .page_total_size_in_bytes  = 2*SZ_1K + 64,
31     .ecc_strength_in_bits      = 1,
32     .ecc_size_in_bytes         = 512,
33     .data_setup_in_ns          = 5,
34     .data_hold_in_ns           = 5,
35     .address_setup_in_ns       = 15,
36     .gpmi_sample_delay_in_ns   = 6,
37     .tREA_in_ns                = 20,
38     .tRLOH_in_ns               = -1,
39     .tRHOH_in_ns               = -1,
40     "MX30LF1208AA",
41     },
```

The image below shows the information you must add for the 1Gbit MX30LF1G08AA.

```
81      .end_of_table              = false,
82      .manufacturer_code         = 0xc2,
83      .device_code               = 0xf1,
84      .cell_technology           = NAND_DEVICE_CELL_TECH_SLC,
85      .chip_size_in_bytes        = 128LL*SZ_1M,
86      .block_size_in_pages       = 64,
87      .page_total_size_in_bytes  = 2*SZ_1K + 64,
88      .ecc_strength_in_bits      = 1,
89      .ecc_size_in_bytes         = 512,
90      .data_setup_in_ns          = 5,
91      .data_hold_in_ns           = 5,
92      .address_setup_in_ns       = 15,
93      .gpmi_sample_delay_in_ns   = 6,
94      .tREA_in_ns                = 20,
95      .tRLOH_in_ns               = -1,
96      .tRHOH_in_ns               = -1,
97      "MX30LF1G08AA",
98      },
```

Now setup the initialization function for the Macronix NAND flash devices. Here we build a new function naming **nand_device_info_fn_macronix**.

```
2152  static struct nand_device_info * __init nand_device_info_fn_macronix(const uint8_t id[])
2153  {
2154      /* SLC device only */
2155      /* Type 2 */
2156      return nand_device_info_search(nand_device_info_table_type_2,
2157                                     ID_GET_MFR_CODE(id), ID_GET_DEVICE_CODE(id));
2158  }
```

Make a new element in the **nand_device_mfr_directory** array and assign to .id variable the NAND_MFR_MACRONIX define and the nand_device_info_fn_macronix to the .fn variable.

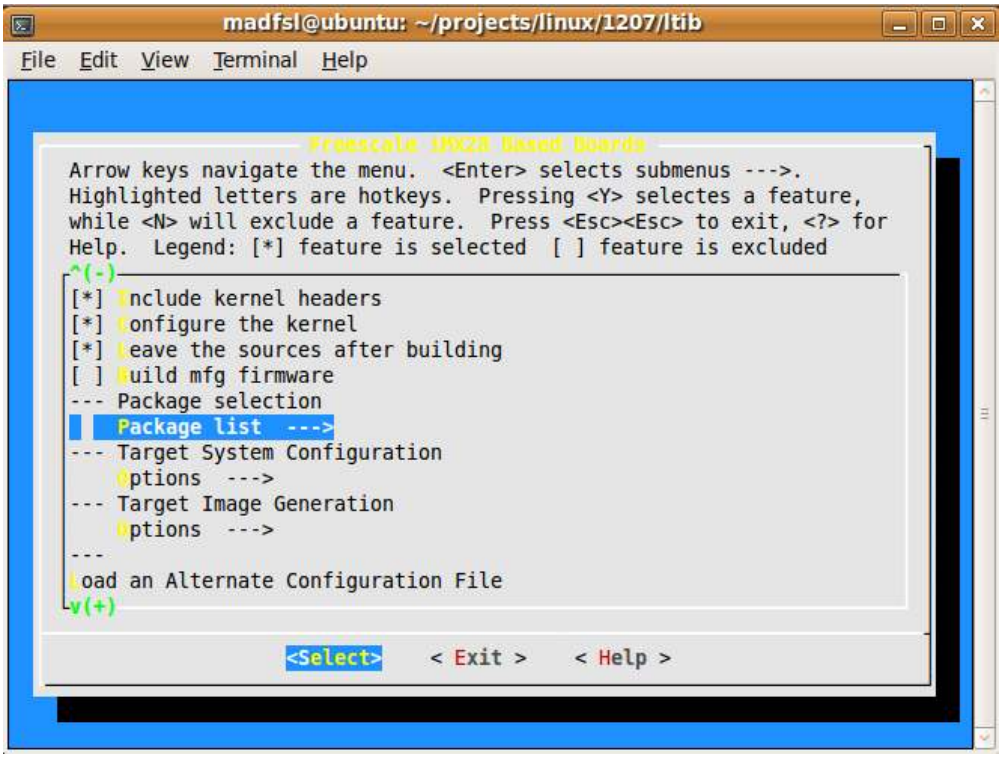
```
2312  static struct nand_device_mfr_info  nand_device_mfr_directory[] __initdata = {
2313      {
2314      .id = NAND_MFR_MACRONIX,
2315      .fn = nand_device_info_fn_macronix,
2316      },
```

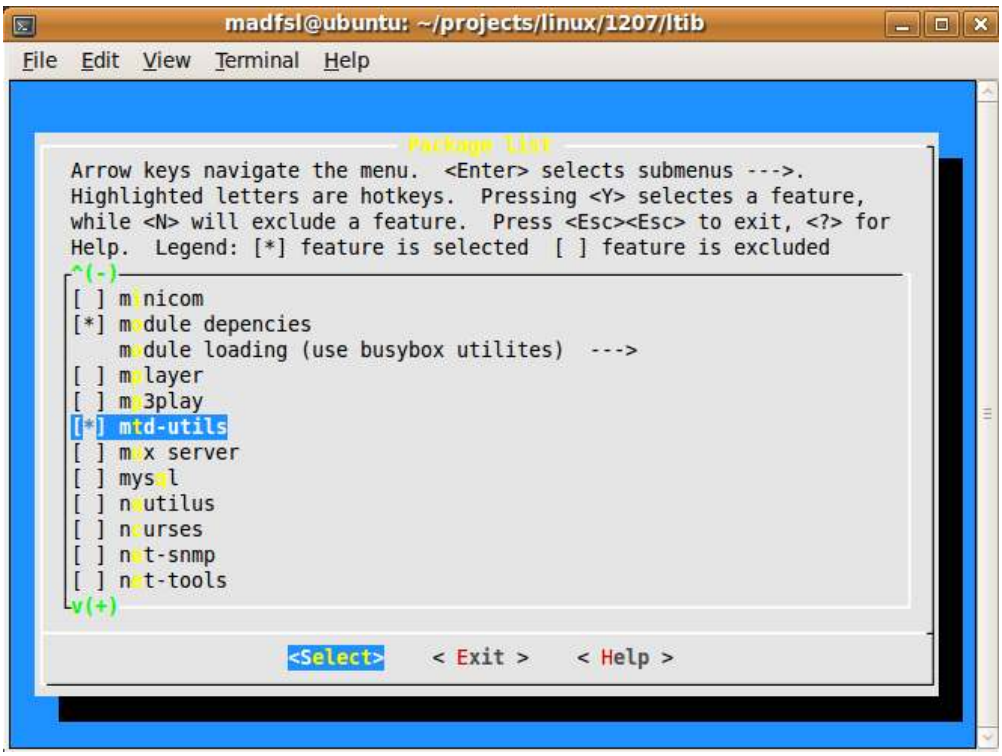
For other Linux versions, please refer to NAND driver patch in our website under the support area for more information.

Linux Kernel Configuration

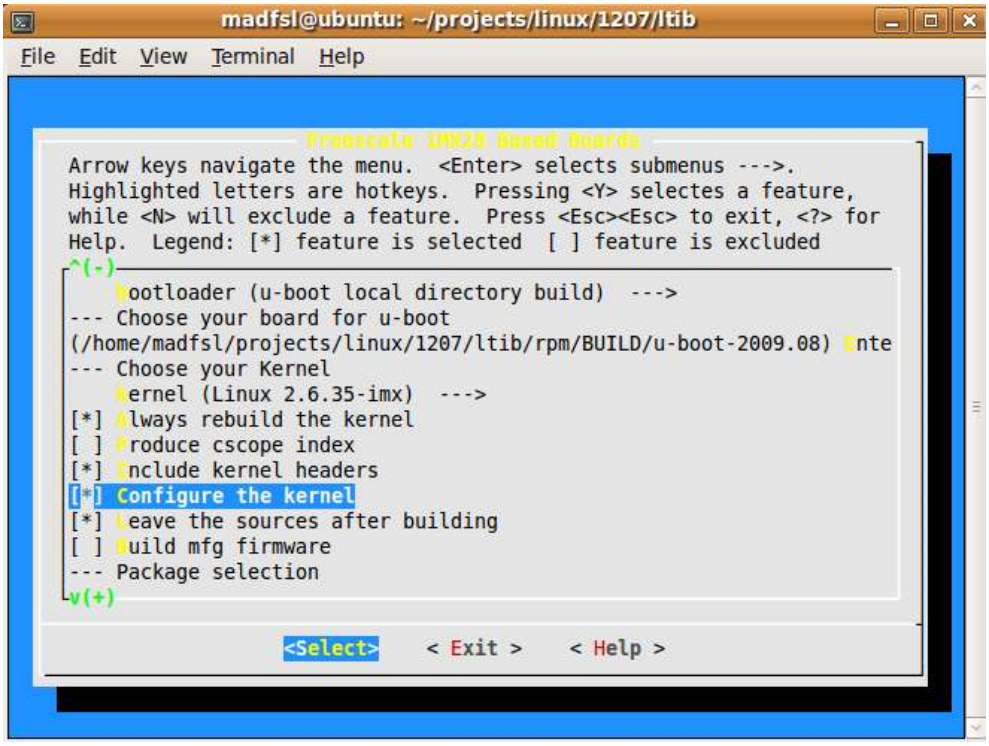
Run ltib script with argument “-m config” to configure the board and enable “**mtd-utils**”, which you can find it in “**Package list**”. The tool is useful in testing flash memory.

```
# ./ltib -m config
```





Choose “Configure the kernel” then exit and save.

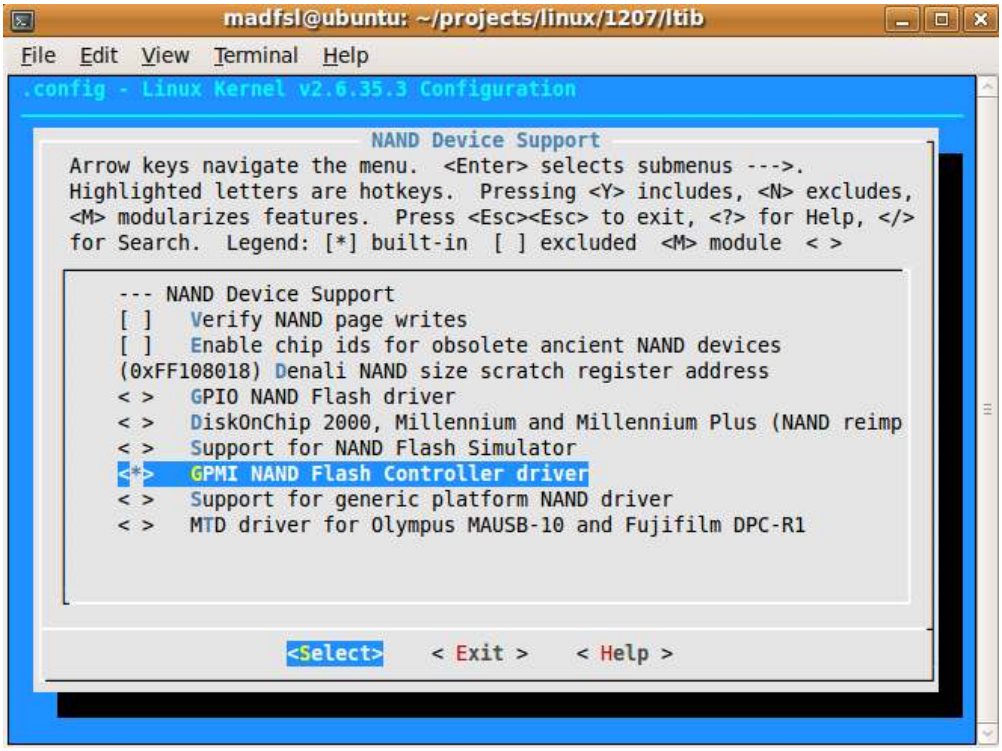


Run ltib to configure Linux kernel and rebuild kernel.

```
# ./ltib
```

In “menuconfig” window, you may need to select the following options for supporting i.MX28’s NAND controller.

<*> Device Drivers ->
    <*> Memory Technology Device (MTD) support ->
        <*> NAND Device Support ->
            <*> GPMI NAND Flash Controller driver



Then, you can follow i.MX28 setup steps to build kernel and rootfs to SD card.

```
# ./ltib -p boot_stream.spec -f
# umount /dev/sdc
# ./mk_mx28_sd /dev/sdc  // sdc is your SD card device
```

Test NAND Device with Different File Systems

Insert SD card boot i.MX28 from SD card with switch setting “1001”. You could check NAND device and GPMI controller’s working status with following command.

```
# cat /proc/mtd
dev:    size    erasesize    name
mtd0: 01400000 00020000 “gpmi-nfc-0-boot”
mtd1: 06c00000 00020000 “gpmi-nfc-0-general-use”
```

You could directly mount NAND device on **mtdblock** with ext2 file system. So how, “mtdblock” is a bad performance solution that is suitable to sequential access data, so we suggest you to try UBIFS.

```
# mkfs.ext2 /dev/mtdblock1
# mount -t ext2 /dev/mtdblock1 /mnt
# umount /mnt
```

Or you could mount NAND device with Journaling Flash File System (**JFFS2**).

```
# mount -t jffs2 /dev/mtdblock1 /mnt
# umount /mnt
```

Or you could mount with Unsorted Block Image File System (**UBIFS**).

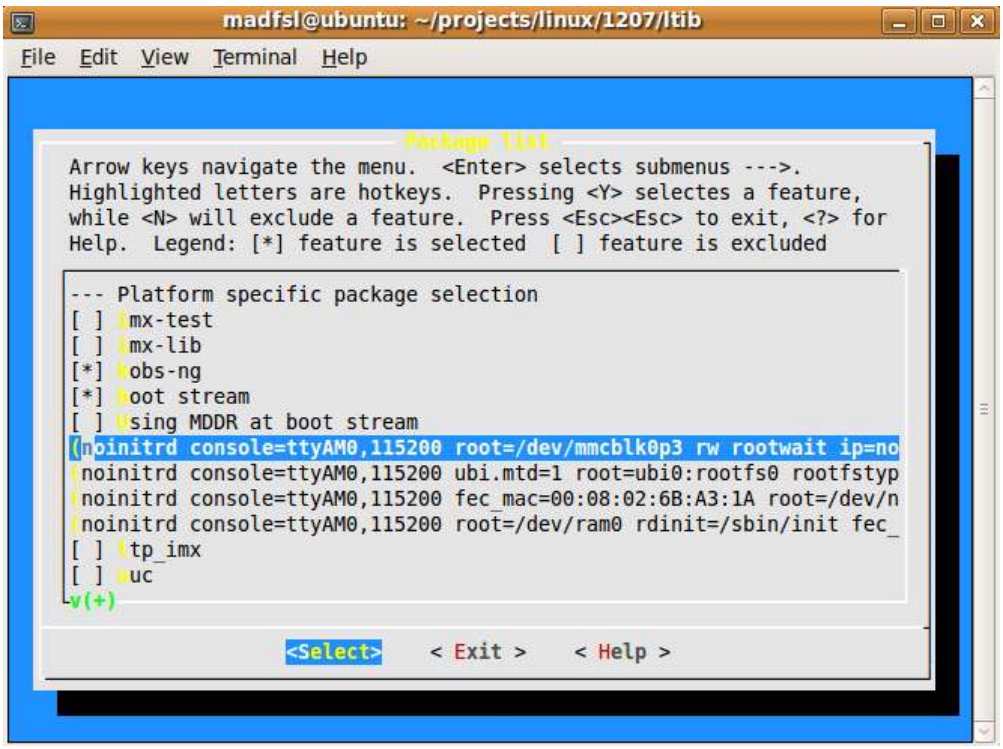
```
# flash_eraseall /dev/mtd1
# ubiattach /dev/ubi_ctrl -m 1
# ubimkvol /dev/ubi0 -N rootfs0 -s 100MiB
# mount -t ubifs ubi0:rootfs0 /mnt
  // or “mount -t ubifs ubi0_0 /mnt”
# umount /mnt
# ubirmvol /dev/ubi0 -n 0          // remove UBI volumn
# ubidetach /dev/ubi_ctrl -m 1     // un-link UBI manager
```

Boot Linux from NAND Flash

Configure the board with command "./ltib -m config" again. Select "Package list" -> "**Boot stream**", and retype the boot stream from the following (1) to (2), which means your root position is on MTD block device instead of MMC device (SD card). And the root file system is also need to be changed to flash file system such as UBIFS. The same procedure, if you want to boot with JFFS2 root file system, you'll need to try option (3).

(1) noinitrd console=ttyAM0, 115200 root=/dev/mmcblk0p3 rw rootwait ip=none gpmi
(2) noinitrd console=ttyAM0, 115200 ubi.mtd=1 root=ubi0:rootfs0 rootfstype=ubifs rw gpmi
(3) noinitrd console=ttyAM0, 115200 root=/dev/mtdblock1 rootfstype=jffs2 rw gpmi

Please refer to the following graph. After your setting, please save and exit.



Rebuild kernel with "./ltib" and "./ltib –p boot_stream.spec -f". Then copy the new kernel and root file system to SD card.

```
# cd ~/projects/linux/1008/ltib
# cp rootfs/boot/imx28_linux.sb /media/disk-1
# cp -rf rootfs tempfs
# rm -rf tempfs/boot              // no need boot directory
# tar -cf nandfs.tar tempfs/*
# cp nandfs.tar /media/disk-1     // copy rootfs to SD card
```

Boot from SD card and copy kernel and root file system to NAND device with UBIFS. These steps are similar in booting with JFFS2.

```
# flash_eraseall /dev/mtd0
# kobs-ng init imx28_linux.sb
  // copy kernel to NAND partition 0
# flash_eraseall /dev/mtd1
# ubiattach /dev/ubi_ctrl -m 1
# ubimkvol /dev/ubi0 -N rootfs0 -s 100MiB
# mount -t ubifs ubi0:rootfs0 /mnt
# tar -xf nandfs.tar -C /mnt   // copy rootfs to NAND
# unmount /mnt
# ubirmvol /dev/ubi0 -n 0
# ubidetach /dev/ubi_ctrl -m 1
```

Halt system and change switch to “0100”, then you can boot from NAND. If you want to boot from NAND with JFFS2 root file system, you can replace above code to the followings.

```
# flash_eraseall /dev/mtd0
# kobs-ng init imx28_linux.sb
# flash_eraseall /dev/mtd1
# mount -t jffs2 /mnt
# tar -xf nandfs.tar -C /mnt
# unmount /mnt
```

For further information or questions, feel free to your local Macronix FAE or contact us directly: flash_model@mxic.com.tw