MX28 芯片下 Linux-2.6.35 移植步骤

----2018 年 9 月

有关于 linux 的移植资料，因篇幅问题，自行在网上寻找，这里详细记录 linux 内核移植到 freescale mx28 芯片的配置修改步骤，其中需要添加的文件自行去移植好的源文件查看。

1、下载 linux-2.6.35。http://git.freescale.com/git/cgit.cgi

2、进入文件目录下 Makefile 文件，修改运行平台和交叉编译工具链路径，或编写脚本直接执行

```
# Note: Some architectures assign CROSS_COMPILE in their arch/*/M
export KBUILD_BUILDHOST := $(SUBARCH)
ARCH            ?= $(SUBARCH)
CROSS_COMPILE   ?= $(CONFIG_CROSS_COMPILE:"%"=%)
```

改为：

```
export KBUILD_BUILDHOST := $(SUBARCH)
ARCH            ?= arm
CROSS_COMPILE   ?= /usr/local/arm-fsl-linux-gnueabi/bin/arm-fsl-linux-gnue
abi-
```

编写的脚本为：

```
#!/bin/bash

export ARCH=arm
export CROSS_COMPILE=/usr/local/arm-fsl-linux-gnueabi/bin/arm-fsl-linux-gnue
abi-
make clean
make mx28evk_defconfig
make uImage
```

3、板级依赖文件在 arch/arm/目录下，配置文件在 arch/arm/configs/下

4、用的板子是周立功的 EasyARM-iMX28，选择 mach-mx28 进行修改。

5、在 arch/arm/mach-mx28/device.c 中

(1)删除或注释 mxs_mmc_hw_init_ssp0（）中部分代码

```
#if 0
        /* Configure write protect GPIO pin */
        ret = gpio_request(MMC0_WP, "mmc0_wp");
        if (ret)
                goto out_wp;

        gpio_set_value(MMC0_WP, 0);
        gpio_direction_input(MMC0_WP);

        /* Configure POWER pin as gpio to drive power to MMC slot */
        ret = gpio_request(MMC0_POWER, "mmc0_power");
        if (ret)
                goto out_power;

        gpio_direction_output(MMC0_POWER, 0);
        mdelay(100);

        return 0;

out_power:
        gpio_free(MMC0_WP);
out_wp:
#endif
```

（2）删除或注释 mxs_mmc_hw_release_ssp0（）中代码

```
static void mxs_mmc_hw_release_ssp0(void)
{
//      gpio_free(MMC0_POWER);
//      gpio_free(MMC0_WP);

}
```

（3）仿照 SSP1 函数，增加 SSP2 操作函数

```
static int mxs_mmc_get_wp_ssp2(void)
{
        return 0;
}

static int mxs_mmc_hw_init_ssp2(void)
{
        int ret = 0;
        return ret;
}
static void mxs_mmc_hw_release_ssp2(void)
{
}
static void mxs_mmc_cmd_pullup_ssp2(int enable)
{
        mxs_set_pullup(PINID_SSP2_MOSI, enable, "mmc2_cmd");
}
```

```
static unsigned long mxs_mmc_setclock_ssp2(unsigned long hz)
{
        struct clk *ssp = clk_get(NULL, "ssp.2"), *parent;

        if (hz > 1000000)
                parent = clk_get(NULL, "ref_io.1");
        else
                parent = clk_get(NULL, "xtal.0");

        clk_set_parent(ssp, parent);
        clk_set_rate(ssp, 2 * hz);
        clk_put(parent);
        clk_put(ssp);

        return hz;
}
```

（4）修改结构体 mmc0_data{}和 mmc1_data{}

```
static struct mxs_mmc_platform_data mmc0_data = {
        .hw_init        = mxs_mmc_hw_init_ssp0,
        .hw_release     = mxs_mmc_hw_release_ssp0,
        .get_wp         = mxs_mmc_get_wp_ssp0,
        .cmd_pullup     = mxs_mmc_cmd_pullup_ssp0,
        .setclock       = mxs_mmc_setclock_ssp0,
//      .caps           = MMC_CAP_4_BIT_DATA | MMC_CAP_8_BIT_DATA
        .caps           = MMC_CAP_4_BIT_DATA | MMC_CAP_DATA_DDR,

        .min_clk        = 400000,
        .max_clk        = 48000000,
        .read_uA        = 50000,
        .write_uA       = 70000,
        .clock_mmc = "ssp.0",
        .power_mmc = NULL,
//      .fastpath_sz = 1024,
};
```

```
static struct mxs_mmc_platform_data mmc1_data = {
        .hw_init        = mxs_mmc_hw_init_ssp1,
        .hw_release     = mxs_mmc_hw_release_ssp1,
        .get_wp         = mxs_mmc_get_wp_ssp1,
        .cmd_pullup     = mxs_mmc_cmd_pullup_ssp1,
        .setclock       = mxs_mmc_setclock_ssp1,
        .caps           = MMC_CAP_4_BIT_DATA | MMC_CAP_8_BIT_DATA
                                | MMC_CAP_DATA_DDR,
        .min_clk        = 400000,
        .max_clk        = 48000000,
        .read_uA        = 50000,
        .write_uA       = 70000,
        .clock_mmc = "ssp.1",
        .power_mmc = NULL,
//      .fastpath_sz = 1024,
};
```

(5)仿照 mmc1 结构体，增加 mmc2 结构体

```
static struct mxs_mmc_platform_data mmc2_data = {
        .hw_init        = mxs_mmc_hw_init_ssp2,
        .hw_release     = mxs_mmc_hw_release_ssp2,
        .get_wp         = mxs_mmc_get_wp_ssp2,
        .cmd_pullup     = mxs_mmc_cmd_pullup_ssp2,
        .setclock       = mxs_mmc_setclock_ssp2,
        .caps           = MMC_CAP_4_BIT_DATA ,
        .min_clk        = 400000,
        .max_clk        = 48000000,
        .read_uA        = 50000,
        .write_uA       = 70000,
        .clock_mmc = "ssp.2",
        .power_mmc = NULL,
};
```

```
static struct resource mmc2_resource[] = {
        {
                .flags  = IORESOURCE_MEM,
                .start  = SSP2_PHYS_ADDR,
                .end    = SSP2_PHYS_ADDR + 0x2000 - 1,
        },
        {
                .flags  = IORESOURCE_DMA,
                .start  = MXS_DMA_CHANNEL_AHB_APBH_SSP2,
                .end    = MXS_DMA_CHANNEL_AHB_APBH_SSP2,
        },
        {
                .flags  = IORESOURCE_IRQ,
                .start  = IRQ_SSP2_DMA,
                .end    = IRQ_SSP2_DMA,
        },
{
                .flags  = IORESOURCE_IRQ,
                .start  = IRQ_SSP2,
                .end    = IRQ_SSP2,
        },
};
```

(6)仿照 SSP2 结构体，增加 SSP3 结构体。改 mx28_init_spi 函数中的 ssp2_resources 改为 ssp3_resources

```
static struct mxs_spi_platform_data spi3_data = {
        .clk = "ssp.3",
};
static struct resource ssp3_resources[] = {
        {
                .start  = SSP3_PHYS_ADDR,
                .end    = SSP3_PHYS_ADDR + 0x2000 - 1,
                .flags  = IORESOURCE_MEM,
        }, {
                .start  = MXS_DMA_CHANNEL_AHB_APBH_SSP3,
                .end    = MXS_DMA_CHANNEL_AHB_APBH_SSP3,
                .flags  = IORESOURCE_DMA,
        }, {
                .start  = IRQ_SSP3_DMA,
                .end    = IRQ_SSP3_DMA,
                .flags  = IORESOURCE_IRQ,
        }, {
                .start  = IRQ_SSP3,
                .end    = IRQ_SSP3,
                .flags  = IORESOURCE_IRQ,
        },
};
```

```
static void __init mx28_init_spi(void)
{
        struct platform_device *pdev;

        pdev = mxs_get_device("mxs-spi", 0);
        if (pdev == NULL || IS_ERR(pdev))
                return;
        pdev->resource = ssp3_resources;
        pdev->num_resources = ARRAY_SIZE(ssp3_resources);
        pdev->dev.platform_data = &spi_data;

        mxs_add_device(pdev, 3);
}
```

(7)改写 audio_clk_init()函数

```
static int audio_clk_init(struct clk *clk)
{
//      struct clk *clk;
//      struct clk *clk1;
        struct clk *pll_clk;
        struct clk *saif_mclk0;
        struct clk *saif_mclk1;
        int ret = -EINVAL;

//      if (audio_plat_data.inited)
//              return 0;
//      clk = clk_get(NULL, "saif.0");
        if (IS_ERR(clk)) {
                pr_err("%s:failed to get clk\n", __func__);
                goto err_clk_init;
        }
        pll_clk = clk_get(NULL, "pll.0");
        if (IS_ERR(pll_clk)) {
                pr_err("%s:failed to get pll_clk\n", __func__);
                goto err_clk_init;
        }
        saif_mclk0 = clk_get(NULL, "saif_mclk.0");
        if (IS_ERR(saif_mclk0)) {
```

```
        ret = clk_set_parent(clk, pll_clk);
        if (ret) {
                pr_err("%s:failed to set parent clk\n", __func__);
                goto err_clk_init;
        }

        ret = 0;
        /*set a default freq of 12M to sgtl5000*/
        clk_set_rate(clk, 12000000);
        clk_enable(clk);
        /*set the saif clk mux, saif0/saif1 both use saif0 clk*/
        __raw_writel(BF_DIGCTL_CTRL_SAIF_CLKMUX_SEL(0x2), \
                        IO_ADDRESS(DIGCTL_PHYS_ADDR) + HW_DIGCTL_CTRL);

        /*enable saif0/saif1 clk output*/
        clk_enable(saif_mclk0);
        clk_enable(saif_mclk1);
err_clk_init:
        return ret;
}
```

同理，修改 audio_clk_finit()

```
static int audio_clk_finit(void)
{
        struct clk *saif_clk;
//      struct clk *saif_clk1;
        struct clk *saif_mclk0;
        struct clk *saif_mclk1;
        int ret = 0;

//      if (audio_plat_data.inited == 0)
//              return 0;
        saif_clk = clk_get(NULL, "saif.0");
        if (IS_ERR(saif_clk)) {
                pr_err("%s:failed to get saif_clk\n", __func__);
                ret = -EINVAL;
                goto err_clk_finit;
        }
        clk_disable(saif_clk);

        saif_mclk0 = clk_get(NULL, "saif_mclk.0");
        if (IS_ERR(saif_mclk0)) {
                pr_err("%s:failed to get saif_mclk\n", __func__);
                goto err_clk_finit;
        }
```

```
        if (IS_ERR(saif_mclk0)) {
                pr_err("%s:failed to get saif_mclk\n", __func__);
                goto err_clk_finit;
        }
        clk_disable(saif_mclk0);

        saif_mclk1 = clk_get(NULL, "saif_mclk.1");
        if (IS_ERR(saif_mclk1)) {
                pr_err("%s:failed to get saif_mclk\n", __func__);
                goto err_clk_finit;
        }
        clk_disable(saif_mclk1);


err_clk_finit:
        return ret;
}
```

(7)修改 mx28_init_audio()

```
void __init mx28_init_audio(void)
{       struct platform_device *pdev;
        pdev = mxs_get_device("mxs-sgtl5000", 0);
        if (pdev == NULL || IS_ERR(pdev))
                return;
        mxs_add_device(pdev, 3);
        audio_plat_data.inited = 0;
        audio_plat_data.saif_mclock = clk_get(NULL, "saif.0");
//      audio_plat_data.saif_mclock1 = clk_get(NULL, "saif.1");
//      audio_plat_data.init = audio_clk_init;
//      audio_plat_data.finit = audio_clk_finit;
        audio_clk_init(audio_plat_data.saif_mclock);
        pdev->dev.platform_data = &audio_plat_data;
}
```

(7) 新增结构体数组 hsadc_resource[]

```
#if defined(CONFIG_MXS_HSADC) || \
    defined(CONFIG_MXS_HSADC_MODULE)
static struct resource hsadc_resource[] = {
        {
                .flags = IORESOURCE_MEM,
                .start = HSADC_PHYS_ADDR,
                .end   = HSADC_PHYS_ADDR + 0xC0 - 1,
        },
        {
                .flags = IORESOURCE_MEM,
                .start = PWM_PHYS_ADDR,
                .end   = PWM_PHYS_ADDR + 0x120 - 1,
        },
        {       //device irq
                .flags = IORESOURCE_IRQ,
                .start = IRQ_HSADC,
                .end   = IRQ_HSADC,
        },
        {       //dma irq
                .flags = IORESOURCE_IRQ,
                .start = IRQ_HSADC_DMA,
                .end   = IRQ_HSADC_DMA,
        {
                .flags = IORESOURCE_DMA,
                .start = MXS_DMA_CHANNEL_AHB_APBH_HSADC,
                .end   = MXS_DMA_CHANNEL_AHB_APBH_HSADC,
        },
};
static void __init mx28_init_hsadc(void)
{
        struct platform_device *pdev;

        pdev = mxs_get_device("mxs-hsadc", 0);
        if (pdev == NULL || IS_ERR(pdev))
                return;
        pdev->resource = hsadc_resource;
        pdev->num_resources = ARRAY_SIZE(hsadc_resource);
        mxs_add_device(pdev, 3);
}
#else
static void __init mx28_init_hsadc(void)
{
}
#endif
```

（8）将新增的 mx28_init_hsadc 加入到 mx28_device_init()中

```
        mx28_init_perfmon();
        mx28_init_otp();
        mx28_init_hsadc();//add
        return 0;
```

6、在 arch/arm/mach-mx28/mx28evk_pins.c 中主要是修改与

板子对应的各种引脚功能。对照着硬件设计原理图修改，需要修改的太多，代码放后面附录中。

7、在 drivers/mtd/nand/nand_base.c 修改 nand_erase_nand()

```
        /*      if (chip->bbt) {
                        int i = (page / pages_per_block) << 1;
                        chip->bbt[i >> 3] |= 0x03 << (i & 0x6);
                        mtd->ecc_stats.badblocks++;
                }*/
                goto erase_exit;
        }
```

在末尾处修改

```
//module_init(nand_base_init);
//module_exit(nand_base_exit);
subsys_initcall(nand_base_init);

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Steven J. Hill <sjhill@realityd
<tglx@linutronix.de>");
MODULE_DESCRIPTION("Generic NAND flash driver
```

8、在 drivers/mtd/nand/nand_bbt.c 修改 check_pattern()

```
end = paglen + td->offs;
```

改为：

```
end =  td->offs;
```

修改 search_bbt()

```
td->version[i] = buf[mtd->writesize
+ td->veroffs];
```

改为：

```
        td->version[i] = buf[td->veroffs];
}
```

9、在 drivers/mtd/nand/nand_device_info.c 修改

```
static struct nand_device_info nand_device_info_table_type_2[] __initdata = {
        {//add
        .end_of_table              = false,
        .manufacturer_code         = 0x01,
        .device_code               = 0xf1,
        .cell_technology           = NAND_DEVICE_CELL_TECH_SLC,
        .chip_size_in_bytes        = 128LL*SZ_1M,
        .block_size_in_pages       = 64,
        .page_total_size_in_bytes  = 2*SZ_1K + 64,
        .ecc_strength_in_bits      = 4,
        .ecc_size_in_bytes         = 512,
        .data_setup_in_ns          = 20,
        .data_hold_in_ns           = 10,
        .address_setup_in_ns       = 20,
        .gpmi_sample_delay_in_ns   = 6,
        .tREA_in_ns                = -1,
        .tRLOH_in_ns               = -1,
        .tRHOH_in_ns               = -1,
        "S34ML01G2",
        },
```

```
{       //add
        .end_of_table              = false,
        .manufacturer_code         = 0xc2,
        .device_code               = 0xda,
        .cell_technology           = NAND_DEVICE_CELL_TECH_SLC,
        .chip_size_in_bytes        = 256LL*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,
        .data_setup_in_ns          = 20,
        .data_hold_in_ns           = 10,
        .address_setup_in_ns       = 20,
        .gpmi_sample_delay_in_ns   = 6,
        .tREA_in_ns                = -1,
        .tRLOH_in_ns               = -1,
        .tRHOH_in_ns               = -1,
        "MX30LF2G",
        },
```

```
        },
        {//add
        .end_of_table              = false,
        .manufacturer_code         = 0xc2,
        .device_code               = 0xf1,
        .cell_technology           = NAND_DEVICE_CELL_TECH_SLC,
        .chip_size_in_bytes        = 128LL*SZ_1M,
        .block_size_in_pages       = 64,
        .page_total_size_in_bytes  = 2*SZ_1K + 64,
        .ecc_strength_in_bits      = 4,
        .ecc_size_in_bytes         = 512,
        .data_setup_in_ns          = 20,
        .data_hold_in_ns           = 10,
        .address_setup_in_ns       = 20,
        .gpmi_sample_delay_in_ns   = 6,
        .tREA_in_ns                = -1,
        .tRLOH_in_ns               = -1,
        .tRHOH_in_ns               = -1,
        "MX30LF1G08AA",
        },
```

```
//add
static struct nand_device_info * __init nand_device_info_fn_mx(const uint8_
t id[])
{
        /* Check for an SLC device. */
        if (ID_GET_CELL_TYPE_CODE(id) == ID_CELL_TYPE_CODE_SLC) {
                /* Type 2 */
                return nand_device_info_search(nand_device_info_table_type_
2,
                                ID_GET_MFR_CODE(id), ID_GET_DEVICE_CODE(id)
);
        }
}
```

```
static struct nand_device_mfr_info  nand_device_mfr_directory[] __initdata = {
        {//add
        .id = 0x01,
        .fn = nand_device_info_fn_mx,
        },
        {
        .id = NAND_MFR_TOSHIBA,
        .fn = nand_device_info_fn_toshiba,
        },
        {
        .id = NAND_MFR_SAMSUNG,
        .fn = nand_device_info_fn_samsung,
        },
        {//add
        .id = NAND_MFR_MX30LF1G08,
        .fn = nand_device_info_fin_mx,
        },
```

10、在 drivers/mtd/nand/nand_ids.c 修改

```
struct nand_manufacturers nand_manuf_ids[] = {
        {0x01, "Spansion"},     //add
        {NAND_MFR_TOSHIBA, "Toshiba"},
        {NAND_MFR_SAMSUNG, "Samsung"},
        {NAND_MFR_FUJITSU, "Fujitsu"},
        {NAND_MFR_NATIONAL, "National"},
        {NAND_MFR_RENESAS, "Renesas"},
        {NAND_MFR_STMICRO, "ST Micro"},
        {NAND_MFR_HYNIX, "Hynix"},
        {NAND_MFR_MICRON, "Micron"},
        {NAND_MFR_AMD, "AMD"},
        {NAND_MFR_SANDISK, "SanDisk"},
        {NAND_MFR_INTEL, "Intel"},
        {0x0, "Unknown"}
};
```

11、在 drivers/mtd/nand/gpmi-nfc/gpmi-nfc.h 修改

```
 */

//add
struct physical_geometry {
        unsigned int  chip_count;
        uint64_t      chip_size_in_bytes;
        unsigned int  block_size_in_bytes;
        unsigned int  page_data_size_in_bytes;
        unsigned int  page_oob_size_in_bytes;
};
```

```
struct gpmi_nfc_data {

        /* System Interface */
        struct device                   *dev;
        struct platform_device          *pdev;
        struct gpmi_nfc_platform_data   *pdata;

        /* Resources */
        struct resources                resources;

        /* Flash Hardware */
        struct nand_device_info         device_info;
        //add
        struct physical_geometry        physical_geometry;//add

        /* NFC HAL */
        struct nfc_hal                  *nfc;
        struct nfc_geometry             nfc_geometry;

        /* Boot ROM Helper */
```

```
struct boot_rom_helper {
        const unsigned int  version;
        const char          *description;
        const int           swap_block_mark;
        int  (*set_geometry)            (struct gpmi_nfc_data *);
//add
        int  (*check_transcription_stamp)(struct gpmi_nfc_data *);
        int  (*write_transcription_stamp)(struct gpmi_nfc_data *);
};
```

11、在 drivers/mtd/nand/gpmi-nfc/gpmi-nfc-hal-common.c 修改

```
 */
int gpmi_nfc_set_geometry(struct gpmi_nfc_data *this)
{
        struct device             *dev      = this->dev;
        //modify
        struct physical_geometry  *physical = &this->physical_geometry;
        struct nfc_geometry       *geometry = &this->nfc_geometry;
        struct boot_rom_helper    *rom      =  this->rom
        unsigned int              metadata_size;
        unsigned int              status_size;
        unsigned int              chunk_data_size_in_bits;
```

修改结构体指针定义，后面的所有使用都修改（不一一列举）

```
        /* Compute the page size, include page and oob. */
//modify
        geometry->page_size_in_bytes =
                        physical->page_data_size_in_buyes +
                        physical->page_oob_size_in_bytes  ;

        /*
         * Compute the total number of ECC chunks in a page. This incl
```

```
        */
//modify
        geometry->ecc_chunk_count = physical->page_data_size_in_bytes /
                                    geometry->ecc_chunk_size_in_bytes;

        /*
         * We use the same ECC strength for all chunks, including the first
```

```
         * only needs what's required to hold the data.
         */
//modify
        geometry->payload_size_in_bytes = physical->page_data_size_in_bytes;

        /*
         * In principle, computing the auxiliary buffer geometry is NFC
```

修改第二个函数：后面的所有使用都修改（不一一列举）

```
 *                    the results of our calculations.
 */
int gpmi_nfc_compute_hardware_timing(struct gpmi_nfc_data *this,
                                     struct gpmi_nfc_hardware_timing *hw)
{
        struct gpmi_nfc_platform_data  *pdata    =  this->pdata;
        struct nfc_hal                 *nfc      =  this->nfc;
//modify
        struct physical_geometry       *physical = &this->physical_geometry;
        struct gpmi_nfc_timing         target    = nfc->timing;
        bool           improved_timing_is_available;
        unsigned long  clock_frequency_in_hz;
        unsigned int   clock_period_in_ns;
```

```
         */
//modify
        if (physical->chip_count > 2) {
                target.data_setup_in_ns    += 10;
                target.data_hold_in_ns     += 10;
                target.address_setup_in_ns += 10;
        } else if (physical->chip_count > 1) {
                target.data_setup_in_ns    += 5;
                target.data_hold_in_ns     += 5;
                target.address_setup_in_ns += 5;
        }

        /* Check if improved timing information is available. */
        improved_timing_is_available =
```

12、在 drivers/mtd/nand/gpmi-nfc/gpmi-nfc-main.c 修改

```
 */
static ssize_t show_device_numchips(struct device *dev,
                                    struct device_attribute *attr, char *buf)
{
        struct gpmi_nfc_data      *this     = dev_get_drvdata(dev);
//      struct nand_chip        *nand = &this->mil.nand;
        struct physical_geometry  *physical = &this->physical_geometry;

        return sprintf(buf, "%d\n", physical->chip_count);
}

/**
```

```
static DEVICE_ATTR(physical_geometry, 0444, show_device_physical_geometry, 0);  //add
static DEVICE_ATTR(nfc_info          , 0444, show_device_nfc_info          , 0);
static DEVICE_ATTR(nfc_geometry      , 0444, show_device_nfc_geometry      ,
```

```
static struct device_attribute *device_attributes[] = {
        &dev_attr_report,
        &dev_attr_numchips,
        &dev_attr_physical_geometry,    //add
        &dev_attr_nfc_info,
        &dev_attr_nfc_geometry,
        &dev_attr_rom_geometry,
        &dev_attr_mtd_nand_info,
        &dev_attr_mtd_info,
        &dev_attr_invalidate_page_cache,
        &dev_attr_mark_block_bad,
        &dev_attr_ignorebad
```

13、在 drivers/mtd/nand/gpmi-nfc/gpmi-nfc-mil.c 修改

```
//add bad block table
static uint8_t bbt_main_pattern[]   = { 'B', 'b', 't', '0' };
static uint8_t bbt_mirror_pattern[] = { '1', 't', 'b', 'B' };

static struct nand_bbt_descr bbt_main_descriptor = {
        .options =
                NAND_BBT_LASTBLOCK |
                NAND_BBT_CREATE    |
                NAND_BBT_WRITE     |
                NAND_BBT_2BIT      |
                NAND_BBT_VERSION   |
                NAND_BBT_PERCHIP,
        .offs       = 1,
        .len        = 4,
        .veroffs    = 5,
        .maxblocks  = 4,
        .pattern    = bbt_main_pattern,
};
```

```
static struct nand_bbt_descr bbt_mirror_descriptor = {
        .options =
                NAND_BBT_LASTBLOCK |
                NAND_BBT_CREATE    |
                NAND_BBT_WRITE     |
                NAND_BBT_2BIT      |
                NAND_BBT_VERSION   |
                NAND_BBT_PERCHIP,
        .offs       = 1,
        .len        = 4,
        .veroffs    = 5,
        .maxblocks  = 4,
        .pattern    = bbt_mirror_pattern,
};

/**
```

```
static int mil_ecc_read_oob(struct mtd_info *mtd, struct nand_chip *nand,
                                                        int page, int sndcm
d)
{
        struct gpmi_nfc_data      *this     = nand->priv;
//add
        struct physical_geometry  *physical = &this->physical_geometry;
        struct boot_rom_helper    *rom      =  this->rom;

        DEBUG(MTD_DEBUG_LEVEL2, "[gpmi_nfc ecc_read_oob] "
                "page: 0x%06x, sndcmd: %s\n", page, sndcmd ? "Yes" : "No");

        /* clear the OOB buffer */
        memset(nand->oob_poi, ~0, mtd->oobsize);

        /* Read out the conventional OOB. */
        nand->cmdfunc(mtd, NAND_CMD_READ0,
                        physical->page_data_size_in_bytes, page);//modify
        nand->read_buf(mtd, nand->oob_poi, mtd->oobsize);

        /*
         * Now, we want to make sure the block mark is correct. In the
```

在以下函数中添加 struct physical_geometry *physical = &this->physical_geometry;
将所有的 mtd->writesize 改成 physical->page_data_size_in_bytes.

```
 */
static int mil_ecc_write_oob(struct mtd_info *mtd,
                                        struct nand_chip *nand, int page)
{
```

增加 mil_set_physical_geometry 函数`

```
//add function
static int mil_set_physical_geometry(struct gpmi_nfc_data  *this)
{
        struct mil                  *mil      = &this->mil;
        struct physical_geometry    *physical = &this->physical_geometry;
        struct nand_chip            *nand     = &mil->nand;
        struct nand_device_info     *info     = &this->device_info;
        unsigned int                block_size_in_pages;
        unsigned int                chip_size_in_blocks;
        unsigned int                chip_size_in_pages;
        uint64_t                    medium_size_in_bytes;

        /*
         * Record the number of physical chips that MTD found.
         */

        physical->chip_count = nand->numchips;

        /*
```

```
        physical->page_data_size_in_bytes =
                                1 << (fls(info->page_total_size_in_bytes) -
1);

        physical->page_oob_size_in_bytes =
                                info->page_total_size_in_bytes -
                                        physical->page_data_size_in_bytes;
```

```
         */
        physical->block_size_in_bytes =
                physical->page_data_size_in_bytes * info->block_size_in_pag
es;

        /* Get the chip size. */

        physical->chip_size_in_bytes = info->chip_size_in_bytes;

        /* Compute some interesting facts. */

        block_size_in_pages  =
                        physical->block_size_in_bytes >>
                                (fls(physical->page_data_size_in_bytes) - 1
);
        chip_size_in_pages   =
                        physical->chip_size_in_bytes >>
                                (fls(physical->page_data_size_in_bytes) - 1
);
        chip_size_in_blocks  =
                        physical->chip_size_in_bytes >>
                                (fls(physical->block_size_in_bytes) - 1);
        medium_size_in_bytes =
```

```
        /* Report. */

        #if defined(DETAILED_INFO)

        pr_info("----------------\n");
        pr_info("Physical Geometry\n");
        pr_info("----------------\n");
        pr_info("Chip Count              : %d\n", physical->chip_count);
        pr_info("Page Data Size in Bytes: %u (0x%x)\n",
                        physical->page_data_size_in_bytes,
                        physical->page_data_size_in_bytes);
        pr_info("Page OOB Size in Bytes : %u\n",
                        physical->page_oob_size_in_bytes);
        pr_info("Block Size in Bytes    : %u (0x%x)\n",
                        physical->block_size_in_bytes,
                        physical->block_size_in_bytes);
        pr_info("Block Size in Pages    : %u (0x%x)\n",
                        block_size_in_pages,
                        block_size_in_pages);
        pr_info("Chip Size in Bytes     : %llu (0x%llx)\n",
                        physical->chip_size_in_bytes,
                        physical->chip_size_in_bytes);
        pr_info("Chip Size in Pages     : %u (0x%x)\n",
```

```
        pr_info("Chip Size in Bytes     : %llu (0x%llx)\n",
                        physical->chip_size_in_bytes,
                        physical->chip_size_in_bytes);
        pr_info("Chip Size in Pages     : %u (0x%x)\n",
                        medium_size_in_bytes, medium_size_in_bytes);

        #endif

        /* Return success. */

        return 0;

}
```

增加 mil_set_mtd_geometry 函数

```
static int mil_set_mtd_geometry(struct gpmi_nfc_data *this)
{
        struct physical_geometry  *physical = &this->physical_geometry;
        struct mil                *mil      = &this->mil;
        struct nand_ecclayout     *layout   = &mil->oob_layout;
        struct nand_chip          *nand     = &mil->nand;
        struct mtd_info           *mtd      = &mil->mtd;

        /* Configure the struct nand_ecclayout. */

        layout->eccbytes          = 0;
        layout->oobavail          = physical->page_oob_size_in_bytes;
        layout->oobfree[0].offset = 0;
        layout->oobfree[0].length = physical->page_oob_size_in_bytes;

        /* Configure the struct mtd_info. */

        mtd->size        = nand->numchips * physical->chip_size_in_bytes;
        mtd->erasesize   = physical->block_size_in_bytes;
        mtd->writesize   = physical->page_data_size_in_bytes;
        mtd->ecclayout   = layout;
        mtd->oobavail    = mtd->ecclayout->oobavail;
```

```
        mtd->oobavail     = mtd->ecclayout->oobavail;
        mtd->oobsize      = mtd->ecclayout->oobavail + mtd->ecclayout->eccby
tes;
        mtd->subpage_sft = 0; /* We don't support sub-page writing. */

        /* Configure the struct nand_chip. */

        nand->chipsize          = physical->chip_size_in_bytes;
        nand->page_shift        = ffs(mtd->writesize) - 1;
        nand->pagemask          = (nand->chipsize >> nand->page_shift) - 1;
        nand->subpagesize       = mtd->writesize >> mtd->subpage_sft;
        nand->phys_erase_shift = ffs(mtd->erasesize) - 1;
        nand->bbt_erase_shift  = nand->phys_erase_shift;
        nand->oob_poi           = nand->buffers->databuf + mtd->writesize;
        nand->ecc.layout        = layout;
        if (nand->chipsize & 0xffffffff)
                nand->chip_shift = ffs((unsigned) nand->chipsize) - 1;
        else
                nand->chip_shift =
                                ffs((unsigned) (nand->chipsize >> 32)) + 32
 - 1;
/* Return success. */
```

```
                nand->chip_shift = ffs((unsigned) nand->chipsize) - 1;
        else
                nand->chip_shift =
                                ffs((unsigned) (nand->chipsize >> 32)) + 32
 - 1;
/* Return success. */

        return 0;

}
```

修改 mil_set_geometry()函数

```
static int mil_set_geometry(struct gpmi_nfc_data  *this)
{//add
        struct device        *dev      =  this->dev;
        struct nfc_geometry  *nfc_geo  = &this->nfc_geometry;
        struct mil           *mil      = &this->mil;

        /* Free the memory for read ID case */
        if (mil->page_buffer_virt && virt_addr_valid(mil->page_buffer_virt
    ))
                dma_free_coherent(dev, nfc_geo->payload_size_in_bytes,
                                mil->page_buffer_virt, mil->page_buffer_ph
    ys);

        /* Set up the various layers of geometry, in this specific order.
    */

        if (mil_set_physical_geometry(this))
                return -ENXIO;

        if (mil_set_nfc_geometry(this))
                return -ENXIO;
```

```
        if (mil_set_nfc_geometry(this))
                return -ENXIO;

        if (mil_set_boot_rom_helper_geometry(this))
                return -ENXIO;

        if (mil_set_mtd_geometry(this))
                return -ENXIO;

         mil->page_buffer_size = nfc_geo->payload_size_in_bytes +
                                        nfc_geo->auxiliary_size_in_bytes;

        mil->page_buffer_virt =
                dma_alloc_coherent(dev, mil->page_buffer_size,
                                        &mil->page_buffer_phys, GFP_DMA);

        if (!mil->page_buffer_virt)
                return -ENOMEM;

        /* Slice up the page buffer. */

        mil->payload_virt = mil->page_buffer_virt;
        mil->payload_phys = mil->page_buffer_phys;
```

```
        /* Slice up the page buffer. */

        mil->payload_virt = mil->page_buffer_virt;
        mil->payload_phys = mil->page_buffer_phys;

        mil->auxiliary_virt = ((char *) mil->payload_virt) +
                                nfc_geo->payload_size_in_bytes;
        mil->auxiliary_phys = mil->payload_phys +
                                nfc_geo->payload_size_in_bytes;

        /* Return success. */

        return 0;

}

#if 0
```

修改 mil_pre_bbt_scan()函数

```
static int mil_pre_bbt_scan(struct gpmi_nfc_data  *this)
{
        struct device             *dev      =  this->dev;
        struct physical_geometry  *physical = &this->physical_geometry;
        struct boot_rom_helper    *rom      =  this->rom;
        struct mil                *mil      = &this->mil;
        struct nand_chip          *nand     = &mil->nand;
        struct mtd_info           *mtd      = &mil->mtd;
        unsigned int              block_count;
        unsigned int              block;
        int                       chip;
        int                       page;
        loff_t                    byte;
        uint8_t                   block_mark;
        int                       error;
```

```
        if (rom->swap_block_mark)
                return 0;

        if (rom->check_transcription_stamp(this))
                return 0;

        /*
         * If control arrives here, we couldn't find a transcription stamp,
 so
         * so we presume the block marks are in the conventional location.
         */

        pr_info("Transcribing bad block marks...\n");

        /* Compute the number of blocks in the entire medium. */

        block_count =
                physical->chip_size_in_bytes >> nand->phys_erase_shift;

        /*
         * Loop over all the blocks in the medium, transcribing block marks
 as
         * we go.
```

```
        for (block = 0; block < block_count; block++) {
                /*
                 * Compute the chip, page and byte addresses for this block
 's
                 * conventional mark.
                 */

                chip = block >> (nand->chip_shift - nand->phys_erase_shift)
 ;
                page = block << (nand->phys_erase_shift - nand->page_shift)
 ;
                byte = block <<  nand->phys_erase_shift;

                /* Select the chip. */

                nand->select_chip(mtd, chip);

                /* Send the command to read the conventional block mark. */

                nand->cmdfunc(mtd, NAND_CMD_READ0,
                                physical->page_data_size_in_bytes, page);
```

```
                block_mark = nand->read_byte(mtd);
                 if (block_mark != 0xff) {
                        pr_info("Transcribing mark in block %u\n", block);
                        mil->marking_a_bad_block = true;
                        error = nand->block_markbad(mtd, byte);
                        mil->marking_a_bad_block = false;
                        if (error)
                                dev_err(dev, "Failed to mark block bad with
 "
                                                        "error %d\n", error
);
                }

                nand->select_chip(mtd, -1);

        }

        /* Write the stamp that indicates we've transcribed the block marks
. */
        rom->write_transcription_stamp(this);
        return 0;
}
```

修改 mil_boot_areas_init()函数

```
{//modify
        struct device                   *dev      =  this->dev;

        struct physical_geometry       *physical = &this->physical_geometry;

        struct boot_rom_geometry        *rom      = &this->rom_geometry;
        struct mil                      *mil      = &this->mil;
        struct mtd_info                 *mtd      = &mil->mtd;
        struct nand_chip                *nand     = &mil->nand;
        struct nand_device_info          *info    = &this->device_info;
        int                             mtd_support_is_adequate;
        unsigned int                    i;
        struct mtd_partition            partitions[7];//modify
        struct mtd_partition            *partitions_ptr;
        struct gpmi_nfc_platform_data *pdata = this->pdata;
        struct mtd_info                 *search_mtd;
        struct mtd_info                 *chip_0_remainder_mtd = 0;
        struct mtd_info                 *medium_remainder_mtd = 0;
        struct mtd_info                 *concatenate[2];

        /*
```

```
         */
         static char  *chip_0_boot_name      = "gpmi-nfc-0-boot";
         static char  *chip_0_remainder_name = "gpmi-nfc-0-remainder";
         static char  *chip_1_boot_name      = "gpmi-nfc-1-boot";
         static char  *medium_remainder_name = "gpmi-nfc-remainder";
//modify
         static char  *general_use_name      = "rootfs 64MB";

         static char *str_uboot = "reserve";
         static char *str_reserve= "reserve";
         static char *str_opt ="opt";

         /* Check if we're protecting the boot areas.*/
         if (!rom->boot_area_count) {
                 /*
                  * If control arrives here, we're not protecting the boot a
reas.
```

```
3                /* Chip 0 Boot */
4                partitions[0].name       = "boot 12MB";
5                partitions[0].offset     = 0;
6                partitions[0].size       = 12*1024*1024;
7                partitions[0].mask_flags = 0;
8
9
0                partitions[1].name       = "env 512k";
1                partitions[1].offset     = MTDPART_OFS_APPEND;
2                partitions[1].size       = 512*1024;
3                partitions[1].mask_flags = 0;
4
5                partitions[2].name       = "reserved 512KB";
6                partitions[2].offset     = MTDPART_OFS_APPEND;
7                partitions[2].size       = 512*1024;
8                partitions[2].mask_flags = 0;
9
0                partitions[3].name       = "env2 2MB";
1                partitions[3].offset     = MTDPART_OFS_APPEND;
2                partitions[3].size       = 2*1024*1024;
3                partitions[3].mask_flags = 0;
4
```

```
            partitions[3].name       = "env2 2MB";
            partitions[3].offset     = MTDPART_OFS_APPEND;
            partitions[3].size       = 2*1024*1024;
            partitions[3].mask_flags = 0;


            partitions[4].name       = "reserved 512KB";
            partitions[4].offset     = MTDPART_OFS_APPEND;
            partitions[4].size       = 512*1024;
            partitions[4].mask_flags = 0;

            partitions[5].name       = general_use_name;
            partitions[5].offset     = MTDPART_OFS_APPEND;
            partitions[5].size       = 64*1024*1024;
            partitions[5].mask_flags = 0;

            partitions[6].name       = "opt      -";
            partitions[6].offset     = MTDPART_OFS_APPEND;
            partitions[6].size       = MTDPART_SIZ_FULL;
            partitions[6].mask_flags = 0;

            /* Construct and register the partitions. */
```

修改 gpmi_nfc_mil_init()函数

```
int gpmi_nfc_mil_init(struct gpmi_nfc_data *this)
{
        struct device                   *dev    =  this->dev;
        struct gpmi_nfc_platform_data   *pdata  =  this->pdata;
        struct mil                      *mil    = &this->mil;
        struct mtd_info                 *mtd    = &mil->mtd;
        struct nand_chip                *nand   = &mil->nand;
//add
        static struct nand_ecclayout     fake_ecc_layout;
        int                             error = 0;

        /* Initialize MIL data. */
```

```
 *      - Disallow partial page writes.
 */
nand->options |= NAND_NO_SUBPAGE_WRITE;
//add
nand->options |= NAND_USE_FLASH_BBT;
nand->bbt_td   = &bbt_main_descriptor;
nand->bbt_md   = &bbt_mirror_descriptor;
nand->badblock_pattern = &gpmi_bbt_descr;

/*
 * Tell the NAND Flash MTD system that we'll be handling ECC wi
ur
```

```
        /* Allocate a temporary DMA buffer for reading ID in the nand_scan(
) */

        memset(&fake_ecc_layout, 0,sizeof(fake_ecc_layout));
        nand->ecc.layout = &fake_ecc_layout;

        /* Allocate a command buffer. */

        mil->cmd_virt =
                dma_alloc_coherent(dev,
                        MIL_COMMAND_BUFFER_SIZE, &mil->cmd_phys, GFP_DMA);

        if (!mil->cmd_virt) {
                error = -ENOMEM;
                goto exit_cmd_allocation;
        }
```

```
        /* Allocate buf  read ID case */
        this->nfc_geometry.payload_size_in_bytes = 1024;
        mil->page_buffer_virt =
                dma_alloc_coherent(dev,
                                this->nfc_geometry.payload_size_in_bytes,
                                &mil->page_buffer_phys, GFP_DMA);
         if (!mil->page_buffer_virt) {
                error = -ENOMEM;
                goto exit_buf_allocation;
        }

        /* Slice up the page buffer. */
        mil->payload_virt = mil->page_buffer_virt;
        mil->payload_phys = mil->page_buffer_phys;

        pr_info("Scanning for NAND Flash chips...\n");

        error = nand_scan(mtd, pdata->max_chip_count);

        if (error) {
                dev_err(dev, "Chip scan failed\n");
                goto exit_nand_scan;
        }
```

修改 gpmi_nfc_mil_exit()函数

```
 */
void gpmi_nfc_mil_exit(struct gpmi_nfc_data *this)
{
        struct mil      *mil = &this->mil;
        struct device   *dev = this->dev;
        /* Shut down partitions as necessary. */
        mil_partitions_exit(this);

        /* Get MTD to let go of our MTD. */
        nand_release(&mil->mtd);

        /* Free all the DMA buffer, if it's been allocated. */
        //mil_free_dma_buffer(this);

         /* Free the page buffer, if it's been allocated. */

        if (mil->page_buffer_virt)
                dma_free_coherent(dev, mil->page_buffer_size,
                                  mil->page_buffer_virt, mil->page_buffer_phy
s);

        mil->page_buffer_size =  0;
        mil->page_buffer_virt =  0;
```

```
        if (mil->page_buffer_virt)
                dma_free_coherent(dev, mil->page_buffer_size,
                                  mil->page_buffer_virt, mil->page_buffer_phy
s);

        mil->page_buffer_size =  0;
        mil->page_buffer_virt =  0;
        mil->page_buffer_phys = ~0;

        /* Free the command buffer, if it's been allocated. */

        if (mil->cmd_virt)
                dma_free_coherent(dev, MIL_COMMAND_BUFFER_SIZE,
                                                mil->cmd_virt, mil->cmd_phy
s);

        mil->cmd_virt =  0;
        mil->cmd_phys = ~0;

}
```

14、在 drivers/mtd/nand/gpmi-nfc/gpmi-nfc-rom-v0.c 修改

```
/* This structure represents the Boot ROM Helper for this version. */
struct boot_rom_helper  gpmi_nfc_boot_rom_helper_v0 = {
        .version                    = 0,
        .description                = "Single/dual-chip boot area, "
                                          "no block mark swapping",
        .swap_block_mark            = false,
        .set_geometry               = set_geometry,
        .check_transcription_stamp = check_transcription_stamp,
        .write_transcription_stamp = write_transcription_stamp,

};
```

15、在顶层目录下 make imx28evk_defconfig

```
djh@ubuntu:~/linux/imx-linux/35.main$ sudo make imx28evk_defconfig
#
# configuration written to .config
#
djh@ubuntu:~/linux/imx-linux/35.main$
```

16、在顶层目录下 make uImage ,如果出现无法生成 uImage 的错误，原因是缺少 mkimage。

```
djh@ubuntu:~/linux/imx-linux/35.main$ sudo make uImage -j4
scripts/kconfig/conf -s arch/arm/Kconfig
  CHK     include/linux/version.h
  CHK     include/generated/utsrelease.h
make[1]: 'include/generated/mach-types.h' is up to date.
  CALL    scripts/checksyscalls.sh
  CHK     include/generated/compile.h
```

解决方法：将 uboot/tools/mkimage 拷贝到/usr/bin/下

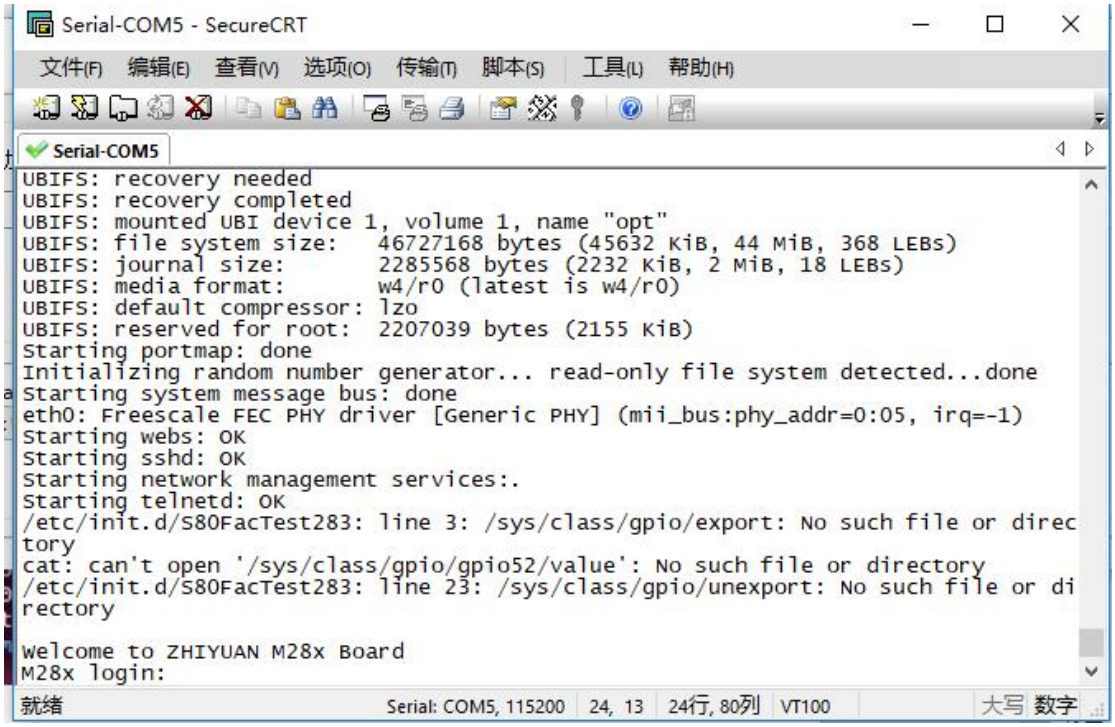
17、make uImage -j4 的命令是使用 4 个处理器同时编译，速度快，缺点是有错误不会停止编译，无法直观看到错误信息。

18、编译完成，信息如下

```
  AS      .tmp_kallsyms3.o
  LD      vmlinux
  SYSMAP  System.map
  SYSMAP  .tmp_System.map
  OBJCOPY arch/arm/boot/Image
  Kernel: arch/arm/boot/Image is ready
  GZIP    arch/arm/boot/compressed/piggy.gzip
  SHIPPED arch/arm/boot/compressed/lib1funcs.S
  AS      arch/arm/boot/compressed/lib1funcs.o
  AS      arch/arm/boot/compressed/piggy.gzip.o
  LD      arch/arm/boot/compressed/vmlinux
  OBJCOPY arch/arm/boot/zImage
  Kernel: arch/arm/boot/zImage is ready
  UIMAGE  arch/arm/boot/uImage
Image Name:   Linux-2.6.35.3
Created:      Wed Sep 19 20:08:45 2018
Image Type:   ARM Linux Kernel Image (uncompressed)
Data Size:    2639012 Bytes = 2577.16 kB = 2.52 MB
Load Address: 40008000
Entry Point:  40008000
  Image arch/arm/boot/uImage is ready
djh@ubuntu:~/linux/imx-linux/35.main$
```

19、使用 freescale 自带的下载工具烧写，具体方法如《EasyARM-iMX28xx Linux 开发指南 20150901 V1.03》

20、下载烧录完成，取下 JP6 跳冒，按下复位键，打印结果如下



20、linux 下当打开文件出现 Read-only file system 时，执行以下命令

mount -o remount rw /