实验六 u-boot-2013.01 移植

【实验目的】

了解 u-boot 的代码结构及移植的基本方法

【实验环境】

1、 ubuntu 14.04 发行版

2、 FS4412 实验平台

3、 交叉编译工具 arm-none-linux-gnueabi-

【注意事项】

1、实验步骤中以“$”开头的命令表示在 ubuntu 环境下执行

【实验步骤】

一、建立自己的平台

1、下载 uboot 源码

在 uboot 官网下载 uboot 源码（这里我们选择 u-boot-2013.01.tar.bz2）

ftp://ftp.denx.de/pub/u-boot/

2、解压 uboot 源码

拷贝 uboot 源码包到 ubuntu 的家目录下，解压并进入其顶层目录

```
$ tar   xvf   u-boot-2013.01.tar.bz2

$ cd   u-boot-2013.01/
```

3、指定交叉编译工具信息

uboot 源码并不知道我们使用的处理器架构及交叉编译工具是什么，这里我们需要自己在 Makefile 中指定

```
$ vi   Makefile
```

将

```
ifeq ($(HOSTARCH),$(ARCH))

CROSS_COMPILE ?=

endif
```

修改为如下内容（注意后边不要有多余的空格），然后保存退出

```
ifeq (arm,arm)

CROSS_COMPILE ?= arm-none-linux-gnueabi-

endif
```

4、添加 Board 信息

因为 uboot 源码并不支持我们的开发板，这里我们需要从源码支持的开发板中找一个硬件与我们最类似的，在其基础上进行修改，这里我们参考的是 samsung 公司的 origen

```
$ cp    -rf    board/samsung/origen/    board/samsung/fs4412

$ mv    board/samsung/fs4412/origen.c    board/samsung/fs4412/fs4412.c
```

因为修改了文件名，所以对应的 Makefile 也要修改

```
$ vi    board/samsung/fs4412/Makefile
```

将

```
ifndef CONFIG_SPL_BUILD

COBJS    += origen.o

endif
```

修改为如下内容，然后保存退出

```
ifndef CONFIG_SPL_BUILD

COBJS    += fs4412.o

endif
```

拷贝 origen 相关的头文件并将其重命名

```
$ cp    include/configs/origen.h    include/configs/fs4412.h
```

修改文件中的信息

```
$ vi    include/configs/fs4412.h
```

将

```
#define CONFIG_SYS_PROMPT        "ORIGEN # "
```

修改为如下内容

```
#define CONFIG_SYS_PROMPT        "fs4412 # "
```

再将

```
#define CONFIG_IDENT_STRING		" for ORIGEN"
```

修改为如下内容，然后保存退出

```
#define CONFIG_IDENT_STRING		" for fs4412"
```

打开 uboot 源码顶层目录下的 boards.cfg 文件

```
$ vi    boards.cfg
```

在

```
origen                  arm     armv7     origen      samsung     exynos
```

后添加如下内容（FS4412 的相关信息），然后保存退出

```
fs4412                  arm     armv7     fs4412      samsung     exynos
```

至此我们在 uboot 源码中给我们的板子添加了“档案”，源码就支持我们的开发板了

5、编译 uboot

在 uboot 源码顶层目录下执行如下命令，指定当前使用的 Board 信息

```
$ make    fs4412_config
```

编译 uboot

```
$ make
```

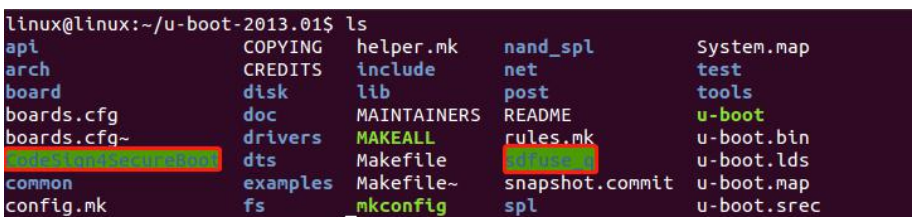
编译完成后会在源码顶层目录下生成 u-boot.bin 文件，但该文件还不能在我们的开发板上运行，因为以上操作我们只是把 origen 相关的文件的名字改成了 fs4412，使 uboot 能识别 fs4412 开发板，但文件中的代码还是 origen 的，和我们的开发板不匹配，所以我们还需要进一步进行修改和配置

二、添加三星加密引导方式

考虑芯片启动的安全性，Exynos4412 需要三星提供的初始引导加密后我们的 u-boot 才能被引导运行，所以我们需要在 uboot 源码中添加三星提供的加密处理代码

1、添加三星加密引导方式

将资料中“移植相关文件”下的 sdfuse_q 和 CodeSign4SecureBoot 目录拷贝到 uboot 源码的顶层目录下（这之后不要执行 make clean 或 make distclean，这会将加密文件清除）



因为添加的加密文件也要编译，所以对应的 Makefile 也要修改

```
$ vi    Makefile
```

在

```
$(obj)u-boot.bin:      $(obj)u-boot
        $(OBJCOPY) ${OBJCFLAGS} -O binary $< $@
        $(BOARD_SIZE_CHECK)
```

后添加如下内容（添加的内容需要 tab 键缩进，否则编译报错），然后保存退出

```
        @#./mkuboot
        @split -b 14336 u-boot.bin bl2
        @+make -C sdfuse_q/
        @#cp u-boot.bin u-boot-4212.bin
        @#cp u-boot.bin u-boot-4412.bin
        @#./sdfuse_q/add_sign
        @./sdfuse_q/chksum
        @./sdfuse_q/add_padding
        @rm bl2a*
        @echo
```

2、添加调试代码（点灯法）

很多时候我们不确定 uboot 是否已经在板子上运行，所以我们在 uboot 源码中添加一段代码使板子上的 LED 点亮，这样如果看到 LED 亮的话就表示 uboot 已经在运行了

打开 uboot 启动后的第一段代码

```
$ vi    arch/arm/cpu/armv7/start.S
```

在第 134 行后添加如下代码（即点亮 LED2），然后保存退出

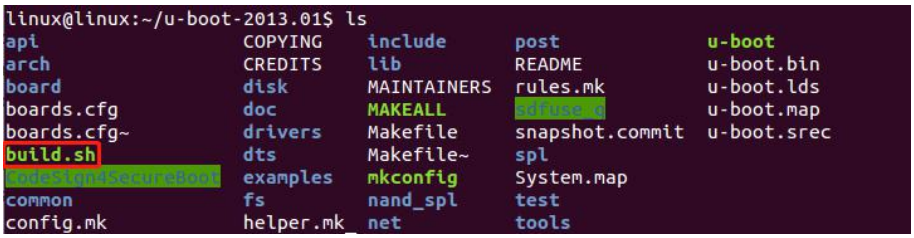
```
ldr r0, =0x11000c40
ldr r1, [r0]
bic r1, r1, #0xf0000000
orr r1, r1, #0x10000000
str r1, [r0]
```

```
ldr r0, =0x11000c44

mov r1, #0xff

str r1, [r0]
```

3、添加编译脚本

使用 make 命令编译时只链接 uboot 源码中的相关代码，而我们添加的初始引导加密的代码不会被连接到 u-boot.bin 中，所以这里我们自己编写编译脚本 build.sh，这个脚本中除了对 uboot 源码进行配置和编译外还将初始引导加密代码链接到了 u-boot.bin 上，最终生成一个完成的 uboot 镜像 u-boot-fs4412.bin

将资料中“移植相关文件”下的 build.sh 拷贝到 uboot 源码的顶层目录下



给编译脚本添加可执行权限

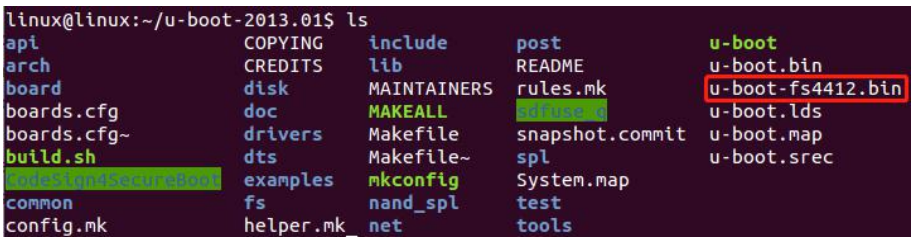
```
$ chmod  777  build.sh
```

4、编译 uboot

通过脚本编译 uboot 源码

```
$ ./build.sh
```

编译完成后在源码的顶层目录下会生成“u-boot-fs4412.bin”



5、测试 uboot

参照之前的实验将生成的 u-boot-fs4412.bin 烧写到 SD 卡中，开发板选择 SD 卡启动，然后上电查看现象，若 LED2 点亮则说明我自己移植的 u-boot 已经能够被加载运行

三、实现串口输出

虽然 uboot 已经能在开发板上加载运行，但是此时的 uboot 还不能在终端上打印信息，原因在于 uboot 源码中对 UART 的配置与我们实际的硬件不匹配

1、修改 UART 源码

```
$ vi    board/samsung/fs4412/lowlevel_init.S
```

在

```
lowlevel_init:
```

后添加如下内容（初始化临时栈）

```
ldr    sp,=0x02060000
```

在

```
beq    wakeup_reset
```

后添加如下内容（关闭看门狗）

```
#if 1
 ldr r0, =0x1002330c
 ldr r1, [r0]
 orr r1, r1, #0x300
 str r1, [r0]
 ldr r0, =0x11000c08
 ldr r1, =0x0
 str r1, [r0]
/* Clear    MASK_WDT_RESET_REQUEST    */
 ldr r0, =0x1002040c
 ldr r1, =0x00
 str r1, [r0]
#endif
```

在

```
uart_asm_init:
 /* setup UART0-UART3 GPIOs (part1) */
 mov r0, r7
 ldr    r1, =EXYNOS4_GPIO_A0_CON_VAL
```

```
str    r1, [r0, #EXYNOS4_GPIO_A0_CON_OFFSET]

ldr    r1, =EXYNOS4_GPIO_A1_CON_VAL

str    r1, [r0, #EXYNOS4_GPIO_A1_CON_OFFSET]
```

后添加如下内容（UART 初始化）

```
ldr    r0, =0x10030000

ldr    r1, =0x666666

ldr    r2, =CLK_SRC_PERIL0_OFFSET

str    r1, [r0, r2]

ldr    r1, =0x777777

ldr    r2, =CLK_DIV_PERIL0_OFFSET

str    r1, [r0, r2]
```

注释掉

```
bl uart_asm_init
```

后的一条语句，然后保存退出

```
#if 0

bl tzpc_init

#endif
```

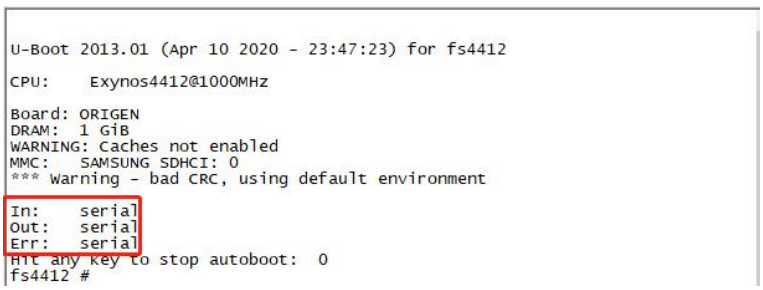
2、编译 uboot

通过脚本编译 uboot 源码

```
$ ./build.sh
```

3、测试 uboot

参照之前的实验将生成的 u-boot-fs4412.bin 烧写到 SD 卡中，开发板选择 SD 卡启动，然后上电查看现象，若终端有打印信息则说明 UART 移植成功



四、网卡移植

虽然可以通过终端输入命令，但此时的 uboot 还不能使用 ping、tftp 等命令，原因在于命令都是操作网络的，而 uboot 源码中网卡的相关配置与我们当前的板子不匹配，所以我们还要对网卡进行移植

1、修改网络初始化代码

```
$ vi    board/samsung/fs4412/fs4412.c
```

在

```
struct exynos4_gpio_part2 *gpio2;
```

后添加如下内容

```
#ifdef    CONFIG_DRIVER_DM9000

#define EXYNOS4412_SROMC_BASE 0X12570000


#define DM9000_Tacs         (0x1)

#define DM9000_Tcos         (0x1)

#define DM9000_Tacc         (0x5)

#define DM9000_Tcoh         (0x1)

#define DM9000_Tah          (0xC)

#define DM9000_Tacp         (0x9)

#define DM9000_PMC          (0x1)


struct exynos_sromc {

 unsigned int bw;

 unsigned int bc[6];

};


void exynos_config_sromc(u32 srom_bank, u32 srom_bw_conf, u32 srom_bc_conf)

{

 unsigned int tmp;
```

```
 struct exynos_sromc *srom = (struct exynos_sromc *)(EXYNOS4412_SROMC_BASE);


 /* Configure SMC_BW register to handle proper SROMC bank */
 tmp = srom->bw;
 tmp &= ~(0xF << (srom_bank * 4));
 tmp |= srom_bw_conf;
 srom->bw = tmp;


 /* Configure SMC_BC register */
 srom->bc[srom_bank] = srom_bc_conf;
}


static void dm9000aep_pre_init(void)
{
 unsigned int tmp;
 unsigned char smc_bank_num = 1;
 unsigned int          smc_bw_conf=0;
 unsigned int          smc_bc_conf=0;


 /* gpio configuration */
 writel(0x00220020, 0x11000000 + 0x120);
 writel(0x00002222, 0x11000000 + 0x140);
 /* 16 Bit bus width */
 writel(0x22222222, 0x11000000 + 0x180);
 writel(0x0000FFFF, 0x11000000 + 0x188);
 writel(0x22222222, 0x11000000 + 0x1C0);
 writel(0x0000FFFF, 0x11000000 + 0x1C8);
```

```
 writel(0x22222222, 0x11000000 + 0x1E0);
 writel(0x0000FFFF, 0x11000000 + 0x1E8);
 smc_bw_conf &= ~(0xf<<4);
 smc_bw_conf |= (1<<7) | (1<<6) | (1<<5) | (1<<4);
 smc_bc_conf = ((DM9000_Tacs << 28)
                | (DM9000_Tcos << 24)
                | (DM9000_Tacc << 16)
                | (DM9000_Tcoh << 12)
                | (DM9000_Tah  << 8)
                | (DM9000_Tacp << 4)
                | (DM9000_PMC));
 exynos_config_sromc(smc_bank_num,smc_bw_conf,smc_bc_conf);
}
#endif
```

在

```
gd->bd->bi_boot_params = (PHYS_SDRAM_1 + 0x100UL);
```

后添加如下内容

```
#ifdef CONFIG_DRIVER_DM9000
 dm9000aep_pre_init();
#endif
```

在文件末尾添加如下内容，然后保存退出

```
#ifdef CONFIG_CMD_NET
int board_eth_init(bd_t *bis)
{
 int rc = 0;
#ifdef CONFIG_DRIVER_DM9000
 rc = dm9000_initialize(bis);
```

```
#endif
 return rc;
}
#endif
```

2、修改网络配置代码

```
$ vi    include/configs/fs4412.h
```

将

```
#undef CONFIG_CMD_PING
```

修改为

```
#define CONFIG_CMD_PING
```

再将

```
#undef CONFIG_CMD_NET
```

修改为

```
#define CONFIG_CMD_NET
```

在文件末尾

```
#endif      /* __CONFIG_H */
```

前添加如下内容，然后保存退出

```
#ifdef    CONFIG_CMD_NET
#define CONFIG_NET_MULTI
#define CONFIG_DRIVER_DM9000    1
#define CONFIG_DM9000_BASE 0x05000000
#define DM9000_IO       CONFIG_DM9000_BASE
#define DM9000_DATA (CONFIG_DM9000_BASE + 4)
#define CONFIG_DM9000_USE_16BIT
#define CONFIG_DM9000_NO_SROM       1
#define CONFIG_ETHADDR 11:22:33:44:55:66
#define CONFIG_IPADDR     192.168.9.200
```

```
#define CONFIG_SERVERIP 192.168.9.120

#define CONFIG_GATEWAYIP     192.168.9.1

#define CONFIG_NETMASK       255.255.255.0

#endif
```

3、编译 uboot

通过脚本编译 uboot 源码

```
$ ./build.sh
```

4、测试 uboot

参照之前的实验将生成的 u-boot-fs4412.bin 烧写到 SD 卡中，开发板选择 SD 卡启动，然后上电查看现象；设置好相关的环境变量，使用网线连接开发板与开发主机，使用 ping 命令连接 ubuntu，若显示“host xxx.xxx.xxx.xxx is alive”则表示网卡移植成功

```
U-Boot 2013.01 (Apr 11 2020 - 00:37:57) for fs4412

CPU:    Exynos4412@1000MHz

Board: ORIGEN
DRAM:  1 GiB
WARNING: Caches not enabled
MMC:   SAMSUNG SDHCI: 0
*** Warning - bad CRC, using default environment

In:    serial
Out:   serial
Err:   serial
Net:   dm9000
Hit any key to stop autoboot:  0
fs4412 # setenv ipaddr 192.168.0.10
fs4412 # ping 192.168.0.100
dm9000 i/o: 0x5000000, id: 0x90000a46
DM9000: running in 16 bit mode
MAC: 11:22:33:44:55:66
operating at 100M full duplex mode
Using dm9000 device
host 192.168.0.100 is alive
fs4412 #
```

五、EMMC 移植

因为 uboot 源码中对 EMMC 的配置与我们的板子不匹配，这里还需要对 EMMC 相关的代码进行修改和配置

1、修改 EMMC 初始化代码

将资料中“移植相关文件”下的 movi.c 拷贝到 uboot 源码的 arch/arm/cpu/armv7/exynos/ 目录下

```
linux@linux:~/u-boot-2013.01/arch/arm/cpu/armv7/exynos$ ls
asm-offsets.s  clock.su     movi.c     pinmux.su  power.su  soc.su    system.su
clock.c        libexynos.o  pinmux.c   power.c    soc.c     system.c
clock.o        Makefile     pinmux.o   power.o    soc.o     system.o
```

因为添加的新文件也要编译，所以对应的 Makefile 也要修改

```
$ vi    arch/arm/cpu/armv7/exynos/Makefile
```

将

```
COBJS	+= clock.o power.o soc.o system.o pinmux.o
```

修改为如下内容，然后保存退出

```
COBJS	+= clock.o power.o soc.o system.o pinmux.o movi.o
```

修改板级文件

```
$ vi	board/samsung/fs4412/fs4412.c
```

在

```
#include <asm/arch/mmc.h>
```

后添加如下内容

```
#include <asm/arch/clk.h>

#include "origen_setup.h"
```

在

```
#ifdef CONFIG_GENERIC_MMC
```

后添加如下内容

```
u32 sclk_mmc4;	/*clock source for emmc controller*/

#define __REGMY(x) (*((volatile u32 *)(x)))

#define CLK_SRC_FSYS	__REGMY(EXYNOS4_CLOCK_BASE +
CLK_SRC_FSYS_OFFSET)

#define CLK_DIV_FSYS3 __REGMY(EXYNOS4_CLOCK_BASE +
CLK_DIV_FSYS3_OFFSET)



int emmc_init()

{

 u32 tmp;

 u32 clock;

 u32 i;

 /* setup_hsmmc_clock */

 /* MMC4 clock src = SCLKMPLL */
```

```
tmp = CLK_SRC_FSYS & ~(0x000f0000);
CLK_SRC_FSYS = tmp | 0x00060000;
/* MMC4 clock div */
tmp = CLK_DIV_FSYS3 & ~(0x0000ff0f);
clock = get_pll_clk(MPLL)/1000000;


for(i=0 ; i<=0xf; i++)    {
        sclk_mmc4=(clock/(i+1));


        if(sclk_mmc4 <= 160) //200
        {
                CLK_DIV_FSYS3 = tmp | (i<<0);
                break;
        }
}
emmcdbg("[mjdbg] sclk_mmc4:%d MHZ; mmc_ratio: %d\n",sclk_mmc4,i);
sclk_mmc4 *= 1000000;


/*
* MMC4 EMMC GPIO CONFIG
*
* GPK0[0]      SD_4_CLK
* GPK0[1]      SD_4_CMD
* GPK0[2]      SD_4_CDn
* GPK0[3:6]   SD_4_DATA[0:3]
*/
writel(readl(0x11000048)&~(0xf),0x11000048); //SD_4_CLK/SD_4_CMD pull-down
enable
```

```
 writel(readl(0x11000040)&~(0xff),0x11000040);//cdn set to be output


 writel(readl(0x11000048)&~(3<<4),0x11000048); //cdn pull-down disable

 writel(readl(0x11000044)&~(1<<2),0x11000044); //cdn output 0 to shutdown the emmc
power

 writel(readl(0x11000040)&~(0xf<<8)|(1<<8),0x11000040);//cdn set to be output

 udelay(100*1000);

 writel(readl(0x11000044)|(1<<2),0x11000044); //cdn output 1


 writel(0x03333133, 0x11000040);


 writel(0x00003FF0, 0x11000048);

 writel(0x00002AAA, 0x1100004C);


#ifdef CONFIG_EMMC_8Bit

 writel(0x04444000, 0x11000060);

 writel(0x00003FC0, 0x11000068);

 writel(0x00002AAA, 0x1100006C);

#endif


#ifdef USE_MMC4

 smdk_s5p_mshc_init();

#endif

}
```

将 board_mmc_init 函数中的内容修改为（之前的内容删除即可）如下内容

```
int board_mmc_init(bd_t *bis)

{

 int i, err;
```

```
#ifdef CONFIG_EMMC
 err = emmc_init();
#endif
 return err;
}
```

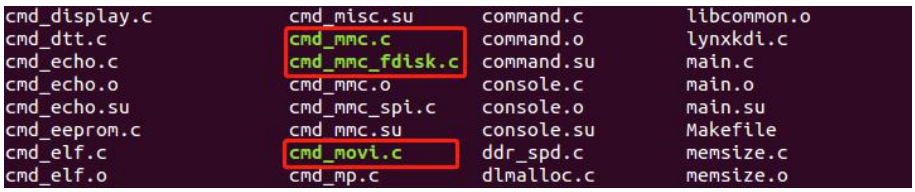
在文件的最末尾添加如下内容，然后保存退出

```
#ifdef CONFIG_BOARD_LATE_INIT
#include <movi.h>
int    chk_bootdev(void)//mj for boot device check
{
 char run_cmd[100];
 struct mmc *mmc;
 int boot_dev = 0;
 int cmp_off = 0x10;
 ulong    start_blk, blkcnt;


 mmc = find_mmc_device(0);


 if (mmc == NULL)
 {
        printf("There is no eMMC card, Booting device is SD card\n");
        boot_dev = 1;
        return boot_dev;
        }
 start_blk = (24*1024/MOVI_BLKSIZE);
 blkcnt = 0x10;
```

```
sprintf(run_cmd,"emmc open 0");
run_command(run_cmd, 0);

sprintf(run_cmd,"mmc read 0 %lx %lx %lx",CFG_PHY_KERNEL_BASE,start_blk,blkcnt);
run_command(run_cmd, 0);

/* switch mmc to normal paritition */
sprintf(run_cmd,"emmc close 0");
run_command(run_cmd, 0);

return 0;
}

int board_late_init (void)
{
int boot_dev =0 ;
char boot_cmd[100];
boot_dev = chk_bootdev();
if(!boot_dev)
{
        printf("\n\nChecking Boot Mode ... EMMC4.41\n");
}
return 0;
}
#endif
```

2、添加 EMMC 命令

将资料中“移植相关文件”下的 cmd_movi.c、cmd_mmc.c、cmd_mmc_fdisk.c 拷贝到 uboot 源码的 common/目录下



因为添加的新文件也要编译，所以对应的 Makefile 也要修改

```
$ vi    common/Makefile
```

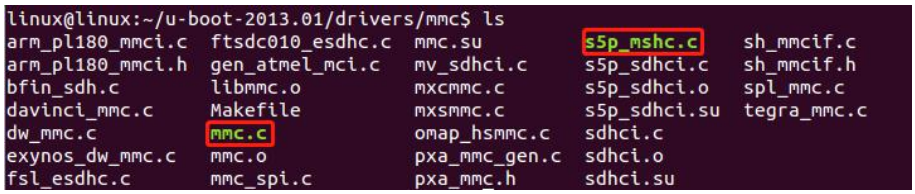
在

```
COBJS-$(CONFIG_CMD_MMC) += cmd_mmc.o
```

后添加如下内容，然后保存退出

```
COBJS-$(CONFIG_CMD_MMC) += cmd_mmc_fdisk.o

COBJS-$(CONFIG_CMD_MOVINAND) += cmd_movi.o
```

将资料中“移植相关文件”下的 mmc.c、s5p_mshc.c 拷贝到 uboot 源码的 drivers/mmc/目录下



将资料中“移植相关文件”下的 mmc.h、movi.h、s5p_mshc.h 拷贝到 uboot 源码的 include/目录下

因为添加的新文件也要编译，所以对应的 Makefile 也要修改

```
$ vi    drivers/mmc/Makefile
```

在

```
COBJS-$(CONFIG_S5P_SDHCI) += s5p_sdhci.o
```

后添加如下内容，然后保存退出

```
COBJS-$(CONFIG_S5P_MSHC) += s5p_mshc.o
```

3、修改 EMMC 配置代码

```
$ vi    include/configs/fs4412.h
```

在文件的末尾

```
#endif    /* __CONFIG_H */
```

前添加如下内容，然后保存退出

```
#define CONFIG_EVT1         1             /* EVT1 */

#ifdef CONFIG_EVT1

#define CONFIG_EMMC44_CH4 //eMMC44_CH4 (OMPIN[5:1] = 4)



#ifdef CONFIG_SDMMC_CH2

#define CONFIG_S3C_HSMMC

#undef DEBUG_S3C_HSMMC

#define USE_MMC2

#endif



#ifdef CONFIG_EMMC44_CH4

#define CONFIG_S5P_MSHC

#define CONFIG_EMMC                    1

#define USE_MMC4

/* #define CONFIG_EMMC_8Bit */

#define CONFIG_EMMC_EMERGENCY

/*#define emmcdbg(fmt,args...) printf(fmt ,##args) */

#define emmcdbg(fmt,args...)

#endif



#endif /*end CONFIG_EVT1*/

#define CONFIG_CMD_MOVINAND
```

```
#define CONFIG_CLK_1000_400_200

#define CFG_PHY_UBOOT_BASE          CONFIG_SYS_SDRAM_BASE + 0x3e00000

#define CFG_PHY_KERNEL_BASE         CONFIG_SYS_SDRAM_BASE + 0x8000


#define BOOT_MMCSD          0x3

#define BOOT_EMMC43         0x6

#define BOOT_EMMC441        0x7

#define CONFIG_BOARD_LATE_INIT
```

4、编译 uboot

通过脚本编译 uboot 源码

```
$ ./build.sh
```

5、测试 uboot

参照之前的实验将生成的 u-boot-fs4412.bin 烧写到 SD 卡中，开发板选择 SD 卡启动，然后上电查看现象；若显示 EMMC 的相关信息则表示 EMMC 移植成功

```
U-Boot 2013.01 (Apr 11 2020 - 01:50:17) for fs4412

CPU:    Exynos4412@1000MHz

Board: ORIGEN
DRAM:  1 GiB
WARNING: Caches not enabled
MMC:   MMC0:    3728 MB
In:    serial
Out:   serial
Err:   serial

MMC read: dev # 0, block # 48, count 16 ...16 blocks read: OK
eMMC CLOSE Success.!!


Checking Boot Mode ... EMMC4.41
Net:   dm9000
Hit any key to stop autoboot:  0
fs4412 #
```

六、电源管理移植

因为 uboot 源码中对电源管理芯片的配置与我们的板子不匹配，后续有可能会导致内核启动卡死，这里还需要对电源管理芯片相关的代码进行修改和配置

1、修改电源管理相关代码

将资料中“移植相关文件”下的 pmic_s5m8767.c 拷贝到 uboot 源码的 drivers/power/pmic/ 目录下

```
linux@linux:~/u-boot-2013.01/drivers/power/pmic$ ls
libpmic.o  muic_max8997.c   pmic_max8997.c  pmic_s5m8767.c
Makefile   pmic_max77686.c  pmic_max8998.c
```

因为添加的新文件也要编译，所以对应的 Makefile 也要修改

```
$ vi    drivers/power/pmic/Makefile
```

在

```
COBJS-$(CONFIG_POWER_MAX77686) += pmic_max77686.o
```

后添加如下内容，然后保存退出

```
COBJS-$(CONFIG_POWER_S5M8767) += pmic_s5m8767.o
```

将添加的函数在头文件中声明

```
$ vi    include/power/pmic.h
```

在

```
int pmic_set_output(struct pmic *p, u32 reg, int ldo, int on);
```

后添加如下内容，然后保存退出

```
void pmic_s5m8767_init(void);
```

修改配置文件

```
$ vi    include/configs/fs4412.h
```

在文件末尾

```
#endif      /* __CONFIG_H */
```

前添加如下内容，然后保存退出

```
#define CONFIG_POWER_S5M8767
```

修改板级文件

```
$ vi    board/samsung/fs4412/fs4412.c
```

在 board_init 函数中

```
#ifdef CONFIG_DRIVER_DM9000
 dm9000aep_pre_init();
#endif
```

后添加如下内容，然后保存退出

```
#ifdef CONFIG_POWER_S5M8767
 pmic_s5m8767_init();
```

```
#endif
```

注释原有的代码

```
$ vi    drivers/power/Makefile
```

将

```
COBJS-$(CONFIG_POWER) += power_core.o
```

修改为（即注释掉）

```
#COBJS-$(CONFIG_POWER) += power_core.o
```

修改架构文件

```
$ vi    arch/arm/cpu/armv7/s5p-common/cpu_info.c
```

在

```
#include <asm/arch/clk.h>
```

后添加如下内容，然后保存退出

```
#include <power/pmic.h>
```

2、编译 uboot

通过脚本编译 uboot 源码

```
$ ./build.sh
```

3、测试 uboot

参照之前的实验将生成的 u-boot-fs4412.bin 烧写到 SD 卡中，开发板选择 SD 卡启动，然后上电查看现象

```
U-Boot 2013.01 (Apr 11 2020 - 03:06:51) for fs4412

CPU:    Exynos4412@1000MHz

Board: ORIGEN
DRAM:  1 GiB
WARNING: Caches not enabled
PMIC: S5M8767(VER5.0)
MMC:   MMC0:    3728 MB
In:    serial
Out:   serial
Err:   serial

MMC read: dev # 0, block # 48, count 16 ...16 blocks read: OK
eMMC CLOSE Success.!!


Checking Boot Mode ... EMMC4.41
Net:   dm9000
Hit any key to stop autoboot:  0
fs4412 #
```

至此，uboot 移植完成