s5pv210 uboot-2012-10 移植(一) 之分析 Alex Ling 的 linaro-2011.10 for mini210

好久好久前就买了 s5pv210 的开发板，一直都是东搞搞西搞搞，一点收获也没有，这次下决心来移植最新的 uboot 到 u-boot-2012.10 上，并通过这个博客记录下来以防时间长给忘了，我的开发板是 QT210 的。s5pv210 的启动分为 BL0，BL1，BL2，BL0 是出厂的时候就固化在 IROM 里的，所以我们的 uboot 就要实现 BL1 和 BL2，BL1 在 uboot 里叫做 u-boot-spl.bin，BL2 就是我们很熟悉的 u-boot.bin 了。在移植之前，我们先看下 Alex Ling 的 linaro-2011.10 for mini210 的 UBOOT 是怎么实现的。这里主要还是分析 SPL 部分，u-boot.bin 是如何生成的现在资料很多，也很复杂，我这个菜鸟也是一知半解的，所以就不分析了。

1.顶层的 Makefile，从中可以知道，我们要想生成 u-boot-spl.bin 就必须配置 COFNIG_SPL，那么 u-boot-spl.bin 依赖什么呢，我们继续

```
1. ALL-$(CONFIG_SPL) += $(obj)spl/u-boot-spl.bin
2.
3. all:		$(ALL-y)
```

搜索发现，是进入到 uboot 顶层目录的 spl 目录下执行 Makefile 的

```
1. $(obj)spl/u-boot-spl.bin:	depend
2. 	$(MAKE) -C spl all
```

2.打开 spl/Makefile 分析，一开始就给我们导出 CONFIG_SPL_BUILD

```
1. CONFIG_SPL_BUILD := y
2. export CONFIG_SPL_BUILD
```

然后分析目标，因为我们的平台是三星的，所以，会有两个目标，一个是不带头信息的 u-boot-spl.bin，一个是(obj)(BOARD)-spl.bin。

```
1. ALL-y	+= $(obj)u-boot-spl.bin
2.
3. ifdef CONFIG_SAMSUNG
4. ALL-y	+= $(obj)$(BOARD)-spl.bin
5. endif
6.
7. all:	$(ALL-y)
```

搜索(obj)(BOARD)-spl.bin，发现，他是通过一个工具生成带头信息的 u-boot-spl.bin

```
ifdef CONFIG_SAMSUNG
$(obj)$(BOARD)-spl.bin: $(obj)u-boot-spl.bin
    $(TOPDIR)/board/$(BOARDDIR)/tools/mk$(BOARD)spl.exe \
        $(obj)u-boot-spl.bin $(obj)$(BOARD)-spl.bin
endif
```

好了，Makefile 就分析到这里，知道了 BL1 是如何生成的了。下面里分析代码了。

首先分析 arch/arm/cpu/armv7/start.S

```
reset:
    bl  save_boot_params
    /*
     * set the cpu to SVC32 mode
     */
    mrs r0, cpsr
    bic r0, r0, #0x1f
    orr r0, r0, #0xd3
    msr cpsr,r0

    /* the mask ROM code should have PLL and others stable */
#ifndef CONFIG_SKIP_LOWLEVEL_INIT
    bl  cpu_init_crit
#endif

/* Set stackpointer in internal RAM to call board_init_f */
call_board_init_f:
    ldr sp, =(CONFIG_SYS_INIT_SP_ADDR)
    bic sp, sp, #7 /* 8-byte alignment for ABI compliance */
    ldr r0,=0x00000000
    bl  board_init_f
```

其中，cpu_init_crit 中主要做初始化了 memory 和串口，接下来就是调用 C 函数 board_init_f 了，在调用 C 函数之前得设置栈，board_init_f 有两个，一个是在 arch/arm/lib/board.c，一个在 board/samsung/mini210/mmc_boot.c 那么道理是那个呢，看各个目录下的 Makefile

1.arch/arm/lib/下的

```
ifndef CONFIG_SPL_BUILD
...
COBJS-y += board.o
...
endif
```

2.board/samsung/mini210/

```
ifdef CONFIG_SPL_BUILD
COBJS  += mmc_boot.o
endif
```

得出，是 board/samsung/mini210/mmc_boot.c 了，打开

```
1. void board_init_f(unsigned long bootflag)
2. {
3.         __attribute__((noreturn)) void (*uboot)(void);
4.         copy_uboot_to_ram();
5.
6.         /* Jump to U-Boot image */
7.         uboot = (void *)CONFIG_SYS_TEXT_BASE;
8.         (*uboot)();
9.         /* Never returns Here */
10.}
```

得出，首先拷贝 BL2 完整的 UBOOT 到 RAM 里去，然后从 ram 里开始执行，然后就没有了。那么是如何拷贝的呢，这就得需要一个文档了 S5PV210_iROM_ApplicationNote_Preliminary_20091126.pdf 中

好了，看看 copy_uboot_to_ram 代码

```
1. typedef u32(*copy_sd_mmc_to_mem)
2. (u32 channel, u32 start_block, u16 block_size, u32 *trg, u32 init);
```

```
1. ch = *(volatile u32 *)(0xD0037488);
2.     copy_sd_mmc_to_mem copy_bl2 =
3.         (copy_sd_mmc_to_mem) (*(u32 *) (0xD0037F98));
```

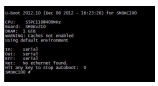
```
1. u32 ret;
2. if (ch == 0xEB000000) {
3.     ret = copy_bl2(0, MOVI_BL2_POS, MOVI_BL2_BLKCNT,
4.         CONFIG_SYS_TEXT_BASE, 0);
5. } else if (ch == 0xEB200000) {
6.     ret = copy_bl2(2, MOVI_BL2_POS, MOVI_BL2_BLKCNT,
7.         CONFIG_SYS_TEXT_BASE, 0);
8. }
```

其中，MOVI_BL2_POS 是 1，MOVI_BL2_BLKCNT 是 49，CONFIG_SYS_TEXT_BASE 是 uboot 在内存里的其实地址。

复制完成之后就会在内存里从 u-boot.bin 的 start.S 出开始重新执行了。

s5pv210 uboot-2012-10 移植(二) 之能够启动进入控制台

这次我们将从官网下载的最新 uboot-2012-10 移植到 s5pv210 开发板上，让其进入控制台，效果如下：



首先，我暂时没采用内核的 SPL，这个将在后面给补上，这里的 BL1 是我自己参考资料写的，我用的是 QT210 开发板，内存 1G，对于不同的开发板，需要重新配置 memory 和修改 uboot 在内存里的地址，也就是 CONFIG_SYS_TEXT_BASE。我的 BL1 代码在这里下载。

一、添加 smdkv210 单板

1.cp -a board/samsung/smdkc100 board/samsung/smdkv210

2.cp include/configs/smdkc100.h include/configs/smdkv210.h

3.vim boards.cfg，在 270 行添加

```
270 smdkv210                arm         armv7       smdkv210        samsung       s5pc1xx
```

4.make smdkv210_config 可以生成 u-boot.bin 了

二、让 u-boot.bin 在内存里启动起来

1.分析我的 BL1 代码可以得知，只拷贝了 u-boot.bin，并没有清楚 bss，在 ls arch/arm/cpu/armv7/start.S +126 中添加

```
reset:
    //by ZheGao clear bss
    ldr r0, =__bss_start
    ldr r1, =__bss_end__
    mov r2, #0x0
1:
    str r2, [r0], #4
    cmp r0, r1
    bne 1b
    //end of clear bss

    bl  save_boot_params
```

2.在 BL0 已经关了看门狗，在 BL1 里已经初始化了 memory 和串口，修改 board/samsung/smdkv210/lowlevel_init.S +39 中给屏蔽

```
.globl lowlevel_init
lowlevel_init:
    mov r9, lr
#if 0
    /* r5 has always zero */
    mov r5, #0

    ldr r8, =S5PC100_GPIO_BASE

```

```
    /* Disable Watchdog */
    ldr r0, =S5PC100_WATCHDOG_BASE     @0xEA200000
    orr r0, r0, #0x0
    str r5, [r0]

    /* setting SRAM */
    ldr r0, =S5PC100_SROMC_BASE
    ldr r1, =0x9
    str r1, [r0]

    /* S5PC100 has 3 groups of interrupt sources */
    ldr r0, =S5PC100_VIC0_BASE         @0xE4000000
    ldr r1, =S5PC100_VIC1_BASE         @0xE4000000
    ldr r2, =S5PC100_VIC2_BASE         @0xE4000000

    /* Disable all interrupts (VIC0, VIC1 and VIC2) */
    mvn r3, #0x0
    str r3, [r0, #0x14]            @INTENCLEAR
    str r3, [r1, #0x14]            @INTENCLEAR
    str r3, [r2, #0x14]            @INTENCLEAR

    /* Set all interrupts as IRQ */
    str r5, [r0, #0xc]             @INTSELECT
    str r5, [r1, #0xc]             @INTSELECT
    str r5, [r2, #0xc]             @INTSELECT

    /* Pending Interrupt Clear */
    str r5, [r0, #0xf00]           @INTADDRESS
    str r5, [r1, #0xf00]           @INTADDRESS
    str r5, [r2, #0xf00]           @INTADDRESS

    /* for UART */
    bl uart_asm_init

    /* for TZPC */
    bl tzpc_asm_init

1:
#endif
    mov lr, r9
    mov pc, lr
```

3.修改第一步的 sp 指针，在 include/configs/smdkv210.h +228 修改

```
//#define CONFIG_SYS_INIT_SP_ADDR   (CONFIG_SYS_LOAD_ADDR - 0x1000000)
#define CONFIG_SYS_INIT_SP_ADDR (0x30000000)
```

4.修改 board/samsung/smdkv210/smdkc100.c +80

```
int checkboard(void)
{
```

```
    printf("Board:\tSMDKv210\n");
    return 0;
}
```

5.修改 include/configs/smdkv210.h +51

```
//#define CONFIG_SYS_SDRAM_BASE     0x30000000
#define CONFIG_SYS_SDRAM_BASE       0x20000000
```

6.修改 include/configs/smdkv210.h +189

```
//#define PHYS_SDRAM_1_SIZE (128 << 20)   /* 0x8000000, 128 MB Bank #1 */
#define PHYS_SDRAM_1_SIZE   (0x40000000)    /* 0x8000000, 128 MB Bank #1 */
```

7.修改 board/samsung/smdkv210/config.mk +16

```
CONFIG_SYS_TEXT_BASE = 0x5ff00000
```

8.修改 arch/arm/config.mk +88

```
#LDFLAGS_u-boot += -pie
```

9.修改 include/configs/smdkv210.h +216

```
//#define CONFIG_ENV_IS_IN_ONENAND  1
#define CONFIG_ENV_IS_NOWHERE
```

10.修改 arch/arm/lib/board.c +384

```
//addr -= gd->mon_len;
addr = 0x5ff00000;
```

11.因为 bl1 已经复制程序到指定的地址就不需要重新定位代码了，但还是需要重新设置栈，所以修改了上面的第 8 步，修改 arch/arm/cpu/armv7/start.S +192

```
ENTRY(relocate_code)
    mov r4, r0  /* save addr_sp */
    mov r5, r1  /* save addr of gd */
    mov r6, r2  /* save addr of destination */

#if 0
    //debug
    ldr r0, =0xE0200C00
    ldr r1, =0x1111
     str r1, [r0]

    ldr r0, =0xE0200C04
    ldr r1, =(7)
```

```
    str r1, [r0]
#endif

    mov sp, r4
    mov r0, r5
    mov r1, r6

    bl board_init_r
#if 0
    /* Set up the stack                         */
stack_setup:
    mov sp, r4

    adr r0, _start
    cmp r0, r6
    moveq   r9, #0      /* no relocation. relocation offset(r9) = 0 */
    beq clear_bss       /* skip relocation */
    mov r1, r6          /* r1 <- scratch for copy_loop */
    ldr r3, _image_copy_end_ofs
    add r2, r0, r3      /* r2 <- source end address      */

copy_loop:
    ldmia   r0!, {r9-r10}       /* copy from source address [r0]    */
    stmia   r1!, {r9-r10}       /* copy to   target address [r1]    */
    cmp r0, r2          /* until source end address [r2]    */
    blo copy_loop

    /*
     * fix .rel.dyn relocations
     */
    ldr r0, _TEXT_BASE      /* r0 <- Text base */
    sub r9, r6, r0      /* r9 <- relocation offset */
    ldr r10, _dynsym_start_ofs  /* r10 <- sym table ofs */
    add r10, r10, r0        /* r10 <- sym table in FLASH */
    ldr r2, _rel_dyn_start_ofs  /* r2 <- rel dyn start ofs */
    add r2, r2, r0      /* r2 <- rel dyn start in FLASH */
    ldr r3, _rel_dyn_end_ofs    /* r3 <- rel dyn end ofs */
    add r3, r3, r0      /* r3 <- rel dyn end in FLASH */
fixloop:
    ldr r0, [r2]        /* r0 <- location to fix up, IN FLASH! */
    add r0, r0, r9      /* r0 <- location to fix up in RAM */
    ldr r1, [r2, #4]
    and r7, r1, #0xff
    cmp r7, #23         /* relative fixup? */
    beq fixrel
    cmp r7, #2          /* absolute fixup? */
    beq fixabs
    /* ignore unknown type of fixup */
    b   fixnext
fixabs:
    /* absolute fix: set location to (offset) symbol value */
    mov r1, r1, LSR #4      /* r1 <- symbol index in .dynsym */
```

```
    add r1, r10, r1     /* r1 <- address of symbol in table */
    ldr r1, [r1, #4]        /* r1 <- symbol value */
    add r1, r1, r9      /* r1 <- relocated sym addr */
    b   fixnext
fixrel:
    /* relative fix: increase location by offset */
    ldr r1, [r0]
    add r1, r1, r9
fixnext:
    str r1, [r0]
    add r2, r2, #8      /* each rel.dyn entry is 8 bytes */
    cmp r2, r3
    blo fixloop
    b   clear_bss
_rel_dyn_start_ofs:
    .word __rel_dyn_start - _start
_rel_dyn_end_ofs:
    .word __rel_dyn_end - _start
_dynsym_start_ofs:
    .word __dynsym_start - _start

clear_bss:
    ldr r0, _bss_start_ofs
    ldr r1, _bss_end_ofs
    mov r4, r6          /* reloc addr */
    add r0, r0, r4
    add r1, r1, r4
    mov r2, #0x00000000     /* clear                */

clbss_l:cmp r0, r1          /* clear loop... */
    bhs clbss_e         /* if reached end of bss, exit */
    str r2, [r0]
    add r0, r0, #4
    b   clbss_l
clbss_e:

/*
 * We are done. Do not return, instead branch to second part of board
 * initialization, now running from RAM.
 */
jump_2_ram:
/*
 * If I-cache is enabled invalidate it
 */
#ifndef CONFIG_SYS_ICACHE_OFF
    mcr p15, 0, r0, c7, c5, 0   @ invalidate icache
    mcr     p15, 0, r0, c7, c10, 4  @ DSB
    mcr     p15, 0, r0, c7, c5, 4   @ ISB
#endif
/*
 * Move vector table
 */
```

```
#if !defined(CONFIG_TEGRA20)
    /* Set vector address in CP15 VBAR register */
    ldr     r0, =_start
    add     r0, r0, r9
    mcr     p15, 0, r0, c12, c0, 0  @Set VBAR
#endif /* !Tegra20 */

    ldr r0, _board_init_r_ofs
    adr r1, _start
    add lr, r0, r1
    add lr, lr, r9
    /* setup parameters for board_init_r */
    mov r0, r5      /* gd_t */
    mov r1, r6      /* dest_addr */
    /* jump to it ... */
    mov pc, lr

_board_init_r_ofs:
    .word board_init_r - _start
#endif
ENDPROC(relocate_code)
#endif
```

重新 make 一下，使用下面命令烧写到 sd 卡里，插入开发板启动即可，看看上面的启动信息，还有很多很多地方要完善~

```
dd iflag=dsync oflag=dsync if=bISD.bin of=/dev/sdb seek=1
dd iflag=dsync oflag=dsync if=u-boot.bin of=/dev/sdb seek=49
```

s5pv210 uboot-2012-10 移植(三) 之支持 SPL

上次的 uboot 的 BL1 是自己实现的，今天就来让 uboot-2012-10 支持 SPL 功能，但不是完全用的 uboot 本身的代码，也不知道这样是好还是坏。

1.分析顶层目录的 Makefile 可以知道，需要添加 CONFIG_SPL 配置，这在前面的已经说过了，跟踪 start.S 代码，得知编译需要 arch/arm/lib/spl.c 文件，查看 arch/arm/lib/Makefile 得知，需要添加 CONFIG_SPL_FRAMEWORK 配置，所以 include/configs/smdkv210.h +70 添加

```
/* SPL */
#define CONFIG_SPL
#define CONFIG_SPL_FRAMEWORK
```

2.因为 SPL 不需要 BSS 清零，所以修改 arch/arm/cpu/armv7/start.S +128

```
#ifndef CONFIG_SPL_BUILD
    //by ZheGao clear bss
    ldr r0, =__bss_start
    ldr r1, =__bss_end__
    mov r2, #0x0
1:
    str r2, [r0], #4
    cmp r0, r1
    bne 1b
    //end of clear bss

    bl  save_boot_params
#endif
```

3.因为 BL1 需要初始化 memory，所以在 lowlevel_init 中添加，

board/samsung/smdkv210/lowlevel_init.S +43

```
#ifdef CONFIG_SPL_BUILD
    bl mem_ctrl_asm_init
#endif
```

4.mem_ctrl_asm_init 在同目录下的 mem_setup.S 文件中，修改

board/samsung/smdkv210/mem_setup.S 添加 memory 的配置

```
#define ELFIN_GPIO_BASE       0xE0200000
#define MP1_0DRV_SR_OFFSET    0x3CC
#define MP1_1DRV_SR_OFFSET    0x3EC
#define MP1_2DRV_SR_OFFSET    0x40C
#define MP1_3DRV_SR_OFFSET    0x42C
#define MP1_4DRV_SR_OFFSET    0x44C
#define MP1_5DRV_SR_OFFSET    0x46C
#define MP1_6DRV_SR_OFFSET    0x48C
#define MP1_7DRV_SR_OFFSET    0x4AC
#define MP1_8DRV_SR_OFFSET    0x4CC
#define MP2_0DRV_SR_OFFSET    0x4EC
#define MP2_1DRV_SR_OFFSET    0x50C
#define MP2_2DRV_SR_OFFSET    0x52C
#define MP2_3DRV_SR_OFFSET    0x54C
#define MP2_4DRV_SR_OFFSET    0x56C
#define MP2_5DRV_SR_OFFSET    0x58C
#define MP2_6DRV_SR_OFFSET    0x5AC
#define MP2_7DRV_SR_OFFSET    0x5CC
#define MP2_8DRV_SR_OFFSET    0x5EC
#define APB_DMC_0_BASE        0xF0000000
#define APB_DMC_1_BASE        0xF1400000
```

```
22.#define ASYNC_MSYS_DMC0_BASE       0xF1E00000
23.
24.#define DMC_CONCONTROL          0x00
25.#define DMC_MEMCONTROL          0x04
26.#define DMC_MEMCONFIG0          0x08
27.#define DMC_MEMCONFIG1          0x0C
28.#define DMC_DIRECTCMD           0x10
29.#define DMC_PRECHCONFIG         0x14
30.#define DMC_PHYCONTROL0         0x18
31.#define DMC_PHYCONTROL1         0x1C
32.#define DMC_RESERVED            0x20
33.#define DMC_PWRDNCONFIG         0x28
34.#define DMC_TIMINGAREF          0x30
35.#define DMC_TIMINGROW           0x34
36.#define DMC_TIMINGDATA          0x38
37.#define DMC_TIMINGPOWER         0x3C
38.#define DMC_PHYSTATUS           0x40
39.#define DMC_CHIP0STATUS         0x48
40.#define DMC_CHIP1STATUS         0x4C
41.#define DMC_AREFSTATUS          0x50
42.#define DMC_MRSTATUS            0x54
43.#define DMC_PHYTEST0            0x58
44.#define DMC_PHYTEST1            0x5C
45.#define DMC_QOSCONTROL0         0x60
46.#define DMC_QOSCONFIG0          0x64
47.#define DMC_QOSCONTROL1         0x68
48.#define DMC_QOSCONFIG1          0x6C
49.#define DMC_QOSCONTROL2         0x70
50.#define DMC_QOSCONFIG2          0x74
51.#define DMC_QOSCONTROL3         0x78
52.#define DMC_QOSCONFIG3          0x7C
53.#define DMC_QOSCONTROL4         0x80
54.#define DMC_QOSCONFIG4          0x84
55.#define DMC_QOSCONTROL5         0x88
56.#define DMC_QOSCONFIG5          0x8C
57.#define DMC_QOSCONTROL6         0x90
58.#define DMC_QOSCONFIG6          0x94
59.#define DMC_QOSCONTROL7         0x98
60.#define DMC_QOSCONFIG7          0x9C
61.#define DMC_QOSCONTROL8         0xA0
62.#define DMC_QOSCONFIG8          0xA4
63.#define DMC_QOSCONTROL9         0xA8
64.#define DMC_QOSCONFIG9          0xAC
65.#define DMC_QOSCONTROL10        0xB0
66.#define DMC_QOSCONFIG10         0xB4
67.#define DMC_QOSCONTROL11        0xB8
68.#define DMC_QOSCONFIG11         0xBC
69.#define DMC_QOSCONTROL12        0xC0
70.#define DMC_QOSCONFIG12         0xC4
71.#define DMC_QOSCONTROL13        0xC8
72.#define DMC_QOSCONFIG13         0xCC
73.#define DMC_QOSCONTROL14        0xD0
74.#define DMC_QOSCONFIG14         0xD4
```

```
75.#define DMC_QOSCONTROL15        0xD8
76.#define DMC_QOSCONFIG15         0xDC
77.
78.#define DMC0_MEMCONFIG_0    0x20E01323  // MemConfig0   256MB config, 8 banks,Mapping Met
   hod[12:15]0:linear, 1:linterleaved, 2:Mixed
79.#define DMC0_MEMCONFIG_1    0x40F01323  // MemConfig1
80.#define DMC0_TIMINGA_REF    0x00000618  // TimingAref   7.8us*133MHz=1038(0x40E), 100MHz
   =780(0x30C), 20MHz=156(0x9C), 10MHz=78(0x4E)
81.#define DMC0_TIMING_ROW     0x28233287  // TimingRow    for @200MHz
82.#define DMC0_TIMING_DATA    0x23240304  // TimingData   CL=3
83.#define DMC0_TIMING_PWR     0x09C80232  // TimingPower
84.
85.#define DMC1_MEMCONTROL     0x00202400  // MemControl   BL=4, 2 chip, DDR2 type, dynamic s
   elf refresh, force precharge, dynamic power down off
86.#define DMC1_MEMCONFIG_0    0x40C01323  // MemConfig0   512MB config, 8 banks,Mapping Me
   thod[12:15]0:linear, 1:linterleaved, 2:Mixed
87.#define DMC1_MEMCONFIG_1    0x00E01323  // MemConfig1
88.#define DMC1_TIMINGA_REF    0x00000618  // TimingAref   7.8us*133MHz=1038(0x40E), 100MHz
   =780(0x30C), 20MHz=156(0x9C), 10MHz=78(0x4
89.#define DMC1_TIMING_ROW     0x28233289  // TimingRow    for @200MHz
90.#define DMC1_TIMING_DATA    0x23240304  // TimingData   CL=3
91.#define DMC1_TIMING_PWR     0x08280232  // TimingPower
92.
93.
94.    .globl mem_ctrl_asm_init
95.mem_ctrl_asm_init:
96.#if 0
97.    ldr r6, =S5PC100_DMC_BASE           @ 0xE6000000
98.
99.    /* DLL parameter setting */
100.    ldr r1, =0x50101000
101.    str r1, [r6, #0x018]            @ PHYCONTROL0
102.    ldr r1, =0xf4
103.    str r1, [r6, #0x01C]            @ PHYCONTROL1
104.    ldr r1, =0x0
105.    str r1, [r6, #0x020]            @ PHYCONTROL2
106.
107.    /* DLL on */
108.    ldr r1, =0x50101002
109.    str r1, [r6, #0x018]            @ PHYCONTROL0
110.
111.    /* DLL start */
112.    ldr r1, =0x50101003
113.    str r1, [r6, #0x018]            @ PHYCONTROL0
114.
115.    /* Force value locking for DLL off */
116.    str r1, [r6, #0x018]            @ PHYCONTROL0
117.
118.    /* DLL off */
119.    ldr r1, =0x50101001
120.    str r1, [r6, #0x018]            @ PHYCONTROL0
121.
```

```
    /* auto refresh off */
    ldr r1, =0xff001010
    str r1, [r6, #0x000]            @ CONCONTROL

    /*
     * Burst Length 4, 2 chips, 32-bit, LPDDR
     * OFF: dynamic self refresh, force precharge, dynamic power down off
     */
    ldr r1, =0x00212100
    str r1, [r6, #0x004]            @ MEMCONTROL

    /*
     * Note:
     * If Bank0 has OneDRAM we place it at 0x2800'0000
     * So finally Bank1 should address start at at 0x2000'0000
     */
    mov r4, #0x0

swap_memory:
    /*
     * Bank0
     * 0x30 -> 0x30000000
     * 0xf8 -> 0x37FFFFFF
     * [15:12] 0: Linear
     * [11:8 ] 2: 9 bits
     * [ 7:4 ] 2: 14 bits
     * [ 3:0 ] 2: 4 banks
     */
    ldr r1, =0x30f80222
    /* if r4 is 1, swap the bank */
    cmp r4, #0x1
    orreq   r1, r1, #0x08000000
    str r1, [r6, #0x008]            @ MEMCONFIG0

    /*
     * Bank1
     * 0x38 -> 0x38000000
     * 0xf8 -> 0x3fFFFFFF
     * [15:12] 0: Linear
     * [11:8 ] 2: 9 bits
     * [ 7:4 ] 2: 14 bits
     * [ 3:0 ] 2: 4 banks
     */
    ldr r1, =0x38f80222
    /* if r4 is 1, swap the bank */
    cmp r4, #0x1
    biceq   r1, r1, #0x08000000
    str r1, [r6, #0x00c]            @ MEMCONFIG1

    ldr r1, =0x20000000
    str r1, [r6, #0x014]            @ PRECHCONFIG

```

```
174.    /*
175.     * FIXME: Please verify these values
176.     * 7.8us * 166MHz %LE %LONG1294(0x50E)
177.     * 7.8us * 133MHz %LE %LONG1038(0x40E),
178.     * 7.8us * 100MHz %LE %LONG780(0x30C),
179.     * 7.8us * 20MHz  %LE %LONG156(0x9C),
180.     * 7.8us * 10MHz  %LE %LONG78(0x4E)
181.     */
182.    ldr r1, =0x0000050e
183.    str r1, [r6, #0x030]          @ TIMINGAREF
184.
185.    /* 166 MHz */
186.    ldr r1, =0x0c233287
187.    str r1, [r6, #0x034]          @ TIMINGROW
188.
189.    /* twtr=3 twr=2 trtp=3 cl=3 wl=3 rl=3 */
190.    ldr r1, =0x32330303
191.    str r1, [r6, #0x038]          @ TIMINGDATA
192.
193.    /* tfaw=4 sxsr=0x14 txp=0x14 tcke=3 tmrd=3 */
194.    ldr r1, =0x04141433
195.    str r1, [r6, #0x03C]          @ TIMINGPOWER
196.
197.    /* chip0 Deselect */
198.    ldr r1, =0x07000000
199.    str r1, [r6, #0x010]          @ DIRECTCMD
200.
201.    /* chip0 PALL */
202.    ldr r1, =0x01000000
203.    str r1, [r6, #0x010]          @ DIRECTCMD
204.
205.    /* chip0 REFA */
206.    ldr r1, =0x05000000
207.    str r1, [r6, #0x010]          @ DIRECTCMD
208.    /* chip0 REFA */
209.    str r1, [r6, #0x010]          @ DIRECTCMD
210.
211.    /* chip0 MRS, CL%LE %LONG3, BL%LE %LONG4 */
212.    ldr r1, =0x00000032
213.    str r1, [r6, #0x010]          @ DIRECTCMD
214.
215.    /* chip1 Deselect */
216.    ldr r1, =0x07100000
217.    str r1, [r6, #0x010]          @ DIRECTCMD
218.
219.    /* chip1 PALL */
220.    ldr r1, =0x01100000
221.    str r1, [r6, #0x010]          @ DIRECTCMD
222.
223.    /* chip1 REFA */
224.    ldr r1, =0x05100000
225.    str r1, [r6, #0x010]          @ DIRECTCMD
```

```
    /* chip1 REFA */
    str r1, [r6, #0x010]          @ DIRECTCMD

    /* chip1 MRS, CL%LE %LONG3, BL%LE %LONG4 */
    ldr r1, =0x00100032
    str r1, [r6, #0x010]          @ DIRECTCMD

    /* auto refresh on */
    ldr r1, =0xff002030
    str r1, [r6, #0x000]          @ CONCONTROL

    /* PwrdnConfig */
    ldr r1, =0x00100002
    str r1, [r6, #0x028]          @ PWRDNCONFIG

    /* BL%LE %LONG */
    ldr r1, =0xff212100
    str r1, [r6, #0x004]          @ MEMCONTROL


    /* Try to test memory area */
    cmp r4, #0x1
    beq 1f

    mov r4, #0x1
    ldr r1, =0x37ffff00
    str r4, [r1]
    str r4, [r1, #0x4]            @ dummy write
    ldr r0, [r1]
    cmp r0, r4
    bne swap_memory
#endif

    /* DMC0 Drive Strength (Setting 2X) */

    ldr r0, =ELFIN_GPIO_BASE

    ldr r1, =0x0000AAAA
    str r1, [r0, #MP1_0DRV_SR_OFFSET]

    ldr r1, =0x0000AAAA
    str r1, [r0, #MP1_1DRV_SR_OFFSET]

    ldr r1, =0x0000AAAA
    str r1, [r0, #MP1_2DRV_SR_OFFSET]

    ldr r1, =0x0000AAAA
    str r1, [r0, #MP1_3DRV_SR_OFFSET]

    ldr r1, =0x0000AAAA
    str r1, [r0, #MP1_4DRV_SR_OFFSET]

```

```
    ldr r1, =0x0000AAAA
    str r1, [r0, #MP1_5DRV_SR_OFFSET]

    ldr r1, =0x0000AAAA
    str r1, [r0, #MP1_6DRV_SR_OFFSET]

    ldr r1, =0x0000AAAA
    str r1, [r0, #MP1_7DRV_SR_OFFSET]

    ldr r1, =0x00002AAA
    str r1, [r0, #MP1_8DRV_SR_OFFSET]


    /* DMC1 Drive Strength (Setting 2X) */

    ldr r0, =ELFIN_GPIO_BASE

    ldr r1, =0x0000AAAA
    str r1, [r0, #MP2_0DRV_SR_OFFSET]

    ldr r1, =0x0000AAAA
    str r1, [r0, #MP2_1DRV_SR_OFFSET]

    ldr r1, =0x0000AAAA
    str r1, [r0, #MP2_2DRV_SR_OFFSET]

    ldr r1, =0x0000AAAA
    str r1, [r0, #MP2_3DRV_SR_OFFSET]

    ldr r1, =0x0000AAAA
    str r1, [r0, #MP2_4DRV_SR_OFFSET]

    ldr r1, =0x0000AAAA
    str r1, [r0, #MP2_5DRV_SR_OFFSET]

    ldr r1, =0x0000AAAA
    str r1, [r0, #MP2_6DRV_SR_OFFSET]

    ldr r1, =0x0000AAAA
    str r1, [r0, #MP2_7DRV_SR_OFFSET]

    ldr r1, =0x00002AAA
    str r1, [r0, #MP2_8DRV_SR_OFFSET]

    /* DMC0 initialization at single Type*/
    ldr r0, =APB_DMC_0_BASE

    ldr r1, =0x00101000         @PhyControl0 DLL parameter setting, manual 0x00101000
    str r1, [r0, #DMC_PHYCONTROL0]

    ldr r1, =0x00000086         @PhyControl1 DLL parameter setting, LPDDR/LPDDR2 Case
    str r1, [r0, #DMC_PHYCONTROL1]
```

```
330.
331.    ldr r1, =0x00101002            @PhyControl0 DLL on
332.    str r1, [r0, #DMC_PHYCONTROL0]
333.
334.    ldr r1, =0x00101003            @PhyControl0 DLL start
335.    str r1, [r0, #DMC_PHYCONTROL0]
336.
337.find_lock_val:
338.    ldr r1, [r0, #DMC_PHYSTATUS]       @Load Phystatus register value
339.    and r2, r1, #0x7
340.    cmp r2, #0x7               @Loop until DLL is locked
341.    bne find_lock_val
342.
343.    and r1, #0x3fc0
344.    mov r2, r1, LSL #18
345.    orr r2, r2, #0x100000
346.    orr r2 ,r2, #0x1000
347.
348.    orr r1, r2, #0x3               @Force Value locking
349.    str r1, [r0, #DMC_PHYCONTROL0]
350.
351.#if 0   /* Memory margin test 10.01.05 */
352.    orr r1, r2, #0x1               @DLL off
353.    str r1, [r0, #DMC_PHYCONTROL0]
354.#endif
355.    /* setting DDR2 */
356.    ldr r1, =0x0FFF2010            @ConControl auto refresh off
357.    str r1, [r0, #DMC_CONCONTROL]
358.
359.    ldr r1, =0x00212400            @MemControl BL=4, 2 chip, DDR2 type, dynamic self refresh, fo
    rce precharge, dynamic power down off
360.    str r1, [r0, #DMC_MEMCONTROL]
361.
362.    ldr r1, =DMC0_MEMCONFIG_0          @MemConfig0 256MB config, 8 banks,Mapping Method[
    12:15]0:linear, 1:linterleaved, 2:Mixed
363.    str r1, [r0, #DMC_MEMCONFIG0]
364.
365.    ldr r1, =DMC0_MEMCONFIG_1          @MemConfig1
366.    str r1, [r0, #DMC_MEMCONFIG1]
367.
368.    ldr r1, =0xFF000000            @PrechConfig
369.    str r1, [r0, #DMC_PRECHCONFIG]
370.
371.    ldr r1, =DMC0_TIMINGA_REF          @TimingAref 7.8us*133MHz=1038(0x40E), 100MHz=780
    (0x30C), 20MHz=156(0x9C), 10MHz=78(0x4E)
372.    str r1, [r0, #DMC_TIMINGAREF]
373.
374.    ldr r1, =DMC0_TIMING_ROW           @TimingRow  for @200MHz
375.    str r1, [r0, #DMC_TIMINGROW]
376.
377.    ldr r1, =DMC0_TIMING_DATA          @TimingData CL=3
378.    str r1, [r0, #DMC_TIMINGDATA]
```

```
379.
380.    ldr r1, =DMC0_TIMING_PWR          @TimingPower
381.    str r1, [r0, #DMC_TIMINGPOWER]
382.
383.    ldr r1, =0x07000000          @DirectCmd  chip0 Deselect
384.    str r1, [r0, #DMC_DIRECTCMD]
385.
386.    ldr r1, =0x01000000          @DirectCmd  chip0 PALL
387.    str r1, [r0, #DMC_DIRECTCMD]
388.
389.    ldr r1, =0x00020000          @DirectCmd  chip0 EMRS2
390.    str r1, [r0, #DMC_DIRECTCMD]
391.
392.    ldr r1, =0x00030000          @DirectCmd  chip0 EMRS3
393.    str r1, [r0, #DMC_DIRECTCMD]
394.
395.    ldr r1, =0x00010400          @DirectCmd  chip0 EMRS1 (MEM DLL on, DQS# disable)
396.    str r1, [r0, #DMC_DIRECTCMD]
397.
398.    ldr r1, =0x00000542          @DirectCmd  chip0 MRS (MEM DLL reset) CL=4, BL=4
399.    str r1, [r0, #DMC_DIRECTCMD]
400.
401.    ldr r1, =0x01000000          @DirectCmd  chip0 PALL
402.    str r1, [r0, #DMC_DIRECTCMD]
403.
404.    ldr r1, =0x05000000          @DirectCmd  chip0 REFA
405.    str r1, [r0, #DMC_DIRECTCMD]
406.
407.    ldr r1, =0x05000000          @DirectCmd  chip0 REFA
408.    str r1, [r0, #DMC_DIRECTCMD]
409.
410.    ldr r1, =0x00000442          @DirectCmd  chip0 MRS (MEM DLL unreset)
411.    str r1, [r0, #DMC_DIRECTCMD]
412.
413.    ldr r1, =0x00010780          @DirectCmd  chip0 EMRS1 (OCD default)
414.    str r1, [r0, #DMC_DIRECTCMD]
415.
416.    ldr r1, =0x00010400          @DirectCmd  chip0 EMRS1 (OCD exit)
417.    str r1, [r0, #DMC_DIRECTCMD]
418.
419.    ldr r1, =0x07100000          @DirectCmd  chip1 Deselect
420.    str r1, [r0, #DMC_DIRECTCMD]
421.
422.    ldr r1, =0x01100000          @DirectCmd  chip1 PALL
423.    str r1, [r0, #DMC_DIRECTCMD]
424.
425.    ldr r1, =0x00120000          @DirectCmd  chip1 EMRS2
426.    str r1, [r0, #DMC_DIRECTCMD]
427.
428.    ldr r1, =0x00130000          @DirectCmd  chip1 EMRS3
429.    str r1, [r0, #DMC_DIRECTCMD]
430.
```

```
    ldr r1, =0x00110400          @DirectCmd  chip1 EMRS1 (MEM DLL on, DQS# disable)
    str r1, [r0, #DMC_DIRECTCMD]

    ldr r1, =0x00100542          @DirectCmd  chip1 MRS (MEM DLL reset) CL=4, BL=4
    str r1, [r0, #DMC_DIRECTCMD]

    ldr r1, =0x01100000          @DirectCmd  chip1 PALL
    str r1, [r0, #DMC_DIRECTCMD]

    ldr r1, =0x05100000          @DirectCmd  chip1 REFA
    str r1, [r0, #DMC_DIRECTCMD]

    ldr r1, =0x05100000          @DirectCmd  chip1 REFA
    str r1, [r0, #DMC_DIRECTCMD]

    ldr r1, =0x00100442          @DirectCmd  chip1 MRS (MEM DLL unreset)
    str r1, [r0, #DMC_DIRECTCMD]

    ldr r1, =0x00110780          @DirectCmd  chip1 EMRS1 (OCD default)
    str r1, [r0, #DMC_DIRECTCMD]

    ldr r1, =0x00110400          @DirectCmd  chip1 EMRS1 (OCD exit)
    str r1, [r0, #DMC_DIRECTCMD]

    ldr r1, =0x0FF02030          @ConControl auto refresh on
    str r1, [r0, #DMC_CONCONTROL]

    ldr r1, =0xFFFF00FF          @PwrdnConfig
    str r1, [r0, #DMC_PWRDNCONFIG]

    ldr r1, =0x00202400          @MemControl BL=4, 2 chip, DDR2 type, dynamic self refresh, fo
rce precharge, dynamic power down off
    str r1, [r0, #DMC_MEMCONTROL]

    /* DMC1 initialization */
    ldr r0, =APB_DMC_1_BASE

    ldr r1, =0x00101000          @Phycontrol0 DLL parameter setting
    str r1, [r0, #DMC_PHYCONTROL0]

    ldr r1, =0x00000086          @Phycontrol1 DLL parameter setting
    str r1, [r0, #DMC_PHYCONTROL1]

    ldr r1, =0x00101002          @PhyControl0 DLL on
    str r1, [r0, #DMC_PHYCONTROL0]

    ldr r1, =0x00101003          @PhyControl0 DLL start
    str r1, [r0, #DMC_PHYCONTROL0]
find_lock_val1:
    ldr r1, [r0, #DMC_PHYSTATUS]       @Load Phystatus register value
    and r2, r1, #0x7
    cmp r2, #0x7               @Loop until DLL is locked
```

```
    bne find_lock_val1
    
    and r1, #0x3fc0
    mov r2, r1, LSL #18
    orr r2, r2, #0x100000
    orr r2, r2, #0x1000
    
    orr r1, r2, #0x3            @Force Value locking
    str r1, [r0, #DMC_PHYCONTROL0]
    
#if 0   /* Memory margin test 10.01.05 */
    orr r1, r2, #0x1            @DLL off
    str r1, [r0, #DMC_PHYCONTROL0]
#endif
    
    /* settinf fot DDR2 */
    ldr r0, =APB_DMC_1_BASE
    
    ldr r1, =0x0FFF2010         @auto refresh off
    str r1, [r0, #DMC_CONCONTROL]
    
    ldr r1, =DMC1_MEMCONTROL        @MemControl BL=4, 2 chip, DDR2 type, dynamic self refresh, force precharge, dynamic power down off
    str r1, [r0, #DMC_MEMCONTROL]
    
    ldr r1, =DMC1_MEMCONFIG_0       @MemConfig0 512MB config, 8 banks,Mapping Method[12:15]0:linear, 1:linterleaved, 2:Mixed
    str r1, [r0, #DMC_MEMCONFIG0]
    
    ldr r1, =DMC1_MEMCONFIG_1       @MemConfig1
    str r1, [r0, #DMC_MEMCONFIG1]
    
    ldr r1, =0xFF000000
    str r1, [r0, #DMC_PRECHCONFIG]
    
    ldr r1, =DMC1_TIMINGA_REF       @TimingAref 7.8us*133MHz=1038(0x40E), 100MHz=780(0x30C), 20MHz=156(0x9C), 10MHz=78(0x4
    str r1, [r0, #DMC_TIMINGAREF]
    
    ldr r1, =DMC1_TIMING_ROW        @TimingRow  for @200MHz
    str r1, [r0, #DMC_TIMINGROW]
    
    ldr r1, =DMC1_TIMING_DATA       @TimingData CL=3
    str r1, [r0, #DMC_TIMINGDATA]
    
    ldr r1, =DMC1_TIMING_PWR        @TimingPower
    str r1, [r0, #DMC_TIMINGPOWER]
    
    
    ldr r1, =0x07000000         @DirectCmd  chip0 Deselect
    str r1, [r0, #DMC_DIRECTCMD]
    
```

```
ldr r1, =0x01000000          @DirectCmd  chip0 PALL
str r1, [r0, #DMC_DIRECTCMD]

ldr r1, =0x00020000          @DirectCmd  chip0 EMRS2
str r1, [r0, #DMC_DIRECTCMD]

ldr r1, =0x00030000          @DirectCmd  chip0 EMRS3
str r1, [r0, #DMC_DIRECTCMD]

ldr r1, =0x00010400          @DirectCmd  chip0 EMRS1 (MEM DLL on, DQS# disable)
str r1, [r0, #DMC_DIRECTCMD]

ldr r1, =0x00000542          @DirectCmd  chip0 MRS (MEM DLL reset) CL=4, BL=4
str r1, [r0, #DMC_DIRECTCMD]

ldr r1, =0x01000000          @DirectCmd  chip0 PALL
str r1, [r0, #DMC_DIRECTCMD]

ldr r1, =0x05000000          @DirectCmd  chip0 REFA
str r1, [r0, #DMC_DIRECTCMD]

ldr r1, =0x05000000          @DirectCmd  chip0 REFA
str r1, [r0, #DMC_DIRECTCMD]

ldr r1, =0x00000442          @DirectCmd  chip0 MRS (MEM DLL unreset)
str r1, [r0, #DMC_DIRECTCMD]

ldr r1, =0x00010780          @DirectCmd  chip0 EMRS1 (OCD default)
str r1, [r0, #DMC_DIRECTCMD]

ldr r1, =0x00010400          @DirectCmd  chip0 EMRS1 (OCD exit)
str r1, [r0, #DMC_DIRECTCMD]

ldr r1, =0x07100000          @DirectCmd  chip1 Deselect
str r1, [r0, #DMC_DIRECTCMD]

ldr r1, =0x01100000          @DirectCmd  chip1 PALL
str r1, [r0, #DMC_DIRECTCMD]

ldr r1, =0x00120000          @DirectCmd  chip1 EMRS2
str r1, [r0, #DMC_DIRECTCMD]

ldr r1, =0x00130000          @DirectCmd  chip1 EMRS3
str r1, [r0, #DMC_DIRECTCMD]

ldr r1, =0x00110440          @DirectCmd  chip1 EMRS1 (MEM DLL on, DQS# disable)
str r1, [r0, #DMC_DIRECTCMD]

ldr r1, =0x00100542          @DirectCmd  chip1 MRS (MEM DLL reset) CL=4, BL=4
str r1, [r0, #DMC_DIRECTCMD]

ldr r1, =0x01100000          @DirectCmd  chip1 PALL
```

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

    ldr r1, =0x05100000          @DirectCmd  chip1 REFA
    str r1, [r0, #DMC_DIRECTCMD]

    ldr r1, =0x05100000          @DirectCmd  chip1 REFA
    str r1, [r0, #DMC_DIRECTCMD]

    ldr r1, =0x00100442          @DirectCmd  chip1 MRS (MEM DLL unreset)
    str r1, [r0, #DMC_DIRECTCMD]

    ldr r1, =0x00110780          @DirectCmd  chip1 EMRS1 (OCD default)
    str r1, [r0, #DMC_DIRECTCMD]

    ldr r1, =0x00110400          @DirectCmd  chip1 EMRS1 (OCD exit)
    str r1, [r0, #DMC_DIRECTCMD]

    ldr r1, =0x0FF02030          @ConControl auto refresh on
    str r1, [r0, #DMC_CONCONTROL]

    ldr r1, =0xFFFF00FF          @PwrdnConfig
    str r1, [r0, #DMC_PWRDNCONFIG]

    ldr r1, =DMC1_MEMCONTROL          @MemControl BL=4, 2 chip, DDR2 type, dynamic self re
fresh, force precharge, dynamic power down off
    str r1, [r0, #DMC_MEMCONTROL]

    mov pc, lr
```

5.修改 board/samsung/smdkv210/Makefile 添加对 mem_setup.S 的选则编译，uboot 里就不需要编译了，因为 BL1 已经初始化过了，在执行一次也没问题的

```
ifndef CONFIG_SPL_BUILD
SOBJS   := lowlevel_init.o
endif

ifdef CONFIG_SPL_BUILD
SOBJS   := lowlevel_init.o mem_setup.o
endif
```

6.修改 arch/arm/lib/spl.c 的 board_init_f 函数

```
void __weak board_init_f(ulong dummy)
{
    __attribute__((noreturn)) void (*uboot)(void);
#if 0
    /* Set the stack pointer. */
    asm volatile("mov sp, %0\n" : : "r"(CONFIG_SPL_STACK));

    /* Clear the BSS. */
```

```
    memset(__bss_start, 0, __bss_end__ - __bss_start);

    /* Set global data pointer. */
    gd = &gdata;

    board_init_r(NULL, 0);
#endif

    /*
    // test
    #define GPH0CON (*(volatile unsigned int *)0xE0200C00)
    #define GPH0DAT (*(volatile unsigned int *)0xE0200C04)

    GPH0CON = (1<<0) | (1<<4) | (1<<8) | (1<<12);
    GPH0DAT = 10;
    */

    uart_init();

    copy_uboot_to_ram();

    //printf ("jump to u-boot image\r\n");
    /* Jump to U-Boot image */
    uboot = (void *)CONFIG_SYS_TEXT_BASE;
    (*uboot)();
}
```

7.在 arch/arm/lib/spl.c 中添加串口的初始化和从 sd 卡拷贝 uboot 到 ddr2 内存里的函数

```
//UART
#define ULCON0      (*(volatile unsigned int *)0xE2900000)
#define UCON0       (*(volatile unsigned int *)0xE2900004)
#define UTRSTAT0    (*(volatile unsigned int *)0xE2900010)
#define UTXH0       (*(volatile unsigned char *)0xE2900020)
#define URXH0       (*(volatile unsigned char *)0xE2900024)
#define UBRDIV0     (*(volatile unsigned int *)0xE2900028)
#define UDIVSLOT0   (*(volatile unsigned int *)0xE290002C)
//GPA0
#define GPA0CON     (*(volatile unsigned int *)0xE0200000)

typedef u32(*copy_sd_mmc_to_mem)
(u32 channel, u32 start_block, u16 block_size, u32 *trg, u32 init);


/* Pointer to as well as the global data structure for SPL */
DECLARE_GLOBAL_DATA_PTR;
gd_t gdata __attribute__ ((section(".data")));

void uart_init (void)
{
    //配置 GPA0_0 - GPA0_1 管脚为串口 0 功能管脚
    GPA0CON = (2<<0) | (2<<4);
```

```
24.
25.     /**
26.     * 传送模式: 0:普通模式
27.     * 校验位: 0 无校验
28.     * 停止位: 1
29.     * 数据位: 8
30.     */
31.     ULCON0 = (0<<6) | (0<<3) | (0<<2) | (3<<0);
32.
33.     /**
34.     * 时钟选则: 0 = PCLK = 66M:
35.     * 发送模式:01 = Interrupt request or polling mode
36.     */
37.     UCON0 = (0<<10) | (1<<2) | (1<<0);
38.
39.     /**
40.     * DIV_VAL = UBRDIVn + (num of 1's in UDIVSLOTn)/16
41.     * DIV_VAL = (PCLK / (bps x 16)) - 1
42.     * DIV_VAL = (66000000 / (115200 * 16)) - 1 = 34.8
43.     * UBRDIV0 = 34
44.     * (num of 1's in UDIVSLOTn)/16 = 0.8
45.     * (num of 1's in UDIVSLOTn) = 12
46.     * 查表得 UDIVSLOT0 = 0xDDDD
47.     */
48.     UBRDIV0 = 34;
49.     UDIVSLOT0 = 0xDDDD;
50.}
51.
52.void copy_uboot_to_ram(void)
53.{
54.     ulong ch;
55.
56.     ch = *(volatile u32 *)(0xD0037488);
57.     copy_sd_mmc_to_mem copy_bl2 =
58.         (copy_sd_mmc_to_mem) (*(u32 *) (0xD0037F98));
59.
60.     u32 ret;
61.
62.     //printf ("boot mmc chanel: %x\r\n", ch);
63.     if (ch == 0xEB000000) {
64.         ret = copy_bl2(0, 49, 1024, CONFIG_SYS_TEXT_BASE, 0);
65.     }
66.
67.     if (ret == 0) {
68.         //printf ("copy error\r\n");
69.         while (1);
70.     }
71.     else
72.         return;
73.}
```

8.arch/arm/include/asm/spl.h +27

```
1. //#include <asm/arch/spl.h>
```

9.分析 spl/Makefile 得知，在 tools/下没有 mk$(BOARD)spl 这个工具，我暂时对这个工具不熟悉，就用自己的吧，我的工具是 mkv210_image，用来制作 BL1 头信息的

```
1. ifdef CONFIG_SAMSUNG
2. $(obj)$(BOARD)-spl.bin: $(obj)u-boot-spl.bin
3.     $(OBJTREE)/tools/mk$(BOARD)spl \
4.         $(obj)u-boot-spl.bin $(obj)$(BOARD)-spl.bin
5. endif
```

10.修改 spl/Makefile，然后拷贝 mkv210_image.c 到 tools/下

```
1. ifdef CONFIG_SAMSUNG
2. $(obj)$(BOARD)-spl.bin: $(obj)u-boot-spl.bin
3.         $(HOSTCC) $(TOPDIR)/tools/mkv210_image.c -o $(TOPDIR)/tools/mkv210_image
4.         $(OBJTREE)/tools/mkv210_image \
5.                 $(obj)u-boot-spl.bin $(obj)$(BOARD)-spl.bin
6. endif
```

11.make，插入 SD 卡，使用下面的命令烧写 SPL 和 uboot.bin 到 SD 卡上

```
1. dd iflag=dsync oflag=dsync if=spl/smdkv210-spl.bin of=/dev/sdb seek=1
```

```
1. dd iflag=dsync oflag=dsync if=u-boot.bin of=/dev/sdb seek=49
```

12.把 SD 卡插到开发板上，上电，就可以在控制台上打印出

```
U-Boot 2012.10 (Dec 07 2012 - 16:00:24) for SMDKC100

CPU:     S5PC110@400MHz
Board:   SMDKv210
DRAM:  1 GiB
WARNING: Caches not enabled
Using default environment

In:    serial
Out:   serial
Err:   serial
Net:   No ethernet found.
Hit any key to stop autoboot:  0
SMDKC100 #
```

这里 UBOOT 有个小 BUG，在执行 spl/Makefile 的时候，include/configs/smdkv210.h 中有下面这种带//注释代码的就不行，他会在 spl/u-boot-spl.lds 上生成出来，导致链接出错，解决方法就是把这种注释全给删了。或改成/* */这种方法

```
1. //#define PHYS_SDRAM_1_SIZE (128 << 20)   /* 0x8000000, 128 MB Bank #1 */
2. #define PHYS_SDRAM_1_SIZE   (0x40000000)    /* 0x8000000, 128 MB Bank
```

s5pv210 uboot-2012-10 移植(四) 之使系统工作在 1000Mhz

在 uboot 原来的代码里，有系统时钟的初始化函数，在 board/samsung/smdkv210/lowlevel_init.S 的 system_clock_init 函数，我大概看了一下，寄存器不一样，而且是汇编写的，所以我就改成用 c 语言来实现，在 BL1 阶段初始化一下，BL2 阶段就不用重新初始化了。

1.arch/arm/lib/spl.c +43 添加

```
1. //SystemClock
2. #define APLL_LOCK     (*(volatile unsigned int *)0xE0100000)
3. #define MPLL_LOCK     (*(volatile unsigned int *)0xE0100008)
4. #define EPLL_LOCK     (*(volatile unsigned int *)0xE0100010)
5. #define VPLL_LOCK     (*(volatile unsigned int *)0xE0100020)
6. #define APLL_CON0     (*(volatile unsigned int *)0xE0100100)
7. #define APLL_CON1     (*(volatile unsigned int *)0xE0100104)
8. #define MPLL_CON      (*(volatile unsigned int *)0xE0100108)
9. #define EPLL_CON0     (*(volatile unsigned int *)0xE0100110)
10.#define EPLL_CON1     (*(volatile unsigned int *)0xE0100114)
11.#define VPLL_CON      (*(volatile unsigned int *)0xE0100120)
12.#define CLK_SRC0      (*(volatile unsigned int *)0xE0100200)
13.#define CLK_DIV0      (*(volatile unsigned int *)0xE0100300)
14.
15.#define SETPLL(mdiv, pdiv, sdiv) ((1<<31)|(mdiv<<16)|(pdiv<<8)|(sdiv<<0))
16.
17.#define APLL_MDIV   250
18.#define APLL_PDIV   6
19.#define APLL_SDIV   1
20.#define APLL_CON0_VAL SETPLL (APLL_MDIV, APLL_PDIV, APLL_SDIV)
21.
22.#define MPLL_MDIV   667
23.#define MPLL_PDIV   12
24.#define MPLL_SDIV   1
25.#define MPLL_CON_VAL SETPLL (MPLL_MDIV, MPLL_PDIV, MPLL_SDIV)
26.
27.#define PCLK_PSYS_RATIO   1
28.#define HCLK_PSYS_RATIO   4
29.#define PCLK_DSYS_RATIO   1
30.#define HCLK_DSYS_RATIO   3
31.#define PCLK_MSYS_RATIO   1
32.#define HCLK_MSYS_RATIO   4
33.#define A2M_RATIO         4
34.#define APLL_RATIO        0
35.#define CLK_DIV0_VAL ( (APLL_RATIO<<0)|\
36.                     (A2M_RATIO<<4)|\
37.                     (HCLK_MSYS_RATIO<<8)|\
38.                     (PCLK_MSYS_RATIO<<12)|\
39.                     (HCLK_DSYS_RATIO<<16)|\
40.                     (PCLK_DSYS_RATIO<<20)|\
```

```
                    (HCLK_PSYS_RATIO<<24)|\
                    (PCLK_PSYS_RATIO<<28) )
```

2. arch/arm/lib/spl.c +135 添加

```
void init_SystemClock (void)
{
    int i;

    //CLK_DIV0 = 0;

    APLL_CON0 = APLL_CON0_VAL;
    MPLL_CON  = MPLL_CON_VAL;

     CLK_SRC0 = 0x1111;

    CLK_DIV0 = CLK_DIV0_VAL;

    //for (i=65535; i>=0; i--);
}
```

3. include/configs/smdkv210.h +178

```
/*#define CONFIG_SYS_PROMPT     "SMDKC100 # "*/
#define CONFIG_SYS_PROMPT       "SMDKV210 # "
```

4. include/configs/smdkv210.h +206

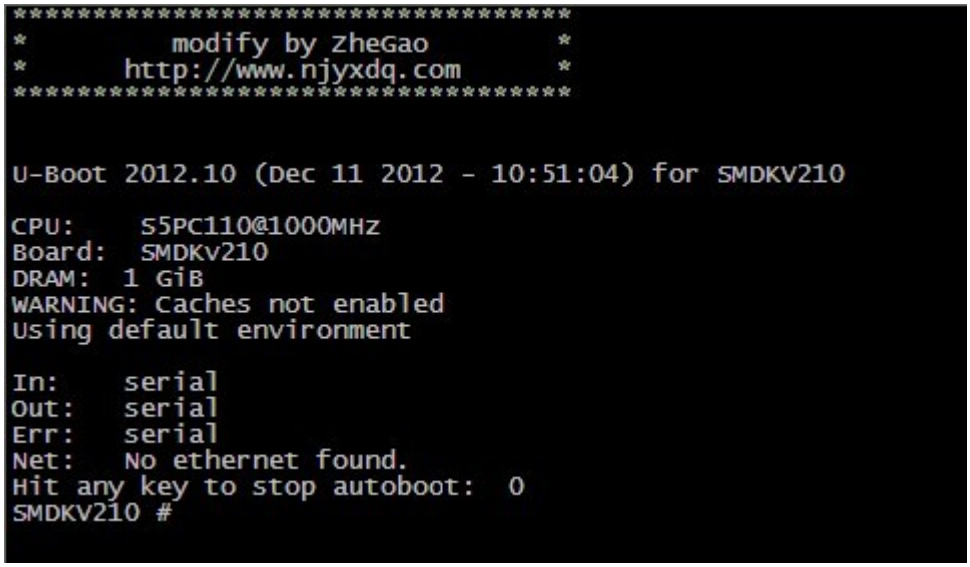
```
/*#define CONFIG_IDENT_STRING       " for SMDKC100"*/
#define CONFIG_IDENT_STRING     " for SMDKV210"
```

5. arch/arm/lib/spl.c +183

```
void __weak board_init_f(ulong dummy)
{
    __attribute__((noreturn)) void (*uboot)(void);
#if 0
    /* Set the stack pointer. */
    asm volatile("mov sp, %0\n" : : "r"(CONFIG_SPL_STACK));

    /* Clear the BSS. */
    memset(__bss_start, 0, __bss_end__ - __bss_start);

    /* Set global data pointer. */
    gd = &gdata;

    board_init_r(NULL, 0);
```

```
15.#endif
16.
17.    /*
18.    // test
19.    #define GPH0CON (*(volatile unsigned int *)0xE0200C00)
20.    #define GPH0DAT (*(volatile unsigned int *)0xE0200C04)
21.
22.    GPH0CON = (1<<0) | (1<<4) | (1<<8) | (1<<12);
23.    GPH0DAT = 10;
24.    */
25.    init_SystemClock();
26.
27.    uart_init();
28.
29.    copy_uboot_to_ram();
30.
31.    //printf ("jump to u-boot image\r\n");
32.    /* Jump to U-Boot image */
33.    uboot = (void *)CONFIG_SYS_TEXT_BASE;
34.    (*uboot)();
35.}
```

6. 好了，make 一下，使用命令烧写 spl/smdkv210-spl.bin 和 u-boot.bin 到 SD 卡里，启动，就会看到下面这个画面了。



s5pv210 uboot-2012-10 移植(五) 之支持 LAN9220 网卡

我的 s5pv210 开发板是 100M 的 LAN9220 网卡芯片，通过 CS5 的总线连接的，对应的地址空间是 0xA8000000，16 位的。

1.跟踪代码发现在 smc9115_pre_init 里配置总线，board/samsung/smdkv210/smdkc100.c +36

```
/*
 * Miscellaneous platform dependent initialisations
 */
static void smc9115_pre_init(void)
{
#define SROM_BW  (*(volatile unsigned int *)0xE8000000)
#define SROM_BC5 (*(volatile unsigned int *)0xE8000018)
#define MP0_1CON (*(volatile unsigned int *)0xE02002E0)

#if 0
    u32 smc_bw_conf, smc_bc_conf;

    struct s5pc100_gpio *const gpio =
        (struct s5pc100_gpio *)samsung_get_base_gpio();

    /* gpio configuration GPK0CON */
    s5p_gpio_cfg_pin(&gpio->k0, CONFIG_ENV_SROM_BANK, GPIO_FUNC(2));

    /* Ethernet needs bus width of 16 bits */
    smc_bw_conf = SMC_DATA16_WIDTH(CONFIG_ENV_SROM_BANK);
    smc_bc_conf = SMC_BC_TACS(0x0) | SMC_BC_TCOS(0x4) | SMC_BC_TACC(0xe)
            | SMC_BC_TCOH(0x1) | SMC_BC_TAH(0x4)
            | SMC_BC_TACP(0x6) | SMC_BC_PMC(0x0);

    /* Select and configure the SROMC bank */
    s5p_config_sromc(CONFIG_ENV_SROM_BANK, smc_bw_conf, smc_bc_conf);
#endif

    unsigned int tmp;

    SROM_BW &= ~((0xf<<20));
    SROM_BW |= ((3<<20));

    tmp = MP0_1CON;
    tmp &= ~((0xf<<20)|(0xf<<12));
    tmp |= ((0x2<<20)|(0x2<<12));
    MP0_1CON = tmp;
}
```

2.跟踪代码，在 board_eth_init 函数里，初始化 LAN9220 网卡，需要传入 CONFIG_SMC911X_BASE 基地址

```
int board_eth_init(bd_t *bis)
{
    int rc = 0;
#ifdef CONFIG_SMC911X
    rc = smc911x_initialize(0, CONFIG_SMC911X_BASE);
    //printf ("rc: %d\n", rc);
#endif
    return rc;
```

```
9. }
```

修改 include/configs/smdkv210.h +241

```
1. /*
2.  * Ethernet Contoller driver
3.  */
4. #ifdef CONFIG_CMD_NET
5. #define CONFIG_SMC911X        1       /* we have a SMC9115 on-board   */
6. #define CONFIG_SMC911X_16_BIT  1       /* SMC911X_16_BIT Mode          */
7. #define CONFIG_SMC911X_BASE    0xA8000000      /* SMC911X Drive Base   */
8. #define CONFIG_ENV_SROM_BANK   3       /* Select SROM Bank-3 for Ethernet*/
9. #endif /* CONFIG_CMD_NET */
```

3.make，把 BL1 和 UBOOT 烧写到 SD 卡里，启动，可以识别网卡芯片，使用 ping 192.168.0.1 会发现不支持 ping 命令

4.搜索代码，发现需要配置 CONFIG_CMD_PING，include/configs/smdkv210.h +246

```
1. #ifdef CONFIG_CMD_NET
2. #define CONFIG_SMC911X        1       /* we have a SMC9115 on-board   */
3. #define CONFIG_SMC911X_16_BIT  1       /* SMC911X_16_BIT Mode          */
4. #define CONFIG_SMC911X_BASE    0xA8000000      /* SMC911X Drive Base   */
5. #define CONFIG_ENV_SROM_BANK   3       /* Select SROM Bank-3 for Ethernet*/
6. #define CONFIG_CMD_PING
7. #endif /* CONFIG_CMD_NET */
```

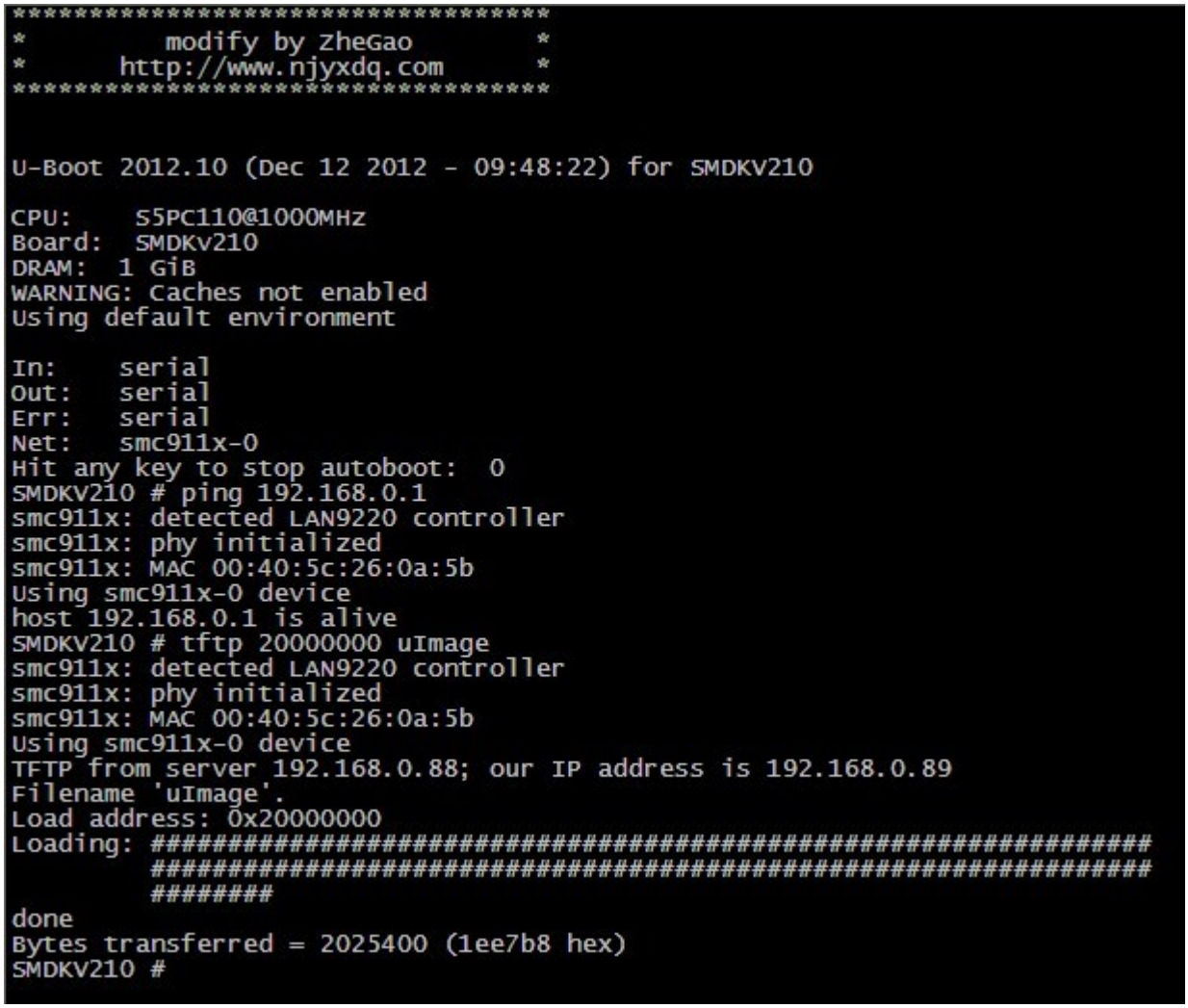
5.make，把 BL1 和 UBOOT 烧写到 SD 卡里，启动，可以识别网卡芯片，使用 ping 192.168.0.1 会发先错误，没有设置网卡物理地址和板子的 ip 地址

6.设置环境变量，ping

7.每次重启后都需要重新配置，很麻烦的， include/configs/smdkv210.h +248

```
1. #define CONFIG_NETMASK      255.255.255.0
2. #define CONFIG_IPADDR       192.168.0.89
3. #define CONFIG_SERVERIP     192.168.0.88
4. #define CONFIG_ETHADDR        00:40:5c:26:0a:5b
5. #define CONFIG_GATEWAYIP       192.168.0.1
```

8.make,烧写到 SD 卡中，上电，ping 可以从控制台里看到，在通过 tftp 命令从服务器端下载 uImage，或者可以 printenv 看看有没有在环境变量里设定。



s5pv210 uboot-2012-10 移植(六) 之支持 NandFlash

我的开发板上外接 256M 的 K9F2G08U0A FLASH 芯片。uboot-2012-10 本身不带 s5pv210 的 nandflash 操作，s5pv210 的 nandflash 和 s3c2410 的寄存器差不多，所以，我就在 s3c2410 的 nandFlash 上进行修改了，因为我不懂 nandflash 的 HW_EEC，所以我实现的是 SOFT_EEC 了。

1.修改 include/configs/smdkv210.h +88

```
/*#undef CONFIG_CMD_NAND*/
#define CONFIG_CMD_NAND
#define CONFIG_SYS_MAX_NAND_DEVICE 1
#define CONFIG_SYS_NAND_BASE 0xB0E00000
#define CONFIG_NAND_S3C2410
```

2.修改 drivers/mtd/nand/s3c2410_nand.c +207，屏蔽了 s3c2410 的 nandflash 初始化，添加了 s5pv210nandflash 总线的初始化和 nandflash 控制器的初始化，并添加 s5pv210_select_chip 函数，其他不变。

```
1.  int board_nand_init(struct nand_chip *nand)
2.  {
3.  #if 0
4.      u_int32_t cfg;
5.      u_int8_t tacls, twrph0, twrph1;
6.      struct s3c24x0_clock_power *clk_power = s3c24x0_get_base_clock_power();
7.      struct s3c2410_nand *nand_reg = s3c2410_get_base_nand();
8.
9.      debug("board_nand_init()\n");
10.
11.     writel(readl(&clk_power->clkcon) | (1 << 4), &clk_power->clkcon);
12.
13.     /* initialize hardware */
14. #if defined(CONFIG_S3C24XX_CUSTOM_NAND_TIMING)
15.     tacls  = CONFIG_S3C24XX_TACLS;
16.     twrph0 = CONFIG_S3C24XX_TWRPH0;
17.     twrph1 =  CONFIG_S3C24XX_TWRPH1;
18. #else
19.     tacls = 4;
20.     twrph0 = 8;
21.     twrph1 = 8;
22. #endif
23.
24.     cfg = S3C2410_NFCONF_EN;
25.     cfg |= S3C2410_NFCONF_TACLS(tacls - 1);
26.     cfg |= S3C2410_NFCONF_TWRPH0(twrph0 - 1);
27.     cfg |= S3C2410_NFCONF_TWRPH1(twrph1 - 1);
28.     writel(cfg, &nand_reg->nfconf);
29. #endif
30.
31.     #define MP0_1CON (*(volatile unsigned int *)0xE02002E0)
32.     #define MP0_3CON (*(volatile unsigned int *)0xE0200320)
33.     #define NFCONF   (*(volatile unsigned int *)0xB0E00000)
34.     #define NFCONT   (*(volatile unsigned int *)0xB0E00004)
35.     #define NFCMMD   (*(volatile unsigned char *)0xB0E00008)
36.     #define NFADDR   (*(volatile unsigned char *)0xB0E0000C)
37.     #define NFDATA   (*(volatile unsigned char *)0xB0E00010)
38.     #define NFSTAT   (*(volatile unsigned int *)0xB0E00028)
39.
40.     #define NFCONF_VAL ((7<<12)|(7<<8)|(7<<4)|(0<<3)|(0<<2)|(1<<1))
41.     #define NFCONT_VAL ((1<<23)|(1<<22)|(1<<2)|(0<<1)|(1<<0))
42.
43.     unsigned int tmp;
44.
```

```
45.     //tmp = MP0_1CON;
46.     //tmp &= ~((0xf<<8)|(0xf<<12));
47.     //tmp |= (0x3<<8) | (0x3<<12);
48.     //MP0_1CON = tmp;
49.     tmp = MP0_1CON;
50.     tmp &= ~(0xf<<16);
51.     tmp |= (0x3<<16);
52.     MP0_1CON = tmp;
53.
54.     tmp = MP0_3CON;
55.     //tmp &= ~((0xf<<0)|(0xf<<4)|(0xf<<8)|(0xf<<12)|(0xf<<16)|(0xf<<20));
56.     //tmp |= (0x2<<0)|(0x2<<4)|(0x2<<8)|(0x2<<12)|(0x2<<16)|(0x2<<20);
57.     tmp &= ~((0xf<<0)|(0xf<<4)|(0xf<<8)|(0xf<<12)|(0xf<<16));
58.     tmp |= (0x2<<0)|(0x2<<4)|(0x2<<8)|(0x2<<12)|(0x2<<16);
59.     MP0_3CON = tmp;
60.
61.     NFCONF = NFCONF_VAL;
62.     NFCONT = NFCONT_VAL;
63.
64.     /* initialize nand_chip data structure */
65.     nand->IO_ADDR_R = (volatile unsigned char *)0xB0E00010;
66.     nand->IO_ADDR_W = (volatile unsigned char *)0xB0E00010;
67.
68.     nand->select_chip = s5pv210_select_chip;
69.
70.     /* read_buf and write_buf are default */
71.     /* read_byte and write_byte are default */
72. #ifdef CONFIG_NAND_SPL
73.     nand->read_buf = nand_read_buf;
74. #endif
75.
76.     /* hwcontrol always must be implemented */
77.     nand->cmd_ctrl = s3c2410_hwcontrol;
78.
79.     nand->dev_ready = s3c2410_dev_ready;
80.
81. #ifdef CONFIG_S3C2410_NAND_HWECC
82.     nand->ecc.hwctl = s3c2410_nand_enable_hwecc;
83.     nand->ecc.calculate = s3c2410_nand_calculate_ecc;
84.     nand->ecc.correct = s3c2410_nand_correct_data;
85.     nand->ecc.mode = NAND_ECC_HW;
86.     nand->ecc.size = CONFIG_SYS_NAND_ECCSIZE;
87.     nand->ecc.bytes = CONFIG_SYS_NAND_ECCBYTES;
88. #else
89.     nand->ecc.mode = NAND_ECC_SOFT;
90. #endif
91.
92. #ifdef CONFIG_S3C2410_NAND_BBT
93.     nand->options = NAND_USE_FLASH_BBT;
94. #else
95.     nand->options = 0;
96. #endif
```

```
97.
98.     debug("end of nand_init\n");
99.
100.    return 0;
101.}
```

3.在 drivers/mtd/nand/s3c2410_nand.c 添加 s5pv210_select_chip 函数

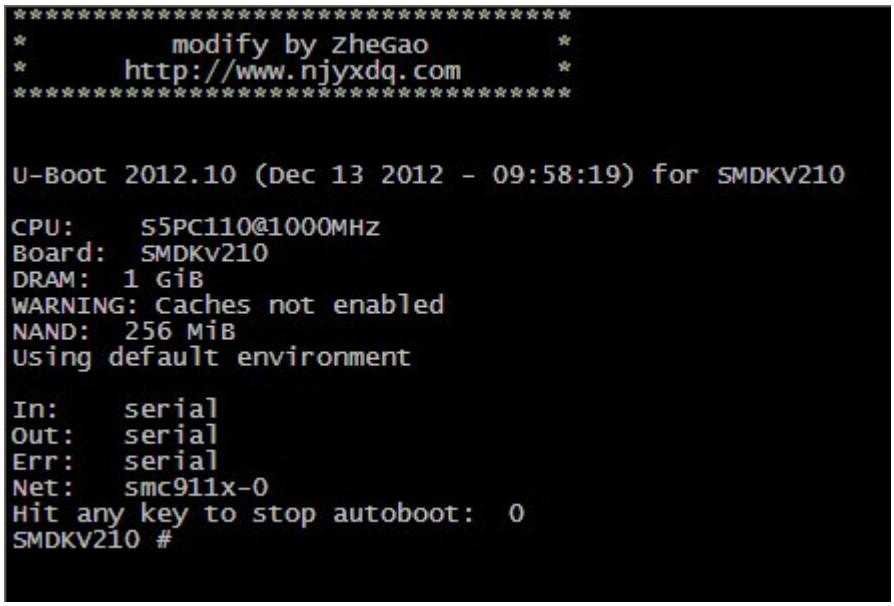
```
1.  static void s5pv210_select_chip(struct mtd_info *mtd, int chipnr)
2.  {
3.      #define NFCONT   (*(volatile unsigned int *)0xB0E00004)
4.
5.      struct nand_chip *chip = mtd->priv;
6.
7.      switch (chipnr) {
8.      case -1:
9.          //chip->cmd_ctrl(mtd, NAND_CMD_NONE, 0 | NAND_CTRL_CHANGE);
10.         NFCONT |= (1<<1);
11.         break;
12.     case 0:
13.         NFCONT &= ~(1<<1);
14.         break;
15.
16.     default:
17.         BUG();
18.     }
19.}
```

4.修改 drivers/mtd/nand/s3c2410_nand.c 的 s3c2410_hwcontrol 函数

```
1.  static void s3c2410_hwcontrol(struct mtd_info *mtd, int cmd, unsigned int ctrl)
2.  {
3.  #if 0
4.      struct nand_chip *chip = mtd->priv;
5.      struct s3c2410_nand *nand = s3c2410_get_base_nand();
6.
7.      debug("hwcontrol(): 0x%02x 0x%02x\n", cmd, ctrl);
8.
9.      if (ctrl & NAND_CTRL_CHANGE) {
10.         ulong IO_ADDR_W = (ulong)nand;
11.
12.         if (!(ctrl & NAND_CLE))
13.             IO_ADDR_W |= S3C2410_ADDR_NCLE;
14.         if (!(ctrl & NAND_ALE))
15.             IO_ADDR_W |= S3C2410_ADDR_NALE;
16.
17.         chip->IO_ADDR_W = (void *)IO_ADDR_W;
18.
19.         if (ctrl & NAND_NCE)
```

```
            writel(readl(&nand->nfconf) & ~S3C2410_NFCONF_nFCE,
                    &nand->nfconf);
        else
            writel(readl(&nand->nfconf) | S3C2410_NFCONF_nFCE,
                    &nand->nfconf);
    }

    if (cmd != NAND_CMD_NONE)
        writeb(cmd, chip->IO_ADDR_W);
#endif

#define NFCMMD   (*(volatile unsigned char *)0xB0E00008)
#define NFADDR   (*(volatile unsigned char *)0xB0E0000C)

    if (cmd != NAND_CMD_NONE) {
        if (ctrl & NAND_CLE) {
            NFCMMD = cmd;
            //printf ("cmd: %x\n", cmd);
            //writeb(cmd, NFCMMD);
        }
        else if (ctrl & NAND_ALE) {
            //writeb(cmd, NFADDR);
            NFADDR = cmd;
            //printf ("addr: %x\n", cmd);
        }
    }

}
```

5.make，使用命令下载到 sd 卡里，上电，观察控制台，可以看到，识别出了 256M 的 nandflash 了，以后就可以对 nandflash 进行操作了。



s5pv210 uboot-2012-10 移植(七) 之支持 SD 卡

我其实对 SD 卡的操作不是很熟悉，所以移植的肯定有问题，在随机赠送的 Kingston 4G 的 SD 卡上可以进行读写，但是我换了张 2G 的 SD 卡，就不行了，原因暂时先不找了，等熟悉了 SD 卡的操作再看看，哪位大侠知道怎么移植的一定要告诉我，不胜感激。

1.include/configs/smdkv210.h +94

```
#define CONFIG_GENERIC_MMC
#define CONFIG_MMC
#define CONFIG_SDHCI
#define CONFIG_S5P_SDHCI
#define CONFIG_CMD_MMC
```

2.在 board/samsung/smdkv210/smdkc100.c 添加

```
int board_mmc_init (bd_t *bis)
{
    //printf ("haha\n");
    #define GPG0CON  (*(volatile unsigned int *)0xE02001A0)
    #define GPG0DRV  (*(volatile unsigned int *)0xE02001AC)
    #define GPG3CON  (*(volatile unsigned int *)0xE0200200)
    #define CLK_DIV4 (*(volatile unsigned int *)0xE0100310)
    #define CLK_SRC4 (*(volatile unsigned int *)0xE0100210)
    #define MOUTMMC (50000000) /* 50MHz */

    unsigned long clk_src, clk_div, mpll, div;

    //初始化 MMC0 MM3 管脚，4bit 模式
    GPG0CON = (0x2<<0)|(0x2<<4)|(0x2<<8)|(0x2<<12)|(0x2<<16)|(0x2<<20)|(0x2<<24);
    GPG3CON = (0x2<<0)|(0x2<<4)|(0x2<<8)|(0x2<<12)|(0x2<<16)|(0x2<<20)|(0x2<<24);
    //GPG0DRV = 0x3fdf;

    //初始化 SD 时钟
    clk_src = CLK_SRC4;
    clk_src &= ~((0xf << 12) | 0xf);
    clk_src |= (0x6 << 12) | 0x6;

        mpll = get_pll_clk(MPLL);
    div = ((mpll + MOUTMMC) / MOUTMMC) - 1;

    clk_div = CLK_DIV4;
    clk_div &= ~((0xf << 12) | 0xf);
    clk_div |= (div << 12) | div;

    CLK_SRC4 = clk_src;
    CLK_DIV4 = clk_div;
```

```
32.
33.     return s5p_sdhci_init(0xEB000000, 0, 4)|s5p_sdhci_init(0xEB300000, 1, 4);
34.}
```

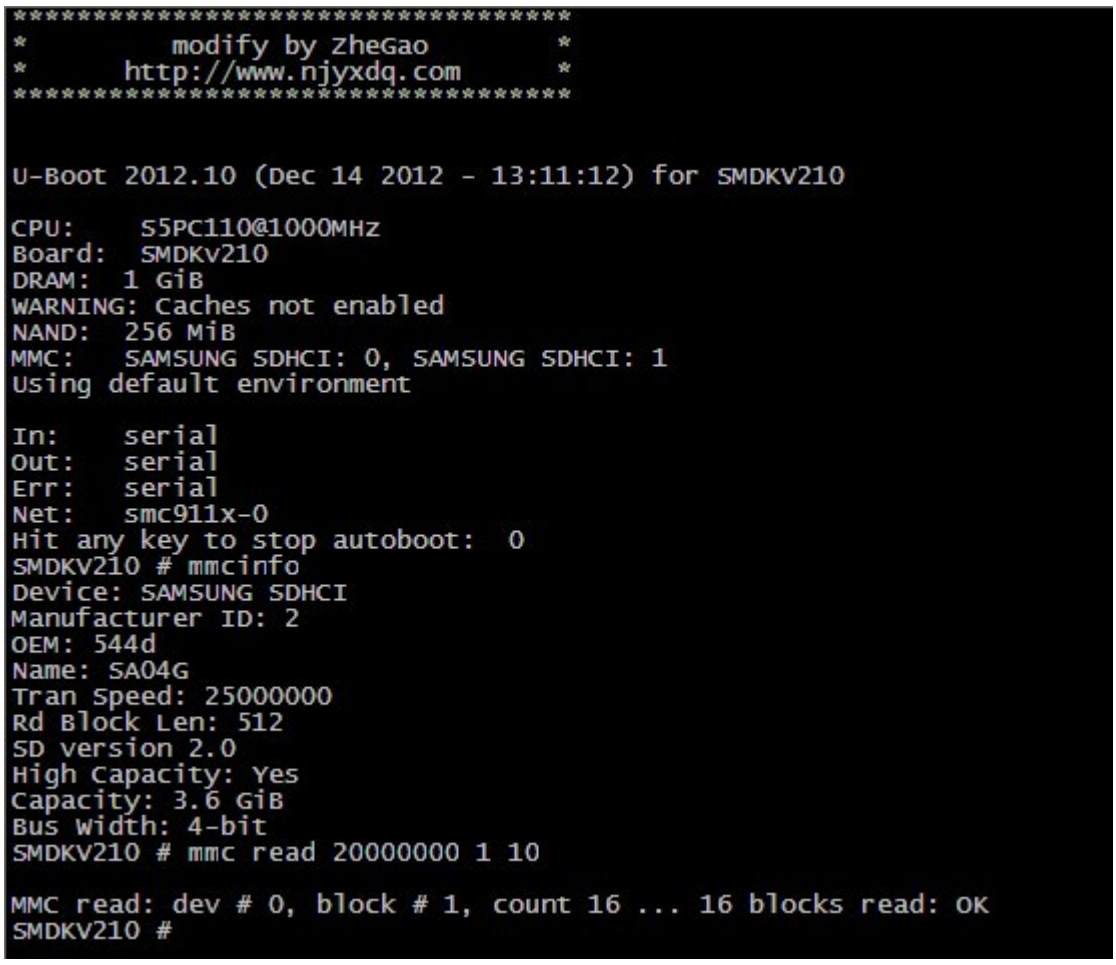
3.修改 drivers/mmc/sdhci.c +133

```
1.  /* Wait max 10 ms */
2.     timeout = 10;
3.  #if 0
4.     sdhci_writel(host, SDHCI_INT_ALL_MASK, SDHCI_INT_STATUS);
5.     mask = SDHCI_CMD_INHIBIT | SDHCI_DATA_INHIBIT;
6.
7.     /* We shouldn't wait for data inihibit for stop commands, even
8.        though they might use busy signaling */
9.     if (cmd->cmdidx == MMC_CMD_STOP_TRANSMISSION)
10.         mask &= ~SDHCI_DATA_INHIBIT;
11.#endif
12.
13.     /*
14.      * PRNSTS
15.      * CMDINHDAT[1] : Command Inhibit (DAT)
16.      * CMDINHCMD[0] : Command Inhibit (CMD)
17.      */
18.     mask = (1 << 0);
19.     if ((data != NULL) || (cmd->resp_type & MMC_RSP_BUSY))
20.         mask |= (1 << 1);
21.
22.     /*
23.      * We shouldn't wait for data inihibit for stop commands, even
24.      * though they might use busy signaling
25.      */
26.     if (data)
27.         mask &= ~(1 << 1);
28.
29.     //mask = 1;
30.     //printf ("mask: %08x\n  read: %08x\n", mask, sdhci_readl(host, SDHCI_PRESENT_STATE));
31.     while (sdhci_readl(host, SDHCI_PRESENT_STATE) & mask) {
32.         if (timeout == 0) {
33.             printf("Controller never released inhibit bit(s).\n");
34.             return COMM_ERR;
35.         }
36.         timeout--;
37.         udelay(1000);
38.     }
```

4.修改 drivers/mmc/mmc.c 把 sd_change_freq 中的最后两个 returne err; 给屏蔽掉

```
1.  //if (err)
2.         //return err;
```

5.make，使用命令烧入 SD 卡里，启动开发板



s5pv210 uboot-2012-10 移植(八) 之支持 SD 卡保存环境变量

本次将实现 SD 卡保存环境变量，uboot 的移植先到这里告一段落了，大体上能用了，但是还很不完善，等到以后需要用到的时候在继续添加，而且中间有移植错误的地方也希望各位大侠给指出来，特别是 SD 卡那里。

1.include/configs/smdkv210.h +233，CONFIG_ENV_OFFSET 自己可以算下，保证不重复就行，我这里是为了保险起见

```
1. /*-----------------------------------------------------------------------
2. * Boot configuration
3. */
4. /*#define CONFIG_ENV_IS_IN_ONENAND  1*/
5. /*#define CONFIG_ENV_IS_NOWHERE*/
6. #define CONFIG_ENV_IS_IN_MMC
7. #define CONFIG_SYS_MMC_ENV_DEV     0
```

```
#define CONFIG_ENV_SIZE     0x4000  /* 16KB */
#define RESERVE_BLOCK_SIZE          (512)
#define BL1_SIZE                    (8 << 10) /*8 K reserved for BL1*/
#define CONFIG_ENV_OFFSET           (RESERVE_BLOCK_SIZE + BL1_SIZE + ((1024 + 512) * 1024
))

/*#define CONFIG_ENV_SIZE         (128 << 10)*/ /* 128KiB, 0x20000 */
/*#define CONFIG_ENV_ADDR         (256 << 10)*/ /* 256KiB, 0x40000 */
/*#define CONFIG_ENV_OFFSET     (256 << 10)*/ /* 256KiB, 0x40000 */
```

2.好了，make 一下，烧写到 SD 卡里，上电

```
**************************************
*          modify by zheGao          *
*       http://www.njyxdq.com        *
**************************************


U-Boot 2012.10 (Dec 15 2012 - 10:01:45) for SMDKV210

CPU:     S5PC110@1000MHz
Board:   SMDKv210
DRAM:  1 GiB
WARNING: Caches not enabled
NAND:  256 MiB
MMC:   SAMSUNG SDHCI: 0, SAMSUNG SDHCI: 1
*** Warning - bad CRC, using default environment

In:    serial
Out:   serial
Err:   serial
Net:   smc911x-0
SMDKV210 #
```

出现*** Warning - bad CRC, using default environment，

```
**************************************
*          modify by zheGao          *
*       http://www.njyxdq.com        *
**************************************


U-Boot 2012.10 (Dec 15 2012 - 10:01:45) for SMDKV210

CPU:     S5PC110@1000MHz
Board:   SMDKv210
DRAM:  1 GiB
WARNING: Caches not enabled
NAND:  256 MiB
MMC:   SAMSUNG SDHCI: 0, SAMSUNG SDHCI: 1
*** Warning - bad CRC, using default environment

In:    serial
Out:   serial
Err:   serial
Net:   smc911x-0
SMDKV210 # saveenv
Saving Environment to MMC...
Writing to MMC(0)... done
SMDKV210 #
```

修改环境变量，保存，重新开机，然后看看有没有保存。