Linux3.0.8平台搭建移植文档——内核移植

一、环境平台介绍

1、内核kernel环境：linux-3.0.8

2、根文件系统工具：busybox-1.16.1

3、yaffs文件系统源码包

4、交叉编译器 arm-linux-4.3.2

二、修改编译内核

1. 解压linux-3.0.8.tar.bz2源码包

```
#tar jxvf linux-3.0.8.tar.bz2
```

2. 修改内核根目录下Makefile文件

```
#vi Makefile
```

修改以下两条信息：

ARCH ? =arm

CROSS_COMPILE ? =/usr/local/arm/4.3.2/bin/arm-linux-（编译的时候会提示某些工具找不到，需要自己做一个链接）

3. 修改链接地址和运行地址

内核自带的链接和运行地址是0x20008000和0x20000100

我们的板子内存地址实际为:0x30000000---0x50000000

所以需要将链接地址改为：0x30008000和0x0x30000100

```
vi arch/arm/mach-s5pv210/Makefile.boot
```

修改内核镜像运行地址为：

```
zreladdr-y      := 0x30008000
params_phys-y    := 0x30000100
```

4. 修改机器类型ID号(mach-type)

bootloader启动内核时，会传递机器类型ID,已启动相应的板级支持代码，我们使用的板子是：SMDKV210在内核源码的“arch/arm/tools/mach-types”文件查找SMDKV210

```
smdkv210      MACH_SMDKV210      SMDKV210          2456
```

在u-boot中输入：bdinfo 可以查看bootloader传递给内核的参数

arch_number = 0x00000998 转为十进制，刚好为：2456 所以这里我们无需修改

5. 修改默认配置文件

```
#cp arch/arm/configs/s5pv210_defconfig .config 或者 make s5pv210_defconfig
#make menuconfig
```

1）选中 physical to virtual 补丁

[*] Patch physical to virtual translations at runtime (EXPERIMENTAL)

2）修改串口支持

System Type --->

(0)S3C UART to use for low-level messages

//内核默认的配置是串口1，我们改为串口0（ctrl+Backspace删除字符）

3）修改板级支持

System Type --->

S5PC110 Machines---> //将与S5PC110有关的东西全部去掉

[]Aqulia

[]GONI

[]SMDKC110

S5PV210 Machines---> //仅保留跟SMDKV210

[*]SMDKV210

[]Torbreck

6. 修改支持NAND FLASH及MTD分区

官方提供的板子在s5pv210芯片后之后采用了三星公司最新推出的onenand内核中提供的是one nand的驱动，但是因为成本及推广的原因，绝大部分厂商还在使用原来的NAND ，内核中没有nand驱动，需要我们自己添加，步骤如下：

1）添加s3c_nand驱动文件：

Step1：复制《基础代码》中s3c_nand.c到linux-3.0.8/drivers/mtd/nand/

复制《基础代码》中regs-gpio.h到arch/arm/mach-s5pv210/include/mach/替换现有文件，以支持大部分GPIO端口的宏定义，便于后期驱动程序使用

Step2：修改指定文件夹下的makefile（指定编译s3c_nand.c）文件

#vi linux-3.0.8/drivers/mtd/nand/Makefile //增加s3c_nand的支持

```
.....
obj-$(CONFIG_MTD_NAND_S3C2410)  += s3c2410.o
obj-$(CONFIG_MTD_NAND_S3C)      += s3c_nand.o
.....
```

Step3：修改Kconfig文件，支持nand的配置和编译

注：所有添加到Kconfig的代码中help条目下的帮助信息都是在一行

#vi linux-3.0.8/drivers/mtd/nand/Kconfig

在以下选项的下面增加配置信息

```
config MTD_NAND_S3C2410_CLKSTOP
...
help
...
```

配置信息：

```
config MTD_NAND_S3C
    tristate "NAND Flash support for S3C SoC"
```

```
    depends on MTD_NAND && (ARCH_S5PC1XX || ARCH_S5PC11X || ARCH_S5PV2XX ||ARCH_S5PV210)
    help
      This enables the NAND flash controller on the S3C.No board specfic support
is done by this driver, each board  must advertise a platform_device for the driver
to attach.

config MTD_NAND_S3C_DEBUG
    bool "S3C NAND driver debug"
    depends on MTD_NAND_S3C
    help
      Enable debugging of the S3C NAND driver

config MTD_NAND_S3C_HWECC
    bool "S3C NAND Hardware ECC"
    depends on MTD_NAND_S3C
    help
      Enable the use of the S3C's internal ECC generator when using NAND. Early
versions of the chip have had problems with incorrect ECC generation, and if using
these,the default of software ECC is preferable.If you lay down a device with the
hardware ECC, then you will currently not be able to switch to software, as there
is no implementation for ECC method used by the S3C.
```

#vi linux-3.0.8/drivers/mtd/Kconfig

我们可以在以下选项的下面增加配置信息

```
config MTD_TESTS
    ......
help
......
```

配置信息：

```
config MTD_PARTITIONS
    bool "MTD partitioning support"
    help
      If you have a device which needs to divide its flash chip(s) up into multiple
'partitions', each of which appears to the user as a separate MTD device, you require
this option to be enabled. If unsure, say 'Y'.Note, however, that you don't need
this option for the DiskOnChip devices. Partitioning on NFTL 'devices' is a different
- that's the 'normal' form of partitioning used on a block device.
```

2）修改配置选项，增加对nand flash及mtd分区的支持

```
make menuconfig
Device Drivers---->
<*>Memory Technology Device(MTD)    support--->
```

```
        [*]   MTD partitioning support
        <*>   Direct char device access to MTD devices
        -*-   Common interface to block layer for MTD 'translation layers'
        <*>   Caching block device access to MTD devices
       <*>NAND Device Support-->
                  <*>NAND Flash support for S3C SoC
                  [ ]S3C NAND driver debug
                  [*]S3C NAND Hardware ECC
```

3）添加对S3c_nand支持相关的宏定义和结构体

#vi arch/arm/plat-samsung/include/plat/nand.h

在结束位置添加以下代码：

```
struct s3c_nand_mtd_info {
     uint chip_nr;
     uint mtd_part_nr;
     struct mtd_partition *partition;
};
```

#vi arch/arm/plat-samsung/include/plat/regs-nand.h

在结束部分添加以下宏定义：

```
/* for s3c_nand.c */
#define S5P_NFREG(x) (x)
#define S3C_NFCONF          S5P_NFREG(0x00)
#define S3C_NFCONT          S5P_NFREG(0x04)
#define S3C_NFCMMD          S5P_NFREG(0x08)
#define S3C_NFADDR          S5P_NFREG(0x0c)
#define S3C_NFDATA8          S5P_NFREG(0x10)
#define S3C_NFDATA          S5P_NFREG(0x10)
#define S3C_NFMECCDATA0          S5P_NFREG(0x14)
#define S3C_NFMECCDATA1          S5P_NFREG(0x18)
#define S3C_NFSECCDATA          S5P_NFREG(0x1c)
#define S3C_NFSBLK          S5P_NFREG(0x20)
#define S3C_NFEBLK          S5P_NFREG(0x24)
#define S3C_NFSTAT          S5P_NFREG(0x28)
#define S3C_NFMECCERR0          S5P_NFREG(0x2c)
#define S3C_NFMECCERR1          S5P_NFREG(0x30)
#define S3C_NFMECC0          S5P_NFREG(0x34)
#define S3C_NFMECC1          S5P_NFREG(0x38)
#define S3C_NFSECC          S5P_NFREG(0x3c)
#define S3C_NFMLCBITPT          S5P_NFREG(0x40)
#define S3C_NF8ECCERR0          S5P_NFREG(0x44)
```

```
#define S3C_NF8ECCERR1          S5P_NFREG(0x48)
#define S3C_NF8ECCERR2          S5P_NFREG(0x4C)
#define S3C_NFM8ECC0            S5P_NFREG(0x50)
#define S3C_NFM8ECC1            S5P_NFREG(0x54)
#define S3C_NFM8ECC2            S5P_NFREG(0x58)
#define S3C_NFM8ECC3            S5P_NFREG(0x5C)
#define S3C_NFMLC8BITPT0        S5P_NFREG(0x60)
#define S3C_NFMLC8BITPT1        S5P_NFREG(0x64)

#define S3C_NFCONF_NANDBOOT     (1<<31)
#define S3C_NFCONF_ECCCLKCON    (1<<30)
#define S3C_NFCONF_ECC_MLC      (1<<24)
#define S3C_NFCONF_ECC_1BIT     (0<<23)
#define S3C_NFCONF_ECC_4BIT     (2<<23)
#define S3C_NFCONF_ECC_8BIT     (1<<23)
#define S3C_NFCONF_TACLS(x)     ((x)<<12)
#define S3C_NFCONF_TWRPH0(x)    ((x)<<8)
#define S3C_NFCONF_TWRPH1(x)    ((x)<<4)
#define S3C_NFCONF_ADVFLASH     (1<<3)
#define S3C_NFCONF_PAGESIZE     (1<<2)
#define S3C_NFCONF_ADDRCYCLE    (1<<1)
#define S3C_NFCONF_BUSWIDTH     (1<<0)

#define S3C_NFCONT_ECC_ENC      (1<<18)
#define S3C_NFCONT_LOCKTGHT     (1<<17)
#define S3C_NFCONT_LOCKSOFT     (1<<16)
#define S3C_NFCONT_MECCLOCK     (1<<7)
#define S3C_NFCONT_SECCLOCK     (1<<6)
#define S3C_NFCONT_INITMECC     (1<<5)
#define S3C_NFCONT_INITSECC     (1<<4)
#define S3C_NFCONT_nFCE1        (1<<2)
#define S3C_NFCONT_nFCE0        (1<<1)
#define S3C_NFCONT_INITECC      (S3C_NFCONT_INITSECC | S3C_NFCONT_INITMECC)

#define S3C_NFSTAT_ECCENCDONE     (1<<7)
#define S3C_NFSTAT_ECCDECDONE     (1<<6)
#define S3C_NFSTAT_ILEGL_ACC      (1<<5)
#define S3C_NFSTAT_RnB_CHANGE     (1<<4)
#define S3C_NFSTAT_nFCE1        (1<<3)
#define S3C_NFSTAT_nFCE0        (1<<2)
#define S3C_NFSTAT_Res1           (1<<1)
#define S3C_NFSTAT_READY        (1<<0)
#define S3C_NFSTAT_CLEAR        ((1<<7) |(1<<6) |(1<<5) |(1<<4))
#define S3C_NFSTAT_BUSY           (1<<0)
#define S3C_NFECCERR0_ECCBUSY     (1<<31)
```

4）添加s3c_nand的platform资源

s3c_nand 驱动遵循了 platform 机制，因此我们要为 nand 驱动添加 platform 资源，因此将此添加到 arch/arm/mach-s5pv210/mach-smdkv210.c 文件中。

#vi arch/arm/mach-s5pv210/mach-smdkv210.c

在文件开头宏定义的最后面空白位置添加platform_device结构体：

```
static struct resource s5p_nand_resource[] = {
        [0] = {
                .start          = S5P_PA_NAND,
                .end            = S5P_PA_NAND + S5P_SZ_NAND - 1,
                .flags          = IORESOURCE_MEM,
        }
};

struct platform_device s5pv210_device_nand = {
        .name                   = "s5pv210-nand",
        .id                     = -1,
        .num_resources          = ARRAY_SIZE(s5p_nand_resource),
        .resource               = s5p_nand_resource,
};
```

还要在mach-smdkv210.c这个文件中的mdkv210_devices数组中添加以下内容：

```
Static struct platform_device *smdkv210_devices[] __initdata = {
    &s5pv210_device_nand,    //add by sunplusedu
    &s3c_device_adc,
    &s3c_device_cfcon,
    &s3c_device_fb,
    &s3c_device_hsmmc0,
    &s3c_device_hsmmc1,
```

5）添加nand的寄存器地址定义

#vi arch/arm/mach-s5pv210/include/mach/map.h

在文件的最后添加以下内容(要在 #endif /* __ASM_ARCH_MAP_H */ 定义的前面)：

```
/* NAND Controller */
#define S5PV210_PA_NAND             (0xB0E00000)
#define S5P_PA_NAND             S5PV210_PA_NAND
#define S5PV210_SZ_NAND             SZ_1M
#define S5P_SZ_NAND             S5PV210_SZ_NAND
```

```
#endif /* __ASM_ARCH_MAP_H */
```

6）增加内核中对于nand时钟的支持

修改以下文件：

```
#vi arch/arm/mach-s5pv210/clock.c
static struct clk init_clocks_off[] = {
//add by sunplsuedu start
{
    .name       = "nand",
    .id     = -1,
    .parent         = &clk_hclk_psys.clk,
    .enable         = s5pv210_clk_ip1_ctrl,
    .ctrlbit    = ((1 << 28) | (1 << 24)),
},
//add by sunplsuedu start end
{
    ……
```

7. 支持yaffs文件系统

我们需要在“\\172.20.220.24\软件共享\00_嵌入式课程所需软件\6_平台\linux_resource\2.6.35.7_tools”下拷贝 yaffs 补丁包，并将补丁包与内核源码放在同一级目录下，解压 yaffs2-34292b4.tar.gz 源码包，进入 yaffs2文件夹，给内核打上补丁使内核支持 yaffs2

```
# ./patch-ker.sh  c  m  ../linux-3.0.8
#cd ../linux-3.0.8
#make menuconfig
      File systems --->
         Miscellaneous filesystems --->
            <*>yaffs2 file system support(选中此选项)
```

8. make zImage

将在arch/arm/boot/下生成编译好的可执行程序zImage下载到开发板即可，如果文件系统是完好的，那么启动后系统会进入正常的命令行界面，否则可以通过打印信息查看内核是否成功启动，一般提示“ Freeing init memory: ”表示成功。

9. 内核镜像下载调式步骤（首先进入u-boot命令）

1)下载内核镜像到内存中：

update app zImage

2)启动内核：

bootm 0x40000000

以上调试方法，是为了提高开发效率，将内核下载到内存运行，当内核调试到完全没问题后即可下载到nand flash， 这样下次开机即可启用新的内核。