Linux3.0.8平台搭建移植文档——USB移植

1. USB系统移植

在 linux-3.0.8版本内核的 usb 控制器驱动中只有 ehci-s5p.c，即只有 ehci 控制器的驱动支持，而缺少 ohci 的驱动支持，因此我们需要在 linux-2.6.35.7版本内核中抽取出 ohci-s5pv210.c 驱动程序，主要进行整体结构的重新规划和函数的修改以适应新版本内核（此外还包括了 bsp 中 usb 控制器的初始化等配置函数的修改）从而形成新的驱动文件：ohci-s5p.c。具体修改情况请对比两个版本中的 ohci 相关源代码（现仍遗留一个问题，内核通过 reboot 命令重启后 ohci 驱动无法正常运行，若通过复位键重新上电重启则没有问题，在解决中），具体移植过程如下所示：

1）移植 EHCI 控制器驱动

该部分移植涉及到以下几个文件：

driver/usb/host/ehci-s5p.c //ehci 控制器驱动程序，实现 platform_driver 被 ehci-hcd.c 包含
arch/arm/plat-s5p/dev-ehci.c //ehci 的 platform_device 资源定义，最终要加入到 bsp 中
arch/arm/mach-s5pv210/mach-smdkv210.c //bsp 还有控制器的配置函数

<a>修改 ehci-s5p.c 文件

将以下头文件包含注释掉,才可编译通过：

```
...
#include <linux/platform_device.h>
//#include <mach/regs-pmu.h>
#include <plat/cpu.h>
...
```

修改 ehci 时钟的获取，在 s5p_ehci_probe()函数中 clk_get 函数调用修改如下：

```
...
    s5p_ehci->hcd = hcd;
    s5p_ehci->clk = clk_get(&pdev->dev, "usb-host");
    ...
```

<b>修改 dev-ehci.c 文件

将 s5p_ehci_set_platdata()函数中以下内容注释掉，对于 phy_init 函数我们后面会在 bsp（mach-smdkv210.c）中进行设置：

```
void __init s5p_ehci_set_platdata(struct s5p_ehci_platdata *pd)
{
    struct s5p_ehci_platdata *npd;
    npd = s3c_set_platdata(pd, sizeof(struct s5p_ehci_platdata),
            &s5p_device_ehci);
/*
    if (!npd->phy_init)
        npd->phy_init = s5p_usb_phy_init;
    if (!npd->phy_exit)
        npd->phy_exit = s5p_usb_phy_exit;
*/
}
```

修改 resource（s5p_ehci_resource 变量的内容），如下所示：

```
static struct resource s5p_ehci_resource[] = {
        [0] = {
                .start      = S5P_PA_EHCI,                  //该平台下没有这个宏的定义，需要
我们接下来添加
                .end        = S5P_PA_EHCI + SZ_256 - 1,
                .flags      = IORESOURCE_MEM,
        },
        [1] = {
                .start      = IRQ_UHOST,            //修改其 ehci 的中断号
                .end        = IRQ_UHOST,
                .flags      = IORESOURCE_IRQ,
        }
};
```

ehci 的 resource 资源中 ehci 控制器寄存器物理地址“S5P_PA_EHCI”未定义，需要在 arch/arm/mach-s5pv210/include/mach/map.h 中添加以下内容：

```
#define S5PV210_PA_USB_EHCI      (0xEC200000)
 /*这里我们自行定义了物理地址，也可用内核提供的基地址+偏移量进行替换*/
#define S5P_PA_EHCI             S5PV210_PA_USB_EHCI
#define S5P_SZ_EHCI             SZ_1M
```

修改 arch/arm/plat-s5p/kconfig 文件，在 config PLAT_S5P 选项中添加 dev-ehci 选项，实现其编译，具体如下：

```
config PLAT_S5P
        bool
        depends on (ARCH_S5P64X0 || ARCH_S5PC100 || ARCH_S5PV210 || ARCH_EXYNOS4)
        default y
        ...
        select S5P_DEV_USB_EHCI
        help
          Base platform code for Samsung's S5P series SoC.
```

<c>修改 arch/arm/mach-s5pv210/mach-s5pv210.c 文件

添加 ehci 的 platform 资源到资源列表（smdkv210_devices）中，如下所示：

```
static struct platform_device *smdkv210_devices[] __initdata = {
        ...
        &s5p_device_ehci,
        ...
};
```

添加 usb 的 phy 初始化函数，在空白位置添加，（其主体为 smdkv210_ehci_pdata 结构体）如下：

```
static int usb_host_phy_init(struct platform_device *pdev, int type)
{
        struct clk *otg_clk;
        otg_clk = clk_get(NULL, "otg");
```

```
        clk_enable(otg_clk);
        printk("[valor-lion debug]: in phy init.....\n");
        if (readl(S5P_USB_PHY_CONTROL) & (0x1<<1))
                return -1;
        __raw_writel(__raw_readl(S5P_USB_PHY_CONTROL) | (0x1<<1),
                        S5P_USB_PHY_CONTROL);
        __raw_writel((__raw_readl(S3C_PHYPWR)
                        & ~(0x1<<7) & ~(0x1<<6)) | (0x1<<8) | (0x1<<5),
                        S3C_PHYPWR);
        __raw_writel((__raw_readl(S3C_PHYCLK) & ~(0x1<<7)) | (0x3<<0),
                        S3C_PHYCLK);
        __raw_writel((__raw_readl(S3C_RSTCON)) | (0x1<<4) | (0x1<<3),
                        S3C_RSTCON);
        udelay(15);
        __raw_writel(__raw_readl(S3C_RSTCON) & ~(0x1<<4) & ~(0x1<<3),
                        S3C_RSTCON);
        return 0;
}
static int usb_host_phy_off(struct platform_device *pdev, int type)
{
        __raw_writel(__raw_readl(S3C_PHYPWR) | (0x1<<7)|(0x1<<6),
                        S3C_PHYPWR);
        __raw_writel(__raw_readl(S5P_USB_PHY_CONTROL) & ~(1<<1),
                        S5P_USB_PHY_CONTROL);
        return 0;
}

static struct s5p_ehci_platdata smdkv210_ehci_pdata = {
        .phy_init = usb_host_phy_init,
        .phy_exit = usb_host_phy_off,
};
```

在文件开始的位置添加头文件：

```
#include <linux/clk.h>
#include <plat/regs-usb-hsotg-phy.h>
#include <plat/ehci.h>
```

设置 ehci 的 platdata，在 smdkv210_machine_init 函数中添加以下函数调用：

```
 static void __init smdkv210_machine_init(void)
{
        ...
        s5p_ehci_set_platdata(&smdkv210_ehci_pdata);
        ...
}
```

2）make menuconfig 配置 ehci 驱动选项

执行 make menuconfig 命令进入以下目录，选中以下选项：

```
Device Drivers  --->
        ...
      [*] USB support  --->
```

进入 USB support 选项，选中如下：

```
--- USB support
<*>   Support for Host-side USB
[ ]     USB verbose debug messages
[ ]     USB announce new devices
          *** Miscellaneous USB options ***
[ ]     USB device filesystem (DEPRECATED)
[*]     USB device class-devices (DEPRECATED)
[ ]     Dynamic USB minor allocation
<*>   EHCI HCD (USB 2.0) support
[ ]     Root Hub Transaction Translators
[*]     Improved Transaction Translator scheduling
[*]   S5P EHCI support
< >   OXU210HP HCD support
...
```

3）移植 OHCI 控制器驱动

该部分移植所需要.c 及.h 文件为自行修改或编写，其主要思路参考 ehci，主要涉及文件有以下几个：

```
driver/usb/host/ohci-s5p.c //ehci 控制器驱动程序，实现 platform_driver 被 ohci-hcd.c 包含
arch/arm/plat-s5p/dev-ohci.c  //ohci 的 platform_device 资源定义，最终要加入到 bsp 中
arch/arm/mach-s5pv210/mach-smdkv210.c      //bsp 还有控制器的配置函数
```

<a>添加 ohci 驱动文件

首先我们将修改好的 ohci-s5p.c 文件拷贝到 driver/usb/host/目录下，具体修改内容请参照原版“linux-2.6.35.7/driver/usb/host/ohci-s5pv210.c”文件；同时我们还为此准备了一个 ohci.h 文件，将其添加到 arch/arm/plat-samsung/include/plat/目录下。

在 driver/usb/host/ohci-hcd.c 文件中加入对 ohci-s5p.c 的包含。

在#if defined(CONFIG_ARCH_S3C2410) || defined(CONFIG_ARCH_S3C64XX) ... #endif 的后面添加以下内容：

```
#if defined(CONFIG_ARCH_S5PV210) || defined(CONFIG_ARCH_S5P6450) || \
    defined(CONFIG_ARCH_S5PV310)
#include "ohci-s5p.c"
#define PLATFORM_DRIVER          ohci_hcd_s5pv210_driver
#endif
```

修改 driver/usb/kconfig 文件

在 config USB_ARCH_HAS_OHCI 选项中添加以下内容，使得 S5P 平台支持 OHCI 驱动

```
config USB_ARCH_HAS_OHCI
      boolean
      # ARM:
```

```
        ...
        default y if PLAT_S5P
        ...
    default PCI
```

<b>添加 dev-ohci.c 文件

将 dev-ohci.c 文件添加到 arch/arm/plat-s5p/目录下（该代码内容具体参考的 dev-ehci.c 文件，主要内容是 platform 资源的定义）

ehci 的 resource 资源中 ehci 控制器寄存器物理地“S5P_PA_OHCI”未定义，需要在 arch/arm/mach-s5pv210/include/mach/map.h 中添加以下内容：

```
#define S5PV210_PA_USB_OHCI     (0xEC300000)        /*这里我们自行定义了物理地址，也可用内核提供的基地址+偏移量进行替换*/
#define S5P_PA_OHCI             S5PV210_PA_USB_OHCI
#define S5P_SZ_OHCI             SZ_1M
```

将 ohci 的 platform_device 数据添加到 arch/arm/plat-samsung/include/plat/devs.h 文件中（该文件声明了内核中定义的 platform_device 数据），如下所示：

```
...
extern struct platform_device s5p_device_ehci;
extern struct platform_device s5p_device_ohci;
 ...
```

修改 arch/arm/plat-s5p/kconfig 文件，在 config PLAT_S5P 选项中添加 dev-ohci 选项，实现其编译，具体如下：

```
config PLAT_S5P
        bool
        depends on (ARCH_S5P64X0 || ARCH_S5PC100 || ARCH_S5PV210 || ARCH_EXYNOS4)
        default y
        ...
        select S5P_DEV_USB_OHCI
```

```
        help

          Base platform code for Samsung's S5P series SoC.
```

添加 S5P_DEV_USB_OHCI 的 config 选项，在 config S5P_DEV_USB_EHCI 选项的后面，如下所示：

```
config S5P_DEV_USB_OHCI
    bool
    help
      Compile in platform device definition for USB OHCI
```

最后我们对 arch/arm/plat-s5p/Makefile 文件进行添加，将 dev-ohci.c 编译进内核，加入如下内容：

```
obj-$(CONFIG_S5P_DEV_USB_OHCI)      += dev-ohci.o
```

<c>修改 arch/arm/mach-s5pv210/mach-s5pv210.c 文件

添加 ohci 的 platform 资源到资源列表（smdkv210_devices）中，如下所示：

```
static struct platform_device *smdkv210_devices[] __initdata = {
      ...
      &s5p_device_ohci,
     ...
};
```

添加 usb 的 phy 初始化接口（其主体为 smdkv210_ohci_pdata 结构体，且和 ehci 公用初始化函数）如下：

```
static struct s5p_ehci_platdata smdkv210_ehci_pdata = {
  .phy_init = usb_host_phy_init,
  .phy_exit = usb_host_phy_off,
};

static struct s5p_ohci_platdata smdkv210_ohci_pdata = {
  .phy_init = usb_host_phy_init,
  .phy_exit = usb_host_phy_off,
};
```

设置 ohci 的 platdata，在 mach_smdkv210.c 文件的 smdkv210_machine_init 函数中添加以下函数调用：

```
static void __init smdkv210_machine_init(void)
{
      ...
      s5p_ohci_set_platdata(&smdkv210_ohci_pdata);
```

```
        ...
    }
```

在文件开始位置添加头文件

```
#include <plat/ohci.h>
```

4）make menuconfig 配置 ehci 驱动选项

执行 make menuconfig 命令进入以下目录，选中以下选项：

```
      Device Drivers  --->
      [*] USB support  --->
            ...
            <*>   OHCI HCD support
            ...
```

5)make

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

2.HID驱动支持

执行 make menuconfig 命令进入以下目录，选中以下选项：

```
Device Drivers  --->
     ...
    [*] HID Devices  --->
     ...
```

此后 usb 接口的 HID 设备都得到了支持，例如 usb 鼠标键盘等...，插入 usb 鼠标或键盘时会有提示信息，否则移植失败。

注意：

1、u 盘设备不支持，原因是没有加入 mass storage 设备的驱动

answer：

解决这个问题，需要我们配置内核 usb storage 驱动的支持，具体配置如下(执行 make menuconfig 命令)：

```
    Device Drivers  --->
          [*] USB support  --->
                <*> Support for Host-side USB
                     <*>   USB Mass Storage support
```

```
[*]     USB Mass Storage verbose debug
```

2、u 盘出入后挂载，无法显示中文（内核不支持中文）

answer：

解决这个问题，需要我们配置文件系统选项中语言支持，具体配置如下(执行 make menuconfig 命令)：

```
File systems  --->
        ...
        -*- Native language support  --->
```

进入 Native language support 选项

```
--- Native language support
(iso8859-1) Default NLS Option
<*>   Codepage 437 (United States, Canada)
....
<*>   Simplified Chinese charset (CP936, GB2312)
....
<*>   NLS UTF-8
```