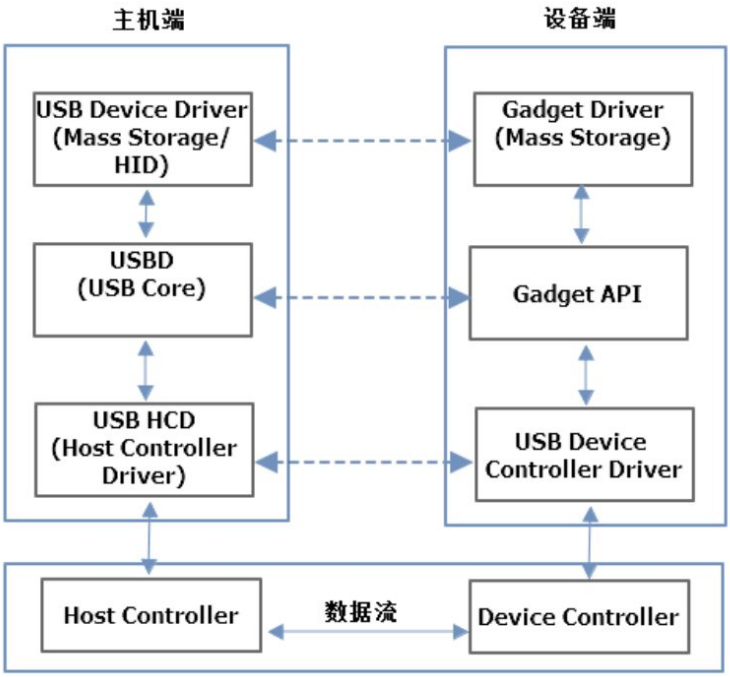
iTOP-4412-驱动-usb 文档 04-主控制器和驱动框架简介

在学习 USB 驱动的过程中，虽然 USB 的内部驱动不需要我们去写，但是还是需要对其有个大概的了解。

1 USB 驱动架构简介

USB 是一种主从结构的系统。主机叫做 Host，从机叫做 Device；开发板作为 USB host 端，USB 鼠标、USB 键盘、USB-WIFI 等等称为设备端；通常，作为 USB device 的设备被称为 Gadget。

如下图所示，是 USB 驱动架构简略图。在主机端（这里需要注意的是，内部驱动和外部驱动都是属于主机端）。



设备端，Gadget API 定义了一个通用的 Gadget Driver 的接口，Gadget Driver 通过 Gadget API 与底层 USB Device Controller Driver 通信。其中 Gadget API 层屏蔽了底层硬件的不同，使 Gadget Driver 注重功能的实现，尽量与硬件无关。设备端的驱动一般是以固件形式在设备端中，由设备端的生产厂商固化在设备端中。

在主机端，有 USB HCD 和 USBD 两个接口层。

USB HCD 的全称为主机控制器驱动(Host Controller Driver)，它是对主机控制器硬件的一个抽象，提供与 USB 系统软件之间的软件接口。

从客户软件的角度看，USBD 控制所有的 USB 设备，因此客户软件对设备的控制和所要发送的数据只要交给 USBD 就可以了。USBD 为客户软件提供命令机制和管道机制。客户软件通过命令机制可以访问所有设备的 0 号端点且与默认管道通信，从而实现对设备的配置和其他一些基本的控制工作。管道机制允许客户和设备实现特定的通信功能。该默认管道描述了一条 USBD 和 USB 设备间通信的逻辑通道。

主机端各层有以下功能：

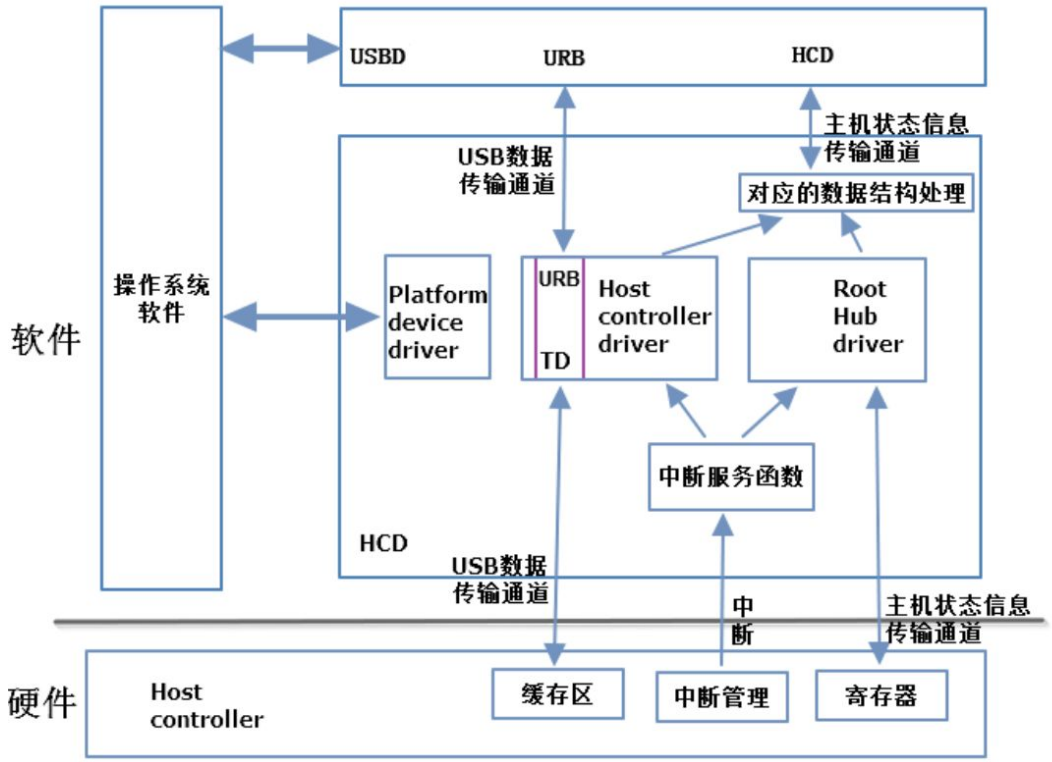
1) 检测连接和移去的 USB 设备；

2) 管理主机和 USB 设备间的数据流；

3) 连接 USB 状态和活动统计；

4) 控制主控制器和 USB 设备间的电气接口，包括能量供应。

如下图所示，是主机端驱动架构，在后面的教程中，我们会详细分析其中的 URB（USB 请求块）和 USB 设备描述符。在 USB 设备通信的整个流程中，USB 描述符用于主机端识别设备端具体是哪个设备，这个过程是由主控制器来完成；USB 请求块用于主机端和设备端的数据传输，提供具体的数据格式定义以及通道。整个驱动架构中的其它部分一般不需要关注。



2 USB 主控制器

本节简单了解一下 4412 的主控制驱动。

2.1 USB 主控制器的功能

USB 主控制器是集成到片上系统的，例如，4412 开发板，主控制器是在 4412 芯片上，代码也是集成在三星原厂提供的内核中的。主控制器主要有一下功能：

1. 解析和维护 URB
2. 负责不同 USB 传输类型的调度工作
3. 负责 USB 数据的实际传输工作
4. 实现虚拟 USB HUB（集线器）的功能

2.2 了解 USB 主控制器驱动

USB 的 USB CORE 在内核源码“drivers/usb/core/”中，如下图所示，可以看到和各种功能对应的内核源码。其中有，USBCORE 核心代码，hub、urb 等等。这些都是具体平台无关的代码，在任意平台中都是通用的核心层代码，给外部驱动提供对应的 API。

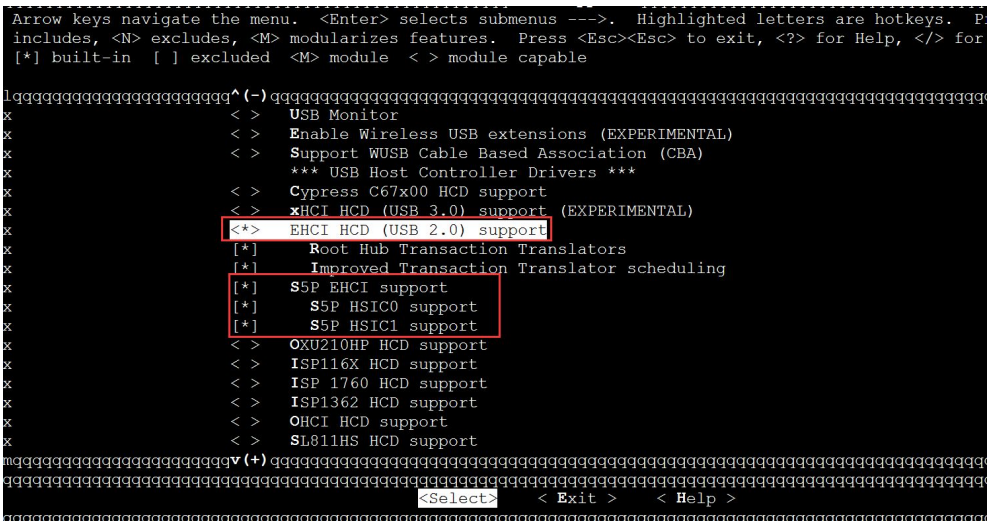
```
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0# ls drivers/usb/core/*.o
drivers/usb/core/buffer.o     drivers/usb/core/driver.o     drivers/usb/core/hub.o       drivers/usb/core/urb.o
drivers/usb/core/built-in.o   drivers/usb/core/endpoint.o   drivers/usb/core/message.o   drivers/usb/core/usbcore.o
drivers/usb/core/config.o     drivers/usb/core/file.o       drivers/usb/core/notify.o    drivers/usb/core/usb.o
drivers/usb/core/devices.o    drivers/usb/core/generic.o    drivers/usb/core/quirks.o
drivers/usb/core/devio.o      drivers/usb/core/hcd.o        drivers/usb/core/sysfs.o
```

另外在内核目录“drivers/usb/serial/”下可以看到前面文档中介绍的 USB 转串口驱动，这个目录里面是 usb 转串口的驱动源码。

USB 主控制在内核源码“drivers/usb/host/”中，如下图所示，可以看到其中只有一个编译生成的“.o”文件。

```
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0# ls drivers/usb/host/*.o
drivers/usb/host/built-in.o  drivers/usb/host/ehci-hcd.o
```

我们在第一篇文档中有介绍到 4412 的主控制器是 USB2.0，使用的是 EHCI 控制器。我们在 menuconfig 中，进入“ Device Drivers”-> “USB support (USB_SUPPORT [=y])”，如下图所示，可以看到“EHCI HCD (USB 2.0) support”默认被配置了。



如上图所示，可以看到“ S5P EHCI support”，这是针对具体平台的配置，如下图所示这个配置定义了宏“CONFIG_USB_EHCI_S5P”。

```
.config - Linux/arm 3.0.15 Kernel Configuration
qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
 lqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq S5P EHCI support qqq
 x CONFIG_USB_EHCI_S5P:
 x
 x Enable support for the S5P SOC's on-chip EHCI controller.
 x
 x Symbol: USB_EHCI_S5P [=y]
 x Type  : boolean
 x Prompt: S5P EHCI support
 x   Defined at drivers/usb/host/Kconfig:197
 x   Depends on: USB_SUPPORT [=y] && USB_EHCI_HCD [=y] && PLAT_S5P [=y]
 x   Location:
 x     -> Device Drivers
 x       -> USB support (USB_SUPPORT [=y])
```

接着使用 source insight 看一下“drivers/usb/host/ehci-hcd.c”的驱动源码，做一下简单了解。

module_init(ehci_hcd_init);入口函数

入口函数 ehci_hcd_init 中，以下代码是注册主控制器驱动的代码。

```
#ifdef PLATFORM_DRIVER
    retval = platform_driver_register(&PLATFORM_DRIVER);
    if (retval < 0)
        goto clean0;
#endif
```

如下图所示，在 menuconfig 中我们可以看到 CONFIG_USB_EHCI_S5P 宏是被定义的，所以 PLATFORM_DRIVER 被定义为 s5p_ehci_driver。

```
#ifdef CONFIG_USB_EHCI_S5P
#include "ehci-s5p.c"
#define PLATFORM_DRIVER         s5p_ehci_driver
#endif
```

接着找一下 s5p_ehci_driver 的定义，在“drivers/usb/host/ehci-s5p.c”文件下。如下图所示，可以看到驱动名称为"s5p-ehci"，USB 的主控制驱动在驱动注册的时候也是使用平台驱动结构体 platform_driver，结构体中也是和字符驱动类似的 move、probe 等等函数。

有驱动注册，那么肯定有设备注册，而且设备名称也是要和驱动名称一样为"s5p-ehci"。

接着我们在平台文件中找一下设备注册。在“arch/arm/mach-exynos/mach-itop4412.c”文件中，搜索宏定义“USB_EHCI_S5P”，如下图所示，可以看到主控制器函数的设备注册代码。

```
#ifdef CONFIG_USB_EHCI_S5P
static struct s5p_ehci_platdata smdk4x12_ehci_pdata;

static void __init smdk4x12_ehci_init(void)
{
        struct s5p_ehci_platdata *pdata = &smdk4x12_ehci_pdata;

        s5p_ehci_set_platdata(pdata);
}
```

如上图所示，smdk4x12_ehci_pdata 结构体变量应该是在调用函数 s5p_ehci_set_platdata(pdata)中初始化的。接着在 source insight 中搜索一下 s5p_ehci_set_platdata 函数，找到了该函数是在“arch/arm/plat-s5p/dev-ehci.c”中定义。如下图所示，该函数中调用了 s5p_device_ehci 结构体来进行初始化，接着搜索一下 s5p_device_ehci 结构体。

```
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;
    if (!npd->phy_suspend)
        npd->phy_suspend = s5p_usb_phy_suspend;
    if (!npd->phy_resume)
        npd->phy_resume = s5p_usb_phy_resume;
}
```

如下图所示，可以看到设备名称注册也是使用的“s5p-ehci”。

```
struct platform_device s5p_device_ehci = {
    .name           = "s5p-ehci",
    .id     = -1,
    .num_resources  = ARRAY_SIZE(s5p_ehci_resource),
    .resource       = s5p_ehci_resource,
    .dev            = {
        .dma_mask = &s5p_device_ehci_dmamask,
        .coherent_dma_mask = 0xffffffffUL
    }
};
```

至此，我们完成分析了主控制的设备注册和驱动注册。具体实现代码更加复杂，但是这部分不需要我们去做，有原厂会提供做好的驱动。

本文档只是让大家对主控制驱动有个感性的认识，在后面文档中的设备描述符、URB（请求块）才是驱动学习的重点。

另外还有具体的 USB 驱动的移植，也比主机驱动和 USB 核心层驱动更重要，希望大家不要花费过多的时间去研究主控制驱动和 USB 核心层代码。而是要在主机驱动和核心层驱动的基础上，移植我们在项目和工程中需要的外围模块。到后面，大家会发现，在移植和使用 USB 外围设备驱动的时候，完全不需要用到 USB 主控制器、USB 驱动框架等等知识。

联系方式

北京迅为电子有限公司致力于嵌入式软硬件设计，是高端开发平台以及移动设备方案提供商；基于多年的技术积累，在工控、仪表、教育、医疗、车载等领域通过 OEM/ODM 方式为客户创造价值。

iTOP-4412 开发板是迅为电子基于三星最新四核处理器 Exynos4412 研制的一款实验开发平台，可以通过该产品评估 Exynos 4412 处理器相关性能，并以此为基础开发出用户需要的特定产品。

本手册主要介绍 iTOP-4412 开发板的使用方法，旨在帮助用户快速掌握该产品的应用特点，通过对开发板进行后续软硬件开发，衍生出符合特定需求的应用系统。

如需平板电脑案支持，请访问迅为平板方案网“http://www.topeet.com”，我司将有能力为您提供全方位的技术服务，保证您产品设计无忧！

本手册将持续更新，并通过多种方式发布给新老用户，希望迅为电子的努力能给您的学习和开发带来帮助。

迅为电子

2018 年 01 月