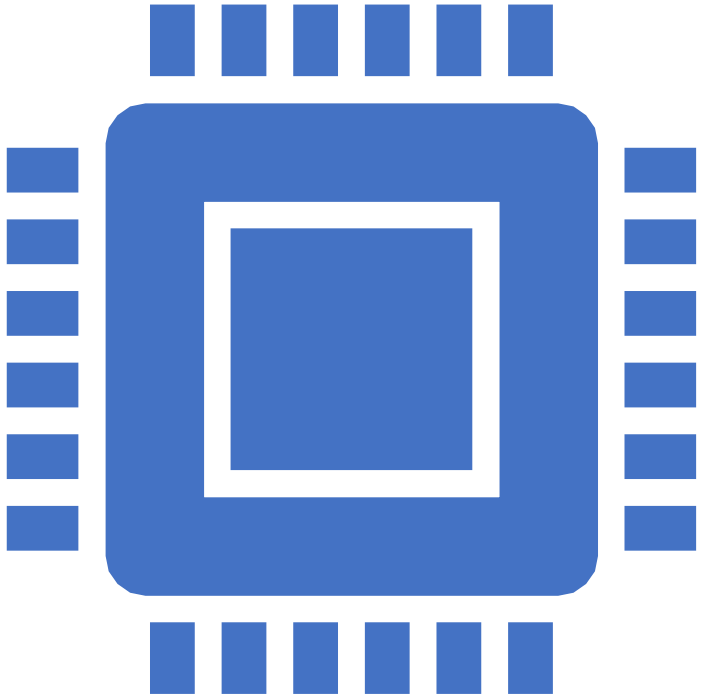


王萌 PLCT实习生 2020.11.11

USB驱动框架及RT-Thread的USB host实现分析

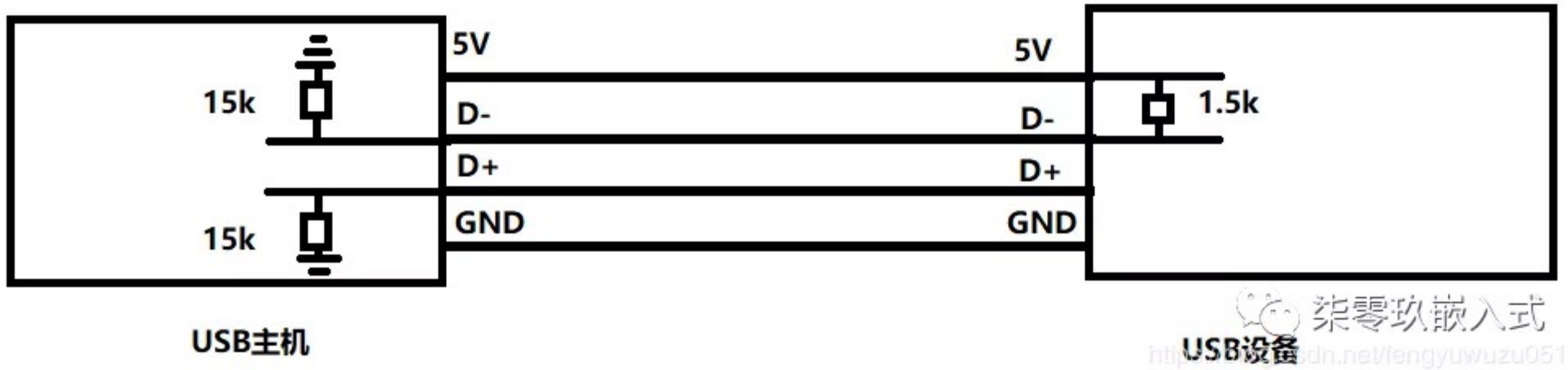
Contents

USB协议简介

- USB(Universal Serial Bus通用串行总线)
- 是一种主从模式的协议
- 分host和device两种工作模式，host和host，device和device之间不能通信
- USB主机发起通信请求，设备进行数据回复，USB设备不具备主动向主机通信的能力。

USB驱动框架

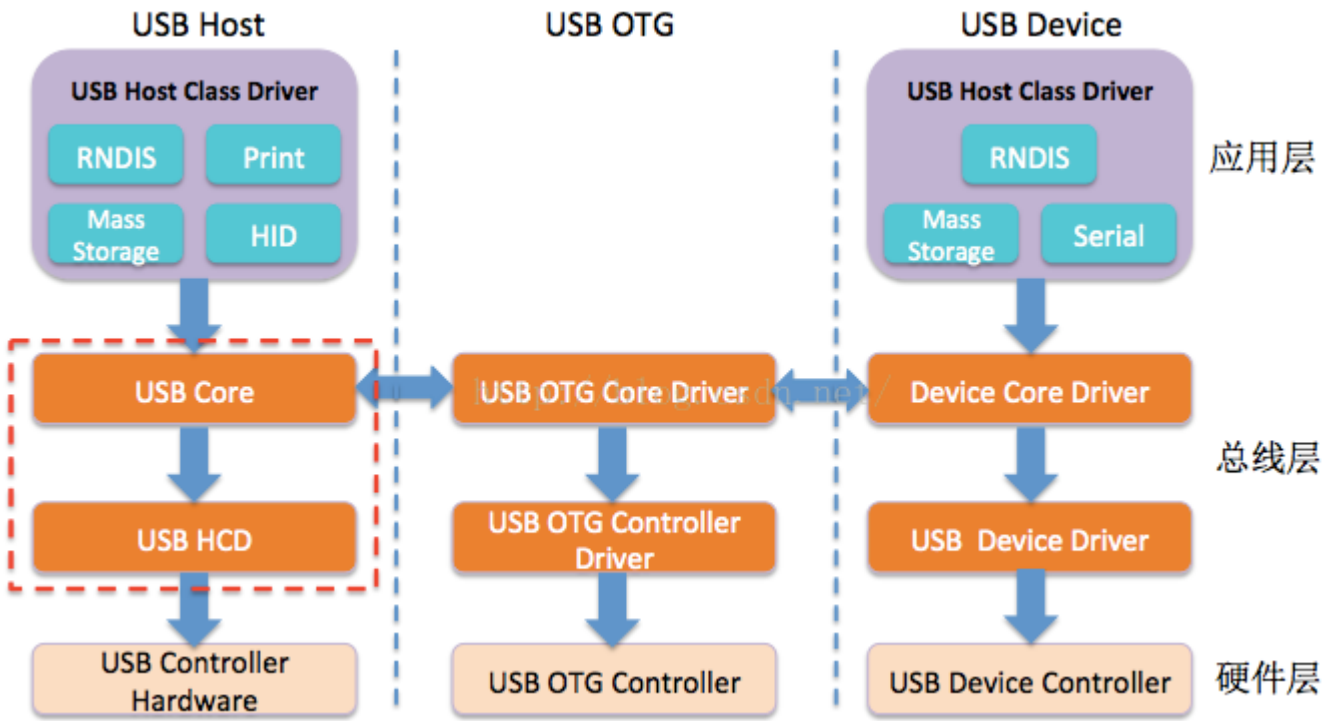
- 物理特性：



- 有一对数据线，采用差分传输模式

USB驱动框架

- 大致的驱动整体框架如下:



USB驱动框架

- 其中，USB HCD即hardware Controller Driver，负责与USB控制器的硬件进行交互
- 有多种不同的接口规范：

- （1）UHCI：Intel提供，通用主机控制接口，USB1.0/1.1；

- （2）OHCI：微软提供，开放主机控制接口，USB1.0/1.1；

- （3）EHCI：增强主机控制接口，USB2.0；

USB驱动框架

- 在USB HCD之上运行的是USB core，这部分是独立于硬件的协议栈

- USB Core为设备驱动程序提供服务，提供一个用于访问和控制USB硬件的接口，而不用考虑系统当前使用的哪种HOST Controller。USB Core将用户的请求映射到相关的HCD，用户不能直接访问HCD。USB Core就是HCD与USB设备的桥梁。
- 在USB Core上方运行的就是各类设备的驱动程序

RT-Thread的USB Host协议栈分析

- rt_usbh_hub_init((uhcd_t)uhc): 初始化根集线器
- 位于drivers/usb/usbhost/core/hub.c
- Root hub运行一个服务和一个消息队列用来管理controller上发生的动作(设备连接/断开/执行callback函数)
- rt_usbh_event_signal()负责向hub发送msg
- rt_usbh_hub_thread_entry()负责接收设备发送的msg并执行相应的操作

RT-Thread的USB Host协议栈分析

- HCD驱动(以STM32f4xx的驱动为例)
- bsp/stm32/libraries/HAL_Drivers/drv_usbh.c
- HCD驱动使用一个uhcd结构体来描述：

```
struct uhcd
{
    struct rt_device parent;
    uhcd_ops_t ops;
    rt_uint8_t num_ports;
    uhub_t roothub;
};
```

RT-Thread的USB Host协议栈分析

- uhcd结构体中的ops成员定义了驱动的具体功能：

```
static struct uhcd_ops _uhcd_ops =
{
    drv_reset_port,
    drv_pipe_xfer,
    drv_open_pipe,
    drv_close_pipe,
};
```

这四个函数就能够完成USB的数据传输

USB的数据传输是单向的，也就是以URB(USB Request Block)请求、URB生成、URB递交、URB释放为主线，这四个函数用于管理信道和数据传输就足够了

RT-Thread的USB Host协议栈分析

- uhcd结构体中的ops函数的实现
- 首先调用drv_usbh.c中的相应函数与rtt的协议栈交互
- 然后调用bsp/stm32/libraries/STM32F4xx_HAL/STM32F4xx_HAL_Driver/Src/stm32f4xx_hal_hcd.c中的对应函数与硬件交互

```
static int drv_pipe_xfer(upipe_t pipe, rt_uint8_t token, void *buffer, int nbytes, int timeouts)
{
    int timeout = timeouts;

    while (1)
    {
        if (!connect_status)
        {
            return -1;
        }
        rt_completion_init(&urb_completion);
        HAL_HCD_HC_SubmitRequest(&stm32_hhcd_fs,
```

RT-Thread的USB Host协议栈分析

- HAL_HCD_xx函数是stm32提供的一组HAL即硬件抽象层，它也不负责和硬件交互，最终会调用bsp/stm32/libraries/STM32F4xx_HAL/STM32F4xx_HAL_Driver/Src/stm32f4xx_ll_usb.c中的函数和硬件交互

```
HAL_StatusTypeDef HAL_HCD_HC_SubmitRequest(HCD_HandleTypeDef *hhcd,
                                           uint8_t ch_num,
                                           uint8_t direction,
                                           uint8_t ep_type,
                                           uint8_t token,
                                           uint8_t *pbuff,
                                           uint16_t length,
                                           uint8_t do_ping)
{
  ...

  return USB_HC_StartXfer(hhcd->Instance, &hhcd->hc[ch_num], (uint8_t)hhcd->Init.dma_enable);
}
```

RT-Thread的USB Host协议栈分析

- 除了驱动部分之外，还需要实现设备探测等功能，需要实现相应的中断处理函数
- bsp/stm32/libraries/HAL_Drivers/drv_usbh.c中绑定了OTG_FS_IRQHandler()，将其绑定到HAL_HCD_IRQHandler函数

```
void OTG_FS_IRQHandler(void)
{
    rt_interrupt_enter();
    HAL_HCD_IRQHandler(&stm32_hhcd_fs);
    rt_interrupt_leave();
}
```