本文档从相对底层的usb\_get\_descriptor函数着手查看代码执行过程。

1 usb\_get\_descriptor(USB设备，类型，指针，数据缓存区地址，大小)，其中只调用了一个函数usb\_control\_msg(USB设备，管道，请求，请求类型，值，指针，数据缓存区地址，大小，超时时间值)。**管道**由设备和端点0计算得出，其中包含了设备号，端点号，最大传输数据大小以及传输方向。**请求**为获取描述符USB\_REQ\_GET\_DESCRIPTOR。**请求类型**为输入USB\_DIR\_IN。值为(type << 8) + index。指针为0。**超时时间值**为100ms。

【拓展请求命令，创建管道】

2 usb\_control\_msg(USB设备，管道，请求，请求类型，值，指针，数据缓存区地址，大小，超时时间值)。此函数下创建了一个名为setup\_packet的命令包，其类型为struct devrequest，将请求类型、请求、设置值、指针与大小打包放入setup\_packet之中。将设备状态设置为未处理后，调用submit\_control\_msg（USB设备，管道，数据缓存区地址，传输长度，命令包）提交控制信息。之后在超时时间内循环查看设备状态，如果设备状态不再是未处理的就结束循环，最后查看跳出循环是否因为超时引起的，超时的话就返回错误代码否则返回接收数据包的长度。

【将命令打包，超时监测】

3 submit\_control\_msg（USB设备，管道，数据缓存区地址，传输长度，命令包）。此函数只调用了\_ohci\_submit\_control\_msg(**开放的主机控制器接口OHCI的地址**,USB设备,管道, 数据缓存区地址，传输长度，命令包)函数，达到控制OHCI底层。

【调用开放的主机控制器接口】

4 \_ohci\_submit\_control\_msg(开放的主机控制器接口OHCI的地址，USB设备，管道，数据缓存区地址，传输长度，命令包)。根据管道中的信息[第一步]解码出该管道的最大传输大小，并检测其是否为0。检测管道中的设备号[第一步]是否是根集线器的设备号。如果是就进行根集线器的重定向：OHCI中保存的根集线器设备设置为当前的USB设备，再调用ohci\_submit\_rh\_msg(OHCI地址，USB设备，管道，数据缓存区地址，传输长度，命令包)。否则调用submit\_common\_msg(OHCI地址，USB设备，管道，数据缓存区地址，传输长度，命令包，间隔)访问其他公共外设，此处间隔为0。

【检验参数最大传输大小，判断管道连接的设备是否为根集线器，OHCI访问设备】

5.1 ohci\_submit\_rh\_msg(OHCI地址，USB设备，管道，数据缓存区地址，传输长度，命令包)。开辟一个名为databuf的数据缓存空间，调用usb\_pipeint对管道进行初始化（将管道设置为中断类型）。从命令包解码出请求类型和请求、值、指针、长度，根据解码出的请求进行相应配置：①如为获取状态则根据需要获取的具体信息（接口信息、端点信息、类信息等）将databuf指针指向相应值或位置，并对长度进行赋值；②如为清除指令，则根据请求类型和设置值（电流过载，端口使能，端口停止，端口供电，端口连接，端口复位等）对相应寄存器ohci->regs->roothub.status赋值，返回数据长度为0无返回数据；③如果为设置指令也是根据设置信息对相关寄存器赋值（端口停止、端口复位、端口供电、端口使能以及地址设置将根集线器设备号设置为wValue），返回数据长度0；④如为获取描述符（设备描述符，配置描述符，字符串描述符等）就将数据缓存区databuf的指针指向本地存储的根集线器描述符，并返会该描述符的长度；⑤如为获取根集线器类描述符就将各种信息依次填入数据缓存区databuf中，计算返回数据的长度；⑥如为获取配置与设置配置同上。进行完上述操作后，计算数据缓存区databuf中有用数据的长度并将其写入USB设备的接收应答长度dev->act\_len中，利用memcpy函数将有用的信息从数据缓存区databuf中拷贝到我们传入的缓存区地址下，最后更新USB设备的状态为start。

【解码命令包，根据命令读取相应信息到缓存区并计算返回数据长度或根据命令信息配置相关寄存器，将缓存区的数据拷贝到我们想要的地址下，更新USB设备的状态dev->status与接收长度dev->act\_len】

5.2 submit\_common\_msg(OHCI地址，USB设备，管道，数据缓存区地址，传输长度，命令包，间隔)。从管道pipe中解码出数据包大小的最大值，确保其大于0。调用ohci\_alloc\_urb函数配置USB请求块urb（USB设备，管道，数据缓存区地址，传输长度，间隔）。用ohci\_get\_ohci\_dev（OHCI地址，USB设备的设备号，管道中断）查找OHCI设备。调用sohci\_submit\_job（OHCI地址，USB设备，管道，数据缓存区地址，传输长度，命令包，USB请求块，间隔）向下位机发送开始传输的命令，用USB\_TIMEOUT\_MS函数提取管道中包含的超时时间值[第一步]，在超时时间内循环用hc\_interrupt函数监测主机控制器中断状态，以此判断传输知否结束，正常结束时传递状态为USB\_ST\_CRC\_ERR，如果超时状态设置为USB\_ST\_CRC\_ERR。最后，更新设备状态dev->status与返回长度dev->act\_len，如果状态异常则清除数据缓存区buffer，释放USB请求块urb。

【配置USB请求块，查找OHCI设备，向下位机发送开始传输的命令，检测中断是否超时，更新USB状态与接收数据的长度，释放请求块urb】

6.1 ohci\_alloc\_urb(USB设备，管道，数据缓存区地址，传输长度，间隔)。创建一个类型为urb\_priv\_t的USB请求块urb，将USB设备、管道、数据缓存区地址，传输长度以及间隔统统打包到USB请求块中。

【将USB设备、管道、数据缓存区地址，传输长度以及间隔统统打包到USB请求块中】

6.2 ohci\_get\_ohci\_dev(OHCI地址，USB设备号，管道中断)，首先根据USB设备号在OHCI的设备列表中查找OHCI之前是否与此设备建立连接(ohci->int\_dev[i].devnum == devnum)，如果有返回对应的OHCI设备地址&ohci->int\_dev[i]。如果没有建立连接就找一个空位置新建设备并返回。否则即找不到设备，又没有空位置返回NULL。

【根据设备号在OHCI设备列表中查找、新建OHCI设备】

6.3 sohci\_submit\_job(OHCI地址，OHCI设备，USB请求块，命令包)。从USB请求块中解码出USB设备，管道，数据缓存区地址，传输长度和间隔。检测OHCI控制器是否挂起，将请求块完成标志urb->finished置为0表示未完成。调用ep\_add\_ed（）函数创建端点ed。根据管道类型设置数据传输区间TD的个数size（批量传输管道和控制传输管道单次传输4096个字节，中断传输一个TD即可）。对端点USB请求块的私有数据purb\_priv结构体赋值。用td\_alloc（）分配数据传输区间TD，如果端点还没有运行，则调用ep\_link（）将端点ed链接到主机控制器HC中，最后td\_submit\_job（）准备传输所有数据！！！

【创建端点，计算数据传输区间TD的个数并进行分配，链接端点，准备开启传输】

6.4 hc\_interrupt(OHCI地址)，检查主机控制器中断触发原因，进行状态检测，判断传输是否顺利结束。确保OHCI与HCCA状态正常，否则返回错误代码。如果是根集线器状态变更，返回0xFF（对通用USB设备来说，这是错误代码之一）。中断发生不可描述的错误时，复位主机控制器HC和总线，返回错误代码-1。调度超限返回错误代码-1。完成了头回写的话读写寄存器regs->intrdisable并调用函数dl\_done\_list（）进行删除已经完成的TD等相关操作。

【监测传输状态】

7.1 ep\_add\_ed(OHCI设备，USB设备，管道，间隔，加载)创建与初始化端点（参数加载为1）。定向创建端点，让端点指针ep定向指向数组ohci\_dev->ed[]中的某个端点，以完成创建。如果端点是新建端点就初始化，创建第一个数据传输区间td[0]。设置新建端点的状态：尾指针，头指针，端点状态，管道类型，信息，中断，加载，OHCI设备当前端点。

【创建端点，并初始化其状态】

7.2 td\_alloc(OHCI设备，USB设备)，循环调用size次函数来分配size个数据传输区间TD。在对应的ohci\_device下的td[]中找一个空节点（未被USB设备占用的td），返回该空节点地址，让purb\_priv->td[i]指针指向它，以完成向设备分配TD的过程。

【查找未被分配的td进行分配】

7.3 ep\_link(OHCI地址，端点)，根据端点类型将端点链接到主机控制器列表中。

【链接端点】

7.4 td\_submit\_job(OHCI地址，USB设备，管道，数据缓存区地址，传输长度，命令包，请求块，间隔)。复位USB切换位（OHCI自己处理数据切换，我们只使用 USB 切换位进行重置即可）。根据管道类型，调用td\_fill（）函数配置每个TD（本质是将数据缓存区buffer一小段一小段得分配给各个TD）。调用ohci\_writel（）函数，向寄存器ohci->regs->cmdstatus写入命令开始数据传输。

【复位数据切换位，配置TD，开启数据传输】

8 td\_fill(OHCI地址，信息，数据缓存区起始地址，长度，USB设备，指针，请求块)。设置端点，接收数据缓存区，信息，当前缓存区指针，缓存区结束地址，下一个TD，尾指针。

【设置TD】

1

**底层**

**核心**

**程序**

**框架**

2

3

4

8

5.1

5.2

6.1

6.2

6.3

6.4

7.1

7.2

7.3

7.4