## 概述

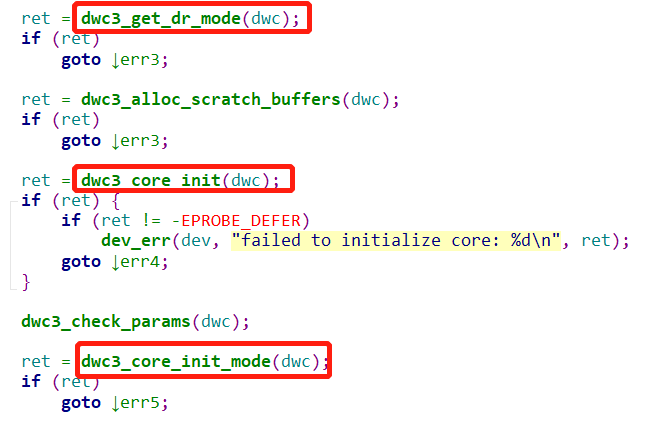
本文从新思的dwc3方案出发，分析USB 主机控制器驱动与根 Hub 的注册过程，以及 USB设备的枚举过程。并不涉及具体USB协议，主要分析驱动框架流程。

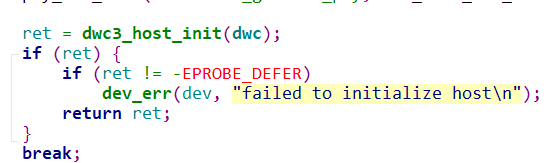
## 第一阶段：添加平台设备即USB主控制器到内核

### 1）添加平台设备到总线

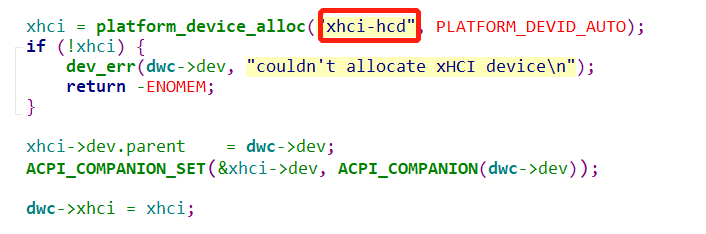
usb控制器是作为一个pci设备工作的，这里从dwc3\_probe()函数展开分析。

1. dwc3\_get\_dr\_mode()获取USB模型，包括host和device，这里只分析主机控制器。
2. dwc3\_core\_init()初始化USB phy。
3. dwc3\_core\_init\_mode()初始化USB模型，该函数调用dwc3\_host\_init函数。

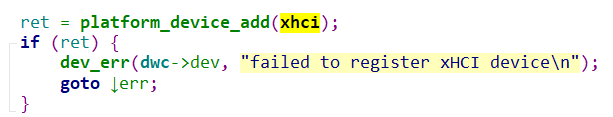


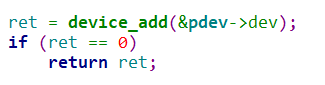
[](file:///C:\Users\zhuming\Documents\youduqt\14581706-102253-zhuming\image\temp\%7b74f3ccb3-3654-4410-8916-79f5f2327d10%7d.png)

接着创建名字为xhci-hcd的平台设备。

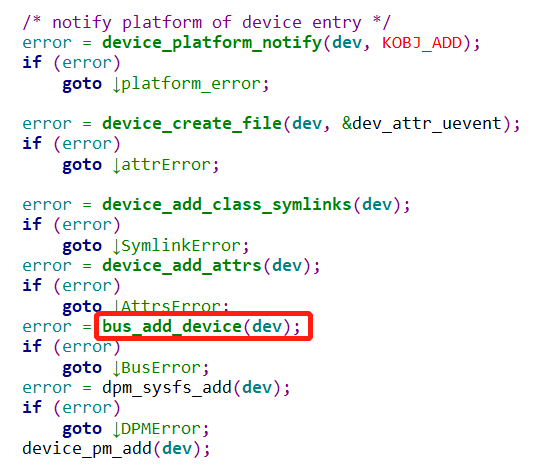


然后通过platform\_device\_add函数，再调用device\_add函数将平台设备添加到内核。

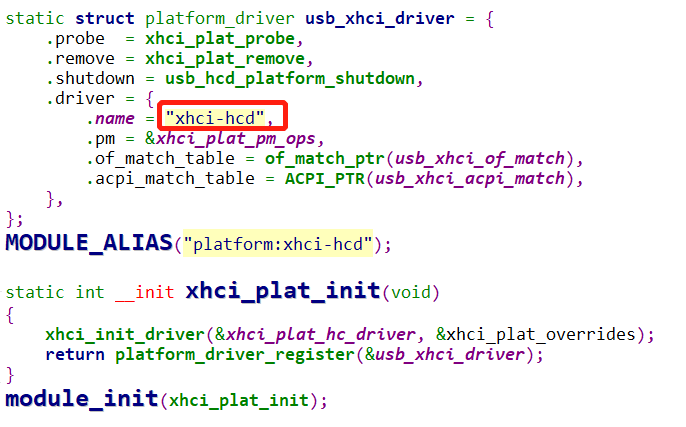
[](file:///C:\Users\zhuming\Documents\youduqt\14581706-102253-zhuming\image\temp\%7b488bb8f7-7eaf-4b06-889a-bd083c0a98ec%7d.png)

[](file:///C:\Users\zhuming\Documents\youduqt\14581706-102253-zhuming\image\temp\%7b0823fa8c-9351-41a5-9363-a2bac8519a2f%7d.png)

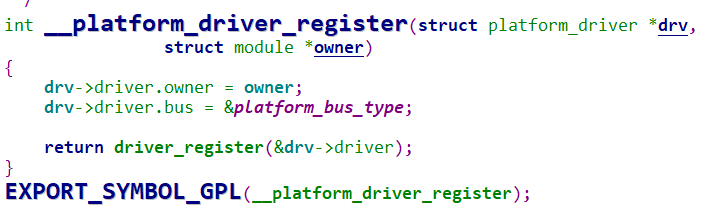
添加到对应总线。



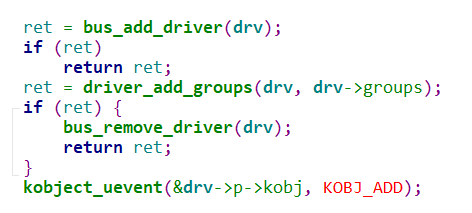
### 2）注册xHCI驱动

xhci\_plat\_init()函数中，会注册名字同样为xhci-hcd的驱动，这样设备就和驱动完成匹配。 

platform\_driver\_register()函数进一步调用driver\_register()

[](file:///C:\Users\zhuming\Documents\youduqt\14581706-102253-zhuming\image\temp\%7b337957ff-2eae-433d-ae2a-b91ee79282a8%7d.png)

bus\_add\_driver()函数将该驱动注册到总线

[](file:///C:\Users\zhuming\Documents\youduqt\14581706-102253-zhuming\image\temp\%7be3e6fb11-7cd8-490f-94a3-4555664d9228%7d.png)

### 3）probe函数

在设备驱动注册最后收尾工作，当设备的device 和其对应的driver 在总线上完成配对之后，系统就调用 [platform](https://link.zhihu.com/?target=http%3A//www.ebhou.com/catalog.asp%3Ftags%3Dplatform)设备的probe函数完成驱动注册最后工作。

经过第1步的bus\_add\_device与第2步的bus\_add\_driver之后，内核驱动模型完成匹配并调用probe函数：

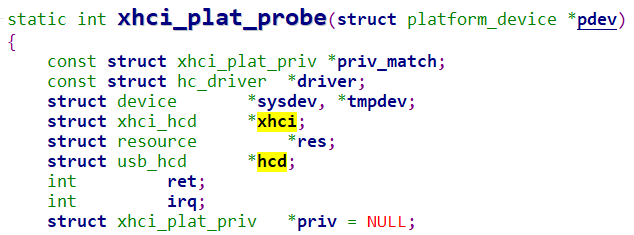
xhci\_plat\_probe()

## 第二阶段：注册root hub，为hub建立设备模型

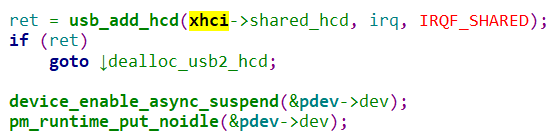
接下来是注册root hub，整个流程会从xhci\_plat\_probe()函数走到add\_device()函数，循环了第一阶段的流程，但是对象不一样了，包括后续USB设备的注册流程，也会走这个循环。

### 1）添加root hub设备到总线

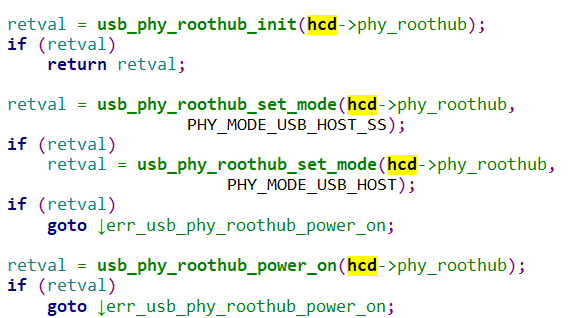
从xhci\_plat\_probe()函数出发

[](file:///C:\Users\zhuming\Documents\youduqt\14581706-102253-zhuming\image\temp\%7b2cddbe99-1d5a-4347-a3c4-c70dc18cb134%7d.png)

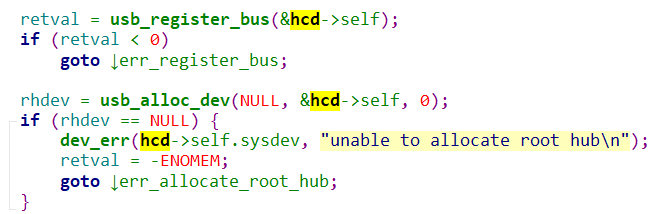
调用usb\_add\_hcd()完成通用主机控制器驱动的初始化与注册

[](file:///C:\Users\zhuming\Documents\youduqt\14581706-102253-zhuming\image\temp\%7bf98ea0bc-7ad7-44d9-a290-82b687186f02%7d.png)

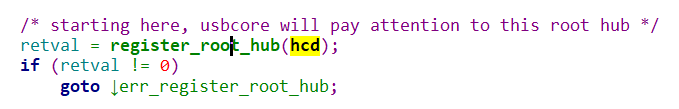
Root hub phy的初始化

[](file:///C:\Users\zhuming\Documents\youduqt\14581706-102253-zhuming\image\temp\%7b725b7aef-15c5-4b3f-b97d-f2f91f05bea8%7d.png)

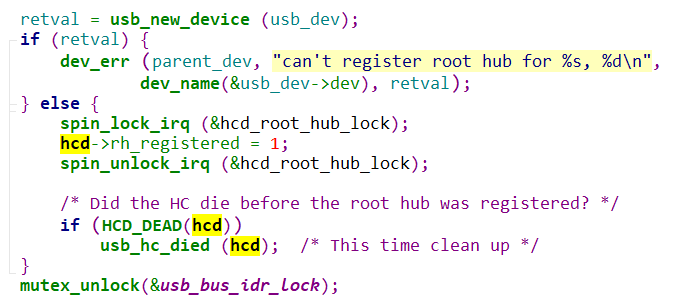
USB core通过usb\_register\_bus()注册USB主机控制器

[](file:///C:\Users\zhuming\Documents\youduqt\14581706-102253-zhuming\image\temp\%7b723493bf-eee1-4a53-8a6f-c942917a3b52%7d.png)

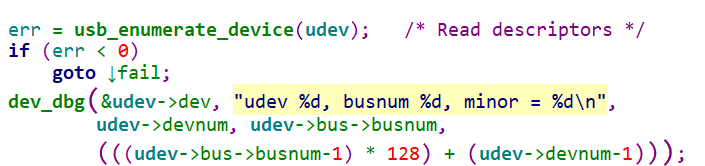
一个主机控制器绑定一个root hub

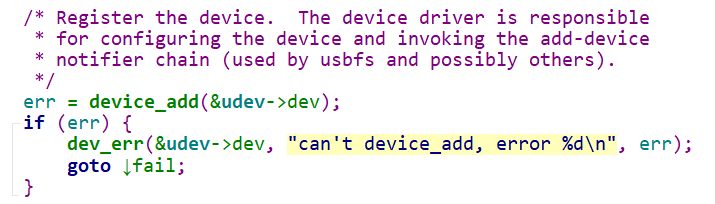
[](file:///C:\Users\zhuming\Documents\youduqt\14581706-102253-zhuming\image\temp\%7b3cc1d31e-2a85-4d3f-9ba6-1c1bb60efa63%7d.png)

register\_root\_hub()函数中调用usb\_new\_device()来注册root hub

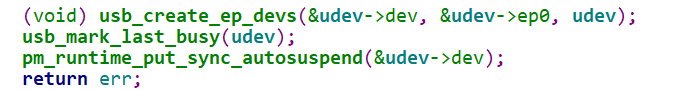
[](file:///C:\Users\zhuming\Documents\youduqt\14581706-102253-zhuming\image\temp\%7b76835e4a-37df-4c6b-82ee-5e523405373f%7d.png)

通过usb\_enumerate\_device()获取该设备的所有描述符，然后通过device\_add()将该设备添加到内核。

[](file:///C:\Users\zhuming\Documents\youduqt\14581706-102253-zhuming\image\temp\%7b5b6b2832-d9e2-49f2-8be9-e488dc45afdb%7d.png)

[](file:///C:\Users\zhuming\Documents\youduqt\14581706-102253-zhuming\image\temp\%7be4c6a74f-527f-48ee-8543-d408ff2511f1%7d.png)

将root hub的端点都注册到内核，并创建设备节点。

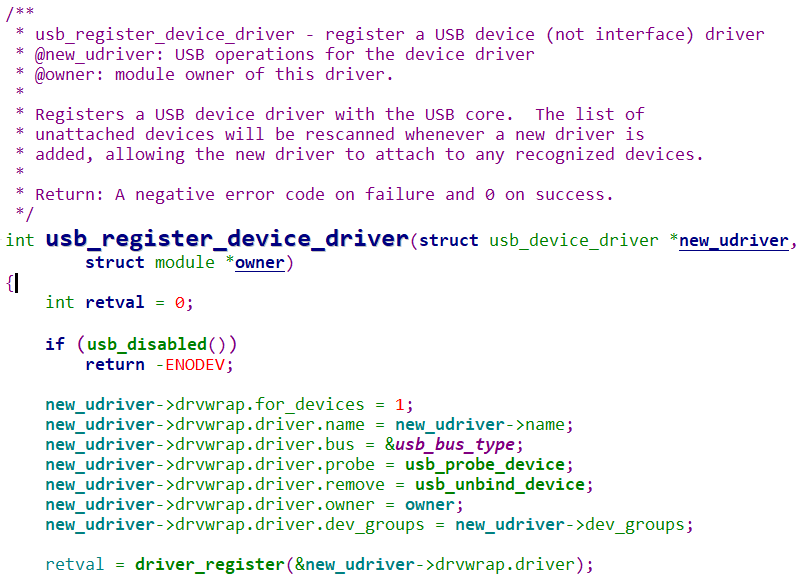
[](file:///C:\Users\zhuming\Documents\youduqt\14581706-102253-zhuming\image\temp\%7be463f2c3-6f49-43e2-bb2d-4de564c0a6ee%7d.png)

### 2）注册通用USB驱动

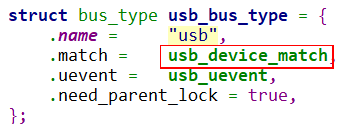
usb\_init()函数是usb子系统的入口，主要完成初始化工作。

[](file:///C:\Users\zhuming\Documents\youduqt\14581706-102253-zhuming\image\temp\%7bd1deaa1a-113f-4cbb-8202-75ed23e777a7%7d.png)

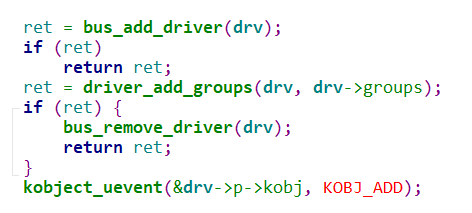
通过usb\_register\_device\_driver()注册通用USB设备驱动

[](file:///C:\Users\zhuming\Documents\youduqt\14581706-102253-zhuming\image\temp\%7beb5ca26e-14ab-4946-aad2-f43a08d4bbfd%7d.png)

该函数里会初始化usb\_bus\_type结构体，定义了一个名字为usb的总线，里面包含通用的usb设备match函数，在后面会有触发。

[](file:///C:\Users\zhuming\Documents\youduqt\14581706-102253-zhuming\image\temp\%7b7c9b73b2-ed47-47d5-a7b8-af107e7008aa%7d.png)

将该驱动注册到总线。

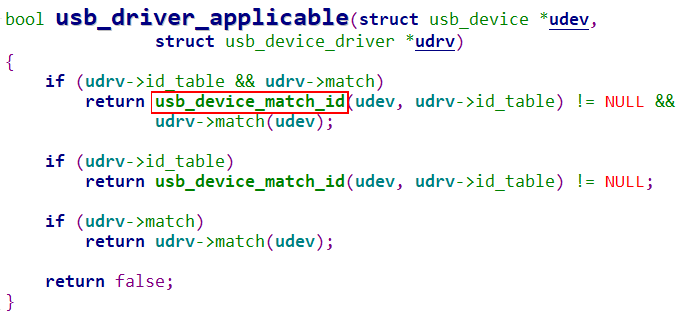
[](file:///C:\Users\zhuming\Documents\youduqt\14581706-102253-zhuming\image\temp\%7be3e6fb11-7cd8-490f-94a3-4555664d9228%7d.png)

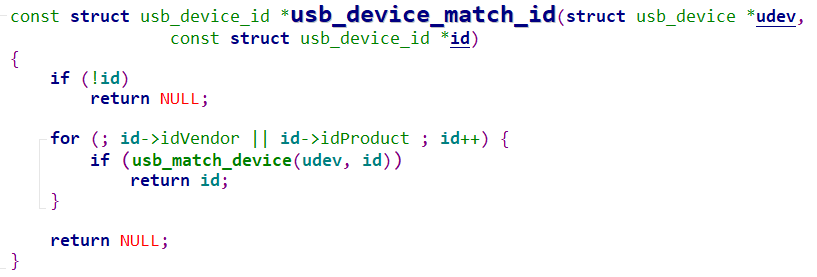
### 3）match通用USB设备驱动

向内核注册设备和驱动，最终均会出发一个设备和驱动配对的过程。 驱动模型就会调用xxx\_match函数，这个函数是负责判断驱动和设备是否配对。

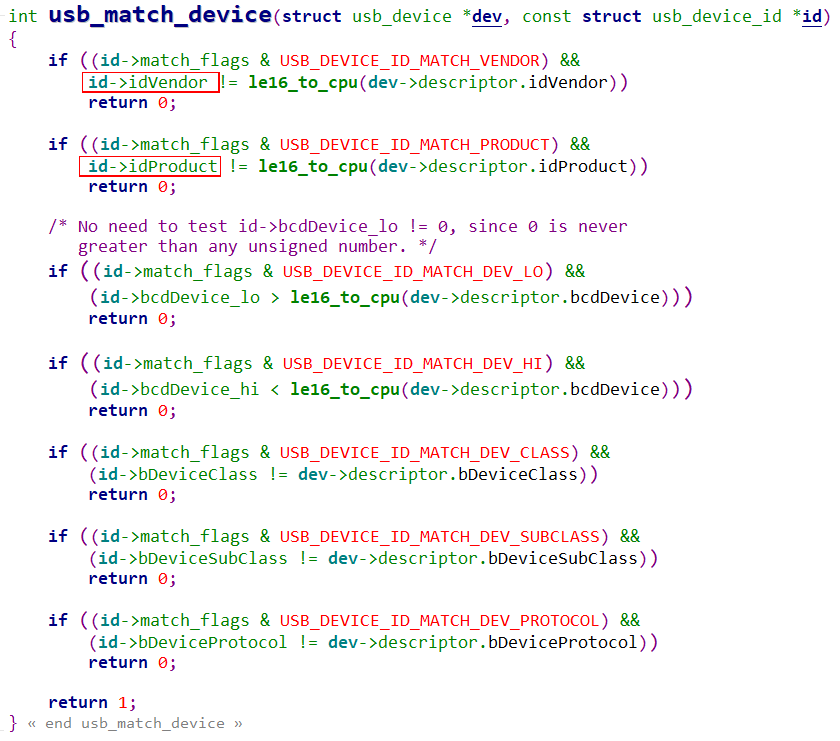
当发现有新的设备加入时，总线会为其寻找驱动，如果找到合适的driver后，更新device中的driver成员，并一并更新driver中的klist\_devices成员，这是一个match的过程，同时将他们绑定。

usb\_device\_match()函数被调用

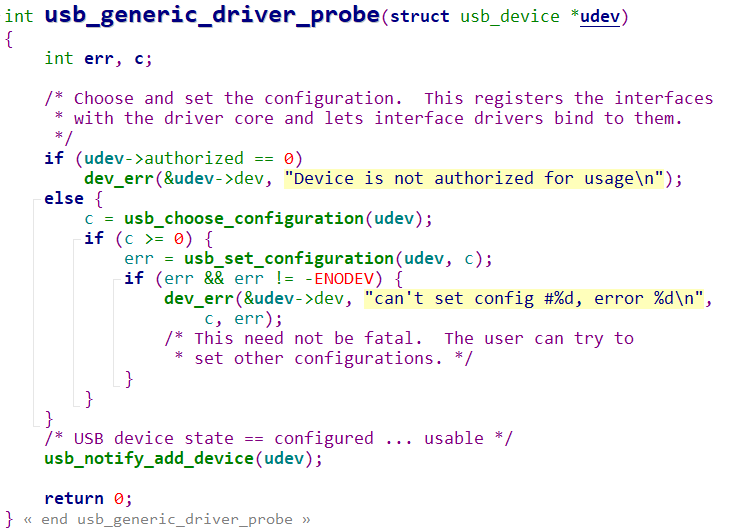
[](file:///C:\Users\zhuming\Documents\youduqt\14581706-102253-zhuming\image\temp\%7b1a51100c-5168-45f3-8382-10c3f48235ff%7d.png)

[](file:///C:\Users\zhuming\Documents\youduqt\14581706-102253-zhuming\image\temp\%7b24f45983-3be7-463c-8d05-00426e892ea2%7d.png)

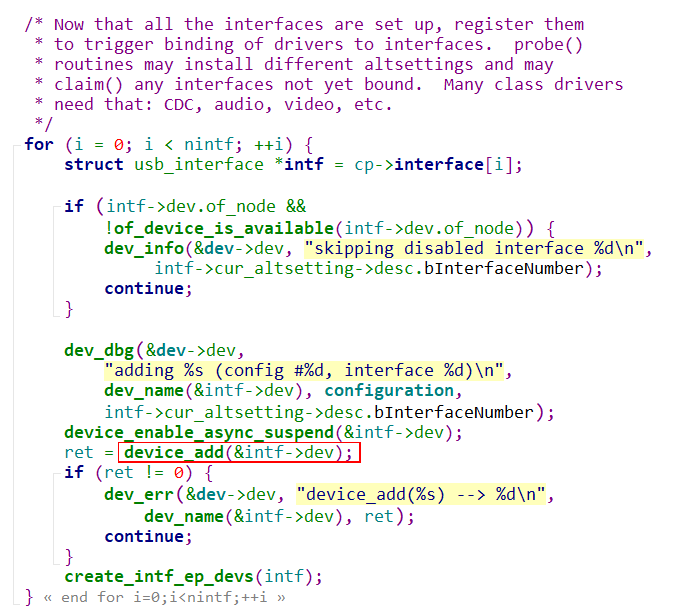
通过PID和VID进行匹配

[](file:///C:\Users\zhuming\Documents\youduqt\14581706-102253-zhuming\image\temp\%7b28adc8df-b59d-4ba0-956e-8e3e3e675db4%7d.png)

进一步调用probe函数，usb\_generic\_driver\_probe()函数中会根据之前枚举获取的配置信息，对相应设备进行配置。

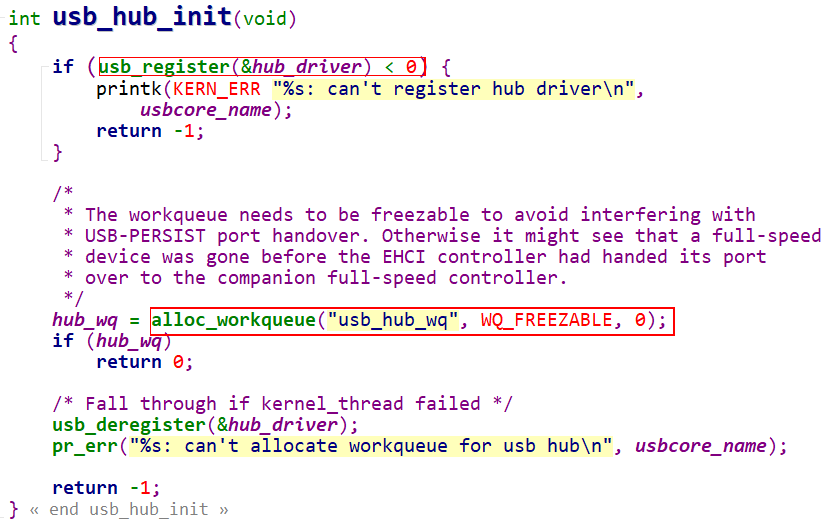
[](file:///C:\Users\zhuming\Documents\youduqt\14581706-102253-zhuming\image\temp\%7b01d0fbae-afe1-4a0a-9478-d0874beb8596%7d.png)

将root hub下的每一个端口注册到内核：

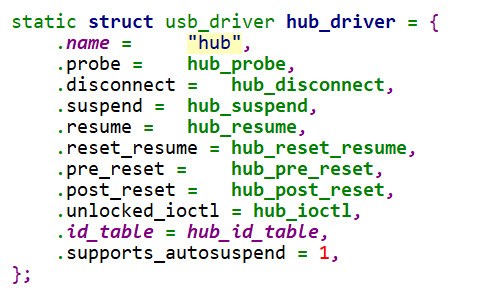
[](file:///C:\Users\zhuming\Documents\youduqt\14581706-102253-zhuming\image\temp\%7b8fc36a07-b2c1-4352-8017-9efb22bad5b8%7d.png)

### 4）注册hub驱动到内核

在注册通用USB设备驱动之前，会进行usb hub的初始化。

[](file:///C:\Users\zhuming\Documents\youduqt\14581706-102253-zhuming\image\temp\%7bea2559b3-ec90-42d7-b22e-53ab33535988%7d.png)

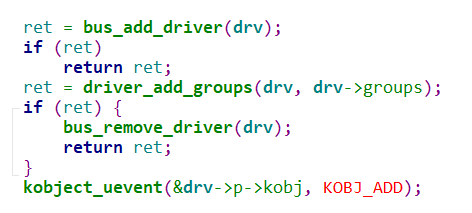
usb\_hub\_init()只完成两件事，一是注册hub驱动，二是创建工作队列，用于处理hub中断下半部事务。

[](file:///C:\Users\zhuming\Documents\youduqt\14581706-102253-zhuming\image\temp\%7b3564cc93-a3b5-465a-a548-94a340d7dbb3%7d.png)

注册USB hub接口驱动

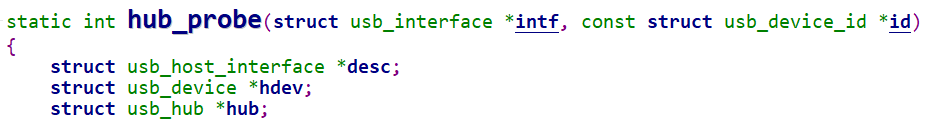
[](file:///C:\Users\zhuming\Documents\youduqt\14581706-102253-zhuming\image\temp\%7b2ae0ab51-bcdd-4a70-877e-29eb4965a48b%7d.png)

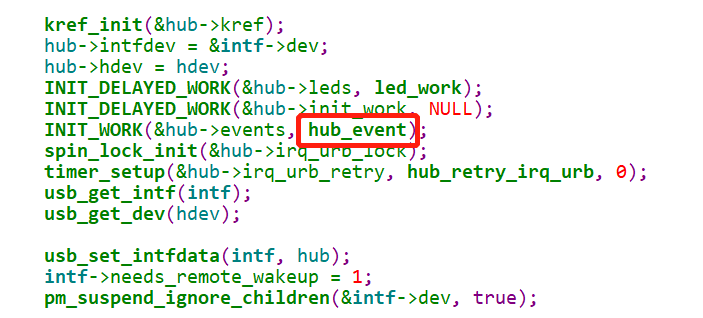
将hub驱动添加到总线

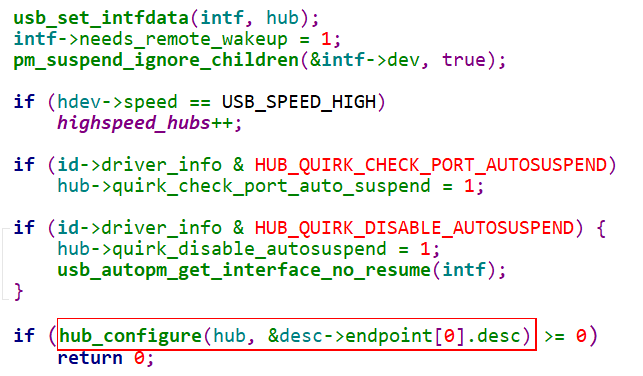
[](file:///C:\Users\zhuming\Documents\youduqt\14581706-102253-zhuming\image\temp\%7be3e6fb11-7cd8-490f-94a3-4555664d9228%7d.png)

### 5）match hub驱动

内核驱动模型中的usb\_device\_match()函数被调用，然后hub\_probe()函数被调用。hub\_probe函数在发现hub设备的时候由内核调用，用于初始化hub设备对象

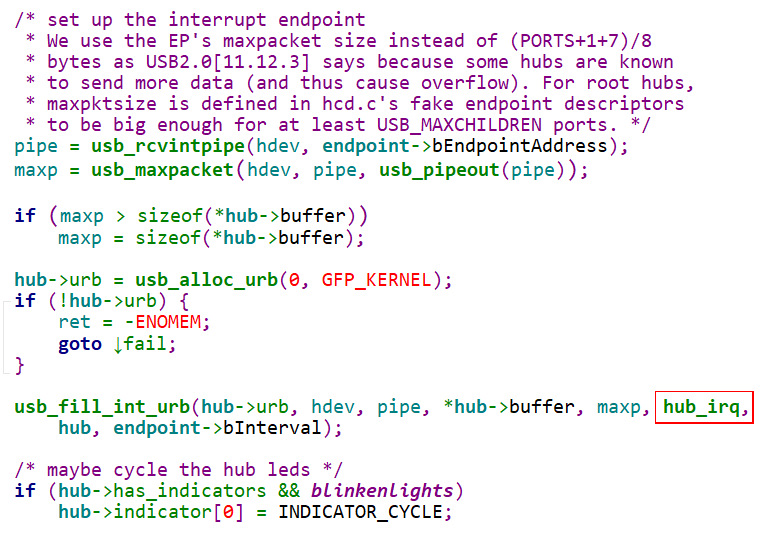


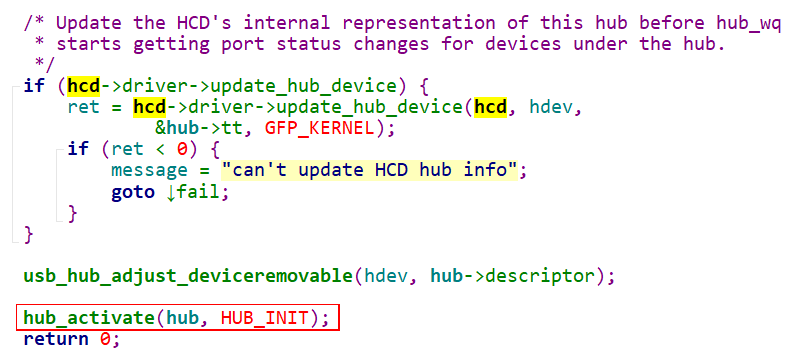


[](file:///C:\Users\zhuming\Documents\youduqt\14581706-102253-zhuming\image\temp\%7b52009af0-74ef-4719-8fcb-33b8ccb46ee8%7d.png)

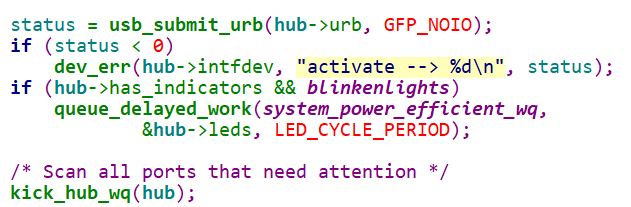
Hub->init\_work用来分配hub设备对象空间，然后进行一系列初始化。INIT\_WORK用来初始workqueue任务，work\_struct实例最重要的成员就是func函数指针，这里把此指针初始化为hub\_event()函数。

接着，hub\_configure()函数初始化一个中断URB，回调函数为hub\_irq()，并激活hub。

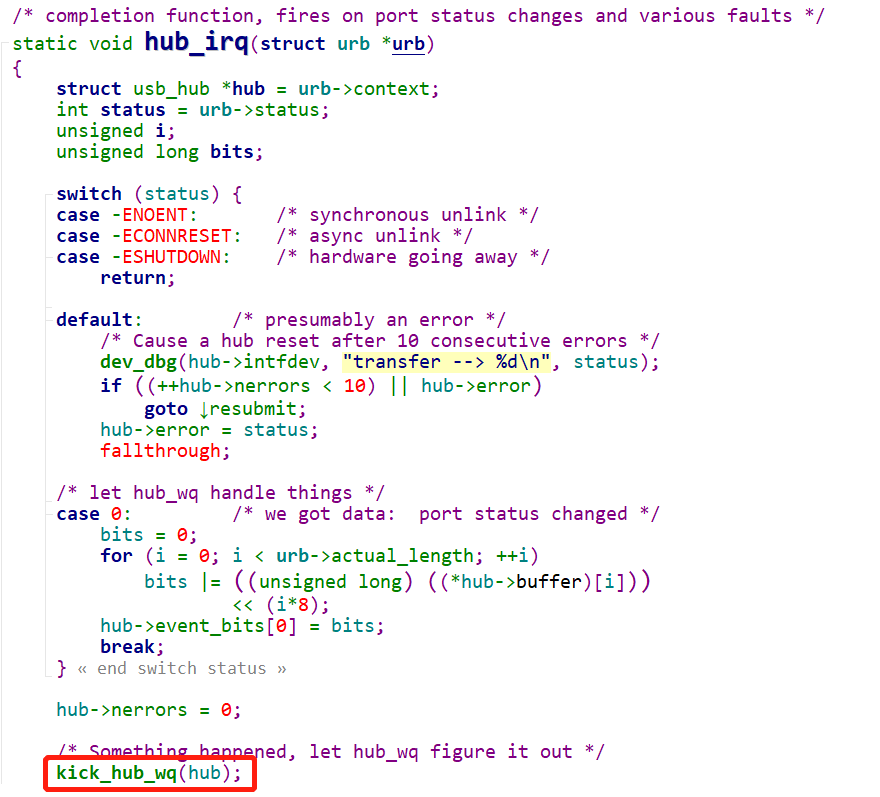
[](file:///C:\Users\zhuming\Documents\youduqt\14581706-102253-zhuming\image\temp\%7b91a334fd-9149-4dff-8643-0b8742509bb0%7d.png)

[](file:///C:\Users\zhuming\Documents\youduqt\14581706-102253-zhuming\image\temp\%7b3f58f058-73e3-4c66-82dd-4746df7cda85%7d.png)

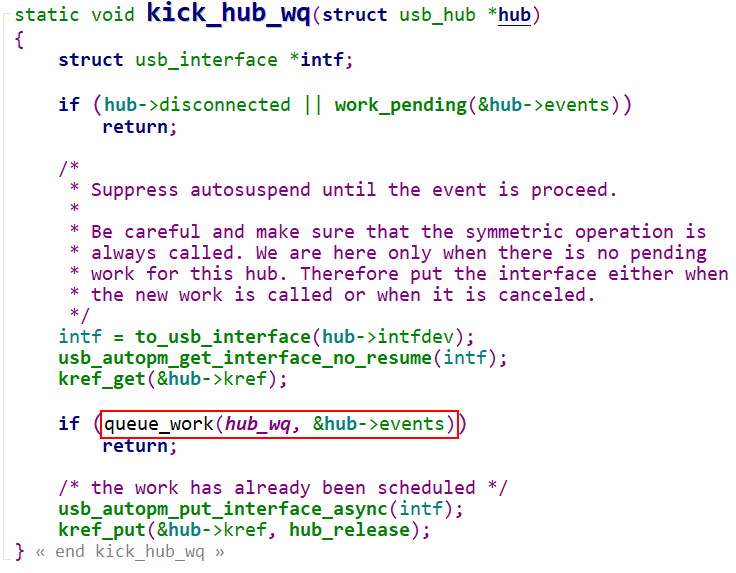
hub\_activate()函数和hub\_irq()函数都会调用kick\_hub\_wq()函数。用来定时检测hub端口状态，如果hub发生变化，将该hub添加到内核链表hub\_event\_list中去，并唤醒内核。

[](file:///C:\Users\zhuming\Documents\youduqt\14581706-102253-zhuming\image\temp\%7b665eb4e9-9a96-4f3d-b2fe-25dc3d296d20%7d.png)

CPU收到USB中断事件后，在中断的上半部通过hub\_irq()处理，这个过程对时间要求比较敏感，所以处理完关键的工作后就把剩余的工作交给工作队列（中断的下半部）来处理。



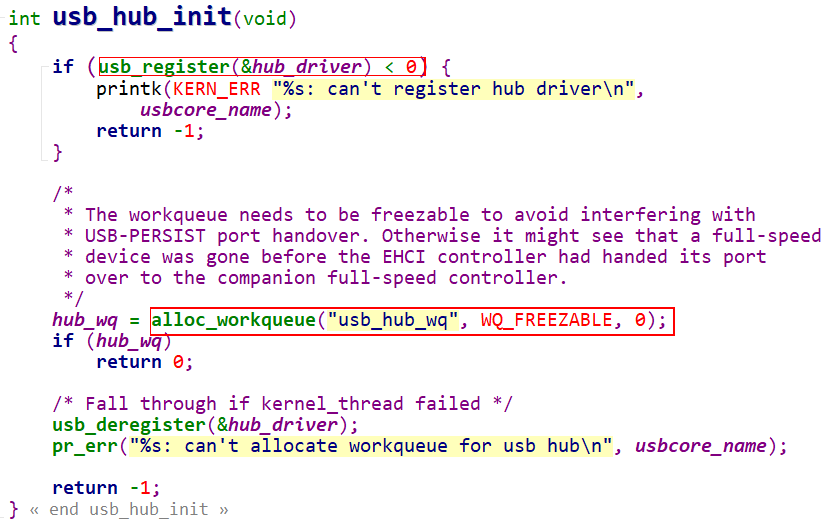
踢工作队列一脚，触发工作队列的任务。内核收到了一些USB事件，但现在还不知道是什么内容，由于时间紧迫，把它交给工作队列异步处理。kick\_hub\_wq()函数实现如下：

[](file:///C:\Users\zhuming\Documents\youduqt\14581706-102253-zhuming\image\temp\%7bc6942664-d0f3-49b2-aa83-48bd4963d0ee%7d.png)

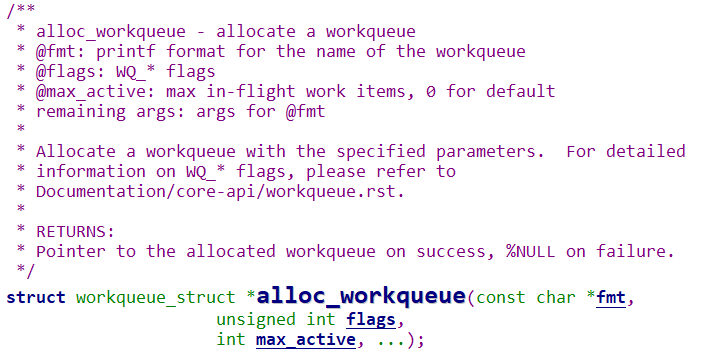
这样就出发了workqueue的任务

### 6）usb\_hub\_init函数

USB控制器通过外部中断与CPU进行异步通讯，由于中断对时间的要求特别敏感，所以USB驱动把主要的工作放在工作队列中处理，这样一来可以有效的提高系统的响应能力。要使用工作队列，第一步，先创建workqueue，在usb\_hub\_init中被调用，如下。

[](file:///C:\Users\zhuming\Documents\youduqt\14581706-102253-zhuming\image\temp\%7bea2559b3-ec90-42d7-b22e-53ab33535988%7d.png)

Alloc\_workqueue()函数声明如下



函数调用流程：

1. queue\_work()

在hub\_irq()函数中被调用，触发workqueue的任务。

1. hub\_thread()

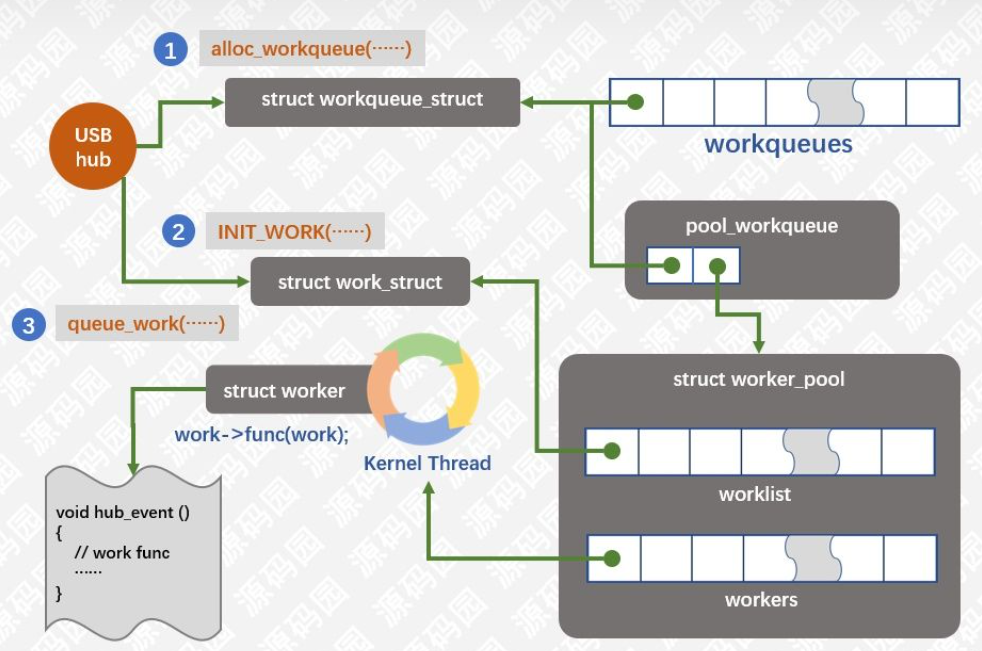
在usb\_hub\_init中，由alloc\_workqueue()函数创建工作队列，并创建内核线程hub\_thread。hub\_thread函数中会调用hub\_events处理hub\_event\_list链表中的hub。当hub\_events函数返回后，hub\_thread进入休眠，等待下一次的唤醒。

c）hub\_event()

### 7）工作队列

以上，USB hub驱动注册过程用到了linux重要的处理机制workqueue。工作队列是内核异步处理机制中的一种，通过内核线程实现，它在进程的上下文中运行，可以重新调度和睡眠。内核线程用线程池进行管理能更加有效的利用资源，经常用于执行中断的下半部程序。

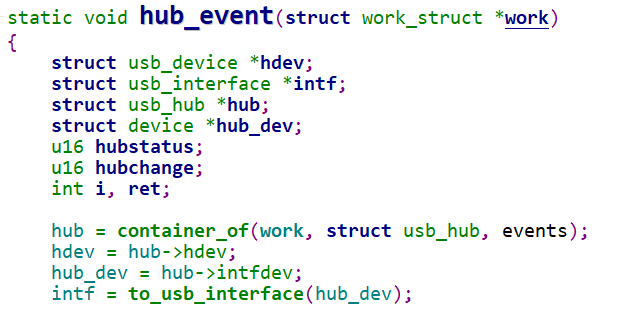
创建队列、初始化工作任务、触发工作队列的任务是工作队列的基本操作步骤，已在上文中用绿色背景标识出来了。下图是从网上找来的workqueue的基本结构图，有时间可以再来详细研究一下。

[](file:///C:\Users\zhuming\Documents\youduqt\14581706-102253-zhuming\image\temp\%7ba62e123a-8e0e-4cb5-81ac-bd118213c1d4%7d.png)

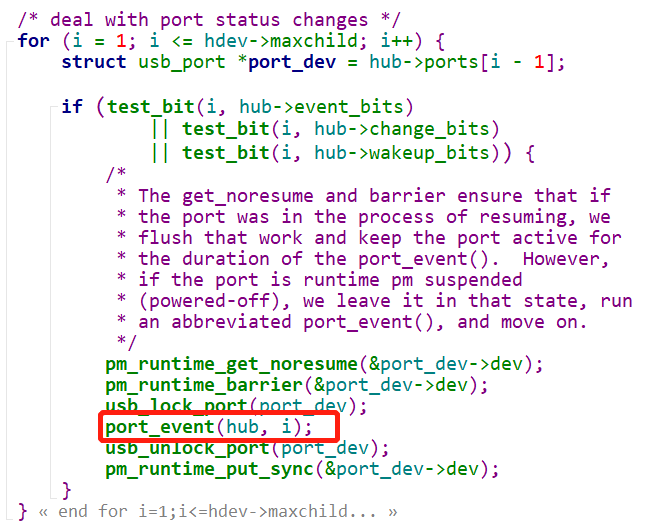
## 第三阶段：检测Hub端口的变化状态。

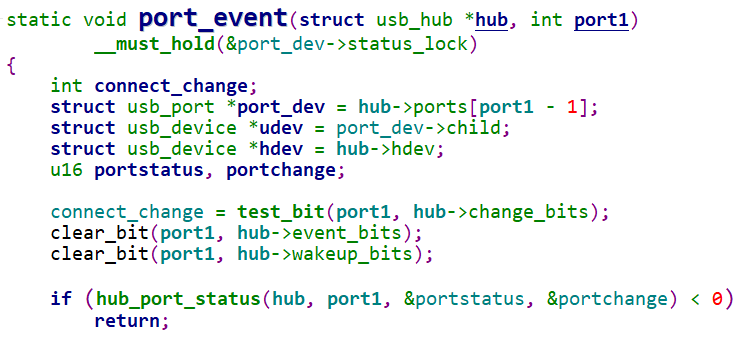
### 1）port状态变化

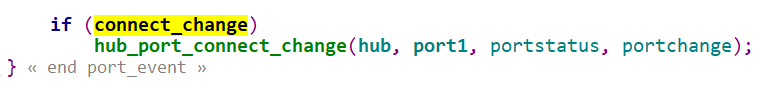
从以上的分析可知，当Hub端口状态发生变化，最终工作队列会调到线程处理函数hub\_event()。

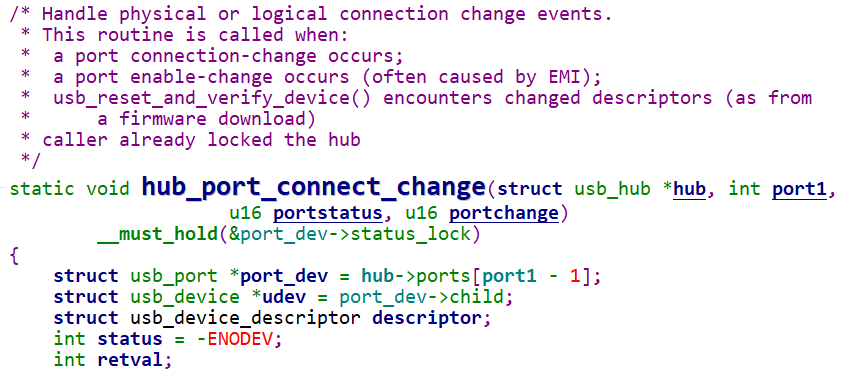


处理每一个port状态变化

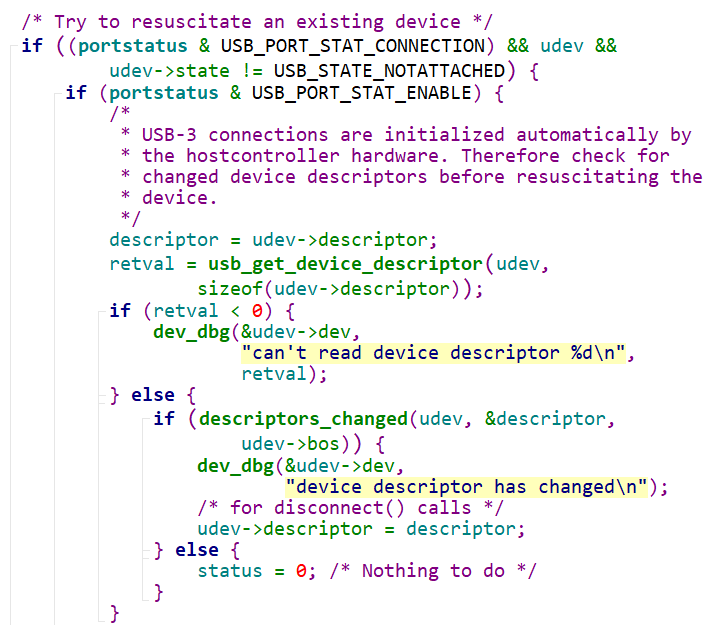




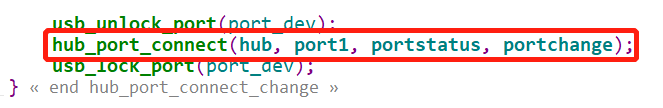




恢复设备前检查设备描述符是否改变

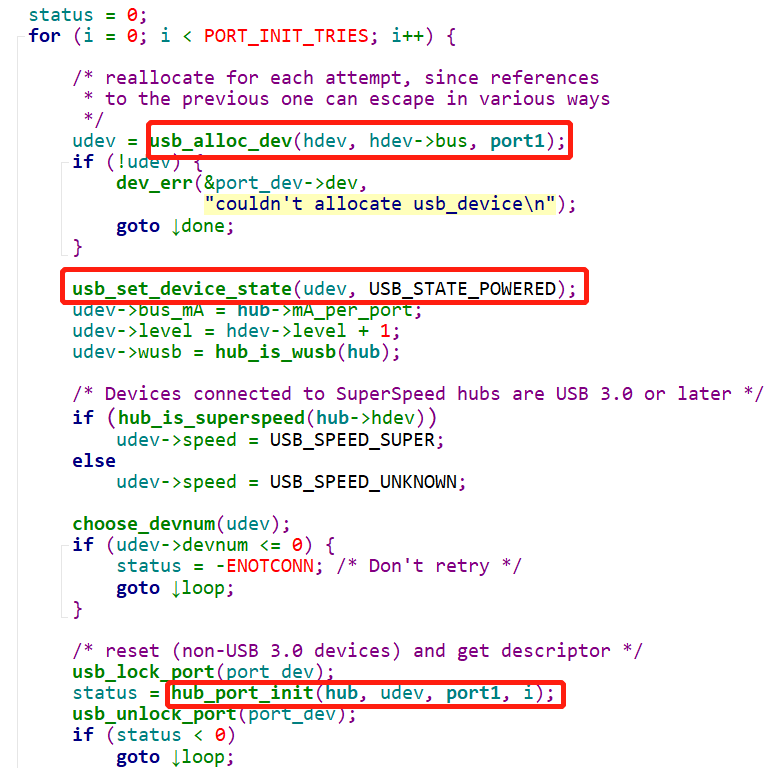


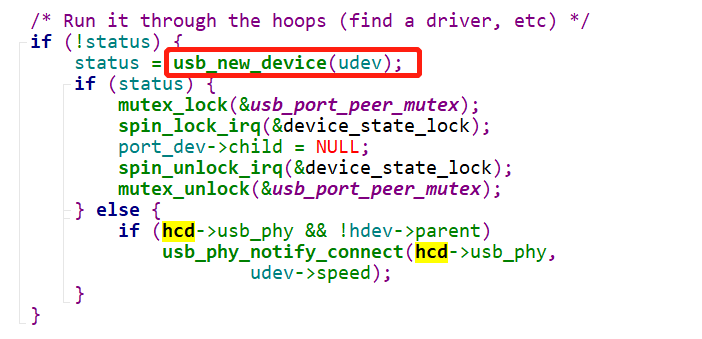
### 2）调用hub\_port\_connect()函数



Hub\_port\_connect()函数中几个重要的步骤：

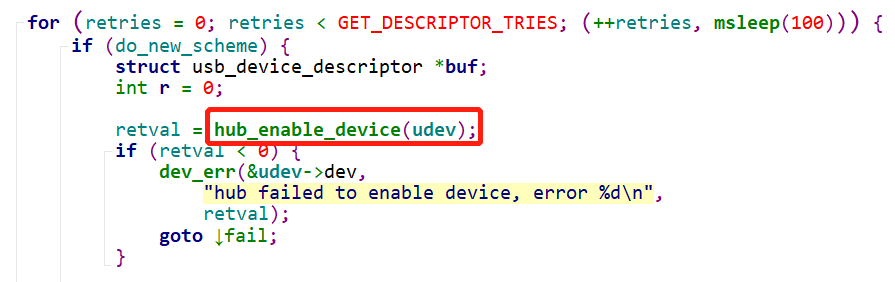
1. 为usb\_device类型的USB设备指针分配内存空间，同时作一些初始化工作
2. 设置USB状态为powered
3. Hub port的初始化
4. 将USB设备添加到内核中





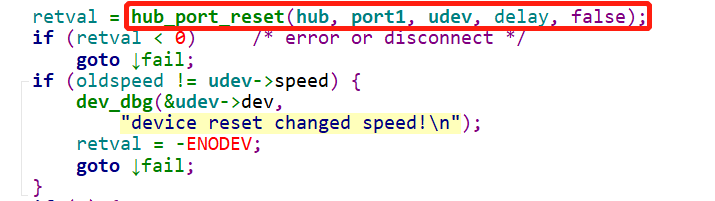
Hub\_port\_init()函数中主要负责

1. 复位设备
2. 设置设备地址
3. 获取设备描述符（这里只获取设备描述符头部信息，有几种配置，长度等基本信息）

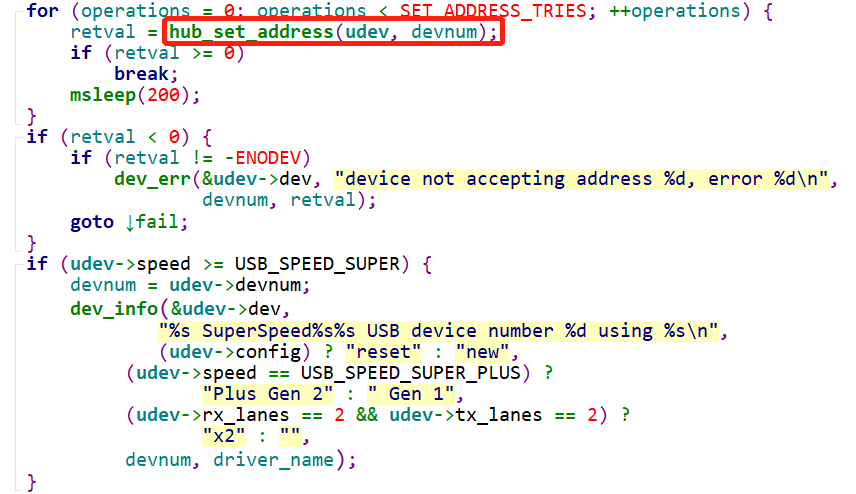


hub\_enable\_device()实际调用的是xHCI驱动里的xhci\_enable\_device()函数。

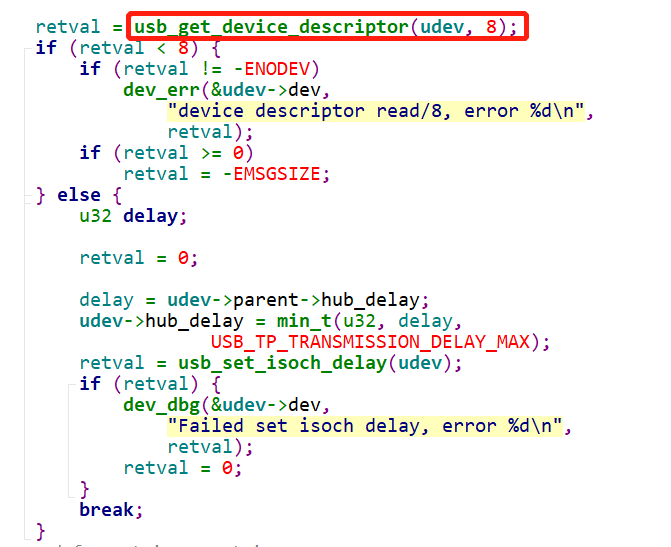
复位设备



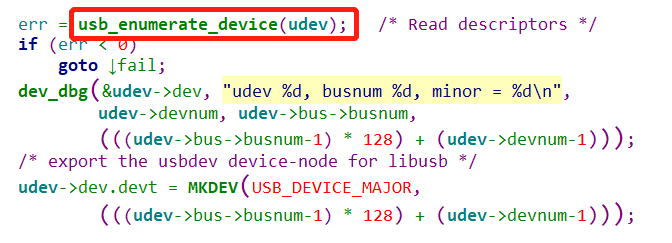
为设备分配地址

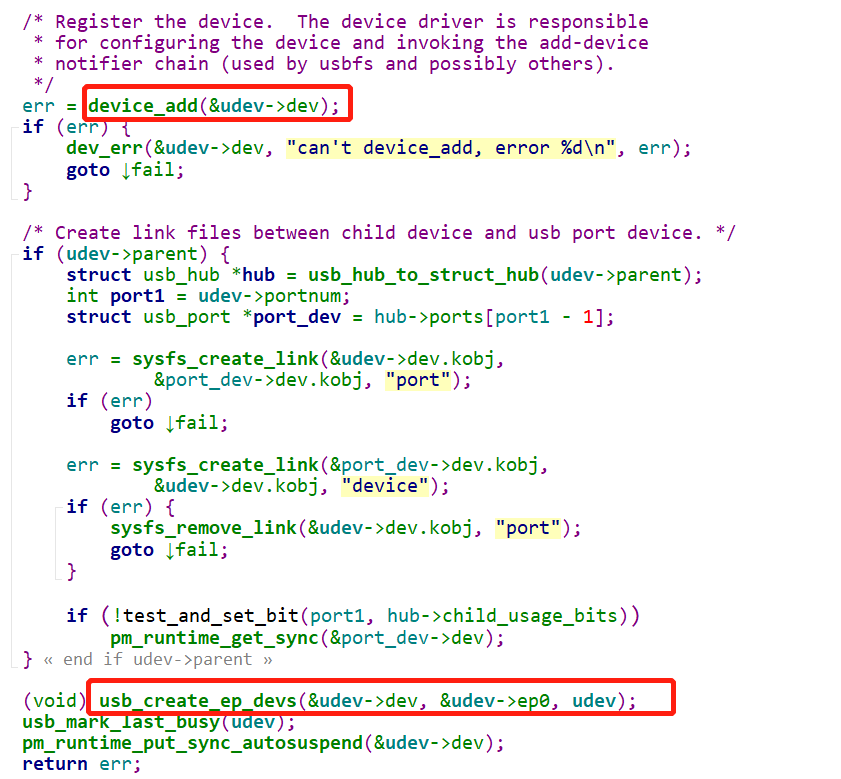


获取通用描述符信息



usb\_new\_device()函数先调用usb\_enumerate\_device()获取该设备所有描述符信息，会调用device\_add来添加设备节点到内核。





1）device\_add()函数

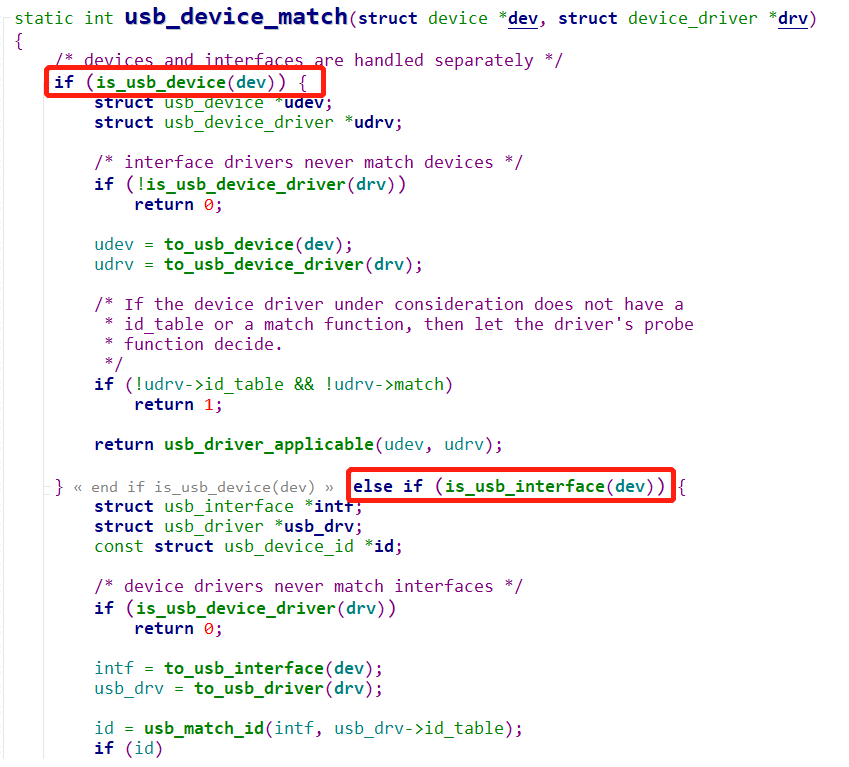
无论插入的是hub设备还是USB设备都要去匹配USB设备驱动usb\_generic\_driver()，匹配成功会将设备接口添加到内核，如果插入hub的是另一个hub，则为hub创建设备模型。

2）usb\_create\_ep\_devs()函数

无论是hub还是USB设备，都有端点，该函数主要作用是配置端点信息并将这些端点添加到内核并创建设备节点。



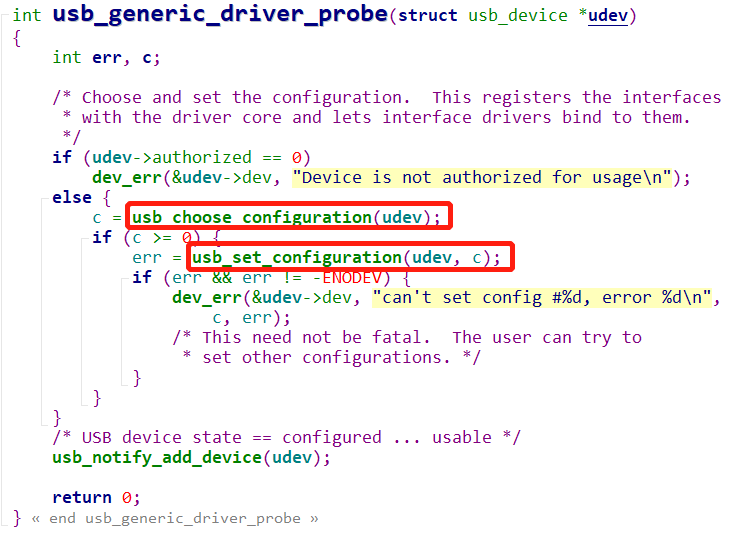
### 3）内核驱动模型调用usb\_device\_match()函数。



根据是add\_device添加的是usb接口还是usb设备。

1. 如果是usb设备，调用usb\_generic\_driver\_probe()函数。
2. 如果是interface，且是hub接口，调用hub\_probe()函数。
3. 如果是interface，且是usb接口，根据实际接口情况调用接口函数，比如cypress\_probe()函数。

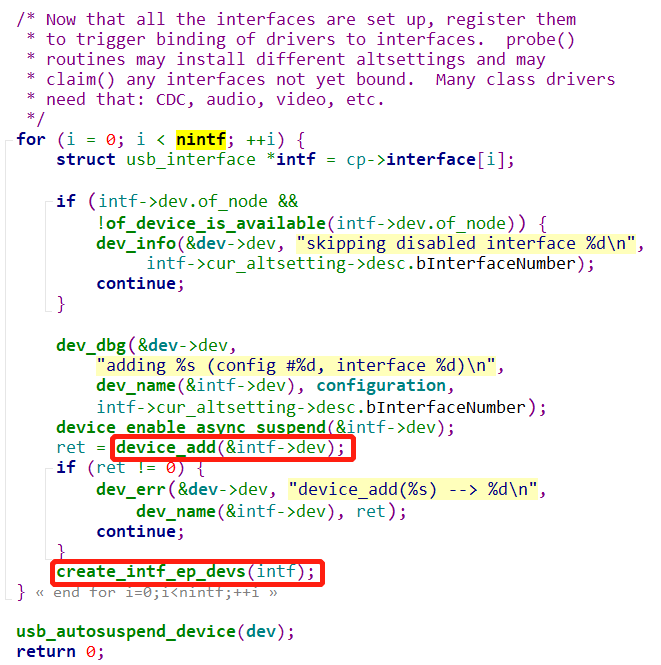
以usb设备为例，上述步骤中已完成设置address，获取描述符等操作，接下来在generic\_driver\_probe()函数中进行choose配置和set配置的操作。



### 4）usb\_set\_configuration()函数

在usb\_set\_configuration()函数主要完成

1. 给USB设备发送SET\_CONFIGURATIO消息，以完成配置
2. 设置设备状态为Configed
3. 初始化该配置中所有接口
4. 注册所有接口到内核，以触发驱动程序与接口的绑定
5. 将每个接口下的所有端点添加到内核并创建设备节点



这样，无论是hub还是usb设备，其上所有接口和端口均以这种方式注册到内核了。在这个过程中，根据hub\_port\_connect\_change()函数的触发，完成了USB设备的枚举过程。