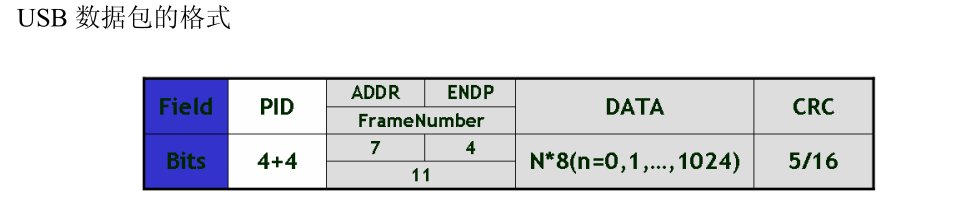
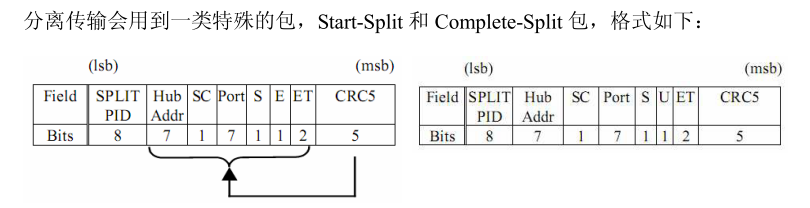
* 1. USB
     1. 数据格式



以PID分类，USB包可分为令牌包，数据包，握手包及特殊包。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | PID | 数据域 | CRC | 备注 |
| 令牌包 | SETUP，  IN，  PUT  SOF | 7addr+4point（或者framenum） | 5位 | SOF是特殊的一类令牌包，其数据域为11个字节的framenum |
| 数据包 |  | N字节数据 | 16位 |  |
| 握手包 | NACK，  ACK，  STALL，  NYET | X | X | 只有PID字段 |
| 特殊包 |  |  |  |  |

分离传输包格式：



* + 1. 端点，数据流

端点：相当于buffer，一个端点buffer可能存在若干个数据包。

数据包—端点—管道（协议层）—传输层（传输类型：一种管道对应一种类型）--事务层（一个事务对应如果次传输）

SYNC是每个packet的起始，EOP是每个packet的结束。

传输类型：控制传输，中断传输，批量传输，同步传输。涉及的内容：传输机制，USB带宽，主机安排的传输任务，最大包长度。

默认管道：控制传输，处理请求类型：标准、基于class类和厂商自定义。

关于管道和传输类型的关系：

----控制管道：控制传输

管道

----流管道：中断，批量，同步传输。

问题：

Q1：其他3中传输类型是否会使用控制管道？

Q2：Host—(广播)—设备，那么设备—(?)—Host呢？

Q3：微帧时间？

S3：USB1.1中定义1 frame为1ms，USB2.0中定义1 frame中有8 microframe，即1 microframe = 125us。

* + 1. 微帧

usb1.0中采用帧（frame）的概念来进行传输时间管理,其中：1 frame = 1 ms,而usb2.0中引入了微帧的概念,一帧分为8个微帧，即每个micro-frame = 125us。

* + 1. 传输事务

传输事务以：令牌包+数据包+握手包的形式进行（除同步传输，同步传输只有令牌包和数据包）。

Bulk传输：发送IN（或者OUT）的令牌包，USB设备接收到令牌包后，进行发送（或者接收）操作，如果发送操作成功，数据发送完成后HOST将返回ACK，否则设备（在接收完HOST的令牌包后）返回错误状态（NAK,STALL,data error）；如果接收操作成功，设备发送ACK，否则返回错误状态。

SETUP令牌包和SETUP事务：SETUP令牌包是一种USB数据包的格式，而SETUP事务是一个传输流程，其中分成:

SETUP数据包发送---设置数据(DATA0)发送---设备返回ACK

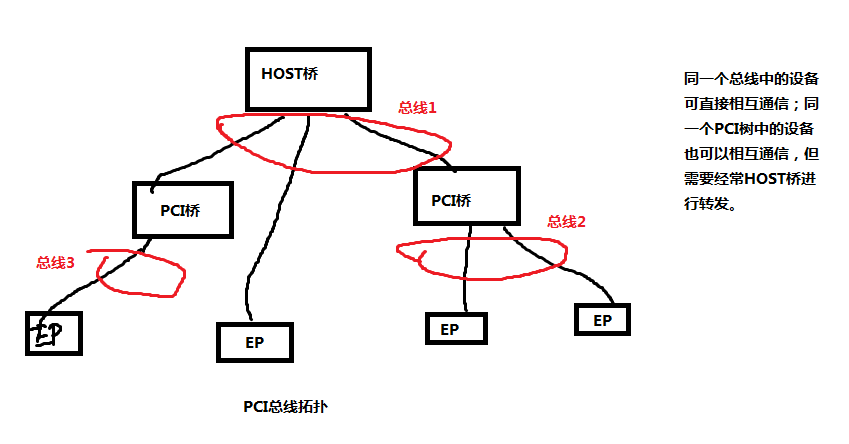
也就是说，SETUP过程中首先会发送SETUP令牌包。

* + 1. 枚举
  1. CAN总线
  2. PCI总线

局部总线：不是作为处理器系统的总线连接cache和主储存器。

外部设备？

PCI桥总线拓扑结构，如下图所示：



PCI总线：由PCI设备和PCI协议组成。

PCI设备：HOST主桥，PCI-TO-PCI桥，PCI设备；PCI设备还分成主、从设备。

PCI总线域：PCI总线地址空间集合，具体表现为BAR等等。

PCI HOST主桥通过初始化PCI设备的BAR空间，实现对PCI设备的存储器空间的访问。

换句话说，Bar空间的地址有**3重**身份，首先他是PCI总线地址，其次他和主设备的存储器空间地址有映射关系，最后他还是从设备的存储器空间地址映射。

PCI桥的base寄存器？

ID译码方式：通过bus number,device num,function num,register num实现。

Bus number使用：HOST桥首先判断该bus number是否在自己这条PCI总线上（即pci bus 0），如果是，则产生type0请求，否则，产生type1请求。当产生type1请求时，pci bus 0总线上的PCI桥将会收到该请求，但只有bus number落在该PCI桥的secondary bus number和subord bus number之间，该PCI桥才会接收刚type1请求，如果该bus number与该PCI桥的secondary num相同，PCI桥将把该type1请求转换为type0请求，转发到其PCI总线中。

* + 1. PCI总线数据传输：Posted和Non-Posted

Posted:多事务并行，Non-Posted:多事务串行。

只有存储器写事务能采用Posted事务，存储器读，IO读写，配置空间读写都是采样Non-Posted事务。此部分属于PCI硬件逻辑控制。

* + 1. PCI总线枚举

枚举过程中使用的是什么事务？配置读写。

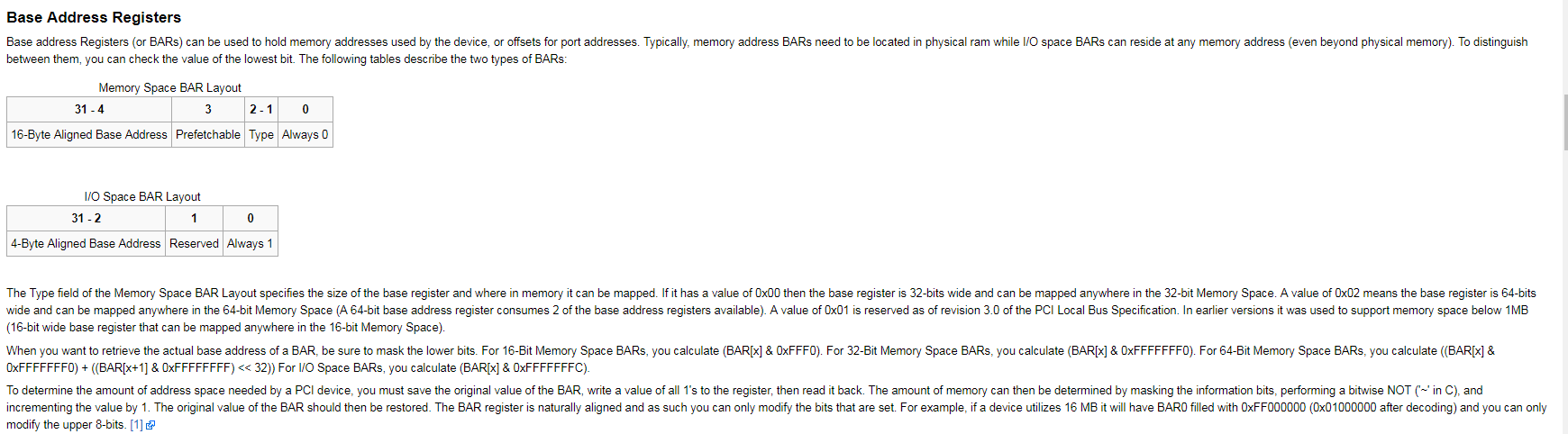
* + 1. PCIe设备bar空间大小

PCIe设备Bar空间大小由设备自己申请，然后通过主PCIe设备进行分配。那么主PCIe如何确定PCIe设备申请的Bar空间大小？

简单说，就是主设备向从设备的bar空间基地址配置寄存器（即配置空间常见的Base Address Register）写1，然后读回，根据最低位1的位置，确定从设备申请的BAR空间大小。比如，读回的值为0xFF000000,则表示从设备申请的Bar空间大小为0x100000，意味着你只能改变Bar空间基地址配置寄存器的高8位。

而主设备如何确定哪块Bar空间需要配置？也是通过上述方法，如果读回的值为0，则表示不需要配置。

下图是维基百科对Bar空间基地址配置寄存器的详细描述。



* + 1. Pcie Mem和Memio空间分配

<PCIe的Mem和Memio空间分配.xmind>