**USB2.0 LPM介绍**

# 目标

在USB2.0 Specification的电源管理中，只定义了正常工作状态(L0)和挂起状态(L2，低功耗状态)，挂起状态要求总线Idle 3ms以上(即3ms内没有任何数据传输)才能进入，且至少挂起3ms，另外，通过Resume信号挂起的设备从挂起状态恢复到正常状态也要经历至少3ms以上的延时。

基于这些规定：

一方面，那些需要即时响应的设备在挂起期间不能得到即时的服务，增加了响应延迟。

另一方面，如果设备想要减少延迟，就只能尽量不进入挂起状态，这样一来，又增加了系统的电源消耗。在早期的桌面机或者服务器上，增加的这点功耗也许不算什么，但是随着移动类设备的发展及节能相关的要求，需要一种更好的电源管理方案来解决上面提到的两个问题，同时还能兼容目前已有的USB2.0设备。

# 参考资料：

1. USB2\_LinkPowerMangement\_ECN.pdf
2. USB2-LPM-Errata-final.pdf;
3. USB2.0 Specification.pdf
4. xHC Specification.pdf

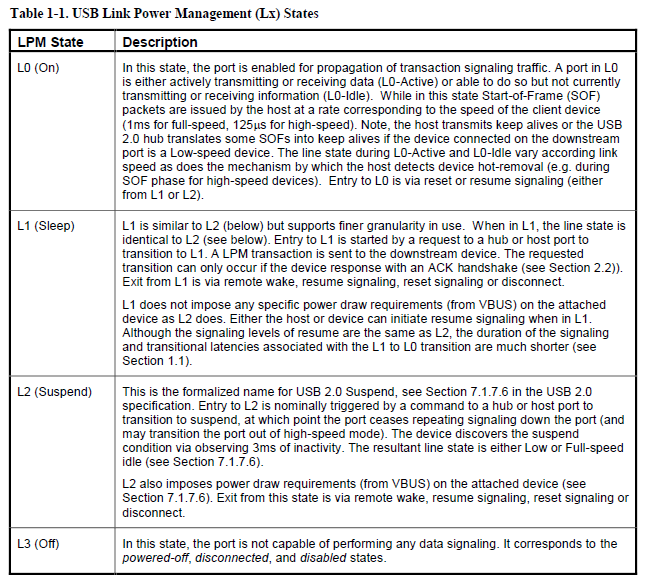
# Link Power Management

USB2.0 LPM规范是USB2.0的一个补充规范，由两个文档共同构成。USB2\_LinkPowerMangement\_ECN.pdf对USB2.0的电源状态进行了重新定义，定义了LPM的基本定义，USB2-LPM-Errata-final.pdf对前一个文档进行勘误，需要结合两个文档一起学习。

在LPM规范中定义了4个状态：

1. L0（正常工作）
2. L1(sleep睡眠状态)
3. L2(suspend挂起状态)，
4. L3(power off正电状态)

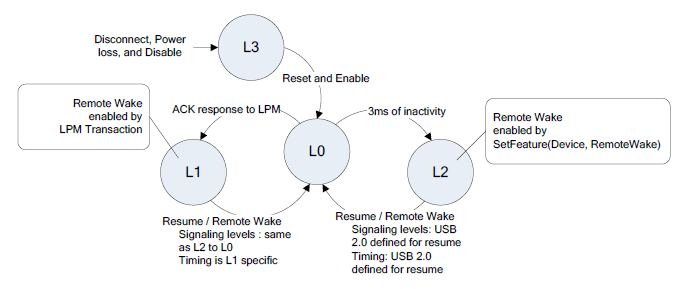
其中L0，L2，L3这三个状态对应的电气状态其实已经在USB2.0规范中进行了定义，只是叫了不同的名字而已，具体的每个状态定义如下表



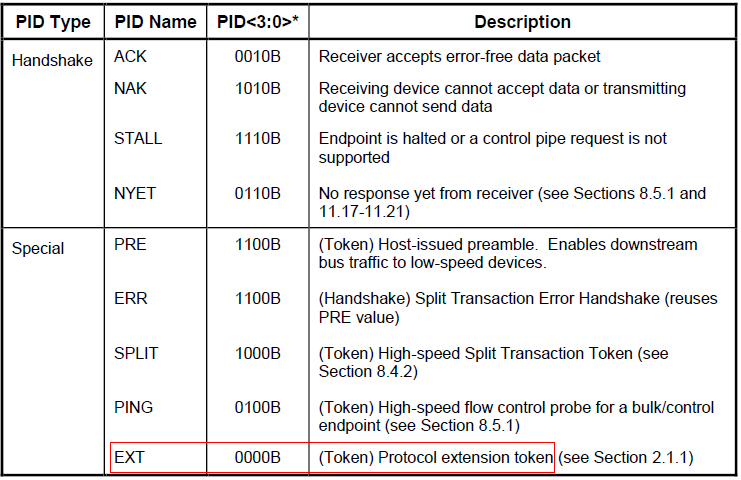
1. L0状态对应USB2.0协议中定义的正常工作状态。
2. L1是新增的一个状态，由Hub主动发起，而不再是由设备自己检测总线状态，并根据总线状态决定是否进入低功耗；
3. L2对应USB2.0协议中定义的suspend状态。
4. L3是设备下电时的状态。

除了定义这4个状态外，LPM还定义了一个新的LPM Transaction。利用这个LPM Transaction, Hub可以通过数据包的形式主动请求设备进入L1低功耗状态，通过这种方式发起的睡眠entry时间远小于3ms，设备也没有至少要睡眠3ms的要求，从而避免了由设备检测总线Idle时间达到3ms这种被动挂起的耗电方式。

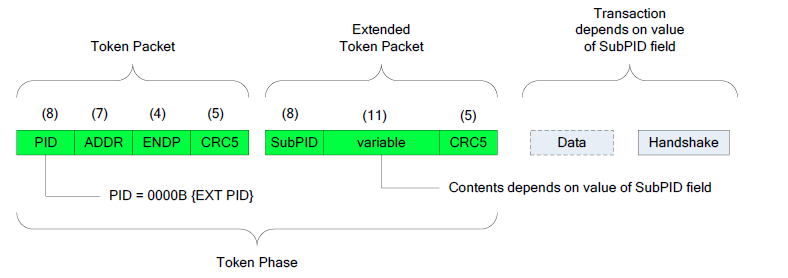
这4个状态的转换关系如下图：



在LPM协议中，利用一个保留的PID定义了一种扩展事务(Ext Transaction)：

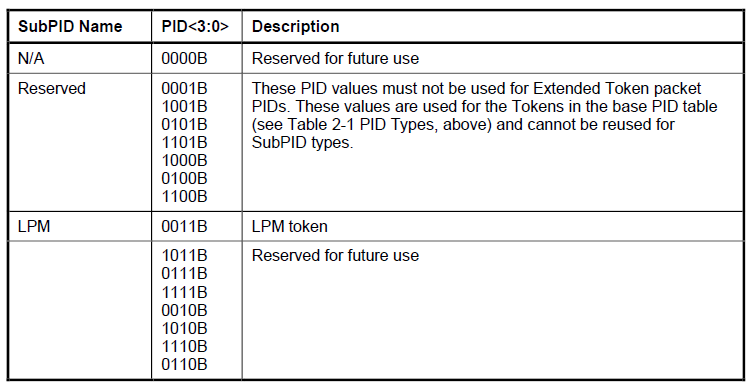


利用这种扩展格式的事务，我们可以定义更多用于不同特性的事务，为未来的进一步扩展留下了空间。Ext Transaction的具体格式：

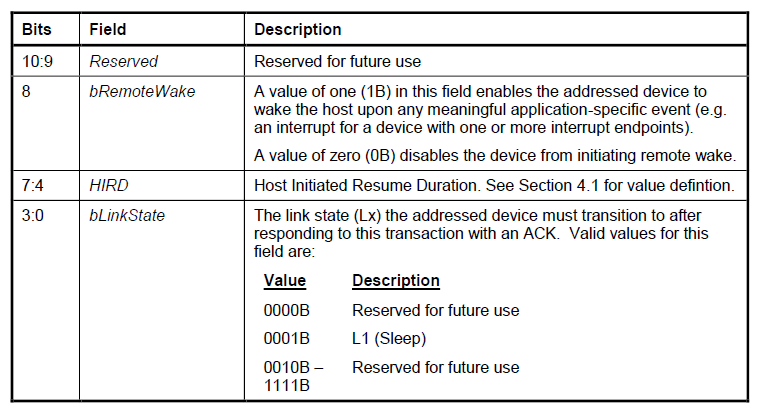


扩展事务由两个连续Token包，外加数据阶段、握手阶段共同构成。数据阶段视扩展事务的特性而定，是可选的。第一个Token包与标准的Token包构成完全相同，由PID, ADDR, ENDP, CRC5共同构成。第二个Token包由SubPID, variable, CRC5三部分构成，variable部分视不同的SubPID而各不相同。Data与Handshake阶段与USB2.0 Spec中定义相同。

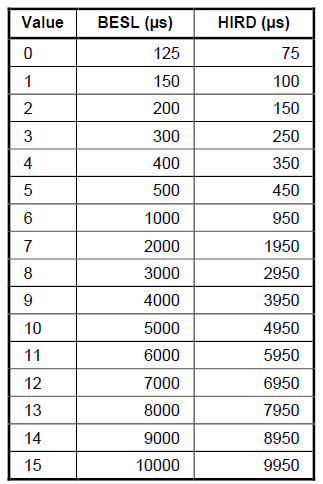
目前已经定义的SubPID如下表：



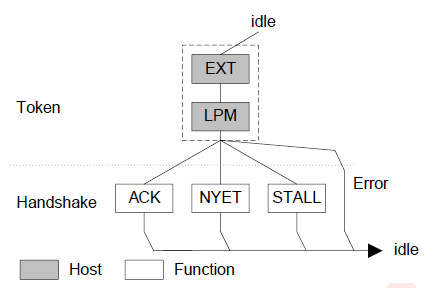
即只定义了LPM SubPID，其variable部分的11比特的具体格式如下：



1. bRemoteWake规定了是否使能设备的远程唤醒功能；
2. HIRD在勘误文档中进行了重新定义，并重命名为BESL（Best Effort Service Latency）,以解决当前HIRD定义中存在的一些缺陷。具体的Host, Device, 以及Software应当以BESL中的定义为准。具体的BESL值表如下：

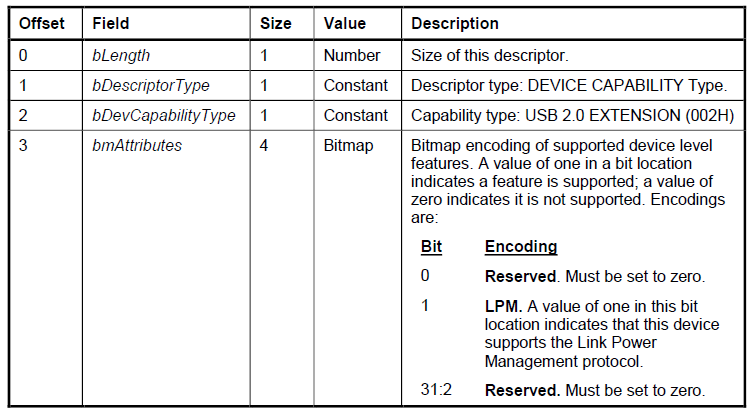


LPM事务由两部分构成，第一部分由两个背靠背的Token包组成(只能由Hub port发起)，第二部分由一个握手包组成(只能由设备进行响应)，具体包序列如下：



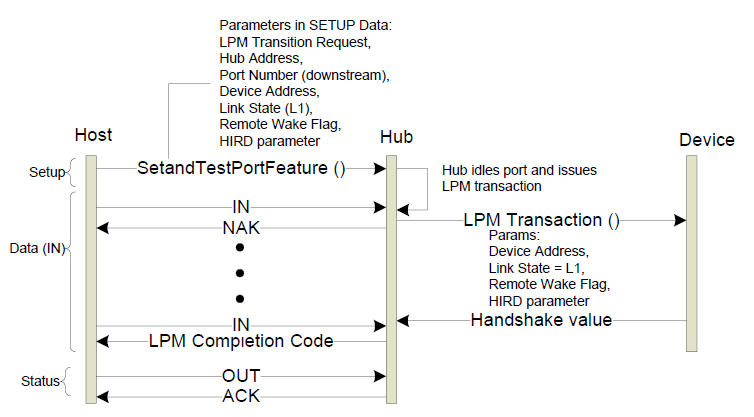
当主机准备好让一个链路进入更低功耗的状态时，会先给链路所对应的Hub发送一个命令，当Hub接收到这个命令后，会根据命令让指定的端口(及链路)进入指定的电源管理状态，当电源管理状态为L1时，则会发起LPM Transaction，设备接收到LPM后，如果设备已经准备好进入L1状态，则会发送ACK进行握手响应，如果设备没有准备好进入L1，则发送NYET进行握手响应，如果设备不支持L1电源状态，则发送STALL进行响应，如果设备检测到令牌阶段有错误或者不支持LPM协议(不理解扩展协议)，设备则不需要进行响应(这会导致Hub检测到事务超时，产生Transaction Error)。

除了定义LPM Transaction时序外，LPM协议为设备定义了一个可选的设备能力描述符(USB 2.0 Extension Descriptor)用于系统软件判断设备是否支持LPM电源管理能力。这个能力描述符是BOS描述符的一部分，系统只能通过获取BOS描述符的方式获取USB 2.0 Extension Descriptor，而不能单独获取USB 2.0 Extension Descriptor描述符。USB 2.0 Extension Descriptor的具体格式如下：



USB2.0协议中规定系统通过给Hub发送Get/Clear/Set Feature命令来控制Hub下游端口的状态，LPM协议在此基础上增加了新的命令用于LPM Transaction。

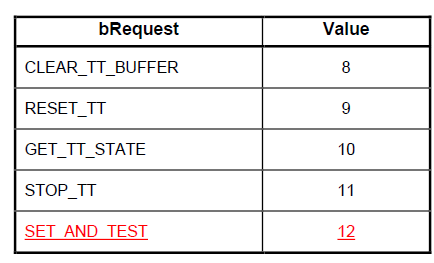
一个完整的LPM Transaction的序列图如下：



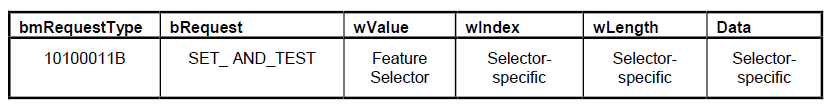
如果Hub在发起LPM Transaction后检测到了Transaction Error（包含超时），它必须发起LPM重试(最多再重试两次)，如果连续三次LPM都出错了，则Hub需要在数据阶段返回ERROR，Hub必须在200us内对Setup请求进行响应。

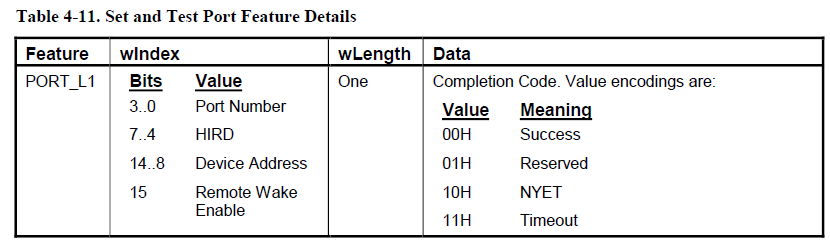
一方面系统软件可以通过命令设置Remote Wake Flag 参数使用或禁用设备的远程唤醒功能。此命令中的Remote Wake Flag会临时覆盖通过Set/ClearFeature() 设置的Remote Wake 特性值*。*另一方面，系统软件可以通过BESL/HIRD参数告诉设备，主机发起的Resume信号会持续多长时间，设备可以根据此信息进行电源管理的优化。

为了能让Host发起L1命令，新增了Set and Test Port Feature命令：

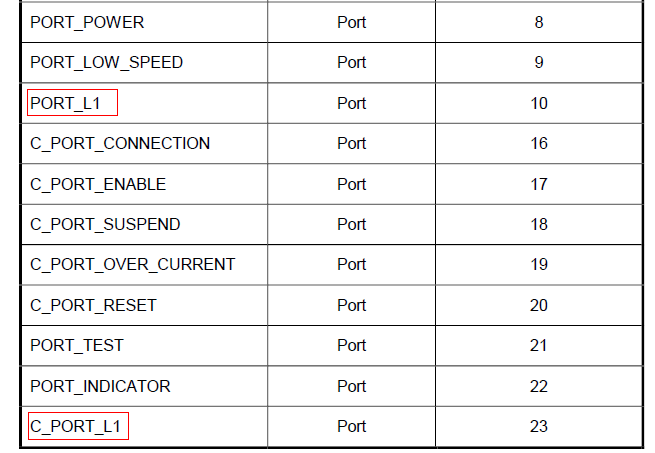


Set and Test Port Feature对应的格式：

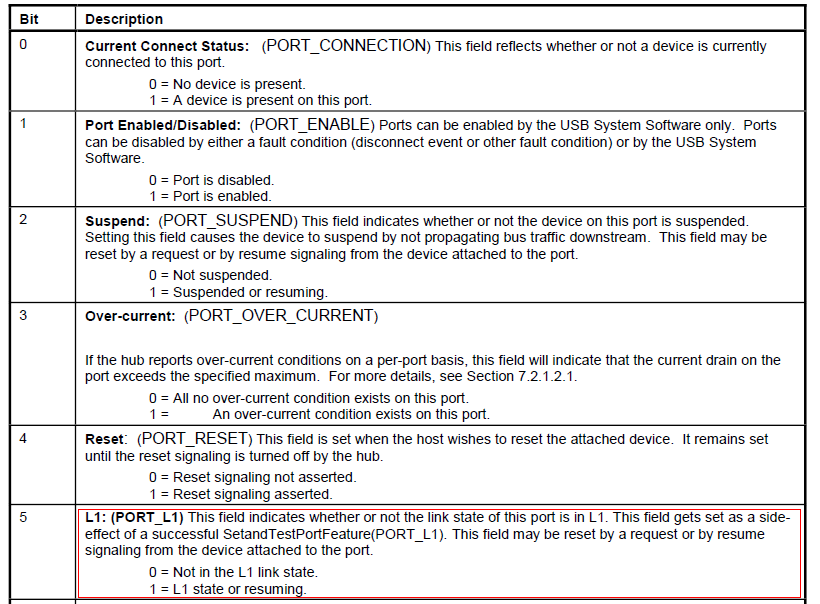




同时也增加了L1相关特性值：



在PortStatus寄存器中也增加了L1相关Bit:



另外，也增加了状态改变相关的Bit：

