

# **增强型主机控制器接口规范**

## **通用串行总线**

日期 :2002年3月12日

修订版 : 1.0



本规范“按原样”提供，没有任何保证，包括任何适销性、非侵权性、适用于任何特定目的的保证，或任何提案、规范或样品。本规范中的信息与英特尔产品相关。本规范未通过禁止反悔或其他方式授予任何知识产权的明示或暗示许可，除非特此授予复制和复制本规范仅供内部使用的许可。

有关进一步的许可协议和要求，请与英特尔联系。

英特尔不承担与使用本规范中的信息有关的一切责任，包括侵犯任何专有权利的责任。除英特尔的此类产品销售条款和条件中另有规定外，英特尔不承担任何责任，且英特尔不承担与英特尔产品的销售和/或使用有关的任何明示或暗示保证，包括与特定用途的适用性、适销性或侵犯任何专利、版权或其他知识产权有关的责任或保证。英特尔产品不适用于医疗、救生或维持生命的应用。

英特尔可能随时更改文档、规格和产品说明，恕不另行通知。

设计者不得依赖于任何标记为“保留”或“未定义”的功能或指令的缺失或特征。英特尔保留这些功能或指令以供将来定义，对于将来对其进行的更改所引起的冲突或不兼容，英特尔概不负责。

在下任何产品订单之前，请与您当地的英特尔销售办事处或分销商联系，以获取最新的文档和/或规格。

本规范或其他英特尔文献中引用的具有订购号的文档副本可从以下网站获取：

英特尔公司  
<http://www.intel.com> 或  
拨打1-800-548-4725

版权所有©英特尔公司1999–2001

\* Third-party  
品牌和名称是其各自所有者的财产。

### 修订历史记录

修订	发布日期	评论
0.8年	1999年7月20日	初次修订
0.8亿	1999年8月10日	在数据结构中添加信息以支持拆分事务等。开始在操作模型中添加信息(第4章)。
0.9转/分1	1999年12月21日	对寄存器空间、数据结构和操作模型的完整定义进行了重大补充。
0.9平方厘米	2000年1月3日	根据0.9rc1反馈更新。
0.9克5	2000年1月12日	根据0.9 rc2反馈更新。内部,增量修订帐户rcX跳转。
0.9	2000年1月13日	0.9 rc5的编辑更改。
0.91	2000年7月6日	黄色封面发布的更新。还包括添加调试端口。
0.95 rc1	2000年9月26日	编辑更新。删除了第5章,传统键盘。
0.95	2000年10月11日	最终编辑更新。
0.96立方厘米1	2001年4月23日	编辑澄清;修复了在中断拆分事务期间处理CErr的错误,以及管理帧封装FS/LS中断、异步园区模式、FS/LS重新平衡和新的EHCI所有权信号量的新功能。
0.96瑞卡2	2001年5月30日	修复了许多打字错误,放宽了对泊车模式的要求,增加了FS/LS设置事务的CErr处理以及许多其他澄清。
0.96立方厘米	2001年6月12日	增加了处理全速等时IN数据流的要求(为了兼容性)。
0.96	2001年6月20日	最终编辑、分页等。
1.0转/分1	2002年1月31日	从行业反馈中积累的编辑变化。
1	2002年3月13日	被许可人审查期间累积的编辑变更。

**软件开发人员注意:**

EHCI规范的修订在编程接口中引入了新功能。软件不得试图在根据旧版本设计的主机控制器上使用规范最新版本中定义的功能。以下提供了每个修订版的新功能摘要。

**Revision Software 可见的新功能**

- |      |  |
|------|--|
| 0.96 | <ul style="list-style-type: none"> <li><input type="checkbox"/> 框架跨度横向节点(FSTN),见第3.7节。</li> <li><input checked="" type="checkbox"/> 再平衡锁定(I-Bit),见第3.6节、第4.12.2.5节</li> <li><input checked="" type="checkbox"/> 异步驻车模式,参见第2.2.4、2.3.1和4.10.3.2节。</li> <li><input checked="" type="checkbox"/> EHCI扩展功能,见第5节。</li> <li><input checked="" type="checkbox"/> 传统支持,请参阅第5.1节</li> </ul> |
|------|--|

**重要贡献者：**

---

约翰·S·霍华德	英特尔公司	Furuya信夫	NEC公司
达伦·艾布拉姆森	英特尔公司	詹姆斯·E·古齐亚克	朗讯
迈克尔·N·德尔	英特尔公司	布莱恩·利特	英特尔公司
约翰·加尼	英特尔公司	布雷德·赫斯勒	英特尔公司
卡尔蒂·瓦迪维卢	英特尔公司	克里斯·罗宾森	微软
Ken Stufflebeam公 司	康柏		

请通过电子邮件将意见发送至 :[ehcisupport@intel.com](mailto:ehcisupport@intel.com)

页面故意留空

# 目录

<b>1. 引言 .....</b>	<b>1</b>
<b>1.1 EHCI产品合规性 .....</b>	<b>2</b>
<b>1.2 体系结构概述 .....</b>	<b>2</b>
1.2.1 接口体系结构 .....	4
1.2.2 EHCI计划数据结构 .....	5
1.2.3 根集线器仿真 .....	5
<b>2. 寄存器接口 .....</b>	<b>7</b>
<b>2.1 PCI配置寄存器 (USB) .....</b>	<b>8</b>
2.1.1 PWRMGT公司□ PCI电源管理接口 .....	8
2.1.2 类□类代码寄存器 .....	9
2.1.3 美国海军基地□寄存器空间基址寄存器 .....	9
2.1.4 SBRN公司□串行总线发布号寄存器 .....	9
2.1.5 帧长度调整寄存器 (FLADJ) .....	10
2.1.6 端口唤醒能力寄存器 (PORTWAKECAP) .....	11
2.1.7 USBLEGSUP公司□USB传统支持扩展功能 .....	11
2.1.8 USBLEGCTLSTS公司□USB旧版支持控制/状态 .....	12
<b>2.2 主机控制器能力寄存器 .....</b>	<b>13</b>
2.2.1 管帽长度□能力寄存器长度 .....	13
2.2.2 h转换□主机控制器接口版本号 .....	14
2.2.3 HCSPARAMS公司□结构参数 .....	14
2.2.4 HCCPARAMS公司□能力参数 .....	15
2.2.5 HCSP-端口路由□配套端口路线说明 .....	16
<b>2.3 主机控制器操作寄存器 .....</b>	<b>17</b>
2.3.1 美国边境管理局□USB命令寄存器 .....	18
2.3.2 USBSTS公司□USB状态寄存器 .....	21
2.3.3 美国银行□USB中断启用寄存器 .....	22
2.3.4 FRINDEX公司□帧索引寄存器 .....	23
2.3.5 控制段□控制数据结构段寄存器 .....	24
2.3.6 牙周膜基□周期帧列表基址寄存器 .....	24
2.3.7 异步地址□当前异步列表地址寄存器 .....	25
2.3.8 配置标志□配置标志寄存器 .....	25
2.3.9 端口SC□端口状态和控制寄存器 .....	26
<b>3. 数据结构 .....</b>	<b>31</b>
<b>3.1 周期性帧列表 .....</b>	<b>31</b>
<b>3.2 异步列表队列头指针 .....</b>	<b>32</b>
<b>3.3 等时(高速)传输描述符(iTD) .....</b>	<b>33</b>
3.3.1 下一个链接点 .....	33

3.3.2 iTD事务状态和控制列表 .....	34
3.3.3 iTD缓冲区页指针列表 (加) .....	35
<b>3.4 拆分事务等时传输描述符 (siTD ) .....</b>	<b>36</b>
3.4.1 下一个链接点 .....	37
3.4.2 siTD端点能力/特性 .....	37
3.4.3 siTD传输状态 .....	38
3.4.4 siTD缓冲区指针列表 (加) .....	39
3.4.5 siTD反向链接指针 .....	40
<b>3.5 队列元素传输描述符 (qTD ) .....</b>	<b>40</b>
3.5.1 下一个qTD指针 .....	41
3.5.2 备用下一个qTD指针 .....	41
3.5.3 qTD代币 .....	42
3.5.4 qTD缓冲区页面指针列表 .....	45
<b>3.6 队列头 .....</b>	<b>46</b>
3.6.1 队列头水平链接指针 .....	46
3.6.2 端点功能/特性 .....	47
3.6.3 转移覆盖 .....	49
<b>3.7 周期帧跨度遍历节点 (FSTN ) .....</b>	<b>51</b>
3.7.1 FSTN正常路径指针 .....	51
3.7.2 FSTN反向链路指针 .....	52
<b>4. 操作模型 .....</b>	<b>53</b>
<b>4.1 主机控制器初始化 .....</b>	<b>53</b>
<b>4.2 端口路由和控制 .....</b>	<b>54</b>
4.2.1 通过EHCI配置 (CF) .....	Bit55的端口路由控制
4.2.2 通过PortOwner和Disconnect事件的端口路由控制 .....	56
4.2.3 端口路由状态机示例 .....	57
4.2.4 端口功率 .....	57
4.2.5 端口报告电流过大 .....	58
<b>4.3 暂停/恢复 .....</b>	<b>59</b>
4.3.1 端口挂起/恢复 .....	59
<b>4.4 时间表遍历规则 .....</b>	<b>61</b>
4.4.1 示例-保持微框架完整性 .....	62
<b>4.5 定期计划框架边界与总线框架边界 .....</b>	<b>64</b>
<b>4.6 定期计划 .....</b>	<b>66</b>
<b>4.7 使用iTDS67管理等时传输</b>	
4.7.1 iTDS67的主机控制器操作模型 .....	
4.7.2 iTDS69的软件操作模型 .....	
<b>4.8 异步时间表 .....</b>	<b>71</b>
4.8.1 将队列头添加到异步调度 .....	71
4.8.2 从异步调度中删除队列头 .....	72
4.8.3 空异步计划检测 .....	74
4.8.4 EOF74之前重新启动异步计划 .....	
4.8.5 异步计划遍历 :启动事件 .....	76
ii 4.8.6 Reclamation Status Bit (USBSTS Register) .....	77
	USB 2.0

4.9 Nak Counter的操作模型 .....	77
4.9.1    裸计数重新加载控制 .....	78
4.10 通过队列头管理控制/批量/中断传输 .....	79
4.10.1    提取队列头 .....	80
4.10.2    前进队列 .....	81
4.10.3    执行事务 .....	81
4.10.4    回写qTD86 .....	86
4.10.5    跟随队列头水平指针 .....	86
4.10.6    用于 .....	qTDs86数据流的缓冲区指针列表
4.10.7    将中断队列头添加到定期调度 .....	88
4.10.8    管理来自队列头的传输完全中断 .....	88
4.11 Ping控制 .....	88
4.12 拆分交易 .....	89
4.12.1    异步传输的拆分事务 .....	90
4.12.2    拆分事务中断 .....	92
4.12.3    同步拆分事务 .....	103
4.13 主机控制器暂停 .....	114
4.14 端口测试模式 .....	114
4.15 中断 .....	115
4.15.1    基于传输/事务的中断 .....	115
4.15.2    主机控制器事件中断 .....	117
<b>5. EHCl扩展能力 .....</b>	<b>121</b>
5.1 EHCl扩展功能 :操作系统到操作系统切换前同步 .....	121
<b>附录A. EHCl-PCI电源管理接口 .....</b>	<b>125</b>
A.1PCI电源管理寄存器接口 .....	125
A.1.1    电源状态转换 .....	126
A.1.2    电源状态定义 .....	126
A.1.3    PCI PME#信号 .....	127
<b>附录B. EHCl 64位数据结构 .....</b>	<b>129</b>
<b>附录C. DEBUG端口 .....</b>	<b>133</b>
C.1定位调试端口 .....	133
C.2使用调试端口字段 .....	134
C.3USB2调试端口寄存器接口 .....	134
C.3.1    调试端口控制寄存器 .....	135
C.3.2    USB PID寄存器 .....	137
C.3.3    数据缓冲器 .....	137
C.3.4    设备地址寄存器 .....	138
C.4    操作模型 .....	138
USB 2.0 .....	i

C.4.1	输出/设置事务 .....	139
C.4.2	IN交易 .....	139
C.4.3	调试软件启动 .....	139
C.4.4	查找调试外围设备 .....	140

**附录D. 高带 ..... 宽等时规则  
141**

## 1. 介绍

增强型主机控制器接口 (EHCI) 规范描述了通用串行总线 (USB) 修订版2.0的主机控制器的寄存器级接口。该规范包括对系统软件和主机控制器硬件之间的硬件/软件接口的描述。

本规范适用于硬件组件设计者、系统构建者和设备驱动程序 (软件) 开发人员。读者应熟悉通用串行总线规范2.0版。尽管进行了尽职调查 ,但本规范与USB 2.0规范之间可能存在冲突。USB 2.0规范优先于所有冲突问题。

EHCI规范的一些关键特征是 :

- ① **全面、稳健地支持所有USB 2.0功能。** 本规范描述了正确支持所有兼容USB 2.0低速、全速和高速设备的主机控制器。这包括新的USB 2.0功能 ,如USB 2.0集线器的拆分事务 ,以及协议的其他扩展 ,如用于高速OUT端点的新PING协议。
- ② **对全速和低速外圈设备的低风险支持。** EHCI规范通过集成 (使用 )USB的现有硬件和软件 ,为根端口上的所有三种设备速度提供支持  
1.1 主机控制器 ,用于支持连接到根端口的全速和低速设备。这使得EHCI能够专注于对高速操作的有效支持 ,而不必在复杂性或性能方面进行任何权衡以支持全速和低速设备。
- ③ **系统电源管理。** 当前的PC架构为积极的电源管理提供了无处不在的支持。USB是提供一致、连贯和稳健的用户体验的关键组件。如果实现包括PCI配置寄存器 ,则主机控制器需要实现PCI电源管理接口。
- ④ **为USB 1.1主机控制器问题提供简单、稳健的解决方案。** EHCI主机控制器规范包含了大量问题的解决方案 ,这些问题已被证明对USB主机控制器有问题。EHCI规范中解决的一些问题包括 :内存抖动、内存访问效率以及与cpu电源管理的冲突。EHCI体系结构为其体系结构提供了新的特定功能和优化 ,以解决遗留问题。
- ⑤ **优化以获得最佳内存访问效率。** EHCI利用一种独特的方法来减少执行USB事务所需的平均存储器访问次数。每个调度数据结构都经过优化以描述大型客户端数据缓冲区 ,从而最大限度地减少了内存占用和遍历调度的内存开销。
- ⑥ **最小化硬件复杂性。** EHCI为软件提供了一个简单的异步接口 ,以向主机控制器提供主机控制器用于在USB上执行事务的参数化工作项。该接口允许软件在主机控制器执行时异步地向接口添加工作 ,而不需要任何同步。该接口支持用于所有接口数据结构的简单硬件分散/收集方法。
- ⑦ **支持32位和64位寻址。** 在本规范的实施寿命内 ,预计EHCI控制器将越来越多地用于支持超过32位可寻址存储器空间的体系结构中。EHCI包括可选的接口扩展 ,支持多达64位的寻址。

本规范介绍了寄存器空间和调度接口数据结构的纯结构定义的两章。这些定义章节包含很少或根本不包含操作需求或使用模型。在定义章节之后 ,使用先前定义的寄存器和调度接口数据结构 ,详细描述了主机控制器的操作模型要求。以下列表总结了本规范的组织结构 :

- ① 第1.1节EHCI产品合规向读者介绍了EHCI合规计划。
- ② 第1.2节概述了EHCI主机控制器的体系结构。

- ① 第2章寄存器接口定义了EHCI主机控制器的寄存器空间。
- ① 第3章数据结构定义了EHCI主机控制器的调度数据结构。
- ① 第4章操作模型定义了EHCI主机控制器操作模型的细节。它侧重于主机控制器硬件的操作要求。它使用示例行为抽象来描述操作需求。
- ① 第5章EHCI扩展能力定义了EHCI特定的扩展能力特征 ,还包含了对每个定义的扩展能力的描述。
- ① 附录A EHCl PCI电源管理接口定义了EHCI主机控制器的PCI电源管理界面的详细信息。
- ① 附录B EHCl 64位数据结构定义了第3章中定义的数据结构的64位版本。
- ① 附录C调试端口定义了到可选调试端口的接口。
- ① 附录D高带宽同步规则列举了主机控制器执行高带宽同步事务所需的行为。

## 1.1 EHCI产品合规性

USB增强型主机控制器接口规范的采用者和贡献者已签署《USB增强型主控制器接口规范-贡献者协议》,以便获得使用和实施本规范的许可。本贡献者协议为贡献者和采用者提供了英特尔及其他采用者和贡献者对其符合USB增强型主机控制器接口规范的产品的某些知识产权的互惠免版税许可。采用者和贡献者可以通过英特尔定义的测试程序证明其符合本规范。

## 1.2 体系结构概述

USB主机系统由许多硬件和软件层组成。图1-1展示了主机系统中协同工作以支持USB2.0的构建块层的概念框图。

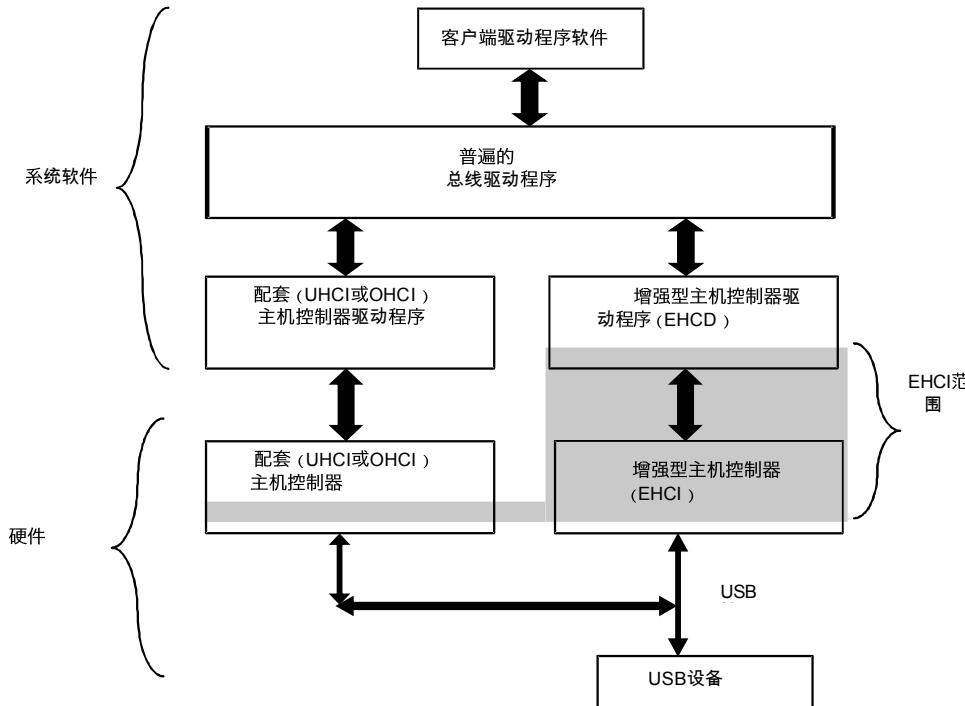


图1-1。通用串行总线,修订版2.0系统框图

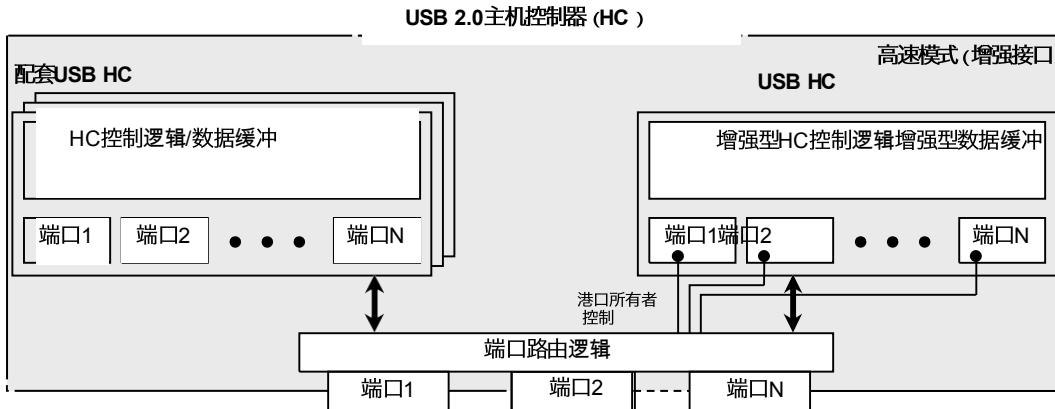
组件层包括：

- ① **客户端驱动程序软件。**该软件在与特定USB设备相对应的主机PC上执行。客户端软件通常是一部分或与USB设备一起提供。
- ② **USB驱动程序(USBD)。**USBD是一种系统软件总线驱动程序,它抽象了特定操作系统的特定主机控制器驱动程序的细节。
- ③ **主机控制器驱动程序(xHCD)。**xHCD提供特定主机控制器硬件和USBD之间的软件层。主机控制器驱动程序的细节取决于特定的主机控制器硬件寄存器接口定义。
- ④ **主机控制器(xHC)。**主机控制器是主机控制器的特定硬件实现。高速USB 2.0功能有一种主机控制器规范,全速和低速主机控制器有两种规范。**通用主机控制器接口(UHCI)**或**用于USB的开放式主机控制器接口**是两个行业标准的USB 1.1主机控制器接口。
- ⑤ **USB设备。**这是一种执行有用的最终用户功能的硬件设备。与USB设备的交互从应用程序通过软件和硬件层流向USB设备。

USB 2.0主机控制器包括一个高速模式主机控制器和0个或多个USB 1.1主机控制器(见图1-2)。高速主机控制器实现了EHCI接口。它用于与连接到USB 2.0主机控制器根端口的高速模式设备之间的所有高速通信。本规范允许与连接到USB 2.0主机控制器的根端口的全速和低速设备进行通信,由配套的USB 1.1主机控制器提供。如果实现不包括配套主机控制器,则主机控制器必须包括永久连接到实现计划使用的每个EHCI端口的高速设备。EHCI控制器无法与全速或低速设备一起工作。

这种架构允许USB 2.0主机控制器提供USB功能,只要常驻操作系统中至少支持USB 1.1软件即可。当操作系统中同时提供USB 1.1和EHCI软件时,即可提供完整的USB 2.0功能。端口收发器路由逻辑是提供这种灵活操作环境的关键。路由逻辑的初始状态(见图1-2)

取决于软件是否已配置EHCI控制器。一旦EHCD驱动程序配置了EHCI控制器，如果连接的设备不是高速设备，它就可以专门将收发器释放到伴随主机控制器端口寄存器。当操作系统不支持EHCI控制器时，端口默认路由到配套主机控制器，并且保留对全速和低速设备的现有USB支持。



**图1-2。USB 2.0主机控制器**

配套主机控制器(cHC)可以是任何USB 1.1主机控制器(例如OHCI或UHCI)。配套主机控制器始终管理连接到根端口的全速和低速USB设备。cHC不知道高速模式主机控制器。它们可以集成到USB 2.0主机控制器中，无需任何修改。

高速设备总是路由到EHCI主机控制器(eHC)并由其控制。运行和配置时，eHC是所有根端口的默认所有者。eHC及其驱动程序最初检测所有设备连接。它在每个端口寄存器中都有可见的额外控制位，用于管理路由逻辑。例如：如果连接的设备不是高速设备，则eHC驱动程序将端口的所有权(以及设备的控制权)释放给配套主机控制器。对于该端口，枚举从初始连接检测点开始，设备在cHC下枚举。否则，eHC保留端口的所有权，并且设备在eHC下完成枚举。

本规范不包括通用主机控制器接口(UHCI)或USB开放式主机控制器接口(OHCI for USB)。本规范定义了增强型主机控制器接口的寄存器和调度接口。

### 1.2.1 接口体系结构

EHCI接口定义了三个接口空间(见图1-3)：

- ① **PCI配置空间。**如果实现包括PCI寄存器，则它们用于系统组件枚举和PCI电源管理。
- ② **寄存器空间。**具体实施的参数和能力，加上操作控制和状态寄存器。这个空间，通常称为I/O空间，必须实现为内存映射的I/O空间。
- ③ **安排接口空间。**这通常是由eHC驱动程序为周期性和异步调度分配和管理的内存。

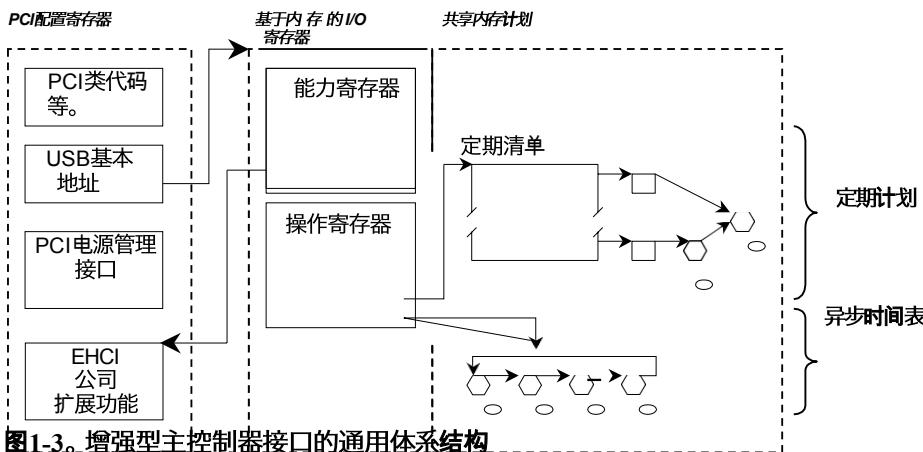


图1-3 增强型主控制器接口的通用体系结构

EHCI提供对两类传输类型的支持：异步和周期性。周期性传输类型包括同步传输和中断传输。异步传输类型包括控制和批量传输。图1-3说明了EHCI计划接口为每种传输类型提供了单独的计划。

周期性时间表基于面向时间的帧列表，该列表表示主控制器工作项的时间滑动窗口。所有的同步传输和中断传输都是通过周期性时间表提供服务的。异步调度是一个简单的调度工作项循环列表，它为所有异步传输提供了循环服务机会。

EHCI主机控制器接口允许软件启用或禁用每个计划。这允许系统软件在SOF流量的情况下保持USB活动，但当两个调度都被禁用时，主机控制器将不会访问调度空间。这使主机控制器在大多数实现中无法访问主存储器，并使移动系统能够更好地管理CPU功率（请参见第4.13节）。

### 1.2.2 EHCI计划数据结构

EHCI使用简单的缓冲区排队数据结构来管理所有中断、批量和控制传输类型。排队数据结构提供自动的、按顺序的数据传输流。软件可以异步地将数据缓冲区添加到队列中并保持流式传输。USB定义的短分组语义在所有处理边界条件下都得到完全支持，而无需软件干预。

全速和低速非同步端点的拆分事务使用相同的数据结构进行管理。拆分事务协议是作为高速执行模型的简单扩展进行管理的。

高速和全速等时传输使用专用（和不同）接口数据结构进行管理。这些数据结构针对每个数据有效载荷的可变性和等时传输类型的面向时间的特性进行了优化。

### 1.2.3 根集线器仿真

USB总线的主机控制器是实现根集线器所必需的。操作寄存器空间包含端口寄存器，端口寄存器包含管理USB规范中每个端口所需的最低硬件状态和控制。主机控制器遍历EHCI调度并遇到导致主机控制器执行USB事务的工作项。这些事务通过所有启用的根端口广播到连接的下游USB设备。

端口寄存器为系统软件提供根据USB规范2.0版操作端口所需的控制和状态信息。支持的功能包括：检测

设备连接、断开连接、执行设备重置、操作端口电源和管理端口电源管理功能。

系统软件应该为USB系统软件堆栈提供一个抽象，允许根集线器端口被系统操纵，就好像它们是外部集线器上的端口一样。

## 2. 寄存器接口

增强型USB主机控制器包含两组软件可访问的硬件寄存器：内存映射主机控制器寄存器和可选PCI配置寄存器。请注意，PCI配置寄存器仅用于实现主机控制器的PCI设备。

- ① **PCI配置寄存器(用于PCI设备)**。除了普通的PCI头、电源管理和设备专用寄存器外，PCI配置空间中还需要两个寄存器来支持USB。普通PCI头和设备专用寄存器超出了本文档的范围(CLASSC寄存器如本文档所示)。注意，HCD不与PCI配置空间交互。此空间仅由PCI枚举器用于识别USB主机控制器，并分配适当的系统资源。
- ② **内存映射的USB主机控制器寄存器**。这个寄存器块被内存映射到不可缓存的内存中(见图1-3)。此内存空间必须从DWord(32位)边界开始。该寄存器空间分为两部分：一组只读能力寄存器和一组读/写操作寄存器。表2-1描述了每个寄存器空间。

请注意，主机控制器不需要支持用于访问存储器映射寄存器空间的独占访问机制(例如PCI LOCK)。因此，如果软件尝试对主机控制器存储器映射寄存器空间的独占访问机制，则结果是未定义的。

**表2-1。增强型接口寄存器集**

抵消	寄存器集	解释
0到N-1	能力寄存器(第2.2节)	功能寄存器指定主机控制器实现的限制、限制和功能。这些值被用作主机控制器驱动程序的参数。
N至N+M-1	操作寄存器(第2.3节)	操作寄存器由系统软件用来控制和监视主机控制器的操作状态。

在本说明书的该修订版中定义的保留位可以在以后的修订版中分配。软件不应假定保留位总是零，并且在写入可修改寄存器时应保留这些位。以下符号用于描述寄存器访问属性：

**RO只读**。如果寄存器是只读的，则写入没有任何作用。

**仅WO写入**。如果一个寄存器只写，则读取会为所有位位置返回一个零。

**R/W 读/写**。具有此属性的寄存器可以被读取和写入。请注意，某些读/写寄存器中的个别位可能是只读的。

**R/WC读/写清除**。具有此属性的寄存器位可以被读取和写入。然而，写入1会清除(设置为0)相应的位，而写入0则没有效果。

辅助井中的寄存器在与核心井中的寄存器不同的条件下被重置。在以下情况下，辅助阱存储器空间寄存器被初始化为其默认值：

- ① 辅助电源井的初始通电，或
- ② *HCReset*中的值为1b(见第2.3.1节)

在以下情况下，核心阱内存空间寄存器被初始化为其默认值：

- ① 断言系统(核心井)硬件重置，或
- ② *HCReset*中的值为1b，或
- ③ 从PCI电源管理D3hot状态转换到D0状态

在辅助电源阱中实现的PCI配置空间寄存器在与核心阱中的寄存器不同的条件下被复位。在以下情况下，辅助阱、配置空间寄存器被初始化为其默认值：

- ① 辅助电源井的初始通电

在以下情况下，核心阱PCI配置空间寄存器被初始化为其默认值：

- ① 断言系统（核心阱）硬件重置，或
- ② 从PCI电源管理D3hot状态到D0状态的转换这些复位条件的异常将在相关的寄存器部分中定义。

## 2.1 PCI配置寄存器(USB)

表2-2列出了EHCI控制器的PCI配置空间寄存器。以下小节描述了有关定义的每组寄存器的详细信息。

表2-2。PCI配置空间寄存器

配置偏移	助记符	登记	动力阱	注册访问
00~08小时	□	根据需要为特定PCI设备注册实现	果心	□
09~10小时	类	类别代码	果心	罗
0厘米~0华氏度	□	根据需要为特定PCI设备注册实现	果心	□
10~13小时	美国海军基地	基址到内存映射的主机控制器寄存器空间	果心	R/W
14~15小时	□	根据需要为特定PCI设备注册实现	果心	□
60小时	SBRN公司	串行总线发布号	果心	罗
第61小时	亚麻布	帧长度调整寄存器	辅助	R/W
62~63小时	舷窗尾水帽	端口唤醒功能寄存器(可选)	辅助	R/W
64~自由流动资金	□	根据需要为特定PCI设备注册实现	果心	□
EECP+0h <sup>1</sup>	USBLEGSUP公司	USB传统支持EHCI扩展功能寄存器	辅助	滚装
EECP+4小时	USBLEGCTLSTS公司	USB旧版支持控制和状态寄存器	辅助	转水，转水

<sup>1</sup> EECP字段在HCCRAMS寄存器中，参见第2.2.4节。

### 2.1.1 PWRMGT公司 □ PCI电源管理接口

需要EHCI主机控制器实现来实现PCI总线电源管理接口规范修订版1.1中定义的PCI电源管理寄存器。有关PCI电源管理的EHCI操作要求，请参阅附录A。

### 2.1.2 类□类代码寄存器

地址偏移 :0910Bh默认值 :

0C0320

h属性 :

RO

大小 :

24位

该寄存器包含与子类代码和基类代码定义相关的设备编程接口信息。该寄存器还标识基类代码和与基类代码相关的函数子类。

一点	描述
23时16分	基类代码 ( <b>BASEC</b> )。0Ch=串行总线控制器。
15时8分	子类代码 ( <b>SCC</b> )。03h=通用串行总线主机控制器。
7:0分	编程接口 ( <b>PI</b> )。20h=符合本规范的USB 2.0主机控制器。

### 2.1.3 美国海军基地□寄存器空间基址寄存器

地址偏移 :1013小时

默认值 :实现 相关属性 : R/W

大小 : 32位

此寄存器包含与D字对齐的内存映射主机控制器寄存器的基址。此寄存器中可写位的数量决定了所需内存空间窗口的实际大小。本规范规定了最低要求。各个实现方式可能有所不同。

一点	描述
31:8	基本地址□转/转。对应于存储器地址信号[31:8]。
7点3分	保留。□ RO。这些位是只读的，并且硬连接到零。
2:1	类型□ RO。此字段有两个有效值： 价值    含义 00b只能映射到32位寻址空间(推荐)。10b    可以被映射到64位寻址空间 中。
0	保留□ RO。此位是只读的，并硬接线至零。

### 2.1.4 SBRN公司□串行总线释放号寄存器

地址偏移 :60小时

默认值 :参见 下面的描述属性 :

RO

大小 : 8位

此寄存器包含通用串行总线规范的发布版本，此通用串行总线主机控制器模块符合该规范。

一点	描述
7:0分	串行总线规格发布号。保留所有其他组合。 Bits[7:0]    释放号 20h            版本2.0

### 2.1.5 帧长度调整寄存器 (FLADJ )

地址偏移 : 61h默认  
值 : 20h属性 : R/W  
大小 : 8位

此寄存器位于辅助电源井中。此功能用于调整产生驱动SOF计数器的时钟的时钟源的任何偏移。当一个新的值被写入这六个比特时，帧的长度被调整。其初始编程值取决于硬件USB时钟的准确性，并由系统BIOS初始化。只有当USBSTS寄存器中的HCHalted位为1时，才应修改此寄存器。主机控制器运行时更改此寄存器的值会产生未定义的结果。除非默认值或BIOS编程值不正确，或者系统从挂起状态返回时正在恢复寄存器，否则USB系统软件不应对其进行重新编程。

一点	描述																				
7点6分	保留。这些位是为将来使用而保留的，并且应读取为零。																				
5点0分	<p><b>帧长度计时值。</b>该寄存器的每一个十进制值变化对应16个高-速度位时间。SOF循环时间(生成SOF的SOF计数器时钟周期数微帧长度)等于59488+的值。默认值为十进制32(20h)，这给出了60000的SOF循环时间。</p> <table> <thead> <tr> <th>框架长度 (#高速位时间) (十进制)</th> <th>FLADJ值 (十进制)</th> </tr> </thead> <tbody> <tr> <td>594880 (</td> <td>00小时 )</td> </tr> <tr> <td>595041 (</td> <td>01小时 )</td> </tr> <tr> <td>595202 (02小时 )</td> <td></td> </tr> <tr> <td>...</td> <td></td> </tr> <tr> <td>5998431 (1小时 )</td> <td></td> </tr> <tr> <td>6000032 (</td> <td>20小时 )</td> </tr> <tr> <td>...</td> <td></td> </tr> <tr> <td>6048062 (3小时 )</td> <td></td> </tr> <tr> <td>6049663 (3小时 )</td> <td></td> </tr> </tbody> </table>	框架长度 (#高速位时间) (十进制)	FLADJ值 (十进制)	594880 (	00小时 )	595041 (	01小时 )	595202 (02小时 )		...		5998431 (1小时 )		6000032 (	20小时 )	...		6048062 (3小时 )		6049663 (3小时 )	
框架长度 (#高速位时间) (十进制)	FLADJ值 (十进制)																				
594880 (	00小时 )																				
595041 (	01小时 )																				
595202 (02小时 )																					
...																					
5998431 (1小时 )																					
6000032 (	20小时 )																				
...																					
6048062 (3小时 )																					
6049663 (3小时 )																					

### 2.1.6 端口唤醒能力寄存器 (PORTWAKECAP )

地址偏移 :62h  
 默认值 :实现 相关属性 : R/W  
 大小 : 16位

此寄存器是可选的。当实现时 ,该寄存器处于辅助电源阱中。此寄存器的预期用途是建立一个关于唤醒事件将使用哪些端口的策略。掩码中的比特位置1-15对应于在当前EHCI控制器上实现的物理端口。一位一表示连接在端口下方的设备可以被启用为唤醒设备 ,并且端口可以被启用用于断开/连接或过电流事件作为唤醒事件。这是一个仅限信息的掩码寄存器。此寄存器中的位不会影响EHCI主机控制器的实际操作。系统特定策略可以通过BIOS将该寄存器初始化为系统特定值来建立。  
 系统软件在启用设备和端口进行远程唤醒时使用此寄存器中的信息。

一点	描述
15:0	端口唤醒功能掩码。该寄存器的位位置0指示寄存器是否实现。位位置0中的1表示寄存器已实现。比特位置1到15对应于在该主机控制器上实现的物理端口。例如 ,比特位置1对应于端口1 ,位置2对应于端口2 ,等等。

### 2.1.7 USBLEG SUP公司□ USB传统支持扩展功能

偏移 : EECP+00h  
 默 认值 实现 相关属性 RO ,  
 R/W  
 大小 : 32位

此寄存器是EHCI扩展功能寄存器。它包括一个特定的功能部分和指向下一个EHCI扩展功能的指针。预操作系统软件 (BIOS) 和操作系统使用此寄存器来协调EHCI主机控制器的所有权。详见第5.1节。

表2-3。USBLEG SUP公司□ USB传统支持扩展功能

一点	描述
31分25秒	保留。这些位是保留的 ,必须设置为零。
24	<b>HC OS拥有的信号量□ 转/转。</b> 0=默认值。系统软件将此位设置为请求EHCI控制器的所有权。当该位读取为1并且 <b>HC BIOS拥有的信号量</b> 位读取为0时 ,获得所有权。
23时17分	保留。这些位是保留的 ,必须设置为零。
16	<b>HC BIOS拥有的信号量□ 转/转。</b> 0=默认值。BIOS设置此位以建立EHCI控制器的所有权。系统BIOS将响应系统软件对EHCI控制器所有权的请求将该位设置为零。
15时8分	下一个 <b>EHCI扩展功能指针□ RO</b> 。此字段指向下一个扩展能力指针的PCI配置空间偏移。值00h表示扩展能力列表的结束。
7:0分	<b>能力ID□ RO</b> 。此字段标识扩展功能。值01h表示该功能为传统支持。此扩展功能需要一个额外的32位寄存器用于控制/状态信息 ,并且此寄存器位于偏移量EECP+04h处。

### 2.1.8 USBLEGCTLSTS公司□ USB旧版支持控制/状态

偏移 : EECP+04h默认  
 值 00000000h  
 AttributeRO、R/W、R/WC  
 大小 : 32位

预操作系统 (BIOS) 软件使用此寄存器为其需要跟踪的每个EHCI/USB事件启用SMI。该寄存器的位[21:16]只是USBSTS寄存器[5:0]的影子位。

表2-4。USBLEGCTLSTS公司□ USB旧版支持控制/状态

一点	描述
31	酒吧里的SMI□ R/WC。0=默认值。每当写入基本地址寄存器(BAR)时，该位被设置为1。
30	SMI on PCI命令□ R/WC。0=默认值。每当写入PCI命令寄存器时，此位都设置为1。
29	SMI关于操作系统所有权变更□ R/WC。0=默认值。每当USBLEGSUP寄存器中的HC OS Owned Semaphore位从1转换为0或从0转换为1时，该位被设置为1
28分22秒	保留。这些位是保留的，应该为零。
21	异步推进的SMI□ RO.0=默认值。USBSTS寄存器中异步前进中断位的影子位，定义见第2.3.2节。 要将该位设置为零，系统软件必须向USBSTS寄存器中的异步前进中断位写入一。
20	主机系统上的SMI错误□ RO.0=默认值。USBSTS寄存器中主机系统错误位的影子位，请参阅第2.3.2节，了解与该位设置为1相关的事件的定义和影响。 要将此位设置为零，系统软件必须将一写入USBSTS寄存器中的主机系统错误位。
19	SMI关于帧列表滚动□ RO.0=默认值。USBSTS寄存器中帧列表翻转位的影子位，定义见第2.3.2节。 要将该位设置为零，系统软件必须向USBSTS寄存器中的帧列表滚动位写入一。
18	端口更改检测上的SMI□ RO.0=默认值。USBSTS寄存器中端口更改检测位的影子位，定义见第2.3.2节。 要将此位设置为零，系统软件必须将一写入USBSTS寄存器中的端口更改检测位。
17	USB上的SMI错误□ RO.0=默认值。USBSTS寄存器中USB错误中断(USBERRINT)位的影子位，定义见第2.3.2节。 要将该位设置为零，系统软件必须向USBSTS寄存器中的USB错误中断位写入一。
16	□ RO.0=默认值。USBSTS寄存器中USB中断(USBINT)位的影子位，定义见第2.3.2节。 要将该位设置为零，系统软件必须向USBSTS寄存器中的USB中断位写入一。
15	启用BAR上的SMI□ 转/转。0=默认值。当此位为1并且BAR上的SMI为1时，主机控制器将发出SMI。
14	启用PCI上的SMI命令□ 转/转。0=默认值。当此位为1并且PCI命令上的SMI为1时，主机控制器将发出SMI。
13	启用操作系统所有权上的SMI□ 转/转。0=默认值。当此位为1且操作系统所有权更改位为1时，主机控制器将发出SMI。
12:6	保留。这些位是保留的，应该为零。

表2-4。USBLEGCTLSTS公司□ USB旧版支持控制/状态(续)

一点	描述
5.	<b>启用异步高级SMI</b> □ 转/转。0=默认值。当此位为1，并且此寄存器中异步前进上的SMI位(上面)为1时，主机控制器将立即发出SMI。 <sup>E</sup>
4.	<b>主机系统上的SMI错误启用</b> □ 转/转。0=默认值。当此位为1，并且此寄存器中的主机系统错误上的SMI位(以上)为1时，主机控制器将立即发出SMI。 <sup>E</sup>
3.	<b>启用SMI帧列表滚动</b> □ 转/转。0=默认值。当该位为1，并且该寄存器中的帧列表翻转上的SMI位(上面)为1时，主机控制器将立即发出SMI。
2.	<b>启用端口更改时的SMI</b> □ 转/转。0=默认值。当此位为1，并且此寄存器中的端口更改检测上的SMI位(上图)为1时，主机控制器将立即发出SMI。 <sup>E</sup>
1.	<b>USB上的SMI错误启用</b> □ 转/转。0=默认值。当此位为1，并且此寄存器中的SMI on USB Error位(上面)为1时，主机控制器将立即发出SMI。 <sup>E</sup>
0	<b>USB SMI启用</b> □ 转/转。0=默认值。当此位为1，并且此寄存器中USB上的SMI完整位(如上)为1时，主机控制器将立即发出SMI。 <sup>E</sup>

注意

事项：<sup>A</sup> 对于所有启用寄存器位，1=启用，0=禁用<sup>B</sup> SMI—系统管理中断<sup>C</sup> BAR—基本地址寄存器<sup>D</sup> MSE—内存空间启用<sup>E</sup> SMI与中断阈值无关

## 2.2 主机控制器能力寄存器

这些寄存器规定了主机控制器实现的限制、限制和功能。

表2-5。增强型主机控制器能力寄存器

抵消	大小	助记符	动力井	注册机构名称
00小时	1.	管帽长度	果心	能力寄存器长度
01小时	1.	保留	果心	N/A
02小时	2.	h转换	果心	接口版本号
04小时	4.	HCSPARAMS公司	果心	结构参数
08小时	4.	HCCPARAMS公司	果心	能力参数
0小时	8.	HCSP-端口路由	果心	配套端口路由描述

### 2.2.1 管帽长度□ 功能寄存器长度

地址： 基地+ (00h)  
 默认值 实现相关属性： RO  
 大小： 8位

此寄存器用作偏移量，添加到寄存器基极，以查找运算寄存器空间的开始。

## 2.2.2 h转换□ 主机控制器接口版本号

地址 : Base+ (02h)  
 默认值 : 0100h  
 属性 : AttributeRO  
 大小 : 16位

这是一个两字节寄存器，包含此主机控制器支持的EHCI修订号的BCD编码。该寄存器的最高有效字节表示主要修订，最低有效字节表示次要修订。

## 2.2.3 HCSPARAMS公司□ 结构参数

地址 : 基址+ (04h)  
 默认值 : 实现相关属性RO  
 大小 : 32位

这是一组作为结构参数的字段：下游端口数量等。

表2-6.HCSPARAMS□ 主机控制器结构参数

一点	描述
31分24秒	保留。这些位是保留的，应该设置为零。
23时20分	调试端口号。可选择的此寄存器标识哪个主机控制器端口是调试端口。该值是调试端口的端口号（基于一）。此字段中的非零值表示存在调试端口。此寄存器中的值不得大于N_ports（请参阅下文）。
19时17分	保留。这些位是保留的，应该设置为零。
16	端口指示灯（P_INDICATOR）。此位指示端口是否支持端口指示器控制。当该位为1时，端口状态和控制寄存器包括用于控制端口指示符的状态的读/写字段。端口指示器控制字段的定义见第2.3.9节。
15时12分	配套控制器（N_CC）的数量。此字段指示与此USB 2.0主机控制器相关联的配套控制器的数量。 此字段中的零表示没有配套主机控制器。不支持端口所有权移交。主机控制器根端口上仅支持高速设备。 此字段中大于零的值表示存在配套的USB 1.1主机控制器。支持港口所有权移交。主机控制器根端口支持高速、全速和低速设备。
11点8分	每个配套控制器（N_PCC）的端口数。此字段指示每个配套主机控制器支持的端口数。它用于向系统软件指示端口路由配置。 例如，如果N_PORTS具有值6并且N_CC具有值2，则N_PCC可以具有值3。约定是，假定第一N_PCC端口被路由到伴随控制器1，下一N_PCC端口到伴随控制器2，等等。在先前的示例中，N_PCC可以是4，其中前4个被路由到伴随控制器1，后两个被路由至伴随控制器2。 此字段中的数字必须与N_PORTS和N_CC一致。

表2-6.HCSPARAMS□ 主机控制器结构参数 (续 )

一点	描述						
7.	<p>端口路由规则。此字段指示此实现所使用的方法 ,用于如何将所有端口映射到配套控制器。该字段的值具有以下解释 :</p> <table> <thead> <tr> <th>价值</th> <th>含义</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>第一个N_PCC端口被路由到编号最低的功能伴随主机控制器 ,下一个N_PCC港口被路由到下一个最低功能伴随控制器 ,依此类推。</td> </tr> <tr> <td>1</td> <td>端口路由由HCSP-PORTROUTE数组的前N_PORTS元素显式枚举。</td> </tr> </tbody> </table>	价值	含义	0	第一个N_PCC端口被路由到编号最低的功能伴随主机控制器 ,下一个N_PCC港口被路由到下一个最低功能伴随控制器 ,依此类推。	1	端口路由由HCSP-PORTROUTE数组的前N_PORTS元素显式枚举。
价值	含义						
0	第一个N_PCC端口被路由到编号最低的功能伴随主机控制器 ,下一个N_PCC港口被路由到下一个最低功能伴随控制器 ,依此类推。						
1	端口路由由HCSP-PORTROUTE数组的前N_PORTS元素显式枚举。						
6点5分	保留。这些位是保留的 ,应该设置为零。						
4.	港口电力控制 (PPC )。此字段指示主机控制器实现是否包括端口功率控制。此位中的1表示端口具有端口电源开关。此位中的零表示端口没有端口电源开关。该字段的值影响每个端口状态和控制寄存器中端口功率字段的功能 (见第2.3.8节 )。						
3点	<p>N_端口。此字段指定在此主机控制器上实现的物理下行端口的数量。该字段的值决定了操作寄存器空间中可寻址的端口寄存器数量 (见表2-8 )。有效值在1H到FH的范围内。</p> <p>此字段中的零未定义。</p>						

## 2.2.4 HCCPARAMS公司□ 能力参数

地址 : 基地+ (08h )  
 默认值 实现相关属性RO  
 大小 : 32位

多模式控制 (时基位功能 ), 寻址能力

表2-7。HCCPARAMS公司□ 主机控制器能力参数

一点	描述
31分16秒	保留。这些位是保留的 ,应该设置为零。
15时8分	<b>EHCI扩展功能指针 (EECP )。</b> 默认值=取决于实施。此可选字段指示是否存在功能列表。值00h表示未实现扩展功能。该寄存器中的非零值指示第一EHCI扩展能力在PCI配置空间中的偏移。如果实现指针值以保持为此类设备定义的PCI头的一致性 ,则指针值必须为40h或更大。
7点4分	<b>等时调度阈值。</b> 默认值=取决于实现。该字段指示相对于执行主机控制器的当前位置 ,软件可以可靠地更新等时进度表的位置。当位[7]为零时 ,最低有效3位的值指示在刷新状态之前主机控制器可以保持一组等时数据结构 (一个或多个 )的微帧的数量。当比特[7]为1时 ,则主机软件假定主机控制器可以缓存整个帧的等时数据结构。有关软件如何使用这些信息来安排等时传输的详细信息 ,请参阅第4.7.2.1节。
3.	保留。此位是保留的 ,应设置为零。

表2-7。HCCPARAMS公司□主机控制器能力参数(续)

一点	描述
2.	<b>异步调度园区功能。</b> 默认值=取决于实施。如果此位设置为1，则主机控制器支持异步调度中高速队列头的泊车功能。通过使用USBCMD寄存器中的 <i>Asynchronous Schedule Park Mode Enable</i> (异步计划驻车模式启用)和 <i>Asynchronousous Schedule</i> 驻车模式计数字段，可以禁用或启用该功能，并将其设置为特定级别。
1.	<b>可编程帧列表标志。</b> 默认值=取决于实施。如果此位设置为零，则系统软件必须使用此主机控制器的1024个元素的帧列表长度。USBCMD寄存器帧列表大小字段是只读寄存器，应设置为零。 如果设置为1，则系统软件可以指定并使用较小的帧列表，并通过USBCMD寄存器帧列表大小字段配置主机控制器。帧列表必须始终在4K页面边界上对齐。这一要求确保了帧列表在物理上总是连续的。
0	<b>64位寻址能力<sup>1</sup>。</b> 该字段记录了该实现的寻址范围能力。该字段的值决定软件应使用第3节(32位)中定义的数据结构还是附录B(64位)中的数据结构。 该字段的值具有以下解释： 0b使用32位地址内存指针的数据结构1使用64位地址内存指示器 的 数据结构

[1] 这与USBBASE地址寄存器映射控制没有紧密耦合。64位寻址能力位指示主机控制器是否可以生成64位地址作为主机。USBBASE寄存器表示主机控制器只需要将32位地址作为从机进行解码。

## 2.2.5 HCSP-端口路由□ 配套端口路由描述

地址：	基 地+ (0Ch)
默认值	实现相关属性RO
大小：	60位

只有当HCSPARAMS寄存器中的“端口路由规则”字段设置为1时，此可选字段才有效。

第4.2节详细描述了在PCI空间内组织配套主机控制器和EHCI主机控制器的规则。该字段用于允许主机控制器实现明确地描述每个实现的端口被映射到哪个伴随主机控制器。

该字段是一个15个元素的半字节数组(每个4位是一个数组元素)。每个阵列位置与主机控制器提供的物理端口一一对应(例如PORTROUTE[0]对应第一个PORTSC端口，PORTROUTE[1]对应第二个PORTSC端口等)。每个元素的值指示该端口路由到哪个配套主机控制器。只有前N PORTS元素具有有效信息。值为零表示端口路由到编号最低的功能配套主机控制器。值为1表示端口被路由到下一个编号最低的功能配套主机控制器，依此类推。

### 2.3 主机控制器操作寄存器

本节定义了增强型主机控制器操作寄存器。这些寄存器位于功能寄存器之后(见第2.1.7节)。操作寄存器基址必须与D字对齐，并通过将第一个功能寄存器(CAPLENGTH, 第2.2.1节)中的值与增强型主机控制器寄存器地址空间的基址相加来计算。所有寄存器的长度都是32位。软件应该只使用DWord访问来读取和写入这些寄存器。

这些寄存器分为两组。在核心功率阱中实现在地址00h到3Fh处的第一组。到所实现的寄存器空间末端的地址40h处的第二组在辅助功率阱中实现。

表2-8。主机控制器操作寄存器

抵消	助记符	注册机构名称	动力井	部分
00小时	美国边境管理局	USB命令	果心	2.3.1
04小时	USBSTS公司	USB状态	果心	2.3.2
08小时	美国银行	USB中断启用	果心	2.3.3
0小时	FRINDEX公司	USB帧索引	果心	2.3.4
10小时	控制段	4G段选择器	果心	2.3.5
14小时	牙周膜基	帧列表基地址	果心	2.3.6
18小时	异步地址	下一个异步列表地址	果心	2.3.7
1C-3Fh型	保留		果心	
40小时	配置标志	配置的标志寄存器	辅助	2.3.8
44小时	端口sc (1-N_PORTS )	端口状态/控制	辅助	2.3.9

### 2.3.1 美国边境管理局□ USB命令寄存器

地址 : 作战基地+ (00h)

默认值 : 00080000h (如果异步计划园区能力为一 , 则为00080B00h )AttributeRO , R/W (取决于字段 ), WO

尺寸 32位

命令寄存器指示要由串行总线主机控制器执行的命令。写入寄存器会导致执行命令。

表2-9。USBCMD-USB命令寄存器位定义

一点	描述																		
31分24秒	保留。这些位是保留的 , 应该设置为零。																		
23时16分	<p><b>中断阈值控制</b>□ 转/转。默认08h。系统软件使用该字段来选择主机控制器发出中断的最大速率。下面定义了唯一有效的值。如果软件将无效值写入该寄存器 , 则结果未定义。</p> <table> <thead> <tr> <th>值</th> <th>最大中断间隔</th> </tr> </thead> <tbody> <tr> <td>00h</td> <td>保留</td> </tr> <tr> <td>01h</td> <td>微型框架</td> </tr> <tr> <td>02h</td> <td>微型相框</td> </tr> <tr> <td>04h</td> <td>微帧</td> </tr> <tr> <td>08h</td> <td>8微帧 (默认值 , 相当于1毫秒 )</td> </tr> <tr> <td>10h</td> <td>16微帧 (2毫秒 )</td> </tr> <tr> <td>20h</td> <td>32微帧 (4毫秒 )</td> </tr> <tr> <td>40h</td> <td>64微帧 (8毫秒 )</td> </tr> </tbody> </table> <p>有关受此寄存器影响的中断 , 请参阅第4.15节。此寄存器中的任何其他值都会产生未定义的结果。</p> <p>当HCHalted位等于零时 , 对该位的软件修改会导致未定义的行为。</p>	值	最大中断间隔	00h	保留	01h	微型框架	02h	微型相框	04h	微帧	08h	8微帧 (默认值 , 相当于1毫秒 )	10h	16微帧 (2毫秒 )	20h	32微帧 (4毫秒 )	40h	64微帧 (8毫秒 )
值	最大中断间隔																		
00h	保留																		
01h	微型框架																		
02h	微型相框																		
04h	微帧																		
08h	8微帧 (默认值 , 相当于1毫秒 )																		
10h	16微帧 (2毫秒 )																		
20h	32微帧 (4毫秒 )																		
40h	64微帧 (8毫秒 )																		
15时12分	保留。这些位是保留的 , 应该设置为零。																		
11	异步计划驻车模式启用 (可选)□ RO或RW。如果HCCPRAMS寄存器中的 <i>Asynchronous Park Capability</i> 位为1 , 则该位默认为1h , 并且为R/W。否则 , 该位必须为零并且为RO。软件使用该位启用或禁用泊车模式。当该位为1时 , 将启用驻车模式。																		
10	保留。此位是保留的 , 应设置为零。																		
9点8分	异步计划驻车模式计数 (可选)□ RO或RW。如果HCCPRAMS寄存器中的 <i>Asynchronous Park Capability</i> 位为1 , 则该字段默认为3h , 为R/W。否则 , 它默认为零 , 并且为RO。它包含允许主机控制器在继续遍历异步调度之前从异步调度上的高速队列头执行的连续事务数。完整操作细节见第4.10.3.2节。有效值为1h至3h。当驻车模式启用为1时 , 软件不得将0写入该位 , 因为这将导致未定义的行为。																		

表2-9。USBCMD-USB命令寄存器位定义(续)

一点	描述
7.	<p><b>灯光主机控制器重置 (可选) □ 转/转。</b>此控制位不是必需的。如果实现,它允许驱动程序重置EHCI控制器,而不会影响端口的状态或与配套主机控制器的关系。例如,PORSTC寄存器不应重置为其默认值,CF位设置不应为零(保留端口所有权关系)。</p> <p>主机软件将该位读取为零表示Light host Controller Reset(轻型主机控制器重置)已完成,主机软件可以安全地重新初始化主机控制器。主机软件将该位读取为1表示Light host Controller Reset(灯光主机控制器重置)尚未完成。</p> <p>如果未实现,则对该字段的读取将始终返回零。</p>
6.	<p><b>异步前进门铃中断 □ 转/转。</b>该位被软件用作门铃,告诉主机控制器在下次推进异步调度时发出中断。软件必须对此位写入1才能按门铃。</p> <p>当主机控制器退出所有适当的缓存调度状态时,它会在USBSTS寄存器中设置异步前进中断状态位。如果USBINTR寄存器中的Interrupt on Async Advance Enable位为1,则主机控制器将在下一个中断阈值断言中断。操作细节见第4.8.2节。</p> <p>主机控制器在设置异步前进中断后将此位设置为零 USBSTS寄存器中的状态位为1。</p> <p>当异步调度被禁用时,软件不应该向该位写入一。这样做将产生未定义的结果。</p>
5.	<p><b>异步计划启用 □ 转/转。</b>默认0b。此位控制主机控制器是否跳过处理异步调度。数值的平均值 0b不处理异步计划 1b 使用ASYNCLISTADDR寄存器访问异步调度。</p>
4.	<p><b>定期计划启用 □ 转/转。</b>默认0b。该位控制主机控制器是否跳过对周期性调度的处理。数值的平均值: 0b不处理定期计划 1b 使用PERIODICLISTBASE寄存器访问定期计划。</p>
3分2秒	<p><b>框架列表大小 □ (R/W或RO)。</b>默认00b。仅当HCCPRAMS寄存器中的可编程帧列表标志设置为1时,此字段为R/W。此字段指定帧列表的大小。帧列表的大小控制帧索引寄存器中的哪些位应用于帧列表当前索引。数值的平均值: 00b1024个元素(4096字节)默认值01b512个元素(2048字节) 10b256元素(1024字节)-用于资源受限的环境11b 保留</p>

表2-9。USBCMD-USB命令寄存器位定义(续)

一点	描述
1.	<p><b>主机控制器复位 (HCRESET) □ 转/转。</b>该控制位由软件用来重置主机控制器。这对根集线器寄存器的影响类似于芯片硬件重置。</p> <p>当软件将一写入该位时，主机控制器将其内部管道、计时器、计数器、状态机等重置为其初始值。USB上当前正在进行的任何事务都将立即终止。USB重置未在下游端口上驱动。</p> <p>PCI配置寄存器不受此重置的影响。所有操作寄存器，包括端口寄存器和端口状态机，都设置为其初始值。端口所有权归配套主机控制器所有，副作用如第4.2节所述。软件必须按照第4.1节所述重新初始化主机控制器，以便将主机控制器恢复到操作状态。</p> <p>当复位过程完成时，主机控制器将此位设置为零。软件无法通过向该寄存器写入零来提前终止重置过程。</p> <p>当USBSTS寄存器中的<i>HCHalted</i>位为零时，软件不应将该位设置为1。试图重置正在运行的主机控制器将导致未定义的行为。</p>
0	<p><b>运行/停止 (RS) □ 转/转。</b>默认0b。1=运行。0=停止。当设置为1时，主机控制器继续执行计划。只要该位被设置为1，主机控制器就继续执行。当该位设置为0时，主机控制器完成USB上的当前和任何主动流水线事务，然后停止。软件清除运行位后，主机控制器必须在16微帧内停止。状态寄存器中的HC Halted位指示主机控制器何时完成其挂起的流水线事务并进入停止状态。除非主机控制器处于“暂停”状态（即USBSTS寄存器中的“暂停”为“一”），否则软件不得向该字段写入“一”。这样做将产生未定义的结果。</p>

### 2.3.2 USBSTS公司口 USB状态寄存器

地址 : 作战基地+ (04h) 默认值  
           :00001000h  
 AttributeRO、R/W、R/WC (取决于字段) 尺寸 32位

此寄存器指示挂起的中断和主机控制器的各种状态。串行总线上事务产生的状态未在此寄存器中指示。软件通过向该寄存器中写入1将其设置为0。有关USB中断条件的更多信息 ,请参阅第4.15节。

表2-10。USBSTS USB状态寄存器位定义

一点	描述
31分16秒	保留。这些位是保留的 ,应该设置为零。
15	异步计划状态口 RO.0=默认值。该位报告异步调度的当前实际状态。如果该位为零 ,则异步调度的状态被禁用。如果此位为1 ,则异步调度的状态为启用状态。当软件转换USBCMD寄存器中的异步调度启用位时 ,主机控制器不需要立即禁用或启用异步调度。当该位和异步调度启用位的值相同时 ,异步调度被启用 (1 )或禁用 (0 )。
14	定期计划状态口 RO.0=默认值。该位报告周期表的当前实际状态。如果该位为零 ,则周期表的状态被禁用。如果此位为1 ,则启用周期计划的状态。当软件转换USBCMD寄存器中的周期调度启用位时 ,主机控制器不需要立即禁用或启用周期调度。当该位和周期调度启用位为相同值时 ,周期调度被启用 (1 )或禁用 (0 )。
13	填海口 RO.0=默认值。这是一个只读状态位 ,用于检测空的异步计划。第4.8.3节描述了空计划检测的操作模型。第4.8.6节描述了该位的有效转换。
12	六氯环己烷口 RO.1=默认值。只要运行/停止位为1 ,该位就为0。由于运行/停止位被软件或主机控制器硬件设置为0 (例如内部错误) ,主机控制器在停止执行后将该位设置为1。
11时6分	保留。这些位是保留的 ,应该设置为零。
5.	异步前进中断口 R/WC。0=默认值。系统软件可以通过向USBCMD寄存器中的 <i>interrupt on Async Advance Doorbell</i> 位写入一个来强制主机控制器在下次主机控制器推进异步调度时发出中断。该状态位指示该中断源的断言。
4.	主机系统错误口 R/WC。当涉及主机控制器模块的主机系统访问过程中发生严重错误时 ,主机控制器将此位设置为1。在PCI系统中 ,将该位设置为1的条件包括PCI奇偶校验错误、PCI主中止和PCI目标中止。当发生此错误时 ,主机控制器会清除命令寄存器中的运行/停止位 ,以阻止计划的TD的进一步执行。
3.	帧列表滚动口 R/WC。当帧列表索引 (见第2.3.4节 )从其最大值翻转为零时 ,主机控制器将该位设置为1。滚动发生的确切值取决于帧列表的大小。例如 ,如果帧列表大小 (如在USBCMD寄存器的帧列表大小字段中编程的 )是1024 ,则帧索引/寄存器在每次FRINDEX[13]切换时翻转。类似地 ,如果大小是512 ,则主机控制器每次FRINDEX[12]切换时都将该位设置为1。

表2-10。USBSTS USB状态寄存器位定义(续)

一点	描述
2.	<b>端口更改检测</b> <input type="checkbox"/> <b>R/WC</b> 。当端口所有者位设置为零的任何端口(见第2.3.9节)具有从零到一的更改位转换,或由于在挂起的端口上检测到J-K转换而导致强制端口恢复位从零到1的转换时,主机控制器将该位设置为一。在系统软件通过向端口的端口所有者位写入一个连接状态更改来放弃对连接端口的所有权后,该位也将被设置为连接状态更改的结果(见第4.2.2节)。 允许将该钻头保持在辅助动力井中。或者,也可以接受的是,在EHCI HC设备的D3到D0转换时,该位加载所有PORTSC更改位的OR(包括:强制端口恢复、过电流更改、启用/禁用更改和连接状态更改)。
1.	<b>USB错误中断(USBERRINT)</b> <input type="checkbox"/> <b>R/WC</b> 。当USB事务的完成导致错误情况(例如,错误计数器下溢)时,主机控制器将此位设置为1。如果发生错误中断的TD也设置了IOC位,则设置该位和USBINT位。有关将导致此位设置为1的USB错误列表,请参见第4.15.1节。
0	<b>USB中断(USBINT)</b> <input type="checkbox"/> <b>R/WC</b> 。主机控制器在USB事务完成时将该位设置为1,这导致设置了IOC位的传输描述符失效。 当检测到短数据包(实际接收的字节数小于预期的字节数)时,主机控制器也将该位设置为1。

### 2.3.3 美国银行 USB中断启用寄存器

地址 : 作战基地+ (08h) 默认值 :  
00000000h  
属性 R/W  
尺寸 32位

该寄存器启用和禁用向软件报告相应的中断。当一个位被设置并且相应的中断被激活时,一个中断被生成到主机。在该寄存器中被禁用的中断源仍然出现在USBSTS中,以允许软件轮询事件。

每个中断使能位描述指示其是否依赖于中断阈值机制(见第4.15节)。

表2-11。USBINTR-USB中断启用寄存器

一点	中断源	描述
31:6	保留。	这些位是保留的,应该为零。
5.	<b>异步提前中断启用</b>	当该位为1,并且USBSTS寄存器中的异步前进中断位为1时,主机控制器将在下一个中断阈值发出中断。通过软件清除异步前进中断位来确认中断。
4.	<b>主机系统错误启用</b>	当此位为1,并且USBSTS寄存器中的主机系统错误状态位为1时,主机控制器将发出中断。通过清除主机系统错误位的软件确认中断。

表2-11。USBINTR-USB中断启用寄存器(续)

一点	描述	
3.	<b>帧列表滚动启用。</b>	当该位为1，并且USBSTS寄存器中的帧列表滚动位为1时，主机控制器将发出中断。通过软件清除帧列表翻转位来确认中断。
2.	<b>端口更改中断启用。</b>	当此位为1，并且USBSTS寄存器中的端口更改检测位为1时，主机控制器将发出中断。通过软件清除端口更改检测位来确认中断。
1.	<b>USB错误中断启用。</b>	当该位为1，并且USBSTS寄存器中的USBERRINT位为1时，主机控制器将在下一个中断阈值发出中断。通过清除USBERRINT位的软件确认中断。
0	<b>USB中断启用。</b>	当该位为1，并且USBSTS寄存器中的USBINT位为1时，主机控制器将在下一个中断阈值发出中断。通过清除USBINT位的软件确认中断。

注：对于所有启用寄存器位，1=启用，0=禁用

### 2.3.4 FRINDEX公司□ 帧索引寄存器

地址： 操作基准+ (0Ch) 默认值：

00000000h

属性： R/W (写入必须是D字写入) 大小  
32位

该寄存器被主机控制器用来索引到周期性帧列表中。寄存器每125微秒更新一次（每微帧更新一次）。比特[N:3]用于在周期性调度执行期间选择周期性帧列表中的特定条目。用于索引的位数取决于系统软件在USBCMD寄存器的帧列表大小字段中设置的帧列表的大小（见表2-9）。

此寄存器必须作为D字写入。字节写入会产生未定义的结果。除非主机控制器处于HCHalted位（USBSTS寄存器第2.3.2节所示的Halted状态，否则无法写入此寄存器。当Run/Stop位设置为1（USBCMD寄存器，第2.3.1节）时，写入此寄存器会产生未定义的结果。写入此寄存器也会影响SOF值。详见第4.5节。

表2-12。FRINDEX公司□ 帧索引寄存器

一点	描述										
31分14秒	保留。										
13:0	<p>框架索引。该寄存器中的值在每个时间帧结束时递增(例如。微框架)。比特[N:3]用于帧列表当前索引。这意味着每个帧列表的位置在移动到之前被访问8次(帧或微帧)下一个索引。以下说明了基于帧列表值的N值</p> <p>USBCMD寄存器中的大小字段。</p> <table> <thead> <tr> <th>USBCMD[帧列表大小]数字元素</th> <th>N</th> </tr> </thead> <tbody> <tr> <td>00b (1024)</td> <td>12</td> </tr> <tr> <td>01b (512)</td> <td>11</td> </tr> <tr> <td>10b (256)</td> <td>10</td> </tr> <tr> <td>11b 保留</td> <td></td> </tr> </tbody> </table>	USBCMD[帧列表大小]数字元素	N	00b (1024)	12	01b (512)	11	10b (256)	10	11b 保留	
USBCMD[帧列表大小]数字元素	N										
00b (1024)	12										
01b (512)	11										
10b (256)	10										
11b 保留											

总线SOF令牌的SOF帧编号值是从该寄存器导出或可替换地管理的。有关主机控制器SOF值管理要求的详细说明 ,请参阅第4.5节。FRINDEX的值必须为125□秒 (1微帧)。SOF值可以被实现为11位影子寄存器。对于这个讨论 ,这个影子寄存器是11位的 ,并且被命名为SOFV。SOFV每8微帧更新一次。(1毫秒)。实现这种行为的一个示例实现是 ,每当FRINDEX[2:0]从零增加到一时 ,就增加SOFV。

软件必须使用FRINDEX的值来导出当前的微帧数 ,这既是为了高速等时调度的目的 ,也是为了提供客户端驱动程序所需的获取微帧数功能。因此 ,如果芯片复位或软件写入FRINDEX ,则FRINDEX的值和SOFV的值必须保持一致。对FRINDEX的写入还必须通过FRINDEX[13:3]写入SOFV[10:0]。为了使更新尽可能简单 ,软件不应在三个最低有效位为111b或000b的情况下写入FRINDEX值。请参阅第4.5节。

### 2.3.5 控制段□ 控制数据结构段寄存器

地址 : 作战基地+ (10h )默认值 :  
00000000h  
属性 : R/W (写入必须是D字写入 )大小 :  
32位

此32位寄存器对应于所有EHCI数据结构的最高有效地址位[63:32]。如果HCCPRAMS中的64位寻址能力字段为零 ,则不使用此寄存器。软件无法写入 ,从该寄存器读取将返回零。

如果HCCPRAMS中的64位寻址能力字段为1 ,则该寄存器与链接指针一起使用 ,以构造到EHCI控制数据结构的64位地址。该寄存器与来自PERIODICLISTBASE、ASYNCLISTADDR或任何控制数据结构链接字段的链接指针连接 ,以构造64位地址。

该寄存器允许主机软件将所有控制数据结构定位在相同的4千兆字节存储器段内。

### 2.3.6 牙周膜基□ 周期帧列表基址寄存器

地址 : 作战基地+ (14h )默认值 :  
未定义  
属性 : R/W (写入必须是D字写入 )大小 :  
32位

此32位寄存器包含系统内存中周期帧列表的起始地址。如果主机控制器处于64位模式 (如

HCCSARAMS寄存器),则每个控制数据结构地址的最高有效32位来自CTRLDSSEGMENT寄存器(见第2.3.5节)。系统软件在主机控制器开始执行调度之前加载该寄存器(见4.1)。该物理内存指针引用的内存结构假定为4-K字节对齐。该寄存器的内容与帧索引寄存器(FRINDEX)相结合,使主机控制器能够按顺序逐步通过周期帧列表。

表2-13。牙周膜基址周期帧列表基址寄存器

一点	描述
31分12秒	基址(低位)。这些位分别对应于存储器地址信号[31:12]。
11:0	保留。必须写成0s。在运行时,这些位的值是未定义的。

### 2.3.7 异步地址□当前异步列表地址寄存器

地址: 作战基地+ (18h) 默认值:  
未定义  
属性: 读/写(写入必须是D字写入)大小: 32位

这个32位寄存器包含要执行的下一个异步队列头的地址。该物理存储器指针所引用的存储器结构被假定为32字节(高速缓存行)对齐。

表2-14。异步地址□当前异步列表地址寄存器

一点	描述
31分5秒	链路指针低(LPL)。这些位分别对应于存储器地址信号[31:5]。此字段只能引用队列头(QH),请参见第3.6节。
4点	保留。这些位是保留的,它们的值对操作没有影响。

### 2.3.8 配置标志□配置标志寄存器

地址: 作战基地+ (40h) 默认值:  
00000000h  
属性 R/W  
尺寸 32位

此寄存器位于辅助电源井中。仅当最初施加辅助电源或响应于主机控制器重置时,才由硬件重置。

表2-15。配置标志□配置标志寄存器位定义

一点	描述
31比1	保留。这些位是保留的,应该设置为零。
0	配置标志(CF)□转/转。默认0b。主机软件将此位设置为配置主机控制器过程中的最后一个操作(请参阅第4.1节)。此位控制默认端口路由控制逻辑。位值和副作用如下所示。有关操作细节,请参见第4.2节。 0b 端口路由控制逻辑默认将每个端口路由到依赖于实现的经典主机控制器。 1bPort路由控制逻辑默认将所有端口路由到此主机控制器。

### 2.3.9 端口SC□ 端口状态和控制寄存器

地址 : 作战基地+ (44h+ (4\*端口号-1))  
     其中 : 端口号为1、2、3...N\_PORTS  
 默认值 : 00002000h (带PPC设置为1) ; 00003000h (PPC设置为零) 属性 :  
     RO、R/w、R/WC (取决于字段)  
 尺寸 32位

主机控制器必须实现一个或多个端口寄存器。主机控制器的特定实例化实现的端口寄存器的数量记录在HCSPARAM寄存器中(第2.2.3节)。

软件使用此信息作为输入参数来确定需要服务的端口数量。所有端口都具有以下定义的结构。

此寄存器位于辅助电源井中。仅当最初施加辅助电源或响应于主机控制器重置时,才由硬件重置。港口的初始条件是:

- ① 未连接设备,
- ② 端口已禁用

如果端口具有端口电源控制,则软件在通过将端口电源设置为1为端口通电后才能更改端口的状态。在端口电源稳定之前,软件不得尝试更改端口状态。主机必须在0到1转换后的20毫秒内使端口电源稳定。

**注1:**当连接设备时,端口状态转换为连接状态,系统软件将像处理任何状态更改通知一样对此进行处理。有关更改事件如何与端口挂起模式交互的操作要求,请参阅第4.3节。

**注2:**如果某个端口被用作调试端口,则当配置标志为零时,该端口可能会报告设备已连接并启用。

表2-16。端口SC□ 端口状态和控制

一点	描述
31分23秒	保留。这些位是为将来使用而保留的,并且在读取时应返回零值。
22	<b>过电流唤醒启用 (WKOC_E)</b> □ 转/转。默认值=0b。将该位写入1使端口能够对作为唤醒事件的过电流条件敏感。有关此位对恢复事件行为的影响,请参见第4.3节。操作模型参见第4.3.1节。 如果端口功率为零,则此字段为零。
21	<b>断开连接唤醒启用 (WKDSCNNT_E)</b> □ 转/转。默认值=0b。将此位写入1使端口能够对设备断开连接作为唤醒事件敏感。有关此位对恢复事件行为的影响,请参见第4.3节。操作模型参见第4.3.1节。 如果端口功率为零,则此字段为零。
20	<b>启用连接唤醒 (WKCNNT_E)</b> □ 转/转。默认值=0b。将此位写入1使端口能够对作为唤醒事件的设备连接敏感。有关此位对恢复事件行为的影响,请参见第4.3节。操作模型参见第4.3.1节。 如果端口功率为零,则此字段为零。

表2-16。SC□ 端口状态和控制 (续 )

一点	描述														
19时16分	<p><b>端口测试控制</b>□转/转。默认值=0000b。当该字段为零时 ,端口未在测试模式下运行。非零值表示其在测试模式下运行 ,并且特定测试模式由该特定值表示。测试模式位的编码为 (保留0110b-1111b) :</p> <table> <thead> <tr> <th>比特</th> <th>测试模式</th> </tr> </thead> <tbody> <tr> <td>0000b</td> <td>测试模式未启用</td> </tr> <tr> <td>0001b</td> <td>测试J_STATE</td> </tr> <tr> <td>0010b</td> <td>测试K_STATE</td> </tr> <tr> <td>0011b</td> <td>测试S0_NAK</td> </tr> <tr> <td>0100b</td> <td>测试包</td> </tr> <tr> <td>0101b</td> <td>测试强制启用</td> </tr> </tbody> </table> <p>关于使用这些测试模式的操作模型 ,请参阅第4.14节 ,关于每个测试模式的详细信息 ,请参阅USB规范2.0版第7章。</p>	比特	测试模式	0000b	测试模式未启用	0001b	测试J_STATE	0010b	测试K_STATE	0011b	测试S0_NAK	0100b	测试包	0101b	测试强制启用
比特	测试模式														
0000b	测试模式未启用														
0001b	测试J_STATE														
0010b	测试K_STATE														
0011b	测试S0_NAK														
0100b	测试包														
0101b	测试强制启用														
15时14分	<p><b>端口指示灯控制</b>。默认值=00b。如果HCSPARAMS寄存器中的P_INDICATOR位为零 ,则写入这些位没有效果。如果P_INDICATOR比特是1 ,则比特编码为 :</p> <table> <thead> <tr> <th>位值</th> <th>含义</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>端口指示灯熄灭</td> </tr> <tr> <td>01b</td> <td>绿色</td> </tr> <tr> <td>10</td> <td>未定义</td> </tr> <tr> <td>11b</td> <td></td> </tr> </tbody> </table> <p>有关如何使用这些位的说明 ,请参阅USB规范2.0版。</p> <p>如果端口功率为零 ,则此字段为零。</p>	位值	含义	00b	端口指示灯熄灭	01b	绿色	10	未定义	11b					
位值	含义														
00b	端口指示灯熄灭														
01b	绿色														
10	未定义														
11b															
13	<p><b>港口所有者</b>□R/W默认值=1b。当CONFIGFLAG寄存器中的配置位进行0b到1b的转换时 ,该位无条件地变为0b。每当配置的比特为零时 ,该比特无条件地变为1b。</p> <p>系统软件使用此字段将端口的所有权释放给选定的主机控制器 (如果连接的设备不是高速设备 )。当连接的设备不是高速设备时 ,软件会将一写入该位。此位中的1表示伴随主机控制器拥有并控制端口</p> <p>4.2操作细节。</p>														
12	<p><b>端口电源 (PP)</b>□R/W或RO。此位的功能取决于HCSPARAMS寄存器中端口功率控制 (PPC) 字段的值。行为如下 :</p> <p><b>PPCPP 操作</b></p> <p>0b1b反 渗透□主机控制器没有端口电源控制开关。 每个端口都是硬连接到电源的。</p> <p>1b1b/0b右□主机控制器具有端口电源控制开关。此位表示开关的当前设置 (0=关闭 ,1=打开 )。当端口上没有电源时 (即PP等于0) ,端口不起作用 ,不会报告连接、分离等。</p> <p>当在通电端口上检测到过电流情况并且PPC为1时 ,主机控制器可以将每个受影响端口中的PP位从1转换为0 (从端口断电 )。</p>														

表2-16。SC□ 端口状态和控制 (续 )

一点	描述										
11时10分	<p><b>线路状态</b>□这些位反映了D+ (位11)和D- (位10)信号线的当前逻辑电平。这些位用于在端口重置和启用序列之前检测低速USB设备。只有当端口启用位为零并且当前连接状态位设置为1时 ,此字段才有效。</p> <p>比特的编码是 :</p> <table> <thead> <tr> <th>Bits[11:10]USB状态</th> <th>解释</th> </tr> </thead> <tbody> <tr> <td>00bSE0Not</td> <td>低速设备 ,执行EHCI</td> </tr> <tr> <td>Not</td> <td>低速设备、执行EHCI复</td> </tr> <tr> <td>位01bK 状态</td> <td>低速设备 ,释放端口11b的所有</td> </tr> <tr> <td>权</td> <td>UndefinedNot 低速设备并 执行EHCI重设。</td> </tr> </tbody> </table> <p>如果端口功率为零 ,则未定义此字段的值。</p>	Bits[11:10]USB状态	解释	00bSE0Not	低速设备 ,执行EHCI	Not	低速设备、执行EHCI复	位01bK 状态	低速设备 ,释放端口11b的所有	权	UndefinedNot 低速设备并 执行EHCI重设。
Bits[11:10]USB状态	解释										
00bSE0Not	低速设备 ,执行EHCI										
Not	低速设备、执行EHCI复										
位01bK 状态	低速设备 ,释放端口11b的所有										
权	UndefinedNot 低速设备并 执行EHCI重设。										
9	保留。此位保留供将来使用 ,读取时应返回零值。										
8.	<p><b>端口重置</b>□<b>转/转</b>。1=端口处于重置状态。0=端口未处于重置状态。默认值为0。当软件将一写入该位 (从零开始 )时 ,USB规范修订版2.0中定义的总线重置序列启动。软件将零写入该位以终止总线复位序列。软件必须将该位保持在足够长的一位 ,以确保USB规范2.0版中规定的重置序列完成。注意 :当软件将该位写入1时 ,还必须将0写入端口启用位。</p> <p>注意 ,当软件将零写入该位时 ,在位状态变为零之前可能存在延迟。直到复位完成后 ,位状态才会读取为零。如果重置完成后端口处于高速模式 ,主机控制器将自动启用此端口 (例如 ,将端口启用位设置为1)。主机控制器必须在软件将该位从1转换为0的2毫秒内终止重置并稳定端口的状态。例如 :如果端口在重置过程中检测到连接的设备是高速的 ,那么主机控制器必须在软件将该位写入0的2ms内使端口处于启用状态。</p> <p>在软件尝试使用此位之前 ,USBSTS寄存器中的<b>HCHalted</b>位应为零。当<b>HCHalted</b>位为1时 ,主机控制器可以将断言的端口重置保持为1。</p> <p>如果端口功率为零 ,则此字段为零。</p>										
7.	<p><b>悬</b>□<b>转/转</b>。1=端口处于挂起状态。0=端口未处于挂起状态。默认值为0。此寄存器的端口启用位和挂起位定义端口状态如下 :</p> <table> <thead> <tr> <th>位[端口启用 ,挂起]</th> <th>端口状态</th> </tr> </thead> <tbody> <tr> <td>0X</td> <td>禁用</td> </tr> <tr> <td>10</td> <td>使可能</td> </tr> <tr> <td>11</td> <td>悬</td> </tr> </tbody> </table> <p>处于挂起状态时 ,除端口重置外 ,数据的下行传播在此端口上被阻止。如果将该位写入1时事务正在进行 ,则阻塞发生在当前事务结束时。在挂起状态下 ,端口对恢复检测很敏感。请注意 ,在端口挂起之前 ,位状态不会改变 ,并且如果USB上当前有事务正在进行 ,则挂起端口可能会延迟。</p> <p>主机控制器忽略对该位的零写入。当出现以下情况时 ,主机控制器将无条件地将该位设置为零 :</p> <ul style="list-style-type: none"> <li>① 软件将强制端口恢复位设置为零 (从一开始)。</li> <li>② 软件将端口重置位设置为1 (从零开始)。</li> </ul> <p>如果主机软件在端口未启用时将该位设置为1 (即端口启用位为零) ,则结果未定义。</p> <p>如果端口功率为零 ,则此字段为零。</p>	位[端口启用 ,挂起]	端口状态	0X	禁用	10	使可能	11	悬		
位[端口启用 ,挂起]	端口状态										
0X	禁用										
10	使可能										
11	悬										

表2-16。SC□ 端口状态和控制 (续 )

一点	描述
6.	<p><b>强制端口恢复□ 转/转。</b> 1=端口上检测到/驱动恢复。0=端口上未检测/驱动恢复 (K状态)。默认值为0。为操作此位而定义的此功能取决于挂起位的值。例如,如果端口未挂起(挂起和启用位为一),并且软件将此位转换为一,则对总线的影响未定义。</p> <p>软件将该位设置为1以驱动恢复信号。如果在端口处于挂起状态时检测到J-to-K转换,则主机控制器将此位设置为1。当该位由于检测到J到K转换而转换为1时,USBSTS寄存器中的端口更改检测位也被设置为1。如果软件将此位设置为1,则主机控制器不得设置端口更改检测位。</p> <p>请注意,当EHCI控制器拥有端口时,恢复序列遵循USB规范修订版2.0中记录的定义序列。只要此位保持为1,就在端口上驱动恢复信号(全速“K”)。软件必须适当地为恢复计时,并在经过适当的时间后将此位设置为零。写入零(从一开始)会导致端口返回高速模式(迫使端口下方的总线进入高速空闲状态)。该位将保持为1,直到端口切换到高速空闲。主机控制器必须在软件将该位设置为零的2毫秒内完成该转换。</p> <p>如果端口功率为零,则此字段为零。</p>
5.	<p><b>过电流变化□R/WC。</b> 默认值为0。1=当过电流激活发生变化时,此位被设置为1。软件通过向该位位置写入一来清除该位。</p>
4.	<p><b>过电流激活□RO。</b> 默认值=0。1=此端口当前存在过电流情况。0=此端口没有过电流情况。当过电流条件被消除时,该位将自动从1转换为0。</p>
3.	<p><b>端口启用/禁用更改□R/WC。</b> 1=端口启用/禁用状态已更改。0=无变化。默认值为0。对于根集线器,只有当端口由于EOF2点存在的适当条件而被禁用时,此位才会被设置为1(有关端口错误的定义,请参阅USB规范第11章)。软件通过向该位写入1来清除该位。</p> <p>如果端口功率为零,则此字段为零。</p>
2.	<p><b>端口启用/禁用□转/转。</b> 1=启用。0=禁用。默认值为0。端口只能由主机控制器作为重置和启用的一部分启用。软件无法通过向该字段写入端口来启用端口。只有当重置序列确定连接的设备是高速设备时,主机控制器才会将该位设置为1。</p> <p>端口可以由故障条件(断开连接事件或其他故障条件)或主机软件禁用。请注意,在端口状态实际更改之前,位状态不会更改。由于其他主机控制器和总线事件,可能会延迟禁用或启用端口。有关端口重置和启用的全部详细信息,请参阅第4.2节。</p> <p>当端口被禁用时(0b),除重置外,数据的下行传播在此端口上被阻止。</p> <p>如果端口功率为零,则此字段为零。</p>
1.	<p><b>连接状态更改□R/WC。</b> 1=当前连接状态的变化。0=无变化。默认值为0。表示端口的“当前连接状态”发生了更改。主机控制器为端口设备连接状态的所有更改设置此位,即使系统软件尚未清除现有的连接状态更改。例如,在系统软件清除已更改的条件之前,插入状态更改了两次,集线器硬件将“设置”一个已设置的位(即,该位将保持设置状态)。软件通过向该位写入1将其设置为0。</p> <p>如果端口功率为零,则此字段为零。</p>

表2-16。SC□ 端口状态和控制 (续 )

一点	描述
0	当前连接状态□RO.1=端口上存在设备。0=不存在设备。默认值为0。此值反映端口的当前状态 ,可能与导致设置连接状态更改位 (位1 )的事件不直接对应。 如果端口功率为零 ,则此字段为零。

### 3. 数据结构

本节定义了用于在HCD (软件) 和增强型主控制器 (硬件) 之间通信控制、状态和数据的接口数据结构。本章中的数据结构定义支持32位内存缓冲区地址空间。附录B说明了接口数据结构的64位版本。该接口由周期调度、周期帧列表、异步调度、等时事务描述符、拆分事务等时传输描述符、队列头和队列元素传输描述符组成。

周期性帧列表是主机控制器接口的所有周期性 (同步和中断传输类型) 支持的根。异步列表是所有批量和控制传输类型支持的根。使用等时事务描述符管理等时数据流 (TD如第3.3节所述)。使用分割事务等时传输描述符管理等时间分割事务数据流 (iTDS如第3.4节所示)。所有中断，控制和批量数据流通过队列头 (第3.6节) 和队列元素传输描述符 (第3.5节中描述的qTD) 进行管理。<sup>1</sup>这些数据结构经过优化，以减少调度的总内存占用，并 (平均) 减少执行USB事务所需的内存访问次数。

请注意，软件必须确保EHCI主机控制器无法访问的接口数据结构跨越4K页面边界。

本章中定义的数据结构 (从主机控制器的角度来看) 是只读字段和读/写字段的混合。主机控制器必须在所有数据结构写入时保留只读字段。

#### 3.1 周期帧列表

此时间表适用于所有周期性传输 (同步传输和中断传输)。周期性调度是使用`PERIODICLISTBASE`地址寄存器和`FRINDEX`寄存器从操作寄存器空间引用的。周期性调度基于一个称为周期性帧列表的指针数组。`PERIODICLISTBASE`地址寄存器与`FRINDEX`寄存器相结合，产生一个进入帧列表的内存指针。周期性帧列表实现了工作随时间推移的滑动窗口。

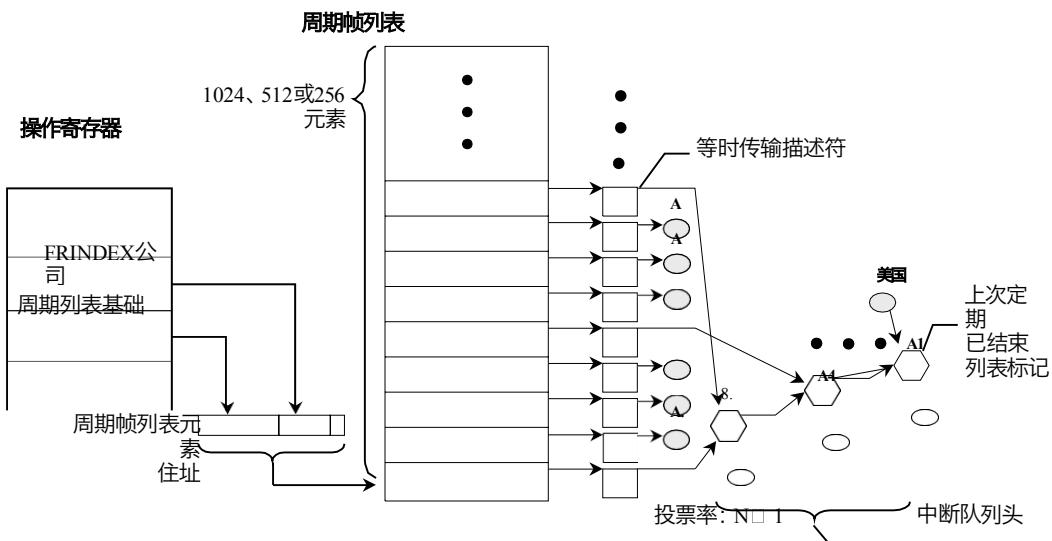
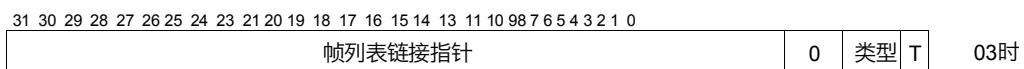


图3-1。定期计划组织

<sup>1</sup>分割事务中断、批量和控制也使用队列头和队列元素传输描述符进行管理。

周期性帧列表是帧列表链接指针的4K页对齐阵列。帧列表的长度可以是可编程的。周期性帧列表的可编程性通过HCCRAMS寄存器导出到系统软件。如果不可编程，则长度为1024个元素。如果可编程，则系统软件可以将长度选择为256、512或1024个元素中的一个。一个实现必须支持所有三种大小。通过系统软件将适当的值写入USBCMD寄存器中的帧列表大小字段来完成对大小（即元素数量）的编程。

帧列表链接指针将主机控制器指向当前微帧的帧的周期性调度中的第一个工作项。链接指针在帧列表中的DWORD边界上对齐。



**图3-2。帧列表元素指针的格式**

帧列表链接指针总是引用32字节对齐的内存对象。所引用的对象可以是用于高速设备的等时传输描述符、拆分事务等时传输描述符（用于全速等时端点）或队列头（用于支持高速、全速和低速中断）。系统软件不应将非定期计划项目放入定期计划中。帧列表指针中的最低有效位用于将主机控制器设置为指针所引用的对象的类型。

最低有效位是T位（位0）。当此位设置为1时，主机控制器将永远不会将帧列表指针的值用作物理内存指针。*Typ*字段用于指示此指针所引用的数据结构的确切类型。值编码为：

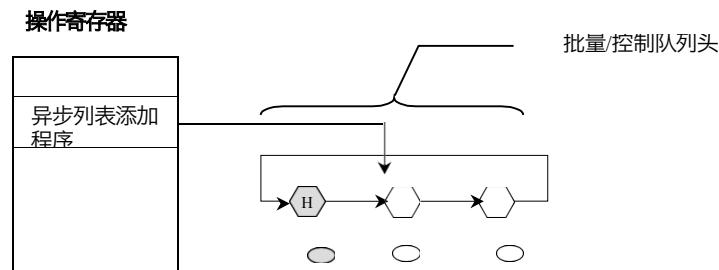
**表3-1。键入字段值定义**

价值	意思
00亿	等时传输描述符（TD，见第3.3节）
01乙	排队头（QH，见第3.6节）
10亿	拆分交易等时转移描述符（SiTD，见第3.4节）。
11磅	框架跨度横向节点（FSTN，见第3.7节）

请参阅第4.4节，了解主机控制器的操作细节和框架列表。

### 3.2 异步列表队列头指针

异步传输列表（基于ASYNCLISTADDR寄存器）是管理所有控制和批量传输的地方。主机控制器仅在该列表到达周期列表末尾、周期列表被禁用或周期列表为空时才使用该列表。



**图3-3。异步计划组织**

异步列表是一个简单的队列头循环列表。ASYNCLISTADDR寄存器只是指向下一个队列头的指针。这为链接到异步列表中的所有队列头实现了纯循环服务。

### 3.3 等时(高速)传输描述符(TD)

等时传输描述符的格式如图3-4所示。此结构仅用于高速等时端点。所有其他传输类型都应使用队列结构。等时TDs必须在32字节的边界上对齐。

31 30 29 28 27 26 25 24 23 21 20 19 18 17 16 15 14 13 11 10 9 8 7 6 5 4 3 2 1 0				
下一个链接指针				0   类型 T   03时
地位	事务处理0长度	国际	PG*	事务处理0偏移*
地位	事务处理1长度	国际	PG*	事务处理1抵销*
地位	事务处理2长度	国际	PG*	事务处理2抵销*
地位	事务处理3长度	国际 奥 委 会	PG*	事务处理3抵销*
地位	事务处理5长度	国际	PG*	事务处理5抵销*
地位	事务处理6长度	国际	PG*	事务处理6抵销*
地位	事务处理7长度	国际	PG*	事务处理7抵销*
缓冲区指针(第0页)			结束点 R	设备地址
			I/O	最大数据包大小 2小时-28小时
缓冲区指针(第2页)			保留	Mult2英尺2英寸 公司
缓冲区指针(第3页)			保留	33-30小时
缓冲区指针(第4页)			保留	37-34小时
缓冲区指针(第5页)			保留	3至38小时
缓冲区指针(第6页)			保留	3英尺3英寸

主机控制器读/写

主机控制器只读

\*注意：如果I/O字段指示OUT，则主机控制器可以修改这些字段。

图3-4。等时事务描述符(TD)

#### 3.3.1 下一个链接指针

iTD的第一个DWord是指向下一个计划数据结构的指针。

表3-2。下一个计划元素指针

一点	描述
31分5秒	<b>链接指针(LP)</b> 。这些位分别对应于存储器地址信号[31:5]。该字段指向另一个等时事务描述符(TD/siTD)、队列头(QH)或FSTN。
4点3分	<b>保留</b> 。这些位是保留的，它们的值对操作没有影响。软件应将该字段初始化为零。

表3-2。下一个计划元素指针（续）

一点	描述						
2:1	<p><b>QH/ (S) iTD/FSTN选择 (类型)</b>。该字段向主机控制器指示所引用的项目是iTID、sITD还是QH。这允许主机控制器在项目被提取后对其执行正确类型的处理。值编码为：</p> <table> <thead> <tr> <th>价值</th> <th>含义</th> </tr> </thead> <tbody> <tr> <td>00biTD (等时传输描述符) 01b 列头)</td> <td>QH (队</td> </tr> <tr> <td>10bsiTD (拆分事务等时传输描述符) 11bFSTN (帧跨遍历节点)</td> <td></td> </tr> </tbody> </table>	价值	含义	00biTD (等时传输描述符) 01b 列头)	QH (队	10bsiTD (拆分事务等时传输描述符) 11bFSTN (帧跨遍历节点)	
价值	含义						
00biTD (等时传输描述符) 01b 列头)	QH (队						
10bsiTD (拆分事务等时传输描述符) 11bFSTN (帧跨遍历节点)							
0	<b>终止 (T)</b> 。1=链接指针字段无效。0=链接指针字段有效。						

有关iTID的主机控制器操作模型，请参阅第4.7节。

### 3.3.2 iTD事务状态和控制列表

字1到8是事务控制和状态的八个槽。每个插槽的格式如图3-4所示。每个交易描述包括：

- ① 状态结果字段
- ② 事务长度（针对OUT事务发送的字节和针对IN事务接收的字节）。
- ③ 缓冲区偏移。*PG*和*Transaction X Offset*字段与缓冲区指针列表一起用于构造事务的起始缓冲区地址。

主机控制器使用每个事务描述中的信息加上缓冲页指针列表的前三个双字中包含的端点信息，在USB上执行事务。

表3-3。iTID事务状态和控制

一点	描述										
31分28秒	<p><b>地位</b>此字段记录主机控制器为此插槽执行的事务的状态。此字段是具有以下编码的位向量：</p> <table> <thead> <tr> <th>位</th> <th>定义</th> </tr> </thead> <tbody> <tr> <td>31</td> <td><b>激活</b>。通过软件设置为1，以使主机控制器能够执行同步事务。当与该描述符相关联的事务完成时，主机控制器将该位设置为0，指示该元素的事务在调度中下一次遇到时不应执行。</td> </tr> <tr> <td>30</td> <td><b>数据缓冲区错误</b>。在状态更新期间由主机控制器设置为1，表示主机控制器无法跟上传入数据的接收（溢出）或在传输期间无法足够快地提供数据（欠载）。第4.15.1.1.2节定义了发生欠载运行错误时主机控制器的要求。如果发生超限情况，则无需采取任何行动。</td> </tr> <tr> <td>29</td> <td><b>检测到巴贝尔</b>。当在此描述符生成的事务期间检测到“牙牙学语”时，主机控制器在状态更新期间将其设置为1。</td> </tr> <tr> <td>28</td> <td><b>事务错误 (KactErr)</b>。在主机未从设备接收到有效响应（超时、CRC、错误PID等）的情况下，在状态更新期间由主机控制器设置为1。此位只能设置为同步in事务。</td> </tr> </tbody> </table>	位	定义	31	<b>激活</b> 。通过软件设置为1，以使主机控制器能够执行同步事务。当与该描述符相关联的事务完成时，主机控制器将该位设置为0，指示该元素的事务在调度中下一次遇到时不应执行。	30	<b>数据缓冲区错误</b> 。在状态更新期间由主机控制器设置为1，表示主机控制器无法跟上传入数据的接收（溢出）或在传输期间无法足够快地提供数据（欠载）。第4.15.1.1.2节定义了发生欠载运行错误时主机控制器的要求。如果发生超限情况，则无需采取任何行动。	29	<b>检测到巴贝尔</b> 。当在此描述符生成的事务期间检测到“牙牙学语”时，主机控制器在状态更新期间将其设置为1。	28	<b>事务错误 (KactErr)</b> 。在主机未从设备接收到有效响应（超时、CRC、错误PID等）的情况下，在状态更新期间由主机控制器设置为1。此位只能设置为同步in事务。
位	定义										
31	<b>激活</b> 。通过软件设置为1，以使主机控制器能够执行同步事务。当与该描述符相关联的事务完成时，主机控制器将该位设置为0，指示该元素的事务在调度中下一次遇到时不应执行。										
30	<b>数据缓冲区错误</b> 。在状态更新期间由主机控制器设置为1，表示主机控制器无法跟上传入数据的接收（溢出）或在传输期间无法足够快地提供数据（欠载）。第4.15.1.1.2节定义了发生欠载运行错误时主机控制器的要求。如果发生超限情况，则无需采取任何行动。										
29	<b>检测到巴贝尔</b> 。当在此描述符生成的事务期间检测到“牙牙学语”时，主机控制器在状态更新期间将其设置为1。										
28	<b>事务错误 (KactErr)</b> 。在主机未从设备接收到有效响应（超时、CRC、错误PID等）的情况下，在状态更新期间由主机控制器设置为1。此位只能设置为同步in事务。										

表3-3。iTDD事务状态和控制 (续)

一点	描述
27分16秒	<p><b>事务X长度。</b>对于OUT, 此字段是主机控制器在事务处理期间将发送的数据字节数。主机控制器不需要更新此字段以反映传输期间传输的实际字节数。</p> <p>对于IN, 字段的初始值是主机期望端点传递的字节数。在状态更新期间, 主机控制器会写回成功接收的字节数。</p> <p>此寄存器中的值是实际字节计数 (例如0 零长度数据, 1 一个字节, 2 两个字节等)。</p> <p>此字段可能包含的最大值为0xC00 (3072)。有关大于0x400 (1024) 的数据包大小的操作要求的说明, 请参阅第4.7节。</p>
15	<b>完成时中断 (OC)。</b> 如果此位设置为1, 则它指定当此事务完成时, 主机控制器应在下一个中断阈值发出中断。
14时12分	<b>页面选择 (PG)。</b> 这些位是由软件设置的, 以指示该槽中的偏移字段应该连接哪个缓冲页指针, 以产生该事务的起始存储器地址。此字段的有效值范围为0到6。
11:0	<b>交易记录X抵销。</b> 此字段是一个值, 它是从缓冲区开始的偏移量, 以字节表示。该字段被连接到相邻PG字段中指示的缓冲区页面指针上, 以产生该事务的起始缓冲区地址。

### 3.3.3 iTD缓冲区页指针列表 (加号)

等时事务描述符的数据字9-15名义上是指向该传输描述符的数据缓冲器的页指针 (4K对齐)。这种数据结构要求相关联的数据缓冲区是连续的 (相对于虚拟内存), 但允许物理内存页是不连续的。提供了七个页面指针来支持8个等时传输的表达式。七个指针允许3 (事务) \*1024 (最大数据包大小) \*8 (事务记录) (24576字节) 与该数据结构一起移动, 而与第一页的对齐偏移无关。

由于每个指针都是4K对齐的页面指针, 因此如表3-4和表3-5中所定义的, 几个页面指针中的最低有效12位用于其他目的。

表3-4。iTDD缓冲区指针页0 (加号)

一点	描述
31分12秒	<b>缓冲区指针 (第0页)。</b> 这是一个指向物理内存的4K对齐指针。对应于存储器地址位 [31:12]。
11点8分	<b>端点编号 (Endpt)。</b> 此4位字段选择用作数据源或汇点的设备上的特定端点编号。
7.	<b>保留。</b> 此位是为将来使用而保留的, 应通过软件将其初始化为零。
6点0分	<b>设备地址。</b> 此字段选择用作数据源或汇点的特定设备。

表3-5。iTД缓冲区指针第1页 (加号)

一点	描述
31分12秒	<b>缓冲区指针 (第1页)。</b> 这是一个指向物理内存的4K对齐指针。对应于存储器地址位[31:12]。
11	<b>方向 (IO)。</b> 0=出局; 1=IN。此字段对高速事务应该使用IN还是OUT PID进行编码。
10:0	<b>最大数据包大小。</b> 这直接对应于关联端点的最大数据包大小 ( <i>wMaxPacketSize</i> )。该字段用于高带宽端点，其中每个事务描述 (例如每个微帧) 发出一个以上的事务。此字段与“多”字段一起使用，以支持高带宽管道。该字段还用于所有IN传输，以检测数据包字符串。参见第节 4.7.1用于操作模型。 软件不应设置大于1024 (400h) 的值。任何较大的值都会产生未定义的结果。

表3-6。iTД缓冲区指针第2页 (加号)

一点	描述						
31分12秒	<b>缓冲区指针。</b> 这是一个指向物理内存的4K对齐指针。对应于存储器地址位[31:12]。						
11时2分	<b>保留。</b> 此位是为将来使用而保留的，应设置为零。						
1点	<b>多。</b> 该字段用于向主机控制器指示每个事务描述 (例如每个微帧) 应该执行的事务的数量。有效值为： <table> <thead> <tr> <th>价值</th> <th>含义</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>保留。此字段中的零会产生未定义的结果。</td> </tr> <tr> <td>01b</td> <td>每个微帧 为该 端 点 发布一 个 事务 10b 每 个 微框架为此端点发布两 个 事 务 11b 每个微框为此端点发布 三个事务</td> </tr> </tbody> </table>	价值	含义	00b	保留。此字段中的零会产生未定义的结果。	01b	每个微帧 为该 端 点 发布一 个 事务 10b 每 个 微框架为此端点发布两 个 事 务 11b 每个微框为此端点发布 三个事务
价值	含义						
00b	保留。此字段中的零会产生未定义的结果。						
01b	每个微帧 为该 端 点 发布一 个 事务 10b 每 个 微框架为此端点发布两 个 事 务 11b 每个微框为此端点发布 三个事务						

表3-7。iTД缓冲区指针第3-6页

一点	描述
31分12秒	<b>缓冲区指针。</b> 这是一个指向物理内存的4K对齐指针。对应于存储器地址位[31:12]。
11:0	<b>保留。</b> 这些位是为将来使用而保留的，并且应该设置为零。

### 3.4 拆分事务等时传输描述符 (iTД)

所有通过事务转换器的全速等时传输都使用iTД数据结构进行管理。该数据结构满足管理拆分事务协议的操作要求。有关iTД的操作行为，请参见第4.12.3节。



图3-5。拆分事务等时事务描述符 (iTID)

### 3.4.1 下一个链接指针

siTD的DWord 0是指向下一个调度数据结构的指针。

表3-8。下一个链接指针

一点	描述
31分5秒	下一个链接指针 (LP)。该字段包含周期表中要处理的下一个数据对象的地址，并分别对应于存储器地址信号[31:5]。
4点3分	保留。这些位必须写成0。
2:1	QH(6) iTD/FSTN选择 类型。该字段向主机控制器指示所引用的项目是iTID/siTD还是QH。这允许主机控制器在项目被提取后对其执行正确类型的处理。值编码为： 价值    含义 00biTD (等时传输描述符) 01bQH (队列头) 10bsiTD (拆分事务等时传输描述符) 11bFSTN (帧跨遍历节点)
0	终止 (T)。1=链接指针字段无效。0=链接指针有效。

### 3.4.2 siTD端点能力/特征

字1和2指定了关于全速端点、父事务转换器的寻址和微帧调度控制的静态信息。

表3-9。端点和事务转换器特性

一点	描述
31	方向 (I/O)。0=出局；1=IN。此字段对全速事务应该是IN还是OUT进行编码。
30分24秒	端口号。此字段是收件人事务转换器的端口号。

表3-9。端点和事务转换器特性（续）

一点	描述
23	<b>保留。</b> 此位是保留的，应设置为零。
22时16分	<b>集线器地址。</b> 此字段保存事务转换器集线器的设备地址。
15时12分	<b>保留。</b> 此字段为保留字段，应设置为零。
11点8分	<b>端点编号 (Endpt)。</b> 此4位字段选择用作数据源或汇点的设备上的特定端点编号。
7.	<b>保留。</b> 此位保留以备将来使用。它应该设置为零。
6点0分	<b>设备地址。</b> 此字段选择用作数据源或汇点的特定设备。

表3-10。微框架进度控制

一点	描述
31分16秒	<b>保留。</b> 此字段保留以备将来使用。它应该设置为零。
15时8分	<b>拆分完成掩码(□帧C掩码)。</b> 此字段（以及状态字节中的Active和SplitX状态字段）用于确定主机控制器应在哪些微帧期间执行完整的拆分事务。当满足使用此字段的条件时，此字段中的全零值具有未定义的行为。 主机控制器使用FRINDEX寄存器的三个低阶位的值来索引到此位字段中。如果FRINDEX寄存器值索引到 □帧C-Mask字段为1，则此siTD是事务执行的候选者。 此掩码集中可能有一个以上的位。
7:0分	<b>拆分开始掩码(□帧S掩码)。</b> 此字段（以及状态字节中的Active和SplitX状态字段）用于确定主机控制器应在哪些微帧期间执行启动拆分事务。 主机控制器使用FRINDEX寄存器的三个低阶位的值来索引到此位字段中。如果FRINDEX寄存器值索引到 □帧S掩码字段为1，则此siTD是事务执行的候选字段。看见 第4.12.3.3.1节为主机控制器执行事务之前必须满足的标准的综合列表。 此字段中的全零值与周期性帧列表中的现有值相结合会产生未定义的结果。

### 3.4.3 siTD传输状态

数据字3-6用于管理传输的状态。

表3-11。siTD传输状态和控制

一点	描述
31	<b>完成时中断 (oc)。</b> 0=事务完成时不要中断。1=事务完成时进行中断。当主机控制器确定分割事务已经完成时，它将在下一个中断阈值断言硬件中断。
30	<b>页面选择 (P)。</b> 用于指示哪个数据页指针应与CurrentOffset字段连接以构造数据缓冲区指针（0选择页面0指针，1选择页面1）。当siTD失效时（活动位从1转换为0），主机控制器不需要写回该字段。
29分26秒	<b>保留。</b> 此字段是为将来使用而保留的，应设置为零。

25分16 秒	要传输的总字节数。此字段由软件初始化为此传输中预期的总字节数。最大值为1023 (3FFh)
------------	--

表3-11。SiTD传输状态和控制 (续)

一点	描述																								
15时8分	<p>□ 帧完成分割进度掩码 (C-prog-Mask)。主机控制器使用此字段来记录执行了哪些拆分完成。行为要求见第4.12.3.3.2节。</p>																								
7:0分	<p><b>地位</b>此字段记录主机控制器为此插槽执行的事务的状态。此字段是具有以下编码的位向量：</p> <table> <thead> <tr> <th>位</th> <th>定义</th> </tr> </thead> <tbody> <tr> <td>7</td> <td><b>主动。</b>通过软件设置为1, 以使主机控制器能够执行同步拆分事务。参见第4.12.3.3节。</td> </tr> <tr> <td>6</td> <td><b>呃。</b>当从事务转换器接收到ERR响应时, 主机控制器将其设置为1。</td> </tr> <tr> <td>5</td> <td><b>数据缓冲区错误。</b>在状态更新期间由主机控制器设置为1, 表示主机控制器无法跟上传入数据的接收(溢出)或在传输期间无法足够快地提供数据(欠载)。在欠载运行的情况下, 主机控制器将发送不正确的CRC(从而使端点处的数据无效)。如果发生超限情况, 则无需采取任何行动。</td> </tr> <tr> <td>4</td> <td><b>检测到故障。</b>当在此描述符生成的事务期间检测到“牙牙学语”时, 主机控制器在状态更新期间将其设置为1。</td> </tr> <tr> <td>3</td> <td><b>事务错误 (KactErr)。</b>在主机未收到来自设备的有效响应(超时、CRC、错误PID等)的情况下, 主机控制器在状态更新期间将其设置为1。此位仅针对in事务设置。</td> </tr> <tr> <td>2</td> <td><b>组合微型框架。</b>主机控制器检测到主机引起的延迟导致主机控制器错过所需的完整拆分事务。</td> </tr> <tr> <td>1</td> <td><b>拆分事务状态 (SplitXstate)。</b>操作细节见第4.12.3.3节。比特编码为：</td> </tr> <tr> <th>价值</th> <th>含义</th> </tr> <tr> <td>0b</td> <td><b>不开始拆分。</b>此值指示主机控制器在S掩码中遇到匹配时向端点发出“启动”拆分事务。</td> </tr> <tr> <td>1bDo完全拆分。</td> <td><b>完全拆分。</b>此值指示主机控制器在C掩码中遇到匹配时向端点发出完整拆分事务。</td> </tr> <tr> <td>0</td> <td><b>保留。</b>此位是为将来使用而保留的, 应设置为零。</td> </tr> </tbody> </table>	位	定义	7	<b>主动。</b> 通过软件设置为1, 以使主机控制器能够执行同步拆分事务。参见第4.12.3.3节。	6	<b>呃。</b> 当从事务转换器接收到ERR响应时, 主机控制器将其设置为1。	5	<b>数据缓冲区错误。</b> 在状态更新期间由主机控制器设置为1, 表示主机控制器无法跟上传入数据的接收(溢出)或在传输期间无法足够快地提供数据(欠载)。在欠载运行的情况下, 主机控制器将发送不正确的CRC(从而使端点处的数据无效)。如果发生超限情况, 则无需采取任何行动。	4	<b>检测到故障。</b> 当在此描述符生成的事务期间检测到“牙牙学语”时, 主机控制器在状态更新期间将其设置为1。	3	<b>事务错误 (KactErr)。</b> 在主机未收到来自设备的有效响应(超时、CRC、错误PID等)的情况下, 主机控制器在状态更新期间将其设置为1。此位仅针对in事务设置。	2	<b>组合微型框架。</b> 主机控制器检测到主机引起的延迟导致主机控制器错过所需的完整拆分事务。	1	<b>拆分事务状态 (SplitXstate)。</b> 操作细节见第4.12.3.3节。比特编码为：	价值	含义	0b	<b>不开始拆分。</b> 此值指示主机控制器在S掩码中遇到匹配时向端点发出“启动”拆分事务。	1bDo完全拆分。	<b>完全拆分。</b> 此值指示主机控制器在C掩码中遇到匹配时向端点发出完整拆分事务。	0	<b>保留。</b> 此位是为将来使用而保留的, 应设置为零。
位	定义																								
7	<b>主动。</b> 通过软件设置为1, 以使主机控制器能够执行同步拆分事务。参见第4.12.3.3节。																								
6	<b>呃。</b> 当从事务转换器接收到ERR响应时, 主机控制器将其设置为1。																								
5	<b>数据缓冲区错误。</b> 在状态更新期间由主机控制器设置为1, 表示主机控制器无法跟上传入数据的接收(溢出)或在传输期间无法足够快地提供数据(欠载)。在欠载运行的情况下, 主机控制器将发送不正确的CRC(从而使端点处的数据无效)。如果发生超限情况, 则无需采取任何行动。																								
4	<b>检测到故障。</b> 当在此描述符生成的事务期间检测到“牙牙学语”时, 主机控制器在状态更新期间将其设置为1。																								
3	<b>事务错误 (KactErr)。</b> 在主机未收到来自设备的有效响应(超时、CRC、错误PID等)的情况下, 主机控制器在状态更新期间将其设置为1。此位仅针对in事务设置。																								
2	<b>组合微型框架。</b> 主机控制器检测到主机引起的延迟导致主机控制器错过所需的完整拆分事务。																								
1	<b>拆分事务状态 (SplitXstate)。</b> 操作细节见第4.12.3.3节。比特编码为：																								
价值	含义																								
0b	<b>不开始拆分。</b> 此值指示主机控制器在S掩码中遇到匹配时向端点发出“启动”拆分事务。																								
1bDo完全拆分。	<b>完全拆分。</b> 此值指示主机控制器在C掩码中遇到匹配时向端点发出完整拆分事务。																								
0	<b>保留。</b> 此位是为将来使用而保留的, 应设置为零。																								

### 3.4.4 SiTD缓冲区指针列表 (加)

数据字4和5是用于传输的数据缓冲区页指针。此结构支持一个物理页面交叉。本节中每个D字的最高有效20位是4K(页)对齐的缓冲区指针。每个D字的最低有效12位被用作附加的传输状态。

表3-12。缓冲区页指针列表 (加)

一点	描述
31分12秒	<p><b>缓冲区指针列表。</b>数据字4和5的位[31:12]是4K页对齐的物理存储器地址。这些比特分别对应于物理地址比特[31:12]。</p> <p>每个指针中较低的12位如下所述进行定义和使用。字段P(参见表3-11)指定了当前活动指针。</p>

表3-12。缓冲区页指针列表 (加) (续)

一点	描述																				
11:0	<p><b>第0页:</b>  <b>当前偏移。</b>页面0指针的12个最低有效位是当前页面指针的当前字节偏移量 (用页面指示符 (P字段) 选择)。当siTD失效时 (活动位从1转换为0)，主机控制器不需要写回该字段。</p> <p>第1页指针的最低有效位分为三个子字段：</p> <p><b>第1页:</b></p> <table> <thead> <tr> <th>位</th> <th>描述</th> </tr> </thead> <tbody> <tr> <td>11:5</td> <td>保留。</td> </tr> <tr> <td>4:3</td> <td> <b>交易头寸 (TP)</b>。该字段与T-计数一起使用，以确定是发送所有、第一、中间还是最后的每个出站事务有效载荷。系统软件必须使用适当的起始值初始化此字段。主机控制器必须在传输的生存期内正确管理此状态。比特编码为：           </td> </tr> <tr> <td></td> <td> <table> <thead> <tr> <th>价值</th> <th>含义</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td><b>全部。</b>整个全速事务数据有效载荷在该事务中 (即小于或等于188字节)。</td> </tr> <tr> <td>01b</td> <td><b>开始。</b>这是大于188字节的全速事务的第一个数据有效载荷。</td> </tr> <tr> <td>10</td> <td><b>中等。</b>这是大于188字节的全速OUT事务的中间有效载荷。</td> </tr> <tr> <td>11b</td> <td><b>结束。</b>这是大于188字节的全速OUT事务的最后一个有效负载。</td> </tr> </tbody> </table> </td> </tr> <tr> <td>2:0</td> <td><b>交易计数 (T计数)</b>。软件使用此传输所需的OUT开始拆分次数初始化此字段。任何大于6的值都将被取消定义。</td> </tr> </tbody> </table>	位	描述	11:5	保留。	4:3	<b>交易头寸 (TP)</b> 。该字段与T-计数一起使用，以确定是发送所有、第一、中间还是最后的每个出站事务有效载荷。系统软件必须使用适当的起始值初始化此字段。主机控制器必须在传输的生存期内正确管理此状态。比特编码为：		<table> <thead> <tr> <th>价值</th> <th>含义</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td><b>全部。</b>整个全速事务数据有效载荷在该事务中 (即小于或等于188字节)。</td> </tr> <tr> <td>01b</td> <td><b>开始。</b>这是大于188字节的全速事务的第一个数据有效载荷。</td> </tr> <tr> <td>10</td> <td><b>中等。</b>这是大于188字节的全速OUT事务的中间有效载荷。</td> </tr> <tr> <td>11b</td> <td><b>结束。</b>这是大于188字节的全速OUT事务的最后一个有效负载。</td> </tr> </tbody> </table>	价值	含义	00b	<b>全部。</b> 整个全速事务数据有效载荷在该事务中 (即小于或等于188字节)。	01b	<b>开始。</b> 这是大于188字节的全速事务的第一个数据有效载荷。	10	<b>中等。</b> 这是大于188字节的全速OUT事务的中间有效载荷。	11b	<b>结束。</b> 这是大于188字节的全速OUT事务的最后一个有效负载。	2:0	<b>交易计数 (T计数)</b> 。软件使用此传输所需的OUT开始拆分次数初始化此字段。任何大于6的值都将被取消定义。
位	描述																				
11:5	保留。																				
4:3	<b>交易头寸 (TP)</b> 。该字段与T-计数一起使用，以确定是发送所有、第一、中间还是最后的每个出站事务有效载荷。系统软件必须使用适当的起始值初始化此字段。主机控制器必须在传输的生存期内正确管理此状态。比特编码为：																				
	<table> <thead> <tr> <th>价值</th> <th>含义</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td><b>全部。</b>整个全速事务数据有效载荷在该事务中 (即小于或等于188字节)。</td> </tr> <tr> <td>01b</td> <td><b>开始。</b>这是大于188字节的全速事务的第一个数据有效载荷。</td> </tr> <tr> <td>10</td> <td><b>中等。</b>这是大于188字节的全速OUT事务的中间有效载荷。</td> </tr> <tr> <td>11b</td> <td><b>结束。</b>这是大于188字节的全速OUT事务的最后一个有效负载。</td> </tr> </tbody> </table>	价值	含义	00b	<b>全部。</b> 整个全速事务数据有效载荷在该事务中 (即小于或等于188字节)。	01b	<b>开始。</b> 这是大于188字节的全速事务的第一个数据有效载荷。	10	<b>中等。</b> 这是大于188字节的全速OUT事务的中间有效载荷。	11b	<b>结束。</b> 这是大于188字节的全速OUT事务的最后一个有效负载。										
价值	含义																				
00b	<b>全部。</b> 整个全速事务数据有效载荷在该事务中 (即小于或等于188字节)。																				
01b	<b>开始。</b> 这是大于188字节的全速事务的第一个数据有效载荷。																				
10	<b>中等。</b> 这是大于188字节的全速OUT事务的中间有效载荷。																				
11b	<b>结束。</b> 这是大于188字节的全速OUT事务的最后一个有效负载。																				
2:0	<b>交易计数 (T计数)</b> 。软件使用此传输所需的OUT开始拆分次数初始化此字段。任何大于6的值都将被取消定义。																				

### 3.4.5 siTD反向链接指针

siTD的Dword 6仅仅是另一个调度链接指针。此指针始终为零，或引用siTD。此指针不能引用任何其他计划数据结构。有关运行模式的详细信息，请参见第4.12.3.3.2.1节。

表3-13。siTD反向链接指针

一点	描述
31分5秒	<b>siTD返回指针。</b> 此字段是指向siTD的物理内存指针。操作细节见第4.12.3.3.2.1节。
4分1秒	<b>保留。</b> 此字段保留以备将来使用。它应该设置为零。
0	<b>终止 (T)</b> 。1=siTD返回指针字段无效。0=siTD返回指针字段有效。使用请参考第4.12.3.3.2.1节。

## 3.5 队列元素传输描述符 (QTD)

此数据结构仅与队列头一起使用 (参见第3.6节)。此数据结构用于一个或多个USB事务。有关行为模型的完整描述，请参见第4.10节。此数据结构用于传输多达20480 (5\*4096) 个字节。该结构包含两个用于队列前进的结构指针，一个传输状态的D字和一个数据缓冲区指针的五元素数组。该结构是32个字节 (或一个32字节的高速缓存行)。此数据结构必须在物理上连续。

与此传输相关联的缓冲区必须实际上是连续的。缓冲区可以在任何字节边界上开始。缓冲区中的每个物理页都必须使用一个单独的缓冲区指针列表元素，无论缓冲区在物理上是否连续。

主机控制器对独立qTD的更新（主机控制器写入）仅在传输引退期间发生（见第4.10.4节）。以下位字段定义中对“qTD”的更新参考是队列头的qTD部分（见图3-7）。

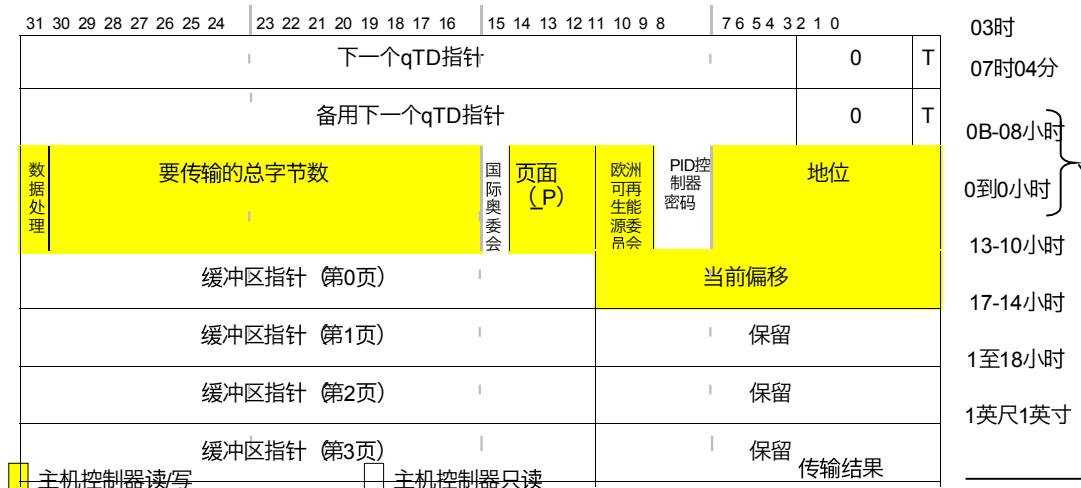


图3-6。队列元素传输描述符框图

队列元素传输描述符必须在32字节边界上对齐。

### 3.5.1 下一个qTD指针

元素传输描述符的第一个DWord是指向另一个传输元素描述符的指针。

表3-14。qTD下一个元素传输指针 (0字0)

一点	描述
31分5秒	下一个传输元素指针。此字段包含要处理的下一个qTD的物理内存地址。该字段分别对应于存储器地址信号[31:5]。
4分1秒	保留。这些位是保留的，它们的值对操作没有影响。
0	终止 (T)。1=指针无效。0=指针有效（指向有效的传输元素描述符）。此位向主机控制器指示队列中不再有有效条目。

### 3.5.2 备用下一个qTD指针

队列元素传输描述符的第二个DWord用于仅支持在短数据包上将数据流推进到下一个客户端缓冲区的硬件。更明确地说，当当前qTD由于短数据包而失效时，主机控制器将始终使用该指针。

表3-15。qTD备用下一个元素传输指针 (1字1)

一点	描述
31分5秒	<b>备用下一个传输元素指针。</b> 此字段包含当前qTD执行遇到短数据包(针对in事务)时要处理的下一个qTD的物理内存地址。该字段分别对应于存储器地址信号[31:5]。
4分1秒	<b>保留。</b> 这些位是保留的,它们的值对操作没有影响。
0	<b>终止(T)。</b> 1=指针无效。0=指针有效(指向有效的传输元素描述符)。此位向主机控制器指示队列中不再有有效条目。

### 3.5.3 qTD代码

队列元素传输描述符的第三个DWORD包含主机控制器执行USB事务所需的大部分信息(剩余的端点寻址信息在队列头中指定)。

注:字段描述队列头中定义的前向引用字段(第3.6节)。必要时,这些前向引用前面加上QH。符号

表3-16。qTD令牌 (1Word 2)

一点	描述
31	<b>数据切换。</b> 这是数据切换序列位。此位的使用取决于队列头中数据切换控制位的设置。数据切换的操作模型见第4.10节。
30分16秒	<b>要传输的总字节数。</b> 此字段指定要使用此传输描述符移动的字节总数。仅当事务成功完成时,此字段才会按事务期间实际移动的字节数递减。 软件可在此字段中存储的最大值为5*4K(5000H)。这是5个页面指针可以访问的最大字节数。 请参阅第4.10.3节,了解当该字段减为零时主机控制器要求的完整说明。 如果当主机控制器获取该传输描述符时该字段的值为零(并且设置了有效位),则主机控制器执行零长度事务并使传输描述符失效。 OUT传输不要求要传输的总字节数是QHD.最大数据包长度的偶数倍。如果软件为OUT传输建立这样一个传输描述符,则最后一个事务将始终小于QHD.最大数据包长度。
15	<b>完成时中断(OC)。</b> 如果此位设置为1,则它指定当此qTD完成时,主机控制器应在下一个中断阈值发出中断。
14时12分	<b>当前页面(C_Page)。</b> 该字段用作qTD缓冲区指针列表的索引(第4.10.6节)。有效值在0H至4H范围内。当qTD失效时,主机控制器不需要写回该字段(见第4.10.4节)。

表3-16。qTD代码 (0字2) (续)

一点	描述												
11时10分	<p><b>错误计数器 (CERR)</b>。此字段是一个2位递减计数器，用于跟踪执行此qTD时检测到的连续错误数。如果此字段在设置过程中被编程为非零值，则如果事务失败，主机控制器会递减计数并将其写回qTD。如果计数器从1计数到0，则主机控制器将qTD标记为非活动，并将导致CERR递减到0的错误的<i>Halted</i>位设置为1和错误状态位。如果USBINTR寄存器中的USB错误中断启用位设置为1，则会产生中断。如果HCD在设置期间将该字段编程为零，则主机控制器将不会计算该qTD的错误，并且该qTD重试次数也没有限制。请注意，中间执行状态的回写是到队列头覆盖区域，而不是qTD。</p> <table> <thead> <tr> <th>错误</th> <th>递减计数器</th> <th>错误</th> <th>递减计数器</th> </tr> </thead> <tbody> <tr> <td>事务错误<sup>1</sup></td> <td>是号</td> <td>数据缓冲区错误<sup>3</sup> 失速探测到<sup>1</sup></td> <td>号 号故障</td> </tr> <tr> <td>无错误<sup>2</sup></td> <td></td> <td></td> <td></td> </tr> </tbody> </table> <p><sup>1</sup>检测到巴贝尔或失速会自动停止排队。因此，计数不递减。  <sup>2</sup>如果EPS字段指示HS设备或队列头在异步调度中 (<i>PIDCode</i>指示in或OUT)，并且总线事务完成，并且主机控制器没有检测到事务错误，则主机控制器应重置CERR以扩展该事务的错误总数。例如，每次成功完成事务时，CERR都应重置为最大值(3)。如果事务开始时的值为00b，则主机控制器决不能重置此字段。  <sup>3</sup>数据缓冲区错误是主机问题。它们不计入设备的重试次数。</p> <p><b>注：</b>当EPS字段被编程为指示全速或低速设备的值时，软件不得将CErr编程为零值。这种组合可能导致未定义的行为。</p>	错误	递减计数器	错误	递减计数器	事务错误 <sup>1</sup>	是号	数据缓冲区错误 <sup>3</sup> 失速探测到 <sup>1</sup>	号 号故障	无错误 <sup>2</sup>			
错误	递减计数器	错误	递减计数器										
事务错误 <sup>1</sup>	是号	数据缓冲区错误 <sup>3</sup> 失速探测到 <sup>1</sup>	号 号故障										
无错误 <sup>2</sup>													
9点8分	<p><b>PID代码</b>。此字段是令牌的编码，应用于与此传输描述符相关联的事务。编码为：</p> <table> <tbody> <tr> <td>00bOUT E1H)</td> <td>01bIN通证</td> <td>通证生成通证 (生成令牌 (69H))</td> </tr> <tr> <td>10bSETUP令牌</td> <td></td> <td>生成令牌 (2DH) 如果端点是中断传输类型，例如。□队列头中的帧S掩码字段为非零。)</td> </tr> <tr> <td>11b</td> <td>保留</td> <td></td> </tr> </tbody> </table>	00bOUT E1H)	01bIN通证	通证生成通证 (生成令牌 (69H))	10bSETUP令牌		生成令牌 (2DH) 如果端点是中断传输类型，例如。□队列头中的帧S掩码字段为非零。)	11b	保留				
00bOUT E1H)	01bIN通证	通证生成通证 (生成令牌 (69H))											
10bSETUP令牌		生成令牌 (2DH) 如果端点是中断传输类型，例如。□队列头中的帧S掩码字段为非零。)											
11b	保留												
7:0分	<p><b>地位</b>主机控制器使用此字段将各个命令执行状态传回HCD。此字段包含在此qTD上执行的最后一笔交易的状态。比特编码为：</p> <table> <thead> <tr> <th>位</th> <th>状态字段描述</th> </tr> </thead> <tbody> <tr> <td>7</td> <td><b>主动</b>。软件将其设置为1，以启用主机控制器执行事务。有关主机控制器何时将该位从1转换为0的操作细节，请参阅第4.10.3节。</td> </tr> </tbody> </table>	位	状态字段描述	7	<b>主动</b> 。软件将其设置为1，以启用主机控制器执行事务。有关主机控制器何时将该位从1转换为0的操作细节，请参阅第4.10.3节。								
位	状态字段描述												
7	<b>主动</b> 。软件将其设置为1，以启用主机控制器执行事务。有关主机控制器何时将该位从1转换为0的操作细节，请参阅第4.10.3节。												

表3-16。qTD代码（0字2）（续）

一点	描述												
7:0分 (续)	<p><b>位 状态字段描述</b></p> <p>6 <b>停止。</b>在状态更新期间，主机控制器将其设置为1，以指示此qTD所寻址的设备/端点发生严重错误。这可能是由牙牙学语、错误计数器倒计时到零或在事务期间从设备接收到STALL握手引起的。每当事务导致Halted位被设置为1时，Active位也被设置为0。</p> <p>5 <b>数据缓冲区错误。</b>在状态更新期间由主机控制器设置为1，表示主机控制器无法跟上传入数据的接收（溢出）或在传输期间无法足够快地提供数据（欠载）。第4.15.1.1.2节定义了发生欠载运行错误时主机控制器的要求。如果发生溢出情况，主机控制器将强制USB上出现超时条件，从而使源处的事务无效。如果主机控制器将该位设置为1，则在传输的持续时间内该位保持为1。</p> <p><b>4检测到 故障。当事务期间检测到“牙牙学语”时，主机控制器在状态更新期间将其设置为1。除了设置该位之外，主机控制器还将Halted位设置为1。由于“牙牙学语”被认为是传输的致命错误，因此将Halted位设置为1可以确保不再因该描述符而发生事务。</b></p> <p>3事务 <b>错误 (XactErr)。</b>在主机未从设备接收到有效响应（超时、CRC、错误PID等）的情况下，在状态更新期间由主机控制器设置为1。有关影响此位的条件摘要，请参阅第4.15.1.1节。如果主机控制器将该位设置为1，则在传输的持续时间内该位保持为1。</p> <p>2 <b>组合微型框架。</b>除非QH.EPS字段指示完全或低速端点，并且队列头在周期性列表中，否则此位将被忽略。当主机控制器检测到主机引起的延迟导致主机控制器错过所需的完整拆分事务时，设置该位。如果主机控制器将该位设置为1，则在传输的持续时间内该位保持为1。</p> <p>1 <b>拆分事务状态 (SplitXstate)。</b>除非QH.EPS字段指示完全或低速端点，否则主机控制器将忽略此位。当是全速或低速设备时，主机控制器使用此位来跟踪拆分事务的状态。用于管理该状态位和拆分事务协议的主机控制器的功能要求取决于端点是在周期性调度中还是在异步调度中。操作细节见第4.12节。比特编码：</p> <table> <thead> <tr> <th>价值</th> <th>含义</th> </tr> </thead> <tbody> <tr> <td>0b</td> <td><b>不开始拆分。</b>此值指示主机控制器向终结点发出“启动”拆分事务。</td> </tr> <tr> <td>1b</td> <td><b>Do完全拆分。</b>此值指示主机控制器向终结点发出完整拆分事务。</td> </tr> </tbody> </table> <p>0 <b>平状态 (P) /ERR。</b>如果QH.EPS字段指示高速设备，PID_Code指示OUT端点，则这是Ping协议的状态位。操作细节见第4.11节。比特编码：</p> <table> <thead> <tr> <th>价值</th> <th>含义</th> </tr> </thead> <tbody> <tr> <td>0b</td> <td><b>退出。</b>该值指示主机控制器向端点发出OUT PID。</td> </tr> <tr> <td>1</td> <td><b>打乒乓球。</b>此值指示主机控制器向端点发出PING PID。</td> </tr> </tbody> </table> <p>如果QH.EPS字段不指示高速设备，则该字段被用作错误指示位。每当周期性拆分事务接收到ERR握手时，主机控制器就会将其设置为1。</p>	价值	含义	0b	<b>不开始拆分。</b> 此值指示主机控制器向终结点发出“启动”拆分事务。	1b	<b>Do完全拆分。</b> 此值指示主机控制器向终结点发出完整拆分事务。	价值	含义	0b	<b>退出。</b> 该值指示主机控制器向端点发出OUT PID。	1	<b>打乒乓球。</b> 此值指示主机控制器向端点发出PING PID。
价值	含义												
0b	<b>不开始拆分。</b> 此值指示主机控制器向终结点发出“启动”拆分事务。												
1b	<b>Do完全拆分。</b> 此值指示主机控制器向终结点发出完整拆分事务。												
价值	含义												
0b	<b>退出。</b> 该值指示主机控制器向端点发出OUT PID。												
1	<b>打乒乓球。</b> 此值指示主机控制器向端点发出PING PID。												

### 3.5.4 qTD缓冲区页指针列表

队列元素传输描述符的最后五个D字是物理内存地址指针的数组。这些指针引用数据缓冲区的各个页面。操作要求参见第4.10.6节。

系统软件将当前偏移字段初始化为当前页面的起始偏移，其中当前页面是通过*C\_page*字段中的值选择的。

**表3-17。qTD缓冲区指针 (D字3-7)**

一 点	描述
31分12秒	<p><b>缓冲区指针列表。</b>列表中的每个元素都是一个4K页面对齐的物理内存地址。每个指针中的低12位是保留的（第一个除外），因为每个内存指针必须引用4K页的开头。字段<i>C_Page</i>（见表3-16）指定当前活动指针。</p> <p>当提取传输元素描述符时，使用<i>C_Page</i>（类似于用于选择数组元素的数组索引）来选择起始缓冲区地址。如果事务跨越4K缓冲区边界，则主机控制器必须检测数据流中的页面跨越边界，递增<i>C_page</i>并前进到列表中的下一个缓冲区指针，并通过新的缓冲区指针结束事务。有关主机控制器操作要求的完整说明，请参阅4.10.6。</p>
11:0	<p><b>当前偏移（保留）。</b>该字段在除第一个指针（例如，第0页）之外的所有指针中都被保留。主机控制器应忽略所有保留位。对于页面0的当前偏移量解释，此字段是当前页面的字节偏移量（由<i>C_page</i>选择）。</p> <p>当qTD失效时，主机控制器不需要写回该字段（见第4.10.4节）。软件应确保保留字段初始化为零。</p>

### 3.6 排队负责人



图3-7。排队头结构布局

#### 3.6.1 队列头水平链接指针

队列头的第一个DWord包含一个链接指针，指向该队列中任何所需处理完成后要处理的下一个数据对象，以及下面定义的控制位。

该指针可以引用一个队列头（第3.6节）或一个等时传输描述符（第3.3节和第3.4节）。它不能引用队列元素传输描述符（见第3.5节）。

表3-18。队列头D字0

一点	描述
31分5秒	队列头水平链接指针 (QHLP)。该字段包含水平列表中要处理的下一个数据对象的地址，并分别对应于存储器地址信号[31:5]。
4点3分	保留。这些位必须写成0。

表3-18。队列头D字0 (续)

一点	描述						
2:1	<p><b>QH/ 6) iTD/FSTN选择 (类型)</b>。该字段向硬件指示链接指针所引用的项是iTД、siTД还是QH。这允许主机控制器在项目被提取后对其执行正确类型的处理。值编码为：</p> <table> <thead> <tr> <th>价值</th> <th>含义</th> </tr> </thead> <tbody> <tr> <td>00biTD (等时传输描述符)</td> <td>01bQH (队列头)</td> </tr> <tr> <td>10bsiTD (拆分事务等时传输描述符)</td> <td>11bFSTN (帧跨遍历节点)</td> </tr> </tbody> </table>	价值	含义	00biTD (等时传输描述符)	01bQH (队列头)	10bsiTD (拆分事务等时传输描述符)	11bFSTN (帧跨遍历节点)
价值	含义						
00biTD (等时传输描述符)	01bQH (队列头)						
10bsiTD (拆分事务等时传输描述符)	11bFSTN (帧跨遍历节点)						
0	<p><b>终止 (T)</b>。1=最后一个QH (指针无效)。0=指针有效。如果队列头在周期列表的上下文中，则该字段中的一位向主机控制器指示这是周期列表的末尾。当队列头在异步调度中时，主机控制器会忽略此位。软件必须确保主机控制器可到达的队列头始终具有有效的水平链路指针。参见第4.8.2节</p>						

### 3.6.2 端点功能/特征

队列头的第二个和第三个D字指定关于端点的静态信息。此信息在终结点的生存期内不会更改。该地区有三种类型的信息：

- ① **端点特征**。这些是USB端点特性，包括寻址、最大数据包大小和端点速度。
- ② **端点功能**。这些是端点的可调整参数。它们影响主机控制器如何管理端点数据流。参见第4.10节。
- ③ **拆分交易特征**。此数据结构用于管理全速和低速数据流，以便通过USB 2.0集线器事务转换器的拆分事务进行批量、控制和中断。还有其他字段用于寻址集线器和调度协议事务（用于周期性）。参见第4.12节。

主机控制器不得修改此区域中的位。

表3-19。端点特征：队列头DWord 1

一点	描述				
31分28秒	<b>裸计数重新加载 (RL)</b> 。此字段包含一个值，主机控制器使用该值来重新加载Nak Counter字段。				
27	<b>控制端点标志 (C)</b> 。如果QH.EPS字段指示端点不是高速设备，并且端点是控制端点，则软件必须将此位设置为1。否则，它应该始终将该位设置为零。				
26分16秒	<b>最大数据包长度</b> 。这直接对应于关联端点的最大数据包大小 ( <i>wMaxPacketSize</i> )。此字段可能包含的最大值为0x400 (1024)。				
15	<b>填海区负责人名单旗 (H)</b> 。该位由系统软件设置，用于将队列头标记为回收列表的头。操作模式见第4.8节。				
14	<p><b>数据切换控制 (DTC)</b>。此位指定主机控制器应在覆盖转换上的何处获取初始数据切换。</p> <table> <tbody> <tr> <td>0b</td> <td>忽略传入qTD中的DT位。主机控制器在队列头中保留DT位。</td> </tr> <tr> <td>1b</td> <td>初始数据切换来自传入的qTD DT位。主机控制器从qTD中的DT位替换队列头中的DT比特。</td> </tr> </tbody> </table>	0b	忽略传入qTD中的DT位。主机控制器在队列头中保留DT位。	1b	初始数据切换来自传入的qTD DT位。主机控制器从qTD中的DT位替换队列头中的DT比特。
0b	忽略传入qTD中的DT位。主机控制器在队列头中保留DT位。				
1b	初始数据切换来自传入的qTD DT位。主机控制器从qTD中的DT位替换队列头中的DT比特。				

表3-19。端点特征：队列头DWord 1 (续)

一点	描述										
13时12分	<p><b>端点速度 (EPS)。</b>这是关联端点的速度。位组合为：</p> <table> <thead> <tr> <th>价值</th> <th>含义</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>全速 (12Mbs)</td> </tr> <tr> <td>01b</td> <td>低速 (1.5Mbs)</td> </tr> <tr> <td>10b</td> <td>高速 (480Mb/s)</td> </tr> <tr> <td>11b</td> <td>保留</td> </tr> </tbody> </table> <p>主机控制器不得修改此字段。</p>	价值	含义	00b	全速 (12Mbs)	01b	低速 (1.5Mbs)	10b	高速 (480Mb/s)	11b	保留
价值	含义										
00b	全速 (12Mbs)										
01b	低速 (1.5Mbs)										
10b	高速 (480Mb/s)										
11b	保留										
11点8分	<b>端点编号 (Endpt)。</b> 此4位字段选择用作数据源或汇点的设备上的特定端点编号。										
7.	<p><b>在下一个事务上停用 0。</b>该位由系统软件用来请求主机控制器将活动位设置为零。完整操作细节见第4.12.2.5节。</p> <p>此字段仅在队列头位于定期计划中且EPS字段指示全速或低速端点时有效。当队列头在异步调度中或EPS字段指示高速设备产生未定义的结果时，将此位设置为1。</p>										
6点0分	<b>设备地址。</b> 此字段选择用作数据源或汇点的特定设备。										

表3-20。端点功能：队列头DWord 2

一点	描述						
31时30分	<p><b>高带宽管道乘法器 (Mult)。</b>该字段是一个乘数，用于将主机控制器作为主机控制器在当前执行中可以提交给端点的连续数据包的数量。主机控制器作出简化的假设，即软件正确地初始化该字段（无论队列头在调度中的位置或其他运行时参数如何）。有关正确的软件操作模型，请参见第4.10.3节。有效值为：</p> <table> <thead> <tr> <th>价值</th> <th>含义</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>保留。此字段中的零会产生未定义的结果。</td> </tr> <tr> <td>01b</td> <td>每个微帧 为该 端 点 发布一 个 事 务 10b 每 个 微框架为此端点发布两 个 事 务 11b 每个微框为此端点发布 三个事务</td> </tr> </tbody> </table>	价值	含义	00b	保留。此字段中的零会产生未定义的结果。	01b	每个微帧 为该 端 点 发布一 个 事 务 10b 每 个 微框架为此端点发布两 个 事 务 11b 每个微框为此端点发布 三个事务
价值	含义						
00b	保留。此字段中的零会产生未定义的结果。						
01b	每个微帧 为该 端 点 发布一 个 事 务 10b 每 个 微框架为此端点发布两 个 事 务 11b 每个微框为此端点发布 三个事务						
29分23秒	<b>端口号。</b> 主机控制器将忽略此字段，除非EPS字段指示全速或低速设备。该值是USB 2.0集线器上的端口号标识符（用于位于以下设备地址Hub Addr的集线器），与此端点关联的全速或低速设备连接到该端口号标识符之下。此信息用于拆分事务协议。参见第4.12节。						
22时16分	<b>集线器地址。</b> 主机控制器将忽略此字段，除非EPS字段指示设备已满或低速。该值是USB 2.0集线器的USB设备地址，与该端点关联的全速或低速设备连接在该集线器之下。此字段用于拆分事务协议。参见第4.12节。						

表3-20。端点功能 :队列头DWord 2 (续)

一点	描述
15时8分	<p><b>拆分完成掩码(□帧C掩码)</b>。主机控制器将忽略此字段，除非EPS字段指示此设备是低速或全速设备，并且此队列头在周期列表中。此字段（以及Active和SplitX状态字段）用于确定主机控制器应在哪些微帧期间执行完整的拆分事务。当满足使用此字段的条件时，此字段中的零值具有未定义的行为。</p> <p>如果FRINDEX寄存器位解码到□帧C-Mask字段为1，则此队列头是事务执行的候选。请参阅第4.10.3节，了解在主机控制器执行事务之前必须满足的标准的综合列表。</p> <p>此掩码集中可能有一个以上的位。参见第4.12.2.1节。</p>
7:0分	<p><b>中断计划掩码(□帧S掩码)</b>。此字段用于所有端点速度。当队列头在异步调度上时，软件应将此字段设置为零。此字段中的非零值表示中断端点。</p> <p>主机控制器使用FRINDEX寄存器的三个低阶位的值作为该位向量中的位位置的索引。如果□帧S掩码字段在索引位位置有一个，则该队列头是事务执行的候选者。请参阅第4.10.3节，了解在主机控制器执行事务之前必须满足的标准的综合列表。</p> <p>如果EPS字段指示端点是高速端点，则执行的事务由包含在执行区域中的PID_Code字段确定。</p> <p>该字段还用于支持拆分事务类型：中断 (IN/OUT)。参见第节 4.12.2.当该字段为非零时，且EPS字段指示这是一个全速或低速设备时，此条件成立。</p> <p>此字段中的零值与周期帧列表中的现有值相结合会产生未定义的结果。</p>

### 3.6.3 传输覆盖

该区域中的九个D字表示主机控制器的事务工作空间。一般的操作模型是主机控制器可以检测覆盖区域是否包含活动传输的描述。如果它不包含活动传输，则它将跟随“队列头水平链接指针”到达下一个队列头。主机控制器永远不会跟随下一个传输队列元素或备用队列元素指针，除非它正在主动尝试推进队列（见第4.10节）。在传输期间，主机控制器将在覆盖区域中保持传输的增量状态。传输完成后，结果会被写回原始队列元素。主机控制器如何使用该区域的完整操作模型如第4.10节所述。

队列头的DWord 3包含指向当前与覆盖相关联的源qTD的指针。在传输完成之后，主机控制器使用该指针将覆盖区域写回到源qTD中。

表3-21。当前qTD链接指针

一点	描述
31分5秒	<b>当前元素事务描述符链接指针。</b> 该字段包含该队列中正在处理的当前事务的地址，并分别对应于存储器地址信号[31:5]。
4点	<b>保留 (R)。</b> 当使用该值作为写入数据的地址时，主机控制器会忽略这些位。实际值可能因使用情况而异。

队列头的D字4-11是事务覆盖区域。该区域具有与第3.5节中定义的队列元素传输描述符相同的基本结构。队列头利用图3-7中定义的页面指针的保留字段来实现对拆分事务状态的跟踪。

该区域被表征为覆盖，因为当队列前进到下一个队列元素时，源队列元素被合并到该区域上。此区域为传输提供执行缓存。部分

4.10描述了队列前进过程中，队列头中的哪些字段被重写。

表3-22。叠加中位的主机控制器规则 (D字5、6、8和9)

德沃	一点	描述
5.	4分1秒	<b>裸体柜台 (NakCnt) □</b> 该字段是每当与该队列头相关联的端点的事务导致Nak或Nyet响应时主机控制器递减的计数器。在回收列表的第一次传递期间(相对于异步列表重新启动条件)执行事务之前，将从RL重新加载此计数器。参见第4.9节。它也在覆盖期间从RL加载。参见第4.10.2节。
6.	31	<b>数据切换。</b> 数据切换控件控制在执行覆盖操作时主机控制器是否保留此位。
6.	15	<b>完成时中断 (OC)。</b> 当执行覆盖操作时，IOC控制位总是从源qTD继承。
6.	11时10分	<b>错误计数器 (C_ERR)。</b> 这个两位字段在覆盖期间从qTD复制，并在队列前进期间写回。详见表3-16。
6.	0	<b>Ping状态 (P) /ERR。</b> 如果EPS字段指示高速端点，则在覆盖操作期间应保留此字段。
8.	7:0分	<b>拆分事务完成拆分进度 (C-prog-mask)。</b> 在任何覆盖过程中，此字段都会初始化为零。 此字段用于跟踪中断拆分事务的进度。有关操作模型的详细信息，请参见第4.12.2节。
9	4点	<b>拆分事务帧标记 (帧标记)。</b> 在任何覆盖过程中，此字段都会初始化为零。 此字段用于跟踪中断拆分事务的进度。有关操作模型的详细信息，请参见第4.12.2节。
9	11时5分	<b>S字节。</b> 在激活qTD之前，软件必须确保qTD中的S字节字段为零。 此字段用于跟踪IN或OUT分割事务期间发送或接收的字节数。有关操作模型的详细信息，请参阅第4.12.2节。

### 3.7 周期性框架跨度遍历节点 (FSTN)

此数据结构仅用于管理跨越主机帧边界的全速和低速事务。完整操作细节见第4.12.2.2节。软件不得在异步计划中使用FSTN。异步计划中的FSTN会导致未定义的行为。软件不得将FSTN功能与HCIVERSION寄存器指示低于0096h的修订实现的主机控制器一起使用。FSTN不是为0.96之前的实现定义的，它们的使用将产生未定义的结果。

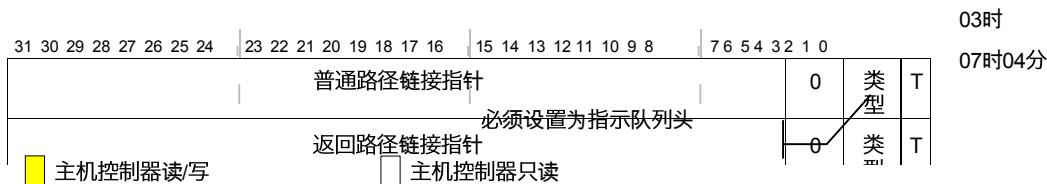


图3-8。框架跨横向节点结构布置图

#### 3.7.1 FSTN正常路径指针

FSTN的第一个DWord包含指向下一个计划对象的链接指针。此对象可以是任何有效的定期计划数据类型。

一点	描述						
31分5秒	正常路径链接指针 (NPLP)。该字段包含周期表中要处理的下一个数据对象的地址，并分别对应于存储器地址信号[31:5]。						
4点3分	保留。这些位必须写成0。						
2:1	<b>QH/ (s )iT D/FSTN选择 (类型)</b> 。该字段向主机控制器指示所引用的项目是iT D/siT D、QH还是FSTN。这允许主机控制器在项目被提取后对其进行正确类型的处理。值编码为： <table border="0"> <tr> <th>价值</th> <th>含义</th> </tr> <tr> <td>00biTD (等时传输描述符)</td> <td>01bQH (队列头)</td> </tr> <tr> <td>10bsiT D (拆分事务等时传输描述符)</td> <td>11bFSTN (帧跨遍历节点)</td> </tr> </table>	价值	含义	00biTD (等时传输描述符)	01bQH (队列头)	10bsiT D (拆分事务等时传输描述符)	11bFSTN (帧跨遍历节点)
价值	含义						
00biTD (等时传输描述符)	01bQH (队列头)						
10bsiT D (拆分事务等时传输描述符)	11bFSTN (帧跨遍历节点)						
0	终止 (T)。1=链接指针字段无效。0=链接指针有效。						

### 3.7.2 FSTN反向路径链接指针

FTSN节点的第二个DWord包含指向队列头的链接指针。如果该指针中的T位为零，则该FSTN为保存位置指示器。其类型字段必须由软件设置，以指示目标数据结构是队列头。如果此指针中的T位设置为1，则此FSTN为恢复指示器。当T位为1时，主机控制器忽略Typ字段。

一点	描述
31分5秒	反向路径链接指针 (BPLP)。此字段包含队列头的地址。该字段分别对应于存储器地址信号[31:5]。
4点3分	保留。这些位必须写成0。
2:1	类型。软件必须确保将此字段设置为指示目标数据结构是队列头。此字段中的任何其他值都会产生未定义的结果。
0	终止(T)。1=链路指针字段无效(即主机控制器不得使用位[31:5](与CTRLDSSEGMENT寄存器结合使用,如果适用)作为有效的存储器地址)。该值还表示该FSTN是一个恢复指示器。 0=链路指针有效(即,主机控制器可以使用位[31:5](如果适用,与CTRLDSSEGMENT寄存器组合)作为有效的存储器地址)。该值还表示该FSTN是一个保存位置指示器。

## 4. 运营模式

通用操作模型用于增强型接口主机控制器硬件和增强型接口主控制器驱动程序(通常称为系统软件)。EHCI主机控制器的每个重要操作特征在单独的章节中进行了讨论。每一节都介绍了主机控制器硬件的操作模型要求。在适当的情况下,还介绍了功能的推荐系统软件操作模型。

### 4.1 主机控制器初始化

当系统引导时,主机控制器被枚举,分配寄存器空间的基址, BIOS将FLADJ寄存器设置为系统特定值。初始通电或 $HCReset$ (硬件或通过USBCMD寄存器中的 $HCReset$ 位)后,所有操作寄存器将处于其默认值,如表4-1所示。硬件复位后,只有未包含在辅助电源井中的操作寄存器才会处于其默认值。

表4-1. 操作寄存器空间的默认值

操作寄存器	默认值(重置后)
美国边境管理局	00080000h(00080B00h,如果异步调度园区能力为一)
USBSTS公司	00001000小时
美国银行	00000000小时
FRINDEX公司	00000000小时
控制段	00000000小时
牙周膜基	未定义
异步地址	未定义
配置标志	00000000小时
端口SC	00002000h(带PPC设置为1);00003000h(PPC设置为零)

为了初始化主机控制器,软件应执行以下步骤

- ① 用4 GB段对CTRLDSSEGMENT寄存器进行编程,其中分配了所有接口数据结构。
- ② 将适当的值写入USBINTR寄存器以启用适当的中断。
- ③ 将周期帧列表的基址写入周期基址寄存器。如果在周期性调度中没有工作项,则周期性帧列表的所有元素的T位都应设置为1。
- ④ 写入USBCMD寄存器以设置所需的中断阈值、帧列表大小(如果适用),并通过设置运行/停止位打开主机控制器。
- ⑤ 写入一个1到CONFIGFLAG寄存器,将所有端口路由到EHCI控制器(见第4.2节)。

此时,主机控制器已启动并运行,端口寄存器将开始报告设备连接等。系统软件可以通过重置过程枚举端口(其中端口处于启用状态)。此时,端口处于活动状态,SOF出现在启用端口的高速端口下方,但计划尚未启用。EHCI主机控制器不会将SOF传输到启用的全速或低速端口。

为了通过异步调度与设备通信,系统软件必须使用控制或大容量队列头的地址写入ASYNDLISTADDR寄存器。然后,软件必须通过向USBCMD寄存器中的异步调度启用位写入一来启用异步调度。为了通过定期计划与设备通信,系统软件必须启用

通过向USBCMD寄存器中的周期性调度启用位写入一来执行周期性调度。请注意，可以在重置（和启用）第一个端口之前打开计划。

无论何时写入USBCMD寄存器，系统软件都必须确保根据预期操作保留适当的位。

## 4.2 端口路由和控制

USB 2.0主机控制器（如第1.1节所述）由一个高速主机控制器组成，该控制器实现EHC编程接口和0到N个USB 1.1配套主机控制器。配套主机控制器（cHC）可以是通用或开放式主机控制器规范的实现。此配置用于提供所需的完整USB 2.0定义端口功能；例如，每个端口的低速、全速和高速能力。图4-1显示了端口路由逻辑的简单框图及其与USB 2.0主机控制器内的高速和配套主机控制器的关系。

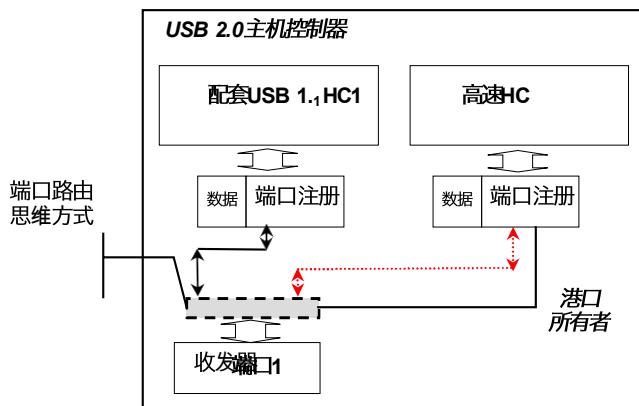


图4-1。USB 2.0主机控制器端口路由框图示例

每个物理端口有一个收发器，每个主机控制器模块都有自己的端口状态和控制寄存器。EHC控制器为每个端口都有端口状态和控制寄存器。每个配套主机控制器只有操作所需的端口控制和状态寄存器。每个收发器可以由EHC控制器或一个伴随主机控制器控制。路由逻辑位于收发器与端口状态和控制寄存器之间。<sup>2</sup>端口路由逻辑由源自EHC控制器的信号控制。EHC控制器具有全局路由策略控制字段和每个端口所有权控制字段。配置标志（CF）位（在第2.3.8节中定义）是全局路由策略控制。在通电或重置时，默认路由策略是到配套控制器（如果存在）。如果系统不包括EHC控制器的驱动程序，而主机控制器包括配套控制器，则端口仍将在全速和低速模式下工作（假设系统包括配套控制器的驱动）。通常，当EHC拥有端口时，伴随主机控制器的端口寄存器看不到来自收发器的连接指示。类似地，当伴随主机控制器拥有端口时，EHC控制器的端口寄存器看不到来自收发器的连接指示。以下各节介绍了端口路由逻辑规则的详细信息。

如果USB 2.0主机控制器包括配套控制器，则必须将其实现为多功能PCI设备。配套主机控制器的功能编号必须小于EHC控制器功能编号。相对于与该EHC控制器相关联的伴随主机控制器，EHC控制器必须是更大的函数号。如果PCI设备

---

<sup>2</sup>路由逻辑不应在收发器的480MHz时钟域中实现。

实现仅包含一个EHCI控制器(即没有配套控制器或其他PCI功能),则根据PCI规范,EHCI主机控制器必须为功能零。

结构参数寄存器(*HCSPARAMS*)中的*N\_CC*字段指示控制器实现是否包括伴随主机控制器。当*N\_CC*具有非零值时,存在伴随主机控制器。如果*N\_CC*的值为零,则主机控制器实现不包括伴随主机控制器。如果主机控制器根端口暴露在全速或低速设备的附件中,则在重置期间,端口将始终无法通过高速啁啾,并且端口将不会启用。系统软件可以将非法情况通知用户。这种类型的实现需要将USB 2.0集线器连接到根端口,以提供完整和低速的设备连接。

系统软件使用主机控制器能力寄存器中的信息来确定端口如何路由到配套主机控制器。参见第2.2.5节和第2.2.<sup>3</sup><sub>3</sub>节

#### 4.2.1 通过EHCI配置(*CF*)位的端口路由控制

USB 2.0主机控制器中的每个端口都可以路由到单个配套主机控制器或EHCI主机控制器。端口路由逻辑由EHCI HC中的两种机制控制:主机控制器全局标志和每个端口控制。配置标志(*CF*)位(在第2.3.8节中定义)用于全局设置路由逻辑的策略。每个端口寄存器都有一个端口所有者控制位,允许EHCI驱动程序显式控制各个端口的路由。每当*CF*位从0转换为1时(此转换仅在程序控制下可用),端口路由无条件地将所有端口寄存器路由到EHCI HC(所有端口所有者位变为0)。当*CF*位是一个时,EHCI驱动程序可以通过端口所有者控制位控制各个端口的路由。同样,每当*CF*位从1转换为0时(由于Aux电源应用、*HCRESET*或软件将0写入*CF*位),端口路由无条件地将所有端口寄存器路由到适当的伴随HC。EHCI HC的*CF*位的默认值(在Aux电源应用或*HCRESET*之后)为零。表4-2根据EHCI HC的*CF*位的值总结了所有端口的默认路由。

端口的视图取决于当前所有者。通用或开放式配套主机控制器将看到端口寄存器位与适当的规范一致。EHCI主机控制器所需的端口位定义对配套主机控制器不可见。

表4-2. 取决于EHCI HC CF位的默认端口路由

HS CF位	默认端口所有权	解释
0磅	同伴HC	配套主机控制器拥有端口,并且系统中仅支持全速和低速设备。确切的端口分配取决于实现。 在此配置中,端口仅作为全速端口和低速端口运行。
1磅	EHCI碳氢化合物	EHCI主机控制器对所有端口具有默认所有权。当端口所有者是EHCI HC时,路由逻辑禁止设备连接事件到达伴随HC的端口状态和控制寄存器。 EHCI HC可以访问本规范中定义的附加端口状态和控制位(见第2.3.9节)。EHCI HC可通过将PORTSC寄存器中的 <i>PortOwner</i> 位设置为1,将端口控制临时释放给配套HC。

<sup>3</sup>如果一个实现包括一组以上的伴随和EHCI主机控制器,则它们被组织为具有混合的EHCI控制器的伴随主机控制器的组。

#### 4.2.2 通过PortOwner和Disconnect事件的端口路由控制

通过CF位操作端口路由是一个极端的过程，不打算在正常操作期间使用。端口所有权转移的正常模式是使用EHCI HC的PORTSC寄存器中的port Owner位（用于从EHCI切换到配套主机控制器）来确定各个端口的粒度。当端口注册设备断开连接时，单个端口所有权将返回给EHCI控制器。当检测到断开连接时，端口路由逻辑立即将端口所有权返回给EHCI控制器。配套主机控制器端口寄存器检测到设备断开连接并正常工作。

在正常操作条件下（假设所有HC驱动程序都已加载并可操作，并且EHCI CF位设置为1），典型的端口枚举序列如下所示：

- ① 初始条件是EHCI是端口所有者。设备已连接，导致端口检测到连接，设置端口连接更改位并发出端口更改中断（如果启用）。
- ② EHCI驱动程序用断言的新连接更改位标识端口，并向集线器驱动程序发送更改报告。集线器驱动程序发出GetPortStatus（）请求并标识连接更改。然后，它发出一个清除连接更改的请求，然后发出一个重置和启用端口的请求。
- ③ 当EHCI驱动程序收到重置和启用端口的请求时，它首先检查PORTSC寄存器中LineStatus位报告的值。如果它们指示连接的设备是全速设备（例如D+被断言），则EHCI驱动器将PortReset控制位设置为1（并将PortEnable位设置为0），从而开始重置过程。软件对重置的持续时间进行计时，然后通过向端口重置位写入零来终止重置信令。当软件读取端口重置位中的零时，重置过程实际上完成了。EHCI驱动程序检查PORTSC寄存器中的PortEnable位。如果设置为1，则连接的设备是高速设备，EHCI驱动程序（根集线器模拟器）会向集线器驱动程序发出更改报告，集线器驱动程序会继续枚举连接的设备。
- ④ 当EHCI驱动程序接收到端口重置和启用请求时，线路状态位可能指示低速设备。此外，当端口重置过程完成时，PortEnable字段可能指示连接了全速设备。在任何一种情况下，EHCI驱动程序都会将PORTSC寄存器中的PortOwner位设置为1，以将端口所有权释放给配套主机控制器。
- ⑤ 当EHCI驱动程序将PortOwner位设置为1时，端口路由逻辑使收发器的连接状态可用于配套主机控制器端口寄存器，并从EHCI HC端口删除连接状态。EHCI PORTSC寄存器通过断开连接更改位观察并报告断开连接事件。EHCI驱动程序检测连接状态更改（通过轮询或端口更改中断），然后向集线器驱动程序发送更改报告。当集线器驱动程序请求该端口状态时，EHCI驱动程序以重置完整更改设置为1、连接更改设置为0和连接状态设置为0作为响应。此信息直接来自EHCI端口寄存器。这将允许集线器驱动程序假定设备在重置期间已断开连接。它将确认更改位并等待下一个更改事件。虽然EHCI控制器不拥有该端口，但它只是保持在端口报告未连接设备的状态。

配套HC的设备连接评估电路激活并检测设备，配套驱动程序检测连接并枚举端口。

当端口路由到伴随HC时，它保持在伴随HC的控制下，直到设备与根端口断开连接（目前忽略EHCI的CF位从1b转换到0b的情况）。当发生断开连接时，伴随HC端口控制和EHCI端口所有权控制都会检测到断开连接事件。在该事件中，端口所有权会立即返回给EHCI控制器。伴随HC堆栈检测断开连接并进行确认，就像在普通的独立实现中一样。EHCI端口寄存器将检测到后续连接，并且该过程将重复。

### 4.2.3 端口路由状态机示例

图4-2举例说明了应如何管理端口所有权。以下各节介绍了每个状态的进入条件。

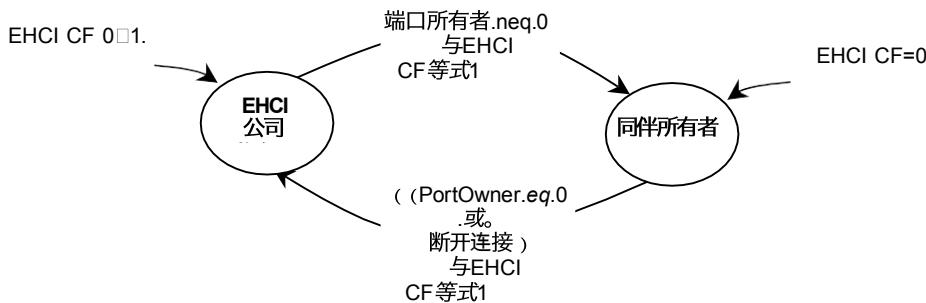


图4-2。端口所有者切换状态机

#### 4.2.3.1 EHCI HC所有者

只要发生以下事件之一，就会进入此状态：

- ① 当CONFIGFLAG寄存器中的EHCI HC的配置标志(*CF*)位从0转换为1时。这表明系统具有用于EHCI HC的主机控制器驱动程序，并且USB 2.0主机控制器中的所有端口必须默认路由到EHCI控制器。
- ② 当端口为配套HC所有并且设备和端口断开连接时。EHCI端口路由控制逻辑会收到断开连接的通知，并将端口路由返回给EHCI控制器。伴随HC的连接状态立即变为断开状态(具有连接更改、启用和启用更改的适当副作用)。配套HC驾驶员将通过将连接状态更改位设置为零来确认断开连接。这允许配套HC的驱动器通过断开过程与端口完全交互。
- ③ 当系统软件将零写入PORTSC寄存器中的*PortOwner*位时。这允许软件从配套主机控制器获得端口的所有权。当这种情况发生时，到伴随HC的路由有效地向伴随HC的端口状态和控制寄存器发出断开连接的信号。

#### 4.2.3.2 同伴HC所有者

只要发生以下事件之一，就会进入此状态：

- ① 当“端口所有者”字段从0转换为1时。
- ② 当HS模式HC的配置标志(*CF*)等于零时。

进入该状态时，路由逻辑允许配套HC端口寄存器检测设备连接。正常端口枚举继续进行。

### 4.2.4 港口电力

HCSPARAMS寄存器中的端口功率控制(*PPC*)位指示USB 2.0主机控制器是否具有端口功率控制功能(参见第2.2.3节)。当该位为零时，则主机控制器不支持端口功率开关的软件控制。在此配置中，端口电源始终可用，并且配套主机控制器必须实现与端口电源始终打开一致的功能。

当*PPC*位为1时，则主机控制器实现包括端口电源开关。每个可用的交换机都有一个输出启用，在本讨论中称为*PortPowerOutputEnable*

(第页)。PPE是基于组合比特PPC比特、EHC配置(CF)比特和单个端口功率(PP)比特的状态来控制的。表4-3说明了总结行为模型。

表4-3。端口电源启用控制规则

CF	CHC2 (聚丙烯)	EHC3 (PP)	物主	第1页	描述
0	0	十、	中国	0	当尚未配置EHC控制器时，该端口由配套主机控制器所有。当配套HC的端口电源选择关闭时，端口电源关闭。
0	1.	十、	中国	1.	与前面的条目类似。当配套HC的端口电源选择打开时，端口电源打开。
1.	0	0	中国	0	端口所有者已关闭端口电源，端口电源已关闭。
1.	0	0	紧急事故控制	0	端口所有者已关闭端口电源，端口电源已关闭。
1.	0	1.	紧急事故控制	1.	端口所有者已打开端口电源，因此端口电源已打开。
1.	0	1.	中国	1.	如果任一HC打开了端口电源，则端口的电源打开。
1.	1.	0	紧急事故控制	1.	如果任一HC打开了端口电源，则端口的电源打开。
1.	1.	0	中国	1.	端口所有者已打开端口电源，因此端口电源已打开。
1.	1.	1.	中国	1.	端口所有者已打开端口电源，因此端口电源已打开。
1.	1.	1.	紧急事故控制	1.	端口所有者已打开端口电源，因此端口电源已打开。

1PPE (端口电源启用)。此位实际上打开了端口电源开关 (如果存在)。

2CHC (配套主机控制器)。

3EHC (EHC主机控制器)。

#### 4.2.5 过电流端口报告

主机控制器是USB上的电源供应器。端口被认为是高功率还是低功率是平台实现的问题。每个EHC PORTSC寄存器都有一个过电流状态和过电流变化位。这些位的功能在USB规范修订版2.0中有规定。

过电流检测和限制逻辑通常位于主机控制器逻辑之外。该逻辑可以与一个或多个端口相关联。当该逻辑检测到过电流情况时，它对配套端口和EHC端口都可用。过电流状态对配套主机控制器端口的影响超出了本文档的范围。过电流情况会影响EHC端口上PORTSC寄存器中的以下位：

- ① 过电流有效位被设置为1。当过电流状态消失时，过电流激活位将从1转换为0。
- ② 过电流变化位被设置为1。在过电流激活位的每次转换时，主机控制器将把过电流改变位设置为1。软件通过向该位写入1，将过电流变化位设置为零。
- ③ 端口启用/禁用位设置为零。当此更改位设置为1时，则USBSTS寄存器中的端口更改检测位设置为一。

- ① 端口功率 (PP) 比特可以可选地被设置为零。USB中没有要求电源供应商在过电流条件下关闭电源。限制电流并保持通电状态就足够了。

当过电流变化位从0转换为1时，主机控制器还将USBSTS寄存器中的端口变化检测位设置为1。此外，如果USBINTR寄存器中的端口更改中断启用位为1，则主机控制器将向系统发出中断。有关主机控制器停止（从设备组件的角度来看已暂停）时过电流检测的汇总行为，请参阅表4-4。

### 4.3 挂起/继续

EHCI主机控制器提供了与为USB 2.0集线器中的各个端口定义的挂起和恢复模型等效的挂起模型。提供控制机制以允许系统软件挂起和恢复各个端口。这些机制允许通过软件启动完全恢复各个端口。提供其他控制机制以参数化主机控制器对外部恢复事件的响应（或灵敏度）。在本讨论中，主机启动或软件启动的简称为简历事件/操作。总线启动的恢复事件称为唤醒事件。唤醒事件的类别包括：

- ① 启用远程唤醒的设备断言恢复信令。与USB 2.0集线器类似，EHCI控制器必须始终响应明确的设备恢复信号并唤醒系统（如有必要）。
- ② 端口连接和断开以及过电流事件。通过使用PORTSC寄存器中的每个端口控制位，可以打开或关闭对这些事件的敏感度。

选择性挂起是每个PORTSC寄存器都支持的功能。它用于将特定端口置于挂起模式。此功能被用作实现在特定操作系统中实现的适当电源管理策略的功能组件。

当系统软件打算挂起整个总线时，它应该有选择地挂起所有启用的端口，然后通过将USBCMD寄存器中的运行/停止位设置为零来关闭主机控制器。然后，可以通过PCI电源管理接口将EHCI模块置于较低的设备状态（见附录a）。

当唤醒事件发生时，系统将恢复操作，系统软件最终将运行/停止位设置为1，并恢复挂起的端口。在确认主机控制器的时钟稳定之前，软件不得将运行/停止位设置为1。这通常在系统实现中得到证实，因为在CPU重新启动之前，系统中的所有时钟都是稳定的。因此，根据定义，如果软件正在运行，系统中的时钟是稳定的，并且USBCMD寄存器中的运行/停止位可以设置为1。PCI电源管理规范中也定义了最小系统软件延迟。有关更多信息，请参阅本规范。

#### 4.3.1 端口挂起/恢复

系统软件通过将一个端口写入适当的PORTSC挂起位，将各个端口置于挂起模式。只有当端口处于启用状态（端口启用位为1）且EHCI为端口所有者（端口所有者位为0）时，软件才能设置挂起位。

主机控制器可以立即评估挂起比特，或者等待直到出现微帧或帧边界。如果立即评估，则在当前事务（如果正在执行）完成之前，端口不会挂起。因此，在主机控制器评估挂起位之前，端口上可能存在多个活动的微帧。主机控制器必须至少在每个帧边界评估挂起位。

系统软件可以通过将一写入强制端口恢复位，在选择性挂起的端口上启动恢复。除非端口报告其处于挂起状态，否则软件不应尝试恢复端口（请参阅第2.3.9节）。如果系统软件在端口未处于挂起的状态时将“强制端口恢复”位设置为1，则产生的行为未定义。为了确保USB设备正常运行，软件必须在端口指示其挂起后等待至少10毫秒（挂起位为

一),然后通过强制端口恢复位启动端口恢复。当强制端口恢复位为1时,主机控制器向端口发送恢复信号。系统软件对恢复的持续时间(标称为20毫秒)进行计时,然后将强制端口恢复设置为0。当主机控制器接收到将强制端口恢复转换为零的写入时,它将完成USB规范中定义的恢复序列,并将强制端口继续和挂起位都设置为零。软件启动的端口恢复不会影响USBSTS寄存器中的端口更改检测位,如果USBINTR寄存器中的“端口更改中断启用”位为1,则不会导致中断。

外部USB事件也可以启动恢复。唤醒事件定义如上。当挂起的端口上发生唤醒事件时,端口会检测到恢复信号,并在100内向下游反映恢复□秒。端口的强制端口恢复位设置为1,USBSTS寄存器中的端口更改检测位设置为一。如果USBINTR寄存器中的端口更改中断启用位为1,则主机控制器将发出硬件中断。

系统软件观察端口上的恢复事件,延迟端口恢复时间(通常为20毫秒),然后通过将零写入端口中的强制端口恢复位来终止恢复序列。主机控制器接收对强制端口恢复的零写入,终止恢复序列,并将强制端口恢复和挂起端口位设置为零。软件可以通过对PORTSC寄存器进行采样并观察“挂起”和“强制端口恢复”位为零来确定端口已启用(未挂起)。

在通过将零写入端口的强制端口恢复位来终止恢复之前,软件必须确保主机控制器正在运行(即USBSTS寄存器中的HCHalted位为零)。如果当“强制端口恢复”设置为零时,HCHalted是一个,则SOF将不会出现在启用的端口上,并且设备将在最长10毫秒内返回到挂起模式。

表4-4总结了唤醒事件。每当检测到恢复事件时,USBSTS寄存器中的端口更改检测位都设置为1。如果端口更改中断启用位是USBINTR寄存器中的一位,则主机控制器也会在恢复事件上生成中断。软件通过清除USBSTS寄存器中的端口更改检测状态位来确认恢复事件中断。

**表4-4。唤醒事件期间的行为**

端口状态和信令类型	信号端口响应	设备状态	
		D0	不是D0
端口已禁用, 收到恢复K状态	无影响	N/A	N/A
端口已挂起,收到恢复K状态	恢复反映在信号端口的下游。PORTSC寄存器中的强制端口恢复状态位设置为1。USBSTS寄存器中的端口更改检测位设置为1。	[1],[2]	[2]
端口已启用、禁用或挂起,并且端口的WKDSCNNT_E位为1。检测到断开连接。	根据初始端口状态,PORTSC Connect(端口连接)和Enable(启用)状态位设置为零,Connect Change(连接更改)状态位设为一。USBSTS寄存器中的端口更改检测位设置为1。	[1],[2]	[2]
端口已启用、禁用或挂起,并且端口的WKDSCNNT_E位为零。检测到断开连接。	根据初始端口状态,PORTSC Connect(端口连接)和Enable(启用)状态位设置为零,Connect Change(连接更改)状态位设为一。USBSTS寄存器中的端口更改检测位设置为1。	[1],[3]	[3]
端口未连接,端口的WKCNT_E位为1。检测到连接。	PORTSC连接状态和连接状态更改位被设置为1。USBSTS寄存器中的端口更改检测位设置为1。	[1],[2]	[2]
端口未连接,并且端口的WKCNT_E位为零。检测到连接。	PORTSC连接状态和连接状态更改位被设置为1。USBSTS寄存器中的端口更改检测位设置为1。	[1],[3]	[3]

表4-4。唤醒事件期间的行为（续）

端口状态和信令类型	信号端口响应	设备状态	
		D0	不是D0
端口已连接，端口的WKOC_E位为1。出现过电流情况。	PORTSC过电流激活、过电流改变位被设置为1。如果端口启用/禁用位为1，则将其设置为零。USBSTS寄存器中的端口更改检测位设置为1	[1], [2]	[2]
端口已连接，并且端口的WKOC_E位为零。出现过电流情况。	PORTSC过电流激活、过电流改变位被设置为1。如果端口启用/禁用位为1，则将其设置为零。USBSTS寄存器中的端口更改检测位设置为1。	[1], [3]	[3]

[1] 如果USBINTR寄存器中的端口更改中断启用位为1，则发出硬件中断。

[2] 如果启用，则断言PME#（注意：PME状态必须始终设置为1）。

[3] 未断言PME#。

#### 4.4 计划遍历规则

主机控制器使用简单的共享内存调度来执行设备的事务。时间表由几个数据结构组成，这些数据结构被组织成两个不同的列表。数据结构旨在提供USB所需的最大灵活性，最大限度地减少内存流量和硬件/软件复杂性。

系统软件为主机控制器维护两个时间表：周期性时间表和异步时间表。周期调度的根是PERIODICLISTBASE寄存器（见第2.3.6节）。PERIODICLISTBASE寄存器是周期帧列表的物理内存地址。周期性帧列表是物理内存指针的数组。框架列表中引用的对象必须是第3节中定义的有效明细表数据结构。在每个微帧中，如果启用了周期性调度（见第4.6节），则主机控制器必须在从异步调度执行之前从周期性调度执行。只有在遇到周期性计划的末尾后，它才会从异步计划执行。主机控制器通过从PERIODICLISTBASE和FRINDEX寄存器构建阵列偏移参考来遍历周期性调度（见图4-3）。它获取元素并开始遍历链接的时间表数据结构图。

周期性调度的结束由其T比特设置为1的调度数据结构的下一个链接指针来标识。当主机控制器在周期列表的水平遍历过程中遇到设置为1的T位时，它将其解释为周期列表结束标记。这导致主机控制器停止按周期性调度工作，并立即转换到遍历异步调度。一旦进行了这种转换，主机控制器就从异步调度开始执行，直到微帧结束。

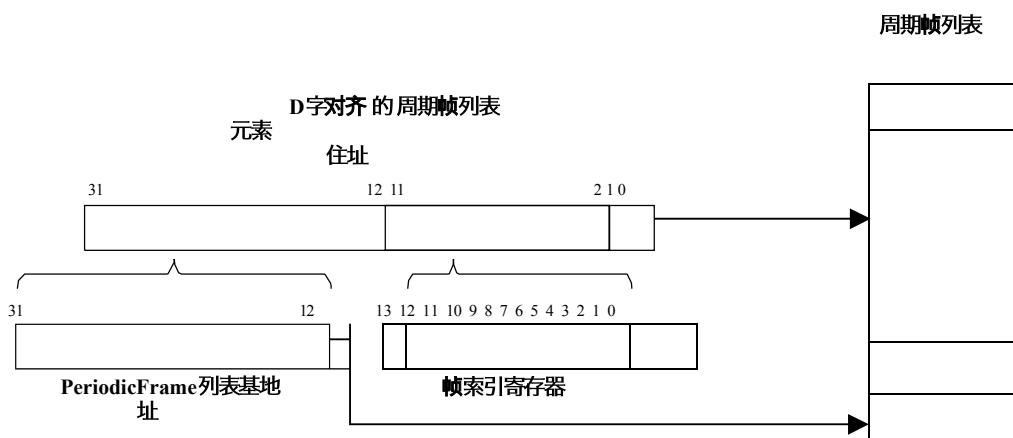
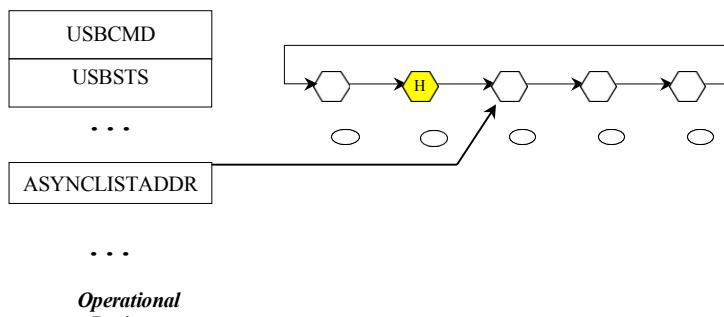


图4-3。指向帧列表数组的指针派生

当主机控制器确定是时候从异步列表执行时，它使用操作寄存器`ASYNCLISTADDR`来访问异步调度，见图4-4。



**图4-4。异步进度表通用格式**

`ASYNCLISTADDR`寄存器包含指向下一个队列头的物理内存指针。当主机控制器转换到执行异步调度时，它从读取`ASYNCLISTADDR`寄存器引用的队列头开始。对于异步调度中的队列头，软件必须将队列头水平指针T位设置为零。有关完整的操作细节，请参见第4.8节。

#### 4.4.1 示例-保持微框架完整性

USB主机控制器的要求之一是保持帧完整性。这意味着HC必须保留微框架边界。例如：SOF数据包必须按时生成（在指定的允许抖动范围内），并且必须强制执行高速EOF1,2阈值。微帧定时点EOF1和EOF2的结束在USB规范修订版2.0中有明确定义。

这一责任的一个含义是HC必须确保其不会启动在微框架结束之前无法完成的交易。更确切地说，主机控制器不应该启动任何事务，因为这些事务不能在EOF1点之前全部完成。为了强制执行此规则，主机控制器必须在每个事务开始之前检查它，以确保它将在微帧结束之前完成。

那么，这次检查到底需要涉及什么呢？从根本上说，必须考虑事务数据有效负载、比特填充和事务开销。下一笔交易需要花费的时间可能非常准确。以输出为例。主机控制器必须从内存中获取所有OUT数据，才能将其发送到USB总线上。主机控制器实现可以预取所有OUT数据，并预计计算令牌和数据包中的实际位数。此外，系统知道目标端点的深度，因此可以密切估计握手的周转时间。此外，主机控制器知道握手包的大小。比特填充的预计效果和对其他开销数的求和可以允许主机控制器准确地知道在EOF1完成OUT事务之前是否有足够的总线时间。要实现这种特殊的方法需要大量的时间和硬件复杂性。

另一种选择是合理猜测是否可以启动下一个事务。下面描述一个示例近似算法。该示例算法依赖于EHCI策略，即在微帧中首先调度周期性事务。合理的假设是，软件永远不会将微框架过度提交给大于规范允许的80%的周期性事务。在可用的剩余20%带宽中，主机控制器具有一定的能力（在本例中）来决定是否执行事务。这种算法的结果是，有时，在某些情况下，本可以执行的事务将不会被执行。然而，在任何情况下，除非框架中有足够的时间来完成事务，否则永远不会启动事务。

#### 4.4.1.1 事务拟合——一种最佳拟合近似算法

计算一条曲线，该曲线表示每个数据包大小的最新开始时间，软件将在该时间调度周期性事务的开始。该曲线是80%带宽曲线。计算另一条曲线，该曲线是每个数据包大小的绝对的、最近允许的开始时间。该曲线表示微帧中每个数据包大小的事务可以开始和完成的绝对最新时间。这两条曲线图如图4-5所示。绘图Y轴表示帧中剩余的字节数次。

80%和最后开始图之间的空间是带宽回收区域。在该算法中，如果谨慎的话，主机控制器可以在此期间跳过事务。

“最佳拟合近似”方法绘制80%和“最后开始”曲线之间的函数( $f(x)$ )。函数 $f(x)$ 为每个事务的最大数据包大小添加一个常数，并将结果与帧中剩余的字节数进行比较。该常数表示比特填充和协议开销的影响的近似值。主机控制器启动其结果位于函数曲线之上的事务。主机控制器不会启动其结果落在函数曲线以下的事务。

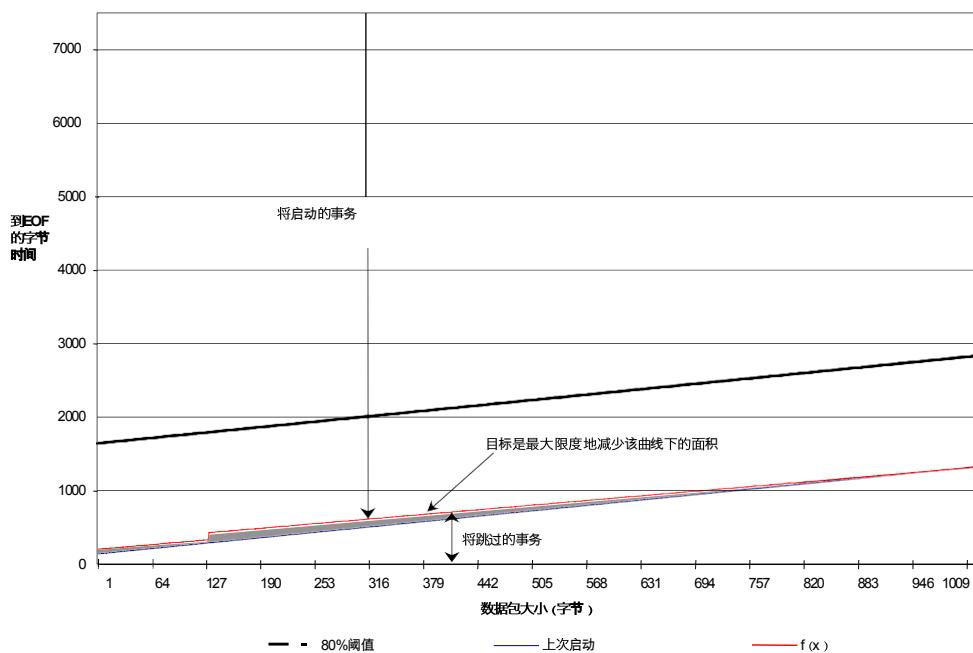


图4-5。最佳拟合近似

在本例中，计算LastStart行是为了假设每个事务的绝对最坏情况总线开销。使用的特定事务是一个开始拆分、长度为零的OUT事务，带有握手。表4-5中列出了组成部分的摘要。组件时间源自USB规范修订版2.0中定义的协议时序。

表4-5。更糟糕情况下的事务计时组件示例

组成部分	位时间	字节时间	解释
拆分令牌	76	9.5	USB核心规范中定义的拆分令牌。 包括同步、令牌、eop等。
主机2主机IPG	88	11	连续之间所需的位次数 主机数据包
代币	67	8.375	USB核心规范中定义的令牌。包括 同步、令牌、eop等。
主机2主机IPG	88	11	同上
数据包 (0个数据字节 )	66.7	8.34	零长度数据包。包括sync、PID、crc16 , eop等。
周转时间	721	90.125	数据包启动器 (主机 )看到的开始时间 对发送的分组的响应。
握手数据包	48	6.	USB核心中定义的握手数据包 规格包括同步、PID、eop等。
	144		全部的

函数 ( $f(x)$ ) 的确切细节取决于特定的实现。然而，很明显，目标是最大限度地减少近似函数和 *Last Start* 曲线之间的曲线下面积，而不低于 *LastStart* 线，同时保持硬件实现的检查尽可能简单。图4-5中的  $f(x)$  是使用以下对每个事务大小数据点的伪代码测试构建的。该算法假设主机控制器保持对帧中剩余比特的跟踪。

```

算法 检查事务willFit (最大数据包大小 ,HC_BytesLeftInFrame )开始
    本地      温度=  最大数据包   大小
    +192本地值=  真

    如果最大数据包大小>=128 ,则温度+=128
    结束如果

    如果温度>HC_BytesLeftInFrame ,则
        Rvalue=FALSE
    结束如果
    返回右值
终止

```

该算法采用两个输入，即事务的当前最大数据包大小和当前微帧中剩余字节数的硬件计数器。它无条件地将192的简单常数添加到最大分组大小，以考虑事务开销和比特填充的一阶效应。如果事务大小大于或等于128字节，则将128的附加常数添加到运行和，以说明大于128的有效载荷的附加最坏情况比特填充。在128处插入了一个拐点，因为  $f(x)$  图越来越接近 *LastStart* 线。

## 4.5 周期性明细表框架边界与总线框架边界

USB规范修订版2.0要求高速总线和USB 2.0集线器以下的全和低速总线的帧边界 (SOF帧编号变化) 严格对齐。对这一要求的超级要求是，USB 2.0集线器通过微帧管道管理完整和低速事务 (见图4-6所示的开始 (SS) 和完成 (CS) 拆分)。将帧边界模型简单、直接地投影到主机控制器接口调度架构中，会在帧边界和为全速率和低速事务转换器周期性管道提供服务所需的调度机制之间产生紧张关系 (硬件和软件都很复杂)。

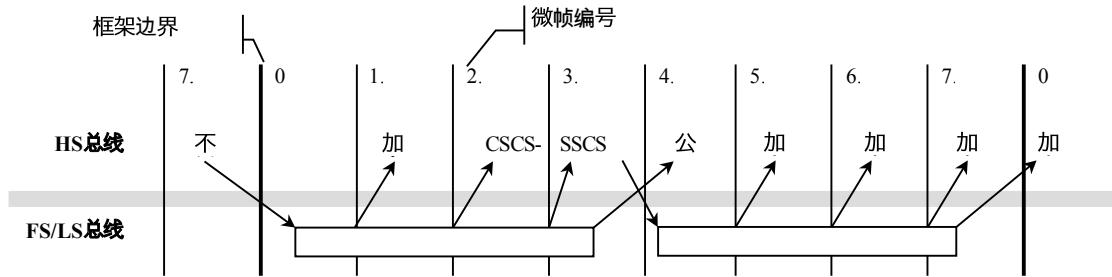


图4-6。HS总线与FS/LS总线的帧边界关系

如图4-6所示，简单的投影引入了帧边界包裹条件，用于在帧的开始和结束进行调度。为了降低硬件和软件的复杂性，主机控制器需要为其帧边界视图实现一个微帧相移。相移消除了帧的开始和帧包裹调度的边界条件。

该相移的实现要求主机控制器使用一个寄存器值来访问周期性帧列表，并且使用另一个值来访问SOF令牌中包括的帧号值。这两个值是分开的，但紧密耦合。周期性帧列表可通过帧列表索引寄存器(FRINDEX)访问，该寄存器记录在第2.3.4节中，最初如第4.4节所示。比特FRINDEX[2:0]表示微帧号。SOF值与FRINDEX[1:3]的值相耦合。FRINDEX[1:3]和SOF值均基于FRINDEX[2:0]递增。要求SOF值从FRINDEX值延迟一个微帧。一个微帧延迟产生主控制器周期调度和总线帧边界关系，如图4-7所示。这种调整允许软件使用周期性调度接口的自然对齐，为完整和低速周期性端点简单地调度周期性启动和完成拆分事务。选择这种相移的原因超出了本说明书的范围。

图4-7说明了周期性调度数据结构如何与调度帧边界和总线帧边界相关。为了帮助演示，定义了两个术语。主机控制器对1毫秒边界的视图称为H帧。高速总线对1毫秒边界的视图被称为B帧。

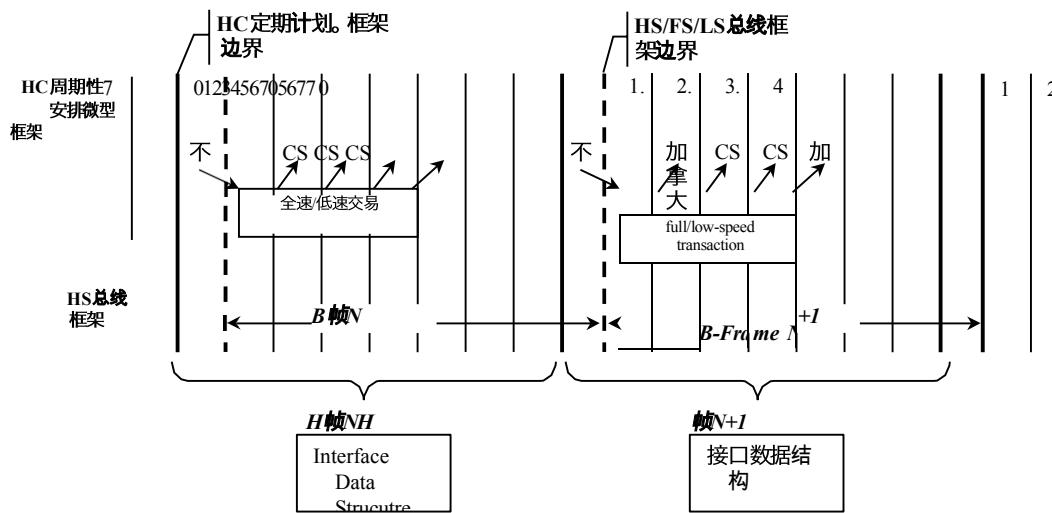


图4-7。周期计划框架边界与总线框架边界的关系

主机控制器的H帧边界对应于FRINDEX[13:3]的增量。H帧的微帧编号由FRINDEX[2:0]跟踪。通过改变SOF令牌的帧号，可以在高速总线上看到B帧边界。高速总线上的微帧编号仅来源于SOF令牌的帧编号（即，高速总线将看到八个具有相同SOF的SOF

帧编号值)。如图4-7所示, H帧和B帧具有固定关系(即,B帧滞后H帧一个微帧时间)。

主机控制器的周期性调度自然地与H帧对齐。软件为相对于H帧的全速和低速周期性端点调度事务。结果是,这些事务在USB 2.0集线器周期性管道的正确时间在高速总线上执行。

如第2.3.4节所述,SOF值可以实现为影子寄存器(在本例中称为SOFV),其滞后于FRINDEX寄存器位[13:3]一个微帧计数。表4-6说明了FRINDEX值和SOFV值之间所需的关系。这种滞后行为可以通过基于FRINDEX[2:0]的从7到0的增量的进位来增加FRINDEX[1:3]并且基于FRINDEX[2:0]从0到1的转变来增加SOFV来实现。

允许软件写入FRINDEX。第2.3.4节提供了在FRINDEX中写入新值时软件应遵守的要求。

**表4-6。FRINDEX和SOFV (SOF值寄存器)的操作**

现在的			下一个		
弗里恩德斯[F]	索夫	FRINDEX公司 [□F ]	弗里恩德斯[F]	索夫	FRINDEX公司 [□F ]
N	N	111b个	N+1	N	000亿
N+1	N	000亿	N+1	N+1	001b号
N+1	N+1	001b号	N+1	N+1	010b美元
N+1	N+1	010b美元	N+1	N+1	2011年
N+1	N+1	2011年	N+1	N+1	100亿
N+1	N+1	100亿	N+1	N+1	101b个
N+1	N+1	101b个	N+1	N+1	110磅
N+1	N+1	110磅	N+1	N+1	111b个

式中[F]=[13:3]; [□F]=[2:0]

## 4.6 定期计划

通过USBCMD寄存器中的周期调度启用位启用或禁用周期调度遍历。如果周期调度使能位被设置为零,则主机控制器简单地不试图通过周期CLISTBASE寄存器访问周期帧列表。同样,当周期性调度启用位为1时,则主机控制器确实使用周期性CLISTBASE寄存器来遍历周期性调度。主机控制器不会立即对“定期计划启用”的修改做出反应。为了消除与拆分事务的冲突,主机控制器仅在FRINDEX[2:0]为零时评估周期性调度启用位。如果计划包含跨越000b微帧的活动拆分事务工作项,则系统软件不得禁用定期计划。

在周期性计划启用位写入零之前,必须从计划中删除这些工作项。

USBSTS寄存器中的周期性调度状态位指示周期性调度的状态。系统软件通过向USBCMD寄存器中的周期性计划启用位写入一(或零)来启用(或禁用)周期性计划。然后,软件可以轮询周期性调度状态位,以确定周期性调度何时进行了期望的转换。除非周期性计划启用位的值等于周期性计划状态位的值,否则软件不得修改周期性计划使能位。

周期性调度用于管理所有同步和中断传输流。周期性时间表的基础是周期性帧列表。软件将调度数据结构链接到周期性帧列表,以生成调度数据结构的图。该图表示USB上适当的事务序列。图4-8显示了周期为1的等时传输(使用iTID和sITIDs)直接链接到周期帧列表。中断传输(由队列头管理)和

具有不同于一个周期的周期的等时流在周期—iTDS之后被链接。中断队列头被链接到按轮询速率排序的帧列表中。较长的轮询速率首先链接（例如，最接近周期性帧列表），然后是较短的轮询速率，在最末端具有轮询速率为1的队列头。

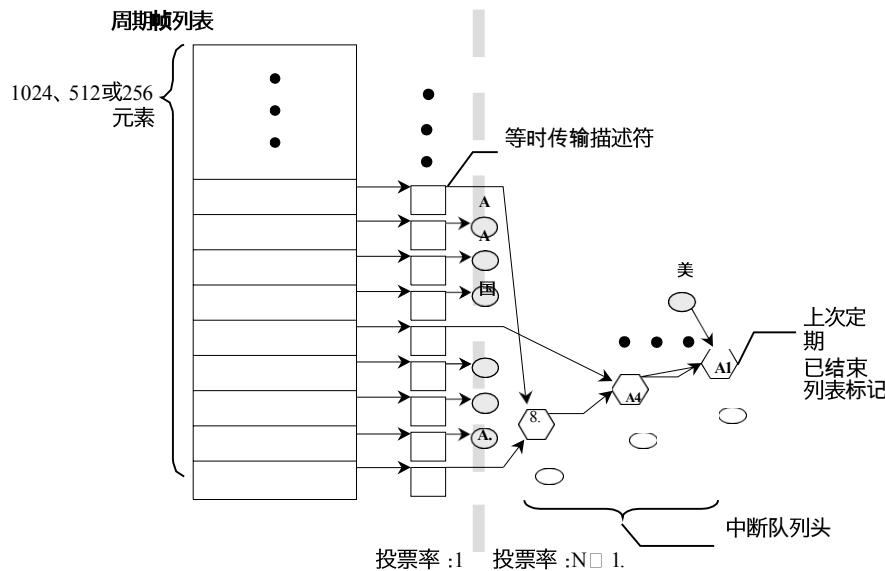


图4-8。周期计划示例

## 4.7 使用iTDS管理等时传输

iTD的结构如3.3所示。iTD有四个不同的部分：

- ① 第一个字段是“下一个链接指针”。此字段仅用于计划链接目的；
- ② 事务描述数组。这个区域是一个八元素数组。每个元素表示单个高速等时端点的一个微帧事务的控制和状态信息。
- ③ 缓冲区页指针数组是指向数据缓冲区的物理存储器指针的7元素数组。这些是指向物理内存的4K对齐指针。
- ④ 端点功能。该区域利用缓冲页指针阵列的未使用的低阶12位。此区域中的字段用于为此iTD执行的所有事务，包括端点寻址、传输方向、最大数据包大小和高带宽乘法器。

### 4.7.1 iTD的主机控制器操作模型

主机控制器使用FRINDEX寄存器位[12:3]来索引到周期性帧列表中。这意味着主机控制器在递增到下一个周期性帧列表元素之前连续八次访问每个帧列表元素。每个iTD包含八个事务描述，它们直接映射到FRINDEX寄存器位[2:0]。每个iTD可以跨越8个微帧的交易。

当主机控制器获取iTD时，它使用FRINDEX寄存器位[2:0]索引到事务描述数组中。如果索引事务描述的状态字段中的活动位设置为零，则主机控制器将忽略iTD，并跟随下一个指针指向下一个调度数据结构。

当索引的活动位为1时，主机控制器继续解析iTD。它存储索引的事务描述和一般端点信息（设备地址、端点编号、最大值）

数据包大小等)。它还使用页面选择(*PG*)字段对缓冲区指针数组进行索引,存储所选的缓冲区指针和下一个顺序缓冲区指针。例如,如果*PG*字段为0,则主机控制器将存储页面0和页面1。

主机控制器通过连接当前缓冲区指针(如使用当前事务描述的*PG*字段所选择的)和事务描述的事务偏移字段来构造物理数据缓冲区地址。主机控制器使用端点寻址信息和I/O位来执行到适当端点的事务。当事务完成时,主机控制器清除活动位,并将任何额外的状态信息写回当前所选事务描述中的“状态”字段。

与iTD相关联的数据缓冲区必须是虚拟连续存储器。提供七个页面指针以支持八个高带宽事务,而不管起始分组到第一页面的偏移对齐。起始缓冲区指针(物理内存地址)是通过将活动事务描述的*PG*(示例值:00B)字段选择的页面指针(示例:页面0指针)与事务偏移量字段连接而构建的。当事务移动数据时,主机控制器必须检测当前缓冲区指针的增量何时会越过页面边界。每个总线事务的大小由最大数据包大小字段中的值确定。

iTD通过*Mult*(乘数)字段支持高带宽管道。当*Mult*字段为1、2或3时,主机控制器为当前微帧中的端点执行指定数量的最大数据包大小的总线事务。换句话说,*Mult*字段表示当前微帧中端点的事务计数。如果*Mult*字段为零,则主机控制器的操作未定义。传输描述用于服务由*Mult*字段指示的所有事务。

对于OUT传输,Transaction *X Length*字段的值表示在微帧期间要发送的总字节数。*Mult*字段必须由软件设置为与*Transaction X Length*和*Maximum Packet Size*一致。主机控制器将以最大数据包大小的部分发送字节。在每个事务之后,主机控制器将事务*X*长度的本地副本递减最大数据包大小。主机控制器发送的字节数始终是“最大数据包大小”或“事务*X*长度”,以较小者为准。主机控制器推进传输描述中的传输状态,更新iTD中的适当记录,并移动到下一个调度数据结构。支持的最大事务大小为3 x 1024字节。

对于IN传输,主机控制器发出*Mult*事务。假设软件已正确初始化iTD,以容纳所有可能的数据。在每个IN事务期间,主机控制器必须使用“最大数据包大小”来检测数据包牙牙学语错误。主机控制器保持在事务*X*长度字段中接收的字节的总和。在微帧的端点的所有事务都完成后,事务*X*长度包含接收到的总字节数。如果事务*X*长度的最终值小于最大数据包大小的值,则从关联端点接收到的数据少于允许的数据。这种短分组条件不会将USBSTS寄存器中的*USBINT*位设置为1。主机控制器将不会检测到这种情况。如果设备发送的字节数超过事务*X*长度或最大数据包大小字节数(以较小者为准),则主机控制器会将巴氏检测位设置为1,并将活动位设置为0。请注意,在这种错误情况下,主机控制器不需要更新iTD字段*Transaction X Length*。

如果*Mult*字段大于1,则主机控制器将自动执行*Mult*的值交易。如果出现以下情况,主机控制器将不会执行所有*Mult*事务:

- ① 端点是一个OUT,在所有*Mult*事务执行(数据用完)之前,Transaction *X Length*变为零,或者
- ② 端点是IN,并且端点传递短数据包,或者在  
已执行多项事务。

微帧的结束可能发生在所有交易机会都被执行之前。当这种情况发生时,传输描述的传输状态被提前以反映所取得的进展,结果被写回iTD,并且主机控制器继续处理下一个微帧。提到

附录D是所有高带宽事务情况下主机控制器所需行为的表格摘要。

#### 4.7.2 iTD的软件操作模型

对等时端点的客户端缓冲区请求可以跨越1到N个微帧。当N大于1时，系统软件可能必须使用多个iTД来读取或写入缓冲区中的数据（如果N大于8，则必须使用一个以上的iTД）。

图4-9说明了系统软件如何将客户端缓冲区映射到周期性时间表（即周期性帧列表和一组iTД）的简单模型。右边是客户对其请求的描述。该描述包括一个缓冲区基地址加上额外的注释，以识别缓冲区的哪些部分应该与每个总线事务一起使用。中间是系统软件用于服务客户端请求的iTД数据结构。每个iTД可以初始化为最多24个事务提供服务，分为八组，每组最多三个事务。每个组映射到一个微框架的交易价值。EHCI控制器不在微帧内提供每个事务的结果。它将每个微帧的事务处理为单个逻辑传输。左侧是主机控制器的帧列表。

系统软件建立从帧列表中适当位置到每个适当iTД的参考。

如果缓冲区很大，则系统软件可以使用一小组iTД为整个缓冲区提供服务。系统软件可以以特定数据流所需的任何模式激活事务描述记录（包含在每个iTД中）。

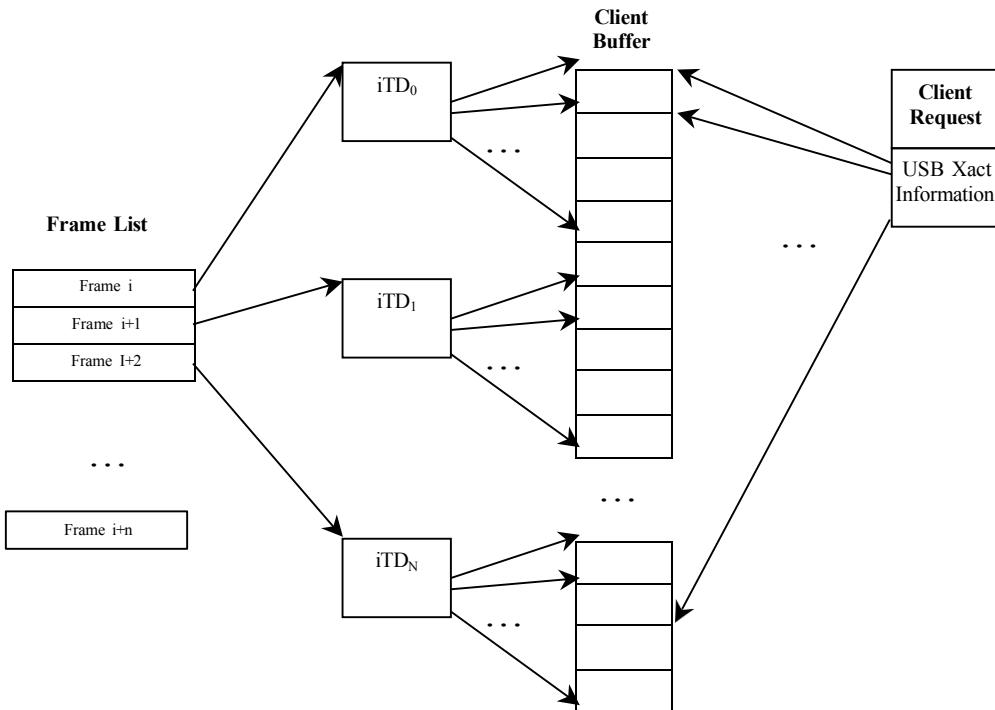


图4-9。iTД与客户端请求缓冲区的关联示例

如上所述，客户端请求包括指向缓冲区底部的指针和到缓冲区的偏移量，以注释在该端点上发生的每个总线事务上将使用哪些缓冲区部分。系统软件必须初始化iTД中的每个事务描述，以确保其使用客户端缓冲区的正确部分。例如，对于每个事务描述，PG字段被设置为索引正确的物理缓冲区页面指针，并且事务偏移字段被设置成相对于正确的缓冲区指针页面（例如，PG字段所引用的相同页面）。当主机控制器执行事务时，它基于FRINDEX[2:0]选择事务描述记录。然后它使用当前页面缓冲区指针（由

*PG*字段)并连接到事务偏移量字段。结果是事务的起始缓冲区地址。当主机控制器移动事务的数据时,它必须注意页面换行条件,并正确地前进到下一个可用的页面缓冲区指针。

系统软件不得在事务描述中使用第6页缓冲区指针,其中传输长度将包裹页面边界。这样做会产生未定义的行为。不需要主机控制器硬件将页面选择器“别名”为页面零。

USB 2.0等时端点可以指定一个大于一的周期。软件可以通过将iTID链接到适当的帧(相对于帧列表)并通过将适当的事务描述元素的有效位设置为1来实现适当的调度。

#### 4.7.2.1 周期性调度阈值

HCCPRAMS能力寄存器中的“等时调度阈值”字段是系统软件的一个指示器,指示主机控制器如何预取和有效缓存调度数据结构。系统软件在将等时工作项添加到定期计划中时使用它。该字段的值向系统软件指示它可以更新等时数据(相对于周期列表中主机控制器执行的当前位置)并且仍然让主机控制器处理它们的最小距离。

iTD和siTD数据结构中的每一个都描述了价值8微帧的交易。主机控制器被允许高速缓存这些数据结构中的一个(或多个),以便减少存储器流量。有三个基本的缓存模型来解释等时数据结构跨越8个微帧的事实。三种缓存模型分别是:无缓存、微帧缓存和帧缓存。

当软件将新的等时事务添加到调度时,它总是执行对FRINDEX寄存器的读取,以确定主机控制器当前正在执行的当前帧和微帧。当然,没有关于主控制器在微帧中的位置的信息,因此必须假设一个微帧的恒定不确定性因子。将主机控制器在哪里执行的知识与高速缓存模型的知识相结合,允许定义简单的算法,以确定软件能够可靠地与正在执行的主机控制器工作的程度。

在*Isochronous Scheduling Threshold*(等时调度阈值)字段中,没有用零值表示缓存。主机控制器可以在周期性调度遍历期间(每微帧)预取数据结构,但总是在微帧结束时转储任何累积的调度状态。在相对于每个微帧的开始的适当时间,主机控制器总是从帧列表开始调度遍历。软件可以使用FRINDEX寄存器的值(加上常数1的不确定性因子)来确定执行主机控制器的大致位置。当没有选择缓存时,软件可以在主机控制器的当前执行位置之前添加接近2微帧的等时事务。

帧缓存用等时调度阈值字段的位[7]中的非零值表示。在帧缓存模型中,系统软件假设主机控制器为整个帧(8个微帧)缓存一个(或多个)等时数据结构。软件使用FRINDEX寄存器的值(加上常数1的不确定性)来确定当前微帧/帧(假设将常数1与微帧编号相加时采用模8运算)。对于任何当前帧N,如果当前微帧是0到6,则软件可以安全地将等时事务添加到帧N+1。如果当前微帧是7,则软件可以将等时事务添加到帧N+2。

微帧缓存用等时调度阈值字段的最低有效3位中的非零值表示。系统软件假设主机控制器缓存一个或多个周期性数据结构,用于等时调度阈值字段中指示的微帧数量。例如,如果计数值为2,则主机控制器在芯片上保持2微帧的状态窗口(当前微帧加上一个)。在每个微帧边界上,主机控制器释放当前微帧状态并开始累积下一个微帧状态。

## 4.8 异步时间表

异步调度遍历通过USBCMD寄存器中的异步调度启用位启用或禁用。如果异步调度启用位设置为零，则主机控制器简单地不尝试通过`ASYNCLISTADDR`寄存器访问异步调度。同样，当异步调度启用位为1时，主机控制器确实使用`ASYNCLISTADDR`寄存器遍历异步调度。对异步调度启用位的修改不一定是立即的。相反，只有下次主机控制器需要使用`ASYNCLISTADDR`寄存器的值来获得下一个队列头时，才会考虑比特的新值。

USBSTS寄存器中的异步调度状态位指示异步调度的状态。系统软件通过向USBCMD寄存器中的异步调度启用位写入一（或零）来启用（或禁用）异步调度。然后，软件可以轮询异步调度状态位，以确定异步调度何时进行了所需的转换。除非异步计划启用位的值等于异步计划状态位的值，否则软件不得修改异步计划启用。

异步计划用于管理所有控制和批量传输。使用队列头数据结构管理控制和批量传输。异步调度基于`ASYNCLISTADDR`寄存器。重置后`ASYNCLISTADDR`寄存器的默认值未定义，当异步调度启用位为零时，调度被禁用。

当计划被禁用时，软件只能写入具有定义结果的寄存器。例如，USBCMD中的异步计划启用位和USBSTS寄存器中的异步时间表状态位为零。系统软件通过将（队列头的）有效内存地址写入该寄存器，实现异步调度的执行。然后，软件通过将异步调度启用位设置为1来启用异步调度。当异步计划状态位为1时，实际上会启用异步计划。

当主机控制器开始为异步调度提供服务时，它开始使用`ASYNCLISTADDR`寄存器的值。它读取第一个引用的数据结构，并开始执行事务，并在适当的时候遍历链表。当主机控制器“完成”处理异步调度时，它会在`ASYNCLISTADDR`寄存器中保留上次访问的队列头的水平指针的值。下次访问异步调度时，这是将要服务的第一个数据结构。这为处理异步调度提供了循环公平性。

当以下事件之一发生时，主机控制器“完成”处理异步计划：

- ① 微帧结束。
- ② 主机控制器检测到空列表条件（即参见第4.8.3节）
- ③ 已通过USBCMD寄存器中的异步计划启用位禁用该计划。

异步列表中的队列头被链接到一个简单的循环列表中，如图4-4所示。队列头数据结构是唯一可以链接到异步调度中的有效数据结构。异步调度中的同步传输描述符（TD或siTD）会产生未定义的结果。

队列头中的最大数据包大小字段的大小被设置为适应所有非同步传输类型使用该数据结构。USB规范2.0版规定了所有传输类型和传输速度的最大数据包大小。系统软件应始终根据核心规范要求对队列头数据结构进行参数化。

### 4.8.1 将队列头添加到异步计划

这是软件需求部分。有两个独立的事件用于将队列头添加到异步计划中。第一个是异步列表的初始激活。第二种方法是将一个新的队列头插入到激活的异步列表中。

激活列表很简单。系统软件将队列头的物理内存地址写入`ASYNCLISTADDR`寄存器，然后通过将`USBCMD`寄存器中的异步调度启用位设置为1来启用列表。

将队列头插入活动列表时，软件必须确保从主机控制器的角度来看，计划始终是一致的。这意味着系统软件必须确保所有队列头指针字段都是有效的。例如，`qTD`指针具有设置为一或引用有效`qTD`的`T`位，并且水平指针引用有效的队列头数据结构。以下算法代表了功能要求：

```
InsertQueueHead (pQHeadCurrent, pQueueHeadNew)
--  
-- 要求：所有输入必须正确初始化。  
--  
-- pQHeadCurrent是指向已在活动列表中的队列头的指针  
-- pQueueHeadNew是指向要添加的队列头的指针  
--  
-- 此算法将新的队列头链接到现有列表  
--  
pQueueHeadNew.HorizontalPointer      = pQHeadCurrent.HorizontalPointer=物理地  
址 (pQueueHeaderNew)  
结束插入队列头
```

#### 4.8.2 从异步计划中删除队列头

这是软件需求部分。有两个独立的事件用于从异步计划中删除队列头。第一个是关闭（停用）异步列表。第二种方法是从激活的列表中提取单个队列头。

软件通过将`USBCMD`寄存器中的异步计划启用位设置为零来停用异步计划。当`USBSTS`寄存器中的异步调度状态位为零时，软件可以确定列表何时空闲。

正常的操作模式是软件在不关闭异步调度的情况下从异步调度中删除队列头。软件不得从计划中删除活动的队列头。软件应首先停用所有活动的`qTD`，等待队列头处于非活动状态，然后从异步列表中删除队列头。软件通过以下算法从异步列表中删除队列头。如图所示，取消连接非常容易。软件仅仅必须确保主机控制器可到达的所有链路指针保持一致。

```
取消连接QueueHead (pQHeadPrevious, pQueueHeadToUnlink, pQHeadNext)
--  
-- 要求：所有输入必须正确初始化。  
--  
-- pQHeadPrevious是指向引用  
-- 要删除的队列头  
-- pQHeadToUnlink是指向要删除的队列头的指针  
-- pQheadNext是指向仍在计划中的队列头的指针。软件  
-- 为该指针提供了以下严格规则：  
--   如果主机软件是一个队列头，则pQHeadNext必须是  
-- 与pQueueheadToUnlink.HorizontalPointer 相同。如果主机软件  
--   取消连接一系列队列头的链接，pQHeadNext必须为  
-- 由软件 设置为计划中剩余的队列头。  
-- 此算法取消队列头与循环列表的链接  
--  
pQueueHeadPrevious.HorizontalPointer=pQueueHeadToUnlink.HorizontalPointer  
pQueueHeaderToUnlink.HorizontalPointer=pQHeadNext  
结束取消连接队列头
```

如果软件删除了`H`位设置为1的队列头，则必须选择仍链接到计划中的另一个队列头，并将其`H`位设为1。这应该在删除队列头之前完成。要求是软件在异步调度中保留一个队列头，其`H`位设置为1。

当软件从异步调度中删除了一个或多个队列头时，不知道主机控制器是否有指向它们的缓存指针。类似地，主机控制器可能保留缓存的信息多长时间是未知的，因为它取决于实现，并且可能受到

进度负荷的实际动态。因此，一旦软件从异步列表中删除了队列头，它就必须保持队列头（链接指针等）的一致性。在知道主机控制器没有指向任何已删除数据结构的指针的本地副本之前，它不能干扰已删除的队列头。

软件用于确定何时修改删除的队列头与主机控制器握手是安全的方法。握手机制允许软件从异步调度中删除项目，然后执行一个简单、轻量级的握手，该握手被软件用作密钥，它可以释放（或重用）与它从异步调度删除的数据结构相关的内存。

握手是用主机控制器中的三个比特来实现的。第一个位是一个命令位（USBCMD寄存器中的异步前进门铃中断位），它允许软件通知主机控制器已从其异步调度中删除某些内容。第二位是状态位（USBSTS寄存器中的异步前进中断位），主机控制器在释放了可能引用刚刚移除的数据结构之一的所有片上状态之后设置该状态位。当主机控制器将该状态位设置为1时，它还将命令位设置为0。第三位是与状态位匹配的中断使能（USBINTR寄存器中的异步前进中断位）。如果状态位是1并且中断使能位是1，则主机控制器将断言硬件中断。

图4-10显示了一个一般示例。在这个例子中，使用上面的算法将连续的队列头（B和C）从调度中取消链接。在取消链接操作之前，主机控制器具有队列头a的副本。取消链接算法要求，当软件取消每个队列头的链接时，将保留在异步调度中的队列头的地址加载到未链接的队列头。

当主机控制器观察到门铃位被设置为1时，它记录本地可达时间表信息。在本例中，本地可达调度信息包括两个队列头（A&B）。只要主机控制器已经遍历超过当前可到达调度信息（即，在该示例中遍历超过队列头（B）），主机控制器就可以设置状态位（并清除门铃位）。

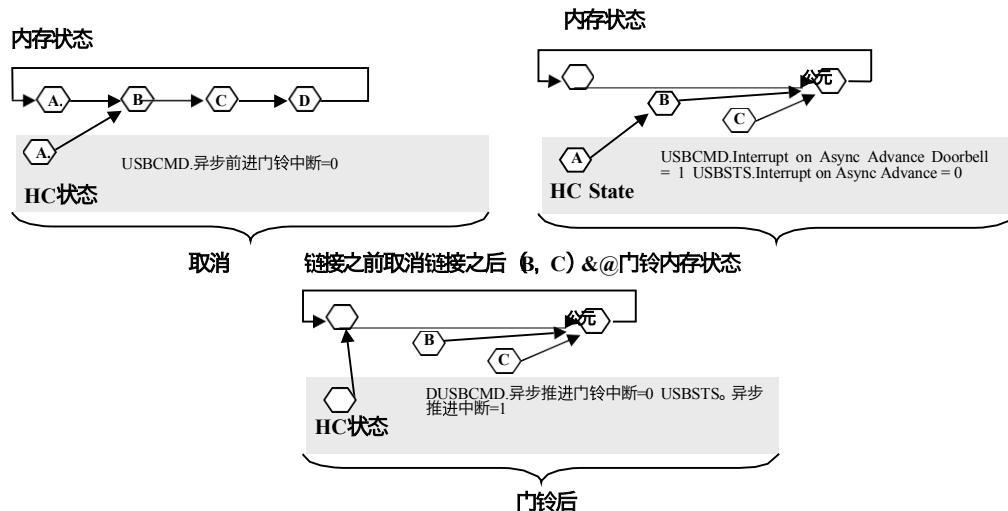


图4-10。通用队列头取消链接方案

或者，在将异步推进状态位设置为1之前，允许主机控制器实现遍历整个异步调度列表（例如，观察队列的头部（两次））。

在软件观察到异步前进中断状态位被设置为1之后，在门铃的断言之后，软件可以重用与被移除的队列头相关联的存储器。在再次使用门铃握手之前，软件应确认USBSTS寄存器中指示的异步前进中断状态。

### 4.8.3 空异步计划检测

增强型主机控制器接口使用两位来检测异步调度何时为空。队列头数据结构（见图3-7）在队列头中定义了一个H位，它允许软件将队列头标记为回收列表的头。增强型主机控制器接口还在USBSTS寄存器（*Reclamation*）中保持1位标志，当增强型接口主机控制器观察到H位设置为1的队列头时，该1位标志被设置为0。当异步调度中的任何USB事务被执行时（或当异步调度开始时，请参见第4.8.5节），状态寄存器中的回收标志被设置为1。

如果增强型主机控制器接口遇到H位为1和*Reclamation*位为0，则EHCI控制器简单地停止异步调度的遍历。

如图4-11所示，是一个调度中H位的示例。

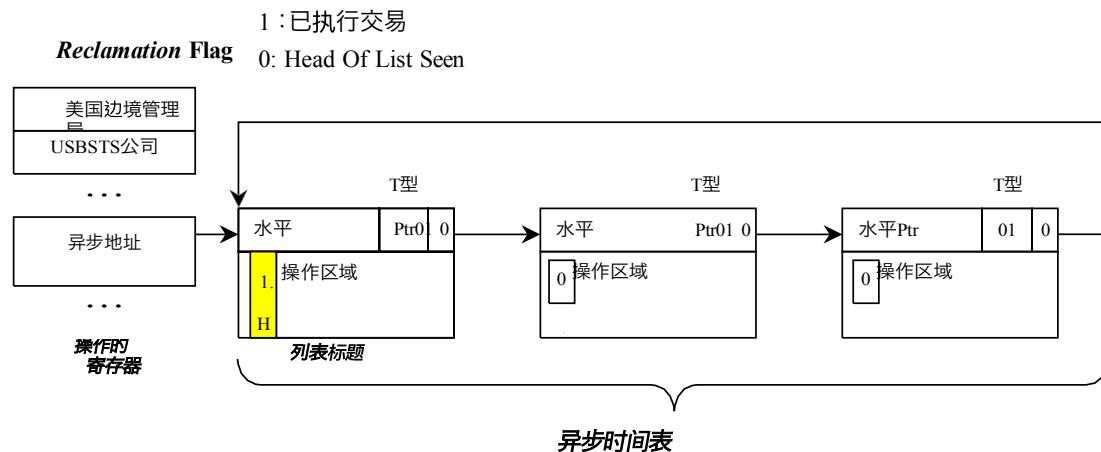


图4-11。异步时间表列表，带注释以标记列表标题

软件必须确保最多有一个H位设置为1的队列头，并且它始终与调度一致。

### 4.8.4 在EOF之前重新启动异步计划

存在许多情况，其中主机控制器将在微帧结束之前很久检测到空列表。重要的是要记住，在许多情况下，由于来自所有端点的Nak/Nyet响应，调度遍历已经停止。

一个特别令人感兴趣的例子是，在微帧的早期发生批量端点的开始分割。考虑到EHCI简单的遍历规则，该事务的完全拆分可能会非常快地Nak/Nyet。如果它是调度中的唯一项目，那么主机控制器将在微帧中很早停止遍历异步调度。为了向该端点提供合理的服务，主机控制器应该在当前微帧结束之前发出完整的拆分，而不是等到下一个微帧。

当主机控制器空闲异步调度遍历的原因是空列表检测时，主机控制器必须实现“唤醒”方法来恢复异步调度的遍历。下面描述一个示例方法。

#### 4.8.4.1

问题是：主机控制器在重新启动之前应该保持空闲多长时间？

这个例子中的答案是基于导出一个清单常量，该常量是主机控制器在重新启动遍历之前保持空闲的时间量。在本例中，清单常量称为`AsyncScheduleSleepTime`，其值为10毫秒。该值是根据第节中的分析得出的。

#### 4.8.4.2. 遍历算法很简单：

- ① 遍历异步调度，直到发生微帧结束事件或检测到空列表。如果事件是微帧结束，请尝试遍历周期性时间表。如果事件是一个空列表，则设置睡眠计时器并进入计划睡眠状态。
- ② 当睡眠计时器到期时，将工作上下文设置为异步计划启动条件，然后进入计划活动状态。启动上下文允许HC重新加载`Nakcnt`字段等，因此HC有机会在计划中运行一次以上的迭代。

这个过程只是在每个微帧中重复它自己。图4-12展示了一个示例状态机，用于管理异步调度遍历策略的活动状态和睡眠状态。有三种状态：主动遍历异步计划、休眠和非活动。后两者在与异步调度的交互方面相似，但非活动状态意味着主机控制器正忙于周期性调度或异步调度未启用。睡眠状态特别是一种特殊状态，其中主机控制器在恢复异步调度的执行之前仅等待一段时间。

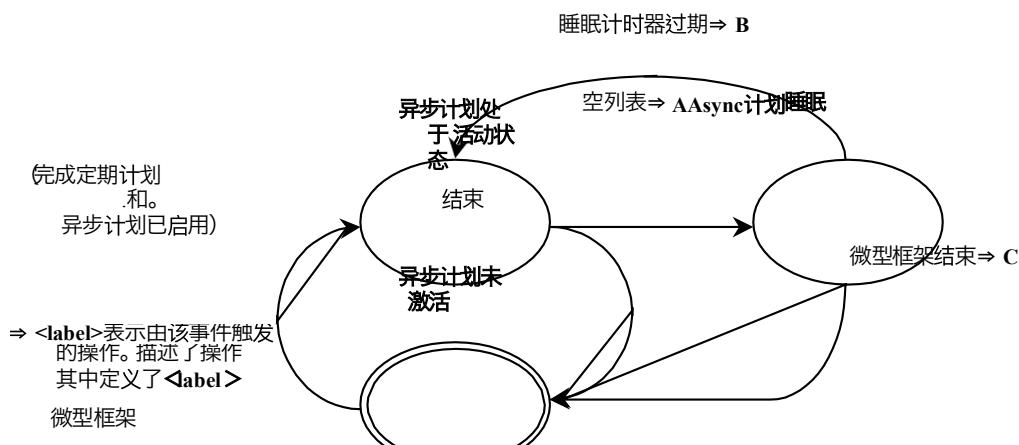


图4-12。用于管理异步调度遍历的示例状态机

表4-7中定义了图4-12中提到的动作。

表4-7。异步调度SM转换操作

操作标签	操作说明
A.	在检测到空列表时，主机控制器设置 <code>AsynchronousTraversal/Sleep Timer</code> 到 <code>AsyncScheduleSleep Time</code> 。
B	当 <code>AsynchronousTraversal/Sleep Timer</code> 到期时，主机控制器将USBSTS寄存器中的 <code>Reclamation</code> 位设置为1，并将Nak计数器重新加载状态机移动到 <code>WaitForListHead</code> （见第4.9节）。
C	主机控制器取消睡眠计时器（ <code>AsynchronousTraversal/Sleep Timer</code> ）。

#### 4.8.4.1.1 异步计划未激活

这是主机控制器重置后遍历状态机的初始状态。当USBCMD寄存器中的异步调度启用位为零时，遍历状态机将不会离开此状态。

当检测到微帧事件结束时，此状态是从**异步调度活动**或**异步调度睡眠**状态进入的。

#### 4.8.4.1.2 异步计划处于活动状态

当定期计划未激活时，此状态是从**异步计划未激活**状态进入的。当*AsynchronousTraversalSleepTimer*过期时，也会从**Async-Sched睡眠**状态进入。

在每次转换到此状态时，主机控制器都会将USBSTS寄存器中的*Reclamation*位设置为1。

在这种状态下，主机控制器将连续遍历异步调度，直到检测到微帧结束或空列表条件为止。

#### 4.8.4.1.3 异步计划睡眠

当检测到计划为空时，将从“**异步计划活动**”状态进入该状态。进入此状态时，主机控制器将*AsynchronousTraversalSleepTimer*设置为*AsyncScheduleSleepTime*。

#### 4.8.4.2 *AsyncScheduleSleepTime*的派生示例

推导是基于对主机控制器下一步可能做什么工作的分析。它假设主机控制器不保持关于异步调度中可能挂起的工作的任何状态。该时间表可以包含高满或低速控制和批量请求的任何可能组合。表4-8总结了时间表中可能出现的一些典型的“一笔交易”，以及交易完成所需的时间（如占地面积或挂钟）。

**表4-8。典型的低速/全速交易时间**

事务处理属性	足迹 (时间)	描述
速度 HS	11.9□s	最坏情况下的全尺寸批量数据事务的最大占地面积。
大小 512	9.45□s	近似最佳情况下的全尺寸批量数据事务的最大占地面积。
类型 大部分		
速度 金融服务	约50□s	近似适用于全尺寸批量数据。一个8字节的低速大约是2倍，或者在90到100之间□s
大小 64		
类型 大部分		
速度 金融服务	约12□s	8字节大容量/控制（即设置）的近似典型值
大小 8.		
类型 Cntrl 公司		

*AsyncScheduleSleepTime*值为10□s提供了系统存储器负载的合理放松，并且仍然为各种传输类型和有效载荷大小提供了良好的服务水平。例如，假设我们在为64字节全速批量请求发出启动拆分后检测到一个空列表。假设这是列表中唯一的东西，主机控制器将在第五个完整的拆分请求期间从集线器获得全速事务的结果。如果全速交易是一个IN，并且它是nak'd，则10□s的睡眠周期将允许主机控制器在第一次完全分割时获得NAK结果。

#### 4.8.5 异步计划遍历：启动事件

一旦HC通过空调度检测（第4.8.3节）使其自身空闲，它将自然激活并在每个微帧开始时从周期调度开始处理。此外，它可能在微帧的早期就闲置了。当这种情况发生时（在微帧早期空闲），HC必须在微帧期间偶尔重新激活，并遍历异步时间表以确定

是否可以取得任何进展。重新启动的要求和方法如第节所述

#### 4.8.4. 异步计划启动事件定义：

- ① 每当主机控制器从周期性调度转换到异步调度时。如果禁用了周期性计划并启用了异步计划，则微帧的开始相当于从周期性计划的转换，或者
- ② 异步调度遍历从休眠状态重新启动（参见第4.8.4节）。

### 4.8.6 回收状态位 (USBSTS寄存器)

空异步调度检测功能（第4.8.3节）的操作取决于USBSTS寄存器中*Reclamation*位的正确管理。主机控制器在遍历异步调度时获取新的队列头后，立即测试空调度（见第4.10.1节）。

如第4.8.5节所述，每当发生异步调度遍历开始事件时，主机控制器都需要将*Reclamation*位设置为1。每当主机控制器在遍历异步调度时执行事务时，*Reclamation*位也设置为1（见第4.10.3节）。

每当主机控制器发现其*H*位设置为1的队列头时，它都将*Reclamation*位设置为0。如果队列头在异步调度中，则软件应仅设置队列头的*H*位。如果软件将中断队列头中的*H*位设置为1，则产生的行为是未定义的。当从周期性调度执行时，主机控制器可以将回收位设置为零。

### 4.9 Nak计数器的操作模型

本节描述了队列头中定义的*NakCnt*字段的操作模型（见第3.6节）。软件不应将此功能用于中断队列头。主机控制器不需要强制执行此规则。

USB协议具有通过Nak响应由设备进行的内置流量控制。除了Ping功能之外，还有几种场景，其中端点可能在大多数时间自然地Nak或Nyet。一个例子是用于控制和批量端点的拆分事务协议的主机控制器管理。所有大容量端点（高速或全速）都通过相同的异步计划提供服务。*Start*分割事务和第一个*Complete*分割事务之间的时间可能非常短（例如，当端点是异步计划中唯一的端点时）。集线器NYET（有效地Naks）完整的拆分事务，直到经典事务完成。这可能导致主机控制器抖动内存，重复获取队列头并执行到集线器的事务，直到经典总线上的事务完成后，集线器才会完成。

队列头中有两个组件字段来支持节流功能：计数器字段（*NakCnt*）和计数器重新加载字段（*RL*）。*NakCnt*被主机控制器用作确定是否执行到端点的事务的标准之一。有两种操作模式与此计数器相关：

- ① **未使用。**当*RL*字段为零时设置此模式。对于通过*RL*字段为零的队列头执行的任何事务，主机控制器忽略*NakCnt*字段。软件必须将此选择用于中断端点。
- ② **裸油门模式。**当*RL*字段为非零时，会选择此模式。在该模式中，*NakCnt*字段中的值表示主机控制器在每个端点上将容忍的Nak或Nyet响应的最大数量。在这种模式下，HC将根据表4-9中列出的令牌/握手标准减少*NakCnt*字段。当端点成功移动数据时，主机控制器必须重新加载*NakCnt*（例如，奖励移动数据的设备的策略）。

表4-9。NakCnt字段调整规则

代币	握手	
	纳克	尼叶
英寸/平	递减 <i>NakCnt</i>	N/A (protocol error)
输出	递减 <i>NakCnt</i>	无操作 <sup>1</sup>
开始拆分	递减 <i>NakCnt</i>	N/A (protocol error)
完全拆分	无行动	减少 <i>NakCnt</i>

<sup>1</sup>对此响应的建议行为是重新加载*NakCnt*。

总之，系统软件通过将重新加载字段 (*RL*) 设置为非零值来启用计数器。如果*NakCnt*为非零，则主机控制器可以执行事务。如果*NakCnt*为零，主机控制器将不会执行事务。

第4.9.1节详细介绍了重新加载机制。

请注意，当异步调度中的所有队列头耗尽所有传输或所有*NakCnt*变为零时，主机控制器将检测到空的异步调度和空闲调度遍历（见第4.8.3节）。

任何时候，主机控制器开始对异步调度进行新的遍历时，都会假设发生启动事件，请参阅第4.8.5节。每次发生“开始事件”时，都会启用“裸计数”重新加载过程。

#### 4.9.1 裸计数重新加载控件

当主机控制器达到队列头的执行事务状态（意味着它具有活动操作状态）时，它进行检查以确定是否应从*RL*重新加载*NakCnt*字段（见第4.10.3节）。如果答案是肯定的，则将*RL*复制到*NakCnet*中。在重新加载之后或者如果重新加载不是活动的，则主机控制器评估是否执行事务。

在异步调度启动事件（启动事件的定义见第4.8.5节）之后，主机控制器必须在第一次通过回收列表期间重新加载队列头中的nak计数器（*NakCnt*，见图3-7）。异步调度应该最多有一个队列头标记为头（见图4-11）。图4-13显示了一个示例状态机，该状态机满足检测异步调度的第一次通过的主机控制器的操作要求。该状态机在主机控制器内部维护，仅用于队列头遍历状态期间nak计数器的门重新加载：执行事务（图4-14）。

如果*RL*字段（见图3-7）设置为零，则主机控制器不执行nak计数器重新加载操作。

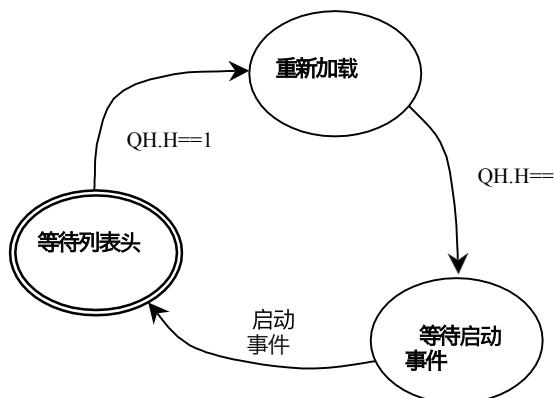


图4-13。用于控制Nak计数器重新加载的示例HC状态机

#### 4.9.1.1 等待列表头

这是初始状态。当发生第4.8.5节中定义的启动事件时，状态机从等待启动事件进入该状态。此状态的目的是等待异步调度头的第一次观测。当主机控制器获取其H位被设置为1的队列头时，就会发生这种情况。

#### 4.9.1.2 重新加载

当主机控制器获取H位设置为1的队列头时，从等待列表头状态进入该状态。在这种状态下，主机控制器将为访问的具有非零nak重新加载值 (RL) 字段的每个队列头执行nak计数器重新加载。

#### 4.9.1.3 等待启动事件

当提取H位设置为1的队列头时，从Do Reload状态进入此状态。在这种状态下，主机控制器将不会执行nak计数器重新加载。

### 4.10 通过队列头管理控制/批量/中断传输

本节概述了主机控制器如何与排队数据结构交互。

队列头使用第3.5节中定义的队列元素传输描述符 (qTD) 结构。一个队列头用于管理一个端点的数据流。队列头结构包含静态端点特性和功能。它还包含一个工作区，从该工作区执行端点的单个总线事务 (参见图3-7中定义的覆盖区)。每个qTD表示一个或多个总线事务，在本规范的上下文中将其定义为传输。

主机控制器使用队列头的一般处理模型很简单：

- ① 读取队列头,
- ② 从覆盖区域执行事务,
- ③ 将事务的结果写回覆盖区域
- ④ 移动到下一个队列头。

如果主机控制器在事务处理过程中遇到错误，主机控制器将在队列头的“状态”字段中设置一个（或多个）错误报告位。Status (状态) 字段累积在执行qTD期间遇到的所有错误（例如，队列头Status (状态) 字段中的错误位为“粘性”，直到传输 (qTD) 完成为止）。当传输完成时，此状态总是被写回到源qTD。

在传输（例如缓冲或停止条件）边界上，主机控制器必须自动前进（无需软件干预）到下一个qTD。此外，硬件必须能够停止队列，这样端点就不会发生额外的总线事务，主机控制器也不会推进队列。

队列头遍历的主机控制器操作状态机示例如图4-14所示。这个状态机是一个主机控制器应该如何遍历队列头的模型。主机控制器必须能够将队列从获取QH状态提前，以避免所有硬件/软件竞争条件。这种简单的机制允许软件简单地将qTD链接到队列头并激活它们，然后主机控制器将始终在它们可到达时找到它们。

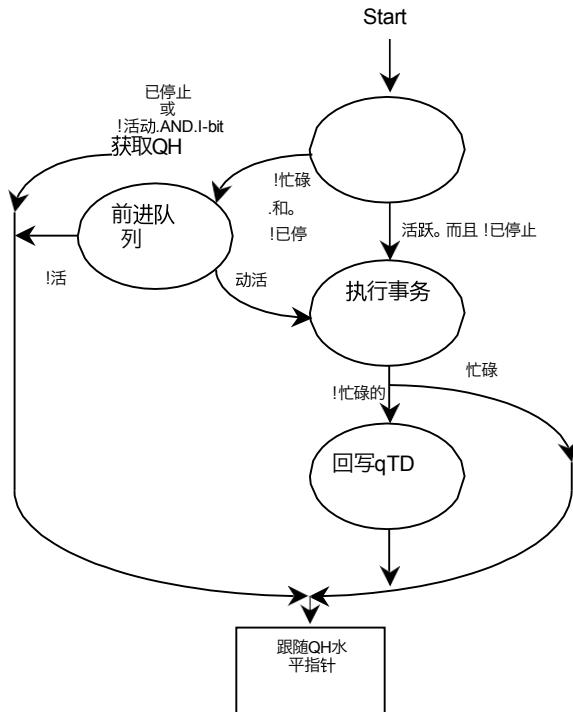


图4-14. 主机控制器队列头遍历状态机

无论传输类型或是否需要拆分事务，此遍历状态机都适用于所有队列头。以下各节介绍了每种状态。每个状态描述都描述了进入条件。**执行事务状态**（第4.10.3节）描述了所有端点的基本要求。第4.12.1节和第4.12.2节描述了需要拆分事务的端点的**执行事务状态**所需扩展的详细信息。

注意：在软件将队列头放入定期或异步列表之前，软件必须确保队列头已正确初始化。队列头应至少初始化为以下内容（队列头的布局请参见第3.6节）：

- ① 有效的静态终结点状态
- ② 对于第一次使用队列头，软件可以将队列头传输覆盖清零，然后设置“下一个qTD指针”字段值以引用有效的qTD。

#### 4.10.1 提取队列头

队列头可以从存储在ASYNCLISTADDR寄存器中的物理地址引用（第2.3.7节）。此外，它也可以从iTД、siTД、FSTN或其他队列头的下一个链接指针字段引用。如果引用链接指针的Typ字段设置为指示队列头，则假定引用图3-7中定义的队列头结构。

在这种状态下，主机控制器执行操作以实现空调度检测（第4.8.3节）和Nak计数器重新加载（第4.9节）。

在获取队列头之后，主机控制器执行以下查询以检测空调度：

- ① 如果队列头不是中断队列头（即S掩码为零），以及
- ② H位是1，并且
- ③ USBSTS寄存器中的Reclamation位为零。

当满足这些标准时，主机控制器将停止遍历异步列表（如第4.8.3节所述）。当不能满足这些标准，主机控制器继续调度遍历。

如果队列头不是中断，并且H位是1并且Reclamation位是1，则主机控制器在完成该状态之前将USBSTS寄存器中的Reclamation比特设置为0。

第4.9节详细描述了Nak计数器的重新加载操作。当已在芯片上读取队列头时，此状态即完成。

### 4.10.2 前进队列

为了推进队列，主机控制器必须找到下一个qTD，调整指针，执行覆盖并将结果写回到队列头。

如果覆盖活动和暂停位设置为零，则从FetchQHD状态进入该状态。在进入该状态时，主机控制器确定使用哪个下一指针来获取qTD，获取qTD并确定是否执行覆盖。注意，如果I位是1并且活动位是0，则主机控制器立即跳过对该队列头的处理，退出该状态并使用指向下一调度数据结构的水平指针。

如果字段Bytes to Transfer（要传输的字节数）不为零，并且Alternate Next qTD Pointer（备用下一个qTD指针）中的T位设置为零，则主机控制器使用Alternate Next qTD指针。否则，主机控制器将使用下一个qTD指针。如果下一个qTD指针的T位设置为1，则主机控制器退出该状态并使用指向下一个调度数据结构的水平指针。

使用所选择的指针，主机控制器获取所引用的qTD。如果获取的qTD的活动位设置为1，则主机控制器将用于到达qTD（下一个或备用下一个）的指针值移动到当前qTD指针字段，然后执行覆盖。如果所获取的qTD的活动位设置为零，则主机控制器中止队列前进，并跟随队列头的水平指针到下一个调度数据结构。

主机控制器根据以下规则执行覆盖：

- ① 覆盖区域中数据切换(dt)字段的值取决于数据切换控制(dtc)位的值（见表3-19）。
- ② 如果EPS字段指示端点是高速端点，则Ping状态字段由主机控制器保留。此字段的值不会因覆盖而更改。
- ③ C-prog-mask字段被设置为零（来自传入qTD的字段被忽略，覆盖区域的当前内容也是如此）。
- ④ 帧标记字段设置为零（来自传入qTD的字段被忽略，覆盖区域的当前内容也是如此）。
- ⑤ 覆盖区域中的NakCnt字段是从队列头的静态端点状态中的RL字段加载的。
- ⑥ 覆盖的所有其他区域由传入的qTD设置。

当主机控制器将写入提交到队列头时，它将退出此状态。

### 4.10.3 执行事务

只有当队列头的“状态”字段中的“活动”位设置为1时，主机控制器才会从“**获取队列头**”状态进入此状态。

在进入该状态时，主机控制器执行一些预操作，然后在承诺执行队列头的事务之前检查一些先决条件标准。

执行的预操作和预条件标准取决于队列头是否是中断端点。当

队列头的*S掩码*字段包含一个非零值。软件有责任确保*S掩码*字段根据传输类型进行适当初始化。如果*EPS*字段指示终点为低速或全速终点，则必须满足其他标准，请参见第4.12.1节和第4.12.2节。

① **中断传输先决条件标准**

如果队列头用于中断端点（例如非零*S掩码*字段），则FRINDEX[2:0]字段必须标识*S掩码*字段中有一个1的位。例如，只有当FRINDEX[2]等于101b时，00100000b的*S掩码*值才会评估为true。如果满足该条件，则主机控制器考虑事务的该队列头。

② **异步传输预操作和先决条件标准**

如果队列头不用于中断端点（例如，零*S掩码*字段），则主机控制器执行一个预操作，然后评估一个预条件标准：预操作为：

- ① 检查Nak计数器重新加载状态（第4.9节）。主机控制器可能需要重新加载Nak计数器字段。此时执行重新加载。

评估的前提条件是：

- ① 无论*NakCnt*字段是否已被重新加载，主机控制器都检查队列头中*NakCst*字段的值。如果*NakCnt*为非零，或者如果*Reload Nak Counter*字段为零，则主机控制器将此队列头视为事务的队列头。

③ **转移类型独立预操作**

无论传输类型如何，主机控制器总是执行至少一个预操作并评估一个预条件。预操作是：

- ① 主机控制器内部事务（向下）计数器*qHTransactionCounter*从队列头的*Mult*字段加载。对于异步列表上的队列头，允许主机控制器实现忽略这一点。这对于中断队列头是强制性的。软件应确保为传输类型适当设置*Mult*字段。

评估的前提条件是：

- ① 主机控制器确定微帧中是否有足够的时间来完成该事务（示例评估方法见第4.4.1.1节）。如果没有足够的时间来完成事务，主机控制器将退出此状态。
- ② 如果中断端点的*qHTransactionCounter*的值为零，则主机控制器退出此状态。

当预操作完成并且满足预条件时，主机控制器将USBSTS寄存器中的*Reclamation*位设置为1，然后开始使用队列头中的端点信息执行一个或多个事务。主机控制器在此状态下迭代*qHTransactionCounter*次执行事务。每个事务执行后，*qHTransactionCounter*将递减一。当发生以下事件之一时，主机控制器将退出此状态：

- ① *qHTransactionCounter*递减为零，或者
- ② 端点使用除ACK、<sup>4</sup>或
- ③ 交易出现交易错误，或
- ④ 队列头中的活动位变为零，或者
- ⑤ 微框架中没有足够的时间来执行下一个事务（关于框架边界测试的实施方法示例，请参见第4.4.1.1节）。

---

<sup>4</sup>请注意，对于高带宽中断OUT端点，如果事务失败，主机控制器可以选择立即重试事务。

每个事务的结果都记录在片上覆盖区域中。如果在事务处理过程中成功移动了数据，则覆盖区域中的传输状态将处于高级状态。为了提高队列头的传输状态，要传输的总字节数字段按事务中移动的字节数递减，切换数据切换位 (*dt*)，将当前页面偏移量提高到下一个适当值（例如，按成功移动的字节数来提高），并将*C\_page*字段更新到适当值（如有必要）。参见第4.10.6节。

请注意，当满足执行事务的所有其他条件时，要传输的总字节数字段可能为零。当这种情况发生时，主机控制器将执行到端点的零长度事务。如果*PID Code*字段指示IN事务，并且设备传送数据，则主机控制器将检测到数据包多路传输条件，在*Status*字段中设置多路传输和暂停位，将*Active*位设置为零，将结果写回源qTD，然后退出该状态。

在In令牌接收到数据PID不匹配响应的情况下，主机控制器必须忽略接收到的数据（例如，不推进接收到的字节的传输状态）。此外，如果端点是中断IN，则主机控制器必须记录事务发生（例如递减*qHTransactionCounter*）。如果*qHTransactionCounter*的结果值大于1，则建议（但不是必需）主机控制器继续执行此终结点的事务。

如果对IN总线事务的响应是Nak（或Nyet）并且*RL*为非零，则*NakCnt*递减1。如果*RL*为零，则不需要主机控制器的回写（对于接收Nak或Nyet响应的事务，并且*CErr*的值没有改变）。如果队列头是中断端点，则软件应将*RL*字段设置为零。不需要主机控制器硬件来强制执行此规则或操作。

在事务已经完成并且主机控制器已经完成结果的后处理（推进传输状态并且可能是*NakCnt*）之后，主机控制器将事务的结果写回到主存储器中的队列头的覆盖区域。

IN事务期间移动的字节数取决于设备端点传递的数据量。设备可以发送的最大字节数是最大数据包大小。OUT事务期间移动的字节数是最大数据包长度字节数或要传输的总字节数，以较小者为准。

如果在事务处理过程中出现事务错误，则主机控制器不会推进传输状态（如上所述）。*CErr*字段递减1，状态字段更新以反映观察到的错误类型。第4.15.1.1节总结了交易错误。

以下事件将导致主机控制器清除队列头的覆盖状态字段中的活动位。当活动位从1转换为0时，覆盖中的传输被认为是完成的。传输完成的原因（清除活动位）决定了下一个状态。

- ① *CErr*字段递减为零。当这种情况发生时，*Halted*位被设置为1，*Active*位被设置为0。这导致硬件无法推进队列，管道将停止。软件必须调解才能恢复。
- ② 设备使用STALL PID对事务进行响应。当这种情况发生时，*Halted*位被设置为1，*Active*位被设置成0。这导致硬件无法推进队列，管道将停止。软件必须调解才能恢复。
- ③ 事务完成后，要传输的总字节数字段为零。请注意，对于长度为零的事务，它在事务启动之前为零。当出现这种情况时，活动位被设置为零。
- ④ PID代码是IN，并且在事务处理期间移动的字节数小于最大数据包长度。当这种情况发生时，活动位被设置为零，并且存在短分组条件。在前进队列状态期间检测到短数据包情况。有关管理低速和全速交易的其他规则，请参阅第4.12节。
- ⑤ PID代码字段指示IN，并且设备发送的字节数超过预期（例如，最大数据包长度或要传输的总字节数，以较小者为准）（例如，数据包胡言乱语）。这导致主机控制器将*Halted*位设置为1。

除了NAK响应（当 $RL$ 字段为零时），主机控制器总是将事务的结果写回到主存储器中的覆盖区域。这包括传输完成的时间。对于高速端点，写回的队列头信息至少包括以下字段：

- ①  $NakCnt$ 、 $dt$ 、要传输的总字节数、 $C\_Page$ 、状态、 $CERR$ 和当前偏移

对于低速或全速设备，写回的队列头信息还包括以下字段：

- ①  $C$ 程序掩码、帧标记和 $S$ 字节。

此状态的持续时间取决于完成事务所需的时间，并且向覆盖层写入的状态已提交。

#### 4.10.3.1 停止排队头

仅为通过队列头（控制、批量和中断）管理的传输类型定义了暂停端点。以下事件表明，如果没有驾驶员的干预，端点已达到无法再进行活动的状态：

- ① 端点可以在事务期间返回STALL握手，
- ② 一个事务有三个连续的错误条件，或者
- ③ 端点上发生数据包巴氏错误。

当这些事件中的任何一个发生时（对于队列头），主机控制器停止队列头并将USBSTS寄存器中的USBERRINT状态位设置为1。要停止队列头， $Active$ 位设置为零， $Halted$ 位设置为1。当队列停止时，可能会设置其他错误状态位。当传输完成时，主机控制器总是将覆盖区域写回到源qTD，而不管原因是什么（正常完成、短分组或停止）。主机控制器不会在导致“暂停”条件的事务上推进传输状态（例如，不需要对要传输的总字节数、 $C\_Page$ 、当前偏移量和 $dt$ 进行更新）。主机控制器必须根据需要更新 $CErr$ 。

当队列头停止时，USBSTS寄存器中的 $USB$ 错误中断位被设置为1。如果USBINTR寄存器中的 $USB$ 错误中断启用位设置为1，则在下一个中断阈值处生成硬件中断。

#### 4.10.3.2 异步计划驻车模式

异步调度园区模式是一种特殊的执行模式，可以由系统软件启用，其中允许主机控制器在异步调度继续水平遍历之前，从异步调度中的高速队列头执行多个总线事务。此功能对“定期”计划中的队列头或其他数据结构没有影响。此功能在意图上类似于周期计划中使用的 $Mult$ 功能。其中，作为 $Mult$ 特征是可针对每个端点进行调整的特征； $park$ 模式是一种应用于异步调度中所有高速队列头的策略。它本质上是对到同一端点的连续总线事务的迭代器的规范。管理总线事务的所有规则以及第4.10.3节中定义的结果均适用。此功能仅指定在移动到异步列表中的下一个队列头之前，允许主机控制器从同一队列头连续执行多少次。该特性应允许主机控制器为那些能够以最大速率移动数据的设备获得更好的总线利用率，同时为所有端点提供公平的服务。

主机控制器通过将HCCPARAM寄存器中的异步调度园区能力位设置为1，将其支持该功能的能力导出到系统软件。该信息为系统软件提供了关键信息，即USBCMD寄存器中的异步调度泊车模式启用和异步调度泊车方式计数字段是可修改的。系统软件通过向异步计划驻车模式启用位写入一来启用该功能。

当泊车模式未启用时（例如USBCMD寄存器中的异步调度泊车模式启用位为零），主机控制器在异步调度的每次遍历中，每个高速队列头执行的总线事务不得超过一个。

启用驻车模式时，主机控制器不得将该功能应用于EPS字段指示低速/全速设备的队列头（即，每次遍历异步调度时，每个低速/全速队列头只允许一个总线事务）。驻车模式只能应用于异步调度中的队列头，其EPS字段指示其为高速设备。

主机控制器必须将驻车模式应用于EPS字段指示高速端点的队列头。主机控制器可以在高速队列头上执行的连续总线事务的最大数量由USBCMD寄存器中的异步调度驻车模式计数段中的值确定。软件不得将异步计划驻车模式启用位设置为1，也不得将异步安排驻车模式计数段设置为零。未定义结果行为。一个示例行为示例描述了实现停车模式的主机控制器的操作要求。

此功能不会影响主机控制器如何处理第4.10.3节中定义的总线事务。它只影响当前队列头可以执行的连续总线事务的数量。所有边界条件、错误检测和报告照常适用。该功能在概念上类似于在周期性调度中使用Mult字段对队列头进行高带宽中断。

当异步调度驻车模式启用位为1，并且首先获取高速队列头并满足执行总线事务的所有标准时，主机控制器从异步调度驻车灯模式计数有效地加载内部递减计数器PM计数。在总线事务之后，PM计数递减。主机控制器可以继续从当前队列头执行总线事务，直到PM计数变为零、检测到错误、用于当前传输的缓冲区耗尽或者端点以流控制或STALL握手进行响应。表4-10总结了影响主机控制器是否继续当前队列头的另一个总线事务的响应。

**表4-10。基于端点响应和剩余传输状态的驻车模式操作**

PID 控 制 器	终 结 点 响 应	交易后的转移状态		行动
		PM计数	要传输的字节	
在	数据[0,1] 带最大数据 包大小的数 据	不是零	非零	允许执行另一个总线事务。 <sup>1, 2</sup>
		不是零	零	退出qTD并转到下一个QH
		零	不在乎	转到下一个QH。
	数据[0,1] 带短数 据包	不在乎	不在乎	退出qTD并转到下一个QH。
	纳克	不在乎	不在乎	转到下一个QH。
	失速, XactErr公司	不在乎	不在乎	转到下一个QH。
输出	确认	不是零	非零	允许执行另一个总线事务。 <sup>2</sup>
		不是零	零	退出qTD并转到下一个QH
		零	不在乎	转到下一个QH。
	NYET, 纳克	不在乎	不在乎	转到下一个QH。
	失速, XactErr公司	不在乎	不在乎	转到下一个QH

表4-10。基于端点响应和剩余传输状态的驻车模式操作 (续)

方向	终结点响应	交易后的转移状态		行动
		PM计数	要传输的字节	
PING (PING)	确认	非零	非零	允许执行另一个总线事务。 <sup>2</sup>
	纳克	不在乎	不在乎	转到下一个QH
	失速, XactErr公司	不在乎	不在乎	转到下一个QH

<sup>1</sup>注意，如果检测到PID不匹配（例如，预期的DATA1和接收的DATA0，或者反之亦然），则主机控制器可以继续从当前高速队列头执行总线事务（如果PM计数不等于零）。

<sup>2</sup>注意，当PM计数为非零时，本规范不要求主机控制器执行另一个总线事务。鼓励实现适当地权衡复杂性和性能。

#### 4.10.4 回写qTD

当活动位设置为零时，从**执行事务**状态进入该状态。回写的源数据是队列头覆盖区域的传输结果区域（见图3-7）。主机控制器使用当前qTD指针字段作为qTD的目标地址。队列头传输结果区域被写回到目标qTD的传输结果区域。这种状态也被称为：qTD退休。必须写回源qTD的字段包括要传输的总字节数、Cerr和状态。

此状态的持续时间取决于qTD写回已提交的时间。

#### 4.10.5 跟随队列头水平指针

当存在以下任何条件时，主机控制器必须使用队列头中指向下一个调度数据结构的水平指针：

- ① 如果活动位是从**执行事务**状态退出时的一位，或者
- ② 当主机控制器退出**回写qTD**状态时，或者
- ③ 如果由于目标qTD未处于活动状态，“**前进队列**”状态未能使队列前进，或者
- ④ 如果Halted位是从**Fetch QH**状态退出时的一位。

不存在主机控制器在使用水平指针读取下一个链接的数据结构之前等待直到当前事务完成的功能要求。但是，它必须等到当前事务完成后才能执行下一个数据结构。

#### 4.10.6 用于带qTD的数据流的缓冲区指针列表

qTD有一个缓冲区指针数组，用于引用数据缓冲区进行传输。此规范要求与传输相关联的缓冲区实际上是连续的。这意味着：如果缓冲区跨越多个物理页面，则必须遵守以下规则（图4-15举例说明）：

- ① 缓冲区的第一部分必须从页面中的某个偏移量开始，并延伸到页面的末尾。
- ② 剩余的缓冲区不能以分散在内存中的小块进行分配。对于第一页之外的每个4K块，每个缓冲区部分都与完整的4K页匹配。最后一部分可能只大到足以占据页面的一部分，必须从页面顶部开始，并在该页面内连续。

qTD中的缓冲区指针列表足够长，可以支持20K字节的最大传输大小。当所有五个缓冲区指针都被使用并且第一个偏移为零时，就会发生这种情况。qTD处理具有任何起始缓冲区对齐的16K字节缓冲区。

主机控制器使用字段 *C\_Page* 字段作为索引值来确定列表中的哪个缓冲区指针应用于启动当前事务。主机控制器对缓冲器的每个物理页使用不同的缓冲器指针。这总是正确的，即使缓冲区在物理上是连续的。

主机控制器必须检测当前事务何时将跨越页面边界，并自动移动到页面指针列表中的下一个可用缓冲区指针。下一个可用指针是通过增加 *C\_Page* 并从列表中拉出下一个页面指针来达到的。软件必须确保有足够的缓冲区指针来移动“要传输的字节”字段中指定的数据量。

图4-15说明了系统软件如何初始化缓冲区指针列表和 *C\_Page* 字段（传输大小为16383字节）的标称示例。*C\_Page* 设置为零。页面0的高20位引用了物理页面的开头。当前偏移（队列头D字7的低12位）保持页面中的偏移，例如2049（例如4096-2047）。其余的页面指针被设置为引用每个后续4K页面的开头。

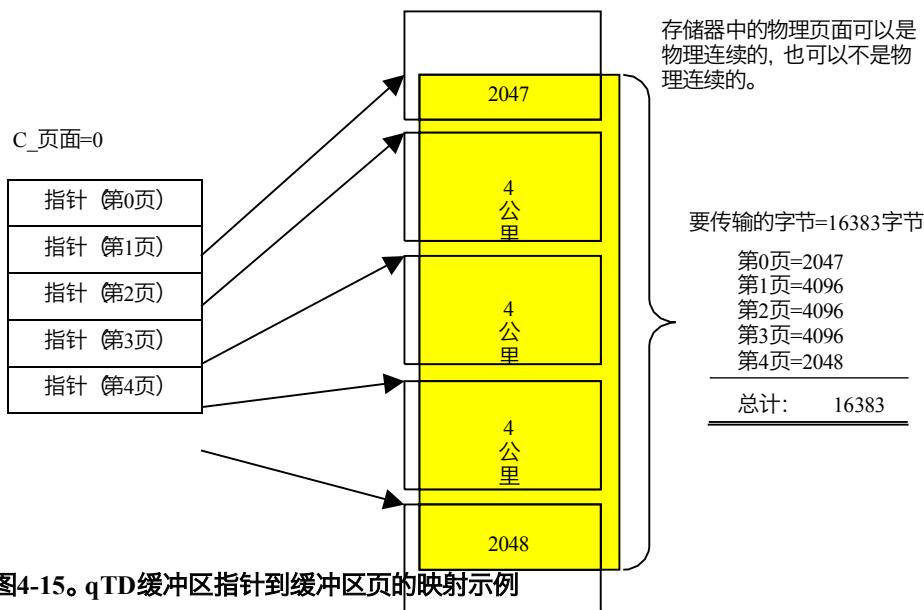


图4-15。qTD缓冲区指针到缓冲区页的映射示例

对于qTD上的第一个事务（假设为512字节事务），主机控制器使用第一个缓冲区指针（因为 *C\_page* 设置为零，所以页面为0）并连接当前偏移字段。512字节在事务处理期间被移动，当前偏移和要传输的总字节被调整512，并被写回到队列头工作区。

在第4次事务处理期间，主机控制器在第0页需要511个字节，在第1页需要一个字节。主机控制器将把 *C\_Page*（增加到1），并使用第1页指针移动事务的最后一个字节。在第4个事务之后，活动页面指针是页面1指针，当前偏移量已滚动到1，并且两者都被写回覆盖区域。事务在缓冲区的其余部分继续进行，必要时主机控制器会自动移动到下一页指针（即 *C\_page*）。

主机控制器如何处理 *C\_Page* 有三个条件。

- ① 当前事务不跨越页面边界。*C\_Page* 的值不由主机控制器调整。
- ② 当前事务确实跨越了页面边界。主机控制器必须检测到页面交叉情况，并在向USB传输数据/从USB传输数据时前进到下一个缓冲区。

- ② 当前事务在页面边界上完成（即，为当前事务移动的最后一个字节是当前页面指针在页面中的最后字节）。在写回事务的状态之前，主机控制器必须递增  $C\_Page$ 。

请注意，主机控制器可以对  $C\_Page$  进行的唯一有效调整是递增一。

#### 4.10.7 将中断队列头添加到周期调度

从周期性帧列表到队列头的链路路径建立了可以在哪些帧中为队列头执行事务。队列头被链接到定期计划中，以便以适当的速率对其进行轮询。系统软件在队列头的  $s$  掩码中设置一个位，以指示在1毫秒周期内应为队列头执行事务的微帧。软件必须确保周期性计划中的所有队列头都将  $S\text{-}Mask$  设置为非零值。在周期调度的上下文中具有零值的  $S$  掩码产生未定义的结果。

如果所需的轮询速率大于一帧，则系统软件可以使用队列头链接和  $S$  掩码值的组合来在调度表中分布相等轮询速率的中断，从而以最有效的方式分配和管理周期性带宽。一些示例如表4-11所示。

表4-11。具有2ms轮询速率的中断传输的周期性参考模式示例

帧#参考序列	描述
0、2、4、6、8等。 $S$ 掩码=01h	2毫秒间隔（16微帧）的队列头被链接到周期性调度中，使得它可以从前一列中指示的周期性帧列表位置到达。此外，队列头中的 $S\text{-}Mask$ 字段设置为01h，表示端点的事务应在该帧的微帧0期间在总线上执行。
0、2、4、6、8等。 $S$ 掩码=02h	b间隔为2毫秒的队列头的另一个示例以与前一示例完全相同的间隔链接到周期性帧列表中。然而， $S\text{-}Mask$ 被设置为02h，指示端点的事务应该在帧的微帧1期间在总线上执行。

#### 4.10.8 管理来自队列头的传输完全中断

当完成的传输 (qTD) 具有设置为1的“完成时中断” (IOC) 位时，或者每当传输 (qTD) 以短分组完成时，主机控制器将设置中断，以在下一个中断阈值用信号通知。如果系统软件需要多个qTD来完成客户端请求（即，类似于控制传输），则中间qTD不需要中断。系统软件可能只需要一次中断就可以通知它完整的缓冲区已经传输。系统软件可能会将IOC设置为更频繁地发生。这样做的动机可能是它希望尽早通知，以便及时重新使用接口数据结构。

### 4.11 Ping控制

USB 2.0为高速设备定义了一个名为Ping的附加协议。所有USB都需要Ping  
2.0高速批量和控制端点。拆分事务流不允许Ping。该协议的扩展消除了Naking OUT端点的不良副作用。*Status*字段有一个Ping State位，主机控制器使用它来确定它将在到端点的下一个事务中使用的下一实际PID（见表3-16）。Ping状态位仅由满足以下条件的队列头的主机控制器管理：

- ① 队列头不是中断，并且

① EPS字段等于高速和

② PIDCode字段等于OUT

表4-12说明了主机控制器负责维护PING协议的状态转换表。有关Ping协议的详细说明，请参阅USB规范2.0版中的第8章。

**表4-12。Ping控制状态转换表**

现在的	事件		下一个
	主办	装置	
打乒乓球	PING (PING)	Nak公司	打乒乓球
打乒乓球	PING (PING)	确认	请勿外出
打乒乓球	PING (PING)	XactErr <sup>1</sup>	打乒乓球
打乒乓球	PING (PING)	货摊	N/C <sup>2</sup>
请勿外出	输出	Nak公司	打乒乓球
请勿外出	输出	奈特	打乒乓球 <sup>3</sup>
请勿外出	输出	确认	请勿外出
请勿外出	输出	XactErr <sup>1</sup>	打乒乓球
请勿外出	输出	货摊	N/C <sup>2</sup>

<sup>1</sup>事务错误 (XactErr) 是指主机错过握手的任何时间。

<sup>2</sup> Ping状态位不需要转换更改。暂停握手会导致端点被暂停（例如。  
活动设置为零，停止设置为一）。需要软件干预才能重新启动队列。

<sup>3</sup>对OUT的Nyet响应意味着设备已经接受了数据，但此时无法再接收数据。主机必须推进传输状态，另外，将Ping状态位转换为Do Ping。

Ping状态位具有以下编码：

价值	含义
0B	执行OUT主机控制器将在此端点的下一个总线事务期间使用OUT PID。
1B	Do Ping主机控制器将在此端点的下一个总线事务期间使用Ping PID。

定义的ping协议（见USB 2.0规范，第8章）允许主机在初始化ping协议时不精确（即，当我们不知道设备上是否有空间时，在Do OUT中启动）。

主机控制器管理Ping状态位。系统软件在初始化队列头时设置队列头中的初始值。主机控制器在所有队列前进过程中保留Ping状态位。

这意味着，当新的qTD被写入队列头覆盖区域时，Ping状态位的先前值被保留。

## 4.12 拆分交易

USB 2.0定义了通过USB 2.0集线器管理USB 1.x数据流的总线协议扩展。本节描述了主机控制器如何使用接口数据结构来管理具有完整和低速设备的数据流，这些设备连接在USB 2.0集线器下，使用拆分事务协议。有关拆分事务协议的完整定义，请参阅USB 2.0规范。

全速和低速设备的枚举方式与高速设备相同，但到全速和低速端点的事务使用高速总线上的拆分事务协议。拆分事务协议是全速或低速事务的封装（或包装）。高速

协议的包装部分寻址到USB 2.0集线器和事务转换器，全速或低速设备连接在该集线器和事务转换转换器下面。

EHCI接口使用专用数据结构来管理全速等时数据流（见第3.4节），批量和中断使用排队数据结构进行管理（见第3.6节）。接口数据结构需要使用USB 2.0集线器的设备地址和事务转换器编号进行编程，该集线器用作该链路的低速/全速主机控制器。以下各节介绍了主机控制器必须如何处理和管理拆分事务协议的详细信息。

#### 4.12.1 异步传输的拆分事务

异步调度中的队列头（EPS字段指示满设备或低速设备）向主机控制器指示它必须使用拆分事务来流式传输该队列头的数据。所有全速批量和全速低速控制都通过异步调度中的队列头进行管理。

软件必须使用事务转换器的适当设备地址和端口号初始化队列头，该事务转换器用作连接端点的链路的完整/低速主机控制器。软件还必须将拆分事务状态位（*SplitXState*）初始化为**Do Start split**。最后，如果端点是控制端点，则系统软件必须将队列头中的控制传输类型（C）位设置为1。如果这不是控制传输类型的端点，则必须通过软件将C位初始化为零。主机控制器使用此信息来正确设置拆分事务总线令牌中的端点类型（ET）字段。当C位为零时，拆分事务令牌的ET字段被设置为指示批量端点。当C位为1时，分割事务令牌的ET字段被设置为指示控制端点。有关详细信息，请参阅USB规范2.0版的第8章。

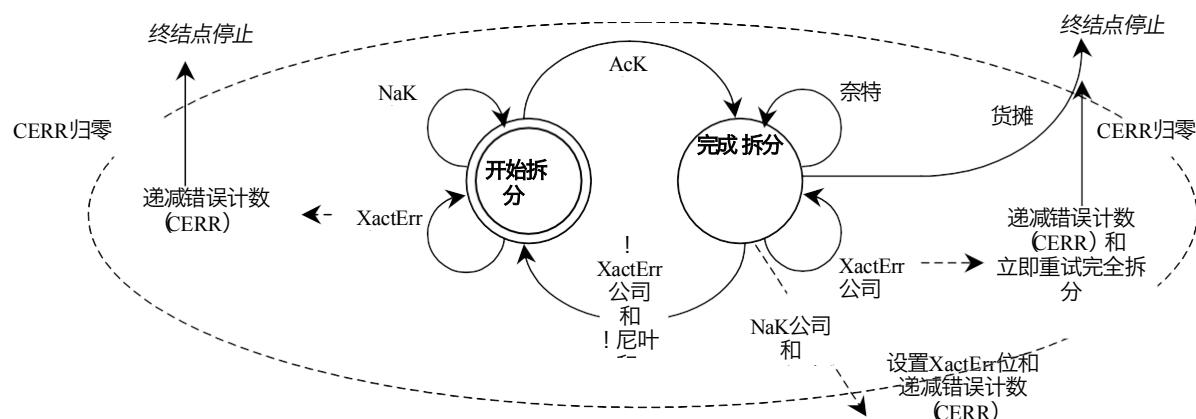


图4-16。主机控制器异步计划拆分事务状态机

##### 4.12.1.1 异步-执行启动拆分

这是软件必须初始化完整或低速异步队列头的状态。只有在完全拆分事务从事务转换器接收到不是Nyet握手的有效响应之后，才从**Do Complete Split**状态进入此状态。

对于处于这种状态的队列头，主机控制器将执行到适当事务转换器的启动拆分事务。如果总线事务在没有错误的情况下完成，并且*PidCode*指示IN或OUT事务，则主机控制器将重新加载错误计数器（*Cerr*）。如果它是一个成功的总线事务，并且*PidCode*指示SETUP，则主机控制器将不会重新加载错误计数器。如果事务转换器以Nak响应，则队列头保持这种状态，并且主机控制器继续到异步调度中的下一个队列头。

如果主机控制器超时事务（无响应或响应错误），则主机控制器递减  $Cerr$ ，然后继续到异步调度中的下一个队列头。

#### 4.12.1.2 异步-完成拆分

只有在开始拆分事务从事务转换器接收到Ack握手之后，才从Do Start Split状态进入此状态。

对于处于这种状态的队列头，主机控制器将执行一个完整的拆分事务到适当的事务转换器。如果事务转换器以Nyet握手进行响应，则队列头保持这种状态，错误计数器被重置，并且主机控制器前进到异步调度中的下一个队列头。当接收到队列头的PidCode指示IN或OUT的总线事务的Nyet握手时，主机控制器将重新加载错误计数器 ( $Cerr$ )。当接收到针对完整的分离总线事务的Nyet握手时，其中队列头的PidCode指示SETUP，主机控制器不得调整 $Cerr$ 的值。

独立于PIDCode，以下响应具有以下效果：

- ① 事务错误 ( $XactErr$ )。超时或数据CRC失败等。错误计数器 ( $Cerr$ ) 递减一，并立即重试完整的拆分事务（如果可能）。如果微帧中没有足够的时间执行重试，则主机控制器必须确保下次主机控制器从异步调度开始执行时，它必须从该队列头开始执行。如果在完全拆分真正完成之前向事务转换器发送了另一个开始拆分（对于某些其他端点），则事务转换器可能会转储结果（这些结果从未传递到主机）。这就是为什么核心规范规定重试必须立即进行的原因。实现这种行为的一种方法是不提前异步调度。当主机控制器在下一个微帧中返回异步调度时，调度中的第一个事务将是该端点的重试。如果 $Cerr$ 为零，则主机控制器必须停止队列。
- ② 纳克。目标端点Nak'd完全或低速事务。传输的状态未进入，状态已退出。  
如果PidCode是SETUP，则Nak响应是协议错误。 $XactErr$ 状态位设置为1，并且 $Cerr$ 字段递减。
- ③ 失速。目标终结点以STALL握手响应。主机控制器设置停止状态字节中的位使qTD失效但不尝试推进队列。

如果PidCode指示IN，则预期以下任何响应：

- ④ DATA0/1。在接收到数据时，主机控制器确保PID与预期的数据切换相匹配，并检查CRC。如果数据包是好的，则主机控制器将推进传输状态，例如，将数据指针移动所接收的字节数，将BytesToTransfer字段递减所接收的比特数，并切换dt位。然后，主机控制器将退出此状态。传输的响应和推进可以触发其他处理事件，例如qTD的退出和队列的推进。

如果数据序列PID与预期不匹配，则会忽略数据，传输状态不会提前，并退出此状态。

如果PidCode指示OUT/SETUP，则预期以下任何响应：

- ⑤ 确认。目标终结点接受了数据，因此主机控制器必须提前传输状态。“当前偏移”字段按“最大数据包长度”或“要传输的字节数”递增，以较小者为准。字段Bytes To Transfer（要传输的字节数）按相同的数量递减，并且数据切换位 (dt) 被切换。然后，主机控制器将退出此状态。  
推进传输状态可能会导致其他处理事件，如qTD的退出和队列的推进（见第4.10节）。

## 4.12.2 拆分事务中断

分割事务中断IN/OUT端点通过用于高速中断端点的相同数据结构进行管理。它们在定期计划中共存。队列头/qTD提供了可靠数据传输所需的一组功能，这是中断传输类型的特点。拆分事务协议完全在这个定义的功能转移框架内进行管理。例如，对于高速端点，主机控制器将访问队列头，执行高速事务（如果满足标准），并根据整个事务的结果提前（或不提前）传输状态。对于低速和全速端点，执行阶段的细节是不同的（即需要多个总线事务才能完成），但操作框架的其余部分是完整的。这意味着转让预付款等按照第4.10条的规定发生，但仅发生在分割交易完成时。

### 4.12.2.1 用于中断的拆分事务调度机制

全速和低速中断队列头具有指示全速或低速的EPS字段，并且具有非零的S掩码字段。主机控制器可以检测这种参数组合，并假设端点是周期性端点。低速和全速中断队列头需要使用拆分事务协议。主机控制器在分割令牌中设置端点类型(ET)字段，以指示事务是中断。这些事务是通过事务转换器的定期管道进行管理的。除非在周期性调度中使用了队列头，否则软件不应设置这些字段来指示队列头是中断。

系统软件通过预算和调度精确地管理每个/事务转换器的周期性管道，在这些微帧期间，每个端点的开始拆分和完成拆分将发生。事务转换器的特性是，高速事务协议必须在显式微帧期间执行，否则管道中的数据或响应信息将丢失。图4-17说明了EHCI周期性调度和队列头数据结构所支持的一般调度边界条件。S和CX标签表示软件可以（分别）安排开始拆分和完成拆分的微帧。

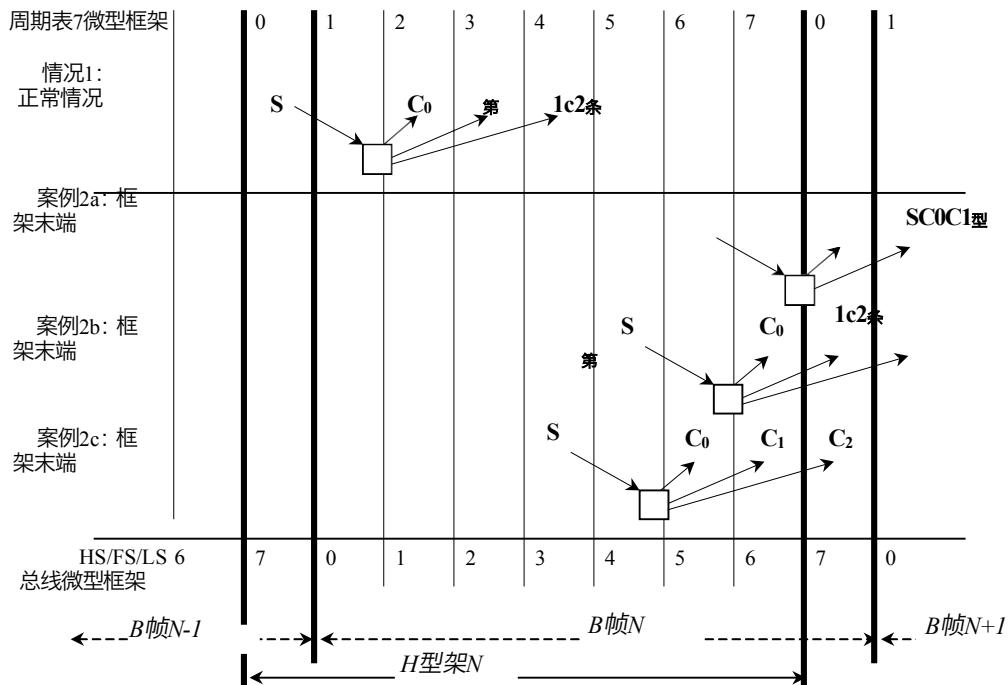


图4-17. 分割事务、中断调度的边界条件

调度案例包括：

- ① 情况1：正常调度情况是整个拆分事务完全由一个帧（在这种情况下为H-frame）约束。
- ② 案例2a到案例2c：USB 2.0集线器管道规则明确规定了必须计划何时以及多少次完整拆分，以考虑在完整/低速链路上最早到最晚的执行。当开始分割在微帧4或更晚时，完全分割可以跨越H帧边界。当这种情况发生时，H帧到B帧的对齐要求队列头可以从连续的周期性帧列表位置到达。除非使用FSTN，否则系统软件无法构建满足此要求的高效时间表。图4-18显示了定期计划的总体布局。

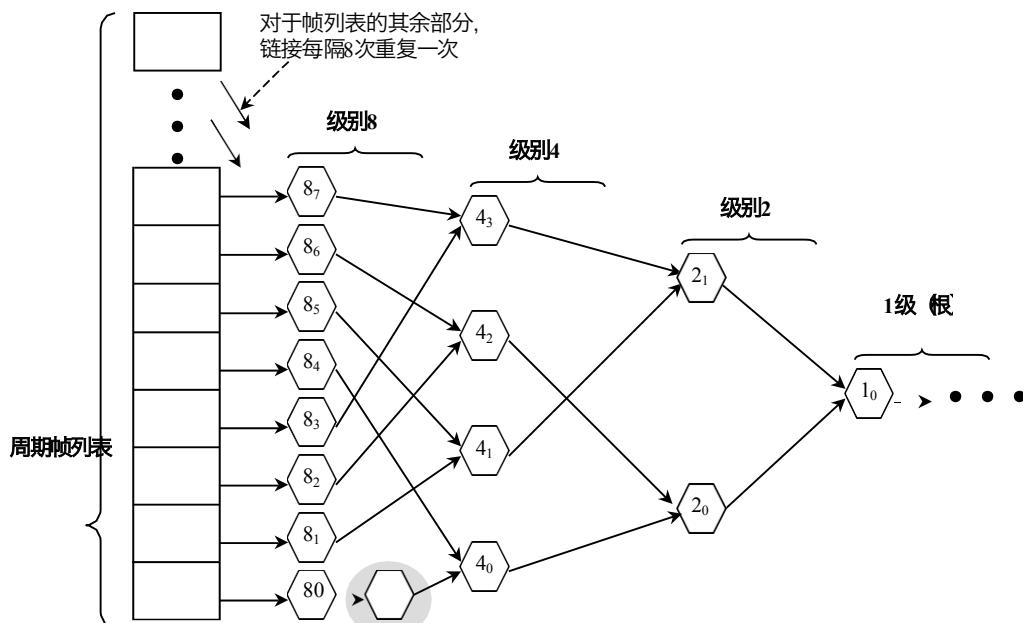


图4-18。利用中断扩展的EHCI周期调度的一般结构

周期帧列表实际上是叶级的二叉树，它总是从叶到根遍历。树中的每个级别对应于 $2^N$ 个轮询速率。软件可以通过将具有相同轮询速率要求的中断队列头分布在帧列表的所有可用路径上，来有效地管理USB上的周期性带宽。例如，系统软件可以调度八个轮询率为8的队列头，并在高速总线带宽分配中对它们进行一次说明。

当一个端点被分配了一个跨越帧边界的执行足迹时，该端点的队列头必须可以从帧列表中的连续位置到达。一个例子是如果<sub>80b</sub>中有这样一个端点。如果没有接口上的额外支持，为了在正确的时间到达<sub>80b</sub>，软件必须将<sub>81</sub>连接到<sub>80b</sub>。然后，它必须移动<sub>41</sub>，然后所有连接的东西都进入与<sub>40</sub>相同的路径。这破坏了二叉树的完整性，并禁止使用扩展技术。

FSTN数据结构用于保持二叉树结构的完整性并使得能够使用扩展技术。第4.12.2.2节定义了使用FSTN的硬件和软件操作模型要求。

以下队列头字段由系统软件初始化，以指示主机控制器何时执行分割事务协议的部分。

- ① **拆分X状态**。这是驻留在队列头的状态字段中的单个位（见表3-16）。此位用于跟踪拆分事务的当前状态。
- ② **□帧S掩码**。这是一个位字段，在该位字段中，系统软件设置与主机控制器应执行开始分割事务的微帧（在H帧内）相对应的位。这总是

由队列头的*Status*字段中的*SplitXState*位的值限定。例如，参考图4-17，情况一，*S*掩码的值为00000001b，表明如果主机控制器遍历队列头，并且*SplitXState*指示**Do\_Start**，并且FRINDEX[2:0]指示的当前微帧为0，则执行开始拆分事务。

- ① □ 框架*C*掩码。这是一个位字段，其中系统软件设置一个或多个对应于主机控制器应执行完整拆分事务的微帧（在*H*帧内）的位。该字段的解释始终由队列头的*Status*字段中的*SplitXState*位的值限定。例如，参考图4-17，情况一，*C*掩码的值为00011100b，表明如果主机控制器遍历队列头，并且*SplitXState*指示**Do\_Complete**，并且FRINDEX[2:0]指示的当前微帧为2、3或4，则执行完整的拆分事务。

软件有责任确保在*s*掩码和*C*掩码中设置位时正确执行*H*帧和*B*帧之间的转换

#### 4.12.2.2 FSTN的主机控制器操作模型

FSTN数据结构用于管理需要从连续帧列表位置（即边界情况2a到2c）到达的低速/全速中断队列头。FSTN本质上是一个后向指针，其意图类似于siTD数据结构中的后向指针字段（见第3.4.5节）。该功能为软件提供了一个简单的基元，用于保存时间表位置，重定向主机控制器以遍历前一帧中所需的队列头，然后恢复原始时间表位置并完成正常遍历。

FSTN的使用有四个组成部分：

1. FSTN数据结构，定义见第3.7节。
2. 保存位置指示器。这始终是一个带有反向路径链接指针的FSTN。*T*位设置为零。
3. 恢复指示灯。这始终是一个FSTN，其反向路径链接点*T*-比特设置为1。
4. 主机控制器FSTN遍历规则。

##### 4.12.2.2.1 FSTN的主机控制器操作模型

当主机控制器在微帧2到7期间遇到FSTN时，它只是遵循节点的正常路径链路指针来访问下一个调度数据结构。请注意，FSTN的*Normal Path Link Pointer.T*位可以设置为1，主机控制器必须将其解释为周期性列表标记的结束。

当主机控制器在微帧0或1中遇到保存位置FSTN时，它将保存正常路径链接指针，并设置一个内部标志，指示它正在恢复路径模式下执行。恢复路径模式修改主机控制器如何遍历时间表的规则，并限制执行总线事务时将考虑的数据结构。主机控制器继续在恢复路径模式下执行，直到遇到恢复FSTN或确定已到达微帧的末尾（请参阅下面列表中的详细信息）。在恢复路径模式下，计划遍历和有限执行的规则如下：

- ① 当遇到作为保存位置指示器的FSTN时，始终遵循正常路径链接指针。主机控制器不能递归地遵循*Save Place* FSTN。因此，在恢复路径模式下执行时，它决不能遵循FSTN的反向路径链接指针。
- ① 不要处理siTD或iTD数据结构。只需遵循其“下一个链接指针”。
- ① 请勿处理EPS字段指示高速设备的QH（队列头）。只需遵循其水平链接指针。
- ① 当QH的EPS字段指示全速/低速设备时，只有当其*SplitXState*为**DoComplete**时，主机控制器才会考虑执行该设备（注意：无论PID代码指示IN还是OUT，这都适用）。附加条件的完整列表见第4.10.3节和第4.12.2.3节

通常必须满足主机控制器发出总线事务的要求。请注意，在恢复路径模式下执行时，主机控制器不得执行启动拆分事务。参见第节  
4.12.2.4.2在恢复路径模式下进行特殊处理。

- ② 当恢复路径遇到作为“恢复”指示器的FSTN时，停止遍历恢复路径。当返回到正常路径遍历时，主机控制器无条件地使用保存位置FSTN的正常路径链接指针的保存值。主机控制器在将计划遍历恢复到保存位置FSTN的正常路径链接指针时，必须清除执行恢复路径的上下文。

如果主机控制器确定在微帧中没有足够的时间来完成对周期性调度的处理，则它放弃对恢复路径的遍历，并清除执行恢复路径的上下文。结果是，在下一个连续微帧开始时，主机控制器开始遍历帧列表。

包括FSTN的周期性计划的遍历示例如图4-19所示。

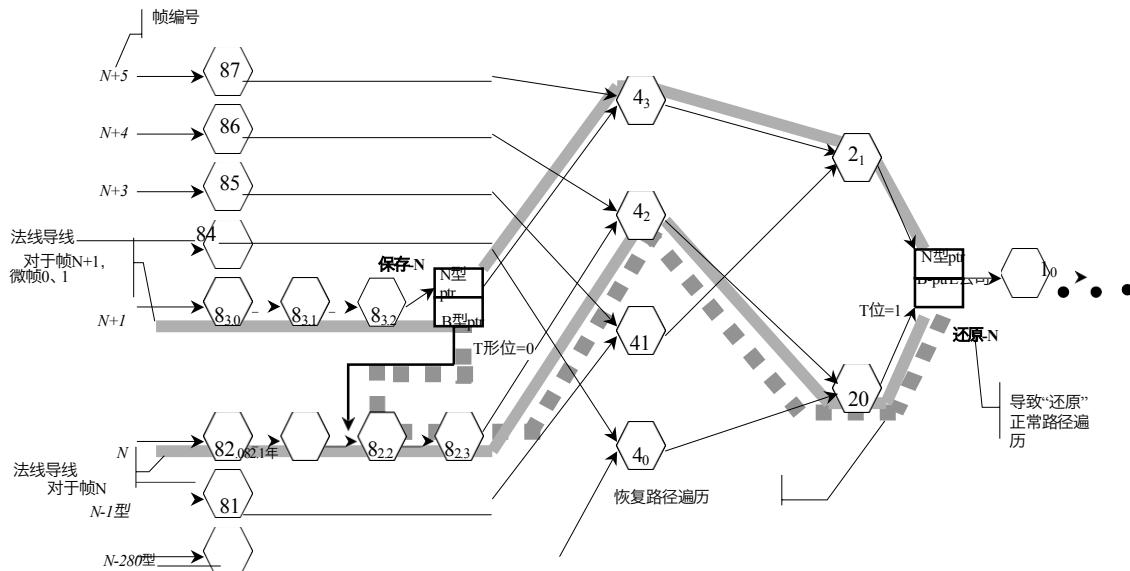


图4-19。示例主机控制器通过FSTN遍历恢复路径

在帧N (微帧0-7) 中，对于该示例，主机控制器将利用其遇到的任何FSTN中的正常路径链路指针遍历所有调度数据结构。这是因为主机控制器尚未遇到保存位置FSTN，因此它未在恢复路径模式下执行。当它在微帧0和1期间遇到Restore-FSTN (Restore-N) 时，它使用Restore-N.Normal Path Link Pointer遍历到下一个数据结构 (即正常调度遍历)。这是因为主机控制器在未以恢复路径模式执行时必须使用恢复FSTN的正常路径链接指针。在帧N期间遍历的节点包括：{82.0, 82.1, 82.2, 82.3, 42, 20, Restore-N, 10...}。

在帧N+1 (微帧0和1) 中，当主机控制器遇到保存路径FSTN (保存-N) 时，它观察到保存-N.Back Path Link Pointer.T比特为零 (保存路径指示符的定义)。主机控制器保存Save-N.Normal Path Link Pointer的值，并遵循Save-N.Back Path Link Pointe。同时，它设置了一个内部标志，指示它现在处于恢复路径模式 (恢复路径在图4-19中用一条大虚线注释)。主机控制器继续遍历恢复路径上的数据结构，并在恢复路径上仅执行如上所述的那些总线事务，直到它到达恢复FSTN (恢复-N)。Restore-N.Back Path Link Pointer.T位设置为1 (恢复指示符的定义)，因此主机控制器通过清除内部恢复路径模式标志退出恢复路径模式，并使用保存位置FSTN的正常路径链接指针 (例如Save-N.Normal Path Link Pointer) 的保存值开始 (恢复) 计划遍历。恢复路径上的节点以粗体显示。

在帧N+1 (微帧2-7) 中, 当主机控制器遇到保存路径FSTN保存-N时, 它将无条件地遵循保存-N正常路径链接指针。在这些微帧期间遍历的节点包括: {<sub>83.0</sub>, <sub>83.1</sub>, <sub>83.2</sub>, Save-A, <sub>43</sub>, <sub>21</sub>, Restore-N, <sub>10</sub>...}。

#### 4.12.2.2 FSTN的软件操作模型

软件必须为主机控制器创建一个一致、连贯的遍历时间表。使用FSTN时, 系统软件必须遵守以下规则:

- ① 每个“保存位置”指示器都需要一个匹配的“恢复”指示器。

保存位置指示器是一个FSTN, 具有有效的反向路径链路指针和等于零的T位。请注意, 必须设置*Back Path Link Pointer.Typ*字段以指示引用的数据结构是队列头。恢复指示器是一个FSTN, 其反向路径链接指针.T位设置为1。

恢复FSTN可以与一个或多个保存位置FSTN相匹配。例如, 如果计划包括轮询率1级别, 则系统软件只需在此列表的开头放置一个*Restore* FSTN, 即可匹配所有可能的*Save place* FSTN。

- ② 如果计划中没有以1的轮询速率级别链接的元素, 并且使用了一个或多个保存位置FSTN, 则系统软件必须确保恢复FSTN的正常路径链接指针的T位设置为1, 因为这将用于标记定期列表的结束。
- ③ 当计划中确实有以1的轮询速率级别链接的元素时, 还原FSTN必须是轮询速率1列表上的第一个数据结构。来自帧列表的所有遍历路径都收敛在轮询速率1列表上。系统软件必须确保在允许主机控制器遍历轮询速率1列表之前退出恢复路径模式。
- ④ 保存位置FSTN的反向路径链接指针必须引用队列头数据结构。引用的队列头必须可以从上一个帧列表位置访问。换句话说, 如果保存位置FSTN可从帧列表偏移量N到达, 则FSTN的反向路径链路指针必须引用可从帧表偏移量N-1到达的队列头。

软件应使时间表尽可能高效。在这种情况下, 这意味着软件在任何单个帧中都不应该有超过一个可访问的保存位置FSTN。请注意, 有时会存在两个 (或更多个, 取决于实现方式), 因为全/低速占地面积会随着带宽调整而变化。例如, 当带宽重新平衡导致系统软件将*Save Place* FSTN从一个轮询速率级别移动到另一个轮询速率级别时, 可能会发生这种情况。在转换过程中, 软件必须保持先前计划的完整性, 直到新计划到位。

#### 4.12.2.3 跟踪中断传输的拆分事务进程

为了正确维护数据流, 主机控制器必须能够检测和报告数据丢失的错误。对于中断IN传输, 数据在进入USB 2.0集线器时会丢失, 但USB 2.0主机系统无法在从事务转换器管道到期之前将数据从USB 2.0集线器获取并进入系统。当检测到丢失的数据情况时, 必须停止队列, 从而向系统软件发出信号, 使其从错误中恢复。当USB 2.0集线器发出、接受并成功执行启动拆分时, 就会出现数据丢失情况, 但完整的拆分在高速链路上出现不可恢复的错误, 或者完整的拆分没有在正确的时间发生。完全拆分可能不会在正确的时间发生的一个原因是由于主机引起的系统延迟, 导致主机控制器无法及时访问系统内存中的时间表而错过总线事务。

中断OUT也可能出现同样的情况, 但结果不是端点停止情况, 而是仅影响传输的进度。

队列头具有以下字段, 用于跟踪每个拆分事务的进度。这些字段用于保持关于哪些 (以及何时) 部分已被执行的增量状态。

- ⑤ C程序掩码。这是一个8比特的矢量, 其中主机控制器跟踪哪些完整的分割已经被执行。由于事务转换器定期管道的性质

需要按顺序执行完全拆分。主机控制器需要检测何时未按顺序执行完整的拆分。这可能是因为主机控制器无法达到基于内存的时间表的系统延迟而导致的。*C-prog-mask*是一个简单的位向量，主机控制器为执行的每个完整分割设置*C-prog-mask*位中的一个。比特位置由执行完全分割的微帧号确定。主机控制器总是在执行完整的拆分事务之前检查*C-prog-mask*。如果之前的完整拆分尚未执行，则意味着跳过了一个（或多个），数据可能会丢失。

- ① 框架标记。此字段由主机控制器在拆分事务的完全拆分部分期间使用，以在必须执行下一个完整拆分时用帧号（*H*-帧号）标记队列头。
- ② *S*字节。此字段可用于存储在开始拆分期间发送的数据有效负载字节数（如果事务是OUT）。*S*字节字段必须用于累积在完整拆分期间接收到的数据有效载荷字节（对于IN）。

#### 4.12.2.4 中断的拆分事务执行状态机

在下面的演示中，所有对微帧的引用都是在*H*帧内的微帧的上下文中。

与异步全速和低速端点一样，分割事务状态机用于管理分割事务序列。除了在队列头中定义的用于调度和跟踪拆分事务的字段之外，主机控制器还计算一个内部机制，该机制也用于管理拆分事务。内部计算机制为：

- ① *cMicroFrameBit*。这是对当前微帧编号的单比特编码。它是主机控制器在每个微帧开始时计算的八位值。它是从*FRINDEX*寄存器的三个最低有效位（即*cMicroFrameBit* = 左移1 (*FRINDEX[2:0]*)）计算得出的。*cMicroFrameBit*最多断言一个比特，该比特始终与当前微帧编号相对应。例如，如果当前微帧为0，则*cMicroFrameBit*将等于0000000 1b。

变量*cMicroFrameBit*用于与*S*掩码和*C*掩码字段进行比较，以确定队列头是否标记为当前微帧的开始或完成split事务。

图4-20说明了用于管理完整中断拆分事务的状态机。每个拆分事务有两个阶段。第一个是单次启动拆分事务，当*SplitXState*处于Do\_start，并且*cMicroFrameBit*中的单个位在*QH.S-mask*中具有相应的活动位时，就会发生该事务。事务转换器不会确认收到周期性启动拆分，因此主机控制器无条件地将状态转换为Do Complete。由于事务转换器管道中的可用抖动，软件将为Do complete状态调度一个以上的完整拆分事务。这简单地转化为在*QH.C*掩码字段中有多个比特被设置为一的事实。

主机控制器保持队列头处于Do Complete状态直到分割事务完成（参见下面的定义），或者错误条件触发三击规则（例如，在主机尝试同一事务三次，并且每次都遇到错误后，主机控制器将停止重试总线事务并停止端点，从而要求系统软件检测该条件并执行依赖于系统的恢复）。

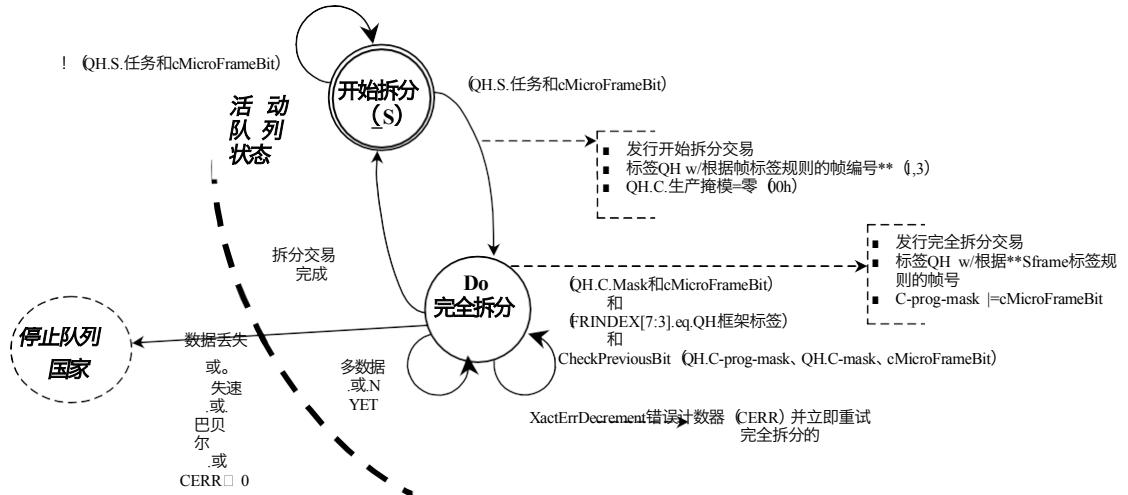


图4-20。用于中断的拆分事务状态机

\*\*帧标签管理规则见第4.12.2.4.3节。

#### 4.12.2.4.1 定期中断-Do Start拆分

这是软件必须初始化一个完整或低速中断队列头StartXState位的状态。只有在分割事务完成后，才能从Do Complete分割状态进入此状态。当以下事件之一发生时，会发生这种情况：

事务转换器使用以下其中一项对完整的拆分事务进行响应：

- ① 纳克。NAK响应是完全或低速端点的NAK响应的传播。
- ② 确认。ACK响应是完全或低速端点的ACK响应的传播。仅发生在OUT端点上。
- ③ 数据0/1。仅适用于IN。指示这是来自此拆分事务终结点的最后一个数据。
- ④ 呃。事务转换器下方低速/全速链路上的事务出现故障（例如超时、CRC错误等）。
- ⑤ 《纽约时报》（最后一期）。主机控制器发出最后一次完全拆分，事务转换器以NYET握手进行响应。这意味着事务转换器没有正确接收到启动拆分，因此它从未执行到完整或低速端点的事务，有关“最后”的定义，请参见第4.12.2.4.2节。

每次主机控制器访问处于该状态的队列头时（在执行事务状态下一次），它都会执行以下测试以确定是否执行启动拆分。

- ⑥ QH.S.掩码采用cMicroFrameBit逐位编码。

如果结果为非零，则主机控制器将发出一个启动拆分事务。如果PIDCode字段指示IN事务，则主机控制器必须将QH.S字节字段清零。在已执行分割事务之后，主机控制器在队列头中设置状态以跟踪分割事务的完整分割阶段的进度。具体来说，它将期望的帧数记录到QH.FrameTag字段中（见第4.12.2.4.3节），将C-prog-mask设置为零（00h），并退出该状态。请注意，主机控制器不得因启动拆分事务的完成而调整CErr的值。

#### 4.12.2.4.2 定期中断-完成拆分

在总线上执行启动拆分事务后，从**Do Start Split**状态无条件进入该状态。每次主机控制器访问处于该状态的队列头时（在**执行事务**状态下一次），它都会进行检查以确定现在是否应该执行完整的拆分事务。

有四个测试来确定是否应该执行完整的拆分事务。

- ① 测试A.*cMicroFrameBit*是逐位的，并带有*QH掩码*字段。非零结果表示软件计划在此微帧期间为此端点进行完全拆分。
- ② 测试B.将*QH.FrameTag*与*FRINDEX*的当前内容进行比较[7:3]。相等表示匹配。
- ③ 测试C 检查完整分割进度位向量，以确定是否设置了前一位，指示前一完整分割已适当执行。  
下面提供了该测试的示例算法：

```
算法布尔函数 previousBit (QH.C-prog-mask, QH.C-mask, cMicroFrameBit)
开始
    --返回值:
    --TRUE-无错误
    --FALSE-错误
    --
布尔值右值=真;
previousBit=cMicroframeBit逻辑向右旋转 ()  

--位式前置位带C掩码的位表示是否有意图
--以在先前的微帧中发送完整的分割。所以，如果
--“前一位”设置在C掩码中，检查C程序掩码以确保
--发生了。
如果 (前一位与QH.C掩码), 则
    如果不是 (previousBit位AND QH.C-prog-mask), 则
        rvalue=FALSE;
    结束如果结
束如果
--如果C-prog-mask在这个位置已经有一个1, 那么一个混叠
--出现错误。它可能会被FrameTag测试捕获, 但是
--无论如何, 这是一种错误情况, 正如在这里可以检测到的那样, 不应该允许
--要执行的事务。
如果 (cMicroFrameBit位AND QH.C-prog-mask), 则
    rvalue=FALSE;
如果结束
返回 (右值)
结束算法
```

- ④ 测试D.检查是否应在此微帧中执行启动拆分。请注意，这与在**Do Start Split (执行启动拆分)**状态下执行的测试相同（请参见第4.12.2.4.1节）。每当评估结果为TRUE且控制器未在*Recovery Path*（恢复路径）模式下进行处理时，这意味着启动拆分应在该微帧中发生。测试D和测试A同时评估为TRUE是系统软件错误。行为未定义。

如果 (A.和.B以及.C.和.not (D))，则主机控制器将执行一个完整的拆分事务。当主机控制器承诺执行完整的拆分事务时，它通过与*cMicroFrameBit*的位“或”来更新*QH.C-prog-mask*。完成完整分割事务后，主机控制器将事务结果记录在队列头中，并将*QH.FrameTag*设置为预期的*H*帧编号（见第4.12.2.4.3节）。对队列头状态的影响以及传输状态的影响取决于事务转换器对完整分割事务的响应。以下回复具有

效果 (注意, 如果递减的结果为零, 则导致 $CErr$ 递减的任何响应都将导致队列头被主机控制器暂停) :

- ① 《纽约时报》(最后一期)。对于每个NYET响应, 主机控制器进行检查以确定这是否是该拆分事务的最后一次完整拆分。在此上下文中, Last被定义为已执行所有计划的完全拆分的条件。如果这是最后一次完全拆分(带有NYET响应), 则队列头的传输状态不会提前(从未接收到任何数据), 并且此状态已退出。事务转换器必须已经用NYET响应了所有clompete拆分, 这意味着没有收到主机控制器发出的开始拆分。启动拆分应在下一个轮询周期重试。

这是否是最后一次完全拆分的测试可以通过将 $QH.C$ 掩码与 $QH.C-prog-mask$ 异或来执行。如果结果全为零, 则所有完全拆分都已执行。当出现这种情况时,  $XactErr$ 状态位设置为1, 并且 $CErr$ 字段递减。

- ② NYET (而非Last)。有关Last的测试, 请参见上面的描述。完整的拆分事务接收到来自事务转换器的NYET响应。不要更新任何传输状态( $C-prog-mask$ 和 $FrameTag$ 除外)并保持此状态。主机控制器不得对此响应调整 $CErr$ 。
- ③ 事务错误(XactErr)。超时、数据CRC失败等。 $CErr$ 字段递减,  $Status$ 字段中的 $XactErr$ 位设置为1。将立即重试完整的拆分事务(如果 $Cerr$ 为非零)。如果微帧中没有足够的时间来完成重试, 并且端点是in, 或者 $Cerr$ 从1减为0, 则队列将停止。这将导致在下一个轮询周期重试整个OUT拆分事务。有关为什么必须立即重试这些错误的详细要求, 请参阅USB规范修订版2.0中的第11章集线器(特别是全速和低速中断部分)。
- ④ 确认。只有当目标端点是OUT时, 才会发生这种情况。目标端点对数据进行了ACK, 而此响应是端点ACK向主机控制器的传播。主机控制器必须提前传输的状态。“当前偏移”字段按“最大数据包长度”或“要传输的字节数”递增, 以较小者为准。字段“要传输的字节”按相同的数据量递减。并且数据切换位( $dt$ )被切换。然后, 主机控制器将退出此队列头的此状态。主机控制器必须重新加载具有此响应的最大值的 $Cerr$ 。

推进转移状态可能会导致其他过程事件, 如qTD的退出和队列的推进(见第4.10节)。

- ⑤ 多数据。此响应将仅发生在IN端点上。事务转换器以零个或多个字节的数据和MDATA PID作为响应。接收到的字节增量以 $QH.S$ 字节为单位累积。主机控制器不得对此响应调整 $CErr$ 。
- ⑥ DATA0/1。此响应可能仅发生在IN端点上。接收到的字节数与累积的字节计数相加(以 $QH.S$ 字节为单位)。传输状态根据结果而提前, 主机控制器将退出此队列头的此状态。

推进传输状态可能会导致其他处理事件, 如qTD的退出和队列的推进(见第4.10节)。

如果数据序列PID与预期不匹配, 则会忽略此拆分事务中接收到的全部数据, 传输状态不会提前, 并退出此状态。

- ⑦ 纳克。目标端点Nak'd完全或低速事务。传输的状态未进入, 并且已退出此状态。主机控制器必须重新加载具有此响应的最大值的 $Cerr$ 。
- ⑧ 呃。在全速或低速事务处理过程中出现错误。ERR状态位设置为1,  $Cerr$ 递减, 传输状态不前进, 并退出此状态。

- ① 失速。队列被暂停（**执行事务状态的退出条件**）。状态字段位：活动位设置为零，暂停位设置为1，qTD失效。

未在列表中枚举的响应或无序接收的响应是非法的，并且可能导致未定义的主机控制器行为。

测试A、B、C和D的其他可能组合可能表明数据或响应丢失。表4-13列出了可能的组合和适当的措施。

**表4-13。中断输入/输出完成拆分状态执行标准**

条件	行动	描述
不是 (A) 不是 (D)	忽略QHD	对于当前微帧，既不安排开始分割，也不安排完全分割。主机控制器应继续执行时间表。
A. 不是 (C)	如果PIDCode=IN 停止QHD  如果PIDCode=OUT 重试开始拆分	进度位检查失败。这意味着错过了一次完整的拆分。存在数据丢失的可能性。如果PIDCode是IN，则必须停止队列头。  如果PIDCode为OUT，则传输状态不前进，状态退出（例如，重试启动拆分）。这是一个由主机引起的错误，不会影响CERR。  在任何一种情况下，将状态字段中的 <i>Missed Micro</i> 帧位设置为1。
A. 不是 (B) C	如果PIDCode=IN 停止QHD  如果PIDCode=OUT 重试开始拆分	QH.FrameTag测试失败。这意味着恰好跳过了一个或多个H帧。这意味着完全拆分并且已错过。存在数据丢失的可能性。如果PIDCode是IN，则必须停止队列头。  如果PIDCode为OUT，则传输状态不前进，状态退出（例如，重试启动拆分）。这是一个由主机引起的错误，不会影响CERR。  在任何一种情况下，将状态字段中的 <i>Missed Micro</i> 帧位设置为1。
A B C 不是 (D)	执行完全拆分	这是主机控制器执行完整拆分事务的非错误情况。
D	如果PIDCode=IN 停止QHD  如果PIDCode=OUT 重试开始拆分	这是一种退化的情况，其中发出了开始拆分，但跳过了所有完整的拆分，并且检测丢失数据的所有可能的干预机会都未能激发。如果PIDCode是IN，则必须停止队列头。  如果PIDCode为OUT，则传输状态不前进，状态退出（例如，重试启动拆分）。这是一个由主机引起的错误，不会影响CERR。  在任何一种情况下，将状态字段中的 <i>Missed Micro</i> 帧位设置为1。  注意：当在恢复路径模式的上下文中执行时，允许主机控制器处理队列头并采取上述操作，或者它可以等到在正常处理模式中访问队列头。无论如何，主机控制器不得在以恢复路径模式执行的上下文中执行启动拆分。

#### 4.12.2.4.3 管理QH.FrameTag字段

队列头中的*QH.FrameTag*字段完全由主机控制器管理。设置规则  
*QH.FrameTag*很简单：

- ① 规则1：如果从**Do Start Split**转换为**Do Complete Split**，并且*FRINDEX[2:0]*的当前值为6。  
*QH.FrameTag*设置为*FRINDEX[7:3]+1*。这适用于起始拆分和完全拆分位于不同*H*帧中的拆分事务（情况2a，见图4-17）。
- ② 规则2：如果*FRINDEX[2:0]*的当前值为7，则*QH.FrameTag*设置为*FRINDEX[7:3]+1*。这适用于在情况2a、2b和2c中停留在**Do Complete Split**（图4-17）。
- ③ 规则3：如果从**Do\_Start Split**转换为**Do Complete Split**并且*FRINDEX[2:0]*的当前值不是6，或者当前处于**Do Complete Split**并且*(FRINDEX[%2:0])*的当前值不为7，则*FrameTag*设置为*FRINDEX[7:3]*。这适用于所有其他情况（图4-17）。

#### 4.12.2.5 重新平衡定期计划

在操作过程中，系统软件必须偶尔调整周期性队列头的*s掩码*和*C掩码*字段。当对周期性调度的调整产生新的带宽预算并且一个或多个队列头被分配新的执行足迹（即，新的*s掩码*和*C掩码*值）时，就会出现这种需要。至关重要的是，系统软件不得在拆分交易过程中将这些掩码更新为新值。为了避免更新时出现任何竞争情况，EHCI主机控制器为系统软件提供了简单的帮助。

系统软件将下一个事务(*I*)上的非激活位设置为1，以向主机控制器发出信号，表明它打算更新该队列头上的*S掩码*和*C掩码*。然后，系统软件将等待主机控制器观察到*I*位为1，并将活动位转换为0。主机控制器如何以及何时将活动位设置为零的规则列举如下：

- ① 如果活动位为零，则不采取任何操作。当*I*位为1时，主机控制器不尝试推进队列。
- ② 如果活动位为1，而*SplitXState*为**DoStart**（与*S掩码*的值无关），则主机控制器将简单地将活动位设置为零。主机控制器不需要将传输状态写回到当前qTD。请注意，如果*S掩码*指示为当前微帧安排了启动拆分，则主机控制器不得发出启动拆分总线事务。它必须将活动位设置为零。

在将*I*位设置为1之前，系统软件必须保存传输状态。这是必需的，以便它能够正确地确定在*I*位被设置为1并且主机控制器执行其最终总线事务并将*Active*设置为0之后发生了什么传输进度（如果有的话）。

系统软件更新*S掩码*和*C掩码*后，必须重新激活队列头。由于活动位和*I*位不能用相同的写入来更新，系统软件需要使用以下算法来一致地重新激活已经通过*I*位停止的队列头。

1. 将*Halted*位设置为1，然后
2. 将*I*位设置为零，然后
3. 在同一写入中，将活动位设置为1，将暂停位设置为0。

将*Halted*位设置为1禁止主机控制器在*I*位变为0和*Active*位变为1之间尝试推进队列。

### 4.12.3 拆分交易等时

全速等时传输通过USB2.0集线器中的USB2.0事务转换器使用拆分事务协议进行管理。EHCI控制器利用siTD数据结构来支持等时分割事务的特殊要求。该数据结构使用等时TDs的调度模型（TD，第3.3节）（TD的操作模型见第4.7节）以及由队列头提供的连续数据特征。这种简单的安排允许单个等时调度模型，并增加了从端点接收的所有数据（每个拆分事务）必须进入连续缓冲区的附加功能。

#### 4.12.3.1 等时间的分割事务调度机制

全速等时事务通过事务转换器的周期性管道进行管理。与全速和低速中断一样，系统软件通过预算和调度精确地管理每个事务转换器的周期性管道，在微帧期间，每个全速等时端点的开始分割和完全分割都会发生。第4.12.2.1节中所述的要求适用。图4-21说明了EHCI周期性调度支持的一般调度边界条件。

**SX**和**CX**标签表示软件可以（分别）安排开始和完成拆分的微帧。**H**型架边界用一条粗粗的大竖线标记。**B**帧边界用一条粗粗虚线标记。图的底部说明了siTD与**H**帧的关系。

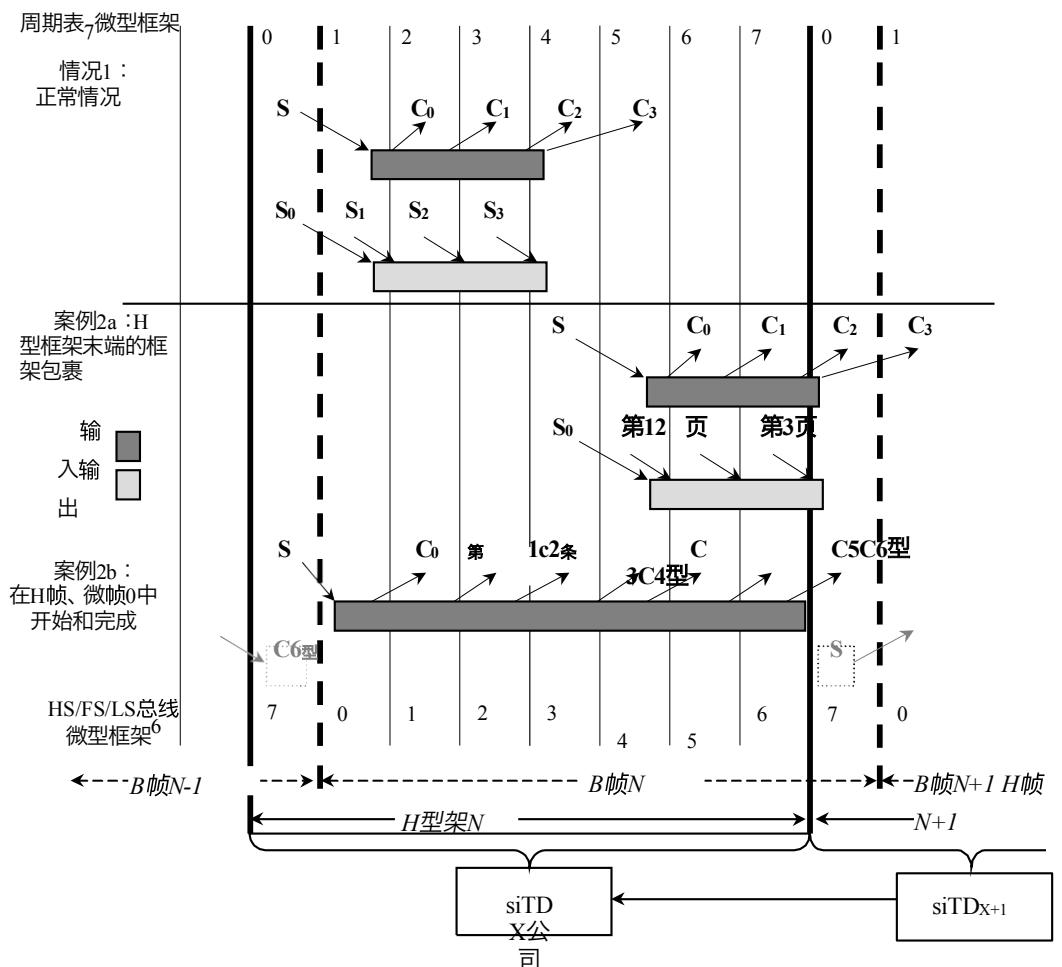


图4-21. 分事务、等时调度的边界条件

当端点是等时OUT时，只有开始拆分，没有完全拆分。当端点是等时IN时，最多有一个开始拆分和一个完全拆分。调度边界情况包括：

- ① 情况1 :整个拆分交易完全受H帧约束。例如 :开始拆分和完成拆分都计划在同一H帧中发生。
- ② 情况2a :这种边界情况是在H帧边界上调度分割事务IN的一个或多个 (最多两个) 完整分割。只有当分割事务有可能在B帧、微帧6或7 (H帧微帧7或0) 中移动数据时, 才会发生这种情况。当出现H帧边界环绕条件时, 分割事务的调度跨越周期性列表中的多个位置。 (例如, 需要相邻周期性帧列表位置中的两个siTD来完全描述分割事务的调度)。

尽管分割事务的调度可以采用两个数据结构, 但每个全速IN等时事务的所有完整分割必须仅使用一个数据指针。出于这个原因, siTD包含一个返回指针, 其使用将在下面描述。

软件决不能在H帧边界上调度全速等时输出。

- ③ 情况2b :这种情况只能发生在非常大的等时IN中。这是唯一允许在同一微帧中发生同一端点的开始分割和完全分割的情况。软件必须通过首先安排大型事务来强制执行此规则。Large被定义为大于579字节最大数据包大小的任何值。

全中断和低速中断队列头所采用的相同机制的子集被用于siTD中, 以调度和跟踪等时分割事务的部分。以下字段由系统软件初始化, 以指示主机控制器何时执行分割事务协议的部分。

- ④ 拆分X状态。这是驻留在siTD的状态字段中的单个位 (见表3-11)。此位用于跟踪拆分事务的当前状态。该钻头的管理规则见第4.12.3.3节。
- ⑤ □帧S掩码。这是一个位字段, 在该位字段中, 系统软件设置与主机控制器应执行开始分割事务的微帧 (在H帧内) 相对应的位。这总是由SplitXState位的值限定的。例如, 参考图4-21中的IN示例, 情况一, S掩码的值为0000000 1b, 表明如果主机控制器遍历siTD, 并且SplitXState指示Do Start Split, 并且FRINDEX[2:0]指示的当前微帧为0, 则执行启动拆分事务。
- ⑥ □框架C掩码。这是一个位字段, 其中系统软件设置一个或多个对应于主机控制器应执行完整拆分事务的微帧 (在H帧内) 的位。该字段的解释始终由SplitXState位的值限定。例如, 参考图4-21中的IN示例, 情况一, C掩码的值为00111100b, 表示如果主机控制器遍历siTD, 并且SplitXState指示Do Complete Split, 并且FRINDEX[2:0]指示的当前微帧为2、3、4或5, 则执行完整的拆分事务。
- ⑦ 后退指针。siTD中的该字段用于使用先前的H-Frame的siTD来完成in分割事务。仅当完整拆分的调度跨越H-Frame边界时才使用此选项。

高速等时分割事务 (包括所有开始和完全分割) 和全速等时事务之间存在一对一的关系。siTD包含 (除其他外) 缓冲区状态和拆分事务调度信息。siTD的缓冲状态总是映射到一个全速等时数据有效载荷。这意味着, 对于任何全速事务有效负载, 都必须使用单个siTD的数据缓冲区。此规则适用于IN和OUT。siTD的调度信息通常也映射到一个高速等时分割事务。该规则的例外是上面提到的H帧边界包裹情况。

siTD数据结构最多描述一帧的高速事务, 并且该描述严格限制在帧边界内。图4-22举例说明了一些例子。顶部是用于上述边界调度情况的全速事务占用的示例。中间是B帧 (HS/FS/LS总线) 和H帧的时间帧参考。底部说明了siTD描述的范围和时间参考之间的关系。每个H型架

对应于周期性帧列表中的单个位置。这意味着每个siTD每次都可以从单个周期性帧列表位置到达。

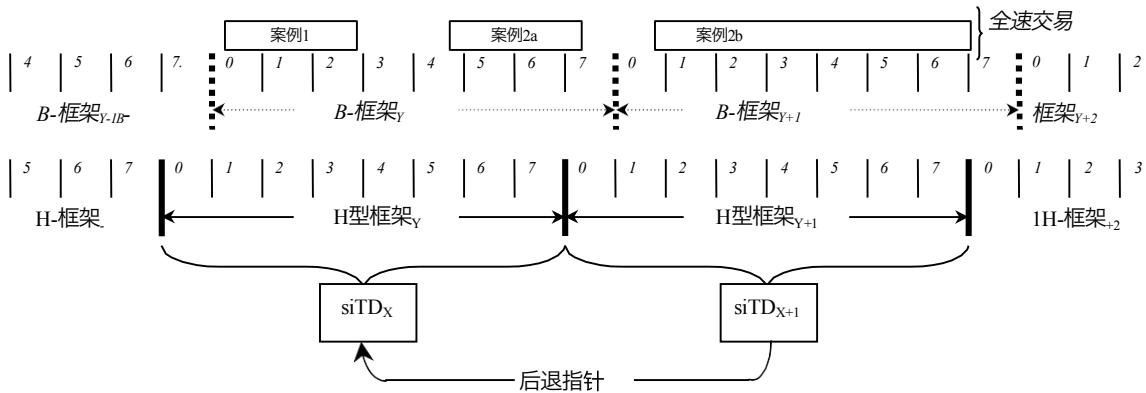


图4-22。siTD调度边界示例

每个案例描述如下：

- ① 情况1：一个siTD足以描述和完成等时分割事务，因为整个等时分割交易被紧密地包含在单个H帧内。
- ② 情况2a、2b：尽管IN和OUT都可以有这些足迹，但OUT总是只需要一个siTD来安排。然而，IN（对于这些边界情况）需要两个siTD来完成等时分割事务的调度。siTD<sub>X</sub>用于始终发布开始拆分和前N个完全拆分。全速事务（对于这些情况）可以在H-Frame<sub>Y+1</sub>的微帧7或H-Frame<sub>Y+2</sub>的微帧0期间在全速总线段上传送数据。使用<sub>siTD<sub>X+1</sub></sub>+2（未显示）安排完整的拆分。提取此数据的完整拆分必须使用siTD<sub>X+1</sub>中的缓冲区指针。主机控制器从H-Frame<sub>Y+1</sub>到达siTD<sub>X+1</sub>的唯一方法是使用siTD<sub>X+2</sub>的反向指针。第4.12.3.3.2.1节介绍了何时使用反向指针的主机控制器规则。

在计算进度表并将进度表数据结构链接到定期进度表时，软件必须应用以下规则：

- ① 软件必须确保启动同步拆分事务，以便在B帧结束之前完成。
- ② 软件必须确保对于单个全速等时端点，在H帧/微帧1中永远不会出现开始分割和完全分割。这是强制性的规则，以便可以区分情况2a和情况2b。根据核心USB规范，可以调度情况2b中所示的长等时事务，使得开始分割在H帧N的微帧1中，并且最后的完全分割将需要发生在H帧N+1的微帧中。然而，不可能区分情况2a和情况2b，这对主机控制器的复杂性有重大影响。

### 4.12.3.2 跟踪同步转账的拆分交易进度

为了正确维护数据流，主机控制器必须能够检测和报告设备到主机数据丢失的错误。等时端点不采用错误停止的概念，但要求主机识别并报告在数据流中观察到的每个数据包的错误。这包括调度遍历问题（跳过的微帧）、超时和接收到的损坏数据。

与中断分割事务类似，分割事务协议的部分必须在它们被调度的微帧中执行。用于管理全中断和低速中断的队列头数据结构有几种机制用于跟踪事务部分何时发生。等时传输使用siTD进行传输，并且数据结构只能通过

精确的微帧，其中需要它们（因此SiTD不需要用于跟踪队列头的所有机制）。软件可以在完整的周期计划中多次重复使用SiTD。然而，它必须确保在主机控制器返回SiTD之前（在未来的微帧中）消耗分割事务N的结果并且重新初始化（激活）SiTD。

分割事务等时输出利用低级协议来指示分割事务数据的哪些部分已经到达。通过字段事务位置（TP）和事务计数（T-计数）在SiTD中暴露对低级别协议的控制。如果OUT分割事务的整个数据有效载荷大于188字节，则将有一个以上的开始分割事务，每个事务都需要适当的注释。如果发生主机延迟，那么从主机接收的注释序列将不完整，这将由事务转换器检测和处理。请参阅第4.12.3.3.1节，了解在一系列开始拆分事务期间如何使用这些字段的说明。

字段 $SiTD.T-Count$ 和 $SiTD.TP$ 由主机控制器用来驱动事务位置注释并对其进行排序。系统软件有责任在每个SiTD中正确初始化这些字段。一旦建立了拆分事务等时端点的预算，与该端点相关联的所有SiTD的S掩码、T计数和TP初始化值都是恒定的。它们保持不变，直到软件重新计算端点的预算并调整定期计划。

对于IN端点，事务翻译器简单地用足够的信息来注释响应数据包，以允许主机控制器识别最后的数据。与拆分事务中断一样，主机控制器有责任检测何时错过了执行完整拆分的机会。SiTD中的以下字段用于跟踪和检测IN等时端点的拆分事务执行中的错误。

- ① C程序掩码。这是一个8比特的矢量，其中主机控制器跟踪哪些完整的分割已经被执行。由于事务转换器周期管道的性质，需要按顺序执行完整的拆分。主机控制器需要检测何时未按顺序执行完整的拆分。这可能是因为主机控制器无法达到基于内存的时间表的系统延迟而导致的。  
*C-prog-mask*是一个简单的位向量，主机控制器为执行的每个完整分割设置一个位。比特位置由执行完全分割的微帧（FRINDEX[2:0]）号决定。主机控制器总是在执行完整的拆分事务之前检查*C-prog-mask*。如果之前的完整拆分尚未执行，则意味着跳过了一个（或多个），数据可能会丢失。在将SiTD的活动位设置为1之前，系统软件需要将该字段初始化为零。

如果事务转换器在执行完所有完整拆分之前返回了最终数据，则传输状态将提前，从而不执行剩余的完整拆分。请参阅第节

4.12.3.3.2关于如何推进转移状态的说明。需要注意的是，IN SiTD仅根据事务转换器对完整拆分事务的响应而失效。这意味着，例如，事务转换器可以使用MDATA PID来响应完全拆分。MDATA的数据有效负载中的字节数可能会导致SiTD字段“要传输的总字节数”减少到零。在执行所有计划的完整拆分之前，可能会出现此响应。在其他接口、数据结构（例如，通过队列头的高速数据流）中，总字节数到传输的转换为零表示传输结束，并导致活动位设置为零。但是，在这种情况下，事务转换器尚未传递结果，主机必须继续进行下一个完整的拆分事务，以提取剩余的事务状态。出现这种情况是因为事务转换器的管道规则（见通用串行总线修订版2.0的第11章）。总之，周期性管道规则要求在微帧边界上，事务转换器将在全速总线管道级中保持接收到的最后两个字节（如果它还没有看到分组结束（EOP）），并且将剩余的字节给予高速管道级。在微帧边界，事务转换器可能已经接收到整个分组（包括两个CRC字节），但没有接收到分组EOP。在下一个微帧中，事务转换器将用MDATA进行响应，并发送所有数据字节（其中两个CRC字节保存在全速流水线阶段）。这可能会导致SiTD将其要传输的总字节数字段减为零，表明它已接收到所有预期数据。主机仍必须执行一个（计划的）完整拆分事务，以便从事务转换器中提取全速事务的结果（例如，事务转换器可能检测到CRC故障，并且必须将此结果转发给主机）。

如果主机经历了导致主机控制器跳过一个或多个（但不是所有）调度的用于等时OUT的分割事务的延迟，则到事务转换器的协议将不一致，并且事务转换器将检测到该问题并对其作出反应。同样，对于导致主机控制器跳过用于等时IN的一个或多个（但不是全部）调度的分割事务的主机延迟，主机控制器使用C-prog-mask来检测错误。然而，如果主机经历了导致其跳过所有siTD的延迟，或者siTD在主机延迟期间到期（例如，发生延迟并且主机控制器不再能够到达siTD以便其报告延迟事件），则系统软件必须检测到siTD尚未被主机控制器处理（例如状态未被高级），并向客户端驱动器报告适当的错误。

#### 4.12.3.3 用于Isochronous的拆分事务执行状态机

在下面的演示中，所有对微帧的引用都是在H帧内的微帧的上下文中。

如果状态字节中的活动位为零，则主机控制器将忽略siTD并继续遍历周期性调度。否则，主机控制器将按照以下规定处理siTD。

拆分事务状态机用于管理拆分事务协议序列。主机控制器使用第4.12.3.2节中定义的字段，加上第4.12.3.2中定义的变量cMicroFrameBit和4.12.2.4跟踪同步分割交易的进展。

图4-23说明了通过等时分割事务管理siTD的状态机。粗体虚线圆圈表示siTD的状态字段中活动位的状态。粗体虚线弧表示这些状态之间的转换。实心圆表示拆分事务状态机的状态，实心弧表示这些状态之间的转换。点弧和方框引用了由于转换或处于状态而发生操作。

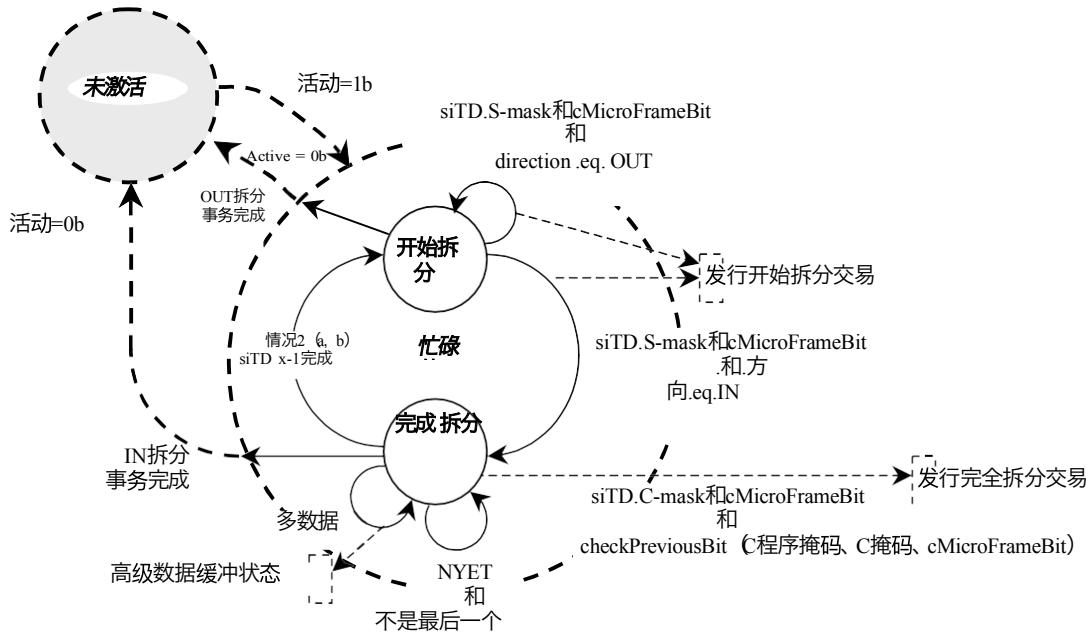


图4-23。用于Isochronous的拆分事务状态机

##### 4.12.3.3.1 周期性等时-Do Start Split

等时分割事务OUT仅使用此状态。分割事务等时IN的siTD被初始化到该状态，或者当情况2a (IN) 或2b调度边界等时分割事务完成时，siTD从Do Complete split转换到该状态。

每次主机控制器在这种状态下达到活动siTD时，它都会对照*cMicroFrameBit*检查siTD.S-mask。如果在适当的位置有一个，siTD将执行一个开始拆分事务。根据定义，主机控制器不能在错误的时间达到siTD。

如果I/O字段指示IN，则开始拆分事务仅包括扩展令牌加上全速令牌。软件必须将*siTD.Total Bytes To Transfer*字段初始化为预期的字节数。这通常是全速端点的最大数据包大小。当启动拆分事务完成时，主机控制器将退出此状态。

本节的其余部分特定于等时OUT端点（即i/O字段指示OUT）。当主机控制器执行用于等时OUT的开始分割事务时，它在开始分割事务中包括数据有效载荷。数据有效载荷的内存缓冲区地址是通过将*siTD.Current Offset*与页面选择器字段（*siTD.P*）所指示的页面指针连接而构建的。此字段中的0选择页面0，1选择页面1。在OUT的开始分割期间，如果数据传输在事务处理期间越过页面边界，则主机控制器必须检测到页面交叉，将*siTD.P-bit*从零更新为一，并开始使用*siTD.page 1*和*siTD.Current Offset*作为内存地址指针。

字段*siTD.TP*用于用当前有效载荷表示拆分事务数据的哪一部分的指示（ALL、BEGIN、MID、END）来注释每个开始拆分事务。在所有情况下，主机控制器只需使用*siTD.TP*中的值来用正确的事务位置代码标记开始拆分。

*T-Count*始终初始化为当前帧的起始分割数。*TP*总是被初始化为第一个所需的事务位置标识符。调度边界情况（见图4-22）用于确定*TP*的初始值。表4-14总结了最初的案例。

表4-14。OUT siTD的TP和T计数字段的初始条件

案例	T-计数	TP公司	描述
1. 2a	=1	全部	当OUT数据有效载荷小于（或等于）188字节时，仅需要一个开始分割来移动数据。单起点拆分必须标记为ALL。
1. 2a	!=1	开始	当OUT数据有效载荷大于188字节时，必须使用一个以上的起始分割来移动数据。初始起始拆分必须用BEGIN标记。

在每个启动拆分事务完成后，主机控制器适当地更新*T-Count*和*TP*，以便正确地注释下一个启动拆分。表4-15说明了所有*TP*和*T-Count*转换，这些转换必须由主机控制器完成。

表4-15。交易头寸（TP）/交易计数（T-Count）转换表

TP 公 司	T计数下 一个	TP公 司 下一个	描述
全部	0	N/A	从全部过渡到完成。
开始	1.	完	从开始到结束的转换。当T计数从2开始时发生。
开始	!=1	中间	从BEGIN到MID的转换。当T计数开始大于2时发生。
中间	!=1	中间	<i>TP</i> 保持在MID，而T计数不等于1（例如大于1）。这种情况可以发生在T计数开始大于3的任何调度边界情况下。
中间	1.	完	从MID到END的转换。这种情况可以发生在T计数开始大于2的任何调度边界情况下。

启动拆分事务不接收来自事务转换器的握手，因此主机控制器总是在总线事务完成后推进siTD中的传输状态。为了推进传输状态，将执行以下操作：

① 将调整 *siTD.Total Bytes To Transfer* (要传输的总字节数) 和 *siTD.Current Offset* (当前偏移量) 字段, 以反映传输的字节数。

② *siTD.P* (页面选择器) 比特被适当地更新。

③ *siTD.TP* 和 *siTD.T-count* 字段按照表4-15中的定义进行适当更新。然后, 这些字段被写回到基于存储器的*siTD*中。

*S掩码*在当前预算的有效期内是固定的。如上所述, 在每个*siTD*中专门设置 *TP* 和 *T-计数*, 以反映要从该*siTD*发送的数据。因此, 无论*S掩码*的值如何, 启动拆分事务的实际数量都取决于 *T-计数* (或等效地, 要传输的总字节数)。当主机控制器检测到所有调度数据都已发送到总线时, 它必须将活动位设置为零。优选的方法是检测 *T-Count*何时由于启动拆分总线事务而减少到零。等效地, 主机控制器可以检测要传输的总字节数何时减少到零。任何一种实现都必须确保, 如果初始条件是要传输的总字节数等于零, 并且 *T-计数*等于1, 则主机控制器将发出具有零长度数据有效载荷的单次启动拆分。软件必须确保 *TP*、*T-计数*和要传输的总字节数设置为从每个*siTD*传递适当数量的总线事务。不一致的组合将产生未定义的行为。

如果主机遇到延迟, 导致主机控制器跳过OUT传输的启动拆分事务, 则传输状态将无法正常进行。事务翻译器将在OUT端点的开始拆分到达时观察到协议冲突 (即, 事务翻译器接收到的事务位置注释将不正确)。

示例场景如第4.12.3.4节所述。

主机控制器实现可以选择性地通过在执行每个调度的开始分割时在 *siTD.C-prog-mask* 中设置适当的位来跟踪OUT分割事务的进度。在执行每个开始拆分之前, 可以使用第4.12.3.3.2节中定义的 *checkPreviousBit 0* 算法来确定是否跳过了开始拆分。主机控制器可以使用该机制来检测遗漏的微帧。然后, 它可以将*siTD*的活动位设置为零并停止执行该*siTD*。这节省了内存和高速总线带宽。

#### 4.12.3.3.2 周期等时-做完全拆分

此状态仅由拆分事务等时IN端点使用。在为IN端点执行启动拆分事务后, 从**Do Start state**无条件地进入此状态。每次主机控制器访问处于这种状态的*siTD*时, 它都会进行大量测试, 以确定是否应该执行完整的拆分事务。下面列出了各个测试。它们被应用的顺序取决于主机控制器当前正在执行的微帧, 这意味着测试可能直到从返回指针引用的*siTD*被提取之后才被应用。

① 测试A.*cMicroFrameBit*是逐位的, 并带有 *siTD.C-mask* 字段。非零结果表示软件计划在此微帧期间为此端点进行完全拆分。此测试始终应用于处于此状态的新获取的*siTD*。

② 测试B.检查 *siTD.C-prog-mask* 位向量, 以确定是否已执行先前的完全拆分。下面是一个示例算法 (这与第4.12.2.4.2节中使用的算法略有不同)。应用此测试的顺序取决于FRINDEX的当前值 [2:0]。如果FRINDEX[2:0]为0或1, 则在使用回指针之前不会应用它。否则会立即应用。

**算法布尔校验previousBit ( siTD.C-prog-mask、siTD.C-mask、cMicroFrameBit) 开始**  
布尔值右值=真；

```

previousBit=cMicroFrameBit向右旋转 (1)

--位式前置位带C掩码的位表示是否有意图
--以在先前的微帧中发送完整的分割。所以，如果
--“前一位”设置在C掩码中，检查C程序掩码以确保
--发生了。

如果上一个Bit bit AND siTD.C掩码，则
    如果不是(前一位位AND siTD.C-prog-mask)，则rvalue=FALSE
        结束如果结
    束如果

    返回右值
结束算法
```

如果测试A为真，且FRINDEX[2:0]为零或一，则这是情况2a或2b的调度边界（见图4-21）。有关处理这种情况的详细信息，请参见第4.12.3.3.2.1节。

如果测试A和测试B评估为真，则主机控制器将使用当前siTD的传输状态执行完整的拆分事务。当主机控制器承诺执行完整的拆分事务时，它通过与cMicroFrameBit的位“或”来更新QH.C-prog-mask。转移状态是根据完整拆分事务的完成状态进行的。要推进IN siTD的传输状态，主机控制器必须：

- ① 减少从siTD接收的字节数。要传输的总字节数，
- ② 根据接收到的字节数调整siTD.Current Offset，
- ③ 如果传输导致主机控制器使用下一页指针，则调整siTD.P（页面选择器）字段，以及
- ④ 根据事务的结果，在siTD.Status字段中设置任何适当的位。

请注意，如果主机控制器遇到siTD.Total Bytes To Transfer为零的情况，并且接收到更多数据，则主机控制器不得将额外数据写入内存。siTD.Status.Active位必须设置为零，siTD.Status.Babble Detected位必须设为1。此事务尝试不需要更新字段siTD.Total Bytes To Transfer、siTD.Current Offset和siTD.P（页面选择器）。

主机控制器必须接受（假设良好的数据包CRC和缓冲区中有足够的空间，如siTD.Total Bytes To Transfer的值所示）MDATA和DATA0/1数据有效载荷，最大可达192字节（包括192字节）。当主机控制器实现接收到数据有效载荷大于192字节的MDATA或DATA0/1时，它可以选择性地将siTD.Status.Active设置为零，将siTD.Status.Babble Detected设置为1。

以下回应具有显著效果：

- ① 呃。全速事务在超时或CRC错误的情况下完成，这是该错误对主机的反映。主机控制器在siTD.Status字段中设置ERR位，并将Active位设置为零。
- ② 事务错误(XactErr)。完整的拆分事务遇到超时、CRC16故障等。siTD.Status字段XactErr字段设置为1，必须立即重试完整的拆分交易。主机控制器必须使用内部错误计数器来计数重试次数，因为siTD数据结构中没有提供计数器字段。主机控制器重试次数不会超过两次。如果主机控制器用尽重试次数或微帧结束，则活动位设置为零。
- ③ DATAx (0或1)。该响应表示分割事务的最终数据已经到达。siTD的传输状态被提前，并且活动位被设置为零。如果要传输的字节字段

没有递减到零（包括DATAx响应中的数据有效载荷的接收），则实际接收到的数据少于预期或允许的数据。此短数据包事件不会将USBSTS寄存器中的USBINT状态位设置为1。主机控制器将不会检测到这种情况。

- ① 《纽约时报》（最后一期）。对于每个NYET响应，主机控制器还进行检查以确定这是否是该拆分事务的最后一次完整拆分。最后一个定义见第4.12.2.4.2节。如果它是最后一次完全拆分（具有NYET响应），则siTD的传输状态不前进（从未接收到任何数据），并且活动位被设置为零。“状态”字段中没有设置任何位，因为这本质上是一个跳过的事务。事务转换器必须已经用NYET响应了所有计划的clompete拆分，这意味着没有收到主机控制器发出的开始拆分。系统软件应将此结果解释为事务被完全跳过。

这是否是最后一次完全拆分的测试可以通过将C掩码与C程序掩码异或来执行。结果为零表示所有完整的拆分都已执行。

- ② MDATA（最后一个）。有关Last的测试，请参见上面的描述。只有在出现错误的情况下才会发生这种情况。要么是全速链路上出现了延迟全速事务完成的胡言乱语情况，要么是软件设置的S掩码和/或C掩码不正确。主机控制器必须将XactErr位设置为1，将Active位设置为0。
- ③ NYET（而非Last）。有关Last的测试，请参见上面的描述。完整的拆分事务接收到来自事务转换器的NYET响应。不要更新任何传输状态（C-prog-mask除外）并保持此状态。
- ④ MDATA（而不是最后一个）。当事务转换器具有拆分事务的部分数据时，它将使用MDATA进行响应。例如，全速事务数据有效载荷从微帧X跨越到X+1，并且在微帧X期间，事务转换器将用MDATA和累积到微帧X结束的数据进行响应。主机控制器推进传输状态以反映接收到的字节数。

如果测试A成功，但测试B失败，则意味着跳过了一个或多个完整的拆分。主机控制器设置Missed Micro Frame状态位并将Active位设置为零。

#### 4.12.3.3.2.1 调度边界情况2a、2b的完全拆分

边界情况2a和2b（仅限IN）（见图4-21）要求主机控制器使用前一个siTD的事务状态上下文来完成拆分事务。表4-16列举了事务状态字段。

**表4-16。siTD拆分事务状态摘要**

缓冲区状态	地位	执行进度
要传输的总字节数P（选择页面） 当前偏移 TP（交易头寸） T计数（交易计数）	状态字段中的所有位	C-程序-任务

注：TP和T计数仅用于主机到设备（OUT）端点。

如果软件已为此数据流的计划编制了帧换行的预算，则它必须初始化siTD.Back Pointer字段以引用有效的siTD，并将siTD.Back-Pointer字段中的siTD.Bback Pointer.T-bit设置为零。否则，软件必须将siTD.Back-Point字段中的siTD.Back Pointer.T位设置为1。下面列出了主机控制器用于解释何时使用siTD.Back Pointer字段的规则。这些规则仅适用于siTD的活动位为1并且SplitXState为Do Complete Split的情况。

- ① 当cMicroFrameBit为1h且siTDX.Back Pointer.T为零时，或
- ② 如果cMicroFrameBit是2h并且siTDX.S-掩码[0]是零

当这些条件中的任何一个适用时，主机控制器必须使用来自*siTDX-1*的事务状态。

为了访问*siTDX-1*，主机控制器在芯片上读取从*siTDX.Back*指针引用的*siTD*。在处理*siTDX-1*时，主机控制器必须从*siTDX*保存整个状态。这是为了适应情况2b的处理。主机控制器不能递归遍历*siTD.Back*指针的列表。

如果*siTDX-1*是活动的（活动位是1，*SplitXStat*是**Do Complete Split**），则如上所述应用测试a和测试B。如果满足执行完全分割的这些标准，则主机控制器执行完全分割并如上所述评估结果。*siTDX-1*的事务状态（见表4-16）根据结果进行适当的高级处理并写回内存。如果*siTDX-1*的活动位的结果状态为1，则主机控制器返回到*siTDX*的上下文，并跟随其下一指针到下一调度项目。无需更新*siTDX*。

如果*siTDX-1*处于活动状态（活动位为1，*SplitXStat*为**Do Start Split**），则主机控制器必须将活动位设置为0，将*Missed Micro Frame*状态位设置为1，并将结果状态写回内存。

如果*siTDX-1*的活动位为零（因为当主机控制器第一次通过*siTDX*的返回指针访问*siTDX-2*时它为零，所以由于检测到错误，它转换为零，或者*siTDX.3*的完整拆分事务的结果将其转换为零），则主机控制器返回*siTDX*上下文并将其*SplitXState*转换为**Do Start split**。主机控制器然后确定情况2b的开始分割边界条件是否存在（即，如果*cMicroframeBit*是1b并且*siTDX.s-mask[0]*是1b）。如果满足该标准，则主机控制器立即执行开始拆分事务，并适当地推进*siTDX*的事务状态，然后跟随*siTDX.Next*指向下一个计划项目的指针。如果不满足该标准，主机控制器只需遵循*siTDX.Next*指向下一个计划项目的指针。注意，在2b边界情况下，当主机控制器返回到*siTDX*的上下文时，*siTDX-1*的分割事务将其活动位设置为零。此外，请注意，软件不应初始化C掩码位0和1设置为1的*siTD*，以及位0设置为一的S掩码。不支持此调度组合，并且未定义主机控制器的行为。

#### 4.12.3.4 同步拆分 事务-处理示例硬件/软件管理同步拆分事务状态机的方式与管

理异步和中断拆分事务状态的方式之间存在重要差异

机器。异步和中断分离事务状态机封装在一个

队列头。数据流的进度取决于每个拆分事务的进度。在某些方面，分割事务状态机是通过**执行事务队列头遍历状态机排序**的（见图4-14）。

Isochronous是一种纯粹面向时间的事务/数据流。接口数据结构被优化以有效地描述需要在特定时间发生的事务。等时分割事务状态机必须在这些面向时间的数据结构中进行管理。这意味着系统软件必须正确地描述跨多个数据结构的拆分事务的调度。然后，主机控制器必须在适当的时间以正确的数据结构进行适当的状态转换。

例如，表4-17说明了调度情况2a全速等时数据流所需的几帧调度。

表4-17。示例案例2a-IN端点的软件调度siTD

siTD <sub>X</sub>		微型框架							最初的拆分X状态	
#	口罩	0	1.	2.	3.	4.	5.	6.	7.	
十、	S-掩码					1.				开始拆分
	C型面罩	1.	1.					1.	1.	
X+1	S-掩码					1.				完成拆分
	C型面罩	1.	1.					1.	1.	
X+2	S-掩码					1.				完成拆分
	C型面罩	1.	1.					1.	1.	
X+3	S-掩码	重复以前的模式								完成拆分
	C型面罩									

此示例显示了事务流的前三个siTD。由于这是情况2a的帧包裹情况，因此用于该端点的所有siTD的S掩码具有10h的值（微帧4中的一个比特）和C3h的C掩码值（在微帧0、1、6和7中的一比特）。此外，软件确保每个siTD的“返回指针”字段引用适当的siTD数据结构（并且“返回指针T”位设置为零）。

第一个siTD的初始SplitXState是**Do Start Split**。主机控制器将在帧X期间访问第一个siTD八次。微帧0和1中的C掩码位被忽略，因为状态为**Do Start Split**。在微帧4期间，主机控制器确定它可以运行开始拆分（并执行），并将SplitXState更改为**Do Complete split**。在微帧6和7期间，主机控制器执行完全的分割。请注意，帧X+1的siTD将其SplitXState初始化为**Do Complete Split**。当主机控制器在H帧X+1期间继续遍历调度时，它将访问第二个siTD八次。在微帧0和1期间，它将检测到必须执行完全拆分。

在H帧X+1，微帧0期间，主机控制器检测到<sub>siTDX+1</sub>的Back Pointer.T位为零，保存<sub>siTDX+1</sub>的状态并获取<sub>siTDX</sub>。它使用<sub>siTDX</sub>的事务状态执行完整的拆分事务。如果<sub>siTDX</sub>拆分事务完成，则siTD的活动位被设置为零，结果被写回<sub>siTDX</sub>。主机控制器保留<sub>siTDX</sub>已失效的事实，并将<sub>siTDX+1</sub>中的SplitXState转换为**Do Start Split**。在这一点上，主机控制器准备在<sub>siTDX+1</sub>到达微帧4时对其进行开始分割。如果拆分事务提前完成（事务完成定义见第4.12.3.3.2节），即在执行所有计划的完全拆分之前，主机控制器将<sub>siTDX</sub>.SplitXState转换为**Do Start split early**，并自然跳过剩余的计划的完整拆分事务。对于该示例，<sub>siTDX+1</sub>直到H帧X+2，微帧1才接收DATA0响应。

在H帧X+2，微帧0期间，主机控制器检测到<sub>siTDX+2</sub>的Back Pointer.T位为零，保存<sub>siTDX+2</sub>的状态并获取<sub>siTDX+1</sub>。如上所述，它执行另一个拆分事务，接收MDATA响应，更新传输状态，但不修改活动位。主机控制器返回到<sub>siTDX+2</sub>的上下文，并遍历其下一个指针，而不对<sub>siTDX+2</sub>进行任何状态更改更新。S

在H帧X+2，微帧1期间，主机控制器检测到<sub>siTDX+2</sub>的s掩码[0]为零，保存<sub>siTDX+2</sub>的状态并获取<sub>siTDX+1</sub>。它执行另一个完整的拆分事务，接收DATA0响应，更新传输状态并将活动位设置为零。它返回到<sub>siTDX+2</sub>的状态，并将其SplitXState更改为**Do Start Split**。在这一点上，主机控制器准备在<sub>siTDX+2</sub>到达微帧4时对其进行开始分割。

<TBD...描述软件如何检测到缺少微帧（不要认为我们关心缺少微帧。有足够的剩余状态来识别，而不是执行所有事务）。

## 4.13 主机控制器暂停

当主机控制器在USBSTS寄存器中的*HCHalted*位为零时，主机控制器将向所有启用的端口发送SOF（开始帧）数据包。启用时间表后，EHCI主机控制器将访问每个微帧主存储器中的时间表。众所周知，主存储器的这种持续ping会给移动系统带来CPU电源管理问题。具体来说，移动系统会根据最近的历史使用情况积极管理CPU的状态。在更积极的节能模式下，CPU可以禁用其缓存。当前的PC体系结构假定总线主控器对主存储器的访问必须是高速缓存一致的。因此，当总线主控器忙于触摸内存时，CPU电源管理软件可以随着时间的推移检测到这种活动，并禁止CPU转换到其最低节能模式。USB控制器是总线主控器，它们访问基于内存的时间表的频率使CPU电源管理软件无法将CPU置于其最低节能状态。

USB主机控制器在挂起时不会访问主存储器。然而，将USB控制器置于挂起状态不起作用的原因有很多，但它们超出了本文档的范围。基本要求是USB控制器需要保持在主存储器之外，同时防止USB总线进入挂起状态。

EHCI控制器提供了可由系统软件操纵的大粒度机制，以改变主机控制器的存储器访问模式。系统软件可以操作USBCMD寄存器中的调度使能位来开启/关闭调度遍历。此方法并非旨在始终应用于限制USB，而是应仅应用于非常特定的配置和使用负载。例如，当只有键盘或鼠标连接到USB时，启发式方法可以检测USB仅很少尝试移动数据的时间，并且可以调整占空比，以允许CPU在更长的时间内达到其低功耗状态。类似地，它可以检测USB负载的增加，并适当地调整占空比，甚至可以达到从未禁用时间表的程度。这里的假设是USB正在移动数据，并且需要CPU来处理数据流。

有人建议，为了为系统提供完整的解决方案，伴随主机控制器还应该提供类似的方法，以允许系统软件禁止伴随主机控制器访问其基于共享存储器的数据结构（时间表列表或其他）。

## 4.14 端口测试模式

EHCI主机控制器必须实现USB规范修订版2.0中所述的端口测试模式*test J\_State*、*test K\_State*、*test Packet*、*test Force\_Enable*和*test SE0\_NAK*。系统只允许测试EHCI控制器拥有的端口（例如，*CF*位为1，*PortOwner*位为0）。系统软件一次最多允许有一个端口处于测试模式。将多个端口置于测试模式将产生未定义的结果。所需的每个端口测试序列为（假设CONFIGFLAG寄存器中的*CF*位为1）：

- ① 通过设置异步计划启用和定期调度USBCMD寄存器中的启用位为零。
- ② 通过将每个相应PORTSC寄存器中的*Suspend*位设置为1，将所有启用的根端口置于*Suspend*状态。
- ③ 将USBCMD寄存器中的运行/停止位设置为零，并等待USBSTS寄存器中的*HCHalted*位转换为1。请注意，EHCI主机控制器实现可以选择性地允许将运行/停止位设置为1的端口测试。但是，所有主机控制器都必须支持端口测试，*Run/Stop*设置为零，*HCHalted*设置为一。
- ④ 将测试端口PORTSC寄存器中的端口测试控制字段设置为所需测试模式对应的值。如果所选测试为*test Force\_Enable*，则USBCMD寄存器中的运行/停止位必须转换回1，以便能够将SOF传输出测试端口。

- ① 测试完成后，系统软件必须确保主机控制器停止 (*HCHalted*位为1)，然后通过将*HCReset*设置为1来终止并退出测试模式。

## 4.15 中断

EHCI主机控制器硬件提供基于多个源的中断能力。中断源有几个一般组：

- ① 由于根据计划执行事务而中断（成功和错误条件），
- ② 主机控制器事件（端口更改事件等），以及
- ③ 主机控制器错误事件

所有基于事务的源都可以通过主机控制器的中断启用寄存器（*USBINTR*，见表2-11）屏蔽。此外，可以标记各个传输描述符以在完成时生成中断。本节介绍每个中断源以及响应中断而发生的处理。

在正常操作期间，中断可以是立即的或延迟的，直到出现下一个中断阈值。中断阈值是通过*USBCMD*寄存器中的*中断阈值控制*字段可调的参数。此寄存器的值控制主机控制器何时代表正常事务执行生成中断。当事务在中断间隔时间段内完成时，直到中断阈值发生时，才发生用信号通知传输完成的中断。例如，默认值为八个微帧。这意味着主机控制器不会比每八个微帧一次更频繁地产生中断。

第4.15.2.4节详细说明了主机系统错误的影响。

如果已计划为当前中断阈值间隔生成中断，则直到将该间隔中最后一个完整事务的状态写回主机存储器之后，才发出中断信号。这有时可能导致直到下一个中断阈值才用信号通知中断。

无论中断的原因是什么，初始中断处理都是相同的。当硬件发出中断信号时，CPU控制被转移到主机控制器的USB中断处理程序。实现传输的精确机制是特定于操作系统的。对于这个讨论，仅仅假设接收到控制。当中断处理程序接收到控制时，它的第一个动作是读取*USBSTS*（USB状态寄存器）。然后，它通过向这些位位置写入1来清除所有中断状态位，从而确认中断。然后，处理程序确定中断是由于调度处理还是其他事件造成的。在确认中断后，处理程序（通过操作系统特定的机制）调度稍后执行的延迟过程调用（*DPC*）。*DPC*例程处理调度执行的结果。所使用的确切机制超出了本文件的范围。

注：当软件将中断使能（在*USBINR*寄存器中，请参阅第2.3.3节）设置为零时，主机控制器不需要取消断言当前活动的中断条件。软件用于确认中断的唯一可靠方法是将*USBSTS*寄存器（第2.3.2节）中的适当状态位从1转换为0。

### 4.15.1 基于传输/事务的中断

这些中断源与传输和事务进程相关。它们都取决于下一个中断阈值。

#### 4.15.1.1 事务处理错误

事务错误是指导致主机控制器认为传输未成功完成的任何错误。表4-18列出了主机可以观察到的作为事务结果的事件/响应。以下段落总结了错误计数器和中断状态的影响。这些错误中的大多数在适当的接口数据结构中设置了*XactErr*状态位。

有一小部分协议错误仅在执行队列头时相关，并且属于错误PID错误的保护伞下，这些错误对明确识别非常重要。当出现这些错误时，将设置队列头中的 $XactErr$ 状态位，并递减 $Cerr$ 字段。当 $PIDCode$ 指示SETUP时，以下响应是协议错误，并导致 $XactErr$ 位被设置为1，并且 $Cerr$ 字段被递减。

- ① EPS字段指示高速设备，并且它向SETUP返回Nak握手。
- ② EPS字段指示高速设备，并且它向SETUP返回Nyet握手。
- ③ EPS字段指示低速或全速设备，并且完全分离接收Nak握手。

表4-18交易错误汇总

事件/结果	队列头/qTD/iTD/siTD副作用		USB状态寄存器(USBSTS)
	欧洲可再生能源委员会	状态字段	
CRC公司	-1个	$XactErr$ 设置为1。	1 <sup>1</sup>
超时	-1个	$XactErr$ 设置为1。	1 <sup>1</sup>
错误PID2	-1个	$XactErr$ 设置为1。	1 <sup>1</sup>
含糊不清地说	N/A	第4.15.1.1.1节	1.
缓冲区错误	N/A	第4.15.1.1.2节	

<sup>1</sup> 如果发生在队列头中，则只有当 $Cerr$ 从1倒计时到0时，才会断言 $USBERRINT$ 。此外，队列被暂停，请参见第4.10.3.1节。

<sup>2</sup> 主机控制器收到来自设备的响应，但无法将PID识别为有效PID。

#### 4.15.1.1 串行总线巴贝尔

当一个设备在USB上传输的数据超过主机控制器对该事务的预期时，它被定义为喋喋不休。一般来说，这被称为“数据包胡言乱语”。当设备发送的数据超过最大长度字节数时，主机控制器将*Babble Detected*位设置为1，并在使用队列头时停止端点（见第4.10.3.1节）。最大长度定义为要传输的总字节数和最大数据包大小的最小值。 $Cerr$ 字段不会针对数据包牙牙学语的情况而递减（仅适用于队列头）。如果IN事务在高速EOF2点进行，则也存在牙牙学语条件。这被称为框架胡言乱语。在适当的调度数据结构中记录一个帧串音条件。此外，主机控制器必须禁用检测到帧间歇的端口。

USBSTS寄存器中的 $USBERRINT$ 位设置为1，如果USBINTR寄存器中的 $USB$ 错误中断启用位为1，则在下一个中断阈值向系统发送硬件中断信号。主机控制器决不能启动将在微帧EOF上喋喋不休的OUT事务。

**注：**当主机控制器检测到数据PID不匹配时，它必须：在总线事务的持续时间内禁用数据包间歇检查，或者仅根据最大数据包大小进行数据包间歇检测。USB核心规范定义了当数据接收器接收到数据PID不匹配时对其的要求（例如，期望DATA0并获得DATA1，反之亦然）。总之，为了推进发射机的数据序列，它必须忽略接收到的数据并用ACK握手进行响应。

EHCI接口允许系统软件为控制、批量或中断IN端点提供缓冲区，这些缓冲区不是设备指定的最大数据包大小的偶数倍。每当设备错过IN端点的ACK时，主机和设备就数据传输的进度而言不同步。主机控制器可能已经将传输推进到小于最大分组大小的缓冲器。该设备将重新发送其具有原始数据PID的最大数据包大小的数据包，以响应下一个in令牌。为了正确管理总线协议，当主机控制器观察到数据PID不匹配时，它必须禁用数据包间歇检查。

#### 4.15.1.1.2 数据缓冲区错误

此事件表示此事务的传入数据溢出或传出数据不足。这通常是由主机控制器不能在必要的等待时间要求内访问存储器中所需的数据缓冲区造成的。这些条件不被视为事务错误，并且不会影响队列头中的错误计数。当这些错误确实发生时，主机控制器通过设置队列头中的数据缓冲区错误位*iTD*或*siTD*来记录发生错误的事实。

如果数据缓冲区错误发生在非同步IN上，主机控制器将不会向端点发出握手。这将强制端点重新发送相同的数据（和数据切换），以响应到端点的下一个in。

如果OUT上发生数据缓冲区错误，主机控制器必须损坏数据包的末尾，这样设备就无法将其解释为良好的数据包。简单地截断数据包是不可接受的。一个可接受的实现选项是1对CRC字节进行补码并发送它们。USB规范2.0版的“事务转换器”部分还建议了其他选项。

#### 4.15.1.2 USB中断（完成中断（OC））

传输描述符（*TD*、*siTD*和队列头（*qTD*））包含一个位，该位可以设置为在其完成时引起中断。与该调度项目相关联的传输的完成导致USBSTS寄存器中的USB中断（USBINT）位被设置为1。此外，如果在与队列头关联的In事务上遇到短数据包，则此事件也会导致USBINT设置为1。如果USBINTR寄存器中的USB中断启用位设置为1，则在下一个中断阈值向系统发出硬件中断信号。如果完成是因为错误，则USBSTS寄存器中的USBERRINT位也被设置为1。

#### 4.15.1.3 短数据包

在控制、批量或中断传输期间接收到小于端点的最大数据包大小的数据包，表示传输完成。每当在队列头执行期间发生短数据包完成时，USBSTS寄存器中的USBINT位被设置为1。如果在USBINTR寄存器中设置了USB中断启用位，则在下一个中断阈值向系统发送硬件中断信号。

### 4.15.2 主机控制器事件中断

这些中断源独立于中断阈值（异步推进中断除外，见第4.15.2.3节）。

#### 4.15.2.1 端口更改事件

端口寄存器包含状态和状态更改位。当状态更改位被设置为1时，主机控制器将USBSTS寄存器中的端口更改检测位设置为1。如果USBINTR寄存器中的端口更改中断启用位为1，则主机控制器将发出硬件中断。端口状态更改位包括：

- ① 连接状态更改
- ① 端口启用/禁用更改
- ① 过电流变化
- ① 强制端口恢复

### 4.15.2.2 帧列表滚动

此事件表示主机控制器已包装帧列表。帧列表的当前编程大小影响该中断发生的频率。如果帧列表大小是1024，则中断将每1024毫秒发生一次，如果是512，则中断每512毫秒发生一次等。

当检测到帧列表翻转时，主机控制器将USBSTS寄存器中的帧列表翻转位设置为1。如果USBINTR寄存器中的帧列表翻转启用位设置为1，则主机控制器发出硬件中断。此中断不会延迟到下一个中断阈值。

### 4.15.2.3 异步前进中断

此事件用于从异步计划中确定删除队列头。每当主机控制器推进异步调度的片上上下文时，它都会评估USBCMD寄存器中异步推进门铃上的中断位的值。如果是1，则将USBSTS寄存器中的“异步前进中断”位设置为1。如果USBINTR寄存器中的Interrupt on Async Advance Enable位为1，则主机控制器在下一个中断阈值发出硬件中断。第4.8.2节对该特性进行了详细说明。

### 4.15.2.4 主机系统错误

主机控制器是总线主控器，并且主机控制器和系统之间的任何交互都可能经历错误。主机错误的类型对主机控制器来说可能是灾难性的（例如主机中止），使得主机控制器不可能以一致的方式继续。在存在非灾难性主机错误（例如奇偶校验错误）的情况下，主机控制器可能会继续运行。对于这些类型的错误，建议的行为是将其升级为灾难性错误并停止主机控制器。基于主机的错误必须导致以下操作：

- ① USBCMD寄存器中的运行/停止位设置为零。
- ② USBSTS寄存器中的以下位已设置：
  - ① 主机系统错误位为1。
  - ② HCHalted位设置为1。
- ③ 如果USBINTR寄存器中的主机系统错误启用位为1，则主机控制器将发出硬件中断。此中断不会延迟到下一个中断阈值。

表4-19总结了针对各种主机错误所需采取的措施。

**表4-19。EHCI主机控制器对主机系统错误的总结行为**

循环类型	主机中止	目标夭折	数据相位奇偶校验
帧列表指针获取（读取）	致命的	致命的	致命[0]
siTD获取（读取）	致命的	致命的	致命[0]
siTD状态回写（写）	致命[0]	致命[0]	致命[0]
iTD获取（读取）	致命的	致命的	致命[0]
iTD状态回写（写入）	致命[0]	致命[0]	致命[0]
qTD获取（读取）	致命的	致命的	致命[0]
qHD状态回写（写）	致命[0]	致命[0]	致命[0]
数据写入	致命[0]	致命[0]	致命[0]
读取的数据	致命的	致命的	致命[0]

[0] 主机控制器实现可能会在不停止的情况下继续运行。但是，建议的行为是停止主机控制器。

注：发生主机系统错误后，软件必须通过USBCMD寄存器中的*HCReset*重置主机控制器，然后才能重新初始化和重新启动主机控制器。

页面故意留空

## 5. EHCl扩展功能

EHCl控制器使用与PCI功能类似的方法导出EHCl特定的扩展功能。如果EHCl控制器实现任何扩展功能，它会在HCCRAMS寄存器的EHCl扩展功能指针(EECP)字段中指定一个非零值。该值是第一个扩展能力寄存器所在的PCI配置空间的偏移量。每个能力寄存器的格式如表5-1所示

表5-1。EHCl扩展能力指针寄存器的格式

一点	描述
31分16秒	特定能力。这些比特的定义和属性取决于特定的能力。
15时8分	下一个EHCl扩展功能指针□ RO。此字段指向下一个EHCl扩展能力指针的PCI配置空间偏移。值00h表示扩展能力列表的结束。
7:0分	能力ID□ RO。此字段标识EHCl扩展功能。有关有效的EHCl扩展功能的列表，请参见表5-2。

“下一个”字段可能包含一个非零值。在这种情况下，它指定下一个EHCl扩展功能所在的PCI配置空间的偏移量。功能链接列表中的最后一个EHCl扩展功能的链接字段必须设置为零。

表5-2。EHCl扩展能力代码

身份证件	名称	描述
00小时	保留	此值为保留值，不得使用。
01小时	操作系统到操作系统切换前同步	操作系统切换同步支持功能。详见第5.1节。

### 5.1 EHCl扩展功能：操作系统切换前同步

系统配置可以包括BIOS(在此也称为Pre-OS软件)中对EHCl控制器的控制的支持。OS切换同步功能提供了允许BIOS启用对EHCl事件的SMI支持的机制，以及用于实现信号量以同步EHCl控制器的所有权更改的一组寄存器。交接机制必须是干净和准确的，每个参与者都必须遵守下面定义的协议。如果不这样做，将导致两个软件代理认为他们各自拥有EHCl的独家所有权，并试图同时使用控制器。注意，这些寄存器并不旨在支持伴随控制器的所有权交换。

操作系统切换同步EHCl扩展功能包括PCI配置空间中的两个连续的32位寄存器。第一个寄存器是USB操作系统切换同步EHCl扩展能力寄存器(USBLEGSUP)，字段定义见第2.1.7节。此寄存器是一个标准的EHCl扩展能力指针，包括一个EHCl扩展功能ID字段和到下一个EHCl-扩展功能的链接。

该寄存器的高16位包含所有权限信号量。一个信号量用于操作系统(OS)，一个用于BIOS。这些信号量可读写。这些字段位于相邻的字节中，这允许每个代理(OS或BIOS)在不覆盖其他所有权限信号量的情况下更新各自的信号量。

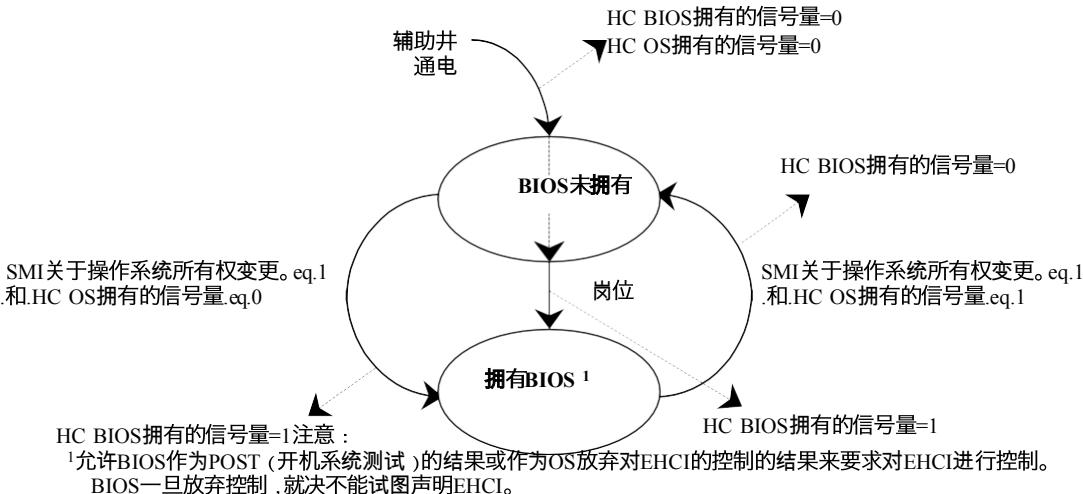
第二个32位寄存器是USB操作系统切换同步控制/状态寄存器(USBLEGCTLTS)，字段定义见第2.1.8节。被遮蔽的特定USBSTS寄存器位表示可以

检测到并使能以产生中断。USBLEGCTLTS寄存器为BIOS提供了将所有EHCI事件、所有必要的重新配置事件和操作系统所有权请求映射到SMI的机制。

下面是两个状态机，它们说明BIOS和OS必须遵守的正确协议（例如，对所有权信号的更新），以便一致地请求和/或放弃EHCI控制器的所有权。这些图中使用的约定是：

- ① 实心弧表示导致状态变化的单个或多个事件。
- ② 带箭头的虚线表示发生的副作用。当连接到实心弧时，解释为该事件会产生副作用。

图5-1说明了BIOS所有权的协议状态机。操作系统切换同步寄存器位于辅助井中，因此，当辅助井电源恢复时，任何从辅助井断电的系统事件都将导致这些寄存器重置为默认值。

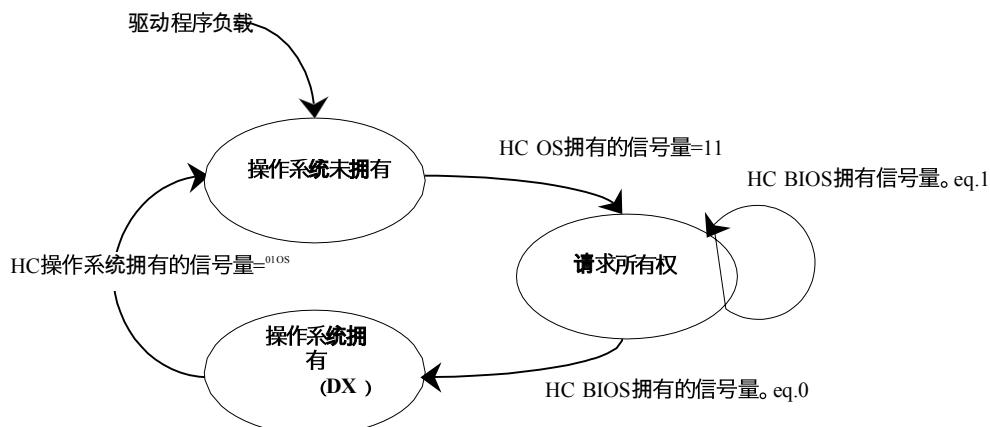


**图5-1。BIOS所有权状态机**

当辅助电源井通电时，USBLEGSUP中的BIOS所有和OS所有信号量将变为其默认值（例如零）。BIOS可以通过将BIOS拥有的信号量设置为1来获得EHCI控制器的所有权。只有当OS拥有的位为零时，BIOS才被允许拥有EHCI控制器的所有权。BIOS然后可以配置它所需要的SMI事件，包括操作系统所有权变更上的SMI。BIOS现在拥有EHCI控制器，因此它可以配置控制器、枚举总线并根据需要使用找到的设备。

最终，操作系统将加载。如果操作系统支持EHCI控制器，则需要对EHCI控制器进行独占控制。操作系统驱动程序必须使用图5-2中定义的协议来请求主机控制器的所有权，然后才能获得所有权并使用控制器。操作系统驱动程序通过将操作系统拥有的信号量设置为1来启动所有权请求。操作系统在尝试使用EHCI控制器之前等待BIOS拥有的位变为零。操作系统必须等待BIOS响应所有权请求的时间超出了本规范的范围。请注意，没有定义类似的SMI类型的事件，允许BIOS向操作系统请求所有权。

如果BIOS已将USBLEGCTLTS寄存器中的操作系统所有权更改上的SMI设置为1，则当操作系统驱动程序将操作系统拥有的信号量设置为1时，它将接收SMI（如上）。BIOS通过通知BIOS它打算放弃对EHCI的控制，观察到操作系统已将操作系统拥有的位的值更改为零。



注意事项：

当USBLEGCTLTS中的SMI on OS Ownership Enable位设置为1时，对OS Owned信号量的修改会导致SMI。

图5-2。操作系统所有权状态机

如果操作系统驱动程序卸载和/或想要放弃EHCI控制器的所有权，则必须将操作系统拥有的信号量设置为零。同样，如果BIOS已将USBLEGCTLTS寄存器中的OS Ownership Enable上的SMI设置为1，则当OS Driver将OS Owned信号量设置为0时，它将接收SMI。BIOS观察到OS已经放弃控制，然后可以适当地接管EHCI控制器的控制。一旦系统软件放弃了对控制器的控制，它就必须如上所述请求所有权。

注意，该机制仅旨在确保主机控制器的所有权交换可以以非常确定和可靠的方式完成。

页面故意留空

## 附录A. EHCI PCI电源管理接口

符合PCI总线电源管理接口规范修订版1.1的高级电源管理功能接口被纳入EHCI。该接口允许EHCI处于各种电源管理状态，为主机系统提供各种节能。

表A-3重点介绍了EHCI对电源管理状态的支持以及每个电源管理状态支持的功能。EHCI实现可以在内部关闭USB时钟并挂起USB收发器（低功耗模式）以提供这些功率节省。每个EHCI供应商用于实现所需行为的方法是特定于实现的。EHCI将根据PCI总线电源管理接口规范修订版1.1和本规范中定义的规则断言PME#并保留芯片上下文。

在退出D0状态之前，控制器软件驱动程序必须将EHCI的所有启用的下游USB端口置于USB挂起状态。这是为了确保所有下游设备处于非活动低功率模式。

表A-3。EHCI对电源管理状态的支持

PCI电源管理状态	按规范要求/可选的状态	评论
D0	必需	完全唤醒向后兼容状态。全功率模式下的所有逻辑。
第1页	可选择的	禁用EHCI总线主机功能的USB睡眠状态。所有USB端口处于挂起状态。由于低延迟返回D0状态，所有逻辑处于低延迟节能模式。
D2	可选择的	禁用EHCI总线主机功能的USB睡眠状态。所有USB端口处于挂起状态。
D3热	必需	深度USB睡眠状态，禁用EHCI总线主机功能。所有USB端口处于挂起状态。
D3冷	必需	完全睡眠向后兼容状态。基于在功率预算内提供下游端口功率的实施能力，所有下游设备要么被暂停，要么被断开。

## A.1 PCI电源管理寄存器接口

EHCI实现遵循PCI电源管理规范修订版1.1中指定的PCI电源管理寄存器接口。EHCI实施的具体要求和说明如下：

- ① 当处于任何支持的设备状态时，主机控制器必须能够断言PME#。但是，如果主机控制器支持无法从D3cold断言PME#的系统（即辅助电源不足或不存在），则D3cold的“PME\_Support”位（PMC寄存器的第15位）必须是可修改的。主板下设备可以使用软件（BIOS）方案来修改在该只读位中报告的值，而其他设备可以使用引脚绑定来确定报告的值。
- ② EHCI报告的Aux\_Current或Data Register值应表示主机控制器设备将消耗的最大电流。它不得包括连接到下游USB2端口的设备所消耗的功率。请注意，如果主机控制器已配置为不从D3cold生成PME#，则Aux\_Current字段或Data Register（D3功耗，D3功耗）必须报告“000”。

所有其他寄存器和字段应遵循PCI PM规范。

### A.1.1 电源状态转换

当施加Vcc并且发生硬件或软件复位时，EHCI从D3冷功率状态进入D0功率状态。软件重置不应影响PCI电源管理寄存器。硬件复位可以是PCI复位输入或可选的通电复位输入。

电源管理软件通过EHCI拥有的PCI电源管理寄存器访问将EHCI转换为D0、D1、D2和D3热电源状态。可以实现额外的功率管理策略，在Vcc被移除时将辅助电源VAUX切换或连续地应用于EHCI。在这种被称为D3cold的电源状态下，EHCI表现出与D3hot电源状态相同的行为（除了不支持配置空间访问），并且不需要额外的EHCI硬件来区分D3hot和D3cold。

根据PCI电源管理规范，EHCI函数在D3hot到D0转换期间断言内部重置。当从D3hot转换到D0时，主机控制器必须保留所有相关的唤醒上下文，以便系统软件处理唤醒请求。在PCI配置空间中，这意味着必须保持PMCSR.PME\_Status和PMCSR.PME\_En位。此外，必须维护PMC.PME\_Support(D3cold)位。

此外，EHCI控制器必须保留满足以下任何标准的功能特定上下文：

1. 在系统初始化期间编程的BIOS配置寄存器
2. 避免USB重新枚举所需的上下文
3. 正确生成唤醒事件所需的上下文
4. 用于确定唤醒事件来源的软件的状态位

具体而言，以下EHCI寄存器在D3hot到D0转换期间不得重置，必须保持在辅助电源井中（见下文第A.1.2节）：

- ① 配置的标志位
- ② 端口状态和控制寄存器

请注意，上述所有寄存器仅在初始Aux通电或软件重置时重置。如果需要，软件必须在随后的初始化序列中明确清除这些位中的任何一个。还可以使用USB命令寄存器中的主机控制器重置机制来清除存储器空间位。

### A.1.2 电源状态定义

本节定义了使用PMCSR PowerState编程时每个电源状态的EHCI行为。电源管理软件可以使用备用寄存器机制将EHCI置于类似状态。EHCI应支持D0、D3hot和D3cold电源状态，建议支持D1、D2电源状态。

当EHCI编程时，表A-4中规定的任何唤醒事件都将设置PMCSR.PME\_Status设置为D0。如果通过PMCSR PME\_En启用，则应发出PCI PME#唤醒信号。一个也是唤醒事件的中断事件可能会导致EHCI向主机发出PCI中断和PME#信号。电源管理软件应设计为处理这种情况，或在EHCI处于D0时屏蔽PME#信号。

软件应在尝试将EHCI移出D0电源状态之前，将每个已启用电源的下游USB端口置于挂起或禁用状态。所有EHCI上下文都保留在除D3cold之外的所有功率状态中。对于D3cold，必须保留上一节中关于D3hot-to-D0内部重置所描述的相同上下文。

表A-4总结了EHCI功率状态的功能和唤醒特性。

表A-4。EHCI电源状态摘要

电源状态	功能特性	唤醒特性(必须设置相关启用)
D0	<ul style="list-style-type: none"> <li>■ 全功能EHCI设备状态</li> <li>■ 无掩码中断功能齐全</li> </ul>	<ul style="list-style-type: none"> <li>■ 在挂起的端口上检测到恢复</li> <li>■ 在端口上检测到连接或断开连接</li> <li>■ 端口上检测到过电流</li> </ul>
第1页	<ul style="list-style-type: none"> <li>■ EHCI应保留PCI配置</li> <li>■ EHCI应保留USB配置</li> <li>■ 硬件屏蔽功能中断</li> <li>■ 所有端口都已禁用或挂起</li> </ul>	<ul style="list-style-type: none"> <li>■ 在挂起的端口上检测到恢复</li> <li>■ 在端口上检测到连接或断开连接</li> <li>■ 端口上检测到过电流</li> </ul>
D2	<ul style="list-style-type: none"> <li>■ EHCI应保留PCI配置</li> <li>■ EHCI应保留USB配置</li> <li>■ 硬件屏蔽功能中断</li> <li>■ 所有端口都已禁用或挂起</li> </ul>	<ul style="list-style-type: none"> <li>■ 在挂起的端口上检测到恢复</li> <li>■ 在端口上检测到连接或断开连接</li> <li>■ 端口上检测到过电流</li> </ul>
D3热	<ul style="list-style-type: none"> <li>■ EHCI应保留PCI配置</li> <li>■ EHCI应保留USB配置</li> <li>■ 硬件屏蔽功能中断</li> <li>■</li> </ul>	<ul style="list-style-type: none"> <li>■ 在挂起的端口上检测到恢复</li> <li>■ 在端口上检测到连接或断开连接</li> <li>■ 端口上检测到过电流</li> </ul>
D3冷	<ul style="list-style-type: none"> <li>■ 保留PCI配置空间中的PME上下文</li> <li>■ 保留EHCI内存空间中的唤醒上下文</li> <li>■ 所有端口都已禁用或挂起</li> </ul>	<ul style="list-style-type: none"> <li>■ 在挂起的端口上检测到恢复</li> <li>■ 在端口上检测到连接或断开连接</li> <li>■ 端口上检测到过电流</li> </ul>

### A.1.3 PCI PME#信号

PCI PME#信号应实现为开路漏极、有源低信号，该信号由EHCI驱动为低电平，以请求改变其当前功率管理状态。PME#除了标准开漏信号外，还有其他电气要求，允许在断电和通电的设备之间共享。有关更多详细信息，请参阅PCI电源管理规范。

页面故意留空

## 附录B.EHCI 64位数据结构

本附录列出了数据结构的64位版本。

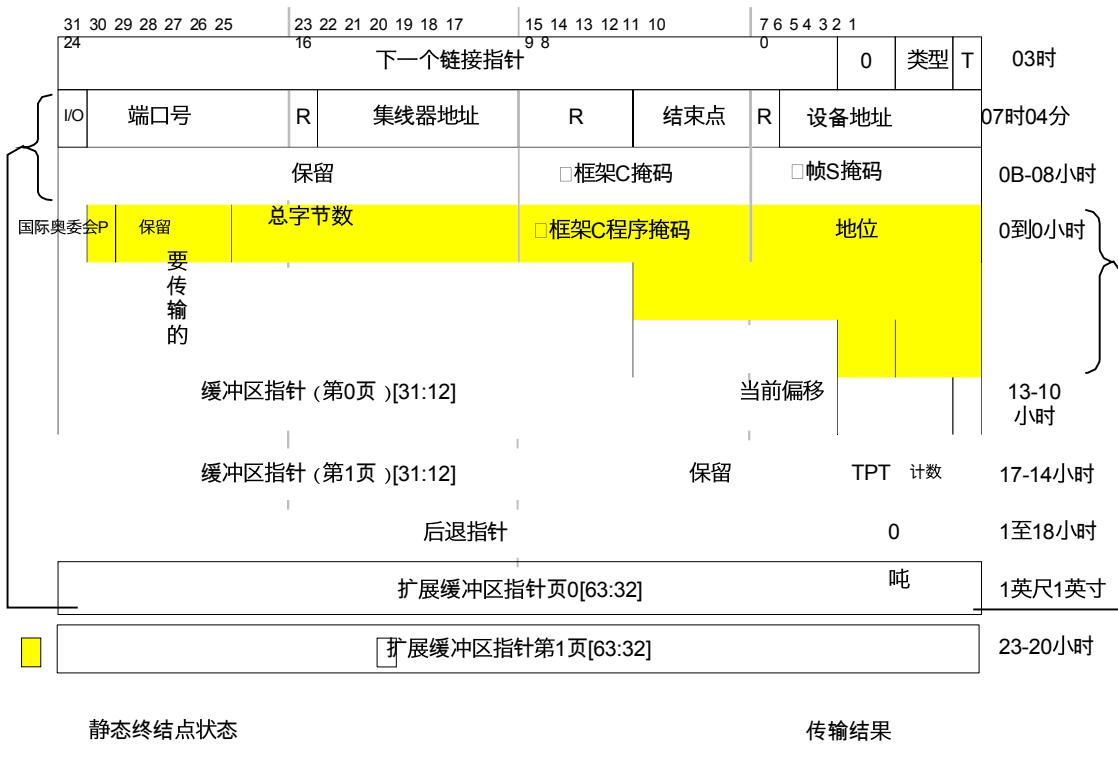
下一个链接指针					0	类型 T	03时
地位	事务处理0长度	国际	PG*	事务处理0偏移*			07时04分
地位	事务处理1长度	国际	PG*	事务处理1抵销*			0B-08小时
地位	事务处理2长度	国际	PG*	事务处理2抵销*			0到0小时
地位	事务处理3长度	国际奥委会	PG*	事务处理3抵销*			13-10小时
							17-14小时
地位	事务处理5长度	国际	PG*	事务处理5抵销*			1至18小时
地位	事务处理6长度	国际	PG*	事务处理6抵销*			1英尺1英寸
地位	事务处理7长度	国际	PG*	事务处理7抵销*			23-20小时
缓冲区指针 (第0页 )[31:12]				结束点 R	设备地址		27-24小时
缓冲区指针 (第1页 )[31:12]				I/O	最大数据包大小		2小时-28小时
缓冲区指针 (第2页 )[31:12]					保留	Mult 公司	2英尺2英寸
缓冲区指针 (第3页 )[31:12]					保留		33-30小时
缓冲区指针 (第4页 )[31:12]					保留		37-34小时
缓冲区指针 (第5页 )[31:12]					保留		3至38小时
缓冲区指针 (第6页 )[31:12]					保留		3英尺3英寸
扩展缓冲区指针页0[63:32]							43至40小时
扩展缓冲区指针第1页[63:32]							47-44小时
扩展缓冲区指针第2页[63:32]							4小时至48小时
扩展缓冲区指针第3页[63:32]							4F-4英寸
扩展缓冲区指针第4页[63:32]							53-50小时
扩展缓冲区指针第5页[63:32]							57至54小时
扩展缓冲区指针第6页[63:32]							第5页至第58页

主机控制器读/写

主机控制器只读

\*注意：如果I/O字段指示OUT，则主机控制器可以修改这些字段。

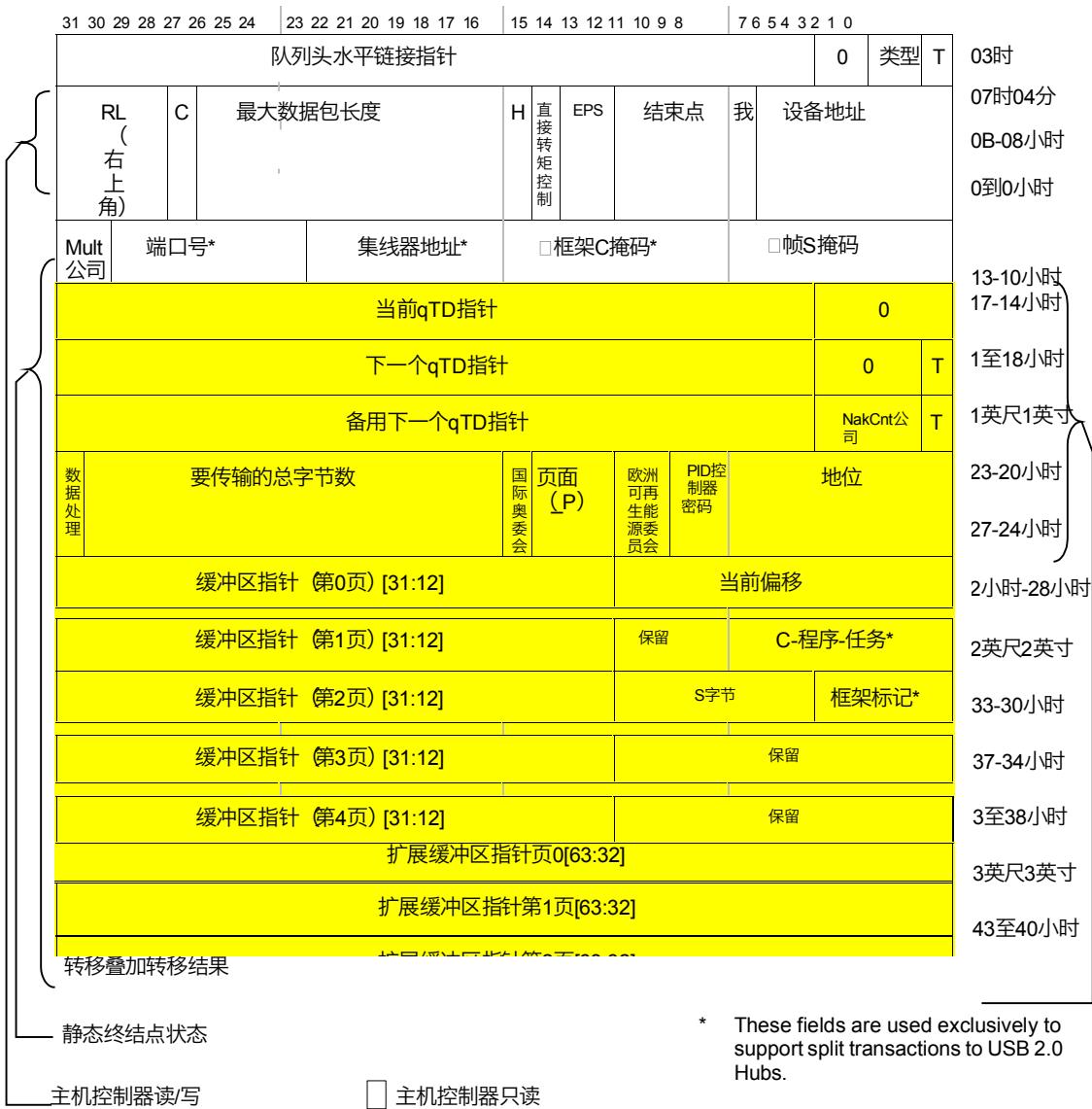
图B-1。64位等时事务描述符 (iTDD-64 )



图B-2。64位拆分事务等时事务描述符 (siTD-64)



图B-3。64位队列元素事务描述符(qTD-64 )



图B-4。64位队列头描述符 (QH-64)

注：不需要64位版本的FSTN。结构指针的高32位必须来自CTRLDSSEGMENT寄存器。

页面故意留空

## 附录C. 调试端口

调试端口是一个可选的实现功能。本附录描述了作为EHCI控制器一部分的USB2调试端口所需的实现和行为。此调试端口实现的具体功能包括：

- ① 仅适用于高速USB调试设备
- ② 针对主机控制器上的特定端口实现
- ③ 只要端口未挂起且主机控制器处于D0电源状态，即可运行。
- ④ 端口驱动USB RESET时功能中断

### C.1 定位调试端口

PCI Capability List<sup>5</sup>用于为软件提供查找和使用调试端口的标准方法。图C-1显示了调试端口功能布局，由三个字段组成。CAP\_ID如表C-1所述，NXT\_PTR如表C-2所示，DEBUG\_PORT如表C-3所示。

调试端口	下一页	队长ID
------	-----	------

图C-1。调试端口能力寄存器布局

表C-1。队长ID

位	类型	领域	描述
7:0分	罗	队长ID	此字段中的值0Ah表示函数支持调试端口。

表C-2。下一页

位	类型	领域	描述
15时08分	罗	下一页	指向功能列表中下一项的指针。列表中的最后一项必须为NULL。

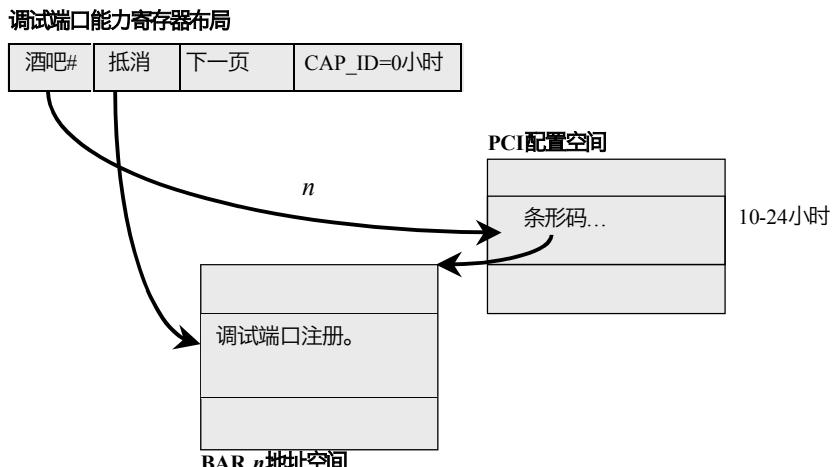
<sup>5</sup> PCI能力列表在PCI本地总线规范中定义（第6.7节能力列表，修订版2.21999年12月18日）。

表C-3。调试端口

位	类型	领域	描述
28分16秒	罗	抵消	该12位字段指示由BAR#指示的BAR内的字节偏移（高达4K）。这个偏移量需要DWORD对齐，因此位16和17总是为零。
31分29秒	罗	酒吧#	一个3位字段，指示可能的6个基址寄存器偏移中的哪一个包含调试端口寄存器。例如，值1h表示第一个BAR（偏移10h），而值5表示在20h处的BAR。该偏移与BAR是32位还是64位无关。例如，如果偏移量为3，则表示偏移量18h处的BAR包含调试端口。偏移量10和14h处的BARs可以被实现，也可以不被实现。此字段为只读字段，只有值1-6h才有效。（允许使用64位BAR。）只允许使用内存BAR。

## C.2 使用调试端口字段

“调试端口”字段由两个子字段组成。第一个是OFFSET，第二个是BAR#。图C-2说明了软件如何使用字段来定位和访问调试端口寄存器。从调试端口能力寄存器的BAR#到PCI配置空间中BARs字段的箭头表示BAR#包含一个指向设备的一个BAR的“指针”，调试端口寄存器映射在该设备中。



图C-2 调试端口能力寄存器

调试端口能力寄存器的OFFSET用作BAR分配的地址空间的偏移量。因此，软件采用基址和OFFSET来确定调试端口寄存器的起始地址。例如，BAR的编程地址为3000 0000h (3 GB)，偏移量为20 h (32字节)。因此，调试寄存器的基址将是3000 0020h。

## C.3 USB2调试端口寄存器接口

EHCI实现使用PCI Capability结构（如上所述）来指示它提供调试端口。USB2调试端口寄存器与标准EHCI寄存器位于由基本地址寄存器(BAR)定义的相同存储区域中。支持这种调试能力的特定EHCI端口由EHCI控制器的HCSPARAMS寄存器中的4位字段(位20-23)指示。

**表C-4。调试端口控制寄存器**

注册机构名称	抵消
控制/状态	0x00小时
USB PID	0x04小时
数据缓冲区 (字节3-0)	0x08小时
数据缓冲区 (字节7-4)	0x0通道
设备地址	0x10小时

**C.3.1 调试端口控制寄存器**

调试端口控制寄存器 (偏移量0h) 允许软件与USB2调试端口交互。与此寄存器相关的硬件不提供任何检查，以确保软件正确编程接口。非法编程时硬件的行为尚不明确。控制寄存器中每个位或字段的定义如下表所示。注意：控制寄存器中的所有位都可以同时写入或读取。软件需要通过执行读-修改-写操作来保留保留字段/位。具有重置值的寄存器将根据HCReset、D3-D0转换或PCI重置进行重置。

**表C-5。控制/状态寄存器位**

位	类型	领域	描述
31	罗	保留	
30	R/W	物主	当调试软件向该位写入一时，调试端口的所有权被强制分配给EHCI控制器（即立即从伴随控制器中夺走）。如果该端口已归EHCI控制器所有，则将该位设置为无效。该位覆盖标准EHCI寄存器中所有与所有权相关的位。重置默认值=0。请注意，该位中的值可能不会影响相关PORTSC寄存器中端口所有者位中报告的值。
29	罗	保留	
28	R/W	已启用	如果调试端口已启用，则此位为1。软件可以通过向其写入零来清除此位。控制器在硬件设置端口启用/禁用更改位（在PORTSC寄存器中）的相同条件下清除该位。（注意：当系统软件清除端口启用/禁用位（在PORTSC寄存器中）时，该位不会被清除。如果相关端口状态和控制寄存器中的端口已启用，软件可以直接设置此位（这是硬件强制执行的）。重置默认值=0。）
27分17秒	罗	保留	
16	R/WC	完成	该比特由HW设置以指示请求完成。将1写入此位将清除它。将0写入此位没有任何效果。重置默认值=0。
15时11分	罗	保留	
10	R/W	正在使用中	由软件设置以指示端口正在使用中。由软件清除，表示端口是空闲的，可以由其他软件使用。重置默认值=0。（此位对硬件没有影响。）

表C-5。控制/状态寄存器位 (续)

位	类型	领域	描述										
9点7分	罗	例外	<p>此字段指示设置错误/良好#时的异常。软件无法清除此字段。重置默认值=000b。</p> <table> <thead> <tr> <th>价值</th><th>含义</th></tr> </thead> <tbody> <tr> <td>000b</td><td>无</td></tr> <tr> <td>001b</td><td>事务错误或胡言乱语：表示USB2事务有错误（CRC、PID错误、超时、数据包胡言乱语等）</td></tr> <tr> <td>010b</td><td>HW错误。端口挂起或重置时，已尝试（或正在进行）请求。</td></tr> <tr> <td>011b-111b</td><td>保留</td></tr> </tbody> </table>	价值	含义	000b	无	001b	事务错误或胡言乱语：表示USB2事务有错误（CRC、PID错误、超时、数据包胡言乱语等）	010b	HW错误。端口挂起或重置时，已尝试（或正在进行）请求。	011b-111b	保留
价值	含义												
000b	无												
001b	事务错误或胡言乱语：表示USB2事务有错误（CRC、PID错误、超时、数据包胡言乱语等）												
010b	HW错误。端口挂起或重置时，已尝试（或正在进行）请求。												
011b-111b	保留												
6.	罗	错误/ 良好#	硬件在设置完成位的同时进行更新。设置时，表示发生了错误。异常字段中提供了错误的详细信息。清除后，表示请求已成功终止。重置默认值=0。										
5.	R/W	去	软件设置此位以使硬件执行请求。当该位已经被设置时，将该位写入1可能导致未定义的行为。将0写入此位没有任何效果。设置时，当硬件设置完成位时，硬件清除该位。（完成位表示请求完成。）重置默认值=0。										
4.	R/W	写入/ 读取#	软件将该位设置为指示当前请求是写入，并将其清除为指示读取。重置默认值=0。										
3点	R/W	数据长 度	<p>对于写入操作，此字段由软件设置，以向硬件指示当软件设置Go时设置write/Read#时，数据缓冲区中有多少字节的数据要传输到控制台。值0h表示应该发送零长度的数据包。1-8的值表示要传输1-8个字节。值9-Fh是非法的，并且未定义硬件在使用时的行为。</p> <p>对于读取操作，此字段由硬件设置，以向软件指示数据缓冲区中有多少字节是有效的，以响应清除写入/读取#时的软件设置Go。值0h表示返回了零长度的数据包。（未定义数据缓冲区的状态。）值1-8表示已接收1-8个字节。in事务完成后，当错误/良好#字段为零（0b）时，硬件不允许返回范围为9-Fh的值。当in事务完成后错误/良好#字段为一（1b）时，此字段中的值无效。</p> <p>数据的传输总是从数据区中的字节0开始，并向字节7移动，直到达到传输大小。重置默认值=0h。</p>										

### C.3.2 USB PID寄存器

此D字寄存器用于在USB调试驱动程序和USB2调试端口之间传递PID信息。调试端口使用其中一些字段生成USB数据包，并使用其他字段将PID信息返回给USB调试驱动。

**表C-6。USB PID寄存器**

位	类型	领域	描述
31分24秒	罗	保留	
23时16分	罗	收到PID	调试端口控制器用接收到的PID为任一方向的事务更新此字段。当控制器发送数据时（写入/读取#被断言），该字段将使用从设备接收的握手PID进行更新。当主机控制器正在接收数据时（写入/读取#未被断言），该字段用数据包PID（如果设备发送数据）或握手PID（如果该设备NAK请求）更新。当控制器将完成位设置为一（1b）并且 <b>控制/状态</b> 寄存器中的错误/良好#字段为零时，该字段有效。如果事务完成（完成位转换为1），错误/良好#字段设置为1，则该字段的内容未定义。重置默认值=未定义。
15时8分	R/W	发送PID	当向USB发送数据时，调试端口控制器发送此PID以开始数据包（即断言写入/读取#）。软件通常会将该字段设置为DATA0或DATA1 PID值。重置默认值=未定义。
7:0分	R/W	令牌PID	调试端口控制器发送此PID作为每个USB事务的令牌PID。软件通常会将该字段设置为IN、OUT或SETUP PID值。重置默认值=未定义。

### C.3.3 数据缓冲区

数据缓冲器由8个字节组成，这些字节被排列为位于调试端口寄存器空间中偏移08h和0Ch处的两个连续的D字。表C-7提供了更多详细信息。

**表C-7。数据缓冲区**

位	类型	领域	描述
63分0秒	R/W	数据缓冲区	最低有效字节在偏移08h处被访问，而最高有效字节则在偏移0Fh处被存取。数据缓冲区中的每个字节都可以单独访问。 在软件启动写入请求之前，必须用数据写入数据缓冲区。对于读取请求，“数据缓冲区”在设置“完成”时包含有效数据，“错误/良好#”为零（0b），“数据长度”指定有效的字节数。如果IN事务完成（完成位转换为1），错误/良好#字段设置为1，则该字段的内容未定义。重置默认值=未定义。

### C.3.4 设备地址寄存器

设备地址寄存器保存生成事务时正确寻址调试设备所需的信息。它指定调试设备的USB设备地址，以及调试端口生成的USB令牌的端点地址。

**表C-8。设备地址寄存器**

位	类型	领域	描述
31分15秒	罗	保留	
14时8分	R/W	USB接口住址	7位字段，用于标识控制器用于所有令牌PID生成的USB设备地址。这是一个R/W字段，在通电复位后设置为7Fh。重置默认值=7Fh。
7点4分	罗	保留	
3点	R/W	USB接口终点	4位字段，用于标识控制器用于所有令牌PID生成的端点。重置默认值=01h。

## C.4 运营模式

USB2调试端口有两种操作模式。模式1是从标准EHCI驱动程序的角度来看，当EHCI主机控制器未运行时（例如，USBSTS寄存器中的*HCHalted*位为1）。在模式1中，控制器需要生成间隔不到2毫秒的“保活”数据包，以防止连接的调试设备挂起。保活数据包是一个独立的32位SYNC字段。

模式2是主机控制器正在运行时（即运行/停止位为1）。在模式2中，SOF数据包的正常传输（或者如果PORTSC寄存器端口启用/禁用位为零，则SYNC保持有效）将防止调试设备挂起。

在这两种模式下，控制器必须至少每125微妙检查一次软件请求的调试事务。

如果调试端口由调试驱动程序启用，并且标准主机控制器驱动程序重置USB端口，则在重置的持续时间内以及直到发送第一个SOF之后，USB调试事务在异常条件（0b010）下完成。

如果标准主机控制器驱动程序挂起USB端口，则在挂起/恢复序列的持续时间内以及直到发送第一个SOF之后，USB调试事务在异常条件（0b010）下完成。

调试寄存器空间中的启用端口控制位独立于相关端口状态和控制寄存器中的类似端口控制位。

下表显示了与调试寄存器中的位以及相关端口状态和控制寄存器中的位数的状态相关的调试端口行为。

**表C-9。端口行为与寄存器设置的摘要**

调试位		EHCI位			调试端口行为
物主	已启用	端口启用	运行/停止	悬	
0	+	+	+	+	调试端口未被使用。正常操作。
1.	0	+	+	+	调试端口未被使用。正常操作。
1.	1.	0	0	+	模式1中的调试端口。发送的SYNC保持有效加上调试流量

表C-9。端口行为与寄存器设置摘要 (续)

调试位		EHCI位			调试端口行为
物主	已启用	端口 启用	运行/ 停止	悬	
1.	1.	0	1.	+	模式2中的调试端口。发送的SYNC保持有效(或SOF)加上调试事务。请注意，此端口上不会发送其他USB事务，因为相关PORTSC寄存器中的端口启用位为零。
1.	1.	1.	0	0	模式2中的调试端口。发送的SYNC保持有效加上调试流量。
1.	1.	1.	0	1.	端口已挂起。未发送调试流量。
1.	1.	1.	1.	0	模式2中的调试端口。调试流量与正常流量穿插在一起。
1.	1.	1.	1.	1.	端口已挂起。未发送调试流量。

#### C.4.1 输出/设置事务

当调试端口被启用并且调试软件设置Go位并且写入/读取#位被设置时，则调试控制器被启用以发送将向调试设备发送数据的调试事务。控制器发送一个令牌包，该令牌包由SYNC、令牌PID字段、设备地址字段、端点字段组成，后面跟着一个5位CRC字段。在发送令牌包之后，控制器发送由SYNC、发送PID、来自数据寄存器的数据的数据长度字节和16位CRC组成的数据包。请注意，数据长度值为零是有效的，在这种情况下，数据包中将不包括任何数据字节。在发送数据包之后，控制器等待来自调试设备的握手响应。如果接收到握手，控制器将接收的PID放入接收的PID寄存器，重置错误/良好#位并设置完成位。如果没有接收到握手PID，则控制器将Exception字段设置为001b，设置Error/Good#位，并设置Done位。

#### C.4.2 IN交易

当调试端口被启用并且调试软件设置Go位并且写入/读取#位被重置时，则调试控制器被启用以发送将从调试设备接收数据的调试事务。控制器发送一个令牌包，该令牌包由SYNC、令牌PID字段、设备地址字段和端点字段组成，后面跟着一个5位CRC字段。在发送令牌分组之后，控制器等待来自调试设备的响应。如果接收到响应，则接收的PID被放入接收的PID寄存器中，并且任何后续字节被放入数据寄存器中。数据长度字段将更新，以显示PID之后接收到的字节数。如果从设备接收到有效的数据包，并且它的总长度为一个字节（表明它是握手数据包），则控制器重置错误/良好#位并设置完成位。如果从设备接收到有效的数据包，并且它的总长度超过一个字节（表明它是一个数据包），则控制器发送ACK握手数据包，重置错误/良好#位并设置完成位。如果没有接收到有效的分组，则控制器将Exception字段设置为001b，设置Error/Good#比特，并设置Done比特。

#### C.4.3 调试软件启动

调试软件启动有两种常见情况：1) 当EHCI控制器尚未由系统主控制器驱动程序初始化时，以及2) EHCI控制器已由系统主控制程序初始化时。调试软件通常知道它必须处理什么情况（通常是情况1），但可以通过检查EHCI CONFIGFLAG寄存器中的Configured位来进行进一步的确定。如果设置了配置标志，则表示系统主机控制器驱动程序已初始化EHCI控制器。以下各节提供了这两种情况的一般启动程序。

#### C.4.3.1 在系统主机控制器驱动程序之前启动

如果在设置所有者位之后，设置了相应PORTSC寄存器中的当前连接状态位，则调试软件可以尝试使用调试端口。调试软件应首先通过确保主机控制器正在运行来重置连接的设备（USBCMD寄存器中的运行/停止位为1，USBSTS寄存器中的 $HCalte$ d位为0），然后设置并清除PORTSC寄存器中的端口重置位。如果连接了高速设备，控制器将自动设置PORTSC寄存器中的端口启用/禁用位，调试软件可以继续。调试软件应设置调试端口控制寄存器中的启用位，然后重置PORTSC寄存器中的端口启用/禁用位（这样系统主机控制器驱动程序在首次加载时就看不到启用的端口）。当端口重置序列完成时，调试软件可以通过将运行/停止位设置为零来关闭主机控制器。

#### C.4.3.2 在系统主机控制器驱动程序之后启动

如果连接了设备（由PORTSC寄存器中的“当前连接状态”位指示），调试软件可以在调试端口控制寄存器中设置“所有者”位，然后直接在调试端口控制器寄存器中设置启用位。

### C.4.4 查找调试外围设备

在启用调试端口功能之后，调试软件可以通过尝试向调试外围设备发送数据来确定是否连接了调试外围设备。如果所有尝试都导致错误（Exception字段表示Transaction error），则所连接的设备不是调试外围设备。

## 附录D.高带宽等时规则

高带宽等时流利用USB协议中的附加PID。本附录中的表格完全列举了EHCI控制器在执行高带宽等时数据流时必须做出的所有必要响应。

每个表格由以下字段组织：

- ① **输入:** 列出行行为数据点的输入或初始条件。输入值为：
  - ① **方向:** 表示交易的方向。
  - ① **Mult:** 这是iTDE实例化中Mult字段的值。这是iTDE使用寿命的恒定值。它是Cnt的初始值（见下文）。该字段是基于设备提供的USB框架参数设置的。它不是相对于缓冲区大小等设置的。
  - ① **Cnt:** 这是事务迭代器。它是内部事务计数器的当前值，对于OUT，最初加载Mult的内容。对于IN，Cnt最初是根据第一个总线事务的PID响应设置的（见下文）。
  - ① **剩余缓冲区:** 剩余缓冲区的数量由当前交易记录中“交易X长度”字段的当前值表示。该字段的初始值由软件设置，以指示可用于该事务记录的缓冲量。它由主机控制器在执行事务和移动数据时进行调整。
- ① **响应:** 列出设备的响应（PID代码和数据大小）以及对iTDE状态字段位和事务迭代器的影响。
  - ① **PID/ 数据大小:** 表示主机刺激、数据PID或设备的其他响应。
- ① **结果:** 列出响应对Status字段和迭代器中的位的影响。

每个表中的每一行都说明了高带宽等时事务的所有输入/响应组合所需的主机控制器行为。本附录中有两张表。第一个枚举OUT事务所需的行为，第二个枚举IN事务所需行为。

表D-1。OUT事务的高带宽行为

目录	输入		回答	后果	解释	
	Mult公司	Cnt公司	剩余缓冲区	PID (数据大小)		
出来	1.	1.	□最大数据包	PID控制器□ DATA0 (最大数据包)	忙碌的□ 0	正常完成1、2或3个高带宽事务 (针对微帧)；发送maxpacketsize字节。
				PID控制器□ DATA1 (最大数据包)		
				PID控制器□ DATA2 (最大数据包)		请注意□应用>的最大数据包只是为了说明软件对Mult或Transaction Length编程错误的情况。
	1.	2.	□ 最大数据包	PID控制器□ DATA0 (Xfer长度)	忙碌的□ 0	1、2或3个高带宽事务的正常完成 (针对帧)；发送缓冲区中可用的字节数。
				PID控制器□ DATA1 (Xfer长度)		
				PID控制器□ DATA2 (Xfer长度)		
	3.2个	2.	④ 最大数据包	PID控制器□ 最大数据包	忙碌的□ 不适用	高带宽序列中的中间事务；使用MDATA PID发送maxpacketsize字节。
				PID控制器□ DATA0 (Xfer长度)	忙碌的□ 0	软件没有为此事务 (微帧) 发送的Mult*MaxpacketSize字节。
	3.	3.	④ 最大数据包	PID控制器□ DATA1 (Xfer长度)		
				PID控制器□ 最大数据包	忙碌的□ 不适用	高带宽序列中的中间事务；使用MDATA PID发送maxpacketsize字节。
	3.	3.	□最大数据包	PID控制器□ DATA0 (Xfer长度)	忙碌的□ 0	软件没有为此事务 (微帧) 发送的Mult*MaxpacketSize字节。
				PID控制器□ MDATA (缓冲区错误)	忙碌的□ 0 BuffErr公司□ 1	主机在能够传递所有数据之前遇到了缓冲区错误。它不得在此端点上执行任何进一步的请求。

任何时候出现缓冲区错误 (在这种情况下是缓冲区运行不足)，主机控制器都将放弃高带宽事务的剩余部分。例如，如果当前PID是MDATA，并且在及时将数据从主存储器获取到HC时出现缓冲区错误，则主机控制器将缓冲区错误状态位设置为1，并立即将活动状态位清除为0。这将导致主机控制器有效地跳过剩余的总线事务 (如果有任何挂起的事务，基于Cnt的值)。

使用状态机模型描述了主机控制器对管理高带宽IN总线事务序列的要求。该模型总结在状态转换表表D-2中。这只是一个示例状态机，其目的是定义主机控制器的操作要求。

本节的目的是明确定义高带宽等时数据流的适当数据PID序列，并设置在高带宽事务序列期间可检测的错误的检测和报告的优先级。

高带宽PID跟踪状态机的前提是当前微帧的数据PID序列由设备对微帧的第一个IN的响应来确定。基于PID响应，主机控制器设置内部计数变量(Cnt)，该内部计数变量用于驱动状态机通过高带宽事务序列的剩余阶段(状态)。

每一个微帧，机器都被初始化为启动状态。在这种状态下，内部计数器的值为“不在乎”(X)。主机控制器发出初始IN，然后将内部计数器(Cnt)设置为接收到的数据PID的值编号(Y)。例如，如果PID响应为DATA2，则Cnt加载值“2”。当PID是DATA1或DATA2时，则执行两个额外的检查。如果这两项检查都没有失败，则主机控制器将转换到“下一步”状态。

1. 数据有效载荷的大小必须等于最大数据包长度(Maxpacket)，并且
2. 主机控制器必须检查启动PID响应是否在Mult中指定的为此端点配置的范围内。如果PID值编号(Y)小于Mult的值，则接收到的数据PID在适当的范围内。例如，如果Mult是2并且设备返回DATA1，则Y=1小于Mult，因此接收到的PID是可接受的。

当在开始状态下接收到的PID为DATA0时，则该微帧的高带宽事务完成，并且主机控制器必须将活动位设置为零。允许有效的DATA0 PID具有小于或等于Maxpacket的数据有效载荷大小。如果检测到牙牙学语错误，则主机控制器将额外地将牙牙学音比特设置为1。

表D-2。IN事务的高带宽行为

当前状态		终结点响应		后果	下一个状态	解释		
Cnt	公司	PID[年]	数据大小					
开始	十、	PID控制器□数据[2,1]	Y□ Mult公司	= Maxpacket		Cnt=[2,1]		
				□ 最大数据包	忙碌的□ 0中, XactErr公司□ 1.	完成		
				⊕ 最大数据包	忙碌的□ 0中, 含糊不清地说□ 1.	完成		
				Y□ Mult公司	不在乎	忙碌的□ 0中, XactErr公司□ 1.		
	PID控制器□数据0			□ 最大数据包	忙碌的□ 0	完成		
				⊕ 最大数据包	忙碌的□ 0中, 含糊不清地说□ 1.	完成		
	下一个	PID控制器□数据2		不在乎	忙碌的□ 0中, XactErr公司□ 1.	完成		
		PID控制器□数据1		= Maxpacket		Cnt=1		
				□ 最大数据包	忙碌的□ 0中, XactErr公司□ 1.	完成		
				⊕ 最大数据包	忙碌的□ 0中, 含糊不清地说□ 1.	完成		
	1.	PID控制器□数据0		不在乎	忙碌的□ 0中, XactErr公司□ 1.	完成		
		PID控制器□数据[2,1]		不在乎	忙碌的□ 0中, XactErr公司□ 1.	完成		
		PID控制器□数据0		□ 最大数据包	忙碌的□ 0	完成		

		② 最大数据包	忙碌的□ 0中, 含糊不清地说□ 1.	完成	大于最大数据包大小的数据有效载荷是一种胡言乱语的情况。
--	--	---------	------------------------	----	-----------------------------

在*Next*状态下，主机控制器发出In令牌，并对照内部计数器 (Cnt) 的值检查PID响应的值编号 (Y)。如果数值 (Y) 等于 (Cnt-1)，则PID响应正确，主机控制器将内部计数器 (Cnt) 设置为接收到的数据PID的数值。

当接收到的PID响应是可接受的并且是DATA1时，则主机控制器还必须检查数据有效载荷的大小是否等于配置的最大分组长度 (Maxpacket)。如果长度检查通过，则PID检查已经通过，并且主机控制器进行最后的牙牙学语检查。如果没有牙牙学语错误，主机控制器将保持在下一个状态，并执行另一个总线事务。如果出现错误，主机控制器会将活动位设置为零。如果长度检查失败，主机控制器会将*XactErr*设置为1。

当接收到的PID响应是可接受的并且是DATA0时，则该微帧的高带宽事务完成，并且主机控制器必须将活动位设置为零。允许数据有效载荷小于或等于配置的最大分组大小。如果检测到一个牙牙学语错误，那么主机控制器将把牙牙学音位设置为1。

每当单个事务在超时中完成时，主机控制器都会将状态位*Active*设置为零，将状态位*XacErr*设置为一。

请注意，此状态机用于说明目的。只要得到的行为是正确的，实现可以适当地优化以尽可能避免算术运算。