ISP1581 编程指南

1. 简介

ISP1581 是一种高速通用串行总线（USB 2.0）接口器件，它为各种范围的微控制器提供了灵活的接口。这些高速微控制器接口增加了系统的数据吞吐量，简化了处理器的应用。

本文简要描述了在带有 16 位数据总线的通用处理器接口上如何实现 ISP1581 硬件和固件的应用。要注意的是，本文在这里所描述的相关硬件不同于 ISP1581 的断开总线工作方式下的硬件，不要将两者混淆。

2. ISP1581 微控制器接口信号

ISP1581 为微控制器接口提供了灵活的配置。对于大多数的微处理器都不要求固定的逻辑。

下面的两张表列出了 ISP1581 引脚的典型连接。

表 2－1 微处理器接口信号连接

ISP1581 信号	微处理器	注释
AD[0][1]	不连接	在 16 位总线上 AD[0]必须与 ISP1581 的地端相连
AD[7:1][2]	地址总线 7 到 1	－
$\overline{CS}$	系统译码芯片选择器	必须用在对 16 位存储单元的访问中
DATA[15:0][3]	16 位数据总线	－
$\overline{DS}$ / $\overline{WR}$	WR	写选通
INT	微控制器的任意中断输入	－
(R/$\overline{W}$)/$\overline{RD}$	RD	读选通

表 2－2 直接存储器存取（DMA）

ISP1581 信号	DMA 控制器	注释
DREQ[4]	DMA 请求输入	－
DACK[5]	DMA 应答输出	－
DIOR	DMA 读信号	当 DMA 控制器和微控制器使用同一个读选通信号将它与 ISP1581 的 $\overline{RD}$ 短接[6]
DIOW	DMA 写信号	当 DMA 控制器和微控制器使用同一个写选通信号将它与 ISP1581 的 $\overline{WR}$ 短接[7]
EOT	DMA 传输结束输出	－

1. 16 位的接口要求能对所有地址进行访问。因此，AD[0]不连接是一些固件要利用这一位来产生混淆字校正代码。
2. AD[7:0] 在通用处理器模式下用作地址总线，在断开总线模式下用作复用地址/数据总线，微控制器利用它来控制 ISP1581。
3. DATA[15:0]在通用处理器模式下用作 DMA 总线和系统总线，通过它来控制 ISP1581。但是，在断开总线模式下 DATA[15:0]仅用作 DMA 总线。
4. ISP1581 的 DMA 核可用作 DMA 主机和 DMA 从机，取决于开始的操作码（DMA 命令寄存器，地址 30H）。DREQ 和 DACK 一直保持高阻态直至执行完 DMA 命令。
5. 见第 4 条。
6. 当 ISP1581 工作在 DMA 的 ACK 模式，读和写信号都必须为高。
7. 见第 6 条。

ISP1581 通过上电复位时评估引脚电平来确定操作码（见表 2-3）。

表 2-3 建立配置

ISP1581 信号	复位时的信号电平	配置
BUS_CONF	1	通用微处理器接口；16 位数据总线和 8 位地址线 [1]
MODE0	1	ISP1581 通过检测 $\overline{RD}$ 来“读”，检测 $\overline{WR}$ 来“写”
ALE	1	若 ALE 未用，置为低

3. 初始化寄存器

固件必须对 ISP1581 的寄存器初始化来配置 I/O 口的信号电平以满足系统设置的要求。

下面的几张表列出了 ISP1581 的方式寄存器、中断配置寄存器、DMA 配置寄存器和 DMA 硬件寄存器的一般初始化方法。

表 3-1 方式寄存器（0CH）

寄存器位（Hex）	寄存器位符号	ISP1581 信号	注释
0C.7	CLKAON	无对应的信号	0－在挂起状态下关掉时钟来降低功耗
0C,3	GLINTENA	INT	1－全局中断使能
0C,2	WKUPCS	$\overline{CS}$ 和 WAKEUP	1－ $\overline{CS}$ 有效，将 ISP1581 从挂起状态唤醒（在 WAKUP 下所有的设置不变）
0C.1	PWROFF	SUSPEND	1－挂起状态 SUSPEND 引脚为高，唤醒时为低 0－唤醒到来时 SUSPEND 引脚产生一个脉冲，其他时间仍然保持低电平
0C.0	SOFTCT	RPU	0－RPU 引脚上的 1.5KΩ电阻与内部 D+断开 1－RPU 引脚上的 1.5KΩ电阻与内部 D-相连（USB 1.1 全速功能）

表 3-2 中断配置寄存器（10H）

寄存器位 [2]（Hex）	寄存器位符号	ISP1581 信号	注释
10.1	INTLVL	INT	1－中断发生时在 INT 引脚上产生一个脉冲 0－只要有任何中断事件等待处理，中断处于有效状态
10.0	INTPOL	INT	1－INT 保持不变，中断到来时变为高 0－INT 保持不变，中断到来时变为低

表 3-3DMA 配置寄存器（38H）

寄存器位 [2]（Hex）	寄存器位符号	ISP1581 信号	注释
38.7	DIS_XFER_CNT	EOT	1－DMA 计数器禁止，DMA 传输的终止完全取决于传输终止（EOT）信号的有效
38.6 到 4	BURST[2:0]	DREQ 和 DACK	决定 DREQ 信号的握手信号
38.3 到 2	MODE[1:0]	DIOR， DIOW 和 DACK	决定是否使用 DIOR 或 DIOW，它们如何协同 DACK 工作
38.0	WIDTH	DATA[15:0]	1－DATA[15:0]用作 DMA 16 位数据总线 0－DMA 仅使用 DATA[7:0](DMA 工作在 8 位模式下)

1 .典型地，此时 A0 与地端相连，微处理器通过 16 位数据线对 ISP1581 进行访问。

2. DMA 寄存器在上电复位、硬件复位、软件复位和 DMA 复位时均被复位。

表 3-4DMA 硬件寄存器（3CH）

寄存器位[1]（Hex）	寄存器位符号	ISP1581 信号	注释
3C.7 到 6	ENDIAN[1:0]	DATA[15:0]	00－MSB 在 DATA[15:8]，LSB 在 DATA[7:0] 01－MSB 在 DATA[7:0]，LSB 在 DATA[15:8]
3C.5	EOT_POL	EOT	1－EOT 高电平有效
3C.4	MASTER	DREQ,DACK DIOR 和 DIOW	信号传输方向随着 DMA 主机或从机功能的变化而改变
3C.3	ACK_POL	DACK	1－有效时 DACK 保持高电平
3C.2	DREQ_POL	DREQ	1－有效时 DREQ 变为高电平
3C.1	WRITE_POL	DIOW[2]	1－DIOW 在高脉冲时有效，下降沿处选通 0－DIOW 在低脉冲时有效，上升沿处选通
3C.0	READ_POL	DIOR[1]	1－DIOR 在高脉冲时有效，下降沿处选通 0－DIOR 在低脉冲时有效，上升沿处选通

1. DMA 寄存器在上电复位、硬件复位、软件复位和 DMA 复位时均被复位

2. DIOR 和 DIOW 可以设置成不同的方式，但 $\overline{RD}$ 和 $\overline{WR}$ 不能设置成这些方式。因此，系统必须工作在不同的硬件连接下，可能要求其它逻辑以满足 DMA 需要。

3. 见第 2 点。

4. ISP1581 固件

USB 是一种主机到从机的结构。图 4-1 所示为 ISP1581 的固件结构，表 4-1 对内部的各种文件进行了描述。

设备是不能启动任何传输过程的，它只能对主机的请求做出响应。在这种结构下，固件总是一直在等待主机命令，再根据命令去执行相应的程序。

mainloop.c 文件跟踪 USB 的中断事件，中断发生时引导它们去执行相应的程序。

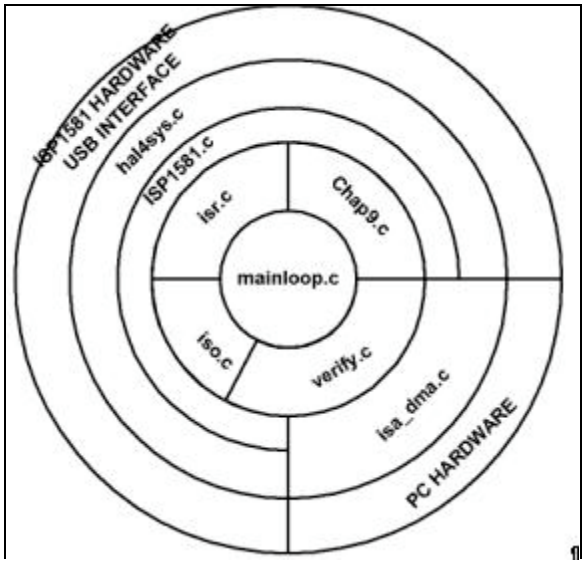


图 4-1 固件结构

表 4-1 固件结构

文件名	功能
mainloop.c	循环扫描 USB 事件；启动设备和系统的工作
isr.c	中断服务程序；对中断进行判定并将事件信息传递给其它的程序
Charp9.c	包含标准 USB 命令，用于在设备和主机之间建立一个基本连接
verify.c	包含厂商定义命令，用于验证数据完整性的显示应用
iso.c	功能与 verify.c 类似，还可用于启动同步（ISO）数据的传输
isa_dma.c	为 ISP1581 和本地存储器之间的 DMA 数据通道设置 DMA 控制器（文件名取决于所使用的微控制器。）
ISP1581.c	ISP1581 命令设置；低电平层实现了对 ISP1581 寄存器和数据端口的访问
hal4sys.c	ISP1581 和微控制器主板之间的实验板的硬件接口设置。直接与微控制器相连，这一层可以不用

4.1 流程

USB 主机和设备之间的数据流动方式有很多种。我们将数据流动的路径称为‘管道’。在 ISP1581 采样测试软件（见图 14-2）中，用到了以下几种类型的管道

- USB 控制传输的控制管道（又称缺省管道）
- Bulk-IN 和 Bulk-OUT 数据管道
- ISO-IN 和 ISO-OUT 数据管道

控制管道由 SETUP，控制 OUT 和控制 IN 端点组成。其它管道只有一个端点，每一个端点对应一个方向。

USB 主机通过控制管道利用 Setup 命令来设立和控制数据管道中的数据流。一个命令的传输分为三个阶段设定阶段、数据阶段（包含了附加的命令参数或其它信息）以及状态阶段。图 4-2 所示为一个数据结构化建立的例子，它所对应的固件程序在图 4-3 中给出。

图 4-2 一个数据结构化的建立

此例中，SETUP 包的内容是一个由主机发送的批量数据传输请求命令，该命令是由厂商定义的。主机发送一个控制 OUT 包，给定了要发送的数据字节数。设备必须配合 DMA 将传输数据保存到本地存储器中。当控制 IN 传输完成后，整个传输的设置也就结束了。这个过程的完成表明主机和设备双方都已经接收到了命令和命令成功执行后的状态。在主机和设备双方约定好后就可以启动 Bulk-OUT 传输了。

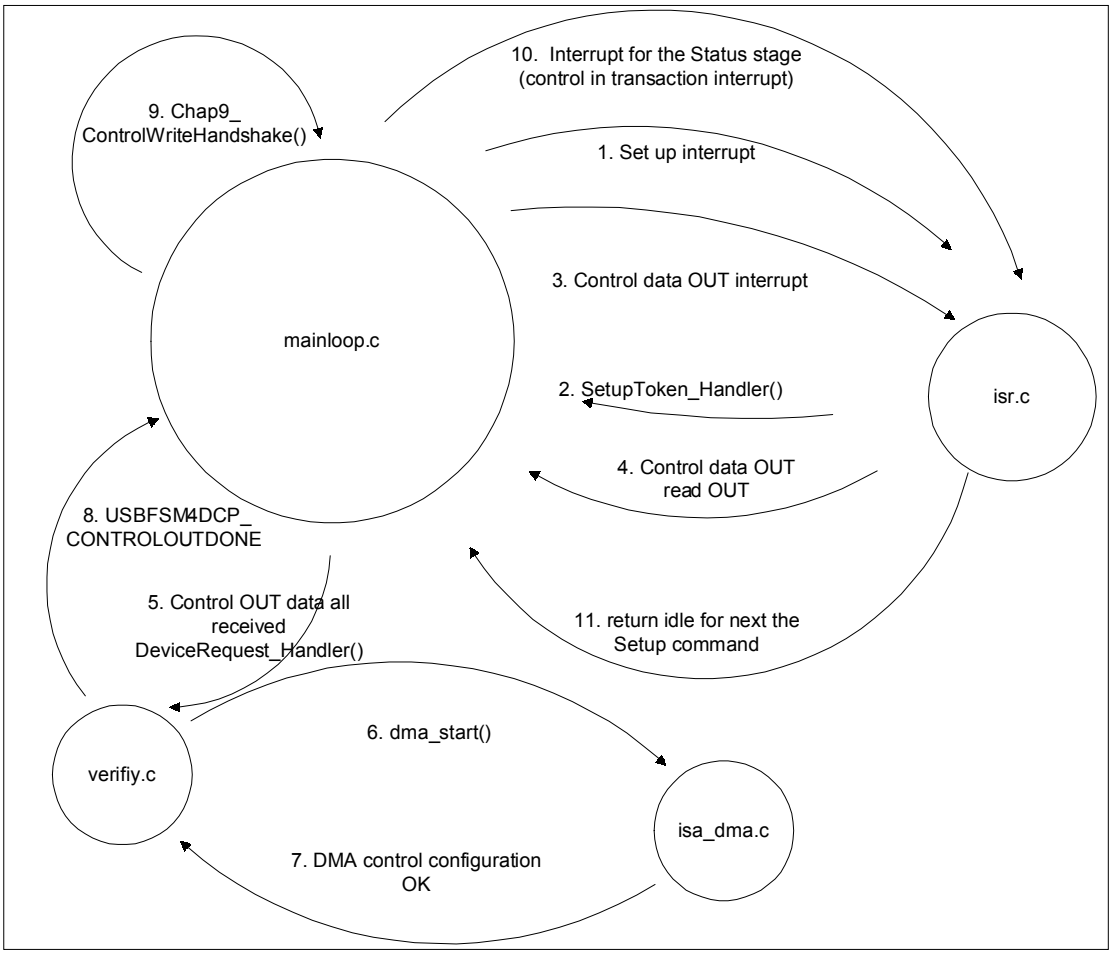


图 4-3 固件程序

4.1.1 USB 命令流程

USB 主机利用控制型传输的三个阶段来实现对设备的操作，这三个阶段为设定阶段、数据阶段和状态阶段。图 4-4 所示为 USB 命令处理的流程图。

设备跟随主机各种状态并对主机的要求做出响应。

数据阶段是可选择性的；如果没有此阶段，控制型传输被看成一个数据 OUT 命令，数据长度为零。这时设备就直接跳转到状态阶段，并且状态阶段的方向为 IN 方向。接着设备发送一个零长度包给 IN 令牌来通知主机它已接收到了命令并执行了设定请求。在 ISP1581 中，控制功能寄存器的 STATUS 位（位 1）必须置 1 来响应 SETUP 传输的状态阶段的 ACK 或 NAK 的产生。

IDLE	设备处于空闲状态，等待主机命令
SETUPPROC	设置过程，接收和解释设定命令
REQUESTPROC	请求过程，设备已执行完命令请求
DATA_IN	数据 IN，设备准备好了每一个主机请求的数据
DATA_OUT	数据 OUT，设备准备好了重新接收主机发送的数据
DATAOUTDONE	数据 OUT 完成，设备已经接收到了设定阶段要求的数据量
CTLRDHDSHAK	控制读握手，设备已完成控制读命令，可以响应状态阶段的请求
CTLWRHDSHAK	控制写握手，设备已成控制写命令，可以响应状态阶段的请求
STALL	设备声明不能完成主机命令的要求。

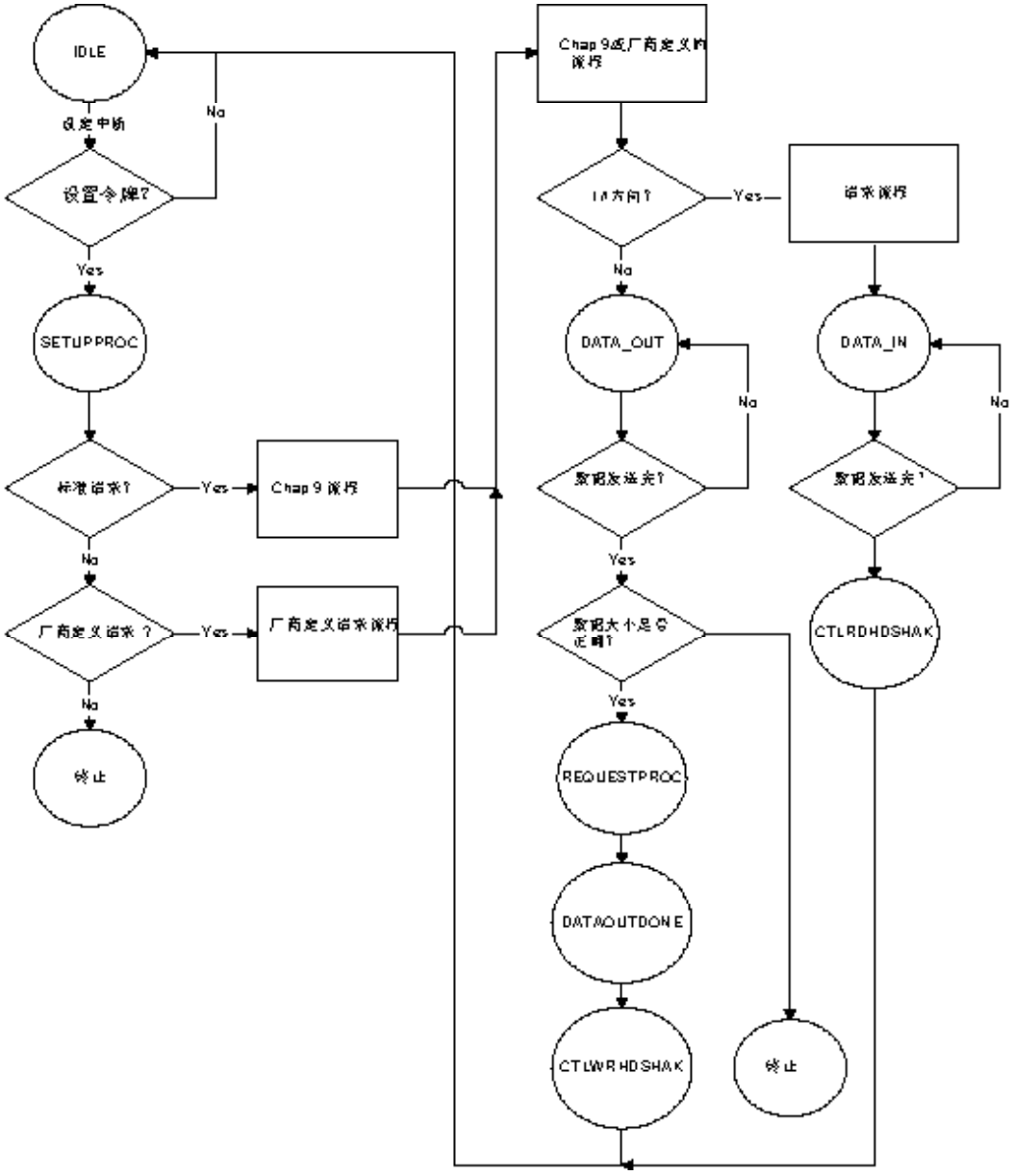


图 4-4 设置处理的设备状态

5. USB 2.0 Chapter 9 命令

主机每增加一个设备，就要执行一系列的命令。这些命令在 Universal Serial Specification Rev 2.0 列出。下面将详细给出固件命令流程。

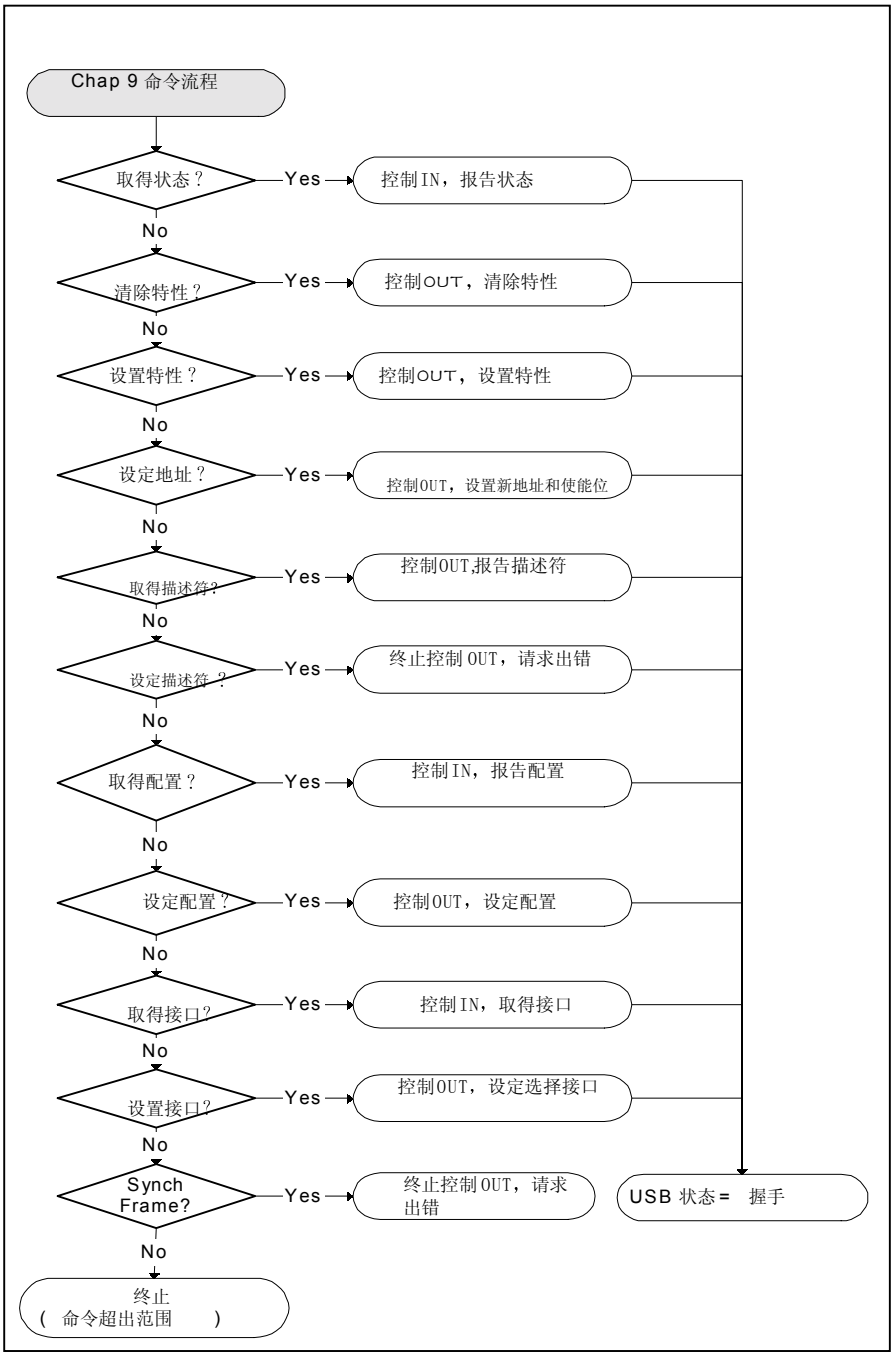
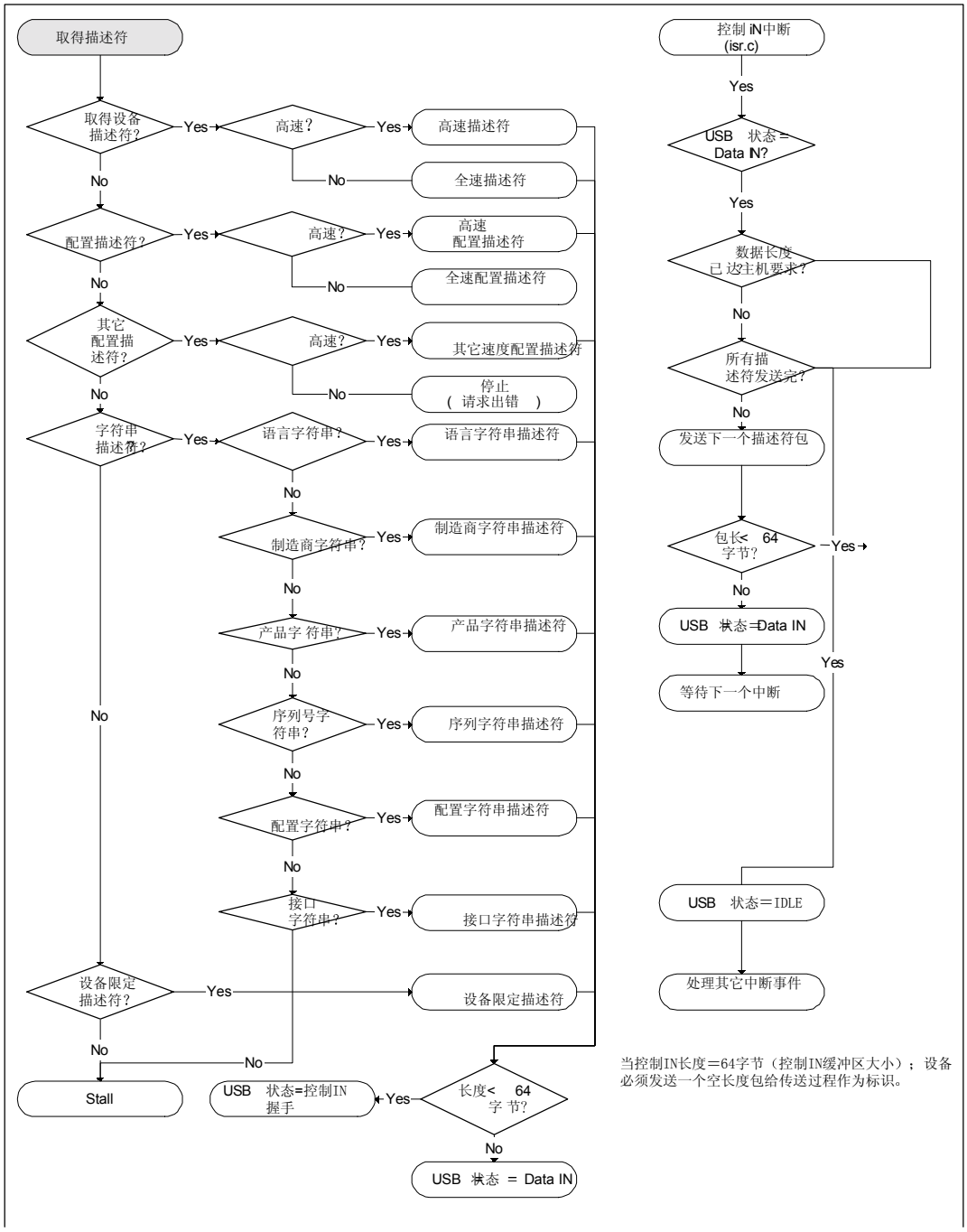


图 5-1 Charper 9 USB 2.0 标准命令处理流程图

5.1 取得描述符

USB 设备利用描述符向主机报告它的属性。描述符是一种有着固定格式的数据结构。它的起始部分是字节宽度域，描述了紧跟其后的描述符的其它所有字段的总字节长度，也标识了描述符类型。在 5-2 流程图中给出了执行主机命令和根据主机请求返回特定的描述符的详细过程。

要了解设备描述符的标准格式，可参考 USB 2.0 规范。



5-2 取得描述符流程图

5.1.1 设备描述符

设备描述符中有关于这个 USB 设备的基本信息。它包含了应用到设备和它们的配置上的全部信息。一个 USB 设备只对应一个设备描述符。

图 5-3 中描述的传输体现了 Philips ISP1581 评估板的设备描述符。设备描述符有高速能力，属于 2.0 版本（0200H，第三和第四个字节的次序颠倒了）。

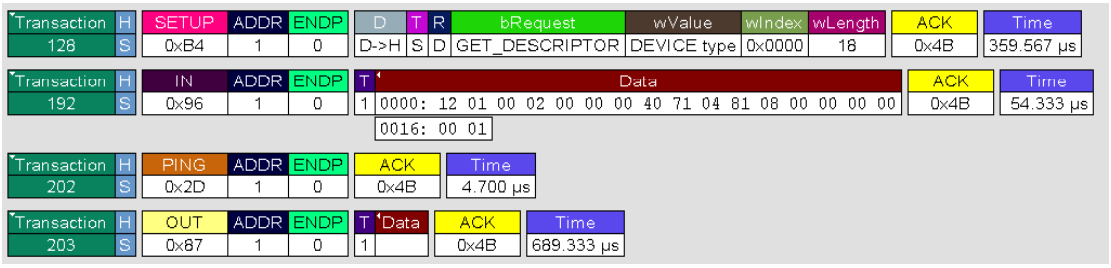


图 5-3 设备描述符

从图 5-3 给定的设备描述符中可以得到以下信息

设备 USB 2.0

厂商 ID 0471(Philips)

产品 ID 0881(Philips 显示评估设备)

这个设备没有分类。

5.1.2 设备限定描述符

设备限定描述符包含的信息是一个高速设备工作在其它速度下发生的改变。

5.1.3 配置描述符（高速）

配置描述符（见图 5-4）提供了一个指定设备的配置信息。ISP1581 评估板包含 2 个中断端点和 2 个 ISO 端点。另外，由于带宽的限制，含有 ISO 端点的设备还必须支持选择性接口。这样，在数据传输要求低带宽的情况下，USB 主机就可选择没有 ISO 接口的缺省接口或恰好需要低带宽的接口工作。

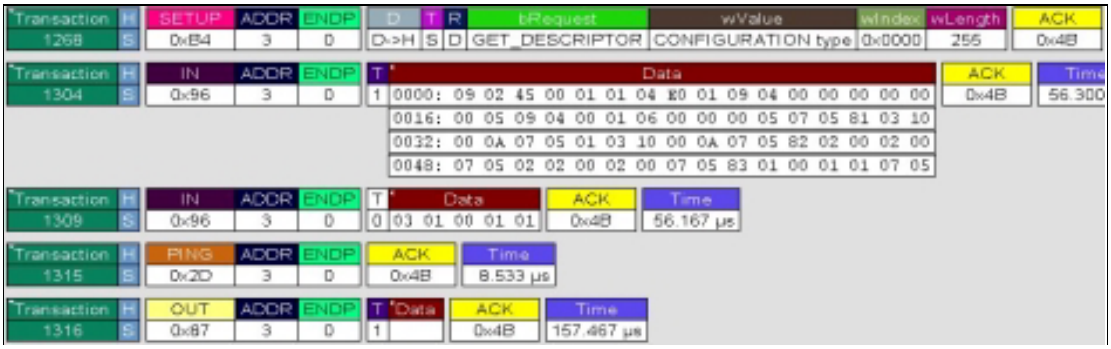


图 5-4 配置描述符

配置描述符包括接口描述符和端点描述符。ISP1581 评估板含有一个配置和一个可选择设置的接口。其中，缺省接口没有端点，选择性接口由 6 个端点组成。这 6 个端点在表 5-1 中列出。

配置	1
接口	1（带可选择设置）
端点号	6

表 5-1 端点配置（在可选择设置情况下）

端点描述	数目	方向	大小
中断端点	1	OUT	16 字节
中断端点	1	IN	16 字节
批量端点	2	OUT	512 字节
批量端点	2	IN	512 字节
ISO 端点	3	OUT	256 字节
ISO 端点	3	IN	256 字节

5.2　设定地址

执行这个请求就是给设备一个新地址。由 ISP1581 决定这个新地址是否有效。设备一旦接收到 IN 令牌的 ACK（状态阶段）信号就要立即将新地址发送出去。

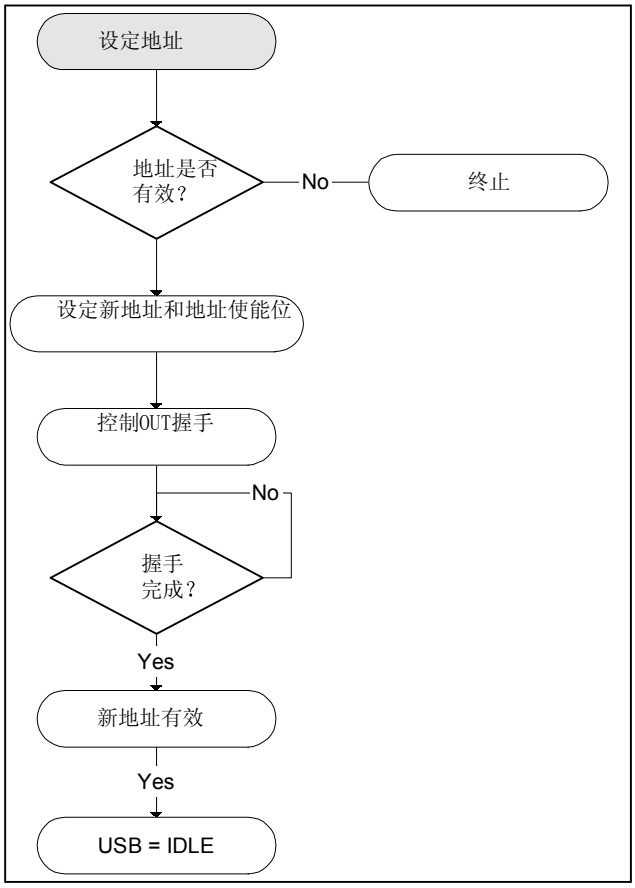


图 5-5　设定地址流程

5.3 设定特性

接受设定特性请求的可以是设备、接口或端点。这里不支持评估板的接口特性，因此对设定接口特性的请求不响应。作为一个高速设备，评估板可以完全实现 TEST_MODE 特性，这样就可以方便进行高速电测试和其它相关测试（详细情况见图 5-6）。

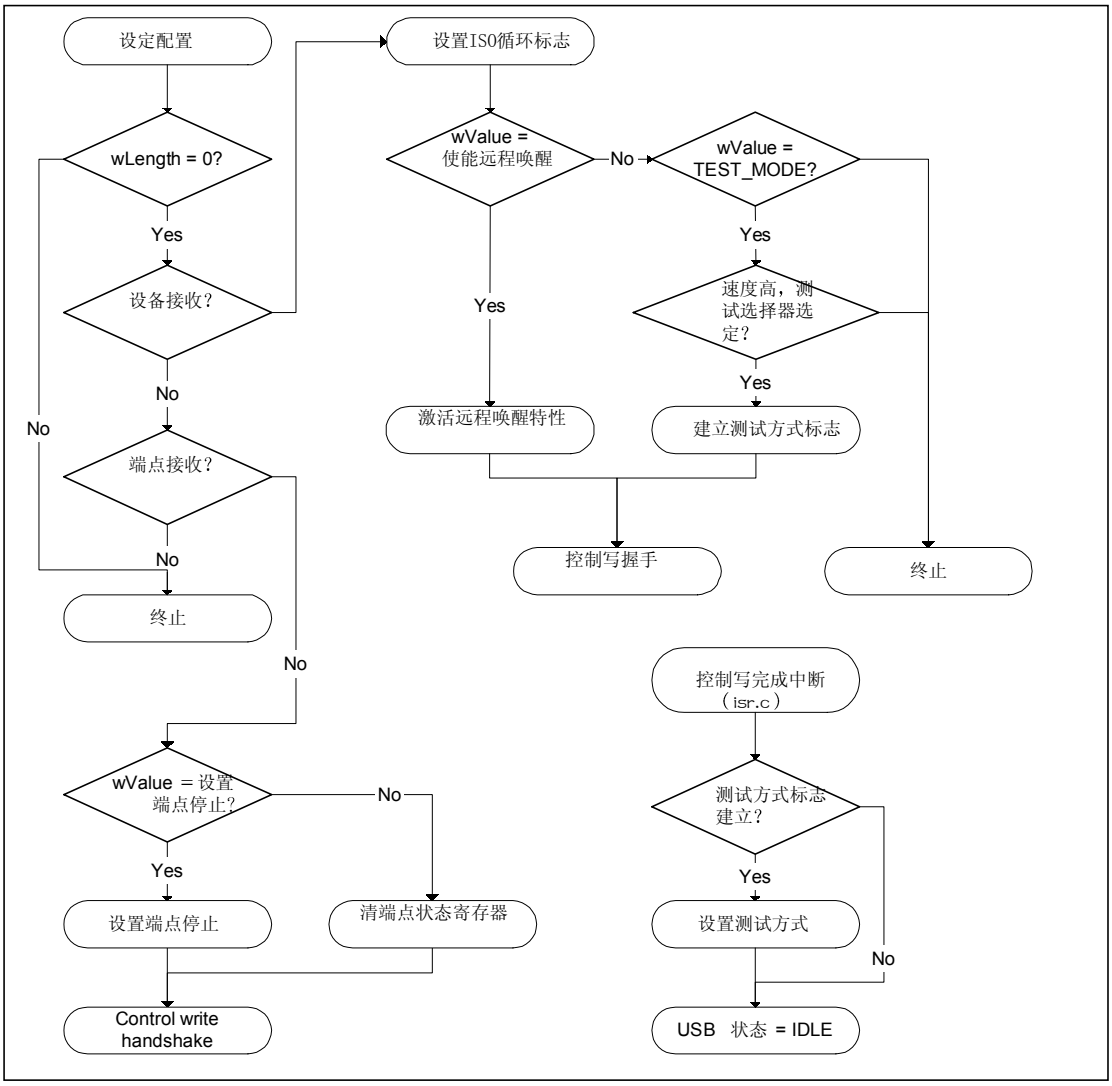


图 5-6　设定特性流程图

6. 设备的启动和高速配置

一个设备的配置在硬件复位和软件复位时被清除。总线复位将清除端点配置。因此，ISP1581 在每一次复位都必须重新启动。

当发生上电复位或软件复位时，设备将先断开再连接（见图 6-1）。

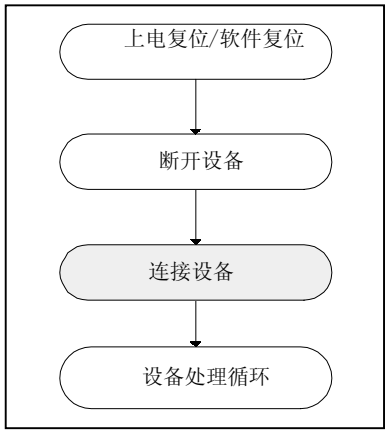


图 6-1 复位过程流程图

图 6-2 所示为引起 ISP1581 初始化的外部事件。

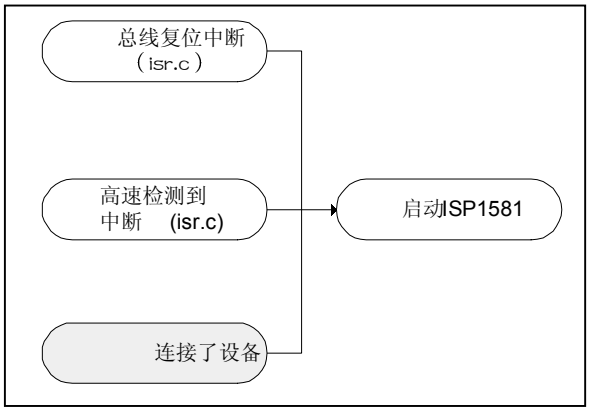


图 6-2 ISP1581 初始化流程图

总线复位后的初始化如图 6-3 所示。

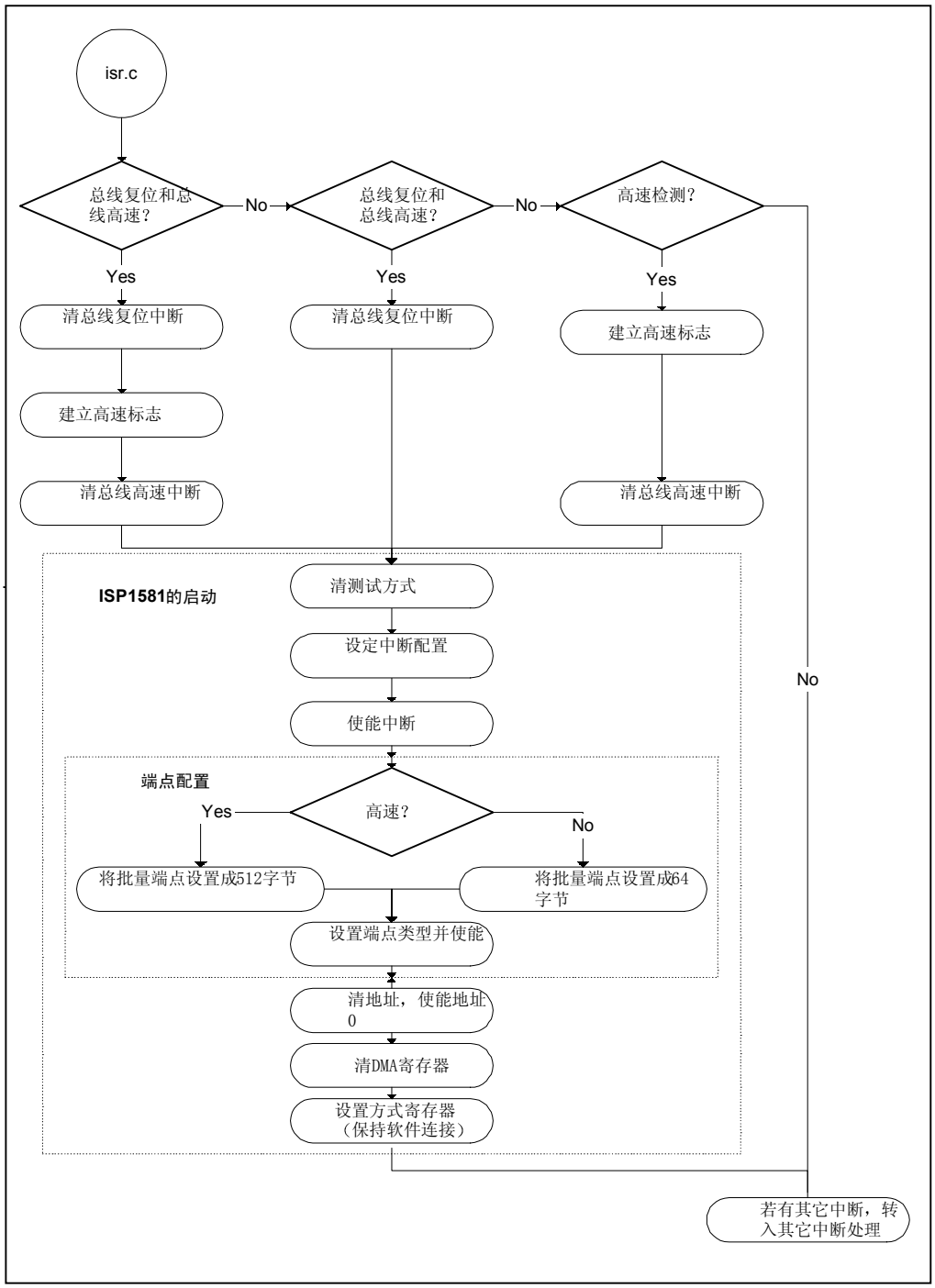


图 6-3 设备配置和高速配置流程图

7. DMA 传输设置

7.1 DMA 复位

DAM 复位用于终止一次未结束的 DMA 传输，它将 DMA 恢复到上电时的默认值。由于在通用 DMA（GDMA）模式下没有 DMA 停止命令，所以在必要时必须要采用其它方法来重新启动一次新的 DMA 传输。

DMA 复位命令发布后，IN 端点的有效数据包仍保存在缓冲区中等待发送到主机。若为 OUT 端点，数据包中的所有数据都被读出后再全部清除，否则，所有数据包的数据都保留在缓冲区中。

DMA 清除缓冲区命令和使缓冲区有效命令分别用于丢弃还未读完的最后数据包或使缓冲区中的部分数据有效。

7.2　连续 DMA 传输命令设置

在传输过程中，要特别注意主机和设备之间的实时跟踪数据传输，因为主机的非模块化数据读/写功能会将其本身所发送的数据根据执行的状态也保存到几个缓冲区中去。比如，在 ISP1581 抽样测试程序中（见图 15-2），每一个 DMA Bulk OUT 命令都在 SETUP 令牌包之后执行，在应用中主机启动一个 Setup 的接连操作之后紧跟着执行 Bulk OUT 命令就会模拟出一个连续的 DMA 流。

尽管此时在它的缓冲区中执行的是上一次 DMA 命令的批量数据传输，但从设备的角度来看是要启动一个新循环的命令设置。因此，固件编程者要对这种情况加以防范。

7.3　带有 SH-3[1]ISP1581 DMA 接口

ISP1581 实行的是基于 I/O 口的访问模式。因此，为了满足 SH-3 的要求,必须对它的信号作一些调整（见表 7-1）。

表 7-1SH-3 信号连接

使用 DIOR 和 DIOW 作为 DMA 的读和写选通信号		
ISP1581 信号	SH-3	注释
$\overline{CS}$	DMA 传输时不必访问	—
DIOR	与 SH-3 的读选通相连	—
DIOW	与 SH-3 的写选通相连	—
DREQ	减少到小于 SH-3 的两个 CLKIO	见图 7-1
使用单独 DACK 模式		
ISP1581 信号	SH-3	注释
$\overline{CS}$	DMA 传输时不必访问	—
DIOR	与 V_{CC}　3.3V 相连	DMA 方向取决于 DMA 命令
DIOW	与 V_{CC}　3.3V 相连	—
DREQ	减少到小于 SH-3 的两个 CLKIO	见图 7-1

1.如果没有特别说明，SH-3 在本文中是指 7709A。

7.3.1　DMA 模式配置

SH-3 实现了存储器到存储器的 DMA 方式。通常要为 DMA 源和 DMA 目的地都设置片选信号。为了避免混淆，可以将 ISP1581 划为两片存储区域。一片用作 PIO 访问，另一片用作 DMA 访问。系统译码电路只在使用 PIO 存储器时认可 $\overline{CS}$ 有效。当对 DMA 进行操作时，直接给 DMA 区域设置地址而不必再使 $\overline{CS}$ 信号有效。

ISP1581 请求信号会持续到 DMA 传输的最后一次访问（见图 7-1）。ISP1581 的 DREQ 不能与 SH-3 直接相连，因为两者的时序不匹配。不管 DMA 操作启动与否，在一个时钟周期的开始 SH—3 就已经采样了两次（也就是说，它可以启动两次 DMA 循环）。即使在第一次传输中 DREQ 无效，SH—3 仍会进行上述操作。在突发模式下，在此期间还甚至可能另外有读/写周期的出现。

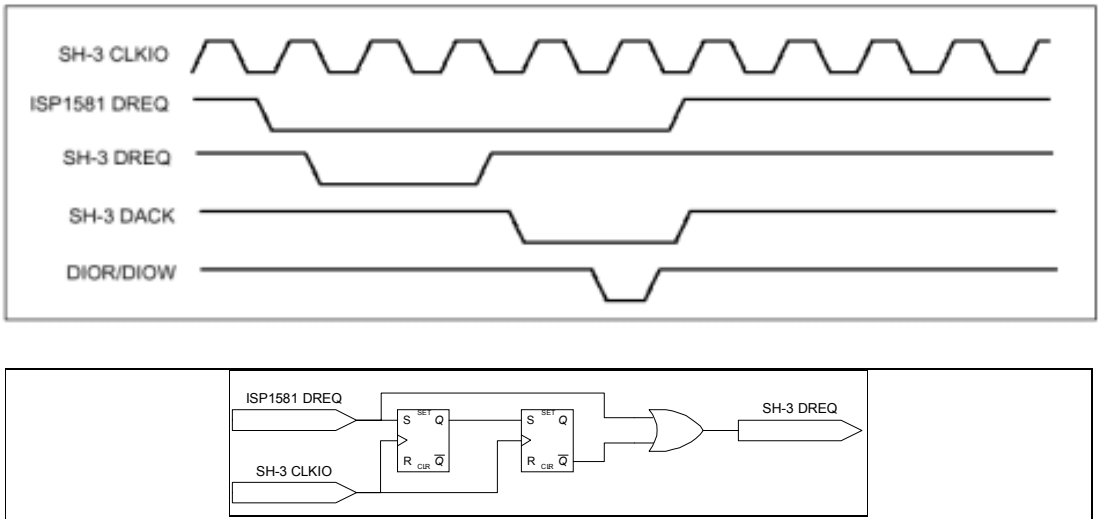


图 7-1　　DMA 请求信号从 ISP1581 到 SH-3

SH-3 必须被配置成周期挪用模式，在这种模式下 DMA 传输能方便地使处理器产生一个空闲周期（边沿方式或电平方式均可，在周期挪用模式下两者没有区别）。

ISP1581 也必须通过编程将其设置成单周期模式，在这种模式下 DREQ 在 DMA 传输过程中无效，传输结束后另一次传输启动时变为有效。

8. ISP1581 停止处理

8.1 停止功能

批量或中断端点支持停止特性的设定和清除（即 STALL 功能）。停止的端点将一直保持 STALL 状态直至主机通过控制管道发布一个清除停止特性命令。如果端点的停止特性被解除（UNSTALL），它将清除缓冲区（双缓冲区）并使数据触发位复位成‘0’（PID DATA0）。在 UNSTALL 后的第一次数据交易中，ISP1581 接收带有 PID DATA0 的数据包给一个 OUT 令牌并发送一个带有 PID DATA0 的数据包给一个 IN 令牌。

8.1.1 停止中断

若端点被配置成单独 ACK 模式，停止状况发生时没有对应的中断产生。因为端点的停止不会对固件产生任何影响，只需要主机发布一个 CLEAR_FEATRUE 请求就可将停止状况解除。

在调试模式下就能够产生中断。但是，这个中断并没有用来提醒用户现在正在处理的是停止中断的标识，只有通过在该停止端点没有任何数据交易之前固件也没有进行其它数据的处理而且也不处于空闲等待的情况来判定此时正在处理该中断。端点后来所接收的封包会将以前的信息覆盖。

8.1.2　端点的停止和退出停止状态

停止一个端点可以随时被停止，通过设置端点状态寄存器的 STALL 位或主机向端点发送设定端点停止特性的请求两种方式都可将端点设定成停止状态。

例 ISP1581_SetEndpointStatus（bEPIndex， epctlfc_stall）；

停止功能优先级最高。因此，即使同时有 IN 端点缓冲区出现数据或 OUT 端点为空的情况主机也不理会而去接收端点停止。

退出（清除）将端点的 STALL 位清除。另外，若缓冲区中还有数据，就要先将端点禁止再让端点使能来清除缓冲区。

Example:

```
void Chap9_ClearFeature(void)
{
        unsigned char endp;
        unsigned char bRecipient  = ControlData.DeviceRequest.bmRequestType    & USB_RECIPIENT;
        unsigned short wFeature  = ControlData.DeviceRequest.wValue;
        unsigned short wEPCFG;
        unsigned char dir = ControlData.DeviceRequest.bmRequestType    & SB_REQUEST_DIR_MASK;
        if(dir)
                Chap9_StallEP0InControlRead();

        if( ControlData.DeviceRequest.wLength   == 0 )
        {
          switch(bRecipient)
          {
              case USB_RECIPIENT_DEVICE:
                        //....
              case USB_RECIPIENT_ENDPOINT:
                 if( ControlData.DeviceRequest.wIndex   & USB_ENDPOINT_DIRECTION_MASK)
                        endp  = (ControlData.DeviceRequest.wIndex*2    + 1);
                else
                        endp  = (ControlData.DeviceRequest.wIndex*2);

                if(wFeature  == USB_FEATURE_ENDPOINT_STALL)
                {
                // Clear the data toggle bit to (set to 0) and clear buffers before clear stall of
                // the endpoint.
                        wEPCFG  = ISP1581_GetEndpointConfig(endp);
                        ISP1581_SetEndpointConfig(endp,   0); // disable endpoint  *
                // Enable endpoint,  clear the buffer and set the data toggle bit to 0.
                        ISP1581_SetEndpointConfig(endp,   wEPCFG);
                        ISP1581_SetEndpointStatus(endp,   0);   // clear the Stall condition of the
                                                                //endpoint.
                        Chap9_ControlWriteHandshake();
                }
                else
                {
                        Chap9_StallEP0InControlWrite();
                }
              break;
              default:
                        Chap9_StallEP0InControlWrite();
                        break;
            }
          }
          else
          {
              Chap9_StallEP0InControlWrite();
          }
}
```

注：一个端点的禁止和再次使能并不会影响端点的存储（缓冲区）分配，也不会干扰其它端点的正常工作。

8.2 协议终止

协议终止仅使用于控制管道。ISP1581 只支持一个控制管道，它由 SETUP，控制 OUT（端点索引 0 OUT）端点和控制 IN（端点索引 0 IN）端点。控制管道又被称为缺省管道。

尽管所有的控制 OUT 和控制 IN 端点在端点状态寄存器中都对应一个 STALL 位，但在物理上这些位共用一个存储单元。ISP1581 内部只有一个寄存器位能够从不同的入口进行访问。可以将这一位设置到控制 OUT 或控制 IN 中。设置好后，从主机到端点 0（OUT 或 IN）的 IN 和 OUT 令牌就会有一个停止应答。这个停止条件会保持到数据交易的启动。

在启动数据交易的过程中，控制 OUT 和控制 IN 端点的数据被刷新，两者的数据触发位置 1。

控制管道没有终止特性（Stall 功能）。

9. 零长度包和短长度包有效

9.1 OUT 方向

对于 OUT 方向的数据传输，利用数据计数寄存器用来记录接收到的数据包长度（以字节形式）。若包长度为奇数个字节，则设备配置成 16 位模式，读出的最后一个字节存放在低半部分，高字节由任意数填充。

9.2 IN 方向

对于 IN 方向的数据传输，有两种方法可使得零长度包和短长度包有效。

9.2.1 使用数据计数器

数据计数器是一个字节计数器。利用它记录下数据包的字节数，再选择一个数据端口将数据包的数据写入，当写入的数据长度达到了数据计数器的值，数据包自动有效。若计数器的值是奇数，意味着发送的

数据包的长度为奇数个字节，利用数据包的最后一个写选通信号来使低字节有效而舍弃高字节。

9.2.2 使用有效缓冲区命令

有效缓冲区命令仅用于短长度包的情况。该命令发布后短长度包就有效，同时包的长度被记录到 ISP1581 中。

注如果写入的数据量与缓冲区长度相等，并且数据后面带有一个缓冲区有效命令，那么缓冲区有效命令这个额外的零长度包也有效。

9.2.3 零长度包

使用数据计数器

向数据计数器中写入 0 可使零长度包有效。

使用有效命令

使一个不含任何数据的缓冲区有效就是使一个零长度包有效。如果写入数据端口寄存器的数据达到了缓冲区的长度并且立刻发布一个有效缓冲区命令，则满长度包和零长度包就都有效。这样所得到的只是一个包，排列顺序是满长度包在前，零长度包在后。

10. 有效缓冲区

如果 IN 端点配置成双缓冲区模式，固件只能通过读中断位来判断是否还有存放新数据的空间。所以，一般不提倡一次填充缓冲区的数据包超过一个满包。固件的计数器必须不间断地对写入的数据量进行记录。如果出现写入的数据多于一个满数据包的长度但另外一个缓冲区仍为空的情况，ISP1581 就无法进行下一步的处理。

11. 配置端点

对端点进行配置有以下几种情况

1. 禁止从 2 OUT 到 7 IN 的所有端点。
2. 配置从 2 OUT 到 7 IN 所有端点的缓冲区长度。（未使用的端点给‘0’。）
3. 设置从 2 OUT 到 7 IN 的端点配置寄存器。（未使用的端点置 0。）

若一个端点禁止而它的缓冲区长度不为零，端点对应的 RAM 保留。但是所有端点的 RAM 总量不能超过总的 RAM 大小，否则，就不执行端点的最后 RAM 空间的分配。

12. 上拉电阻

1.5KΩ的上拉电阻与 $V_{CC(3.3)}$输出相连以便于检测到下行端口对高速或全速设备的插入。如果 V_{BUS} 不供电，电阻就不能给 D+提供电流。比如，当下行端口工作在掉电模式，微控制器就不断地查询 V_{BUS}，V_{BUS} 一旦上电，就将其与上拉电阻断开。

上拉电阻一端与 RPU 相连，另一端与 3.0～3.6V DC 相连。若 ISP1581 转换到高速模式，就必须将上拉电阻移开。因此，一般来说电阻不直接与 V_{BUS} 相连。而是把它连接到一个电压源上，这个电压源来自或受控于 V_{BUS}，这样来实现 V_{BUS} 移开时上拉电阻也能随之断开与它的连接。

13. USB 2.0 测试

对于 IN 令牌，一个 USB 2.0 高速设备必须支持四种测试方式 Test-J、Test-K、Test-Packet 和 Test-SEO-NAK。

USB 2.0 主机通过 SET-FEATURE 命令来启动测试方式。设备在执行完该设置命令后进入 3ms 的测试模式。

对于测试包来说，固件必须先确定控制 IN 端点的测试模式，再对测试使能位进行设置。一旦设定好了测试方式，设备断开与主机的连接，通过人工测量信号来终止测试过程。USB 2.0 测试的细节如流程图 13-1 所示。

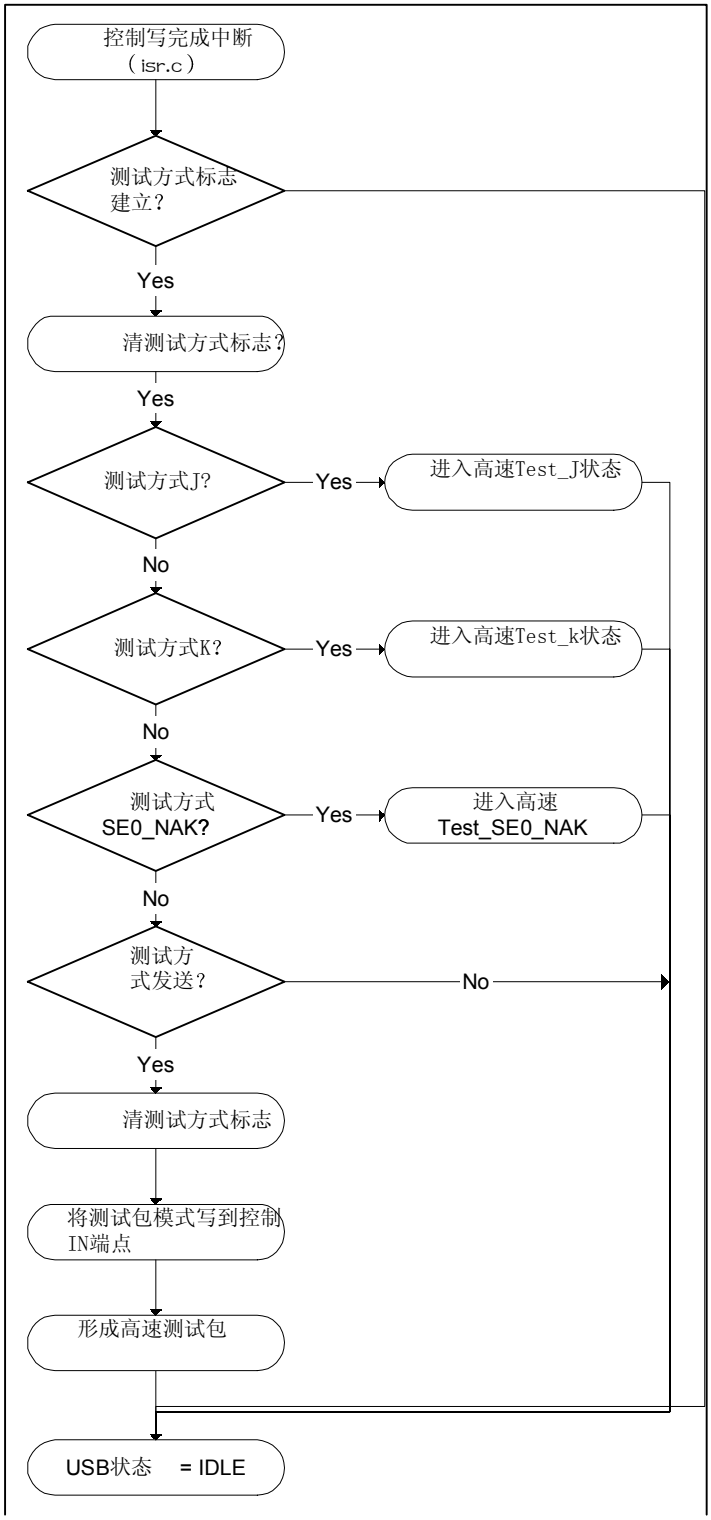


图 13-1　测试方式流程图

测试模式结构

```
USB_TESTPACKET   bTestPacket  =
{
//      0x00,  0x00,
//      0x00,  0x80,  // SYNC pattern  will  be  added  by  ISP1581
        0xc3,
        0x00,  0x00,
        0x00,  0x00,
        0x00,  0x00,
        0x00,  0x00,
        0x00,  0xaa,
        0xaa,  0xaa,
        0xaa,  0xaa,
        0xaa,  0xaa,
        0xaa,  0xee,   //aa*4
        0xee,  0xee,
        0xee,  0xee,
        0xee,  0xee,
        0xee,  0xfe,   //ee*4
        0xff,  0xff,
        0xff,  0xff,
        0xff,  0xff,
        0xff,  0xff,
        0xff,  0xff,   //FF*11
        0xff,  0x7f,
        0xbf,  0xdf,
        0xef,  0xf7,
        0xfb,  0xfd,
        0xfc,  0x7e,
        0xbf,  0xdf,
        0xef,  0xf7,
        0xfb,  0xfd,
        0x7e,
        0xb6,  0xce    // CRC
//      0xff,  0xf7    // Bit  stuff  as  end  of  the  packet  added  by  ISP1581.
```

14. 应用驱动程序和程序代码

14.1 登录应用驱动程序

若设备报告指明该设备是一个 Philips 的待测试设备，就登录到测试驱动程序（见图 14-1）。驱动程序按照设备配置描述符报告来建立管道。

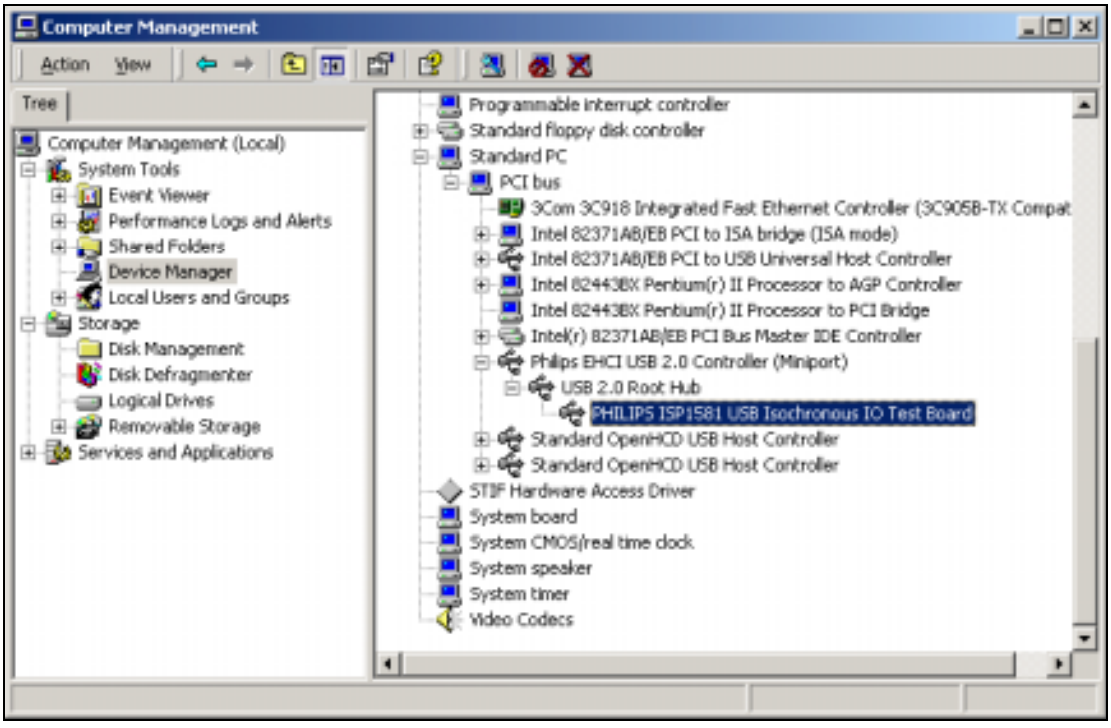


图 14-1　　设备驱动登录

14.2 测试程序应用

利用 Philips 验证程序进行测试（见图 14-2）。在固件中还有几个厂商定义命令（见图 15）。

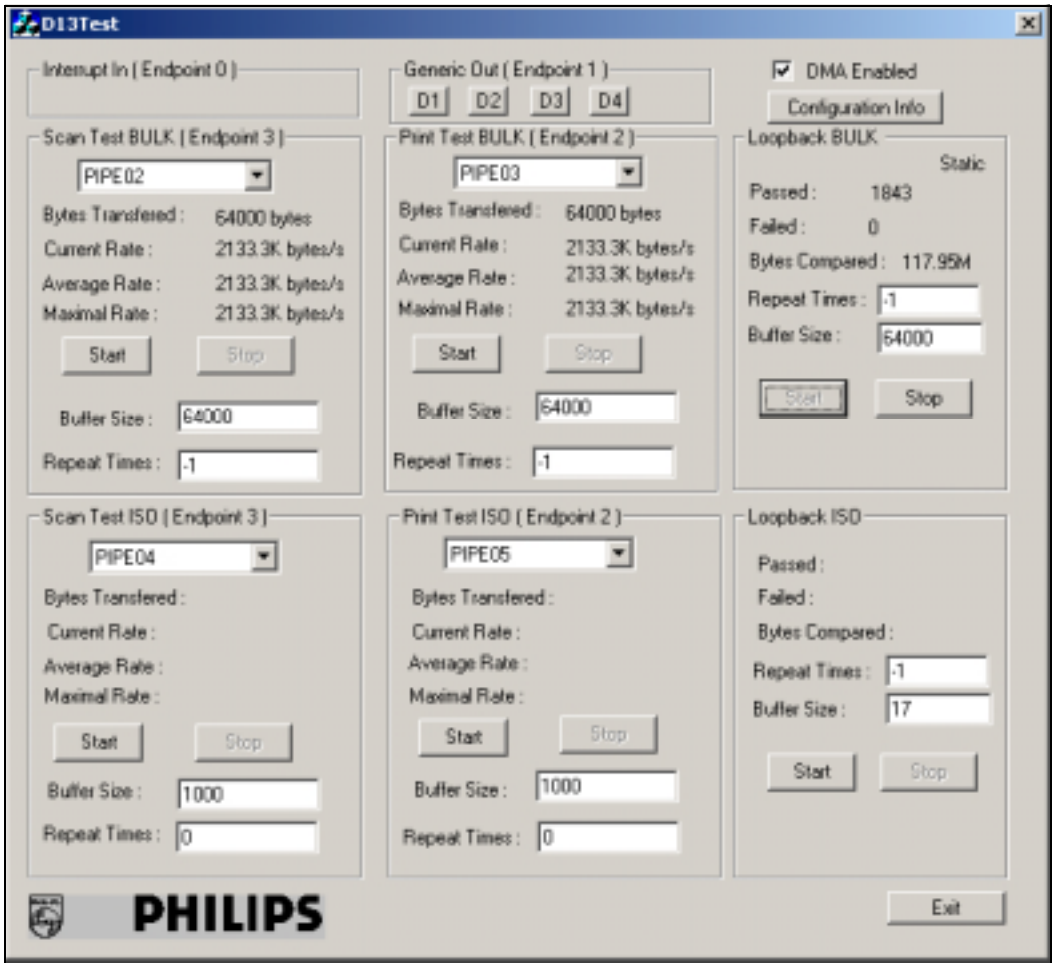


图 14-2 测试程序

15. 厂商定义命令处理固件程序

厂商定义了几个命令用来设置 Bulk 和 ISO 数据。图 15-1 所示为厂商定义命令处理的流程图。可将这些命令分类如下

- 取得固件版本（wIndex＝GET-FIRMWARE-REQUEST）
- 设定和取得 TWIN 配置（wIndex＝TWIN-CINFIGURATION）
- 将批量数据写到设备中或从设备中读出数据（wIndex＝SETUP-DMA-REQUEST）
- 设定 ISO 传输（bmRequest＝EnableIsoMode）

批量数据传输的 wIndex 列表

```
#define     SETUP-DMA-REQUEST                   0X0471
#define     GET-FIRMWARE-VERSION                0X0472
#define     TWIN-CINFIGURATION                  0X0473
#define     TWIN-CLEAR-CURRENT-FILE             0X0006
#define     TWIN-CURRENT-FILE-INDEX             0X0001
#define     TWIN-CURRENT-FILE-SIZE              0X0002
#define     TWIN-CURRENT-FILE-INDEX-LENGTH      0X01
#define     TWIN-CURRENT-FILE-SIZE-LENGTH       0X04
```

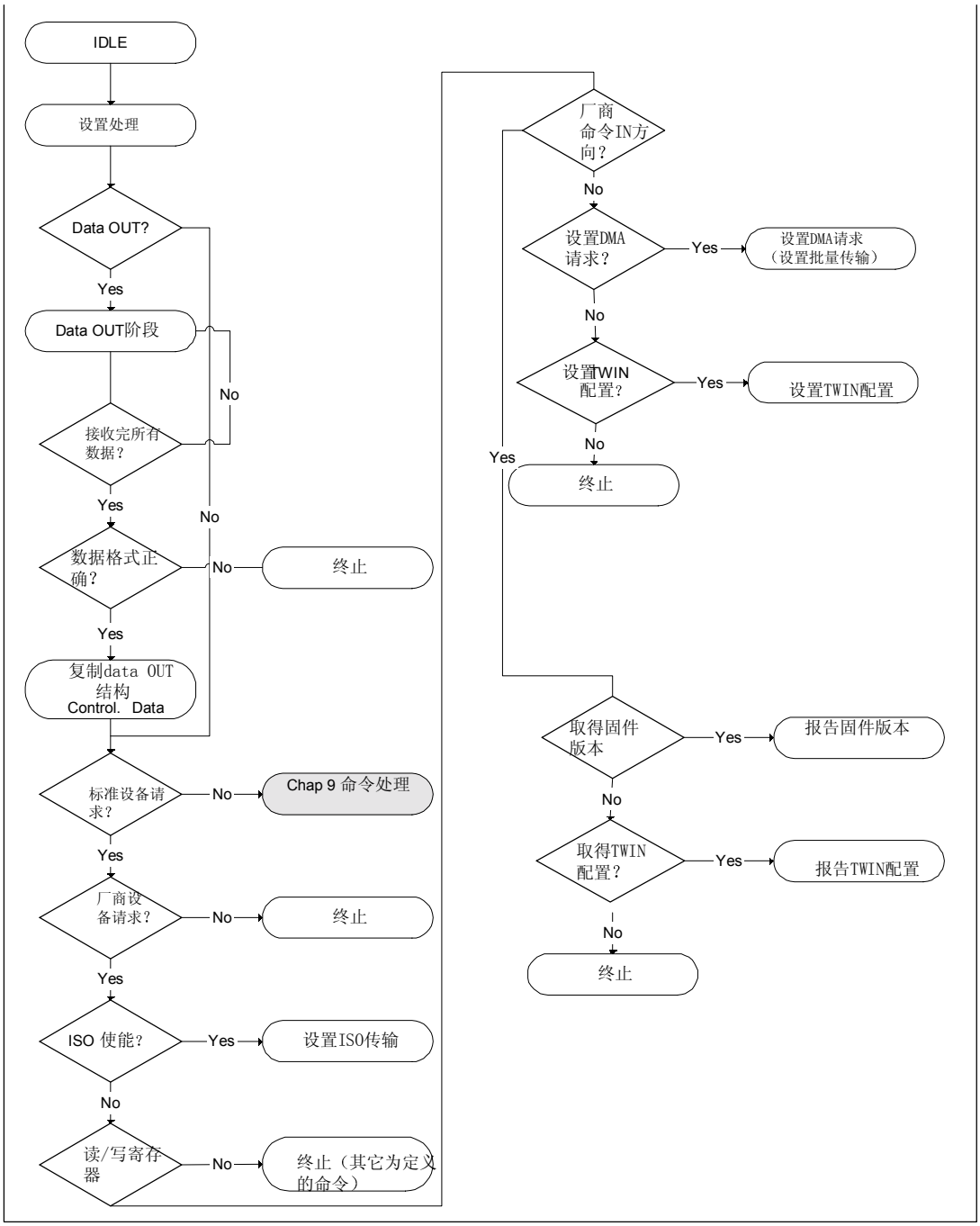


图 15-1　　厂商定义命令处理流程图

15.1 取得固件版本

一旦测试软件开始运行，它首先发送一个设备固件版本的厂商请求用以判定所使用的评估板和选择要测试的功能。图 15-2 所示为取得固件版本命令的流程图。

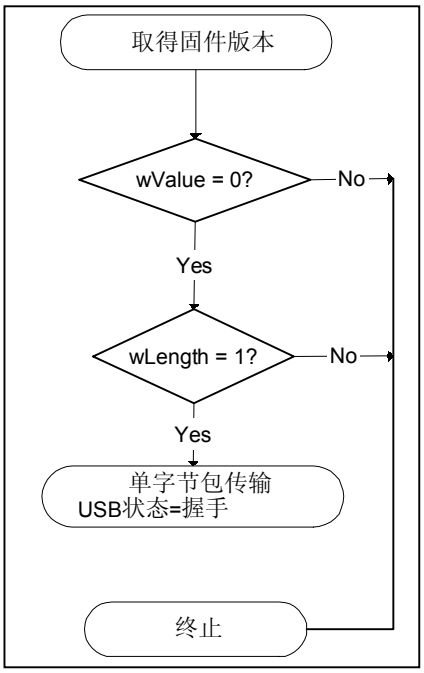


图 15-2　取得固件版本命令流程图

15.2　取得和设定 TWIN 配置

批量数据传输之前，测试软件从配置结果开始执行（类似于配置期间扫描仪的工作）。整个数据传输的过程分为几个阶段，每个阶段的传输数据量可达 64K 字节。为了验证，还能够通过软件重新取出设定的配置。取得 TEIN 配置和设置 TWIN 配置的流程图分别在图 15-3 和图 15-4 中给出。

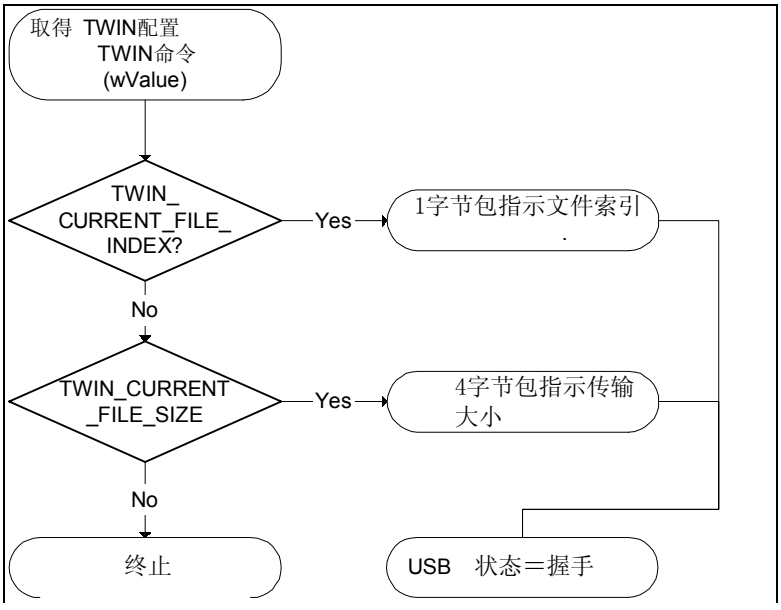


图 15-3　　取得 TWIN 配置流程图

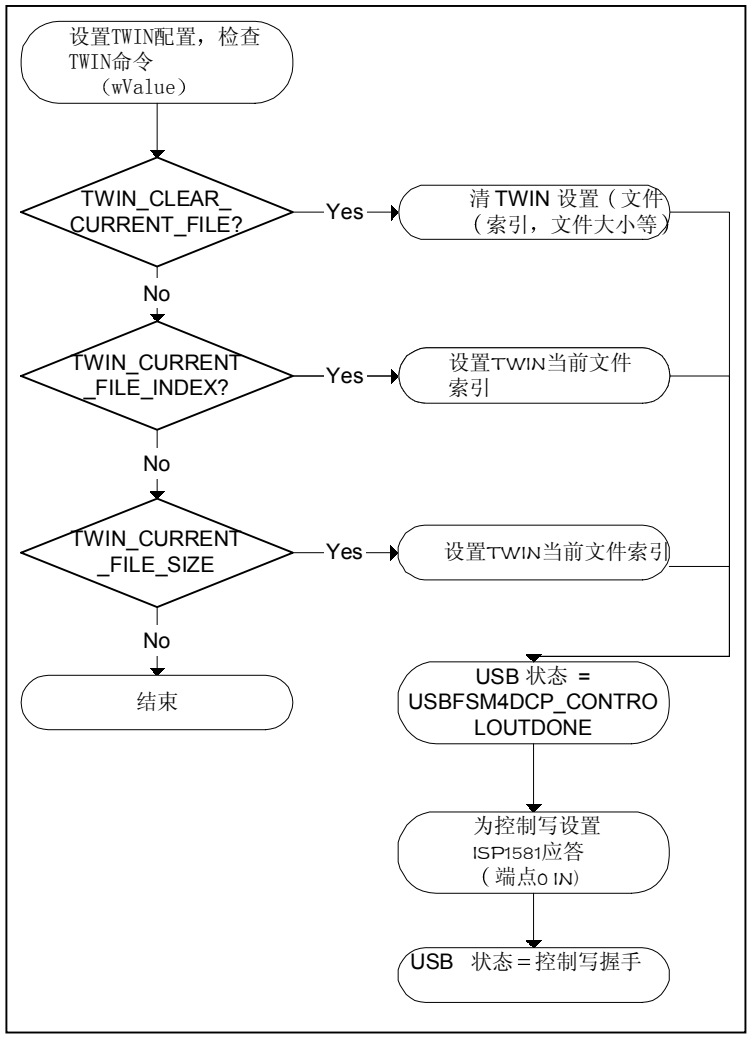


图 15-4 设置 TWIN 配置流程图

15.3 设置 DMA 请求（设置批量传输）

设置 DMA 请求包括以下信息

- 数据大小
- 数据传输方向；IN 或 OUT
- 页索引；指示传输的页的次序
- 数据定位；支持多页传输，每一页有一个文件索引

图 15-5 所示为设置 DMA 请求流程图

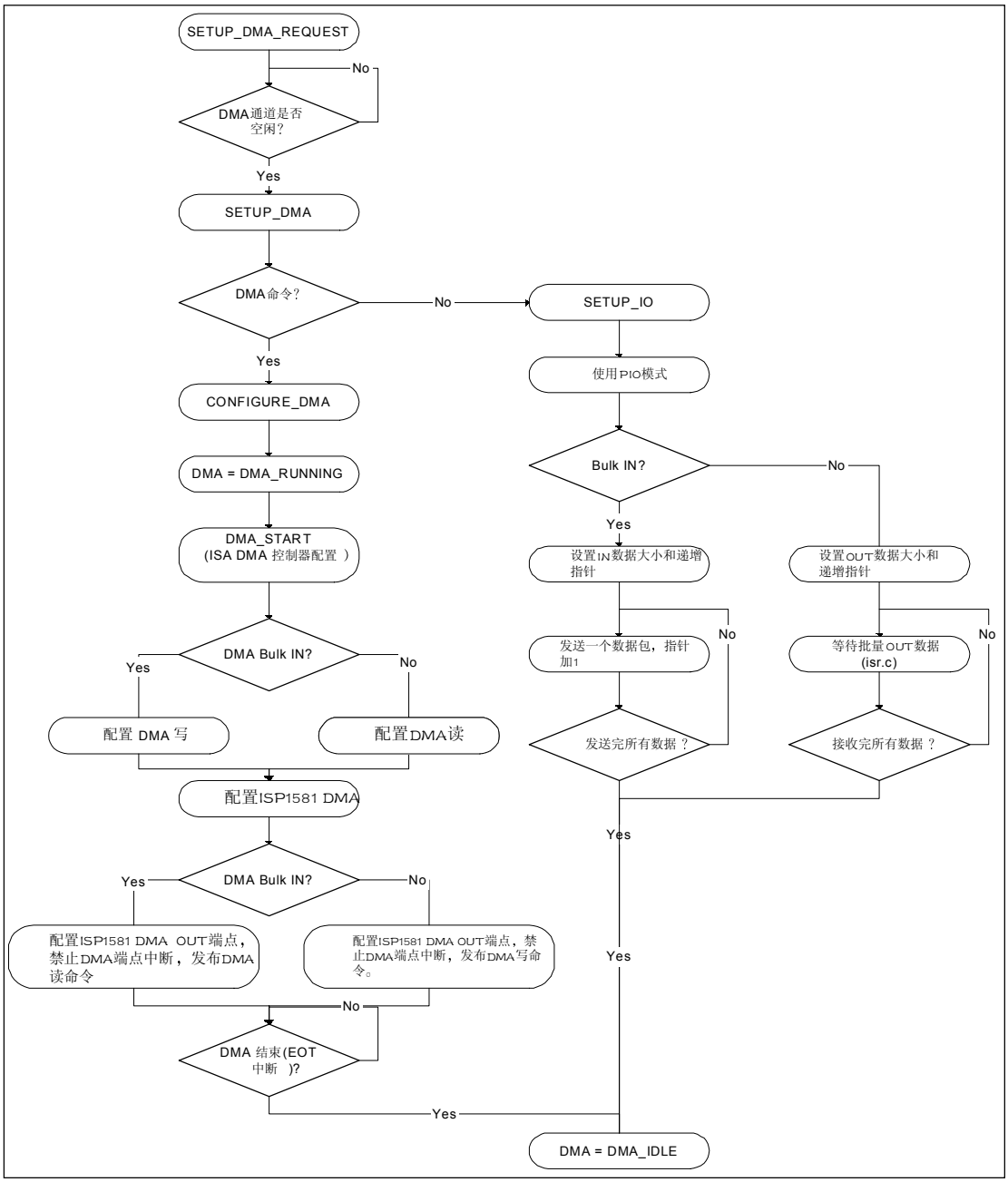


图 15-5　　设置 DMA 请求流程图

15.4　设置 ISO 传输

设置 ISO 传输的细节如图 15-6 所示。

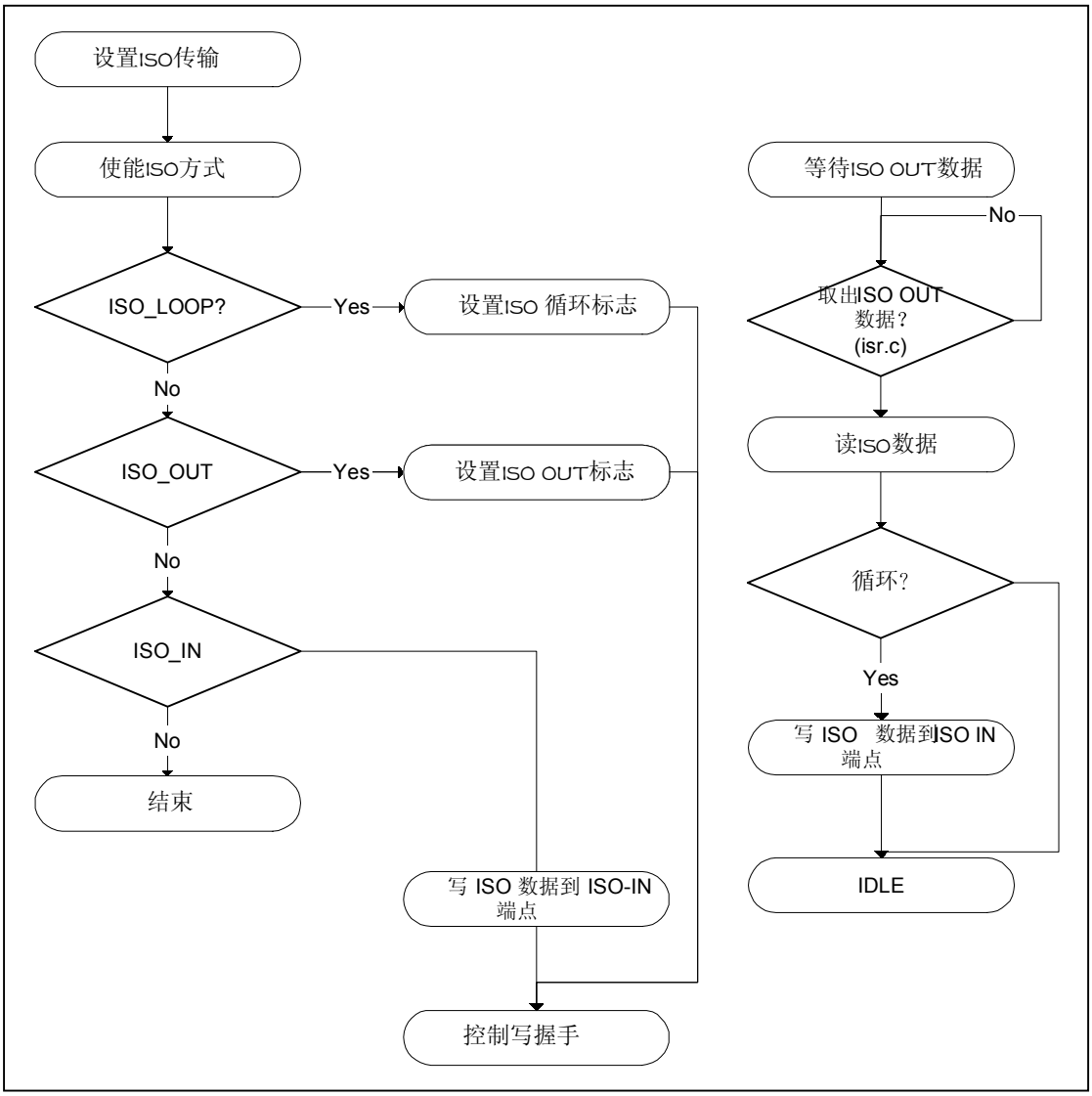


图 15-6 设置 ISO 传输流程图

16. 应用详情

厂商定义命令在测试应用中的详细情况在以下部分给出。

16.1 取得固件版本

下图描述的传输说明了取得固件版本命令的处理过程。

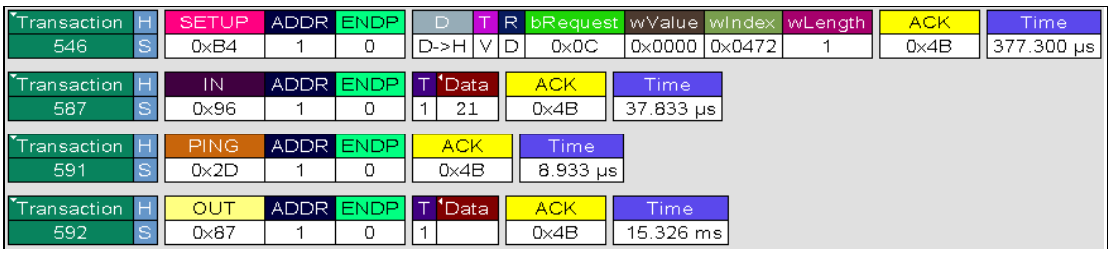


图 16-1 取得固件版本

从图 16-1 中可得出以下信息

请求索引 0472

请求值 0H

请求设备的固件版本。设备报告所有值。

16.2 清除当前文件

下图描述的传输说明了清除特性命令的处理过程。

Transaction	H	SETUP	ADDR	ENDP	D	T	R	bRequest	wValue	wIndex	wLength	ACK	Time
593	S	0xB4	1	0	H->D	V	D	0x0C	0x0006	0x0473	1	0x4B	11.067 µs

Transaction	H	OUT	ADDR	ENDP	T	Data	NYET	Time
594	S	0x87	1	0	1	01	0x69	488.000 µs

Transaction	H	IN	ADDR	ENDP	T	Data	ACK	Time
644	S	0x96	1	0	1		0x4B	5.296 ms

图 16-2 清除当前文件

图 16-2 提供了以下信息

请求索引 0473

请求值 0006H

这个命令用于请求设备清除存储的图象来释放部分空间来存储一个新图象。

16.3 设置文件索引

下图描述的传输说明了清除特性命令的处理过程。

Transaction	H	SETUP	ADDR	ENDP	D	T	R	bRequest	wValue	wIndex	wLength	ACK	Time
193417	S	0xB4	1	0	D->H	V	D	0x0C	0x0001	0x0473	1	0x4B	372.567 µs

Transaction	H	IN	ADDR	ENDP	T	Data	ACK	Time
193457	S	0x96	1	0	1	01	0x4B	61.700 µs

Transaction	H	PING	ADDR	ENDP	ACK	Time
193465	S	0x2D	1	0	0x4B	7.233 µs

Transaction	H	OUT	ADDR	ENDP	T	Data	ACK	Time
193466	S	0x87	1	0	1		0x4B	5.809 ms

图 16-3 设置文件索引

从图 16-3 可以得出

请求索引 0473

请求值 0001H

设定设备准备将图象的文件编为 1 索引。

16.4 批量传输设置

从 Philips 扫描显示中取得图象数据命令。

Transaction	H	SETUP	ADDR	ENDP	D	T	R	bRequest	wValue	wIndex	wLength	ACK	Time
374083	S	0xB4	1	0	H->D	V	D	0x0C	0x0000	0x0471	6	0x4B	9.733 µs

Transaction	H	PING	ADDR	ENDP	ACK	Time
374084	S	0x2D	1	0	0x4B	8.500 µs

Transaction	H	OUT	ADDR	ENDP	T	Data	NYET	Time
374085	S	0x87	1	0	1	00 C0 1C 00 40 81	0x69	2.028 ms

图 16-4 批量传输设置

从图 16-4 中可得到以下信息

请求索引 0471

请求值 0H

控制写传输以压缩数据方式传输。

图 16－4 的数据格式在下面的表中有详细说明。

表 16-1 数据格式

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
LSB	－	MSB	LSB	MSB	－
00	0C	1C	00	40	81
图象地址偏移			当前传输长度		操作命令（见表 16-2）

表 16-2 操作命令

Bit 7	6	5	4	3	2	1	Bit 0
1－使用 DMA 传输 0－微处理器直接对缓冲器进行操作	不用						1－扫描图象（批量数据 IN） 0－将图象数据存储到设备中（批量数据 OUT）