

RS232 接口转 USB 接口的通信方法

USB 作为一种新的 PC 机互连协议,使外设到计算机的连接更加高效、便利。这种接口适合于多种设备,不仅具有快速、即插即用、支持热插拔的特点,还能同时连接多达 127 个设备,解决了如资源冲突、中断请求 (IRQs) 和直接数据通道 (DMAs) 等问题。因此,越来越多的开发者欲在自己的产品中使用这种标准接口。而 RS232 是单个设备接入计算机时,常采用的一种接入方式,其硬件实现简单,因此在传统的设备中有很多采用了这种通信方式。一般的 IC 卡门禁考勤系统也使用 RS232 接口与 PC 机通信。如果将 USB 技术应用于 IC 卡门禁考勤系统与 PC 机之间的数据通信,这样,不仅能使 IC 卡门禁考勤设备具备 USB 通信的诸多优点,而且对 PC 机而言还可以节余 1 个 RS232 串口为其它通信所用。

1. USB 系统概述

USB 规范描述了总线特性、协议定义、编程接口以及其它设计和构建系统时所要求的特性。USB 是一种主从总线,工作时 USB 主机处于主模式,设备处于从模式。USB 系统所需要的唯一的系统资源是,USB 系统软件所使用的内存空间、USB 主控制器所使用的内存地址空间 (I/O 地址空间) 和中断请求 (IRQ) 线。USB 设备可以是功能性的,如显示器、鼠标或者集线器之类。它们可以作低速或者高速设备实现。低速设备最大速率限制在 1.5 Mb/s,每一个设备有一些专有寄存器,也就是端点 (endpoint)。在进行数据交换时,可以通过设备驱动间接访问它。每一个端点支持几种特殊的传输类型,并且有一个唯一的地址和传输方向。不同的是端点 0 仅用作控制传输,并且其传输可以是双向的。

系统上电后,USB 主机负责检测设备的连接与拆除、初始化设备的列举过程,并根据设备描述表安装设备驱动后自动重新配置系统,收集每个设备的状态信息。设备描述表标识了设备的属性、特征并描述了设备的通信要求。USB 主机根据这些信息配置设备、查找驱动,并且与设备通信。

典型的 USB 数据传输是由设备驱动开始的,当它需要与设备通信时,设备驱动提供内存缓冲区,用来存放设备收到或者即将发送的数据。USB 驱动提供 USB 设备驱动和 USB 主控制器之间的接口,并将传输请求转化为 USB 事务,转化时需要与带宽要求及协议结构保持一致。某些传输是由大块数据构成的,这时需要先将它划分为几个事物再进行传输。

具有相似功能的设备可以组成一类,这样便于分享共有的特性和使用共同的设备驱动程序。每个类可以定义其自己的描述符,如: HID 类描述符和 Report 描述符。HID 类是由人控制计算机系统的设备组成的,它定义了一个描述 HID 设备的结构,并且表明了设备的通信要求。HID 设备描述符必须支持端点输入中断,固件也必须包括一个报告描述符,表明接收和发送数据的格式。在 IC 卡门禁考勤系统引入 RS232 到 USB 的接口转换模块后,从系统所具有的特性来看,应该属于 HID 设备。因此,两种特殊的 HID 类请求必须被支持: SetReport 和 GetReport。这些请求使设备能接收和发送一般的设备信息给主机。在没有中断输出终端时,SetReport 是主机发送数据给 HID 设备的唯一方式。

2. 系统要求

为了实现 IC 卡门禁考勤系统中 RS232-USB 的接口转换,需要 1 台支持 USB 的主机,同时还要提供主机上用于与外设通信的驱动,一般由操作系统提供。此外,还需开发在主机上执行的客户端应用程序。在设备端,需要提供具有 USB 接口的主控制器芯片,以及编写主控制器上执行的 USB 通信代码和用于执行外设功能的相关代码。

2.1 主机要求

主机必须能够通过设备驱动接收 USB 数据,并且使这些数据对处理这些请求的应用程序有效。在主机中必须有一个驱动负责处理 USB 传输、辨识设备、向 USB 设备收发数据;同时,还需要有一个设备驱动-虚拟串行口,仿真真实的串口。这个驱动必须能够像真实的串口接收和发送 USB 数据。

从应用的观点,设备驱动必须能收发数据,可以通过使用一个虚拟化的串口或通过转化为 USB 数据实现。微软提供了一个叫作 USB POS 的设备驱动,它允许应用程序访问 USB 设备时,好像它们连接到标准串口上一样。系统大致结构方框图如图 1 所示。

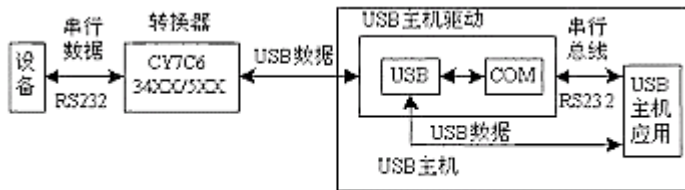


图 1

设备要求

在定义即将使用的微控制器时,必须说明一些通信要求,如:通信速率、频率、传输的数据量等。考虑到 IC 卡门禁考勤系统有效的通信速率,可以把转换器作为一个低速的设备使用,低速设备通信速度可以在 10-100 Kb/s 的范围变化。考虑到传输的数据量和传输的频率,此系统中使用中断的传输类型。中断传输可以在 2 个方向进行,但不能同时进行,这种类型的传输要求在规定的时间内完成相当大数据量的传输任务。

对于转换模块,它可以用于 PC 机的数据收发,操作系统提供了 HID 驱动,允许使用中断传输模式。对于低速设备的一个事务,中断传输最大的包容量是 8 字节,如果需要发送大量的数据,则必须把它分割为很多事务。

转换模块要定义的另一个特性是所需端点数。如上所述,端点是微控制器在 USB 通信过程中所用用来发送和接收数据的缓冲区。此系统中,该转换器定义了 2 个端点:一个端点(端点 0)用来控制传输,另一个

端点是中断输入端点，定义为发数据给 PC 机。

根据以上要求，通过研究比较现有的微控制器，考虑到如内存空间、价格和开发包等因素，我们选用 Cypress 家族的一种 8 位 RISC 微控制器 CY7C634XX/5XX。它使用哈佛总线结构，是对较高 I/O 要求的低速应用设备的低价解决方案。

图 2 为 IC 卡门禁考勤系统 USB 通信实现硬件方框原理图。

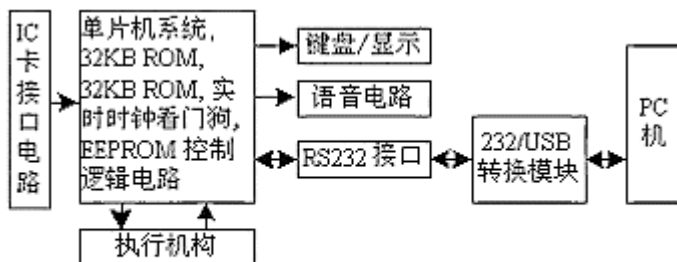


图 2

软件设计和执行

系统软件由 6 部分组成：定义描述符、设备检测和列举、端点中断服务程序、USB 数据交换模块、串行口数据交换模块、USB/Serial 模块接口。下面简要描述其中部分模块程序的功能和实现思想。

3.1 描述符定义

描述符是数据结果或信息的格式化块，它可以使主机知道这个设备。每个描述符包含了这个设备整体的信息或者某个元素的信息。所有的 USB 外设必须响应对标准的 USB 描述符的请求。

该系统中使用了 1 个接口和 2 个终端(控制和中断输入)。由于受 Win98 的限制还不能使用中输出终端，因此为了解决这个问题，我们通过端点 0 中使用 SetReport 传输 PC 机欲送往 IC 卡门禁考勤设备的数据。

数据接收是在 Output Reports 中完成的。它根据送往 IC 卡门禁考勤设备最大的数据量，系统定义为 16K 个 8 位域。发送数据给主机是在输入报告中完成的，它是 8K 个 8 位域。

3.2 设备检测和列举

当 1 个 USB 人机接口类 (HID) 设备第一次连接到总线，它将被总线供电但仍然非功能性等待 1 个总线复位。D-端的上拉电阻通知 Hub 连接上了新的设备，主机也同时知道了新连接的 USB 设备，并将它复位。紧跟输入包之后，主机发送 1 个配置包，从缺省地址 0 处读取设备描述符。读到描述符后，主机将分配一个新的地址给设备，并继续查询关于设备描述、配置描述、人机报告描述的信息，设备将对新分

配的地址作出反应。根据从设备处返回的信息，主机知道了被设备支持的数据终端的数量，完成列举过程。

列举结束后，Windows 将把新的设备加入到控制面板的设备管理器中显示。

为此，在微控制器中必须写入访问描述符的代码，这样便于对主机在列举设备时发送的请求作出有效的辨识和响应。在设备方面需要创建一个 INF 文件，使 Windows 能够辨识设备，并且为设备找到其驱动。

由于操作系统提供了简单的 INF 文件，因此，开发中只需要编写写入到微控制器中的程序。

3.3 数据发送和接收过程

发送数据到门禁考勤系统是通过控制端点 0 中使用 SetReport 来完成的。主机先向门禁考勤系统请求发送数据，设备响应请求后，主机便开始执行。当有数据到达设备的终端 0 时，将对设备产生一个中断。此时，相应的中断服务程序便将数据复制到数据缓冲区。一旦进入端点 0 的中断服务程序，所有的中断必须关闭，确保能够正确地复制数据。

微处理器的数据缓冲区编程为可以接收 64 个字节，这个值是存放在设置包的包头请求信息中。从主机处接收到的最大包大小，是根据它将发送给门禁考勤系统的最大数据量来决定的。

系统还使用了 Put_command 线程，通过 1 个 I/O 端口引脚，向门禁考勤系统串口发送数据。在执行此线程时，根据串口通信协议插入了起始位、停止位以及相应的延时。

从门禁考勤系统接收数据的过程是利用端点 1 完成的。端点 1 配置为 1 个中断输入端点，当有 1 个起始位到达引脚时，GPIO 中断必须打开，并关闭所有其它类型中断。设计中通过使用 1 个 Get_Serial 线程来收集 I/O 引脚发出的串行数据，并把它存入数据缓冲区。同时该线程负责检验接收到的起始位和停止位的正确性。当收到 8 个字节时，将接收缓冲区中的数据复制到终端 1 的缓冲区，并且允许微处理器响应中断输入请求。

考虑到一般串行口的有效波特率的范围在 300~19 200 bps，我们按处于最大波特率 19 200 bps 的情况来考虑。传输 1 个字符需要时间接近 0.75 ms；而 1 个输入中断大约每 10 ms 送 1 个 8 字节的数据包。因此，设计 1 个 128 字节的快速数据缓冲区便可以保证不会丢失数据。