RS－485总线转USB接口微模块研制

张 淳[1],徐世杰[2],王一鸣[3]
(1.重庆大学 高电压与电工新技术教育部重点实验室 电气工程学院,重庆 400030;
2.河海大学 计算机及信息工程学院,江苏 常州 213022;
3.康卡迪亚大学,加拿大)

摘要：详细介绍了基于通用串行总线(USB)专用芯片PDIUSBD12(D12)及其相关电路的硬件设计方案、转换装置微处理器AT89C52(MCU)的firmware程序设计方案、基于Windows驱动模型标准(WDM)技术的Windows下的驱动程序设计方案和基于基础类(MFC)技术的Windows下试验程序的设计方案。列举了研制工作中的难点:USB总线枚举过程的调试和Windows下WDM模式的驱动程序设计,并给出了相应的解决方案和调试经验。

关键词：USB；RS－485；数据转换；远程监控；驱动程序
中图分类号：TN 911.7 **文献标识码**：B **文章编号**：1006－6047(2006)05－0070－05

通用串行总线(USB)具有传输速度快、支持热插拔、易于扩充多个设备等优点[1]。但是USB同样存在着传输距离短的缺点,不利于工业现场使用。目前,工业现场大多使用的是RS－485总线方式,它存在速度慢、可靠性差等缺点,因而结合USB与RS－485的特点,设计了USB转换接口。同时,当前一些电脑,比如笔记本电脑,只有一个串口,而有时在工业现场需要使用多个串口设备[2],因而本装置在综合考虑了现场要求和成本的基础上也加入了RS－232的转接功能。

1 微模块的原理与应用方式

图1给出了转换装置的整体设计框架。

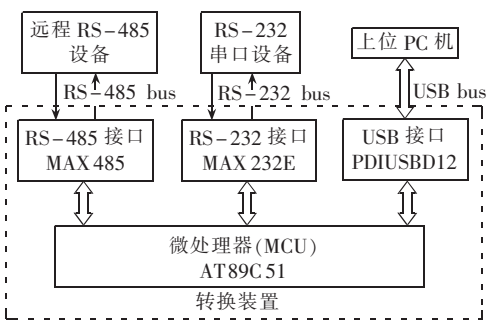

图1 接口电路原理框图
Fig.1 Construction of interface

转换装置由3个主要模块实现。其中,USB接口模块与上位PC机以USB方式通信;RS－232接口模块实现笔记本电脑等设备的串口扩充,它将USB数据通过下位微处理器转发给RS－232设备,实现对RS－232设备的控制;RS－485转换模块主要用于工业现场数据采集和远程控制,利用转换装置接收来自RS－485总线的数据并通过USB传送至PC机进行分析处理,而PC机向设备发送控制字、数据等的过程正好相反。

根据所要完成的功能应分成电平转换与数据格式转换。RS－232和USB,RS－485和USB这2组电平转换由MAX 232,MAX 485,PDIUSBD12(D12)芯片完成。数据格式转换则主要由软件完成,即下位机微处理器(MCU)的firmware的设计完成。微处理器从一个端口读入数据,根据该端口的数据协议格式解包,然后再根据所要转发端口的数据协议格式打包,最后进行转发。由于RS－232与RS－485最重要的是进行工作方式设置,设计难度主要来自于USB数据包的打包与解包,笔者采用了Philips公司的D12作为USB接口芯片完成主要工作,因而工作量大大减少,减小了开发难度与开发周期。

2 硬件设计

2.1 芯片介绍

采用Philips公司的D12作为USB接口芯片,该芯片完全符合USB1.1协议规范,支持本地DMA技术,是一款性价比很高的USB接口器件。图2是D12的内部功能方框图[3]。

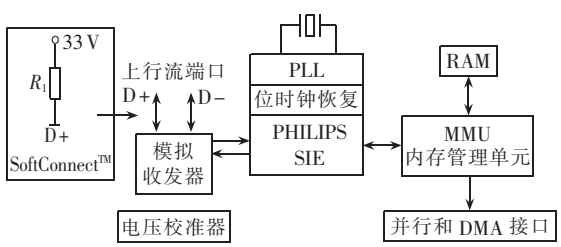

图2 D12的内部功能方框图
Fig.2 Internal function construction of D12

收稿日期:2005－06－03;修回日期:2005－11－21

a. PLL:集成了 6~48 M 时钟乘法 PLL,该 PLL 工作不需要其他外部元件。

b. PHILIPS SIE:实现了全部的 USB 协议层,完全由硬件实现而不需要固件的参与。该模块的功能包括同步模式的识别、并行/串行转换、位填充/解除填充、CRC 校验/产生、地址识别和握手评估/产生。

c. SoftConnect™:与 USB 的连接通过 1.5 kΩ 上拉电阻 R_1 将 D+置为高。允许系统微控制器在决定与 USB 建立连接之前完成初始化时序,USB 总线连接可以重新初始化而不需要物理断开。

d. GoodLink™:该技术可提供良好的 USB 连接指示,便于 USB 状态的判断。

e. MMU 和集成 RAM:两者作为 USB 数据的缓存区,允许微控制器以它的速度对 USB 信息包读写。

f. 并行和 DMA 接口:并行接口速度快,能直接与主微控制器连接。同时,在主端点(端点 2)和局部共享存储器之间也可使用 DMA 传输。

2.2　硬件电路设计

本装置设计重点在于 RS-232 和 USB,RS-485 和 USB 转换接口硬件电路[4]的设计。图 3 给出了硬件电路的设计框图。选用的芯片除了 D12,其他主要芯片还有 RS-232 接口芯片为 MAX232,RS-485 接口芯片为 MAX485,下位 MCU 选用 AT89C52,它负责整个装置的控制及数据的转发。

D12 采用复用总线方式,它的 DATA0~DATA7 8 路数据地址复用总线直接与 AT89C52 的 P0.0 口直接相连,同时 AT89C52 的 ALE 与 D12 的 ALE 相连,为使 AT89C52 既能对 D12 进行命令控制,又可与它进行数据通信,因而采用不同的地址方式区分命令与数据。对 D12 的偶地址赋值表示微处理器与 D12 实现数据通信;对 D12 的奇地址赋值表示微处理器对 D12 进行命令字的写入。

AT89C52 的串行接口通过一块可编程逻辑电路 GAL16V8 与 MAX232 和 MAX485 芯片相连。当 D12 接收到上位 PC 机传送的数据,AT89C52 将数据从 D12 的 FIFO 中读出,再根据命令字的要求判断是对 RS-232 还是 RS-485 的转换,然后通过控制 GAL16V8 芯片打开相应的端口,将数据转发给 MAX232 或者 MAX485 转换电平。而 RS-232 或 RS-485 转换为 USB 的流程正好相反,首先根据控制字打开相应的转换端口,数据经 MAX232 或MAX485 电平转换通过 AT89C52 的串口传递给微处理器,AT89C52 将数据重新转发给 D12,再经该芯片传送给上位 PC 机,完成一次转换过程。

AT89C52 的 P1.0 与可编程逻辑器 GAL16V8 的 SEL 相连,当 SEL 为高电平时,打开与 MAX232 通信;当 SEL 为低电平时,打开与 MAX485 的通信。图 4 给出了 GAL16V8 的控制时序逻辑计算机仿真结果。

3　软件设计

软件分为上位 PC 机软件和下位微处理器固件 firmware 设计,两者协同工作完成转换装置的功能。

3.1　MCU firmware 设计

3.1.1　软件功能化模块

下位 MCU 固件 firmware 设计的目标除要数据转换外,还要使 USB 传输的速率最大[5],因而整个软件设计采用前后台方式,前台为主循环完成 USB 的基本功能和所要实现的功能,后台为中断服务程序(ISR)。由于 D12 完全由中断驱动,因而后台实际上是进行 USB 数据的传输。前后台之间的通信与数据交换通过事件标志 EPPFLAGS 和数据缓存区 CONROL_XFER 实现。例如,D12 从 USB 总线上收到一

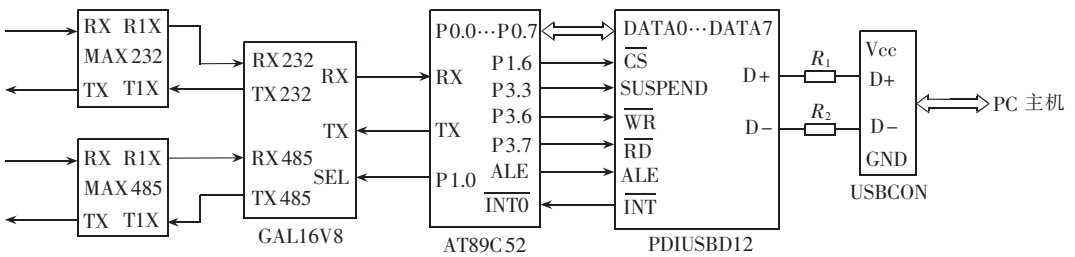

图 3　硬件电路设计框图

Fig.3　Hardware design

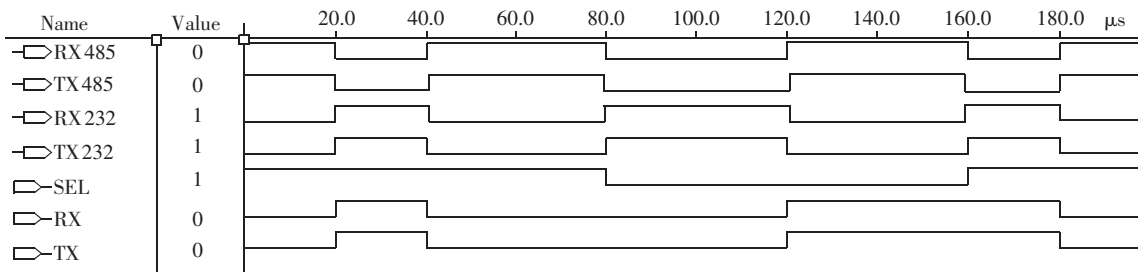

图 4　控制时序逻辑计算机仿真结果

Fig.4　Simulation of controller

个数据包,那么就产生一个对微处理的中断,微处理响应中断。前台服务被中断,开始后台 ISR 服务,此时,微处理器将数据从 D12 的 FIFO 读取到微处理器的数据缓存区并将 FIFO 清零,当 ISR 服务完成后,前台继续原来的工作。

为了软件的可移植性和易维护性,在固件 firmware 设计中采用了模块化设计方案,将整个固件查分成 6 个子模块:主循环、中断服务程序、标准请求处理、厂商请求处理、D12 命令接口、硬件提取层。图 5 给出了固件 firmware 设计的基本结构,箭头表示数据传输方向。

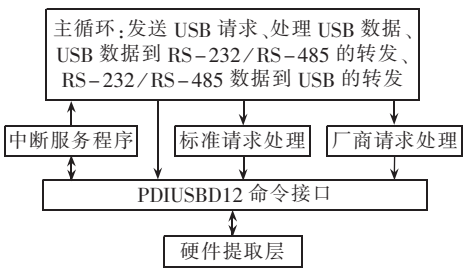

图 5 固件 firmware 设计

Fig.5 Firmware design

a. 硬件提取层是对单片机的 I/O 口、数据总线等硬件接口进行操作;

b. D12 命令接口是对 D12 器件进行操作的模块子程序集;

c. 中断服务程序是当 D12 向单片机发送中断请求时,读取 D12 的中断传输的数据,并设定事件标志“EPPFLAGS”和 Setup 包数据缓存区“CONOL_XFER”传输给主循环程序;

d. 标准请求处理程序是对 USB 的标准设备请求处理;

e. 厂商请求处理程序是对用户添加的厂商请求进行处理;

f. 主循环是发送 USB 请求、处理 USB 总线事件、USB 数据到 RS-232/RS-485 的转发、RS-232/RS-485 数据到 USB 的转发。

3.1.2 USB 设备的枚举

固件 firmware 设计中最重要的一个环节就是 USB 设备的枚举过程[6],只有成功完成该过程主机和 USB 设备才能正常通信。在这个过程中完成了主机和 USB 设备的一次握手:主机发送一个枚举请求的 Setup 包,USB 设备响应该请求并返回相应的信息,设备向主机发送 USB 特定信息后完成枚举过程。

在本设计中转换装置的枚举过程如下:

a. 设备连接是转换装置接入 USB 总线,并 USB 上电;

b. 主机检测到设备,发出复位指令;

c. 设备缺省状态是设备接收到复位信号后,就使用缺省地址(00H)对其进行寻址;

d. 获取设备描述符前 8 个字节,即主机发送请求代码 8006000100004000,AT89C52 通过 D12 发送设备描述符;

e. 地址分配是主机发送地址分配请求 0005820000000000,D12 收到请求后使能 0x82 地址,并对端点 0(控制端口)写入一个空数据;

f. 获取全部设备描述符是主机读取 **d** 中剩下的设备描述符,其中包括 VID(厂商 ID),PID(设备 ID);

g. 获取配置描述符是 D12 将所有配置描述符,包括配置描述符、端口描述符、端点描述符发给主机;

h. 设备配置是主机根据各个描述符对 D12 进行配置;

i. 安装驱动是主机找到该 USB 设备,主机安装并配置驱动程序;

j. 如果以上步骤都成功,USB 设备枚举成功,此时如果 USB 总线空闲 3ms,设备进入挂起节电状态。

其中 **a~c** 为一般 USB 设备必需的过程,**d~j** 为该装置特定的枚举过程。

3.2 PC 机软件设计

3.2.1 *接口驱动程序开发*

本装置是自行设计的特定设备,因而需要开发专用的 Windows 驱动程序[6],这是整个系统中开发难点。

当前 Windows 操作系统家族流行的驱动程序有 VxD 和 WDM 2 种形式,由于 USB 只支持 WDM 模式,因而在系统设计中采用了该方式[6]。

图 6 给出一个 WDM 驱动的基本结构对应关系[7]。其中,2 个驱动是必需的,它们是处于最低层的成为物理设备对象 PDO,它所对应的是总线驱动,在本设计中它对应 USB 总线驱动,该驱动由 Windows 系统提供,称为 USBD;FDO 称为功能设备对象,它对应了功能驱动程序,即设备驱动程序。它包含了硬件工作的所有细节,负责初始化 I/O 操作,处理 I/O 操作完成时所产生的中断事件,并为用户提供一种适当的设备控制方式。

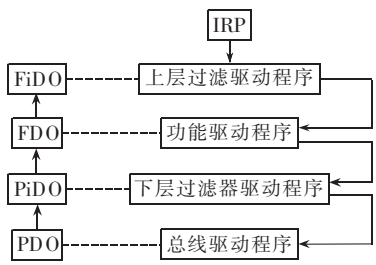

图 6 WDM 驱动的基本结构

Fig.6 Construction of WDM

一个 Windows 驱动程序包含后缀名为 INF 及 SYS 2 个文件。INF 文件控制与安装驱动程序相关的大部分活动,通过该文件告诉操作系统哪一个文件需要复制到用户硬盘上,应该添加或修改哪一个注册表项等信息。由于该文件是一个文本文件,只需遵循 USB 标准的 INF 文件格式,修改相应的一些

配置即可。主要包含以下几个关键字段:版本信息[Version];硬件制造商及硬件设备信息[Manufacturer];设备的安装与文件的拷贝路径等[DestinationDirs];字符串部分[Strings]。.SYS 文件驱动的核心部分,下面将详细介绍它的编写过程。

Microsoft 公司生产 Windows 操作系统系列,它的源代码是保密的。而驱动程序的编写需要涉及到 Windows 的一些内核机制,需要了解 Windows 的核心代码。Microsoft 提供的 DDK(Driver Development Kit)可帮助开发驱动程序[8]。编写驱动程序 .SYS 文件采用 VC++6.0 和DDK 作为开发平台。表 1 给出了该程序工程所包含的主要文件与作用。

表 1 文件列表

Tab.1 File list

文件名	作　用
D12.c	含有基本例程分发的处理代码
D12Power.c	含有对电源管理的 IRPs 的处理代码
D12Pnp.c	含有对即插即用 IRPs 的处理代码
D12Ioctl.c	含有对混合 IRPs 的处理代码
D12Ocrw.c	含有对基本的 I/O 及块数据传输处理代码
D12Ioctl.h	定义设备驱动程序的 IOCTLS
D12.h	定义设备扩展信息数据结构
D12Guid829.h	定义在用户级对硬件设备调用的 GUID 识别符

以 D12.c 文件中的 DriverEntry 例程为例介绍代码的编写过程。DriverEntry 例程(驱动程序入口地址):驱动程序和一般 DOS 和 Windows C 语言不同,它没有作为入口的 Main 函数,它以操作系统称为 DriverEntry 的函数作为入口。I/O 管理程序在驱动程序首次装入时调用该例程,DriverEntry 执行所有初始化任务。DriverEntry 需要传递一个参数指针 IN DRIVER_OBJECT DriverObject,指向一个刚被初始化的驱动程序对象,该对象就代表驱动程序对象。例程将初始化与为其他一些例程分配入口地址的指针。如:D12_Unload;D12_PnPAddDevice;D12_Dispatch;D12_ProcessIOCTL;D12_ProcessPowerIrp。

这些例程又如相应的由 DDK 提供的 IRP 包处理例程,如 IRP_MJ_CLOSE 完成了当用户关闭文件时,调用它清扫系统。DriverEntry 例程拥有一个返回类型为 NTSTATUS 的返回值,它是系统定义的表示状态的一组常数,如返回 STATUS_ SUCCESS 表明函数成功;如果函数失败,应该返回 NTSTATUS.H 中的一个错误代码,或者返回用户定义的错误代码。这时 STATUS_SUCCESS 的值为 0。

以上只是简单的讲述了一个例程的编写过程,驱动程序的开发是一个非常复杂的过程,需要开发者多查阅文献[8]及相关的开发手册。在调试阶段可多采用如 SoftICE,Windbg 等内核调试工具辅助调试,由于驱动程序开发要涉及 Windows Ring 0 层的核心代码[9],仅仅一个小的内存分配问题都可能引起操作系统的死机甚至崩溃[9],因而用 ghost 等恢复软件做好备份也不失为一个比较好的开发技巧。

3.2.2　测试程序

在驱动程序基础上编写转换装置的测试程序将非常方便[10]。由于 Windows 系统中,把每个设备都抽象为文件,此时的测试程序只需通过几条简单的文件操作 API 函数,通过已编写好的驱动程序就完成与转换装置的通信。

4　结语

研制的基于 D12 的 RS-485 转 USB 接口微模块,为工业现场的远程监控提供了一个可行的解决方案;同时也对现有设备的串口扩充方式提供一个有效的途径。

参考文献:

[1] 毛春利,周国荣. 基于 USB 的通用机器人控制器[J]. 工业控制与应用,2005,24(1):1-4.
MAO Chun-li,ZHOU Guo-rong. A versatile robot controller with USB interface[J]. **Industry Control and Application**,2005,24(1):1-4.

[2] 王国媚,须海江,马小兵,等. 基于 FPGA 和 USB 的高速数据传输[J]. 国外电子元器件,2005(5):1-2.
WANG Guo-mei,XU Hai-jiang,MA Xiao-bing,et al. High speed data transferring,recording and displaying system based on FPGA and USB[J]. **Foreign Electronic**,2005(5):1-2.

[3] 周立功. PDUSBD12 USB 固件编程与驱动开发[M]. 北京:北京航空航天大学出版社,2003.

[4] 骆展鸿,冯惠力,叶梧. TMS320C67x 的信号处理系统USB 接口扩展实现[J]. 计算机工程与应用,2005(16):100-102,126.
LUO Zhan-hong,FENG Hui-li,YE Wu. The implementation of USB interface extending in the TMS320C67x signal processing system[J]. **Computer Engineering and Application**,2005(16):100-102,126.

[5] 黄卫华,朱向东,沈绪榜. 一种高速 USB 设备控制器 IP 核的设计与实现[J]. 微电子与计算机,2005,22(5):1-3.
HUANG Wei-hua,ZHU Xiang-dong,SHEN Xu-bang. Design and implementation of a high-speed USB device controller IP core[J]. **Microelectronic and Computer**,2005,22(5):1-3.

[6] BAKER A,LOZANO J. Windows 2000 设备驱动程序设计指南[M]. 2 版. 施诺,译. 北京:机械工业出版社,2001.

[7] ONEY W. Programming the Windows driver model[M]. USA:Microsoft Press,2000.

[8] 杨成. 用 DDK 开发 Windows USB 驱动程序[J]. 程序员,2002(合订本):131-133.
YANG Cheng. Development of Windows USB driver

based on DDK[J]. **Programmer**,2002(file):131-133.
[9] 尤晋元,史美林. Windows 操作系统原理[M]. 北京:机械工业出版社,2001.
[10] 方舒燕,陈新军. USB 通信驱动程序设计[J]. 郑州大学学报:工学版,2005,26(1):1-3.
FANG Shu-yan,CHEN Xin-jun. USB Communication driver design[J]. **Journal of Zhengzhou University: Engineering Science**,2005,26(1):1-3.

(**责任编辑**:汪仪珍)

作者简介:

张 淳(1980-),男,江苏苏州人,硕士研究生,主要从事嵌入式系统与变压器局部放电方面的研究(**E-mail**:a5868@263.net)。

Research on interface between USB and RS-485

ZHANG Chun[1],XU Shi-jie[2],WANG Yi-ming[3]
(1. The Key Laboratory of High Voltage Engineering and Electrical New Technology, Ministry of Education,China,Chongqing University,Chongqing 400030,China;
2. Dept. of Computer Science & Information Engieering,Hohai University, Changzhou 213022,China;3. Concordia University,Canada)

Abstract: The USB(Universal Serial Bus)-based chip PDIUSBD12 and the hardware design of its peripheral circuits are introduced in detail,as well as the firmware design on microprocessor AT 89C52 for data exchange,WDM(Windows Driver Model) technology-based driver application design on Windows platform and MFC(Microsoft windows Foundation Class) technology-based test application design on Windows platform. Pinch points are summarized,such as the debugging for USB enumeration and driver application in WDM mode. Associated measures and experiences are provided.

Key words: USB; RS-485; data exchange; remote supervision; driver application