Modbus 协议在串口通讯中的研究及应用

肖 凯[1]，张贤斌[2]

（1. 长江水利委员会陆水枢纽管理局，湖北 赤壁 437302；
2. 长江工程职业技术学院，湖北 赤壁 437302）

摘 要：提出了Modbus 通讯协议在485串行通信中的运用，并给出了利用ATMEL公司新推出的一款高档AVR单片机ATmega 8 实现485串行通信的方法。

关键词：485串行通信；Modbus 通讯协议；CRC校验；ATmega 8单片机

中图分类号：TN915.2　　**文献标识码**：B　　**文章编号**：1673－0496（2007）01－0030－03

Research and Application of Modbus Protocol in Serial Communication

XIAO Kai[1], ZHANG Xian-bin[2]

(1,Lushui Project Management Bureau of CWRC, Chibi 437302, China;
2,Changjiang Engineering Vocational College, Chibi 437302, China)

Abstract :The application of Modbus protocol in 485 serial communication is displayed, and 485 serial communication method in high class AVR SCM ATmega8 is introduced too.

Key words:485 serial communication; Modbus protocol; CRC checking; ATmega8 SCM

随着计算机技术的不断发展，利用微机进行数据通讯在现代工业生产和过程控制中得到了越来越广泛的应用。485串行通信接口使用差分信号输出，故其抗干扰性强，传输距离远。当采用Modbus通讯协议后，可以方便快速地实现对不同生产现场的控制设备进行组网，便于集中监控。

1 Modbus 通讯协议简介

Modbus 协议是应用于电子控制器上的一种通用语言，也是一种通用的工业标准。通过此协议，控制器相互之间、控制器经由网络（例如以太网）和其它设备之间可以通信。此协议定义了一个控制器能认识使用的消息结构，而不管它们是经过何种网络进行通信的。它描述了一控制器请求访问其它设备、回应来自其它设备的请求以及怎样侦测错误并记录的过程，制定了消息域格局和内容的公共格式。

收稿日期：2006—04—25

作者简介：肖 凯（1974— ），男，贵州纳雍人，助工，中专，主要从事电力自动化产品的研发工作。

当在Modbus网络上通信时，此协议决定了每个控制器所要识别的控制设备的地址，可以按地址识别发来的消息，进一步决定要产生何种动作。如果需要回应，控制器将生成反馈信息并用Modbus协议发出。在其它网络上，包含了Modbus协议的消息可以转换为在此网络上使用的帧或包结构。这种转换也扩展了根据具体的网络解决节地址、路由路径及错误检测的方法。

1.1 两种传输方式

控制器可以设置为两种传输模式（ASCII或RTU）中的任何一种。在标准的Modbus网络通信中，用户选择想要的传输模式以及串口通信参数（波特率、校验方式等），在配置每个控制器的时候，需要注意在一个Modbus网络上，所有设备都必须选择相同的传输模式和串口参数。两种传输模式格式如表1、表2。

所选的ASCII或RTU方式仅适用于标准的Modbus网络，它定义了在这些网络上连续传输消息段的每一位，以及决定怎样将信息打包成消息域和如何解码。

1.2 Modbus消息帧

两种传输模式中，传输设备会将Modbus消息转为有起

表1 ASCII模式格式

3AH	地址	功能代码	数据长度	数据1	……	数据n	LRC高字节	LRC低字节	回车	换行

表2 RTU模式格式

地址	功能代码	数据长度	数据1	……	数据n	CRC高字节	CRC低字节

点和终点的帧，这就允许接收的设备在消息起始处开始工作，读地址分配信息，判断出哪一个设备被选中（广播方式则传给所有设备）。

1.2.1 关于起始位

当工作在ASCII模式，消息以冒号（：）字符（ASCII码3AH）开始，以回车换行符结束（ASCII码 0DH, 0AH）。其它域传输字符可以使用十六进制。网络上的设备不断侦测“：”字符，当有一个冒号接收到时，每个设备都解码下一个域（地址域）来判断它是否发给自己。

工作在RTU模式时，消息发送至少要以3.5个字符的停顿间隔开始。传输的第一个域是设备地址，可以使用的传输字符是十六进制。网络设备不断侦测网络总线，包括停顿间隔的时间。当第一个域（地址域）接收到消息，每个设备都进行解码以判断消息是否发给自己。在最后一个字符传输之后，一个至少3.5个字符时间的停顿标定了消息的结束，而另一个新的消息可在此停顿后开始。

1.2.2 地址域

消息帧的地址域包含两个字符(ASCII)或8Bit (RTU)。从设备地址可以自0到247 (十进制)。主设备通过将需要联络的从设备的地址放入到消息地址域中来选择从设备。当从设备发送回应消息的同时也把自己的地址放到回应的地址域中，以便使主设备知道是哪一个设备作出了回应。规定地址0用作广播地址，以便所有的从设备都能识别。

1.2.3 功能域

消息帧中的功能代码域包含了两个字符(ASCII)或8Bits (RTU)。可能的代码范围是十进制的1到255。全部的功能代码含义可查阅Modbus协议详细说明。如常用的 “03H”代码表示主设备需要从设备读取一组保持寄存器的值。

当从设备回应时，它使用功能代码域来指示是正常回应（无误）还是有某种错误发生（异议回应）。对正常回应，从设备仅回应相应的功能代码；对异议回应，从设备返回的是正常代码值加上80H后的结果值。

1.2.4 数据域

数据域是由两个十六进制数集合构成的，它这可以是由一对ASCII字符组成或由一RTU字符组成，范围从00H到FFH。如主设备发给从设备消息的数据域中包含的附加信息，即从设备必须执行由功能代码所定义的信息，包括寄存器地址、要处理项的数目、域中实际数据字节数等。

1.2.5 错误检测域

由于通信过程中线路和设备会受到不可预测的干扰，为了保证通信数据的可靠和可信性，必须对通讯数据进行错误检测。标准的Modbus网络有LRC和CRC两种错误检测方法。当选用ASCII模式作字符帧，错误检测域则包含两个ASCII字符。这是使用LRC（纵向冗长检测）方法对消息内容计算得出的，不包括开始的冒号符及回车换行符。LRC字符附加在回车换行符前面；当选用RTU模式作字符帧，错误检测域包含一16Bits值(用两个8位的字符来实现)。错误检测域的内容是通过对消息内容进行循环冗长检测（即CRC检测）得出的。CRC域附加在消息的最后，添加时注意先是低字节然后是高字节。

2 利用ATmega8单片机进行Modbus协议串行通讯的设计

2.1 ATmega8单片机简介

ATmega8单片机是ATMEL公司新推出的一款高档AVR单片机，它采用先进的RISC精简指令集结构，拥有130条功能强大且大部分为单时钟周期指令，工作速度高达1M/1MIPS。内部集成了8K字节的FLASH程序存储器，512字节的EEPROM，1K字节的内RAM；32个通用的工作寄存器中含有X、Y、Z三个16位的地址指针。其外部接口功能强大，有3 个定时计数器，3个PWM通道，8路A/D转换口，1个看门狗定时器，1个模拟比较器，1个可编程的USART串行接口等。由于其运算速度快和功能强大，用来做Modbus协议的串行通讯是比较理想的。

2.2 485串行通讯的原理图设计

电路原理的核心由一片ATmega8单片机和一片MAXIM公司的MAX1487芯片组成，MAX1487组成的差分平衡系统抗干扰能力强，适用于半双工通信，通信线上最多可挂128个收发器。其输入输出的差动电压符合RS485的标准，共模输入电压的范围高达-7V～+12V，可见用它作485串行通讯的接口芯片是很好的。

ATmega8 CPU工作在8M的晶振下，工作速度高达8MIPS，CPU串行口RXD和TXD引脚直接与MAX1487的输出引脚RO和输入脚DI相连；另外MAX1487的接收、发送（如图中的RE和DE脚）由CPU PB口的PB5位进行控制。PB5位清0由软件控制，能使MAX1487为“读”，PB5位置1则能使MAX1487为“发送”。A、B两信号为485的两差分输出信号485+和485－，采用双绞线将两信号接至上位机，原理图如图1。

系统通电后，初始化本系统为从机，CPU置MAX1487为读并随时响应上位机主机的串口接收中断，一旦上位机有数据下来，CPU将进入接收中断服务子程序进行数据接收和处理；根据对接收数据的分析，CPU将决定是否响应上位机的命令。若响应上位机命令，则根据要求进入发送中断子程序发送数据。

3 软件设计

根据Modbus通信协议的具体要求，充分利用ATmega8 CPU强大的软硬件资源来进行设计，串行口初始化时波特率为9 600，8位数据位，2位停止位，无校验。串行口初始化代码如下：

```
LDI   R16,0                ;
STS   ComSendC, R16        ; 数据发送计数器为初值0
```

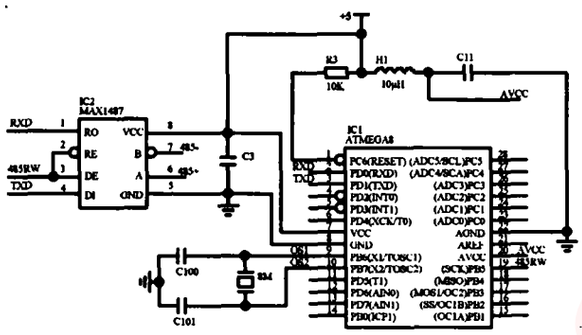

图 1　485串行通讯的原理图

```
LDI   FCOM,0               ;
LDI   R16, LOW(ComSendBu) ; 数据区首址低8位送
STS   ComSendPL,R16        ; 发送指针低8位
LDI   R16,HIGH(ComSendBu); 数据区首址高8位送
STS   ComSendPH,R16        ; 发送指针高8位
LDI   R16,0                ;
STS   ComRecC,R16          ; 数据接收计数器初值
LDI   R16, LOW(ComRecBu); 数据区首址低8位送
STS   ComRecPL,R16         ; 接收指针低8位
LDI   R16,HIGH(ComRecBu) ; 数据区首址高8位送
STS   ComRecPH,R16;          接收指针高8位
LDI   R16,$10              ;
MOV   DevAdd,R16           ; 装置站号初值=16=10H
SBI   DDRB,5               ;
SBI   DDRB,4               ;
CBI   PORTB,5              ; 初始化MAX1487为读
LDI   R16,0B00110011       ; set baud rate
OUT   UBRRL,R16            ;
LDI   R16,0B00000000       ; 波特率9600
OUT   UBRRH,R16            ;
LDI   R16,(1<<RXEN)|(1<<TXEN)|(1<<RXCIE)|
(1<<TXCIE);
OUT   UCSRB, R16           ;
LDI   R16,(1<<URSEL)|(3<<UCSZO)|(1<<USBS) ;
OUT   UCSRC, R16           ;
```

串口的初始化工作完成后，系统启动串行口的工作。当有上位机的数据进入时，系统立即进入接收中断进行数据处理。分析上位机下来的数据并进行如下的工作：①看站地址是否与本机相符；②检测数据的长度；③数据存放到接收缓冲区；④做CRC校验判断数据的合法性和有效性；⑤根据上位机要求进行相关的工作，并使串口进行数据发送。

控制程序流程图见图2。

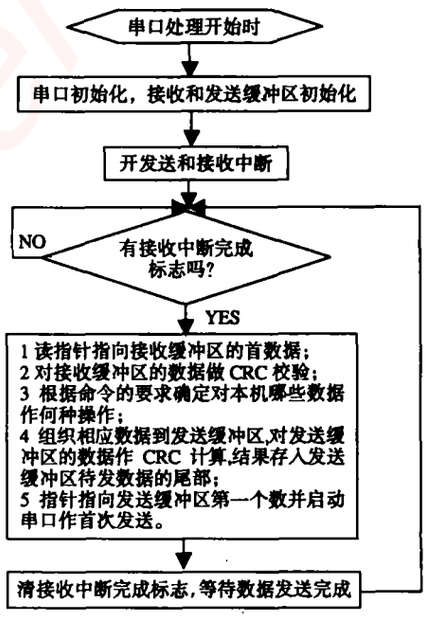

图 2　控制程序流程图

4 实际应用及结论

本文给出的基于Modbus通讯协议的485串行通讯设计已在陆水自动化设备厂的新型微机励磁装置中得到了较好的应用。实际运行情况表明，通过采用Modbus通讯协议的485串行通信，保证了在发电厂这样干扰比较恶劣的生产运行环境中通信数据的安全、可靠和稳定。

嵌入式资源免费下载

总线协议：

1. 基于 PCIe 驱动程序的数据传输卡 DMA 传输
2. 基于 PCIe 总线协议的设备驱动开发
3. CANopen 协议介绍
4. 基于 PXI 总线 RS422 数据通信卡 WDM 驱动程序设计
5. FPGA 实现 PCIe 总线 DMA 设计
6. PCI Express 协议实现与验证
7. VPX 总线技术及其实现
8. 基于 Xilinx FPGA 的 PCIE 接口实现
9. 基于 PCI 总线的 GPS 授时卡设计
10. 基于 CPCI 标准的 6U 信号处理平台的设计
11. USB30 电路保护
12. USB30 协议分析与框架设计
13. USB 30 中的 CRC 校验原理及实现
14. 基于 CPLD 的 UART 设计
15. IPMI 在 VPX 系统中的应用与设计
16. 基于 CPCI 总线的 PMC 载板设计
17. 基于 VPX 总线的工件台运动控制系统研究与开发
18. PCI Express 流控机制的研究与实现
19. UART16C554 的设计
20. 基于 VPX 的高性能计算机设计
21. 基于 CAN 总线技术的嵌入式网关设计
22. Visual C 串行通讯控件使用方法与技巧的研究
23. IEEE1588 精密时钟同步关键技术研究
24. GPS 信号发生器射频模块的一种实现方案
25. 基于 CPCI 接口的视频采集卡的设计
26. 基于 VPX 的 3U 信号处理平台的设计
27. 基于 PCI Express 总线 1394b 网络传输系统 WDM 驱动设计
28. AT89C52 单片机与 ARINC429 航空总线接口设计
29. 基于 CPCI 总线多 DSP 系统的高速主机接口设计
30. 总线协议中的 CRC 及其在 SATA 通信技术中的应用
31. 基于 FPGA 的 SATA 硬盘加解密控制器设计

VxWorks:

1. 基于 VxWorks 的多任务程序设计
2. 基于 VxWorks 的数据采集存储装置设计
3. Flash 文件系统分析及其在 VxWorks 中的实现
4. VxWorks 多任务编程中的异常研究
5. VxWorks 应用技巧两例
6. 一种基于 VxWorks 的飞行仿真实时管理系统
7. 在 VxWorks 系统中使用 TrueType 字库
8. 基于 FreeType 的 VxWorks 中文显示方案
9. 基于 Tilcon 的 VxWorks 简单动画开发
10. 基于 Tilcon 的某武器显控系统界面设计
11. 基于 Tilcon 的综合导航信息处理装置界面设计
12. VxWorks 的内存配置和管理
13. 基于 VxWorks 系统的 PCI 配置与应用
14. 基于 MPC8270 的 VxWorks BSP 的移植
15. Bootrom 功能改进经验谈
16. 基于 VxWorks 嵌入式系统的中文平台研究与实现
17. VxBus 的 A429 接口驱动
18. 基于 VxBus 和 MPC8569E 千兆网驱动开发和实现
19. 一种基于 vxBus 的 PPC 与 FPGA 高速互联的驱动设计方法
20. 基于 VxBus 的设备驱动开发
21. 基于 VxBus 的驱动程序架构分析
22. 基于 VxBus 的高速数据采集卡驱动程序开发

Linux:

1. Linux 程序设计第三版及源代码
2. NAND FLASH 文件系统的设计与实现
3. 多通道串行通信设备的 Linux 驱动程序实现
4. Zsh 开发指南-数组
5. 常用 GDB 命令中文速览
6. 嵌入式 C 进阶之道
7. Linux 串口编程实例
8. 基于 Yocto Project 的嵌入式应用设计
9. Android 应用的反编译
10. 基于 Android 行为的加密应用系统研究
11. 嵌入式 Linux 系统移植步步通

12. 嵌入式 CC++语言精华文章集锦
13. 基于 Linux 的高性能服务器端的设计与研究
14. S3C6410 移植 Android 内核
15. Android 开发指南中文版
16. 图解 Linux 操作系统架构设计与实现原理（第二版）
17. 如何在 Ubuntu 和 Linux Mint 下轻松升级 Linux 内核
18. Android 简单 mp3 播放器源码
19. 嵌入式 Linux 系统实时性的研究
20. Android 嵌入式系统架构及内核浅析
21. 基于嵌入式 Linux 操作系统内核实时性的改进方法研究
22. Linux TCP IP 协议详解
23. Linux 桌面环境下内存去重技术的研究与实现

Windows CE：

1. Windows CE.NET 下 YAFFS 文件系统 NAND Flash 驱动程序设计
2. Windows CE 的 CAN 总线驱动程序设计
3. 基于 Windows CE.NET 的 ADC 驱动程序实现与应用的研究
4. 基于 Windows CE.NET 平台的串行通信实现
5. 基于 Windows CE.NET 下的 GPRS 模块的研究与开发
6. win2k 下 NTFS 分区用 ntldr 加载进 dos 源代码
7. Windows 下的 USB 设备驱动程序开发
8. WinCE 的大容量程控数据传输解决方案设计
9. WinCE6.0 安装开发详解
10. DOS 下仿 Windows 的自带计算器程序 C 源码
11. G726 局域网语音通话程序和源代码
12. WinCE 主板加载第三方驱动程序的方法
13. WinCE 下的注册表编辑程序和源代码
14. WinCE 串口通信源代码
15. WINCE 的 SD 卡程序[可实现读写的源码]
16. 基于 WinCE 的 BootLoader 研究

PowerPC：

1. Freescale MPC8536 开发板原理图
2. 基于 MPC8548E 的固件设计

3. 基于 MPC8548E 的嵌入式数据处理系统设计
4. 基于 PowerPC 嵌入式网络通信平台的实现
5. PowerPC 在车辆显控系统中的应用
6. 基于 PowerPC 的单板计算机的设计
7. 用 PowerPC860 实现 FPGA 配置
8. 基于 MPC8247 嵌入式电力交换系统的设计与实现
9. 基于设备树的 MPC8247 嵌入式 Linux 系统开发
10. 基于 MPC8313E 嵌入式系统 UBoot 的移植
11. 基于 PowerPC 处理器 SMP 系统的 UBoot 移植

ARM：

1. 基于 DiskOnChip 2000 的驱动程序设计及应用
2. 基于 ARM 体系的 PC-104 总线设计
3. 基于 ARM 的嵌入式系统中断处理机制研究
4. 设计 ARM 的中断处理
5. 基于 ARM 的数据采集系统并行总线的驱动设计
6. S3C2410 下的 TFT LCD 驱动源码
7. STM32 SD 卡移植 FATFS 文件系统源码
8. STM32 ADC 多通道源码
9. ARM Linux 在 EP7312 上的移植
10. ARM 经典 300 问
11. 基于 S5PV210 的频谱监测设备嵌入式系统设计与实现
12. Uboot 中 start.S 源码的指令级的详尽解析
13. 基于 ARM9 的嵌入式 Zigbee 网关设计与实现
14. 基于 S3C6410 处理器的嵌入式 Linux 系统移植
15. CortexA8 平台的μC-OSⅡ及 LwIP 协议栈的移植与实现

Hardware：

1. DSP 电源的典型设计
2. 高频脉冲电源设计
3. 电源的综合保护设计
4. 任意波形电源的设计
5. 高速 PCB 信号完整性分析及应用
6. DM642 高速图像采集系统的电磁干扰设计

7. 使用 COMExpress Nano 工控板实现 IP 调度设备
8. 基于 COM Express 架构的数据记录仪的设计与实现
9. 基于 COM Express 的信号系统逻辑运算单元设计
10. 基于 COM Express 的回波预处理模块设计
11. 基于 X86 平台的简单多任务内核的分析与实现
12. 基于 UEFI Shell 的 PreOS Application 的开发与研究
13. 基于 UEFI 固件的恶意代码防范技术研究