第35卷 第12期 计 算 机 工 程 2009年6月
Vol.35 No.12 Computer Engineering June 2009
·开发研究与设计技术· 文章编号：1000—3428(2009)12—0245—03 文献标识码：A 中图分类号：TP311

QNX操作系统下的Modbus串口通信设计

许 森，潘海鹏，任 佳，苏 洁
(浙江理工大学自动化研究所，杭州 310018)

摘 要：在分析QNX设备驱动程序体系结构和Modbus协议的基础上，提出QNX 操作系统串行接口设备驱动程序设计的一般方法，用C语言设计Modbus在QNX下的串口通信程序，经过和西门子S7-200 PLC长时间的严格通信测试，结果表明该方法实时性强、稳定性好、可靠性高，取得令人满意的通信效果。

关键词：QNX操作系统；现场总线；串口通信；设备驱动程序

Design of Modbus Serial Port Communication in QNX Operating System

XU Sen, PAN Hai-peng, REN Jia, SU Jie
(Institute of Automation, Zhejiang Sci-Tech University, Hangzhou 310018)

【Abstract】This paper presents a general design method on serial interface device driver, which is based on the analysis of Modbus protocol and QNX Operating System(QNX OS) device driver architecture. It specifies serial port communication in C program under QNX OS, and tests its communication with SIEMENS S7-200 PLC. The test result proves that the method is real-time, stable and reliable.

【Key words】QNX Operating System(QNX OS); Modbus; serial port communication; device driver

1 概述

QNX操作系统是实时多任务操作系统之一，它建立在微内核和完全地址空间保护基础之上，具有实时、稳定、可靠等优点，已被广泛应用在工业自动化、航空航天、汽车、电信等领域，其性能已被无数用户在实践中证实。Modbus协议是工业自动化协议中使用最普遍的协议之一，许多工业仪器和设备都将该协议作为通信标准，如施耐德 Twido 系列和Modicon M340系列PLC、西门子S7-200 PLC、台达VFD-M系列变频器、威纶通MT8000系列人机界面。很多厂商在开发工业仪器过程中将 Modbus 集成为其中的一部分，使仪器具备开放性。

由于QNX系统并不集成Modbus串口通信协议，因此本文主要介绍 QNX 操作系统中串行接口设备驱动程序的设计及Modbus协议的实现。

2 QNX操作系统

QNX操作系统是加拿大QNX软件系统公司开发的一种分布式、多用户、多任务嵌入式实时操作系统，是一个类Unix操作系统，遵循POSIX 1003.1-2001(即POSIX.1)标准，这使得Linux, Unix以及很多的开源程序很容易移植到QNX中。QNX 由一个微内核和一些可以根据需要进行定制的系统模块组成，其内核一般为几十KB，即使加上其他必要的模块，所占用的空间也很小，而且能保持其实时、多任务的系统特征。QNX的实时性主要体现在中断响应延时和上下文切换延时上，对于常用平台，QNX响应都在微秒级，是目前实时性最强的操作系统之一，能满足苛刻的实时性要求。QNX还具有可嵌入的图形用户界面，并且支持多种处理器。

2003年加拿大航天局经过研究，从市场中的48种实时操作系统(Real-Time Operating System, RTOS)中挑出20种，并对这20种进行了详细的评测[1]，QNX操作系统排名第一，可以看出QNX系统综合性能是非常优秀的。

3 QNX操作系统设备驱动程序的体系结构

在QNX系统中，所有的设备和Unix 操作系统一样，是以设备文件的形式来处理的。从总体上来看，QNX下的设备驱动程序包括2层：(1)硬件屏蔽/抽象层(Hardware Shielding/abstraction Layer, HSL)；(2)硬件获取层(Hardware Access Layer, HAL)。HSL提供对硬件的逻辑接口；而HAL负责对硬件的操作[2]。QNX的设备驱动程序体系结构如图1所示。

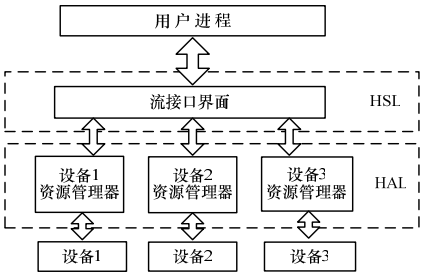

图1 QNX的设备驱动程序体系结构

在HAL层，资源管理器接收到HSL层传递的信息后，立即对硬件进行相应的操作，然后将结果返回HSL层。QNX在设备驱动程序 HAL 层提供一个统一的设备驱动程序编制框架，按照这个框架编写的程序就是资源管理器。资源管理

基金项目：浙江省科技计划基金资助项目(2006C31016, 021101039)
作者简介：许 森(1982－)，男，硕士研究生，主研方向：智能控制，嵌入式系统；潘海鹏，教授；任 佳，讲师、博士；苏 洁，硕士研究生
收稿日期：2008-11-12 **E-mail：**pan@zstu.edu.cn

器是一个用户级的服务程序，它接收来自其他程序的消息，并且随时可以和硬件设备进行通信[3]。与其他操作系统不同，QNX 的资源管理器是一种普通的进程，因此，可以随意起动、配置和停止，支持硬件的热插拔。

由于内核接口与用户层分开，因此资源管理器访问内核是通过消息机制进行信息传递的，以保证内核与接口的可靠通信。由于资源管理器都在独自的保护地址空间中运行，因此在 QNX 系统中随意加载资源管理器并不影响系统的其他部分，而且不会因个别设备驱动程序故障而导致整个系统崩溃[3]，这种体系结构为驱动程序的设计和调试带来很大方便。

4 Modbus通信协议

Modbus 通信协议是 Modicon 公司开发的应用于工业控制网络的主从式通信协议，在工业控制中得到广泛应用，现已成为流行的开放性工业标准之一。Modbus 协议描述了控制器请求访问其他设备的过程以及如何响应来自其他设备的请求，怎样侦测错误并记录错误，制定了消息域格式和消息内容的公共格式。

Modbus 的接口可以采用 RS-232/422/485 规范，支持 Modbus 的设备可以连在一起组成主-从访问的网络。在主节点轮询(即逐一单独访问从节点)时，从节点执行相应操作并返回一个应答信息；主节点也可以对网络上所有的从节点进行广播通信，此时从设备不作任何回应。如果从设备产生正常的回应，应答消息中的功能代码和查询消息中的功能代码相同，数据段包含了从设备的相关数据。如果有错误发生，功能代码将被修改，以指出回应消息是错误的，此时数据段包含了描述该错误的代码。

Modbus 通信协议有 2 种串行传输模式：ASCII 模式和 RTU 模式，与这 2 种模式对应的帧格式是 ASCII 帧格式和 RTU 帧格式[4]。ASCII 方式的主要优点是字符发送的时间间隔可达到 1 s 而不产生错误，RTU 方式的主要优点是在同样的波特率下，可比 ASCII 方式传送更多的数据，为了达到更高的通信效率，本文采用 RTU 方式。配置网络时，在同一个 Modbus 网络上的所有设备必须选择相同的传输模式、帧格式和串口参数。

5 QNX系统中Modbus通信的设计与实现

织机采用电子送经和电子卷取系统既能提高产品质量，解决织疵问题，又可调节纬密，更新织造品种，而且结构简单，灵敏度高，是当前送经和卷取系统发展的方向之一[5]。本文研究的剑杆织机电子送经和电子卷取系统结构如图 2 所示。

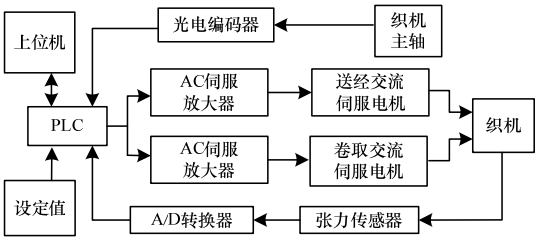

图 2　电子送经电子卷取结构

系统综合经纱张力实际值、张力设定值和织机主轴转速，通过 PLC 控制交流伺服放大器对送经伺服电机转速进行调整，使经纱保持在设定的张力范围内。同时，PLC 根据主轴电机的转速和预先设定的纬密值计算出卷取电机的转速，通过控制卷取伺服电机的转速，将已经成型的织物引离织口。

本系统中 PLC 采用西门子 S7-200，上位机采用三星 2440 芯片构成硬件平台，QNX 系统作为软件平台。上位机采用 Modbus 协议与 S7-200 PLC 进行通信，从而可以实现一台上位机对现场的多台织机进行实时监控。

在 QNX 操作系统中，外设一般分为字符型设备和块设备，串行端口和并行端口都是典型的字符型设备。因此，通过串口实现的 Modbus 通信应采用字符型设备的方式处理。QNX 系统在同一时刻可能运行多个字符型设备，因此，系统内部提供一个 io-char 模块来处理，io-char 模块包含了所有 POSIX 相关的代码，同时也扩充了一些实时系统常用的 I/O 接口代码。总之，字符型外设都继承了这些特性，驱动程序可以调用 io-char 模块实现对设备的控制[3]。io-char 模块接口如图 3 所示。

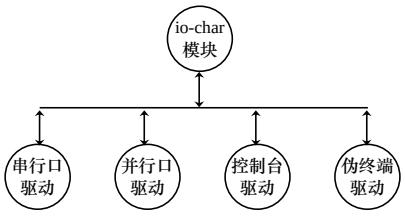

图 3　io-char 模块接口结构

io-char 模块负责管理应用程序和驱动程序之间的数据流。数据在驱动程序和 io-char 模块之间的传输是通过 FIFO 内存队列机制实现的，每一个设备有 3 个队列：一个原始输入队列，一个规范输入队列和一个输出队列[3]，整个串行端口的体系结构如图 4 所示。

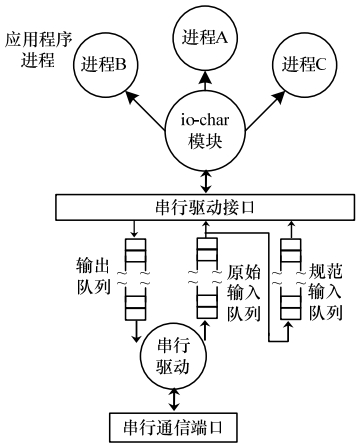

图 4　QNX 系统下的串行口 I/O

串行驱动接收到外设发送的数据后，将收到的数据放入原始输入队列中，应用程序可以到原始输入队列或规范输入队列中取数据。应用程序将需要的数据放到输出队列中，驱动程序负责把输出队列中的数据传送给目的设备。

在 QNX 系统下，串行口与/dev 下的设备文件的对应关系如下：

	串行口 1	串行口 2
设备文件	/dev/ser1	/dev/ser2

QNX 系统下的串口通信一般按如下的步骤进行：(1)打开串口设备文件，存储原有的串口设置，接收使能，配置串口参数；(2)进行协议相关处理如按协议打包数据；(3)向串口写数据；(4)读取串口收到的数据，数据无错则结束进程。具体流程如图 5 所示。

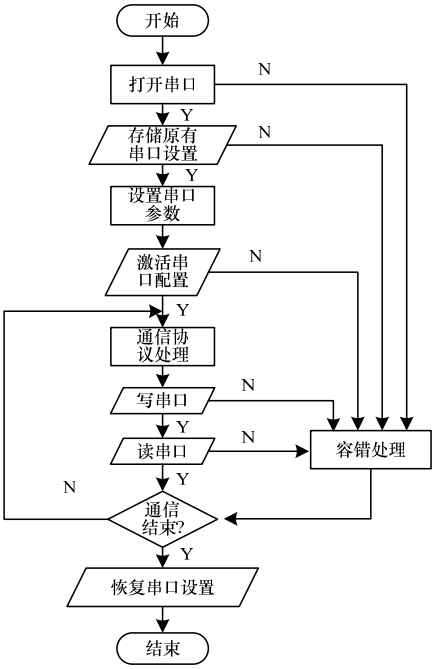

图 5　QNX 下串口通信流程

本文在 QNX 系统下，使用串口方式和 Modbus 协议实现上位机与西门子 S7-200 通信，使用 Modbus 的 RTU 模式，串口采用原始输入模式，非阻塞方式，波特率为 9 600 b/s，8 位数据位，1 位停止位，无奇偶校验位，从设备地址为 2 的西门子 S7-200 CPU 226 中读所有的输出线圈状态。最简单的程序代码如下：

```
int MODBUS_readcoil(unsigned int dev_addr,unsigned int fc_num,unsigned int startno,unsigned int endno,unsigned char * buf)
{ buf[0]=dev_addr;
  buf[1]=fc_num;
  buf[3]=(unsigned char)startno;
  buf[2]=(unsigned char)(startno>>8);
  int coilnum=endno-startno+1;
  buf[5]=(unsigned char)coilnum;
  buf[4]=(unsigned char)(coilnum>>8);
  short sCRC = CalCRC(buf, 6);
  buf[7] = (char)sCRC;
  buf[6] = (char)(sCRC>>8);
}/*Modbus 协议 RTU 模式组包*/
main()
{int fd,temp_crc;
unsigned char readbuff[256];
unsigned char writebuff[8];
struct termios old_termi,new_termi;
fd=open("/dev/ser1", O_RDWR|O_NOCTTY);
  /* 打开串口文件 */
if( fd<0 )
{ printf("open the serial port fail! errno is: %d\n", errno);
  _exit(EXIT_FAILURE);  /*打开串口失败*/}
if ( tcgetattr( fd, &old_termi)  != 0) /*存储原来的设置*/
{ printf("get the terminal parameter error when set baudrate! errno is: %d\n", errno);  /*获取终端相关参数时出错*/
  exit(EXIT_FAILURE);}
bzero(&new_termi,sizeof(new_termi)); /*将结构体清零*/
new_termi.c_cflag|= (CLOCAL|CREAD); /*忽略调制解调器状态行，接收使能*/
new_termi.c_lflag&=~(ICANON|ECHO|ECHOE);
/*选择为原始输入模式*/
new_termi.c_oflag&=~OPOST;  /*选择为原始输出模式*/
new_termi.c_cc[VTIME] = 5;    /*设置超时时间为 0.5 s/
new_termi.c_cc[VMIN] = 7;/*最少返回的字节数是 7*/
cfsetispeed(&new_termi, B9600);  /* 设置输入拨特率 */
cfsetospeed(&new_termi, B9600);  /* 设置输出拨特率 */
new_termi.c_cflag &= (~CSIZE);/* 设置数据位数,8 位数据 */
new_termi.c_cflag |= CS8; new_termi.c_cflag &= ~PARENB;
/*设置奇偶校验为无校验*/
new_termi.c_iflag &=~ ISTRIP; new_termi.c_cflag&= ~CSTOPB;
  /*设置停止位为:一位停止位*/
tcflush(fd, TCIOFLUSH);  /* 刷新输入输出流 */
if(tcsetattr(fd,TCSANOW,&new_termi)!= 0)/* 激活串口配置 */
{ printf("Set serial port parameter error!\n");
  exit(EXIT_FAILURE);}
MODBUS_readcoil(2,1,0,15,writebuff);
if(write(fd, writebuff, 8)==-1)
/* 把 Modbus 协议数据发送到串口 */
{       printf("write serial port error!\n");
  exit(EXIT_FAILURE);}
if(read(fd, readbuff, 7)==-1)      /* 从串口读数据，数据存储到 readbuff 中 */
{       printf("read serial port error!");
  exit(EXIT_FAILURE);}
for(int i=0;i<7;i++)
{ printf("%02X  ",readbuff[i]); /* 显示读回的数据 */  }
printf("\n");
tcsetattr(fd,TCSANOW,&old_termi);  /* 恢复原来的设置 */
close(fd);  /*关闭串口*/}
```

本文测试的硬件平台是 S7-200 CPU 226，QNX 平台是 QNX6.2.1，考虑到程序的可移植性，编写的 C 语言程序调用的都是符合 POSIX.1 标准的函数，程序中省略了 CalCRC() 子程序。本文对以上程序进行了扩展，加入了严密的容错机制和 S7-200 PLC 支持的所有 Modbus 功能，通过剑杆织机与上位机的实践运行，进行了长时间的严格测试，测试结果见表 1。

表 1　系统通信过程测试结果

功能号	QNX 发送数据 (十六进制)	PLC 响应数据 (十六进制)	误码率
1	02,01,00,00,00, 08,3D,FF	02,01,01,F0, 51,88	10E-8
2	02,02,02,00,00, 10,79,B5	02,02,02,F0, 00,B9,B8	10E-7
3	02,03,00,04,00, 02,85,F9	02,03,04,00,00, 00,00,C9,33	10E-9
4	02,04,00,00,00, 02,71,F8	02,04,04,00,00, 00,00,C8,84	10E-8
5	02,05,00,00,FF, 00,8C,09	02,05,00,00,FF, 00,8C,09	10E-8
6	02,06,00,02,00, 0F,68,3D	02,06,00,02,00, 0F,68,3D	10E-9
15	02,0F,00,00,00, 10,02,FF,FF F7,60	02,0F,00,00,00, 10,54,34	10E-8
16	02,10,00,00,00, 02,04,AB,CD, 23,45,95,F3	02,10,00,00,00, 02,41,FB	10E-7

从以上的测试结果可以看出，S7-200 PLC 和上位机的通信过程性能稳定，可靠性高，且系统实时响应快，性能指标完全满足实际需求。 (下转第 253 页)