doi: 10.3969/j.issn.1006-1576.2011.06.029

基于 VxWorks 嵌入式系统的多串口驱动程序开发

官琴，王璐
（中国兵器工业第五八研究所 军用电子产品事业部，四川 绵阳 621000）

摘要：针对传统单串口和双串口无法满足某些设备的通信需求问题，介绍一种基于 VxWorks 操作系统共享一个中断源的多串口驱动程序。详细介绍基于 PowerPC 体系结构 VxWorks 嵌入式操作系统的多串口驱动程序开发，给出其驱动组成的详细定义。结果表明，该驱动简单、方便，使用者可直接调用封装好的函数便能完成串口驱动程序。

关键词：VxWorks；驱动开发；标准串口
中图分类号：TP311.52 **文献标志码**：A

Multi-Serial Driver Development Based on VxWorks Embedded System

Guan Qin, Wang Lu
(Dept. of Armament Products, No. 58 Research Institute of China Ordnance Industries, Mianyang 621000, China)

Abstract: Aiming at traditional single serial ports and multi-serial port can't satisfy certain type equipment telecommunication demand, introduce a VxWorks operating system based on shared an interrupt source of multi-serial port driver. Based on the PowerPC architecture, introduce in detail VxWorks embedded operating system of multi-serial port driver development, and present the detailed definition of driving. The results show that the driver is simple, convenient, users can be directly call packaged function and realize serial driver.

Keywords: VxWorks; driver development; standard serial port

0 引言

串口是一种常见和通用的接口。因为传统的单串口和双串口无法满足某些设备的通信需求，必须对串口进行扩展。通常的扩展方法是一个串口对应一个中断源。对于嵌入式系统，不断地响应中断既浪费时间又浪费资源。因此，笔者介绍一种基于 VxWorks 操作系统共享一个中断源的多串口驱动程序，以保证实时性和可靠性。

1 VxWorks 串口驱动开发

1.1 VxWorks串口驱动概述

在 VxWorks 系统中，串行设备是一种特殊的设备。它既支持 VxWorks 下的 I/O 系统，又支持目标机代理的接口；它必须既能以中断方式工作，又能在轮询模式下工作；还要提供命令行编辑能力以及对输入、输出的数据进行缓冲。因为支持 Target Agent 接口，所以可以使用串行设备实现 Tornado 的目标机服务器（Target Server）与 VxWorks 的 Target Agent 之间的通信。串行设备驱动程序的工作模式如图 1。其中，与硬件设备无关的部分由 WindRiver 在 VxWorks 的虚拟设备 ttyDrv 中实现，因此，笔者只需根据系统给出的接口实现驱动程序，并将其挂接到虚拟设备 ttyDrv 上即可。

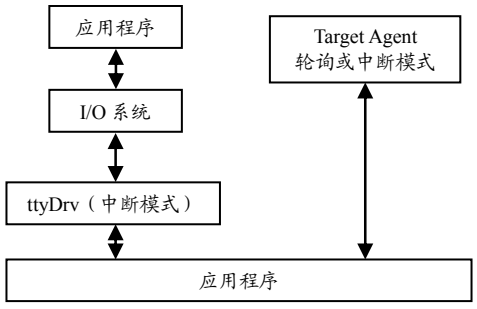

图 1 串行设备驱动程序的工作模式

1.2 虚拟设备ttyDrv

虚拟设备 ttyDrv 管理着 I/O 系统和真实系统之间的通信。ttyDrv 的职责如图 2。

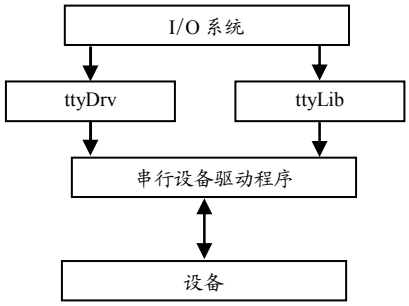

图 2 ttyDrv 的职责

作为一个字符型设备，虚拟设备 ttyDrv 将自身的入口点函数挂在 I/O 系统上，创建设备描述符并

收稿日期：2011-01-09；修回日期：2011-04-01
作者简介：官琴（1979—），女，四川人，大学本科，工程师，从事软件开发研究。

将其加入到设备列表中。当用户有 I/O 请求包到达 I/O 系统中时，I/O 系统会调用 ttyDrv 相应的函数请求。ttyDrv 可以处理 select( )函数，并管理了缓冲区的互斥，提供了任务的同步操作。

另一方面，ttyDrv 负责与实际的设备驱动程序进行信息交换。通过设备驱动程序提供的回调函数及必要的数据结构，ttyDrv 对系统的 I/O 请求作相应处理后，传递给设备驱动程序，由设备驱动程序来完成实际的 I/O 操作。

2　驱动组成

2.1　驱动的基础构架

本驱动程序基于 VxWorks 系统下的库文件 st16552Sio.c 和 st16552Sio.h 编写而成，实现了初始化、回调函数及其所需要的结构体、回调安装函数、发送接收函数、I/O 控制函数和中断服务程序。

在 st16552Sio.h 中有一个结构体 SIO_CHAN，包含了设备的基本信息和提供给高层协议的回调函数。详细定义如下：

```
typedef  struct  st16552_chan     {
SIO_CHAN  sio;                  //标准SIO_CHAN成员
STATUS (*getTxChar) ();     //发送函数指针
STATUS (*putRcvChar) ();   //接收函数指针
void *    getTxArg;
void *    putRcvArg;
UINT8 *regs;                    //寄存器
UINT8   level;                  //中断号
UINT8   ier;
UINT8   lcr;
UINT32  channelMode;        //中断或回环模式
UINT32  regDelta;             //寄存器地址空间
int         baudRate;           //当前波特率
UINT32  xtal;                   //时钟频率
    } st16552_CHAN;
```

SIO_CHAN 结构体是驱动层和 VxWorks 系统的中间体。因此在 sysSerial.c 文件中必须定义一个 st16552_CHAN 对象；st16552DevInit 函数主要完成 sio_chan 指向的驱动和回调函数的安装；st16552Int 函数主要完成中断处理函数的装载。

2.2　硬件组成

该驱动对应的CPU芯片型号为amcc405，串口芯片型号为ST16C554。一片ST16C554扩展4个串口，该系统使用2片ST16C554扩展8个串口，使用CPLD地址译码 8 个串口共用一个中断源。

2.3　软件流程

2.3.1　串口的初始化

在BSP包的sysSerial.c中包含st16552Sio.h文件。定义一个结构体如下：

```
LOCAL st16552_CHAN    xycChan[8];
```

主要的初始化代码如下：

```
for(i=0;i<8;i++)
{
   intDisable(xycParas[i].intLevel);
   xycChan[i].level = xycParas[i].intLevel;
   xycChan[i].baudRate = 9600;
   xycChan[i].regs = (UINT8*)xycParas[i].baseAdrs;
   xycChan[i].regDelta = 1;
   xycChan[i].xtal = 1843200;
   st16552DevInit(&xycChan[i]);
}
```

设置 8 个串口中断号、波特率、寄存器基地址、及时钟频率。

2.3.2　串口中断服务程序

串口中断服务函数的代码如下：

```
void commIsr(void)
{
   unsigned int    ch,bit;
   int ii;
   ch=*((unsigned char *)CPLD_STATUS_ADDR);
   bit=1;
   for(ii=0;ii<8;ii++){
       if((ch&bit))
          st16552Int(&xycChan[ii]);
     }
   bit<<=1;
}
```

其中：ch为共享中断源地址，由CPLD控制。8个数据位分别表示8个串口是否有中断。

安装中断服务程序的函数代码如下：

```
void sysSerialHwInit1_2(void)
{
    int intLevel = intLock ();
    (void)
    intConnect(INUM_TO_IVEC(INT_VEC_UART2),
    commIsr, NULL);
    intEnable (INT_LVL_UART2);
    intUnlock (intLevel);
 }
```

2.4 串口操作

2.4.1 串口的初始化代码

```
ff=fopen("/tyCo/3","r+");
res=ioctl(fd,SIO_BAUD_SET,baudrate);
res=ioctl(fd,SIO_MODE_SET,SIO_MODE_INT)
opt=(CLOCAL|CREAD|CS8)&~(HUPCL|STOPB
|PARENB);
ioctl(fd,SIO_HW_OPTS_SET,opt);
```

2.4.2 串口读操作代码

```
unsigned char buf[20];
while(1){
if(read(fd,buf,1)>0)
    logMsg("com data %x\n",buf[0]);
```

2.4.3 串口写操作代码

```
unsigned char buf[20];
buf[0]=0x55;
write(fd,buf,1);
```

以上代码可以不用修改直接使用tornado编译。

3 结论

通过以上操作，使 8 个串口能同时稳定工作。由于使用者不必详细了解串口的工作机制，直接调用其中封装好的函数就可以完成串口驱动程序，且上层的应用使用方法和标准串口使用方法一样，因此，使用起来十分简单、方便。

参考文献：

[1] WindRiver . VxWorks Programmer's Guide[Z]. 北京: 清华大学出版社.

[2] 杨平, 周启平. VxWorks 下设备驱动程序及 BSP 开发指南[Z]. 北京: 中国电力出版社.

[3] 官琴, 宋方伟, 基于 Vxworks 操作系统的 IEEE1394 协议通讯程序实现[J]. 兵工自动化, 2010, 29(6): 95.

****

（上接第 93 页）

包括 C safe-error 和 M safe-error 攻击。此外，现有的算法仍然面临遭受故障分析的威胁，如何寻找一种有效的防护算法是今后的研究内容之一。

参考文献：

[1] Boneh D, DeMillo R, Lipton R. On the importance of checking cryptographic protocols for faults[C]. Heidelberg : In Fumy, W. (ed.) EUROCRYPT 1997. LNCS, 1997: 37-51.

[2] Boscher A, Naciri R, Prouff E. CRT RSA algorithm protected against fault attacks[C]. In: Sauveron D, Markantonakis K, Bilas A, Quisquater, J.-J. (eds.) WISTP 2007. Springer, Heidelberg, 2007, 4462: 237-252.

[3] Aum¨ uller C, Bier P, Fischer W, Hofreiter P, Seifert J.-P. Fault attacks on RSA with CRT: Concrete results and practical countermeasures[C]. In: Kaliski Jr., B.S., Ko¸ c, C¸ .K., Paar, C. (eds.) CHES 2002. Springer, Heidelberg, 2003, 2523: 260-275.

[4] Shamir A. Method and apparatus for protecting public key schemes from timing and fault attacks. United States Patent *5,991,415[P]. 1999-11-23. Also presented at the rump session of EUROCRYPT 1997.

[5] Yen, S.-M., Kim S, Lim S, Moon S. RSA speedup with Chinese Remainder Theorem immune against hardware fault cryptanalysis[J]. IEEE Transactions on Computers, 2003, 52(4): 461-472.

[6] Joye M, Pailler P, Yen, S.-M. Secure evaluation of modular functions[J]. International Workshop on Cryptology and Network Security 2001, 2001: 227-229.

[7] Wagner D. Cryptanalysis of a provably secure CRT-RSA algorithm[C]. New York: 11th ACM Conference on Computers and Communications Security, ACM Press, 2004: 92-97.

[8] Yen, S.-M., Joye M. Checking before output may not be enough against fault based cryptanalysis[J]. IEEE Transactions on Computers, 2000, 49(9): 967-970.

[9] Ors S B, Batina L, Preneel B, Vandewalle J. Hardware implementation of a Montgomery modular multiplier in a systolic array[C]. In: Proceedings of the 17th International Symposium on Parallel and Distributed Processing, IEEE Computer Society, Los Alamitos, 2003: 1-8.

[10] Giraud C. Fault resistant RSA implementation[C]. D. Walter C, Ko, c, C, .K., Paar, C. (eds.) CHES 2003. Springer, Heidelberg, 2003, 2779: 142-151.

[11] Aumuller C, Bier P, Fischer W, Hofreiter P, Seifert J.-P. Fault attacks on RSA with CRT: Concrete results and practical countermeasures[C]. In: Kaliski Jr, B.S., Ko¸ c, C¸ .K., Paar, C. (eds.) CHES 2002. Springer, Heidelberg, 2003, 2523: 260-275.

[12] Skorobogatov S, Anderson R-J. Optical fault induction attacks[C]. In: Kaliski Jr.,B.S., Ko¸ c, C.K, Paar, C. (eds.) CHES 2002. Springer, Heidelberg, 2003, 2523: 2-12.

[13] Seifert J.-P. On authenticated computing and RSA-based authentication[C]. New York: Proc. of ACM conference on computer and communications security 2005, ACM Press, 2005: 122-127.

[14] Blomer J, Otto M. Wagner's attack on a secure CRT-RSA algorithm reconsidered[C]. In: Breveglieri, L., Koren, I., Naccache, D., Seifert, J.-P. (eds.) FDTC 2006. Springer, Heidelberg, 2006, 4236: 13-23.