嵌入式 Linux 下 CAN 总线驱动程序设计

宋扩东，卢珞先
武汉理工大学信息工程学院通信与信息系统，武汉(430070)
E-mail: song_0011@sohu.com

摘　要：以基于 ARM9 核心的 S3C2410 微处理器与 CAN 总线控制器 MCP2515 为例，讨论了嵌入式 Linux 操作系统下的设备驱动程序开发。在构建好硬件系统的基础上，详细介绍了基于嵌入式 Linux2.6 内核下的 CAN 总线驱动程序的开发过程，并结合 CAN 总线分析测试仪，给出分析测试结果。

关键词：CAN；驱动；嵌入式 Linux；ARM9;
中图分类号：TP368

1. 系统硬件结构介绍

系统中 CAN 总线主要用来完成 S3C2410 开发板和 CAN 总线分析仪的数据传输。在 S3C2410 开发平台上，MCP2515 芯片用作 CAN 控制器，MCP2551 芯片用作 CAN 收发器，S3C2410 微处理器用作节点控制器。如图 1 所示。

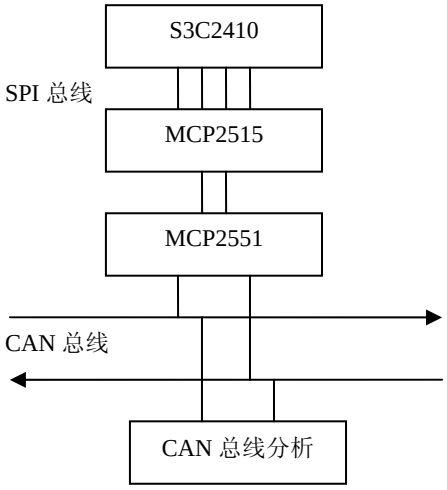

图 1 系统硬件结构

1.1 S3C2410 的 SPI 接口简介

SPI(Serial Peripheral Interface)[1]是一个同步串行外围接口，允许 MCU 与各种外围设备以串行方式进行通信。S3C2410 微处理器包括两路 SPI，每一路分别有两个 8 位转移寄存器，用来发送和接收数据。在 SPI 进行传输时,数据同时发送和接收[2]。如果只想发送数据，接收到的数据将是无效的；如果只想接收数据，应当发送全是 1 的数据。

1.2 MCP2551 功能简介

MCP2551 是 CAN 协议控制器和物理总线之间的接口。这个器件向总线提供了差动的发送能力，向 CAN 控制器提供了差动的接收能力。

1.3 MCP2515 功能简介

MCP2515是一款独立CAN控制器,是为简化连接CAN总线的应用而开发的。图2 简要显示了MCP2515的结构框图[3]。该器件主要由三个部分组成[4]:

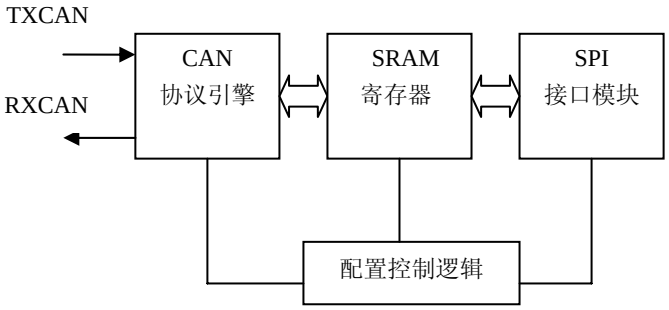

图 2 MCP2515 的结构框图

（1）CAN 协议引擎(处理所有总线上的报文发送和接收) ；

（2）用来为器件及其运行进行配置的控制逻辑和SRAM寄存器；

(3) SPI协议模块，用来实现SPI通信模式。

2. 驱动程序设计

2.1 Linux 设备驱动程序简介

Linux设备驱动程序是内核的一部分，它工作在内核态，设备驱动程序必须向Linux核心或者其他所在的子系统提供一个标准的接口[5]。大多数设备驱动程序可以在需要的时候以核心模块方式加载，在不需要的时候卸载，同时Linux设备驱动程序有几个固定的功能模块，诸如Linux内核注册设备时的初始化模块，用于系统卸载模块时删除设备驱动程序的模块等。

2.2 嵌入式 Linux2.6 内核新增特点

实时可靠性是嵌入式应用较为普遍的要求，相比Linux2.4内核，Linux 2.6 已经在内核主体中加入了提高中断性能和调度响应时间的改进，其中有三个最显著的改进[6]：采用可抢占内核、更加有效的调度算法以及同步性的提高。同时，Linux 2.6 内核的移植更加方便，新增了对许多新的体系结构和处理器类型的支持，而且可以支持大容量内存模型、微控制器，还改善了I/O子系统，增添了更多的多媒体应用功能等。

2.3 CAN 总线驱动程序设计

1. 驱动初始化函数static int __init MCP2515_init()

模块初始化初始化函数中首先通过ioremap()函数将S3C2410的SPI寄存器的物理地址映射到内核空间[7]，这样才可以在驱动程序中访问和配置S3C2410的SPI寄存器。在正确的配置S3C2410的SPI寄存器后，通过register_chrdev()函数为MCP2515注册设备驱动，分配主设备号，这样通过在文件系统的设备文件目录中创建对应的设备文件后，就可以用 Linux的系统函数来操作MCP2515了。最后通过enable_irq()函数使能S3C2410相应的中断 引 脚, 通过set_irq_type()函数初始化MCP2515的中断触发方式等。

2. 中断注册函数request_irq()

由于CAN总线接收数据时必须与系统以中断方式交换数据,所以必须注册中断。下面是具体代码request_irq (IRQ_EINT0, mcp2515_int，SA_INTERRUPT，device_name，NULL)；其中函数request_irq ()的第1个参数是设备申请的中断号，第2个参数是中断处理函数指针，

第3个参数是与中断管理有关的位掩码选项,，SA _ INTERRUPT表示中断处理程序是一个快速中断处理程序，第4个参数是设备名称，第5个参数是申请时告诉系统的设备标志。该函数返回值为0表示申请成功，返回负数表示失败，这样当中断发生时，在中断处理函数mcp2515_int ()中读取CAN状态寄存器CANSTAT，判断RXB0是否装入报文，如果是则把报文通过SPI接收数据寄存器读取到buffer中，等待系统函数read()读取。

3. 文件结构file_operations

这个结构存在于< linux/fs. h >中，文件结构中定义了很多操作，这些操作主要是实现系统调用。内核可以通过文件结构来访问驱动程序的函数。

```
static struct file_operations mcp2515_fops=
{
        // mcp2515_fops所属的设备模块
        .owner    =THIS_MODULE,
        //打开设备
        .open     =MCP2515_open,
        //释放设备
        .release    =MCP2515_release,
        //向总线发送数据
        .write     =MCP2515_write,
        //总线读取数据
        .read      =MCP2515_read,
};
```

读写CAN总线的功能可以由Linux系统函数read ()和write()实现，通过传递不同的参数给驱动程序中的MCP2515_read()和MCP2515_write()，我们可以读取和写入相应的数据。驱动中的CAN总线读函数MCP2515_read()是程序设计的难点，其中采用了Linux内核阻塞机制，核心代码如下：

```
static ssize_t MCP2515_read(struct file *file,const char *buffer,size_t count,loff_t *ppos)
{
        uchar i;
        uchar receive_data[11];
        //使读进程阻塞于此，等待中断发生时唤醒。
        interruptible_sleep_on(&my_wait);
        //打印调试信息
        printk("sleep over\n");
        //读取  CAN数据帧的标识符高位
        receive_data[0]=spi_read_2510(RXB0SIDH);
        //读取CAN数据帧的标识符低位
        receive_data[1]=spi_read_2510(RXB0SIDL);
        //读取CAN数据帧的数据长度
        receive_data[2]=spi_read_2510(RXB0DLC);
        //片选信号拉低
      s3c2410_gpio_setpin(S3C2410_GPB0,0);
      //延时1s确保片选完成
      mdelay(1000);
```

```
        //发读命令字
        spi_tx(cmd_read);
        //发接收缓冲器地址
        spi_tx(RXB0D0);
        for(i=0;i<8;i++)
        {
                //提供S3C2410的SPI读数据所需时钟脉冲信号
                  spi_tx(0xff);
                //等待SPI读一字节数据完成
                  while(!(SPI_STAT0&0x1));
                //读取CAN数据帧的数据字段
                  receive_data[i+3]=SPI_RDAT0;
        }
        //片选信号拉高
        s3c2410_gpio_setpin(S3C2410_GPB0,1);
        //延时1s
        mdelay(1000);
        //将收到的CAN数据帧传递到用户空间
        copy_to_user(buffer,receive_data,sizeof(receive_data); printk("receive ok\n");
        return 0;
}
```

(4) 卸载驱动程序模块函数代码如下,实现释放中断和注销设备的作用。

```
static void __exit MCP2515_exit()
{
        //释放中断号
        free_irq(IRQ_EINT0,NULL);
        //注销设备驱动
        unregister_chrdev(majornum,device_name);
        printk("MCP2515 driver exit\n");
}
```

3. 实验结果

3.1 CAN 总线分析测试仪

为了测试分析上述 CAN 总线驱动是否成功加载，并完成了 CAN 总线上的数据通信任务，实验室采用广州周立功单片机发展有限公司的 CAN 总线分析测试仪- CANalyst CAN，该仪器可以实时的发送、接收 CAN 总线上的数据帧，并支持在 PC 上图表显示发送和接收的 CAN 总线数据。

3.2 数据处理实验结果

结合 CAN 总线分析仪 CANalyst CAN，实验中通过应用程序，经 S3C2410 的 SPI 总线向 CAN 总线分析仪发送标准 CAN 侦“0，1，2，3，4，5，6，7，8”，在 CAN 总线分析仪的图形界面上的显示结果如图 3 所示，可以看出在该驱动程序下的数据收发完全正常，达到了预期的设计目标。

图 3 CAN 分析仪测试数据显示

4. 结论

本文中设计的 CAN 总线驱动程序可在 ARM9 嵌入式系统上正常运行,很好的完成了 CAN 总线数据的正常传输，测试结果完全正确，完成了设计需要。而在嵌入式 Linux2.6 内核下的 CAN 总线驱动程序的开发，对其他 Linux 驱动程序的开发有一定的启示作用。

参考文献

[1] Microchip，MCP2515,Stand-Alone CAN Controller With SPI Interface，2003
[2] Microchip，AN215,A Simple CAN Node Using the MCP2510 and PIC12C67X，2002
[3] 饶运涛等，现场总线 CAN 原理与应用技术，北京航空航天大学出版社，2003.6
[4] 王继国，孙新亚，CAN 控制器芯片 MCP2510 在远程监测系统中的应用，电子技术应用,2004(4)
[5] Alessandro Rubini，Linux 设备驱动程序，中国电力出版社, 2006.1
[6] 潘巨龙，黄宁，ARM9 嵌入式 Linux 系统构建与应用，北京航空航天大学出版社，2006.6
[7] 骆耀祖，Linux 操作系统分析，清华大学出版社, 2004.5

Design of CAN Driver Based on Embedded Linux

Song Kuodong,Lu Luoxian
Department of Communication and Information System,College of Information Engineering,Wuhan University of Technology,Wuhan（430070）

Abstract
This paper took CAN and S3C2410 as an example and discussed the development of device driver in Linux operation system. Based on hardware design, details on CAN's driver development based on embedded Linux 2.6 Kernel and give results with CAN analyst device.
Keywords: CAN; Driver; ARM9; Embedded Linux

作者简介：宋扩东，男，1981 年生，硕士研究生（在读）。主要研究方向是嵌入式系统、数据通信、虚拟仪器技术和图像采集技术。

Linux公社（LinuxIDC.com）于2006年9月25日注册并开通网站，Linux现在已经成为一种广受关注和支持的一种操作系统，IDC是互联网数据中心，LinuxIDC就是关于Linux的数据中心。

LinuxIDC.com提供包括Ubuntu，Fedora，SUSE技术，以及最新IT资讯等Linux专业类网站。

并被收录到Google 网页目录-计算机 > 软件 > 操作系统 > Linux 目录下。

Linux公社（LinuxIDC.com）设置了有一定影响力的Linux专题栏目。

包括:

Ubuntu专题

Fedora专题

RedHat专题

SUSE专题

红旗Linux专题

Android专题