基于 MCP2515 的多路 CAN 总线接口及驱动程序设计※

■北京交通大学 徐晓东 路红英 张宁

摘 要 结合实际需求,提出一种多路 CAN 总线接口的设计方法。系统硬件主体采用 AT91RM9200 和 MCP2515,操作系统采用 ARM-Linux。详细介绍硬件的接口设计,针对该硬件接口设计分析 ARM-Linux 下的驱动程序的设计方法,并对驱动程序实现过程中需要注意的问题进行了深入分析。经过测试,该设计方案可以满足应用的要求。

关键词 AT91RM9200 MCP2515 ARM-Linux CAN SPI 驱动程序

引 言

在铁路系统中,为了保证列车的安全运行,需要对铁轨及周围状况进行实时检测。目前采用的方法是在铁路沿线安装多个检测设备,用于检测洪水、大风、泥石流等自然灾害及轨温等参数。这些设备一般采用的通信方式是 RS232、RS485 或 CAN,并通过专线连接至监控中心的各个监控设备。这种方式极大浪费了线路资源,也不易于设备的统一管理。因此,需要一种安装在铁路沿线的设备,它将附近的检测设备发送的信息统一收集并通过一条专线直接送往监控中心。为了与多个检测设备通信,必须同时具有多个 RS232、RS485 和 CAN 接口。基于这种应用需要,本文提出了扩展多个 CAN 总线接口的方法。

1 系统结构

1.1 芯片介绍

系统采用 Atmel 公司的 AT91RM9200(以下简称"9200")作为 MCU。该处理器基于 ARM920T 内核,主频为 180 MHz 时,性能可达到 200MIPS;最高主频为 209 MHz。该处理器还具有丰富的外设资源,非常适合工业控制领域的应用[1];采用的操作系统是 ARM-Linux,内核版本为 2.4.19。

目前主流的 CAN 协议控制器一般采用 I/O 总线(SJA1000 等)或 SPI 接口(MCP2515 等)与 MCU 进行通信。由于本设计采用 PC/104 总线扩展卡的方式来扩展多个 RS232 和 RS485 接口,没有多余的 I/O 片选线可用,因此最终选用 9200 的 SPI 接口与 MCP2515 进行多路 CAN 总线接口的扩展。

MCP2515 是 Microchip 公司推出的具有 SPI 接口的独立 CAN 控制器。它完全支持 CAN V2.0B 技术规范,通信速率最高可达 1 Mbps,内含 3 个发送缓冲器、2 个接收缓冲器、6 个 29 位验收滤波寄存器和 2 个 29 位验收屏蔽寄存器[2];它的 SPI 接口时钟频率最高可达 10 MHz,可满足一个 SPI 主机接口扩展多路 CAN 总线接口的需要。

1.2 系统硬件接口

图 1 是 9200 与 MCP2515 的接口原理框图,通过 9200 的 SPI 接口,连接了 5 个 MCP2515。由于 9200 的 SPI 从设备片选线数量有限,故采用片选译码方式,NPCS0 可作为普通的外部中断线使用(NPCS0 与 IRQ5 复用引脚)。由于 9200 的外部中断线资源有限,故采用中断线共享的方式,即分别有两个 MCP2515 共享同一中断线,最后一个 MCP2515 独占一条中断线,以满足不同通信速率下数据处理的需要。

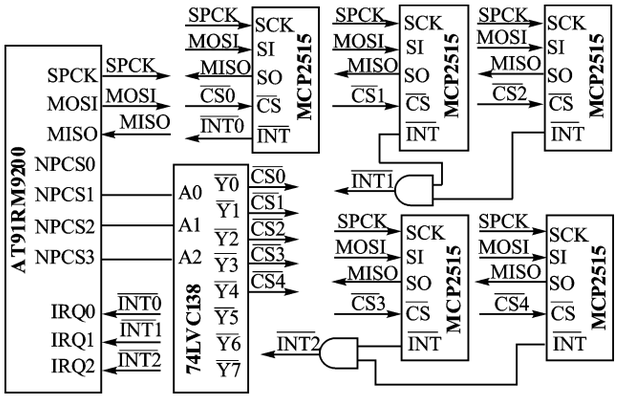

图 1 AT91RM9200 与 MCP2515 接口原理框图

图 2 是 MCP2515 的外围 CAN 总线接口框图,图中省略了 MCP2515 和 9200 的接口部分。由于设备需要安装在铁路沿线,必须具有防雷击的能力。因此 MCP2515 与 CAN 总线收发器(TJA1050)之间采用高速光耦进行完全的电气隔离,并且光耦两端电路的电源也必须用电源隔离模块隔离开,这样才能真正起到隔离的作用。在 TJA1050 的 CANH 和 CANL 引脚与地之间连接 2 个 30 pF 的电容,可以过滤 CAN 总线上的高频干扰;2 个二极管可以在总线电压发生瞬变干扰时起保护作用。光耦正常工作时输入电流为 10 mA 左右,内部发光二极管的正向电压降为 1.7 V 左右,因此要特别注意输入端串联电阻的阻值选择。

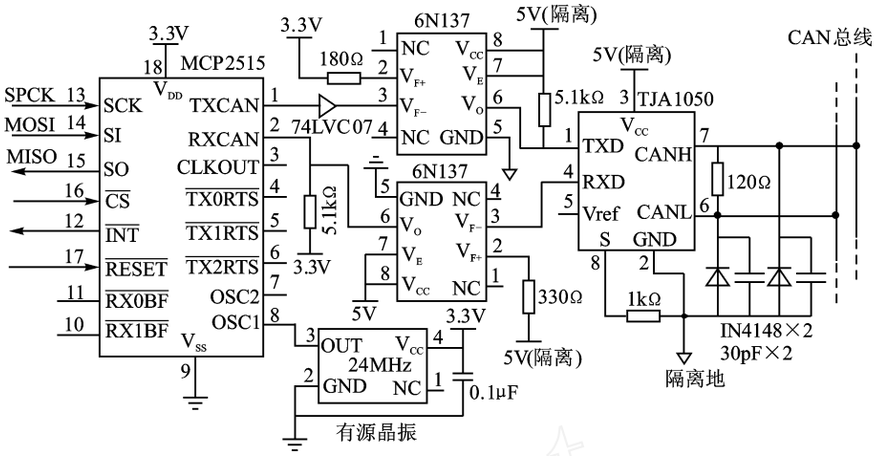

图 2 MCP2515 CAN 总线接口电路

需要注意的是,SPI 使能后,只有在 SPI_TDR(发送数据寄存器)中有数据时,才会根据片选配置(固定外设或可变外设)使能相应片选;而 SPI_TDR 中无数据时,则片选自动禁用。因此,9200 向 MCP2515 连续发送多个字节时,要保证在前一个字节传输完毕前,后一个字节就被写入到 SPI_TDR 中,以避免片选被自动禁用;同时,在传输完每一个字节后,还要读取 SPI_RDR(接收数据寄存器)。

下面以 MCP2515 的读指令为例,说明图 4 所示的驱动程序完成一次读指令操作(只读一个字节数据)的过程,并假设 9200 SPI 采用固定外设的片选配置方式。其他指令的软件实现流程与读指令类似。

2 SPI 主机的工作方式

9200 通过 SPI 接口与 5 个 MCP2515 进行通信,9200 的 SPI 控制器工作在主机模式,MCP2515 工作在从机模式。MCP2515 支持多个指令(如复位指令、读指令、写指令等),以便于 9200 通过 SPI 接口对 MCP2515 的内部寄存器进行读/写操作。9200 SPI 控制器作为主机时工作模式流程如图 3 所示[1]。

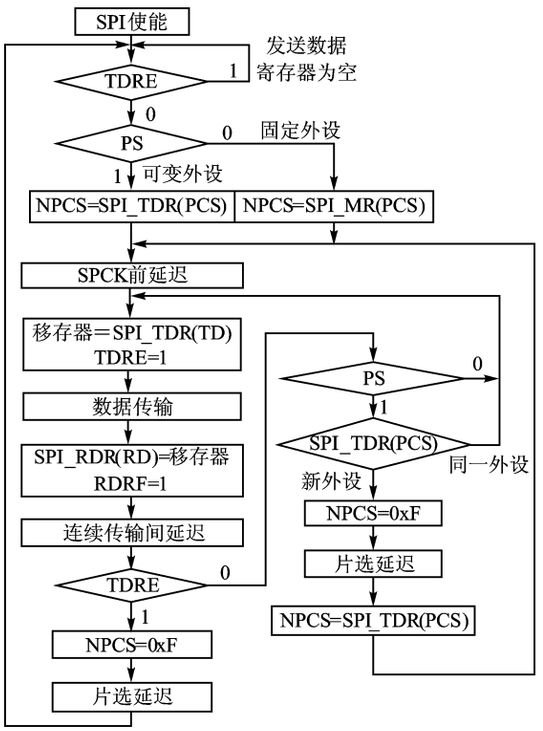

图 3 AT91RM9200 SPI 控制器主机模式流程

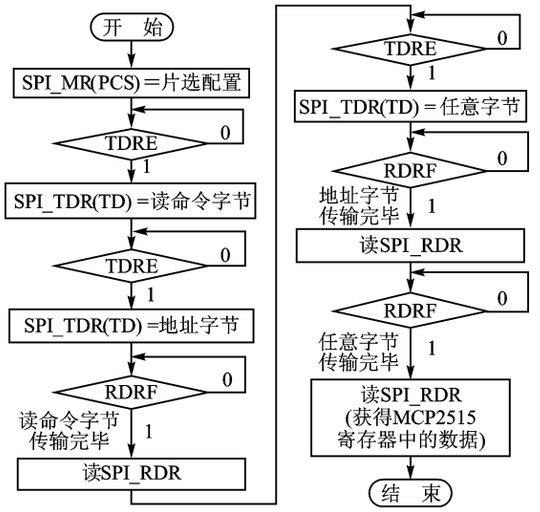

图 4 SPI 读指令操作软件流程

3 驱动程序设计

驱动程序是应用程序与硬件之间的中间软件层,它完全隐蔽了设备工作的细节。Linux 操作系统根据设备中

信息传送方式的不同,将设备分成3种类型:字符设备、块设备和网络设备[3]。9200与MCP2515的通信都是通过SPI接口以字节为单位进行的,因此MCP2515属于字符设备。由于5个MCP2515共享9200的一个SPI接口,因此采用一个驱动程序来管理所有的MCP2515,这样做有利于对所有设备进行统一管理。

3.1 驱动程序中定义的主要数据结构

CAN总线通信是基于报文帧的,在驱动程序中,无论发送数据还是接收数据都是基于报文帧的操作[4],因此需要设计合适的数据结构以满足数据操作的需要。

3.1.1 接收与发送CAN报文帧结构体

```
typedef struct {
    unsigned char node_num;
    unsigned int  id;
    unsigned char dlc;
    unsigned char data[8];
    int ext_flag;
    int rtr_flag;
} CanFrame;
```

其中,node_num为MCP2515的节点号(0～4),id为CAN报文帧的标识符,dlc为数据长度(0～8),data为CAN报文帧的数据缓冲区,ext_flag用于标识CAN报文帧是否为扩展帧,rtr_flag用于标识CAN报文帧是否为远程帧。

3.1.2 设备配置结构体

(1) 波特率和报文滤波配置结构体

```
typedef struct{
    unsigned char   node_num;
    CanBaudRate   baudrate;
    CanFilter       filter;
    int br_flag;
    int fi_flag;
} CanDevConfig;
```

其中,node_num为MCP2515的节点号(0～4),baudrate为CAN总线通信速率,filter为报文滤波配置结构,br_flag用于标识波特率配置是否有效,fi_flag用于标识报文滤波配置是否有效。baudrate和filter的数据结构类型定义如下:

```
typedef enum {
    _125kbps,
    _250kbps,
    _500kbps,
    _1Mbps
} CanBaudRate;

typedef struct{
    unsigned int mask_0;
    unsigned int mask_1;
    unsigned int filter_0;
    unsigned int filter_1;
    unsigned int filter_2;
    unsigned int filter_3;
    unsigned int filter_4;
    unsigned int filter_5;
}CanFilter;
```

(2) 工作模式配置结构体

```
typedef struct{
    unsigned char node_num;
    unsigned char oper_mode;
} CanDevMode;
```

其中,node_num意义同上,oper_mode表示该节点的工作模式。MCP2515共有5种工作模式,分别是配置模式、休眠模式、仅监听模式、回环模式和正常模式。一般设备都工作在正常模式。

3.1.3 环形数据接收缓冲区结构体

```
typedef struct {
    CanFrame can_recv_buf[RECV_BUF_SIZE];
    int recv_pos;
    int read_pos;
    wait_queue_head_t   wq;
} CanDev;
```

其中,can_recv_buf为接收CAN报文帧环形数据缓冲区,recv_pos和read_pos分别表示数据存入和读出缓冲区的位置;wq定义的是一个等待队列,用于实现阻塞型read操作。

3.2 驱动程序接口

驱动程序的接口主要分为3个部分:初始化与退出函数接口,完成设备安装和卸载等操作;文件系统接口,由file_operations数据结构来完成;与设备的接口,完成对设备的读/写等操作。

3.2.1 初始化与退出函数

在安装驱动程序时,操作系统会调用初始化函数进行设备注册、设备初始化以及安装中断处理例程等操作。参考文献[3]详细论述了设备注册的方法,而这里主要讨论设备初始化时的配置方法。在本驱动程序中,设备初始化分两步:一是对9200的SPI控制器初始化,二是对5个MCP2515初始化。

在卸载设备驱动程序时会调用退出函数,退出函数主要完成设备的注销和中断释放。

参考文献[3]详细论述了中断处理例程的安装、设备注销和中断释放的方法,此处不再详述。

3.2.2 中断接收服务例程

MCP2515收到CAN报文帧后,产生中断并将$\overline{\text{INT}}$引脚置低。9200响应外部中断,并调用和外部中断相对应

的中断处理例程。中断处理例程共有 3 个：at91_mcp2515_irq_handler_0 响应 IRQ0 的中断，at91_mcp2515_irq_handler_1_2 响应 IRQ1 的中断，at91_mcp2515_irq_handler_3_4 响应 IRQ2 的中断。其中 IRQ0 只和一个 MCP2515 相连，而 IRQ1 和 IRQ2 分别被两个 MCP2515 所共享。IRQ0 和 IRQ1 的中断处理流程分别如图 5 和图 6 所示，IRQ2 与 IRQ1 的中断处理流程相同。

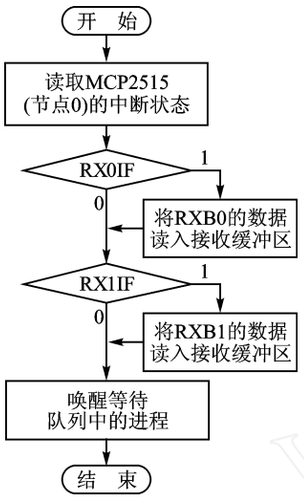

图 5 IRQ0 中断处理流程

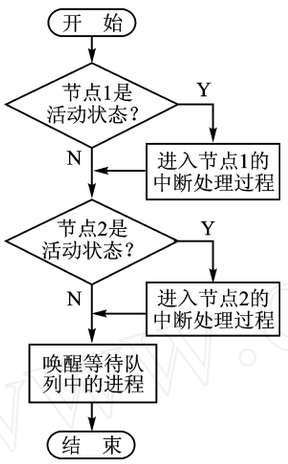

图 6 IRQ1 中断处理流程

需要注意的是，在图 5 的处理流程中并没有清中断操作。这是因为采用了 RX 读缓冲区指令读取 MCP2515 RX 缓冲区中的数据。该指令操作结束后，MCP2515 会自动清除相应的接收中断标志位。

3.2.3 文件系统接口定义

文件系统接口 struct file_operations 的成员全部是函数指针，这些指针指出了设备驱动程序所提供的入口点位置。本驱动程序所定义的 file_operations 为：

```
static struct file_operations at91_mcp2515_fops = {
    owner:      THIS_MODULE,
    write:      at91_mcp2515_write,
    read:       at91_mcp2515_read,
    ioctl:      at91_mcp2515_ioctl,
    open:       at91_mcp2515_open,
    release:    at91_mcp2515_release,
};
```

3.2.4 ioctl 函数

ioctl 函数用于对设备进行配置。我们在 ioctl 函数中实现了两个命令：IOCTRL_CONFIG_CAN_DEV 用于配置节点的 CAN 总线波特率和报文滤波，IOCTRL_SET_OPER_MODE 用于配置节点的工作模式。这两种配置命令所对应的配置参数都是指向应用层相应数据结构的指针，两个配置结构在 3.1.2 小节已经介绍过了。

用 IOCTRL_CONFIG_CAN_DEV 命令配置波特率和报文滤波时，在配置完成后，如果节点处于 INACTIVE 状态，则需要使能节点内部的接收中断，使能节点所对应的外部中断，并将节点状态设置为 ACTIVE。在通常情况下，通过 ioctl 函数对需要配置的节点执行完 IOCTRL_CONFIG_CAN_DEV 命令后，还要再对配置过的节点执行 IOCTRL_SET_OPER_MODE 命令，使节点处于正常的工作模式。

3.2.5 关于竞争问题

本系统是单 CPU 系统，采用 Linux 2.4.19 内核，且是非抢占式的；同时，此设计的驱动程序也只允许一个进程打开并操作该设备。在这种情况下，驱动程序中所涉及的竞争问题主要就是中断处理程序内核代码和其他设备操作的内核代码之间的资源竞争。在上文中所提到的所有设备操作中，都要通过 9200 的 SPI 接口与 MCP2515 进行通信。9200 与 MCP2515 进行通信都是以命令字节开始的，并且在一个命令操作过程中(一般会连续传输多个字节)，片选和时钟是不能被禁用的，否则操作就会失败。因此，MCP2515 的一个完整的命令操作就是一个临界区域，在对 MCP2515 进行一个命令操作的过程中必须禁用所有的中断，以保证命令操作的正常执行。在驱动程序中，采用 local_irq_save 和 local_irq_restore 函数对中断禁用和恢复，在这两个函数调用之间，就是对 MCP2515 执行一个命令操作的代码。

结 语

本文针对特有的应用需求提出的多路 CAN 总线接口和驱动程序设计，经过测试，可以稳定正常地运行。关于驱动程序的编译和运行方法，参考文献[3]有很好的说明。上层的测试程序编写也比较简单，但要注意数据结构的定义和底层驱动程序的一致性。本文侧重介绍设计的基本方法和实现基本的功能。MCP2515 本身提供了许多的功能，在实现基本功能的基础上，也可以根据自己的应用需要再进行功能扩展。

编者注：本文为期刊缩略版，全文见本刊网站 www.mesnet.com.cn。

参考文献

[1] Atmel corporation. AT91RM9200 data sheet, 2006.
[2] Microchip Technology Inc. MCP2515 data sheet, 2005.
[3] 魏永明，骆刚，姜君. LINUX 设备驱动程序[M]. 第 2 版. 北京：中国电力出版社，2004.
[4] 杜尚丰，曹晓钟，徐津. CAN 总线测控技术及其应用[M]. 北京：电子工业出版社，2007.

徐晓东(硕士研究生)，主要研究方向为嵌入式系统及其应用；路红英(高级工程师)，主要研究方向为计算机应用和多媒体技术；张宁(教授)，主要研究方向为计算机测控、视频技术以及嵌入式系统。

(收稿日期：2007-10-31)