CAN/GPRS 车载网关设计与应用

张立国[1] 丁志刚[2,3] 彭娟春[2,3]
[1](上海市计算技术研究所 上海 200040)
[2](上海计算机软件技术开发中心 上海 201112)
[3](上海嵌入式系统应用工程技术研究中心 上海 201112)

摘 要 汽车内部网的应用和无线车载数据的传输受到越来越多的重视。从应用的角度详细描述了车载 CAN/GPRS(Controller Area Network/General Packet Radio Service)网关的硬件设计和软件设计。同时,针对目前将 CAN 设备驱动实现为字符设备驱动带来的缺点,提出了将 CAN 设备驱动实现为网络设备驱动,使用 Socket 进行 CAN 通信。最后,通过测试验证 Socket CAN 通信和 GPRS 无线联网。

关键词 Linux PXA3xx CAN GPRS

DESIGN AND APPLICATION OF CAN/GPRS VEHICULAR GATEWAY

Zhang Liguo[1] Ding Zhigang[2,3] Peng Juanchun[2,3]
[1](*Shanghai Institute of Computing Technology, Shanghai 200040, China*)
[2](*Shanghai Development Center of Computer Software Technology, Shanghai 201112, China*)
[3](*Shanghai Engineering Technology Research Center of Embedded System Application, Shanghai 201112, China*)

Abstract The application of automobile internal networks and the transmission of wireless vehicular data are attracting growing attentions. In terms of the application, we expatiate on the designs of hardware and software of CAN/GPRS vehicular gateway in the paper. Meanwhile, in light to the drawback brought about by implementing CAN device driver to the character driver, we propose to implement CAN device driver to the network devices drivers and using Socket to conduct CAN communication. At last, we validate Socket CAN communication and GPRS wireless internet by test.

Keywords Linux PXA3xx CAN GPRS

0 引 言

控制器局域网 CAN 为串行通信协议,采用差分信号传输,错误检测能力强,通信距离远,能有效地支持高安全等级分布实时控制。CAN 总线由于其成本低廉、可靠性高、抗干扰能力强、通信方式灵活等特点,目前已经广泛应用于汽车内部网络中[1]。

然而现在汽车还需要和外界交换数据,许多信息需要无线传输,所以选择具有覆盖面广、接入速度快、按流量收费等优点的 GPRS 进行无线数据传输。

目前 Linux 下 CAN 设备大多被视为字符设备,存在许多局限。因此,文中提出将 Marvell PXA3xx 处理器上扩展的独立 CAN 控制器 MCP2515 驱动设计成网络接口驱动,使用 Berkeley Socket API 和 Linux 网络协议栈进行 Socket CAN 通信。事实证明,使用 Socket 进行 CAN 的通信,不仅解决了字符设备驱动固有的局限性,而且使 CAN 和 GPRS 在应用层都能用 Socket 进行编程,操作更加方便。这样,汽车内部数据通过 CAN 总线进行传输,而与外界交互的数据则通过 CAN/GPRS 无线网关进行转发。

1 CAN/GPRS 网关硬件结构

网关硬件平台采用 Marvell 最新的基于 XScale 架构的 PXA3xx 处理器。PXA3xx 是一款高主频低功耗的嵌入式处理器,其主频高达 624MHz,为手持式设备提供了很强的处理能力和更长的续航能力。外围设备包括 LCD 触摸屏、100M LAN、音频解码处理芯片、6 个带 LED 灯的矩阵键盘、SD 卡接口、UART 以及 SPI 控制器等[2]。

独立 CAN 控制器 MCP2515 支持 CAN 总线的 V2.0B 技术规范,支持最大 1Mb/s 的可编程波特率。MCP2515 能发送和接收标准和扩展数据帧以及远程帧,其自带的 2 个验收屏蔽寄存器和 6 个验收滤波寄存器可以用于过滤掉不想要的报文,MCP2515 通过 SPI 接口与 CPU 连接[3]。

GE863-GPS 是目前市场上 GPRS/GPS 二合一模块中最小的

收稿日期:2010 - 01 - 25。上海市科委重大项目(08dz1500405)。
张立国,硕士生,主研领域:嵌入式系统软件。

的通信模块。基于 Telit 成熟的 GPRS 核心技术,集成了最新的高灵敏度的 Sirf starIII 芯片 GPS 接收机。其独特的 BGA 封装,对设计空间要求比较严格的应用场合来讲有重要的意义[4]。

网关的硬件结构如图 1 所示。

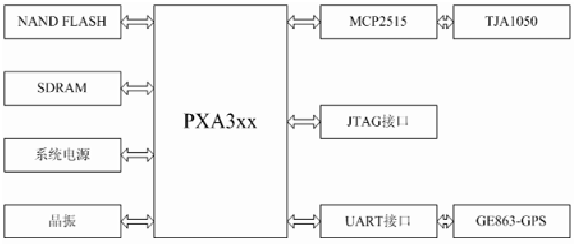

图 1　CAN/GPRS 网关硬件结构

2　CAN/GPRS 网关软件设计

2.1　Socket CAN 简介

Socket CAN 的设计克服了将 CAN 设备驱动实现为字符设备驱动带来的局限性,因为字符设备驱动直接操纵控制器硬件进行帧的收发、帧的排队以及一些高层传输协议只能在应用层进行实现。另外,字符设备驱动只能单进程进行访问,不支持多进程同时操作。

Socket CAN 的设计基于新的协议族 PF_CAN。协议族 PF_CAN 一方面向应用程序提供 Socket 接口,另一方面它构建在 Linux 网络体系结构中的网络层之上,从而可以更有效地利用 Linux 网络子系统中的各种排队策略。CAN 控制器被注册成一个网络设备,这样控制器收到的帧就可以传送给网络层,进而传送到 CAN 协议族部分(帧发送过程的传递方向与此相反)。同时,协议族模块提供传输层协议动态加载和卸载接口函数,更好地支持使用各种 CAN 传输层协议(目前协议族中只包括两种 CAN 协议:CAN_RAW 和 CAN_BCM)。此外,协议族还支持各种 CAN 帧的订阅,支持多个进程同时进行 Socket CAN 通信,每个进程可以同时使用不同协议进行各种帧的收发[5]。

2.2　CAN 子系统[6]

CAN 子系统实现协议族 PF_CAN,主要包括三个 C 文件:af_can. c、raw. c 和 bcm. c。其中 af_can. c 是整个子系统的核心管理文件,raw. c 和 bcm. c 分别是 raw 和 bcm(broadcast manager)协议的实现文件。CAN 子系统与其他模块的关系如图 2 所示。

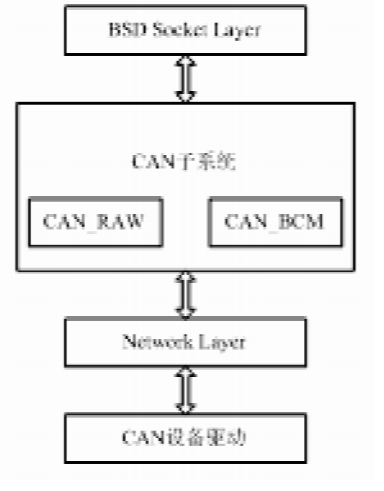

图 2　CAN 子系统与其他模块的关系

af_can. c 是 Socket CAN 的核心管理模块。can_create( )创建 CAN 通信所需的 socket,当应用程序调用 socket( )创建 socket 时,就会间接调用此函数;can_proto_register(struct can_proto *)和 can_proto_unregister(struct can_proto *)可被用来动态加载和卸载 CAN 传输协议,传输协议在此用 struct can_proto 表示。CAN_RAW 和 CAN_BCM 协议分别定义如下:

```
static struct can_proto raw_can_proto __read_mostly = {
    .type = SOCK_RAW,
    .protocol = CAN_RAW,
    .capability = -1,
    .ops = &raw_ops,
    .prot = &raw_proto
};
static struct can_proto bcm_can_proto __read_mostly = {
    .type = SOCK_DGRAM,
    .protocol = CAN_BCM,
    .capability = -1,
    .ops = &bcm_ops,
    .prot = &bcm_proto
};
```

can_rcv( )是网络层用来接收包的操作函数,与包对应的操作函数通过定义 struct packet_type 来指明:

```
static struct packet_type can_packet __read_mostly = {
    .type = __constant_htons(ETH_P_CAN),
    .dev = NULL,
    .func = can_rcv
};
```

can_rcv( )中调用 can_rcv_filter( )对收到的 CAN 帧进行过滤处理,只接收用户通过 can_rx_register( )订阅的 CAN 帧;与 can_rx_register( )对应的函数 can_rx_unregister( )用来取消用户订阅的 CAN 帧。

raw. c 和 bcm. c 分别是协议族里用来实现 raw socket 和 bcm socket 通信所需的文件。利用 CAN_RAW 和 CAN_BCM 协议均能实现帧 ID 的订阅,并且利用 CAN_BCM 协议还能进行帧内容的过滤。文件中各个函数之间的调用关系如图 3 所示。

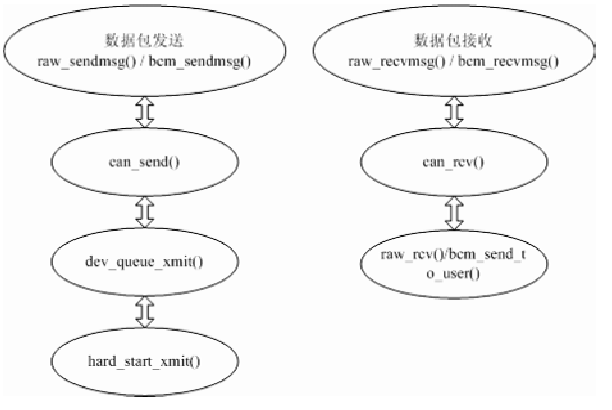

图 3　函数之间的调用关系

其中,raw_rcv( )和 bcm_send_to_user( )是传输层用来接收 CAN 帧的操作函数,当 can_rcv( )接收到用户订阅的帧时,就会调用这两函数将帧放到接收队列中,然后 raw_recvmsg( )或 bcm_recvmsg( )就会通过调用 skb_recv_datagram( )来从接收队列中取出帧,从而把帧进一步传送到上层;

raw_sendmsg( )和 bcm_sendmsg( )通过调用 can_send( )进行帧的发送,而 can_send( )会调用 dev_queue_xmit( )来向 CAN

设备传输一个 CAN 帧，然后 dev_queue_xmit( )会间接调用 CAN 设备驱动 hard_start_xmit( )来真正实现帧的发送。

2.3 Socket CAN 驱动程序设计

Linux 驱动程序是 Linux 内核主要组成部分，其功能主要是操作硬件设备，为用户屏蔽设备的工作细节，并向用户提供透明访问硬件设备的机制。Linux 驱动程序支持 3 种类型的设备：字符设备、块设备和网络设备[7]。

独立 CAN 控制器 MCP2515 实现 CAN 协议的物理层与数据链路层，本文的应用中将其作为网络设备处理。同时，MCP2515 通过 SPI 接口连接到 PXA3xx SPI 控制器，其也被视为 SPI 从设备。因此，驱动程序不仅要操作 CAN 控制器，而且要能很好地与 Linux CAN 子系统、网络子系统、Linux SPI 核心进行交互。CAN 网络设备的定义如下：

```
struct net_device can_dev = {
.type = ARPHRD_CAN,
.mtu = sizeof(struct can_frame),
.tx_queue_len = 20,
.flags = IFF_NOARP,
.features = NETIF_F_NO_CSUM,
.open = mcp2515_open,
.stop = mcp2515_stop,
.hard_start_xmit = mcp2515_hard_start_xmit,
.tx_timeout = mcp2515_tx_timeout,
.watchdog_timeo = HZ
};
```

结构体中部分成员的功能描述如下：

mcp2515_open( )负责对 CAN 控制器进行复位初始化，并且调用 netif_wake_queue( )通知上层启动新的传输。

mcp2515_stop( )使控制器停止工作并让其睡眠。

mcp2515_hard_start_xmit( )负责将帧数据写到控制器的相应寄存器和发送缓冲区中，并进行发送。

mcp2515_tx_timeout( )在发送超时时被调用，用来提示用户。

网络设备发送超时时间，由 dev_watchdog( )使用，监视发送过程。

SPI 从设备信息定义如下：

```
static struct spi_board_info mcp2515_spi_board_info = {
  .modalias = "mcp2515",
  .max_speed_hz = 1000000,
  .bus_num = 1,
  .platform_data = &mcp2515_info,
  .irq = IRQ_GPIO(81)
};
static struct mcp2515_platform_data mcp2515_info = {
  .oscillator_frequency = 16000000,
.bitrate = 400000 //400kbps
};
```

另外，驱动中还涉及几个比较重要函数。mcp2515_can_probe( )在驱动被注册时被调用，主要功能是定义并初始化 CAN 网络设备、SPI 从设备，调用 request_irq( )向系统申请 IRQ 资源，配置 SPI 总线以及 SPI 控制器，最后调用 register_netdev( )进行网络设备注册；mcp2515_hw_rx( )完成将收到的帧封装成 struct sk_buff 结构，然后调用 netif_rx( )将数据递交给上层处理；mcp2515_irq_work_handler( )负责处理来自控制器的中断，主要有 4 种中断：唤醒中断、错误中断、发送中断以及接收中断。通过控制器中断寄存器的标志位来判断中断类型。若是唤醒中断，则执行 mdelay(10)延迟 10ms，等待控制器被唤醒；若是错误中断，则根据错误标志寄存器的内容，构造相应错误类型的错误帧，然后调用 netif_rx( )将其递交给上层；若是发送中断，则调用 netif_wake_queue( )，允许上层调用 mcp2515_hard_start_xmit( )进行帧的发送；若是接收中断，则调用 mcp2515_hw_rx( )进行帧的接收。

2.4 Linux 下 GPRS 联网的实现

GPRS 联网是通过 PPP 协议来完成的。PPP 是在串行连接的数据链路层上实现 IP 以及其他网络协议的一种机制。Linux 下实现 GPRS 联网，首先需要增加 Linux 内核对 PPP 协议的支持。接着移植 Linux 下 PPP 的应用程序工具包，把交叉编译后得到的 pppd，chat，ppp-on，ppp-on-dialer 程序加入根文件系统。还需编写 3 个脚本 gprs，gprs-connect 和 gprs-disconnect。其中，gprs 脚本是用来设置 GE863-GPS 的启动信息，包括串口波特率、数据的接收形式、定义 gprs-connect、gprs-disconnect 脚本属性等。gprs-connect 脚本用来向 GE863-GPS 发送接入 GPRS 网络 AT 命令。gprs-disconnect 脚本用来向 GE863-GPS 发送断开 GPRS 网络的 AT 命令。

由于 GPRS 网络底层是基于 TCP/IP 协议的，因此当向外界发送数据时，只需使用 Socket 接收函数把 CAN 数据包中的数据域提取出来，并通过 Socket 发送出去，这时 Linux 会自动给数据加上 UDP 或 TCP 包头，这样 CAN 数据包就被转化为 UDP 或 TCP 数据包并发送出去了；当从外界接收数据时，仍使用 Socket 接收 UDP 或 TCP 数据包，并通过 Socket 接收函数把 UDP 或 TCP 数据包数据域中的数据提取出来，并通过 Socket 发送函数把数据发送到 CAN 总线上[9]。

3 系统测试与分析

3.1 Socket CAN 通信测试与分析

Socket CAN 的通信测试是通过 CAN 总线，将导航控制平台与 NI 公司的 PXI-8106 嵌入式控制器进行连接来完成。PXI 上自带 CAN 的板卡，可利用 LabVIEW[9] 软件进行 CAN 帧的收发。

测试的应用程序同一般的网络通信程序类似，核心代码分别如下：

1）测试程序一（利用 raw 协议，只接收 ID 为 5 的帧）：

```
s = socket (PF_CAN, SOCK_RAW, CAN_RAW);
rfilter[0].can_id = 0x5;
rfilter[0].can_mask = 0x5;
setsockopt(s, SOL_CAN_RAW, CAN_RAW_FILTER, &rfilter, sizeof(rfilter));
bind (s, (struct sockaddr *)&addr, sizeof(addr));
write (s, &frame, sizeof(struct can_frame));
read (s, &frame, sizeof(struct can_frame));
```

2）测试程序二（利用 bcm 协议，进行帧内容过滤，只接收 ID 为 0x123，最高数据字节与刚收到的前一帧不同的帧）：

```
struct {
    struct bcm_msg_head msg_head;
    struct can_frame frame[4];
} txmsg,rxmsg;
s = socket(PF_CAN, SOCK_DGRAM, CAN_BCM));
```

```
txmsg. msg_head. opcode = RX_SETUP;
txmsg. msg_head. can_id = 0x123;
txmsg. msg_head. nframes = 1;
U64 _ DATA ( &txmsg. frame [ 0 ]) = ( _ _ u64 ) 0xFF00000000000000ULL; //frame content mask
connect( s, ( struct sockaddr * )&addr, sizeof( addr));
write ( s, &txmsg, sizeof( txmsg));
read ( s, &rxmsg, sizeof( rxmsg));
```

Socket CAN 通信测试过程如下:

1) 导航控制平台同时运行两测试程序,分别发送 5 个 CAN 帧。同时,在 PXI 端,通过 LabVIEW 接收到的帧情况如图 4 所示。

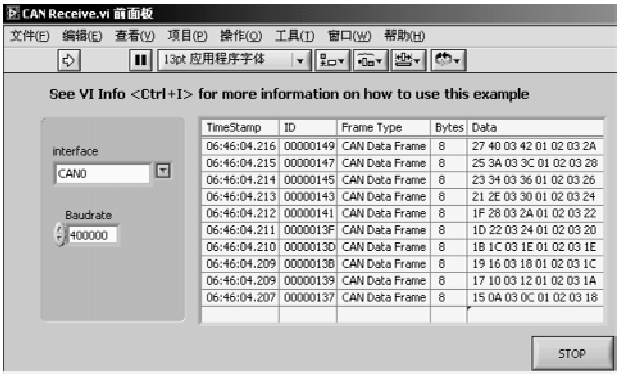

图 4　LabVIEW 接收到的帧

2) 在 PXI 端,利用 LabVIEW 连续发送 5 个 ID 为 0x24、5 个 ID 为 0x5、10 个 ID 为 0x123(其中 6 个最高数据字节为 0x13)的 CAN 帧。导航控制平台同时运行两测试程序接收到的帧情况如图 5 所示。

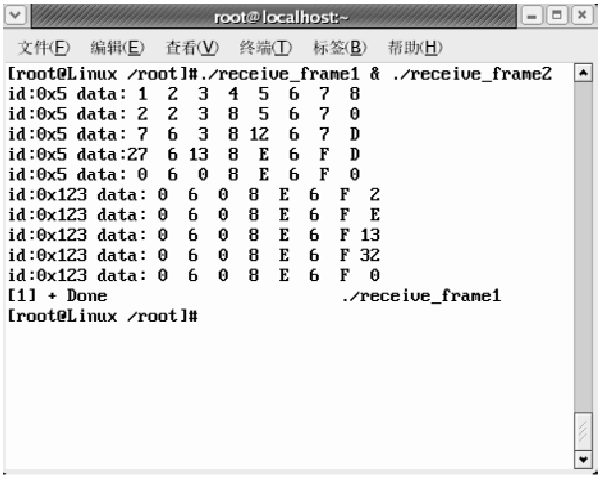

图 5　测试程序接收到的帧

图 4 表明,PXI 端 LabVIEW 正确地接收到了导航控制平台两测试程序发送的 10 个帧;图 5 表明,导航控制平台两测试程序分别接收到了各自订阅的帧,舍弃了不满足要求的帧。因此,利用 Socket CAN,不仅可以实现正确的 CAN 网络通信,而且支持多个进程同时使用不同的协议进行帧的订阅。同时,它使得应用程序与驱动程序之间的耦合性变弱,驱动程序的更改不会影响到应用程序。这些都很好地克服了字符设备驱动带来的缺点。

3.2　GPRS 联网测试与分析

GPRS 联网测试是通过在硬件平台上插入一张 SIM 卡,然后运行 Linux 下 PPP 相关的应用程序来进行。测试结果如图 6 所示。

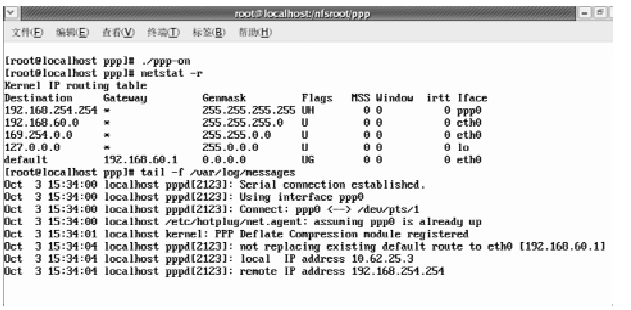

图 6　PPP 拨号接入 GPRS 网络

图 6 表明 GPRS 模块已经获取了本地动态 IP 地址 10.62.25.3,证明该模块已经接入 GPRS 网络。

4　结　语

本文将 CAN 控制器 MCP2515 驱动设计成网络接口驱动,利用 Socket 进行 CAN 通信,克服了将 CAN 设备驱动实现为字符设备驱动带来的局限性,同时通过 Linux PPP 协议实现了 GPRS 无线联网,这些都为 CAN/GPRS 车载网关的实现提供了方便。所设计的网关设备在实验运行过程中,能长时间地正常稳定运行,基本上达到了设计的预期目标。

参　考　文　献

[1] 王彦堂,李贻斌,宋锐. 基于 ARM Linux 平台的 CAN 设备驱动程序设计与实现[J]. 计算机工程与应用,2007,43(15):79-82.

[2] Marvell Corp. Marvell PXA3xx Processor Family Developer Manual [EB/OL]. 2008[2009-10-10],http://www. marvell. com.

[3] Microchip Corp. MCP2515 Datasheet [EB/OL]. 2005[2009-10-10],http://www. microchip. com.

[4] Telit Corp. GE863_Product_Description [EB/OL]. 2007[2009-10-10],http://www. telitchina. com.

[5] Urs Thuermann, Oliver Hartkopp. Socket CAN [EB/OL]. 2008[2009-10-10], http://lxr. linux. no/linux + v2. 6. 28. 5/Documentation/networking/can. txt.

[6] Urs Thuermann, Oliver Hartkopp. Linux CAN Subsystem[CP/OL]. 2008[2009-10-10], http://lxr. linux. no/linux + v2. 6. 28. 5/net/can/.

[7] Jonathan Corbet, Alessandro Rubini, Greg Kroah-Hartman. LINUX 设备驱动程序[M]. 魏永明,耿岳,钟书毅,译. 中国电力出版社,2007.

[8] 农毅. 基于 CAN 总线和 GPRS 的无线车载数据传输[J]. 计算机工程,2008,34(18):239-245.

[9] National Instrument Corp. LabVIEW Programmer Reference Manual [EB/OL]. 1998[2009-10-10]. http://www. ni. com.

(上接第 81 页)

[6] David Crandall, Dan Cosley, Daniel Huttenlocher. Feedback Effects between Similarity and Social Influence in Online Communities[C]// KDD'08, August 24 - 27, 2008, Las Vegas, Nevada, USA.

[7] Borah B, Bhattacharyya DK. An Improved Sampling-Based DBSCAN for Large Spatial Databases[C]//0-7803-8243-9104117. 000 2004 IEEE.

[8] Viswanath P. Rajwala Pinkesh. l-DBSCAN: A Fast Hybrid Density Based Clustering Method[C]//Proceedings of the 18th International Conference on Pattern Recognition (ICPR'06).