基于 MCF51JM128 的 USB-CAN 适配器①

张剑武，陈闻杰，琚小明

(华东师范大学 软件学院，上海 200062)

摘 要：为了实现车载计算平台与汽车内部电子模块的数据通信，通常需要设计一个 CAN 总线适配器。提出了一种基于 USB 的 CAN 总线适配器的设计与实现方法。在设计中采用了内部集成有 USB OTG 模块和 MSCAN 模块的 MCF51JM128 微控制器，使得适配器在设计和实现上简单易行。同时对该适配器在 Linux 系统上驱动程序的实现方法作出了描述。

关键词：USB-CAN 适配器；CAN 总线；USB 驱动；汽车电子；MCF51JM128

USB-to-CAN Adapter Based on MCF51JM128

ZHANG Jian-Wu, CHEN Wen-Jie, JU Xiao-Ming

(Software Engineering Institute, East China Normal University, Shanghai 200062, China)

Abstract: In order to achieve the communication between on-board computer and vehicle sub-systems, in general, we need to design a CAN bus adapter. This article describes one method to design and implement a CAN bus adapter based on USB. As USB OTG module and MSCAN module is integrated in the MCF51JM128 microcontroller, the design and implementation of the adapter of MFC51JM128 microcontroller is simpler and easier. The way to realize the adapter driver on Linux system is also included.

Keywords: USB-CAN Adapter; CAN Bus; USB Driver; auto-electronic; MCF51JM128

1 引言

随着汽车电子化和信息化的不断深入，汽车诊断系统越来越多的被使用。基于 OSGi 的远程汽车诊断系统，它通过 OSGi 的平台，实时获取或按需获取传感器侦测到的信息，并通过无线接口(3G)发送回服务商的测试系统进行分析，得出诊断结论和维修建议。在该系统中计算平台和汽车内部 OBD(On Board Diagnostic)系统通信时，需要用到一个 CAN(Controller Area Network)总线适配器[1]。CAN 是一种现场总线，它有效的支持了分布式控制系统与实时系统，被广泛应用于汽车内部[2]。目前 CAN 总线适配器的设计方法主要有：(1)采用 ASIC 或 FPGA 器件来实现，该方法有很大的灵活性，可以让器件达到最高的使用效率，但开发成本较高，开发难度较大[3]；(2)使用微控制器、CAN 总线控制器和 USB 总线控制器等器件来实现，该方法成本较低，设计和实现较为容易，但缺少灵活性[4]。本文提出了一种基于 MCF51JM128 微控制器的 USB 接口的 CAN 总线适配器，它充分利用了 MCF51JM128 高集成度的优点，电路设计简单，固件程序结构清晰、易于开发，同时也具有 USB 设备传输速度高易于扩展使用灵活的优点。

2 适配器设计概述

USB-CAN 适配器使用到的器件主要为 MCF51J－M128 微控制器，该控制器内部集成有 USB 总线控制器和 CAN 总线控制器。适配器设计的核心是固件程序的开发，适配器固件程序主要有三部分构成：USB 模块驱动、CAN 模块驱动和 USB-CAN 适配器应用程序。如下图 1 所示，适配器应用程序通过调用 USB 驱动来接收 USB host 的命令，并进行处理，应用程序通过调用 MSCAN 驱动程序从 CAN 总线上接收数据，再通过调用 USB 驱动把这些数据转送到 USB host。

① 收稿时间:2010-08-16;收到修改稿时间:2010-09-11

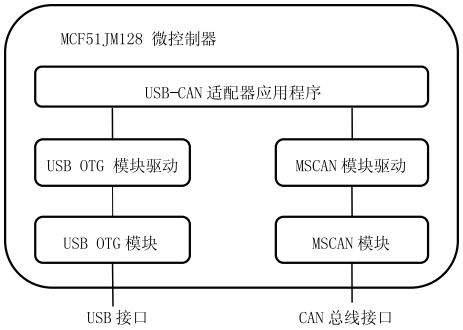

图 1　适配器系统框图

下面将分别从适配器的电路设计、固件设计和驱动开发三个方面进行阐述，其中固件设计部分将分别阐述 USB 模块驱动、MSCAN 模块驱动和应用程序三个部分。

3　适配器硬件设计

如下图 2 所示，适配器主要由微控制器芯片 MCF51JM128 和 CAN 收发器芯片 PCA82C250 构成。MCF51JM128 是 Freescale 公司基于 V1 ColdFire 内核的一款 32 位微控制器芯片，其内部集成了 128KB 的 Flash 和 16KB 的 SRAM，具有 MSCAN 控制器模块、USB OTG 控制器模块等丰富的片内资源[5]。其中的 USB OTG 模块是一个双模式控制器，提供了有限的 USB 主机功能和 USB 2.0 全速设备功能，拥有 16 个双向的端点，支持 DMA 和 FIFO 数据流接口。MSCAN 模块实现了 CAN 2.0 A/B 规范，支持扩展帧和远程帧，具有 5 个可以按 FIFO 方式组织的接收缓冲存储区和三个具有优先级控制的发送缓冲存储区，其最高通信速率为 1Mbps。PCA82C250 是 Philips 半导体生产的 CAN

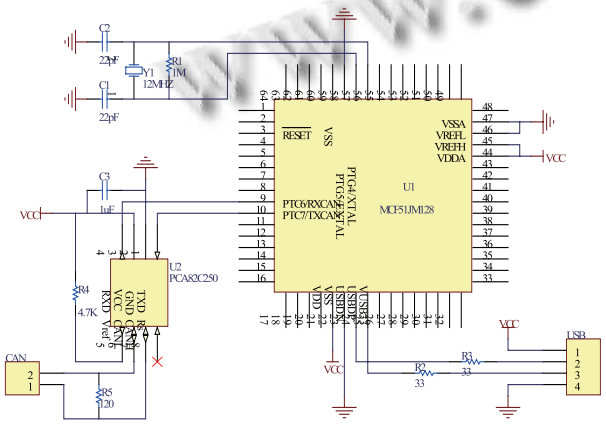

图 2　USB-CAN 适配器电路图

收发器，是 CAN 协议控制器和物理总线之间的接口，它对总线提供差动发送能力，对 CAN 控制器提供差动的接收能力，速度可达 1Mbps。

4　适配器硬件设计

适配器固件主要由三部分构成：USB OTG 模块驱动、MSCAN 模块驱动和 USB-CAN 适配器应用程序。USB OTG 模块驱动负责管理 USB OTG 控制器，处理对 USB 设备的枚举和请求，及侦听 USB 数据包等。MSCAN 模块驱动负责配置和初始化 MSCAN 模块，完成 CAN 信息帧的发送和接收。接收主机发送来的命令并执行，以及发送和接收 CAN 信息帧。

4.1 USB 模块驱动

USB OTG 模块驱动实现了 USB 通讯的设备端，它可被看作两层，底层是 USB 驱动，管理 USB 控制器，实现 USB 协议的协议控制，并对主机提出的请求作出相应的回应，上层是 USB 设备类协议，实现 USB-CAN 适配器的通信协议[6]。底层 USB 驱动使用由 Freescale 公司提供的 CMX USB 协议栈来实现，上层 USB 设备类驱动根据适配器选择的 USB 端点来实现。

在 USB 通信中需要使用到两类端点，一类是用于传输数据到外设的输出端点，另一类是用于传输数据到主机的输入端点。在 USB-CAN 适配器的设备类协议中，使用了 5 个端点，端点 0 用于 USB 设备的枚举和 USB 设备的标准控制，端点 1 和端点 3 用于主机传输命令和数据到 USB 设备，端点 2 和端点 4 用于设备传输命令和握手数据到主机。其中端点 1 和端点 2 的缓冲区大小为 16 字节，端点 3 和端点 4 的缓冲区大小为 32 字节。

在适配器 USB 端点上传输的 USB 包由三部分组成：外设编号、命令字段和数据字段。外设编号是对要被配置外设的唯一标识，它占用 USB 包的第一个字节。命令字段为需要被外设执行的动作的编码，占用 USB 包的第二个字节。数据字段为命令参数或返回命令时的格式数据，其长度因命令的不同而不同。

usb 模块驱动程序提供了 usb_init()、usb_send()、usb_receive()、usb_ep_is_busy()及 usb_get_state()等函数。其中 usb_ep_is_busy()用来检查 USB 端点的状态，usb_get_state()用来返回当前 USB 设备的状态。固件中的应用程序通过调用这些函数实现适配器的 USB 通

信功能。

4.2 MSCAN 模块驱动

MSCAN 模块驱动程序通过对微控制器 MCF51JM128 中 MSCAN 模块的控制，实现了对 CAN 数据的发送、接收和存储。驱动程序使用中断方式接收 CAN 总线上的数据。当中断发生时，中断处理函数把接收到的数据以帧的形式存储到环形队列的接收缓冲区中，应用程序通过对该接收缓冲区的访问来取走数据。其中帧的数据定义如下：

```
typedef struct can_frame {
        uword8   id[4];
        uword8   data[8];
        uword8   length;
        uword8   priority;
        uword8   timeh;
        uword8   timel;
}s_can_frame;
```

驱动程序使用轮询方式发送数据到 CAN 总线上。当有数据需要发送时，驱动程序首先选择一个发送缓冲区，并检查该缓冲区是否有可用的空间。如果有，就复制要发送的数据到该缓冲区，否则标记该缓冲区的可以空间为 0。

MSCAN 驱动程序提供了 can_init()、can_tx()、can_rx()和 can_rxstat()等函数，应用程序通过这些函数接收 CAN 总线上的数据和发送数据。

4.3 适配器固件的应用程序

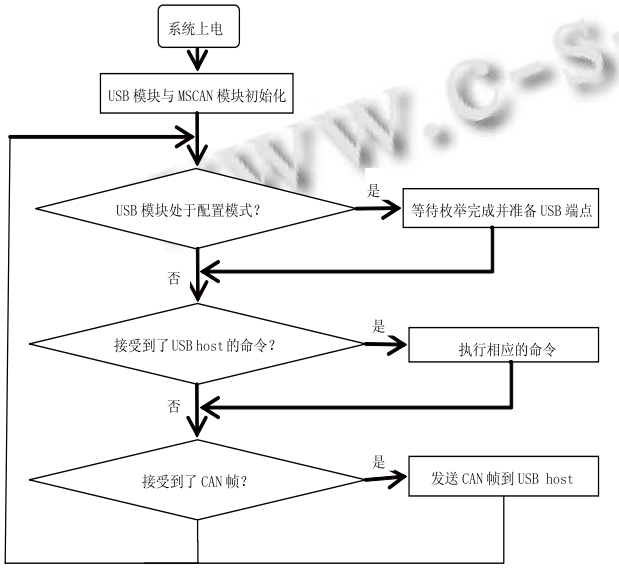

图 3 固件应用程序流程图

USB-CAN 适配器的核心是固件中的应用程序部分，其流程图如下图 3 所示。当系统加电时，程序首先完成对 USB 模块和 MSCAN 模块的初始化，在 USB 设备枚举完成之后，程序将准备好用于接收 USB host 命令的管道，应用程序将执行它的主要任务：接收并执行 USB host 发送来的命令，发送 CAN 总线上接收来的数据到 USB host。

根据 USB 协议的规定，USB 端点在配置之前，USB 设备必须被枚举并且进入配置模式。程序通过使用 USB 驱动的 API 函数 usb_get_state()来获得当前 USB 设备的状态，如果其返回状态为 USBST_CONFIGURED，则表明该 USB 设备是处于配置模式的。USB 驱动的 API 函数 usb_receive()实现了对 USB 管道的设置，在程序中对管道 1 和管道 3 进行了配置，用于接收 USB 主机的命令。在完成对 USB 端点的配置之后，程序将进入其主任务。在主任务中，函数 wait_for_commands()首先通过调用 USB 驱动的 API 函数 usb_ep_is_busy()检查 USB 端点 1 和端点 3 是否忙，如果不忙，则程序就从内部端点缓冲区接收命令和外设编号。接着函数 handle_commands()根据接收到的命令调用相应的命令处理函数。同时，程序通过函数 can_rxstat()检查是否从 CAN 总线上接收到了数据，如果接收到了，则程序调用函数 can_rx()从 MSCAN 驱动中取出接收到的数据，并调用函数 send_smg_to_host()把数据发送到 USB 主机。其中函数 send_msg_to_host()是调用 USB 驱动的 API 函数 usb_send()将数据通过端点 4 发送到 USB host 的。

5 Linux系统中适配器驱动程序的设计

Linux 系统提供了完整的 USB 驱动程序框架，根据适配器的 USB 端点设置，可以容易的开发出适配器的 Linux 驱动程序[7]。

USB-CAN 适配器的 Linux 设备驱动主要由两部分组成：usb_driver 的成员函数和适配器的打开、关闭、读和写等函数。其中适配器的 usb_driver 结构体 usb2can_driver 定义如下所示：

```
static struct   usb_driver   usb2can_driver =
{
    .name        =   “usb2can”,
    .probe       =   usb2can_probe,
    .disconnect  =   usb2can_disconnect,
```

```
    .id_table      =   id_table,
};
```

其成员函数主要包括 probe()函数和 disconnect()函数。probe()函数根据 usb_interface 的成员找出批量输入和输出端点，将端点地址、缓冲区等信息存入驱动程序定义的 usb2can 结构体中，并将 usb2can 结构体实例的指针传入 usb_set_intfdata()函数作为 USB 接口的私有数据，最后去注册 USB 设备。disconnect()函数完成与 probe()函数相反的工作，它设置接口数据为空，注销 USB 设备。其中 usb2can 结构体定义如下：

```
struct usb2can {
    struct usb_device      *udev;
    struct usb_interface   *interface;
    struct urb             *bulk_in_urb;
    unsigned char          *bulk_in_buffer;
    size_t                  bulk_in_size;
    size_t                  bulk_in_filled;
    size_t                  bulk_in_copied;
    __u8            bulk_out_endpoint_1_Addr;
    __u8            bulk_in_endpoint_2_Addr;
    __u8            bulk_out_endpoint_3_Addr;
    __u8            bulk_in_endpoint_4_Addr;
    struct kref             kref;
    struct semaphore        limit_sem;
    struct usb_auchor       submitted;
};
```

在 Linux 系统中该适配器为字符设备，其 open 函数通过 usb_find_interface()获取 usb 接口，再通过 usb_get_intfdata()函数获得接口的私有数据并赋予 file->private_data；release 函数进行资源的释放，减少在 open()中增加的引用计数；read 函数实现从适配器读取 CAN 数据；write 函数实现向适配器发送 CAN 数据；ioctl函数对适配器进行配置。

6 总结

本文介绍了一种 USB-CAN 适配器的设计与实现方法，并实现了该适配器在 Linux 系统上的驱动程序。与其他设计方法相比，该方案电路设计简单，固件程序结构清晰、容易实现。该适配器已成功应用到了基于 OSGi 的远程汽车诊断系统中，并达到了较为理想的效果。

致谢：本文是Intel亚太研发有限公司资助项目“Research on MID Architecture Vehicle Telematics Based on the GENIVI Platform”的一部分。

参考文献

1 黄光亮,秦树人,王见.基于车载诊断系统的汽车虚拟仪表.中国测试,2009,35(5):81−84.
2 蒋建文,林勇.CAN 总线通信协议的分析和实现.计算机工程,2002,28(2):219−221.
3 罗晓齐,毛君.基于 FPGA 的 CAN_USB 协议转换的研究.工业控制计算机,2009,22(11):36−38.
4 谢勤岚,曹汇敏.USB 与 CAN 总线转换电路的设计.计算机测量与控制,2009,17(9):1843−1845.
5 Freescale Semiconductor Inc. MCF51JM128 Coldfire Integrated Microcontroller Reference Manual.2008. [2010-3-27]. http://www.freescale.com/
6 姜波,陈英,胡涛,邹静.可移植的 USB 协议栈原理与技术研究.计算机工程与应用,2003,39(28):156−158.
7 金鑫,孙松林,景晓军.Linux 下有中断端点的 USB 设备驱动的实现.计算机工程与设计,2010,31(7):1576−1579.