基于 ARM 920T 嵌入式通信控制系统设备驱动开发

王松月,杨福兴
(北京邮电大学 自动化学院,北京 100876)

摘要: 传统的 8/16 位微控制器由于速度慢、功耗大且实现 Internet 协议困难,已经不能满足大量信息管理的需要。提出一种基于 ARM 920T 和 Linux 的嵌入式通信控制系统,介绍了系统架构和在工业系统的应用。论述了用动态模块加载法和静态编译进内核法开发系统外扩设备控制器局域网(CAN)总线以及 RS-422/RS-485 总线驱动程序的流程,分析了驱动的调试方法;对开发嵌入式系统驱动程序的 2 种方法作了对比。

关键词: 嵌入式系统; ARM; Linux; 驱动
中图分类号: TP 311.52;TP 36 **文献标识码**: B **文章编号**: 1006-6047(2006)06-0075-04

0 引言

通信管理机主要用于各智能设备及系统之间的数据交换。它是各设备之间数据交换的通信枢纽,是自动化系统不可缺少的重要设备。传统的 8/16 位微控制器由于速度慢、功耗大且实现 Internet 协议十分困难,已经越来越不能满足高速发展的工业系统对大量信息管理的需要。随着 ARM(Advanced RISC Machines)工业级芯片的不断成熟及嵌入式 Linux 系统的不断完善,开发一种基于 ARM 和 Linux 的嵌入式系统用于工业系统通信管理很有必要。而设备驱动是整个系统的重要组成部分,是连接软、硬件平台的桥梁。

本文提出一种基于 ARM 芯片 EP 9315 嵌入式通信控制系统,介绍了用动态模块加载法和静态编译进内核法开发系统外扩设备控制器局域网(CAN)总线和 RS-422/RS-485 总线驱动程序的过程。

1 系统整体架构及工作原理

所设计的嵌入式通信控制系统以 32 位 ARM 芯片 EP 9315 为核心,运行嵌入式 Linux 操作系统,可实现大容量信息采集与计算、通信管理和工业控制等功能。

EP 9315 是 Cirrus Logic 公司开发的基于 ARM 920 T 处理器核的工业级芯片,支持 32 位 ARM 指令集和 16 位 Thumb 指令集,5 级整数流水线,提供 1.1 MIPS/MHz 的哈佛结构,内存管理单元(MMU)支持 Window CE、Linux 等嵌入式操作系统。主频最高可达 200 MHz,配备 Maverick Crunch 协处理器,可用于浮点运算、DSP 运算和媒体处理,内置图像加速卡,图像显示和处理十分迅捷。芯片内部集成有串口 RS-232、通用串行总线(USB)、网口、外挂硬盘和音频等控制器,支持彩色液晶、触摸屏和小键盘,功能十分强大①。

其系统架构如图 1 所示。

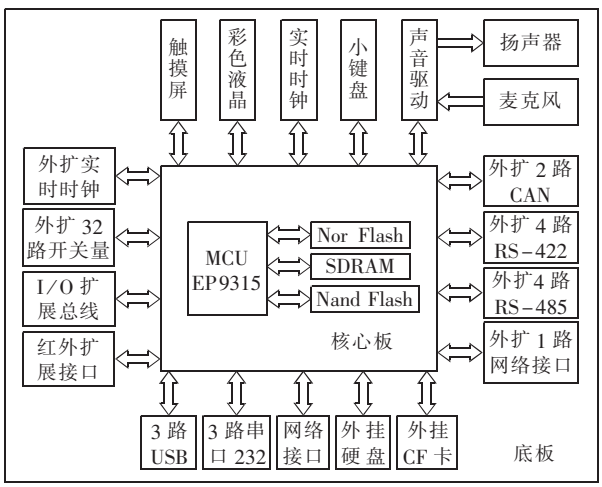


图 1 系统架构图
Fig.1 System architecture

为完善其通信管理功能,本系统外扩了 4 路电气隔离 RS-485/422 串口、2 路光电隔离 CAN V 2.0 总线、1 路变压器隔离 10/100 Mbit/s 以太网和 32 路开关量输入/输出功能,并且配备实时时钟,可以掉电保持。为增加其可靠性,系统设计有看门狗和电源监测电路,并且采取电磁滤波屏蔽(EMI)设计,使系统具有较强的抗干扰能力。

此嵌入式通信控制系统配置了异步串行收发(UART)、USB1.1、CAN 2.0B 和以太网等工业总线标准接口,可作为网关,通过简单的现场总线将其他 8/16 位微处理器(MCU)组成的嵌入式工业控制设备连接,下位机的数据和信息从 RS-422/RS-485 总线进入嵌入式控制器,经过协议转换后通过 Internet 发送到远程主机上。CAN 节点可收集模拟量输入、

收稿日期:2005-11-30;修回日期:2006-02-27

① EP 9315 User Manual. http://www.cirrus.com.

输出和开关量输入、输出等信息。通过CAN总线传送并存储在系统内存中,生成CAN数据文件,用户通过网络可访问和修改工业现场数据。

嵌入式控制器本身还可作为工业控制的1个节点使用,实时采集、存储、传输,通过键盘和LCD可查询设备状态和设置设备参数。该嵌入式通信控制系统将基于TCP/IP的以太网和现场总线技术有效地结合,使得孤立的现场设备作为网络节点有机地连接,易于集中化管理和控制,体现了网络化和开放性是当前工业控制领域的发展方向。

2 搭建系统软件开发环境

在设备驱动程序的开发过程中,调试内核和驱动程序采用Ftp-Telnet远程开发模式。如图2所示。

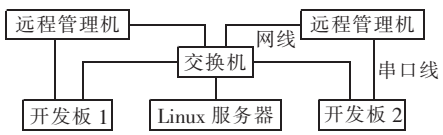


图2 Ftp及Telnet远程开发模式

Fig.2 Remote development mode of Ftp and Telnet

首先,在Linux服务器上建立交叉编译环境,交叉编译就是在一个系统平台上生成另一系统平台上的可执行代码。可以在www.kegel.com上下载交叉编译软件包crosstool-2.95.3.tar.gz,解压并修改/etc目录下的profile脚本文件,设置交叉编译工具路径,系统以后再编译程序时就会自动找到适合的编译工具。由于目标板平台基于ARM处理器,所有生成的交叉编译工具都会打上arm-linux-前缀。

开发人员通过网络用Telnet方式远程登陆到Linux服务器,把需要编译的内核或者驱动程序用文件传输协议(Ftp)传到服务器上,编译完成后把可执行代码用Ftp协议下载回自己主机,然后,通过网络下载到开发板上,通过串口控制开发板。

这种方法可使开发者在熟悉的Windows环境下开发调试,使用熟悉的编辑软件修改程序,避免了Linux开发界面单一的缺点和在不同界面下切换的麻烦。同时,多人可以共享1台高性能的Linux服务器,可以节约资源且方便高效。

嵌入式软件系统的结构可分为操作系统、设备驱动、应用中间件和应用系统4个层次。设备驱动程序在功能上分为2层:硬件控制层和接口封装层。硬件控制层负责对硬件模块按功能提供各种控制接口;接口封装层负责将硬件控制层封装成标准的应用接口[2]。

Linux内核提供了2种机制开发设备驱动程序:一种直接把驱动程序编译到内核中;另一种则是驱动程序编译成可动态加载和卸载的驱动模块[3]。以下分别介绍用这2种方法开发CAN总线和串口RS-422/RS-485的驱动程序。

3 动态模块加载法开发CAN总线驱动程序

CAN属于现场总线的范畴,它是一种有效支持分布式控制或实时控制的串行通信网络,本系统采用2片Philips公司生产的SJA1000作为独立的CAN控制器[4]。以下介绍驱动程序实现中的关键问题。

3.1 物理地址到虚拟地址的映射

EP9315内部带有MMU,用户不能直接对底层物理地址操作,必须在内核中将物理地址映射到相应的虚拟地址上,操作虚拟地址以读/写寄存器。有2种方法可实现这种映射[5]。

3.1.1 静态映射

在inux/include/asm-arm/arch-ep93xx/regmap.h中定义物理地址、虚拟地址和映射长度。

```
#define  CAN_BASE_VIRT  0xFFB00000
// CAN中虚拟地址
#define  CAN_BASE_PHYS  0x31E000000
// CAN中物理地址
#define  CAN_SIZE  0x00100000  // 长度
```

在linux/arch/arm/mach-ep93xx/mm.c中将上面的定义添加实现映射的结构中。

```
static struct map_desc ep93xx_io_desc[]__initdata={
// Virtual Address Physical Address Size
Domain R W C B
Last {IO_BASE_VIRT,IO_BASE_PHYS,IO_SIZE,
DOMAIN_IO,0,1,0,0},
{CAN_BASE_VIRT,CAN_BASE_PHYS,CAN_SIZE,
DOMAIN_IO,1,1,0,0} LAST_DESC};
```

3.1.2 动态映射

Linux在io.h头文件中声明了函数ioremap(),用于将I/O内存资源的物理地址映射到核心虚地址空间(3GB-4GB)中,函数原型如下:

```
void*ioremap(unsigned long phys_addr,unsigned
long size,unsigned long flags);
```

函数void iounmap(void *addr)用于取消ioremap()所作的映射,参数addr是指核心虚地址的指针。这2个函数都是实现在mm/ioremap.c文件中。

映射完成后,就可以象读/写RAM那样直接读/写系统的I/O内存资源了。

3.2 字符设备的注册

Linux系统的设备分为字符、块和网络3种设备,字符设备存取时只需较少缓存,应用程序可像访问文件一样使用标准系统调用打开、读、写和关闭它。

当字符设备初始化时,它的设备驱动程序向Linux核心登记,通过系统函数register_chrdev()将设备加入到系统设备列表中,在chadevs向量表中增加1个device_struct数据结构条目,该设备的主设备标识符(例如对于can设备是91)作为该向量表的索引,1个设备的主标识符是固定的。chadevs向量表中的device_struct数据结构包含2个元素:一个是登记的设备驱动程序名称的指针,另一个是指向一组文件操

作的指针。这块文件操作本身位于这个设备的字符设备驱动程序中，每个都处理特定的文件操作，比如打开、读、写和关闭，/proc/devices 中字符设备的内容来自 chrdevs 向量表。注销与注册相反，调用 unregister_chrdev()函数[6]。

3.3　CAN 总线驱动程序的 3 个主要组成部分

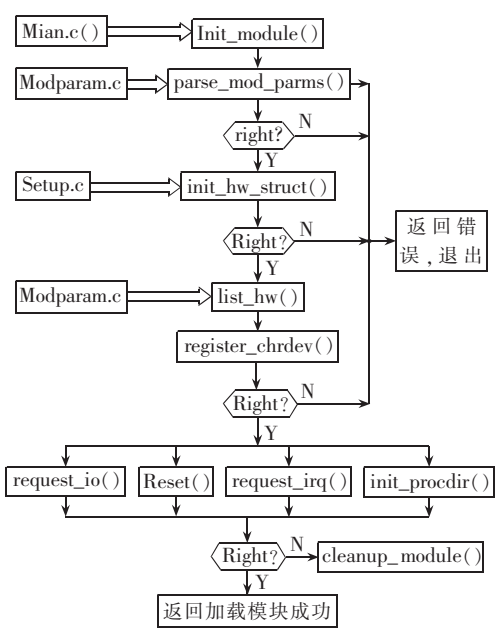
a. 自动配置和初始化子程序，完成硬件设备的检测和初始化。

b. 服务于 I/O 请求的子程序，控制硬件发送和接收数据，打开、关闭及属性设置。file-operations 结构为 Linux 提供的服务于 I/O 请求的子程序的代码实现提供了一系列入口点，和字符设备一起注册到系统中，在文件作用域中进行定义[7]。

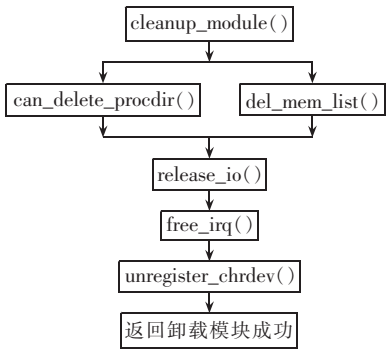
c. 中断服务子程序在 Linux 系统中，并不是直接从中断向量表中调用设备驱动程序的中断服务子程序，而是由 Linux 系统接收硬件中断，再由系统调用中断服务子程序。调用 request_irq()注册中断处理程序，free_irq()释放中断[8]。

3.4　CAN 驱动程序的加载

编写 Makefile 文件，将驱动程序交叉编译生成 can.o 的二进制可执行文件并下载到目标板，执行 insmod can.o，就可以将驱动程序加载到内核空间运行了。Linux 内核维护了一张所有内核所用资源的符号表(称为内核资源符号表)。当一个模块被加载，程序 init_module()函数开始执行，Linux 就会修改内核资源符号表，将该模块所提供的服务和资源加入进去。这样另一个模块载入时，如果需要就可以引用该模块的资源了。卸载一个模块时，程序从 clean_module()函数开始执行，需要知道当前模块是否正在被使用。如果没有被使用，在卸载时要能够通知该模块，以便由它自己释放已被它占用的系统资源。同时，Linux 还要从内核资源符号表中删除该模块提供的所有资源和服务[9]。程序模块加载和卸载流程图如图 3 所示。



(a) 模块加载流程



(b) 模块卸载流程

图 3 CAN 驱动程序动态流程

Fig.3 Dynamic flow of CAN driver program

4　静态编译进内核方法开发驱动程序

RS-422 与 RS-485 都是串行数据接口标准，数据信号采用差分传输方式。RS-422 支持单机发送、多机接收的单向、平衡传输规范；RS-485 标准增加了多点、双向通信能力[10]。本系统采用 Philips 公司生产的 St16C554D 作为复用驱动器。

4.1　串口驱动程序开发步骤

与 CAN 总线一样，串口也属于字符设备，驱动程序架构和 CAN 设备相似。以下仅介绍串口驱动程序 serial.c 静态编译进内核的方法。

步骤 1　改动 serial.c 的源代码。新建函数 int init_serial(void)，将设备注册在此处：

result=register_chrdev(92,“serial”,&serial_fops)。

步骤 2　将 serial.c 复制到 ../drivers/char 目录下，并在../drivers/char 目录下 mem.c 的 intchr_dev_init()函数中增加如下代码：

```
#ifdef  CONFIG_TESTDRIVE
init_serial();          #endif
```

步骤 3　在 ../drivers/char 目录下 Makefile 中增加如下代码：

```
ifeq($(CONFIG_TESTDRIVE),y)
L_OBJS+=serial.o
Endif
```

步骤 4　在 ../arch/m68knommu 目录下 config.in 中字符设备段里增加如下代码：

```
Bool‘support for testdrive’CONFIG_TESTDRIVE y
```

步骤 5　运行 make menuconfig (在 menuconfig 的字符设备选项里可以看到刚刚添加的“support for testdrive”选项，并且已经被选中。

步骤 6　文件系统的相应改变。在 ../dev/目录下依次创建串口设备：mknod/dev/serial1 C 92 0，C 表示字符设备，92 是主设备号，最后一位是次设备号。

对文件系统的目录进行添加即可。至此，已经在 Linux 中增加了一个新的设备驱动程序。通过命令 lsmod 可以查看新的设备驱动程序是否已经加载到内核中了。接下来进行 Linux 内核编译，一般 Linux 内核全部编译需要下述命令：

make xconfig(menuconfig)(dep)(clean)(bzImage)(modules)(modules_install)

将编译好的内核重新下载到目标板上就可使用串口的驱动程序了。

4.2 驱动程序的调试方法

在驱动程序的开发过程中,调用内核打印错误函数 printk( ),将调试信息打印到超级终端上进行简单的调试。通过将 2 路 RS-422 或者 2 路 RS-485 对通,编写测试代码,配合使用示波器检测串口设备收发数据的正确性。CAN 驱动程序也使用同样的调试方法。

5 比较 2 种方法开发驱动程序的特点

使用模块可以使得内核更加紧凑和灵活,修改内核时也不必全部重新编译内核,这样也避免了修改内核时重启机器模块;链接到内核,它的作用和静态链接的内核目标代码完全等价,这样在调用模块函数时,就可不显示传递消息。由于嵌入式 Linux 不像桌面 Linux 那样可以灵活地使用 insmod/rmmod 加载、卸载设备驱动程序,使用静态编译进内核的方法有利于内核运行的稳定性,不容易产生系统恐慌。

同时,模块的引入也带来了一些问题,由于内核所占的内存空间是不会被换出的,链接进内核的模块会给整个系统带来一定的性能和内存方面的损失,而且装入内核的模块会成为内核的一部分,可以修改内核中的其他部分。模块的使用不当可能导致系统崩溃,而且模块的版本维护,以及内核符号表的更新,模块之间的依赖性都是编写模块程序时要时时关注的问题。静态编译进内核的方法调试不是很方便,每次都需要重新编译内核。这 2 种方法各有利弊,可以根据具体情况选择使用。

6 结语

随着微控制器和网络技术的发展,工业系统中使用 ARM 和 Linux 相结合的嵌入式技术必将越来越广泛。本文介绍的基于ARM 920 T 嵌入式通信控制系统在宜昌、南京等电力系统得到了很好的应用。利用 Internet 和嵌入式技术实现工业控制领域的网络连接和远程监控具有很好的市场前景。

参考文献:

[1] 赵炯. Linux 内核完全注释[M]. 北京:机械工业出版社,2004.

[2] 李善平,刘文峰,李程远,等. Linux 内核 2.4 版源代码分析大全[M]. 北京:机械工业出版社,2002.

[3] 邬宽明. CAN 总线原理和应用系统设计[M]. 北京:北京航空航天大学出版社,2003.

[4] 秦贵和,徐华中. ARM9 嵌入式技术及 Linux 高级实践教程[M]. 北京:北京航空航天大学出版社,2005.

[5] LOVE R. Linux 内核设计与实现[M]. 陈莉君,康华,张波,译. 北京:机械工业出版社,2004.

[6] MOLAY B. Unix/Linux 编程实践教程[M]. 扬宗源,黄海涛,译. 北京:清华大学出版社,2004.

[7] WALL K. GNU/Linux 编程指南[M]. 张辉,译. 北京:清华大学出版社,2005.

[8] YAGHMOUR K. 构建嵌入式 Linux 系统[M]. 韩存冰,改编. 北京:中国电力出版社,2004.

[9] 陈坚,孙志月. Modem 通信编程技术[M]. 西安:西安电子科技大学出版社,2003.

[10] 习博,方彦军. 嵌入式监测系统中网络通信的研究与实现[J]. 电力自动化设备,2004,24(7):68-71.
XI Bo,FANG Yan-jun. Research and implementation of network communication in embedded monitoring system[J]. Electric Power Automation Equipment,2004,24(7):68-71.

[11] 严亚勤,吴文传,张伯明. 基于嵌入式 Linux 的网络 RTU[J]. 电力自动化设备,2004,24(9):27-29.
YAN Ya-qin,WU Wen-chuan,ZHANG Bo-ming. A network RTU based on embedded Linux[J]. Electric Power Automation Equipment,2004,24(9):27-29.

[12] 严晓蓉. 电力自动化系统中的数据处理[J]. 电力自动化设备,2005,25(3):96-99.
YAN Xiao-rong. Data processing in electric power automation system[J]. Electric Power Automation Equipment,2005,25(3):96-99.

(**责任编辑**:汪仪珍)

作者简介:

王松月(1981-),男,江苏新沂人,硕士研究生,研究方向为嵌入式系统与计算机控制(**E-mail**:Wangsy_bupt@sina.com);

杨福兴(1964-),男,江西高安人,教授,研究方向为嵌入式系统与计算机控制、物流信息系统。

Driver development for embedded communication control system based on ARM 920 T

WANG Song-yue,YANG Fu-xing

(Beijing University of Posts and Telecommunications,Beijing 100876,China)

Abstract: The traditional 8/16 bit MCU(Micro-Control Unit) can not meet requirements of mass information management because of its slow speed,heavy consumption and difficulty to implement Internet protocol. An embedded communication control system based on ARM(Advanced RISC Machines) 920 T and Linux is proposed. The system architecture and its applications in industrial systems are introduced,and the way to develop drivers for peripherals with CAN(Controller Area Network) bus and RS-422/RS-485 bus using the dynamic module loading method and the static compiling to kernel method is discussed. The debugging of the driver is analyzed. Two methods of developing the embedded system driver program are contrasted.

Key words: embedded system; ARM; Linux; driver