嵌入式 Linux 下步进电机驱动开发

汤冬云,张 毅
(重庆大学 软件学院,重庆 400044)

摘 要:为了实现在嵌入式系统中对步进电机进行控制且尽可能少占系统资源,现基于嵌入式处理器 S3C2410 和 Linux 系统进行步进电机驱动开发,包括相关硬件体系结构和软件系统组成及测试固化方法。实现了动态编译下步进电机驱动的开发及电机控制。阐述了在 Linux 系统下驱动程序的设计流程和内核模块化编程思想,驱动程序模块化并在需要的时候加载到内核是节省嵌入式硬件资源的主要方法之一。为一般嵌入式系统下编写驱动程序步骤提供参考。

关键词:步进电机;驱动程序;内核模块;嵌入式 Linux
中图分类号:TP319 **文献标识码**:A **文章编号**:1673-629X(2012)06-0167-04

Stepping Motor Driving Development under Embedded Linux System

TANG Dong-yun, ZHANG Yi
(College of Software, Chongqing University, Chongqing 400044, China)

Abstract: In order to achieve the stepper motor control in the embedded systems and account systems as less resources, it has implemented the stepping motor drive under the embedded processor S3C2410 and the Linux system. That includes the associated hardware architecture and software systems and test curing methods. It has achieved the step motor-driven development and motor control under dynamic compilation. It elaborates the driver program design processes in Linux system and kernel in modular program. Designed driver module and loaded into the kernel while needing is one of the main methods of saving embedded hardware resources. And it provides a reference for the general steps in writing a driver in embedded system.

Key words: stepping motor; driver program; kernel modular; embedded Linux

0 引 言

伴随着 Linux 嵌入式技术不断的发展进步,在嵌入式系统下编写固定的设备驱动越来越受到人们的重视。在当今各种嵌入式操作系统中,Linux 开源、根据需要可以任意裁剪的特性,更是吸引了人们的眼球。Linux 系统将存储器和外设分为 3 个基础大类:字符设备、块设备、网络设备。字符设备是指那些不经过系统的快速缓冲,以串行顺序依次进行访问的设备[1]。块设备要经过系统的快速缓冲,以数据块为单位的数据读写。网络设备是面向数据包的接受和发送。步进电机具有控制性能好、定位精确、抗干扰能力强等特点被广泛用于各种自动控制系统中。作为执行元件,是机电一体化的关键产品。文中实现了 S3C2410 嵌入式 Linux 下步进电机驱动程序。

收稿日期:2011-10-10;**修回日期**:2012-01-13

作者简介:汤冬云(1984-),男,湖北宜昌人,硕士研究生,研究方向为嵌入式软件开发;张 毅,副教授,博士,研究方向为软件工程、企业应用集成、办公自动化、嵌入式软件开发、企业信息化。

1 硬件平台

1.1 控制系统的硬件组成

本系统采用基于 ARM920T 内核的 S3C2410V6 微处理器作为系统的中央控制器,该芯片主频 200MHz,最高可达 266MHz,存储器分为 64M 的 SDRAM(其时钟频率高达 100MHz)、64M Nand Flash 和 1M Nor Flash,外部接口包括用于通信和下载的串口、USB 接口、JTAG 接口和 GPIO 接口。ULN2003 是集成达林顿管 IC,内部还集成了一个消线圈反电动势的二极管,由双列 16 脚封装,最大驱动电压为 50V,电流 500mA。具有电流增益高、工作电压高、温度范围广、带负载能力强等特点,适用于高速大功率驱动系统。

1.2 步进电机参数指标

步进电机是将电脉冲信号转变为角位移或线位移的开环控制单元组件,在非超载的情况下,电机的转速、停止位置只跟脉冲信号的频率和脉冲数有关,不受负载的变化,当步进驱动器接收到一个脉冲信号,就按预设的方向转动一个步距角。由于它的转动是按固定角度一步一步的运行,所以可以控制脉冲的数目来控

制角位移量,从而达到准确的定位[2]。本系统采用四相八拍步进电机 28BYJ48,工作电压为 DC5V ~ DC12V。

28BYJ48 型号步进电机静态指标:相数,产生不同对级 N、S 磁场的激磁线圈对数,常用 m 表示($m=4$)。拍数,完成一个磁场周期性变化所需要脉冲数或导电状态,用 n 表示($n=8$),或指电机转过一个齿距角所需要的脉冲数,本电机运行方式为 A-AB-B-BC-C-CD-D-DA-A。步距角,对应一个脉冲信号,电机转子转过的角位移用 θ 表示,θ = 360 度/(转子齿数 * 运行拍数)。定位转矩,电动机在不通电状态下,电动机转子自身的锁定力矩。静力矩,电动机在额定静态电作用下,不作旋转运动时,电动机转轴的锁定力矩。

1.3 步进电机的控制系统及接口电路

基于 ARM 的步进电机典型控制系统框图如图 1 所示。系统控制经过 ARM 单片机、脉冲信号、信号分配、功率放大、步进电机、负载 6 个过程。ARM 单片机产生脉冲信号,本系统由 S3C2410 产生脉冲信号,其占空比约为 0.3 ~ 0.4,占空比越大,电机转速越高。信号分配,不同的步进电机工作方式不同,28BYJ48 工作方式为四相八拍,即 A-AB-B-BC-C-CD-D-DA-A。功率放大,驱动系统中最重要的部分是功率放大模块,由于从 ARM 的 GPIO 口输出的脉冲电流都很小,需要加一个功率放大器才能驱动步进电机。步进电机在一定的转速下的转矩取决于它的动态平均电流而非静态电流,步进电机的平均动态电流越大其转矩越大,要使得平均电流达到最大,需要驱动系统尽量克服电动机的反电动势[3]。

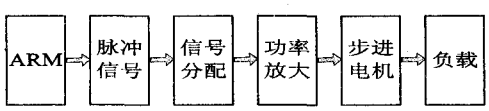

图 1 步进电机控制系统图

由于本系统采用的四相八拍工作模式,系统采用四路 I/O 进行脉冲分配,这四路 I/O 口是由处理器的 GPIO 扩展出来的,通过 ULN2003 功率放大后,进入步进电机的各相绕组。在四相八拍的工作模式下,脉冲的分配信号如表 1 所列。

表 1 脉冲分配信号

序号	正转脉冲序列	反转脉冲序列	序号	正转脉冲序列	反转脉冲序列
1	0001(A)	1001(DA)	5	0100(C)	0110(BC)
2	0011(AB)	1000(D)	6	1100(CD)	0010(B)
3	0010(B)	1100(CD)	7	1000(D)	0011(AB)
4	0110(BC)	0100(C)	8	1001(DA)	0001(A)

系统控制电路如图 2 所示。控制步进电机的 I/O 口地址为:

```
# define StepMtr _ Add * ( volatile U16 * ) (0x38001000)
```

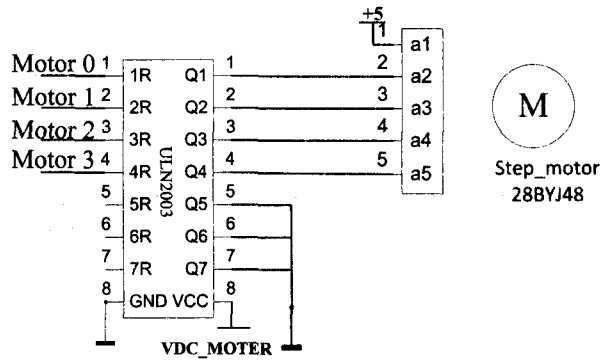

图 2 步进电机控制电路

2 驱动程序设计

设备驱动是操作系统的重要组成部分,对于特定的硬件设备,对应的设备驱动程序是不同的。设备驱动程序的任务包括自动配置、初始化子程序、检测需要驱动的设备是否存在以及是否能正常工作。如果一切正常,则对该设备及相关设备驱动程序运行需要的环境进行初始化[4]。设备驱动程序的流程图如图 3 所示。

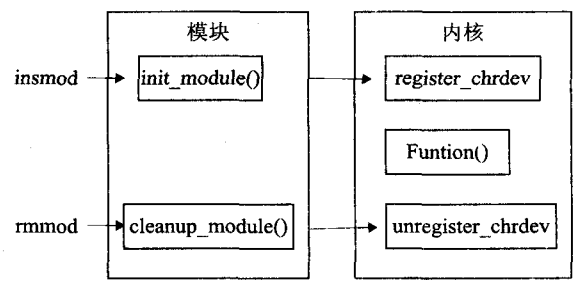

图 3 设备驱动程序的流程图

步进电机是将脉冲信号转变为角位移或线位移的控制元件,电机的状态只取决于脉冲信号的频率和脉冲数,即对电机发送一个脉冲信号,电机就转过一定的步距角[5]。这样,通过控制发送脉冲序列的个数,就可以达到控制角位移量的目的。发送脉冲频率不同,电机的转速和加速度就不一样,通过控制发送脉冲的频率来控制电机的转速,达到调速的目的。发送脉冲信号的顺序不同,相应的电机转向也不一样,按照正转脉冲序列发送脉冲信号,电机就正向旋转。反之,发送反转脉冲序列,电机就逆向旋转,以此来控制电机的旋转方向。

步进电机在系统中认为是只写的字符设备,它只响应应用程序传送给驱动的步进量、正反转等其它参数。

2.1 数据结构的定义

设备驱动运行在内核空间,而应用程序运行在用

户空间,内核空间为用户提供一个与计算机硬件相一致的窗口,即用户进程通过设备文件跟硬件打交道[6]。驱动程序通过设备文件进行操作时就可以对硬件进行直接操作了,而对设备文件操作需要设备驱动程序提供的入口点由数据结构向系统说明。file_operations 数据结构提供文件系统的入口点函数,也就是访问设备驱动的函数[7]。该数据结构定义在<linux/fs. h>中的函数指针表。file_operations 数据结构的每一个成员的名字对应着一个系统调用,用户就是利用系统调用对设备文件进行操作。系统调用通过对设备文件的主设备号找到相应的设备驱动程序,然后读取该结构中相应的指针,把控制权交给该函数[8]。步进电机定义的 file_operations 数据结构为:

```
Static struct file_operations Step_Mtr_fops = {
ioctl:step_mtr_ioctl,
open: step_mtr_open,
release: step_mtr_release,
};
```

从以上定义的数据结构可以看出,实现了步进电机的控制、打开和释放函数。步进电机为只写设备,只接受发出的脉冲命令,必要时候需要改变步进电机的状态,所以需要 ioctl 函数,ioctl 函数实现了步进电机脉冲信号分配、电机的正反转动、启动和停止。

2.2　设备的初始化

驱动模块都有一个入口函数,该函数为 static Step_Mtr_module_init(Step_Mtr_init)载入驱动模块后,程序跳转到 Step_Mtr_init(void)函数,来对设备进行初始化。在这个函数内,主要调用 register_chrdev()来完成字符设备在系统中的注册并建立与文件之间的关联[9]。主要向内核注册了设备的主设备号、设备名称和传递 fops 这个指针变量,在本系统中是通过下面调用来完成注册:

```
register_chrdev(STEPMTR_MAJOR,"Step_mtr",&Step_Mtr_fops);
```

其中 STEPMTR_MAJOR 为定义的主设备号,Step_mtr 为设备名称,Step_Mtr_fops 为上文定义的文件指针变量。在初始化函数里面还要实现步进电机的初始化,申请控制步进电机的端口,用下面的函数实现:

```
apply_region(StepMtr_Add,1,"Step_mtr");
```

因为步进电机需要使用系统的 I/O 端口,而在 Linux 系统中操作端口用的是虚拟地址而非实际物理地址,所以要修改内核代码,使得调用步进电机的控制程序时候,使用的虚拟地址能够映射到步进电机实际接口所接的端口地址[10]。打开文件 linux/arch/arm/mach-s3c2410/smdk. c,在结构体 static struct map_desc 中的 smdk_io_desc[ ]数组中添加一行元素{0x38001000,0x38001000,DOMAIN_IO,0,1,0,0},这样在向虚拟地址 StepMtr_Add 操作时候就能对实际的相应的物理地址进行操作。

2.3　设备的打开和释放

当用户调用驱动程序时候,会调用 step_mtr_open()函数,在这个函数中要设置对应的端口配置器 GPFCON 和端口上拉寄存器 GPFUP,使用了 Port F 的前 4 个端口,相应的端口应该定义为输出,根据端口 F 配置寄存器的定义,GPFCON = 0x00000155,这里要禁止端口的上拉寄存器,所以 GPFUP = 0x0000000F。同时还需要判断当前驱动程序是否被打开,定义 IOpen 来记忆驱动程序的状态,如果其值大于 1,认为已经有其它的应用程序打开了这个驱动程序,这时候返回 Busy,表明该程序处于忙碌状态。当调用 step_mtr_release()函数时,关闭驱动程序的调用,将 IOpen 清零。

2.4　设备的操作

设备的具体操作是调用 step_mtr_ioctl (struc inode * inode, struct file * flip, unsigned int cmd, unsigned long arg)函数。上文定义了脉冲分配信号,根据表 1,定义正转时对应的结构体为:

```
unsigned char co_rotating[ ] = {0x01,0x03,0x02,
0x06,0x04,0x0c,0x08,0x09};
```

同理,定义反转时对应的结构体为:

```
unsigned char re_rorating[ ] = {0x09,0x08,0x0c,
0x04,0x06,0x02,0x03,0x01};
```

通过 cmd 的值来判断是正转还是反转,如果 cmd = 1,则代表正转,这时系统调用电机驱动函数 stepmtr_driver(),并将正转对应的结构体 co_rotating[ ]的地址传给该函数,使得系统的 I/O 端口能按定义的脉冲序号发出脉冲,驱动电机正转;同理,当 cmd = 0 时定义电机反转,将 re_rorating[ ]的地址传给驱动函数,驱动电机反转。

2.5　设备驱动程序的加载

在 Linux 系统中,驱动程序主要有两种方式进行编译,一种是静态编译,另外一种就是动态编译。静态编译是指直接将驱动程序编译到系统中,可以随时的对它进行调用,不需要安装。这种做法的优点是调用驱动程序的时候方便,但是这种方法会带来很多弊端。首先,增加了内核空间,本来嵌入式硬件资源就很有限,对不经常使用的驱动程序静态编译到内核中就浪费了一定的内存空间。其次,如果编译好的内核下载到目标板上重新启动有错误,需要重复的进行系统编译,比较费时费力[11]。动态编译很好地解决了静态编译的不足,动态编译在调用驱动程序的时候会因为要寻找驱动程序模块而增加系统资源的占用和运行时间,但是与系统内核消耗的资源来比显得微不足道。

动态编译最大的方便在于用户对某一硬件的驱动程序开发和调试的时候，不用反复重新启动系统就可以动态的卸载旧的版本，加载新的版本。这种机制被称为模块，模块本身不被编译入内核印象，从而控制了内核大小，而且模块一旦被加入，它就和内核中其它部分完全一样。

把写好的步进电机驱动程序编译成模块，然后调用 insmod 把模块加载到内核，加载的时候系统会自动检测此模块是否已经被加载，如果已经被加载，内核会自动调用模块的初始化函数。模块卸载是用 rmmod 函数来完成的，在调用时会先检测此模块是否能被卸载，如果能被卸载则系统调用该函数清除函数。

驱动程序采用的模块化编程，当需要改驱动程序时，通过加载模块，将该驱动添加到系统内核[12]。当然，不需要该驱动程序的时候，就可以将该驱动程序卸载下来，这样保证了 Linux 系统内核的大小。在嵌入式开发过程中，系统资源有限，模块化编程的灵活方式节约了资源。

3 程序测试

将编写好的驱动程序在 PC 机上进行交叉编译（本系统采用的交叉编译版本为 arm-linux-gcc-3.4.5-glibc-2.3.6），生成 Step_Mtr.o 文件，然后将该文件通过 FTP 或其它方式传输到目标机上。在加载前需要修改文件的读写权限，其命令为：

#chmod 777 Step_Mtr.o

然后加载该模块，其命令为：

#insmod Step_Mtr.o

在加载完成后，可以用下面的命令查看是否有加载的驱动程序，其命令为：

#lsmod

接着建立设备节点，命令为：

#mknod /dev/ Step_Mtr c STEPMTR_MAJOR 0

其中/dev/ Step_Mtr 为步进电机设备驱动的设备名，c 表示本设备为字符设备，STEPMTR_MAJOR 为在上文定义的主设备号，0 为从设备号。

这时候驱动程序一切就绪，编写相应的驱动测试程序就能查看电机的状态。

上面的过程主要用来调试，在掉电之后不能保存，所以需要将程序固化，需要将该程序和模块添加到根文件系统。

将驱动程序和测试应用程序添加到 cramfs 根文件系统 Step 中，然后将 mkcramfs 文件复制到 Step 所在的目录，并在该目录下运行命令：

#mkcramfs Step step_driver.cramfs

该命令运行成功后，会在该目录下生成 step_driver.cramfs 根文件系统，然后将其烧写在系统中，目标板掉电后重新启动驱动程序能继续使用。

4 结束语

本系统主要采用了 S3C2410 处理器的 ARM9 单片机，结合步进电机控制器 ULN2003，对步进电机的控制。在 Linux 系统下，设备驱动是内核与硬件之间的接口，对硬件的操作都是通过驱动程序来实现的。文中介绍了嵌入式 Linux 系统下驱动程序的原理并结合步进电机的驱动程序，给出了在 Linux 系统下编写驱动程序的一般方法。

参考文献：

[1] 王黎明. ARM9 嵌入式系统开发与实践[M]. 北京：北京航天航空大学出版社，2008.

[2] Weerakoon T S, Samaranayake L. Development of a novel drive topology for a five phase stepper motor[C]//Proceedings of 11th international conference on electrical machines and systems. [s. l.]:[s. n.],2008.

[3] 张付详，刘振宇. 多通道步进电机控制器设计及 Linux 驱动实现[J]. 制造业自动化，2011，33(2)：47-50.

[4] 白复东. 嵌入式 Linux 驱动程序开发[J]. 信息技术，2009(9)：185-189.

[5] Mizutani K, Hayashi S, Matsui N. Modeling and Control of Hybrid Stepping Motors[C]//Proc. of IEEE/IAS Annual Meeting. [s. l.]:[s. n.],1993.

[6] Rodriguez C S, Fischer G, Smolski S. Linux 内核编程[M]. 陈莉君，译. 北京：机械工业出版社，2006.

[7] 宋宝华. LINUX 设备驱动开发详解[M]. 北京：人民邮电出版社，2008.

[8] 孙泽田. 嵌入式设计及 linux 驱动开发指南[M]. 北京：电子工业出版社，2005.

[9] 张国栋. 基于嵌入式单片机的步进电机控制系统[J]. 电脑知识与技术，2011，7(24)：6017-6019.

[10] 周晓光. 基于 S3C2440A 的嵌入式视频系统设计[J]. 电子测量技术，2006，29(6)：84-86.

[11] 钟伟弘，关保国，张善青. 步进电机的驱动及微机控制[J]. 天津理工学院学报，2000，16(2)：66-68.

[12] 刘晓明，程铁汉，邵 敏. 基于 ARM7 的便携式工业打印机[J]. 计算机技术与发展，2008，18(7)：187-189.