ZLG7290 扩展键盘在 ARM Linux 系统中的应用

李直霖
(中煤科工集团重庆研究院,重庆 400037)

摘要:针对人机界面按键需求多、资源要求少,模块化,方便移植等问题,提出了将一款支持 I^2C 总线数据格式、提供键盘中断信号的键盘扫描管理芯片 ZLG7290 应用于 ARM Linux 系统中。按键通过 I^2C 总线主机通信的方式,克服了传统方法的不足,节约了硬件资源。研究结果表明,该方案便捷、灵活地解决了上述矛盾,稳定、可靠,具有实际应用推广价值。

关键词:嵌入式;驱动;Linux;ARM;I^2C;ZLG7290
中图分类号:TP211$^+$.5 **文献标志码**:A **文章编号**:1001 - 4551(2011)10 - 1253 - 04

Application of ZLG7290 extended keyboard in ARM Linux system

LI Zhi-lin
(Chongqing Institute of Coal Science Research Institute, Chongqing 400037, China)

Abstract: Aiming at the man-machine interface for the key demand, resource requirements of small, modular, easy to transplant and other issues, a support for I^2C-bus data format was taken to provide keyboard interrupt management chip ZLG7290 keyboard scanning used in ARM Linux system. The communication with host computer was realized by I^2C bus, the lack of traditional method was overcomed, the hardware resource was saved. The research results indicate that the program convenient and flexible solves the above contradiction, it is stable, reliable, and it has practical application value.

Key words: embedded; driver; Linux; ARM; I^2C; ZLG7290

0 引 言

目前实际的嵌入式应用中,很多都对输入/输出提出了要求。输入的方式多用键盘,广泛采用的键盘方式还是用 GPIO 口处理[1]。如果按键较少,一般采用一个 I/O 对应一个按键的方式,这样电路和程序简单、可靠;如果按键较多,一般采用矩阵扫描方式,这样用较少的硬件资源来实现较多的逻辑功能;但是如果硬件资源非常有限,而按键需求又比较多的情况下(如 30 个按键以上),以上两种方法明显满足不了要求。

本研究提出了一种采用 I^2C 总线的 ZLG7290 方式,硬件平台采用 S3C2440,操作系统采用 Linux2.6.29 核,按键通过 I^2C 总线与主机通信的方式,克服传统两种方法的不足,节约了大量的硬件资源。

1 硬件电路

广州周立功单片机发展有限公司的 ZLG7290 芯片[2]支持 I^2C 总线数据格式,提供 I^2C 串行接口和键盘中断信号,方便与处理器接口,可驱动 64 只独立 LED 和 64 个按键。

33 个按键的键盘模块电路图如图 1 所示。

ZLG7290 的 I^2C 接口传输速率可达 32 Kbit/s,S3C2440 的 I^2C 总线串行时钟信号 2440_SCL 与 ZLG7290 的 SCL 相连,数据信号 2440_SDA 与 ZLG7290 的 SDA 相连,S3C2440 的外部中断 9 与 ZLG7290 的 INT 脚相连。该芯片的从地址是确定的,为 70H。

方案应用到其他硬件平台方法也是类似的,可方便移植。

收稿日期:2011 - 04 - 22
作者简介:李直霖(1977 -),男,四川仁寿人,主要从事嵌入式系统及微机保护装置等方面的研究. E-mail:linuxarm@yeah.net

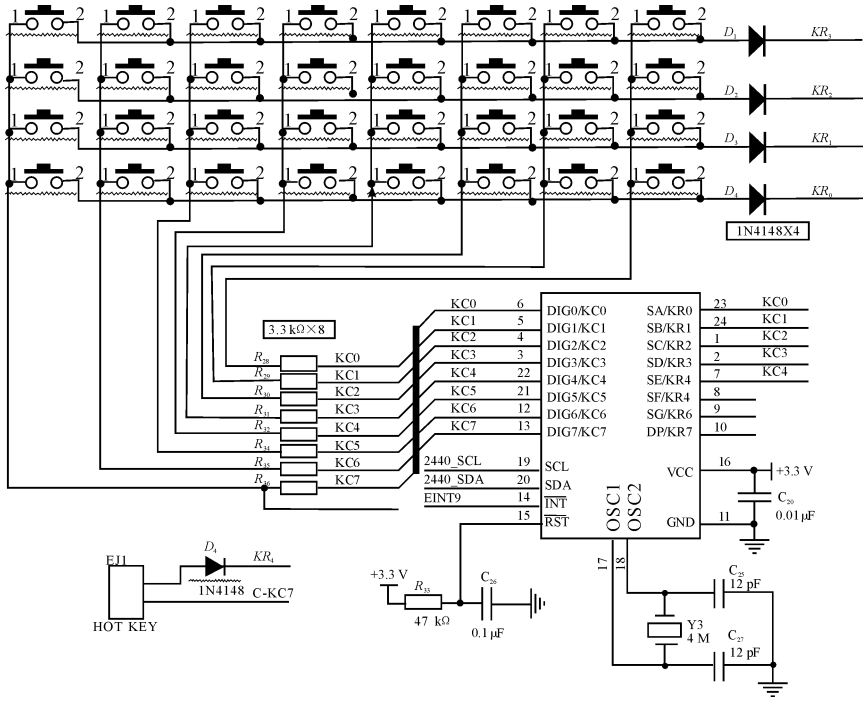

图 1 键盘模块电路图

2 设备驱动程序

设备驱动程序是操作系统和机器硬件之间的接口。在本应用中,虽说在功能上是键盘,但是从另一个角度看,其实是一个 I^2C 总线的设备,所以可利用 Linux I^2C 驱动体系结构来完成。

Linux 中 I^2C 的驱动分为两个部分:总线驱动(BUS)和设备驱动(DEVICE)。其中总线驱动的职责是为系统中每个 I^2C 总线增加相应的读写方法。但是总线驱动本身并不会进行任何的通讯,它只是存在于那里,等待设备驱动调用其函数。设备驱动则是与挂在 I^2C 总线上的具体的设备通讯的驱动。通过 I^2C 总线驱动提供的函数,设备驱动可以忽略不同总线控制器的差异,不考虑其实现细节地与硬件设备通讯。

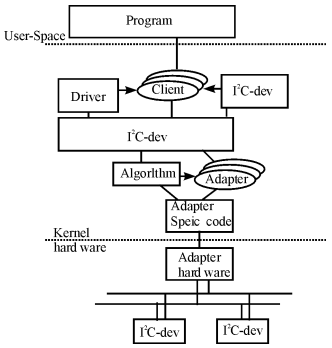

图 2 I^2C 驱动架构图

S3C2440 内嵌 I^2C 总线控制器,如图 2 所示,每一条 I^2C 对应一个 Adapter(适配器)。适配器驱动的作用是让用户能够通过它发出符合 I^2C 标准协议的时序。在 Linux 内核源代码中的 drivers/i2c/busses 目录下包含着一些适配器的驱动。如 S3C2440 的驱动 i2c-s3c2410. c。再通过 i2c core 层将 i2c 设备与 i2c adapter 关联起来。

本研究利用 Linux I^2C 驱动体系结构来完成,驱动源程序包括 i2c-dev. c, i2c-core. c, i2c-algo-xxx. c, i2c-s3c2410. c。以上 4 个模块依次是下层与上层的关系。要使用 ZLG7290 设备,用户要做的是在 i2c-dev. c 的基础上编写总线字符型驱动源程序 i2c-dev7290. c,其他 3 个源程序无需更改。

Linux 内核[3]内部通过 file 结构识别设备,通过 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。在 i2c-dev7290. c 驱动程序[4]中 i2cdev_fops 定义如下:

```
static const struct file_operations i2cdev_fops = {
    . owner          = THIS_MODULE,
    //指向拥有该结构的模块,内核使用该结构维护模块使用计数
    . llseek         = no_llseek,
    . read           = i2cdev_read,//读接口函数
    . write          = i2cdev_write,//写接口函数
    . unlocked_ioctl  = i2cdev_ioctl,
    . open           = i2cdev_open,/打开设备函数
    . release        = i2cdev_release,//释放设备函数
    . fasync         = globalfifo_fasync,//异步通知函数
```

```
};
```

以下是该设备驱动的关键修改部分:

异步通知函数向进程发送 SIGIO 信号,通知访问设备的进程,表示设备已经准备好 I/O 读写了,避免主动查询,提高程序效率。使用异步通知需增加一个 struct fasync_struct 的结构指针,然后实现 fasync 接口函数:

```
static int globalfifo_fasync(int fd, struct file *filp, int mode)
{
    struct i2c_client *dev = filp->private_data;
    return fasync_helper(fd, filp, mode, &async_queue);
    }
```

i2cdev_release 的主要任务是清理未结束的输入输出操作,释放资源,用户自定义排他标志的复位等:

```
static int i2cdev_release(struct inode *inode, struct file *file)
{
    …
    globalfifo_fasync( -1, file, 0);
    return 0;
}
```

i2c_irq_handle 中断处理函数的实现:

```
static void i2c_irq_handle(int irq, void *dev_id, struct pt_regs
*regs)
{
    kill_fasync(&async_queue, SIGIO, POLL_IN);//发送异步
通知信号
    }
```

i2c_dev_init 初始化函数完成驱动模块加载:

```
static int __init i2c_dev_init(void)
{
    …
    res = register_chrdev(I2C7290_MAJOR, "i2c7290",
&i2cdev_fops);
    //注册设备
    …
    if(request_irq(IRQ_EINT9, i2c_irq_handle, IRQF_DIS-
ABLED, "ZLG7290", NULL))
    //申请中断
    …
}
```

i2c_dev_exit 退出函数完成驱动模块卸载:

```
static void __exit i2c_dev_exit(void)
    {
    free_irq(IRQ_EINT9, NULL);
    i2c_del_driver(&i2cdev_driver);
    class_destroy(i2c_dev_class);
    unregister_chrdev(I2C7290_MAJOR,"i2c7290");
}
```

驱动程序完成,编译工作可由 Makefile 文件完成,最终生成 *.ko 的文件加载进内核,在 Linux 的/dev/目录下创建设备节点 i2c7290。

3　上层应用程序

关于 I^2C 操作的两个重要数据结构[5]如下:

```
    struct i2c_msg {
    __u16 addr;      /* slave address */
    __u16 flags;
//flags = = 0 means write operations
// flags = = 1 means read  operations
    __u16 len;
//msg length
    __u8 *buf;
// pointer to msg data
    };
    struct i2c_rdwr_ioctl_data {
    struct i2c_msg *msgs;
//pointers to i2c_msgs
    int nmsgs;
// number of i2c_msgs
    };
    struct i2c_rdwr_ioctl_data data;
    unsigned int fd, i,oflags;
    unsigned char i2cAddr = 1;
    unsigned char buft;
```

input_handle 中断处理程序的实现如下:

```
void input_handler(int signum)
{//对 i2c_msg 结构赋值
    (data.msgs[0]).addr = 0x70 > >1;   //从器件地址
    (data.msgs[0]).buf = &i2cAddr;
    (data.msgs[0]).flags = 0;  //表示写
    (data.msgs[0]).len = 1;  //长度为 1 个字节

    (data.msgs[1]).addr = 0x70 > >1;  //从器件地址
    (data.msgs[1]).buf = &buft;  //读到的数据放到 buft
    (data.msgs[1]).flags = 1;  //表示读
    (data.msgs[1]).len = 1;  //长度为 1 个字节
    ioctl(fd, I2C_RDWR, &data);
    …
    }
```

应用主程序实例如下[6]:

```
    int main(int argc, char * *argv)
    {
    //应用程序通过驱动来访问外部 I2C 器件,首先要通过
open()来打开设备;
```

```
fd = open("/dev/i2c7290/0", O_RDWR);
…
data.nmsgs = 2;
data.msgs = (struct i2c_msg *)malloc(data.nmsgs * sizeof(struct i2c_msg));

signal(SIGIO, input_handler);  //信号驱动(SIGIO)的异步 I/O 与 input_handler 建立联系
fcntl(fd, F_SETOWN, getpid());  // 应用可以接收驱动的 SIGIO 信号
oflags = fcntl(fd, F_GETFL);
fcntl(fd, F_SETFL, oflags | FASYNC);
while(1)
{
usleep(100);
}
…
}
```

应用程序完成后,编译生成可执行文件。运行执行文件,当按键发生时,异步通知,中断函数执行,将键值读入 buft 中[7-8]。

4 结束语

本研究从实际应用的角度全面的阐述了如何在 ARM Linux 系统中采用 I^2C 总线的方式来解决键盘扫描硬件资源的问题,比起传统的 I/O 口和矩阵扫描的方式,该方案具有占用较少的资源,获得较快的响应,同时移植方便,稳定可靠等特点。实际产品已稳定运行 3 年多,从未出现问题,具有广泛的应用与参考价值。

参考文献(References):

[1] 杜春雷. arm 体系结构与编程[M]. 北京:清华大学出版社,2003.

[2] 广州周立功单片机发展有限公司. ZLG7290 用户手册[EB/OL]. [2008 - 11 - 20]. http://www.embedtools.com/pro_kaifa/arm/easyarm2103/zlg7290b_app.pdf

[3] CORBET J, KROAH-HARTMAN G, RUBINI A. Linux Device Drivers[M]. 3rd ed. O'Reilly, 2005.

[4] 毛德操,胡希明. Linux 内核源代码情景分析[M]. 杭州:浙江大学出版社,2001.

[5] 孙 琼. 嵌入式 Linux 应用程序开发详解[M]. 北京:人民邮电出版社,2006.

[6] 孙桂鸿. 基于 ARM 的计算组成原理实验平台[J]. 机电技术,2010(5):35-37.

[7] 汤建斌,蒋 庆,蔡晋辉,等基于 ARM 微处理器的自动称重选别系统[J]. 轻工机械,2009,27(1):58-60.

[8] 戴钦来,马钧华. 基于 ARM 和 DSP 的多轴伺服系统以太网通信[J]. 轻工机械,2009,27(1):62-66.

[编辑:张 翔]

(上接第 1252 页)

价进行估计。它类似于标准的粒子滤波,两者具有相似的步骤:粒子传播、重采样、代价(权重)计算。而主要不同点在于前者使用统计假设,而后者依据较强的概率假设。标准粒子滤波利用重要性函数去构建后验概率分布以估计最优状态,而 CRPF 利用代价函数去构建自适应的高斯分布以估计最优状态。

在实际应用中,先验概率没有办法获得,而重要性函数的选择也很困难,没有统一的方法,使得标准粒子滤波的应用受到限制。而代价函数从 $\| y_t - f_y(x_t^{(i)}) \|^q$ 出发,不需要知道先验概率分布,实际的应用价值,同时准确性也能到提高。

代价参考粒子滤波,而是从最优理论的目标函数出发,结合数理统计的方法,解决了非线性轨迹逼近问题。

参考文献(References):

[1] MÍGUEZ J, BUGALLO M F, DJURIE P M. A new class of particle filters for random dynamic system with unknown statistics[J]. **EURASIP Journal on Applied Signal Processing**, 2004(15):2278-2294.

[2] DJURIE P M, BUGALLO M F. Cost-reference particle filtering for dynamic with nonlinear and conditionally linear states [C]. Proceeding of the Nonlinear Statistical Signal Processing Workshop: Cambridge (UK), 2006.

[3] MÍ GUEZ J. Analysis of selection methods for cost-reference particle filtering with application to maneuvering target tracking and dynamic optimization [C]// In proceeding of Digital Signal Processing, 2007:787-807.

[4] QI C, BONDON P. A new unscented particle filter[C]// IEEE International Conference on Acoustics, Speech and Signal Processing, 2008:3417-3420.

[5] YU zhi-jun, WEI Jian-ming, LIU Tao. Energy-efficient collaborative target tracking algorithm using cost-reference filtering in wireless sensor networks[J]. **The Journal of China Universities of Posts and Telecommunications**, 2009, 6(1):9-15.

[6] ZHANG Jian , WU Cheng-dong, JIA Zi-xi, et al. Target Tracking using Gaussian Cost reference Particle Filter in WSN Based on Multi-Modality Information[C]//Control and Decision Conference 2010:1388 - 1392.

[编辑:张 翔]