



浙江工业大学

# 实验报告

课程：嵌入式系统 A

第八次实验

姓 名 凌智城

学 号 201806061211

专业班级 通信工程 1803 班

老 师 黄国兴

学 院 信息工程学院

提交日期 2021 年 6 月 23 日

# 实验 7：I2C 驱动实验

## 1 实验目的

- 1) 熟悉 I2C 协议的原理；
- 2) 熟悉 Linux 下 I2C 的驱动构架；
- 3) 熟悉 Linux 驱动程序的编写；
- 4) 熟悉 Linux 下模块的加载等。

## 2 实验内容

- 1) 编写 I2C 协议的驱动程序；
- 2) 编写 Makefile；
- 3) 以模块方式加载驱动程序；
- 4) 编写 I2C 测试程序。

## 3 实验步骤

### 步骤 1：硬件连接

查看串口号，通过 putty 软件使用串口通信方式连接实验箱。



图 8-1 查看实验箱端口号

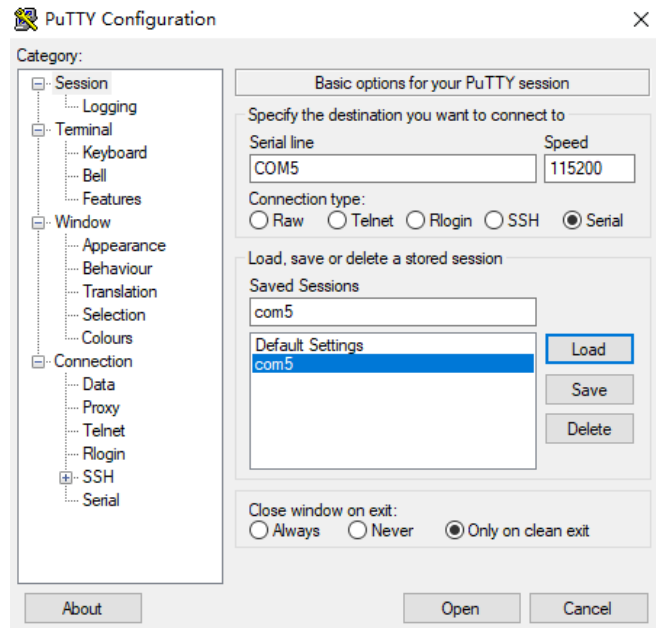


图 8-2 打开 putty 连接

输入启动参数启动内核

```
DM365 EVM :>setenv bootargs 'mem=110M console=ttyS0,115200n8 root=/dev/nfs rw nf
sroot=192.168.1.152:/home/shiyan/share/filesys_test ip=192.168.1.52:192.168.1.15
2:192.168.1.1:255.255.0::eth0:off eth=00:40:01:C1:56:02 video=davincifb:vid0
=OFF:vid1=OFF:osd0=640x480x16,600K:osd1=0x0x0,0K dm365_imp.oper_mode=0 davinci_
capture.device_type=1 davinci_enc_mgr.ch0_output=LCD'
DM365 EVM :>boot

Loading from NAND 1GiB 3,3V 8-bit, offset 0x800000
Image Name: Linux-2.6.18-plc_pro500-davinci_
Image Type: ARM Linux Kernel Image (uncompressed)
Data Size: 1995772 Bytes = 1.9 MB
Load Address: 80008000
Entry Point: 80008000
## Booting kernel from Legacy Image at 80700000 ...
Image Name: Linux-2.6.18-plc_pro500-davinci_
Image Type: ARM Linux Kernel Image (uncompressed)
Data Size: 1995772 Bytes = 1.9 MB
Load Address: 80008000
Entry Point: 80008000
Verifying Checksum ...
```

图 8-3 输入启动参数启动

```
Welcome to MontaVista(R) Linux(R) Professional Edition 5.0.0 (0801921).

login[743]: root login on 'console'
/*****Set QT environment*****/
[root@zjut ~]#
```

图 8-4 实验箱启动成功并且用 root 登录

## 步骤 2：编译 I2C 驱动

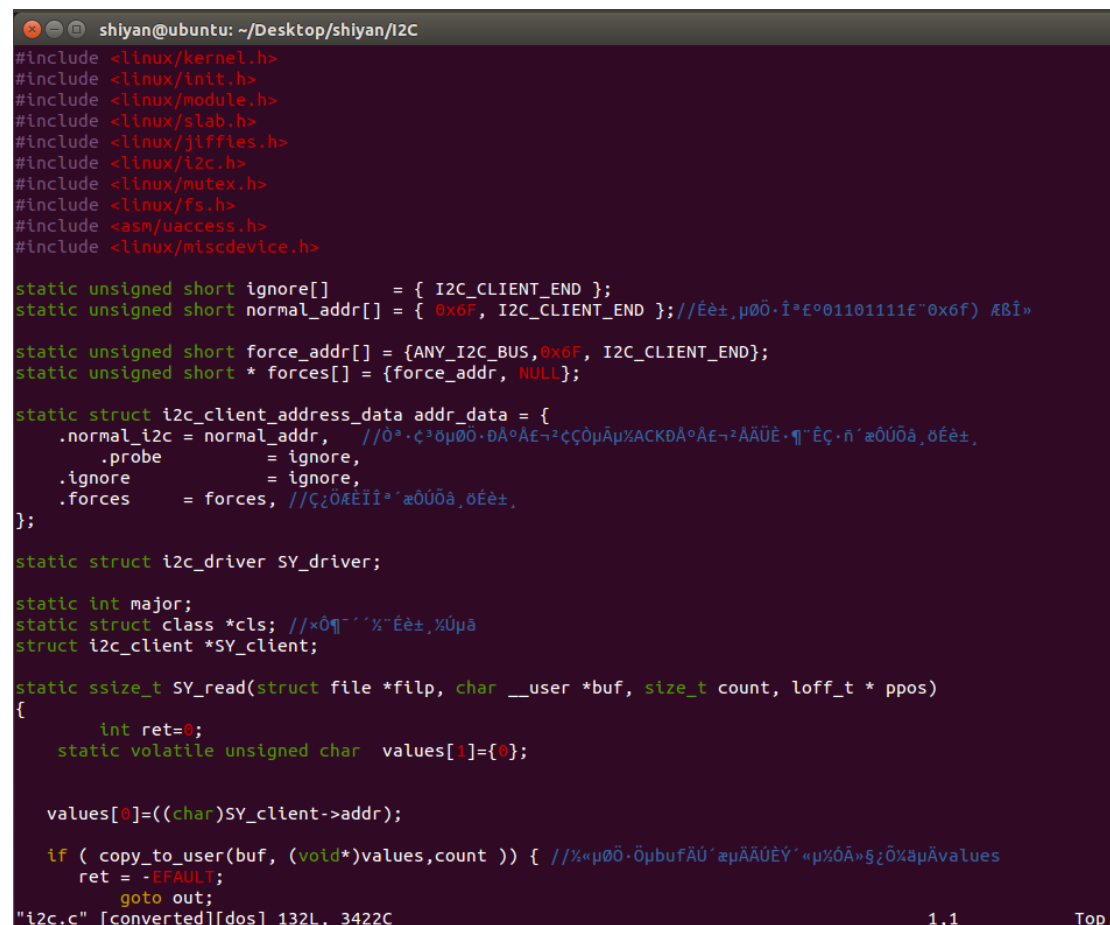
进入学生目录，`mkdir I2C`，创建 I2C 驱动文件夹，编写 I2C 驱动程序。

设备驱动源码参考文件夹 I2C 驱动实验/I2C/i2c.c。



Makefile 文件如下:

```
KDIR:=/home/shiyan/kernel-for-mceb
CROSS_COMPILE = arm_v5t_le-
CC = $(CROSS_COMPILE)gcc
.PHONY: modules clean
obj-m := i2c.o
modules:
    make -C $(KDIR) M=`pwd` modules
clean:
    make -C $(KDIR) M=`pwd` modules clean
```



```
shiyan@ubuntu: ~/Desktop/shiyan/I2C
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/module.h>
#include <linux/slab.h>
#include <linux/jiffies.h>
#include <linux/i2c.h>
#include <linux/mutex.h>
#include <linux/fs.h>
#include <asm/uaccess.h>
#include <linux/miscdevice.h>

static unsigned short ignore[] = { I2C_CLIENT_END };
static unsigned short normal_addr[] = { 0x6F, I2C_CLIENT_END };

static unsigned short force_addr[] = { ANY_I2C_BUS, 0x6F, I2C_CLIENT_END };
static unsigned short * forces[] = { force_addr, NULL };

static struct i2c_client_address_data addr_data = {
    .normal_i2c = normal_addr,
    .probe = ignore,
    .ignore = ignore,
    .forces = forces,
};

static struct i2c_driver SY_driver;

static int major;
static struct class *cls;
struct i2c_client *SY_client;

static ssize_t SY_read(struct file *filp, char __user *buf, size_t count, loff_t * ppos)
{
    int ret=0;
    static volatile unsigned char values[1]={0};

    values[0]=((char)SY_client->addr);

    if ( copy_to_user(buf, (void*)values,count )) {
        ret = -EFAULT;
        goto out;
    }

    return ret;
}
```

图 8-8 i2c.c

i2c 驱动如下:

```
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/module.h>
#include <linux/slab.h>
#include <linux/jiffies.h>
#include <linux/i2c.h>
#include <linux/mutex.h>
#include <linux/fs.h>
```

```

#include <asm/uaccess.h>
#include <linux/miscdevice.h>

static unsigned short ignore[] = { I2C_CLIENT_END };
static unsigned short normal_addr[] = { 0x6F, I2C_CLIENT_END };//设备地址为：
01101111 (0x6f) 七位

static unsigned short force_addr[] = {ANY_I2C_BUS,0x6F, I2C_CLIENT_END};
static unsigned short * forces[] = {force_addr, NULL};

static struct i2c_client_address_data addr_data = {
    .normal_i2c=normal_addr,    //要发出地址信号，并且得到 ACK 信号，才
    能确定是否存在这个设备
    .probe        = ignore,
    .ignore        = ignore,
    .forces        = forces, //强制认为存在这个设备
};

static struct i2c_driver SY_driver;

static int major;
static struct class *cls; //自动创建设备节点
struct i2c_client *SY_client;

static ssize_t SY_read(struct file *filp, char __user *buf, size_t count, loff_t * ppos)
{
    int ret=0;
    static volatile unsigned char  values[1]={0};

    values[0]=((char)SY_client->addr);

    if ( copy_to_user(buf, (void*)values,count) ) { //将地址值 buf 内存的内容传到
    用户空间的 values
        ret = -EFAULT;
        goto out;
    }
    out:
    return ret;
}

//定义字符设备结构体
static struct file_operations SY_fops = {

```

```

        .owner = THIS_MODULE,
        .read  = SY_read,
};

static int SY_detect(struct i2c_adapter *adapter, int address, int kind)
{
    printk("SY_detect\n");
    // 构建一个 i2c_client 结构体；收费数据主要靠它，里面有 .address .adapter .driver
    SY_client = kzalloc(sizeof(struct i2c_client), GFP_KERNEL);
    SY_client->addr      = address;
    SY_client->adapter = adapter;
    SY_client->driver  = &SY_driver;
    strcpy(SY_client->name, "SY");
    i2c_attach_client(SY_client); // 等要卸载驱动时，会调用 I2C_detach
    printk("SY_probe with name = %s, addr = 0x%x\n", SY_client->name, SY_client->addr);

    major = register_chrdev(0, "SY", &SY_fops); // 申请字符设备主设备号

    cls = class_create(THIS_MODULE, "SY"); // 创建一个类，然后在类下面创建一个设备
    class_device_create(cls, NULL, MKDEV(major, 0), NULL, "SY");

    return 0;
}

static int SY_attach(struct i2c_adapter *adapter)
{
    return i2c_probe(adapter, &addr_data, SY_detect);
}

static int SY_detach(struct i2c_client *client)
{
    printk("SY_detach\n");
    class_device_destroy(cls, MKDEV(major, 0));
    class_destroy(cls);
    unregister_chrdev(major, "SY");

    i2c_detach_client(client); // client 结构体
    kfree(i2c_get_clientdata(client)); // 释放 client 的内存
}

```

```

    return 0;
}

//定义 i2c_driver 结构体
static struct i2c_driver SY_driver = {
    .driver = {
        .name = "SY",
    },
    .attach_adapter = SY_attach,
    .detach_client = SY_detach,
};

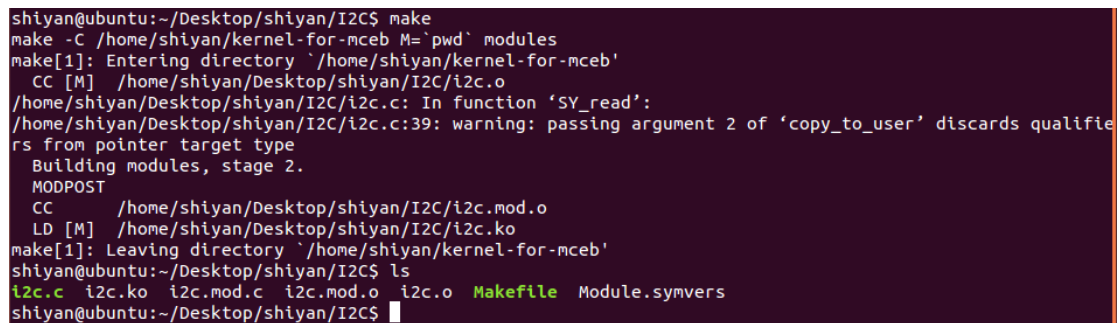
static int SY_init(void)
{
    printk("SY_init\n");
    i2c_add_driver(&SY_driver); //注册 i2c 驱动
    return 0;
}

static void SY_exit(void)
{
    printk("SY_exit\n");
    i2c_del_driver(&SY_driver);
}

module_init(SY_init);
module_exit(SY_exit);
MODULE_LICENSE("GPL");

```

执行 `make` 命令，生成 `i2c.ko` 等文件



```

shiyang@ubuntu:~/Desktop/shiyang/I2C$ make
make -C /home/shiyang/kernel-for-mceb M='pwd' modules
make[1]: Entering directory '/home/shiyang/kernel-for-mceb'
  CC [M] /home/shiyang/Desktop/shiyang/I2C/i2c.o
/home/shiyang/Desktop/shiyang/I2C/i2c.c: In function 'SY_read':
/home/shiyang/Desktop/shiyang/I2C/i2c.c:39: warning: passing argument 2 of 'copy_to_user' discards qualified
rs from pointer target type
  Building modules, stage 2.
  MODPOST
  CC      /home/shiyang/Desktop/shiyang/I2C/i2c.mod.o
  LD [M]  /home/shiyang/Desktop/shiyang/I2C/i2c.ko
make[1]: Leaving directory '/home/shiyang/kernel-for-mceb'
shiyang@ubuntu:~/Desktop/shiyang/I2C$ ls
i2c.c  i2c.ko  i2c.mod.c  i2c.mod.o  i2c.o  Makefile  Module.symvers
shiyang@ubuntu:~/Desktop/shiyang/I2C$

```

图 8-9 make 生成 `i2c.ko` 等文件

`i2c.ko` 驱动文件生成成功后，将其复制到挂载的文件系统 `modules` 的目录下。

虚拟机: `sudo cp i2c.ko /home/shiyang/share/filesys_test/modules`

服务器: `sudo cp i2c.ko /home/stX/filesys_test/modules`



```
[root@zjut ~]# df
Filesystem          1k-blocks      Used Available Use% Mounted on
tmpfs                53868         92     53776   0% /tmp
tmpfs                10240         32     10208   0% /dev
tmpfs                53868          0     53868   0% /dev/shm
192.168.1.152:/home/shiyan/share/filesys_test 43218944 10347520 30652416 25%
/mnt/mtd
[root@zjut ~]#
```

图 8-10 成功挂载文件系统

```
[root@zjut /]# ls
Settings      init          mnt           proc          sys           ver.txt
bin           lib           modules       root          tmp           wav
dev           linuxrc       nfs           sbin          usr
etc           lost+found    opt           shm           var

[root@zjut /]# cd mnt
[root@zjut mnt]# cd mtd
[root@zjut mtd]# ls
Settings      init          mnt           proc          sys           ver.txt
bin           lib           modules       root          tmp           wav
dev           linuxrc       nfs           sbin          usr
etc           lost+found    opt           shm           var

[root@zjut mtd]# cd modules
[root@zjut modules]# ls
at24cxx.ko      ov5640_i2c.ko      rtutil5370ap.ko
davinci_dm365_gpios.ko  rt5370ap.ko        rtutil5572sta.ko
egalax_i2c.ko    rt5370sta.ko        ts35xx-i2c.ko
fml188_i2c.ko    rt5572sta.ko        ttyxin.ko
i2c.ko           rtnet5370ap.ko
lcd.ko           rtnet5572sta.ko

[root@zjut modules]#
```

图 8-11 在实验箱中查看已经有了相应文件

### 步骤 3：加载驱动

- 1) 在驱动加载之前查看已经加载的驱动模块，使用命令 `lsmod`。

```
[root@zjut /]# lsmod
Module              Size  Used by    Tainted: P
dm365mmmap 5336 0 - Live 0xbflc1000
edmak 13192 2 - Live 0xbflbc000
irqk 8552 0 - Live 0xbflb8000
cmemk 28172 0 - Live 0xbflb0000
ov5640_i2c 9572 1 - Live 0xbflac000
rtnet5572sta 53620 0 - Live 0xbfl9d000
rt5572sta 1574024 1 rtnet5572sta, Live 0xbf01b000
rtutil5572sta 79988 2 rtnet5572sta,rt5572sta, Live 0xbf006000
egalax_i2c 16652 0 - Live 0xbf000000
[root@zjut /]#
```

图 8-12 lsmod 查看已加载的驱动模块

- 2) 执行 `cd /sys/bus/i2c`，进入目录查看当前加载的设备地址和设备名字。所有的 I2C 设备都在 `sysfs` 文件系统中显示。在当前目录文件夹下的 `devices` 下，

是当前挂载在 I2C 的设备地址。在 drivers 目录下是挂载 I2C 上的设备驱文件。

```
[root@zjut /]# ls
Settings      init          mnt           proc          sys           ver.txt
bin           lib           modules       root          tmp           wav
dev           linuxrc       nfs           sbin          usr
etc           lost+found    opt           shm           var

[root@zjut /]# cd sys
[root@zjut sys]# ls
block      class      firmware     kernel       vendor
bus        devices    fs           module

[root@zjut sys]# cd bus
[root@zjut bus]# ls
i2c        mmc        scsi         spi          usb-serial
mdio_bus   platform   serio        usb

[root@zjut bus]# cd i2c
[root@zjut i2c]# ls
devices  drivers
[root@zjut i2c]# ls devices/
0-0004  0-0018  0-003c
[root@zjut i2c]# ls drivers/
OV5640 channel0 Video Decoder I2C driver
aic3x I2C Codec
davinci_evm
dev_driver
egalax_i2c
i2c_adapter
[root@zjut i2c]#
```

图 8-13 查看当前加载的设备地址和设备名

- 3) 执行 `insmod /modules/i2c.ko`。在驱动加载成功后会打印出设备的 I2C 注册地址。具体添加设备的地址查看数据手册，每一个设备都会有一个固定地址。添加设备地址为 0x6f，转化成二进制为一个 8 位的数据。I2C 协议地址为 7 位。

```
[root@zjut /]# insmod /modules/i2c.ko
[ 1104.720000] SY_init
[ 1104.750000] SY_detect
[ 1104.800000] SY_probe with name = SY, addr = 0x6f
[root@zjut /]#
```

图 8-14 手动加载驱动模块

- 4) 驱动加载成功之后，查看 devices 和 drivers 目录，会增加一个 0-006f 文件。

```
[root@zjut /]# cd /sys/bus/i2c
[root@zjut i2c]# ls devices/
0-0004  0-0018  0-003c  0-006f
[root@zjut i2c]# ls drivers/
OV5640 channel0 Video Decoder I2C driver
SY
aic3x I2C Codec
davinci_evm
dev_driver
egalax_i2c
i2c_adapter
[root@zjut i2c]#
```

图 8-15 设备 SY 注册成功

- 5) 执行 `cat /proc/device`, 显示加载的 I2C 设备驱动程序创建了一个主设备号为 245 名为 SY 的设备节点。

```
[root@zjut i2c]# cat /proc/devices
Character devices:
1 mem
4 /dev/vc/0
4 tty
4 ttyS
5 /dev/tty
5 /dev/console
5 /dev/ptmx
7 vcs
10 misc
13 input
14 sound
21 sg
29 fb
81 video4linux
89 i2c
90 mtd
108 ppp
116 alsa
128 ptm
136 pts
180 usb
188 ttyUSB
189 usb_device
199 dm365_gpio
245 SY
246 dm365mmap
247 edma
```

图 8-16 创建了设备号 245 名为 SY 的设备节点

#### 步骤 4: 编写测试程序, 并进行调试

测试程序在文件夹 I2C 驱动实验/I2C/i2c\_test 下, i2c\_test.c.就是对应的测试文件。用来查看设备的地址。

i2c\_test.c 测试代码:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
int main(int argc, char **argv)
{
    int fd,ret;
    unsigned char values[0];
    fd = open("/dev/SY", O_RDWR);
```

```

if (fd < 0)
{
    printf("can't open /dev/SY\n");
    return -1;
}
ret= read(fd,values,sizeof(unsigned char));
if (ret >= 0){
    printf("reading data is OK \n");
}
else{
    printf("read data is fialed \n",ret);
}
printf("SY address = 0x%x\n",values[0]);
return 0;
}

```

重置生效环境变量，使用交叉编译工具编译测试程序，并将编译后生成的可执行文件挂载到实验箱上运行调试。

```
$ arm_v5t_le-gcc i2c_test.c -o i2c_test
```

```

shiyang@ubuntu:~/Desktop/shiyang/I2C$ ls
i2c.c i2c.ko i2c.mod.c i2c.mod.o i2c.o Makefile Module.symvers
shiyang@ubuntu:~/Desktop/shiyang/I2C$ vim i2c_test.c
shiyang@ubuntu:~/Desktop/shiyang/I2C$ arm_v5t_le-gcc i2c_test.c -o i2c_test
shiyang@ubuntu:~/Desktop/shiyang/I2C$ ls
i2c.c i2c.ko i2c.mod.c i2c.mod.o i2c.o i2c_test i2c_test.c Makefile Module.symvers
shiyang@ubuntu:~/Desktop/shiyang/I2C$

```

图 8-17 生成 i2c\_test 可执行文件

将交叉编译生成的 i2c\_test 文件拷贝到挂载的文件系统目录下

```

shiyang@ubuntu:~/Desktop/shiyang/I2C$ cp i2c_test /home/shiyang/share/filesys_test/opt/dm365
shiyang@ubuntu:~/Desktop/shiyang/I2C$

```

putty 实验箱窗口执行 i2c\_test，读出当前设备地址。

```

[root@zjut /]# cd opt
[root@zjut opt]# cd dm365
[root@zjut dm365]# ls
3g_guard.sh      dev.pcap      i2c_test_8bit  pollcsq
4g_mceb.sh      dev1.pcap     i2c_test_at24  r_agc.sh
Config.dat      dev_app_cl    image         recv
a.out          dev_app_pn    io           rtc_test
adctest        dm365mmap.ko  iptables.sh   script
agc.sh         edmak.ko      irqk.ko       sip_app
agc_check.sh   encode        lcd_evm       startup_mceb.sh
amixer         encode.log    lcdtest       task_db_1.sh
arecord        encode_mceb   led           task_db_2.sh
blend          getip.sh     led_on.bin    temp
call           gpiotest     led_on_elf    tvp2ov.sh
check_u6100     gps_app      lm.sh         uart57600
check_u9600     gpscfg.xml   longPressKey  wlw
clear.sh       guard_wodma.sh ls            wlw.tar.gz
cmemk.ko       hello        myThread     wlwov
czzq          helloworld   ov2tvp5151.sh
daemon         i2c_test     play
data          i2c_test_5151_1 pnrtc
[root@zjut dm365]# i2c_test
reading data is OK
SY address = 0x6f
[root@zjut dm365]#

```

图 8-18 测试执行程序

## 4 心得与体会

这是嵌入式系统的最后一次实验，通过这次实验，对 I2C 驱动程序的编写有更为深入的了解；I2C 为常见的总线接口，后续学习和工作中很有可能会经常用到，所以我们还是要继续深入了解熟悉 I2C 相关驱动程序，对于其驱动的理解将为我们后续从事嵌入式开发打下基础。