LINUX(ANDROID) IIC DRIVER SOLUTION

November 8, 2011

Bee

Abstract

This document describes the compiling and installing details of pixcir IIC touch screen solution driver for Linux kernel(including Android) on ARM platform,.All compiling work is under Ubuntu 10.04.This driver could support 5 fingers touch for TangoC solution.

Attention： Make sure the Linux or Android kernel version and compile tools' version are same as the kernel the pixcir_i2c_ts.ko used. The Linux kernel is 2.6.30 or later could support ABS_MT event which is needed by multi fingers touch. It need request_threaded_irq function so could not compile with earlier than 2.6.30.

Linux coordinate resolution is 0 ~ MAX-1,
PIXCIR TP resolution is 1 ~ MAX.

Suitable List This Driver Could Be Used

Support Linux kernel Version

2.6.30~ Support Multi touch

Contents

1 IIC Driver

Following steps are under Ubuntu 10.04.

In the src folder include three files:

pixcir_i2c_ts.c	--- .c source code
pixcir_i2c_ts.h	--- .h source code
Makefile	--- for compiling kernel module

1.1 Rebuild Kernel

>>>The follow example are based on the samsung s3c6410 dev board<<<

The IIC device driver used “probe” mode. So the driver will probe IIC device when loading the driver by matching the name in the i2c_device_id structure and in the i2c_board_info structure.If the string is same the Linux kernel will load the driver and continue to run. For example in the driver source file the i2c_device_id structure as show below.

```
static const struct i2c_device_id pixcir_i2c_ts_id[] = {
  { "pixcir_ts", 0 },
  { }
};
MODULE_DEVICE_TABLE(i2c, pixcir_i2c_ts_id);
```

And the i2c_board_info is defined in the file mach-smdk6410.c whose path is /arch/arm/mach-s3c64xx in the kernel source files of s3c6410.If the mach init file do not include the “struct i2c_board_info” we need add it by self. It is need IIC slave device information including name,slave device address,irq and so on.The definition is as below.

```
static struct i2c_board_info i2c_devs0[] __initdata = {
  {
I2C_BOARD_INFO("pixcir_ts", 0x5c),
.irq = S3C_EINT(11),
  },
};
```

`.irq = S3C_EINT(11)` should be modified according to your interrupt source. Then regist the board information in the function smdk6410_machine_init. smdk6410_machine_init or your platform initial function.

```
static void __init smdk6410_machine_init(void)
      {
        …
       I i2c_register_board_info(0, i2c_devs0, ARRAY_SIZE(i2c_devs0));
        …
      }
```

At last rebuild the Linux kernel.

2 Pixcir IIC Touch Screen Driver

2.1 Compile pixcir_i2c_ts Driver

The follow steps will build the ko file out of the kernel.
Before the follow steps, make sure you have installed the compile toolchain in Ubuntu and edit the Makefile correctly.

First**,** you will find the following definition in pixcir_i2c_ts.h file. These are macro definitions according to customer's touch panel resolution and which solution used..And according to your PCB board please define attb and reset pin.

```
#define ATTB XXXX          //define your ATTB(INT) pin
#define get_attb_value     //define function to get the value of the attb pin
#define RESETPIN_CFG   //define function to config the reset pin
#define RESETPIN_SET0   //define function to set the reset pin to 0
#define RESETPIN_SET1   //define function to set the reset pin to 1
```

For Example:

```
/*****************touchscreen resolution setting*************/
//#define BUTTON

static int attb_read_val(void);
static void tangoC_init(void);
static void tangoC_reset(void);
#define X_MAX 1024
#define Y_MAX 768


//Platform gpio define
//#define S5PC1XX
```

```
#ifdef S5PC1XX
  #include <plat/gpio-bank-e1.h> //reset pin GPE1_5
  #include <plat/gpio-bank-h1.h> //attb pin GPH1_3
  #include <mach/gpio.h>
  #include <plat/gpio-cfg.h>

  #define ATTB         S5PC1XX_GPH1(3)
  #define get_attb_value gpio_get_value
  #define RESETPIN_CFG
  s3c_gpio_cfgpin(S5PC1XX_GPE1(5),S3C_GPIO_OUTPUT)
  #define RESETPIN_SET0
  gpio_direction_output(S5PC1XX_GPE1(5),0)
  #define RESETPIN_SET1
  gpio_direction_output(S5PC1XX_GPE1(5),1)

#else      //mini6410

  #include <plat/gpio-cfg.h>
  #include <mach/gpio-bank-e.h>
  #include <mach/gpio-bank-n.h>
  #include <mach/gpio.h>

  #define ATTB         S3C64XX_GPN(11)
  #define get_attb_value gpio_get_value
  #define RESETPIN_CFG
  s3c_gpio_cfgpin(S3C64XX_GPE(1),S3C_GPIO_OUTPUT)
  #define RESETPIN_SET0
  gpio_direction_output(S3C64XX_GPE(1),0)
  #define RESETPIN_SET1
  gpio_direction_output(S3C64XX_GPE(1),1)
#endif

static int attb_read_val(void)
{
  return get_attb_value(ATTB);
}

/*static void tangoC_init(void)
{
  RESETPIN_SET0;
}*/
```

Third, after these modifications, you could compile pixcir_i2c_ts as a kernel

module with Makefile in the src folder.

```
pixcir@pixcir-laptop:~/pixcir-i2c-ts$ make
make ARCH=arm CROSS_COMPILE=/usr/local/arm/4.5.1/bin/arm-linux-  -C
/home/pixcir/linux2.6.36-for-mini6410/linux-2.6.36  M=/home/pixcir/pixcir-i2c-ts
modules
make[1]: Entering directory `/home/pixcir/linux2.6.36-for-mini6410/linux-2.6.36'
  CC [M]  /home/pixcir/pixcir-i2c-ts/pixcir_i2c_ts.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC       /home/pixcir/pixcir-i2c-ts/pixcir_i2c_ts.mod.o
  LD [M]  /home/pixcir/pixcir-i2c-ts/pixcir_i2c_ts.ko
make[1]: Leaving directory `/home/pixcir/linux2.6.36-for-mini2440/linux-2.6.36'
```

If you see the information which is similar with the above, it success and will create pixcir_i2c_ts.ko file.

2.2 Insmod Driver Manually

Following steps are under Linux or Android system which is running on the develop board.

You can copy the pixcir_i2c_ts.ko to the Linux or Android file system running on the dev board and insmod it.

```
# insmod pixcir_i2c_ts.ko
input: pixcir_ts as /class/input/input0
pixcir_i2c_ts_driver_v3.0.0-005c: insmod successfully!

# lsmod
pixcir_i2c_ts 4068 0 - Live 0xbf000000
```

After running lsmod command, you will find pixcir_i2c_ts module as above.

And also you could find a device named pixcir_i2c_ts0 under /dev folder.Which is the device the app programmer could be used to communicate with touch panel device.

2.3 Testing Touch Screen Detection

In this section we will determine which driver is used on our touch screen device.

After the steps before. Connect the touch screen to your device correctly and load the driver.

Run the command cat /proc/bus/input/devices would list something similiar to the following:

```
I: Bus=0018 Vendor=0000 Product=0000 Version=0000
N: Name="pixcir_ts"
P: Phys=
S: Sysfs=/class/input/input0
U: Uniq=
H: Handlers=mouse1 event1
B: EV=b
B: KEY=400 0 0 0 0 0 0 0 0 0 0
B: ABS=650000 3
```

Name ="xxxx" means a touch screen ID. Handlers=mouse1 event1 means the touch screen reported to /dev/input/event1.
If there is no "xxxx" after Name=, you need to update pixcir_i2c_ts.c.

If you have the tool getevent you could run the command
#getevent /dev/input/event1
to test the deriver’s report events to the kernel. After running the command and touching the touch screen,if you can get the information as bellow it is successful.

```
# getevent /dev/input/event1
0001 014a 00000001
0003 0000 00000126
0003 0001 000001d0
0000 0000 00000000
0001 014a 00000000
0003 0000 00000000
0003 0001 00000000
0001 014a 00000001
0003 0000 000000fd
0003 0001 00000154
0000 0000 00000000
0001 014a 00000000
0003 0000 00000126
0003 0001 000001cf
0003 0000 00000000
0003 0001 00000000
0000 0000 00000000
```

Now it can support two fingers touch.

2.4 Calibration Function

When the touch panel install to the electronic device, because the environment of the touch panel is changed after the device on and it need to do calibration for the touch panel. One time calibration is enough.

In the app program, could follow the steps to do calibration:
Step 1: Open the device pixcir_i2c_ts0
Step 2: Change the touch panel to calibration mode and write the calibration command 0x3A 0x03 to the touch panel.
Step 3: If rceive the correct return value(here is 2),success..
Step 4:Close the device

Sample code for app to do calibration :

```
#define TS ("/dev/pixcir_i2c_ts0")
#define  CALIBRATION_FLAG    1

fd = open(TS, O_RDWR));

    //CALIBRATION
    buf[0] = 0x3A;
    buf[1] = 0x03;

   ioctl(touchfd,CALIBRATION_FLAG,arg); //change the touch panle to calibration mode
                                        //parameter arg is not nsed here

    ret = write(touchfd,buf,count);     //write the calibration command to tp by IIC
    if(ret==2){
        printf("calibration successfule\n");
    }
    else
        printf("calibration faild\n");
close(fd);
```

2.5 Bootloader Function

App will call this function to update the TP firmware.

```
#define BOOTLOADER      7
```

2.6 Reset TP Function

```
#define RESET_TP      9
```

2.7 Read Bootloader Status Function

```
#define    BOOTLOADER_STU          12
```

2.8 Read Attb Value Function

```
#define ATTB_VALUE          13
```

2.9 Default Read/Write Interface

These modes could use the tools pixcir supplied to finish.

More details please refer the terminal tool spec.

3 Conclusions

This touch screen driver for android which just need kernel version support ABS_MT event and request_threaded_irq function. The org kernel version should be 2.6.30 or later.

4 Release History

version	content	pages	author	reviewer	date
3.0	Initial document	7	Bee	-	2011-07-14
3.1	Add bootloader function Add reset tp function Add read bootloader status Add universal W/R interface	11	Bee	-	2011-9-9
3.2	Arrange point by pixcir slot Use 0x09 INT MODE Add power management	11	Bee	-	2011-11-08