LINUX(ANDROID) IIC DRIVER SOLUTION

July 15, 2011

Hangsang

Abstract

This document describes the compiling and installing details of pixcir IIC touch screen solution driver for Linux kernel(including Android) on ARM platform, kernel version is 2.6.36.All compiling work is under Ubuntu 10.04.This driver could support two fingers touch.

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 two fingers touch.The Linux kernel 2.6.29 in Android has been modified by Google which could support ABS_MT event.

Suitable List This Driver Could Be Used

Linux kernel Version

2.6.26~2.6.29	Support Single touch
2.6.29 google version	Support Multi touch
2.6.30~	Support Multi touch

Hardware Platform

ARM Serial CPU

Pixcir IC set

ATMEL_168	default needn't define in pixcir_i2c_ts.h
R8C_3GA_2TG	define R8C_3GA_2TG in pixcir_i2c_ts.h
R8C_AUO_I2C	define R8C_AUO_I2C in pixcir_i2c_ts.h
M48	define M48 in pixcir_i2c_ts.h
M48_F32	define M48_F32 in pixcir_i2v_ts.h

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 used. Then register the board information in the function smdk6410_machine_init. smdk6410_machine_init

```
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 ATMEL_168
//#define R8C_3GA_2TG
//#define R8C_AUO_I2C
#define M48
#define M48_F32

#ifdef R8C_AUO_I2C
  #ifndef R8C_3GA_2TG
  #define R8C_3GA_2TG
  #endif
#endif
```

```
#define TOUCHSCREEN_MINX 0
#define TOUCHSCREEN_MAXX 800
#define TOUCHSCREEN_MINY 0
#define TOUCHSCREEN_MAXY 480


#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)
```

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
/home/pixcir/pixcir-i2c-ts/pixcir_i2c_ts.c: In function 'pixcir_i2c_ts_probe':
/home/pixcir/pixcir-i2c-ts/pixcir_i2c_ts.c:137: warning: ISO C90 forbids mixed
declarations and code
  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
pixcir_i2c_init
pixcir_i2c_ts_probe
input: pixcir_ts as /class/input/input0
pixcir_i2c_ts_driver_v2.3.1-005c: 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
```

```
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 0x37 0x03 to the touch panel. If using the R8C AUO solution the

command is 0x78 0x03.
Step 3: Delay 1~5s which depend on the touch panel size.
Step 4:Close the device

Sample code for app to do calibration :

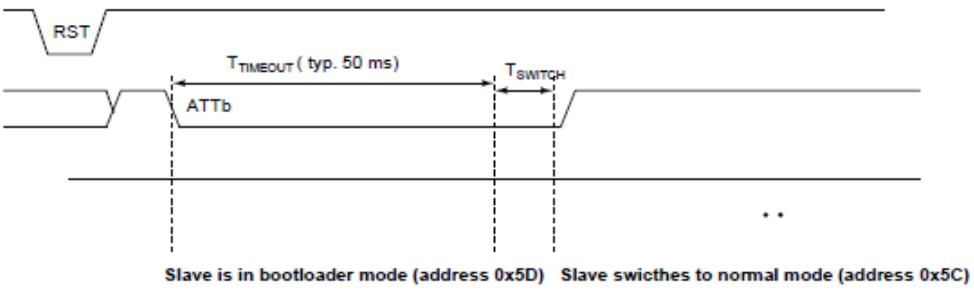
```
#define TS ("/dev/pixcir_i2c_ts0")
#define  CALIBRATION_FLAG    1

fd = open(TS, O_RDWR));

    //CALIBRATION
    buf[0] = 0x37;
    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
    mdelay(3000);                 //delay several seconds
    if(ret==2){
        printf("calibration successfule\n");
    }
    else
        printf("calibration faild\n");
close(fd);
```

2.5 Bootloader Function

Touch panel's firmware could be updated by this function.The firmware file must be *.pix.

It is needed to use the reset pin of the touch panel to use this function to download the firmware.

The reset pin's sequence like this:



2.6 Normal Mode

In this mode touch information including touch,oldtouch,position1 and position2 could be read.The AUO solution has no oldtouch.

The app program could enter this mode by the follow function:

```
#define  NORMAL_MODE      2

ioctl(touchfd,NORMAL_MODE,arg);
```

2.7 Debug Mode

In this mode the rawdata could be read.
The app program could enter this mode by the follow function:

```
#define  DEBUG_MODE       3

ioctl(touchfd, DEBUG_MODE,arg); //parameter arg define the Tango number(1 or 2 usually)
```

2.8 Read Version

In this mode the firmware version could be read.
The app program could enter this mode by the follow function:

```
#define  VERSION_FLAG         6
```

2.9 Internal Mode

In this mode the rawdata after calibration could be read.
The app program could enter this mode by the follow function:

```
#define  INTERNAL_MODE        4
```

2.10 Button_debug_mode

In this mode the button raw data could be read .
The app program could enter this mode by the follow function:

```
#define  BUTTON_DEBUG_MODE    14
```

2.11 Button_normal

In this mode the button_flags register could be read to check the button touch status and support eight buttons solution.
The app program could enter this mode by the follow function:

```
#define  BUTTON_NORMAL        15
```

These modes could use the tools pixcir supplied to finish.

More details please refer the pixi2c tool spec.

3 Conclusions

The touch screen driver for linux or android2.0(2.1,2.2).Adroid2.0(2.1,2.2) kernel version is 2.6.29, linux kernel is 2.6.30.if you want to use multi-touch the kernel need to support ABS_MT event.The org kernel version should be 2.6.30 or later.and If the android kernel is 2.6.29 it has been modified.

4 Review History

version	content	pages	author	reviewer	date
1.0	Initial document	7	Bee	Reed	2010-07-19
2.0	1.Add interface for app 2.Add calibration,debug,norm and bootloader mode 3.Add R8C(Pixcir&AUO solution	12	Bee	Reed	2010-12-06
2.1	1.Add read firmware version 2.Read internal data	12	Bee	Reed	2011-1-14
2.3.1	1.Add M48 solution 2.pixcir_i2c_ts0	12	Bee		2011-5-16
2.5	1.Add M48_F32 Solution 2.Add Internal mode (m48，m48_F32 tested)	12	Hangsang	Bee	2011-07-15
2.6	1.Add button_normal 2.Add button_debug_mode (m48，m48_F32 tested)	12	Hangsang	Bee	2011-08-03