

Multi-Touch linux device driver專題

NXP IMX6ULL ARM M7

2024/07/27
余志佳

Embedded linux and Embedde device driver

- 專題說明
- Embedded linux device driver
 - Linux multi-touch driver
- Note

專題說明:

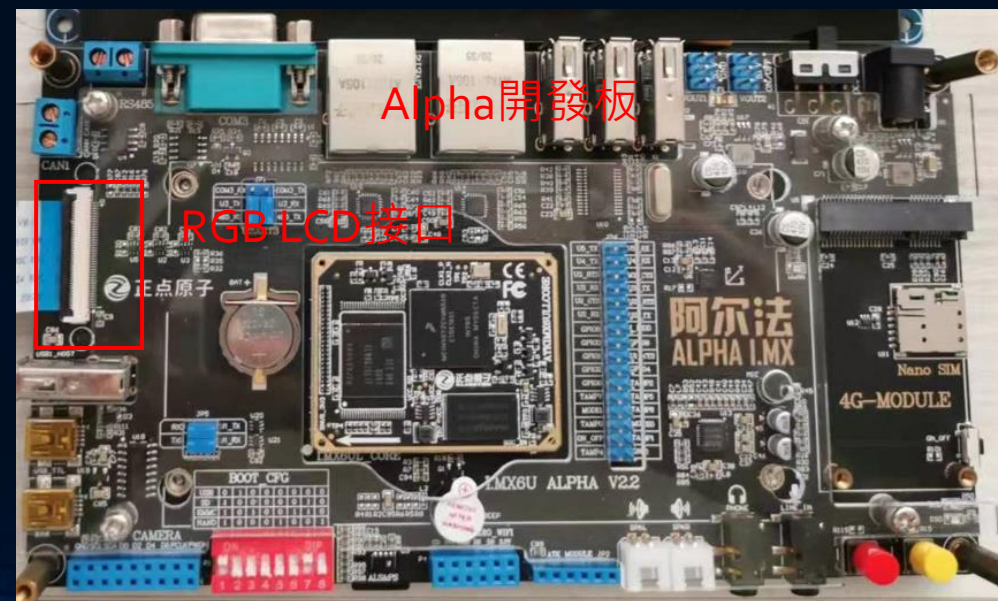
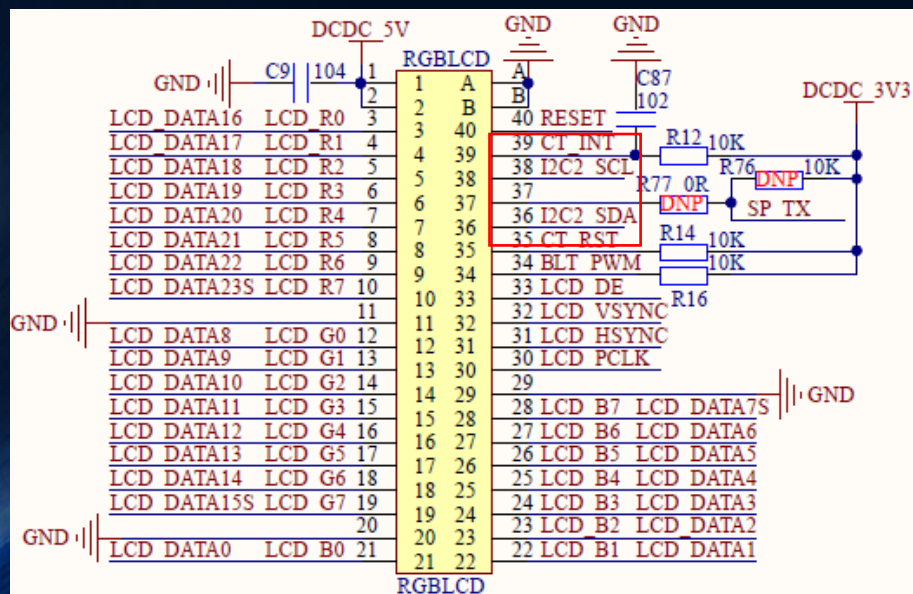
1. Embedded linux device driver : Linux multi-touch driver

在移植好linux系統環境下，本專題編寫 linux 多點電容觸摸屏驅動，其觸摸控制 IC 採用FocalTech的 controller "FT5426"，其結構採用I2C接口，MT Type B協議。

2. 我們要做的觸控式螢幕的驅動從大框架上來看就是個I2C的設備驅動，一旦螢幕被觸摸，FT5426給SOC(NXP I.MX6ULL)觸發一個外部中斷，中斷處理函數就會從IC裡獲取到觸摸的相關資訊。透過input子系統，觸控式螢幕屬於輸入裝置，必然也屬於這個input子系統。所以我們需要通過input子系統按照Linux的內核規定的規則上報一個輸入事件。然後內核通過相關的協議(MT Type B)去分析觸摸的資訊。

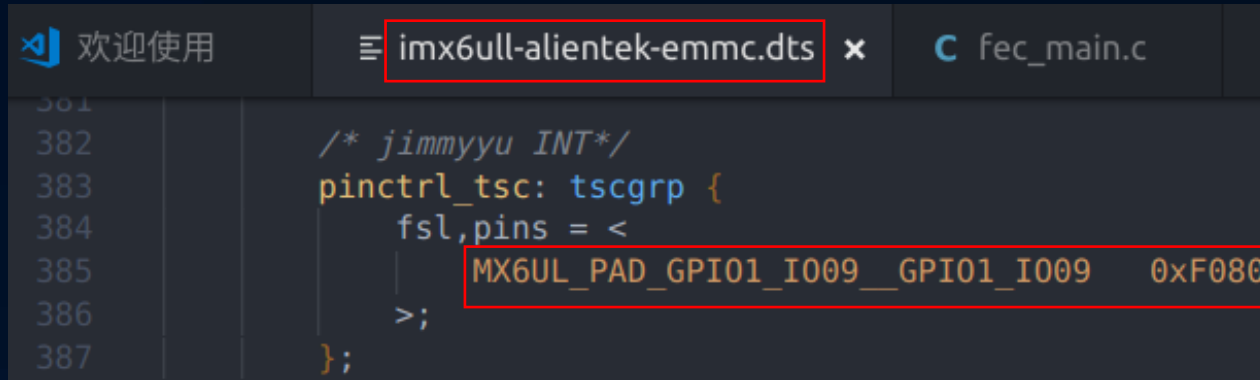
Linux Multi-Touch driver：1.觸摸屏芯片(FT5426)簡介和周邊電路

1. 屏幕採用ATK-7016模塊，為7吋TFT LCD+電容觸摸屏(驅動IC:FT5426)組合起來。SOC(NXP I.MX6ULL)透過I2C接口讀取觸摸屏的觸摸座標數據。
2. 觸摸屏有4個IO連接CPU，分別為SCL、SDA、RST和INT。SCL和SDA是I2C引腳，RST是復位引腳，INT是中斷引腳。
3. 觸摸屏和TFT LCD做在一起，在Alpha開發板底板的RGB LCD接口上，原理圖如下所示:由下圖左可知，觸摸屏SCL/SDA接到SOC的I2C2 IO，INT連接到SOC的GPIO1_IO09，RST連接到SOC的SNVS_TAMPER9。



Linux Multi-Touch driver : 2.驅動程序編寫:修改設備樹

1. 添加FT5426所使用的IO:在linux kernel工程的imx6ull-alientek-emmc.dts文件添加FT5426所使用的4個IO(INT , RST , SCL , SDA)。



```
382  /* jimmyyu INT*/
383  pinctrl_tsc: tscgrp {
384      fsl,pins = <
385          MX6UL_PAD_GPI01_I009__GPI01_I009    0xF080
386      >;
387  };
```

2. 復位引腳使用的是SNVS_TAMPER9 , 因此要添加到iomuxc_snvs節點下。

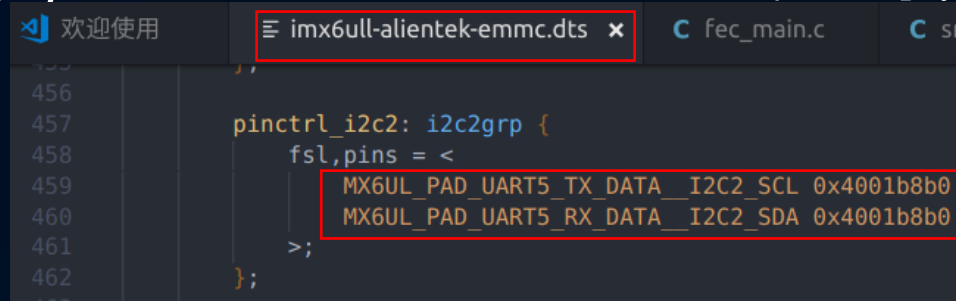
新建子節點pinctrl_tsc_reset



```
687  };
688
689  /*jimmyyu MT RST*/
690  pinctrl_tsc_reset: tsc_reset{
691      fsl,pins = <
692          MX6ULL_PAD_SNVS_TAMPER9__GPI05_I009    0x10B0
693      >;
694  };
```

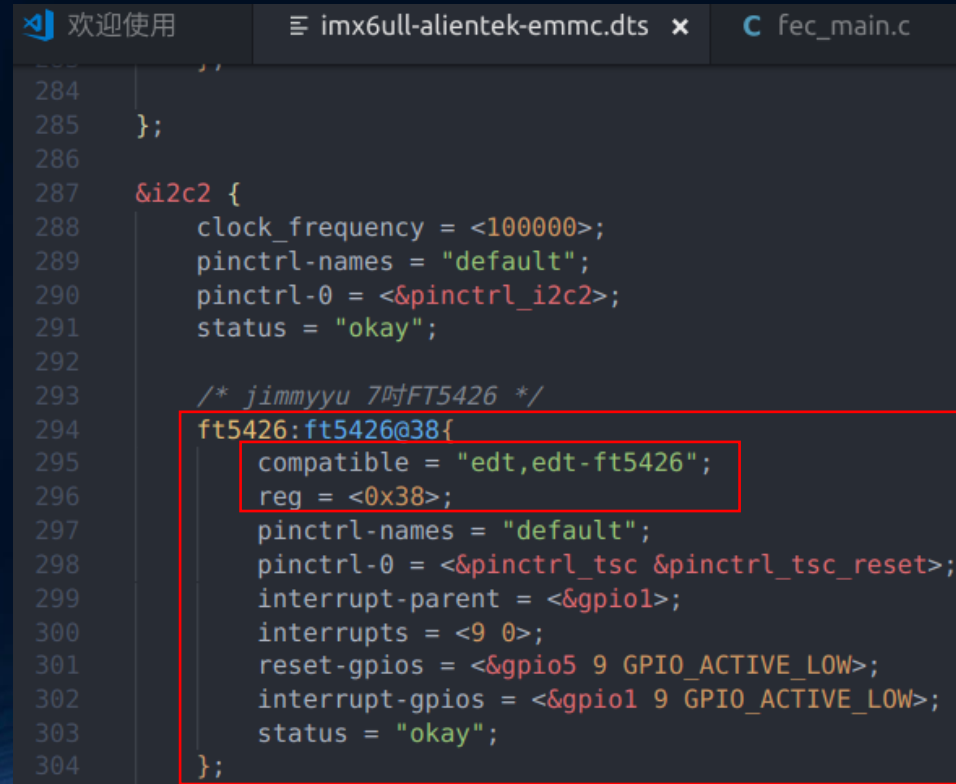
Linux Multi-Touch driver : 2.驅動程序編寫:修改設備樹

- 繼續添加I2C2的SCL和SDA IO，因為SOC NXP官方已經添加在linux kernel工程/imx6ull-alientek-emmc.dts裡，我們就不需要去修改。



```
456
457 pinctrl_i2c2: i2c2grp {
458     fsl,pins = <
459         MX6UL_PAD_UART5_TX_DATA__I2C2_SCL 0x4001b8b0
460         MX6UL_PAD_UART5_RX_DATA__I2C2_SDA 0x4001b8b0
461     >;
462 };
```

- 添加FT5426節點:FT5426是掛載在I2C2下，因此需要向I2C2節點下添加一個子節點。



```
284
285 };
286
287 &i2c2 {
288     clock_frequency = <100000>;
289     pinctrl-names = "default";
290     pinctrl-0 = <&pinctrl_i2c2>;
291     status = "okay";
292
293     /* jimmyyu 7吋FT5426 */
294     ft5426:ft5426@38{
295         compatible = "edt,edt-ft5426";
296         reg = <0x38>;
297         pinctrl-names = "default";
298         pinctrl-0 = <&pinctrl_tsc &pinctrl_tsc_reset>;
299         interrupt-parent = <&gpio1>;
300         interrupts = <9 0>;
301         reset-gpios = <&gpio5 9 GPIO_ACTIVE_LOW>;
302         interrupt-gpios = <&gpio1 9 GPIO_ACTIVE_LOW>;
303         status = "okay";
304     };
```


Linux Multi-Touch driver : 2.驅動程序編寫:編寫FT5426驅動

1. 創建vscode工程，新建ft5426.c，定義觸摸點狀態，FT5426寄存器相關宏定義和觸摸屏設備結構體。 FT5426支援5點觸摸。

```
歡迎使用  C ft5426.c  x
33  #define MAX_SUPPORT_POINTS      5          /* 5点触摸 */
34  #define TOUCH_EVENT_DOWN        0x00       /* 按下 */
35  #define TOUCH_EVENT_UP          0x01       /* 抬起 */
36  #define TOUCH_EVENT_ON          0x02       /* 接触 */
37  #define TOUCH_EVENT_RESERVED    0x03       /* 保留 */
38
39  /* FT5X06寄存器相关宏定义 */
40  #define FT5X06_TD_STATUS_REG     0X02       /* 状态寄存器地址 */
41  #define FT5X06_DEVICE_MODE_REG  0X00       /* 模式寄存器 */
42  #define FT5426_IDG_MODE_REG     0XA4       /* 中断模式 */
43  #define FT5X06_READLEN          29         /* 要读取的寄存器个数 */
44
45  struct ft5x06_dev{
46      struct device_node *nd;                /* 设备节点 */
47      int irq_pin,reset_pin;                 /* 中断和复位IO */
48      int irq_num;                          /* 中断号 */
49      void *private_data;                   /* 私有数据 */
50      struct i2c_client *client;            /* I2C客户端 */
51      struct input_dev *input;              /* input结构体 */
52
53  };
54  static struct ft5x06_dev ft5x06;
```

Linux Multi-Touch driver : 2.驅動程序編寫:編寫FT5426驅動

5. FT5426的ft5x06_ts_driver註冊與註銷。

```
371
372  /*传统的匹配表*/
373  static struct i2c_device_id ft5x06_ts_id[] = {
374      {"edt-ft5206", 0},
375      {"edt-ft5426", 0},
376      {}
377  };
378
379  /*设备树匹配表*/
380  static struct of_device_id ft5x06_of_match[] = {
381      { .compatible = "edt,edt-ft5206" },
382      { .compatible = "edt,edt-ft5406" },
383      { /* sentinel*/}
384  };
385
386  /*i2c_driver*/
387  static struct i2c_driver ft5x06_ts_driver = {
388      .driver = {
389          .owner = THIS_MODULE,
390          .name = "edt_ft5x06",
391          .of_match_table = of_match_ptr(ft5x06_of_match),
392      },
393      .id_table = ft5x06_ts_id,
394      .probe = ft5x06_ts_probe,
395      .remove = ft5x06_ts_remove,
396
397  };
```

```
395      .remove = ft5x06_ts_remove,
396  };
397
398
399  /*驱动入口函数*/
400  static int __init ft5x06_init(void)
401  {
402      int ret;
403      ret = i2c_add_driver(&ft5x06_ts_driver);
404      return ret;
405  }
406
407  /*驱动出口函数*/
408  static void __exit ft5x06_exit(void)
409  {
410      i2c_del_driver(&ft5x06_ts_driver);
411  }
412
413  module_init(ft5x06_init);
414  module_exit(ft5x06_exit);
415  MODULE_LICENSE("GPL");
416  MODULE_AUTHOR("jimmyyu");
```


Linux Multi-Touch driver : 2.驅動程序編寫:編寫FT5426驅動

6. 編寫FT5426暫存器讀寫API，用於向I2C設備在probe時寫多個寄存器數據初始化及中斷時讀取多個寄存器的觸摸數據。

```
85  * @description : 从FT5X06读取多个寄存器数据
86  * @param - dev:  ft5x06设备
87  * @param - reg:  要读取的寄存器首地址
88  * @param - val:  读取到的数据
89  * @param - len:  要读取的数据长度
90  * @return      :  操作结果
91  */
92  static int ft5x06_read_regs(struct ft5x06_dev *dev, u8 reg, void *val, int len)
93  {
94      int ret;
95      struct i2c_msg msg[2];
96      struct i2c_client *client = (struct i2c_client *)dev->client;
97
98      /* msg[0]为发送要读取的首地址 */
99      msg[0].addr = client->addr;          /* ft5x06地址 */
100     msg[0].flags = 0;                    /* 标记为发送数据 */
101     msg[0].buf = &reg;                  /* 读取的首地址 */
102     msg[0].len = 1;                      /* reg长度*/
103
104     /* msg[1]读取数据 */
105     msg[1].addr = client->addr;          /* ft5x06地址 */
106     msg[1].flags = I2C_M_RD;            /* 标记为读取数据*/
107     msg[1].buf = val;                   /* 读取数据缓冲区 */
108     msg[1].len = len;                   /* 要读取的数据长度*/
109
110     ret = i2c_transfer(client->adapter, msg, 2);
111     if(ret == 2) {
112         ret = 0;
113     } else {
114         ret = -EREMOTEIO;
115     }
116     return ret;
117 }
```

```
117 }
118
119 /*
120  * @description : 向ft5x06多个寄存器写入数据
121  * @param - dev:  ft5x06设备
122  * @param - reg:  要写入的寄存器首地址
123  * @param - val:  要写入的数据缓冲区
124  * @param - len:  要写入的数据长度
125  * @return      :  操作结果
126  */
127 static s32 ft5x06_write_regs(struct ft5x06_dev *dev, u8 reg, u8 *buf, u8 len)
128 {
129     u8 b[256];
130     struct i2c_msg msg;
131     struct i2c_client *client = (struct i2c_client *)dev->client;
132
133     b[0] = reg;                          /* 寄存器首地址 */
134     memcpy(&b[1], buf, len);             /* 将要写入的数据拷贝到数组b里面 */
135
136     msg.addr = client->addr;              /* ft5x06地址 */
137     msg.flags = 0;                        /* 标记为写数据 */
138
139     msg.buf = b;                          /* 要写入的数据缓冲区 */
140     msg.len = len + 1;                    /* 要写入的数据长度 */
141
142     return i2c_transfer(client->adapter, &msg, 1);
143 }
```

Linux Multi-Touch driver : 2.驅動程序編寫:編寫ICM20608驅動

7. probe函數執行:當設備與驅動匹配，probe函數就會執行，此函數完成ft5426設備及中斷的初始化，調用input_set_abs_params函數設置EV_ABS事件需要上報ABS_X、ABS_Y、ABS_MT_POSITION_X、ABS_MT_POSITION_Y,最後調用input_register_device函數向系統註冊ft5426設備。

```
C ft5426.c
288 static int ft5x06_ts_probe(struct i2c_client *client,
289                             const struct i2c_device_id *id)
290 {
291     int ret = 0;
292     //int value = 0;
293     printk("ft5x06_probe!\r\n");
294
295     ft5x06.client = client;
296
297     /* 1, 获取设备树中的中断和复位引脚 */
298     ft5x06.irq_pin = of_get_named_gpio(client->dev.of_node, "interrupt-gpios", 0);
299     ft5x06.reset_pin = of_get_named_gpio(client->dev.of_node, "reset-gpios", 0);
300     /* 2, 复位FT5x06 */
301     ret = ft5x06_ts_reset(client, &ft5x06);
302     if (ret < 0) {
303         goto fail;
304     }
305     /* 3, 初始化中断 */
306     ret = ft5x06_ts_irq(client, &ft5x06);
307     if (ret < 0) {
308         goto fail;
309     }
310     /* 4, 初始化FT5X06 */
311     ft5x06_write_reg(&ft5x06, FT5X06_DEVICE_MODE_REG, 0); /* 进入正常模式 */
312     ft5x06_write_reg(&ft5x06, FT5426_IDG_MODE_REG, 1); /* FT5426中断模式 */
```

```
C ft5426.c
316
317     /* 5, input设备注册 */
318     ft5x06.input = devm_input_allocate_device(&client->dev);
319     if (!ft5x06.input) {
320         ret = -ENOMEM;
321         goto fail;
322     }
323     ft5x06.input->name = client->name;
324     ft5x06.input->id.bustype = BUS_I2C;
325     ft5x06.input->dev.parent = &client->dev;
326
327     __set_bit(EV_SYN, ft5x06.input->evbit);
328     __set_bit(EV_KEY, ft5x06.input->evbit);
329     __set_bit(EV_ABS, ft5x06.input->evbit);
330     __set_bit(BTN_TOUCH, ft5x06.input->keybit);
331
332     /* Single touch */
333     input_set_abs_params(ft5x06.input, ABS_X, 0, 1024, 0, 0);
334     input_set_abs_params(ft5x06.input, ABS_Y, 0, 600, 0, 0);
335     /* Multi touch */
336     input_set_abs_params(ft5x06.input, ABS_MT_POSITION_X, 0, 1024, 0, 0);
337     input_set_abs_params(ft5x06.input, ABS_MT_POSITION_Y, 0, 600, 0, 0);
338     ret = input_mt_init_slots(ft5x06.input, MAX_SUPPORT_POINTS, 0);
339     if (ret) {
340         goto fail;
341     }
342
343     ret = input_register_device(ft5x06.input);
344     if (ret)
345         goto fail;
346
347     return 0;
348
349 fail:
350     return ret;
351
352 }
```

Linux Multi-Touch driver : 2.驅動程序編寫:

8. 編寫中斷函數:通過ft5x06_read_regs函數讀取FT5426觸摸點訊息暫存器數據，參考spec，從0X02地址開始，共29暫存器，for循環則為上報觸摸點座標數據，使用TypeB時序，最後通過input_sync函數上報SYN_REPORT事件。

```
C ft5426.c
177  * @param - dev_id : 设备结构。
178  * @return      : 中断执行结果
179  */
180  static irqreturn_t ft5x06_handler(int irq, void *dev_id)
181  {
182      struct ft5x06_dev *multidata = dev_id;
183
184      u8 rdbuf[29];
185      int i, type, x, y, id;
186      int offset, tplen;
187      int ret;
188      bool down;
189
190      offset = 1; /* 偏移1, 也就是0X02+1=0x03,从0X03开始是触摸值 */
191      tplen = 6; /* 一个触摸点有6个寄存器来保存触摸值 */
192
193      //printf("ft5x06_handler\r\n");
194      memset(rdbuf, 0, sizeof(rdbuf)); /* 清除 */
195
196      /* 读取FT5X06触摸点坐标从0X02寄存器开始, 连续读取29个寄存器 */
197      ret = ft5x06_read_regs(multidata, FT5X06_TD_STATUS_REG, rdbuf, FT5X06_READLEN);
198      if (ret) {
199          goto fail;
200      }
201
202      /* 上报每一个触摸点坐标 */
203      for (i = 0; i < MAX_SUPPORT_POINTS; i++) {
204          /* 提取每个触摸点坐标, 上报Type B格式 */
205          u8 *buf = &rdbuf[i * tplen + offset]; /* 获取每个触摸点原始数据起始地址 */
206
207          /* 以第一个触摸点为例, 寄存器TOUCH1_XH(地址0X03), 各位描述如下:
208           * bit7:6 Event flag 0:按下 1:释放 2:接触 3:没有事件
209           * bit5:4 保留
210           * bit3:0 X轴触摸点的11~8位。
211           */
212          type = buf[0] >> 6; /* 获取触摸类型 */
213          if (type == TOUCH_EVENT_RESERVED)
214              continue;
```

```
C ft5426.c
215
216      /* 我们所使用的触摸屏和FT5X06是反过来的 */
217      x = ((buf[2] << 8) | buf[3]) & 0xffff;
218      y = ((buf[0] << 8) | buf[1]) & 0xffff;
219
220      /* 以第一个触摸点为例, 寄存器TOUCH1_YH(地址0X05), 各位描述如下:
221       * bit7:4 Touch ID 触摸ID, 表示是哪个触摸点
222       * bit3:0 Y轴触摸点的11~8位。
223       */
224      /* 上报数据 */
225      id = (buf[2] >> 4) & 0x0f;
226      down = type != TOUCH_EVENT_UP;
227
228      input_mt_slot(multidata->input, id); /* ABS_MT_SLOT */
229      input_mt_report_slot_state(multidata->input, MT_TOOL_FINGER, down);
230      /* ABS_MT_TRACKING_ID */
231
232      if (!down)
233          continue;
234
235      input_report_abs(multidata->input, ABS_MT_POSITION_X, x); /* ABS_MT_POSITION_X */
236      input_report_abs(multidata->input, ABS_MT_POSITION_Y, y); /* ABS_MT_POSITION_Y */
237
238      input_mt_report_pointer_emulation(multidata->input, true);
239      input_sync(multidata->input); /* SYN_REPORT */
240
241      fail:
242      return IRQ_HANDLED;
243
244
245  }
```


Linux Multi-Touch driver : 3.運行測試:編譯驅動程式和測試

1. 編譯驅動程式:在multitouch vscode工程修改Makefile，內容如下。

```
C ft5426.c  ●  M Makefile  x  C ft5426.mod.c
1  KERNELDIR := /home/jimmyyu/linux/IMX6ULL/linux/linux-imx-rel_imx_4.1.15_2.1.0_ga_alientek
2  CURRENT_PATH := $(shell pwd)
3
4  obj-m := ft5426.o
5
6  build: kernel_modules
7
8  kernel_modules:
9      $(MAKE) -C $(KERNELDIR) M=$(CURRENT_PATH) modules
10 clean:
11     $(MAKE) -C $(KERNELDIR) M=$(CURRENT_PATH) clean
```

2. 第4行設置obj-m變量的值為“ft5426.o”，將文件編譯成ft5426.ko，則make編譯成功後可生成“ft5426.ko”的驅動模塊文件。

```
jimmyyu@jimmyyu-virtual-machine:~/linux/IMX6ULL/Linux_Drivers/24_multitouch$ ls
ft5426.c  ft5426.ko  ft5426.mod.c  ft5426.mod.o  ft5426.o  Makefile  modules.order
Module.symvers  multitouch.code-workspace
```

3. 將編譯好的ft5426驅動模塊ft5426.ko和測試APP icm20608App拷貝到rootfs/lib/modules/4.1.15/文件夾下。

```
jimmyyu@jimmyyu-virtual-machine:~/linux/nfs/rootfs/lib/modules/4.1.15$ ls
atomicAPP  dtsled.ko  icm20608App  ledAPPZD  modules.alias  mutex.ko
atomic.ko  ft5426.ko  ledAPP      leddevice.ko  modules.dep  semaphore.ko
beep.ko    gpioled.ko  ledApp_z    leddriver.ko  modules.symbols  spinlockAPP
```

Linux Multi-Touch driver : 4.運行測試:觸摸數據輸出終端

4. 啟動開發板，加載驅動模塊: 執行 `modprobe ft5426.ko`, 顯示 `ft5426 probe` 表示設備與驅動匹配成功，顯示 `device id:edt-ft5426`, 內核節點位址

```
/lib/modules/4.1.15 # ls
atomic.ko      gpioled.ko      leddevice.ko    mutex.ko
atomicAPP      icm20608App     leddriver.ko    semaphore.ko
beep.ko        ledAPP          modules.alias   spinlockAPP
dtsled.ko      ledAPPZD        modules.dep
ft5426.ko      ledApp_z        modules.symbols
/lib/modules/4.1.15 # depmod
/lib/modules/4.1.15 # modprobe ft5426.ko
ft5x06_probe!
input: edt-ft5426 as /devices/platform/soc/2100000.aips-bus/21a4000/i2c/i2c-1/1-0038/input/input1
```

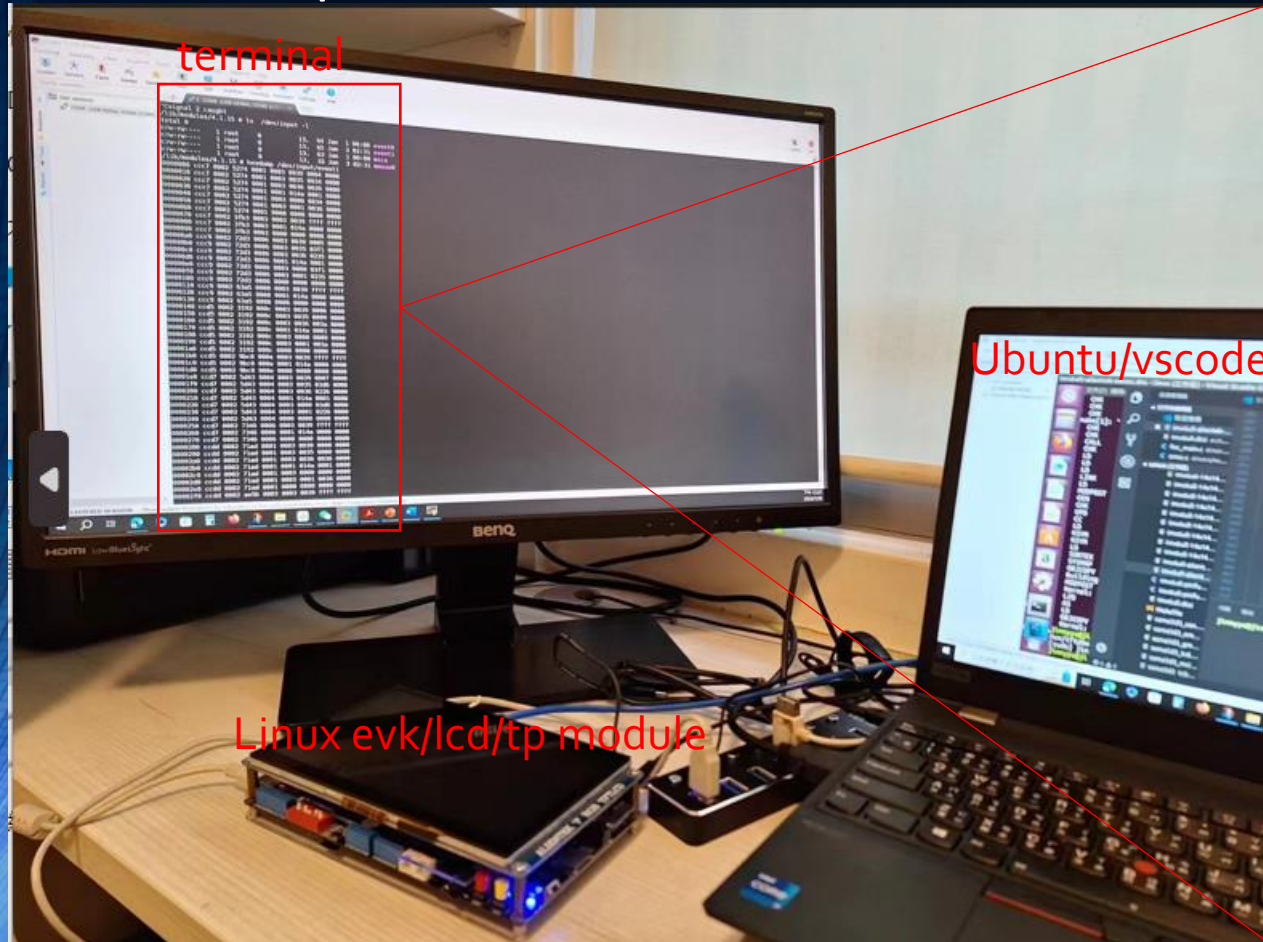
Ft5426對應的i2c2節點與reg位址如下與終端機顯示一致:

```
歡迎使用  imx6ull-alientek-emmc.dts  imx6ull.dtsi  fec_n
946         status = "disabled";
947     };
948
949     i2c2: i2c@021a4000 {
950         #address-cells = <1>;
951         #size-cells = <0>;
952         compatible = "fsl,imx6ul-i2c", "fsl,imx21-i2c";
953         reg = <0x021a4000 0x4000>;
954         interrupts = <GIC_SPI 37 IRQ_TYPE_LEVEL_HIGH>;
955         clocks = <&clks IMX6UL_CLK_I2C2>;
956         status = "disabled";
957     };
958
959     i2c3: i2c@021a8000 {
```

```
歡迎使用  imx6ull-alientek-emmc.dts  imx6ull.dtsi
285     };
286
287     &i2c2 {
288         clock_frequency = <100000>;
289         pinctrl-names = "default";
290         pinctrl-0 = <&pinctrl_i2c2>;
291         status = "okay";
292
293         /* jimmyyu 7吋FT5426 */
294         ft5426:ft5426@38{
295             compatible = "edt,edt-ft5426";
296             reg = <0x38>;
297             pinctrl-names = "default";
298             pinctrl-0 = <&pinctrl_tsc &pinctrl_tsc_reset>;
299             interrupt-parent = <&gpio1>;
300             interrupts = <9 0>;
301             reset-gpios = <&gpio5 9 GPIO_ACTIVE_LOW>;
302             interrupt-gpios = <&gpio1 9 GPIO_ACTIVE_LOW>;
303             status = "okay";
304         };
```


Linux Multi-Touch driver : 3.運行測試:ft5426觸摸數據輸出終端

5. 驅動加載成功後會生成event1設備，輸入“**hexdump /dev/input/event1**”指令後會在終端顯示觸摸數據。用手指觸摸LCD左上角，座標值(0x22,0x3f)，觸摸LCD右下角，座標值(0x3d3,0x226)，為合理的座標範圍。此LCD為(1024,600) RGB pixel。



```
2. COM9 (USB-SERIAL CH340 (COM x
/lib/modules/4.1.15 #
/lib/modules/4.1.15 # ls /dev/input -l
total 0
crw-rw---- 1 root 0 13, 64 Jan 1 00:00 event0
crw-rw---- 1 root 0 13, 65 Jan 3 03:02 event1
crw-rw---- 1 root 0 13, 63 Jan 1 00:00 mice
crw-rw---- 1 root 0 13, 32 Jan 3 03:02 mouse0
/lib/modules/4.1.15 # hexdump /dev/input/event1
00000000 cef1 0002 6ac3 0009 0003 0039 0016 0000
00000010 cef1 0002 6ac3 0009 0003 0035 0030 0000
00000020 cef1 0002 6ac3 0009 0003 0036 002f 0000
00000030 cef1 0002 6ac3 0009 0001 014a 0001 0000
00000040 cef1 0002 6ac3 0009 0003 0000 0030 0000
00000050 cef1 0002 6ac3 0009 0003 0001 002f 0000
00000060 cef1 0002 6ac3 0009 0000 0000 0000 0000
00000070 cef1 0002 ff62 000b 0003 0039 ffff ffff
00000080 cef1 0002 ff62 000b 0001 014a 0000 0000
00000090 cef1 0002 ff62 000b 0000 0000 0000 0000
000000a0 cef3 0002 2acc 0004 0003 0039 0017 0000
000000b0 cef3 0002 2acc 0004 0003 0035 0022 0000
000000c0 cef3 0002 2acc 0004 0003 0036 003f 0000
000000d0 cef3 0002 2acc 0004 0001 014a 0001 0000
000000e0 cef3 0002 2acc 0004 0003 0000 0022 0000
000000f0 cef3 0002 2acc 0004 0003 0001 003f 0000
00000100 cef3 0002 2acc 0004 0000 0000 0000 0000
00000110 cef3 0002 e768 0006 0003 0039 ffff ffff
00000120 cef3 0002 e768 0006 0001 014a 0000 0000
00000130 cef3 0002 e768 0006 0000 0000 0000 0000
00000140 cef5 0002 3420 0000 0003 0039 0018 0000
00000150 cef5 0002 3420 0000 0003 0035 03d3 0000
00000160 cef5 0002 3420 0000 0003 0036 0229 0000
00000170 cef5 0002 3420 0000 0001 014a 0001 0000
00000180 cef5 0002 3420 0000 0003 0000 03d3 0000
00000190 cef5 0002 3420 0000 0003 0001 0229 0000
000001a0 cef5 0002 3420 0000 0000 0000 0000 0000
000001b0 cef5 0002 086b 0005 0003 0039 ffff ffff
000001c0 cef5 0002 086b 0005 0001 014a 0000 0000
000001d0 cef5 0002 086b 0005 0000 0000 0000 0000
000001e0 cef6 0002 c037 0006 0003 0039 0010 0000
```

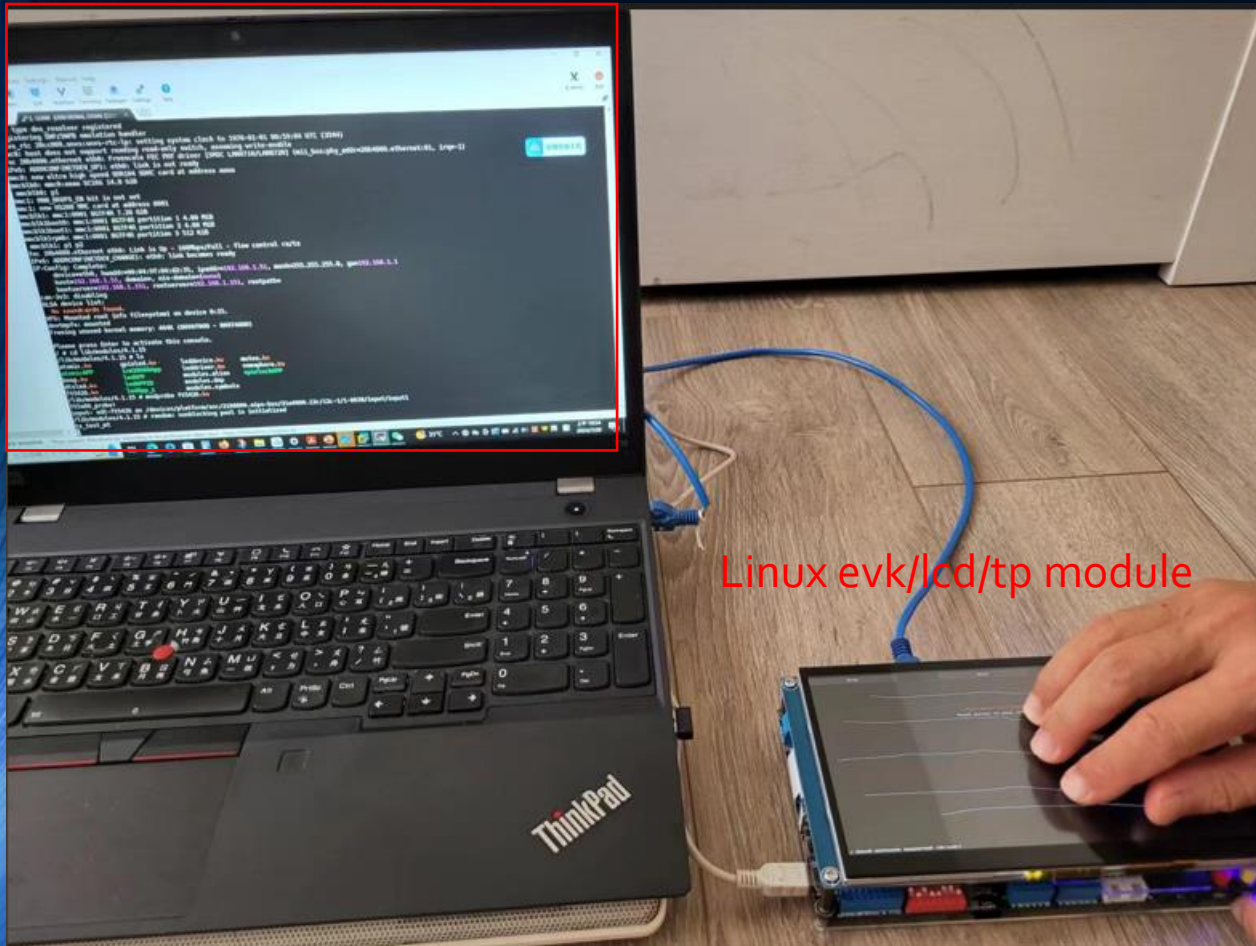
觸摸LCD左上角座標

觸摸LCD右下角座標

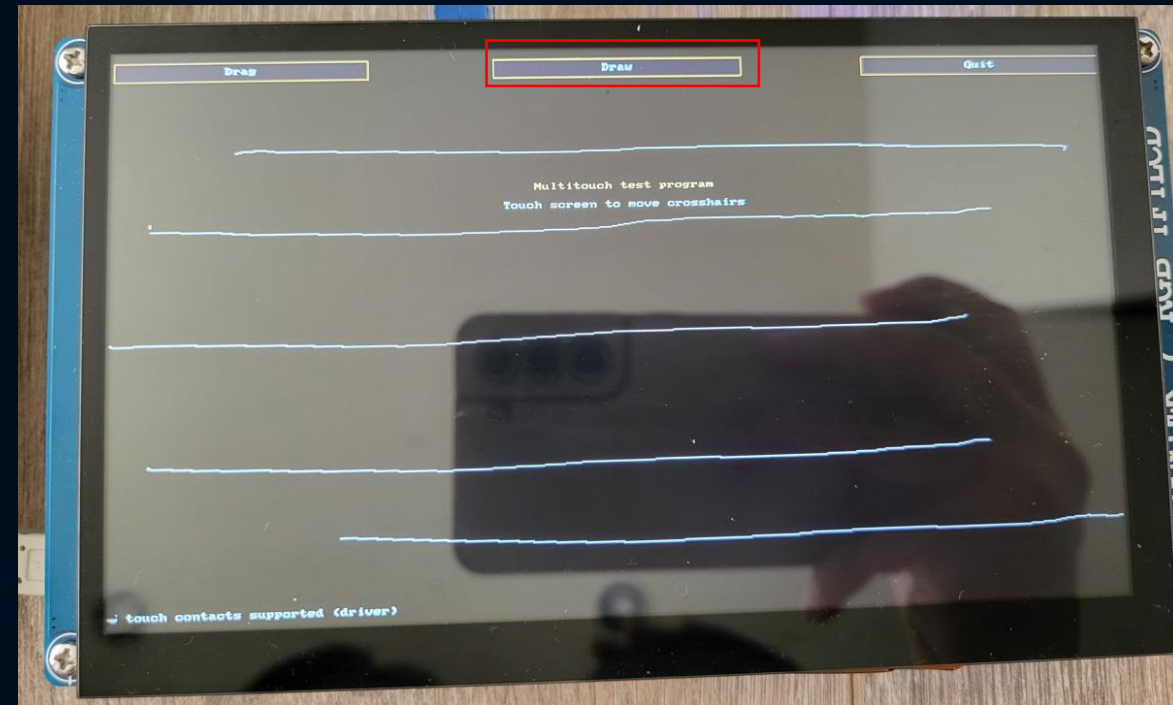
Linux Multi-Touch driver : 3.運行測試:tslib測試

6.tslib為開源的第三方庫，可用多點電容觸摸測試，先編譯安裝好tslib，再拷貝到開發板的根文件系統，啟動開發板，加載ft5426模塊，執行“ts_test_mt”命令，LCD會打開如左下畫面，有Drag/Draw/Quit三個功能按鈕，點擊Draw繪製功能，5個手指一起划過屏幕，屏幕上有5條線，說明5點電容觸摸正常。

terminal

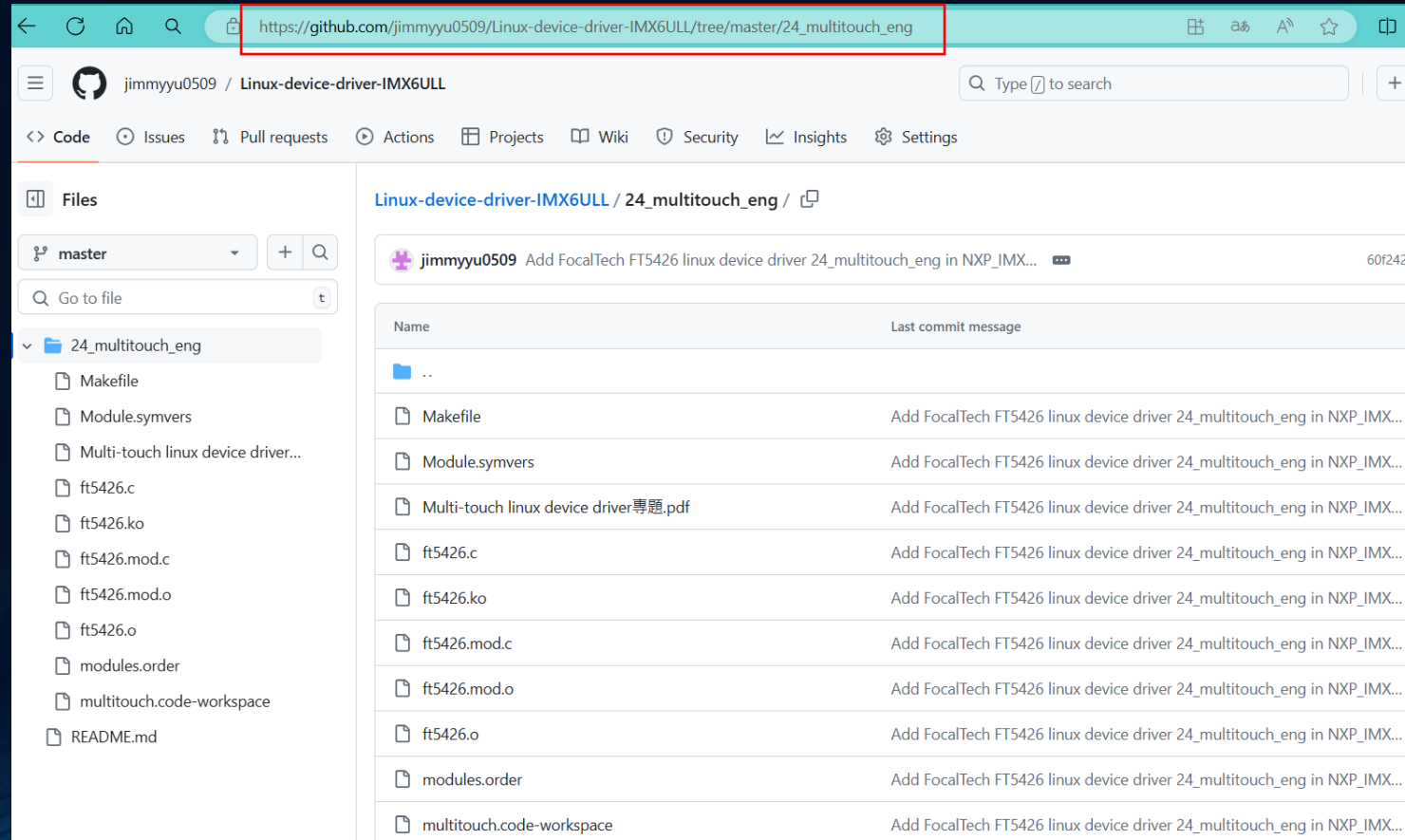


Linux evk/lcd/tp module



Note :

1. 本專題驅動採用NXP I.MX6ULL的SOC，Alientek的Alpha EVK開發板和其教程文檔搭建而成。
2. 本專題驅動github分享連結: https://github.com/jimmyyu0509/linux-device-driver-IMX6ULL/tree/master/24_multitouch_eng



The background is a deep blue gradient. On the left side, there is a faint, light blue grid pattern. On the right side, there are several curved, concentric lines that create a sense of depth and movement, resembling a tunnel or a stylized eye.

Thank You!