BCD Touch IC - Driver Programming Guide

V0.0.0.2

(1)說明

本文件提供了平版或手機系統與 BCD 觸控晶片連接的驅動程序建置方式說明。支援的晶片型號為 3852, 3890, 3891。

(2)通訊

觸發方式：負緣觸發(Falling Edge Trigger)

觸控點上報格式：

Byte	Name of Bytes	Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	ID	Touch ID Code	0xB1							
1	KEY_CODE	Key Code	NONE : 0x00 HOME : 0x01 BACK : 0x02 MENU: 0x03							
2	Points	Actual Points	0-> No point, 1~5-> Points							
3	X1_Low	X coordinate for the piont 1	X Position (bit7~0) of the point 1							
4	X1_High		X Position (bit15~8) of the point 1							
5	Y1_Low	Y coordinate for the piont 1	Y Position (bit7~0) of the point 1							
6	Y1_High		Y Position (bit15~8) of the point 1							
7	Point ID/Status	Report point ID & Status	ID(bit7~bit4) for point 1: 0~4 Status(bit3~0) for point1: 0x3->Touch,0x0->No Touch							
8	X2_Low	X coordinate for the piont 2	X Position (bit7~0) of the point 2							
9	X2_High		X Position (bit15~8) of the point 2							
10	Y2_Low	Y coordinate for the piont 2	Y Position (bit7~0) of the point 2							
11	Y2_High		Y Position (bit15~8) of the point 2							
12	Point ID/Status	Report point ID & Status	ID(bit7~bit4) for point 2: 0~4 Status(bit3~0) for point2: 0x3->Touch,0x0->No Touch							
13	X3_Low	X coordinate for the piont 3	X Position (bit7~0) of the point 3							
14	X3_High		X Position (bit15~8) of the point 3							
15	Y3_Low	Y coordinate for the piont 3	Y Position (bit7~0) of the point 3							
16	Y3_High		Y Position (bit15~8) of the point 3							
17	Point ID/Status	Report point ID & Status	ID(bit7~bit4) for point 3: 0~4 Status(bit3~0) for point3: 0x3->Touch,0x0->No Touch							
18	X4_Low	X coordinate for	X Position (bit7~0) of the point 4							

19	X4_High	the piont 4	X Position (bit15~8) of the point 4
20	Y4_Low	Y coordinate for the piont 4	Y Position (bit7~0) of the point 4
21	Y4_High		Y Position (bit15~8) of the point 4
22	Point ID/Status	Report point ID & Status	ID(bit7~bit4) for point 4: 0~4 Status(bit3~0) for point4: 0x3->Touch,0x0->No Touch
23	X5_Low	X coordinate for the piont 5	X Position (bit7~0) of the point 5
24	X5_High		X Position (bit15~8) of the point 5
25	Y5_Low	Y coordinate for the piont 5	Y Position (bit7~0) of the point 5
26	Y5_High		Y Position (bit15~8) of the point 5
27	Point ID/Status	Report point ID & Status	ID(bit7~bit4) for point 5: 0~4 Status(bit3~0) for point5: 0x3->Touch,0x0->No Touch
28	Reserved		
29	Reserved		
30	Reserved		

觸控晶片 I2C 命令表

Command	**BYTE 1**	**BYTE 2**	**BYTE 3**	**BYTE 4**
Sleep	0x0E	0x13	Reserved	0x00
Resume	0x0E	0x01	0x00	0x00
Disable	0x0E	0x03	0x00	0x00
Enable	0x0E	0x01	0x00	0x00
Reset	0x0E	0x12	0x00	0x00

(3)代碼說明

標頭檔(Header File)設定

驅動程序建立時需引用標頭檔 “tu_drvs.h”, 並於標頭檔中對以下參數進行設定：

參數名稱	MACRO	範例
驅動程序名稱	TU_I2C_NAME	"tu_drvs"
是否使用 Sleep/Suspend	CONFIG_PM	N/A
I2C 裝置位置	DEV_I2C_ADDRESS	0x5F(固定)
Tx 通道數	CHANNEL_X_SIZE	15
Rx 通道數	CHANNEL_Y_SIZE	10
最大觸摸指數	MAX_POINT_SIZE	5

Probe

```
static int tu_probe(.........)
{
        .........
        //設定裝置上報的解析度
        input_set_abs_params(input_dev, ABS_MT_POSITION_X, 0, AA_X_SIZE, 0, 0);
        input_set_abs_params(input_dev, ABS_MT_POSITION_Y, 0, AA_Y_SIZE, 0, 0);
    .........
}
```

IRQ

```
static void tu_i2c_work(struct work_struct *work)
{
    .........
    //讀取上報的封包數據
    ret = i2c_master_recv(tu->client, read_buf, BUF_SIZE  )

    //偵測數據為報點或者按鍵
        if (read_buf[TU_RMOD] == 0xb1)                  //報點(Point Report)
        {
                touchcnt = read_buf[TU_POINTS];

                if( touchcnt==0 )
                {
                        input_report_key(tu->dev, BTN_TOUCH, 0);
                        input_report_abs(tu->dev, ABS_MT_TOUCH_MAJOR, 0);
                        input_mt_sync(tu->dev);
                }
                else
                {
                        idx_x_low = TU_1_POS_X_LOW;
                        idx_x_hi  = TU_1_POS_X_HI;
                        idx_y_low = TU_1_POS_Y_LOW;
                        idx_y_hi  = TU_1_POS_Y_HI;
                        idx_id_st = TU_1_ID_STATUS;

                        for( i=0; i<touchcnt; i++ )
                        {
                                tu->x = COORD_INTERPRET(read_buf[idx_x_hi],
read_buf[idx_x_low]);
                                tu->y =
(COORD_INTERPRET(read_buf[idx_y_hi],read_buf[idx_y_low]));

                                tu->w = (read_buf[idx_id_st]&0x0f);
                                tu->id = (read_buf[idx_id_st]>>4)&0x0f;

                                tu_report(tu);

                                idx_x_low  += MAX_POINT_SIZE;
                                idx_x_hi   += MAX_POINT_SIZE;
                                idx_y_low  += MAX_POINT_SIZE;
                                idx_y_hi   += MAX_POINT_SIZE;
                                idx_id_st  += MAX_POINT_SIZE;
                        }
                }
        }
        else if (read_buf[TU_RMOD] == 0xb2) //報按鍵
        {
                switch (read_buf[TU_KEY_CODE])
                {
                        case TOUCH_KEY_HOME:
                                input_event(tu->dev, EV_KEY, KEY_HOME, !!
read_buf[TU_KEY_CODE]);
                                prev_key = KEY_HOME;
                                break;
```

```
                        case TOUCH_KEY_BACK:
                                input_event(tu->dev, EV_KEY, KEY_BACK, !!
read_buf[TU_KEY_CODE]);
                                prev_key = KEY_BACK;
                                break;
                        case TOUCH_KEY_MENU:
                                input_event(tu->dev, EV_KEY, KEY_MENU, !!
read_buf[TU_KEY_CODE]);
                                prev_key = KEY_MENU;
                                break;
                        case TOUCH_KEY_REL:
                                input_event(tu->dev, EV_KEY, prev_key, !!
read_buf[TU_KEY_CODE]);
                                break;
                        case TOUCH_KEY_VOL_UP:
                                input_event(tu->dev, EV_KEY, KEY_VOLUMEUP, !!
read_buf[TU_KEY_CODE]);
                                prev_key = KEY_VOLUMEUP;
                                break;
                        case TOUCH_KEY_VOL_DOWN:
                                input_event(tu->dev, EV_KEY, KEY_VOLUMEDOWN, !!
read_buf[TU_KEY_CODE]);
                                prev_key = KEY_VOLUMEDOWN;
                                break;
                        case TOUCH_KEY_CALL:
                                input_event(tu->dev, EV_KEY, KEY_SEND, !!
read_buf[TU_KEY_CODE]);
                                prev_key = KEY_SEND;
                                break;
                        default:
                                dev_dbg(&tu->client->dev, "Unknown Android Key %02x",
read_buf[TU_KEY_CODE]);
                                break;
                }

        }

        input_sync(tu->dev);
        .........
}
```

Sleep(suspend)

```
static int tu_suspend(.........)
{
        .........

   //傳送 4 Byte 命令至觸控晶片, 使晶片進入休眠
   ret = i2c_smbus_write_i2c_block_data(touch_i2c_client, 0, 4, command_list[0]);
      .........
        return 0;
}
```

Resume

```
static int tu_resume(.........)
{
        .........
  //傳送 4 Byte 命令至觸控晶片, 使晶片回復運作
  ret = i2c_smbus_write_i2c_block_data(touch_i2c_client, 0, 4, command_list[1]);
      .........
}
```