ILITEK
I Love Innovation

Linux I²C
Touch Device Driver

Version: V0_0_0_6
Document: ILITEK_LINUX_I2C_DRIVER.pdf

ILI TECHNOLOGY CORP.
8F, No.1, Taiyuan 2nd-St., Jhubei City, Hsinchu County 302, Taiwan, R.O.C.
Tel.886-3-5600099; Fax.886-3-5600055
http://www.ilitek.com

Content

Sections ***Pages***

1. Introduction

This document introduces how the touch panel driver with i2c works on different platforms, the guide of porting, and some configurations may be set up before users porting it on its own device. This version of driver integrates the supports on several platforms such as Qualcomm, MTK, Rockchip, which are all defined on the header file ***ilitek_ts.h***. Users can choose one of them by replacing the macro's name depending on their platform, as figer 1. Note that users should be aware of the differences in their Makefile while they are modifying the define.

```
00065: #define ILITEK_PLAT_QCOM                1
00066: #define ILITEK_PLAT_MTK                 2
00067: #define ILITEK_PLAT_ROCKCHIP            3
00068: #define ILITEK_PLAT_ALLWIN              4
00069: #define ILITEK_PLAT_AMLOGIC             5
00070:
00071: #define ILITEK_PLAT                     ILITEK_PLAT_QCOM
00072:
```

Figer 1. The definiation of platform in ***ilitek_ts.h***

The type of support Touch IC	ILI230X、ILI231X、ILI251X
I2C Slave address (7 bits)	0x41
Upgrade firmware automatically at boot stage	It shoule be included the header file named "ilitek_fw.h", or used bin file
Support platforms	Qualcomm, Rockchip, MTK (with DTS), Allwinner, Amlogic. (without the list, users might port the driver with the define of QCOM on their platforms.)

2. File description

This section describes the purpose of each files in the driver.

2.1 ilitek_ts.h

Provide customer modifications . It includes common definations, declation, and macros for each C files. Most of macros used to enable some functions or features are located on the header file.

2.2 ilitek_common.h

It includes common definations, declation, and macros for each C files. Most of macros used to enable some functions or features are located on the header file.

2.3 ilitek_platform_init.c

It places board information, registering i2c device driver with kernel and callback probe function once i2c device been detected.

2.4 ilitek_main.c

Most of main features with Touch IC and other settings related to the input events and suspend/resume are implmeneted in this C file.

2.5 ilitek_update.c

The C file deals with the process of upgrade firmware.

2.6 ilitek_tool.c

It is mainly used to communicate with user space by its device nodes for debugging, commanding, or looking up information from touch ICs.

2.7 ilitek_protocol.c

It is extern functions and ilitek protocol.

3. Porting Guide

This section guides users how they use the driver poring on their own platforms. Note that the scenarios described on below might be different based on kernel version, board configurations or other specific requirements. The following is just common steps which are all verified.

1. Move the directory of this driver into the path ***kernel/drivers/input/touchscreen/.*** It might be different based on platform's request.
2. Add the path of our driver into kernel Makefile, which locates under ***touchscreen/.***

```
obj-$(CONFIG_TOUCHSCREEN_TPS6507X)  += tps6507x-ts.o
obj-$(CONFIG_TOUCHSCREEN_VTL_CT36X) += vtl_ts/
obj-y += ilitek_lim/
```

Once it has been done, the driver will be compiled by kernel as built-in. It can also be defined in Kconfig configured by menuconfig if users prefer to do it.

3. The next following step is the most important when it comes to port our driver on a platform. The first thing users should make sure is what the method is used to configure I2C bus and gpios on their platform certainly. In general, there are divided into two ways where allow users to modifiy several hardware settings. If users's platform is set up by board configuration, the borad file is then nomally placed on the path *"linux/arch/arm/mach-xxx/board-xxx.c".* The following figure, for example, shows that how we did configuration with i2c bus in kernel on customer's platform:

```
00986: static struct i2c_board_info __initdata i2c_tpd={
00987:     I2C_BOARD_INFO("ilitek_i2c", 0x41),
00988:     //.platform_data     = &ilitek_pdata,
00989:                          名称        地址
00990: };

02243:     i2c_register_board_info(2, &i2c_tpd, 1);        总线号
02244:
```

On the other hand, if users's platform is configrued by DTS, they may then find the dts file that is normally located at *"linux/arch/arm/boot/dts/xxx-xxx.dts"* to write ther own settings. More importantly, Users must know which of I2C buses number appied on and INT/Reset pin used on TP device on their platform before adding the information in .dts.

```
ilitek@41 {
    compatible = "tchip,ilitek";
    reg = <0x41>;
    interrupt-parent = <&msm_gpio>;
    interrupts = <13 0x0>;
    vdd-supply = <&pm8916_l17>;
    vcc_i2c-supply = <&pm8916_l6>;
    ilitek,irq-gpio = <&msm_gpio 13 0x0>;
    ilitek,reset-gpio = <&msm_gpio 12 0x0>;
    ilitek,vbus = "vcc_i2c";
    ilitek,vdd = "vdd";
    ilitek,name = "ilitek_i2c";
};
```

4. Description of macros and defines in head file

4.1 Support which of platforms

```
#define ILITEK_PLAT_QCOM        1
#define ILITEK_PLAT_MTK         2
#define ILITEK_PLAT_ROCKCHIP    3
#define ILITEK_PLAT_ALLWIN      4
#define ILITEK_PLAT_AMLOGIC     5
#define ILITEK_PLAT             ILITEK_PLAT_QCOM
```

As said before, these macros are confirmed to the driver which of probe functions it should run on. If users could not find their platform on the list, they may then define ILITEK_PLAT_QCOM as alternative.

4.2 Tool with debug

```
#define ILITEK_TOOL
```

It is enabled for the tool with debug.

4.3 Firmware tunning message

```
#define ILITEK_TUNING_MESSAGE
```

It is enabled for the firmware dumping its debug messeages.

4.4 ESD protection

```
#define ILITEK_ESD_PROTECTION
```

It is used to protect ESD happening if it's enabled.

4.5 Mutiple-touch protocol in Linux

```
#define ILITEK_TOUCH_PROTOCOL_B
```

It tells Linux kernel to either use protocol B, or protocol A to interpret the packet of finger report from firmware.

4.6 LCM resolution

```
#define ILITEK_USE_LCM_RESOLUTION
```

It tells driver whether to use LCM resolution while registering input subsystem.

4.7 Key with coordination

```
#define ILITEK_REPORT_KEY_WITH_COORDINATE
```

Some of platform (Ex. MTK) may require the key reported from driver should match the coordination. This featue depends on the behavior of system.

4.8 Rotate axises

#define ILITEK_ROTATE_FLAG 0

It will rotate the axis of X and Y with finger report, just setting it as non-zero it would be enabled.

4.11 Revert axis of X

#define ILITEK_REVERT_X 0

If it is set as non-zero, the value of axis of X will be changed from maximum to minimum, or from minimum to maximum.

4.12 Revert axis of Y

#define ILITEK_REVERT_Y 0

If it is set as non-zero, the value of axis of Y will be changed from maximum to minimum, or from minimum to maximum.

4.13 Regulator power

#define ILITEK_ENABLE_REGULATOR_POWER_ON

If it is enabled, the driver uses the function of regulator provided by kernel to control TP's voltage.

4.14 GPIO numbers

#define ILITEK_GET_GPIO_NUM

Enabling the macros the driver calls kerne's macro (of_get_named_gpio) to get GPIO numbers defined on .dts and apply them to the two variables (ILITEK_RESET_GPIO, ILITEK_IRQ_GPIO) as correct gpio number. If it is disabled, users should then fill the gpio number in the variables by themselves.

4.15 The behavior of gesture wake up

#define ILITEK_CLICK_WAKEUP	0
#define ILITEK_DOUBLE_CLICK_WAKEUP	1
#define ILITEK_GESTURE_WAKEUP	2
#define ILITEK_GESTURE	ILITEK_CLICK_WAKEUP

1. Sing-click wake up (by driver), which is mainly used to SA Large.
2. Double-click wake up (by drive), which is mainly used to SA Large
3. Gesture wake up (by firmware), which is mainly used to SA Small
4. Acoording to the above define, it tells driver that which of behaviors should react when it comes to gesture wake up.

4.16 Upgrade firmware at boot stage

```
#define ILITEK_UPDATE_FW
```

The process of upgrade firmware will run on booting time if it is enabled. Since the function requires a particular header file to be included, it might be disabled before every thing has been tuned in priority.

4.17 Check INT status

```
#define ILI_UPDATE_BY_CHECK_INT
```

Some of specific Touch ICs (Ex. ILI2302/ILI2312) have the ability to speed up the process of upgrade firmware. Enabling the macro the driver will check the status of INT and sending next data if INT is being polled low or high (means that touch IC is ready or not).

4.18 Upgrade firmware with bin file

```
#define ILITEK_UPGRADE_WITH_BIN                0
```

The file we use to upgrade firmware has two formats, .hex and .bin. Once users enable the macros, they should specify the name and the path where locates the .bin file.

4.19 LCM reslution

```
#define TOUCH_SCREEN_X_MAX          (1080)   //LCD_WIDTH
#define TOUCH_SCREEN_Y_MAX          (1920)   //LCD_HEIGHT
```

If users have enabled the macro *ILITEK_USE_LCM_RESOLUTION* on their system, or enabled *ILITEK_USE_MTK_INPUT_DEV* on MTK platform , those two macros should be enabled and set them correctly.

4.20 Input device for MTK

```
#define ILITEK_USE_MTK_INPUT_DEV
```

Since on MTK platforms they usually use its own input structure to regitster with kernel, it will use *tpd->dev* to register input subsystem in kernel.

4.21 The level of debug message

```
#define ILITEK_ERR_LOG_LEVEL          (1)
#define ILITEK_INFO_LOG_LEVEL         (3)
#define ILITEK_DEBUG_LOG_LEVEL        (4)
#define ILITEK_DEFAULT_LOG_LEVEL      (3)
```

The level represents the number as ERR, INFOR and DEBUG, respectively. Users can adjust the number to see the corresponding debug message.

5. Description of main functions

This section introduces important functions related to platform's settings and some tips are also included.

5.1 ilitek_power_on

This function will be called when ILITEK_ENABLE_REGULATOR_POWER_ON is enabled. Note that the member *vdd* and *vdd_i2c* of structure *ilitek_data* should be set according to the requirement of platform.

5.2 ilitek_get_gpio_num

This function gets the gpio of INT and RESET from the two macros, ILITEK_IRQ_GPIO and ILITEK_RESET_GPIO, which will be appied automatically if platforms use .dts as configuration.

5.3 ilitek_request_gpio

This function requests gpio number that users have been set on kernel. The output of reset will be highe and irq as input when the request is accepted. The operations of gpio might be different with platforms.

5.4 ilitek_reset

The duration of delay of polling reset pin from low to high needs larger than the initial time of touch IC. The function of *tpd_gpio_output (or other customered function)* may cause this action unexpection, so users should ensure that the haward reset works correctly.

5.5 ilitek_read_tp_info

1. Set ilitek_repeat_start to false when it is judged that the IC is ILI2511.

2. For large-size ICs, when the 0xC0 command reads 0x55 (that is, BL mode), the flag of the forced upgrade will be set, forcing the upgrade flag:ilitek_data->force_update.

3. The keyinfo of the button information is set to a size of 10, and when the number of keys is greater than 10, the size of the member keyinfo needs to be modified in the struct ilitek_ts_data structure.

5.6 ilitek_request_irq(void)

Users should be aware of the correction of irq gpio number given by kernel.

6. Function description

Power on upgrade function

1. Turn on the ILITEK_UPDATE_FW macro contorl
2. If you use ili file, you need to provide ili file to use this function.
3. Whether to upgrade the judgment method

```
tp_log_info("ilitek df_startaddr=0x%X, df_endaddr=0x%X, df_checksum=0x%X, df_len = 0x%d\n", df_startaddr, df_endaddr, df_checksum, df_l
if (!(ilitek_data->force_update)) {
    for (i = 0; i < 8; i++) {
        tp_log_info("ilitek_data.firmware_ver[%d] = %d, firmware_ver[%d] = %d\n", i, ilitek_data->firmware_ver[i], i, firmware_ver[i]);
        if (firmware_ver[i] < ilitek_data->firmware_ver[i]) {
            i = 8;
            break;
        }
        if (firmware_ver[i] > ilitek_data->firmware_ver[i]) {
            break;
        }
    }
    if (i >= 8) {
        if (!ilitek_data->ic_2120) {
            tp_log_info("firmware version is older so not upgrade\n");
        }
        else {
            tp_log_info("firmware version is same so not upgrade\n");
            if (ILITEK_UPGRADE_WITH_BIN) {
                if (CTPM_FW_BIN) {
                    vfree(CTPM_FW_BIN);
                    CTPM_FW_BIN = NULL;
                }
            }
        }
        return 1;
    }
} ? end if !(ilitek_data->force_... ?
```

4. Upgrade flow selection

```
if ((ilitek_data->mcu_ver[0] == 0x11 || ilitek_data->mcu_ver[0] == 0x10) && ilitek_data->mcu_ver[1] == 0x25) {
    df_startaddr = 0xF000;
    ret = ilitek_upgrade_2511(df_startaddr, df_endaddr, ap_startaddr, ap_endaddr, CTPM_FW);
    if (ret < 0) {
        tp_log_err("ilitek_upgrade_2511 err ret = %d\n", ret);
        //goto Retry;
    }
} else {
    df_startaddr = 0x1F000;
    if (df_startaddr < df_endaddr) {
        ilitek_data->has_df = true;
    } else {
        ilitek_data->has_df = false;
    }
    ret = ilitek_upgrade_2302or2312(df_startaddr, df_endaddr, df_checksum, ap_startaddr, ap_endaddr, ap_checksum, CTPM_FW);
    if (ret < 0) {
        tp_log_err("ilitek_upgrade_2302or2312 err ret = %d\n", ret);
        //goto Retry;
    }
}
```

5. Gesture function

   #define ILITEK_CLICK_WAKEUP 0 // Single touch on wake up

   #define ILITEK_DOUBLE_CLICK_WAKEUP 1 //Double clock on wake up

   #define ILITEK_GESTURE_WAKEUP 2 //It FW set wake up

   Turn on this feature to ensure that the TP is not powered off during sleep.

   Double click to wake up the parameter description:

   #define DOUBLE_CLICK_DISTANCE 1000 // Maximum distance of double click coordinates.

   #define DOUBLE_CLICK_ONE_CLICK_USED_TIME 800 // The longest time spent on frist click

   #define DOUBLE_CLICK_NO_TOUCH_TIME 1000 // Interval between the second click.

   #define DOUBLE_CLICK_TOTAL_USED_TIME (DOUBLE_CLICK_NO_TOUCH_TIME + (DOUBLE_CLICK_ONE_CLICK_USED_TIME * 2)) // Double click total time

6. Glove function

   6.1 cat /proc/ilitek/ enable_glove_mode

   Enable Glove mode.

   6.2 cat /proc/ilitek/ disable_glove_mode

   Disable Glove mode.

7. Troubleshooting

7.1 Probe function will not be called by kernel

Users should look up the board configuration (located at under *match-xxx/* or *dts/*) to ensure that the settings of I2C Bus number and gpio and their name are correct. For example, on board file, users might check the name of *ILITEK_TS_NAME* that is the same as the name of I2C device registered on kernel. On the other hand, users might also check whether the name of **comptiable** in structure *ilitek_touch_match_table* is the same as the name of **compatiable** in .dts, both of them must be matched.

7.2 I2C communication does not work

First of all, users have to make sure that the I2C bus name and slave address are all correct. Secondly, providing voltage to Touch IC in correct is also a main point to be checked. Finally, if it does not still work, users could use LA to catch the wave of I2C to see what happends on it during ommunication with touch IC

Revision History

Version No.	Date	Page	Description
0.0.1	2011/03/07	All	Firstly release
0.0.2	2011/05/12	3	Modified driver file name.
0.0.3	2011/09/30	3	Modified version id
0.0.4	2012/11/26	3	Method of adding idc files
0_0_0_5	2017/07/14	15	Modified driver structure
0_0_0_6	2019/5/7	13	Remove ILI2120 function add glove mode control.