

TDDI Driver for Android/Linux

Version : V2.5

ILI TECHNOLOGY CORP.

10F, No.1, Taiyuan 2St., Jhubei City, Hsinchu County 302, Taiwan, R.O.C.

Tel.886-3-5600099; Fax.886-3-5600055

<http://www.ilitek.com>

Table of Contents

Section	Page
前言.....	4
驅動架構.....	5
驅動移植.....	7
相關功能說明.....	13

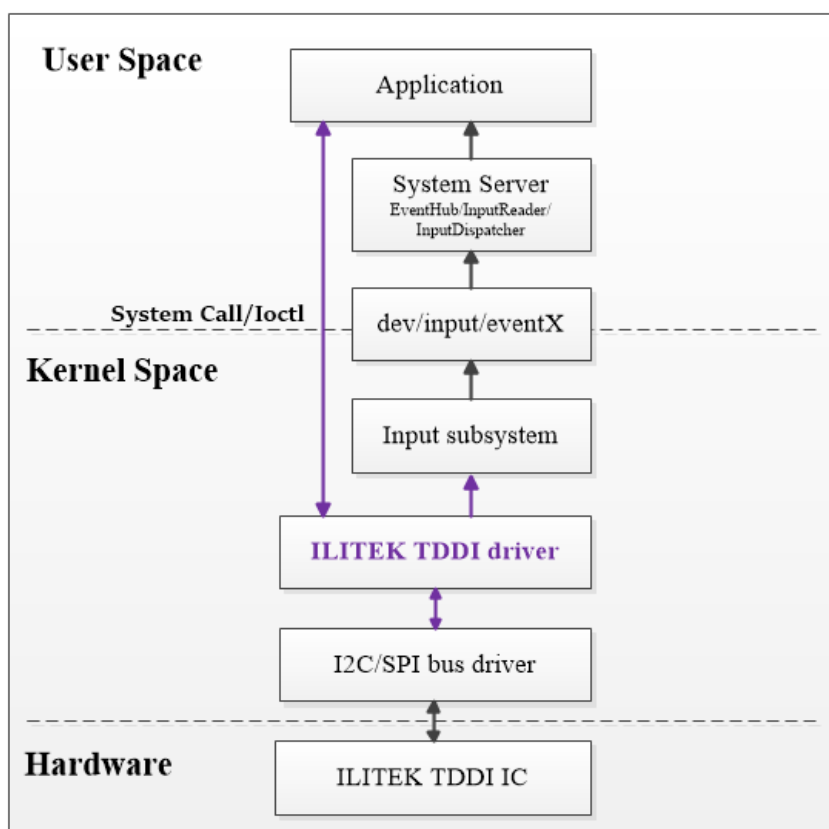
Revision History

Modifier	Description	Date	Version
Ryder Hsu	Release draft version	2019/2/26	0.2
Dicky Chiang	Add Debug 、Probe flow and IC configs	2019/3/13	2.0
Dicky Chiang	Remove DDI Reg example of description	2019/3/19	2.1
Dicky Chiang	Add description of dump iram data	2019/5/6	2.2
Dicky Chiang	Corrected parameters in R/W TP register	2019/5/6	2.3
Dicky Chiang	Add introduction of macro definition	2019/9/6	2.4
Dicky Chiang	Add pinctrl and module compatibility	2019/11/15	2.5

前言

該文件針對 V2.x 驅動做說明，包括驅動架構、驅動移植、功能調試等。

驅動 V2.x 是開發用於 Android/Linux 系統，透過 Linux 內核的輸入子系統(Input Subsystem)上報觸控的事件給上層，上層透過內核的節點和 IOCTL 與驅動溝通，所以開發上只需專注在內核階層。



[圖一]Android 系統架構圖

V2.x 驅動兼容多種平台和 ILITEK TDDI IC：

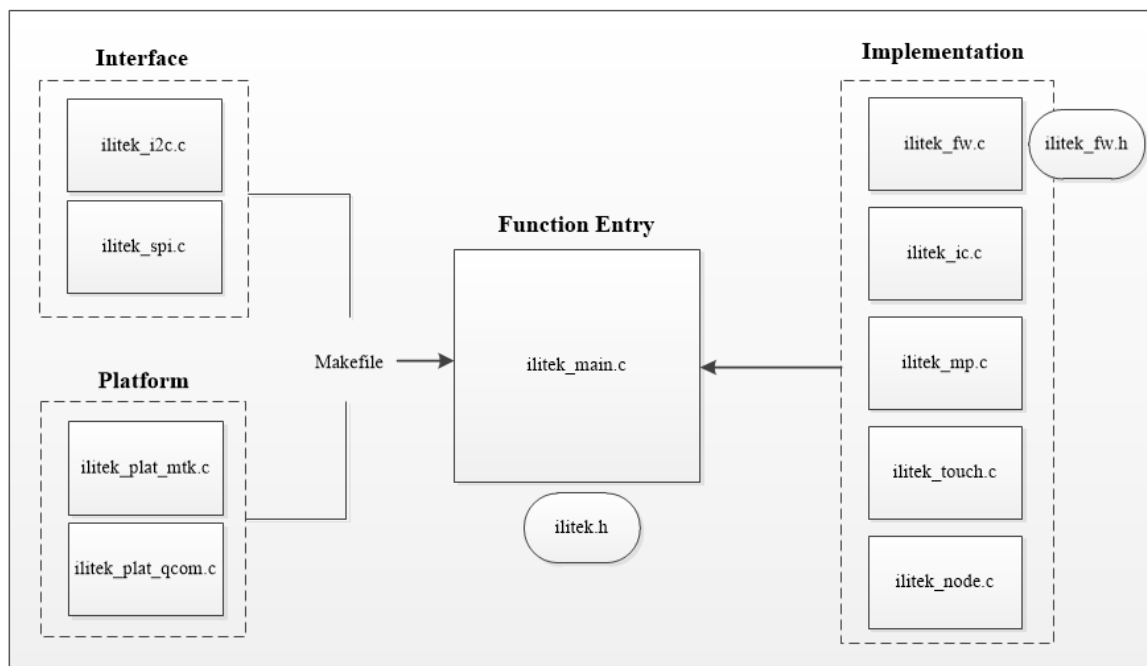
Driver Ver.	Protocol Ver.	Platform	ILITEK IC
2.0.5.0	5.0 - 5.6	MTK Qualcomm SPRD	ILI9881F ILI9881H ILI7807G

[表一]驅動兼容的平台及 IC

驅動架構

驅動 V2.x 的設計理念：

- 統一管理事件入口，這些事件的入口函式會受到互斥鎖(mutex)及原子變量(atomic)的保護，以解決同步競態的問題，保證各個事件的獨立運行。
- 功能的實現歸納成五大類：IC、Touch、FW、MP、Node，功能區分更加明確及集中，易於維護和追蹤。
- 將通訊介面、IC 型別、及平台的差異，集中在少數檔案做設定及管理，增加程式的可讀性及維護性。



[圖二]驅動檔案架構

驅動各檔案內容簡介：

File	Description
ilitek_plat_mtk.c	MTK 平台初始化，設置 GPIO、ISR、Suspend/Resume 等
ilitek_plat_qcom.c	QCOM 平台初始化，設置 GPIO、ISR、Suspend/Resume 等
ilitek_i2c.c	I2C 讀寫
ilitek_spi.c	SPI 讀寫
ilitek_main.c	事件入口管理，FW 更新、產測、報點、workqueue 和休眠控制
ilitek_ic.c	IC 相關操作，包含取得 Chip ID、ICE Mode、Soft Reset 等
ilitek_flash.c	處理有 Flash 的 FW 更新
ilitek_hostdl.c	處理沒有 Flash 的 FW 更新
ilitek_mp.c	整機產測，需搭配正確的 FW 及 ini 檔
ilitek_touch.c	報點，載入 Gesture Code 及 MP Code
ilitek_node.c	建立與上層溝通的節點、IOCTL
ilitek.h	所有檔案共用的頭文件，定義 Linux 頭文件、IC 及 Flash 位址、功能開關、Debug 訊息、驅動版本等
ilitek_fw.h	定義各種模組的路徑和名稱，用於開機燒錄的 ILI、一般更新用的 HEX 以及產測設定檔 ini。
Makefile	選擇要編譯的通訊介面及平台（僅支援 built-in，如要用 module 需自行修改）。

[表二]驅動檔案說明

驅動移植

1. 編譯驅動

※ 以下範例的路徑皆是以 MTK X20 作參考

1.1 將驅動整個目錄置放於特定路徑，置放的路徑為：

`aosp/kernel-3.18/drivers/input/touchscreen/mediatek`

1.2 在 Kernel 的 *Kconfig* 及 *Makefile* 加入編譯選項，並在 *defconfig* 打開選項

Kconfig path : `aosp/kernel-3.18/drivers/input/touchscreen/mediatek/Kconfig`

```
config TOUCHSCREEN_MTK_ILI9881
    bool "ILI9881 for Mediatek package"
    default n
    help
        Say Y here if you have ILI9881 touch panel.

        If unsure, say N.

        To compile this dirver as a module, choose M here: the
        module will be called.
```

Makefile path : `aosp/kernel-3.18/drivers/input/touchscreen/mediatek/Makefile`

```
obj-$(CONFIG_TOUCHSCREEN_MTK_ILI9881) += new_tddi/
```

deconfig path : `aosp/kernel-3.18/arch/arm64/configs/ amt6797_64_open_defconfig`

```
CONFIG_TOUCHSCREEN_MTK_ILI9881=y
```

1.3 設定驅動的 *Makefile*，按照使用平台與 IC 通訊介面進行選擇性編譯

```
BUILD_INFAE := i2c
BUILD_PLATFORM := mtk
```

2. 在驅動設定通訊介面

在標頭檔(ilitek.h)設置通訊介面的設定：

```
#define TDDI_INTERFACE      BUS_SPI /* BUS_I2C(0x18) or BUS_SPI(0x1C) */
```

3. DTS 設置

DTS(Device Tree Source)是用於配置硬體資訊的重要檔案，以 I2C 裝置為例，需在 DTS 配置 I2C 的 Slave ID address 及 GPIO 設定，才能讓驅動與硬體正常通訊。

要讓驅動與對應的 DTS 連結，要確認 DTS 的名稱與驅動註冊的名稱相同。驅動在平台所屬的.c 檔(ilitek_plat_mtk.c 或 ilitek_plat_qcom.c)有 DTS 名稱的定義：

```
#define DTS_OF_NAME      "mediatek, cap_touch"
```

不同平台或不同 code base 在 DTS 設置上可能會有差異，但仍可依照這個寫法再做些微修改：

```
&i2c4 {
    #address-cells = <1>;
    #size-cells = <0>;
    clock-frequency = <400000>;
    mediatek,use-open-drain;
    cap_touch@41 {
        compatible = "mediatek, cap_touch";
        reg = <0x41>;
        status = "okay";
    };
};
```

```
&spi1 {
    #address-cells = <1>;
    #size-cells = <0>;
    ilitek@1 {
        compatible = "mediatek, cap_touch";
        reg = <0>;
        spi-max-frequency = <10000000>;
        status = "okay";
    };
};
```

通過上述步驟的設定，基本上可以完成初步的移植，可以從 kernel log 確認結果。如下 log 顯示驅動有被成功加載，DTS 與通訊介面有設定完成，並且正確讀到 CHIP ID：

```
ILITEK: (tpd_local_init, 407): TPD init device driver
ILITEK: (ilitek_tddi_dev_init, 987): TP Interface: SPI
ILITEK: (ilitek_spi_probe, 682): ilitek spi probe
ILITEK: (core_spi_setup, 656): spi clock = 10000000
ILITEK: (core_spi_setup, 672): name = cap_touch, bus_num = 1, cs = 0, mode = 0, speed = 10000000
ILITEK: (ilitek_plat_probe, 365): platform probe
ILITEK: (ilitek_plat_gpio_register, 176): TP INT: 1
ILITEK: (ilitek_plat_gpio_register, 177): TP RESET: 0
ILITEK: (ilitek_tddi_init, 864): driver version = 2.0.5.0.191115
ILITEK: (ilitek_tddi_reset_ctrl, 728): TP HW RST
ILITEK: (ilitek_plat_tp_reset, 36): edge delay = 5
ILITEK: (ilitek_tddi_ic_spi_speed_ctrl, 664): Enable spi speed up
ILITEK: (ilitek_tddi_ic_get_info, 1144): CHIP: PID = 78071b00, ID = 7807, TYPE = 1b, UER = 0, OTP = 0, ANA = 1
```


4. Probe 流程

v2.x 版本為了把通訊界面和平台做完整切割，不同於傳統的把 probe 放在同一個函式，會先通過平台的設定，決定使用哪一個界面，再去調用對相應的界面做驅動初始化。所以可以看到主結構體都是在 **ilitek_i2c.c** 或者 **ilitek_spi.c** 做內存配置和設定初始值，最後再返回平台設定，然後開始做功能性的初始化。

```
static int ilitek_spi_probe(struct spi_device *spi)
{
    struct touch_bus_info *info =
        container_of(to_spi_driver(spi->dev.driver),
            struct touch_bus_info, bus_driver);

    ipio_info("ilitek spi probe\n");

    if (!spi) {
        ipio_err("spi device is NULL\n");
        return -ENODEV;
    }

    iddev = devm_kzalloc(&spi->dev, sizeof(struct ilitek_tddi_dev), GFP_KERNEL);
    if (ERR_ALLOC_MEM(iddev)) {
        ipio_err("Failed to allocate iddev memory, %ld\n", PTR_ERR(iddev));
        return -ENOMEM;
    }

    iddev->i2c = NULL;
    iddev->spi = spi;
    iddev->dev = &spi->dev;
    iddev->hwif = info->hwif;
    iddev->phys = "SPI";

    iddev->write = ilitek_spi_write;
    iddev->read = ilitek_spi_read;

    iddev->spi_speed = ilitek_tddi_ic_spi_speed_ctrl;
    iddev->actual_tp_mode = P5_X_FW_DEMO_MODE;

    if (TDDI_RST_BIND)
        iddev->reset = TP_IC_WHOLE_RST;
    else
        iddev->reset = TP_HW_RST_ONLY;

    iddev->rst_edge_delay = 1;
    iddev->fw_open = FILP_OPEN;
    iddev->fw_upgrade_mode = UPGRADE_IRAM;
    iddev->mp_move_code = ilitek_tddi_move_mp_code_iram;
    iddev->gesture_move_code = ilitek_tddi_move_gesture_code_iram;
    iddev->esd_recover = ilitek_tddi_wq_esd_spi_check;
    iddev->ges_recover = ilitek_tddi_touch_esd_gesture_iram;
    iddev->gesture_mode = P5_X_FW_GESTURE_NORMAL_MODE;
    iddev->wtd_ctrl = ON;
    iddev->report = ENABLE;
    iddev->netlink = DISABLE;
    iddev->debug_node_open = DISABLE;
}
```

```
static int ilitek_i2c_probe(struct i2c_client *i2c, const struct i2c_device_id *id)
{
    struct touch_bus_info *info =
        container_of(to_i2c_driver(i2c->dev.driver),
            struct touch_bus_info, bus_driver);

    ipio_info("ilitek i2c probe\n");

    if (!i2c) {
        ipio_err("i2c client is NULL\n");
        return -ENODEV;
    }

    if (i2c->addr != TDDI_I2C_ADDR) {
        i2c->addr = TDDI_I2C_ADDR;
        ipio_info("i2c addr doesn't be set up, use default : 0x%x\n", i2c->addr);
    }

    if (!i2c_check_functionality(i2c->adapter, I2C_FUNC_I2C)) {
        ipio_err("i2c functions are not supported\n");
        return -ENODEV;
    }

    iddev = devm_kzalloc(&i2c->dev, sizeof(struct ilitek_tddi_dev), GFP_KERNEL);
    if (ERR_ALLOC_MEM(iddev)) {
        ipio_err("Failed to allocate iddev memory, %ld\n", PTR_ERR(iddev));
        return -ENOMEM;
    }

    iddev->i2c = i2c;
    iddev->spi = NULL;
    iddev->dev = &i2c->dev;
    iddev->hwif = info->hwif;
    iddev->phys = "I2C";

    iddev->write = ilitek_i2c_write;
    iddev->read = ilitek_i2c_read;

    iddev->spi_speed = NULL;
    iddev->actual_tp_mode = P5_X_FW_DEMO_MODE;

    if (TDDI_RST_BIND)
        iddev->reset = TP_IC_WHOLE_RST;
    else
        iddev->reset = TP_HW_RST_ONLY;

    iddev->rst_edge_delay = 100;
    iddev->fw_open = FILP_OPEN;
    iddev->fw_upgrade_mode = UPGRADE_FLASH;
    iddev->mp_move_code = ilitek_tddi_move_mp_code_flash;
    iddev->gesture_move_code = ilitek_tddi_move_gesture_code_flash;
}
```

5. IC 設定

v2.x 為了因客戶需求做單一化處理，又要兼顧 IC 不同的設定，這些區分都因集中化管理以方便篩選，具體內容可參考 **ilitek_ic.c** 裡面的 **ilitek_tddi_ic_check_support** 函式。

6. 模組兼容

一個 IIC 會結合多個不同面板廠家提供給客戶，所以在驅動裡要分辨別出現不同模組再指向不同的定義和相對應的固件是很重要的。鑑於客戶辨認模組型號的方法不同，這裡驅動已寫好一個函式，而使用者只需寫好對應的實作方法和回傳返回值，即可完成在驅動裡完成模組兼容。具體方法請參考 `ilitek_update_tp_module_info` 和 `ilitek_get_tp_module`。

```
static int ilitek_get_tp_module(void)
{
    /*
     * TODO: users should implement this function
     * if there are various tp modules been used in projects.
     */
    return 0;
}

static void ilitek_update_tp_module_info(void)
{
    int module;

    module = ilitek_get_tp_module();

    switch(module) {
    case MODEL_CSOT:
        idev->md_name = "CSOT";
        idev->md_fw_filp_path = CSOT_FW_FILP_PATH;
        idev->md_fw_rq_path = CSOT_FW_REQUEST_PATH;
        idev->md_ini_path = CSOT_INI_NAME_PATH;
        idev->md_ini_rq_path = CSOT_INI_REQUEST_PATH;
        idev->md_fw_ili = CTPM_FW_CSOT;
        idev->md_fw_ili_size = sizeof(CTPM_FW_CSOT);
        break;
    case MODEL_AUO:
```

※ 若需修改定義，請參閱 [ilitek_fw.h](#)：

7. Pin control

若是 MTK 平台，一般 GPIO Pin 的控制基本上都會寫在它們自己的 TPD 子系統來管理，但若是 Qcom 平台，可能就需要利用 Linux pinctrl system 來管理 GPIO 有關的邏輯及腳位控制。目前驅動並無 pinctrl 的實現，但下面提供一些實現方法供參閱：

● 初始化

```
static int ilitek_pinctrl_init(void)
{
    int ret = 0;

    idev->ts_pinctrl = devm_pinctrl_get(idev->dev);
    if (IS_ERR_OR_NULL(idev->ts_pinctrl)) {
        ipio_err("Failed to get pinctrl");
        ret = PTR_ERR(idev->ts_pinctrl);
        idev->ts_pinctrl = NULL;
        return ret;
    }

    idev->int_default = pinctrl_lookup_state(idev->ts_pinctrl,
        "default");
    if (IS_ERR_OR_NULL(idev->int_default)) {
        ipio_err("Pin state[default] not found");
        ret = PTR_ERR(idev->int_default);
        goto exit_put;
    }

    idev->int_out_high = pinctrl_lookup_state(idev->ts_pinctrl,
        "int-output-high");
    if (IS_ERR_OR_NULL(idev->int_out_high)) {
        ipio_err("Pin state[int-output-high] not found");
        ret = PTR_ERR(idev->int_out_high);
    }

    idev->int_out_low = pinctrl_lookup_state(idev->ts_pinctrl,
        "int-output-low");
    if (IS_ERR_OR_NULL(idev->int_out_low)) {
        ipio_err("Pin state[int-output-low] not found");
        ret = PTR_ERR(idev->int_out_low);
        goto exit_put;
    }

    idev->int_input = pinctrl_lookup_state(idev->ts_pinctrl,
        "int-input");
    if (IS_ERR_OR_NULL(idev->int_input)) {
        ipio_err("Pin state[int-input] not found");
        ret = PTR_ERR(idev->int_input);
        goto exit_put;
    }

    return 0;
exit_put:
    devm_pinctrl_put(idev->ts_pinctrl);
    idev->ts_pinctrl = NULL;
    idev->int_default = NULL;
    idev->int_out_high = NULL;
    idev->int_out_low = NULL;
    idev->int_input = NULL;
    return ret;
}
```

```
ilitek_pinctrl_init();
pinctrl_select_state(idev->ts_pinctrl, idev->int_input);
```

● DTS

```

102     },
103     int-input {
104         ts int input: ts_int_input { liufurong, 12 days
105             mux {
106                 pins = "gpio65";
107                 function = "gpio";
108             };
109             config {
110                 pins = "gpio65";
111                 drive-strength = <8>;
112                 input-enable;
113                 bias-pull-up;
114             };
115         };
116     };
117 };
118 /* HS70 code for HS70-133 by liufurong at 2019/10/31 end */
119

```


```

1 &l2c_3{
2     status = "disable";
3 };
4 &spl_3{
5     status = "ok";
6     ilitek@0{
7         compatible = "tchip,ilitek";
8         reg = <0x0>;
9         /* HS70 add for SR-ZQL1871-01-177 by gaozhengwei at 2019/11/02 start */
10        lcm_lab-supply = <&lcd_b_lldo_vreg>;//Adding Power Properties
11        lcm_tbb-supply = <&lcd_b_ncp_vreg>;
12        /* HS70 add for SR-ZQL1871-01-177 by gaozhengwei at 2019/11/02 end */
13        spi-max-frequency = <100000000>;
14        interrupt-parent = <&tlmm>;
15        interrupts = <65 0x02>;
16        touch,irq-gpio = <&tlmm 65 0x02>;
17        touch,reset-gpio = <&tlmm 64 0x01>;
18        /* HS70 code for HS70-133 by liufurong at 2019/10/31 start */
19        pinctrl-names = "default","int-output-high","int-output-low","int-input";
20        pinctrl-0 = <&ts_int_default>;
21        pinctrl-1 = <&ts_int_output_high>;
22        pinctrl-2 = <&ts_int_output_low>;
23        pinctrl-3 = <&ts_int_input>;
24        /* HS70 code for HS70-133 by liufurong at 2019/10/31 end */
25    };
26    focaltech@1 {
27        compatible = "focaltech,fts";
28        reg = <0x1>;

```

相關功能說明

這個部份會說明驅動的一些支援功能如何開啟，以及提供的偵錯方式。

 修改整機產測的 csv 及 ini、FW 更新的 hex、電池狀態等定義

DEBUG_DATA_FILE_SIZE	CSV 文件的大小
DEBUG_DATA_FILE_PATH	儲存 debug (by command 0xFA) 資料的路徑
CSV_LCM_ON_PATH	儲存 MP 亮屏 CSV 路徑
CSV_LCM_OFF_PATH	儲存 MP 暗屏 CSV 路徑
INI_NAME_PATH	MP Ini 設定檔路徑
UPDATE_FW_FILP_PATH	固件更新路徑 (by filp_open)
UPDATE_FW_REQUEST_PATH	固件更新路徑 (by request_firmware)
POWER_STATUS_PATH	讀取電源狀態路徑
DUMP_FLASH_PATH	讀取 Flash 資料儲存路徑
DUMP_IRAM_PATH	讀取 IRAM 資料儲存路徑

- 支援功能的開關，依序為 DDI 與 TP pin 腳的綁定(決定 reset 方式)、報點 protocol、報點壓力參數、ESD 檢查、充電狀態、手勢功能、regulator 供電以及 TP/DDI 休眠的順序 (預設開為 DDI 先)

VDD_VOLTAGE	VDD 電壓值 (如開啟 REGULATOR_POWER)
VCC_VOLTAGE	VCC 電壓值 (如開啟 REGULATOR_POWER)
SPI_CLK	SPK Clock
SPI_RETRY	SPI retry 次數
TR_BUF_SIZE	報點的緩衝大小
WQ_ESD_DELAY	每隔某時間，調用 workqueue 檢查 IC 狀態
WQ_BAT_DELAY	每隔某時間，調用 workqueue 檢查電源狀態
MT_B_TYPE	設置 B 或者 A 類協議
TDDI_RST_BIND	TP RST 和 DDI RST 是否綁定
MT_PRESSURE	開啟上報壓力值
ENABLE_WQ_ESD	開啟 ESD 檢查
ENABLE_WQ_BAT	開啟電源狀態檢查
ENABLE_GESTURE	開啟手勢
REGULATOR_POWER	使用 regulator 註冊電源管理
TP_SUSPEND_PRIO	如果設置為 ENABLE，表示 TP 會先於 DDI 做 suspend (MTK 平台需改 kernel)
RESUME_BY_DDI	由 LCM Driver 調用 TP resume
BOOT_FW_UPDATE	開機更新固件
I2C_DMA_TRANSFER	使用 i2c dma 傳輸
SPI_DMA_TRANSFER_SPLIT	讀寫 SPI 時，會根據定義的大小來做分割

- 定義 Screen 的寬度與長度，這會影響座標轉換的結果。

TOUCH_SCREEN_X_MIN	向 input subsystem 註冊上報的屏幕最小寬度
TOUCH_SCREEN_Y_MIN	向 input subsystem 註冊上報的屏幕最小高度
TOUCH_SCREEN_X_MAX	向 input subsystem 註冊上報的屏幕最大寬度
TOUCH_SCREEN_Y_MAX	向 input subsystem 註冊上報的屏幕最大高度
MAX_TOUCH_NUM	最大支援觸摸手指數
TPD_HEIGHT	轉坐標用，需固件配合
TPD_WIDTH	轉坐標用，需固件配合

✚ 修改固件更新的條件 (Flash only)

- 如果客戶有需求針對更新條件做修改，請參考 ilitek_flash.c 代碼裡，有判斷版本大小，如下圖：

```
/* Check FW version */
ipio_info("New FW ver = 0x%x, Current FW ver = 0x%x\n", tfd.new_fw_cb, idev->chip->fw_ver);
if (tfd.new_fw_cb != idev->chip->fw_ver) {
    ipio_info("FW version not matched, do upgrade\n");
    return UPDATE_FAIL;
}
```

- 不帶 Flash 的 TDDI，則一律更新，不需要版本判斷。

✚ 開啟驅動的 debug 訊息 (kernel log 可見)

- 用 cat 節點做開關設置。每執行一次，開關則轉態一次
adb shell "cat /proc/ilitek/debug_level"

✚ Firmware 更新

- 使用 adb 把 firmware push 到/sdcard 底下並命名為 ILITEK_FW
adb push FW_TDDI_TRUNK_FB.hex /sdcard/ILITEK_FW
- 進行更新
cat /proc/ilitek/fw_upgrade

✚ 產測分別為亮屏和暗屏測試，由兩個不同的節點組成，在執行前，需確認有正確的 ini 檔案有被放置相對應的路徑方可執行。

- 使用 adb 把 ini 檔案放置指定路徑裡，預設為 sdcard 底下
adb push mp.ini /sdcard/
- 執行產測
adb shell cat /proc/ilitek/mp_lcm_on_test (亮屏)
adb shell cat /proc/ilitek/mp_lcm_off_test (暗屏)
- 產測結果會根據亮暗屏測試，放置不同的目錄，預設為 sdcard 底下，可透過 ilitek.h 修改。

```
sdcard_rw      2010-01-01 00:02 ilitek_mp_lcm_off_log
sdcard_rw      2010-01-01 00:01 ilitek_mp_lcm_on_log
```

✚ 讀寫 TP 的 register 可透過節點 rw_tp_reg 來完成。共有 5 組參數需要輸入，其中若做讀取的話，只需輸入前 3 組參數即可。使用方式如下：

- Parameter 1：控制 MCU (0 為 ON，1 為 STOP)
- Parameter 2：讀寫控制 (0 為 Read，1 為 Write)
- Parameter 3：暫存器地址
- Parameter 4：設置寫的值
- Parameter 5：設置寫的長度


```
# echo 0x0,0x0,0x4009C > /proc/ilitek/rw_tp_reg → MCU on ; Read
# cat /proc/ilitek/rw_tp_reg
# echo 0x1,0x1,0x41000,0x0,0x4 > /proc/ilitek/rw_tp_reg → MCU stop ; Write
# cat /proc/ilitek/rw_tp_reg
```

※ 記得後面都要接 `cat /proc/ilitek/rw_tp_reg` 才算完成。

節點 `show_delta_data` 和 `show_raw_data` 可以取得當前一張 frame 的資料，每輸入一次，即取一次。
需搭配正確的 firmware 才可正常運行

```
# cat /proc/ilitek/show_delta_data (or show_raw_data)
```

```
===== Rawdata =====
Header 0xb7 ,Type 4, Length 34050
[ 1] 7808 8467 8465 8493 8479 8495 8483 8438 8580 8583 8613 8633 8653 8641 8654 8651 8649 8538
[ 2] 7904 8637 8636 8676 8673 8692 8679 8641 8768 8397 7866 7896 7925 7914 7934 7929 7925 8315
[ 3] 7823 8542 8543 8592 8587 8607 8588 8544 8668 7907 8425 8459 8486 8480 8486 8488 8484 7810
[ 4] 8403 8595 8598 8643 8642 8666 8639 8605 8720 7905 8385 8426 8450 8450 8465 8456 8446 7792
[ 5] 8496 8697 8704 8749 8749 8767 8744 8705 8296 7962 8454 8492 8516 8512 8522 8513 8511 7841
[ 6] 8597 7701 7703 7745 7736 7757 7714 8790 7781 7904 8400 8437 8466 8462 8463 8456 8452 7780
[ 7] 7931 8627 8629 8674 8674 8682 8640 8602 8676 8443 7869 7910 8487 8483 8489 8484 8477 8306
[ 8] 8540 8745 8745 7671 7678 7684 8757 8720 7755 8644 8072 8670 8692 8678 8687 8685 8676 8504
[ 9] 8578 7679 7674 7705 7710 7718 7687 8767 8343 8083 8550 8589 8614 8602 8608 8599 8599 8433
[10] 8638 7781 7772 8298 8304 8311 8280 7741 7809 8557 8530 8568 8598 8587 8597 8595 8592 8476
[11] 8494 8628 8627 8659 8657 8660 8631 8586 8646 7966 8427 8464 8492 8491 8498 8492 8498 8380
[12] 8545 8746 8749 7669 8770 7670 8750 8713 7718 8639 8599 8642 8669 8659 8670 8669 8666 8501
[13] 8578 7672 8772 7694 7688 7696 7671 8736 8301 8279 8738 8771 7707 7706 7715 7716 7715 8631
[14] 8576 7673 8765 7694 7689 7692 7663 8729 7736 7713 8708 8746 7689 7679 7691 7688 7697 8618
[15] 8634 7727 7718 7744 7743 7752 7723 7675 8343 7727 8709 8743 7686 7684 7685 7688 7693 8606
[16] 7700 7779 7767 7798 7803 7803 7770 7726 7865 8602 8545 8579 8604 8601 8603 8600 8605 8445
[17] 7708 7794 7777 7810 7808 7815 7779 7732 7862 7709 8680 8713 8739 8738 8741 8735 8741 8576
[18] 7715 7796 7770 7800 7803 7809 7773 7728 8393 7714 8682 8700 8740 8736 8745 8738 8735 8576
[19] 8659 7748 7728 7755 7754 7759 7730 7686 8359 7783 8773 7710 8278 7734 7739 7738 8281 8655
[20] 7793 7868 7842 7872 7872 7878 7799 8469 7794 8766 7712 7742 7737 7740 7724 7739 8656
[21] 7824 7896 7875 7901 7895 7898 7867 7824 8497 8372 7731 7753 7776 7773 7779 7756 7776 7718
[22] 7930 8003 7976 7999 8006 8010 7972 7934 8604 7852 7746 7765 7792 7788 7790 7774 7794 7732
[23] 7855 7927 7897 7923 7923 7925 7890 7848 8516 8310 8743 7686 7712 7710 7711 7693 7711 8632
[24] 7856 7927 7896 7913 7914 7918 7889 7842 8516 8336 8760 7702 7732 7726 7736 7716 7734 8655
[25] 7748 7814 7786 7804 7804 7812 7778 7731 8414 8366 7705 7724 7757 7754 7760 7737 7758 8675
[26] 7909 7974 7956 7970 7970 7975 7933 8077 8570 7788 8709 8746 7716 7702 7716 7706 7723 8638
[27] 7886 7952 7926 7939 7938 7945 7907 7868 8540 7900 7707 8363 8380 8377 8388 8378 8399 7811
[28] 7855 7919 7894 7902 7907 7909 7874 7832 8505 8474 7822 7855 7870 7871 7877 7871 7897 7839
[29] 7929 7996 7968 7978 7979 7985 7947 7909 8580 7978 8363 8384 8403 8398 8407 8404 8424 7841
[30] 7970 8038 8012 8017 8020 8019 7989 7948 8616 8492 7808 7834 7854 7851 7856 7863 7885 7833
[31] 7908 8509 7946 7951 7952 7955 7924 7884 8513 8510 7830 7853 7871 7872 7877 7878 7908 7857
[32] 7833 7987 7873 7832 7848 7839 7770 7704 8244 7836 7731 7749 7702 8647 7717 8622 8626 8714

===== Deltadata =====
Header 0xb7 ,Type 4, Length 34049
[ 1] 6 -3 -3 -6 -9 -3 -18 -12 -3 -3 -12 0 15 3 -3 -3 -9 -3
[ 2] 12 3 6 -6 15 12 3 9 6 0 -9 0 9 -3 -6 3 -3 0
[ 3] 0 -3 3 -6 -9 -9 0 0 3 -3 -15 -6 3 6 -12 -3 -12 -9
[ 4] 12 -6 3 -12 3 -3 0 -6 -3 0 -9 -6 -3 3 -9 -9 -3 -6
[ 5] 6 -6 0 -12 6 3 0 -6 -6 3 -12 -15 6 6 9 3 -6 0
[ 6] 9 -3 6 0 3 0 0 -12 -3 0 0 -6 0 9 -3 0 -12 -6
[ 7] 0 -6 0 -3 3 6 0 -6 0 3 -18 -6 3 9 -12 -3 -15 -15
[ 8] 3 0 6 -9 -3 9 3 6 3 0 -12 -6 3 -3 -6 -6 -9 -6
[ 9] 9 -3 -3 -12 -9 0 0 0 3 6 0 3 12 -9 -9 -6 -6
[10] 6 -6 0 -12 0 0 -3 -6 0 -6 -12 -6 6 6 0 -3 -3 -6
[11] 6 -3 6 -15 3 -3 6 -3 0 6 -6 6 3 15 3 -9 -3 -9 -9
[12] 0 0 0 -6 0 0 -6 -6 -3 6 -9 -3 9 -6 -12 -3 -12 0
[13] 3 -6 6 0 -6 -6 -6 -3 3 -3 -12 -3 0 3 3 0 -12 -9
[14] 6 3 0 -3 3 9 9 9 6 9 -9 0 3 9 -3 -6 -3
[15] 15 -6 0 -6 3 3 -6 -3 9 0 -18 -9 6 0 9 -6 -15 -6
[16] 0 0 0 -9 3 0 0 -6 0 -6 -15 -6 3 9 -12 -3 -15 -6
[17] 6 -3 -3 -6 -6 3 -3 6 6 0 -18 -9 6 9 -9 -3 -6 3
[18] -3 -6 -6 -15 -3 -3 -12 -15 -12 0 -15 -6 9 9 -3 0 -9 3
[19] -6 9 3 -3 0 0 3 -3 -3 -3 -15 -6 6 3 -3 -6 -12 0
[20] 12 -3 0 -6 -3 0 -3 -3 -3 -3 -21 -12 0 -3 -3 -6 3
[21] 6 -6 -3 -15 -6 3 3 -9 -12 0 -12 -3 0 0 -12 -6 -12 -6
[22] 3 6 0 -6 0 -3 0 0 0 3 -9 0 3 6 -6 -3 -9 3
[23] 0 -3 -6 -3 3 3 -3 -6 -3 6 -9 6 3 12 -12 0 -9 0
[24] 6 -3 -3 -3 -9 -3 -6 -9 -3 6 -6 -9 0 6 -9 -12 0
[25] -3 -6 -3 -6 -6 0 0 -6 9 15 0 15 12 3 9 6 3 -3
[26] 6 -6 -3 -12 -12 -6 -9 -9 3 6 -9 -6 12 9 0 9 3 0
[27] 3 0 0 -6 -3 9 6 0 3 -9 -12 0 9 6 -9 0 -15 -6
[28] 12 -3 -6 -3 0 0 0 -3 -6 3 -12 0 6 6 -9 -6 -6 -6
[29] 3 3 -6 -18 -6 -6 -6 -3 6 -3 6 6 6 -9 0 -6 0
[30] 0 -12 -3 -12 3 0 9 -15 3 -6 -18 3 9 6 -15 -6 -12 -3
[31] 3 6 -12 -12 -3 3 3 0 3 -3 -18 -6 3 3 -3 0 -12 -3
[32] 12 3 0 -12 3 3 3 -6 3 -3 -9 -3 12 9 -3 -3 -9 -12
```

當 TP 有 Flash IC 時，可利用 `ioctl` 輸入命令把 flash 資料印出並儲存至指定目錄檔案（預設為 `/sdcard/flash_dump`），需輸入 2 組參數做為起始和結束位址。

- Parameter 1：指定動作 `dumpflashdata`
- Parameter 2：起始位址
- Parameter 3：結束位址

```
# echo dumpflashdata,0x0,0x80 > /proc/ilitek/ioctl
```

```
Offset: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000: 48 00 66 D4 48 00 65 14 48 00 65 32 48 00 65 50 H.fTH.e.H.e2H.eP
00000010: 48 00 65 6E 48 00 65 8C 48 00 65 AA 48 00 65 C8 H.enH.e.H.e*H.eH
00000020: 48 00 65 E6 48 00 66 3C 48 00 66 3E 48 00 66 40 H.efH.f<H.f>H.f@
00000030: 48 00 66 42 48 00 66 44 48 00 66 46 48 00 66 48 H.fBH.fDH.fFH.fH
00000040: 48 00 66 4A 48 00 66 4C 48 00 66 4E 48 00 66 50 H.fJH.fLH.fNH.fP
00000050: 48 00 66 52 48 00 66 54 48 00 66 56 48 00 66 58 H.fRH.fTH.fVH.fX
00000060: 48 00 66 5A 48 00 66 5C 48 00 66 5E 48 00 66 60 H.fZH.f\H.f^H.f`
00000070: 48 00 66 62 48 00 66 64 48 00 66 66 48 00 66 68 H.fbH.fdH.ffH.fh
00000080: 48 H
```


注意！該檔案 flash_dump 必需要用 UltraEdit 或其它可以看 16 進制的應用程式開啟才可正常顯示。

如果 TP 沒有帶 Flash 而是燒錄資料到 iram (意即 host download)，同樣也可以操作節點來讀到 iram 的資料，預設儲存路徑為 /sdcard/iram_dump，需輸入 2 組參數做為起始和結束位址。

- Parameter 1：指定動作 **dumpiramdata**
- Parameter 2：起始位址
- Parameter 3：結束位址

```
# echo dumpiramdata,0x0,0xffff > /proc/ilitek/ioctl
```

讀寫 DDI 操作也可以利用輸入命令至 ioctl 達成效果，以下為操作範例：

- Parameter 1：指定動作 **getddiregdata**
- Parameter 2：Page
- Parameter 3：DDI register

```
# echo getddiregdata,0x0,0xa > /proc/ilitek/ioctl
```

- Parameter 1：指定動作 **setddiregdata**
- Parameter 2：Page
- Parameter 3：DDI register
- Parameter 4：寫的值

```
# echo setddiregdata,0x0,0x28,0x0 > /proc/ilitek/ioctl
```

讀寫 IO (I2C/SPI) 同樣可以透過 ioctl 來完成，指令動作分為三類：讀、寫和讀寫同時(可加 delay)。

- Parameter 1：指令動作 **iow**
- Parameter 2：寫的長度
- Parameter 3 ... N：寫的資料

```
# echo iow,0x2,0xF6,0x22 > /proc/ilitek/ioctl
```

```
# echo iow,0x3,0x1,0x6,0x5 > /proc/ilitek/ioctl
```

- Parameter 1：指令動作 **ior**
- Parameter 2：讀取的長度

```
# echo ior,0x4 > /proc/ilitek/ioctl
```

- Parameter 1：指令動作 **iowr**
- Parameter 2：寫的長度
- Parameter 3：讀的長度
- Parameter 4：延遲時間
- Parameter 5 ... N：寫的資料

```
# echo iowr,0x2,0x4,0x1,0xF6,0x22 > /proc/ilitek/ioctl
```

還有其它指令動作供調試用，像是做不同類型的 Reset、Workqueue 關閉、手勢開關和讀取 TP 資訊等等，請自行參閱源代碼。

```

if (strcmp(cmd, "hwreset") == 0) {
    ilitek_tddi_reset_ctrl(TP_HW_RST_ONLY);
} else if (strcmp(cmd, "icwholereset") == 0) {
    ilitek_ice_mode_ctrl(ENABLE, OFF);
    ilitek_tddi_reset_ctrl(TP_IC_WHOLE_RST);
    ilitek_ice_mode_ctrl(DISABLE, OFF);
} else if (strcmp(cmd, "iccodereset") == 0) {
    ilitek_ice_mode_ctrl(ENABLE, OFF);
    ilitek_tddi_reset_ctrl(TP_IC_CODE_RST);
    ilitek_ice_mode_ctrl(DISABLE, OFF);
} else if (strcmp(cmd, "getinfo") == 0) {
    ilitek_tddi_ic_get_info();
    ilitek_tddi_ic_get_protocol_ver();
    ilitek_tddi_ic_get_fw_ver();
    ilitek_tddi_ic_get_core_ver();
    ilitek_tddi_ic_get_tp_info();
    ilitek_tddi_ic_get_panel_info();
} else if (strcmp(cmd, "enableicemode") == 0) {
    if (data[1] == ON)
        ilitek_ice_mode_ctrl(ENABLE, ON);
    else
        ilitek_ice_mode_ctrl(ENABLE, OFF);
} else if (strcmp(cmd, "disableicemode") == 0)
    ilitek_ice_mode_ctrl(DISABLE, OFF);
} else if (strcmp(cmd, "enablewqesd") == 0) {
    ilitek_tddi_wq_ctrl(WQ_ESD, ENABLE);
} else if (strcmp(cmd, "enablewqbat") == 0) {
    ilitek_tddi_wq_ctrl(WQ_BAT, ENABLE);
} else if (strcmp(cmd, "disablewqesd") == 0) {
    ilitek_tddi_wq_ctrl(WQ_ESD, DISABLE);
} else if (strcmp(cmd, "disablewqbat") == 0) {
    ilitek_tddi_wq_ctrl(WQ_BAT, DISABLE);
} else if (strcmp(cmd, "gesture") == 0) {
    iddev->gesture = !iddev->gesture;
    ipio_info("gesture = %d\n", iddev->gesture);
} else if (strcmp(cmd, "gesturenormal") == 0) {
    iddev->gesture_mode = P5_X_FW_GESTURE_NORMAL_MODE;
    ipio_info("gesture mode = %d\n", iddev->gesture_mode);
} else if (strcmp(cmd, "gesture") == 0) {
    iddev->gesture_mode = P5_X_FW_GESTURE_INFO_MODE;
    ipio_info("gesture mode = %d\n", iddev->gesture_mode);
} else if (strcmp(cmd, "netlink") == 0) {
    iddev->netlink = !iddev->netlink;
    ipio_info("netlink flag= %d\n", iddev->netlink);
} else if (strcmp(cmd, "switchtestmode") == 0) {
    tp_mode = P5_X_FW_TEST_MODE;
    ilitek_tddi_switch_mode(&tp_mode);
} else if (strcmp(cmd, "switchdebugmode") == 0) {
    tp_mode = P5_X_FW_DEBUG_MODE;
    ilitek_tddi_switch_mode(&tp_mode);
} else if (strcmp(cmd, "switchdemomode") == 0) {
    tp_mode = P5_X_FW_DEMO_MODE;
    ilitek_tddi_switch_mode(&tp_mode);
}

```