1. 專案23\_Moulti-touch:

* 一般而言觸控螢幕有分成電阻屏和電容屏，目前市面上的攜帶式裝置都是以電容屏居多。絕大多數觸控電容屏是需要一個觸控IC的，而這個驅動IC大多是透過I2C介面來實現。
* Linux系統下，電容驅動框架協議:
  + 1. 電容觸控式I2C介面，也就是需要I2C設備驅動(I2C設備驅動框架)。
    2. 觸控IC透過中斷腳位，透過中斷獲取觸摸訊號(中斷驅動框架)。
    3. 電容觸控屏得到的是觸摸絕對位置(Input子系統)。

1. Linux系統中多點觸控協議:

* Type A: 適用於無法區分或追蹤觸控點的情況，此類型的設備會上報原始數據。
* Type B: 用於能夠透過硬體追蹤並區分觸控點的觸控設備，此類型設備會通過 slot更新某一個觸控點的信息，一般的多點電容觸控屏IC都具備此功能。
* 實際在Linux系統中，觸控點的信息通過一系列的ABS\_MT事件（有的資料也稱為消息）上報給Linux kernel，只有ABS\_MT事件是用於多點觸控的。ABS\_MT件定義在文件 include/uapi/linux/input-event-codes.h中。
* 在ABS\_MT事件中，最常用的是ABS\_MT\_SLOT(用來報告觸摸點ID)、ABS\_MT\_POSITION\_X(用來報告觸摸點X座標)、ABS\_MT\_POSITION\_Y(用來報告觸摸點Y座標)和ABS\_MT\_TRACKING\_ID(用來區分觸摸點)。
* 簡單總結一下，MT協議隸屬於Linux的input子系統，驅動通過大量的ABS\_MT事件向Linux內核上報多點觸摸坐標數據。根據觸控IC的不同，分為Type A和Type B兩種類型，不同的類型其上報時序不同，目前使用最多的是Type B類型。

1. tslib使用:

* tslib是一個開源library用來給電阻屏或電容屏做校準，打開buildroot的圖形化介面，下make menuconfig指令，透過此路徑找到Target packages -> Libraries -> Hardware handling -> [\*] tslib，把tslib勾選，重新編譯就可以使用了。
* 電容屏可以不用校準，如果是電阻屏就要先進行校準！校準的話輸入如下命令：ts\_calibrate校準完成以後如果不滿意，或者不小心對電容屏做了校準，那麼直接刪除掉 /etc/pointercal 文件即可。 最後我們使用 ts\_test\_mt這個命令來測試觸控屏工作是否正常，以及多點觸控是否有效。

1. Multi-touch驅動實作:
2. 首先，先定義設被結構體:

* **暫存器定義**：定義了FT5426控制器的各個寄存器地址。(如表一，圖一)
* **常數定義**：定義了觸摸事件和控制器支持的最大觸摸點數量。(如表二，圖一)
* **結構體 edt\_ft5426\_dev**：用於存儲與FT5426觸摸控制器相關的設備信息，包括I2C客戶端、輸入設備、復位GPIO和中斷GPIO。(如表三，圖一)

(表一:暫存器定義)

|  |  |  |
| --- | --- | --- |
| **宏名稱** | **值** | **描述** |
| FT5426\_DEVIDE\_MODE\_REG | 0x00 | 模式寄存器 |
| FT5426\_TD\_STATUS\_REG | 0x02 | 狀態寄存器 |
| FT5426\_TOUCH\_DATA\_REG | 0x03 | 觸摸數據的起始寄存器 |
| FT5426\_ID\_G\_MODE\_REG | 0xA4 | 中斷模式寄存器 |

(表二:常數定義)

|  |  |  |
| --- | --- | --- |
| **宏名稱** | **值** | **描述** |
| MAX\_SUPPORT\_POINTS | 5 | FT5426 支持的最大觸摸點數量 |
| TOUCH\_EVENT\_DOWN | 0x00 | 觸摸按下事件 |
| TOUCH\_EVENT\_UP | 0x01 | 觸摸抬起事件 |
| TOUCH\_EVENT\_ON | 0x02 | 觸摸接觸事件 |
| TOUCH\_EVENT\_RESERVED | 0x03 | 保留事件 |

(表三: 結構體 edt\_ft5426\_dev定義)

|  |  |  |
| --- | --- | --- |
| **成員變數** | **類型** | **描述** |
| client | struct i2c\_client \* | 指向 I2C 客戶端結構體的指標 |
| input | struct input\_dev \* | 指向輸入設備結構體的指標 |
| reset\_gpio | int | 用於復位的 GPIO 引腳 |
| irq\_gpio | int | 用於中斷的 GPIO 引腳 |



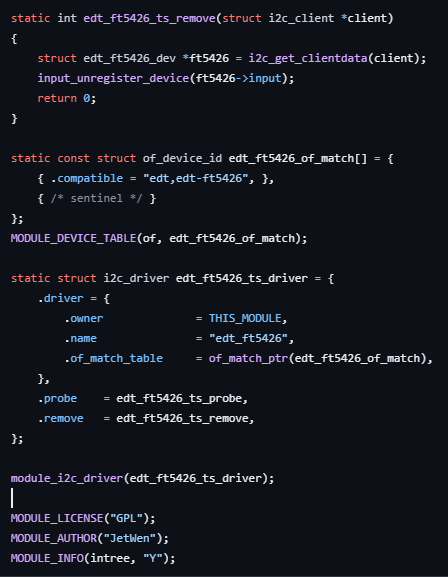
(圖一: 結構體 edt\_ft5426\_dev，常數定義，暫存器定義內容)

1. 接著，註冊一個 I2C驅動程序，用於處理FT5426觸摸控制器(如下圖二及表四)

* **函數 edt\_ft5426\_ts\_remove**
* **功能**：移除 I2C 客戶端設備，解除對應的輸入設備的註冊。
* **參數**：client - 指向 I2C 客戶端結構體的指標。
* **返回值**：總是返回 0，表示成功。
* **兼容性表 edt\_ft5426\_of\_match**
* **功能**：定義設備樹中的兼容性條目，用於匹配FT5426觸摸控制器。
* **內容**：包含一個兼容性條目 edt,edt-ft5426。
* **I2C 驅動結構 edt\_ft5426\_ts\_driver**
* **功能**：定義和註冊 I2C 驅動程序，用於處理FT5426觸摸控制器。
* **主要成員**：
  + driver：包含驅動程序的基本信息，如模塊所有者、名稱和設備樹匹配表。
  + probe：指向驅動程序的探測函數（在這段代碼中未展示）。
  + remove：指向驅動程序的移除函數 edt\_ft5426\_ts\_remove。

(表四:I2C子系統匹配FT5426觸控設備)

|  |  |
| --- | --- |
| **步驟** | **說明** |
| 移除函數 | 定義 edt\_ft5426\_ts\_remove 函數，解除輸入設備的註冊並移除 I2C 客戶端設備。 |
| 兼容性表 | 定義 edt\_ft5426\_of\_match 表，用於匹配 FT5426 觸摸控制器的設備樹條目。 |
| I2C 驅動結構 | 定義 edt\_ft5426\_ts\_driver 結構，包含驅動程序的基本信息、探測函數和移除函數。 |
| 模塊註冊 | 使用 module\_i2c\_driver 宏註冊 I2C 驅動程序 edt\_ft5426\_ts\_driver。 |
| 模塊許可證 | 定義模塊的許可證為 GPL。 |
| 模塊作者 | 定義模塊作者為 "JetWen"。 |
| 模塊信息 | 提供模塊的其他信息，如 MODULE\_INFO 用於標記模塊是否包含在內核樹中。 |

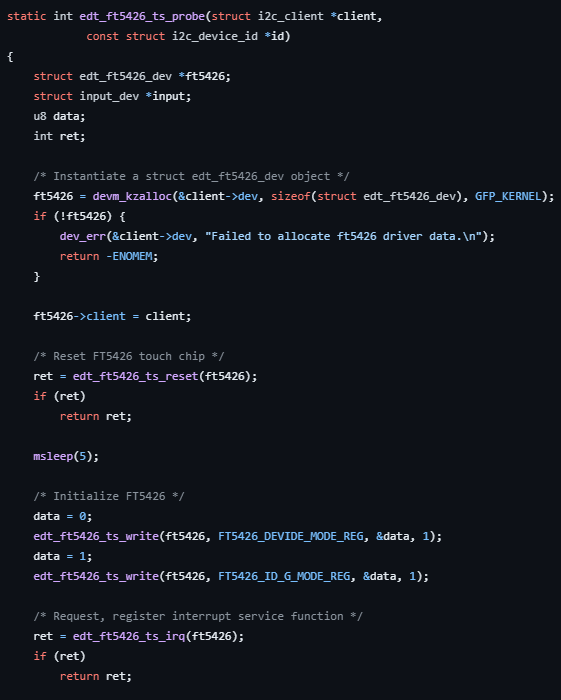


(圖二:I2C子系統匹配FT5426觸控設備)

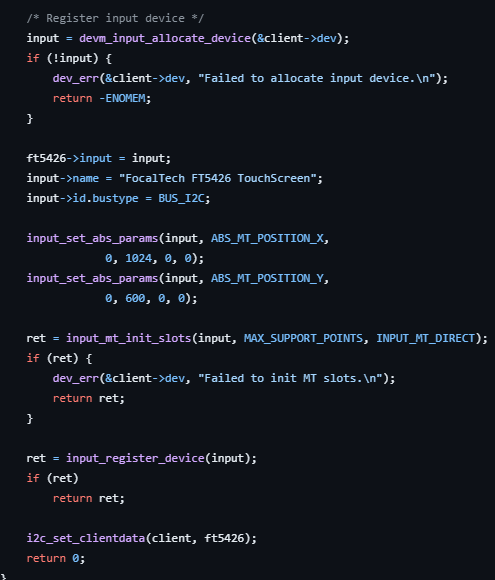
1. 接下來實現，edt\_ft5426\_ts\_probe 函數，用於在 I2C 驅動程序探測到 FT5426 觸摸屏設備時進行初始化。(如下表五，及圖三，圖四)

(表五: edt\_ft5426\_ts\_probe函數內部功能步驟與解析)

|  |  |
| --- | --- |
| **步驟** | **說明** |
| 分配內存 | 分配一個 edt\_ft5426\_dev 結構體對象。 |
| 設置 I2C 客戶端指標 | 將 I2C 客戶端指標設置為 ft5426 結構體的成員。 |
| 復位 FT5426 芯片 | 調用 edt\_ft5426\_ts\_reset 函數來復位 FT5426 觸摸芯片。 |
| 延遲 | 調用 msleep(5) 延遲 5 毫秒，等待復位完成。 |
| 初始化 FT5426 | 將 0 寫入 FT5426\_DEVIDE\_MODE\_REG 寄存器，將 1 寫入 FT5426\_ID\_G\_MODE\_REG 寄存器。 |
| 請求並註冊中斷服務函數 | 調用 edt\_ft5426\_ts\_irq 函數來請求並註冊中斷服務函數。 |
| 分配輸入設備 | 分配一個輸入設備 input。 |
| 設置輸入設備屬性 | 設置輸入設備的名稱、總線類型和絕對值參數（X 和 Y 軸的位置）。 |
| 初始化多點觸控槽 | 調用 input\_mt\_init\_slots 初始化多點觸控槽，支持最大觸控點數量為 MAX\_SUPPORT\_POINTS。 |
| 註冊輸入設備 | 註冊輸入設備，使其在系統中可用。 |
| 設置客戶端數據 | 調用 i2c\_set\_clientdata 將 ft5426 結構體與 I2C 客戶端關聯起來。 |



(圖三:Probe函式內容實現)

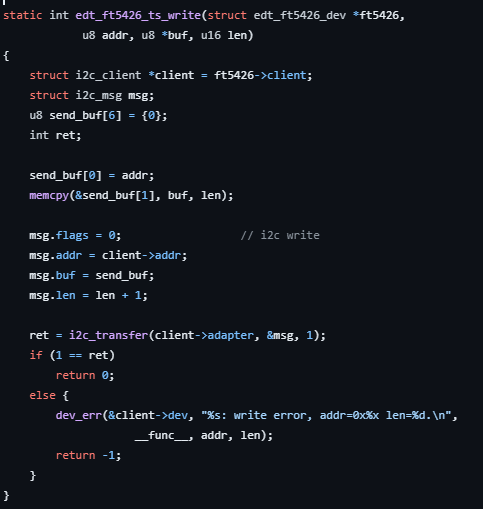


(圖四:Probe函式內容實現)

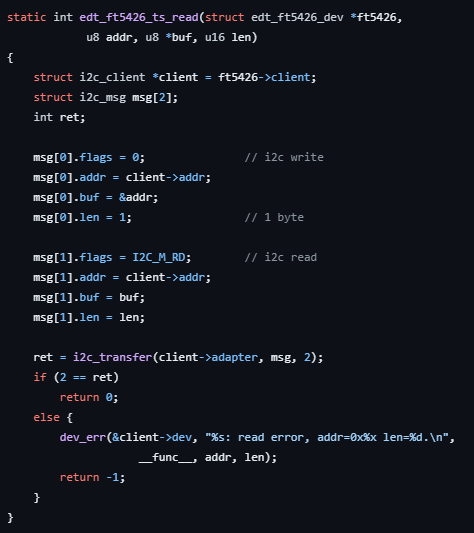
1. 接著完善，函式edt\_ft5426\_ts\_write 和 edt\_ft5426\_ts\_read，用於向 FT5426 觸摸控制器寫入和讀取數據。(如表六，圖五，圖六)

(表六: 函式edt\_ft5426\_ts\_write 和 edt\_ft5426\_ts\_read內部實現功能及解析)

|  |  |
| --- | --- |
| **步驟** | **說明** |
| **初始化 I2C 客戶端** | 從 ft5426 結構體中獲取 I2C 客戶端指標。 |
| **準備發送緩衝區** | 將目標寄存器地址和要寫入的數據複製到發送緩衝區 send\_buf。 |
| **設置 I2C 消息（寫入）** | 初始化 i2c\_msg 結構體，設置為寫入操作，設置目標地址和緩衝區。 |
| **發送 I2C 消息（寫入）** | 調用 i2c\_transfer 函數發送 I2C 消息，檢查返回值判斷是否成功。 |
| **設置 I2C 消息（讀取）** | 初始化兩個 i2c\_msg 結構體，一個用於寫入目標寄存器地址，另一個用於讀取數據。 |
| **發送 I2C 消息（讀取）** | 調用 i2c\_transfer 函數發送 I2C 消息，檢查返回值判斷是否成功。 |
| **錯誤處理** | 如果操作失敗，打印錯誤信息並返回錯誤代碼。 |



(圖五: edt\_ft5426\_ts\_write內容實現)

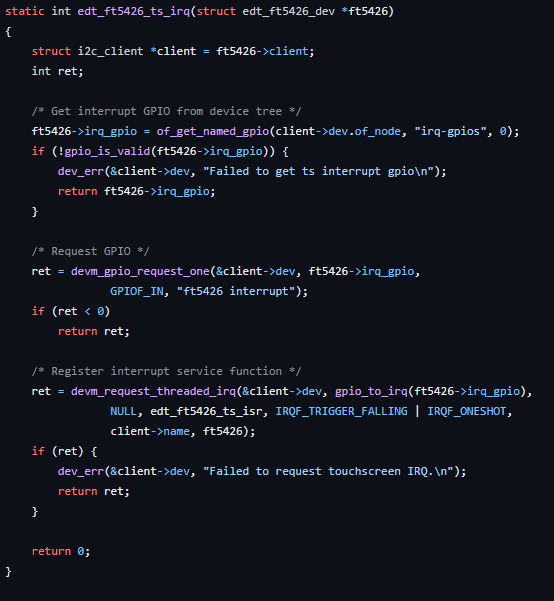


(圖六: edt\_ft5426\_ts\_read內容實現)

1. 實現edt\_ft5426\_ts\_irq 函數，用於設置和請求觸摸屏控制器 FT5426 的中斷。(如下表七，圖七)

(表七: edt\_ft5426\_ts\_irq 函數內部實現功能及解析)

|  |  |
| --- | --- |
| **步驟** | **說明** |
| 獲取中斷 GPIO | 從設備樹中獲取中斷 GPIO，如果獲取失敗，打印錯誤信息並返回錯誤代碼。 |
| 請求 GPIO | 請求中斷 GPIO，將其配置為輸入，如果請求失敗，返回錯誤代碼。 |
| 註冊中斷服務函數 | 註冊中斷服務函數 edt\_ft5426\_ts\_isr，設置中斷觸發類型為下降沿觸發和單次觸發，如果註冊失敗，返回錯誤代碼。 |
| 返回成功 | 如果所有操作成功，返回 0。 |



(圖七: edt\_ft5426\_ts\_irq 函數內部實現)

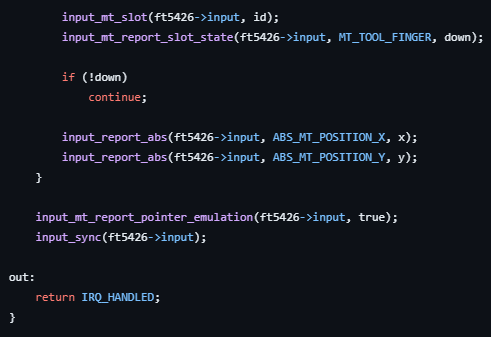
1. 定義 edt\_ft5426\_ts\_isr 函數，這是一個中斷服務函數（ISR），用於處理來自 FT5426 觸摸控制器的中斷事件。(如下表八，圖八，圖九)

(表八: edt\_ft5426\_ts\_isr 函數內部實現功能及解析)

|  |  |
| --- | --- |
| **步驟** | **說明** |
| **讀取觸摸數據** | 從 FT5426 的狀態寄存器 FT5426\_TD\_STATUS\_REG 開始讀取 29 個寄存器的數據。如果讀取失敗，跳轉到結束處理。 |
| **處理每個觸摸點** | 對於每個觸摸點，根據讀取到的數據解析觸摸事件類型、X 和 Y 軸坐標、觸摸點 ID。 |
| **檢查觸摸事件類型** | 檢查觸摸事件類型，如果事件類型為保留類型，則跳過該觸摸點。 |
| **解析觸摸點坐標** | 解析觸摸點的 X 和 Y 軸坐標。 |
| **解析觸摸點 ID** | 解析觸摸點的 ID，並判斷觸摸事件是否為按下事件。 |
| **報告觸摸點狀態** | 使用 input\_mt\_slot 和 input\_mt\_report\_slot\_state 函數報告觸摸點狀態。 |
| **報告觸摸點坐標** | 如果是按下事件，使用 input\_report\_abs 函數報告觸摸點的 X 和 Y 軸坐標。 |
| **模擬指標事件** | 使用 input\_mt\_report\_pointer\_emulation 函數模擬指標事件。 |
| **同步事件** | 使用 input\_sync 函數同步事件。 |
| **返回中斷已處理** | 返回 IRQ\_HANDLED 表示中斷已處理 |



(圖八: edt\_ft5426\_ts\_isr 函數內部實現)



(圖九: edt\_ft5426\_ts\_isr 函數內部實現)

1. edt\_ft5426\_ts\_reset 函數，用於對 FT5426 觸摸控制器進行Reset。

(如下表九，圖十)

(表九:edt\_ft5426\_ts\_reset 函數內部實現功能及解析)

|  |  |
| --- | --- |
| **步驟** | **說明** |
| **獲取復位 GPIO** | 從設備樹中獲取復位 GPIO，如果獲取失敗，打印錯誤信息並返回錯誤代碼。 |
| **請求 GPIO** | 請求復位 GPIO，將其配置為高電平輸出，如果請求失敗，返回錯誤代碼。 |
| **延遲 20 毫秒** | 延遲 20 毫秒，等待復位準備完成。 |
| **拉低復位 GPIO** | 將復位 GPIO 拉低，持續 5 毫秒，以觸發復位。 |
| **拉高復位 GPIO** | 將復位 GPIO 拉高，結束復位過程。 |
| **返回成功** | 如果所有操作成功，返回 0。 |



(圖十: edt\_ft5426\_ts\_reset 函數內部實現)

※總結:

整體開發流程如下表十，結合了I2C驅動開發，Input子系統和中斷處理的觀念，時做Multi-touch觸控螢幕的驅動，接著進行測式或debug就需要藉由tslib協助，而前面也已經說過如何啟用tslib。

(表十:Multi-touch驅動開發實作流程)

|  |  |
| --- | --- |
| **步驟** | **說明** |
| **包含必要的頭文件** | 包含 Linux 核心模塊開發所需的各種標頭文件。 |
| **定義常量和數據結構** | 定義 FT5426 的寄存器地址、支持的最大觸摸點數量和觸摸事件類型，定義 edt\_ft5426\_dev 結構體。 |
| **I2C 寫操作函數** | 定義 edt\_ft5426\_ts\_write 函數，用於向 FT5426 寄存器寫入數據。 |
| **I2C 讀操作函數** | 定義 edt\_ft5426\_ts\_read 函數，用於從 FT5426 寄存器讀取數據。 |
| **觸摸屏復位函數** | 定義 edt\_ft5426\_ts\_reset 函數，用於對 FT5426 觸摸控制器進行硬體復位。 |
| **中斷服務函數** | 定義 edt\_ft5426\_ts\_isr 函數，用於處理 FT5426 觸摸控制器的中斷事件。 |
| **請求中斷函數** | 定義 edt\_ft5426\_ts\_irq 函數，用於設置和請求觸摸屏控制器 FT5426 的中斷。 |
| **I2C 驅動程序探測函數** | 定義 edt\_ft5426\_ts\_probe 函數，初始化設備，包括內存分配、復位、設置寄存器、請求中斷、註冊輸入設備等。 |
| **I2C 驅動程序移除函數** | 定義 edt\_ft5426\_ts\_remove 函數，釋放資源，包括解除輸入設備的註冊。 |
| **設備樹兼容性表** | 定義 edt\_ft5426\_of\_match 表，用於匹配 FT5426 觸摸控制器的設備樹條目。 |
| **I2C 驅動結構** | 定義 edt\_ft5426\_ts\_driver 結構，包含驅動程序的基本信息、探測函數和移除函數。 |
| **模塊註冊** | 使用 module\_i2c\_driver 宏註冊 I2C 驅動程序 edt\_ft5426\_ts\_driver。 |
| **模塊信息** | 定義模塊的許可證、作者和其他信息。 |