專案32\_ADC:

* ADC(Analog to Digital Converter)中文名稱是模數轉換器。它可以將外部的模擬信號轉換成數字信號。對於GPIO口來說，高於某個電壓值，它讀出來的只有高電平，低於就是低電平。假如我想知道具體的電壓數值，就要借助於ADC的幫助，它可以將一個範圍內的電壓精確地讀取出來。(如表一整理)

(表一:ADC重要參數)

|  |  |
| --- | --- |
| **ADC參數** | **描述** |
| 測量範圍 | 測量範圍決定了外接設備信號輸出電壓範圍，不能超過 ADC 的測量範圍。 若不符合，需設計相關電壓轉換電路。 |
| 分辨率 | 最小測量刻度。例如 0-5V 測量範圍、12 位分辨率時，那麼我們能測出來的最小電壓就是 5V 除以 2 的 12 次方， 最小電壓為 0.00122V。分辨率越高，信號越精確。 |
| 精度 | 測量結果的準確度。在 12 位分辨率下的最小測量值是 0.00122V，但最高精度只能到 11 位即 0.00244V。 |
| 採樣時間 | ADC 採集外部電壓信號的瞬間保持電路。保持信號的時間即為採樣時間。 |
| 採樣率 | 每秒內採集的次數。採樣率越高越好，低採樣率可能丟失部分信息，是衡量 ADC 性能的重要指標。 |

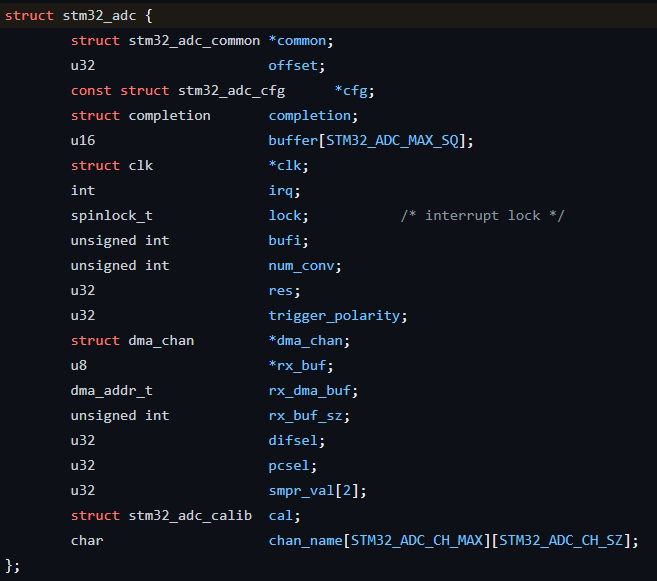
ADC驅動實作:

* ADC驅動部分ST已經幫我們寫好了，其中包含兩個部分。stm32-adc-core.c 和stm32-adc.c。stm32-adc-core.c是ADC核心層，主要用於ADC電源等初始化，我們重點著重的是stm32-adc.c這個文件。stm32-adc.c主體框架是 platform，配合IIO驅動框架實現ADC驅動。

1. ST自己將ADC外部設備抽象成了結構體，stm32\_adc就相當於自定義的設備結構體。(如下圖一，表二整理)

(表二: stm32\_adc結構體成員變數及功能整理)

|  |  |
| --- | --- |
| **成員變數名稱** | **功能描述** |
| common | 指向通用 ADC 配置結構體的指針 |
| offset | ADC 的偏移量 |
| cfg | 指向 ADC 配置的結構體 |
| completion | 用於同步的完成變量 |
| buffer | 保存 ADC 轉換結果的緩衝區 |
| clk | 指向時鐘的指針 |
| irq | 中斷號 |
| lock | 中斷鎖 |
| bufi | 緩衝區索引 |
| num\_conv | 轉換的次數 |
| res | 解析度 |
| trigger\_polarity | 觸發極性 |
| dma\_chan | 指向 DMA 通道的指針 |
| rx\_buf | 接收緩衝區的指針 |
| rx\_dma\_buf | DMA 接收緩衝區地址 |
| rx\_buf\_sz | 接收緩衝區大小 |
| difsel | 差分選擇 |
| pcsel | 預處理選擇 |
| smpr\_val | 採樣時間配置 |
| cal | 校準值 |
| chan\_name | 通道名稱 |

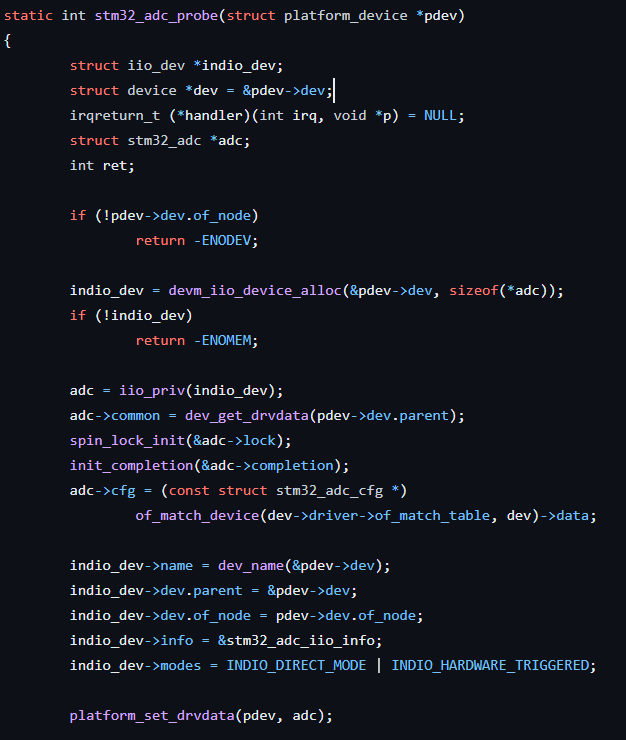


(圖一: stm32\_adc結構體內容)

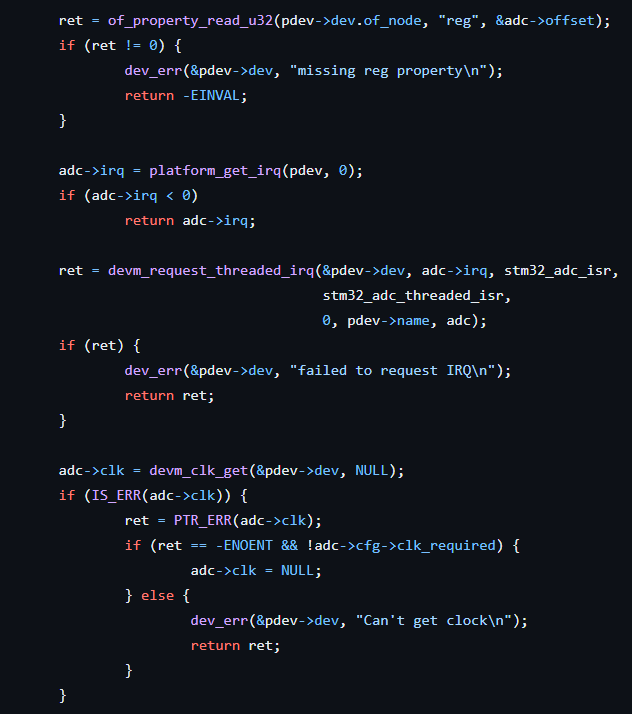
1. 接著是stm32-probe()函數實現(如下表三，圖二~圖五):

(表三:probe()函數內部程式碼解析)

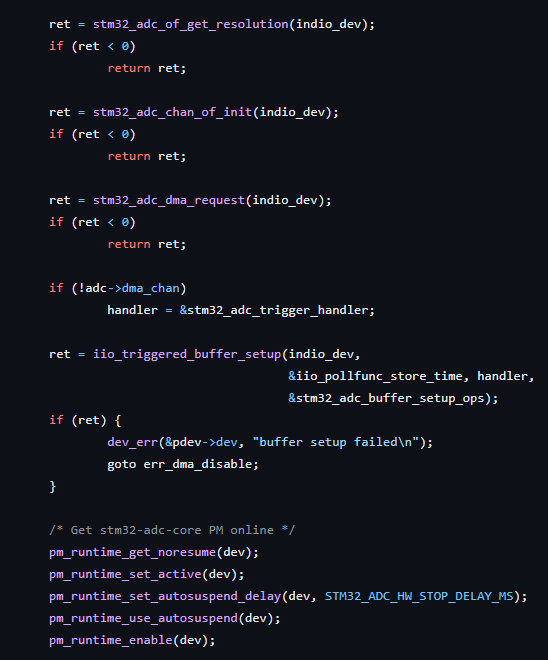
|  |  |  |
| --- | --- | --- |
| **步驟** | **功能描述** | **相關程式碼片段** |
| 1 | 檢查設備節點是否存在 | if (!pdev->dev.of\_node) return -ENODEV; |
| 2 | 分配並初始化 IIO 設備結構 | indio\_dev = devm\_iio\_device\_alloc(&pdev->dev, sizeof(\*adc)); if (!indio\_dev) return -ENOMEM; |
| 3 | 獲取和初始化 stm32\_adc 結構 | adc = iio\_priv(indio\_dev); adc->common = dev\_get\_drvdata(pdev->dev.parent); ... |
| 4 | 設置 IIO 設備屬性 | indio\_dev->name = dev\_name(&pdev->dev); indio\_dev->dev.parent = &pdev->dev; ... |
| 5 | 保存設備專用數據 | platform\_set\_drvdata(pdev, adc); |
| 6 | 讀取設備屬性 | ret = of\_property\_read\_u32(pdev->dev.of\_node, "reg", &adc->offset); ... |
| 7 | 獲取中斷號並請求中斷 | adc->irq = platform\_get\_irq(pdev, 0); ... ret = devm\_request\_threaded\_irq(&pdev->dev, adc->irq, ... |
| 8 | 獲取時鐘並初始化 | adc->clk = devm\_clk\_get(&pdev->dev, NULL); if (IS\_ERR(adc->clk)) { ... |
| 9 | 獲取 ADC 分辨率 | ret = stm32\_adc\_of\_get\_resolution(indio\_dev); if (ret < 0) return ret; |
| 10 | 初始化 ADC 通道 | ret = stm32\_adc\_chan\_of\_init(indio\_dev); if (ret < 0) return ret; |
| 11 | 請求 DMA | ret = stm32\_adc\_dma\_request(indio\_dev); if (ret < 0) return ret; |
| 12 | 設置 IIO 觸發緩衝區 | ret = iio\_triggered\_buffer\_setup(indio\_dev, &iio\_pollfunc\_store\_time, handler, ... |
| 13 | 電源管理設置 | pm\_runtime\_get\_noresume(dev); pm\_runtime\_set\_active(dev); ... |
| 14 | 啟動硬件 | ret = stm32\_adc\_hw\_start(dev); if (ret) goto err\_buffer\_cleanup; |
| 15 | 註冊 IIO 設備 | ret = iio\_device\_register(indio\_dev); if (ret) { dev\_err(&pdev->dev, "iio dev register failed\n"); ... |
| 16 | 錯誤處理和清理 | 在函數的末尾部分，有一系列的錯誤處理代碼，用於在初始化失敗時進行清理和釋放資源。 |



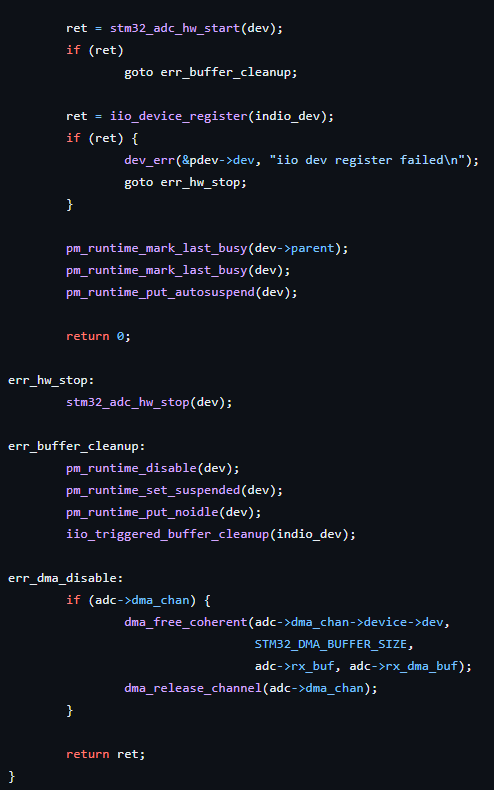
(圖二:probe()函式內容)



(圖三:probe()函式內容)

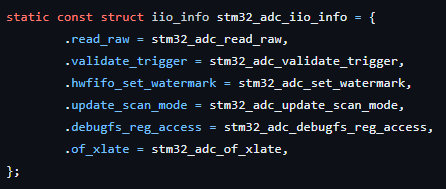


(圖四:probe()函式內容)

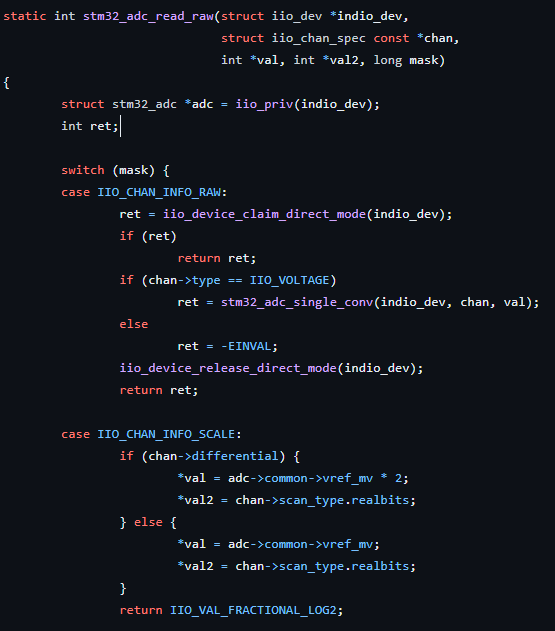


(圖五:probe()函式內容)

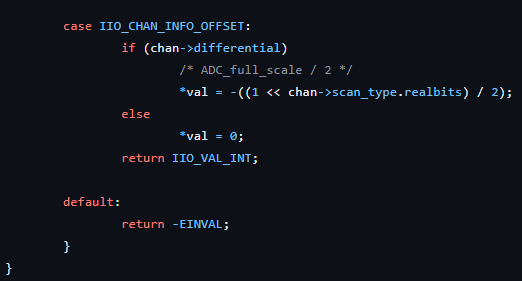
1. 最後解析stm32\_adc\_iio\_info結構體(如圖六)重點注意stm32\_adc\_read\_raw函式(如圖七，圖八)，他主要是對User Space發送數據(如表四整理)。



(圖六: stm32\_adc\_iio\_info結構體內容)



(圖七: stm32\_adc\_read\_raw函式內容)



(圖八: stm32\_adc\_read\_raw函式內容)

(表四: stm32\_adc\_read\_raw函式功能解析)

|  |  |
| --- | --- |
| **stm32\_adc\_read\_raw函式步驟** | **功能描述** |
| 讀取電壓值 | type 值為 IIO\_VOLTAGE，也就是讀取電壓值。 |
| 調用單次讀取函數 | 調用 stm32\_adc\_single\_conv 函數來完成 ADC 單次讀取。 |
| 設置採樣率與配置通道 | stm32\_adc\_single\_conv 函數會設置採樣率、 配置通道、使用硬體觸發、開啟轉換，最後等待轉換完成中斷發生。 |
| 返回分辨率 | 返回 ADC 對應的分辨率。 |
| 返回偏移值 | 返回差分 ADC 的偏移值。 |

※總結:

由於ST中ADC驅動程式非常的長由於篇幅原因，列出重點程式碼以及相關解析。