1. 專案22\_SPI:

* SPI是一種高速，全雙工，同步的通訊協議方式。在Linux底下SPI總線和I2C總線很類似，主機控制驅動和設備控制驅動的 分離思想來實現。所謂主機控制驅動就是指SOC的SPI控制器驅動，我們只需要注重在SPI設備驅動開發。
* 而在Linux Kernel中SPI總線框架也是分成三部分:

1. SPI核心層:

核心層是 Linux 的 SPI 子系統的核心代碼部分，提供了核心數據結構的定義、 SPI 控制器驅動和設備驅動的註冊、註銷、管理等 API。在 Linux 系統中，SPI 核心層的代碼位於 drivers/spi/spi.c。

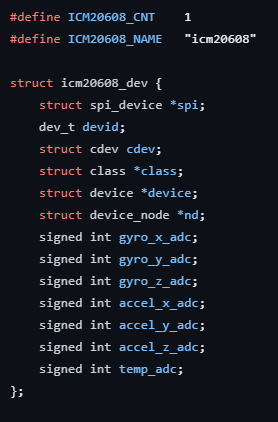
1. SPI控制器驅動層:

每種處理器平台都有自己的 SPI 控制器驅動程序，它的職責是為系統中的 SPI 總線實現相應的讀寫方法。每個控制器都有一條特定的 SPI 總線的讀寫。SPI 子系統使用 struct spi\_master 數據結構體來描述 SPI 控制器。

1. SPI設備驅動層:

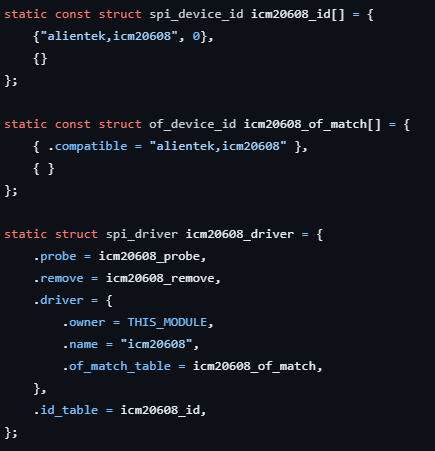
是SPI從設備的驅動，也就是SPI的外部設備。Linux內核使用spi\_driver結構體來表示 SPI 設備驅動，我們在編寫SPI設備驅動的時候需要實現spi\_driver。spi\_driver結構體定義在 include/linux/spi/spi.h文件中。

1. SPI驅動實作:
   1. 首先定義設備結構體，並宣告struct spi\_device成員變數 (如下圖一)。



(圖一:設備結構體內容)

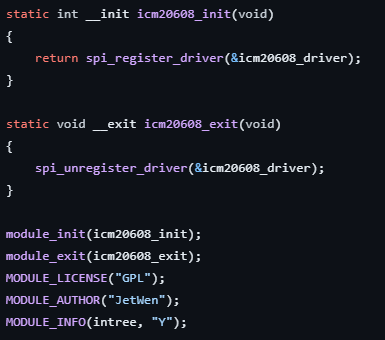
* 1. 接著定義id\_table匹配結構體，其中spi\_device\_id確保驅動程序可以在不使用Device Tree的系統上也正常工作。(如下圖二)。



(圖二:id\_table結構體內容)

* 1. 接著定義驅動出/入口函數，以及描述本驅動的相關訊息(如下圖三)。

其中驅動出/入口函數透過spi\_register\_driver()和spi\_unregister\_driver()函數，來向Linux Kernel註冊和註銷SPI總線。



(圖三:驅動出入口函數內容實現)

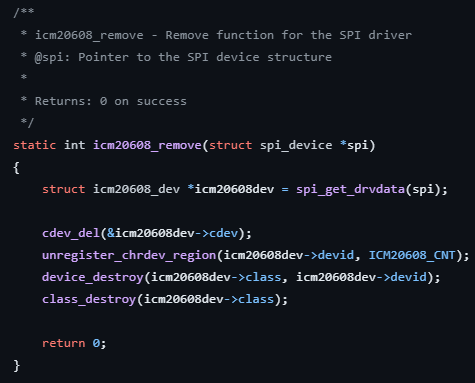
* 1. 再來根據spi\_driver及構體完善probe()函數(如下圖四，圖五)和remove()函數(如下圖六)。其中，透過probe()函除了呼叫icm20607\_reginit()來初始化設備之外，還透過spi\_set\_drvdata()函式將spi設備資料打包成private\_data。並且在remove()函式中使用spi\_get\_drvdata()獲取private\_data。



(圖四:probe()函式實現)

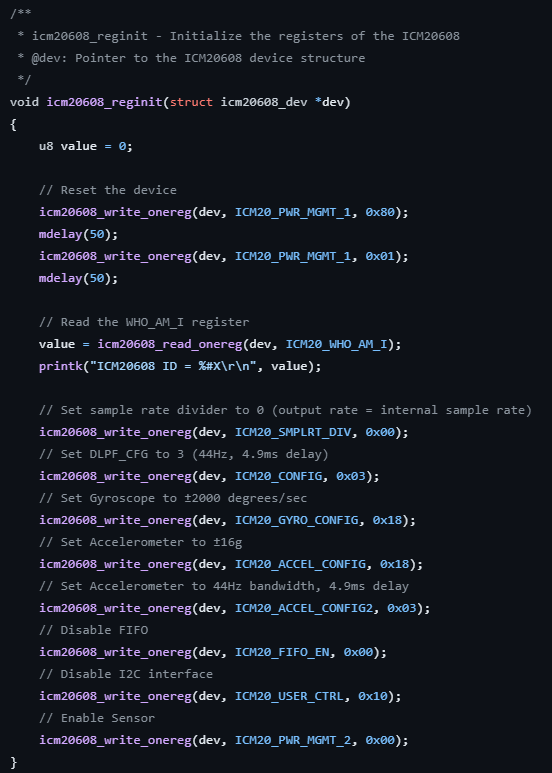


(圖五:probe()函式實現)



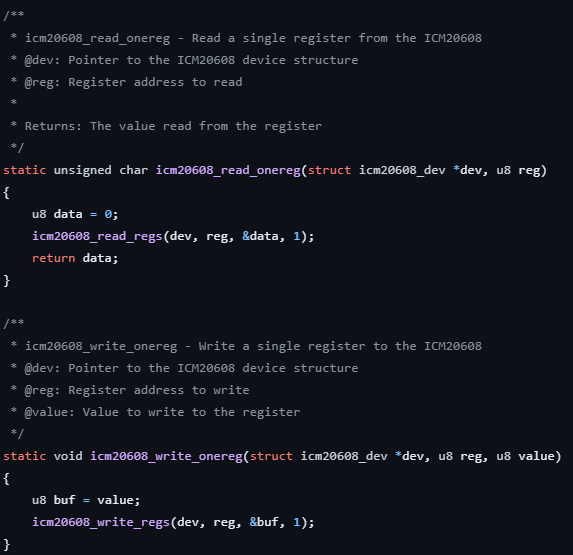
(圖六:remove()函式)

* 1. icm20608\_reginit()函數實現，就是參照DataSheet去設定內部暫存器



(圖七: icm20608\_reginit()函數內容)

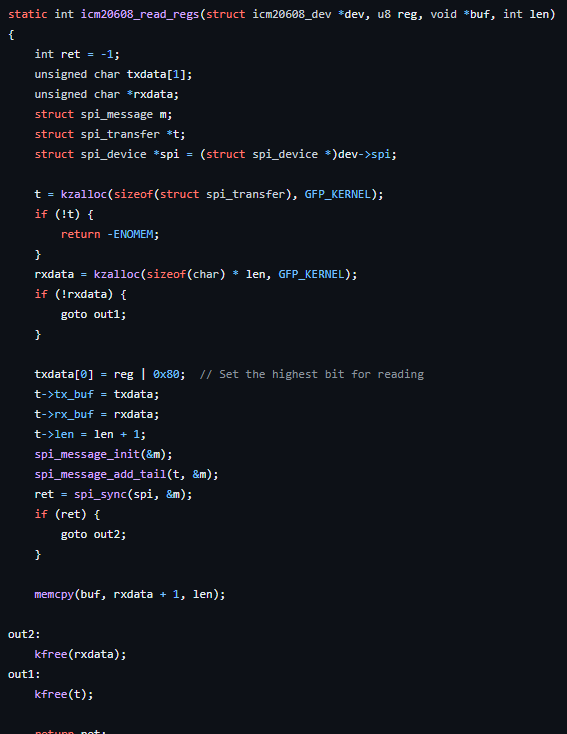
* 1. 透過icm20608\_read\_onereg()去讀取單一暫存器，再透過呼叫icm20608\_read\_regs()對多個暫存器進行讀取動作。接著，透過icm20608\_write\_onereg()去讀取單一暫存器，再透過呼叫icm20608\_write\_regs()對多個暫存器進行讀取動作(如下圖七，圖八，圖九)。



(圖七: icm20608\_read\_onereg()和icm20608\_write\_onereg()函式內容)



(圖八:icm20608\_write\_regs()函式內容)



(圖九:icm20608\_read\_regs()函式內容)

* 1. 最後實現file\_operations結構體中的成員函式(如下圖十)



(圖十:file\_operations結構體成員函式)

※結論:

整體流程與I2C總線驅動框架差不多，設備結構體中定義成員變數，file\_operations定義read()，write()，release()函式，再透過DataSheet去編寫讀取和寫入暫存器的函式。最後在透過id\_table匹配驅動和設備，匹配完成後編寫probe()和remove()函式，以及驅動出/入口函式向Linux Kernel註冊/註銷SPI總線就大功告成了。