PWM驅動開發:

* PWM是一個很常用到的功能，我們可以通過PWM來控制電機速度，也可以使用PWM來控制LCD的背光亮度。這裡將展示Linux系統下PWM驅動開發。
* PWM 信號有兩個關鍵的術語：頻率和占空比。頻率就是開關速度，把一次開關算作一個周期，那麼頻率就1秒內進行了多少次開關。占空比就是一個周期內高電位時間和低電位時間的比例，一個周期內高電平時間越長占空比就越大，反之占空比就越小。

PWM子系統:

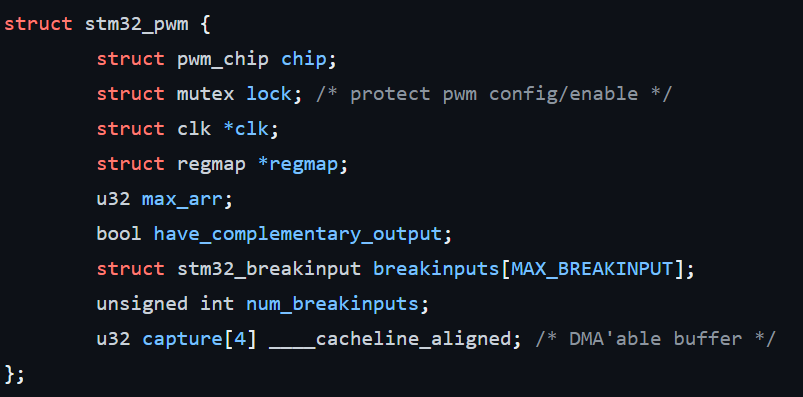
* Linux內核提供了一個PWM子系統框架，編寫PWM驅動的時候一定要符合這個框架。PWM子系統的核心是pwm\_chip結構體，定義在文件include/linux/pwm.h中。
* 其中，特別重要的是pwm\_ops結構體，它是PWM外設的各種操作函數集合，編寫PWM外設驅動的時候需要開發人員來實現。
* PWM子系統驅動的核心是初始化pwm\_chip結構體，然後向Kernel註冊初始化完成後的pwm\_chip。註冊需要透過pwmchip\_add函式。(如下表一)
* 卸載PWM驅動的時候需要將前面註冊的pwm\_chip從內核移除，這裡要用到pwmchip\_remove函數。(如下表一)

(表一: pwmchip\_remove及pwmchip\_add功能整理)

|  |  |
| --- | --- |
| **函數** | **功能** |
| pwmchip\_add | 將初始化完成的 pwm\_chip 結構體註冊到Kernel中 |
| pwmchip\_remove | 將已註冊的 pwm\_chip 從Kernel中移除 |

PWM驅動實作:

* PWM驅動廠商ST已寫好了，這裡就欣賞一下別人的寫法。
  1. 首先依舊是定義設備結構體(如圖一，表一)。

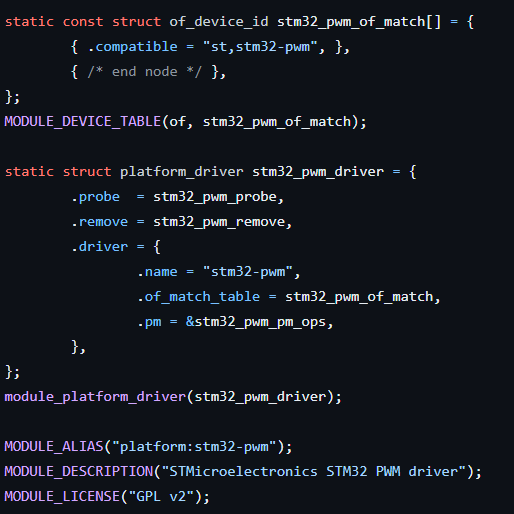


(圖一:設備結構體內成員變數)

(表一:設備結構體成員變數功能整理)

|  |  |  |
| --- | --- | --- |
| **成員名稱** | **類型** | **描述** |
| chip | struct pwm\_chip | PWM 子系統的核心結構體 |
| lock | struct mutex | 用於保護 PWM 配置和使能的互斥鎖 |
| clk | struct clk \* | 指向時鐘結構體的指針 |
| regmap | struct regmap \* | 指向寄存器映射結構體的指針 |
| max\_arr | u32 | 最大自動重裝載值 |
| have\_complementary\_output | bool | 是否有互補輸出 |
| breakinputs | struct stm32\_breakinput[MAX\_BREAKINPUT] | 中斷輸入數組 |
| num\_breakinputs | unsigned int | 中斷輸入的數量 |
| capture | u32[4] | 可供 DMA 使用的緩衝區，並且以快取行對齊 (\_\_\_\_cacheline\_aligned) |

* 1. 接著定義註冊platform子系統給Kernel以及ID\_Table(如下圖二)

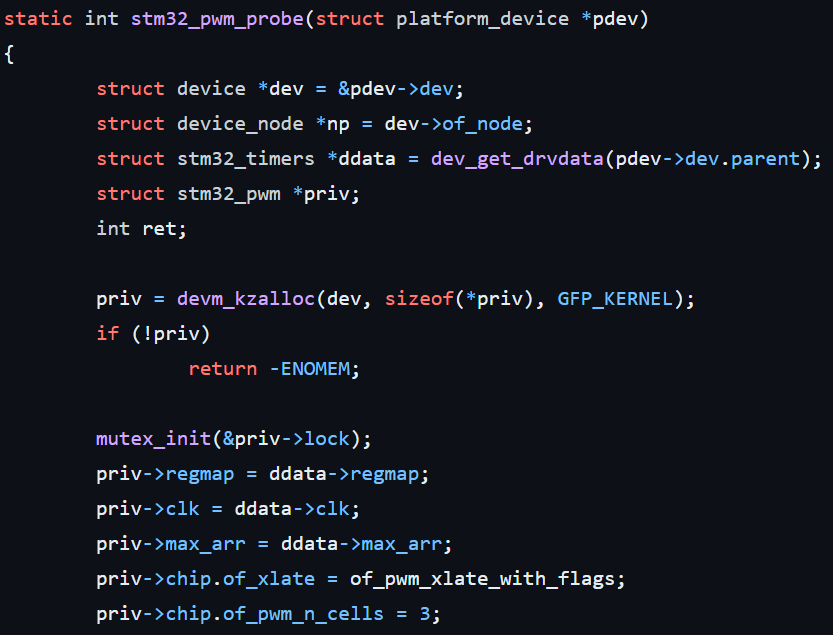


(圖二:ID\_Table和platform子系統註冊)

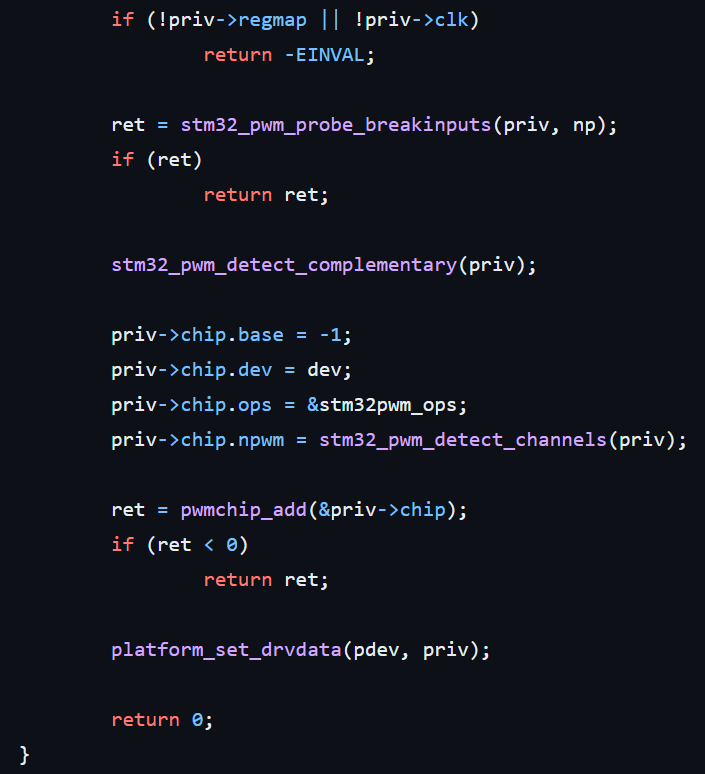
* 1. 緊接著實現圖二中綁定的所有函數stm32\_pwm\_probe，stm32\_pwm\_remove。



(圖三:remove函數內容)

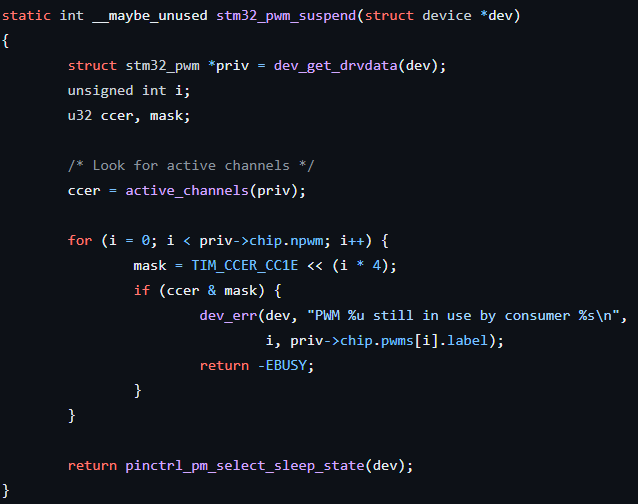


(圖四:probe函式內容)

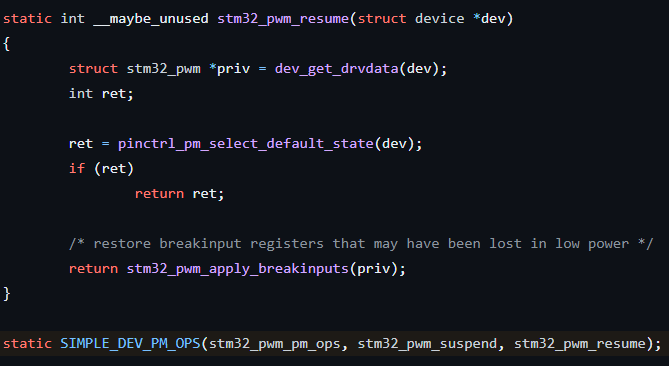


(圖五:probe函式內容)

* 1. 定義一個簡單的設備電源管理操作結構體，用於PWM驅動的電源管理（如掛起和恢復操作）。具體來說，這行程式碼使用了 SIMPLE\_DEV\_PM\_OPS宏來創建stm32\_pwm\_pm\_ops結構體，該結構體包含了兩個函數指針：一個指向掛起操作函數（stm32\_pwm\_suspend），另一個指向恢復操作函數（stm32\_pwm\_resume）。

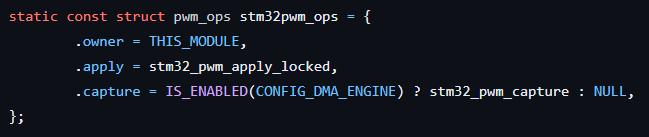


(圖六:stm32\_pwm\_suspend函數內容)



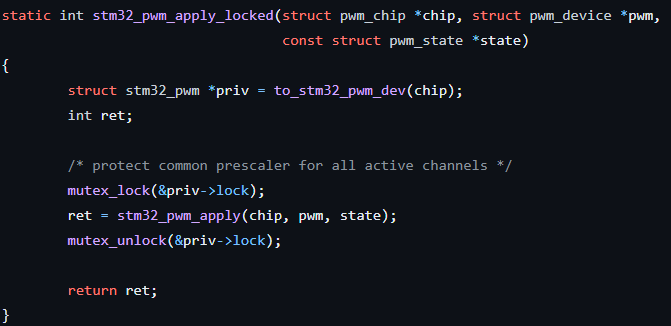
(圖七: stm32\_pwm\_resume函數內容)

* 1. 定義pwm\_ops操作結構體，這個結構體是用來配置一切PWM設備的。(如下圖八)



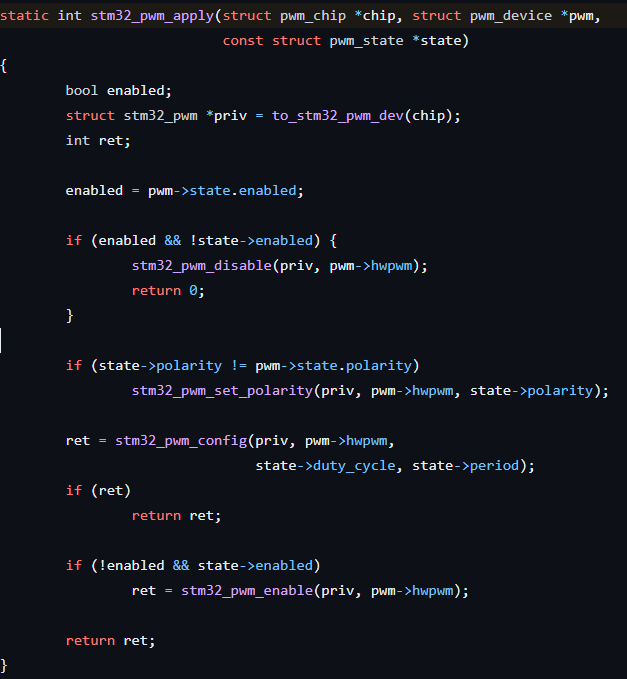
(圖八:PWM\_OPS操作結構體)

* 其中，apply是設置和應用 PWM 設備的配置其綁定函式為stm32\_pwm\_apply\_locked函式。(如下圖九)



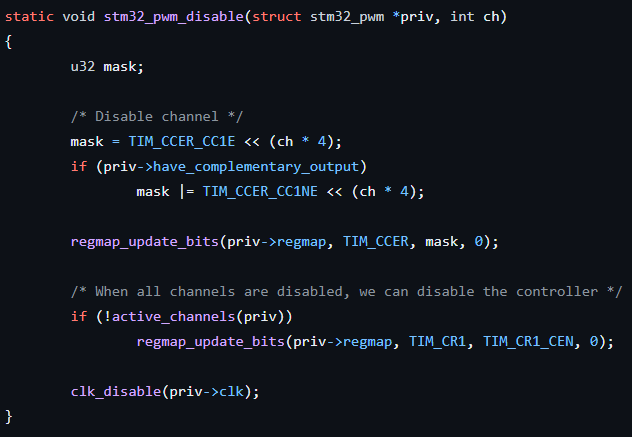
(圖九: stm32\_pwm\_apply\_locked函式內容)

* 而在stm32\_pwm\_apply\_locked函式中實際式透過stm32\_pwm\_apply()函式執行PWM配置的，主要功能是根據新的PWM狀態更新PWM設備的配置。如:禁用或使能PWM設備，設置PWM的極性(如下圖十)

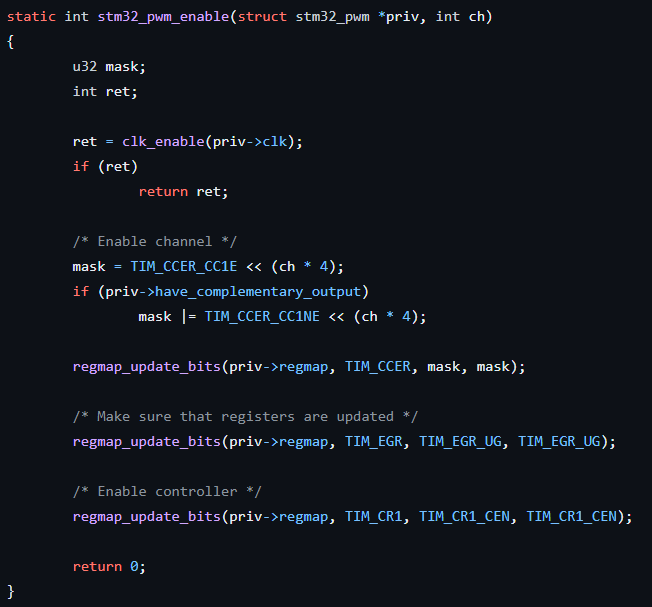


(圖十:stm32\_pwm\_apply函式內容)

* stm32\_pwm\_apply函式中禁用或使能PWM設備，是透過stm32\_pwm\_disable和stm32\_pwm\_enable()函式完成的。(如圖十一，圖十二)



(圖十一: stm32\_pwm\_disable函式內容)



(圖十二: stm32\_pwm\_ensable函式內容)

※總結:

PWM開發流程就是如同上述，其餘部分就是DMA和TIM暫存器的設置。和本章主題Linux系統PWM開發較無關，此處就不再贅述了。