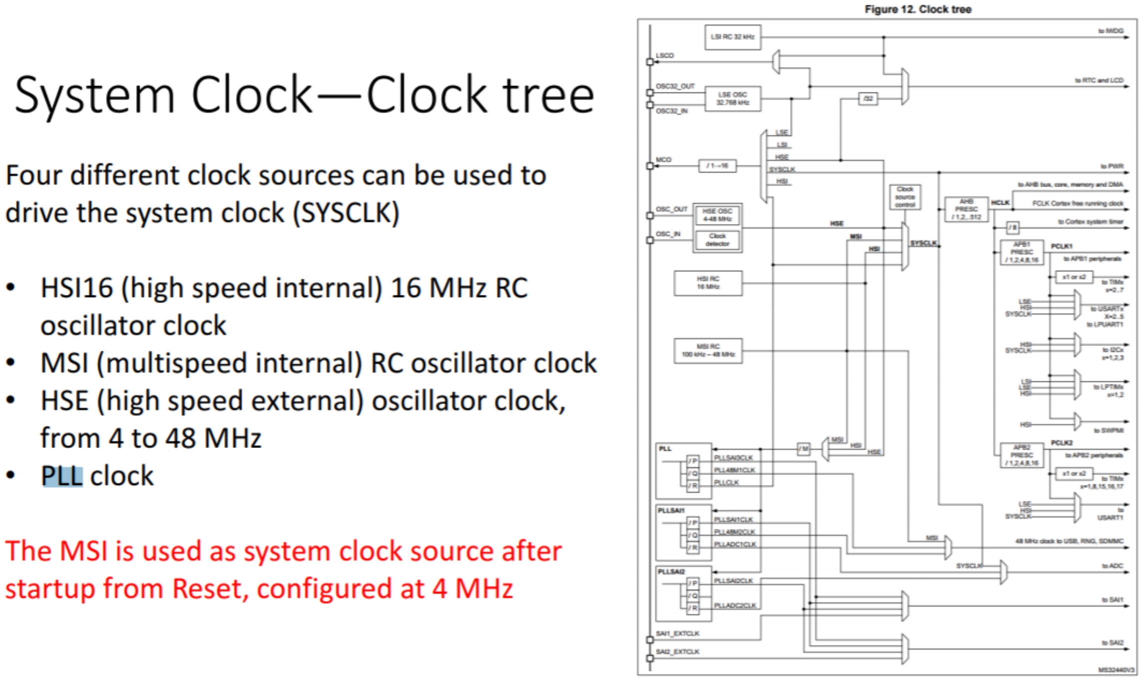
Lab6 STM32 Clock and Timer

1. Lab objectives 實驗目的

- 瞭解 STM32 的各種 clock source 使用與修改
- 瞭解 STM32 的 timer 使用原理
- 瞭解 STM32 的 PWM 使用原理與應用

2. Lab theory 實驗原理

2.1. Clock



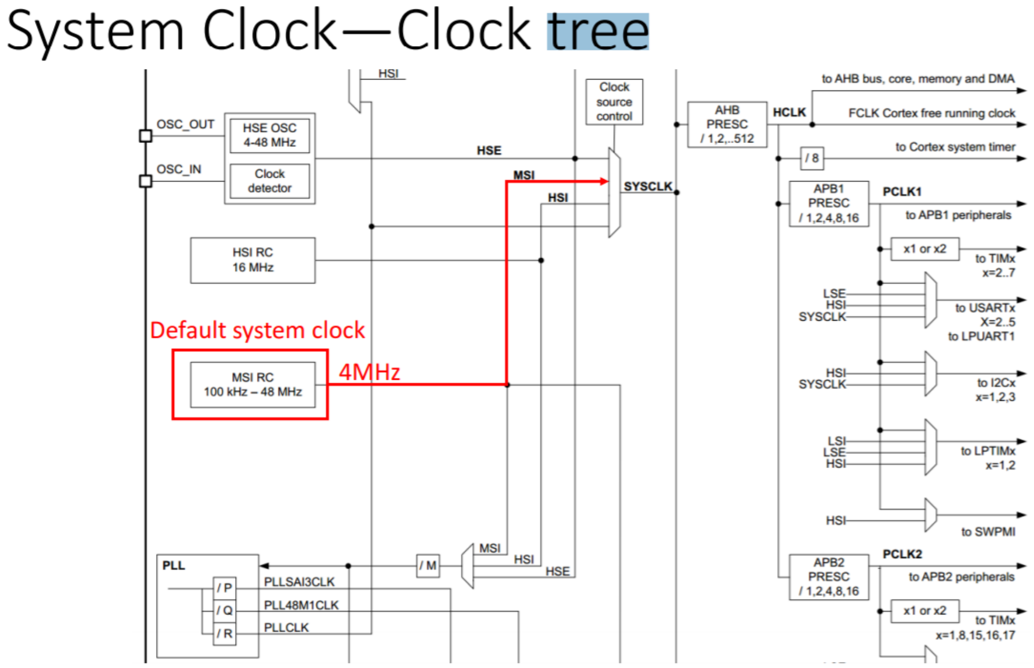
System Clock—Clock tree

Four different clock sources can be used to drive the system clock (SYSCLK)

- HSI16 (high speed internal) 16 MHz RC oscillator clock
- MSI (multispeed internal) RC oscillator clock
- HSE (high speed external) oscillator clock, from 4 to 48 MHz
- PLL clock

The MSI is used as system clock source after startup from Reset, configured at 4 MHz

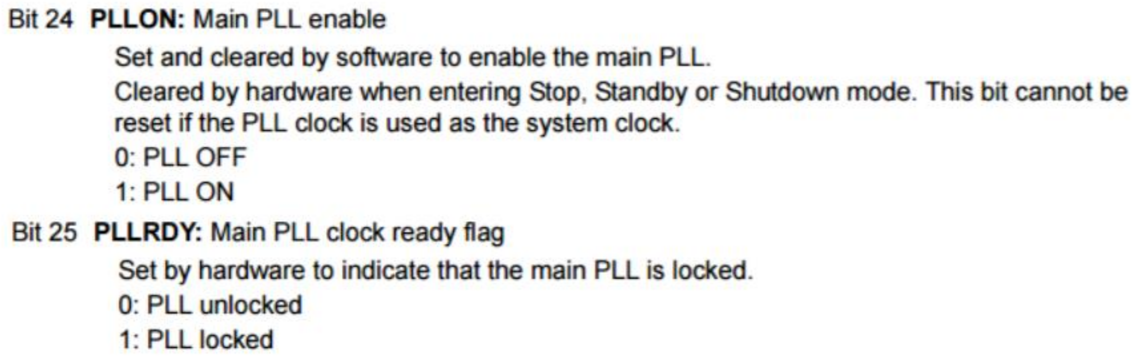
Figure 12. Clock tree



System Clock—Clock tree

RCC_CR

- XXXON:開啟哪一種系統 clock，4 選 1(MSI,HSI,HSE,PLL)
- XXXRDY:狀態回報，是否已開啟完成
- 其中 PLL 還需要再開其他 3 個 clock 之一，當作參考源。



Bit 24 **PLLON:** Main PLL enable

Set and cleared by software to enable the main PLL.

Cleared by hardware when entering Stop, Standby or Shutdown mode. This bit cannot be reset if the PLL clock is used as the system clock.

0: PLL OFF

1: PLL ON

Bit 25 **PLLRDY:** Main PLL clock ready flag

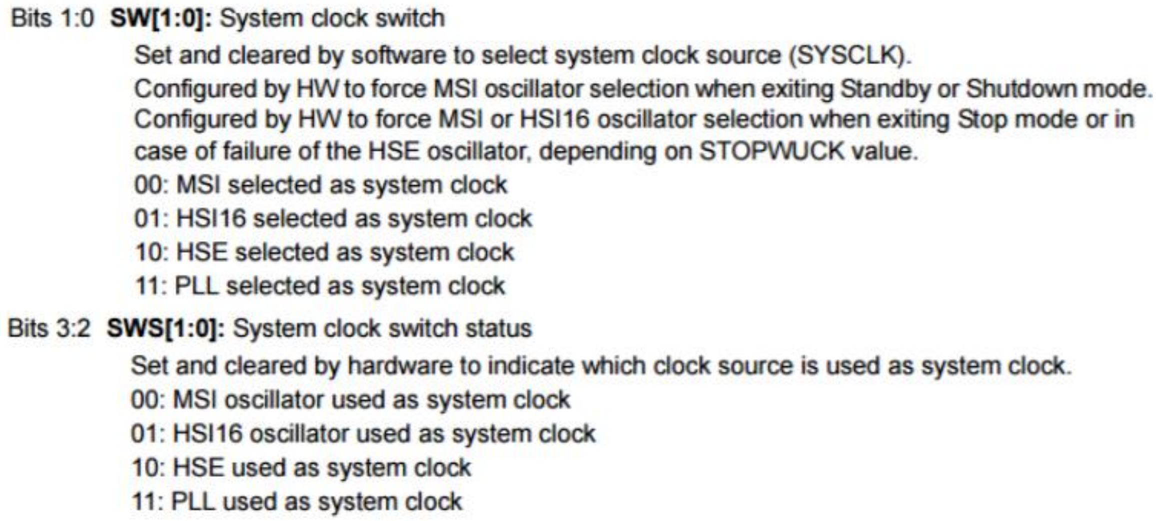
Set by hardware to indicate that the main PLL is locked.

0: PLL unlocked

1: PLL locked

RCC_CFGR

- SW:指定系統 clock 是用哪一種 config，4 選 1
- SWS:狀態回報，是否已指定完成



Bits 1:0 **SW[1:0]:** System clock switch

Set and cleared by software to select system clock source (SYSCLK).

Configured by HW to force MSI oscillator selection when exiting Standby or Shutdown mode.

Configured by HW to force MSI or HSI16 oscillator selection when exiting Stop mode or in case of failure of the HSE oscillator, depending on STOPWUCK value.

00: MSI selected as system clock

01: HSI16 selected as system clock

10: HSE selected as system clock

11: PLL selected as system clock

Bits 3:2 **SWS[1:0]:** System clock switch status

Set and cleared by hardware to indicate which clock source is used as system clock.

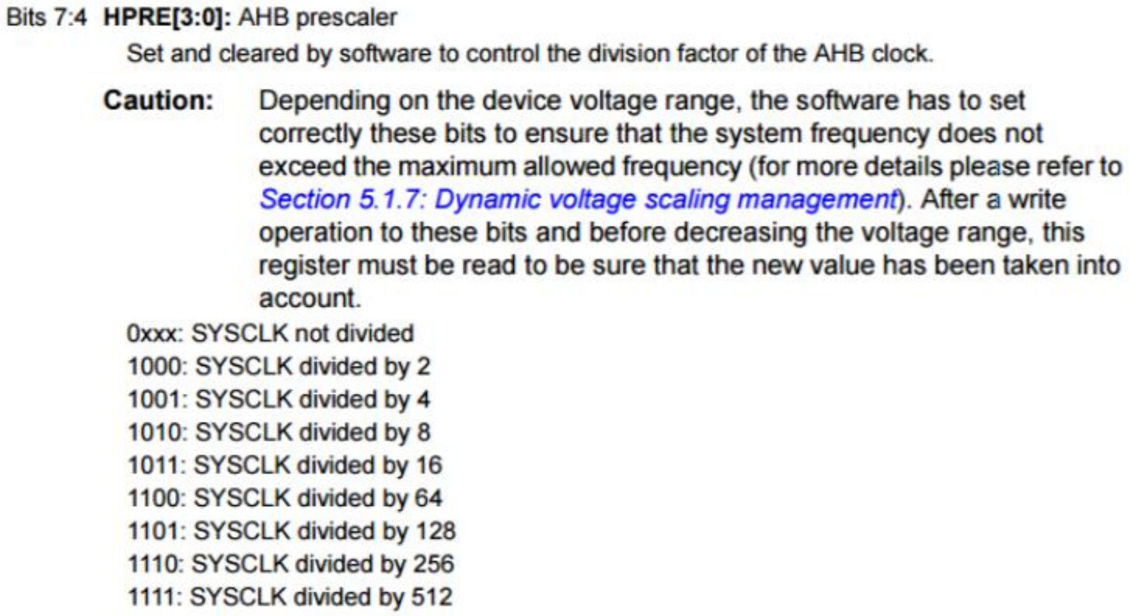
00: MSI oscillator used as system clock

01: HSI16 oscillator used as system clock

10: HSE used as system clock

11: PLL used as system clock

- HPRE: prescaler，選擇倍率



Bits 7:4 **HPRE[3:0]:** AHB prescaler

Set and cleared by software to control the division factor of the AHB clock.

Caution: Depending on the device voltage range, the software has to set correctly these bits to ensure that the system frequency does not exceed the maximum allowed frequency (for more details please refer to *Section 5.1.7: Dynamic voltage scaling management*). After a write operation to these bits and before decreasing the voltage range, this register must be read to be sure that the new value has been taken into account.

0xxx: SYSCLK not divided
1000: SYSCLK divided by 2
1001: SYSCLK divided by 4
1010: SYSCLK divided by 8
1011: SYSCLK divided by 16
1100: SYSCLK divided by 64
1101: SYSCLK divided by 128
1110: SYSCLK divided by 256
1111: SYSCLK divided by 512

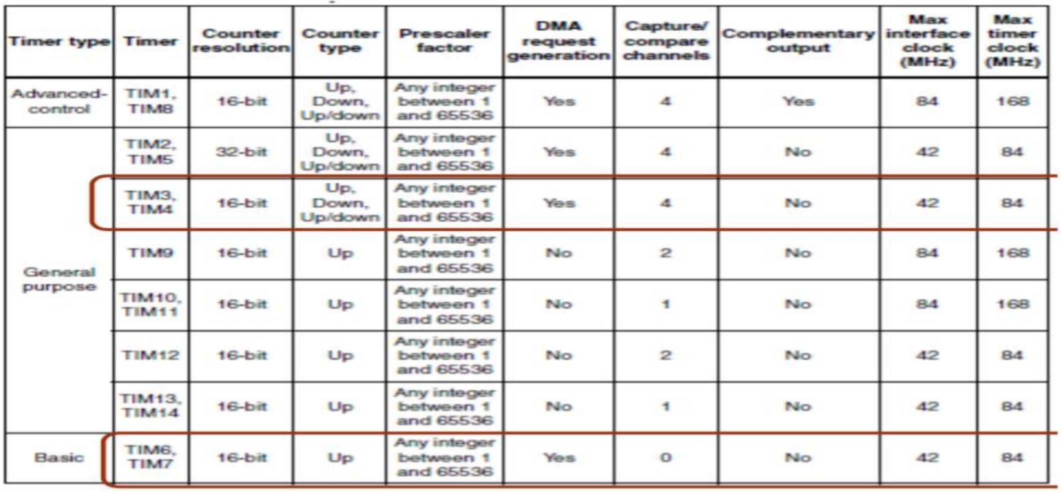
如何開啟一個 clock:

1. RCC->CR 四選一個，並等待 RDY

2. RCC->CFGR 的 SW 一樣四選一，並等待 SWS。

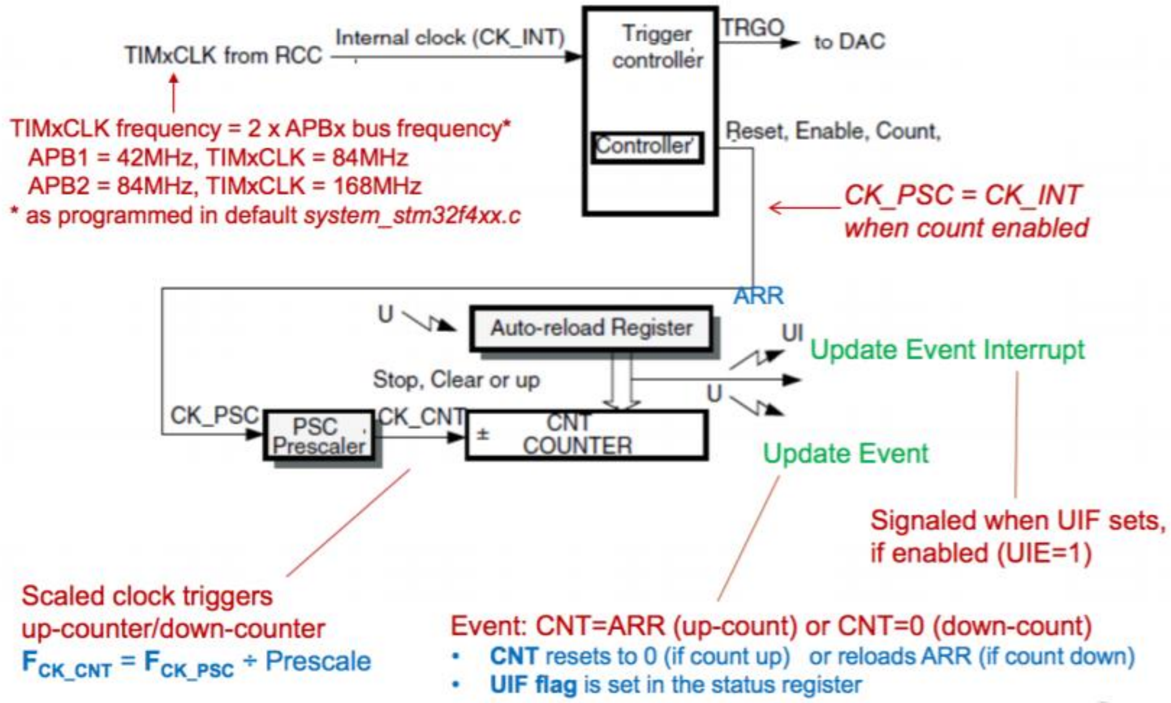
3. 設定 RCC->CFGR 的 HPRE，如果要更動倍率的話。

2.2. Timer and Counter：Timer 就是利用 counter 數經過幾次 clock 後，做出一些反應的設計。Timer 有很多種：



Timer type	Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/ compare channels	Complementary output	Max interface clock (MHz)	Max timer clock (MHz)
Advanced-control	TIM1, TIM8	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	Yes	84	168
General purpose	TIM2, TIM5	32-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	42	84
	TIM3, TIM4	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	42	84
	TIM9	16-bit	Up	Any integer between 1 and 65536	No	2	No	84	168
	TIM10, TIM11	16-bit	Up	Any integer between 1 and 65536	No	1	No	84	168
	TIM12	16-bit	Up	Any integer between 1 and 65536	No	2	No	42	84
	TIM13, TIM14	16-bit	Up	Any integer between 1 and 65536	No	1	No	42	84
Basic	TIM6, TIM7	16-bit	Up	Any integer between 1 and 65536	Yes	0	No	42	84

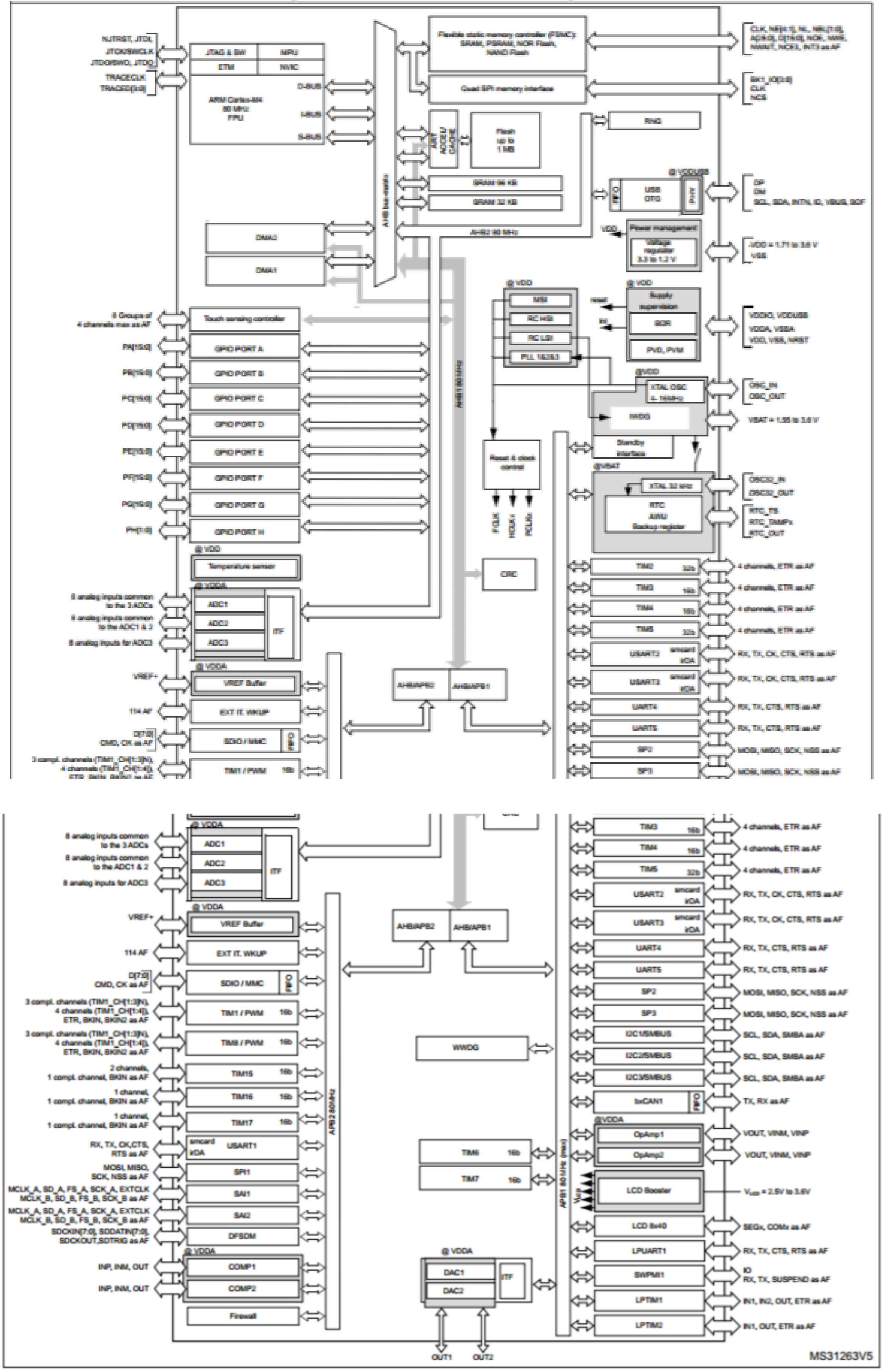
最基本的 Timer 6/7 也是需要設定 CR、ARR、PSC。


TIMxCLK from RCC
Internal clock (CK_INT)
Trigger controller
TRGO
to DAC
Controller
Reset, Enable, Count,
TIMxCLK frequency = 2 x APBx bus frequency*
APB1 = 42MHz, TIMxCLK = 84MHz
APB2 = 84MHz, TIMxCLK = 168MHz
* as programmed in default system_stm32f4xx.c
CK_PSC = CK_INT
when count enabled
ARR
U
Auto-reload Register
UI
Update Event Interrupt
Stop, Clear or up
U
CK_PSC
PSC Prescaler
CK_CNT
± CNT COUNTER
Update Event
Signaled when UIF sets,
if enabled (UIE=1)
Scaled clock triggers
up-counter/down-counter
$F_{CK\_CNT} = F_{CK\_PSC} \div$ Prescale
Event: CNT=ARR (up-count) or CNT=0 (down-count)
• CNT resets to 0 (if count up) or reloads ARR (if count down)
• UIF flag is set in the status register

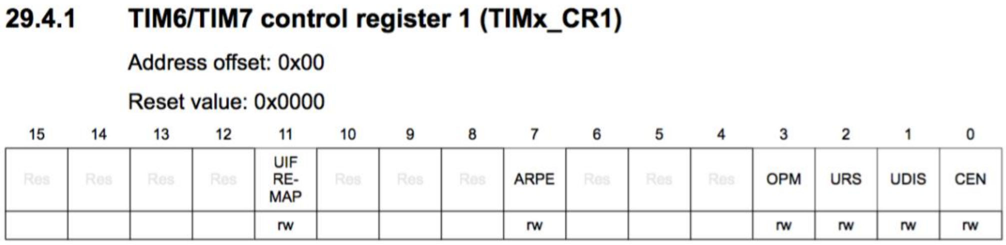
3 種 clock 狀態：原本板子 clock (CK_INT)、 開啟 enable 後 clock (CK_PSC)，最後除頻後，變成你用的所參考的 counter (CK_CNT)。

Timer 大多定義在 AHB1APB1 下，記得去 Enable。

Figure 1. STM32L476xx block diagram



- TIMx_CR1 (timer control reg)：Timer 設定控制



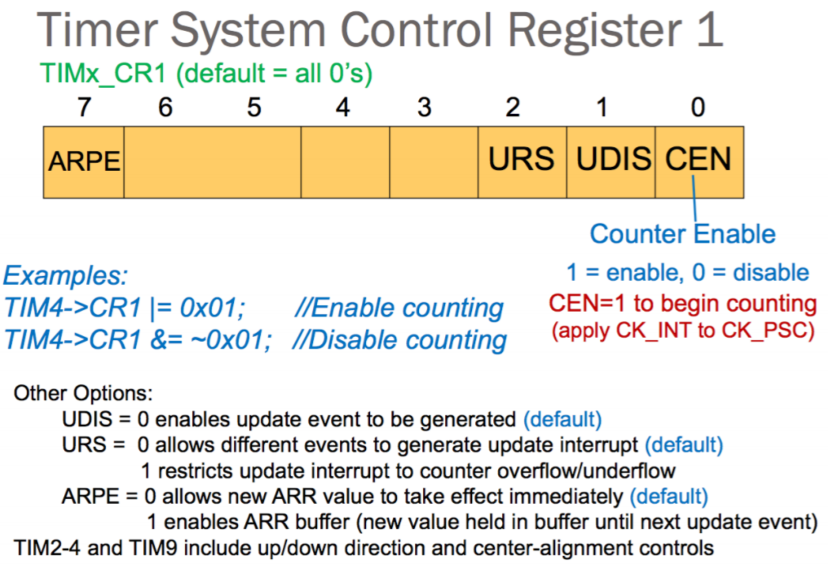
29.4.1 TIM6/TIM7 control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	UIF RE-MAP	Res	Res	Res	ARPE	Res	Res	Res	OPM	URS	UDIS	CEN
				rw				rw				rw	rw	rw	rw

Example:



Timer System Control Register 1

TIMx_CR1 (default = all 0’s)

7	6	5	4	3	2	1	0
ARPE					URS	UDIS	CEN

Counter Enable

1 = enable, 0 = disable

CEN=1 to begin counting

(apply CK_INT to CK_PSC)

Examples:

```
TIM4->CR1 |= 0x01;      //Enable counting
TIM4->CR1 &= ~0x01;  //Disable counting
```

Other Options:

UDIS = 0 enables update event to be generated (default)

URS = 0 allows different events to generate update interrupt (default)

1 restricts update interrupt to counter overflow/underflow

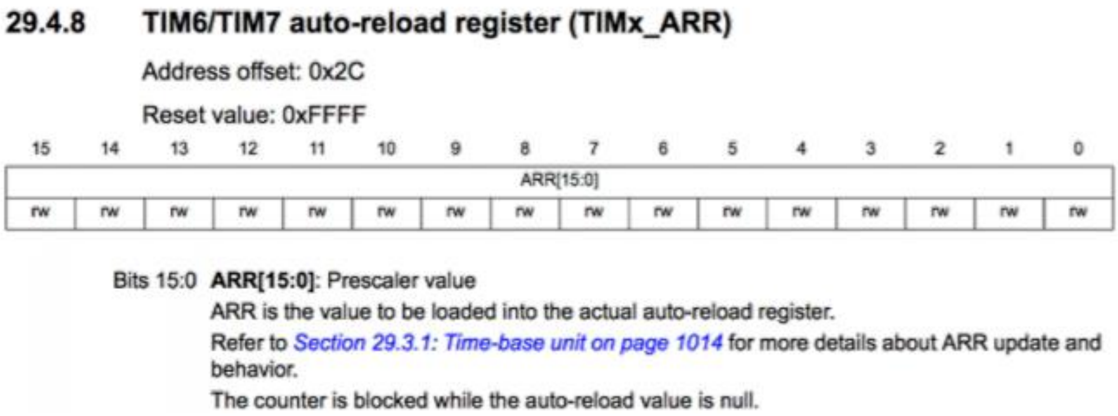
ARPE = 0 allows new ARR value to take effect immediately (default)

1 enables ARR buffer (new value held in buffer until next update event)

TIM2-4 and TIM9 include up/down direction and center-alignment controls

---

- TIMx_ARR (auto reload reg)：看數到多少要給 event。



29.4.8 TIM6/TIM7 auto-reload register (TIMx_ARR)

Address offset: 0x2C

Reset value: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

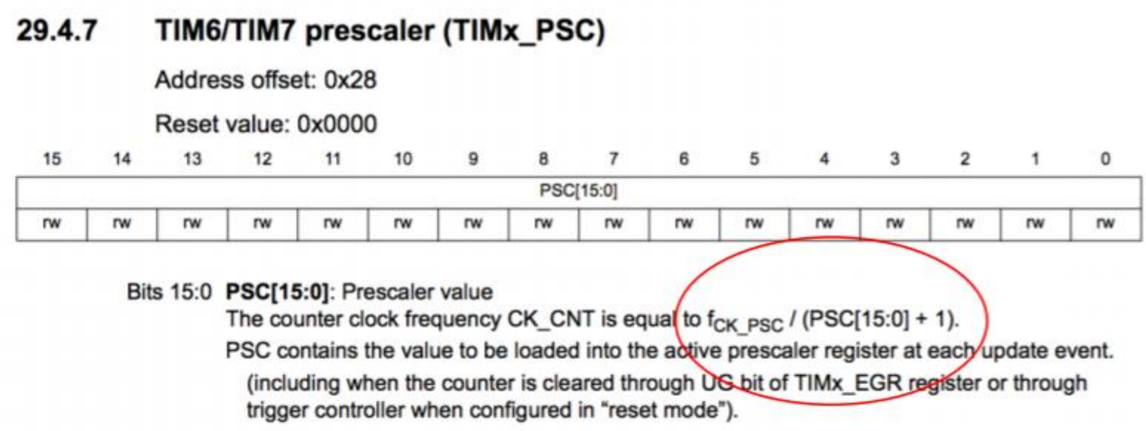
Bits 15:0 **ARR[15:0]**: Prescaler value

ARR is the value to be loaded into the actual auto-reload register.

Refer to *Section 29.3.1: Time-base unit on page 1014* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

- TIMx_PSC (Pre-scaler reg)：把原時脈做除頻 or 增頻(直接給值)



29.4.7 TIM6/TIM7 prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

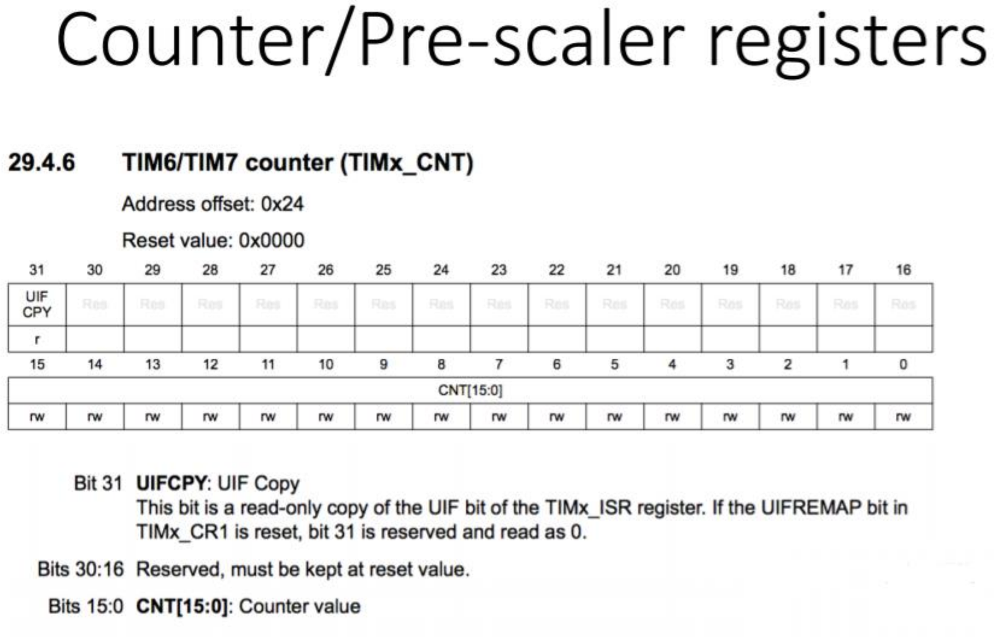
Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency CK_CNT is equal to f_{CK_PSC} / (PSC[15:0] + 1).

PSC contains the value to be loaded into the active prescaler register at each update event.

(including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

- TIMx_CNT：只是讓你知道現在數到哪而已。



Counter/Pre-scaler registers

29.4.6 TIM6/TIM7 counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
r															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy

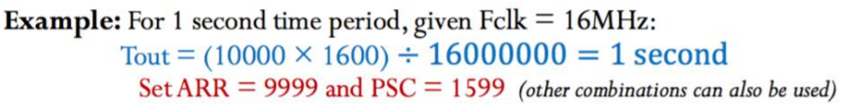
This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in TIMx_CR1 is reset, bit 31 is reserved and read as 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

如何開啟一個 timer:

1. RCC 的 APB1ENR1 設定成 TIMxEN，打開 TIMx。
2. 設定 TIMx 的 PSC 值，他決定多少 cycle 加一次 counter
3. 設定 TIMx 的 ARR 值，他決定 timer 加到多少 cycle 會重數



Example: For 1 second time period, given Fclk = 16MHz:

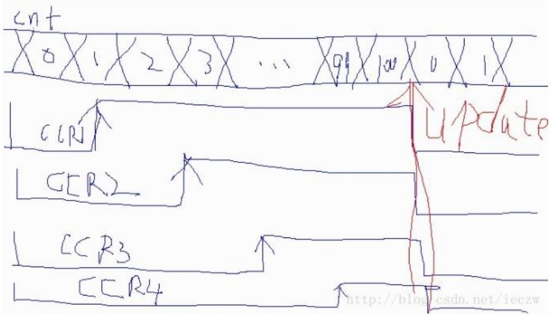
Tout = (10000 × 1600) ÷ 16000000 = 1 second

Set ARR = 9999 and PSC = 1599 *(other combinations can also be used)*

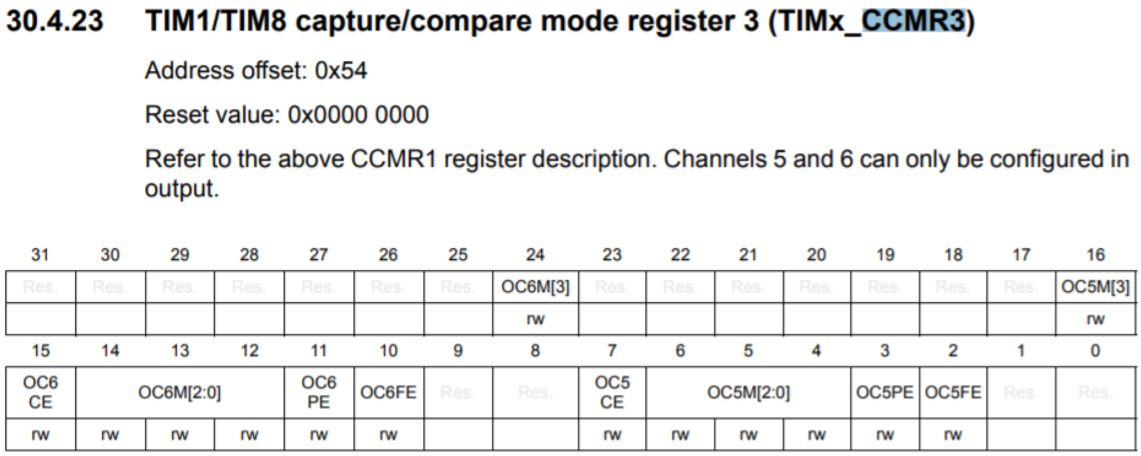
4. timer->EGR = TIM_EGR_UG :重新初始化
5. Start: TIMx->CR1 |= TIM_CR1_CEN; 設定 TIMx 的 CR1 變成 CEN

如何設定一個 PWM:

1. CCR1~4:一個 timer 能分出 4 個不同 duty cycle 的 channel。CCR 的值須與所用 timer 的 ARR 配合。



2. CCMR1~2: 控制輸出波的設定。TIMx_CCMR1 控制 CH1 和 CH2，而 TIMx_CCMR2 控制 CH3 和 CH4



30.4.23 TIM1/TIM8 capture/compare mode register 3 (TIMx_CCMR3)

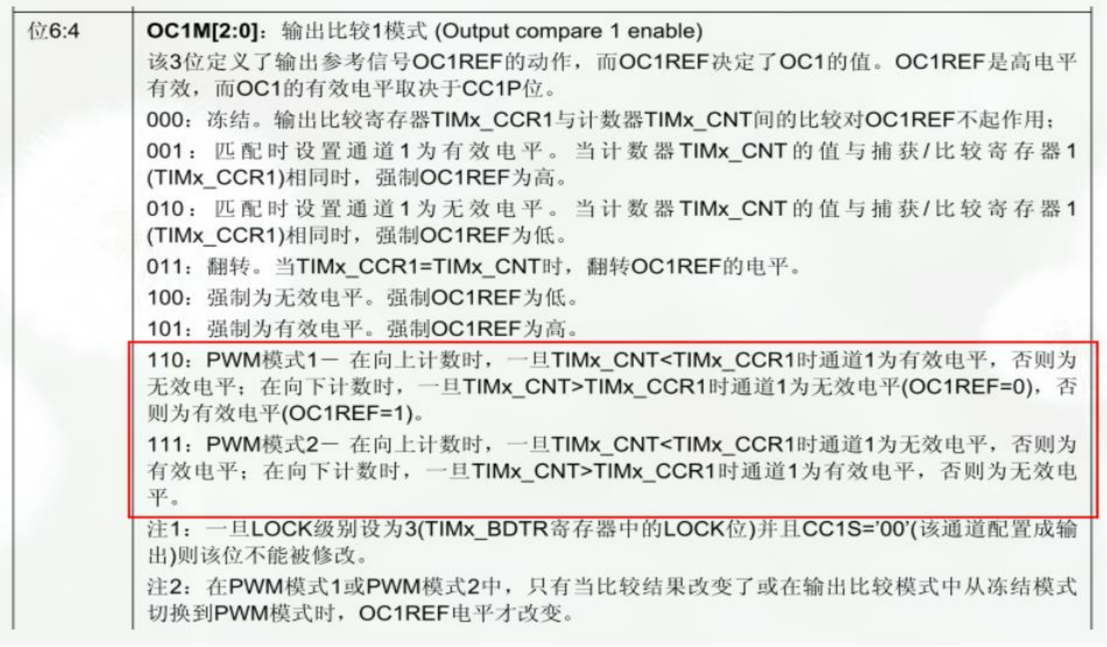
Address offset: 0x54

Reset value: 0x0000 0000

Refer to the above CCMR1 register description. Channels 5 and 6 can only be configured in output.

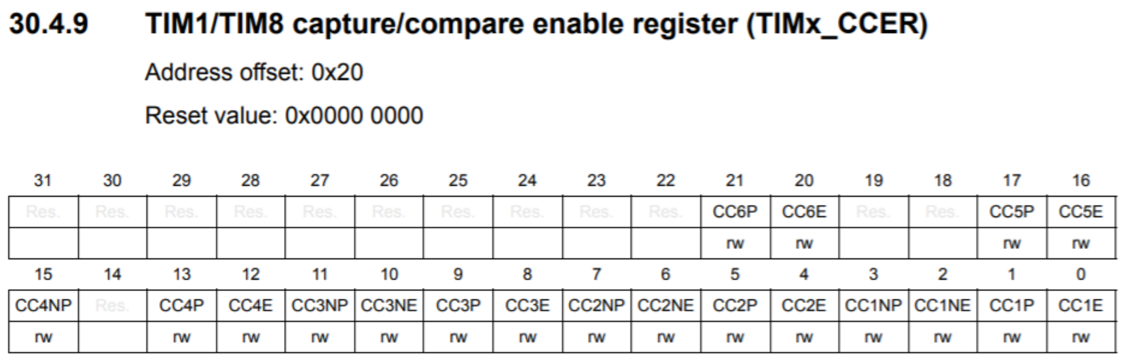
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC6M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC5M[3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC6 CE	OC6M[2:0]			OC6 PE	OC6FE	Res.	Res.	OC5 CE	OC5M[2:0]			OC5PE	OC5FE	Res.	Res.
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw		



位6:4	**OC1M[2:0]**：输出比较1模式 (Output compare 1 enable) 该3位定义了输出参考信号OC1REF的动作，而OC1REF决定了OC1的值。OC1REF是高电平有效，而OC1的有效电平取决于CC1P位。 000：冻结。输出比较寄存器TIMx_CCR1与计数器TIMx_CNT间的比较对OC1REF不起作用； 001：匹配时设置通道1为有效电平。当计数器TIMx_CNT的值与捕获/比较寄存器1 (TIMx_CCR1)相同时，强制OC1REF为高。 010：匹配时设置通道1为无效电平。当计数器TIMx_CNT的值与捕获/比较寄存器1 (TIMx_CCR1)相同时，强制OC1REF为低。 011：翻转。当TIMx_CCR1=TIMx_CNT时，翻转OC1REF的电平。 100：强制为无效电平。强制OC1REF为低。 101：强制为有效电平。强制OC1REF为高。 110：PWM模式1－ 在向上计数时，一旦TIMx_CNT<TIMx_CCR1时通道1为有效电平，否则为无效电平；在向下计数时，一旦TIMx_CNT>TIMx_CCR1时通道1为无效电平(OC1REF=0)，否则为有效电平(OC1REF=1)。 111：PWM模式2－ 在向上计数时，一旦TIMx_CNT<TIMx_CCR1时通道1为无效电平，否则为有效电平；在向下计数时，一旦TIMx_CNT>TIMx_CCR1时通道1为有效电平，否则为无效电平。 注1：一旦LOCK级别设为3(TIMx_BDTR寄存器中的LOCK位)并且CC1S='00'(该通道配置成输出)则该位不能被修改。 注2：在PWM模式1或PWM模式2中，只有当比较结果改变了或在输出比较模式中从冻结模式切换到PWM模式时，OC1REF电平才改变。

3.　CCER: 開啟輸出波(CCRx 就選 CCxE 改)



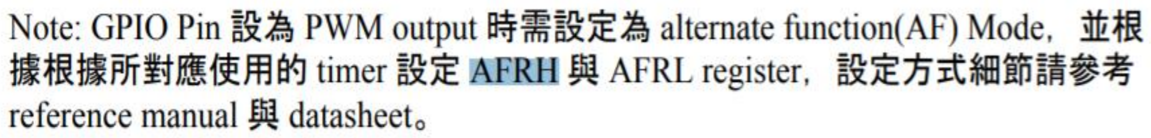
30.4.9　TIM1/TIM8 capture/compare enable register (TIMx_CCER)

Address offset: 0x20

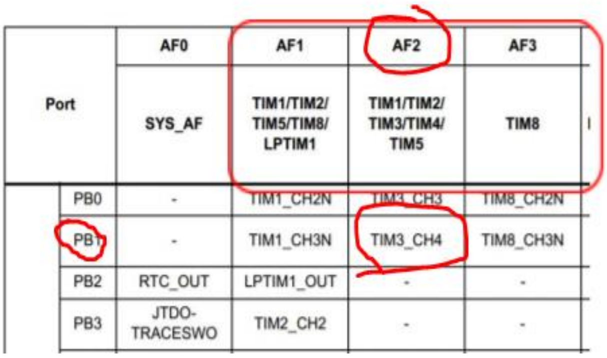
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E
										rw	rw			rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CC4NP	Res.	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

4.　AF Mode: (ex: output 為 PB1 時，綁定 AF2 加上 TIM3 的 CH4)

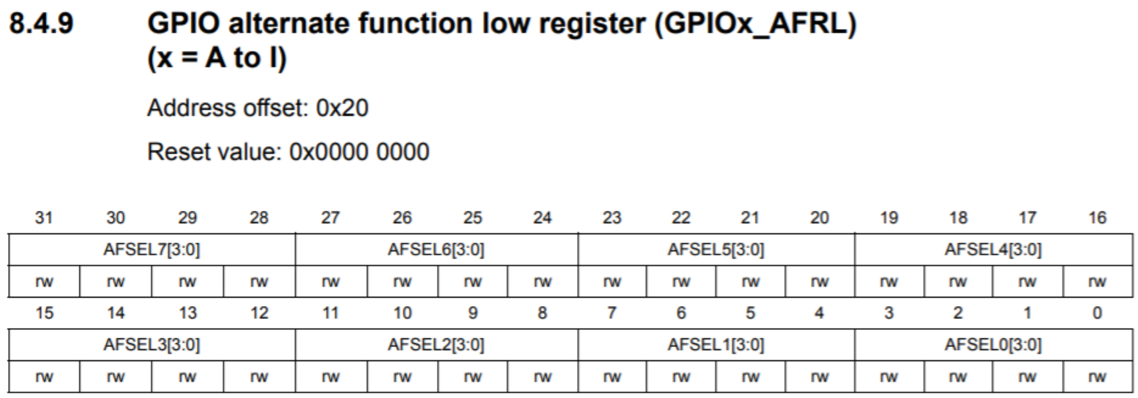


Note: GPIO Pin 設為 PWM output 時需設定為 alternate function(AF) Mode，並根據根據所對應使用的 timer 設定 AFRH 與 AFRL register，設定方式細節請參考 reference manual 與 datasheet。



Port		AF0	AF1	AF2	AF3
		SYS_AF	TIM1/TIM2/ TIM5/TIM8/ LPTIM1	TIM1/TIM2/ TIM3/TIM4/ TIM5	TIM8
	PB0	-	TIM1_CH2N	TIM3_CH3	TIM8_CH2N
	PB1	-	TIM1_CH3N	TIM3_CH4	TIM8_CH3N
	PB2	RTC_OUT	LPTIM1_OUT	-	-
	PB3	JTDO- TRACESWO	TIM2_CH2	-	-

註：For each GPIO pins 0 through 7 are set in AFR[0] which is called GPIOx_AFRL.　For GPIO pins 8 through 15 set the corresponding 4 bits for the desired pin in AFR[1] which is called GPIOx_AFRH

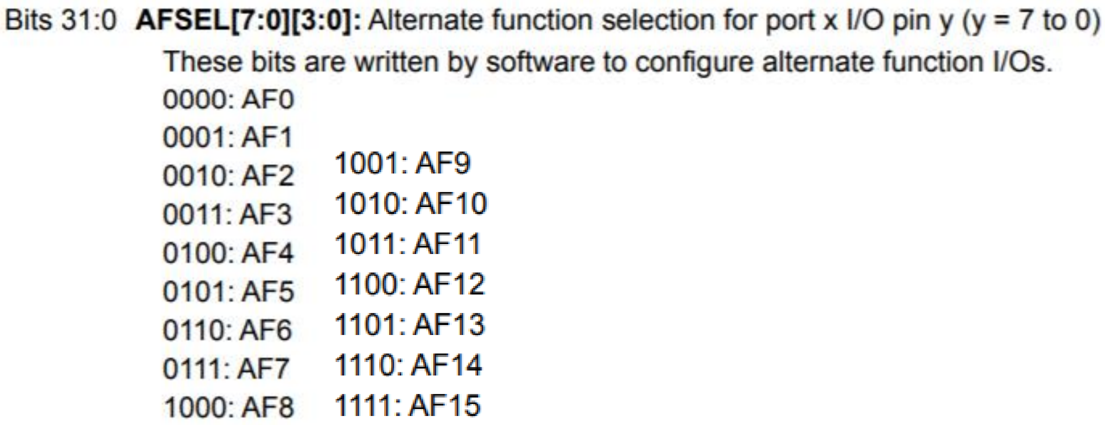


8.4.9　GPIO alternate function low register (GPIOx_AFRL) (x = A to I)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw



Bits 31:0 **AFSEL[7:0][3:0]:** Alternate function selection for port x I/O pin y (y = 7 to 0)

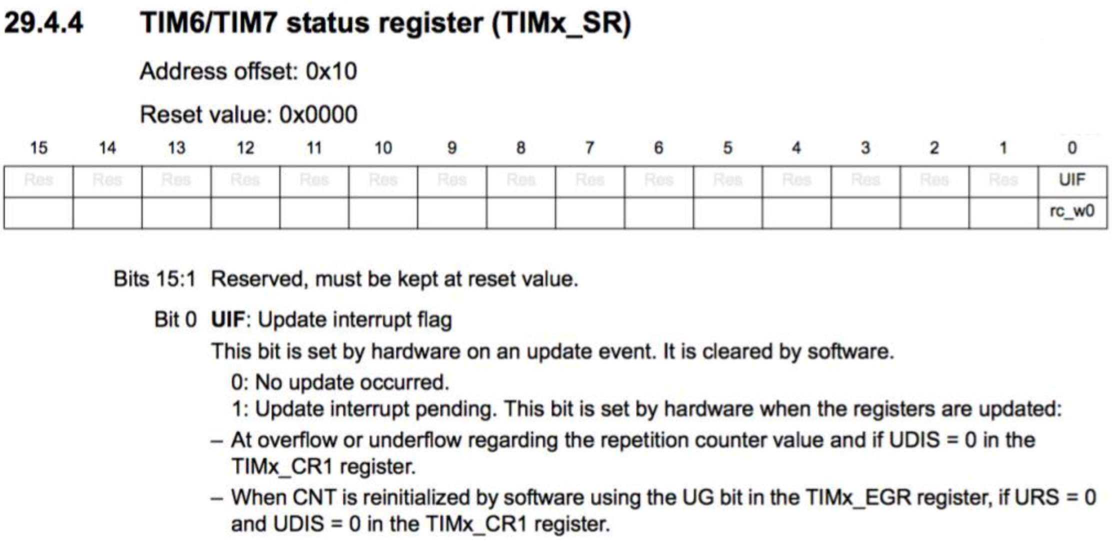
These bits are written by software to configure alternate function I/Os.

0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8
1001: AF9
1010: AF10
1011: AF11
1100: AF12
1101: AF13
1110: AF14
1111: AF15

GPIOB->AFR[0] = 0x2; 因為 PB3 要開 AF2

其他 Interrupt 用：

- TIMx_SR (status reg)：Interrupt 用。



29.4.4 TIM6/TIM7 status register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	UIF
															rc_w0

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

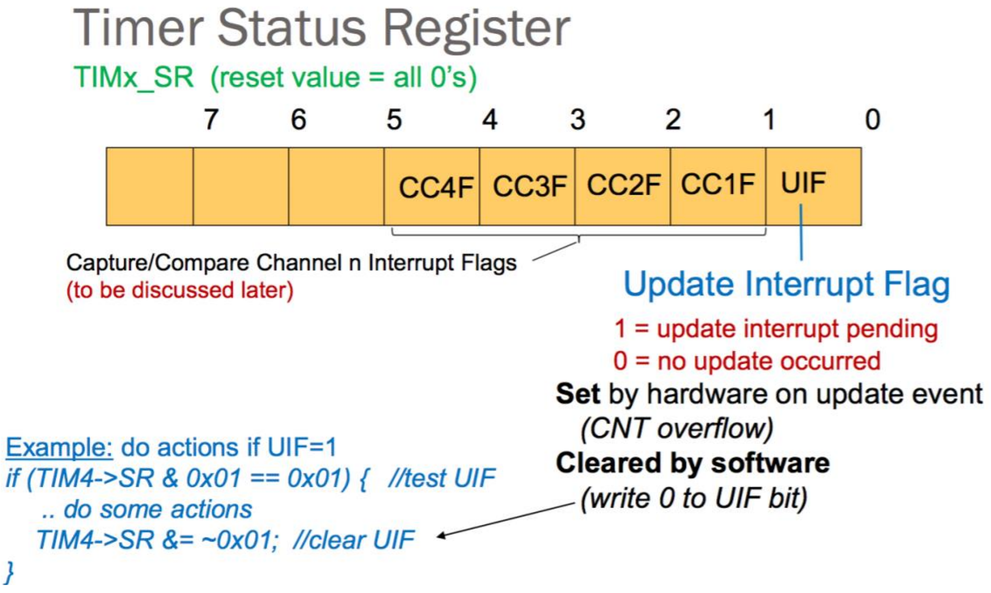
0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow or underflow regarding the repetition counter value and if UDIS = 0 in the TIMx_CR1 register.

– When CNT is reinitialized by software using the UG bit in the TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

Example:



Timer Status Register

TIMx_SR (reset value = all 0's)

7	6	5	4	3	2	1	0
			CC4F	CC3F	CC2F	CC1F	UIF

Capture/Compare Channel n Interrupt Flags
(to be discussed later)

Update Interrupt Flag

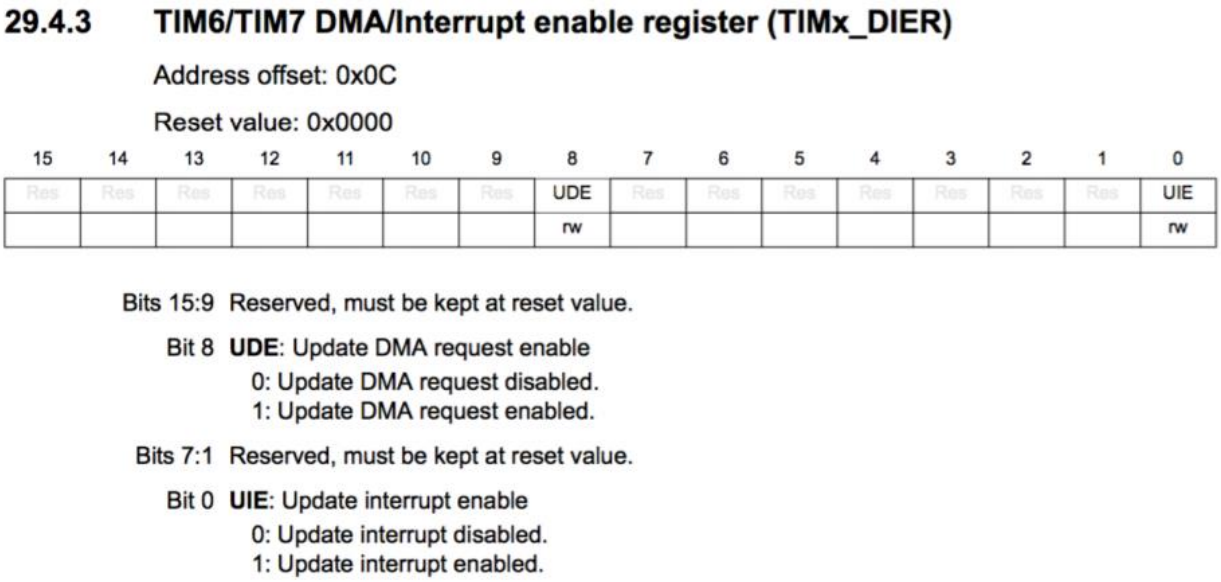
1 = update interrupt pending
0 = no update occurred

Set by hardware on update event
(CNT overflow)
Cleared by software
(write 0 to UIF bit)

Example: do actions if UIF=1

```
if (TIM4->SR & 0x01 == 0x01) {   //test UIF
    .. do some actions
   TIM4->SR &= ~0x01;  //clear UIF
}
```

- TIMx_DIER (interrupt enable reg)：Interrupt 用。



29.4.3 TIM6/TIM7 DMA/Interrupt enable register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	UDE	Res	Res	Res	Res	Res	Res	Res	UIE
							rw								rw

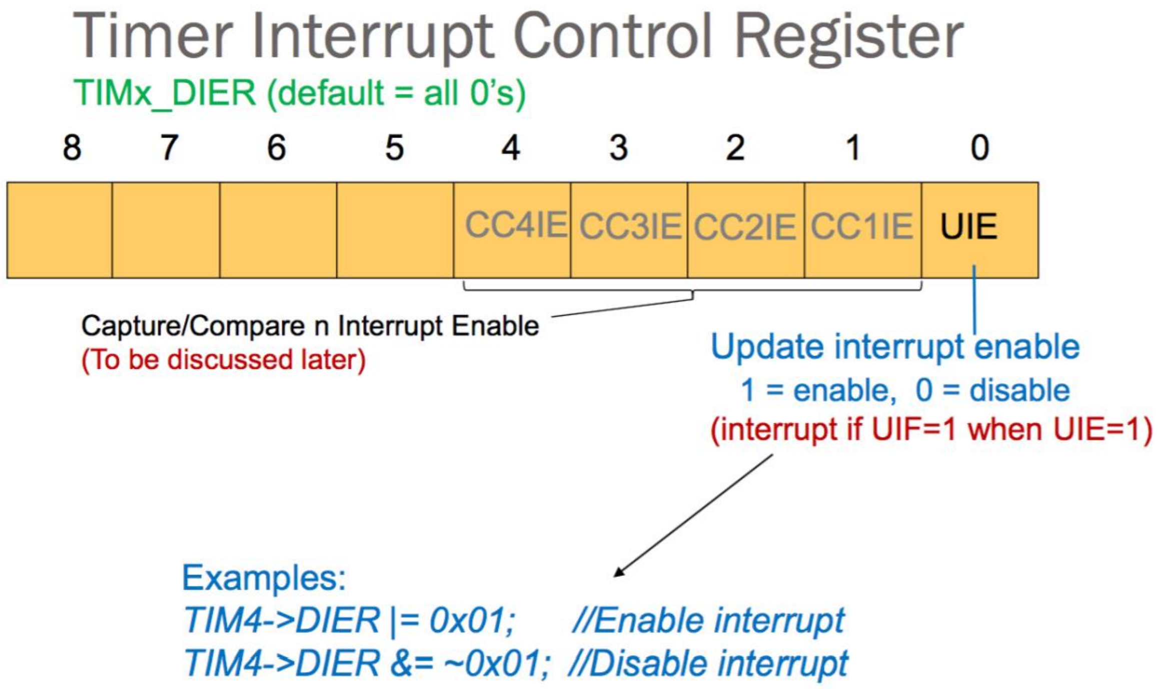
Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled.
1: Update DMA request enabled.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled.
1: Update interrupt enabled.

Example:



Timer Interrupt Control Register

TIMx_DIER (default = all 0’s)

8	7	6	5	4	3	2	1	0
				CC4IE	CC3IE	CC2IE	CC1IE	UIE

Capture/Compare n Interrupt Enable
(To be discussed later)

Update interrupt enable
1 = enable, 0 = disable
(interrupt if UIF=1 when UIE=1)

Examples:
TIM4->DIER |= 0x01; //Enable interrupt
TIM4->DIER &= ~0x01; //Disable interrupt

3. Steps 實驗步驟

3.1 Lab6.1: Modify system initial clock (20%)

實驗所要進行的目的是指定 pll 為 system clk，並以 msi 作為 pll 參考的依據。

如同前一個 lab，stm32 的元件大部分都以 stucter 存在 header 中以直接使用。

- 先開啟 msi，和選定要開啟的 range，開啟的方式是將 enable 與 cr 進行 or。

  (在進行開啟的動作後，應在 while 迴圈放入 msirdy 製作閘門，去檢查動作是否完成。
- 在調整 pll 相關設定前，要先將 pll disable，包括 cfgr 的 sw 設定。
- 選擇 pll 參考的 clk，在這邊選擇的是 msi，選擇好後利用 SET_REG 去將值放入 RCC_PLLCFGR_PLLSRC。
- 根據頻率去調整係數，將 GPIOB 的 ODR 作為當前頻率的 mode

  (0b1 -> 1MHz, 0b11 -> 6MHz …)
- 除 1MHz 特別將 msirange 改為 4 外，其他頻率皆可利用 pllr/(pllm*plln)推得
- pll 設定結束後，將 pllcfgr 和 pll enable，再將 sw 寫入

以上為 pll 和 msi 相關設定過程。除此之外，還須加上 gpio 設定、delay 和 button 偵測的部分。這部分可從先前 lab 複製。

GPIO 方面，開啟 ABC，A 作為 output 操控 led，B 作為 output 以供 c 檔內的 flag，C 則作為 input 接受 button。

考量到高頻率時，程式執行過快，在一次 push 的過程便變化太多次，在每次 push 後加入一個小 delay 的動作(這個 delay 會根據當前頻率的不同，調整為不同的 cycle 數。

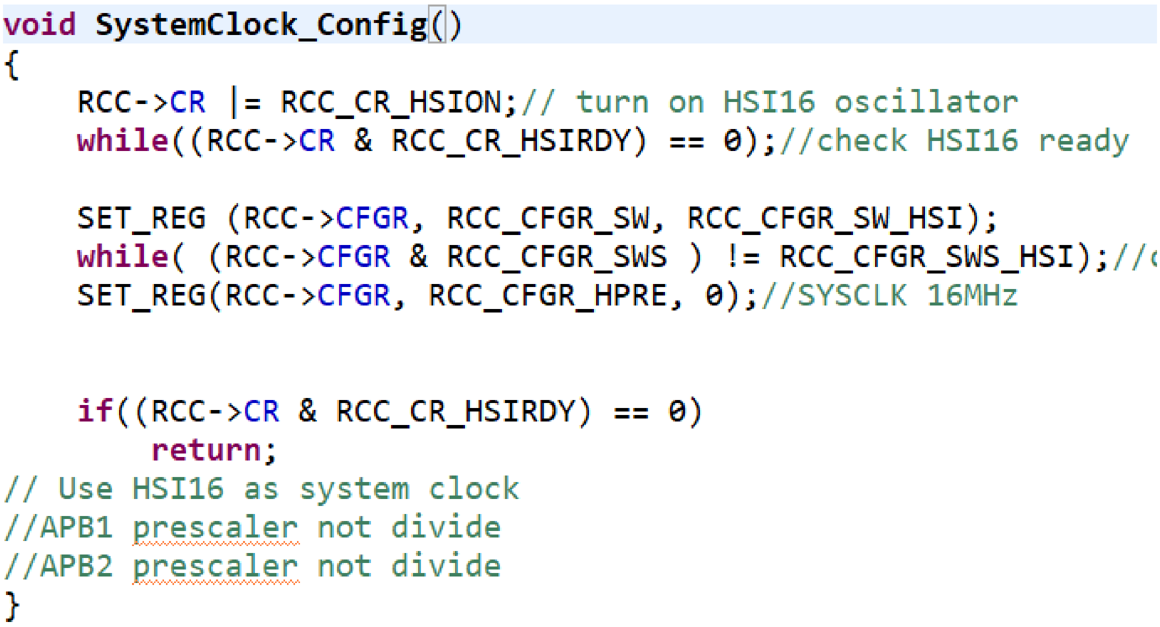
3.2 Lab6.2: Timer 計時器

完成以下的 main.c 中的 Timer_init() 與 Timer_start(); 並使用 STM32 timer 實做一個計時器會從 0 上數(Upcounting) TIME_SEC 秒的時間。顯示到小數點以下第二位，結束時 7-SEG LED 停留在 TIME_SEC 的數字。請使用 polling 的方式取得 timer CNT register 值並換算成時間顯示到 7-SEG LED 上。

0.01 ≤ TIME_SEC ≤ 10000.00 (超過範圍請直接顯示 0.00)

1. 設定 Clock (RCC_CR_xxxON, RCC_CFGR_SW_xxx, RCC_CFGR_HPRE)

- RCC 底下設定 System Clock (預設: MSI = 4Hz)：
  - ➪ CR 選擇 clock 來源(HSI16)
  - ➪ 等待設定完成(HSIRDY)
  - ➪ CFGR 切換到該 clock 來源的設定(SW_HSI)
  - ➪ 等待切換完成(SWS_HSI)
  - ➪ 如果需要，設定 System Clock prescaler



```
void SystemClock_Config()
{
    RCC->CR |= RCC_CR_HSION;// turn on HSI16 oscillator
    while((RCC->CR & RCC_CR_HSIRDY) == 0);//check HSI16 ready

    SET_REG (RCC->CFGR, RCC_CFGR_SW, RCC_CFGR_SW_HSI);
    while( (RCC->CFGR & RCC_CFGR_SWS ) != RCC_CFGR_SWS_HSI);//
    SET_REG(RCC->CFGR, RCC_CFGR_HPRE, 0);//SYSCLK 16MHz


    if((RCC->CR & RCC_CR_HSIRDY) == 0)
        return;
// Use HSI16 as system clock
//APB1 prescaler not divide
//APB2 prescaler not divide
}
```

2. 設定 Timer (APB1ENR1_TIMxEN, TIMx_PSC, TIMx_ARR, TIMx_EGR_UG, TIMx_CR1_EN)

- Timer 底下設定：

⇨ 開啟 APB1(x1) 或 APB2(x2)

⇨ 設定 PSC (prescalar)，多少時脈加一次 counter

⇨ 設定 ARR，counter 加到多少時 Reload

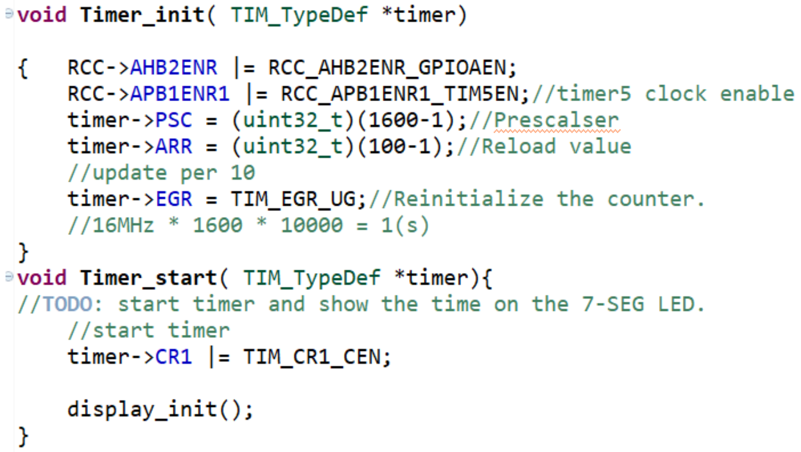
⇨ HIS (16 Hz) * (1 / PSC) * (1 / ARR) = 0.01 秒

Counter 每 0.01 秒會數完並 Reload 1 次。

⇨ (其他: DIR:倒數設定…等)

⇨ EGR: 重置 Timer 值

⇨ CR1 設定 CEN: 打開 Timer



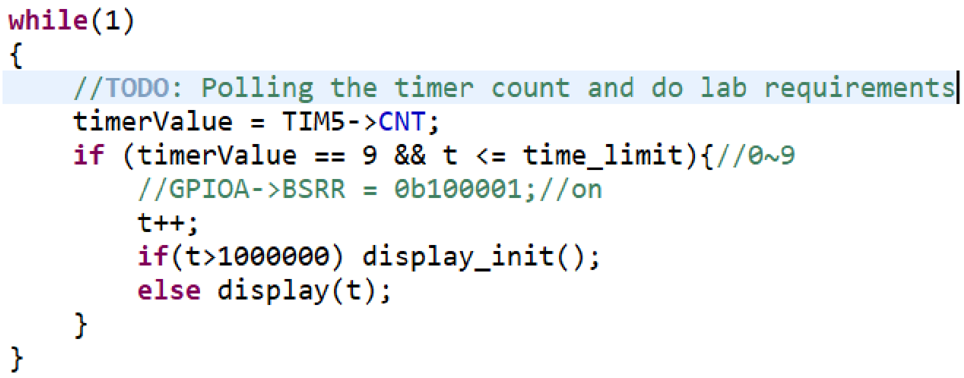
```
void Timer_init( TIM_TypeDef *timer)

{    RCC->AHB2ENR |= RCC_AHB2ENR_GPIOAEN;
     RCC->APB1ENR1 |= RCC_APB1ENR1_TIM5EN;//timer5 clock enable
     timer->PSC = (uint32_t)(1600-1);//Prescalser
     timer->ARR = (uint32_t)(100-1);//Reload value
     //update per 10
     timer->EGR = TIM_EGR_UG;//Reinitialize the counter.
     //16MHz * 1600 * 10000 = 1(s)
}
void Timer_start( TIM_TypeDef *timer){
//TODO: start timer and show the time on the 7-SEG LED.
     //start timer
     timer->CR1 |= TIM_CR1_CEN;

     display_init();
}
```

- 利用 polling TIMx->CNT 取得 counter 值，用上次的 lab 顯示

只要在 timer 到達某一個特定值更新一下 Max7219 顯示值即可 (累加單位: 0.01 秒，例如 10 秒時 t = 1000)



```
while(1)
{
    //TODO: Polling the timer count and do lab requirements
    timerValue = TIM5->CNT;
    if (timerValue == 9 && t <= time_limit){//0~9
        //GPIOA->BSRR = 0b100001;//on
        t++;
        if(t>1000000) display_init();
        else display(t);
    }
}
```

3.3 Lab6.3: Music keypad

請利用 timer 產生並輸出 Duty cycle 為 50% 的 PWM 訊號，並以 Lab5 中的 keypad 為鍵盤，當使用者在按下不同 keypad 按鍵時產生特定頻率(參考下表)的 PWM 方波給蜂鳴器，沒按鍵或按到沒功能的鍵時請不要發出聲音。

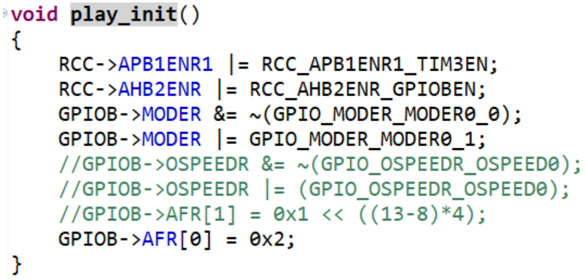
不同 output port 有不同的要求，例如 pb1 就要去設定 TIM3 底下的 CCER, CCR4, CCMR2, 和 GPIOB2 底下的 AFR[0]

1. 設定 PWM (GPIOx_AFR[], TIMx_CCER, TIMx_CCMRn, TIMx_CCRn)

⇨ 開啟 APB1ENR1 下的 timer3 (TIM3EN)

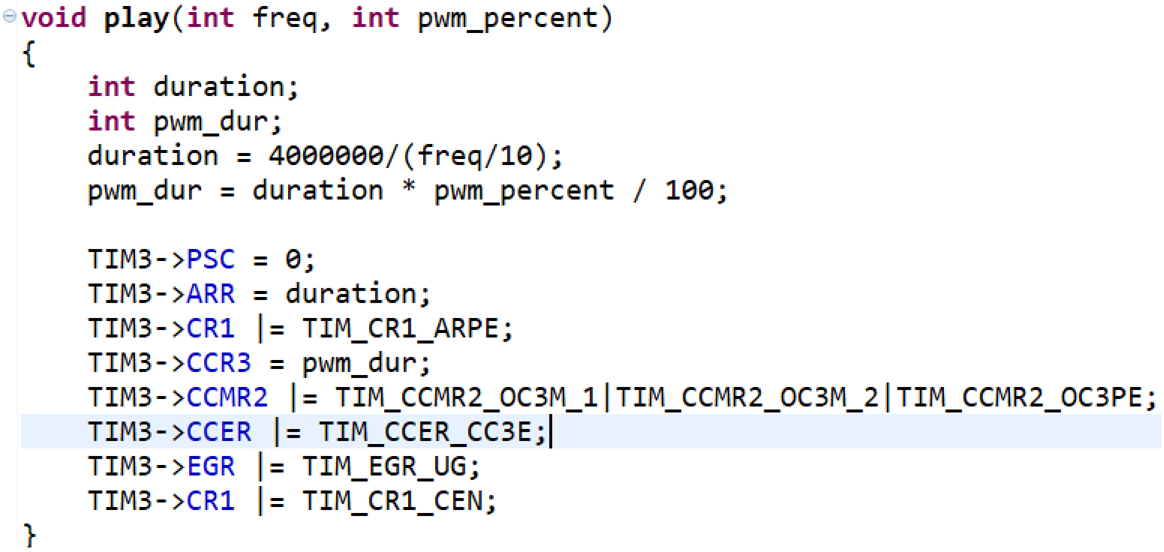
⇨ 開啟 output port(pb1)

⇨ 開啟 AFRL 的 AF3



```
void play_init()
{
	RCC->APB1ENR1 |= RCC_APB1ENR1_TIM3EN;
	RCC->AHB2ENR |= RCC_AHB2ENR_GPIOBEN;
	GPIOB->MODER &= ~(GPIO_MODER_MODER0_0);
	GPIOB->MODER |= GPIO_MODER_MODER0_1;
	//GPIOB->OSPEEDR &= ~(GPIO_OSPEEDR_OSPEED0);
	//GPIOB->OSPEEDR |= (GPIO_OSPEEDR_OSPEED0);
	//GPIOB->AFR[1] = 0x1 << ((13-8)*4);
	GPIOB->AFR[0] = 0x2;
}
```

2. 設定輸出 PWM 的 Timer 設定(比正常的多了 CCER, CCMRn, CCRn 要設



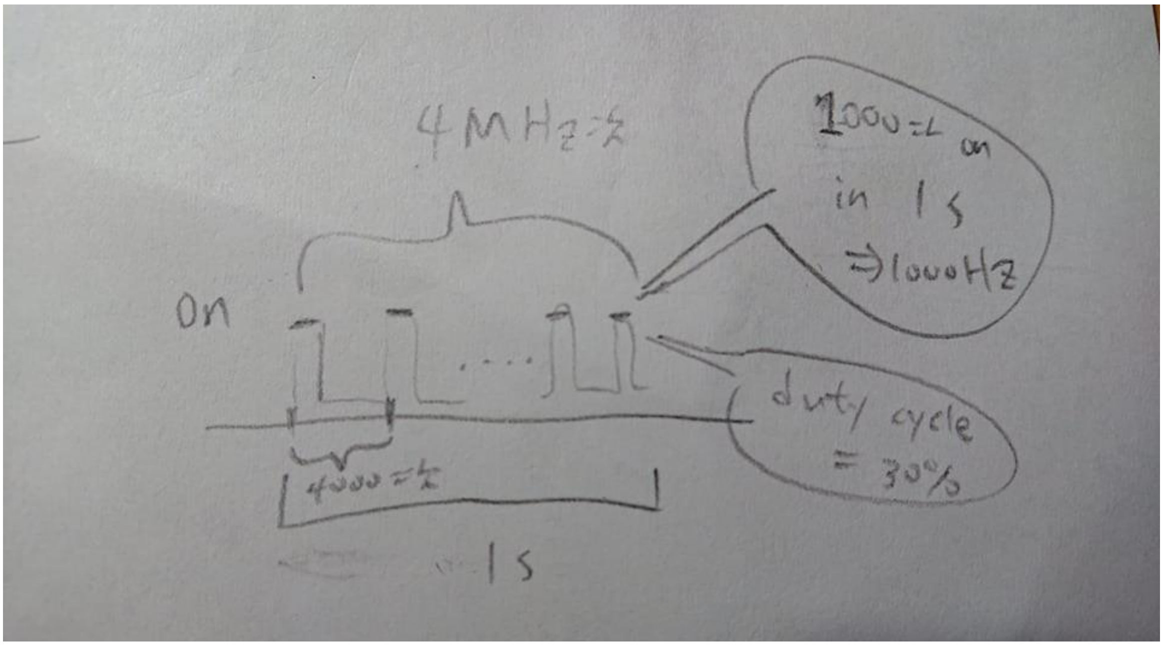
```
void play(int freq, int pwm_percent)
{
	int duration;
	int pwm_dur;
	duration = 4000000/(freq/10);
	pwm_dur = duration * pwm_percent / 100;

	TIM3->PSC = 0;
	TIM3->ARR = duration;
	TIM3->CR1 |= TIM_CR1_ARPE;
	TIM3->CCR3 = pwm_dur;
	TIM3->CCMR2 |= TIM_CCMR2_OC3M_1|TIM_CCMR2_OC3M_2|TIM_CCMR2_OC3PE;
	TIM3->CCER |= TIM_CCER_CC3E;
	TIM3->EGR |= TIM_EGR_UG;
	TIM3->CR1 |= TIM_CR1_CEN;
}
```

⇨ PSC=0，不要製造多餘的計算麻煩

⇨ 計算 ARR 和 CCR3，使其為適當的值

例如在 clock rate 4MHz 下設定一個頻率 1000Hz, duty cycle =30% 的 PWM：設定 ARR 讓 Timer 每 4000(=4M/1000) 次就 reload，再讓 CCR3 只有 ARR 數的過程中 (30) % 為高電位。



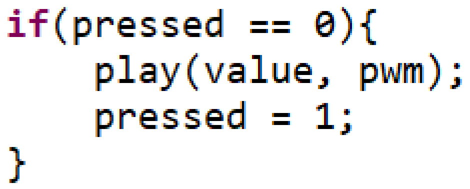
⇨ CR1 的 ARPE 區段設為 1，使 Timer 會 Auto reload & preload

⇨ 因為是用 CCR3，所以要開啟 CCER3

⇨ CCR3 的設定要在 CCMR2 改，去 CCMR2 把 OC3M[2:0]設為 PWM mode (110 or 111)

⇨ 其他就跟正常 Timer 一樣: EGR 重置，CR1 打開

值得注意的是對 Timer 的設定最好要用一個類似 lock 的東西保護，不要每次 polling 都去 call setting function, 不然他會有類似啟動蜂鳴器會發出的雜音，而不是乾淨的聲音。



```
if(pressed == 0){
    play(value, pwm);
    pressed = 1;
}
```

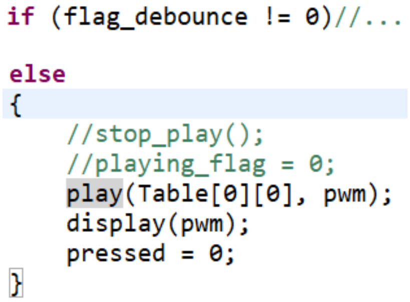
3.4 Lab6.4: Modify LED brightness

在前一實驗中的 keypad 增加 2 個功能按鈕用以調整 PWM 輸出的 Duty cycle (範圍 10%~90%，每按一次鍵調整 5%)。使用 LED 去替換蜂鳴器。

設*為加 5%，#為減 5%

把 3.3 改為不停輸出，只是按鍵從調頻率變成調 duty cycle

一次按鈕只調一次：所以在沒進 debounce 時才改變 duty cycle



```
if (flag_debounce != 0)//...

else
{
    //stop_play();
    //playing_flag = 0;
    play(Table[0][0], pwm);
    display(pwm);
    pressed = 0;
}
```

4. Feedback 實驗心得或建議

Init pwn 相關設定時，不能放在 while loop 讓他每次 polling 前都 Init 一次，似乎會來不及反應，導致沒有輸出聲音。但是之前的 max7219_init 並不會這樣。可能是 timer 結構非常複雜吧

這次 lab 給的時間有點太短，因為太多新架構需要理解了