實驗七 STM32 Clock and Timer

1. 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

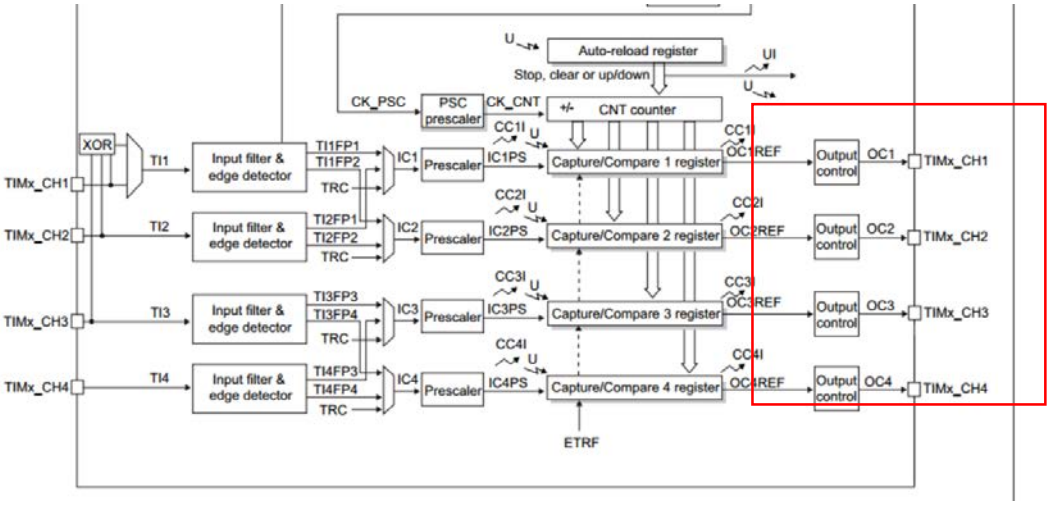
2. 實驗原理

2.1. Timer and Counter

請參考上課 009-MCSL-CounterTimer 講義。

2.2. Timer PWM output mode

在 STM32 系統中要利用 Timer 產生 PWM 輸出，主要通過 capture/compare mode register(TIMx_CCMR1)與 TIMx_CCRx registers 設定並利用 TIMx_CCER 啟動之。

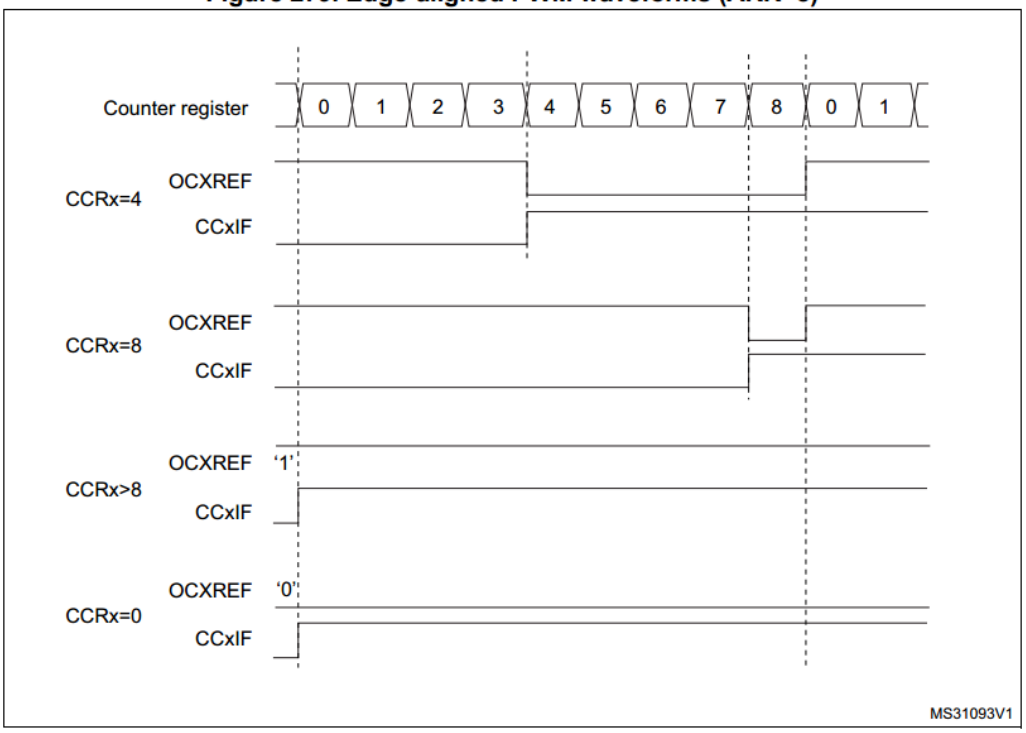


而一般 PWM 有分 mode1 與 mode2 兩種模式，而在計數器上數模式時其對應的輸出為

- PWM mode1: Channel is active as long as TIMx_CNT<TIMx_CCR1 else inactive.
- PWM mode2: Channel is inactive as long as TIMx_CNT<TIMx_CCR1 else active.

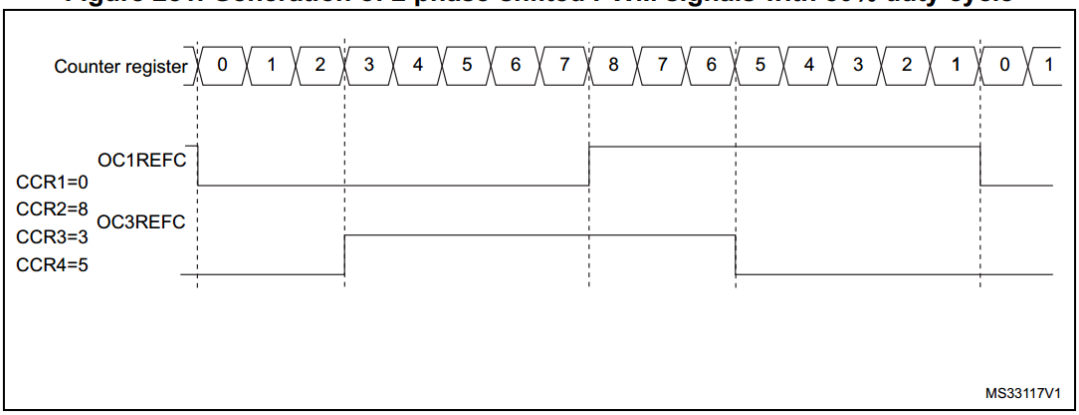
另外依不同的特殊用途又可分 Combined PWM mode 與 Asymmetric PWM mode。

Figure 279. Edge-aligned PWM waveforms (ARR=8)



在 ARR 設定為 8 時，不同 CCR 值設定下 OCxREF 所對應的訊號輸出
以上圖範例來說當 CCRx=4,ARR=8 可以得到 duty cycle(在單位時間內 1 準位與 0 準位的比例)為 50%的波形，CCRx=8,ARR=8 可得 duty cycle= 1-1/8=87.5%的波形。
要輸出比較複雜的 PWM 與不同 duty cycle 波形也可以利用 Asymmetric PWM mode 來達成。

Figure 281. Generation of 2 phase-shifted PWM signals with 50% duty cycle



其他 PWM 設定細節用途請參閱 Reference manual chapter 27.3.9 PWM mode

3. 實驗步驟

3.1. Modify system initial clock(20%)

- 請利用先前 lab 所實作的 GPIO_init 與 delay_1s，可呼叫之前的 assembly function 或是用 C 重新實作，初始化 GPIO 與 delay。
- 修改 SYSCLK 的 clock source 以及相關的 prescaler 使得 CPU frequency(HCLK)為 1MHz。
- 觀察修改前後 LED 燈閃爍的頻率。
- 當使用者按下 user button 便依以下順序改變 CPU system clock(HCLK)，1MHz -> 6MHz -> 10MHz ->16MHz -> 40MHz ->1MHz ->…

main.c

```
void GPIO_init();
void 4MHz_delay_1s();
void SystemClock_Config(){
    //TODO: Change the SYSCLK source and set the corresponding
Prescaler value.
}

int main(){
   SystemClock_Config();
   GPIO_init();
   while(1){
      if (user_press_button())
      {
        //TODO: Update system clock rate
      }
      GPIOA->BSRR = (1<<5);
      4MHz_delay_1s ();
      GPIOA->BRR = (1<<5);
      4MHz_delay_1s ();
   }
}
```

Note: 有些 CPU 頻率設定須由 PLLCLK 內的倍頻器與除頻器達成，此時須將 SYSCLK source 改成 PLLCLK 並依以下流程設定 RCC_PLLCFGR register 設定。

To modify the PLL configuration, proceed as follows:

1. Disable the PLL by setting PLLON to 0 in *Clock control register (RCC_CR)*.
2. Wait until PLLRDY is cleared. The PLL is now fully stopped.
3. Change the desired parameter.
4. Enable the PLL again by setting PLLON to 1.
5. Enable the desired PLL outputs by configuring PLLPEN, PLLQEN, PLLREN in *PLL configuration register (RCC_PLLCFGR)*.

其中 PLL clock 頻率計算為 f(VCO clock) = f(PLL clock input) × (PLLN / PLLM)
最終可輸出給 system clock 頻率為 f(PLL_R) = f(VCO clock) / PLLR

3.2. 計時器(30%)

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。(建議使用擁用比較高 counter resolution 的 TIM2~TIM5 timer)，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)

例如 TIME_SEC 為 12.7 時的 demo 影片：https://goo.gl/F9hh35

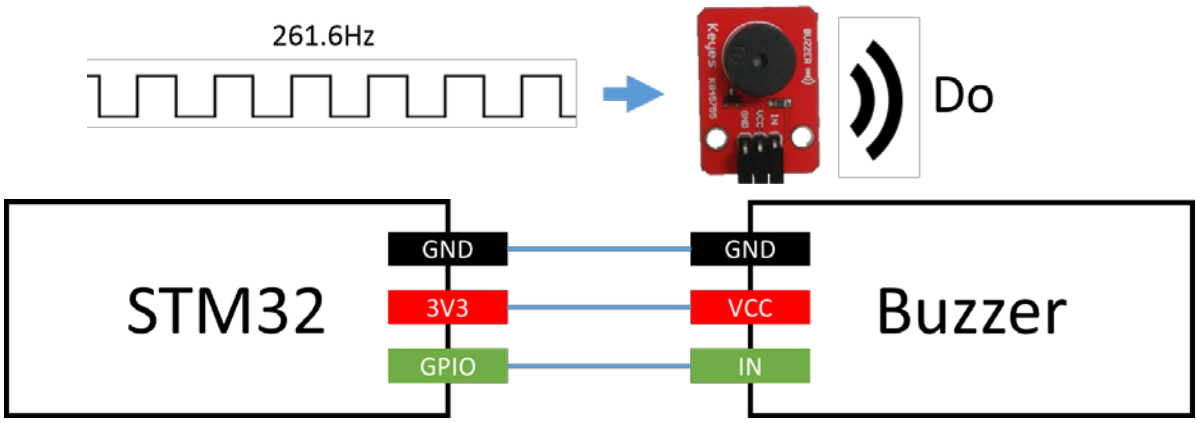
Note: 7-SEG LED 驅動請利用之前 Lab 所實作的 GPIO_init()、max7219_init()與 Display ()函式呈現(須改成可呈現 2 個小數位)。

main.c

```
#include "stm32l476xx.h"
#define TIME_SEC 12.70
extern void GPIO_init();
extern void max7219_init();
extern void Display();
void Timer_init( TIM_TypeDef *timer)
{
    //TODO: Initialize timer
}
void Timer_start(TIM_TypeDef *timer){
    //TODO: start timer and show the time on the 7-SEG LED.
}
int main()
{
   GPIO_init();
   max7219_init();
   Timer_init();
   Timer_start();
   while(1)
   {
      //TODO: Polling the timer count and do lab requirements
   }
}
```

3.3. Music keypad(35%)

蜂鳴器分為有源(自激式)蜂鳴器和無源(他激式)蜂鳴器。有源蜂鳴器將驅動電路直接設計到蜂鳴器中，因此只需提供直流電壓就可以發出聲音，但其缺點是聲音的頻率無法更改。無源蜂鳴器外部需提供震盪波形才會發出聲音，其聲音的頻率就是輸入波的頻率。我們這次 LAB 使用的是無源蜂鳴器。



蜂鳴器的 VCC 接 3.3V、GND 接 GND、IN 接 GPIO 腳位。

請利用 timer 產生並輸出 Duty cycle 為 **50%的 PWM 訊號**，並以 Lab6 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。本次實驗會需要設定 GPIOx_AFRH、GPIOx_AFRL、TIMx_CCER、TIMx_CCMR1、TIMx_CCR1…等 registers。

Note: 參考 RM0351 Reference manual 瞭解這些 register 的功能完成此次實驗。並利用 STM32L476xx 找到 timer channel 所對應的腳位。

	X0	X1	X2	X3
Y0	Do	Re	Mi	
Y1	Fa	So	La	
Y2	Si	HDo		
Y3				

Keypad 對應音名

音名	Do	Re	Mi	Fa	So	La	Si	HDo
頻率(Hz)	261.6	293.7	329.6	349.2	392.0	440.0	493.9	523.3

音名頻率對應表

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
Port B	PB0	-	TIM1_CH2N	TIM3_CH3	TIM8_CH2N
	PB1	-	TIM1_CH3N	TIM3_CH4	TIM8_CH3N
	PB2	RTC_OUT	LPTIM1_OUT	-	-
	PB3	JTDO-TRACESWO	TIM2_CH2	-	-
	PB4	NJTRST	-	TIM3_CH1	-
	PB5	-	LPTIM1_IN1	TIM3_CH2	-
	PB6	-	LPTIM1_ETR	TIM4_CH1	TIM8_BKIN2
	PB7	-	LPTIM1_IN2	TIM4_CH2	TIM8_BKIN
	PB8	-	-	TIM4_CH3	-
	PB9	-	IR_OUT	TIM4_CH4	-
	PB10	-	TIM2_CH3	-	-
	PB11	-	TIM2_CH4	-	-
	PB12	-	TIM1_BKIN	-	TIM1_BKIN_COMP2
	PB13	-	TIM1_CH1N	-	-
	PB14	-	TIM1_CH2N	-	TIM8_CH2N
	PB15	RTC_REFIN	TIM1_CH3N	-	TIM8_CH3N

PortB AF mode selection table

範例：https://goo.gl/4MuIFv

```
extern void GPIO_init();
void GPIO_init_AF(){
//TODO: Initial GPIO pin as alternate function for buzzer. You can
choose to use C or assembly to finish this function.
}
void Timer_init(){
    //TODO: Initialize timer
}
void PWM_channel_init(){
    //TODO: Initialize timer PWM channel
}
int main(){
   GPIO_init();
    GPIO_init_AF();
   Timer_init();
    PWM_channel_init();
     //TODO: Scan the keypad and use PWM to send the corresponding
frequency square wave to buzzer.
}
```

3.3.1. Music 音色實驗(15%)

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%)，觀察是否會影響蜂鳴器所發出的聲音大小或音色。

Note: 須注意頻率與 duty cycle 的關係來設定 timer ARR 與 CCR registers。可用 LED 測試 duty cycle 是否有改變，成功應會看到 LED 隨著 duty cycle 不同而有明暗變化。