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。

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。

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

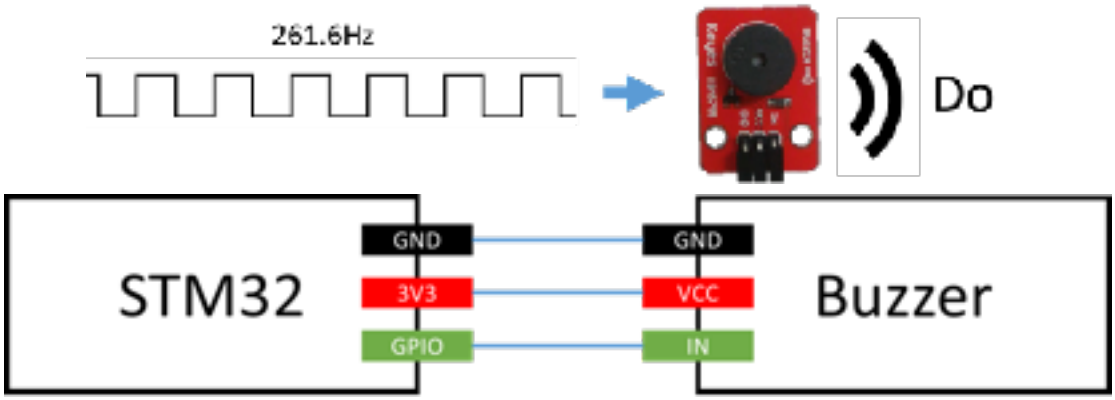
例如 TIME_SEC 為12.7時的demo影片：https://goo.gl/F9hh35
Note: 7-SEG LED驅動請利用之前Lab所實作的GPIO_init()、max7219_init()與Display ()函式呈現(須改成可呈現2個小數位)。

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。

3.1.1Music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。

4.實驗步驟

4.1.

基本上按照助教的步驟initialize就可以完成。其中，若要對RCC_PLLCFGR做更動，必須先disable RCC_CR。我試了幾次都失敗，後來才發現原來要disable RCC_CR，要先將sysclk從PLL換成別種，然後才能disable。

因為sysclk最後的頻率是透過除頻器與倍頻器達成，所以達成目標頻率的比例可以任選。但是要注意的是，8<=PLLN<=86，超過此範圍即產生錯誤。

我的設定為:

	1MHz	6MHz	10MHz	16MHz	40MHz
f(PLL_input)	8MHz	8MHz	8MHz	8MHz	8MHz
PLLN	8	48	80	80	80
PLLM	8	8	8	5	2
PLLR	8	8	8	8	8

4.2.

老師的講義中有這題的範例code，只要把down count 改成 up count，以及一些修正就大功告成了。其中，老師計算dis_val（convert counter value to time）的方式在某些情況下會造成overflow:

```
int dis_val = TIME_SEC*100*timerValue/TIM_ARR_VAL;
```

只看總值的話，Counting time不太可能overflow(怎麼可能數超過2^16秒），但是因為計算方法是先乘法再除法，因此在算乘法且還沒算除法前就有機會overflow。我的修改方式是先將常數的部分計算出來，儲存至一浮點數中，最後再乘以timerValue。

```
float temp=TIME_SEC*100/TIM_ARR_VAL;
long long int dis_val = temp*timerValue;
```

不過因為stm32板子要做浮點數運算的話，要先開啟FPU，所以在init階段時要加FPU_enable。

4.3.

若希望輸出對應的音色，就要調整至對應的頻率。故prescale value=fcpu/音頻
例如：do=261MHz，cpu預設=4*10^6 MHz，prescale=4000000/261。所以，keypad scan到的地方如果是在我規定的位置中，就將新的prescale value丟進timer_init裡，對timer的頻率做更改。

4.3.1.

相較於4.3，需要再增加兩個按鈕，一個增加duty cycle的比例，一個降低。我選擇的是keypad上的”A”(增加比例)，”B”(降低比例)。

實驗理論：

Duty cycle=Ton/Toff(%)，高電位時間除以低電位時間的百分比。模擬出的電壓為 V=Von*Dutycycle(%)，可以知道duty cycle越高，模擬出來的電壓就越高。當duty cycle為100%時，電壓最高，蜂鳴器的音量會最大，反之亦然。

實驗結果：音量方面，duty cycle 90%和duty cycle 10%在我聽起來都一樣大聲，為了比較，我還特地錄下聲音重複播放。唯一能證明音量確實改變的現象，就是當

duty cycle為0%時，按下按鈕是聽不見聲音的。音色方面，duty cycle 90%時聲音聽起來比較扁平，10 %時聲音聽起來較立體。