光機電實驗 ADC DAC DMA

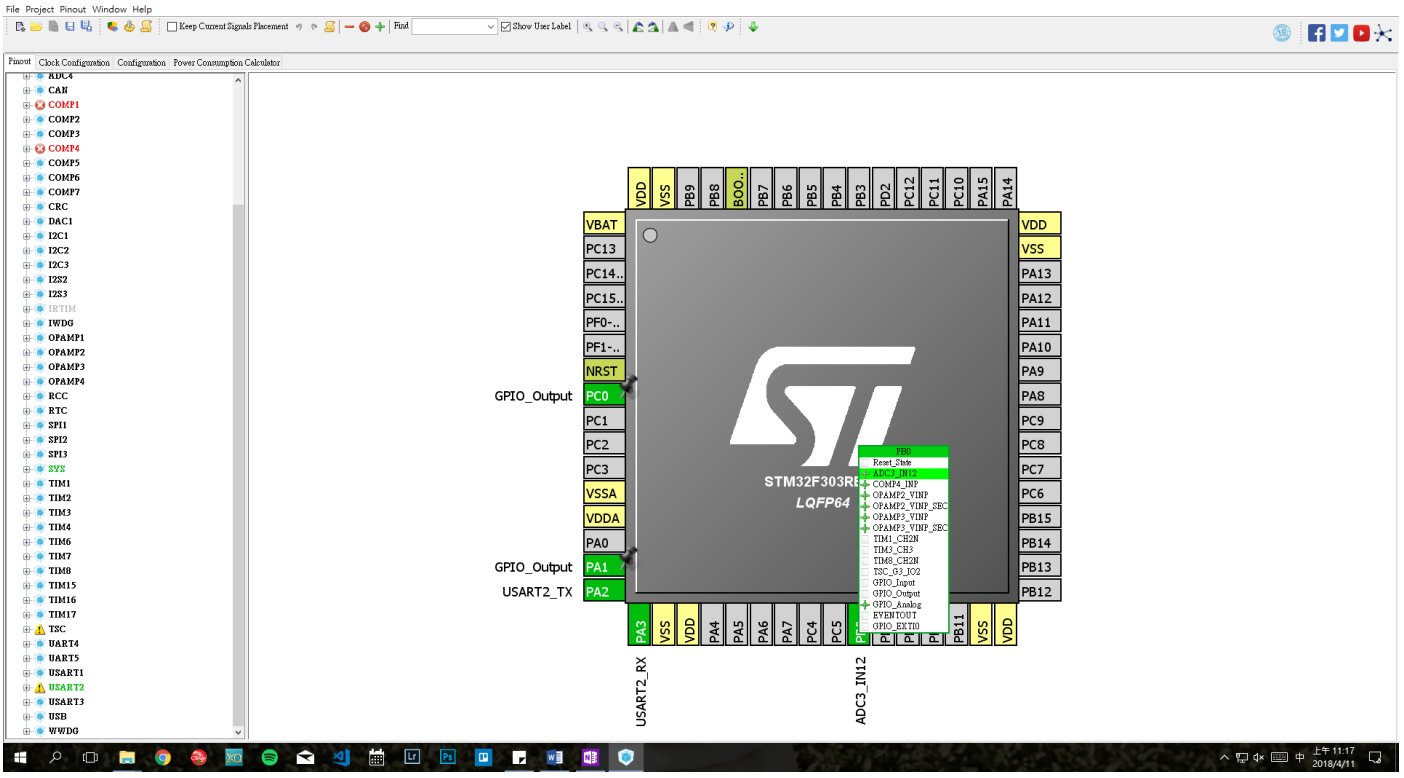
ADC 是將類比轉換至數位的模組。HAL library 裡面已經將大部分的工作做好，我們只需要呼叫 function 就好。

在這裡講到的類比，代表連續的電壓 (0v ~3.3v)，數位則是 ADC 模組提供的 solution，**STM32F303 的 solution 最高是 12-bit**，也就是從 $2^0 \sim 2^{12}$ 共 4096 個值。Resolution 可以選擇 12/10/8/6 四種不同值，解析度越低，轉換速度越快但還原後誤差越大。而 ADC 的時鐘(Clock prescaler)預設為 PLL，也可以換成來自 AHB clock 並加上 prescaler。另一個參數為 sampling time，是 ADC 轉換用到的電容的充電時間，這個值跟輸入阻抗有關。阻抗越大，需要的充電時間越長。而可選參數的一個 ”cycle” 時間即跟上面提到的時鐘來源頻率有關。

接下來的範例，我們會實作

1. ADC　　接上可變電阻，讀取 ADC 轉換後的值
2. DAC　　將 ADC 的值用 DAC 轉換為類比電壓，當作 LED 的電壓
3. ADC multiple channels
4. ADC multiple channels using DMA
5. DAC DMA

一、 ADC

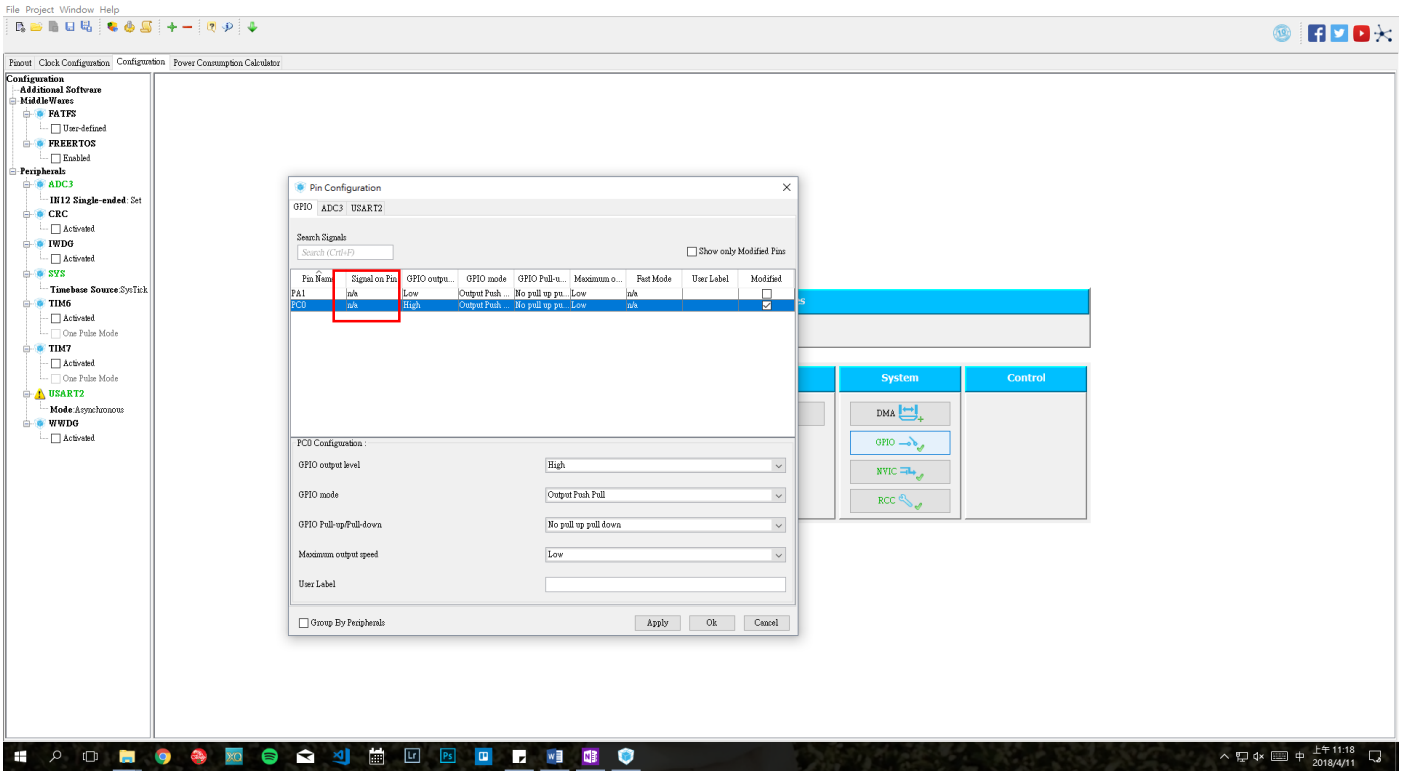


開啟:　PA1 : GPIO_Output

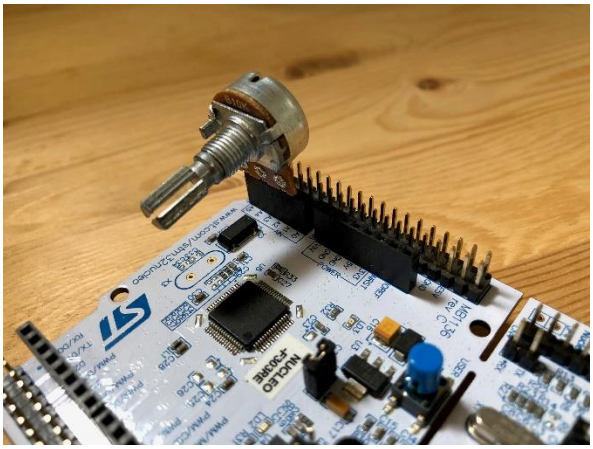
PC0 : GPIO_Output

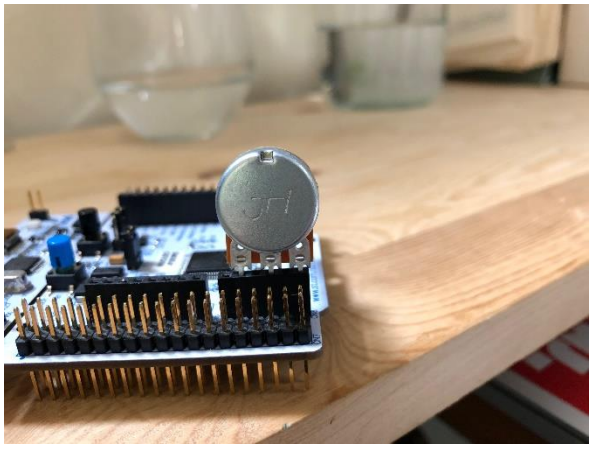
PB0 : ADC3_IN12

USART2



PC0 設為 High, PA1 設為 Low





插入可變電阻

重點 HAL lib function:

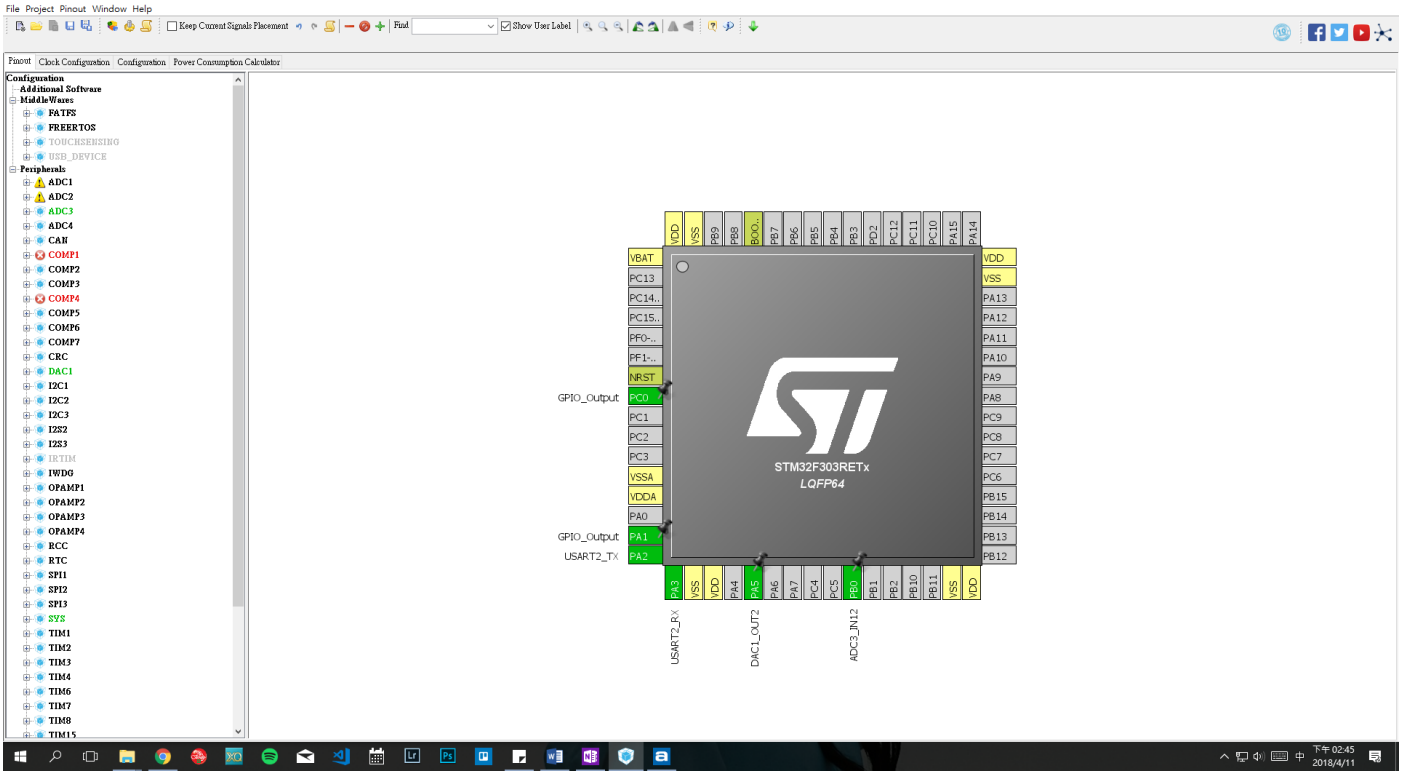
```
//開始 hadc3
HAL_ADC_Start(&hadc3)
//等待 hadc3 轉換完成
while(HAL_ADC_PollForConversion(&hadc3,0xFFFF) != HAL_OK);
//取得 hadc3 轉換後的值，放入自訂義的 uint32_t adc_result
uint32_t adc_result = HAL_ADC_GetValue(&hadc3);
```

詳細程式碼請見 github。

搭配之前學過的 UART，可以將轉換後的值傳到電腦。轉動可變電阻，電腦上的序列埠監控視窗即可以看到不同的值。

二、 DAC

在這個範例，我們要把剛剛由 ADC 轉換出來的值放入 DAC 中



將 PA5 設定為 DAC1_OUT2，其餘與第一個範例相同

重點 HAL lib function:

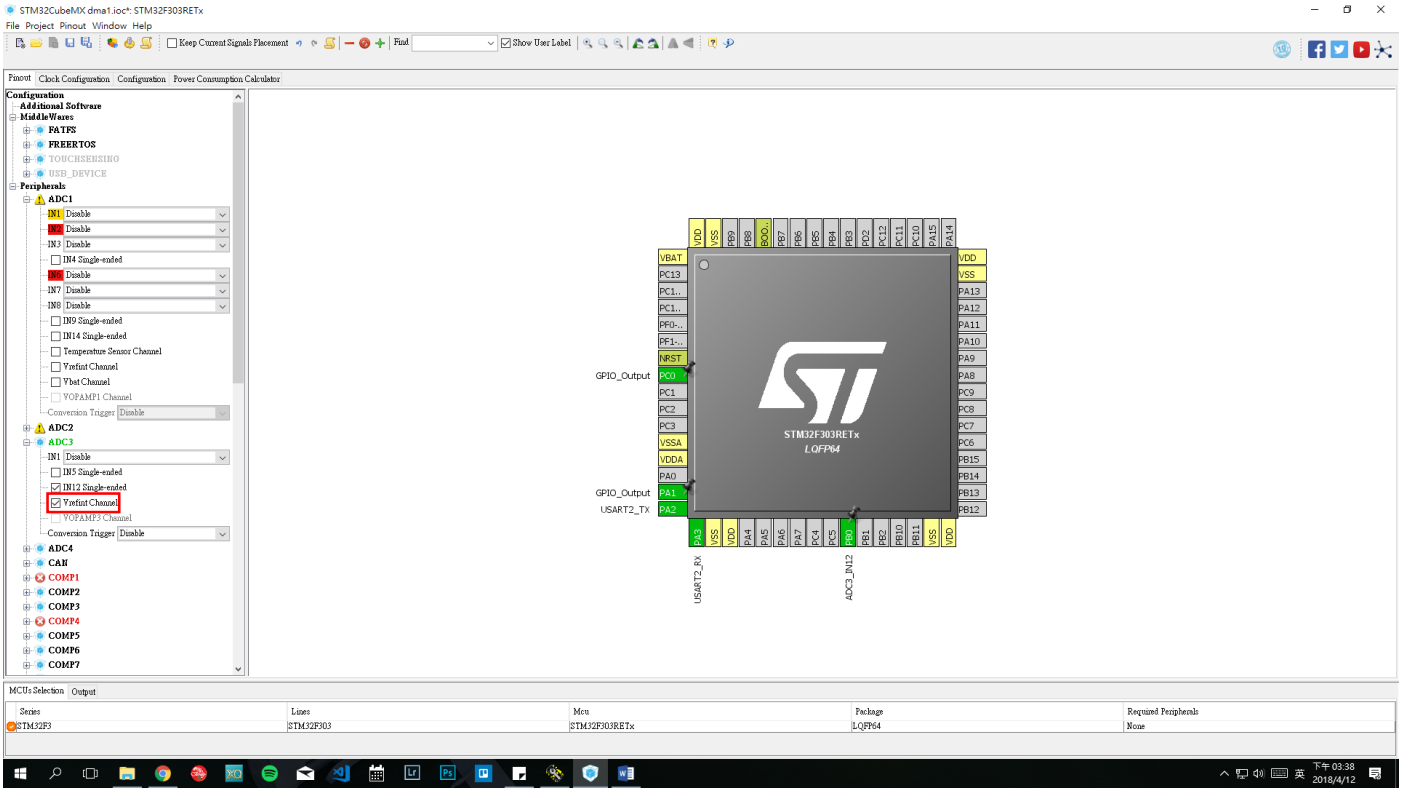
```
//開啟 hdac1 的 channel 2
HAL_DAC_Start(&hdac1, DAC_CHANNEL_2);
//將 hdac1_channel2 的值設為 adc_result，對齊方式為 DAC_ALIGN_12B_R
HAL_DAC_SetValue(&hdac1,DAC_CHANNEL_2,DAC_ALIGN_12B_R,adc_result);
```

完整程式碼請見 github。

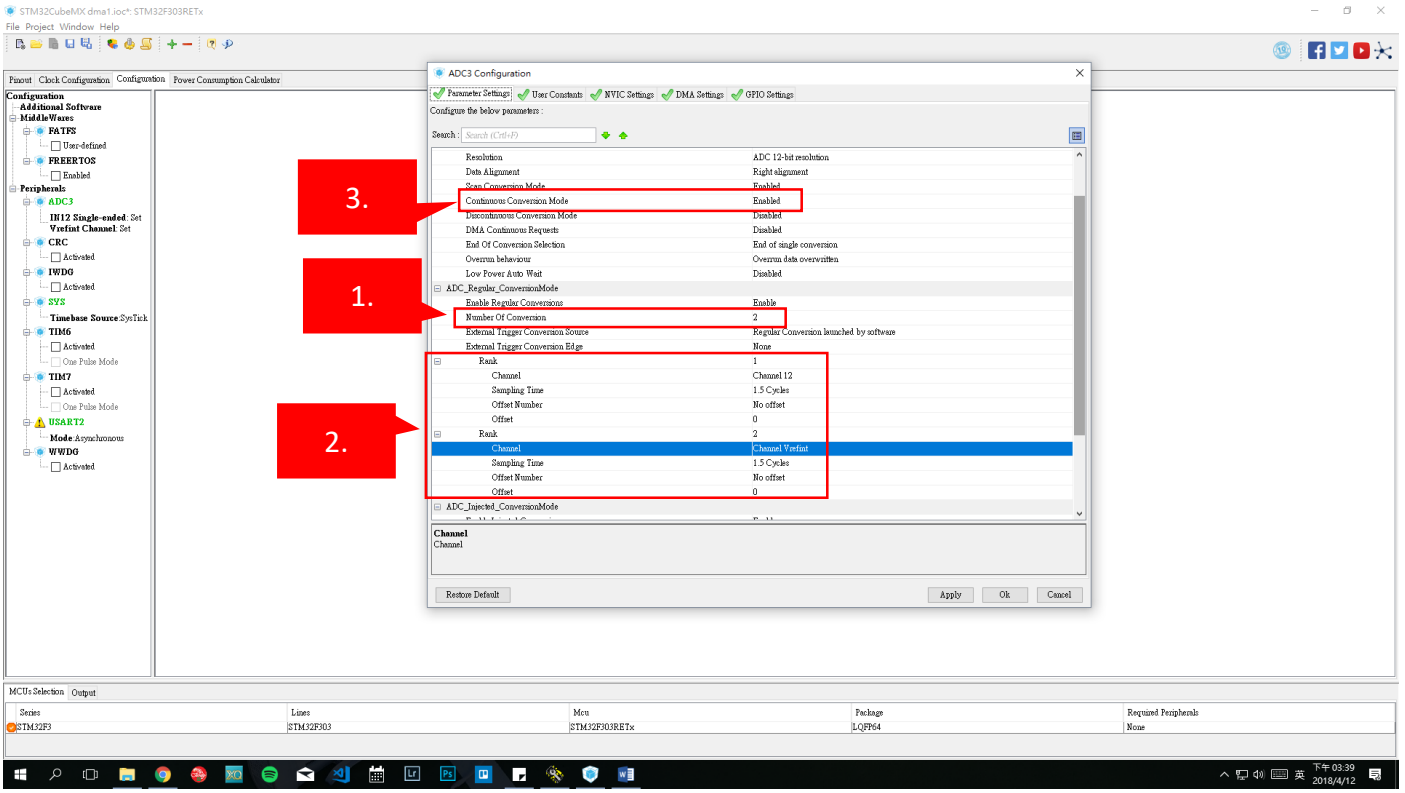
如此可以看到 LED 燈隨著可變電阻的調整而有明暗變化

三、 ADC multiple channels

在前面的範例中，我們是使用一個 ADC 下的一個 channel。當需要轉換一個 ADC 下的多個 channel 時，在 CubeMX 需要做以下更改。



除了第一個範例的 Pin 腳外，另外勾選 ADC3 的 Vrefint channel



1. ADC_Regular_ConversionMode 的 Num Of Conversion 設為欲轉換 channel 的個數，此例為 2
2. Rank 中設定兩個 channel 的先後順序此例為 Channel 12 (PB0) -> Vrefint
3. Continuous Conversion Mode 設為 Enable

Function 呼叫順序:

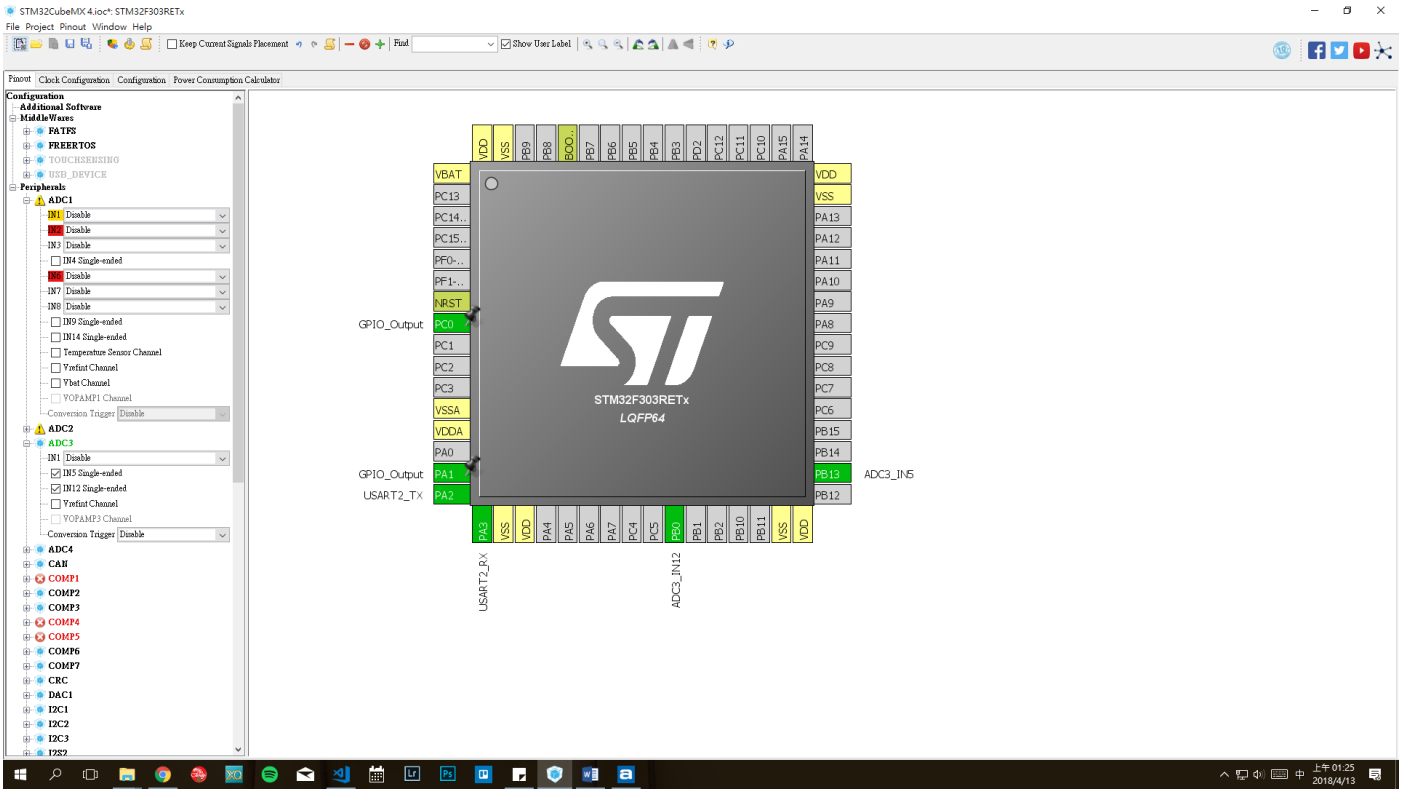
Start → PollForConversion → GetValue (for rank1) → GetValue (for rank2) → Stop

```
HAL_ADC_Start(&hadc3);
HAL_ADC_PollForConversion(&hadc3,0xFFFF);
int a=HAL_ADC_GetValue(&hadc3);
int b=HAL_ADC_GetValue(&hadc3);
HAL_ADC_Stop(&hadc3);
```

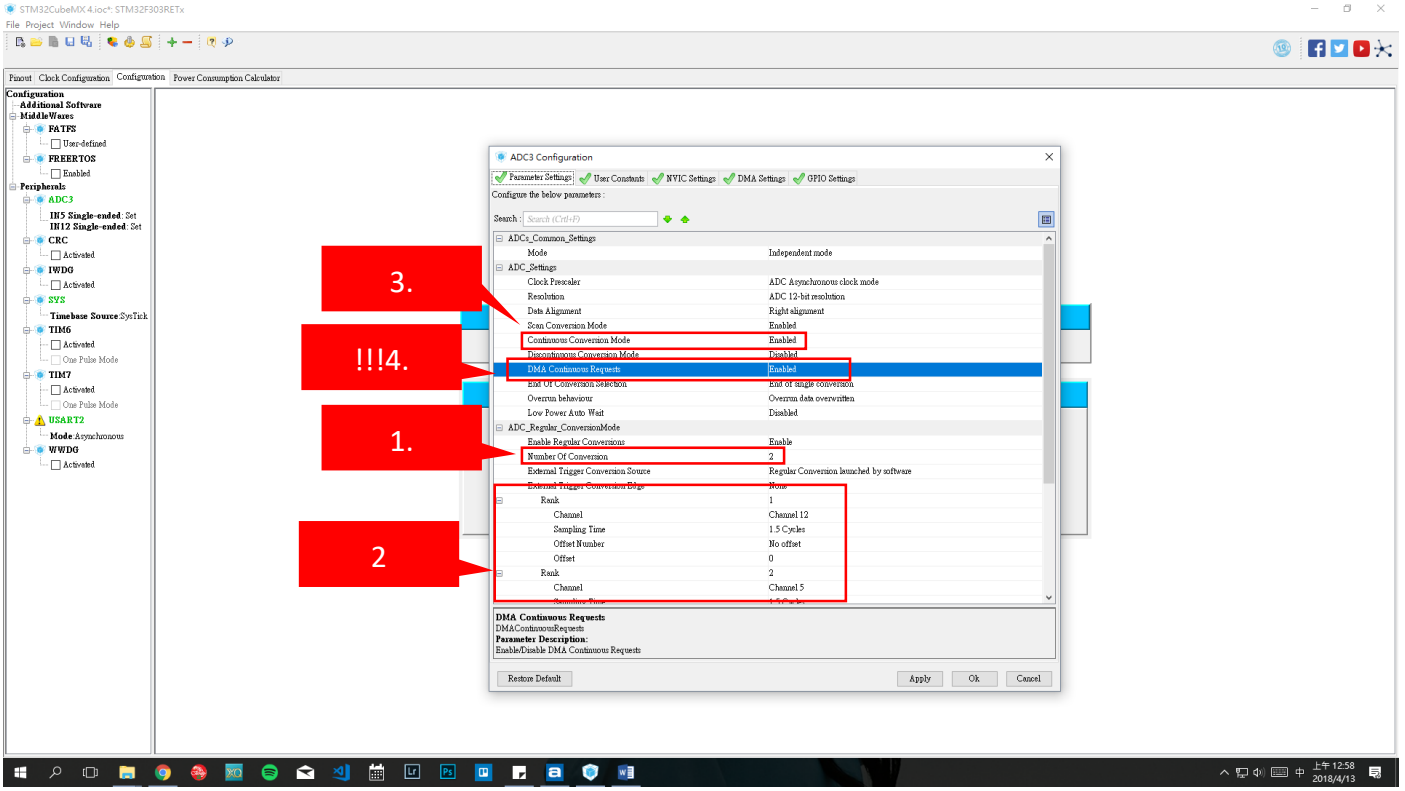
完整程式請見 [github]

四、 ADC multiple channels using DMA

這裡將使用 DMA (直接記憶體存取) 來進行多通道 ADC 的採樣。DMA 可以在不使用 CPU 資源的情況下，讓記憶體直接跟周邊設備進行資料傳輸。

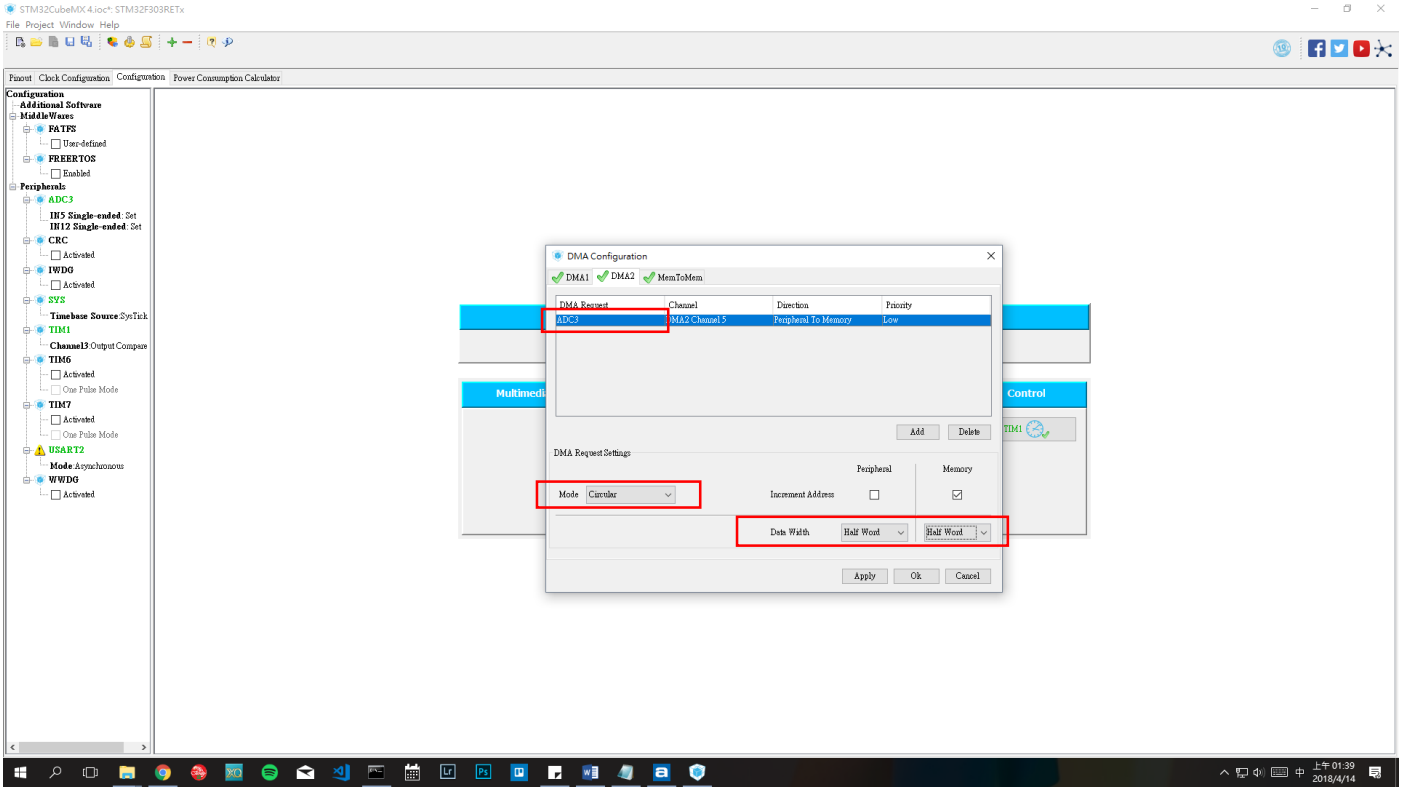


除了第一個範例的 Pin 腳外，另外勾選 ADC3 的 channel 5 (PB13)

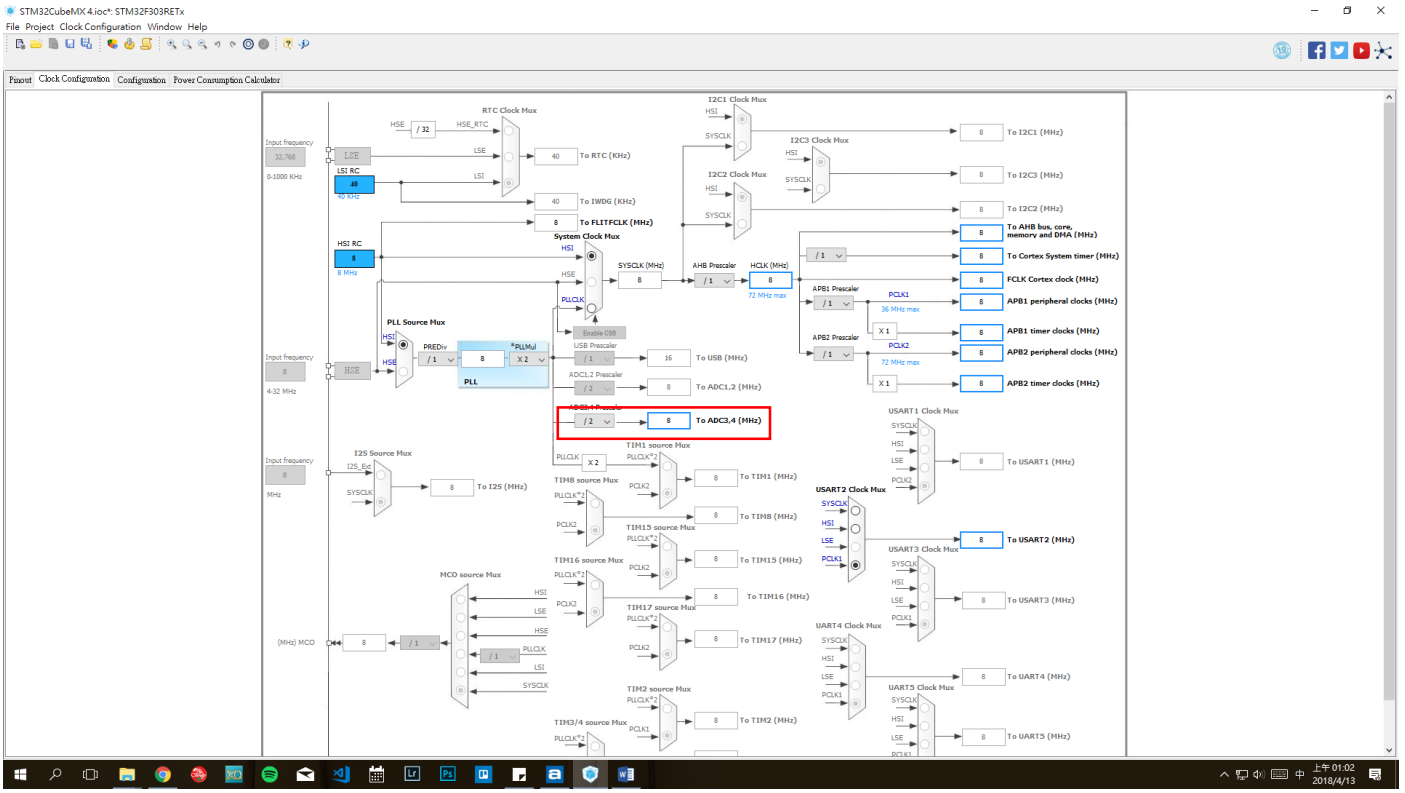


設定前三步跟上一個範例一樣。

不同的是第四步要在 ADC 設定中 Enable DMA continuous Requests



DMA Settings 中，DMA Request 選擇 ADC3，Mode 為 Circular，Data width 選擇 half word。這裡選擇 half word (16bit)的原因是因為採樣值是 12bit，若使用 32bit 的 Word 會浪費資源。



Clock configuration 中，將 ADC3,4 的值改為 8MHz (配合 DMA)

HAL lib function

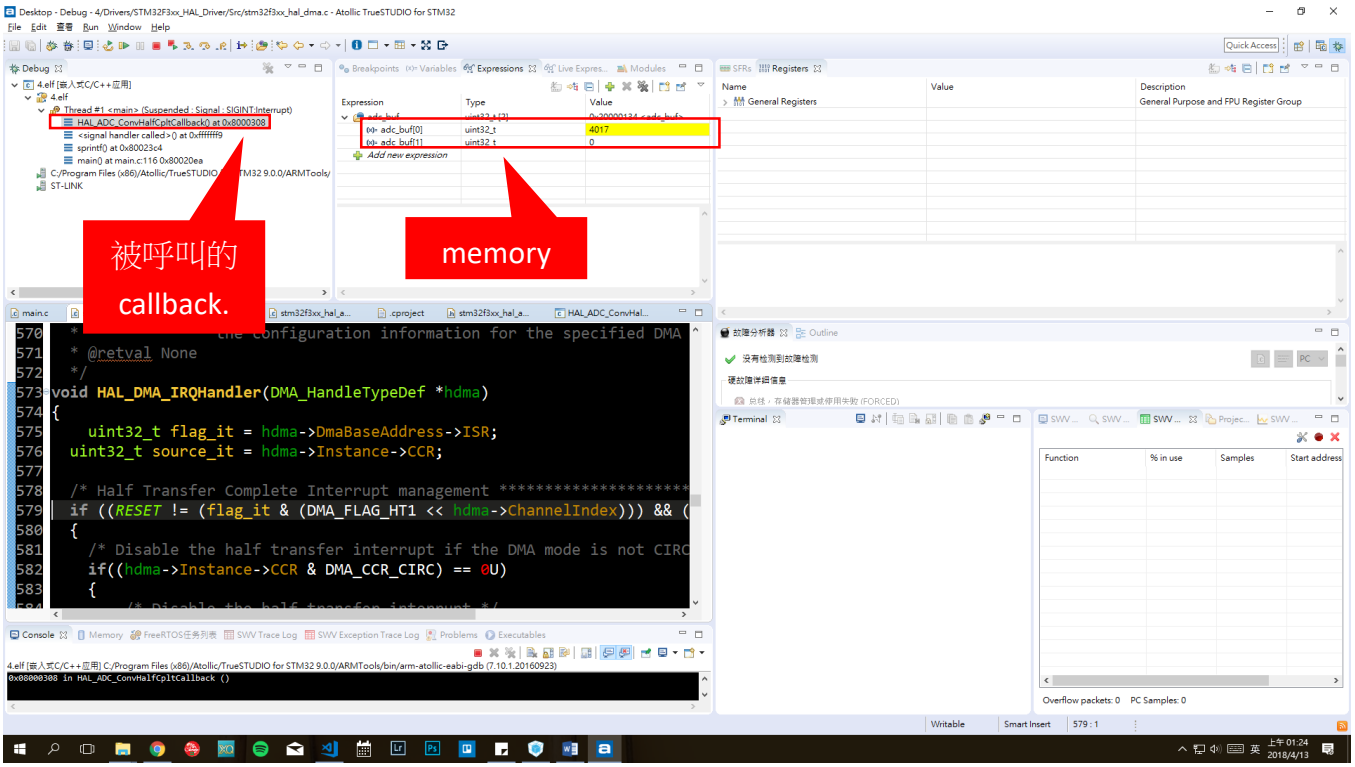
```
/**
  * @param  hadc ADC handle
  * @param  pData The destination Buffer address.
  * @param  Length The length of data to be transferred from ADC to memory.
  */
HAL_StatusTypeDef HAL_ADC_Start_DMA(ADC_HandleTypeDef* hadc, uint32_t* pData,
uint32_t Length)
```

部分 Code，完整請見 github。

```
/* USER CODE BEGIN PFP */
/* Private function prototypes -----------------------------------------------*/
uint32_t adc_buf[2]; //宣告存放採樣結果的 array
/* USER CODE END PFP */
```

```
/* USER CODE BEGIN 0 */
//這些 callback 會在每次轉換被呼叫，可以試試看在這裡加上 UART 之類的程式
//詳細說明可見 stm32fxx_hal_adc.c
void HAL_ADC_ConvHalfCpltCallback(ADC_HandleTypeDef* hadc){
}
/* USER CODE END 0 */
```

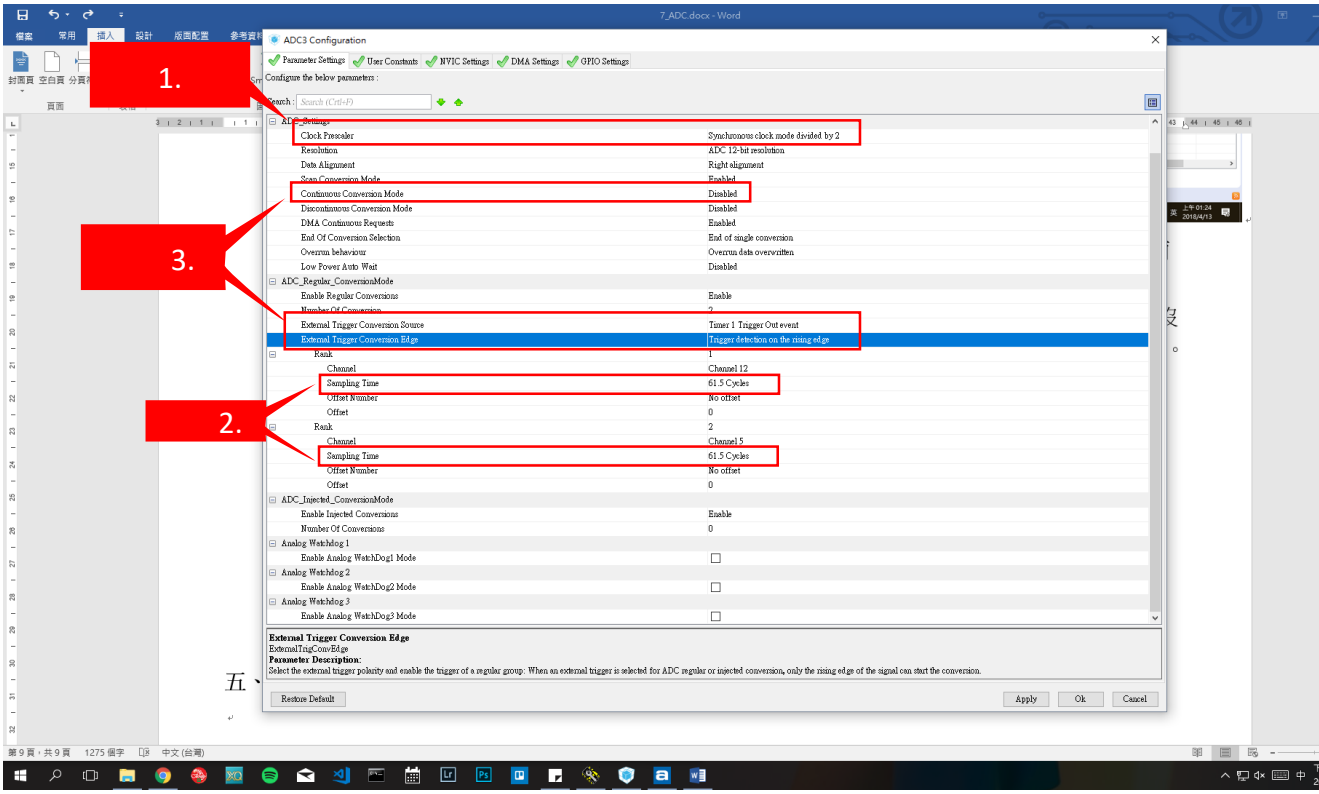
```
//放在 main 中，while 前
HAL_ADC_Start_DMA(&hadc3,adc_buf,2); //對 ADC3 的 2 個通道做 DMA5 轉換，值存入 adc_buf
```



把 channel 5 所在的 PB13 接地，可變電阻調到最小電阻。經由 debugger 可以看到成功讀取兩個 ADC channel 的採樣值。

此時就可在 while loop 中做其他事情，例如 toggle LED。但你可能會發現並沒有如預期運作。這是因太過頻繁的 DMA 中斷會讓其他程序無法執行。這時有幾種做法讓轉換的頻率降低。

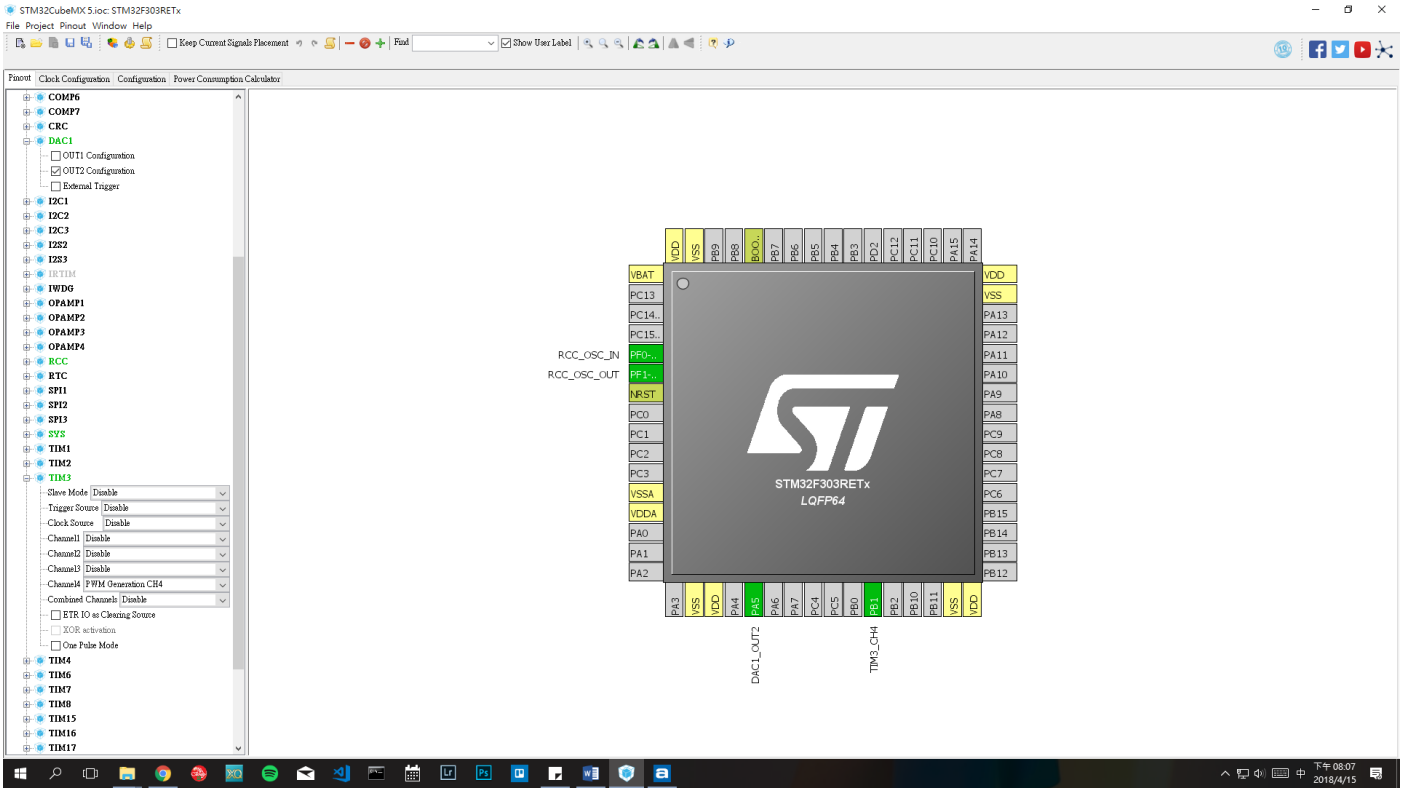
1. 降低 ADC 的 clock frequency。例如調整為 Synchronous clock mode divided by 2。
2. 升高 ADC 的 sampling rate。例如調整為 61.5 cycles。
3. 取消連續轉換(Continuous Conversion Mode = Disable)，並使用 TIM 來定時觸發轉換。



如此使用 DMA，就可以隨時取用那兩個存放採樣值的變數，不用再自己 pollforconversion 及 getvalue。

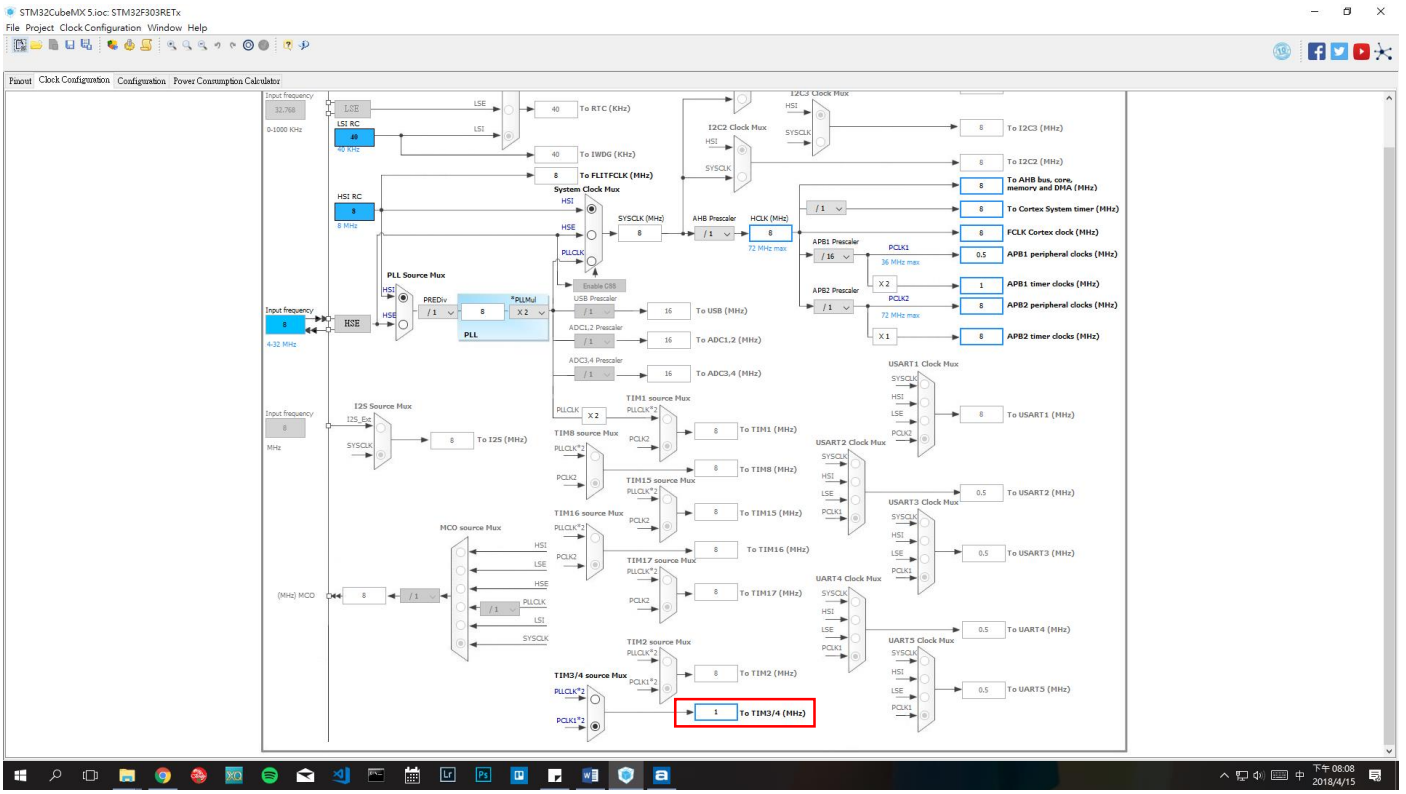
五、 DAC DMA

同樣的，DAC 也可以使用 DMA。在 DMA 模式下，DAC 是由 timer 產生的時鐘訊號控制。這裡我們使用 TIM3_CH4。

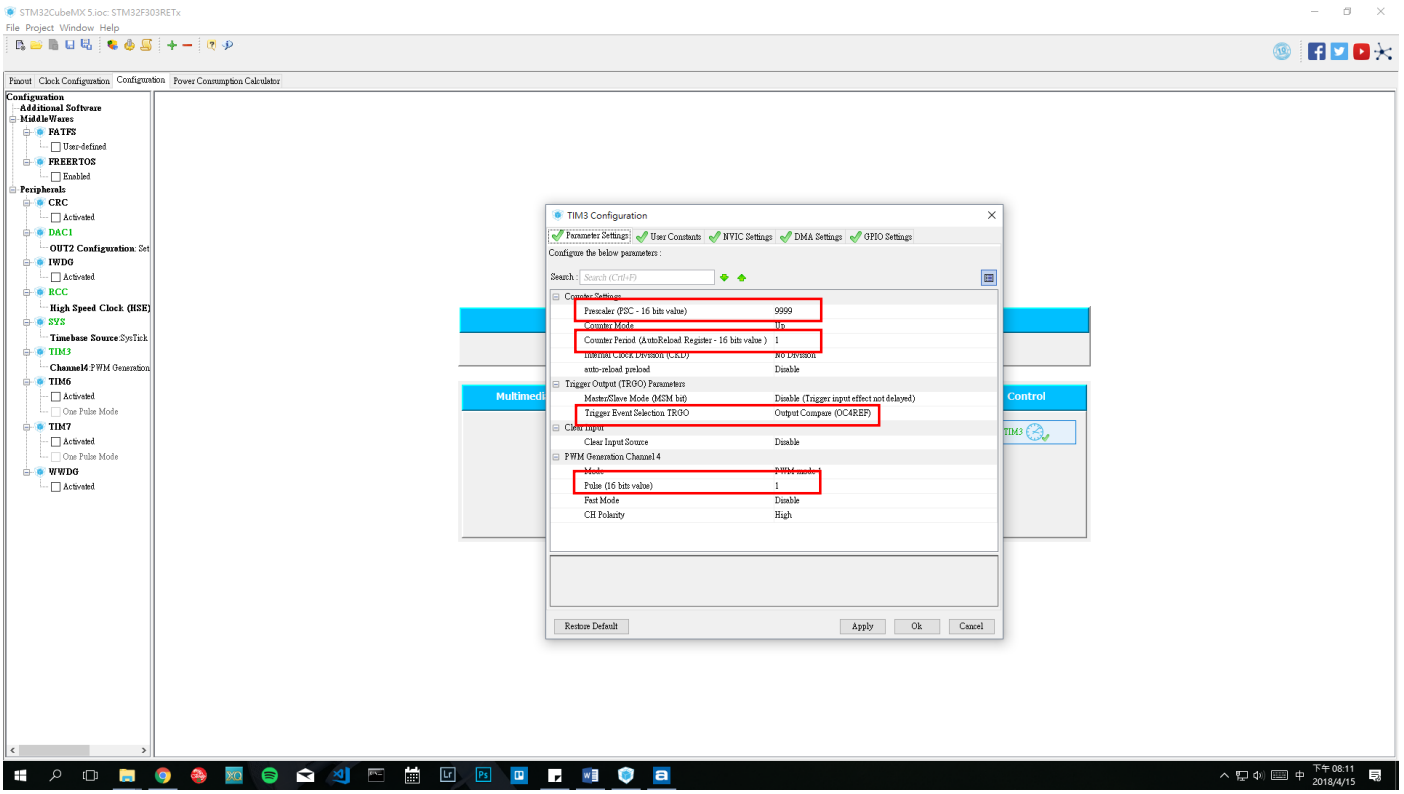


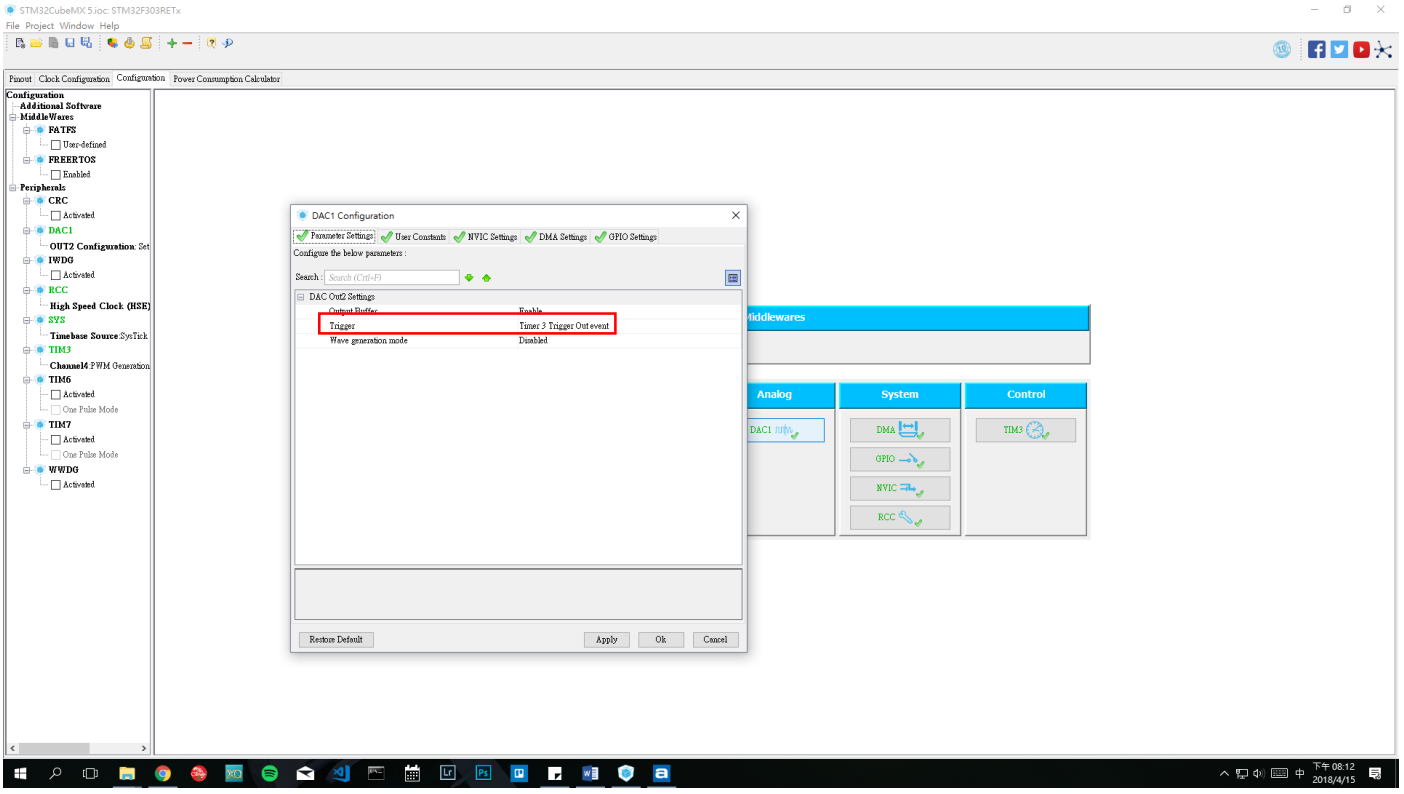
開啟 TIM3_CH4 為 PWM Generation CH4，開啟 DAC1_OUT2

由於 DAC 最大 clock rate 為 1MS/s，因此對 TIM 做以下設定

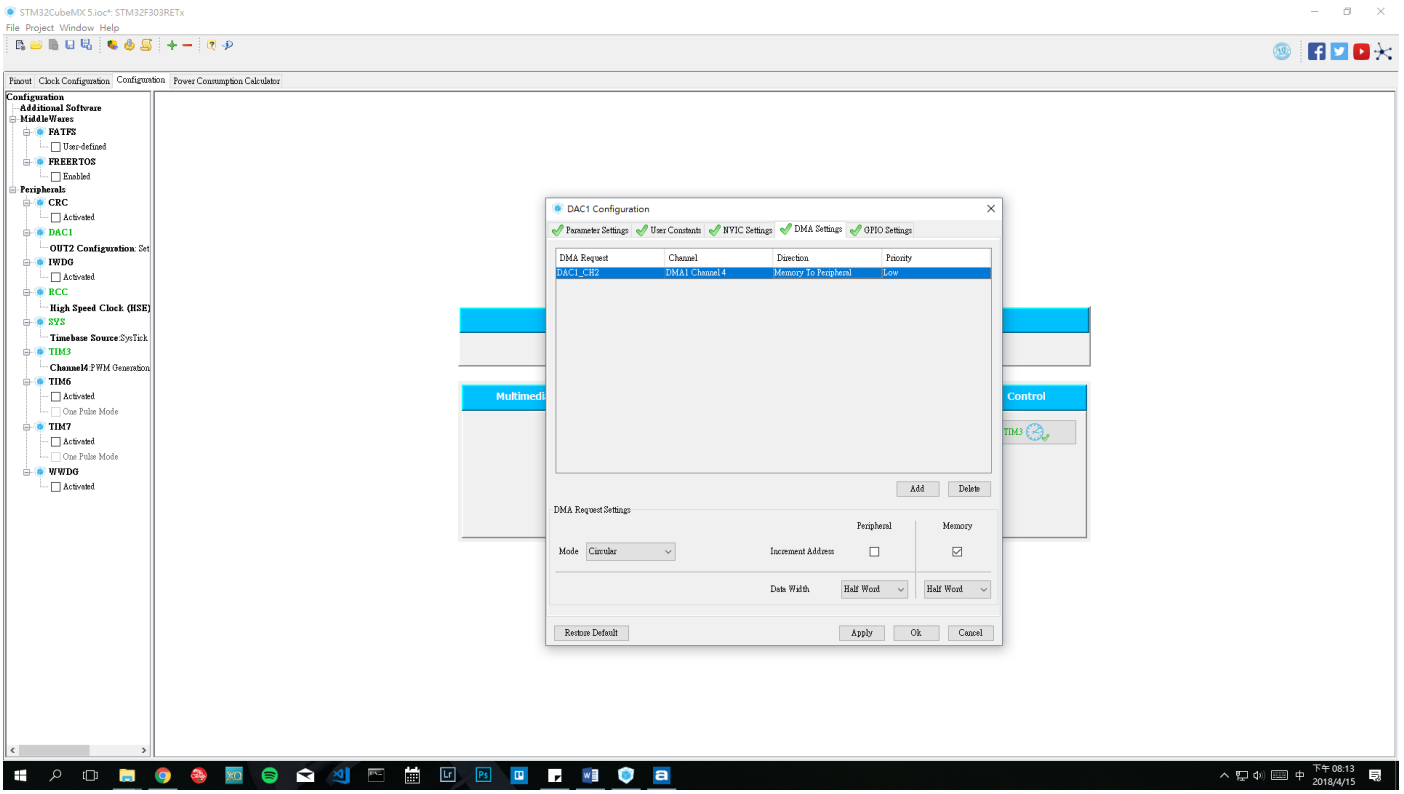


TIM3 clkin 設為 1MHz





將 DAC 的 Trigger 設為 Timer3 Trigger Out Event



開啟 DAC DMA，Mode=Circular

程式重點如下，完整程式請見 github。

//宣告 DMA 使用的 memory

```
uint16_t sine[100]={2048,2112,2176,2240,2303,2364,2425,2484,2541,2597,2650,2701,
2749,2794,2837,2876,2913,2945,2975,3000,3022,3040,3054,3064,3070,3072,3070,3064,
3054,3040,3022,3000,2975,2945,2913,2876,2837,2794,2749,2701,2650,2597,2541,2484,
2425,2364,2303,2240,2176,2112,2048,1984,1920,1856,1793,1732,1671,1612,1555,1499,
1446,1395,1347,1302,1259,1220,1183,1151,1121,1096,1074,1056,1042,1032,1026,1024,
1026,1032,1042,1056,1074,1096,1121,1151,1183,1220,1259,1302,1347,1395,1446,1499,
1555,1612,1671,1732,1793,1856,1920,1984};
```

//main 中啟動 TIM 及 DAC DMA

```
/* USER CODE BEGIN 2 */
HAL_TIM_PWM_Start(&htim3,TIM_CHANNEL_4);
HAL_DAC_Start_DMA(&hdac1,DAC_CHANNEL_2,sine,100,DAC_ALIGN_12B_R);
/* USER CODE END 2 */
```

若要開啟兩個 DAC 且讓他們同步，注意程式必須為「DAC_CH1 → DAC_CH2 → PWM」。如下

```
/* USER CODE BEGIN 2 */
HAL_DAC_Start_DMA(&hdac1,DAC_CHANNEL_1,sine,100,DAC_ALIGN_12B_R);
HAL_DAC_Start_DMA(&hdac1,DAC_CHANNEL_2,sine,100,DAC_ALIGN_12B_R);
HAL_TIM_PWM_Start(&htim3,TIM_CHANNEL_4);
/* USER CODE END 2 */
```

可以用示波器比較先開啟 PWM 及後開啟 PWM，兩者產生出來的訊號差異。