

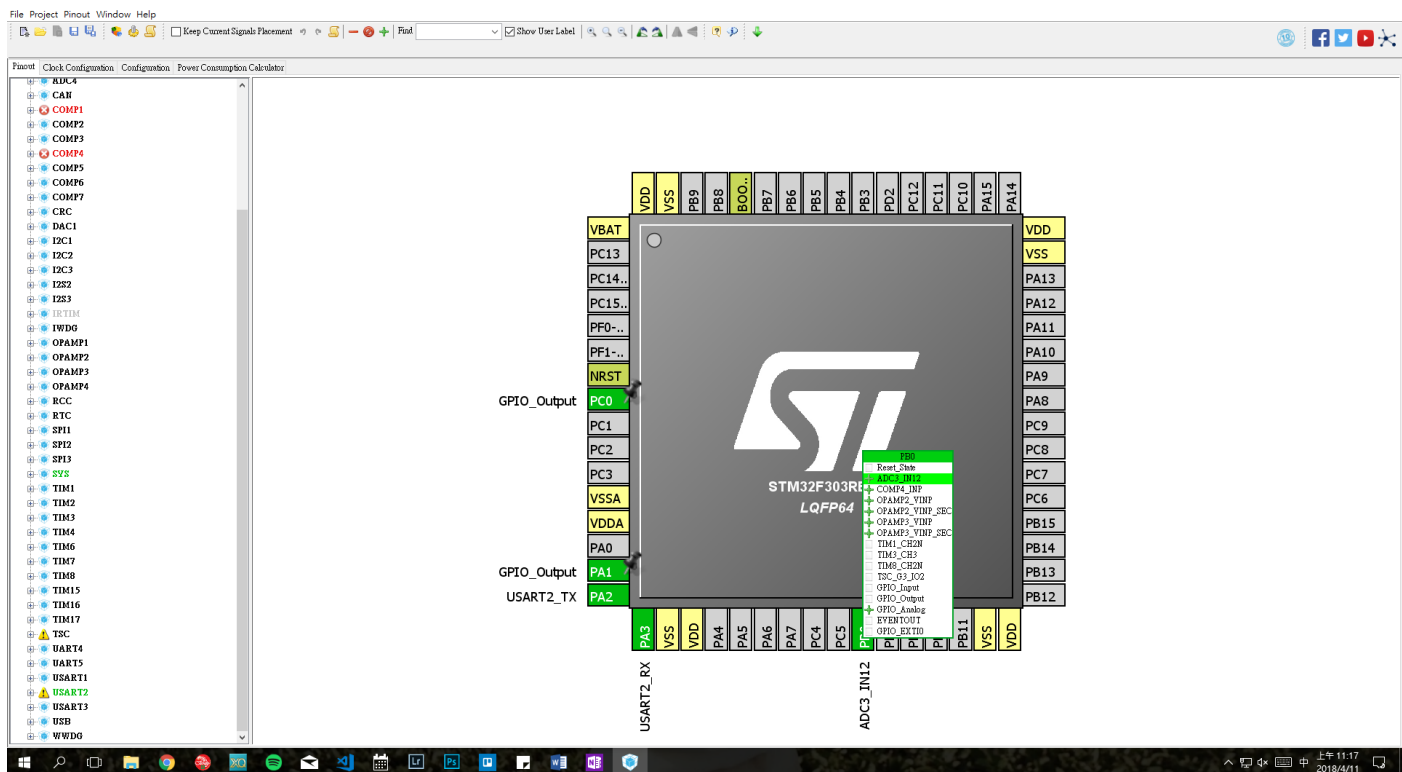
ADC 是將類比轉換至數位的模組。HAL library 裡面已經將大部分的工作做好，我們只需要呼叫 function 就好。

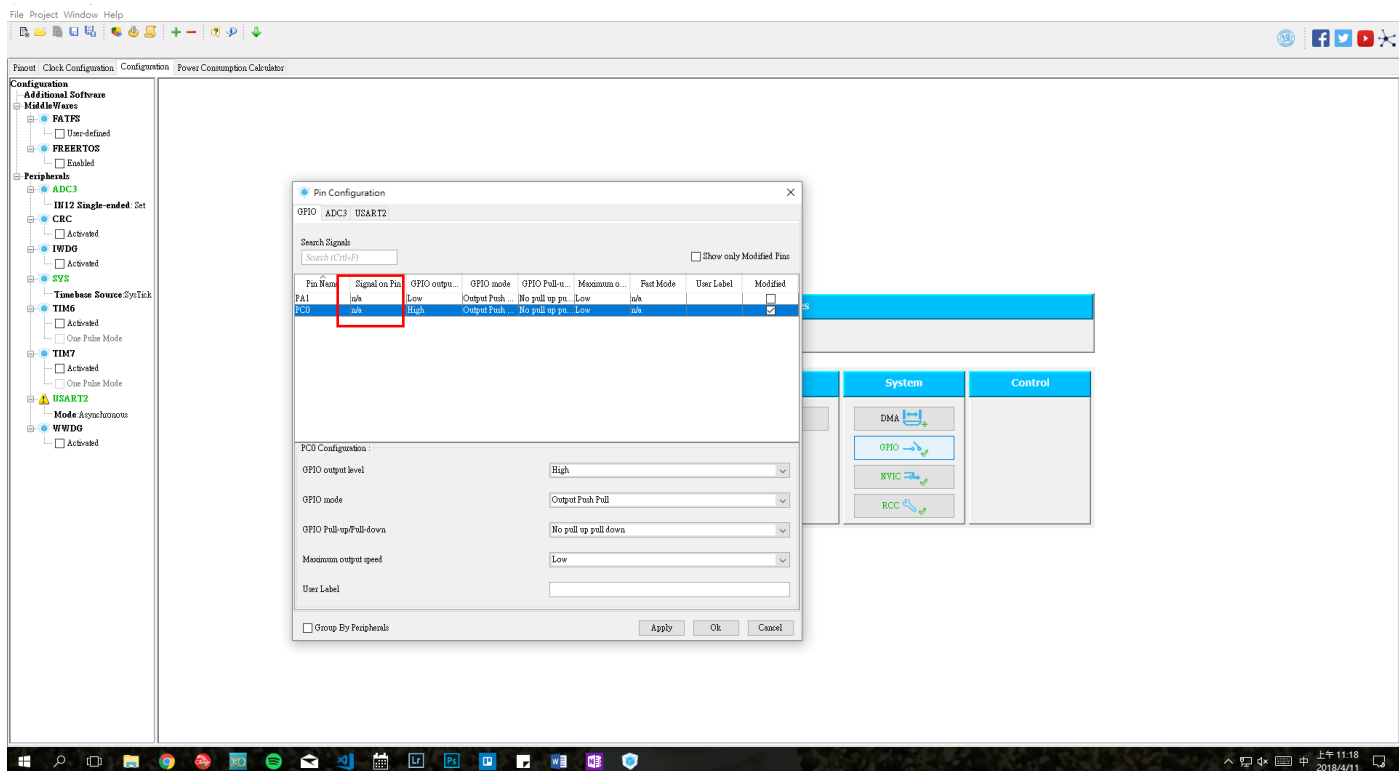
在這裡講到的類比，代表連續的電壓 ( $0\text{v} \sim 3.3\text{v}$ )，數位則是 ADC 模組提供的 solution，STM32F303 的 solution 最高是 12-bit，也就是從  $2^0 \sim 2^{12}$  共 4096 個值。Resolution 可以選擇 12/10/8/6 四種不同值，解析度越低，轉換速度越快但還原後誤差越大。而 ADC 的時鐘(Clock prescaler)預設為 PLL，也可以換成來自 AHB clock 並加上 prescaler。另一個參數為 sampling time，是 ADC 轉換用到的電容的充電時間，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

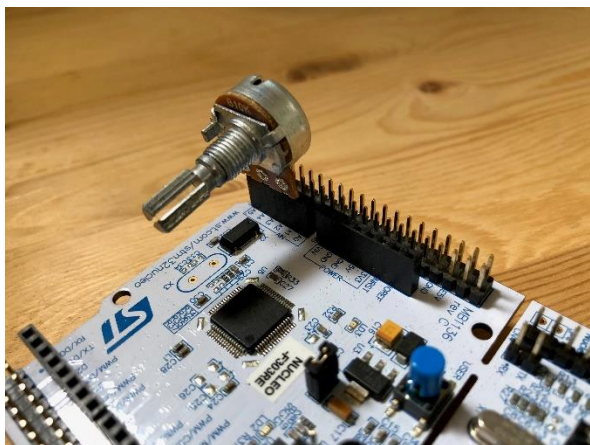
1. ADC 接上可變電阻，讀取 ADC 轉換後的值
2. DAC 將 ADC 的值用 DAC 轉換為類比電壓，當作 LED 的電壓
3. ADC multiple channels
4. ADC multiple channels using DMA
5. DAC DMA

## 一、ADC





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
```

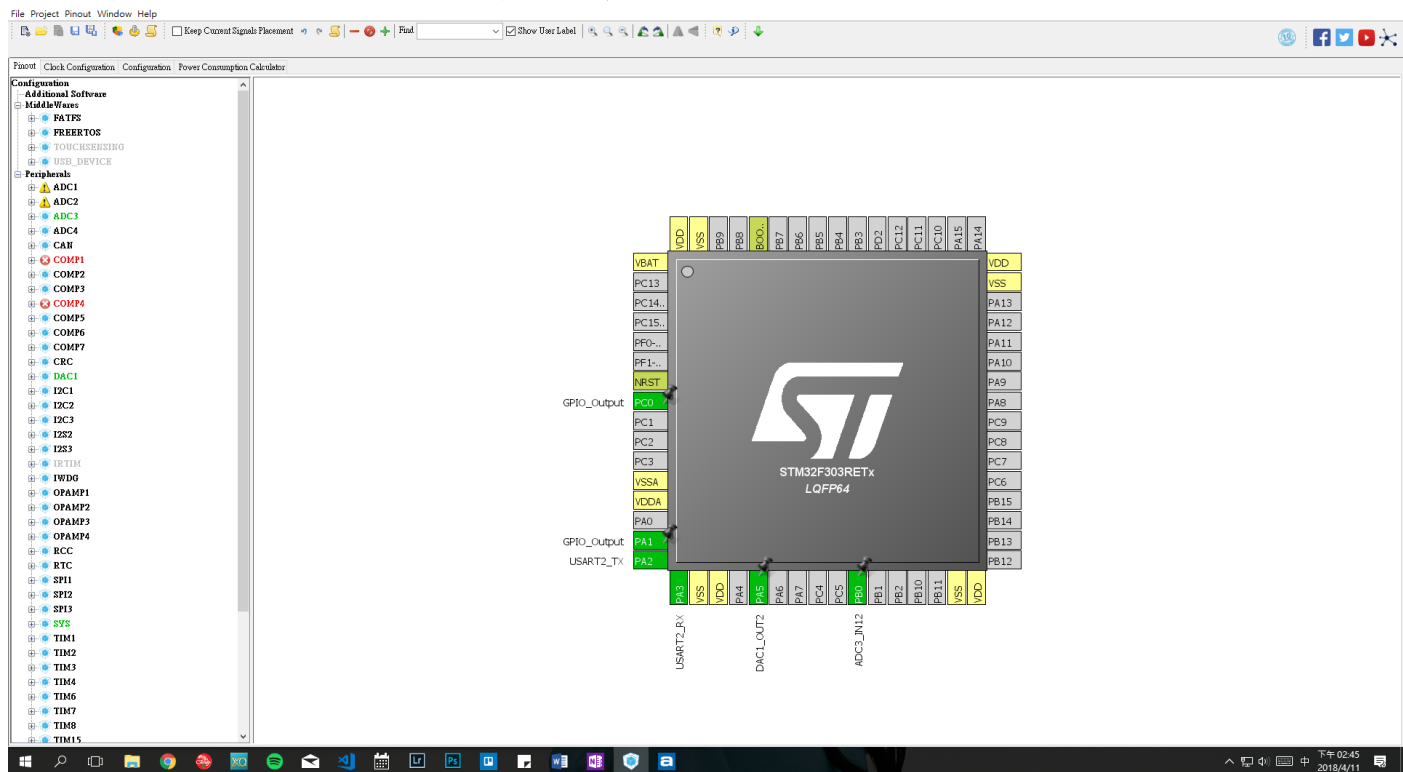
```
uint32_t adc_result = HAL_ADC_GetValue(&hadc3);
```

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
```

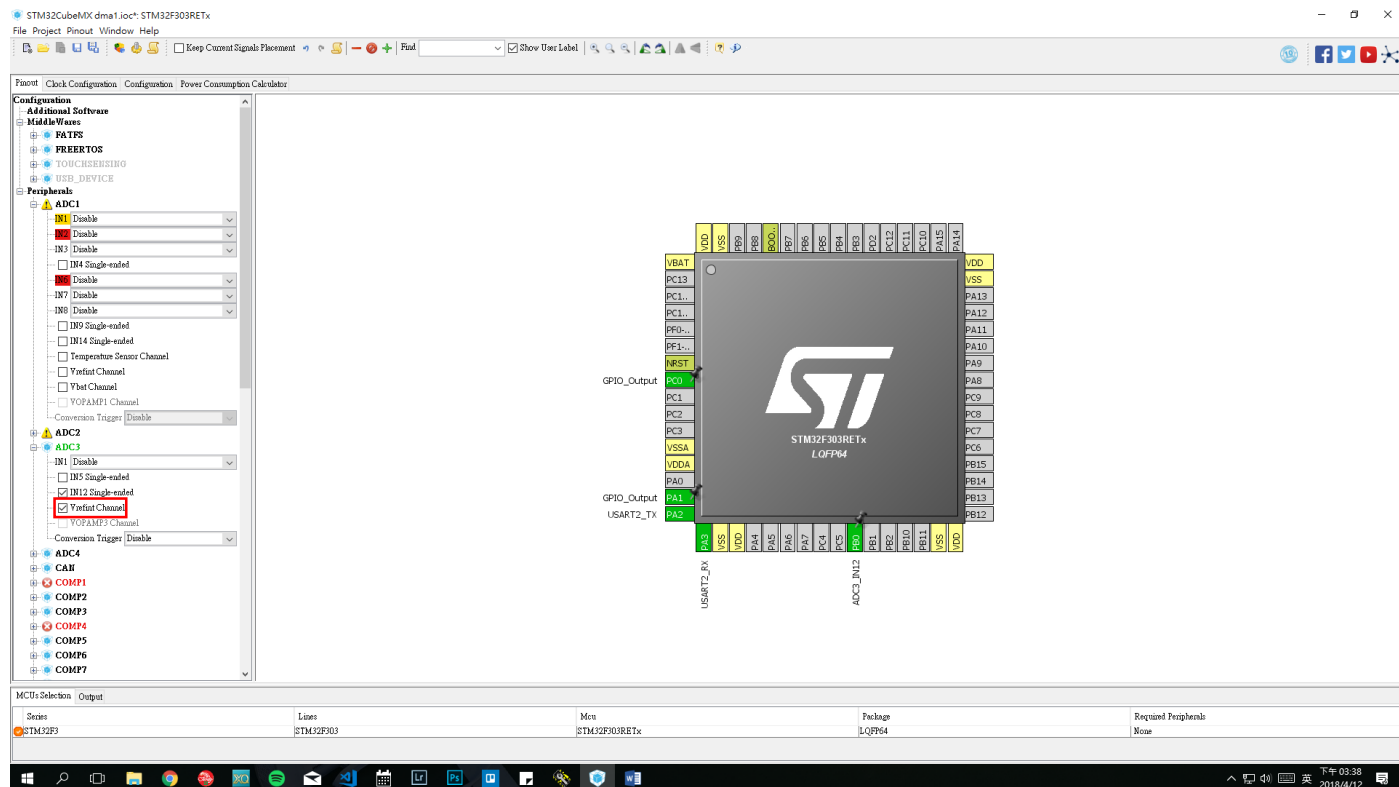
```
HAL_DAC_SetValue(&hdac1, DAC_CHANNEL_2, DAC_ALIGN_12B_R, adc_result);
```

完整程式碼請見 [github](#)。

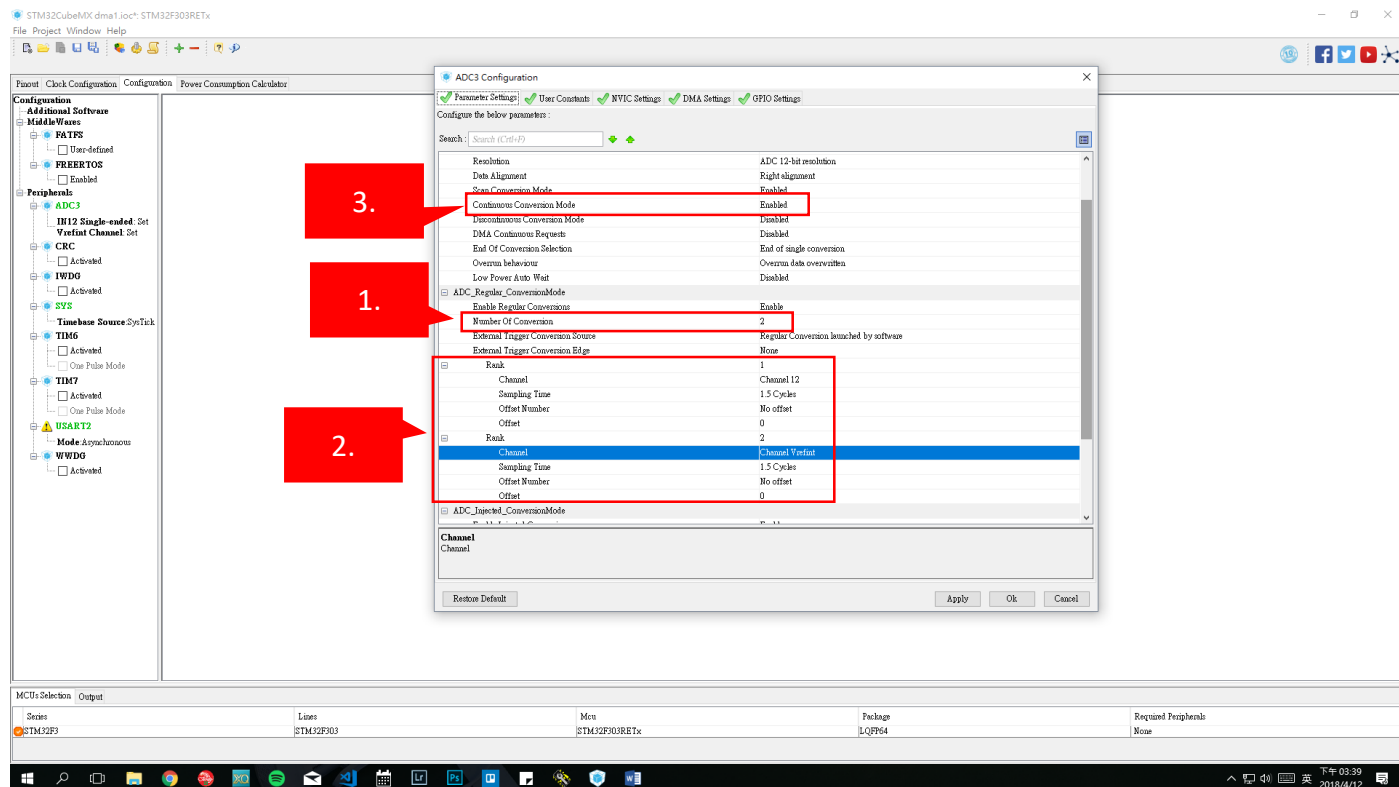
如此可以看到 LED 燈隨著可變電阻的調整而有明暗變化

### 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:

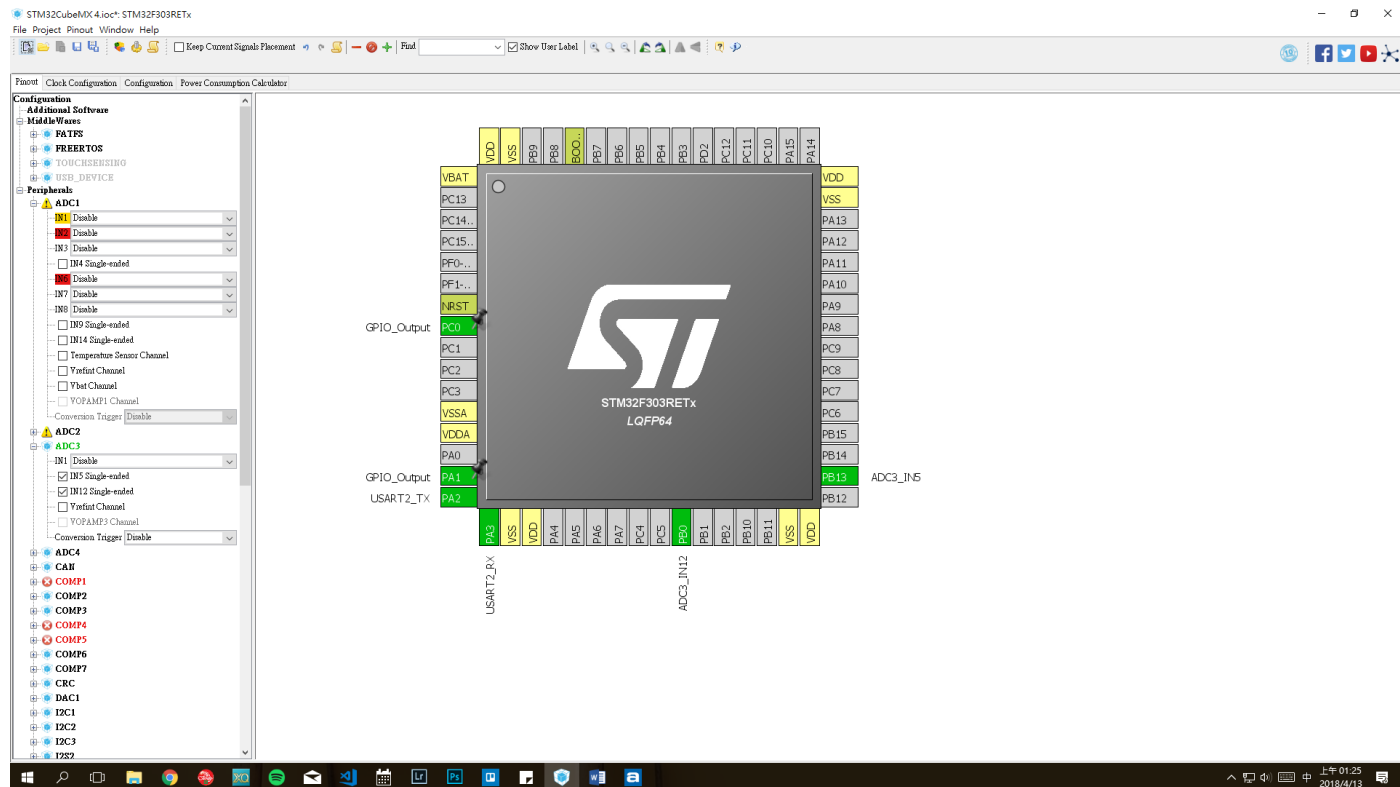
Start → PollForConversion → GetValue (for rank1) → GetValue (for rank2) → Stop

```
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);
```

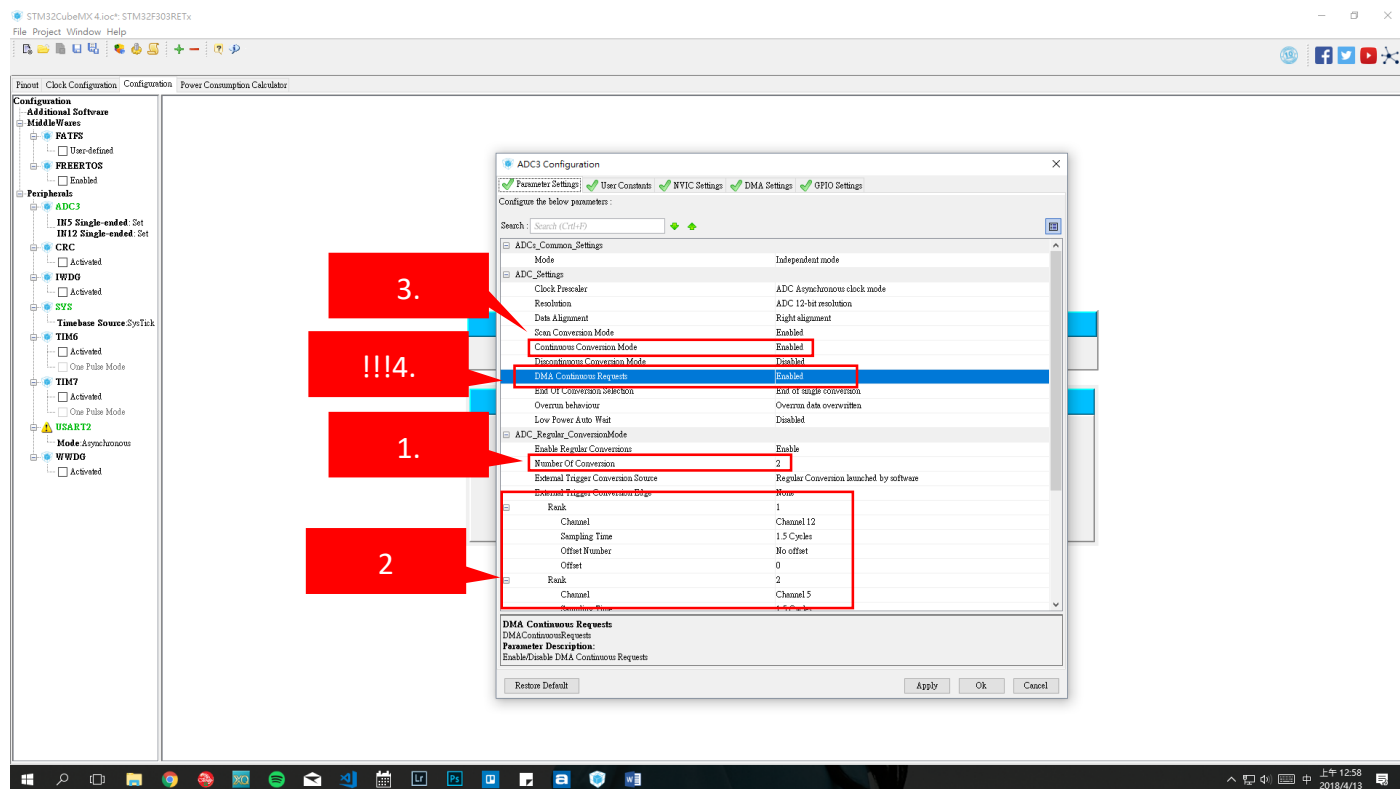
完整程式請見 [github](#)

#### 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。

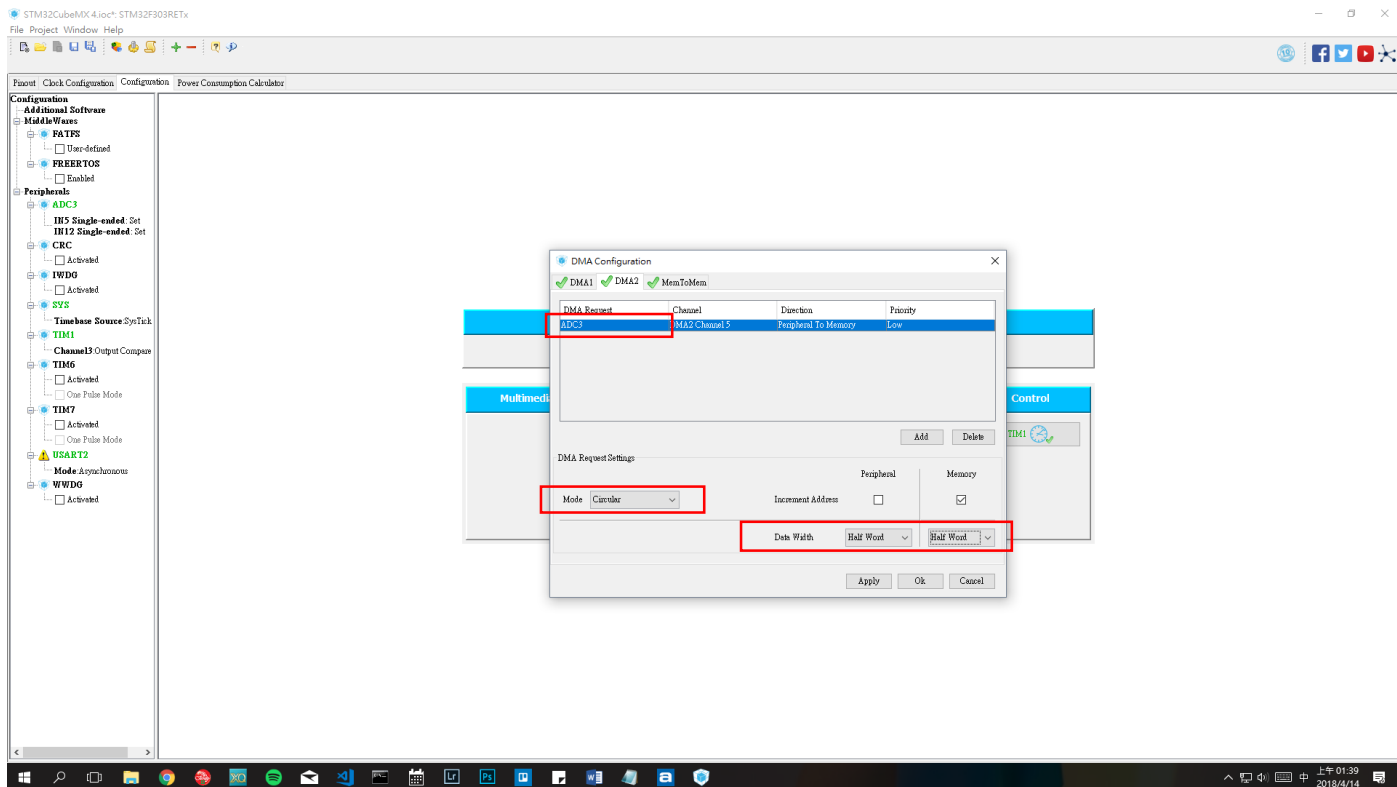


除了第一個範例的 Pin 腳外，另外勾選 ADC3 的 channel 5 (PB13)

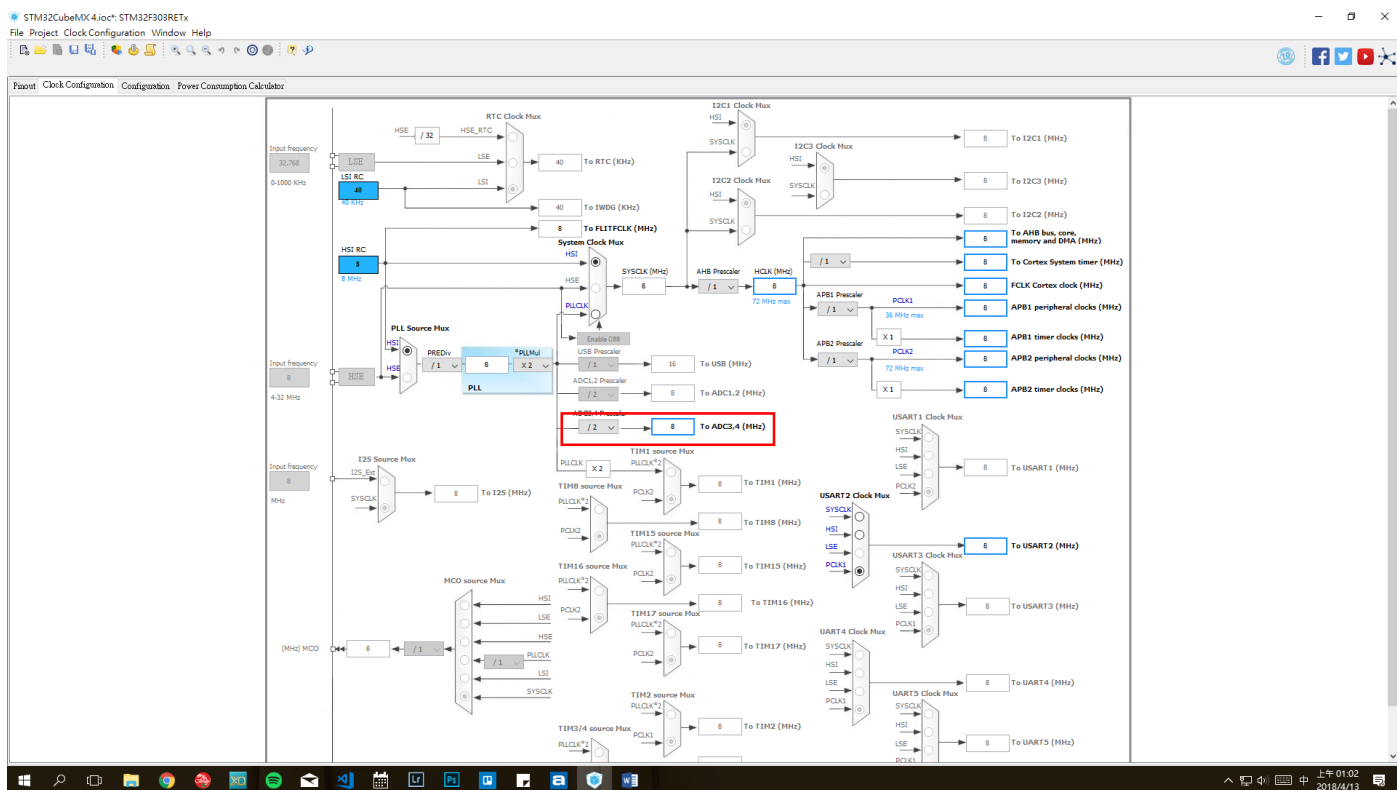


設定前三步跟上一個範例一樣。

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)
```

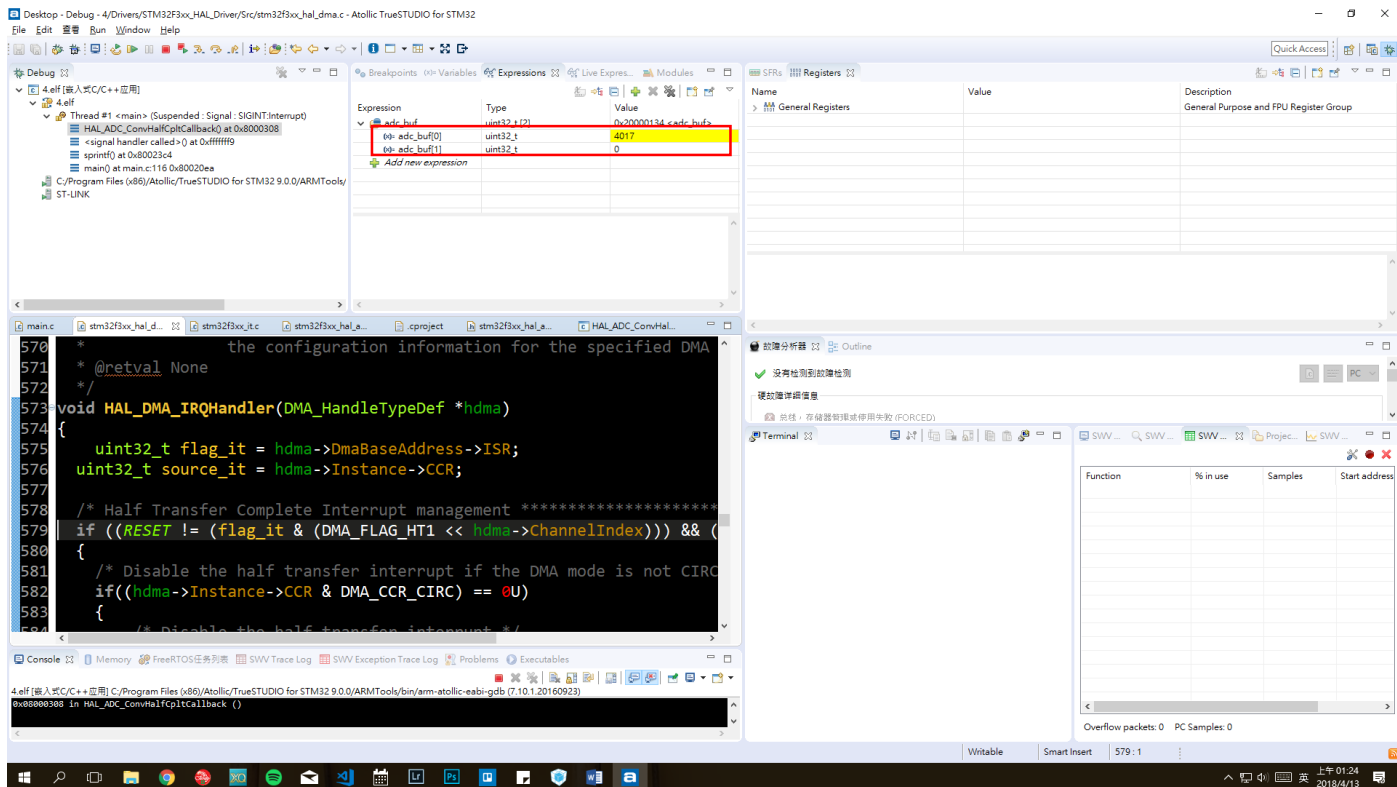
部分 Code，完整請見 [github](#)。

```
/* 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 會在每次轉換完成後被呼叫，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){
}
void HAL_ADC_ConvCpltCallback(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
```

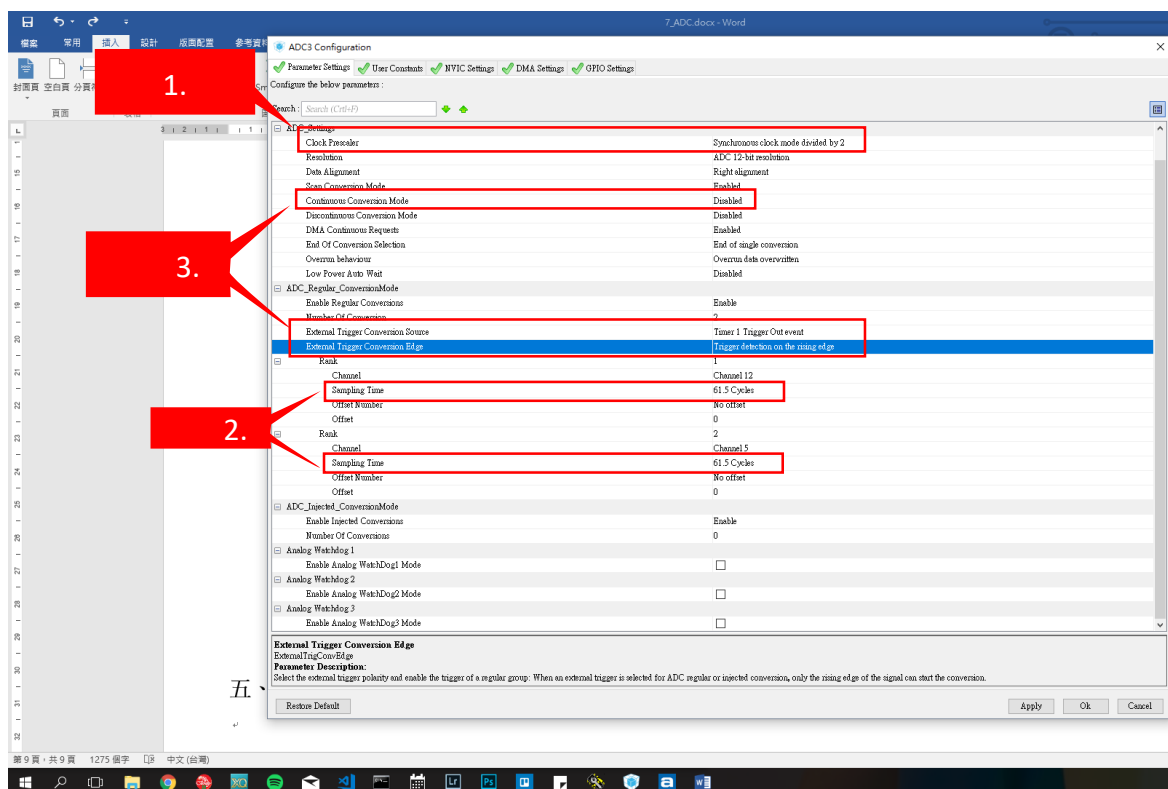




把 channel 5 所在的 PB13 接地，可變電阻調到最小電阻。經由 debugger 可以看到成功讀取兩個 ADC channel 的採樣值。

此時就可以在 main while loop 中做其他事情，例如 toggle LED。你可能會發現 while loop 並沒有如預期般運作。這是因為太過頻繁的 DMA 中斷，會讓其他程序無法到 stack 的頂端被執行。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。  
這時 ADC 的時鐘來源就會是 AHB clock，prescaler=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、