市立楊梅高中 107 學年度優質化計畫

B-3-2 子計畫

務實致用特色實作社群

嵌入式微控制器 STM32L053(ARM)

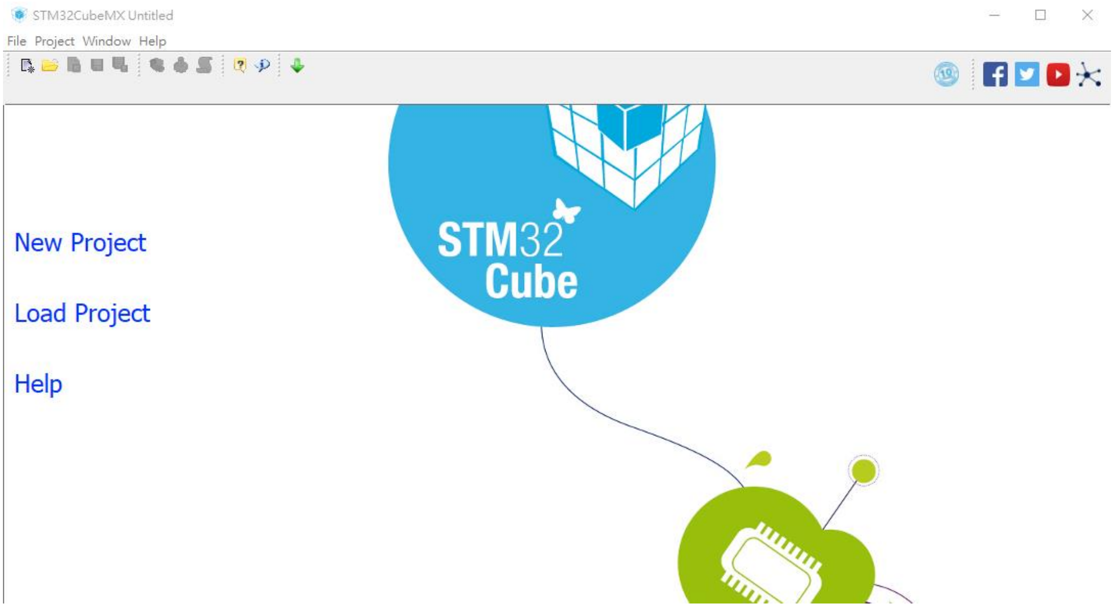
中華民國 108 年 4 月

目　錄

點亮一顆 LED

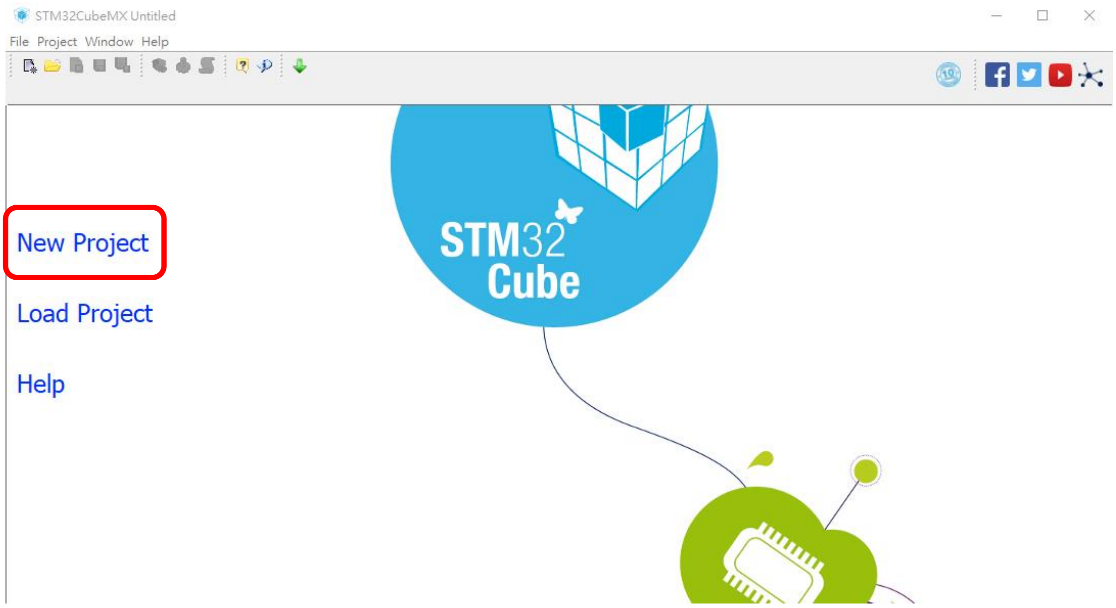
檔案名稱: 01-Start_Led.doc
首先建立一個目錄作為專案(Project)的目錄。本例設為「Start_Led-1」

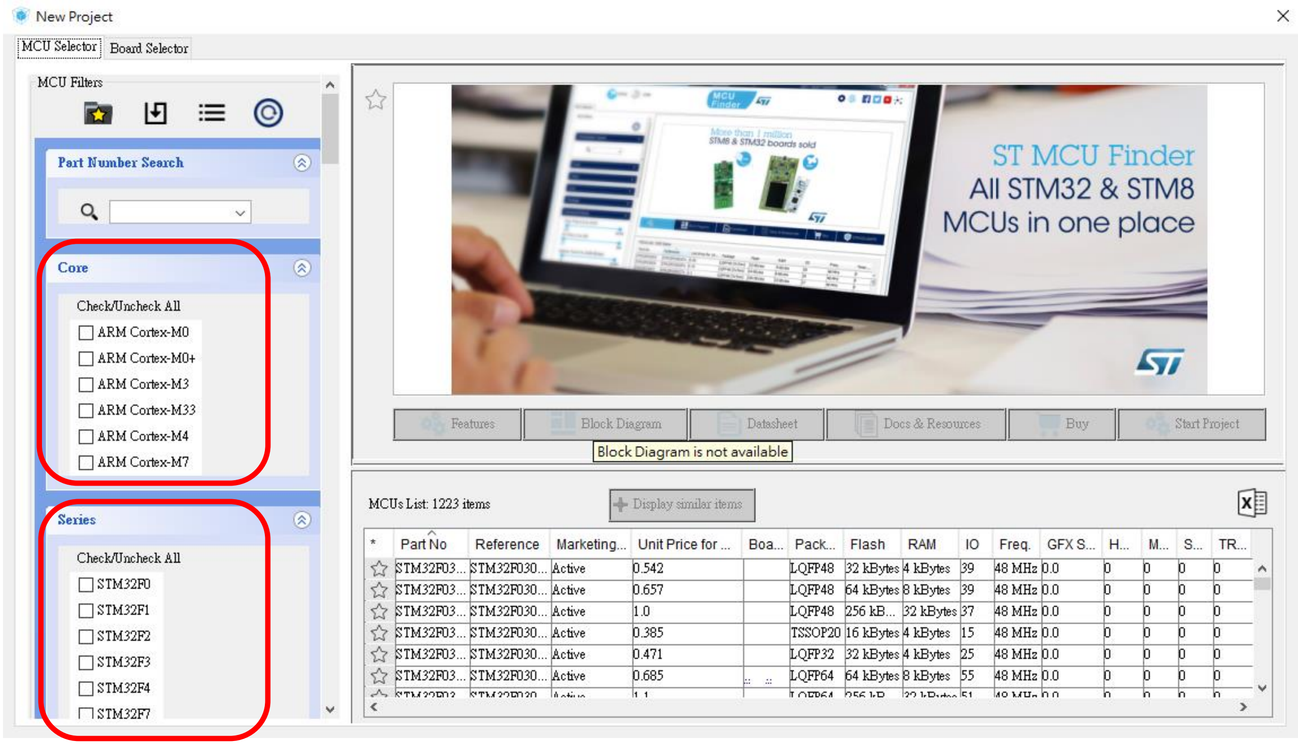
開啟 STM32 的 CUBE 程式如下圖一:



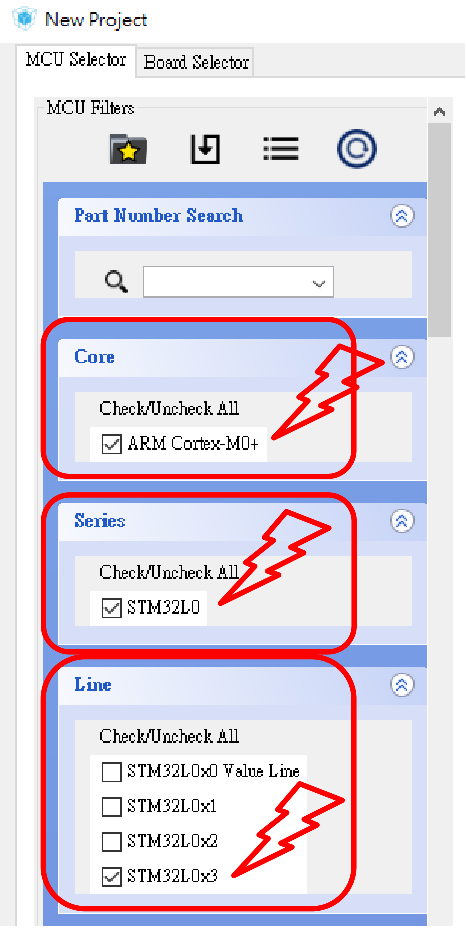
本次使用的版本為 4.27.0，如下圖二:

點選 New Project 選項如下圖

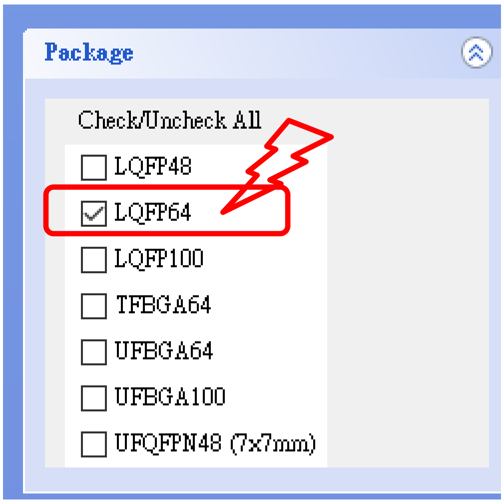




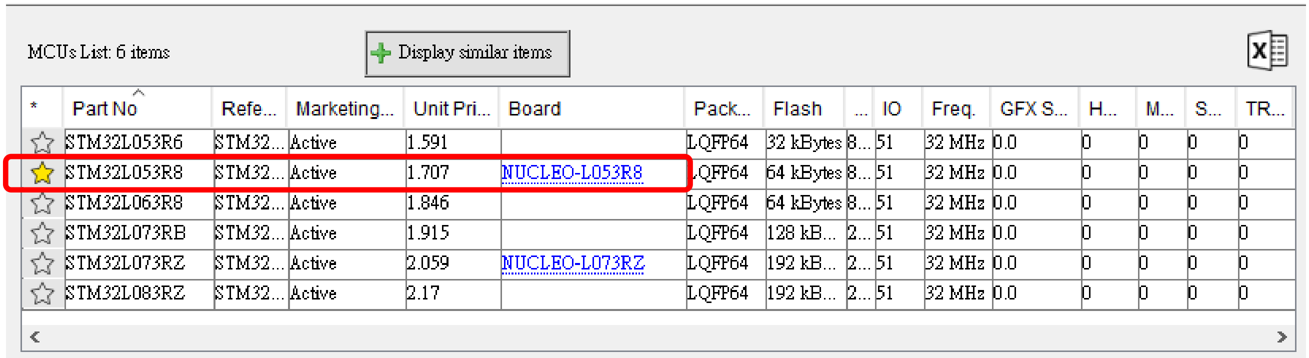
如下圖點選 MCU 的 Core，Serial 及 Line:本範例使用 STM32L053R8T6



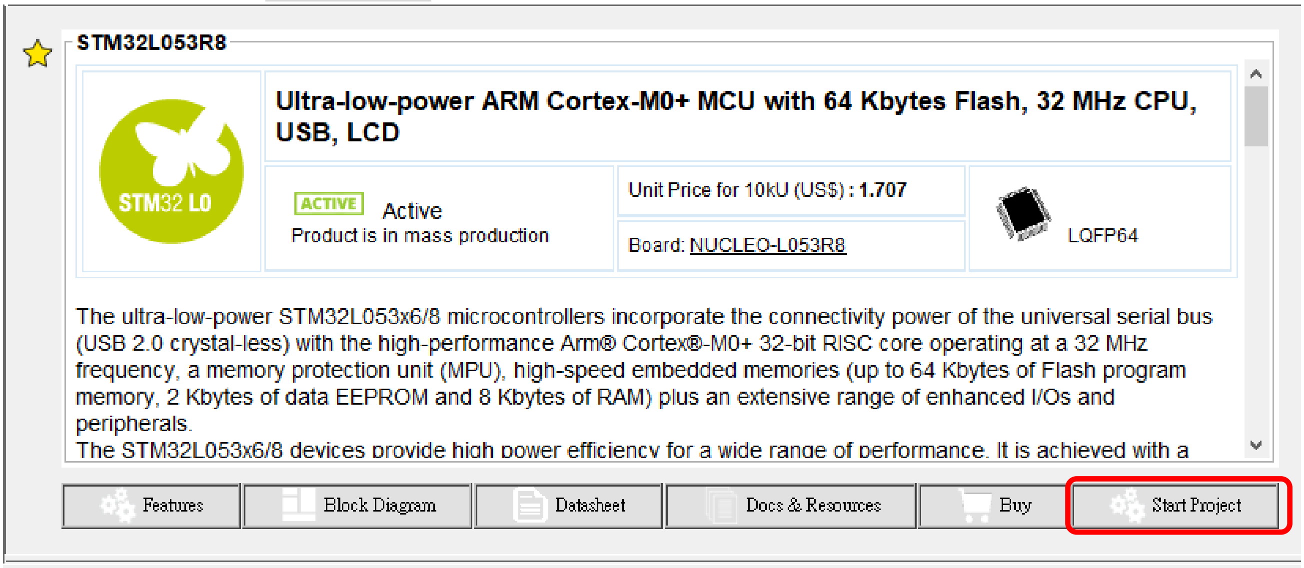
接著再點選 Package(LQFP64)如下圖

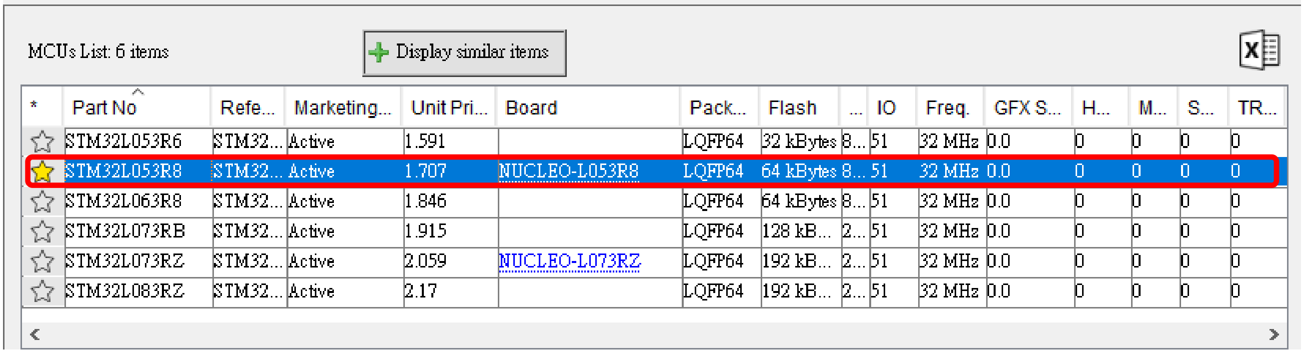


再點選 MCU 的 Mode，

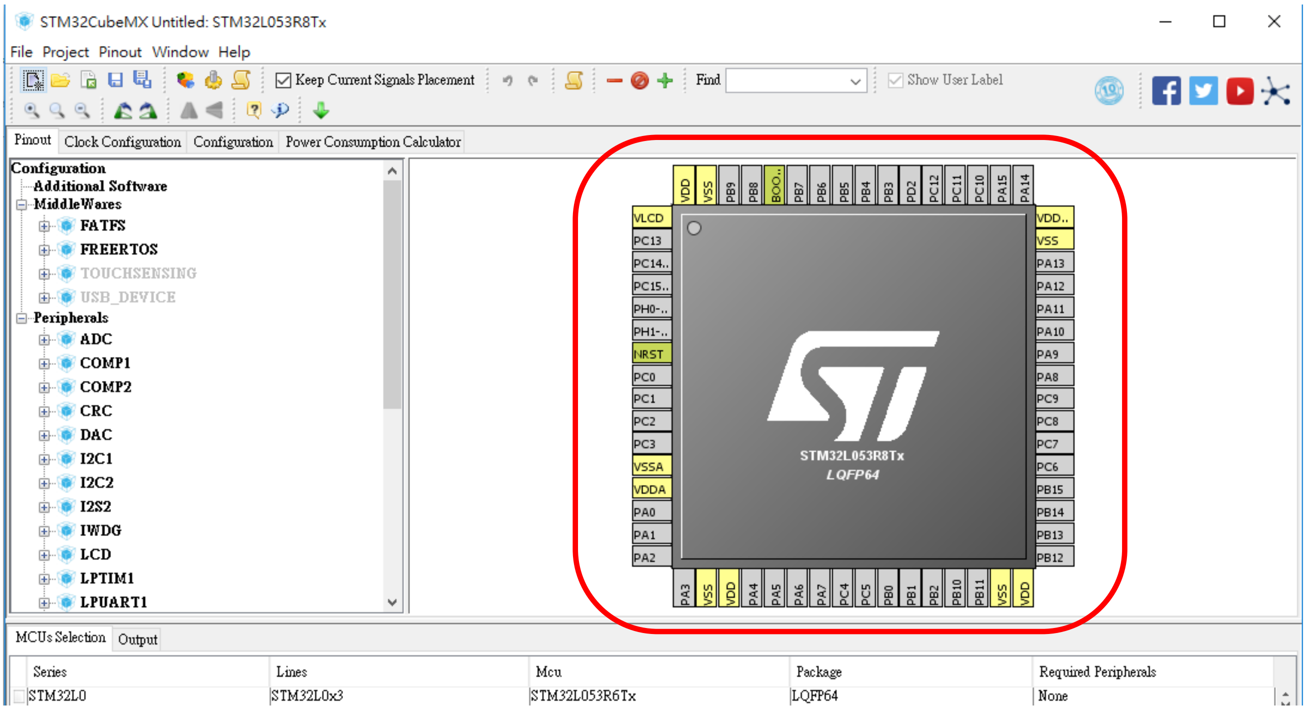


點選完後再選右上 **Start Project**





當點選後顯示如下圖，單須確認是否選到正確的晶片 MCU



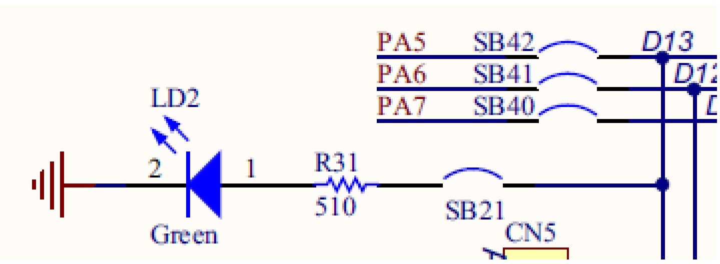
說明:

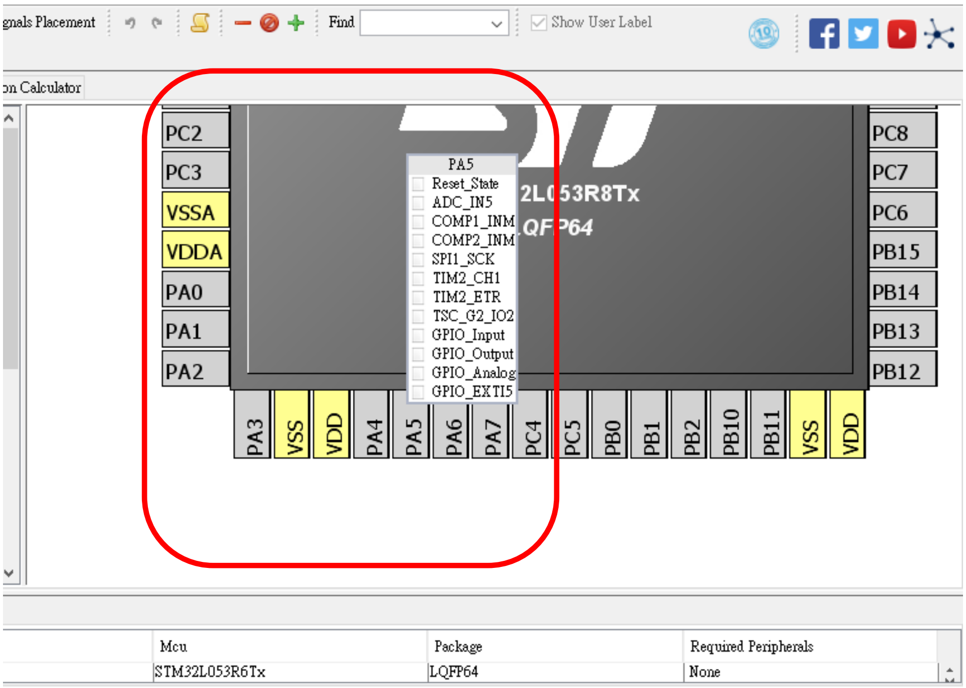
1. 將滑鼠的指標移至晶片的上方或視窗區域，轉動滾輪放大或所小圖示。
2. 按住滑鼠右鍵後，移動滑鼠可以移動 MCU 的位置。

再本範例中欲使用 PA5 的接腳(Pin)作為輸出到 LED 方式:

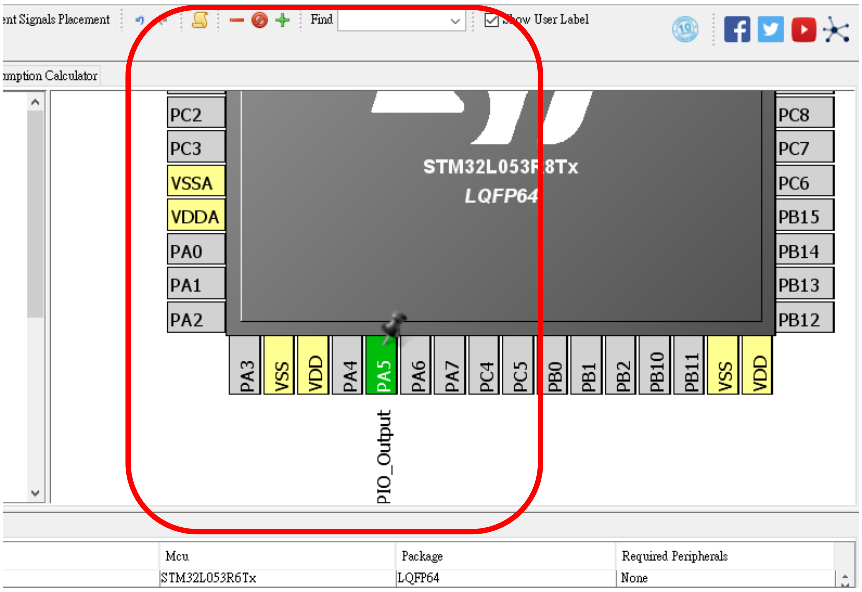
1. 將滑鼠移置 PA5 後按下左鍵。
2. 點選倒數第三項左邊的空格「GPIO Output」

板上線路圖參考如下:

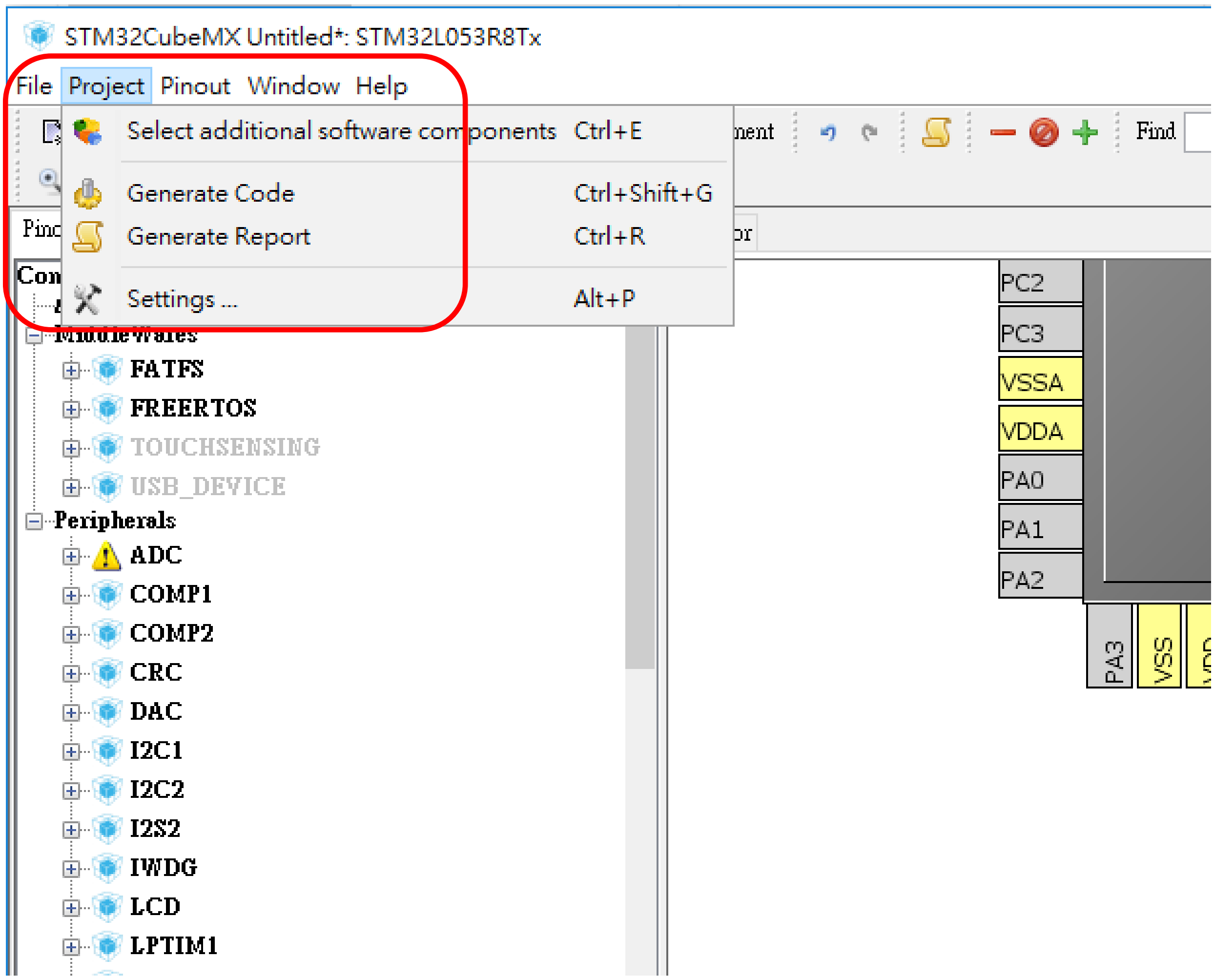



gnals Placement
Find
Show User Label
on Calculator
PC2
PC3
VSSA
VDDA
PA0
PA1
PA2
PA5
Reset_State
ADC_IN5
COMP1_INM
COMP2_INM
SPI1_SCK
TIM2_CH1
TIM2_ETR
TSC_G2_IO2
GPIO_Input
GPIO_Output
GPIO_Analog
GPIO_EXTI5
2L053R8Tx
QFP64
PC8
PC7
PC6
PB15
PB14
PB13
PB12
PA3
VSS
VDD
PA4
PA5
PA6
PA7
PC4
PC5
PB0
PB1
PB2
PB10
PB11
VSS
VDD
Mcu
STM32L053R6Tx
Package
LQFP64
Required Peripherals
None

如下圖示

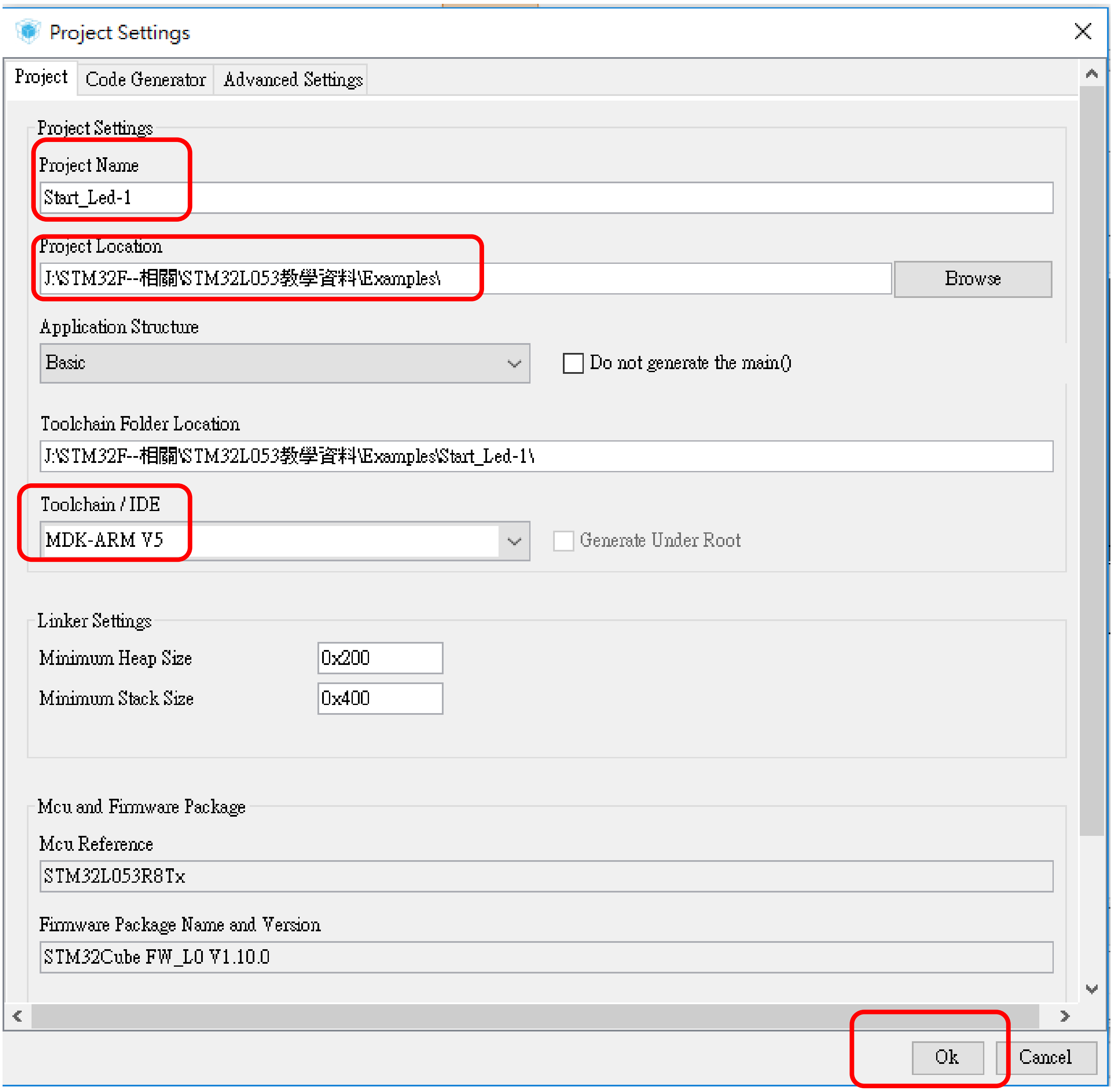

ent Signals Placement
Find
Show User Label
umption Calculator
PC2
PC3
VSSA
VDDA
PA0
PA1
PA2
STM32L053R8Tx
LQFP64
PC8
PC7
PC6
PB15
PB14
PB13
PB12
PA3
VSS
VDD
PA4
PA5
PA6
PA7
PC4
PC5
PB0
PB1
PB2
PB10
PB11
VSS
VDD
PIO_Output
Mcu
STM32L053R6Tx
Package
LQFP64
Required Peripherals
None

點選左上方的 Project →Setting:以便專案的設定



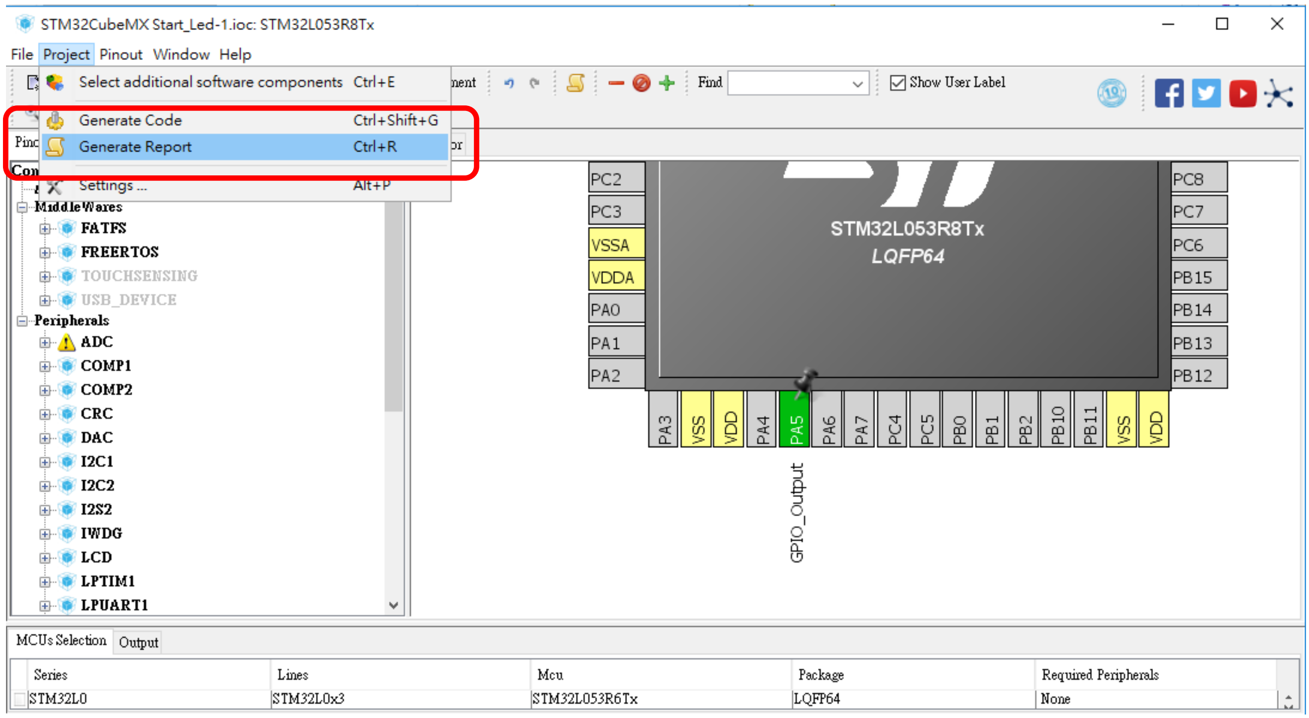
顯示如下圖，並參考如下填入:

1. Project Name
2. Project Location
3. Tool Chain → MDK-ARM V5
4. OK(右下角)

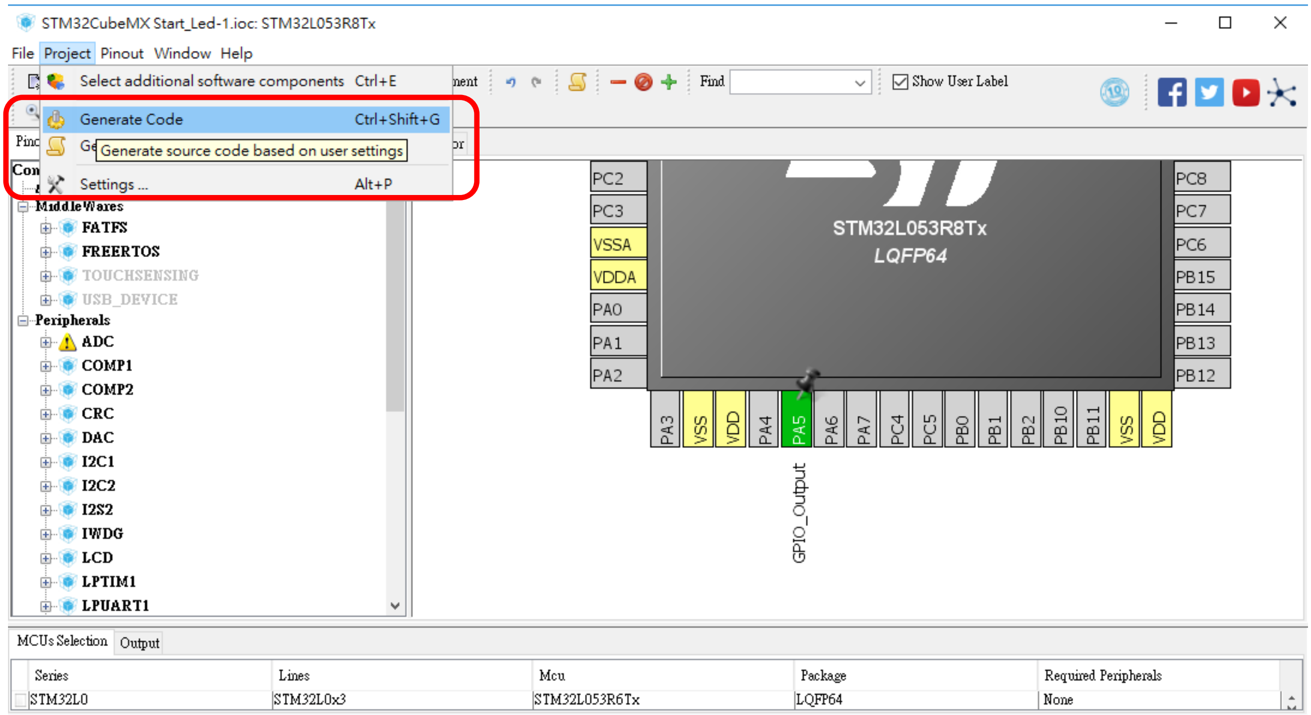


本範例為初次使用自動產生程式碼方式，因此使用其預設值即可。

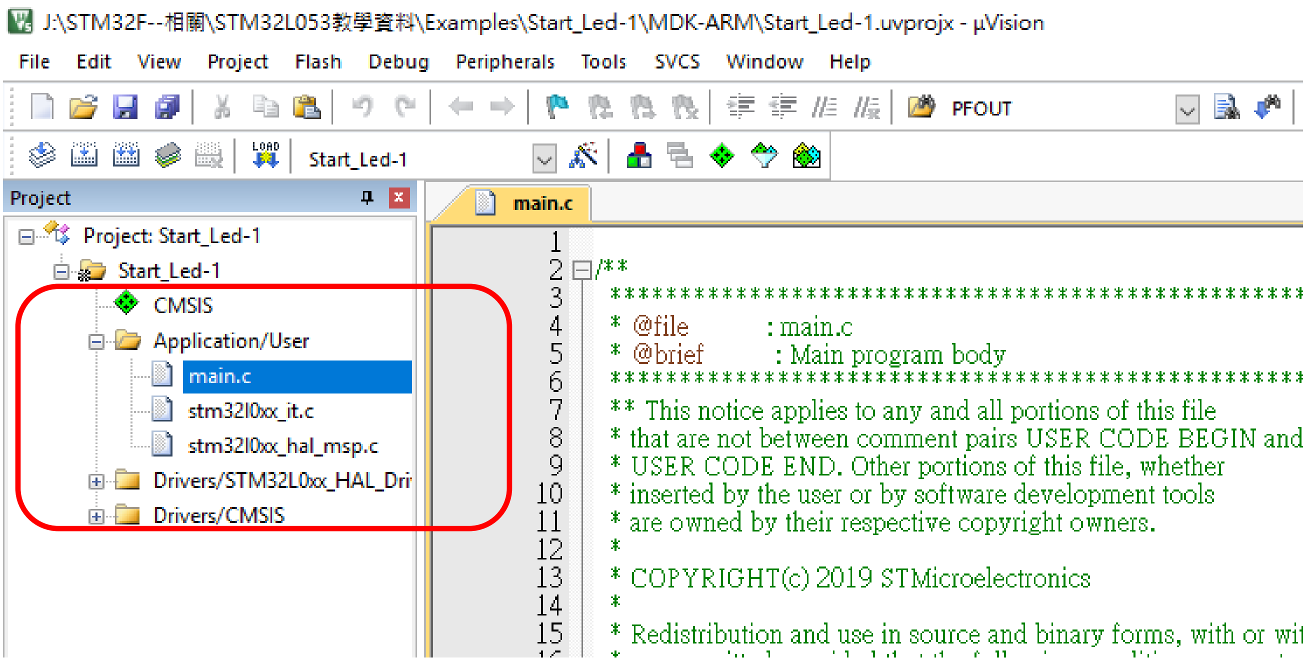
接下來可以利用 Cube 來產生報告。
點選 Project→Generate Report 如下圖:
這個 Report 是 PDF 檔，可以檢視我們設定是否正確，最好在最後設定完成後檢視。



接後點選 Project→Generate Code 來自動產生預設的程式碼



在產生程式碼後開啟後如下圖:



根據 HAL 驅動說明書找到我們需要的 API，然後根據說明添加代碼：（根據說明得到使 PA5 輸出高電平的代碼為 *HAL_GPIO_WritePin (GPIOA, GPIO_PIN_5, GPIO_PIN_SET);*）

25.2.8 HAL_GPIO_WritePin

Function Name	**void HAL_GPIO_WritePin (GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin, GPIO_PinState PinState)**
Function Description	Sets or clears the selected data port bit.
Parameters	• **GPIOx:** where x can be (A..K) to select the GPIO peripheral for STM32F429X device or x can be (A..I) to select the GPIO peripheral for STM32F40XX and STM32F427X devices. • **GPIO_Pin:** specifies the port bit to be written. This parameter can be one of GPIO_PIN_x where x can be (0..15). • **PinState:** specifies the value to be written to the selected bit. This parameter can be one of the GPIO_PinState enum values: GPIO_PIN_RESET: to clear the port pinGPIO_PIN_SET: to set the port pin
Return values	• None
Notes	• This function uses GPIOx_BSRR register to allow atomic read/modify accesses. In this way, there is no risk of an IRQ occurring between the read and the modify access.

輸入程式碼如下:

1. 在 while(1)下方:
   /*USER CODE BEGIN 3*/及/* USER CODE END 3 */中間開始輸入，以避免程式重新產生時被覆蓋。
2. 在程式輸入時會自動出現關鍵字以便選擇。(MDK 5.xx 版後)

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{

/* USER CODE END WHILE */

/* USER CODE BEGIN 3 */
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET); //Low Active ON(板上LED "LED2"-->Active HIGH
  HAL_Delay(1000);   //1秒為 1000個 clock
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
  HAL_Delay(1000);

}
/* USER CODE END 3 */
```

(程式碼會附在後面)

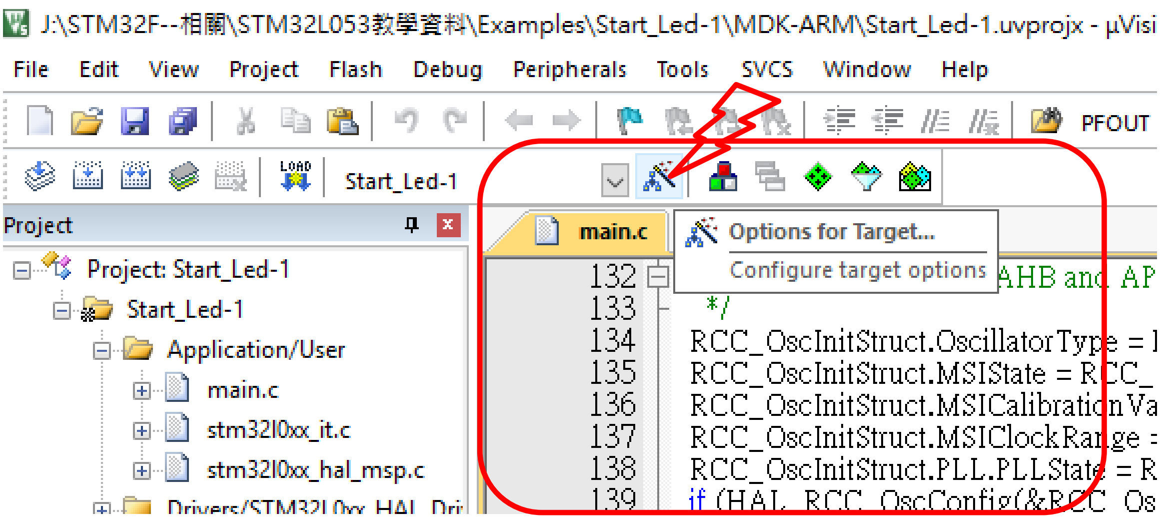
程式碼:

```
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
HAL_Delay(1000);
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
HAL_Delay(1000);
```

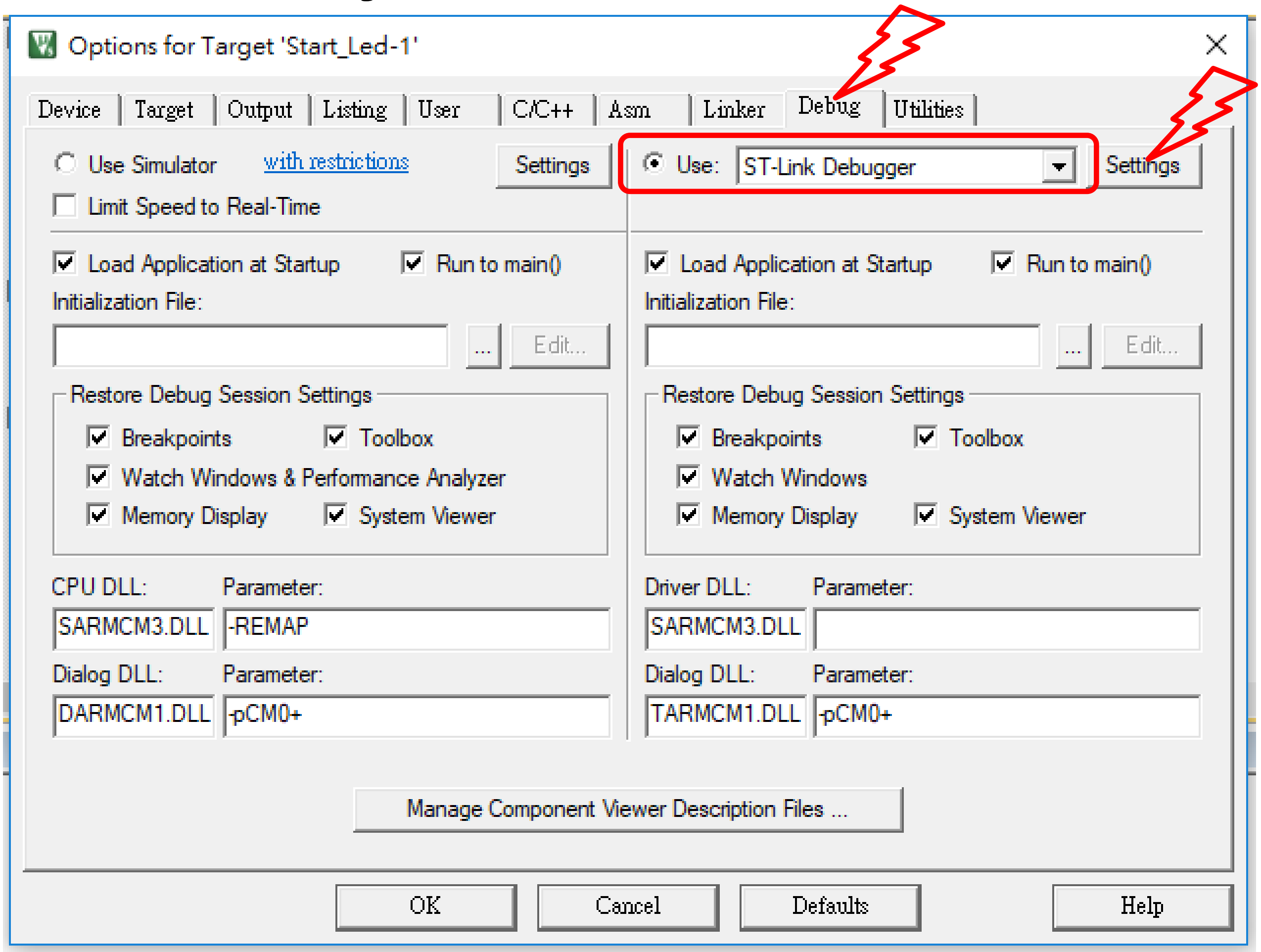
注意:

1. 我們編寫程式時，需將程式碼放在/*USER CODE BEGIN N */ 與 /*USER CODE END N */內，以免在 CUBE 產生程式碼時被覆蓋。
2. 按下 Reset 按鈕開始執行程式，可以看到 LED2 以每秒時間閃爍一次。
3. 亦可事先在 Keil 中設定燒錄完成後自動執行程式。

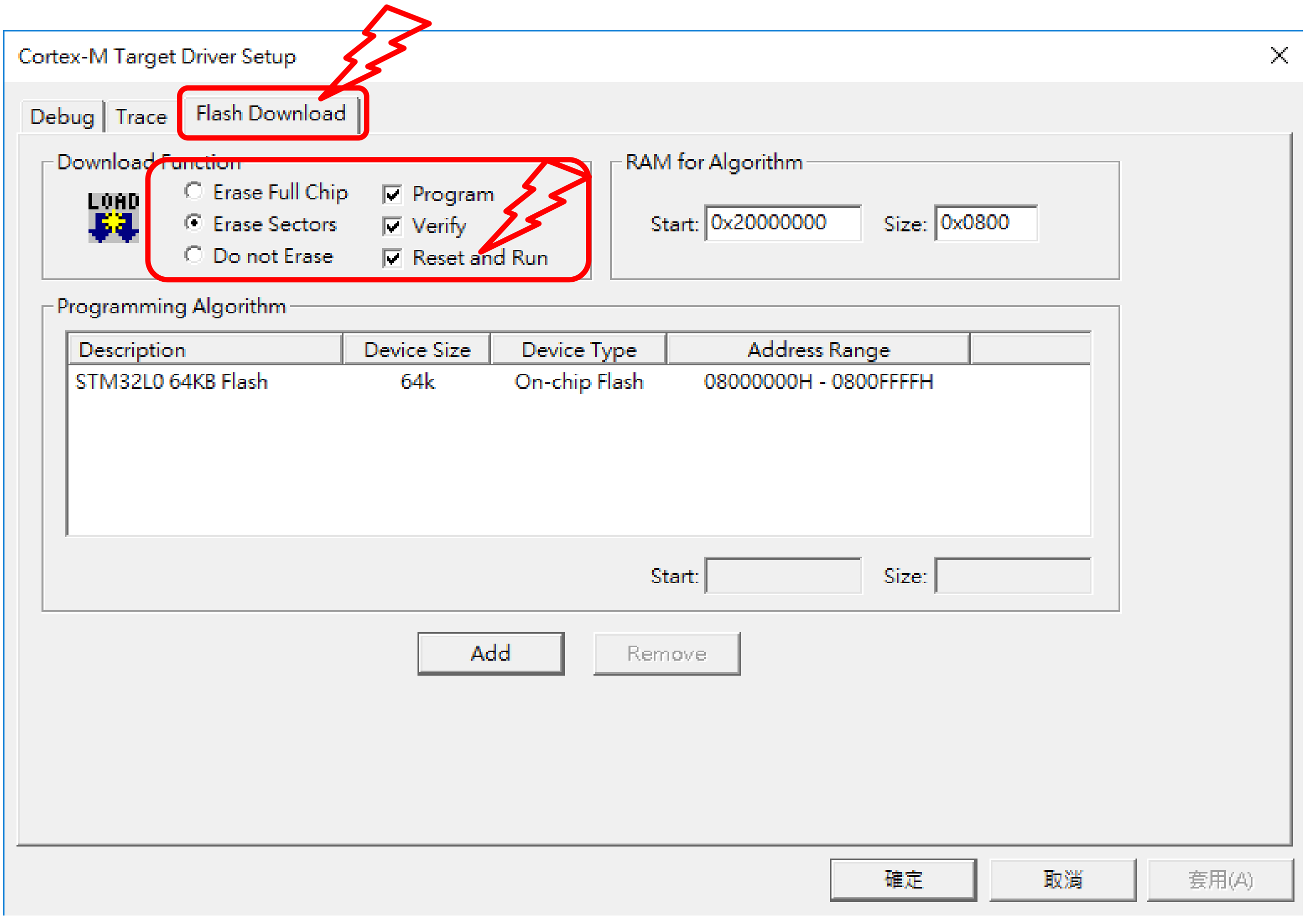
方式:在 MDK 的 Option for Target...(仙女棒) 按鈕點選。



點選 Option for Target...(仙女棒)後即可開啟另一視窗，如下:
接者點選 Debug 後，注意在紅框中需選設在 ST-Link Debugger 或相容的 Tool。
確認後再點選右邊的 settings 選項。



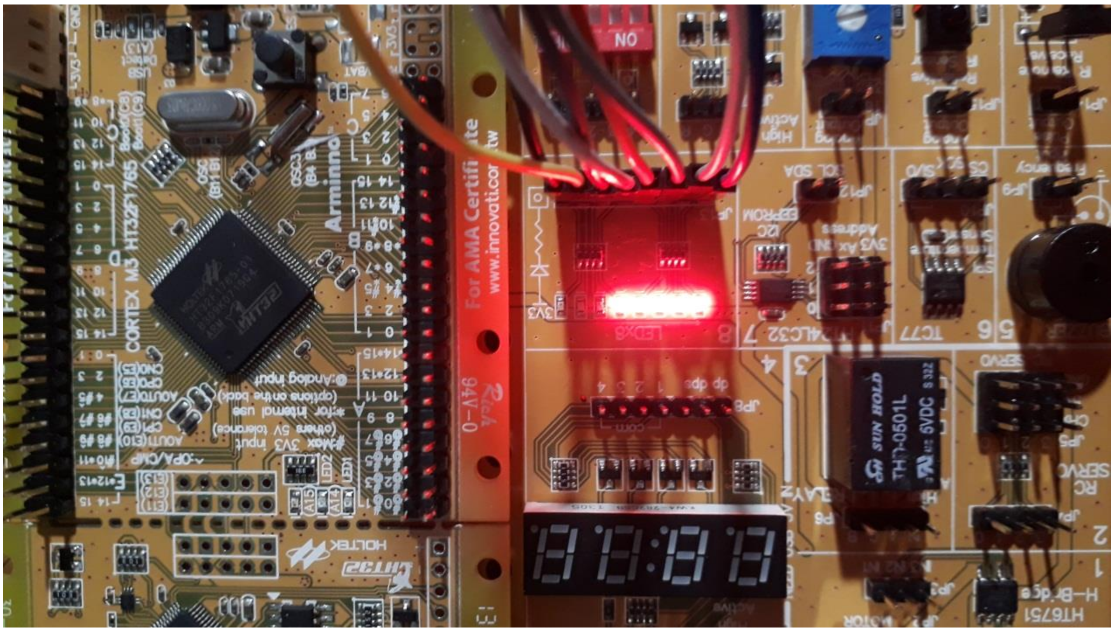
點選上示的選項後，會出現如下示的視窗。
接著點選紅框中的 Flash Download 選項，接著點選下面紅框中的 Reset and Run 選項，這樣會在程式下載完後自動執行，不用再按 Reset 鍵。



跑馬燈

檔案名稱: 02-ledx8-shift.doc

本例可參考前例的練習，說明方式如下：功能要求使 8 顆 LED 由右至左依序點亮，重複執行。如下圖：



程式簡單的方式如下：
HAL_GPIO_WritePin() → 寫入一個值到輸出的腳位。HAL 是指使用 HAL Library 庫。
GPIOC → 是指 ARM 的輸出入腳位群 ”C”，本例即 PC0 ~ PC7。
GPIO_PIN_0 → 是指腳位 0，其餘相同。
GPIO_PIN_SET → 是指 Logic High (1)。
GPIO_PIN_RESET →是指 Logic Low (0)。
HAL_Delay(100) → 使用 HAL Library 的 Delay 函數。括號內的數值是 ms 單位。

將以下的程式碼複製於主程式 main.c
/* Infinite loop */
/* USER CODE BEGIN WHILE */內的 while 函數內。

CODE ：如下

```
/* LED Shift 1 --> Code */
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_0, GPIO_PIN_SET);
```

```
HAL_Delay(100);
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_1, GPIO_PIN_SET);
HAL_Delay(100);
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_2, GPIO_PIN_SET);
HAL_Delay(100);
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_3, GPIO_PIN_SET);
HAL_Delay(100);
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_4, GPIO_PIN_SET);
HAL_Delay(100);
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_5, GPIO_PIN_SET);
HAL_Delay(100);
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, GPIO_PIN_SET);
HAL_Delay(100);
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7, GPIO_PIN_SET);
HAL_Delay(100);


HAL_GPIO_WritePin(GPIOC, GPIO_PIN_0, GPIO_PIN_RESET);
HAL_Delay(100);
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_1, GPIO_PIN_RESET);
HAL_Delay(100);
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_2, GPIO_PIN_RESET);
HAL_Delay(100);
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_3, GPIO_PIN_RESET);
HAL_Delay(100);
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_4, GPIO_PIN_RESET);
HAL_Delay(100);
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_5, GPIO_PIN_RESET);
HAL_Delay(100);
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, GPIO_PIN_RESET);
HAL_Delay(100);
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7, GPIO_PIN_RESET);
HAL_Delay(100);
```

使用中斷控制 LED

檔案名稱: 03-使用中斷控制 LED

一、API 說明

HAL 庫一共包含如下 6 個 IO 操作函數：

1、讀取某個 Pin(接腳)的電位狀態： HAL_GPIO_ReadPin()

2、寫入某個 Pin(接腳)的電位狀態： HAL_GPIO_WritePin()

3、翻轉某個 Pin(接腳)的電位狀態： HAL_GPIO_TogglePin()

4、鎖定某個 Pin(接腳)的配置狀態（直到下次重定）： HAL_GPIO_LockPin()

5、外部中斷服務函數： HAL_GPIO_EXTI_IRQHandler()

6、外部中斷回呼函數： HAL_GPIO_EXTI_Callback()

具體使用方法參見 **User Manual-STM32L0 HAL and Low Layer drivers**。

2.11.2 GPIOs

GPIO HAL APIs are the following:

- HAL_GPIO_Init() / HAL_GPIO_DeInit()
- HAL_GPIO_ReadPin() / HAL_GPIO_WritePin()
- HAL_GPIO_TogglePin ().

46/1466 DocID026232 Rev 6

registers.

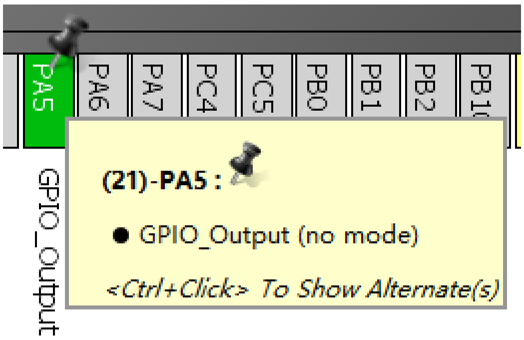
5. To get the level of a pin configured in input mode use HAL_GPIO_ReadPin().
6. To set/reset the level of a pin configured in output mode use HAL_GPIO_WritePin()/HAL_GPIO_TogglePin().
7. To lock pin configuration until next reset use HAL_GPIO_LockPin().
8. During and just after reset, the alternate functions are not active and the GPIO pins are configured in input floating mode (except JTAG pins).

二、GPIO 使用示例

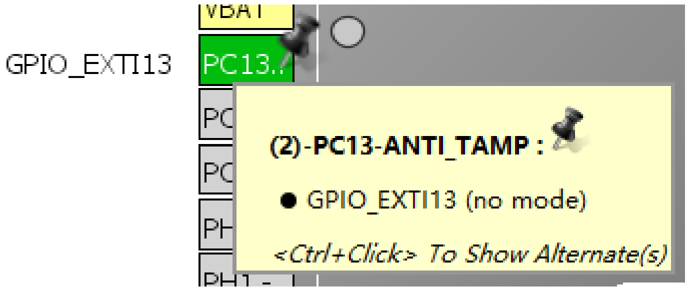
示例功能：使用按鍵（PC13）翻轉 LED（PA5）電位狀態。

1、使用 STM32CubeMX 配置好 Pin(接腳)功能以及巢狀中斷控制器：

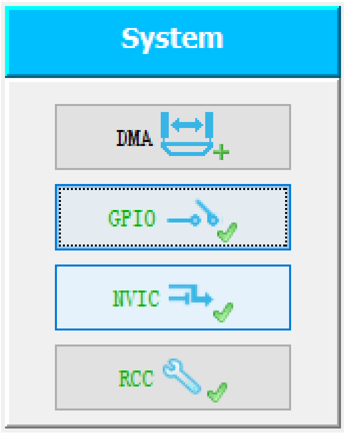
配置 LED 燈 Pin(接腳)為輸出模式



將按鍵 Pin(接腳)掛接到外部中斷 13 號線上

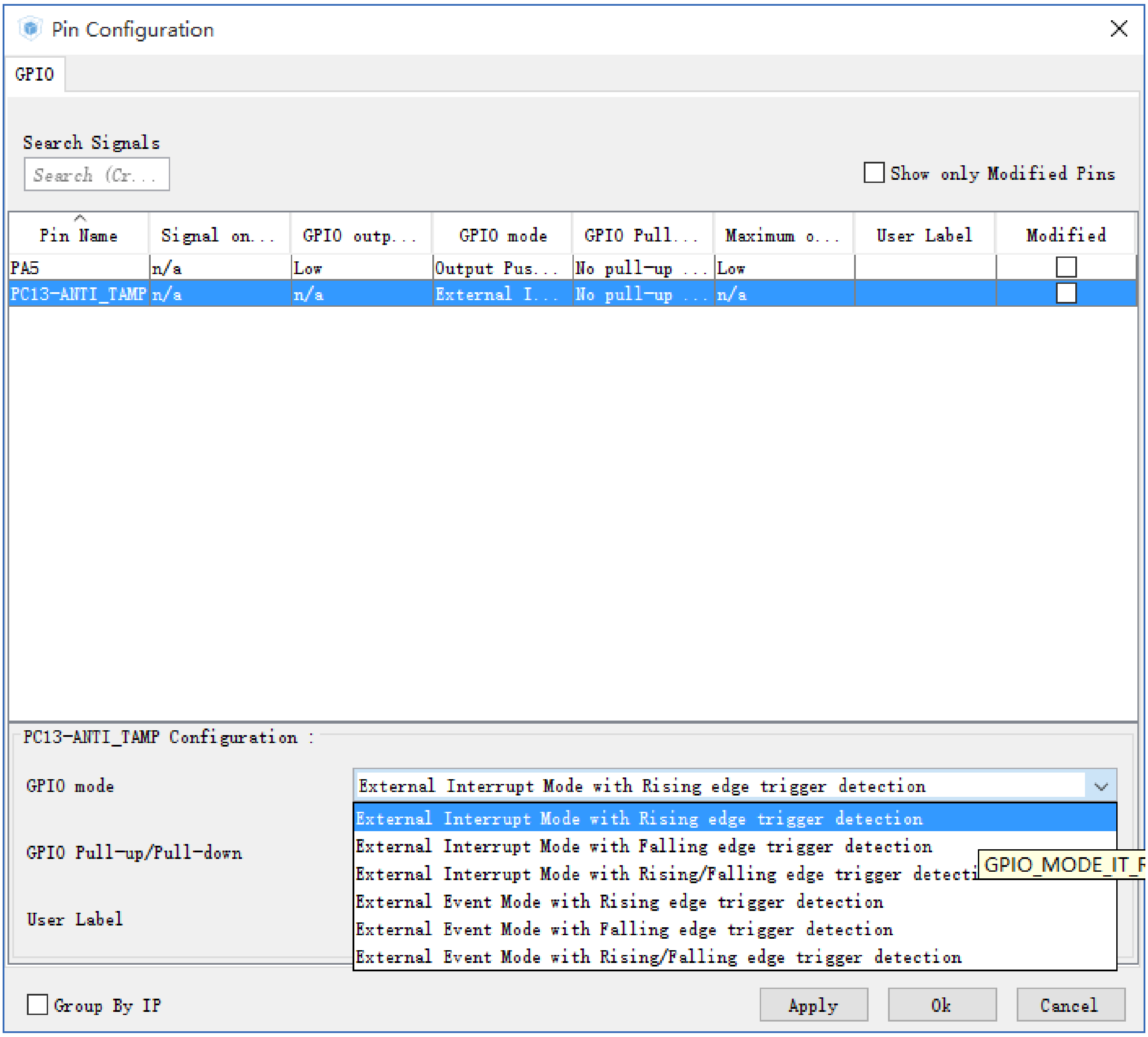


點開 GPIO 設定按鈕

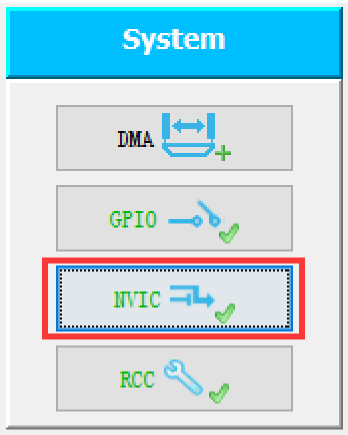


選擇 GPIO 模式為上升緣觸發的外部中斷模式

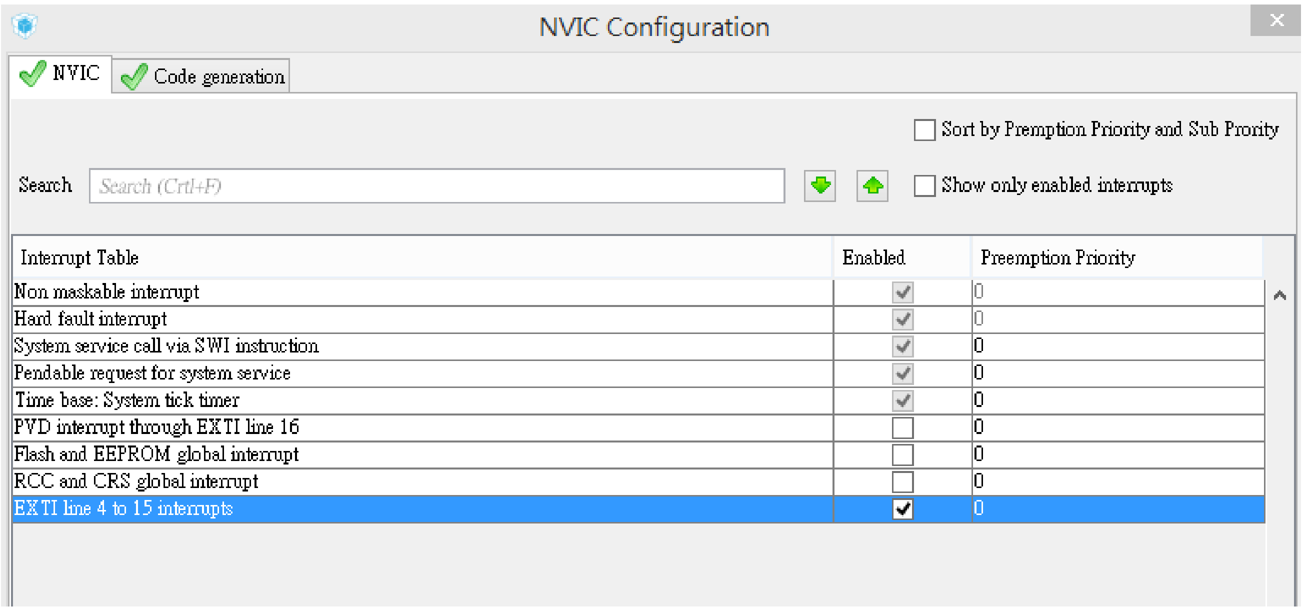
（何謂上升緣觸發、何謂外部中斷，以及其他模式，請自行參考其他有關資料。）

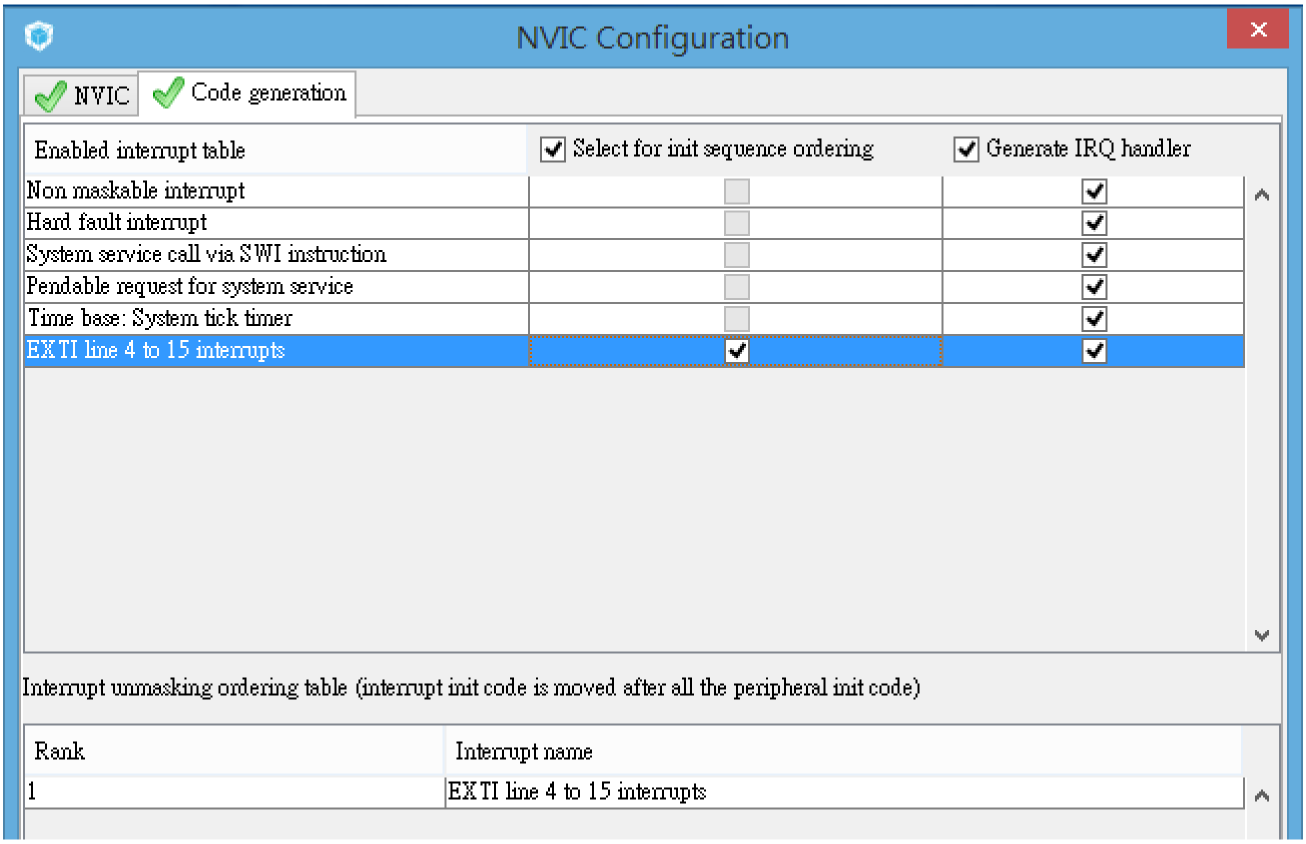


點選巢狀中斷控制器(NVIC)設定按鈕



勾選置能外部中斷 4~15，並且生成相應初始化程式碼





2、在生成的工程中的對應位置定義外部中斷回 Call 函數：

```
/* USER CODE BEGIN 4 */
void HAL_GPIO_EXTI_Callback (uint16_t GPIO_Pin)
{
    if(GPIO_Pin == GPIO_PIN_13)
        HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
}

/* USER CODE END 4 */
```

3、編譯、下載程式至開發板，並且按下 RESET 按鈕。

三、程式碼分析

1、在開發板的開機檔案 startup_stm32l053xx.s 中將 EXTI15_10_IRQHandler 函數註冊為外部中斷 10~15 號線的中斷服務函數，當外部中斷 10~15 號線產生外部中斷時由硬體調用 EXTI15_10_IRQHandler 函數，中斷當前運行的程式，CPU 開始執行中斷服務函數內的程式，執行完之後繼續運行中斷前的程式；

2、因為 STM32L053 的硬體結構決定了外部中斷 10~15 號線共用一個中斷向量，因此只能註冊一個中斷服務函數，而 HAL 框架使用 HAL_GPIO_EXTI_IRQHandler()函數和 HAL_GPIO_EXTI_Callback()函數使使用者看來每個外部中斷線都擁有自己的中斷服務函數（後面會講解這兩個函數）；

3、在 stm32l0xx_it.c 檔中定義了 EXTI15_10_IRQHandler 函數，該函式呼叫了 HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_13);語句；

4、HAL_GPIO_EXTI_IRQHandler()在 stm32l0xx_hal_gpio.c 檔中定義了，該函數判斷外部中斷是由哪一號外部中斷線產生的，並且清除中斷掛起寄存器中的對應位，然後調用外部中斷回呼函數 HAL_GPIO_EXTI_Callback()，並將產生外部中斷的中斷線作為參數傳遞給外部中斷回呼函數 HAL_GPIO_EXTI_Callback()；

5、而上面的所有工作都由 STM32CubeMX 幫我們做好了，我們只需要在 main.c 檔中重定義 HAL_GPIO_EXTI_Callback()函數就行了；

6、因為所有外部中斷都會調用 HAL_GPIO_EXTI_Callback()函數，所以我們需要在 HAL_GPIO_EXTI_Callback()函數內部根據輸入的 GPIO_Pin 參數判斷是哪一號外部中斷線的產生了外部中斷，然後根據不同的外部中斷執行不同的程式碼；

7、因此我們使用語句 `if(GPIO_Pin == GPIO_PIN_13)`判斷該外部中斷是否是由外部中斷 13 號線產生的，然後執行 `HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);`語句翻轉 LED 燈的電位狀態；

提示：我們不應該在中斷服務函數或者中斷回 Call 函數內執行過多的程式碼，這樣做是為了最大程度地減少中斷嵌套。有關中斷嵌套和中斷優先順序的內容請自行參考相關資料，中斷優先順序（不僅限於外部中斷）可以在 STM32CubeMX 中的 NVIC Configuration 中設置。

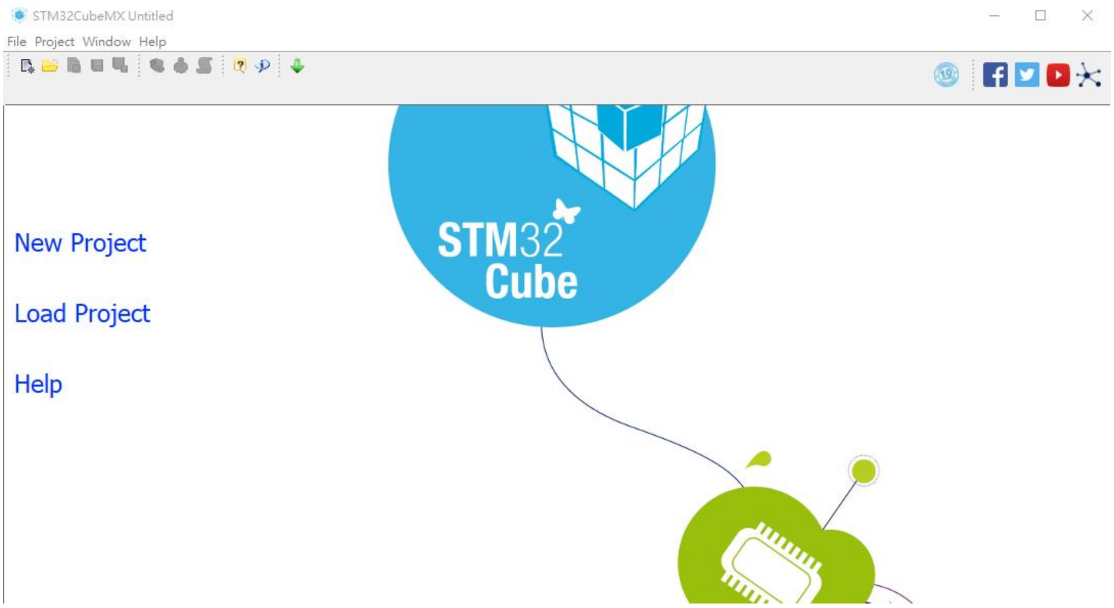
利用 Buzzer 發出聲音

檔案名稱: 04-Buzzer.doc

本例將利用 PC13 的 GPIO PIN 接一個按鈕(板子上)，使得 Buzzer 發聲。
PC13 的接腳為輸入，因此須利用內部的 Pull up 電阻將電位提升，以便使得此按鈕為 Low Active(Low 動作)。

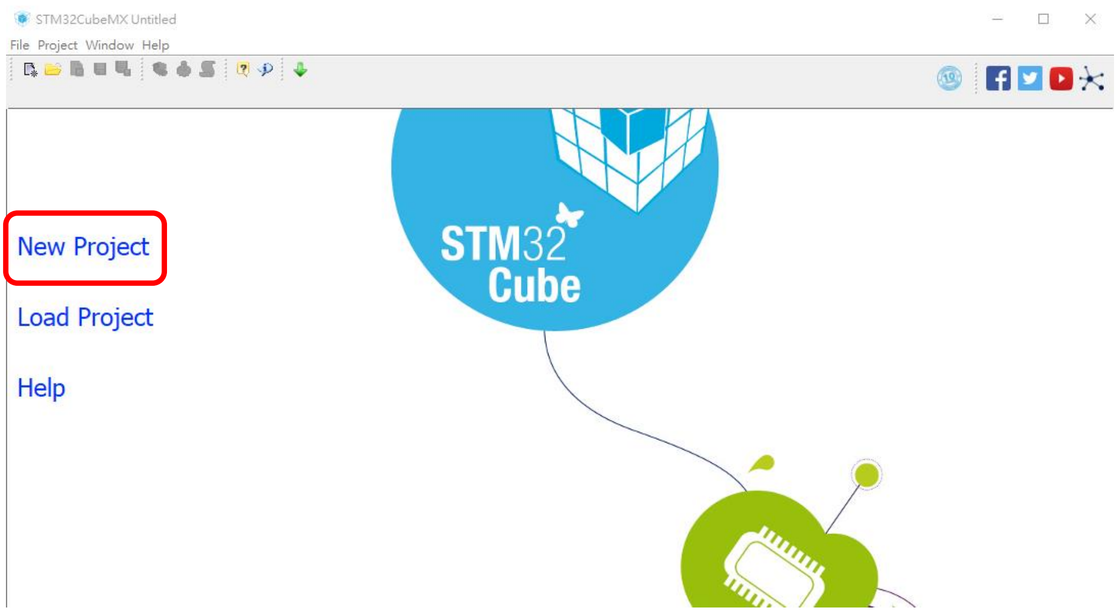
首先建立一個目錄作為專案(Project)的目錄。本例設為「3_Buzzer1」。

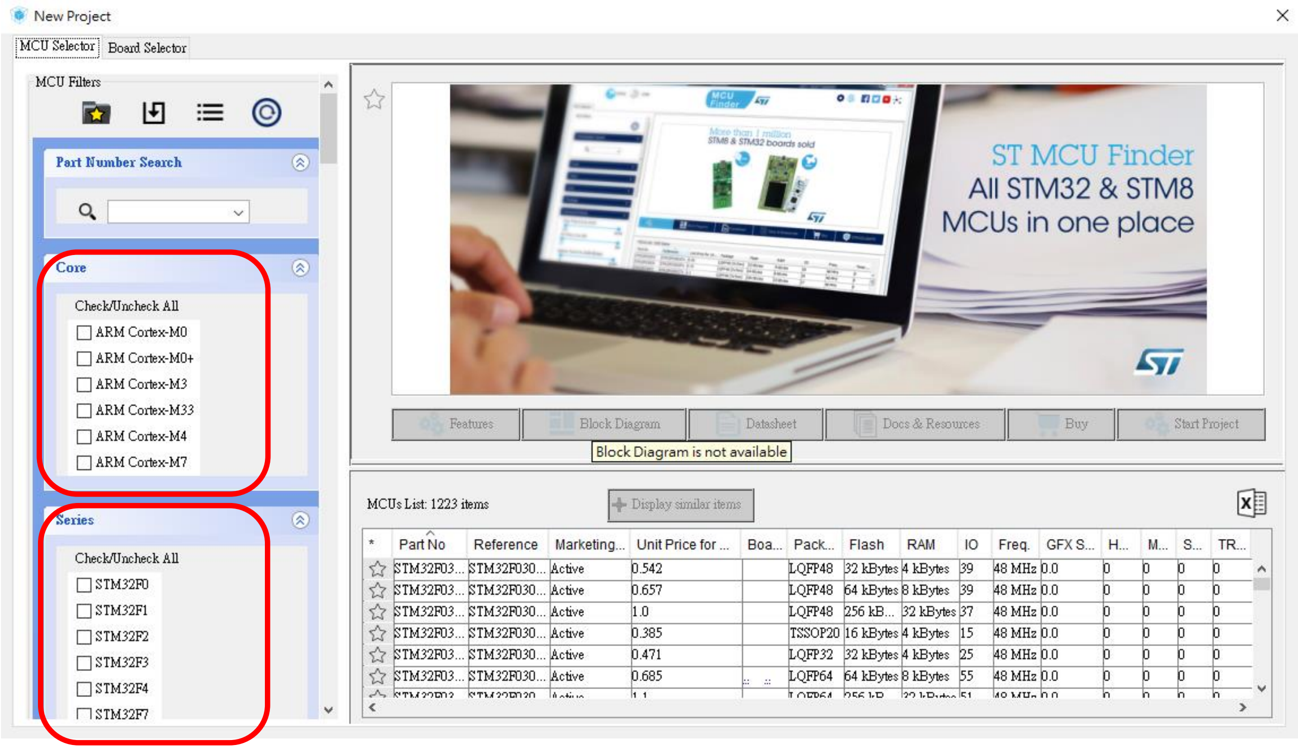
開啟 STM32 的 CUBE 程式如下圖一:



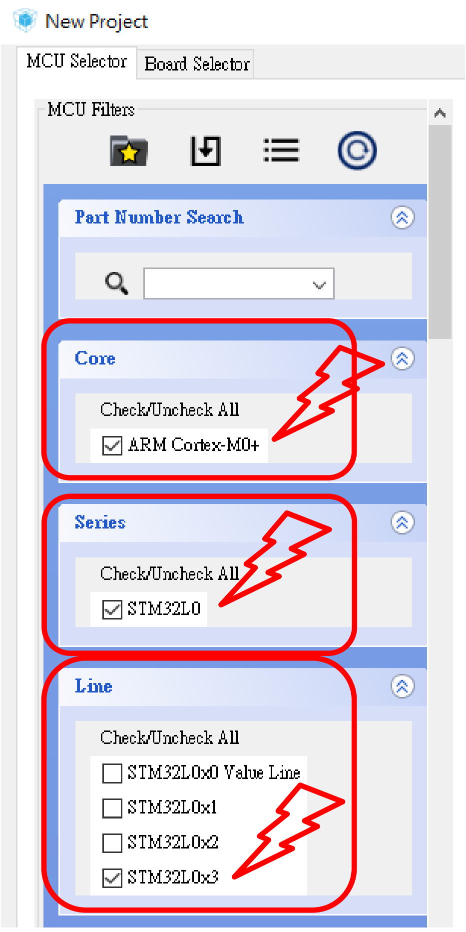
本次使用的版本為 4.27.0，如下圖二:

點選 New Project 選項如下圖

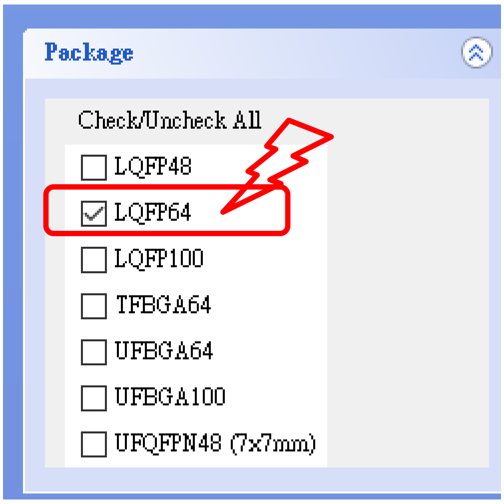




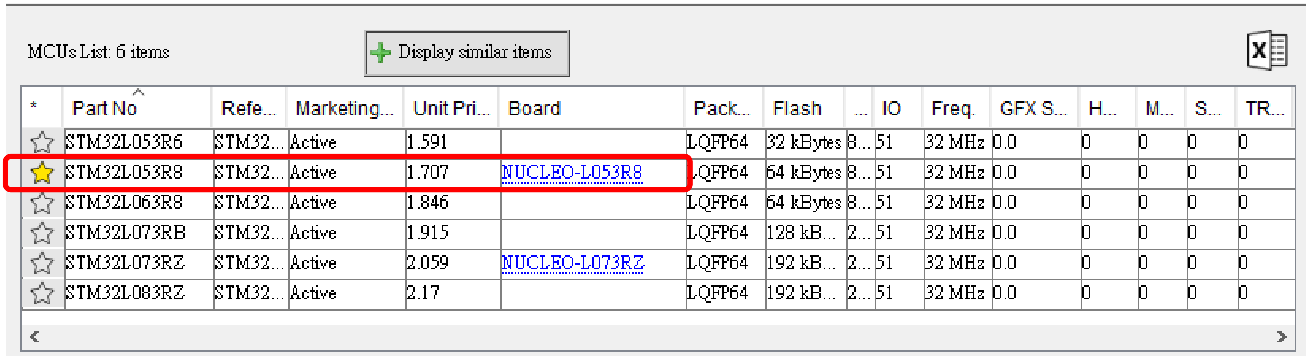
如下圖點選 MCU 的 Core，Serial 及 Line:本範例使用 STM32L053R8T6



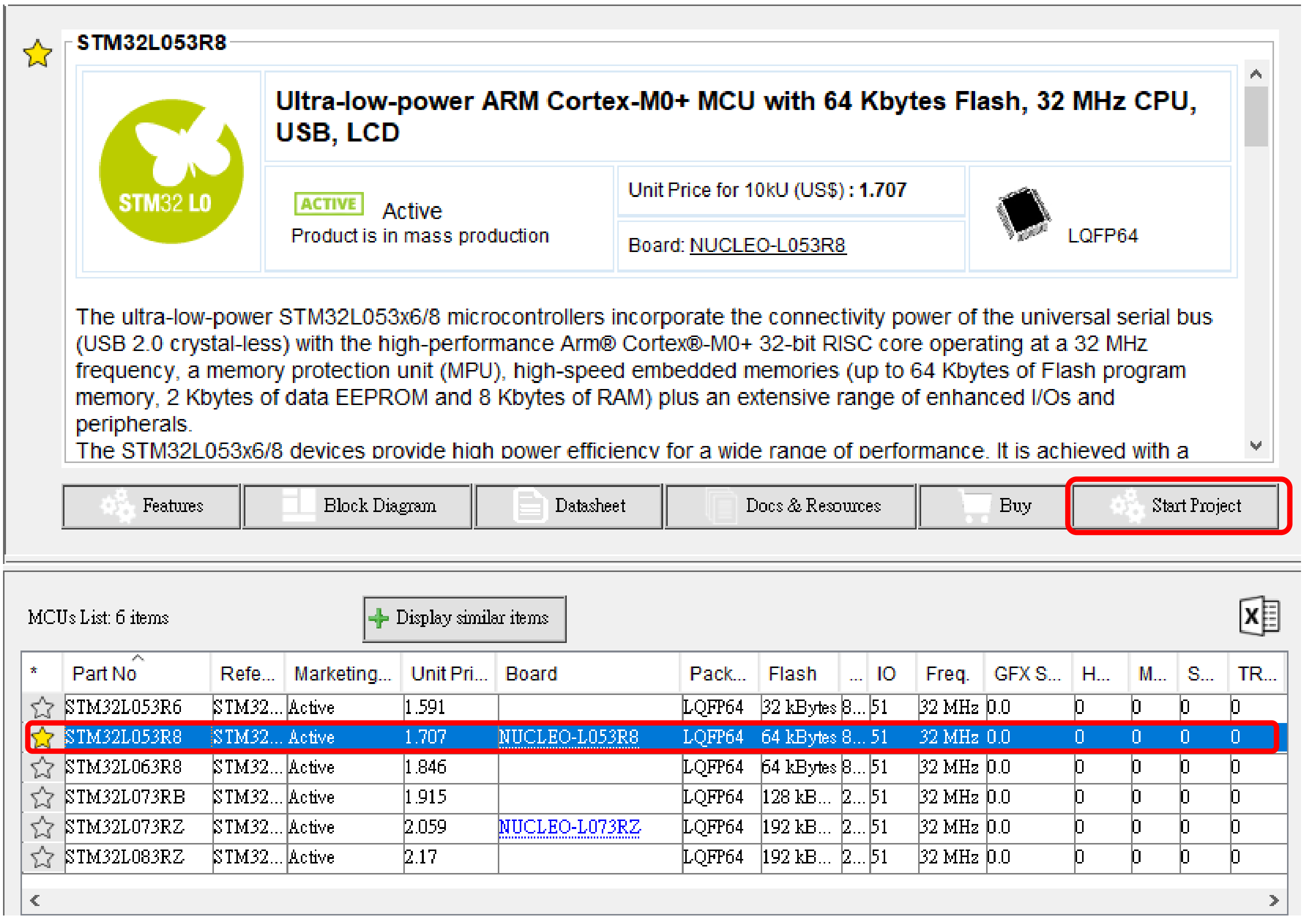
接著再點選 Package(LQFP64)如下圖



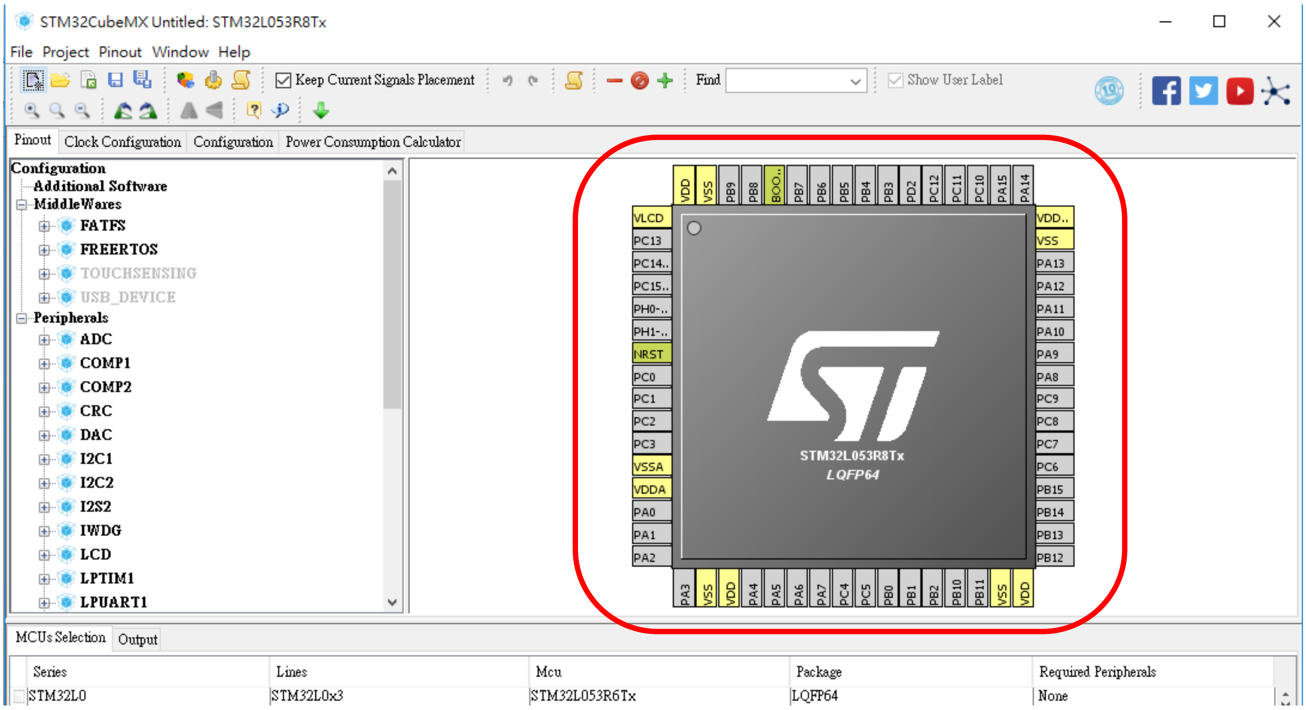
再點選 MCU 的 Mode，



點選完後再選右上 **Start Project**


STM32L053R8
Ultra-low-power ARM Cortex-M0+ MCU with 64 Kbytes Flash, 32 MHz CPU, USB, LCD
STM32 L0
ACTIVE
Active
Product is in mass production
Unit Price for 10kU (US$) : 1.707
Board: NUCLEO-L053R8
LQFP64
The ultra-low-power STM32L053x6/8 microcontrollers incorporate the connectivity power of the universal serial bus (USB 2.0 crystal-less) with the high-performance Arm® Cortex®-M0+ 32-bit RISC core operating at a 32 MHz frequency, a memory protection unit (MPU), high-speed embedded memories (up to 64 Kbytes of Flash program memory, 2 Kbytes of data EEPROM and 8 Kbytes of RAM) plus an extensive range of enhanced I/Os and peripherals.
The STM32L053x6/8 devices provide high power efficiency for a wide range of performance. It is achieved with a
Features
Block Diagram
Datasheet
Docs & Resources
Buy
Start Project
MCUs List: 6 items
Display similar items
* Part No Refe... Marketing... Unit Pri... Board Pack... Flash ... IO Freq. GFX S... H... M... S... TR...
STM32L053R6 STM32... Active 1.591 LQFP64 32 kBytes 8... 51 32 MHz 0.0 0 0 0 0
STM32L053R8 STM32... Active 1.707 NUCLEO-L053R8 LQFP64 64 kBytes 8... 51 32 MHz 0.0 0 0 0 0
STM32L063R8 STM32... Active 1.846 LQFP64 64 kBytes 8... 51 32 MHz 0.0 0 0 0 0
STM32L073RB STM32... Active 1.915 LQFP64 128 kB... 2... 51 32 MHz 0.0 0 0 0 0
STM32L073RZ STM32... Active 2.059 NUCLEO-L073RZ LQFP64 192 kB... 2... 51 32 MHz 0.0 0 0 0 0
STM32L083RZ STM32... Active 2.17 LQFP64 192 kB... 2... 51 32 MHz 0.0 0 0 0 0

當點選後顯示如下圖，單須確認是否選到正確的晶片 MCU

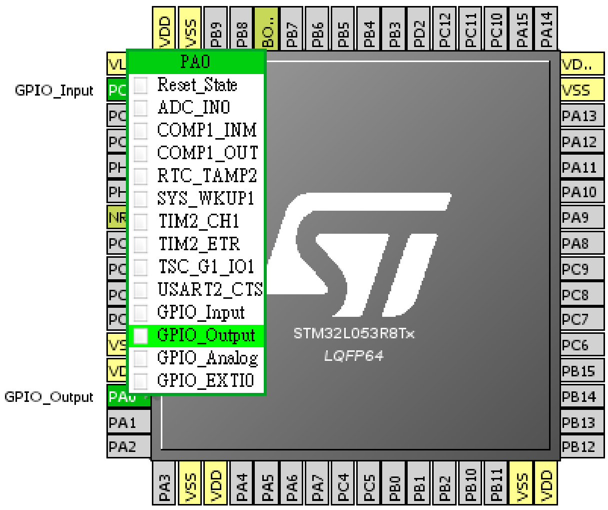


設定 PA0，說明如下:

3. 將滑鼠的指標移至晶片的上方或視窗區域，轉動滾輪放大或所小圖示。
4. 按住滑鼠右鍵後，移動滑鼠可以移動 MCU 的位置。

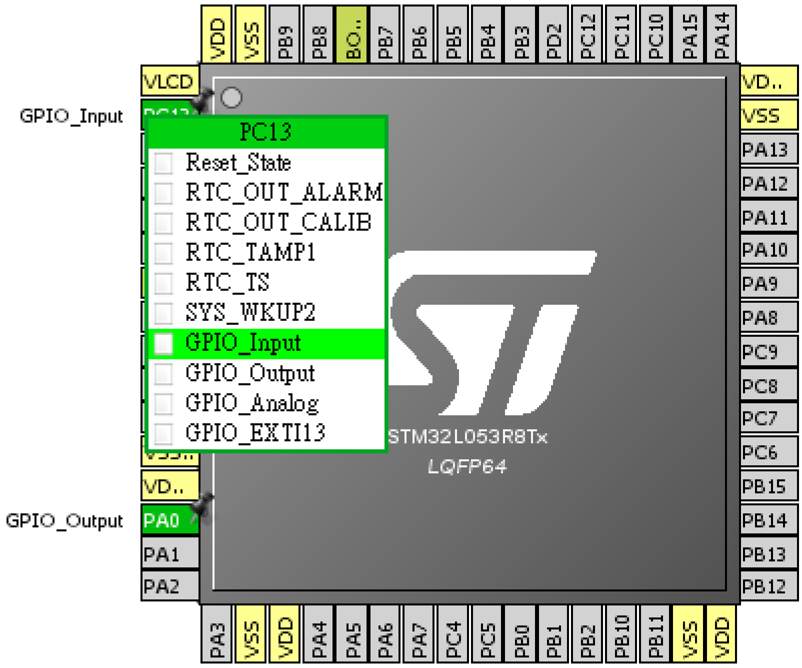
再本範例中欲使用 PA0 的接腳(Pin)作為輸出到 LED 方式:

3. 將滑鼠移置 PA0 後按下左鍵。
4. 點選倒數第三項左邊的空格「GPIO Output」

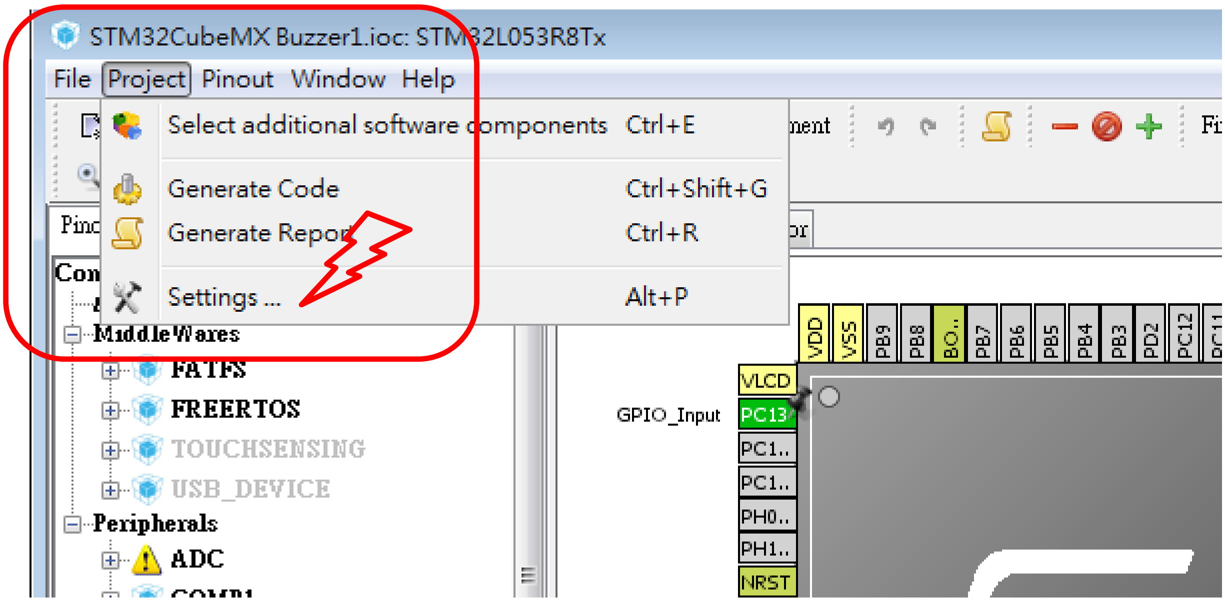


設定 PC13，說明如下:
將滑鼠的指標移至晶片的上方或視窗區域，轉動滾輪放大或所小圖示。
按住滑鼠右鍵後，移動滑鼠可以移動 MCU 的位置。

再本範例中欲使用 PC13 的接腳(Pin)作為輸入方式:
將滑鼠移置 PC13 後按下左鍵。
點選倒數第三項左邊的空格「GPIO Input」

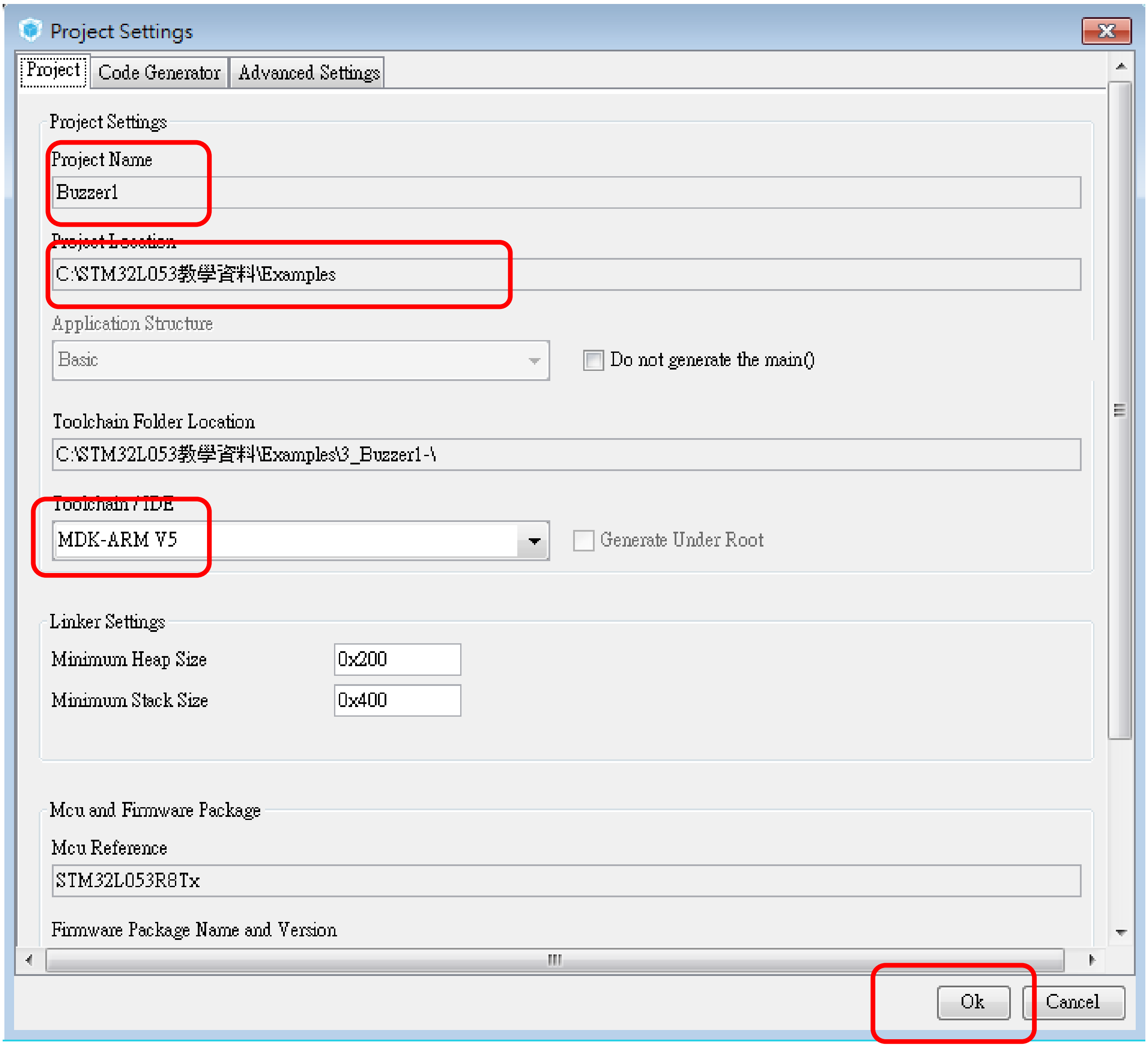


點選左上方的 Project →Setting:以便專案的設定

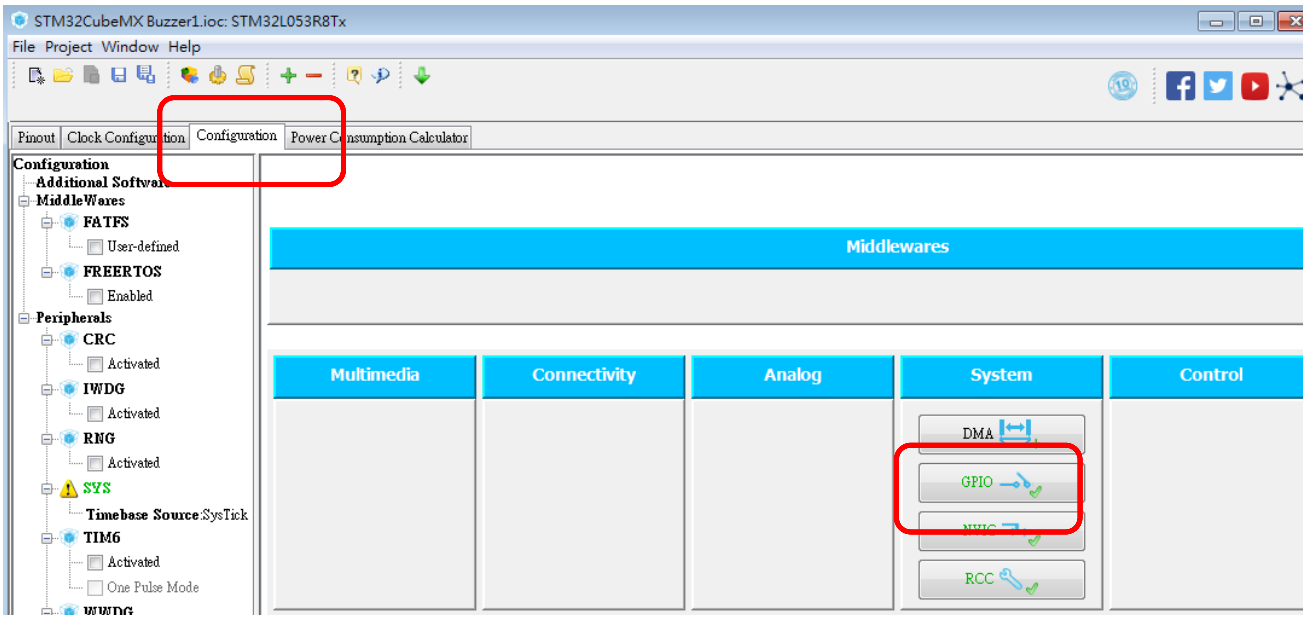


顯示如下圖，並參考如下填入:

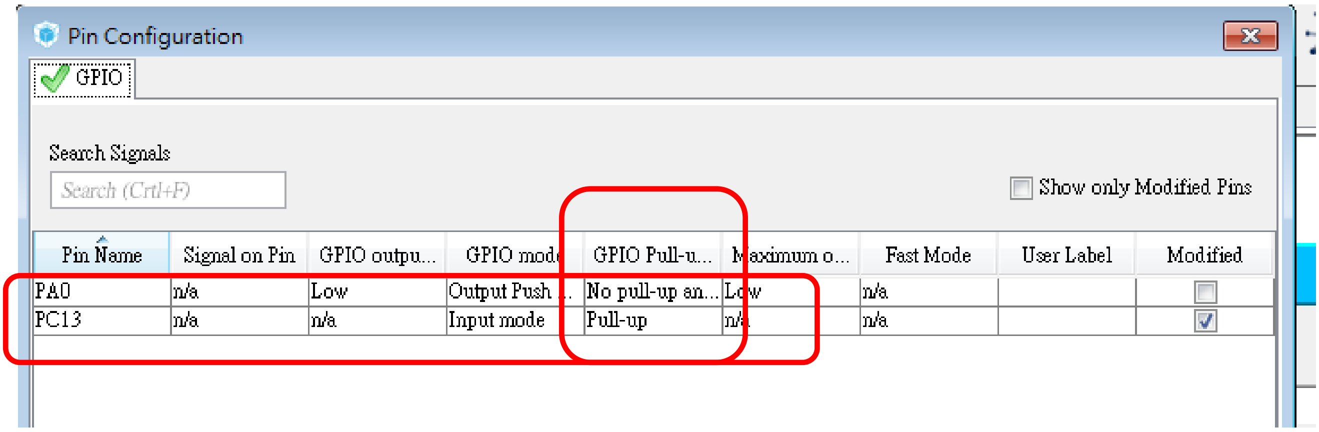
5. Project Name
6. Project Location
7. Tool Chain → MDK-ARM V5
8. OK(右下角)



接下來要設定 PC13 為 PULL High 輸入，如下圖選擇「Configuration」,「GPIO」,

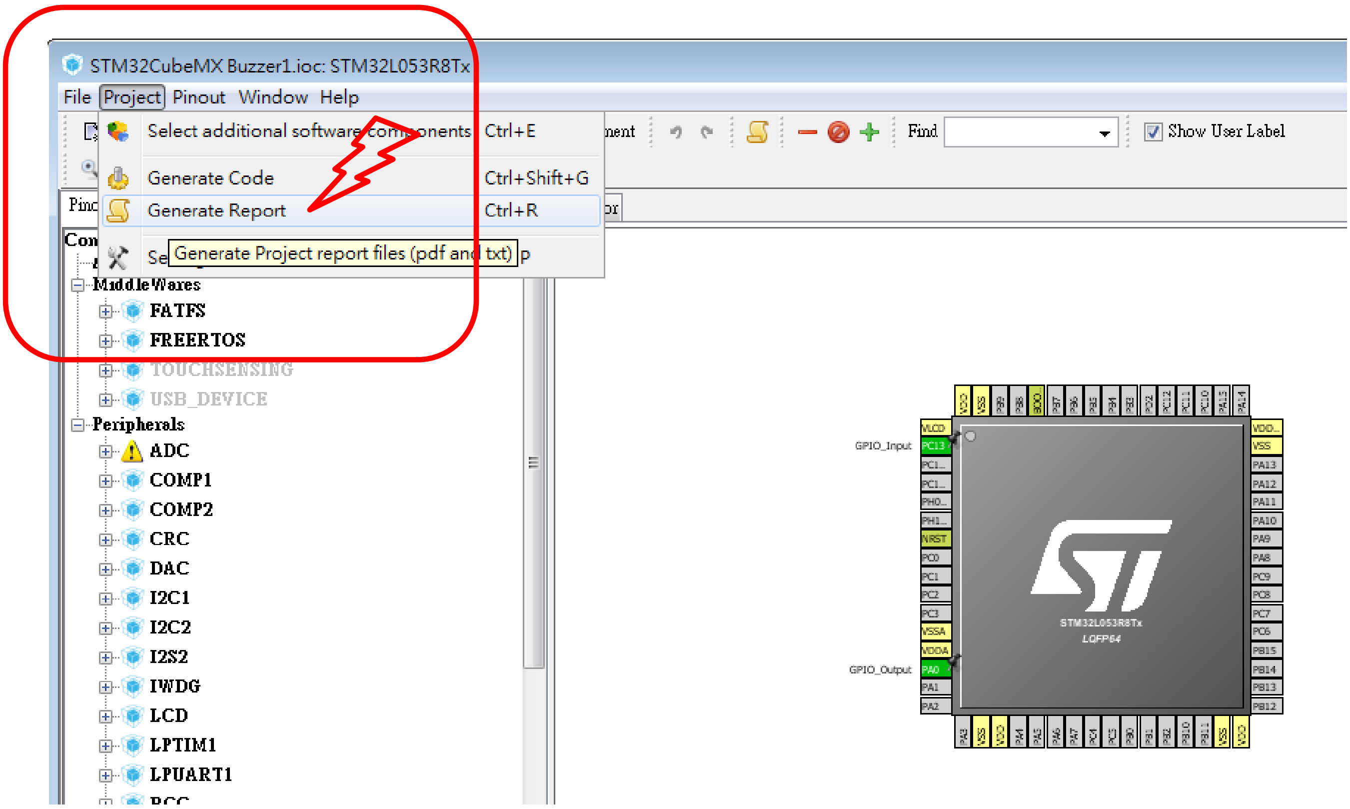


在 PC13 的位置，並點選 GPIO 「Pull-up」。

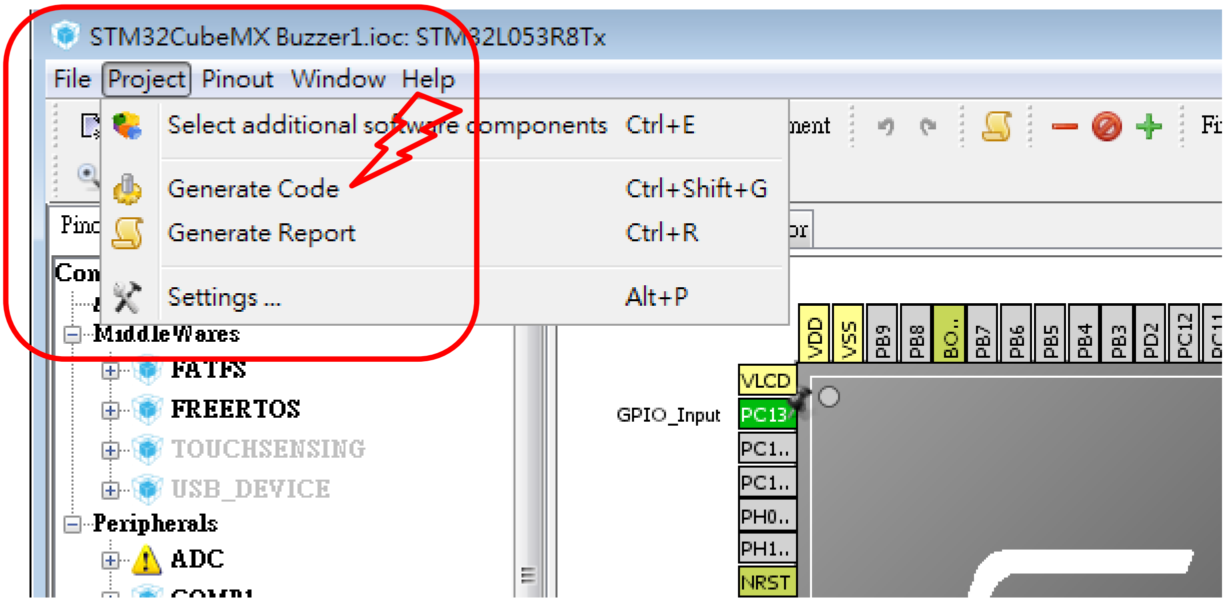


設定完成後，點選「OK」即可。

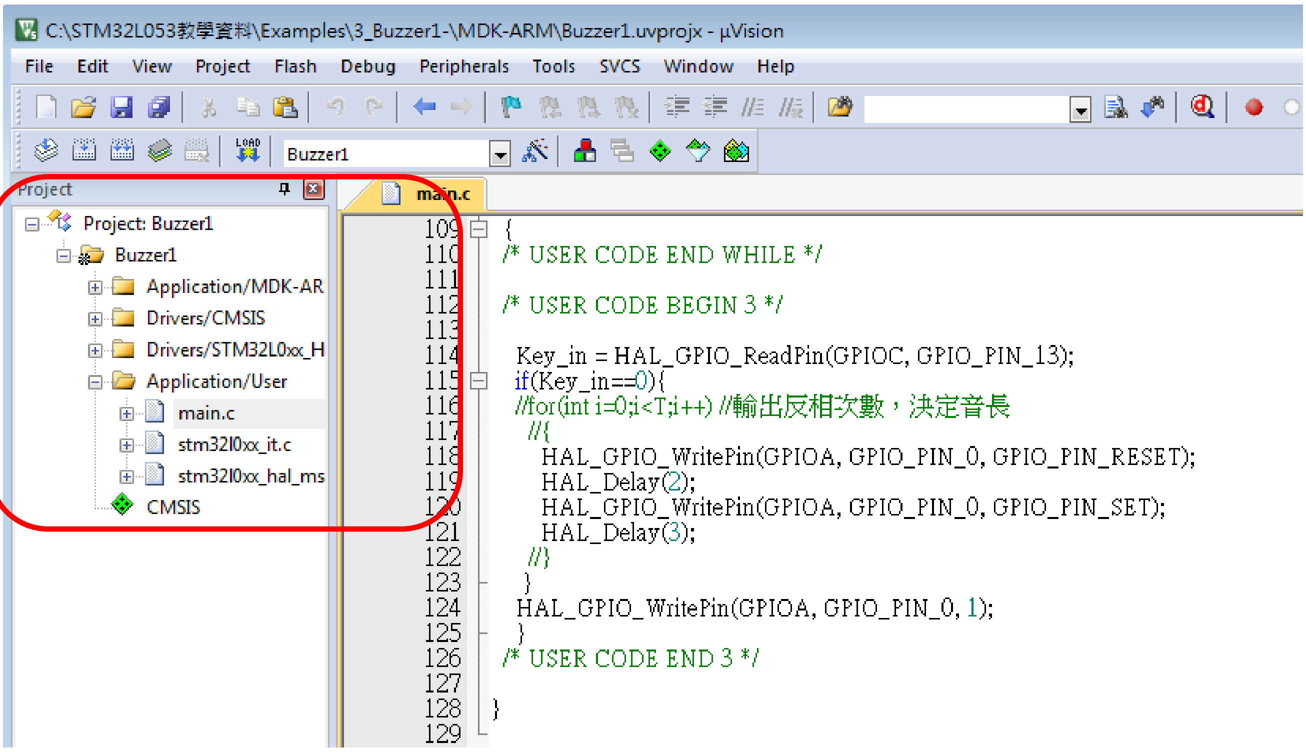
接下來可以利用 Cube 來產生報告。
點選 Project→Generate Report 如下圖:
這個 Report 是 PDF 檔，可以檢視我們設定是否正確，最好在最後設定完成後檢視。



本範例為初次使用自動產生程式碼方式，因此使用其預設值即可。
點選左上方的 Project →Generate Code:以產生程式碼。



在產生程式碼後開啟後並開啟專案檔後如下圖:



根據 HAL 驅動說明書找到我們需要的 API，然後根據說明添加代碼：（根據說明得到使 PA5 輸出高電位的代碼為 *HAL_GPIO_WritePin (GPIOA, GPIO_PIN_5, GPIO_PIN_SET);*）

25.2.8 HAL_GPIO_WritePin

Function Name	**void HAL_GPIO_WritePin (GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin, GPIO_PinState PinState)**
Function Description	Sets or clears the selected data port bit.
Parameters	• **GPIOx:** where x can be (A..K) to select the GPIO peripheral for STM32F429X device or x can be (A..I) to select the GPIO peripheral for STM32F40XX and STM32F427X devices. • **GPIO_Pin:** specifies the port bit to be written. This parameter can be one of GPIO_PIN_x where x can be (0..15). • **PinState:** specifies the value to be written to the selected bit. This parameter can be one of the GPIO_PinState enum values: GPIO_PIN_RESET: to clear the port pinGPIO_PIN_SET: to set the port pin
Return values	• None
Notes	• This function uses GPIOx_BSRR register to allow atomic read/modify accesses. In this way, there is no risk of an IRQ occurring between the read and the modify access.

根據 HAL 驅動說明書找到我們需要的 API，然後根據說明添加代碼：（根據說明得到使 PC13 輸出高電位的代碼為 *HAL_GPIO_ReadPin (GPIOC, GPIO_PIN_13);*）

- __PPP_CLK_SLEEP_ENABLE/__PPP_CLK_SLEEP_DISABLE to peripheral clock during low power (Sleep) mode.

2.11.2 GPIOs

GPIO HAL APIs are the following:

- HAL_GPIO_Init() / HAL_GPIO_DeInit()
- HAL_GPIO_ReadPin() / HAL_GPIO_WritePin()
- HAL_GPIO_TogglePin ().

46/1466 DocID026232 Rev 6

輸入程式碼如下:

3. 在 while(1)下方:
   /*USER CODE BEGIN 3*/及/* USER CODE END 3 */中間開始輸入，以避免程式重新產生時被覆蓋。
4. 在程式輸入時會自動出現關鍵字以便選擇。(MDK 5.xx 版後)

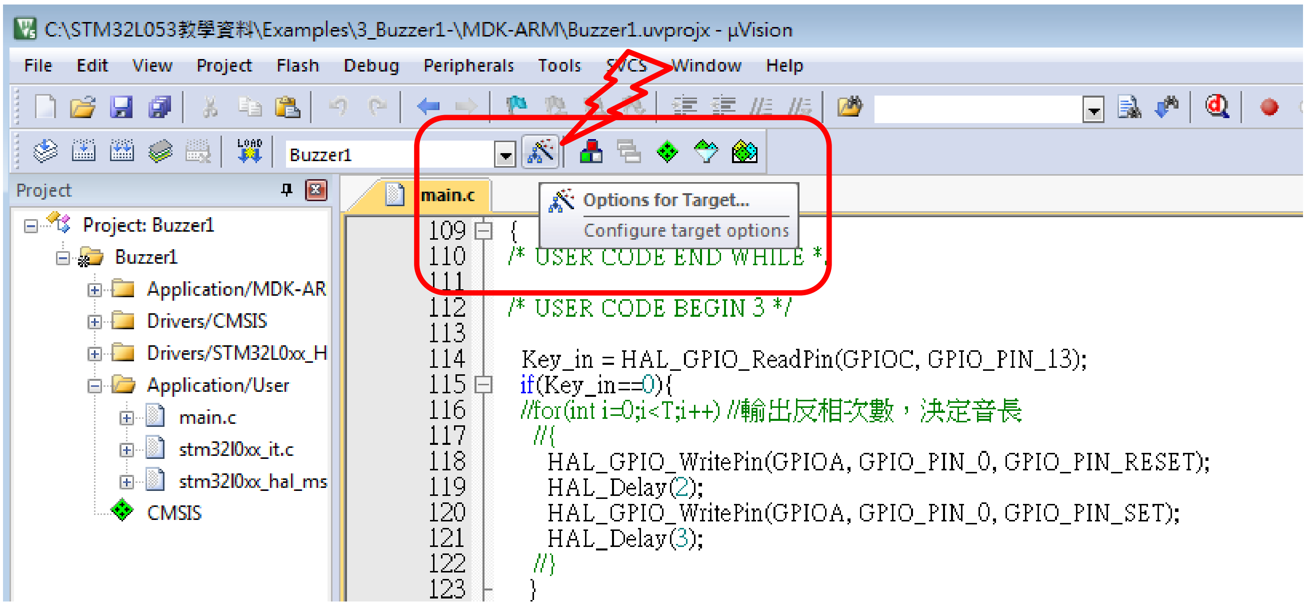
程式碼如下：

```
while (1)
  {
  /* USER CODE END WHILE */
  /* USER CODE BEGIN 3 */

        Key_in = HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13);
        if(Key_in==0){
        //for(int i=0;i<T;i++) //輸出反相次數，決定音長
    //{
                HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_RESET);
                HAL_Delay(2);
                HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_SET);
                HAL_Delay(3);
          //}
     }
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_SET);
        }
```

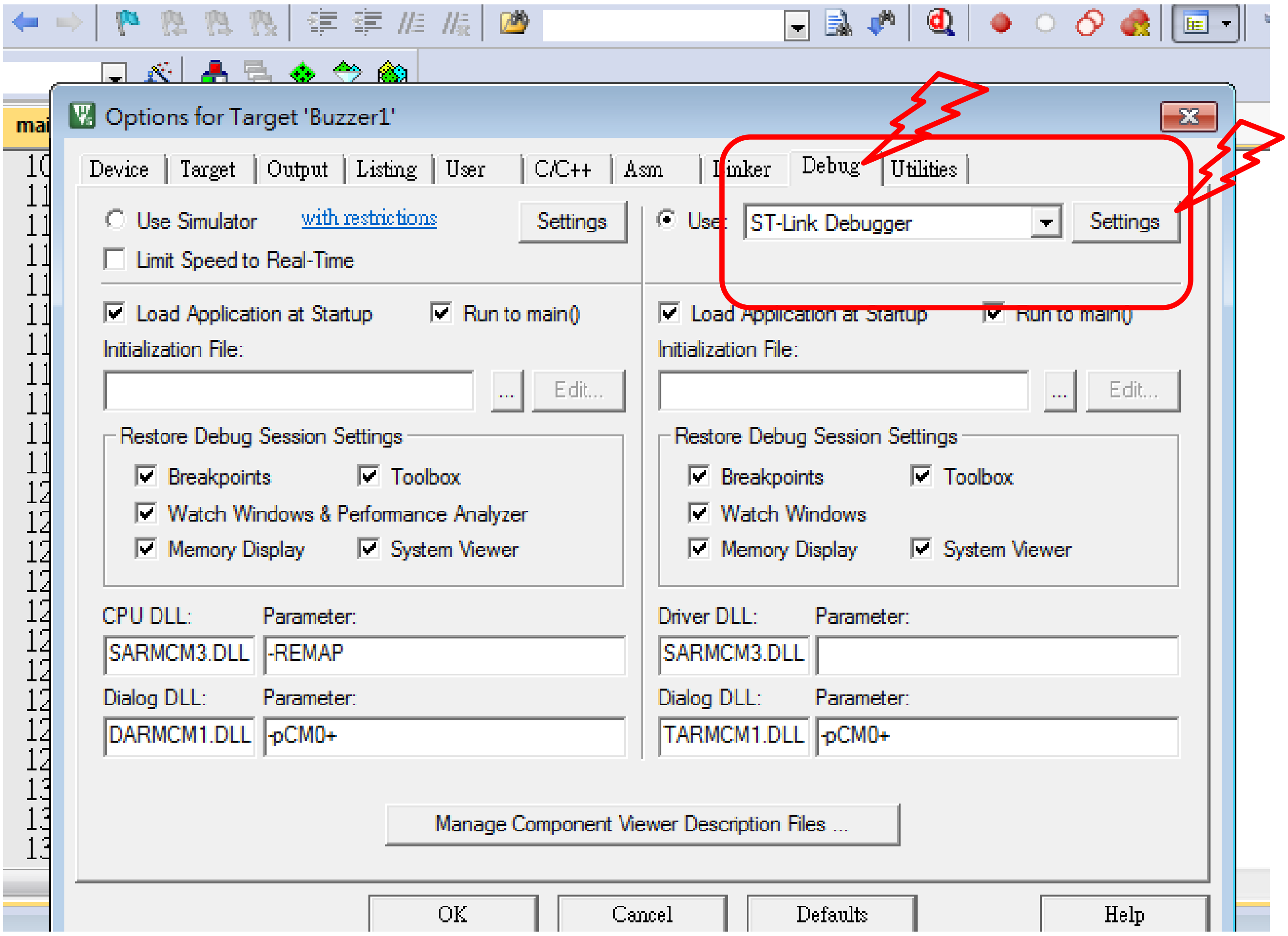
注意:

4. 我們編寫程式時，需將程式碼放在/*USER CODE BEGIN N */ 與 /*USER CODE END N */內，以免在 CUBE 產生程式碼時被覆蓋。
5. 按下 Reset 按鈕開始執行程式，可以聽到 PA0 的 Buzzer 聲音。
6. 亦可事先在 Keil 中設定燒錄完成後自動執行程式。

方式:在 MDK 的 Option for Target...(仙女棒) 按鈕點選。

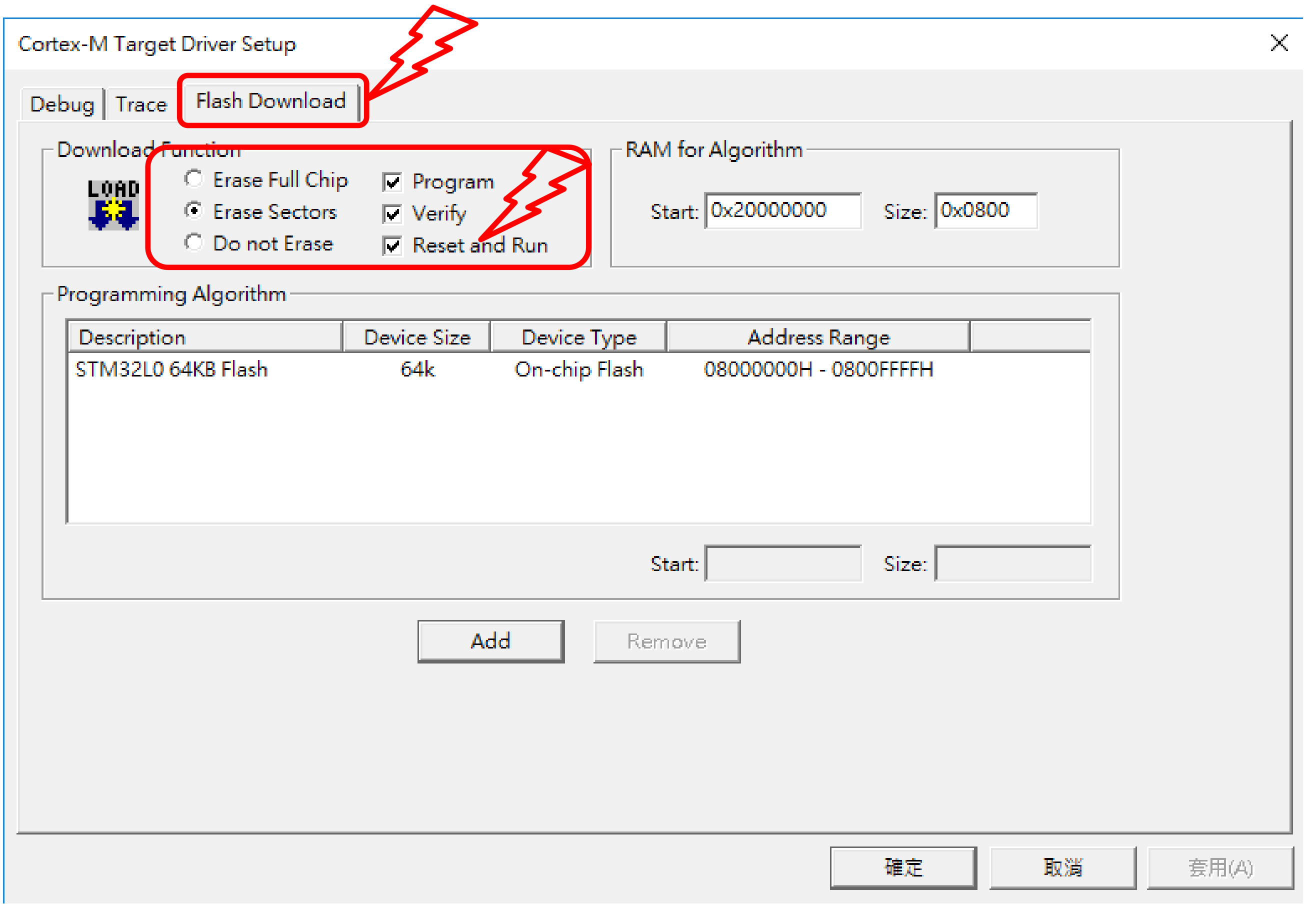


點選 Option for Target...(仙女棒)後即可開啟另一視窗，如下:
接者點選 Debug 後，注意在紅框中需選設在 ST-Link Debugger 或相容的 Tool。
確認後再點選右邊的 settings 選項。



點選上示的選項後，會出現如下示的視窗。

接著點選紅框中的 Flash Download 選項，接著點選下面紅框中的 Reset and Run 選項，這樣會在程式下載完後自動執行，不用再按 Reset 鍵。


Cortex-M Target Driver Setup
Debug
Trace
Flash Download
Download Function
LOAD
Erase Full Chip
Erase Sectors
Do not Erase
Program
Verify
Reset and Run
RAM for Algorithm
Start: 0x20000000
Size: 0x0800
Programming Algorithm
Description
Device Size
Device Type
Address Range
STM32L0 64KB Flash
64k
On-chip Flash
08000000H - 0800FFFFH
Start:
Size:
Add
Remove
確定
取消
套用(A)

利用 PWM 驅動 LED

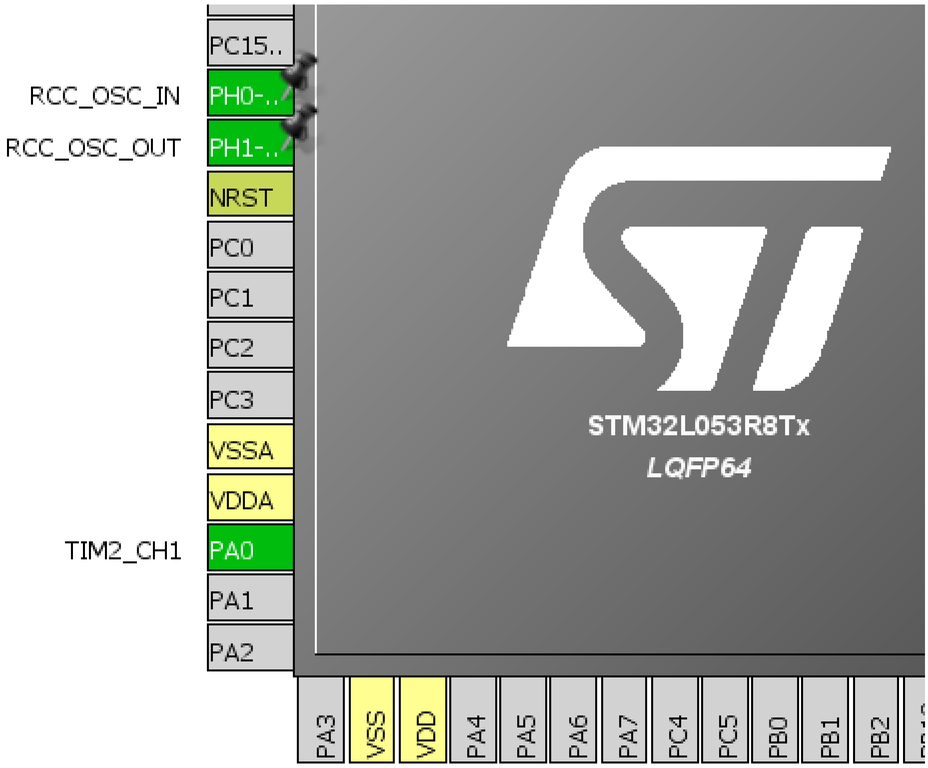
檔案名稱: 05-pwm_led.doc

本例練習利用 PWM 的方式，設定 LED 產生漸明漸滅的方式，本例會用到 HAL 庫及 Timer 計時。由於 STM 對於 RCC 有其相對的 GPIO，因此，PH0，PH1 綁訂在 PA0 ~PA3。本例僅用一顆 LED，所以設定在 PA0 作為輸出。

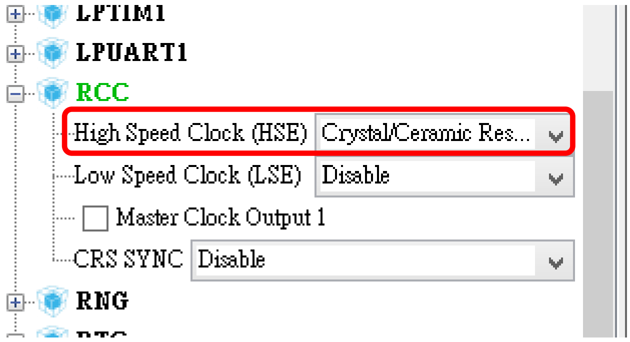
參考下圖

先利用 STM 公司提供的 CUBE，本例使用的版本為 4.27.0(如圖)， 如果開啟舊檔時可能會有版本差異問題時，先選擇移植的選項，有部分較新的專案需升級到版本 5 以上。

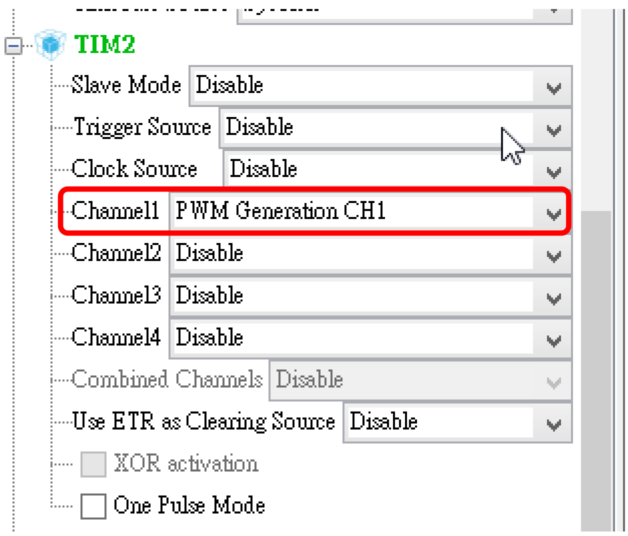
接下來將 PH0 點選為 RCC_OSC_IN，將 PH1 點選為 RCC_OSC_OUT，注意不用點選 PA0。



點開在 CUBE 左邊的選單 RCC，檢查 High Speed Clock(HSE)

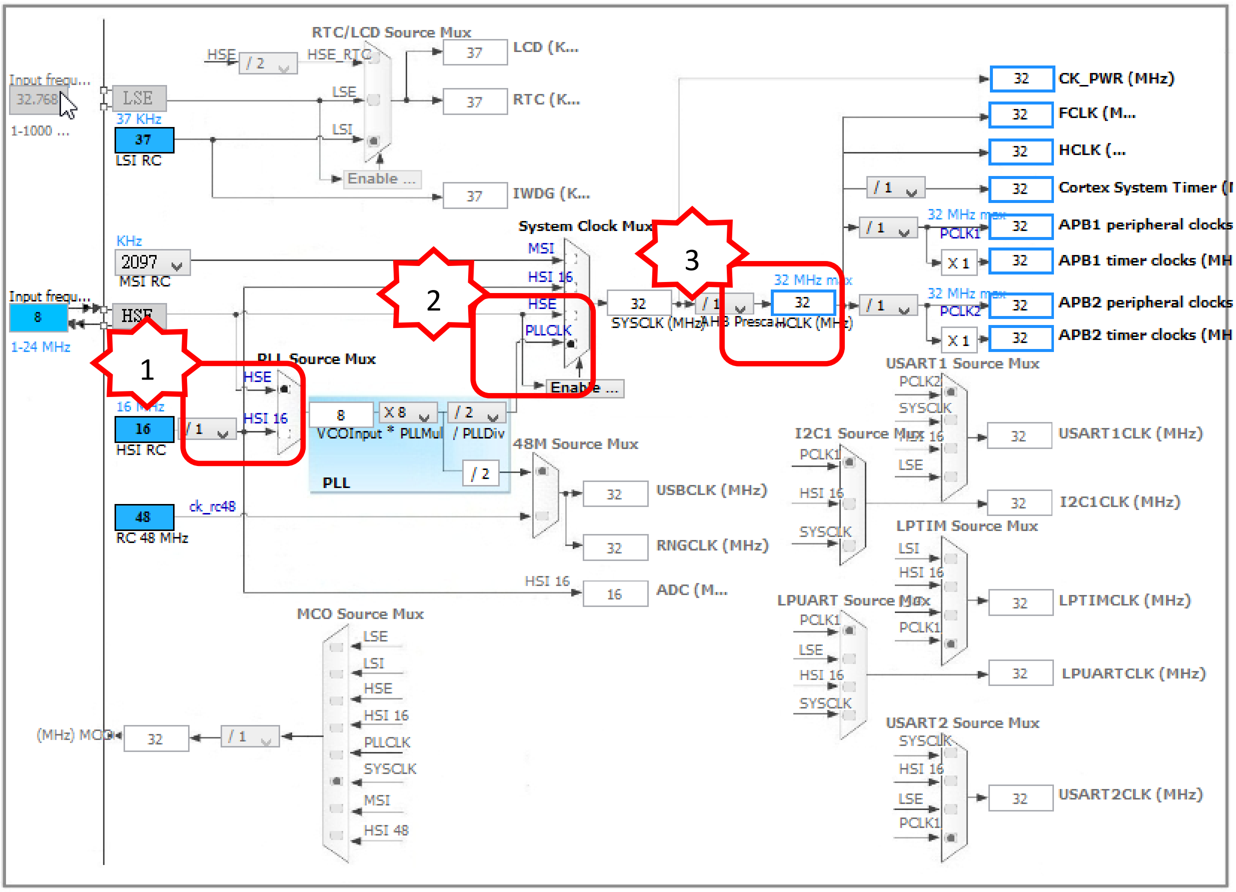


接著要選輸出的 PIN，因此先點開 Channel1 並下拉到 PWM Generation CH1 即可。



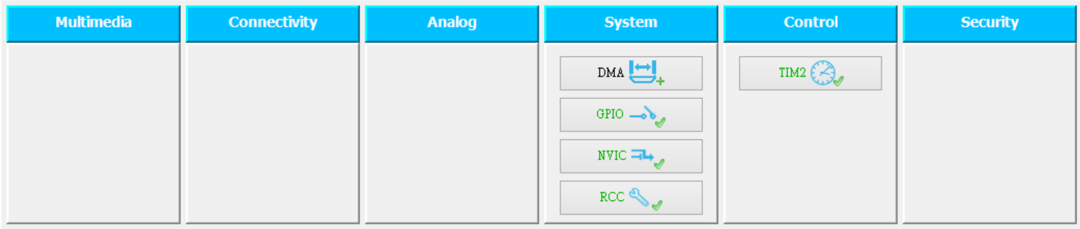
點選 Clock Configuration:

參考下圖，將 1 的 MUX 點選 HSE，將 2 的 MUX 點選 PLLCLK，將 2 的頻率設為最大 32。

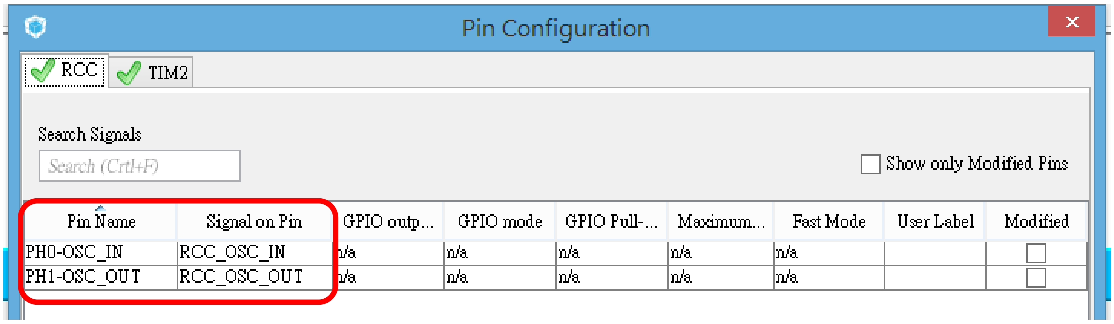


接下來的設定不一定要照順序，但為了講解方便起見，會依序由上而下，由左而右。

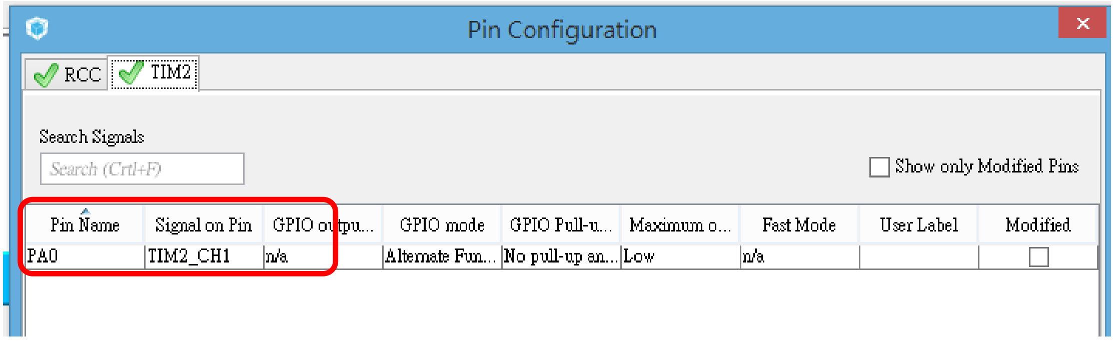
現進入設定的選項模式，由於 ARM 的站存器眾多，且不易了解，應用時容易出錯，因此 STM 公司將其繁雜的設定工作交由第三方開發較為簡便設定，雖然簡便，仍有些設定的關係稍微了解即可，且日後 STM 公司也不再出 CMSIS 庫及標準暫存器庫了，但過去的仍然可相容使用。以下，參考下圖:



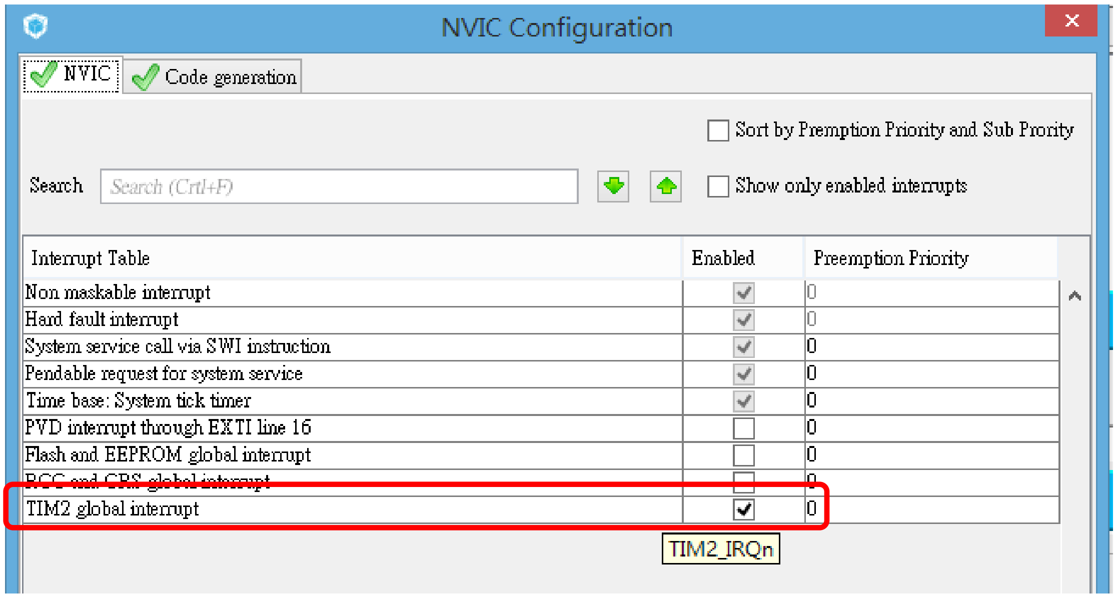
點選 GPIO 項目進入設定，再選擇 RCC 確認一下即可，不做點選。



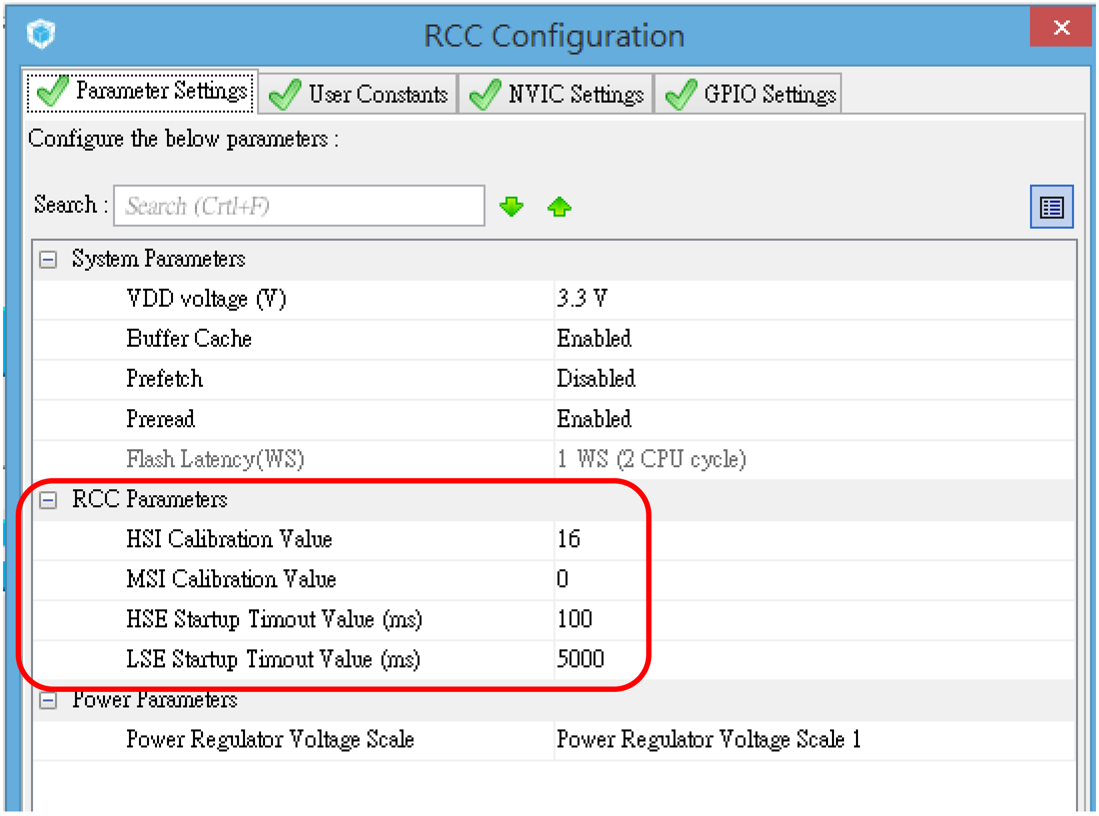
再選擇 TIM2 確認一下為 PA0 即可，不做點選項。



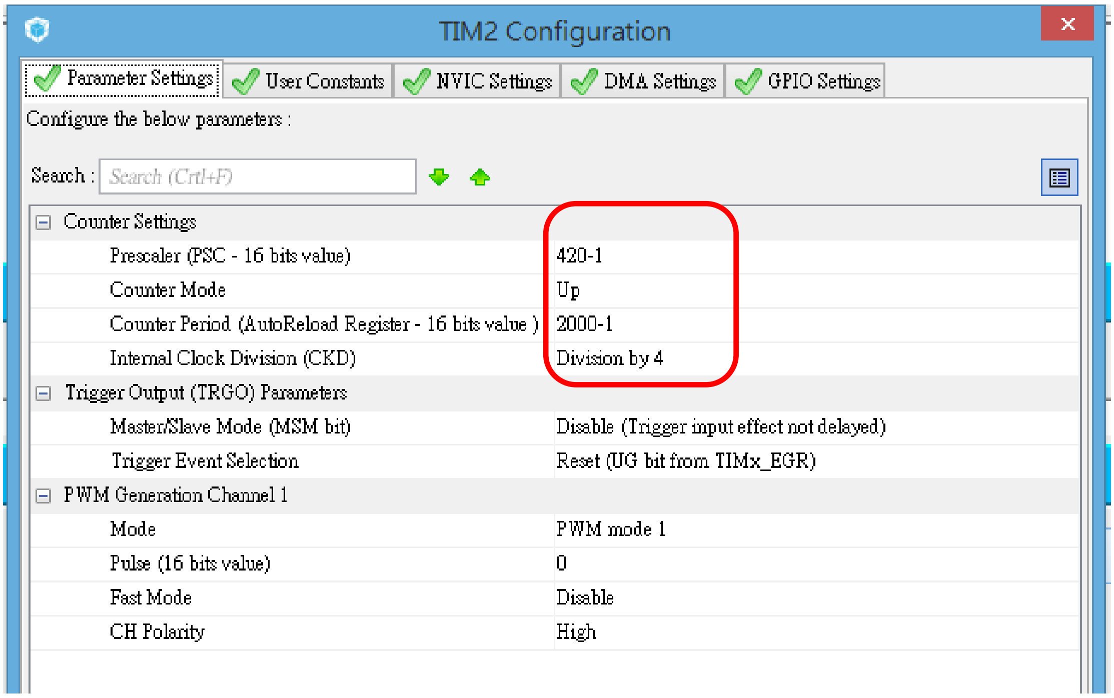
參考下圖
點選 Enable 項



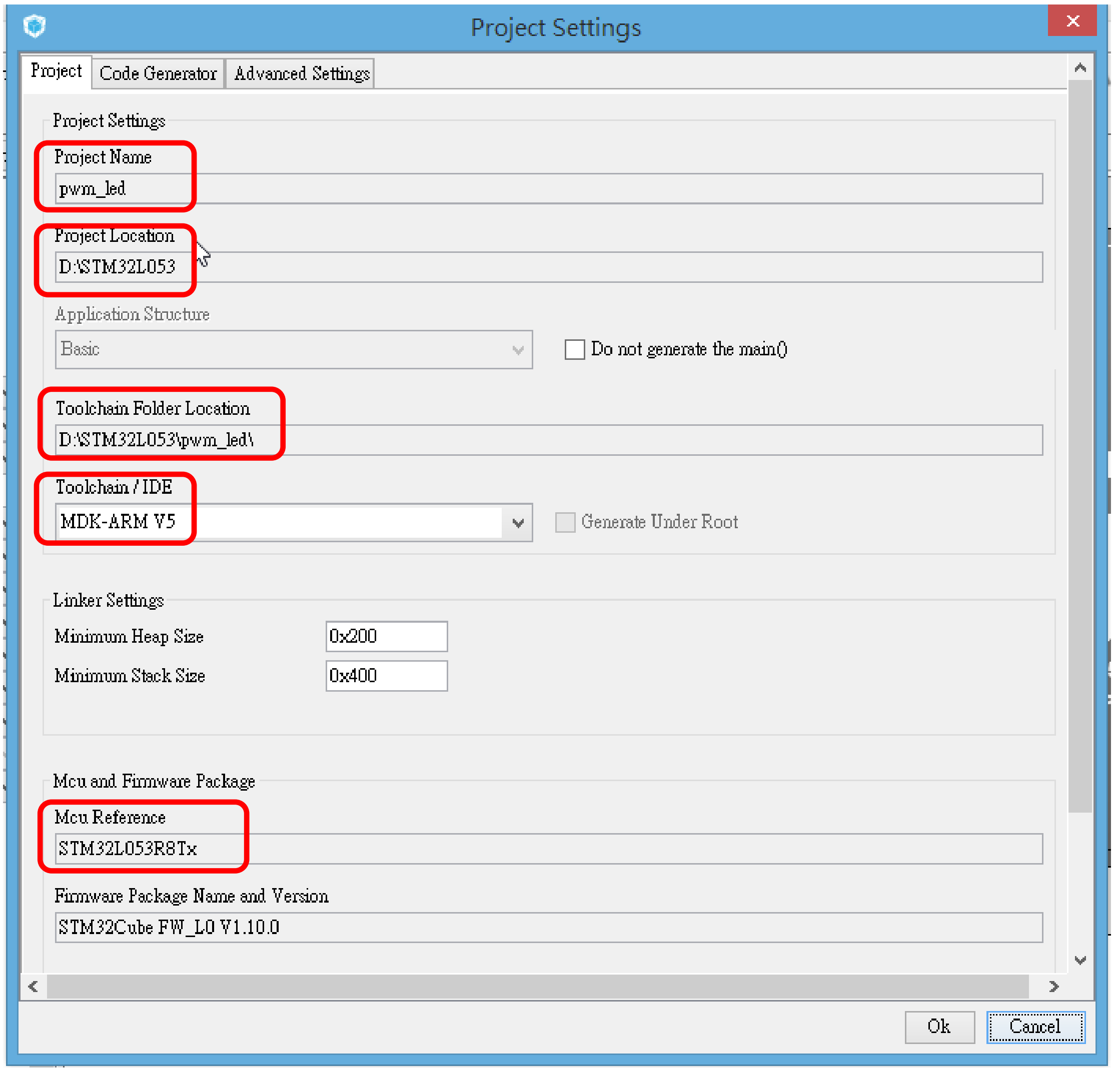
接著確認 RCC Parameters(這是內建的，所以不用調整)。



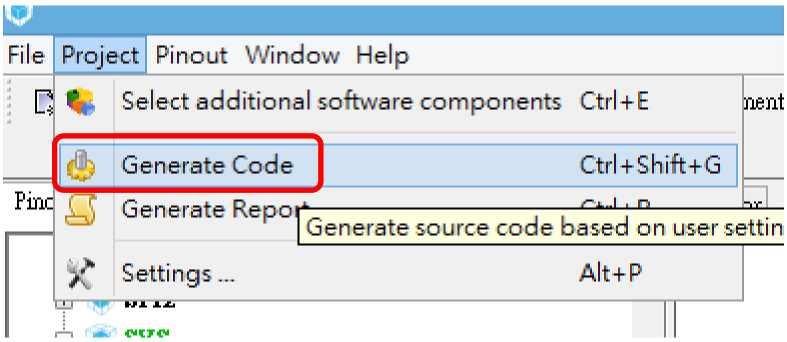
接著設定 TIM2 的 Counter 值，如 prescaler ...等參數值。
這個項目很重要，須確實輸入。



Project 設定項目再確認，如下圖:



以上設定完後，可選擇 Project --> Generate Code 來產生程式碼。



至此，STM 的 CUBE 工作已完成，接下來僅需鑽寫主程式即可。
注意: 將程式碼填入 USER CODE 區內，且不要將其原有的註解刪除，以避免程式碼在自動產生時被刪除。

```c
/* USER CODE BEGIN 0 */

void user_pwm_setvalue(uint16_t value)
{
  TIM_OC_InitTypeDef sConfigOC;

  sConfigOC.OCMode = TIM_OCMODE_PWM1;
  sConfigOC.Pulse = value;
  sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
  sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
  HAL_TIM_PWM_ConfigChannel(&htim2, &sConfigOC, TIM_CHANNEL_1);
  HAL_TIM_PWM_ConfigChannel(&htim2, &sConfigOC, TIM_CHANNEL_2);
  HAL_TIM_PWM_ConfigChannel(&htim2, &sConfigOC, TIM_CHANNEL_3);
  HAL_TIM_PWM_ConfigChannel(&htim2, &sConfigOC, TIM_CHANNEL_4);
  HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
  HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_2);
  HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_3);
  HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_4);
}

/* USER CODE END 0 */
```

Main 主程式內的變數宣告，如下圖:

```c
int main(void)
{
  /* USER CODE BEGIN 1 */
      int pwm_value = 0;
      int step;

  /* USER CODE END 1 */

  /* MCU Configuration--------------------
```

程式 While 迴圈，如下圖:

```
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
  /* USER CODE END WHILE */
  /* USER CODE BEGIN 3 */
      HAL_Delay(250);
    if(pwm_value == 0) step = 100;
    if(pwm_value == 2000) step = -100;
    pwm_value += step;
    user_pwm_setvalue(pwm_value);
  }
  /* USER CODE END 3 */

}
```

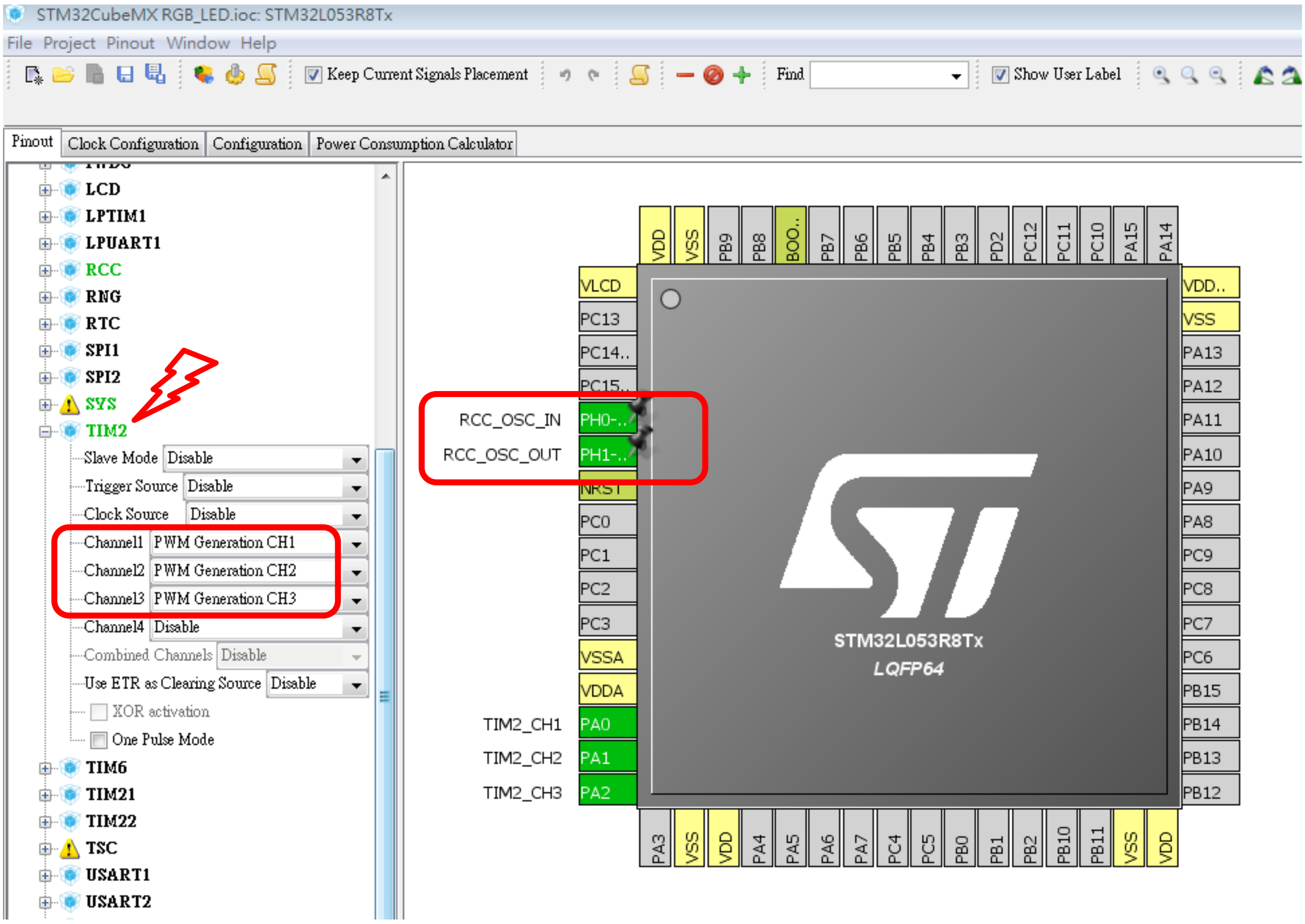
驅動三色 LED

檔案名稱: 06-RGB_LED.doc

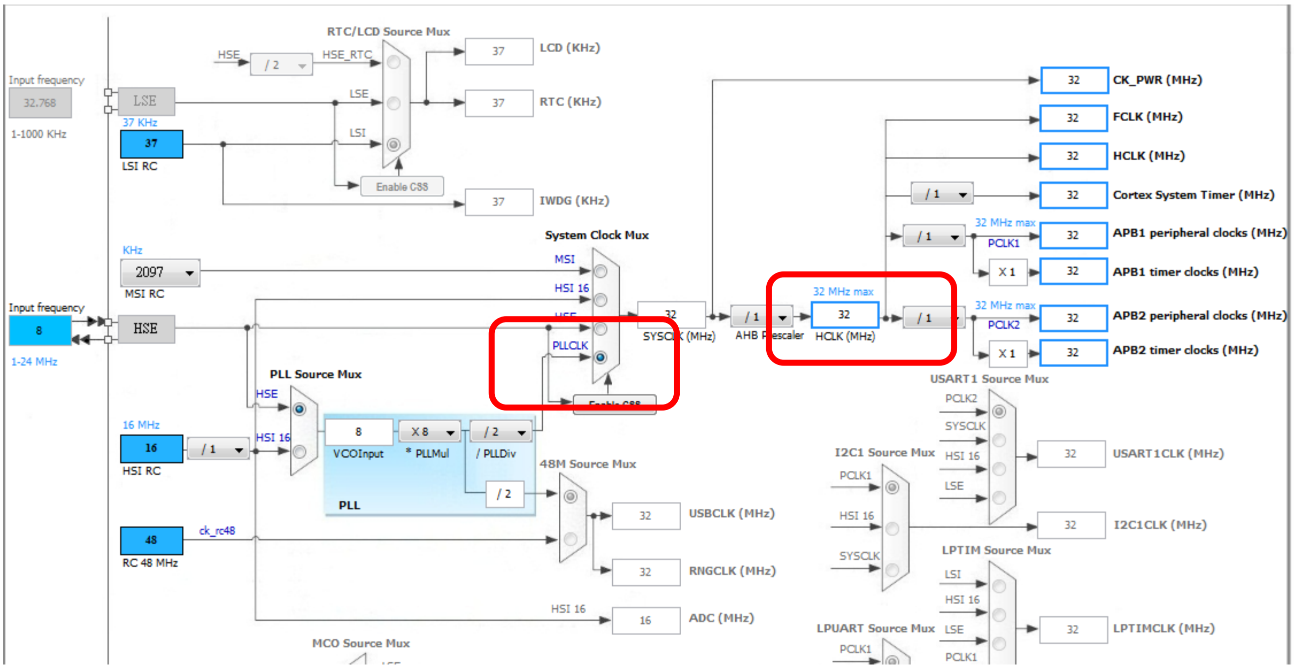
本例是利用晶片內部的計時器及相對應的輸出腳位推動 LED。STM 的本顆晶片有特定相對的輸出腳位，有別於其他廠家的晶片，如 Cypress，Nuvoton 的部分晶片有提供內建的 CPLD 可重新規劃輸出輸入腳位。
首先參考「RGB_LED-參考設計.pdf」先建立 Cube 專案檔，依照下列步驟：

選擇欲使用的晶片可參考其他範例建立 Cube 專案檔。
在建立專案檔後依下列方式設定

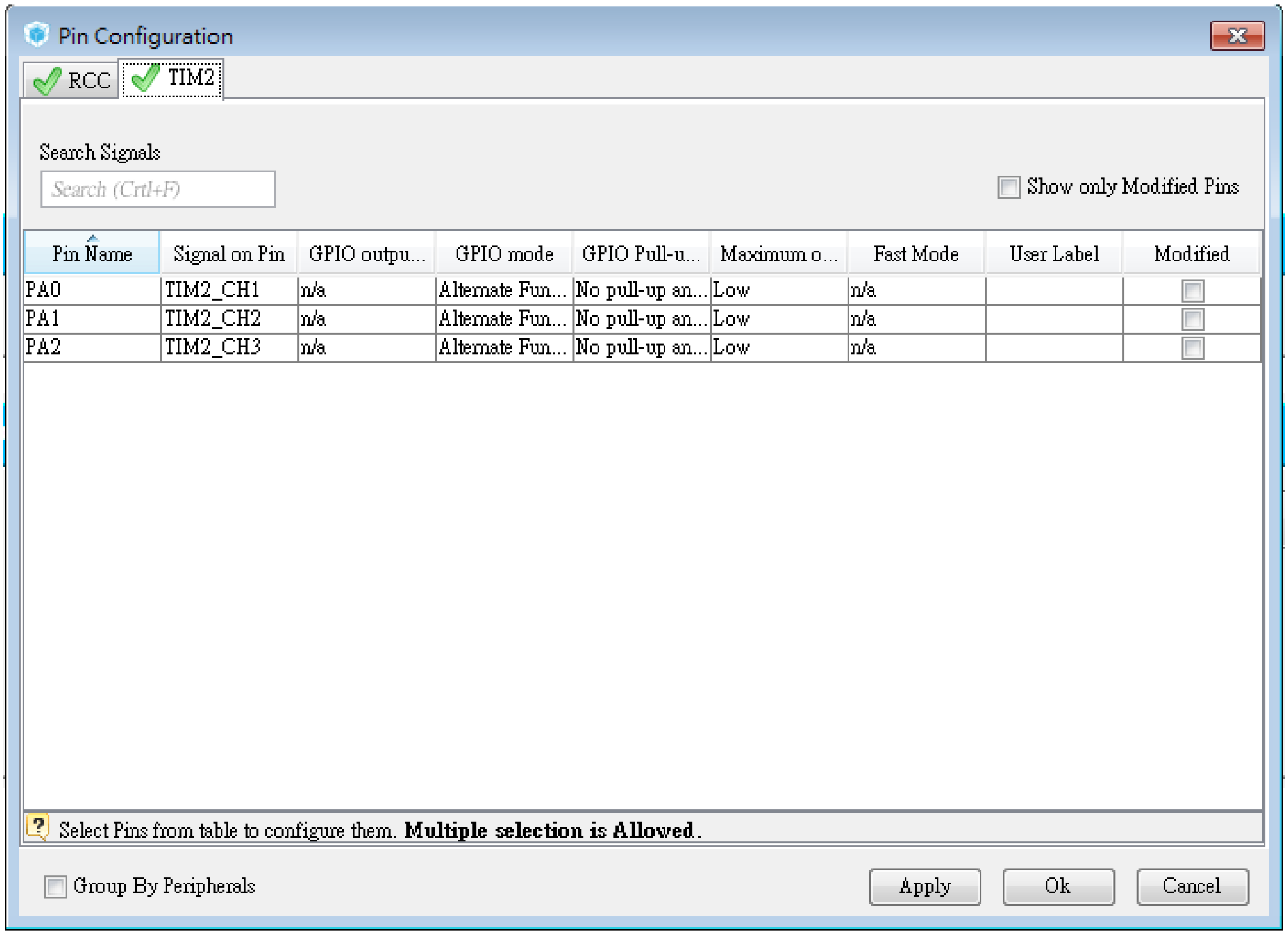
首先在 PH0 及 PH1 點選 RCC_OSC_IN 及 RCC_OSC_OUT。以選擇即設定內部 Clock。
在 Pinout 選項內的 TIM2，應將 Channel1 ~ Channel1 依序選擇 PWM Generation CH1 ~ PWM Generation CH3，將利用這三個 Pin 作為驅動 LED 用，此時可看到 PA0 ~ PA2 三個腳位已被自動設定。



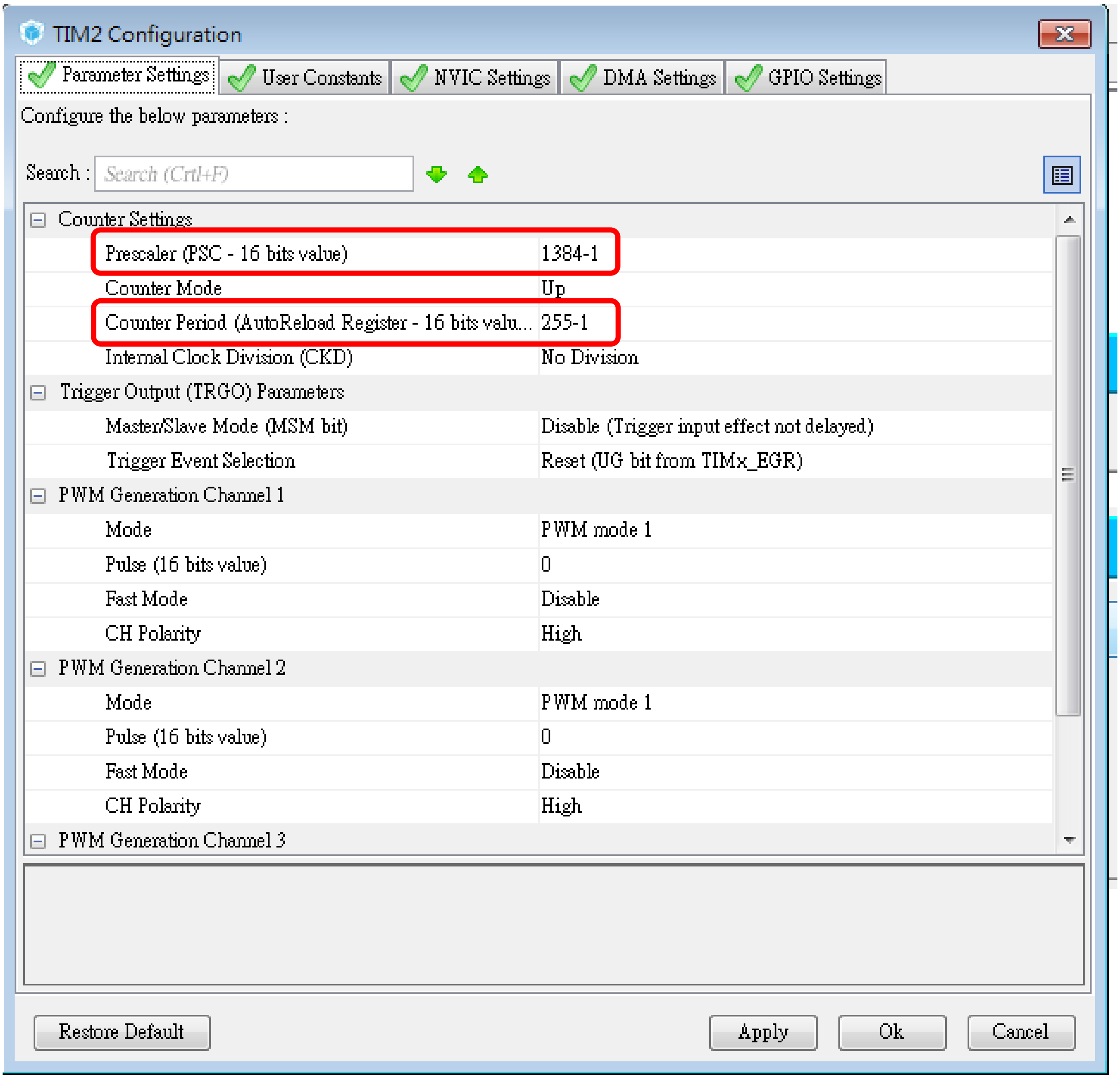
接著在「Clock configuration」選項內點選 clock 的來源為 PLLCLK，再將 HCLK 的頻率設定為最大（即 32MHz）。



在 Configuration →GPIO 選項內觀察 TIM2 選項內的 PA0~PA2。如果有需要可再設定其他功能，在本例中均不另做設定。



接著在 Configuration→Parameter Setting 內的 Counter settings，將 Prescacler 及 Counter period 設定值如下圖。



在 main.c 主程式中加入以下程式碼：

```
/* USER CODE END PFP */

/* USER CODE BEGIN 0 */


// set values to the led


void rgb_set (uint8_t red, uint8_t blue, uint8_t green)
{
  htim2.Instance->CCR1 = red;
  htim2.Instance->CCR2 = blue;
  htim2.Instance->CCR3 = green;
  //htim2.Instance->CCR4 = green;
}

/* USER CODE END 0 */
```

```
  MX_TIM2_Init();
  /* USER CODE BEGIN 2 */
    HAL_TIM_PWM_Start (&htim2, TIM_CHANNEL_1);
    HAL_TIM_PWM_Start (&htim2, TIM_CHANNEL_2);
    HAL_TIM_PWM_Start (&htim2, TIM_CHANNEL_3);
    //HAL_TIM_PWM_Start (&htim2, TIM_CHANNEL_4);
  /* USER CODE END 2 */

```

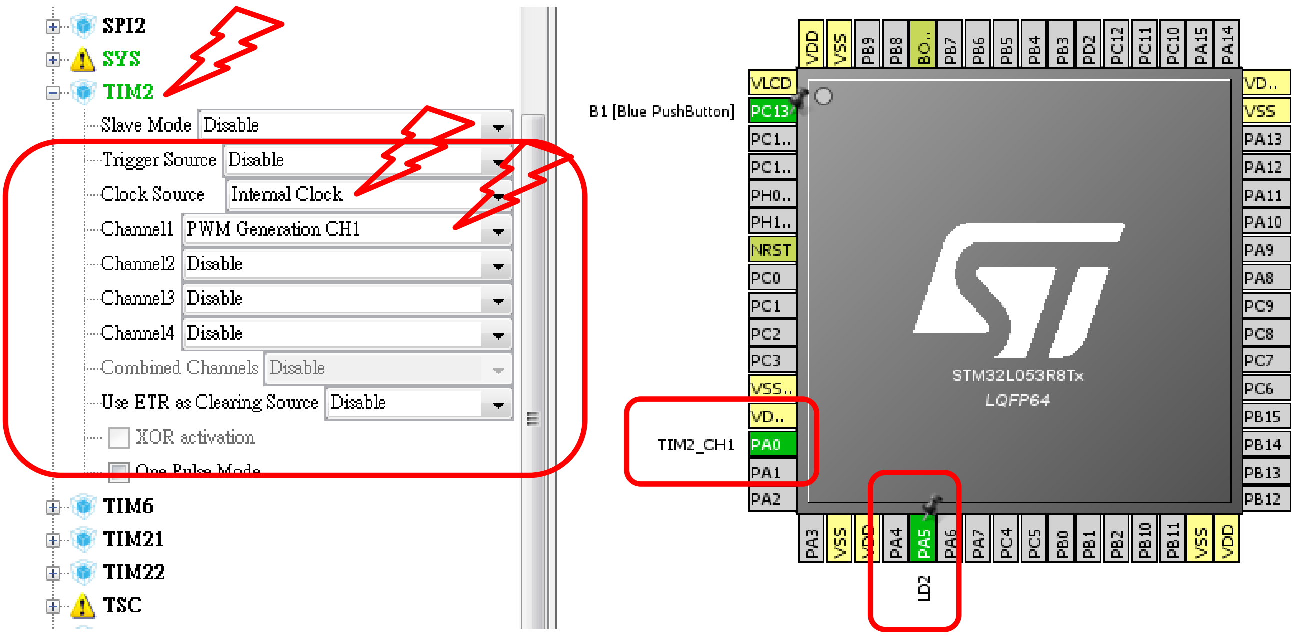
```
while (1)
{
/* USER CODE END WHILE */

/* USER CODE BEGIN 3 */

  rgb_set (255,0,0); // only red
  HAL_Delay (1000);

  rgb_set (0,255,0); // only blue
  HAL_Delay (1000);

  rgb_set (0,0,255); // only green
  HAL_Delay (1000);

  rgb_set (255,255,0);
  HAL_Delay (1000);

  rgb_set (0,255,255);
  HAL_Delay (1000);

  rgb_set (255,0,255);
  HAL_Delay (1000);

  rgb_set (192,192,192);
  HAL_Delay (1000);

}
/* USER CODE END 3 */
```

驅動 WS2812B 彩色燈條

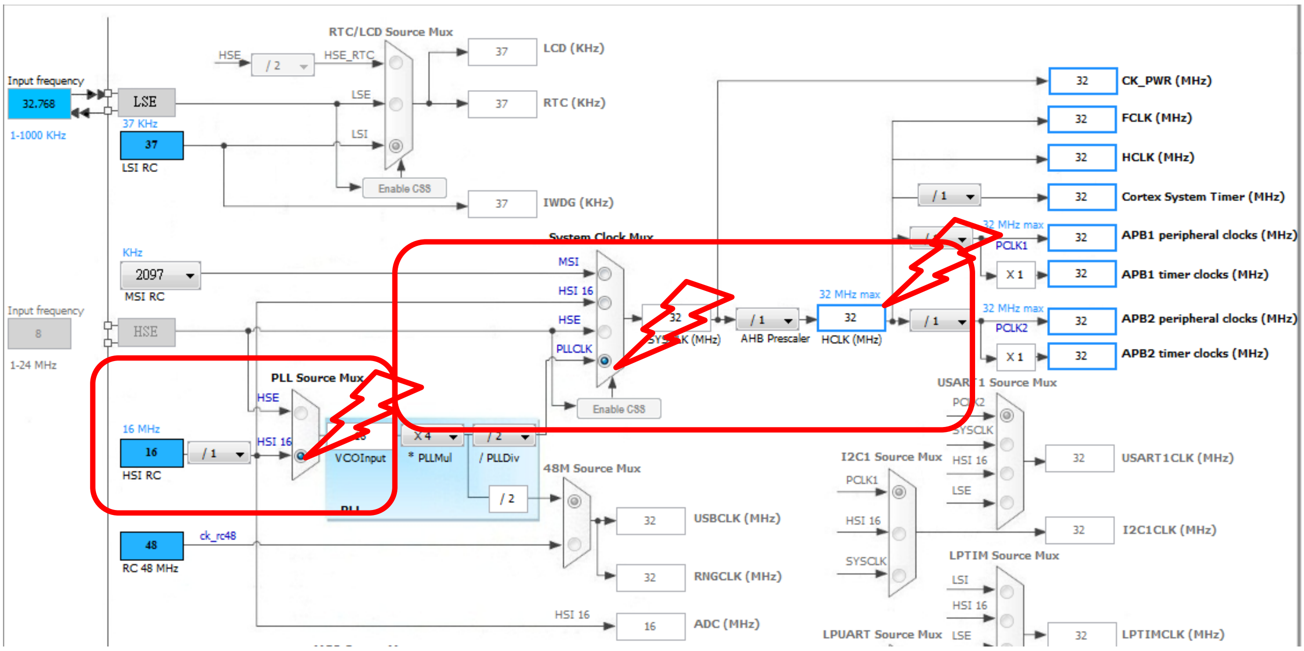
檔案名稱: 07-驅動 WS2812B 彩色燈條.doc

本例是利用 PWM 方式送出訊號來驅動 WS2812B 的串珠燈條,詳細及控制方式可參考 WS2812B 的規格書。
燈條的接線：紅色線接+5V，白色線接 GND，綠色線接到 PA0。
開始時 LD2(版上的 LED)接腳是 PA5，會一直閃爍，當按下 PC13(板子上藍色按鈕)時，所有動作會停止，PC13 在程式定義是 B1，放開按鈕後燈條則繼續進行閃爍。
本例我使用的燈條依公式 15 顆 LED。

先依序點選 Pinout→TIM2→Channel1→PWM Generation CH1 如圖，在 Clock Source→Internal Clock。
PA0 腳位會呈現綠色，表示已經選上了，並與 TIMER2 形成了相依。
接著將 PA5 的腳位設為輸出，並改腳位名稱(Lable)為 LD2。



在 Clock configuration 設定內，Clock 來源選擇 HSI，並選擇 System Clock 選為 PLLCLK，並將 HCLK 用手動方式設為最大值(32MHz)。



請將以下程式碼載入主程式中。

```
/* USER CODE BEGIN 0 */
#define H_VAL 26
#define L_VAL 14
#define N_LEDS 15                    //LED ???, ?????15?
#define BITS_PER_LED (3*8)
#define BIT_BUF_SIZE (N_LEDS * BITS_PER_LED)
uint16_t ws2812BitBuf[BIT_BUF_SIZE + 1]; //DMA transfer needs one byte more with ZERO,
because it will be output after DMA transfer has finished

void ws2812_set_color(int led, uint8_t r, uint8_t g, uint8_t b)
{
   if (led >= N_LEDS) return;
   int i = led * BITS_PER_LED;
   uint8_t mask;
   mask = 0x80;
   while(mask) {
      ws2812BitBuf[i] = (mask & g)?H_VAL:L_VAL;
      mask >>= 1;
      i++;
   }
   mask = 0x80;
   while(mask) {
      ws2812BitBuf[i] = (mask & r)?H_VAL:L_VAL;
      mask >>= 1;
```

```
        i++;
    }
    mask = 0x80;
    while(mask) {
        ws2812BitBuf[i] = (mask & b)?H_VAL:L_VAL;
        mask >>= 1;
        i++;
    }
}
/* USER CODE END 0 */

// ==========================================================
/* USER CODE BEGIN 2 */
//memset(ws2812BitBuf, 0, sizeof(ws2812BitBuf));
    for (int led = 0; led < N_LEDS; led++) ws2812_set_color(led, 2, 2, 2);
    HAL_TIM_PWM_Start_DMA(&htim2, TIM_CHANNEL_1, (uint32_t*)ws2812BitBuf,
sizeof(ws2812BitBuf) / sizeof(ws2812BitBuf[0]));
    HAL_Delay(10);
    /* USER CODE END 2 */

// ==========================================================
/* Infinite loop */
    /* USER CODE BEGIN WHILE */
    while (1)
    {

    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
static int cursorled = 0;
        static uint8_t r = 0;
        static uint8_t g = 0;
        static uint8_t b = 0;
        static uint8_t state = 0;
        if (GPIO_PIN_SET == HAL_GPIO_ReadPin(B1_GPIO_Port, B1_Pin)) {
            for (int led = 0; led < N_LEDS; led++) ws2812_set_color(led, r, g, b);
            cursorled++;
            if (N_LEDS <= cursorled) {
                cursorled = 0;
            }
            switch (state) {
                case 0:
                    r++;
                    if (g) g--;
                    b = 0;
                    if (r == 255) {
                        state++;
```

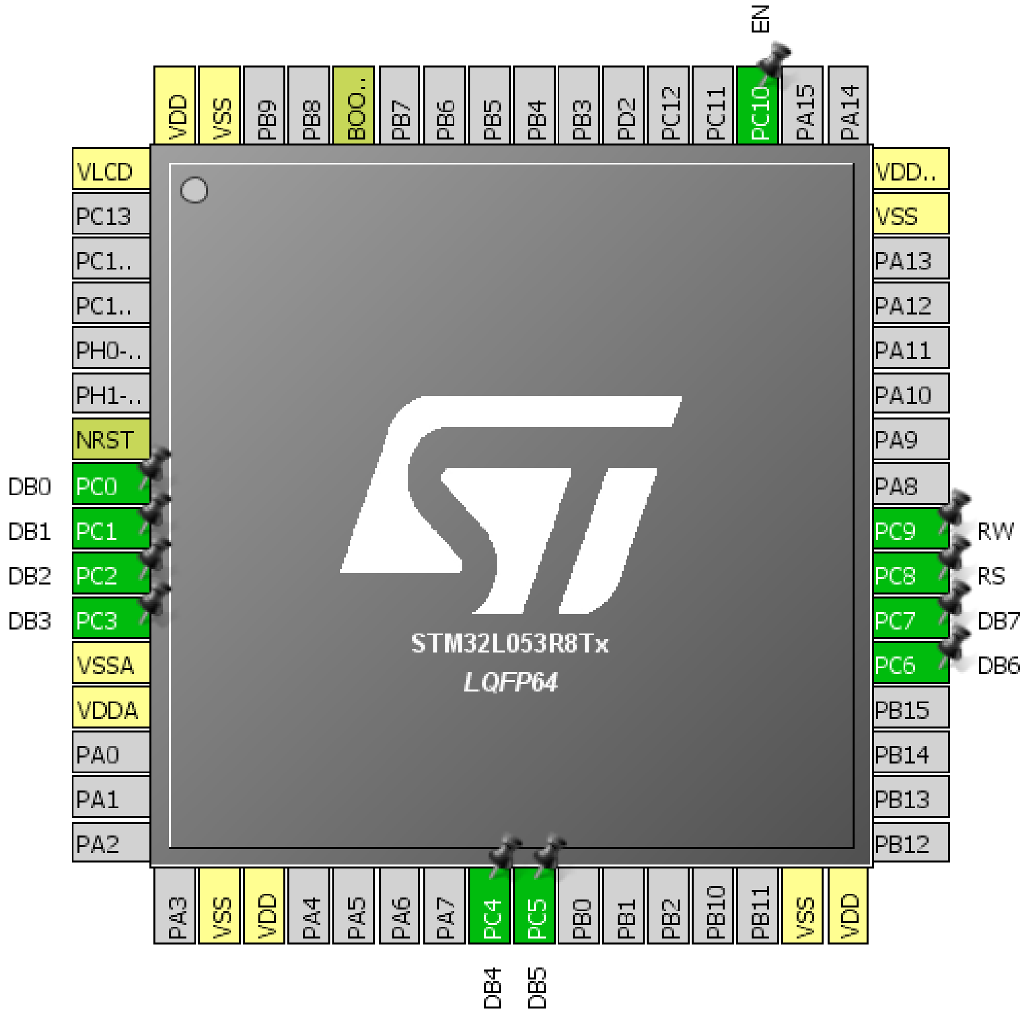
```
                }
                break;
            case 1:
                if (r) r--;
                g = 0;
                b++;
                if (b == 255) {
                    state++;
                }
                break;
            case 2:
                r = 0;
                g++;
                if (b) b--;
                if (g == 255) {
                    state++;
                }
                break;
            default:
                state = 0;
                break;
        }
        ws2812_set_color(cursorled, 255, 255, 255);
        HAL_TIM_PWM_Stop_DMA(&htim2, TIM_CHANNEL_1);
        HAL_TIM_PWM_Start_DMA(&htim2, TIM_CHANNEL_1, (uint32_t*)ws2812BitBuf,
sizeof(ws2812BitBuf) / sizeof(ws2812BitBuf[0]));
        HAL_GPIO_TogglePin(LD2_GPIO_Port, LD2_Pin);
        HAL_Delay(50);

//          HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_SET);
      }
      else {
        HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);
      }
    }
    /* USER CODE END 3 */

}
```

LCD 顯示器範例

專案名稱：08-LCDx1A2(HAL)

在本例中，先假設已經十分熟悉 CUBE 專案的建置，及設定的方式。
本範例中以 GPIO Pin 作為驅動 LCD 的方式，僅需簡單的設定即可，關於 LCD 的規範可參考 LCD 1602 的規格書。

如下圖所示，並參考 Pins configuration 設定各個 Pin 並加入該 Pin 的 Lable。



3. Pins Configuration

Pin Number LQFP64	Pin Name (function after reset)	Pin Type	Alternate Function(s)	Label
1	VLCD	Power		
7	NRST	Reset		
8	PC0 *	I/O	GPIO_Output	DB0
9	PC1 *	I/O	GPIO_Output	DB1
10	PC2 *	I/O	GPIO_Output	DB2
11	PC3 *	I/O	GPIO_Output	DB3
12	VSSA	Power		
13	VDDA	Power		
18	VSS	Power		
19	VDD	Power		
24	PC4 *	I/O	GPIO_Output	DB4
25	PC5 *	I/O	GPIO_Output	DB5
31	VSS	Power		
32	VDD	Power		
37	PC6 *	I/O	GPIO_Output	DB6
38	PC7 *	I/O	GPIO_Output	DB7
39	PC8 *	I/O	GPIO_Output	RS
40	PC9 *	I/O	GPIO_Output	RW
47	VSS	Power		
48	VDD_USB	Power		
51	PC10 *	I/O	GPIO_Output	EN
60	BOOT0	Boot		
63	VSS	Power		
64	VDD	Power		

參考以下接腳方式連接 LCD 及 STM32L053 的接腳。

/************* LCD1******LCD 顯示器實習**********
*動作：由 LCD 顯示兩行文字,令其閃爍或移位
*接線：PC0~7-->DB0~7, PC8~10-->RS、RW、EN
***/

```
//*****宣告副程式
void LCD_Data(uint8_t dat);   //傳送資料到LCD
void LCD_Cmd(uint8_t Cmd);    //傳送命令到LCD
void LCD_init(void);              //LCD的啟始程式

//#define    Data GPIOC->ODR     //資料BUS輸出
```

Source Code：（可將以下的原始碼複製到如下的位置下）

/* USER CODE BEGIN PV */

/* Private variables ---*/

void SystemClock_Config(void);

static void MX_GPIO_Init(void);

註；以上不用複製

```
//*****宣告副程式
void LCD_Data(uint8_t dat);    //傳送資料到 LCD
void LCD_Cmd(uint8_t Cmd);          //傳送命令到 LCD
void LCD_init(void);                //LCD 的啟始程式
//#define      Data GPIOC->ODR       //資料 BUS 輸出
```

繼續如下圖，將 Source Code 複製到重確的位置。

```
int main(void)
{
  /* USER CODE BEGIN 1 */
    uint8_t i;          //資料計數
   HAL_Init();
   SystemClock_Config();
   MX_GPIO_Init();
   LCD_init();    //重置及清除LCD
   //LCD_Cmd(0x0F);//0000 1111
     //bit2:D=1,顯示幕ON
     //bit1:C=1,顯示游標
     //bit0:B=1,游標閃爍
   //LCD_Cmd(0x04); //0000 0100,
     //bit1:I/D=0，游標左移反向顯示

   LCD_Cmd(0x85);      //游標由第一行第5個字開始顯示
   for(i='0'; i<='9';i++) //字元a~j
    {
     LCD_Data(i);   //字元送到LCD顯示
     HAL_Delay(300);//延時，慢速逐一顯示
      }

   LCD_Cmd(0xC5); //游標由第二行第5個字開始顯示
   for(i='A'; i<= 'J';i++)//LCD顯示字元A~J
    {
         LCD_Data(i);  //字元送到LCD顯示
       HAL_Delay(100);//延時，慢速逐一顯示
      }
  /* USER CODE END 1 */
```

```
/* USER CODE BEGIN 1 */
      uint8_t i;          //資料計數
   HAL_Init();
   SystemClock_Config();
   MX_GPIO_Init();
   LCD_init();      //重置及清除 LCD
   //LCD_Cmd(0x0F);//0000 1111
     //bit2:D=1,顯示幕 ON
     //bit1:C=1,顯示游標
     //bit0:B=1,游標閃爍
   //LCD_Cmd(0x04); //0000 0100,
     //bit1:I/D=0，游標左移反向顯示

   LCD_Cmd(0x85);        //游標由第一行第 5 個字開始顯示
   for(i='0'; i<='9';i++) //字元 a~j
```

```
    {
     LCD_Data(i);    //字元送到 LCD 顯示
     HAL_Delay(300);//延時，慢速逐一顯示
       }

  LCD_Cmd(0xC5); //游標由第二行第 5 個字開始顯示
  for(i='A'; i<= 'J';i++)//LCD 顯示字元 A~J
    {
          LCD_Data(i);    //字元送到 LCD 顯示
       HAL_Delay(100);//延時，慢速逐一顯示
      }
 /* USER CODE END 1 */
```

```
/* USER CODE BEGIN 4 */
/*************************************************************
*函數名稱: LCD_Data
*功能描述: 傳送資料到文字型LCD
*輸入參數：dat
*************************************************************/
void LCD_Data(uint8_t dat)   //傳送資料到LCD
{
    //uint8_t dly=2;
      //Data=dat;                  //資料送到BUS
            HAL_GPIO_WritePin(GPIOC, dat , GPIO_PIN_SET);
            HAL_GPIO_WritePin(GPIOC, ~dat, GPIO_PIN_RESET);

    //RS_1; RW_0; EN_1; //資料寫入到LCD內

        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_8, GPIO_PIN_SET);         //RS --> 1
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_9, GPIO_PIN_RESET);       //RW --> 0
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_10, GPIO_PIN_SET);        //EN --> 1
    //while(dly--);
        HAL_Delay(2);
      //EN_0;              //禁能LCD
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_10, GPIO_PIN_RESET);    //EN --> 0
      HAL_Delay(1);     //LCD等待寫入完成
}
```

```
/* USER CODE BEGIN 4 */
/*************************************************************
*函數名稱: LCD_Data
*功能描述: 傳送資料到文字型 LCD
*輸入參數：dat
*************************************************************/
void LCD_Data(uint8_t dat)   //傳送資料到 LCD
{
     //uint8_t dly=2;
```

```
    //Data=dat;                         //資料送到 BUS
            HAL_GPIO_WritePin(GPIOC, dat , GPIO_PIN_SET);
            HAL_GPIO_WritePin(GPIOC, ~dat, GPIO_PIN_RESET);
  //RS_1; RW_0; EN_1; //資料寫入到 LCD 內

      HAL_GPIO_WritePin(GPIOC, GPIO_PIN_8, GPIO_PIN_SET);              //RS --> 1
      HAL_GPIO_WritePin(GPIOC, GPIO_PIN_9, GPIO_PIN_RESET);            //RW --> 0
      HAL_GPIO_WritePin(GPIOC, GPIO_PIN_10, GPIO_PIN_SET);             //EN --> 1
  //while(dly--);
      HAL_Delay(2);
    //EN_0;                    //禁能 LCD
      HAL_GPIO_WritePin(GPIOC, GPIO_PIN_10, GPIO_PIN_RESET);       //EN --> 0
    HAL_Delay(1);       //LCD 等待寫入完成
}
```

```
/****************************************************************
*函數名稱: LCD_Cmd
*功能描述: 傳送命令到文字型LCD
*輸入參數：Cmd
*************************************************************/
void LCD_Cmd(uint8_t Cmd) //傳送命令到LCD
{
    //Data=Cmd;                          //命令送到BUS
            HAL_GPIO_WritePin(GPIOC, Cmd , GPIO_PIN_SET);
            HAL_GPIO_WritePin(GPIOC, ~Cmd, GPIO_PIN_RESET);
    //RS_0; RW_0; EN_1; //命令寫入到LCD內
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_8, GPIO_PIN_RESET);          //RS --> 0
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_9, GPIO_PIN_RESET);          //RW --> 0
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_10, GPIO_PIN_SET);           //EN --> 1
    HAL_Delay(1);
    //EN_0;                //禁能LCD
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_10, GPIO_PIN_RESET);     //EN --> 0
    HAL_Delay(1);
}
```

```
/****************************************************************
*函數名稱: LCD_Cmd
*功能描述: 傳送命令到文字型 LCD
*輸入參數：Cmd
*************************************************************/
void LCD_Cmd(uint8_t Cmd) //傳送命令到 LCD
{
  //Data=Cmd;                         //命令送到 BUS
```

```
            HAL_GPIO_WritePin(GPIOC, Cmd , GPIO_PIN_SET);
            HAL_GPIO_WritePin(GPIOC, ~Cmd, GPIO_PIN_RESET);
    //RS_0; RW_0; EN_1; //命令寫入到 LCD 內
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_8, GPIO_PIN_RESET);       //RS --> 0
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_9, GPIO_PIN_RESET);       //RW --> 0
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_10, GPIO_PIN_SET);        //EN --> 1
    HAL_Delay(1);
    //EN_0;            //禁能 LCD
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_10, GPIO_PIN_RESET);      //EN --> 0
    HAL_Delay(1);
}
```

```
/*****************************************************************
 *函數名稱: LCD_init
 *功能描述: 啟始化文字型LCD
 *****************************************************************/
void LCD_init(void)     //LCD的啟始程式
{
    LCD_Cmd(0x38); //0011 1000,8bit傳輸,顯示2行,5*7字型
    LCD_Cmd(0x38); //bit4:DL=1,8bit傳輸,
    LCD_Cmd(0x38); //bit3:N=1,顯示2行
                                     //bit2:F=0,5*7字型
    LCD_Cmd(0x0c); //0000 1100,顯示幕ON,不顯示游標,游標不閃爍
                                     //bit2:D=1,顯示幕ON
                                     //bit1:C=0,不顯示游標
                                       //bit0:B=0,游標不閃爍
    LCD_Cmd(0x06); //0000 0110,//顯示完游標右移,游標移位禁能
                                     //bit1:I/D=1,顯示完游標右移,
                                     //bit0:S=0,游標移位禁能
    LCD_Cmd(0x01); //清除顯示幕
    LCD_Cmd(0x02); //游標回原位
}
/* USER CODE END 4 */
```

```
/*****************************************************************
 *函數名稱: LCD_init
 *功能描述: 啟始化文字型 LCD
 *****************************************************************/
void LCD_init(void)     //LCD 的啟始程式
{
    LCD_Cmd(0x38); //0011 1000,8bit 傳輸,顯示 2 行,5*7 字型
    LCD_Cmd(0x38); //bit4:DL=1,8bit 傳輸,
    LCD_Cmd(0x38); //bit3:N=1,顯示 2 行
                                  //bit2:F=0,5*7 字型
    LCD_Cmd(0x0c); //0000 1100,顯示幕 ON,不顯示游標,游標不閃爍
                                  //bit2:D=1,顯示幕 ON
```

```
                                                                                //bit1:C=0,不顯示游標
                                                                                   //bit0:B=0,游標不閃爍
        LCD_Cmd(0x06); //0000 0110,//顯示完游標右移,游標移位禁能
                                                                                //bit1:I/D=1,顯示完游標右移,
                                                                                //bit0:S=0,游標移位禁能
        LCD_Cmd(0x01); //清除顯示幕
        LCD_Cmd(0x02); //游標回原位
}
/* USER CODE END 4 */
```

注意：以上均在 User Code 4 之內，要輸入的 Source Code。