關於使用其他開發版

1. 助教在 Lab 課教的方法，只有在「課程指定開發板」上面測試和驗證過。

2. 不論是用哪一型號開發板，只要你能自己克服問題，完成 Lab 要求的功能目標，就會給 Lab 分數。

例如：你的板子上面沒有三軸加速規，所以無法偵測晃動，你可以去電子材料行買一個三軸加速規模組，並用杜邦線接到板子上，自己寫程式讀取訊號輸入，以間接完成偵測晃動輸入訊號的要求。但如何具體完成這個功能，需要同學自行找資料解決問題，助教能幫忙的有限。

每個 Lab 暫訂的功能需求：
Lab 0: STM32 系列開發板，cortex-M4 系列 CPU，且要能找方法確定 FreeFTOS 能在上面跑
Lab 1: 讀取按鈕輸入，改變 3 個 LED 的亮暗模式
Lab 2: 用版上的 usart 功能與 ttl 線，將 FreeRTOS 執行時的某些資料結構資訊輸出到螢幕
Lab 3: 偵測板子晃動，並透過 ISR 改變 3 個 LED 亮暗模式
Lab 4: 修改 FreeRTOS 記憶體管理方式，並用版上的 usart 功能與 ttl 線將資訊輸出到螢幕
Lab 5: 使用 MicroSD card adapter 讀入 SD-Card 內的音樂檔，並透過板上耳機孔輸出音樂

每個 Lab 都是針對課程指定的開發板量身訂做，碩一助教們也是在開學前努力學習與調整每一個 Lab 的相關內容，以確保能正確執行，因此若遇到非課程指定開發板造成的問題，助教們也沒辦法馬上找到方法幫你，必須請同學自行查資料解決問題。只要最後成果與 Lab 要求相同，一樣能拿到分數。
另外，Final Project 並沒有指定使用的開發板，因此可以使用其他開發板。

Lab1 ~ Lab5 共用規定

- 請在 Lab 1 上課前，先完成 Lab 0 的內容。
- 上課時間: 14:10~17:10；地點@新館一樓 65104
- 每一個 Lab 最晚都會在上課當天中午 12:00 前上傳投影片到 moodle，

  為避免教室網路訊號不好，請同學在 14:00 上課前先下載投影片至電腦中。
- 每一個 Lab 佔總分 8%, 獨立計分. (Final Project 佔總分 60%)
- Lab 完成後, 要在 7 天內寫好 Lab report 上傳 moodle。
- 若 Lab 下課前有做完，我們會現場幫你評分。
- 若 Lab 下課前沒做完，會有補交機制(各 Lab 規定方式可能不同)，

  期限內有完成就不會扣分(期限為 7 天內，超過不計分)。

Lab 0 : Porting

若你使用的不是課程指定開發版，請視情況自行調整步驟，並確保能執行 FreeRTOS

OS-Lab Email : oslab@mail.csie.ncku.edu.tw

Download STM32CubeIDE (下載專用的IDE)

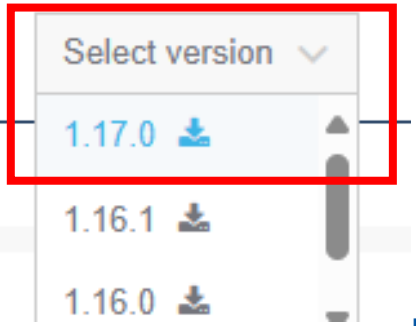
Integrated Development Environment for STM32

https://www.st.com/en/development-tools/stm32cubeide.html

Download STM32CubeIDE (下載專用的IDE)

根據電腦的作業系統，選擇對應的安裝檔

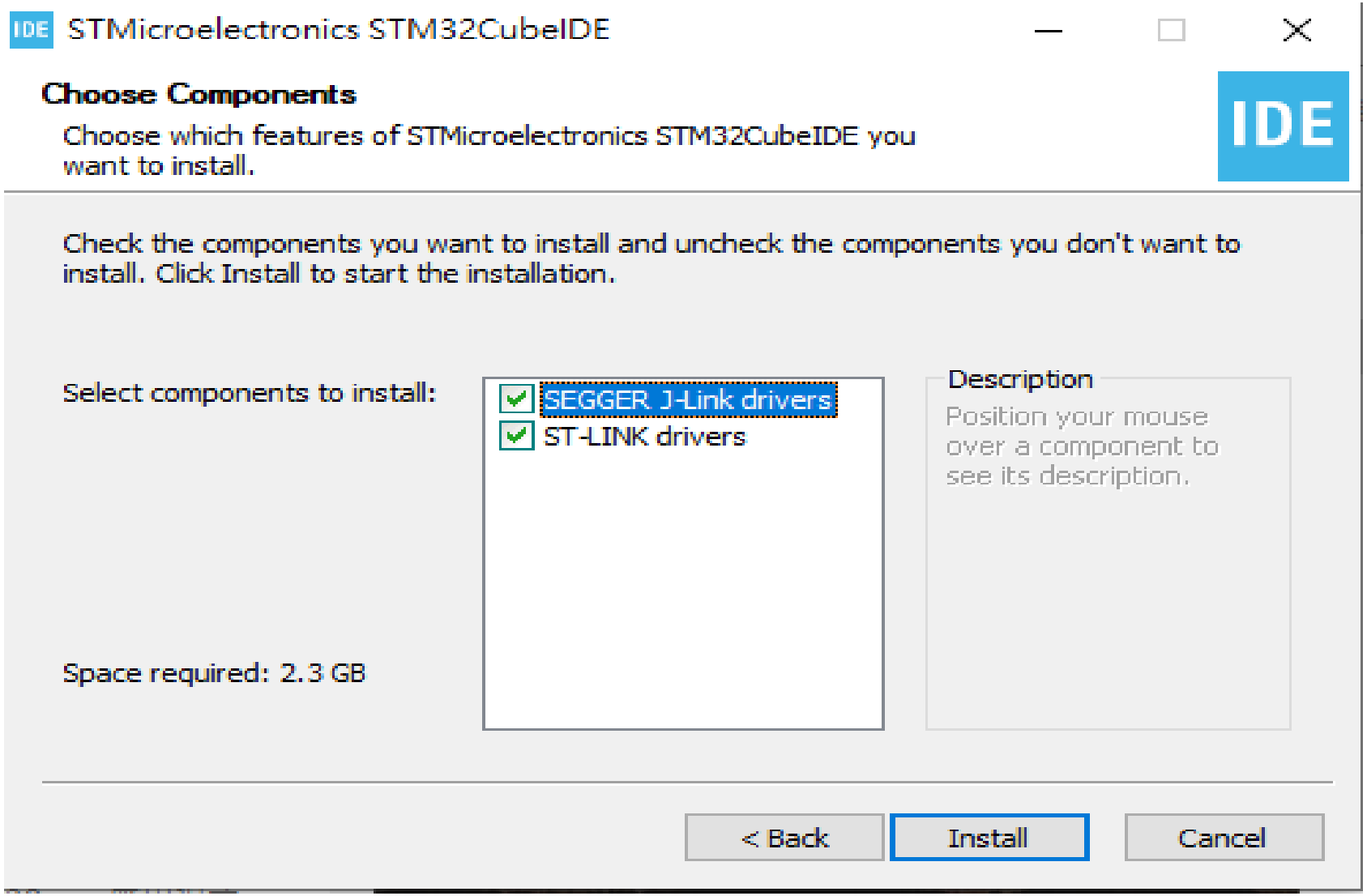
	Part Number	General Description	Latest version	Download	All versions
+	STM32CubeIDE-DEB	STM32CubeIDE Debian Linux Installer	1.17.0	Get latest	Select version
+	STM32CubeIDE-Lnx	STM32CubeIDE Generic Linux Installer	1.17.0	Get latest	Select version
+	STM32CubeIDE-Mac	STM32CubeIDE macOS Installer	1.17.0	Get latest	Select version
+	STM32CubeIDE-RPM	STM32CubeIDE RPM Linux Installer	1.17.0	Get latest	Select version
+	STM32CubeIDE-Win	STM32CubeIDE Windows Installer	1.17.0	Get latest	Select version



1.17.0 版本的 IDE

下載的 IDE 需登入才能正常使用，可先在此網站註冊

Install STM32CubeIDE (安裝專用的IDE)



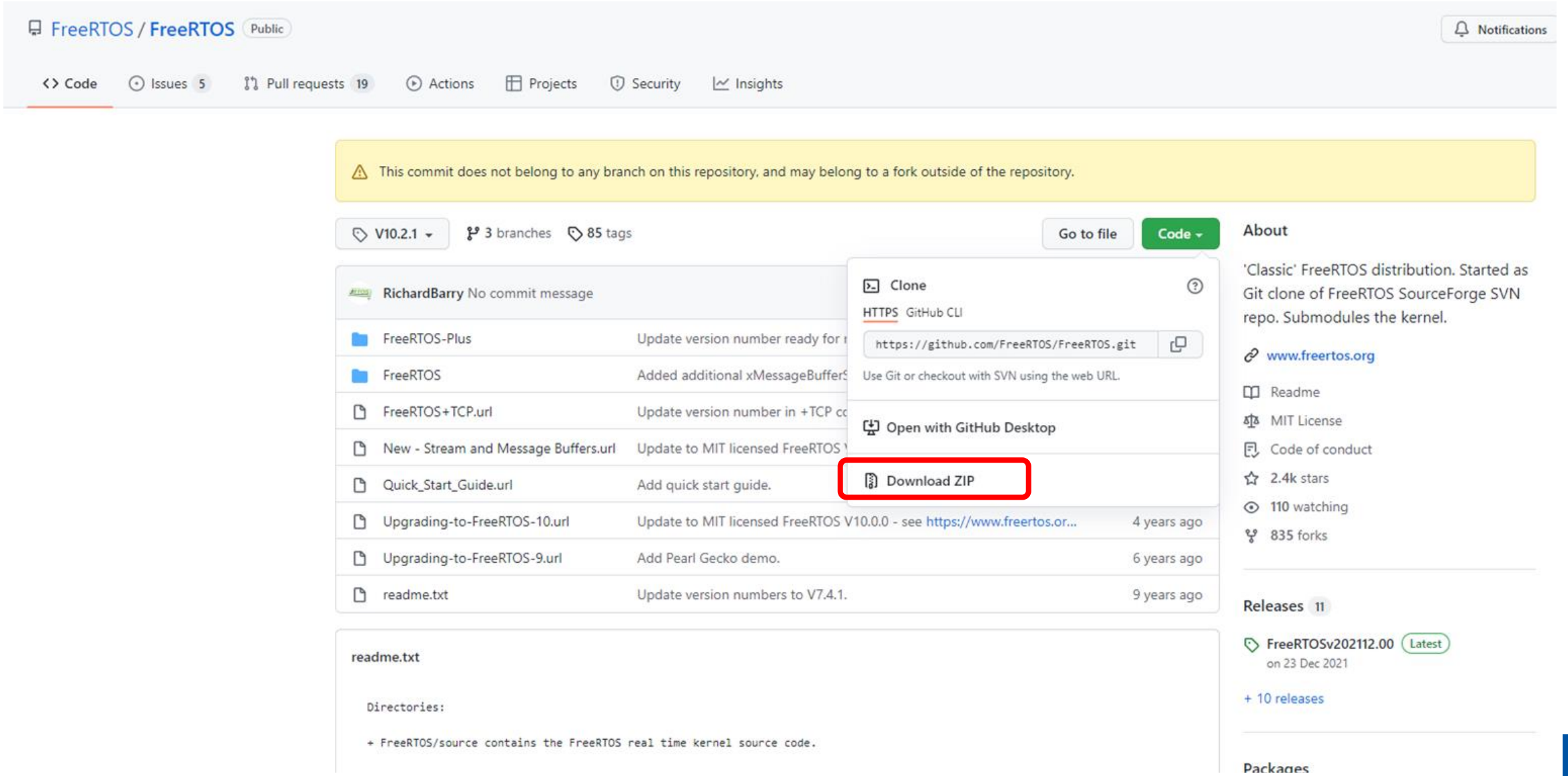
Download FreeRTOS source code (下載FreeRTOS作業系統原始碼)

先下載下來，解壓縮，之後會用到

Github: FreeRTOS v10.2.1

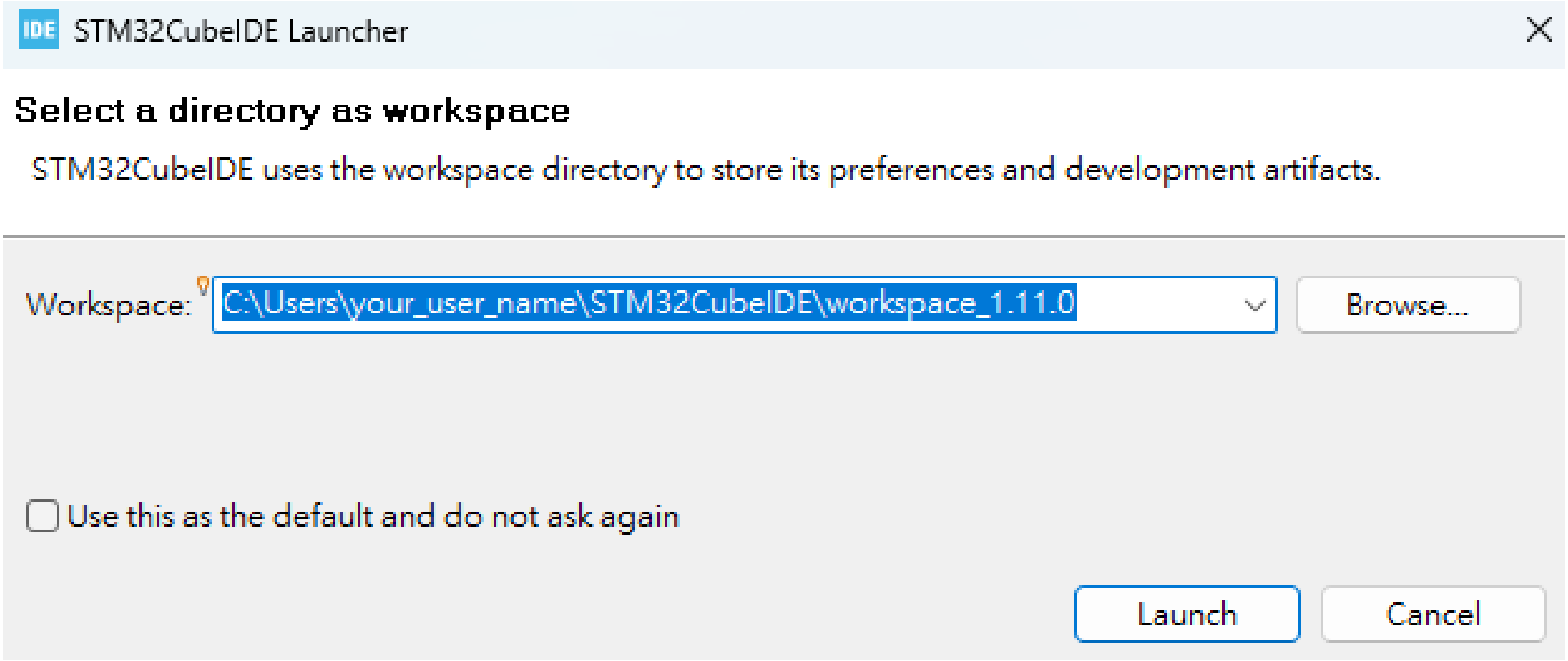
https://github.com/FreeRTOS/FreeRTOS/tree/V10.2.1

Download FreeRTOS source code (下載FreeRTOS作業系統原始碼)



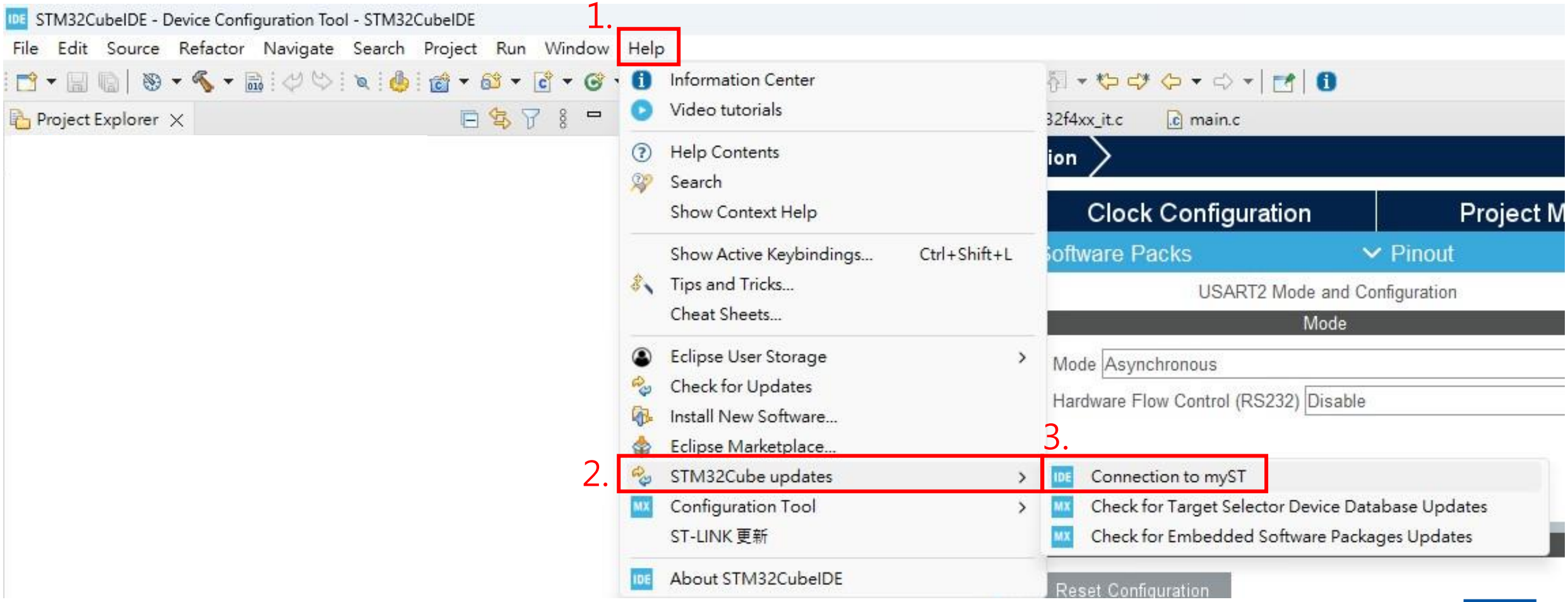
Create STM32 project with STM32CubeIDE (在IDE建立新專案)

Workspace 我們沒有規定，也可以直接用預設的

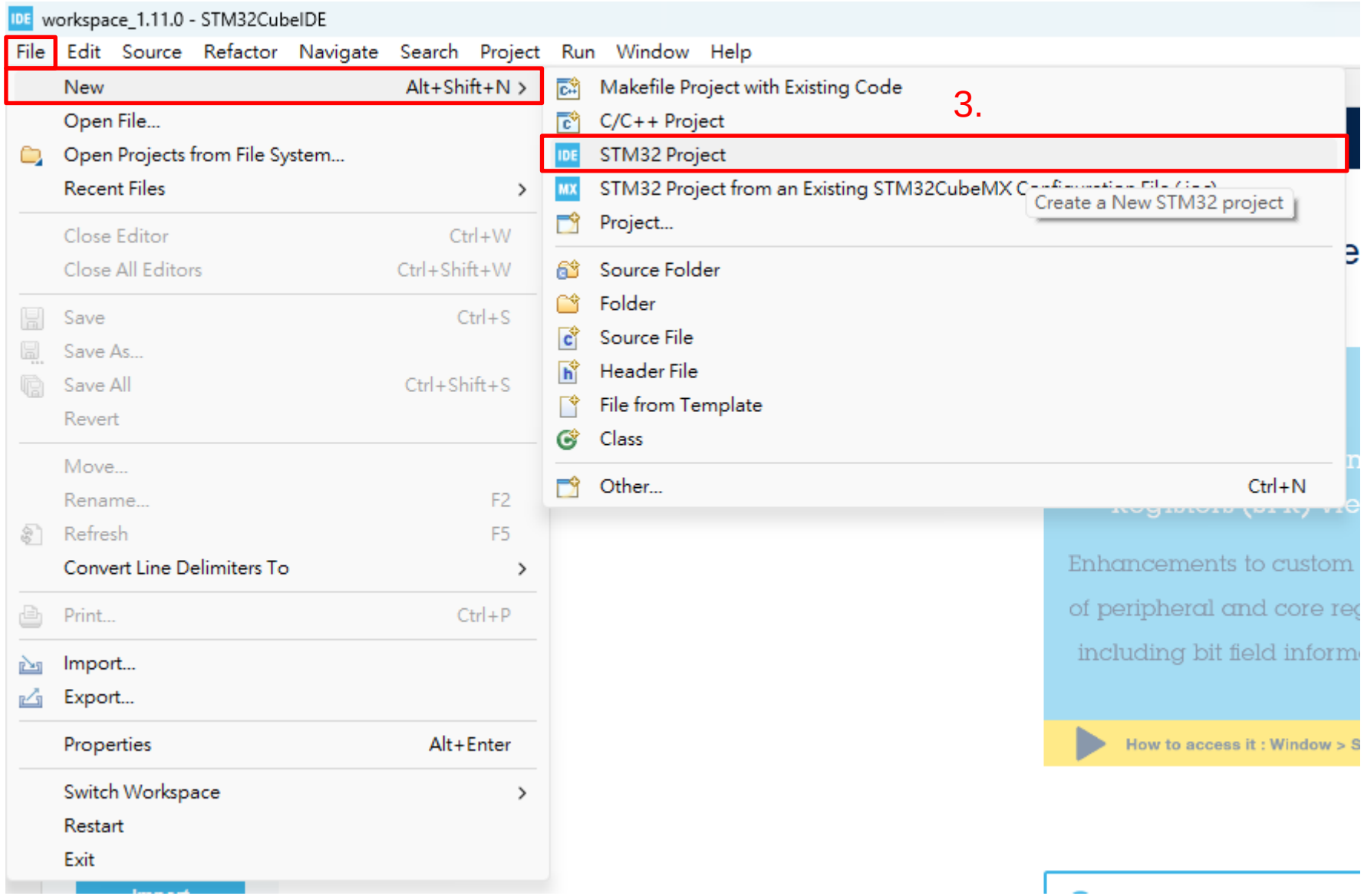


Create STM32 project with STM32CubeIDE (在IDE建立新專案)

登入 myST



Create STM32 project with STM32CubeIDE (在IDE建立新專案)

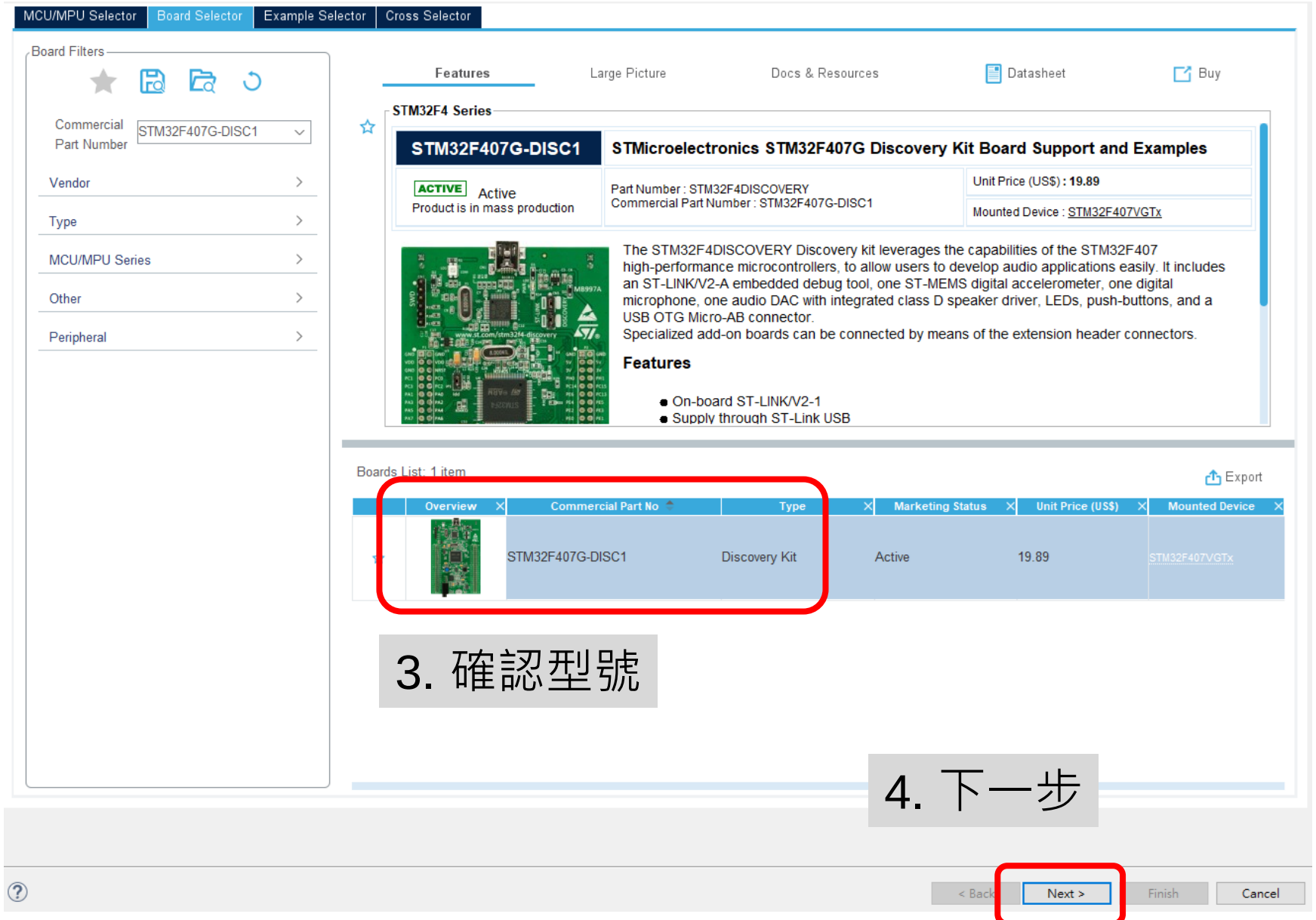


Create STM32 project with STM32CubeIDE (在IDE建立新專案)

1. 輸入 STM32F407VG

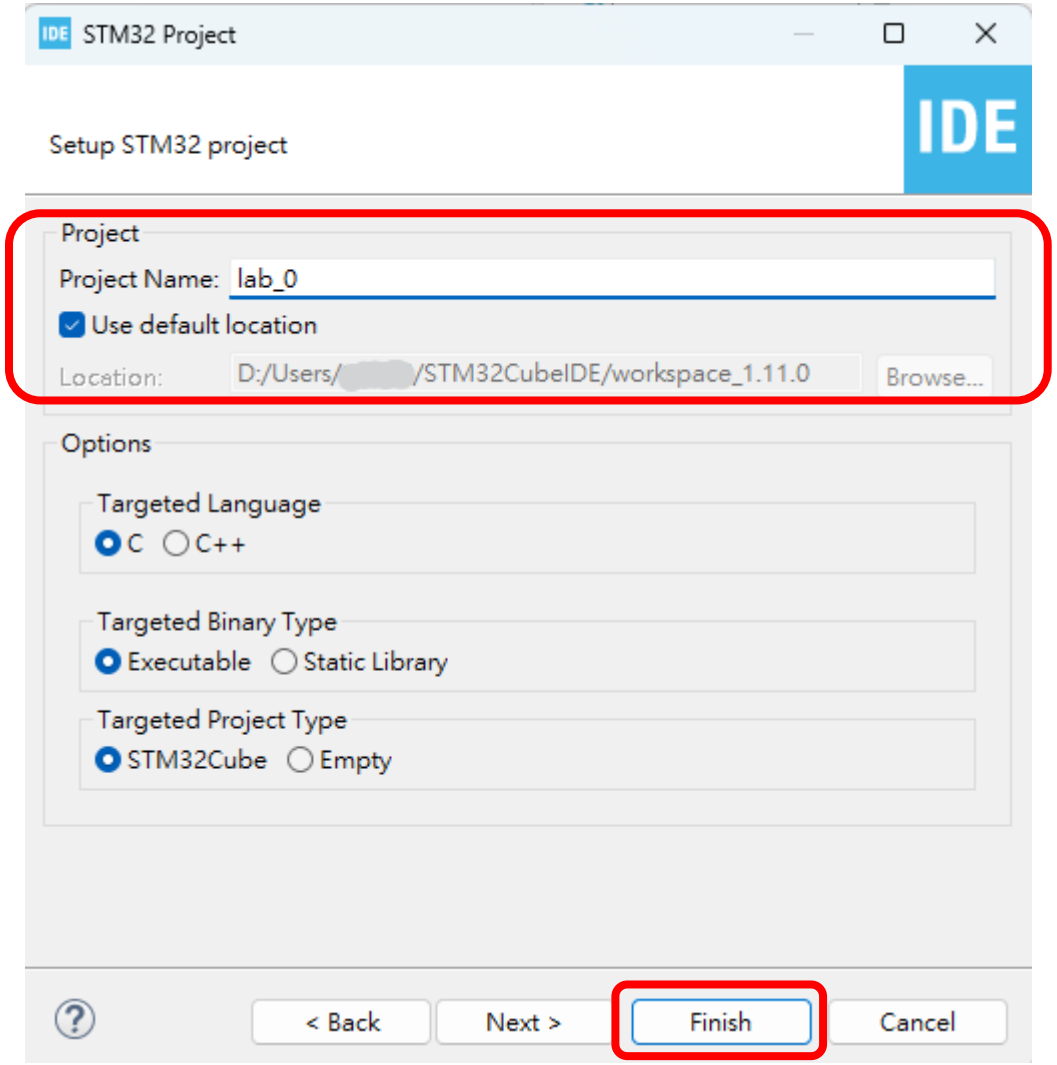
2. 選擇我們的開發版

Create STM32 project with STM32CubeIDE (在IDE建立新專案)

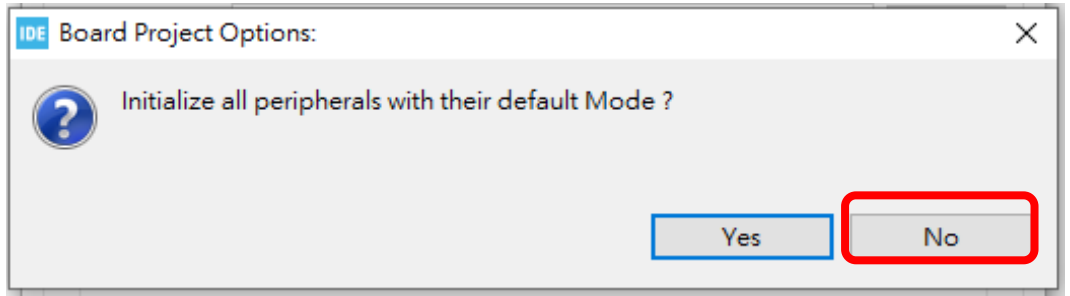


Setup STM32 project

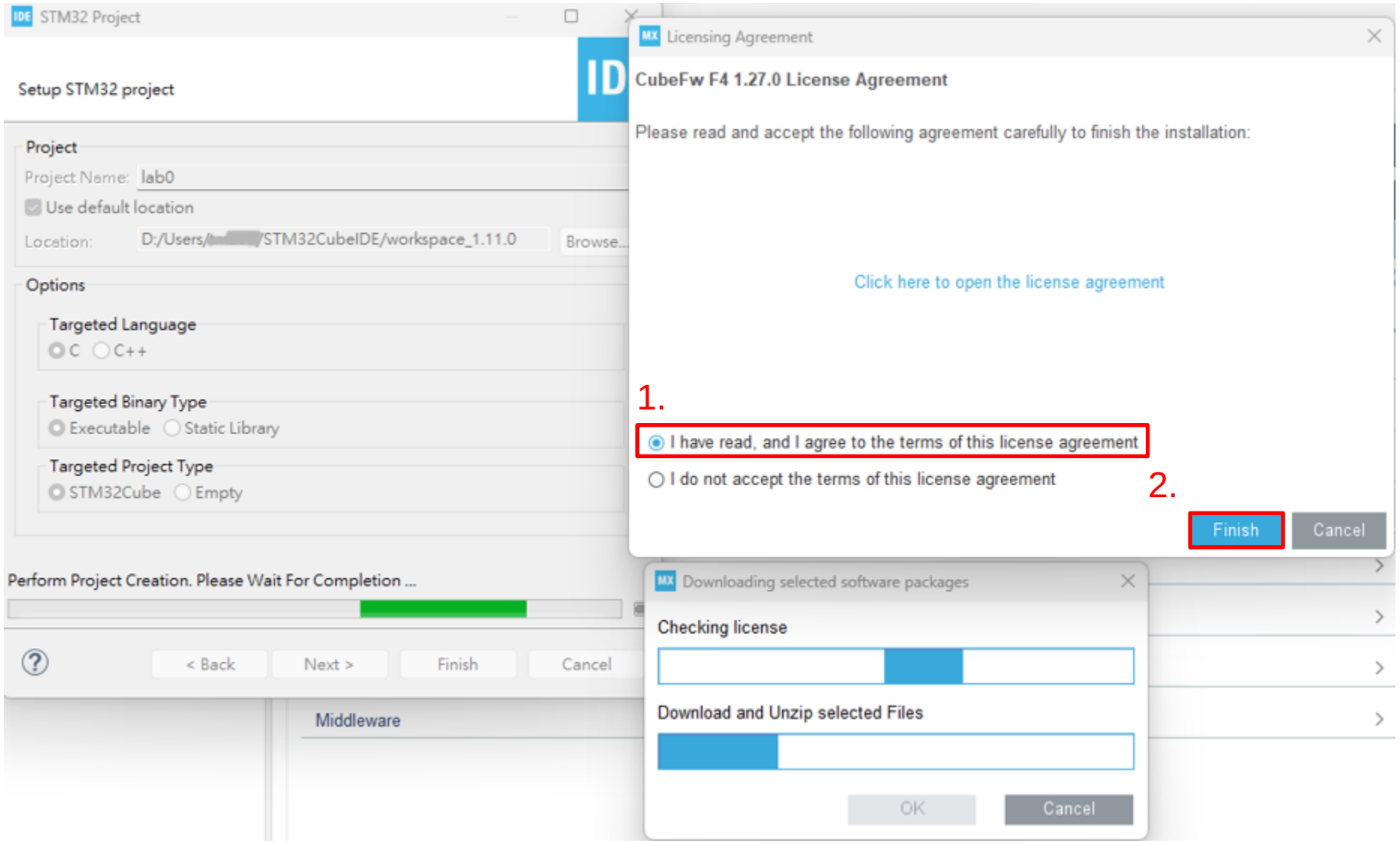
輸入 project name，
選一個不會忘記的location



快速建立 project



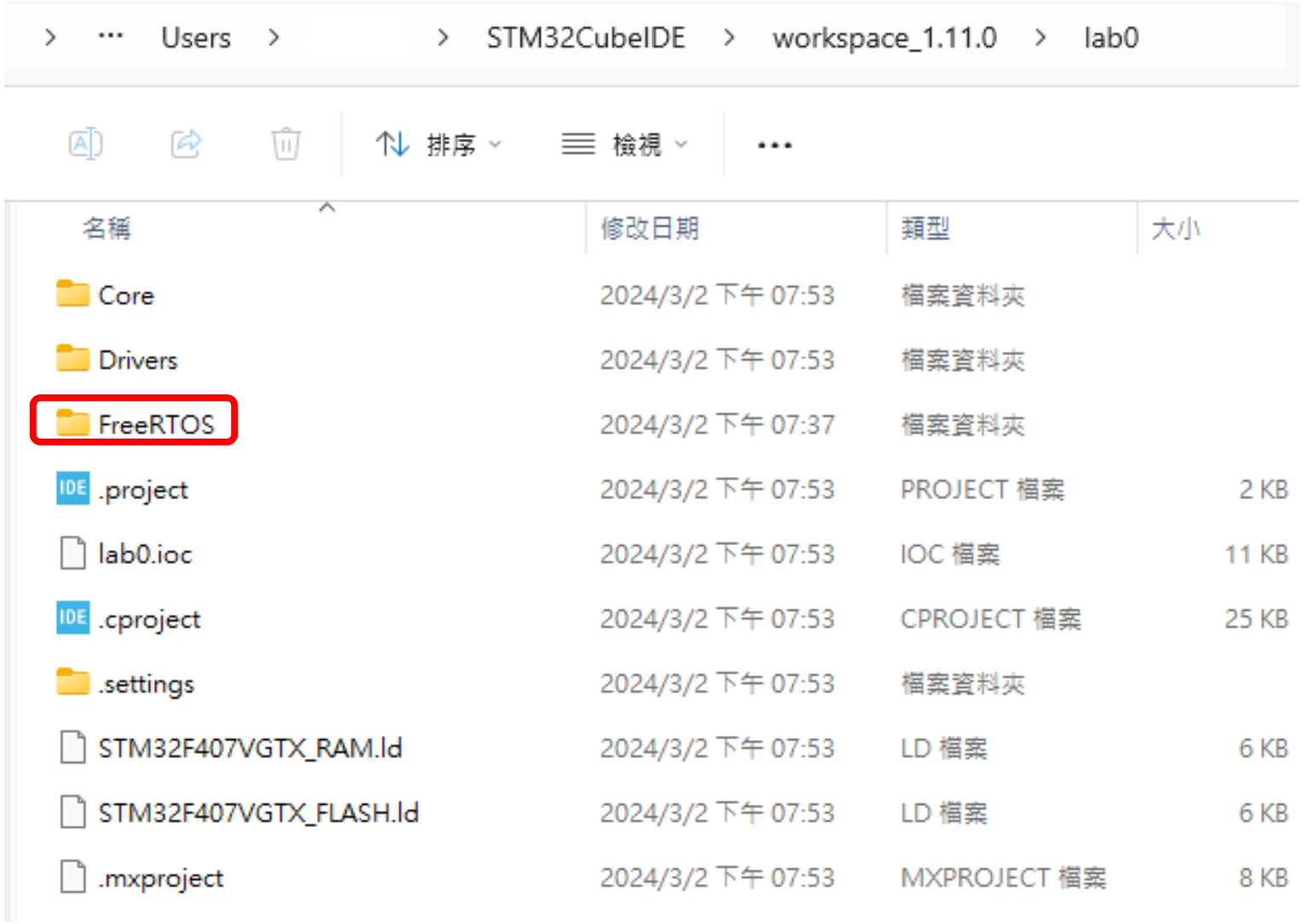
Setup STM32 project



Porting FreeRTOS

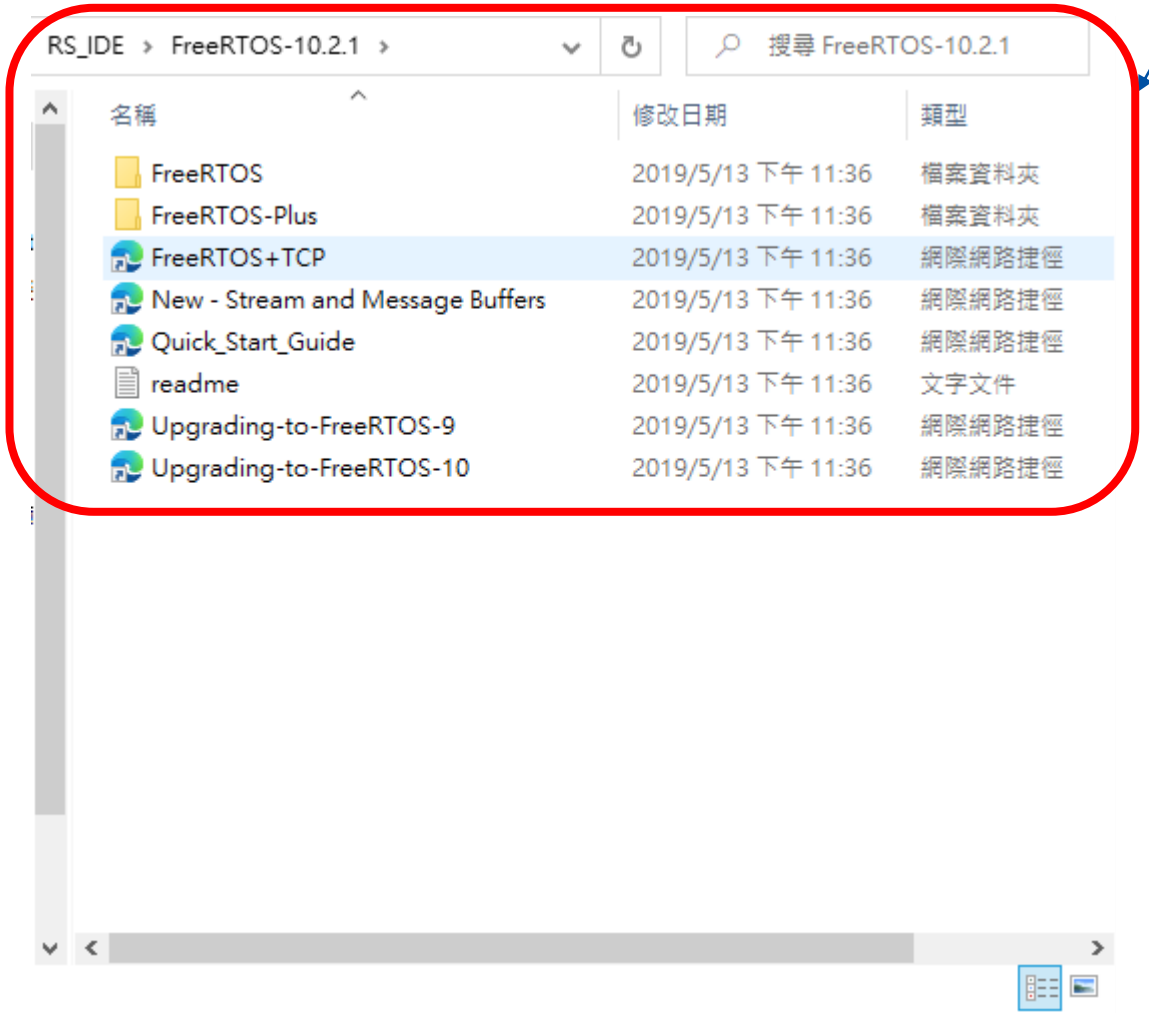
根據你當初建立project的設定，找到你 STM32 project 檔的位置(投影片p.13)

按右鍵
新增一個資料夾，
取名為：FreeRTOS



Porting FreeRTOS 1/2

把之前(投影片p.7)下載的FreeRTOS檔案 解壓縮出來 ，



Porting FreeRTOS 2/2

(非課程指定開發版，需要複製的檔案可能不同)

複製 FreeRTOS 裡面部分的檔案 到 STM32 的專案資料夾(FreeRTOS)
要複製的檔案如下：(藍字是FreeRTOS原始檔案)

➔ FreeRTOS/Source/include (整個 include 資料夾) → 放在專案的 FreeRTOS/
FreeRTOS/Source/*.c (Source內所有的.c 檔) →放在專案的 FreeRTOS/

➔ FreeRTOS/Demo/CORTEX_M4F_STM32F407ZG-SK/FreeRTOSConfig.h
→放在專案的 FreeRTOS/include/

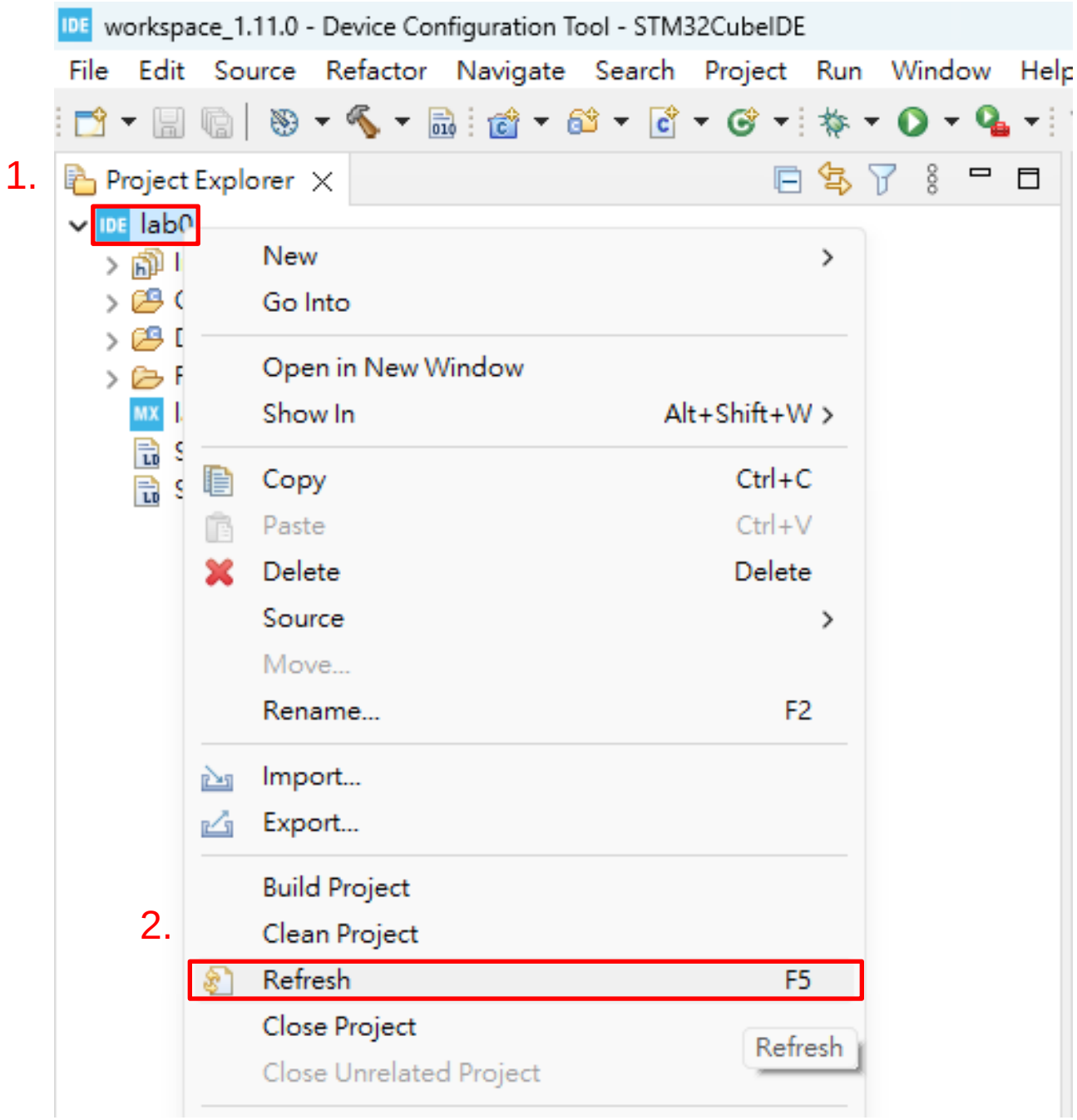
然後在專案的 FreeRTOS/ 新增一個資料夾: portable

➔ FreeRTOS/Source/portable/MemMang/heap_4.c
→在專案的 FreeRTOS/portable/ 新增一個資料夾: MemMang
→在專案的 FreeRTOS/portable/MemMang/， 放heap_4.c

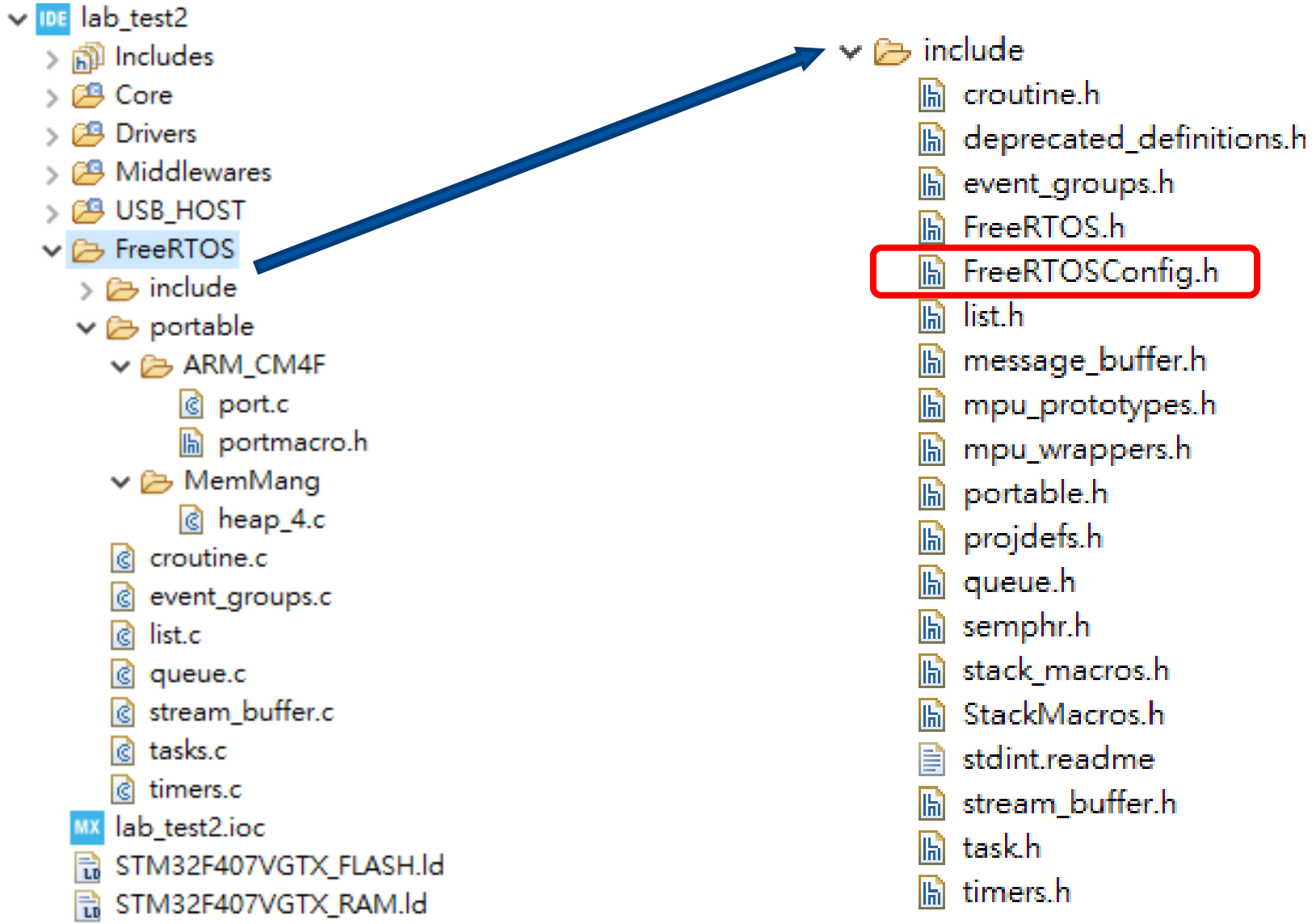
➔ FreeRTOS/Source/portable/GCC/ARM_CM4F (整個 ARM_CM4F 資料夾)
→放在專案的 FreeRTOS/portable/

File Tree

在project檔上按右鍵選 Refresh

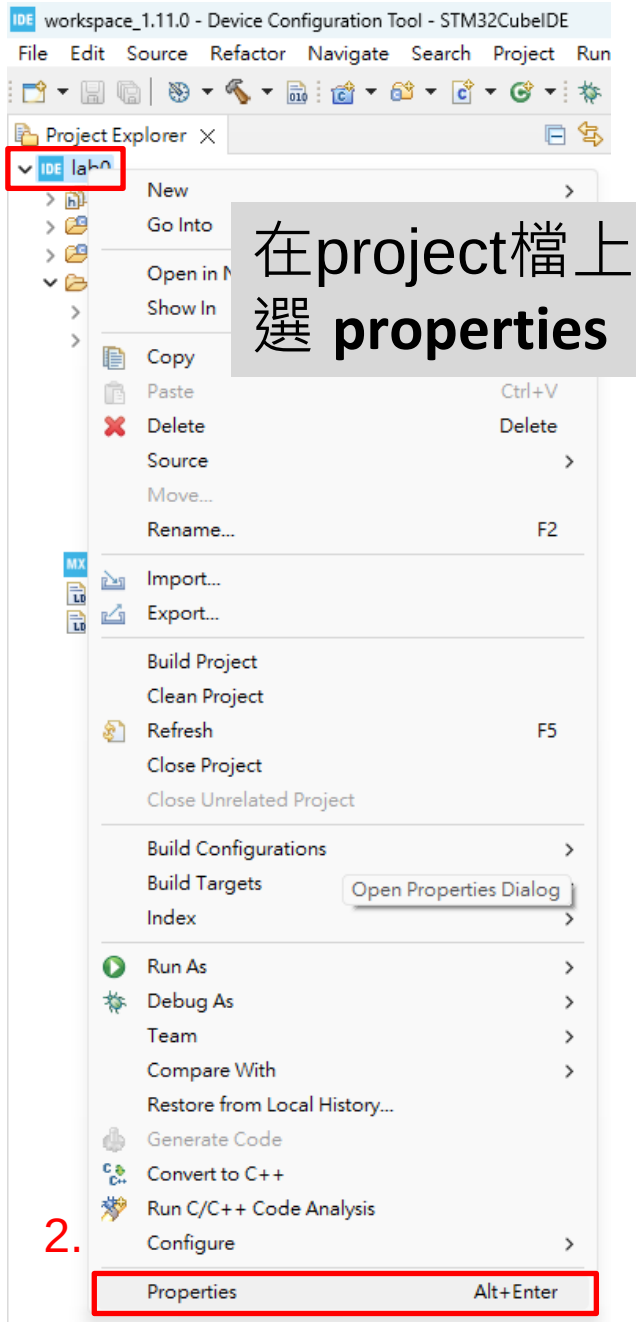


File Tree　看起來應該會像這樣



修改 Paths and Symbols - Source location

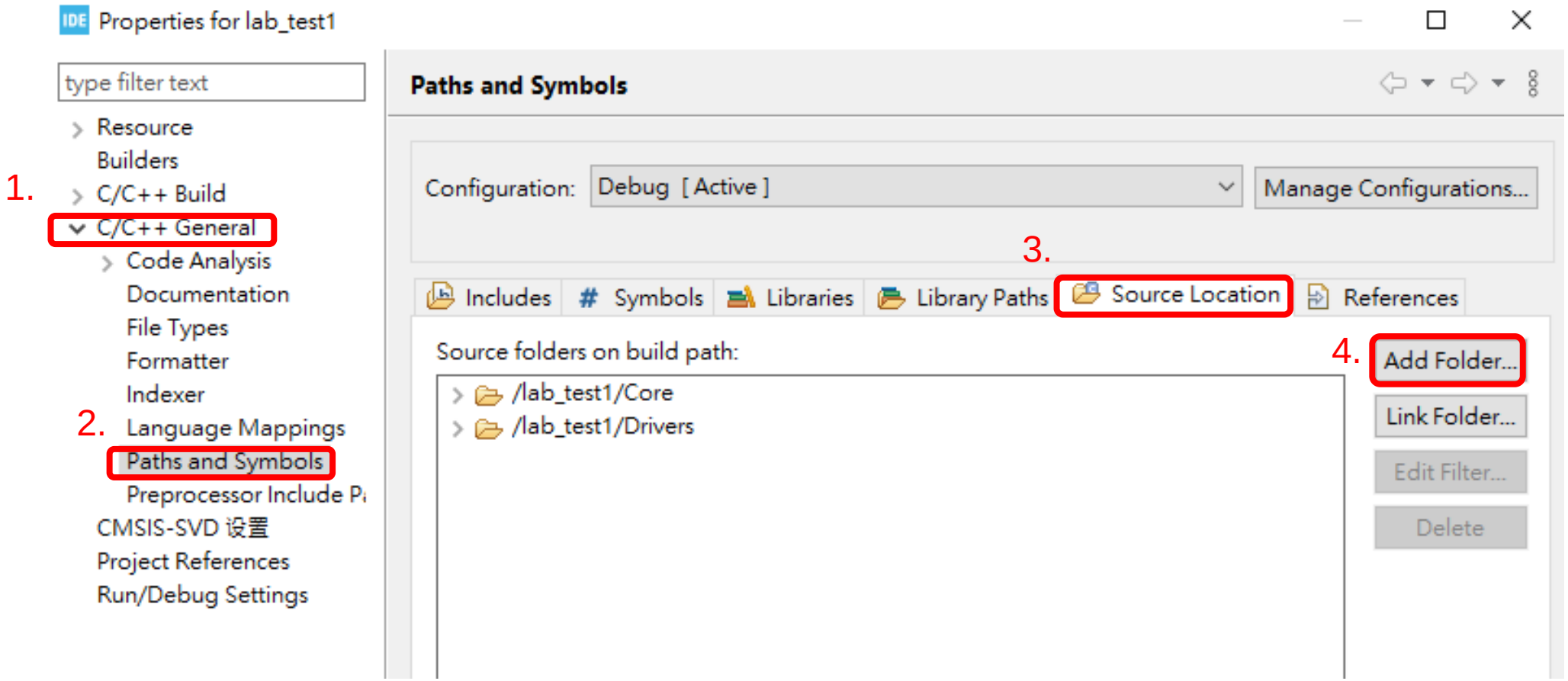
At the top of your project label, right-click **-> properties -> C/C++ General -> Paths and Symbols -> Source Location,** click Add Folder, and add the FreeRTOS folder, as shown in the figure below.



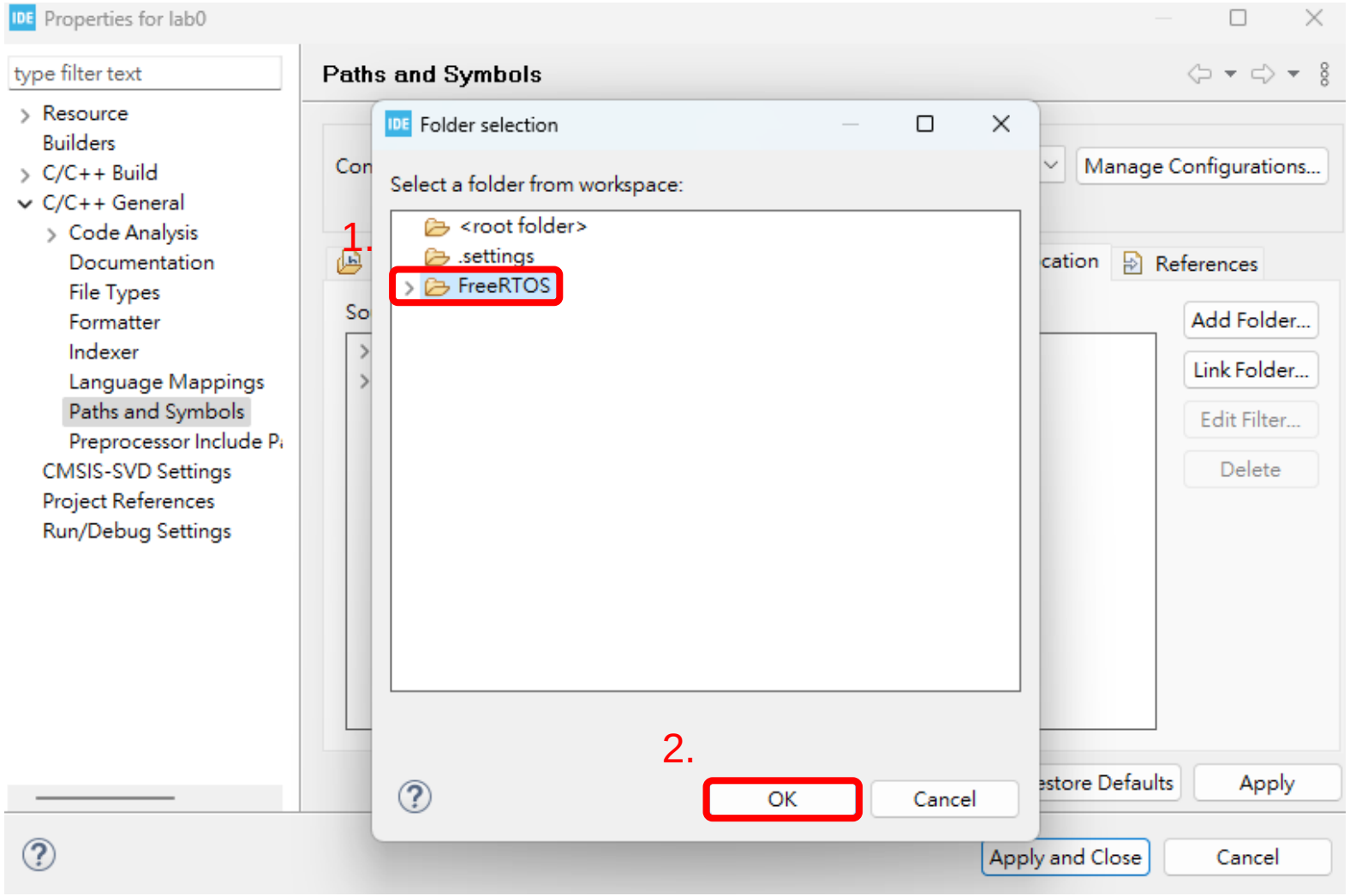
在project檔上按右鍵選 **properties**

修改 Paths and Symbols - Source location

click **C/C++ General -> Paths and Symbols -> Source Location,** click **Add Folder,** and **add the FreeRTOS folder,** as shown in the next slide

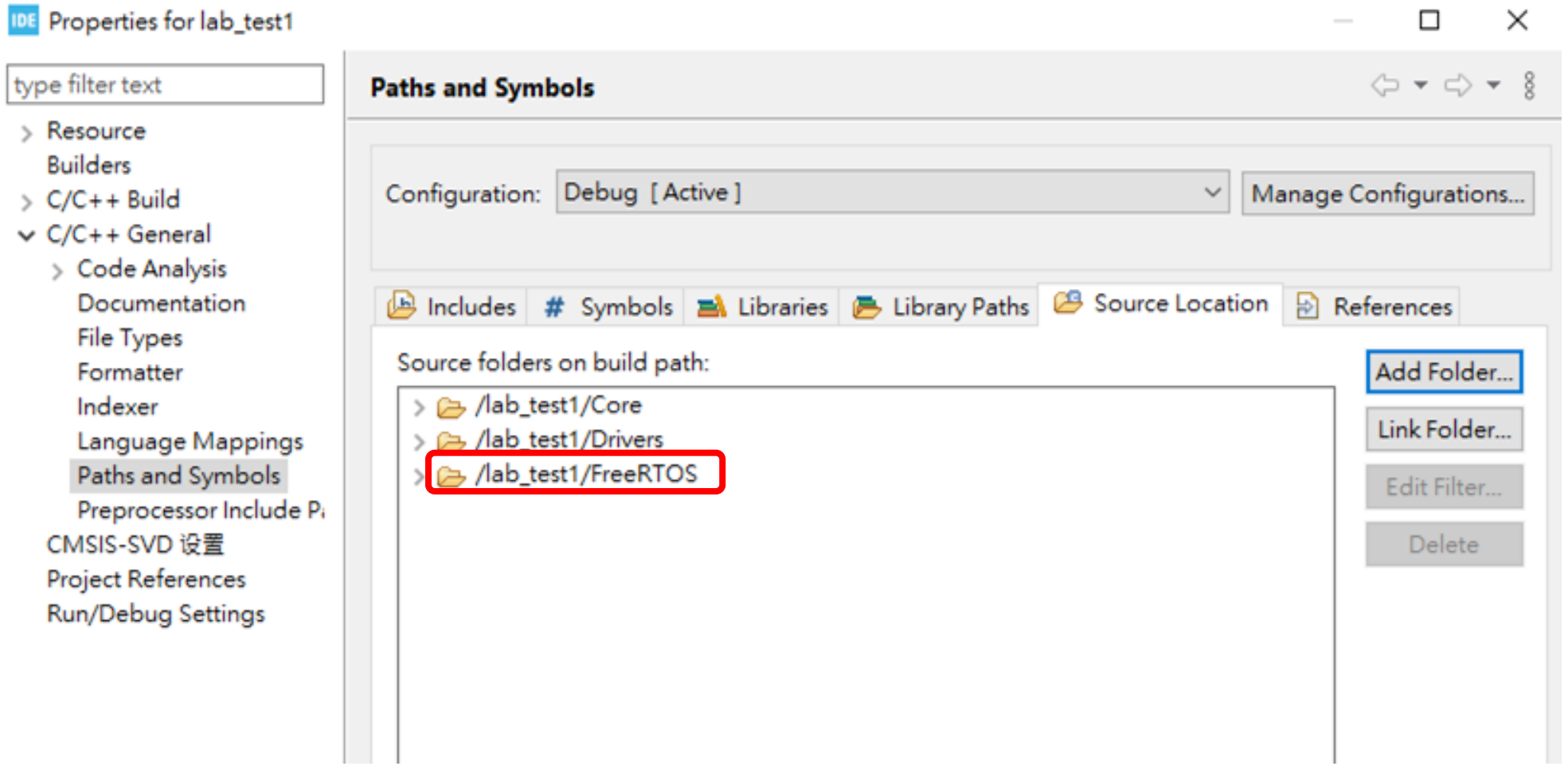


修改 Paths and Symbols - Source location 1/6



修改 Paths and Symbols - Source location 2/6

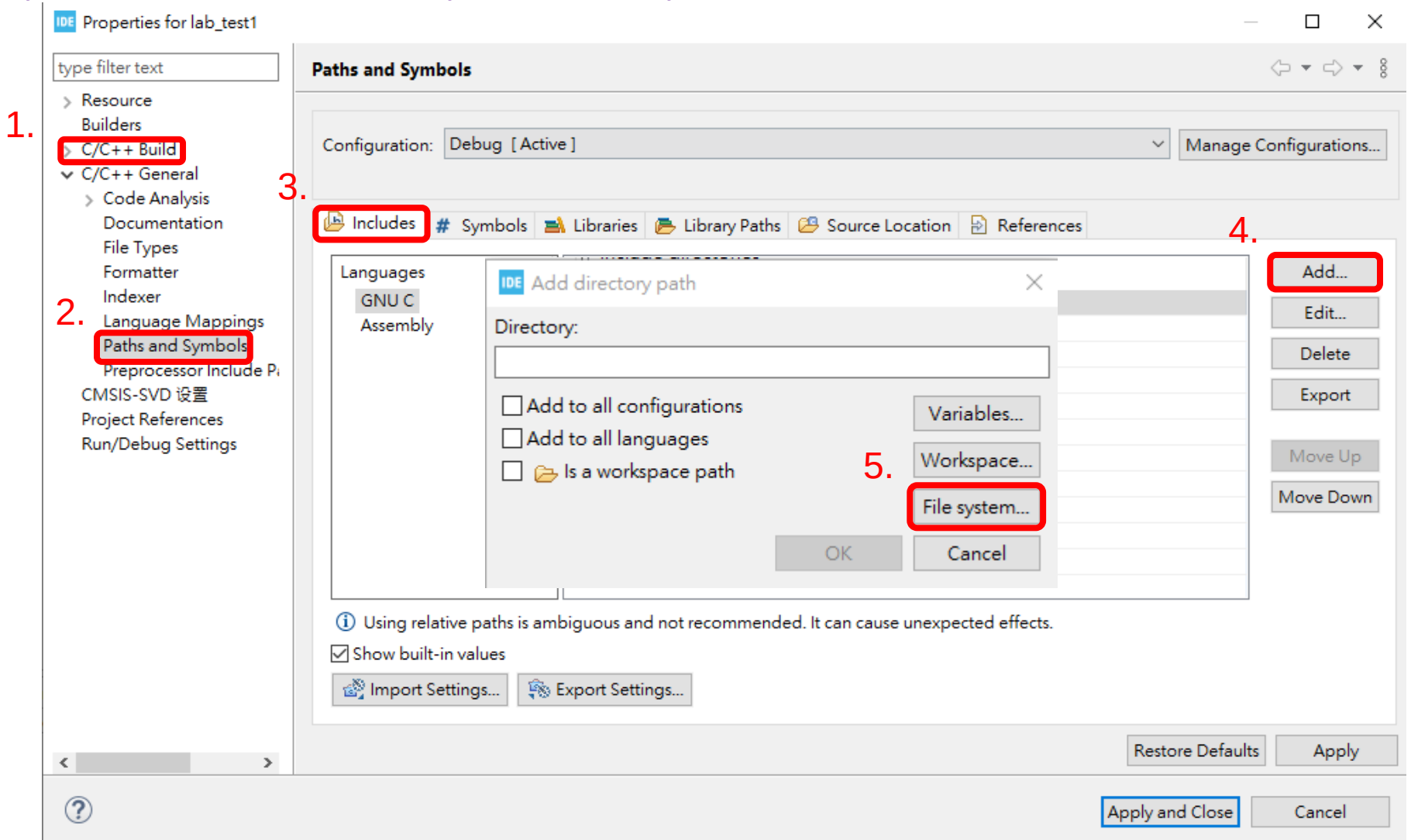
The result :



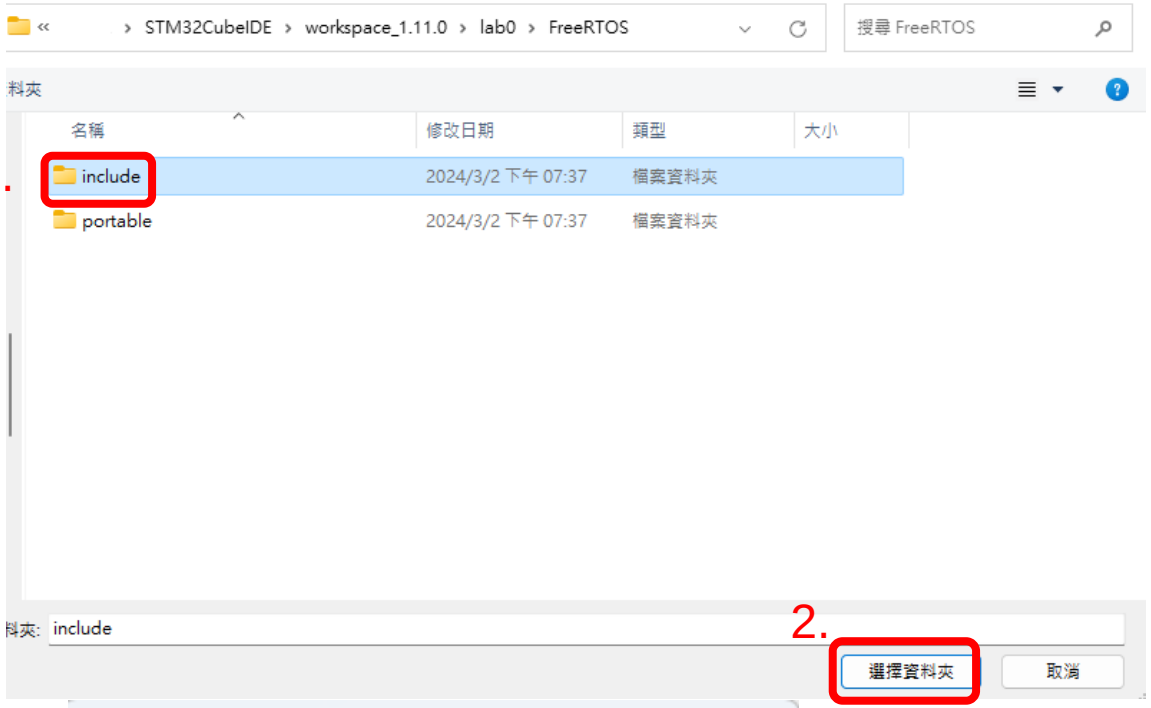
修改 Path and Symbols – Includes 3/6

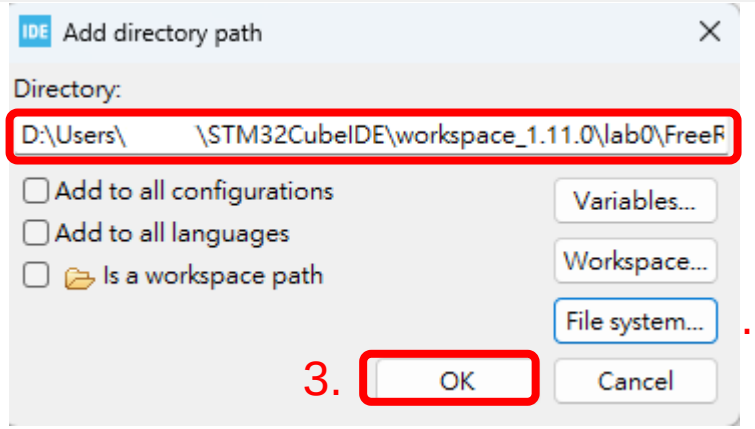
At the same properties window **-> C/C++ General -> Paths and Symbols -> Includes**, click **Add**, and add **include** & **ARM_CM4F** in the FreeRTOS folder.

Note that you need to select "File System" after you click add.

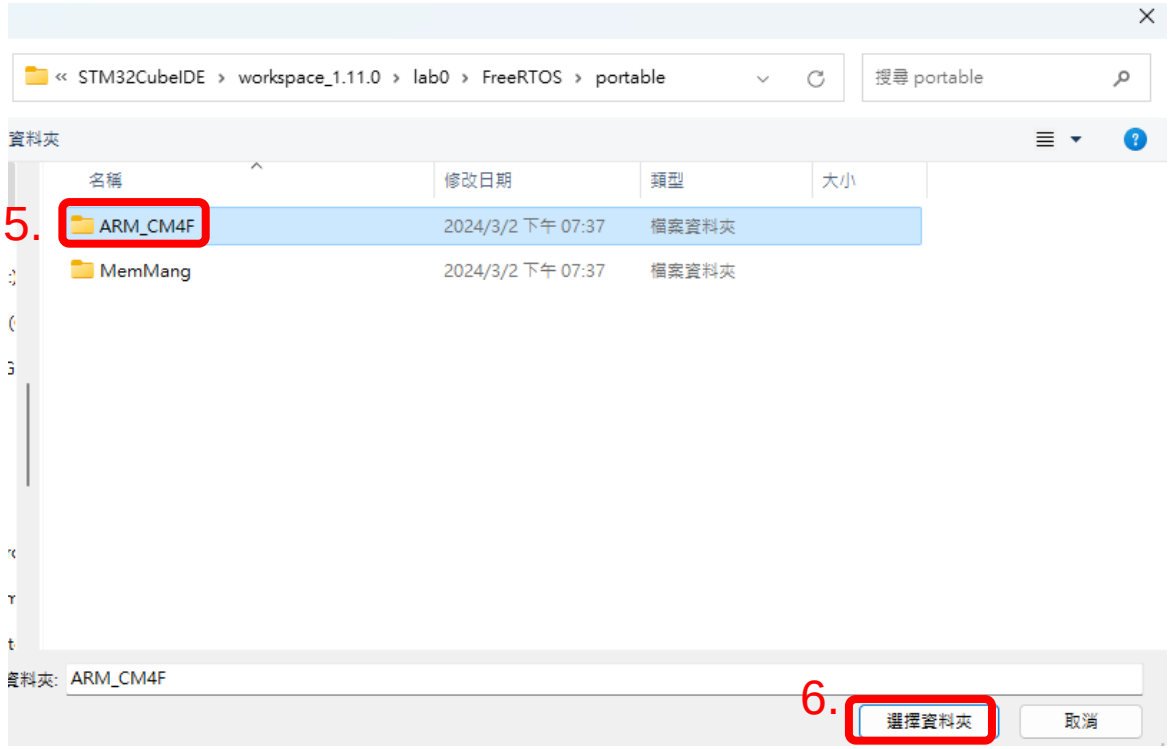


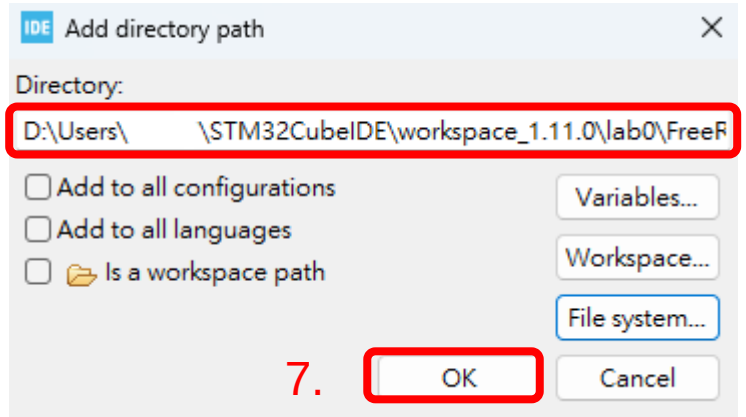
修改 Path and Symbols 4/6





出現 下頁 4

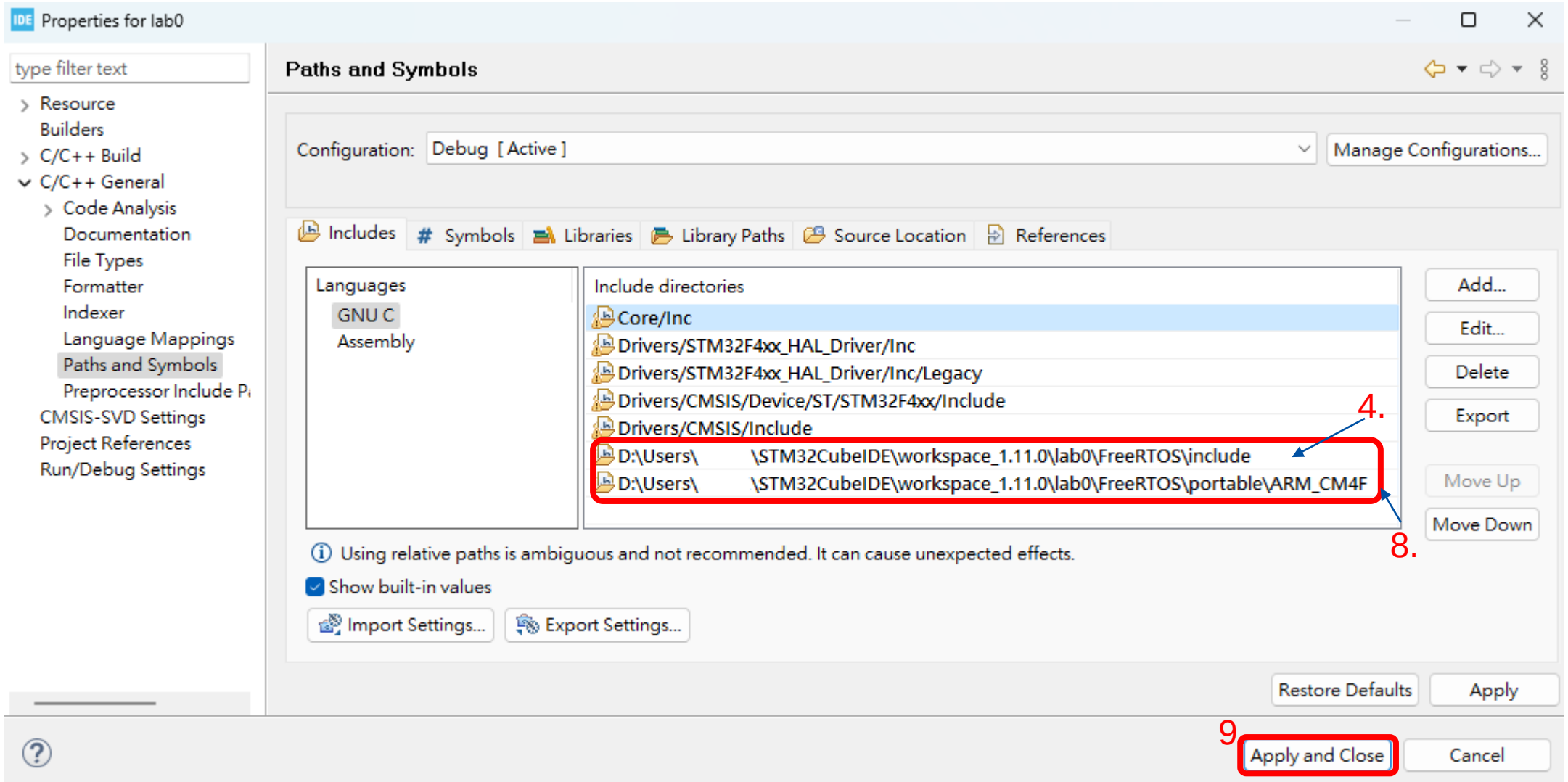




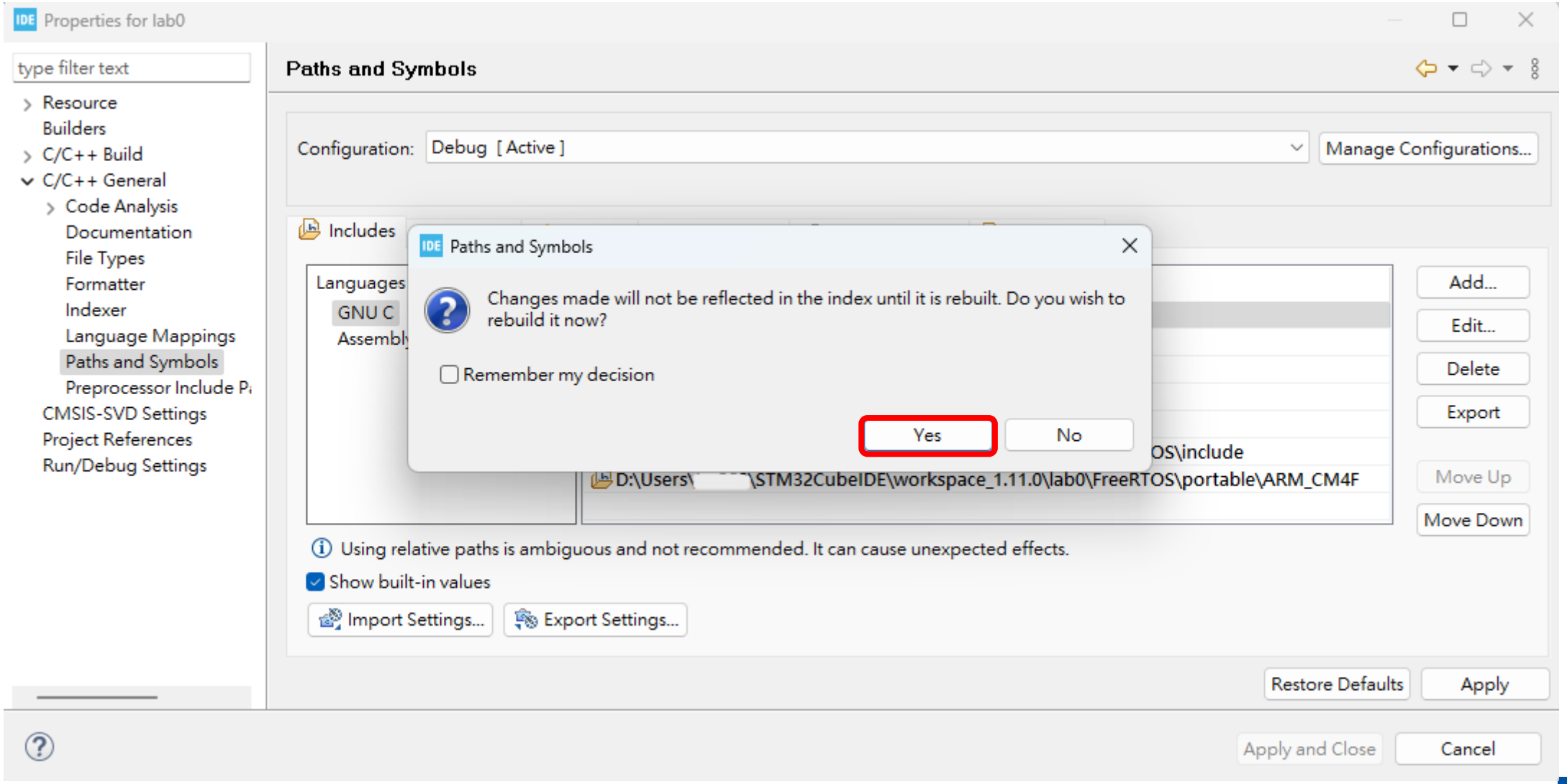
出現 下頁 8

修改 Path and Symbols – Includes 5/6

At the same properties window **-> C/C++ General -> Paths and Symbols -> Includes**,
click **Add**, and add **include** & **ARM_CM4F** in the FreeRTOS folder. (加 include 和 ARM_CM4F 這兩個資料夾)
記得是**找**專案資料夾(投影片p.11)下的。After you add, it will look like the picture below:



修改 Path and Symbols 6/6



到這裡檔案複製與設定的動作就結束了，

接下來是程式碼的修改。

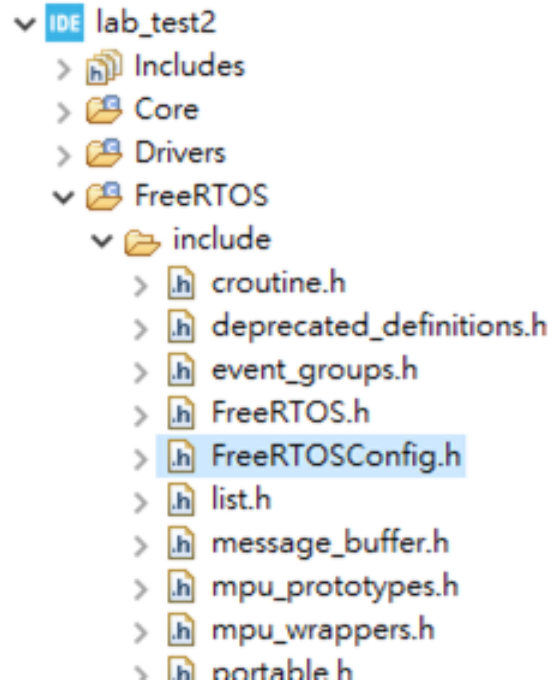
Modify FreeRTOSConfig.h

將 #ifdef __ICCARM__

改成 #if defined(__ICCARM__) || defined(__CC_ARM) || defined(__GNUC__)

或是 #ifdef __GNUC__

```
44 /* Ensure stdint is only used by the compiler, and not the assembler. */
45 #if defined(__ICCARM__) || defined(__CC_ARM) || defined(__GNUC__)
46     #include <stdint.h>
47     extern uint32_t SystemCoreClock;
48 #endif
```



Modify FreeRTOSConfig.h

Change all the following config to 0, otherwise you need to write related HOOK function.

```
#define configUSE_IDLE_HOOK                     1
#define configUSE_TICK_HOOK                     1
#define configUSE_MALLOC_FAILED_HOOK  1
#define configCHECK_FOR_STACK_OVERFLOW  2
```

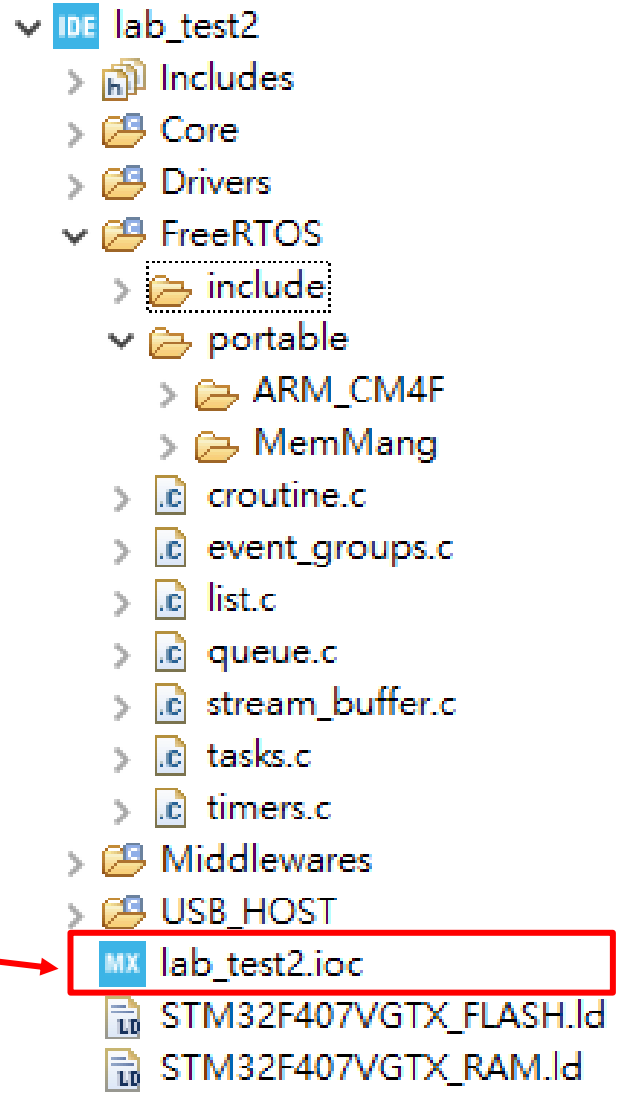
```
#define configUSE_IDLE_HOOK                     0
#define configUSE_TICK_HOOK                     0
#define configUSE_MALLOC_FAILED_HOOK  0
#define configCHECK_FOR_STACK_OVERFLOW  0
```

Modify FreeRTOSConfig.h

After you change.

```
50  #define configUSE_PREEMPTION            1
51  #define configUSE_IDLE_HOOK             0
52  #define configUSE_TICK_HOOK             0
53  #define configCPU_CLOCK_HZ              ( SystemCoreClock )
54  #define configTICK_RATE_HZ              ( ( TickType_t ) 1000 )
55  #define configMAX_PRIORITIES            ( 5 )
56  #define configMINIMAL_STACK_SIZE        ( ( unsigned short ) 130 )
57  #define configTOTAL_HEAP_SIZE           ( ( size_t ) ( 75 * 1024 ) )
58  #define configMAX_TASK_NAME_LEN         ( 10 )
59  #define configUSE_TRACE_FACILITY        1
60  #define configUSE_16_BIT_TICKS          0
61  #define configIDLE_SHOULD_YIELD         1
62  #define configUSE_MUTEXES               1
63  #define configQUEUE_REGISTRY_SIZE       8
64  #define configCHECK_FOR_STACK_OVERFLOW  0
65  #define configUSE_RECURSIVE_MUTEXES     1
66  #define configUSE_MALLOC_FAILED_HOOK    0
67  #define configUSE_APPLICATION_TASK_TAG  0
68  #define configUSE_COUNTING_SEMAPHORES   1
69  #define configGENERATE_RUN_TIME_STATS   0
```

修改 Basic Timer

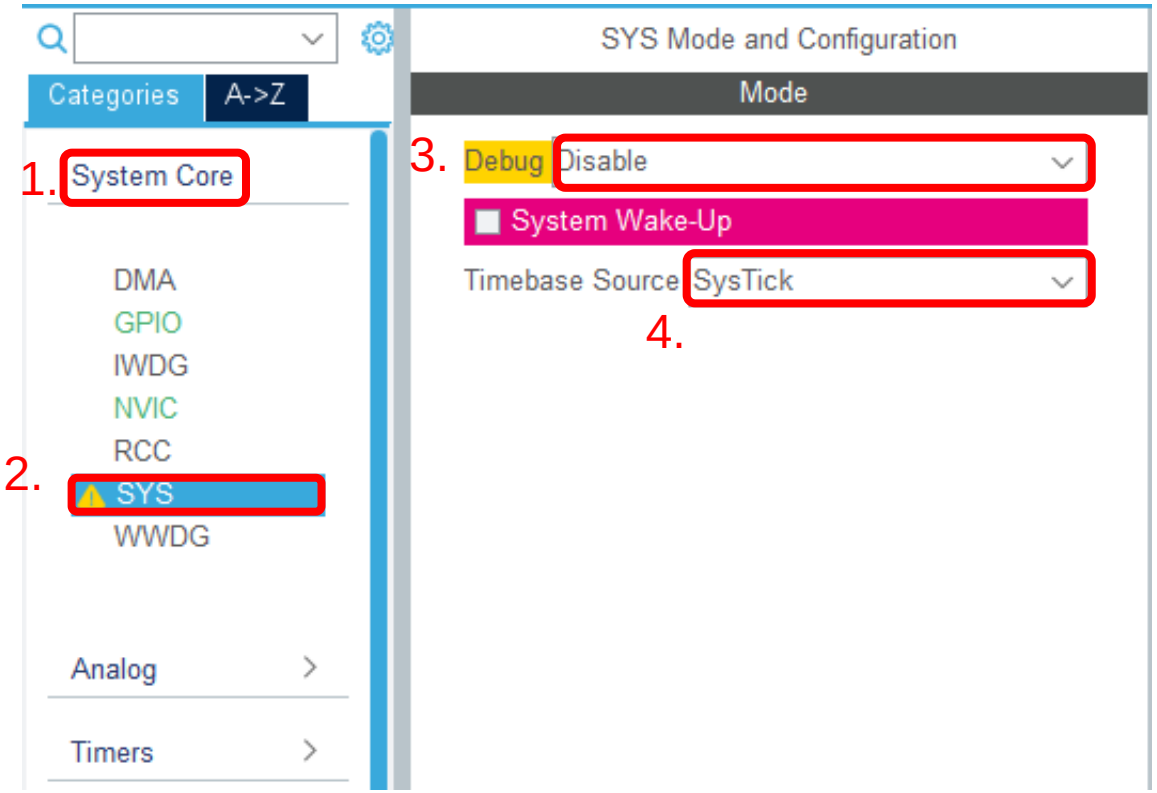


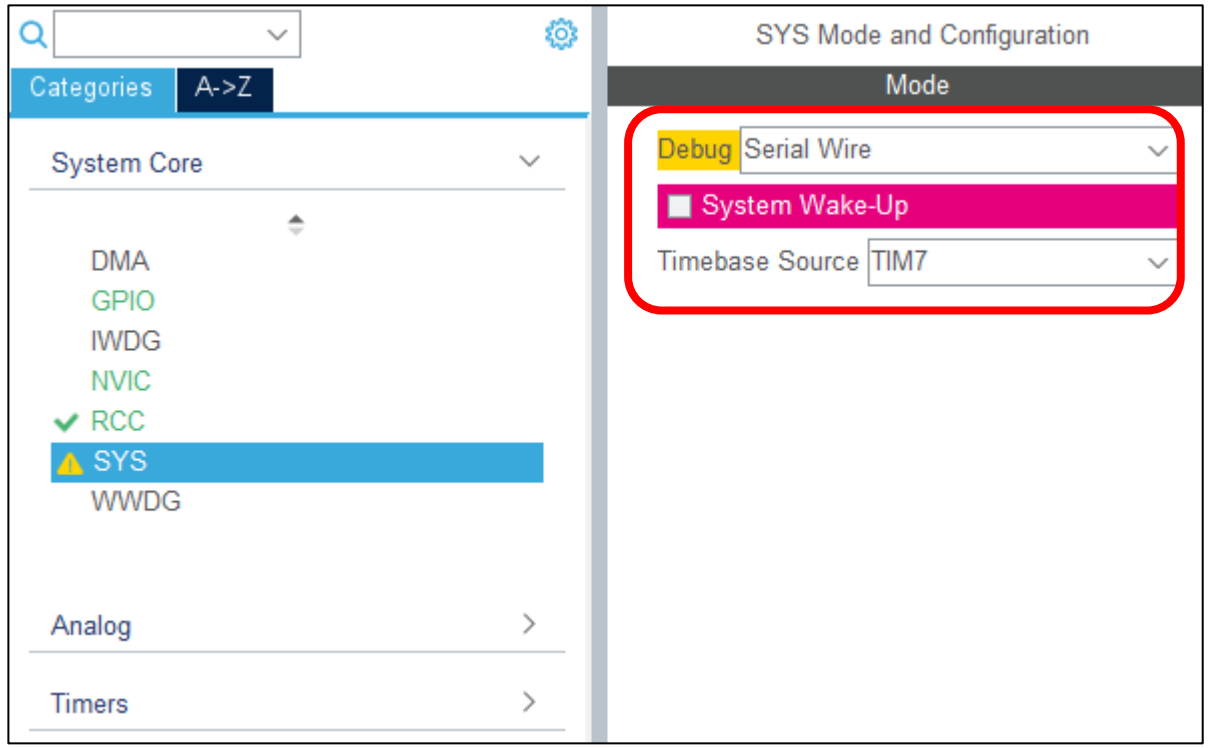
Click “project name.ioc”
(按下去)

修改 Basic Timer

System Core -> **SYS**，Change **Timebase Source** to Tim6 or **Tim7**.

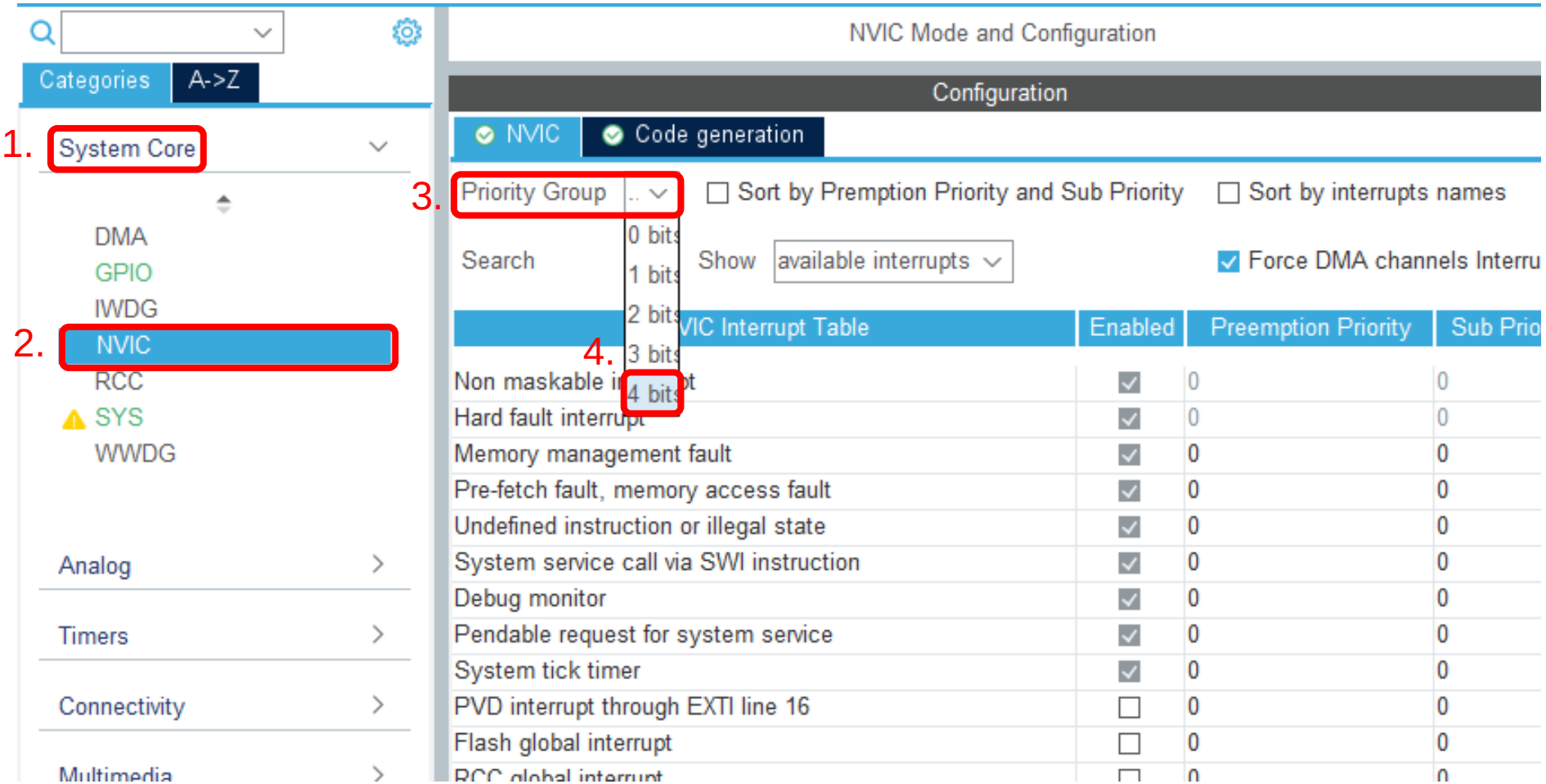
Note that **Debug** is set to **Serial wire**.



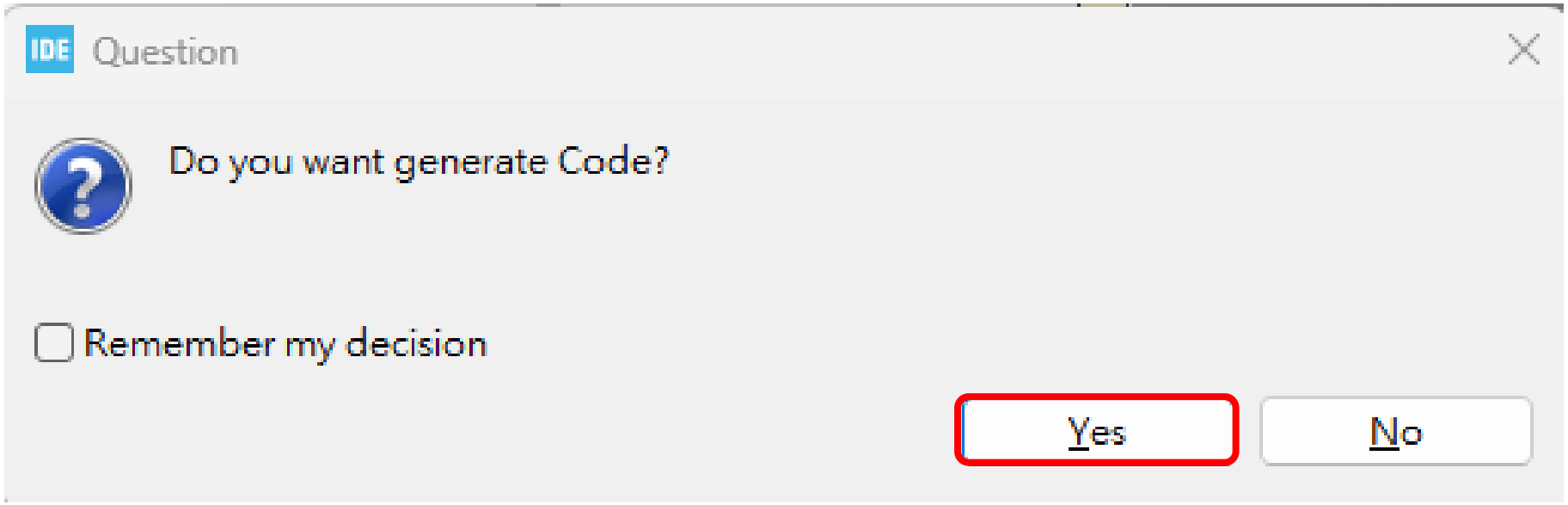


修改 NVIC

System Core -> **NVIC**，Change **Priority Group** to **4 bits**. , then ctrl + S (存檔)



Overwriting problem



注意：每次修改 .ioc 檔, Ctrl + S 後,
你的某些code會被覆蓋掉
Please see the next page to avoid overwriting.

Overwriting problem

/* USER CODE BEGIN ... */

your code; (在上下這種類型註解之間的code，不會被複寫重置。除此之外
(BEGIN 和 END 之間))

/* USER CODE END ... */

```
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */
```

```
/* USER CODE BEGIN 2 */
//Queue1Handle = xQueueCreate(10,sizeof(int));
xTaskCreate(LEDTask_App,"LEDTask",128,(void *) NULL,0,NULL);
xTaskCreate(LBtnTask_App,"LBtnTask",128,(void *) NULL,0,NULL);

vTaskStartScheduler();
/* USER CODE END 2 */

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

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

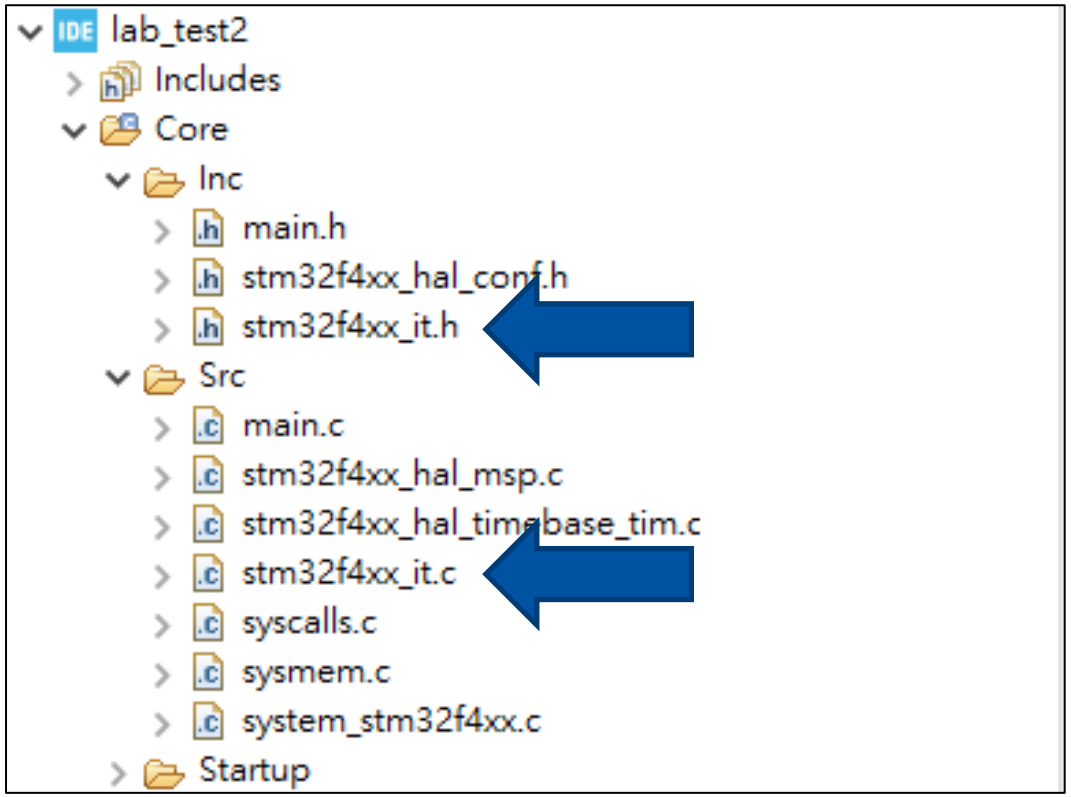
Use FreeRTOS interrupt instead

我們要註解掉某些 handler，避免和 FreeRTOS的衝突

In **stm32f4xx_it.c (裡面有定義)** and **stm32f4xx_it.h (裡面有宣告)**

- PendSV_Handler
- SVC_Handler
- SysTick_Handler

上面三個都要註解掉



Use FreeRTOS interrupt instead

我們要註解掉某些 handler，避免和 FreeRTOS的衝突，

以 SysTick_Handler 為例：

stm32f4xx_it.c

```
183   * @brief This function handles System tick timer.
184   */
185 //void SysTick_Handler(void)
186 //{
187 //  /* USER CODE BEGIN SysTick_IRQn 0 */
188 //
189 //  /* USER CODE END SysTick_IRQn 0 */
190 //
191 //  /* USER CODE BEGIN SysTick_IRQn 1 */
192 //
193 //  /* USER CODE END SysTick_IRQn 1 */
194 //}
```

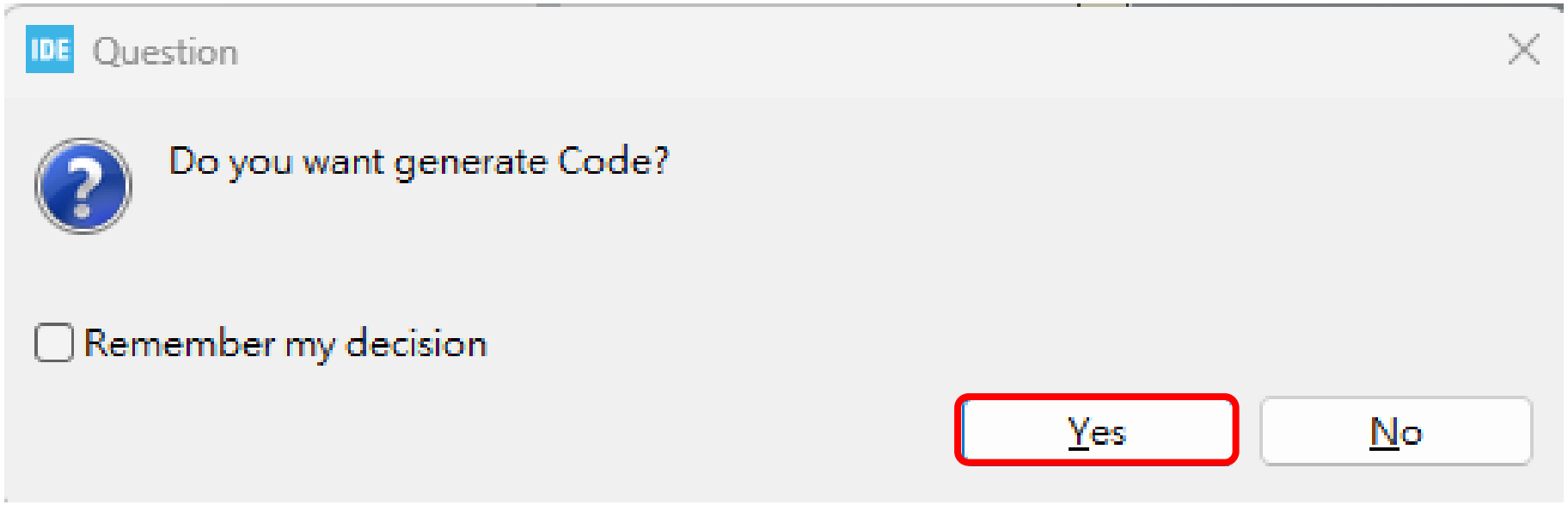
stm32f4xx_it.h

```
57 //void SysTick_Handler(void);
58 void TIM7_IRQHandler(void);
59 void OTG_FS_IRQHandler(void);
```

沒註解 會噴的 error messages

```
c:\st\stm32cubeide_1.8.0\stm32cubeide\plugins\com.st.stm32cube.ide.mcu.externaltools.gnu-tools-for-stm32.9-2020-q2-update.win32_2.0.0.20210531134
C:/Users/crlin/STM32CubeIDE/workspace_1.8.0/lab_test1/Debug/../FreeRTOS/portable/ARM_CM4F/port.c:244: multiple definition of `SVC_Handler'; ./Cor
c:\st\stm32cubeide_1.8.0\stm32cubeide\plugins\com.st.stm32cube.ide.mcu.externaltools.gnu-tools-for-stm32.9-2020-q2-update.win32_2.0.0.20210531134
C:/Users/crlin/STM32CubeIDE/workspace_1.8.0/lab_test1/Debug/../FreeRTOS/portable/ARM_CM4F/port.c:435: multiple definition of `PendSV_Handler'; ./
c:\st\stm32cubeide_1.8.0\stm32cubeide\plugins\com.st.stm32cube.ide.mcu.externaltools.gnu-tools-for-stm32.9-2020-q2-update.win32_2.0.0.20210531134
C:/Users/crlin/STM32CubeIDE/workspace_1.8.0/lab_test1/Debug/../FreeRTOS/portable/ARM_CM4F/port.c:489: multiple definition of `SysTick_Handler'; .
```

Note! 每次你重新gen code (例如修改 .ioc 檔後存檔)，那些handler又會覆寫

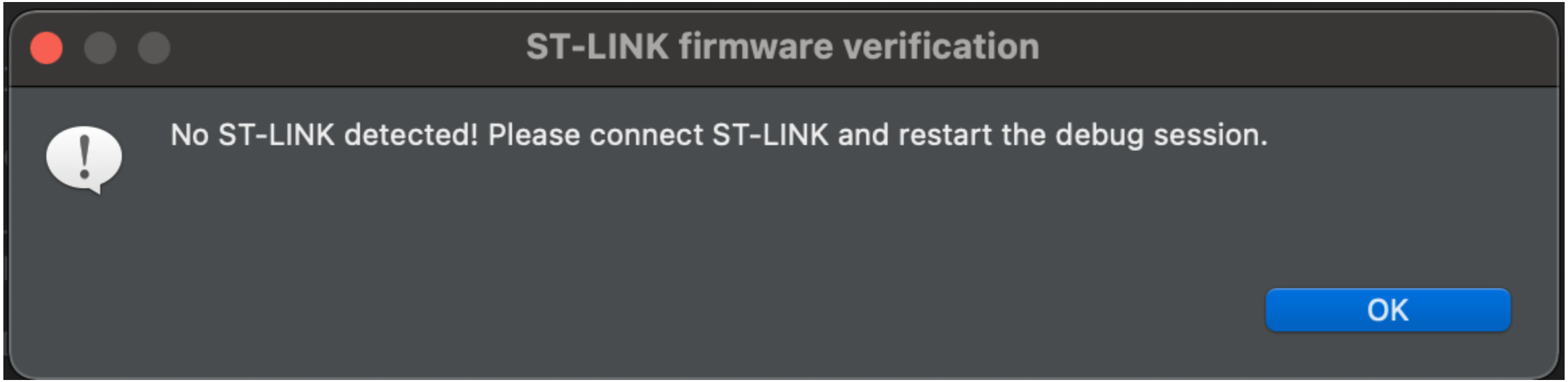


記得再去那兩個檔案，重新把這三個 handler 註解掉:

- PendSV_Handler
- SVC_Handler
- SysTick_Handler

(stm32f4xx_it.c ， stm32f4xx_it.h)

如果你遇到下圖的問題，可能是USB孔沒插好，或是driver沒裝好



Driver 連結:

ST-LINK, ST-LINK/V2, ST-LINK/V2-1, STLINK-V3 boards firmware upgrade.

https://www.st.com/en/development-tools/stsw-link007.html

測試 Porting 是否完成

如果你上面的步驟都做完了，可以測測看是否完成。

1. 在 main.c 內補 2 個 header

```
/* USER CODE END Header */
/* Includes ------------------
#include "main.h"
#include "FreeRTOS.h"
#include "task.h"
```

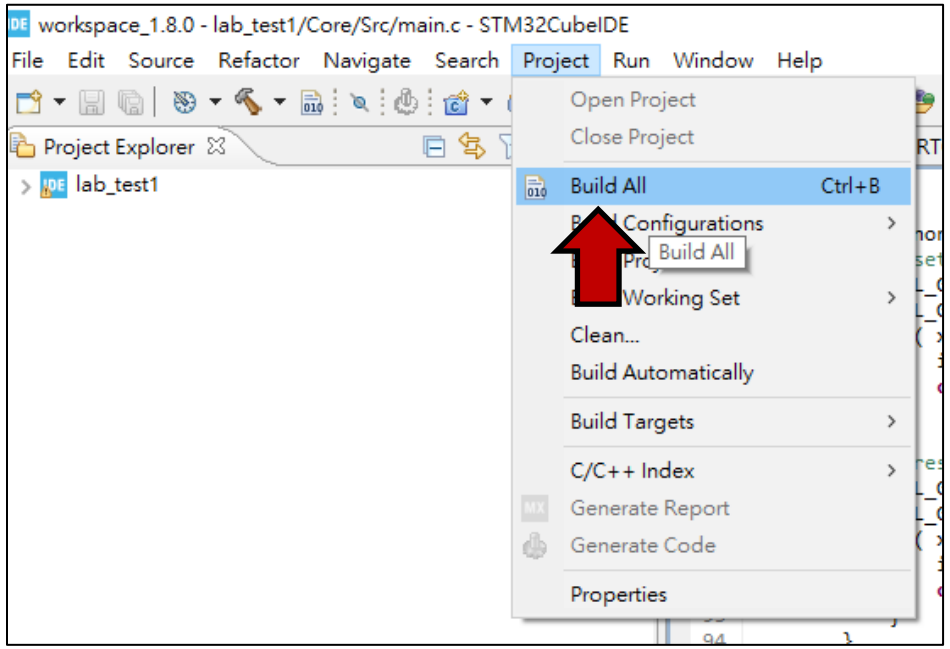
2.

在 main.c 內的 main() 函式最後面的 while 迴圈上面，補 vTaskStartScheduler();

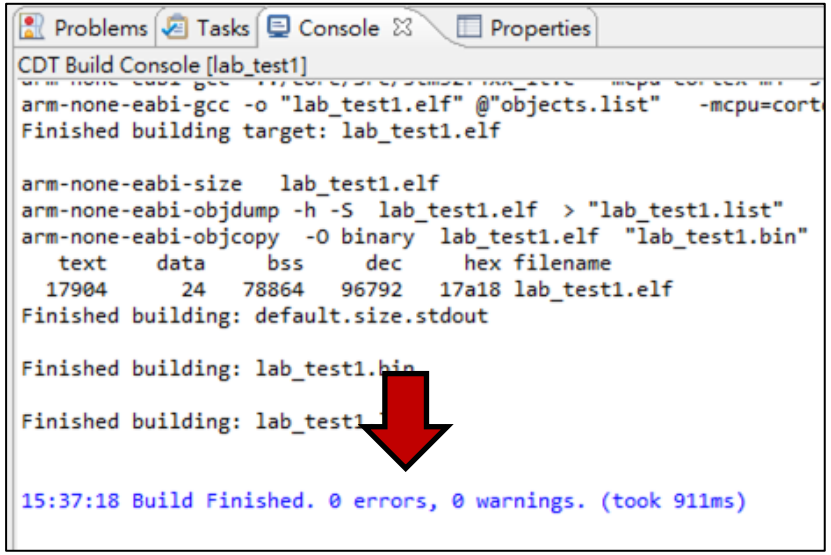
```
  /* USER CODE BEGIN 2 */
  vTaskStartScheduler();
  /* USER CODE END 2 */

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

3. Build Project



4. 確認是否有 error message



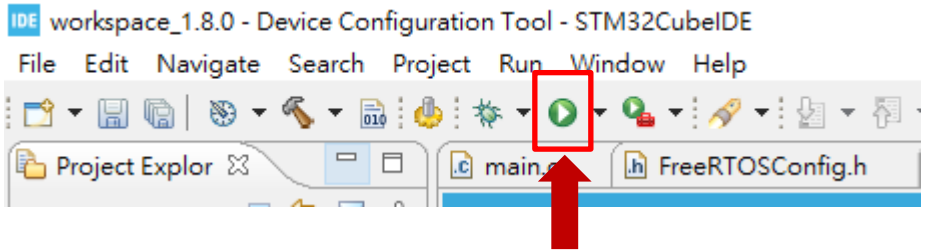
(Optional) 測試 開發板 是否連接正確 (可以不用做)

如果你已經拿到你的的開發版，並且用 mini-USB 連接上電腦，
我們提供了一個簡單的測試檔 (下面連結內是一個程式檔(main.c))，
使用方法是把你原本專案下的 Core/Src/main.c 先另外找個地方存起來，
然後 Core/Src/ 下換成這個 main.c。

https://drive.google.com/file/d/1b3Uak-bNEkV6UnHfl_gnk_7wPGdkn3vn/view

正確執行的話，會看到板子的綠燈在閃爍。

你也可以動手寫一些程式自己玩玩看。



右邊這4格
預設應該
是綠色的

之後的 Lab 可能會用到的參考資料: (建議先下載 前2個)

- User manual (開發板使用說明書):

Discovery kit with STM32F407VG MCU - User manual
https://www.st.com/resource/en/user_manual/dm00039084-discovery-kit-with-stm32f407vg-mcu-stmicroelectronics.pdf

- Reference manual (開發板硬體 SPEC 規格書):

STM32F405/415, STM32F407/417, STM32F427/437 and STM32F429/439 advanced Arm®-based 32-bit MCUs - Reference manual
https://www.st.com/resource/en/reference_manual/dm00031020-stm32f405-415-stm32f407-417-stm32f427-437-and-stm32f429-439-advanced-arm-based-32-bit-mcus-stmicroelectronics.pdf

- FreeRTOS API Reference (如何使用 FreeRTOS 提供的 API):

FreeRTOS API Reference
https://www.freertos.org/a00106.html