

ECM-XF/ECM-XFU範例程式使用導引

訊成科技股份有限公司

V.06
2022.12.25

ECM-XF範例列表	3
ECM-XFU-SK / ECM-XF-PCIe範例列表	3
STM32範例環境建立導引(NUCLEO-F401RE)	4
STM32 F401RE 腳位建立導引	20
STM32 H7 腳位建立導引	20
新唐範例環境建立導引	22
修改範例程式為實際應用導引	25
常見錯誤排除	29
維護紀錄	30

範例程式說明

ECM-XF提供針對不同應用的多種範例，使用者可依實際需要選擇一個類似的範例參考，並修改成適合的應用程式。

若不是使用STM32或是新唐的控制晶片，則僅需替換SPI控制層韌體(與實際硬體相關)，及控制晶片實際SPI連接腳位之設定，其餘部分仍可參考範例程式。XF與XFU除了介面不同，預設的SPI Data Size 不同外，其餘控制流程皆相同，因此各平台間的範例可相互參考使用。

	與上位的介面	預設SPI Data Size	SPI Data Size 範圍
ECM-XF IC	SPI	112 Bytes	32~1408 Bytes
ECM-XF-PCle	PCle	112 Bytes	32~1408 Bytes
ECM-XFU-SK	USB 或 SPI (擇一)	996 Bytes	使用SPI介面32~1408Bytes 使用USB介面固定992Bytes

ECM-XF範例列表

提供STM32 F401與新唐M487的範例，STM32或新唐透過SPI介面與EXM-XF溝通，STM32的範例路徑為Core\ECM\example，新唐範例路徑為User\example

- ex_CSP
驅動器馬達配置及CSP (Cyclic Sync Position mode)模式範例
- ex_PP
驅動器馬達配置及PP (Profile Position mode)模式範例
- ex_PV
驅動器馬達配置及PV (Profile Velocity mode)模式範例
- ex_PT
驅動器馬達配置及PT (Profile Torque mode)模式範例
- ex_home
驅動器馬達配置及Homing (Homing mode)模式範例
- ex_rta
驅動器馬達Real Time Application範例

ECM-XFU-SK / ECM-XF-PCIe範例列表

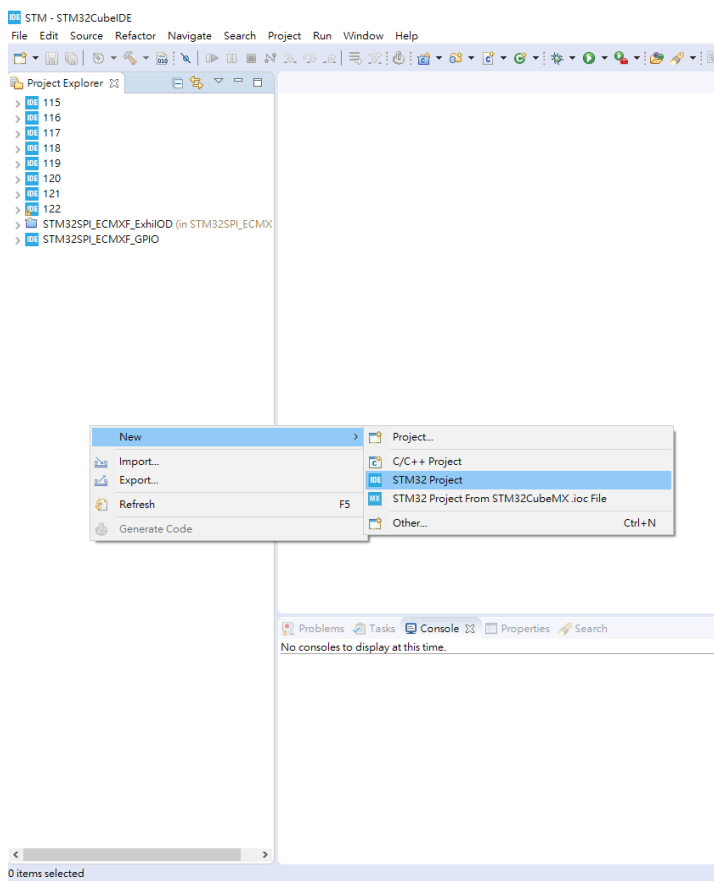
提供Windows電腦透過USB介面與ECM-XFU-SK溝通的範例，並有C語言與C#與語言兩種，可透過微軟Visual Studio邊程軟體進行編譯。

- DA_AD_QEI_LIO
晶片上DAC、ADC、QEI及GPIO的功能範例
- Drive_CSP
驅動器馬達配置及CSP (Cyclic Sync Position mode)模式範例
- Drive_CSP_500us
驅動器馬達配置及CSP (Cyclic Sync Position mode)模式，週期時間500us範例
此範例當中每次傳送2筆(TEST_PDO_TO_FIFO_ONCE = 2)週期命令資料
- Drive_CSV
驅動器馬達配置及CSP (Cyclic Sync Velocity mode)模式範例
- Drive_Homing_PP
驅動器馬達配置及Homing (Homing mode)模式、PP (Profile Position mode)模式範例
- Drive_IO
驅動器馬達配置及IO類型從站混合範例
- EEPROM
主站透過SII(Slave Information Interface)讀取從站EEPROM資料範例
- IO
IO類型從站操作範例
- SimpleTest / Sample
不配置從站(使用預設PDO)進入OP後直接交換資料，若FIFO有累積將顯示狀態

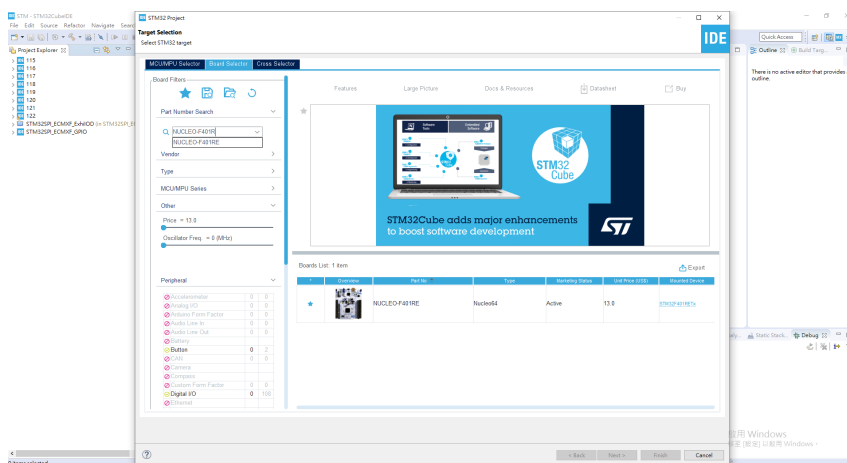
請注意，XF_SetDevType的參數將決定USB介面或是PCIe介面，請務必確認此參數正確。

STM32範例環境建立導引(NUCLEO-F401RE)

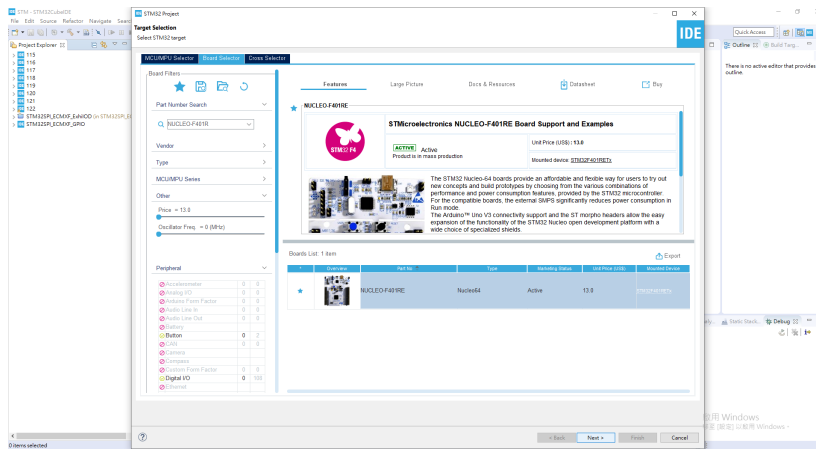
1. 開啟STM32Cube IDE.
2. 在左側視窗點擊滑鼠右鍵並選擇”STM32 Project” 或選擇 “Create a New STM32 project”



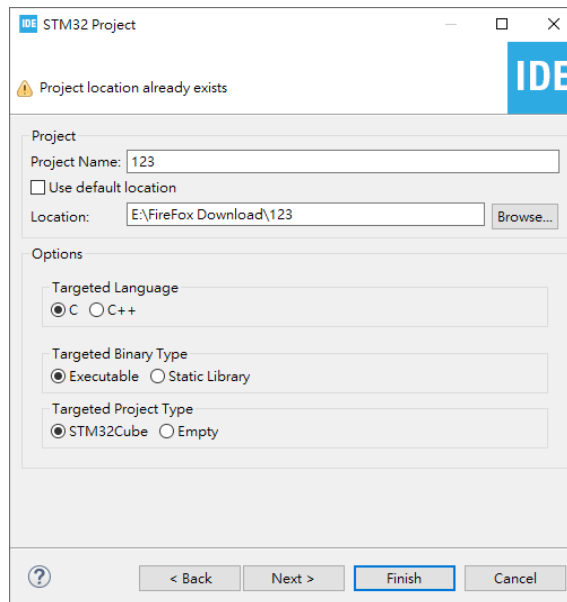
3. 選擇所使用的板子。這邊例子以STM32 NUCLEO-F401RE為例



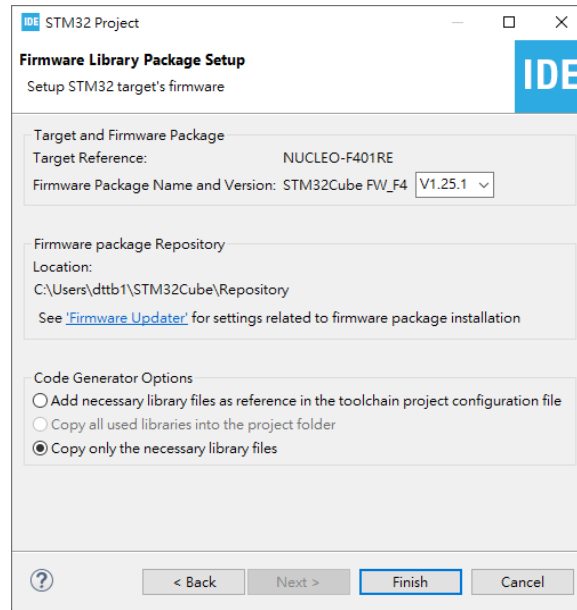
4. 選擇完成後點選下一步



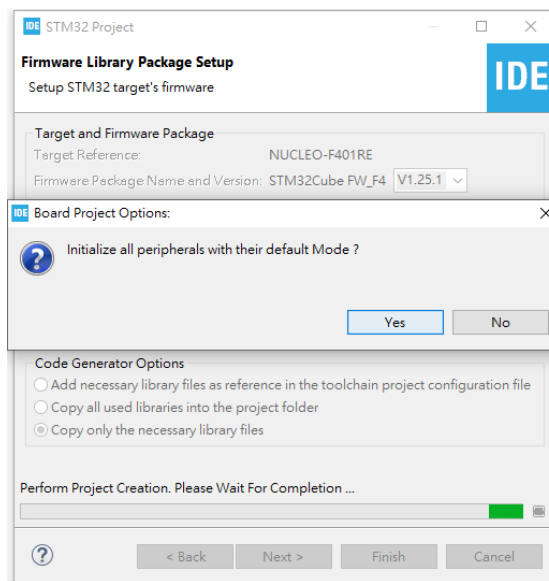
5. 輸入專案名稱與專案位置



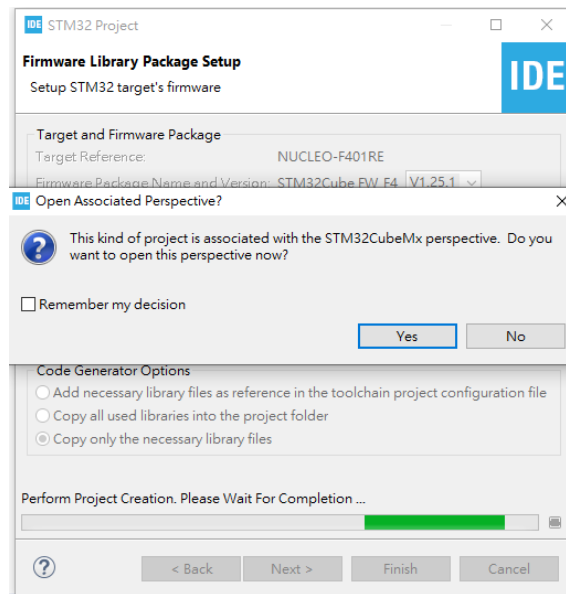
6. 選擇韌體並點選完成。如果沒有目前所選擇的版本，系統會自動下載所選擇的版本



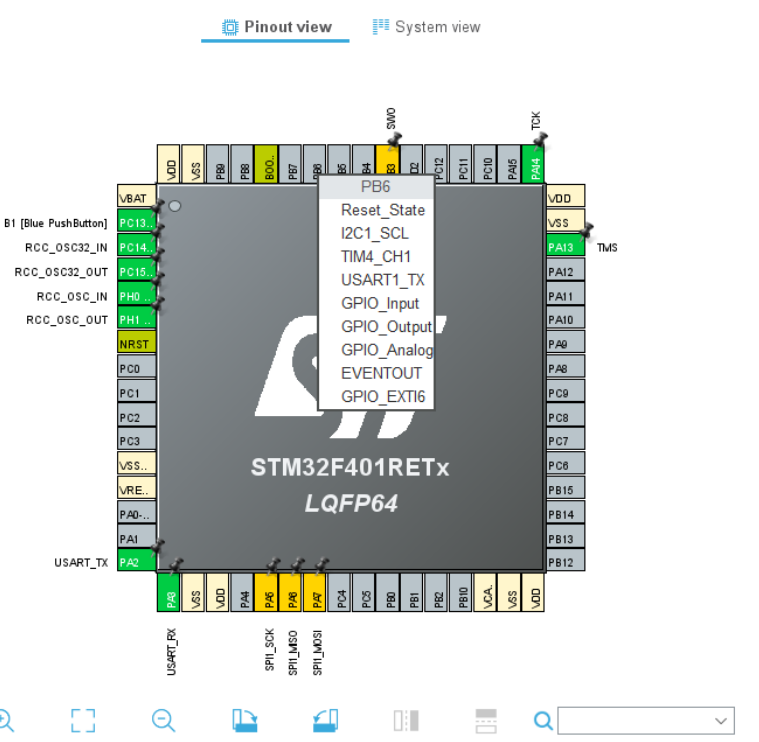
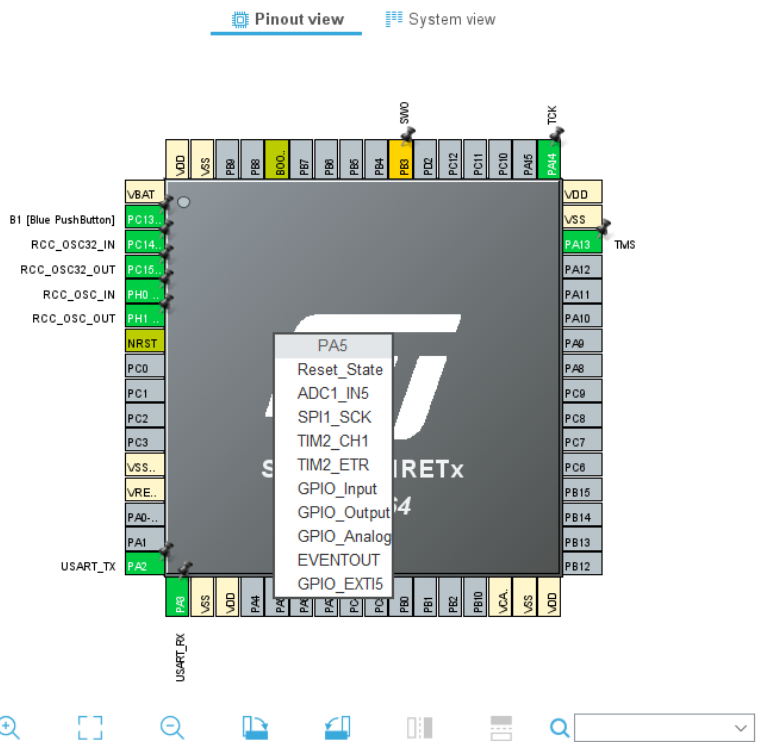
7. 視窗詢問初始化所有外部裝置至預設模式時選擇"是"



8. 詢問開啟STM32CubeMx時選擇"是"並開始定義腳位



9. 在腳位定義圖中，選擇PA5為SPI1_SCK、PA6為SPI1_MISO、PA7為SPI1_MOSI與PB6為GPIO_Output



10. 點選左側“System Core”對GPIO進行進階設定，將“GPIO output level”調整至“High”

The screenshot shows the STM32CubeIDE interface for configuring GPIO pins. The left sidebar is under 'System Core' with 'GPIO' selected. The main window is titled 'GPIO Mode and Configuration' and shows a configuration table for GPIO pins. The 'PB6' row is selected, and its configuration is shown in a detailed view below the table. The 'GPIO output level' is set to 'High'.

Pin ...	Signal o...	GPIO o...	GPIO m...	GPIO P...	Maximu...	User La...	Modified
PB6	n/a	High	Output ...	No pull-...	Low		<input checked="" type="checkbox"/>
PC13-A...	n/a	n/a	Externa...	No pull-...	n/a	B1 [Blu...	<input checked="" type="checkbox"/>

Configuration details for PB6:

- GPIO output level: High
- GPIO mode: Output Push Pull
- GPIO Pull-up/Pull-down: No pull-up and no pull-down
- Maximum output speed: Low
- User Label: (empty)

11. 點選"Timers"中的TIM2並將"Clock Source"改為"Internal Clock"後，將下方"Counter Period"數值改為"0xFFFFFFFF"

The screenshot shows the STM32CubeMX configuration interface for the TIM2 timer. The left sidebar lists various system components, with 'Timers' expanded to show TIM1 through TIM11. TIM2 is selected and highlighted in blue. The main panel is titled 'TIM2 Mode and Configuration' and is divided into two sections: 'Mode' and 'Configuration'.

Mode Section:

- Slave Mode: Disable
- Trigger Source: Disable
- Clock Source: Internal Clock
- Channel1: Disable
- Channel2: Disable
- Channel3: Disable
- Channel4: Disable
- Combined Channels: Disable
- Use ETR as Clearing Source
- XOR activation

Configuration Section:

- Reset Configuration button
- User Constants, NVIC Settings, and DMA Settings are all checked.
- Parameter Settings is checked.
- Configure the below parameters:
- Search (Ctrl+F) field
- Counter Settings:
 - Prescaler (PSC - 16 bits val...): 0
 - Counter Mode: Up
 - Counter Period (AutoReload...): 0xffffffff
 - Internal Clock Division (CKD): No Division
 - auto-reload preload: Disable

12. 接下來跳至“Connectivity”設定“SPI1”，模式設定“Full-Duplex Master”並將鮑率設置在2~8之間

Screenshot of the SPI1 Mode and Configuration settings in a development tool. The left sidebar shows the 'Connectivity' category expanded to 'SPI1'. The main panel shows 'Mode' set to 'Full-Duplex Master' and 'Hardware NSS Signal' set to 'Disable'. The 'Configuration' section includes 'Reset Configuration', 'DMA Settings', 'GPIO Settings', 'Parameter Settings', 'User Constants', and 'NVIC Settings'. Under 'Parameter Settings', 'Frame Format' is 'Motorola', 'Data Size' is '8 Bits', and 'First Bit' is 'MSB First'. Under 'Clock Parameters', 'Prescaler (for Baud Rate)' is '8', 'Clock Polarity (CPOL)' is 'Low', and 'Clock Phase (CPHA)' is '1 Edge'. Under 'Advanced Parameters', 'CRC Calculation' is 'Disabled'.

SPI1 Mode and Configuration	
Mode	
Mode	Full-Duplex Master
Hardware NSS Signal	Disable
Configuration	
Reset Configuration	
DMA Settings	
GPIO Settings	
Parameter Settings	
User Constants	
NVIC Settings	
Frame Format	Motorola
Data Size	8 Bits
First Bit	MSB First
Clock Parameters	
Prescaler (for Baud Rate)	8
Clock Polarity (CPOL)	Low
Clock Phase (CPHA)	1 Edge
Advanced Parameters	
CRC Calculation	Disabled

13. 接下來跳到USART2設定

The screenshot shows the STM32CubeMX Pinout & Configuration window. The left sidebar lists various components, with USART2 selected under the Connectivity section. The main window displays the USART2 Mode and Configuration settings.

Pinout & Configuration | **Clock Configuration** | Software Pack

Search: []

Categories: A->Z

USART2 Mode and Configuration

Mode

Mode: Asynchronous

Hardware Flow Control (RS232): Disable

Configuration

Reset Configuration

✓ NVIC Settings | ✓ DMA Settings | ✓ GPIO Settings

✓ Parameter Settings | ✓ User Constants

Configure the below parameters :

Search (Ctrl+F) []

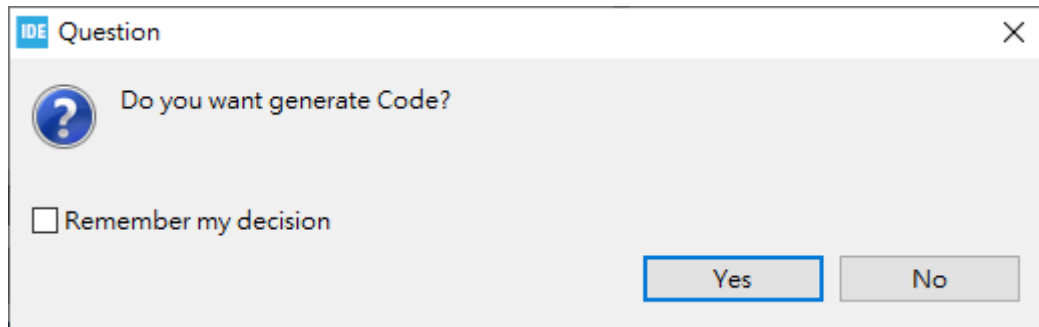
Basic Parameters

Baud Rate	115200 Bits/s
Word Length	8 Bits (including Parity)
Parity	None
Stop Bits	1

Advanced Parameters

Data Direction	Receive and Transmit
Over Sampling	16 Samples

14. 完成上面步驟後，點擊儲存標示儲存並自動產生程式



15. 如果在生成的main.c “Private variables”中看到剛設定的SPI、TIM2表示有之前的設定有完成，若未看到相關資訊，可先返回

```
/* Private variables -----*/  
SPI_HandleTypeDef hspi1;  
  
TIM_HandleTypeDef htim2;  
  
UART_HandleTypeDef huart2;
```

16. 點開“main.h”並依下圖加入下方程式碼

```
/* USER CODE BEGIN Includes */  
#include "platform.h"  
#include "EcmUsrDriver.h"  
/* USER CODE END Includes */  
  
/* Exported macro -----*/  
/* USER CODE BEGIN EM */  
#ifndef PRINTF  
#define PRINTF( str, ...) \  
do{ \  
int n; \  
n = sprintf( printbuf, (str), ##_VA_ARGS_); \  
HAL_UART_Transmit( &huart2, (uint8_t *)printbuf, n, 0xffffffff); \  
}while(0)  
#endif  
#ifndef GETCHAR  
#define GETCHAR userGetchar  
#endif  
/* USER CODE END EM */  
  
/* USER CODE BEGIN EFP */  
extern UART_HandleTypeDef huart2;  
extern char printbuf[];  
/* USER CODE END EFP */
```

17. 開啟“main.c”並加入下方程式碼

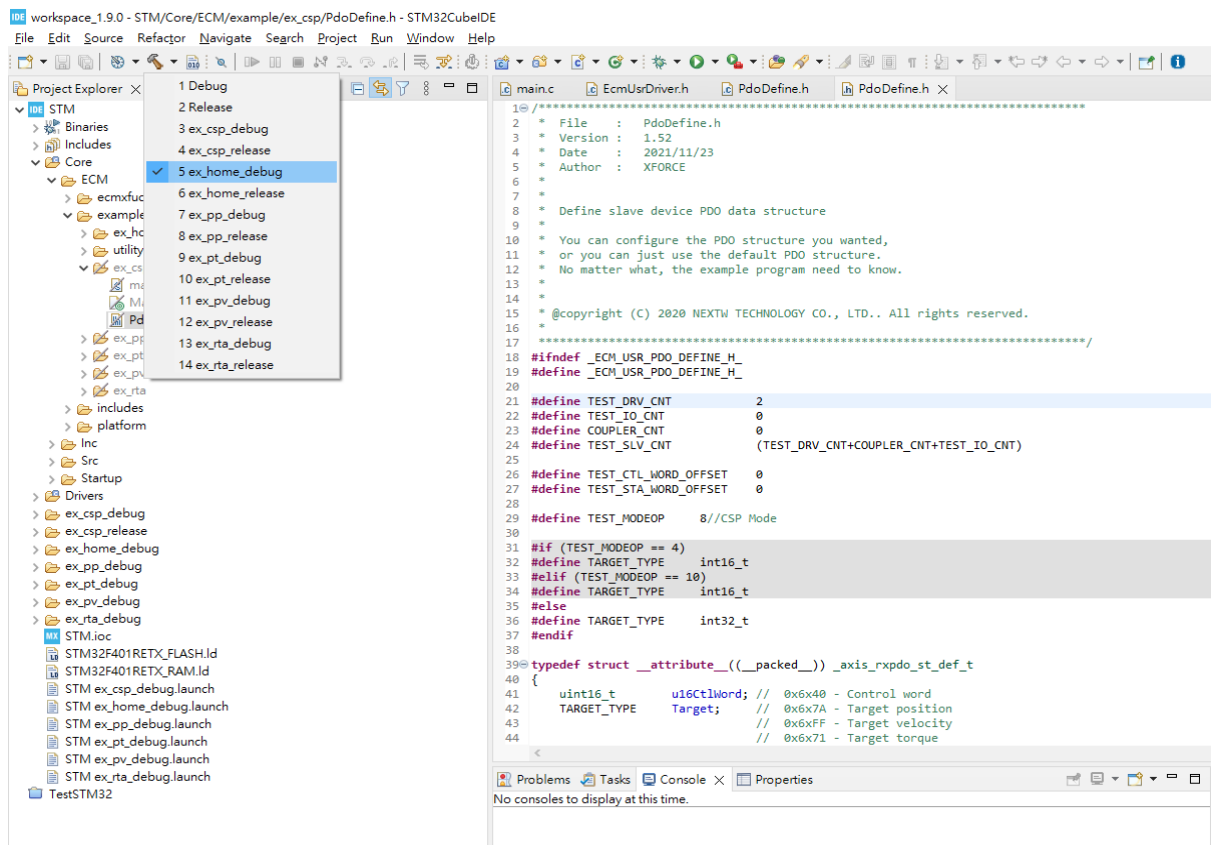
```
/* USER CODE BEGIN PFP */  
char printbuf[128];  
int main_ini(void);  
/* USER CODE END PFP */  
  
/* USER CODE BEGIN 2 */  
main_ini();  
/* USER CODE END 2 */  
  
/* USER CODE BEGIN TIM2_Init 2 */  
HAL_TIM_Base_Start(&htim2);  
/* USER CODE END TIM2_Init 2 */
```

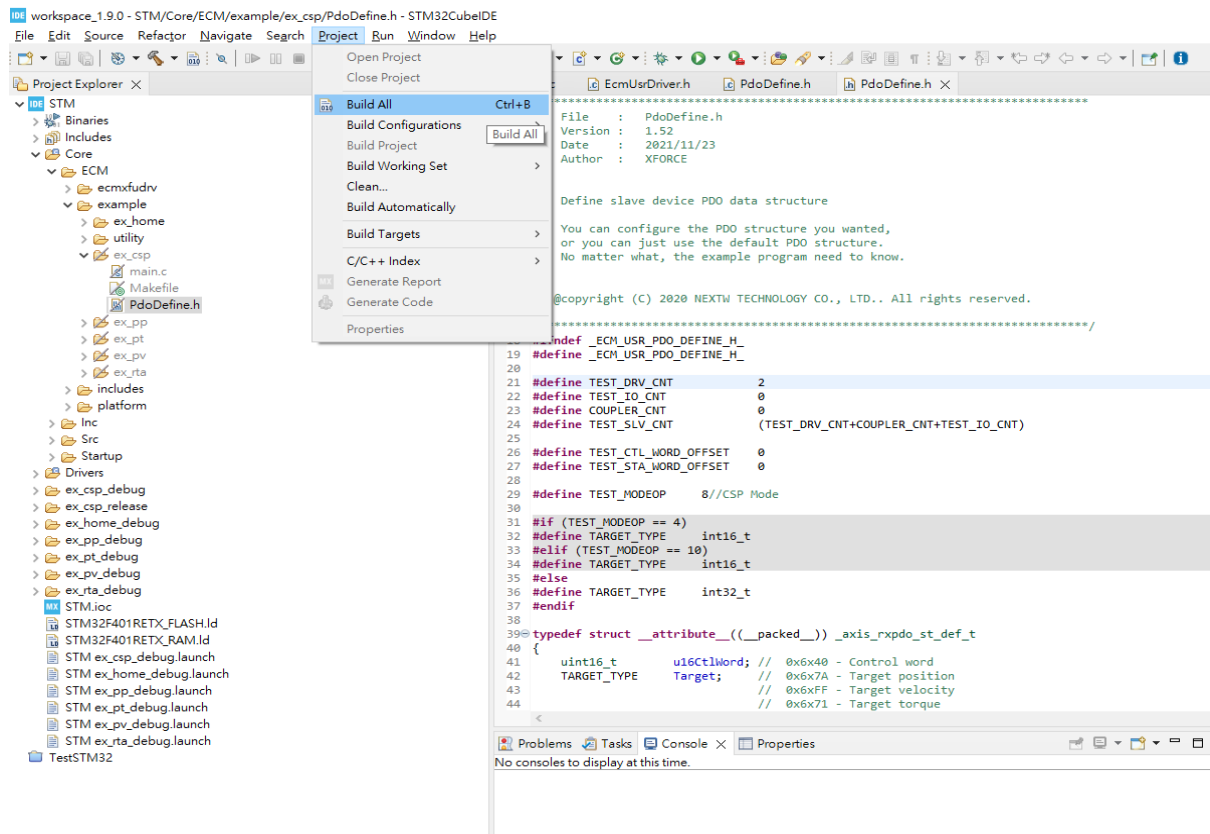
18. 複製EcmDriver.h、EcmUsrDriver.h、PdoDefine.h、platform.h與Utility.h(部分範例)貼入“Inc”資料夾

19. 複製crc32.c(部分範例無此檔案)、EcmUsrDriver.c、main_ini.c(主程式:部分範例為不同名)與platform.c、Utility.c(部分範例)貼入Src資料夾

20. 已準備好執行範例程式

20. 點擊槌子圖示，並選取欲執行的程式來進行Build，或是點擊Project底下的Build all來建置程式

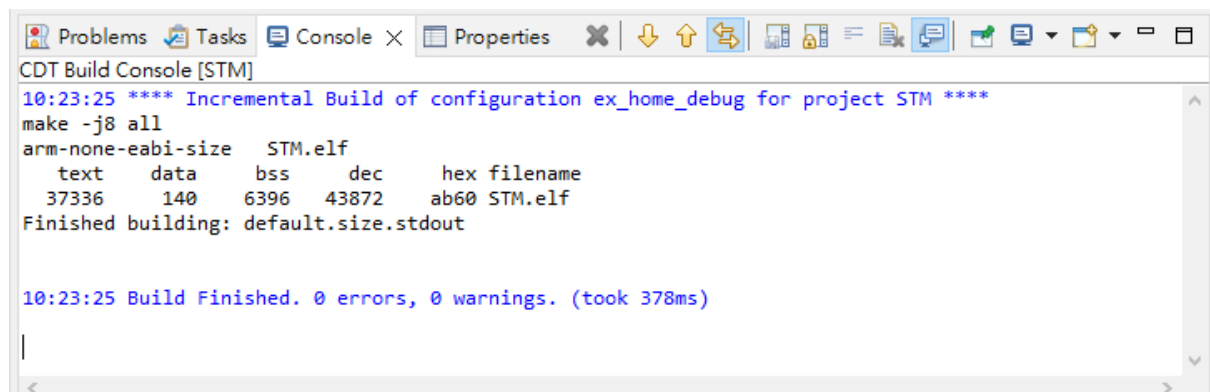




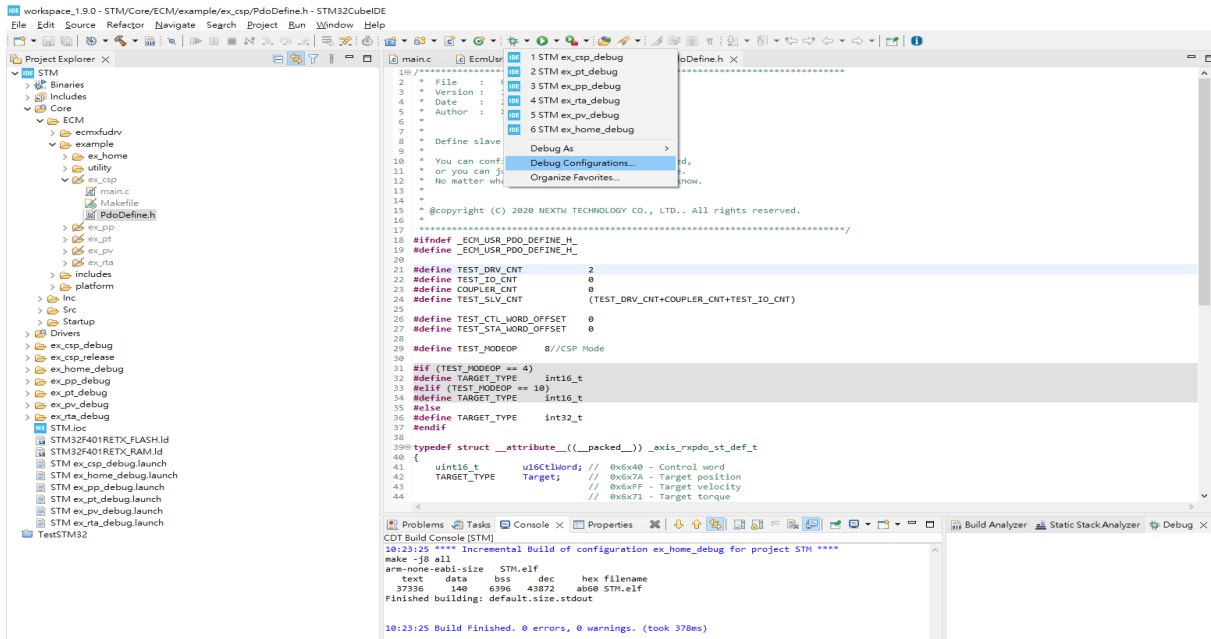
點擊後會開始建置build

在Console欄中顯示完成訊息

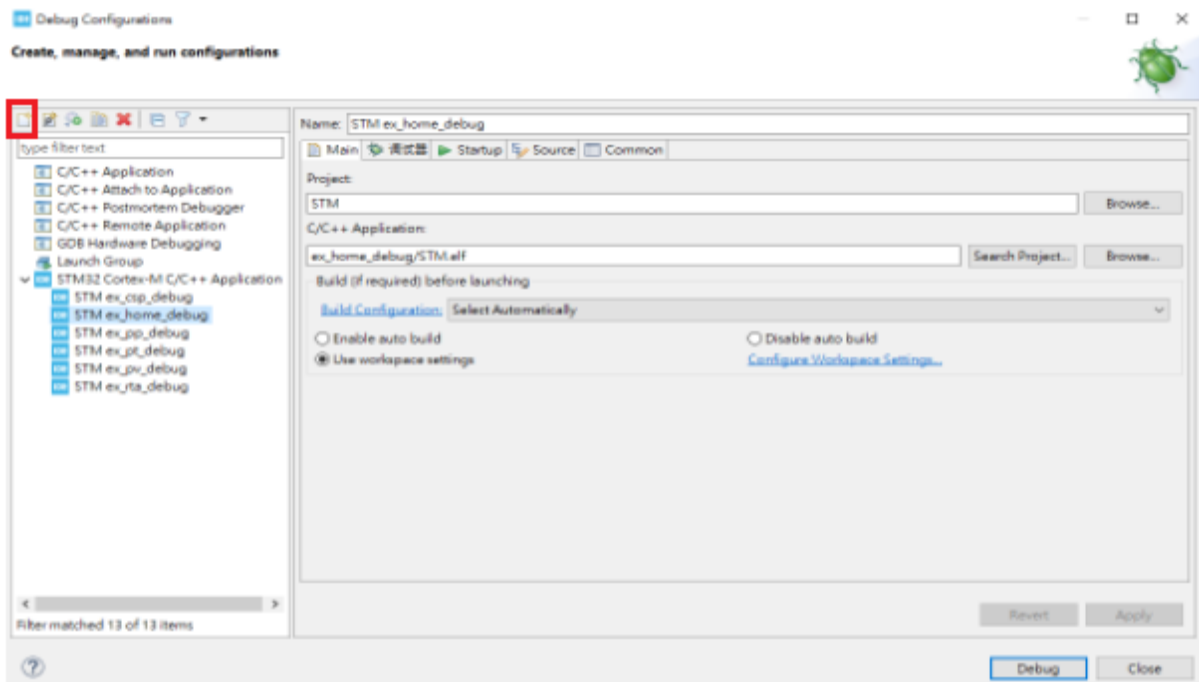
(若有錯誤請檢查錯誤原因並排除, 注意程式路徑不可包含非英文語系文字)



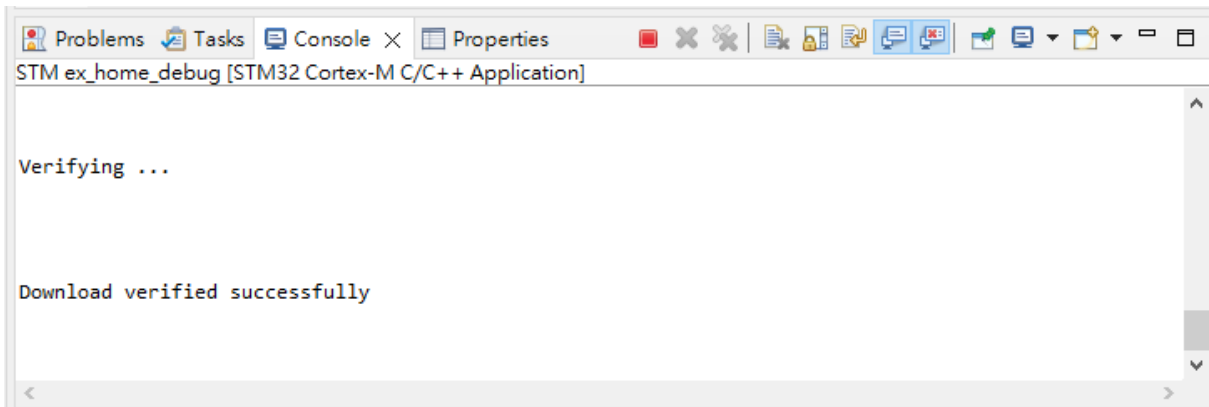
21. 下一步要進行Debug, 在上方工具列中點擊蟲子的圖示, 並點選Debug Configurations.



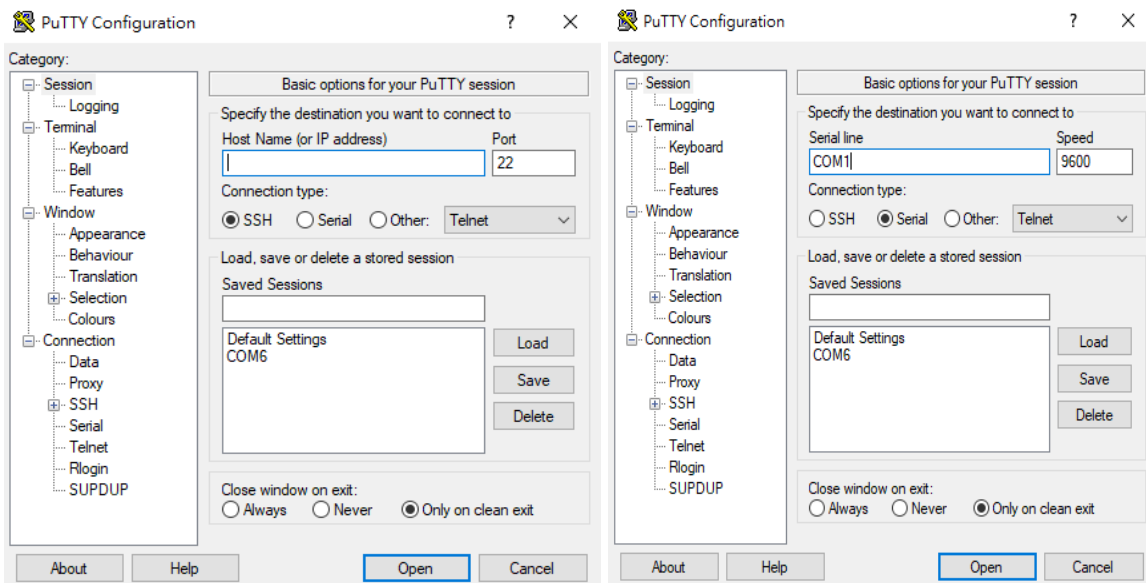
在Debug Configurations中點選STM32, 並選擇要Debug的程式再按下Debug就會開始偵錯, 若沒有出現請按上方加的圖示 (New launch configurationM, 紅框處)



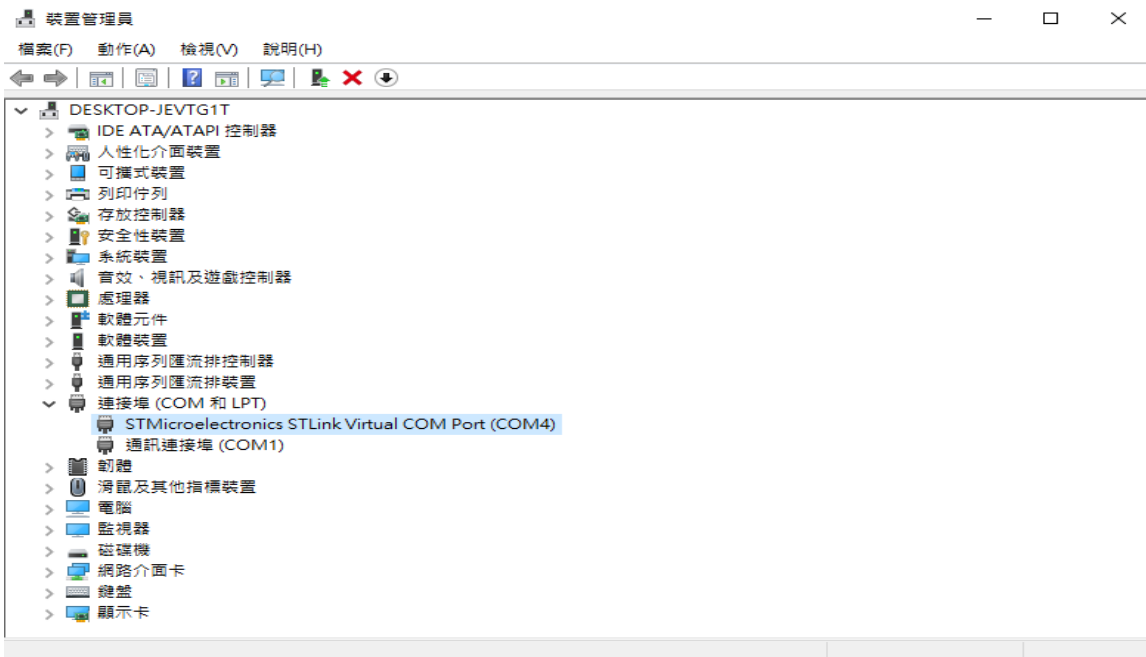
程式會下載至MCU中並在Console顯示相關訊息



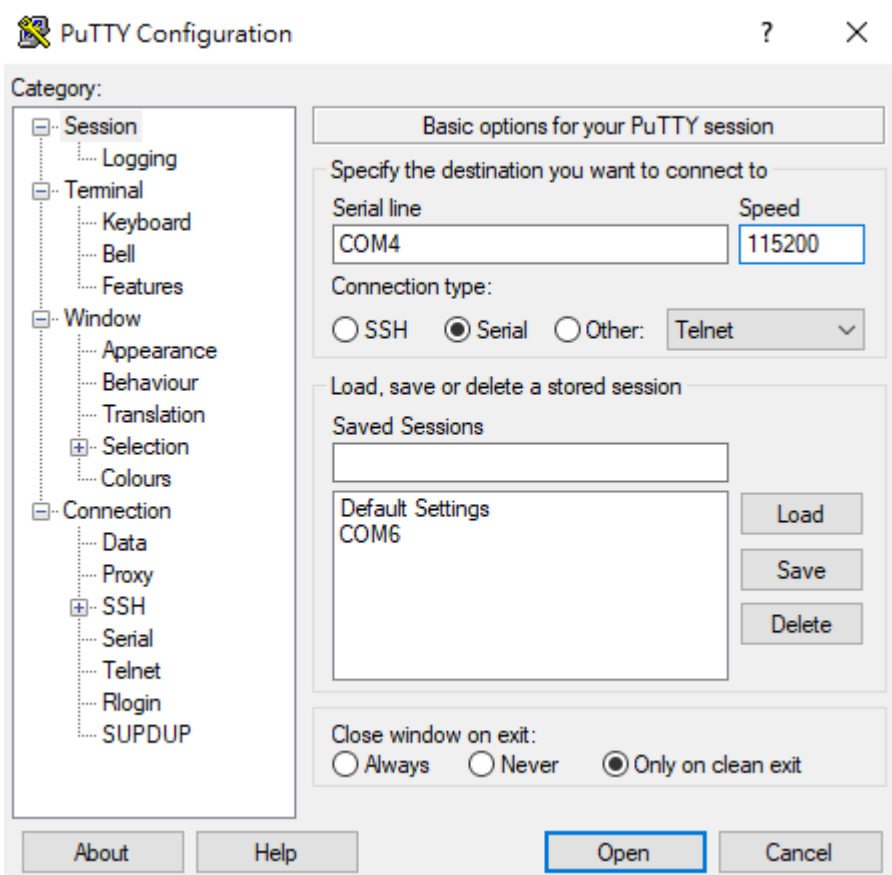
22.開啟序列通訊埠程式 (如Putty) , 並在Connection type中勾選Serial



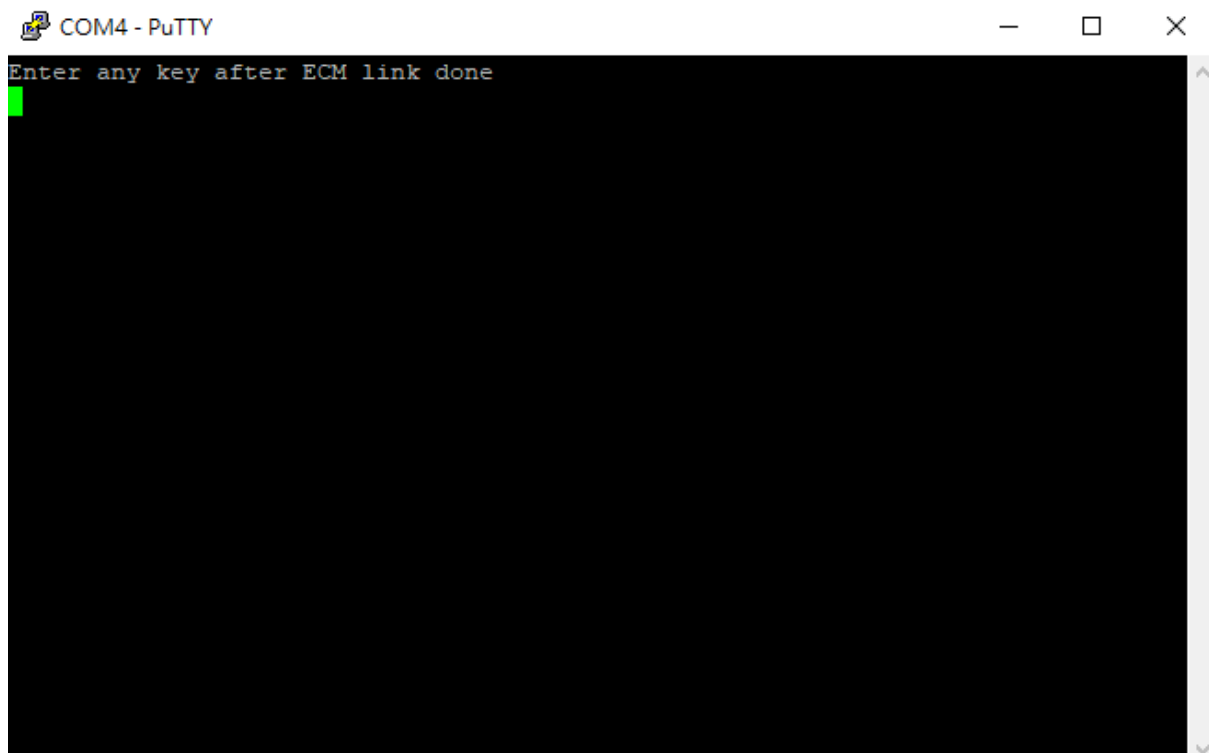
再在Serial line中輸入對應到的COM port(可以在裝置管理員中的連接埠確認, 本例為COM4)



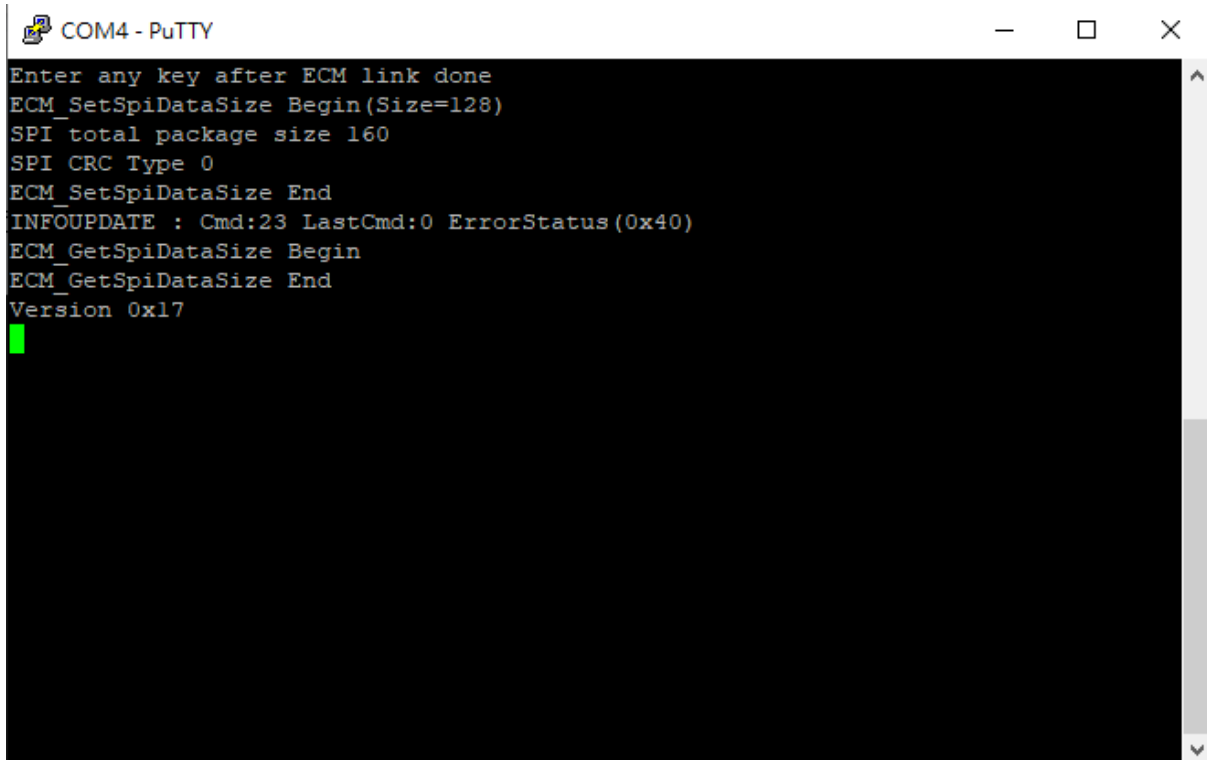
設定好Serial line後, speed輸入 115200, 接著就可以點擊Open



23.進入到Putty或其他RS232序列通訊公用程式，開始執行(按下STM F401RE版上的reset鍵,或點選STM32CubeIDE中[執行]按鈕)，畫面可得程序中printf的文字



確認ECM-XFU-SK已連結至從站，且網路口燈亮起，按下鍵盤上任意鍵就會開始執行程式

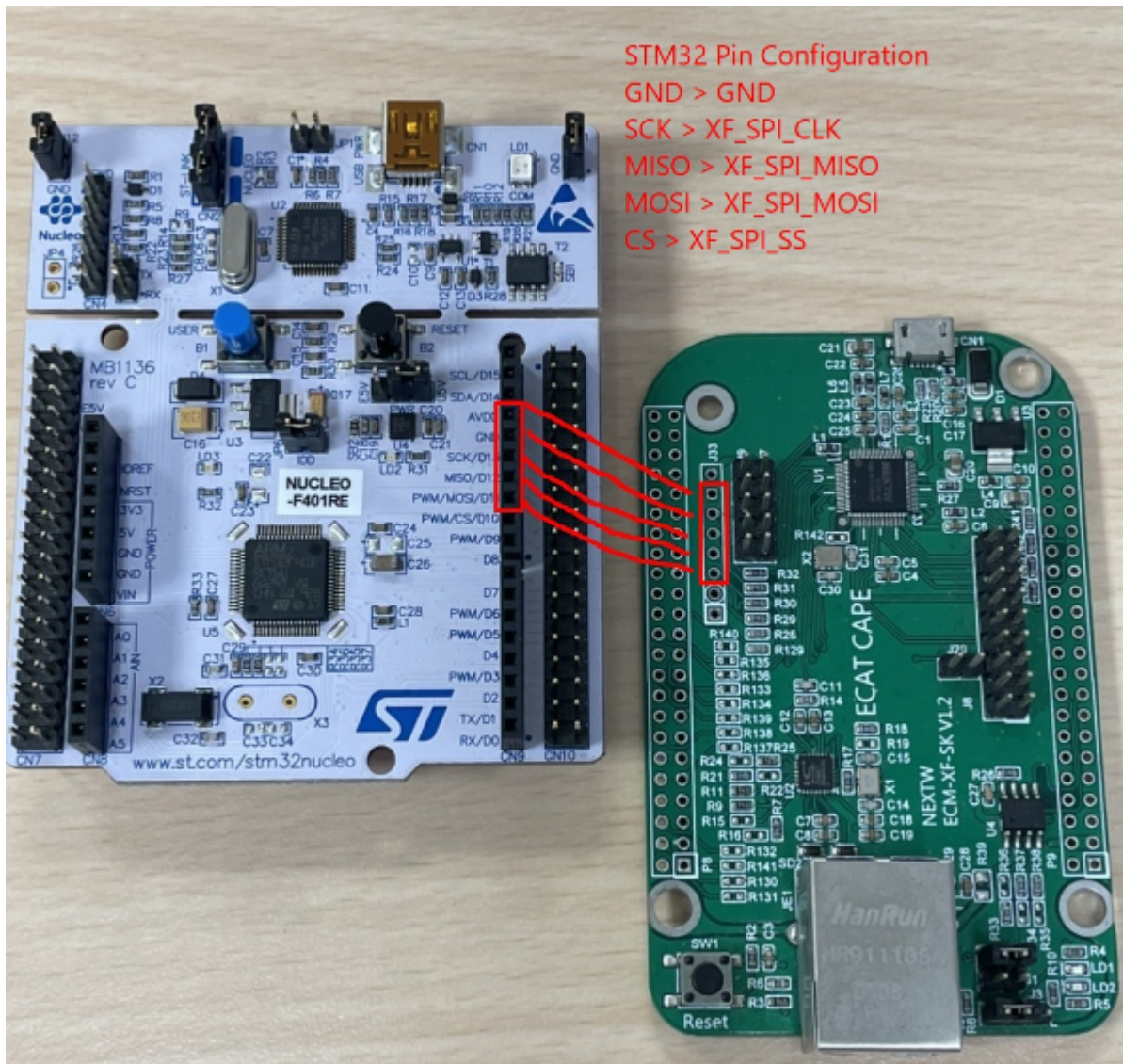


```
COM4 - PuTTY
Enter any key after ECM link done
ECM_SetSpiDataSize Begin(Size=128)
SPI total package size 160
SPI CRC Type 0
ECM_SetSpiDataSize End
INFOUPDATE : Cmd:23 LastCmd:0 ErrorStatus(0x40)
ECM_GetSpiDataSize Begin
ECM_GetSpiDataSize End
Version 0x17
█
```

有出現版本號表示成功代表SPI連線傳輸正常

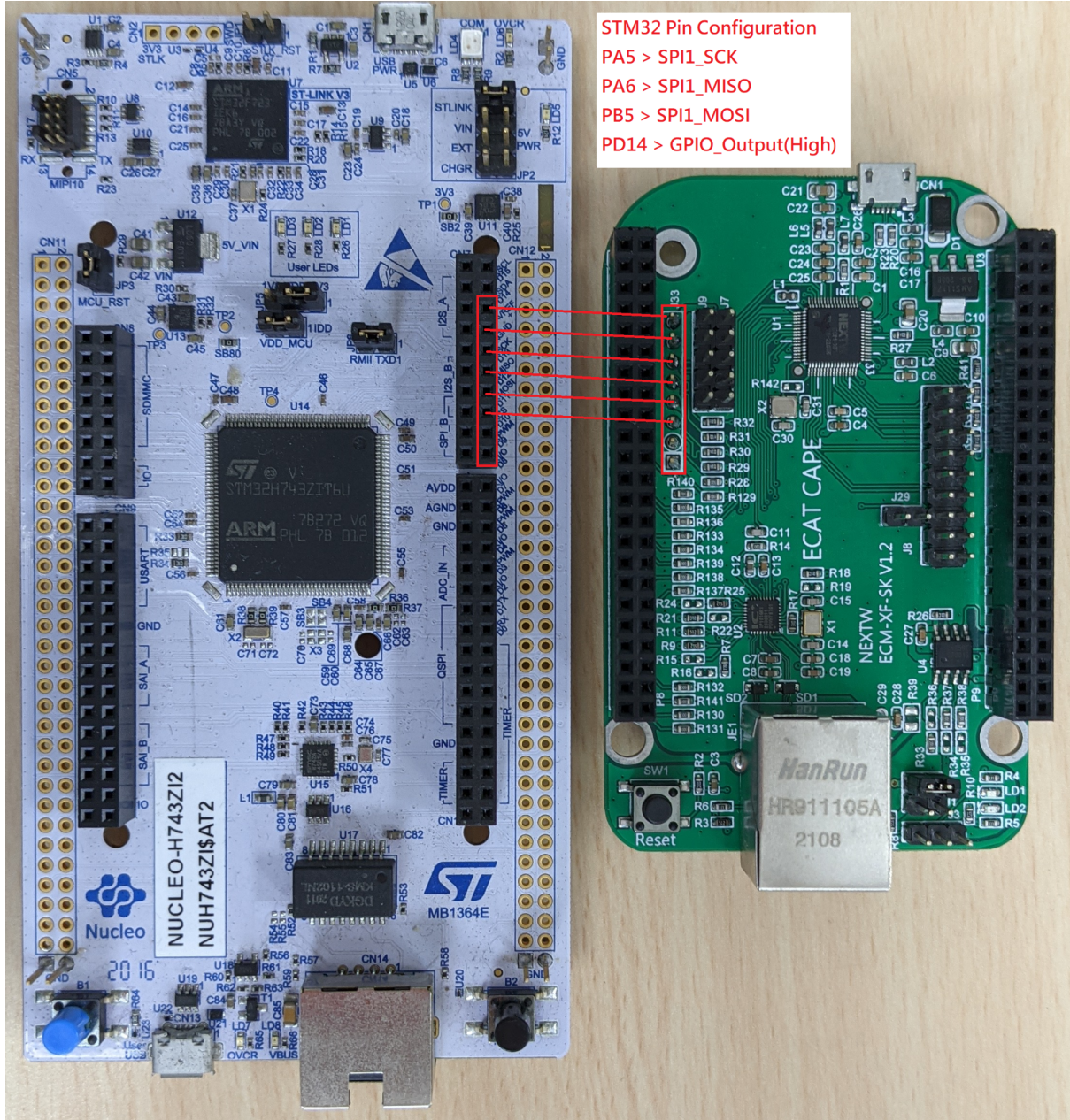
若出現failed訊息可能為接觸不良，調整版子接合處並按下ECM-XF-SK上的reset鍵重新執行

STM32 F401RE 腳位建立導引



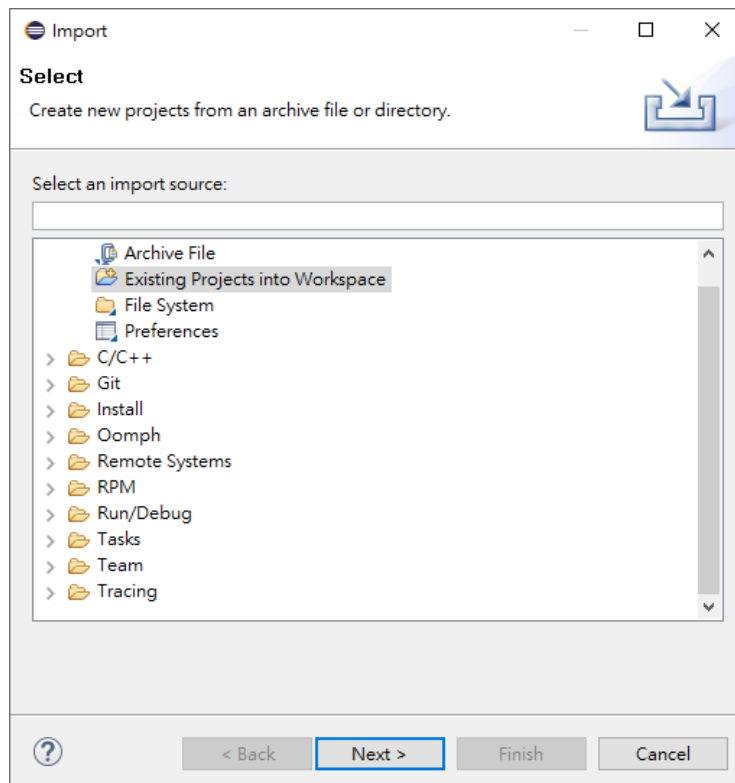
STM32 H7 腳位建立導引

STM32所生成之main.c與main.h不與F4版本共用，請使用H7範例為範本修改
請使用H7專用platform.h與platform.c
需要F4版本範例可以直接複製main_ini.c覆蓋至H7 main_ini.c



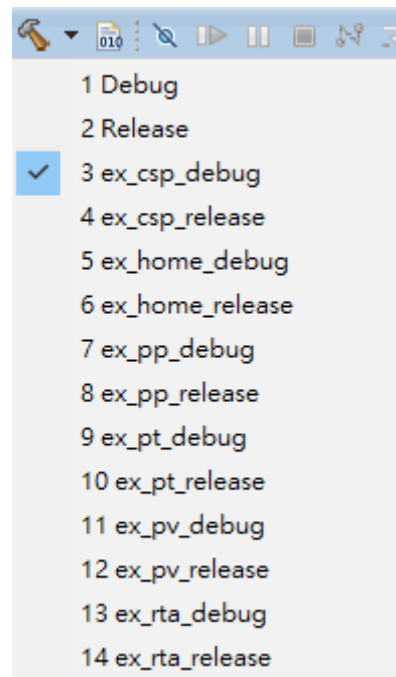
新唐範例環境建立導引

1. 至新唐官網依作業系統下載Nueclipse GCC(開源IDE), 連結為官方下載頁面:
[M487 Ethernet Series - Nuvoton](#)
2. 在同一頁面下載Nu-Link Keil driver
3. 同一頁面下載M480_BSP_CMSIS
4. 安裝IDE與Keil driver
5. 下載範例程式並放入workspace位置
6. 開啟Eclipse並在Project Explorer的位置滑鼠右鍵選擇”Import”
7. 在匯入畫面依以下順序操作: General > existing project into workspace

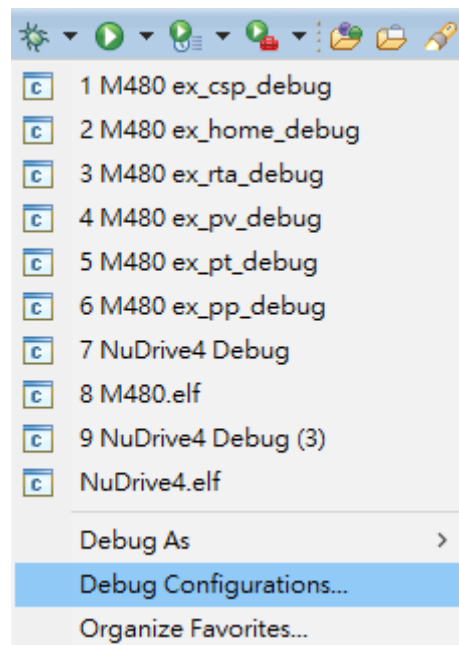


8. 選擇預計開啟的根目錄，範例程式會在中間的視窗出現並點選”Finish”

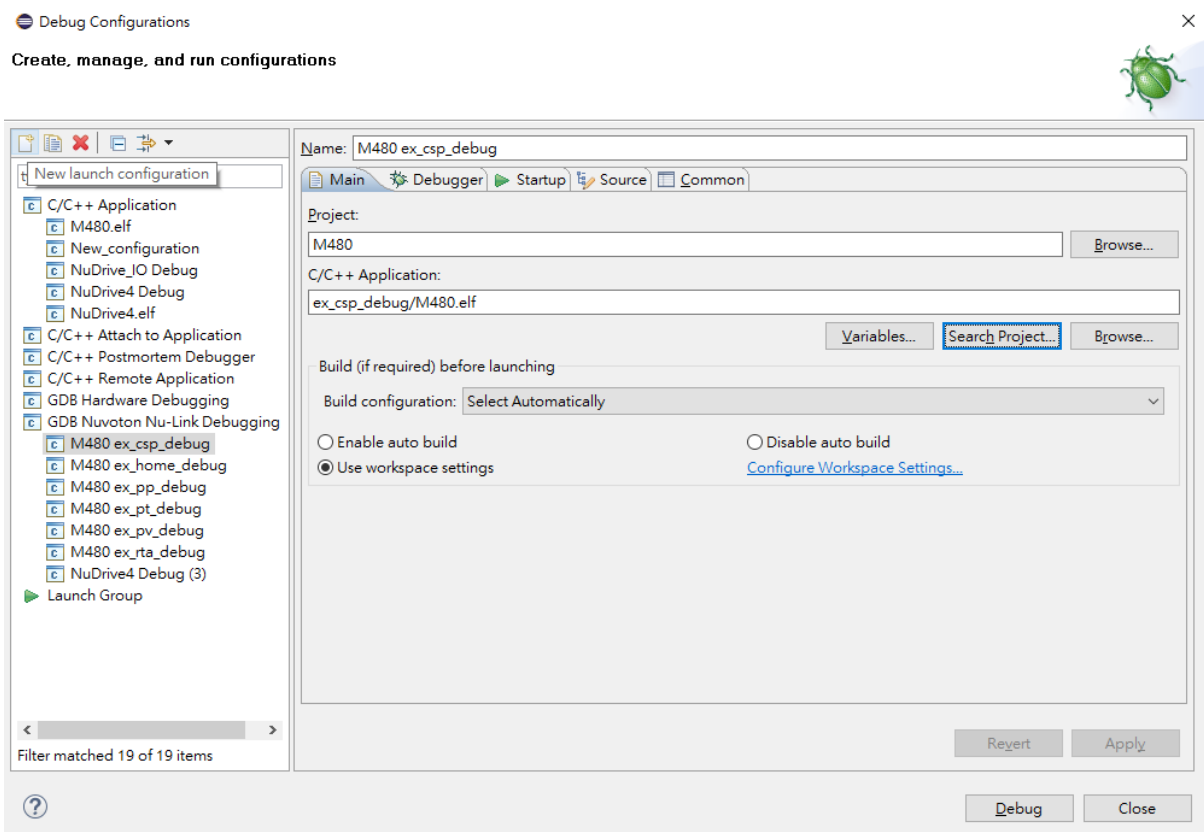
9. 在成功匯入專案後，可以使用左上方工具列的槌子按鈕選擇要進行build的專案



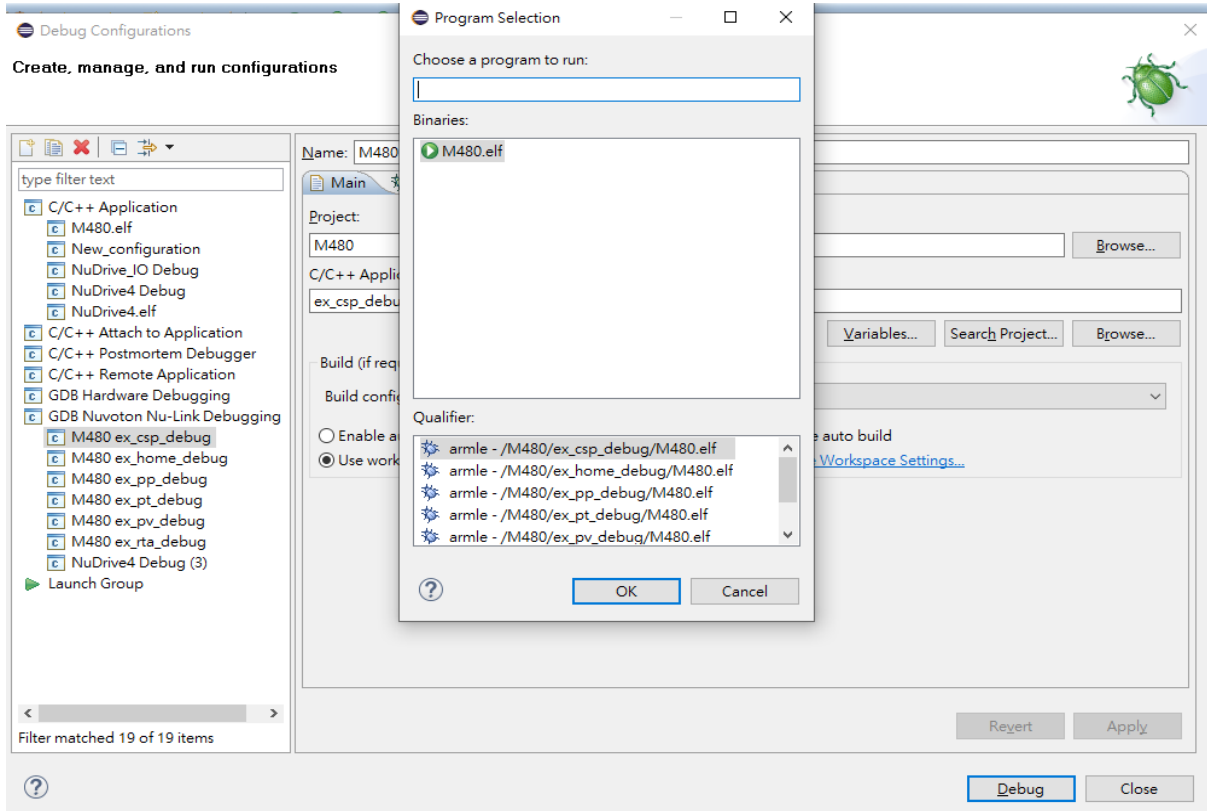
10. 選擇完畢後，右鍵點選或上方工具列點選建置(build)專案。專案可以允許建置 Debug模式與Release模式，建議使用Debug模式，在此選取Debug Configurations。



11. 當專案建置完成後，設定Debug配置與Run配置。在GDB Nuvoton Nu-Link Debugging點選左上圖標名稱為”New launch configuration”，來建立新的configuration接下來應會自動生成剛剛建置完成的專案名稱。



12. 建立新的New launch configuration後，點選右方Search Project，確認選取到相對應名稱的檔案後點擊OK，都完成後再點擊Debug，系統會自動將程式寫入晶片



修改範例程式為實際應用導引

正確開啟範例程式後可依以下步驟順序進行改寫

使用Drive的應用以STFDrive或NuDrive為範本進行修改，中間會有其他參考別範例中的做法

PdoDefine.h設定

1. 點選並進入PdoDefine.h中後，在#define中輸入各類型從站(如Drive或IO或其他)數量與單站軸數。目前馬達控制器發展已有單控制器可多軸控制，所以若是一個控制器可以控制兩馬達進行運動，這樣的情況就會是一個從站與兩個軸數，下圖TEST_DRV_CNT代表DRV類型的從站數量，TEST_IO_CNT代表IO類型的從站數量。

```
#define TEST_DRV_CNT      1
#define TEST_IO_CNT      0
```

2. 完成設定後拉至下方繼續設定各類型從站的資料結構(PDO structure)。資料結構分為RxPDO(主站傳給從站的object)與TxPDO(從站傳給主站的object)兩種，不同類型的從站如果資料結構不相同，應該分開設定。

下圖設定了RxPDO共有兩個Object，分別為16bit的Control Word與32bit的Target。TxPDO也有兩個Object，分別為16bit的Status Word與32bit的Actual。若為TEST_HSP_DEV型式的子站，則RxPDO共有三個Object，分別為16bit的Control Word、32bit的Target及32bit的Out。TxPDO也有三個Object，分別為16bit的Status Word、32bit的Actual及32bit的In。

```
typedef struct __attribute__((__packed__)) _axis_rxpdo_st_def_t
{
    uint16_t    u16CtlWord;    // 0x6x40 - Control word
    int32_t     n32Target;    // 0x6x7A - Target position

#ifdef TEST_HSP_DEV
    int32_t     n32Out;       // 0x6xFE
#endif
}AXIS_RXPDO_ST_DEF_T;

typedef struct __attribute__((__packed__)) _axis_txpdo_st_def_t
{
    uint16_t    u16StaWord;    // 0x6x41 - Status word
    int32_t     n32Actual;    // 0x6x64 - Actual position

#ifdef TEST_HSP_DEV
    int32_t     n32In;       // 0x6xFD
#endif
}AXIS_TXPDO_ST_DEF_T;
```

STM32系列中main_ini.c或Nuvoton系列中main.c設定

1. 首先要先設定SPI通訊時間，STM32系列中的SPI通訊時間必須另外安裝STM32CubeMX來設定，Nuvoton系則以main.c中的#define TEST_SPI_FREQ進行設定或是尋找UserSys_Init()並填入SPI傳輸值。同時輸入欲測試的週期時間，SPI相關設定可以參閱上位控制晶片(如MCU)的說明。

```
#define TEST_SPI_FREQ      24000000
#define DC_ACTIVE_CODE    0x300
#define BASE_CYCTIME      1000000
```

```
UserSys_Init(TEST_SPI_FREQ);
```

- 範例中關於TEST_CYCTIME_NS是定義週期時間，透過MULTI及DIVID的相關define可以計算最後的週期時間。另外RPM與PPR相關的設定、速度相關設定是在OP階段馬達運轉的運動參數。

```

/*
 * TEST_CYCTIME_DIVID
 * 1 : for TEST_CYCTIME_MULTI
 * 2 : 0.5ms
 * 4 : 0.25ms
 * 8 : 0.125ms
 *
 * TEST_CYCTIME_MULTI
 * 1 : for TEST_CYCTIME_DIVID
 * 2 : 2ms
 * 4 : 4ms
 */
#define TEST_CYCTIME_DIVID 1
#define TEST_CYCTIME_MULTI 1
#define TEST_CYCTIME_NS ((BASE_CYCTIME*TEST_CYCTIME_MULTI)/TEST_CYCTIME_DIVID)
#define ONE_SEC_CYC_CNT (1000000000/TEST_CYCTIME_NS)

```

- 設定完成後下方為FIFO設置，FIFO為一暫存空間並依照順序與週期時間將命令由主站送至從站，若週期時間較短(如小於1ms)時，如果每次SPI或USB交換只送一次命令，很有可能會造成RxFIFO為空，當RxFIFO為空時，會保持原值送出，在CSP模式下，很可能造成速度不連續，導致馬達運轉會有震動的情況，此時就應加大每週期時間所傳輸的命令數量，這邊就會修改TEST_PDO_TO_FIFO_ONCE的數值，2為每周期間所傳輸的命令數量為兩筆。實際上可觀察RxFIFO數量可否累積來決定TEST_PDO_TO_FIFO_ONCE是否過小。

```

#define TEST_PDO_TO_FIFO_ONCE 2
#define TEST_RXFIFO_CNT 40
#define TEST_TXFIFO_CNT TEST_PDO_TO_FIFO_ONCE

```

- 設定完成後跳至主程式int main_ini()或int main()中開始進行程式的編寫與說明，一開始會先將記憶體的位置清空以確保接下來運作正常

```

uint64_t u64Data;
int nStartFIFO = 0;
int i = 0, DrvIdx = 0, AxisIdx = 0, YasIdx = 0, nCnt1Sec=0, nRunTimeCnt=0;
int nVel = 0, nSumVel = 0, n32DO=0x5555;
int nret = 0, nServoState = 1;
int nLogStart = 0;
int n32CurPos[TEST_AXIS_CNT];
uint16_t u16LastSW[TEST_AXIS_CNT], u16StalWord[TEST_DRV_CNT][N_AXIS_IN_ONE_DRV];
uint32_t u32CycTimeCnt = 0, u32RunTimeCnt = 0, u32LogFifoCnt=0;
uint8_t u8LEDAxis = 0;
uint8_t u8LEDBit = 0;
uint8_t u8Version = 0, u8FifoCnt = 0, u8FifoCntMax = TEST_RXFIFO_CNT;
uint8_t u8State = 0, u8WkcErrCnt = 0, u8CrcErrCnt = 0, u8IsSlvAlive = 0;
uint8_t u8LastState = 0, u8LastWkcErrCnt = 0, u8LastCrcErrCnt = 0;
uint16_t u16RxPDOSize = 0, u16TxPDOSize = 0, u16SpiSize = 0;
int nDriveRxPDOSize = 0, nDriveTxPDOSize = 0;
int8_t SlaveCnt = 0;
RXPDO_ST_DEF_T *pAllDevRx;
TXPDO_ST_DEF_T *pAllDevTx;
AXIS_RXPDO_ST_DEF_T *pRxPDOAxis;
AXIS_TXPDO_ST_DEF_T *pTxPDOAxis;
memset(RxPDOData, 0, sizeof(RxPDOData));
memset(TxPDOData, 0, sizeof(TxPDOData));
memset(nPos, 0, sizeof(nPos));

```

- EtherCAT主站的第一個命令為ECM_InitLibrary(&u16SpiSize)。當中會執行SPI資料大小的設定與確認，同時會確認主站IC的韌體版本等功能。u16SpiSize為欲

設定新SPI資料大小，在C語言中若ECM_InitLibrary(0)代表不更改SPI Data Size。

```
u8Version = ECM_InitLibrary(&u16SpiSize);
```

6. 當確認項目皆正常執行後，開始進行EtherCAT初始化
ECM_EcatInit(DCActCode, CycleTime)，並進入Init state
ECM_EcatInit(DCActCode, CycleTime)中DCActCode值0為關閉DC sync功能、0x300啟動Sync0、0x700同時開啟Sync0與Sync1，CycleTime為週期時間，單位為ns。

※此函數會針對"所有" Slave 下達相同的DCActCode，若對於不同Slave要設定不同的DCActCode，則需使用ECM_CMD_ECATCH_SYNC（命令代碼50）

```
ECM_EcatInit(DC_ACTIVE_CODE, (BASE_CYCTIME*TEST_CYCTIME_MULTI) / TEST_CYCTIME_DIVID);
```

7. 確認主站與從站連接後透過ECM_StateCheck(Slave, ExpectedState, Timeout)進入Pro-OP開始設定PDO配置，如果從站已有預設配置則只需要給予相對應的Map Index即可，如為特殊配置則需配置PDO內容，SetPdoConfig提供最多3組PDO各8個Object的配置大小，多數範例配置內容在main_ini.c或main.c中，目前唯Drive範例在另外在Utility.c中。如果使用ConfigDrive進行配置，內部已有Reconfig與ShowPDOConfig的動作，所以無需另外加入Reconfig動作，其他範例則需在完成Configure PDO後加入Reconfigure的動作與ShowPDOConfig。
ECM_StateCheck(Slave, ExpectedState, Timeout)中Slave如為0xFF則表示針對所有從站同時命令，ExpectedState則是欲前往之狀態(Pro-OP、Safe-OP與OP狀態)，Timeout為逾時等待時間，如出現無法狀態切換時可能等待時間過短。

```
ConfigDrive(1, 0, (TEST_DRV_CNT - 1), 1, N_AXIS_IN_ONE_DRV, 0x1602, 0x1A02);
```

或

```
RxPDOConfig[i].SmaIdx = RxPDO_ASSIGN_IDX;  
RxPDOConfig[i].PDOCnt = 1;  
RxPDOConfig[i].MapIdx[0] = RxPDO_MAP_IDX;  
RxPDOConfig[i].ObjsCnt[0] = 2;  
SetPdoConfigTbl(&RxPDOConfig[i], 0, 0, 0x6040, 0, 16); //control word // 16 bits = 2 bytes for TEST_RXPDO_SIZE  
// the 1st parameter is PDO_CONFIG_HEAD  
// the 2nd parameter is 0 due to RxPDOConfig.PDOCnt = 1  
// the 3rd parameter is 0 for the first object as RxPDOConfig.ObjsCnt[0] = 2  
// the 4th parameter is a control word index 0x6040  
// the 5th parameter is a sub-index for 0x6040  
// the 6th parameter is the bit size for 4th parameter  
SetPdoConfigTbl(&RxPDOConfig[i], 0, 1, 0x607A, 0, 32); //target position // 32 bits = 4 bytes for TEST_RXPDO_SIZE  
TxPDOConfig[i].SmaIdx = TxPDO_ASSIGN_IDX;  
TxPDOConfig[i].PDOCnt = 1;  
TxPDOConfig[i].MapIdx[0] = TxPDO_MAP_IDX;  
TxPDOConfig[i].ObjsCnt[0] = 2;  
SetPdoConfigTbl(&TxPDOConfig[i], 0, 0, 0x6041, 0, 16); //status word // 16 bits = 2 bytes for TEST_TXPDO_SIZE  
SetPdoConfigTbl(&TxPDOConfig[i], 0, 1, 0x6064, 0, 32); //actual position // 32 bits = 4 bytes for TEST_TXPDO_SIZE
```

```
-ECM_EcatReconfig();
```

8. 完成配置後進行ECM_CheckMEMSpace(TEST_PDO_FIFO_ONCE)，此步驟為確認各部分相關記憶體的大小，參數需輸入每次SPI/USB傳輸包含幾筆週期命令。

```
ECM_CheckMEMSpace(TEST_PDO_TO_FIFO_ONCE);
```

9. 檢查確認沒有問題後利用ECM_StateCheck(Slave, ExpectedState, Timeout)進入Safe-OP狀態

```
ECM_StateCheck(0xFF, EC_STATE_SAFE_OP, 1000);
```

10. 可進入Safe-OP狀態後繼續執行ECM_StateCheck(Slave, ExpectedState, Timeout)進入OP狀態
11. 進入OP狀態後執行ECM_CheckDCStable()確認DC狀態穩定

```
ECM_CheckDCStable();
```
12. 當確認完成後，以ECM_Drv402SM_StateSet(Axes, ServoOn/OffState)或ECM_Drv402SM_Enable(Axes, Slaves)啟用402 state machine至servo on階段，此階段為自動切換402 state machine的方式，如果從站無法使用以上兩種切換方式時，可以參考STF4HSP或NuHSP中的手動切換模式進行單步切換至Servo On狀態

```
ECM_Drv402SM_StateSet((DrvIdx*N_AXIS_IN_ONE_DRV) + AxisIdx, SERVO_ON_STATE);
```

```
PdoExchangeAndGet402State(DrvIdx, AxisIdx, &u8State);
```

或

```
ECM_Drv402SM_Enable(0, 0);
```
13. Servo On完成後若為CSP模式，必須將編碼器(實際位置)與命令的目標位置對齊，可利用AlignmentPosition()，並使用ECM_InitFIFO()初始化FIFO同時使用ECM_ClearFIFO(Direction)將FIFO內部清空後以ECM_EnableFIFO(Enable)來啟用FIFO功能。

```
ECM_ClearFIFO(0), 0為Tx與Rx
```

```
ECM_EnableFIFO(1), 1為Enable、0為Disable
```

```
ECM_InitFIFO();
```

```
ECM_ClearFIFO(0); // 0 for TX and RX both
```

```
PRINTF("ClearFIFO\r\n");
```

```
ECM_EnableFIFO(1); // Enable FIFO
```

```
PRINTF("EnableFIFO\r\n");
```
14. 完成上方步驟後已完成相關相關設定，接下來的資料交換皆會在while迴圈中不斷重複運行，以判斷FIFO的數量為基準，如果FIFO已經快要塞滿時，則選擇暫停塞入命令，此外持續將命令塞入FIFO，速度與位置相關設定在Utility.c中可依需求進行編寫。

常見錯誤排除

在執行相關設定時如果跳出Error相關問題，請依以下進行排除

階段	顯示	解決方法
ECM_Init_Library	wait ASYNC done timeout	請確認主站與從站連接是否正常，主站與從站網口燈號是否亮起
ECM_Init_Library	u8ErrorStatus 0x40	初次設定會跳出的警告，Novoton系列按Enter繼續
通期	wait ASYNC done timeout	等待時間不足或從站錯誤
通期	CRC Error	SPI傳輸訊號錯誤，請確認SPI主站與EtherCAT主站

		IC連線
--	--	------

Drive 402自動切換與手動切換

Drive 402 指的是驅動器上的電源控制 (servo on、激磁或使能)，使用者可以透過 Control Word (0x6040) 控制，並透過 Status Word (0x6041) 來確認狀態。另一個方式是透過 ECM_CMD_402_CONFIG_SET 來指定 Control Word 及 Status Word 在該子站的位置的偏移量 (offset)，隨後再透過 ECM_CMD_402_STATE_SET 來設定希望到達的電源狀態。

- 在 STF4Drive 範例中用 ECM_Drv402SM_StateSet() 與 PdoExchangeAndGet402State() 來進行 Servo ON
- 其他範例中以 ECM_Drv402SM_Enable() 的方式進行 Servo On
ECM_Drv402SM_Enable 中的參數分別為軸編號 (TbIdx) 與從站編號 (SlvIdx) 所以當為一對一控制器時兩站 Enable 就會變成
ECM_Drv402SM_Enable(0, 0);
ECM_Drv402SM_Enable(1, 1);
如果為一對二控制器且有兩站時命令就會變成
ECM_Drv402SM_Enable(0, 0); // 第0站第0個軸，整體的第0軸
ECM_Drv402SM_Enable(1, 0); // 第0站第1個軸，整體的第1軸
ECM_Drv402SM_Enable(2, 1); // 第1站第0個軸，整體的第2軸
ECM_Drv402SM_Enable(3, 1); // 第1站第1個軸，整體的第3軸
- 手動切換的方式請參考下面程式碼，手動切換是透過 Control Word 來切換

```
for(DrvIdx=0;DrvIdx<TEST_SLAVE_CNT;DrvIdx++){
    for(AxisIdx=0;AxisIdx<N_DRV_IN_ONE_SLV;AxisIdx++){
        j = 0;
        while(1){
            nret = ECM_EcatPdoFifoDataExchange(PDO_FIFO_DEFAULT_CNT, RxData, TxData, u16PDOSize, &u8FifoCnt, &u8WkcErrCnt, &u8CrcErrCnt);
            if(nret>0){
                u16LogStatus[(DrvIdx*N_DRV_IN_ONE_SLV)+AxisIdx] = pTxPDDData->DRIVE_GROUP_0[DrvIdx].HSP[AxisIdx].u16StaWord;

                PDOstate[(DrvIdx*N_DRV_IN_ONE_SLV)+AxisIdx] = pTxPDDData->DRIVE_GROUP_0[DrvIdx].HSP[AxisIdx].u16StaWord & CIA402_SW_STATE_MASK;
                PRINTF("DrvIdx = %d, AxisIdx = %d, state = 0x%x\r\n", DrvIdx, AxisIdx, PDOstate[(DrvIdx*N_DRV_IN_ONE_SLV)+AxisIdx]);

                if(PDOstate[(DrvIdx*N_DRV_IN_ONE_SLV)+AxisIdx]==CIA402_SW_NOTREADYTOSWITCHON){
                    UserDelay(1000);
                }
                if(PDOstate[(DrvIdx*N_DRV_IN_ONE_SLV)+AxisIdx]==CIA402_SW_SWITCHEDONDISABLED){
                    pRxPDDData->DRIVE_GROUP_0[DrvIdx].HSP[AxisIdx].u16CtrlWord = 0x6; //Control word: Shutdown = 0x6
                }
                if(PDOstate[(DrvIdx*N_DRV_IN_ONE_SLV)+AxisIdx]==CIA402_SW_READYTOSWITCHON){
                    pRxPDDData->DRIVE_GROUP_0[DrvIdx].HSP[AxisIdx].u16CtrlWord = 0x7; //Control word: Switch on = 0x7
                }
                if(PDOstate[(DrvIdx*N_DRV_IN_ONE_SLV)+AxisIdx]==CIA402_SW_SWITCHEDON){
                    pRxPDDData->DRIVE_GROUP_0[DrvIdx].HSP[AxisIdx].u16CtrlWord = 0xF; //Control word: Enable operation = 0xF
                }
                if(PDOstate[(DrvIdx*N_DRV_IN_ONE_SLV)+AxisIdx]==CIA402_SW_OPERATIONENABLED){
                    j++;
                    if(j==3){
                        break;
                    }
                }
                if(PDOstate[(DrvIdx*N_DRV_IN_ONE_SLV)+AxisIdx]==CIA402_SW_QUICKSTOPACTIVE){
                    pRxPDDData->DRIVE_GROUP_0[DrvIdx].HSP[AxisIdx].u16CtrlWord = 0x0; //Control word: Disable voltage = 0x0
                }
                if(PDOstate[(DrvIdx*N_DRV_IN_ONE_SLV)+AxisIdx]==CIA402_SW_FAULTREACTIONACTIVE){
                    UserDelay(1000);
                }
                if(PDOstate[(DrvIdx*N_DRV_IN_ONE_SLV)+AxisIdx]==CIA402_SW_FAULT){
                    pRxPDDData->DRIVE_GROUP_0[DrvIdx].HSP[AxisIdx].u16CtrlWord = 0x0; //Control word: Fault reset = 0x0->0x80
                    nret = ECM_EcatPdoFifoDataExchange(PDO_FIFO_DEFAULT_CNT, RxData, TxData, u16PDOSize, &u8FifoCnt, &u8WkcErrCnt, &u8CrcErrCnt);
                    UserDelay(1000);
                    pRxPDDData->DRIVE_GROUP_0[DrvIdx].HSP[AxisIdx].u16CtrlWord = 0x80;
                    nret = ECM_EcatPdoFifoDataExchange(PDO_FIFO_DEFAULT_CNT, RxData, TxData, u16PDOSize, &u8FifoCnt, &u8WkcErrCnt, &u8CrcErrCnt);
                    UserDelay(1000);
                }
            }
        }
    }
}
```


維護紀錄

版本	日期	說明
01	2021.02.09	綜合ECM-XF-MCU User Guide, Sample Code Explanation, and STM32 Setup Instruction(NUCLEO-F401RE)
	2021.02.17	修改新唐環境建立導引說明
02	2021.05.06	更新STF4Drive與NuDrive
03	2021.06.28	更新範例修改指引
04	2021.06.29	修正修改指引細項
05	2021.08.17	新增STM32 H7腳位設定
06	2022.12.25	修正範例說明