透過MU課程的綜合應用來幫助您快速解鎖 dsPIC33CH 雙核 MCU 的強大功能

A Leading Provider of Smart, Connected and Secure Embedded Control Solutions

dsPIC33CH 雙核 MCU 開發範例下載

- https://www.microchip.com.tw/modules/tad_uploader/index.php?of_cat_sn=32

檔案名稱	日期	大小
PIC33CH_MailCore_Project	2024-01-22 17:16:08	365.6 KB
dsPIC33CH_SecondaryCore_Project	2024-01-22 17:16:33	231.5 KB

介紹

- 本研討會內容涵蓋 **dsPIC33CH** 雙核系列數位信號控制器(DSC)的架構。主要對於**Microchip** 的單核微控制器架構有一定經驗的開發人員，介紹 雙核 **dsPIC33CH** 的特性和優勢，內容包括基本架構和週邊、共享資源、每個核運行時的應用程序和啟動以及 二核間的通信。 我們還將如何利用不同的週邊，來加速需即時運算的嵌入式控制應用程序、提高安全應用程序的可靠性並降低總體應用程序成本。

介紹

- **MU 課程參考：**
  - MPLAB® X IDE介紹
  - MPLAB® Code Configurator 介紹
  - dsPIC33CH 雙核 架構
  - dsPIC33CH 雙核燒錄和除錯
- **Demoboard：**
  - APP-ALL MCU 2023

dsPIC33CH簡介

- **高性能核心：**
  - dsPIC33CH擁有16位的高性能核心，支持多數據操作（SIMD），具有優越的數字訊號處理能力。
- **Dual Core：**
  - dsPIC33CH採用雙核心架構，包括主核（Master Core）和副核（Slave Core），可以同時執行多個任務，實現多核共同工作。

dsPIC33CH簡介

- **多種Timers：**
  - 這些器件配備了多個高精度的Timers，用於控制定時、計數、PWM生成等應用。
- **類比訊號處理：**
  - dsPIC33CH具有多個高解析度的類比到數位轉換器（ADC）通道和數位到類比轉換器（DAC），支持精確的類比訊號處理。
- **多種型號：**
  - dsPIC33CH系列有多種型號，以滿足不同性能和功能需求。

dsPIC33CH簡介

- 溝通介面：
  - dsPIC33CH支持多種溝通介面，如UART、SPI、I2C、CAN，使其能夠連接到其他設備進行溝通。
- 開發工具：
  - Microchip提供了完整的開發工具，包括集成開發環境（IDE）、Debugger、Demoboard等，使開發過程更加簡單。

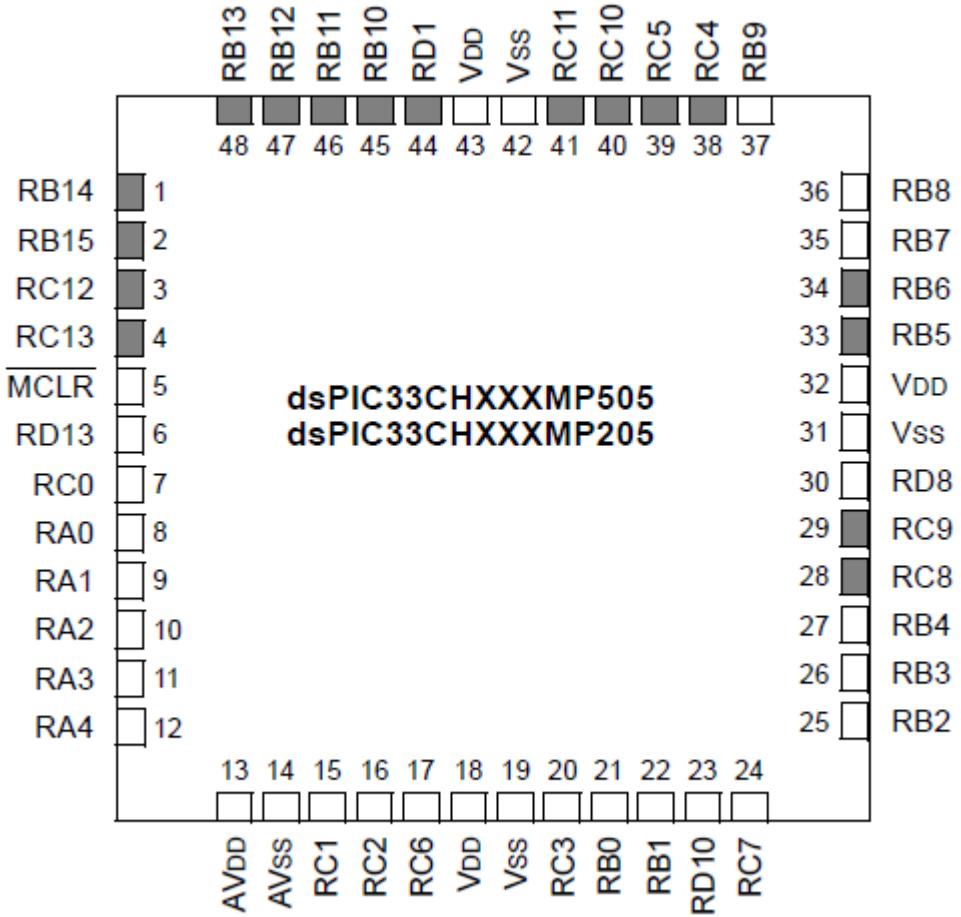
架構和功能

dsPIC33CH Dual Core

MICROCHIP

架構和功能

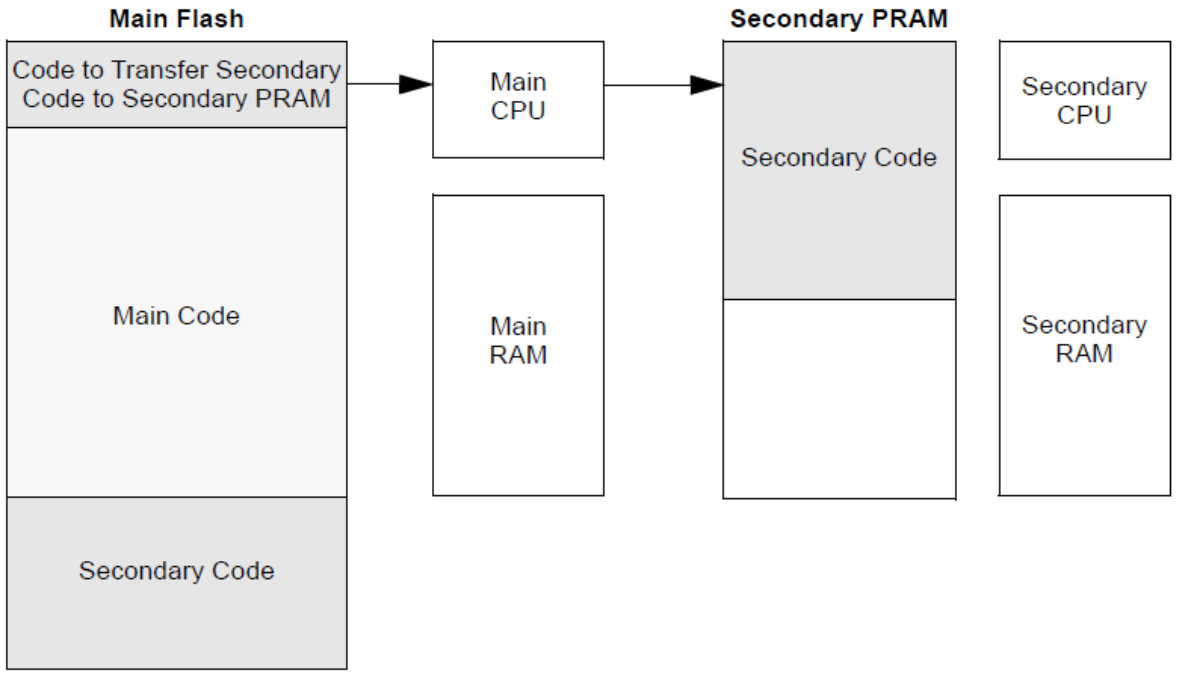
dsPIC33CH256MP505 Pin Diagrams(48-Pin TQFP/UQFN)



架構和功能

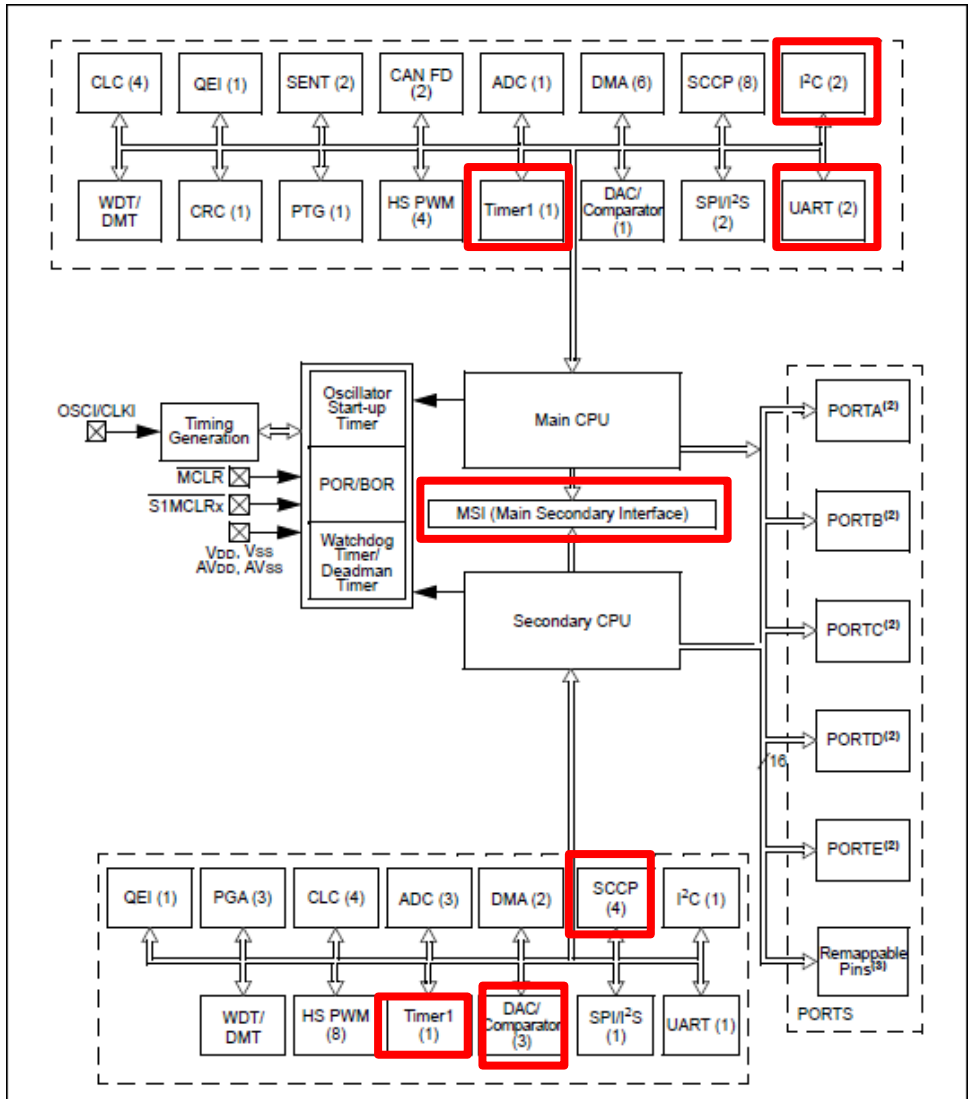
Main Flash 與 Secondary PRAM 關係

- **After a POR, the Main Loads the Code to the Secondary PRAM and then Enables the Secondary to Start Executing the Code:**



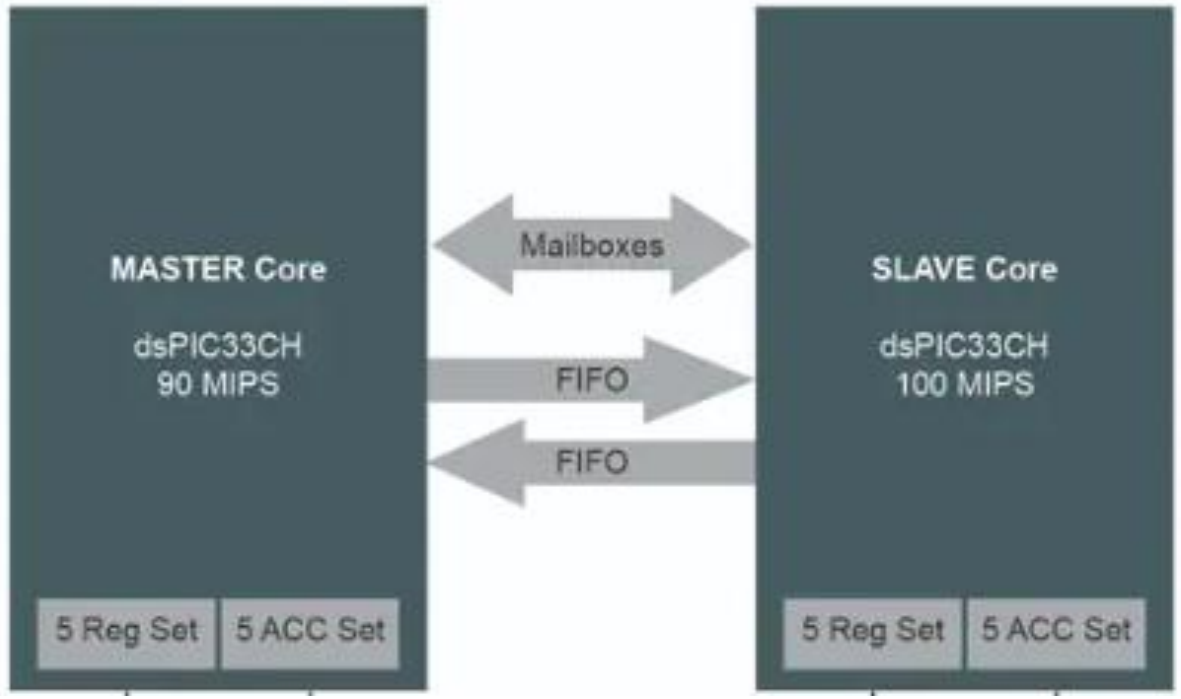
架構和功能

dsPIC33CH512MP508 FAMILY BLOCK DIAGRAM



架構和功能

Difference from Mailboxes and FIFO



MICROCHIP

環境設置

Main & Secondary

環境設置

MPLAB X IDE/ XC16 Compiler/ MCC

- **MPLAB X IDE 6.15 (Windows) Download**

https://www.microchip.com/en-us/toolsresources/develop/mplab-x-ide#tabs

Title		Version Number	Date	
MPLAB X IDE (Windows)	9927b8ef... 217e	6.15	10 Aug 2023	Download
MPLAB X IDE (Linux)	6628a28e... 4c61	6.15	10 Aug 2023	Download
MPLAB X IDE (macOS)	bcf010bf... 578a	6.15	10 Aug 2023	Download
MPLAB X IDE Release Notes		6.15	10 Aug 2023	Download

Notes about MPLAB X IDE

MICROCHIP

環境設置

MPLAB X IDE/ XC16 Compiler/ MCC

- **XC16 Compiler 2.10 (Windows) Download**

https://www.microchip.com/en-us/tools-resources/develop/mplab-xc-compilers/xc16

MPLAB XC16 Compiler Downloads

Title		Version Number	Date	
MPLAB XC16 C-Compiler (Windows)	0fc3beb4... 7d2d	2.10	18 Apr 2023	Download
MPLAB XC16 C-Compiler (macOS)	db0ac553... de55	2.10	18 Apr 2023	Download
MPLAB XC16 C-Compiler (Linux)	d64d5e73... 5dc8	2.10	18 Apr 2023	Download

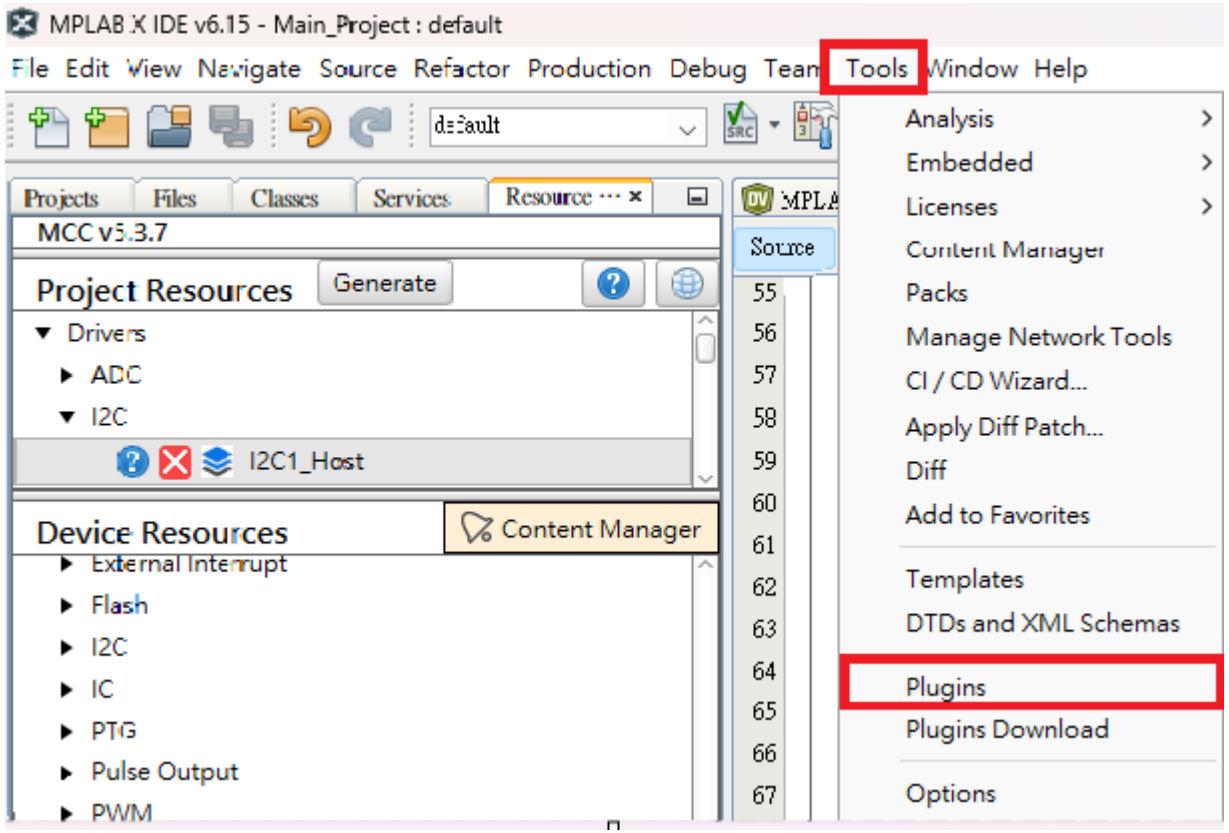
MICROCHIP

環境設置

MPLAB X IDE/ XC16 Compiler/ MCC

- **MCC 5.3.7 Install**

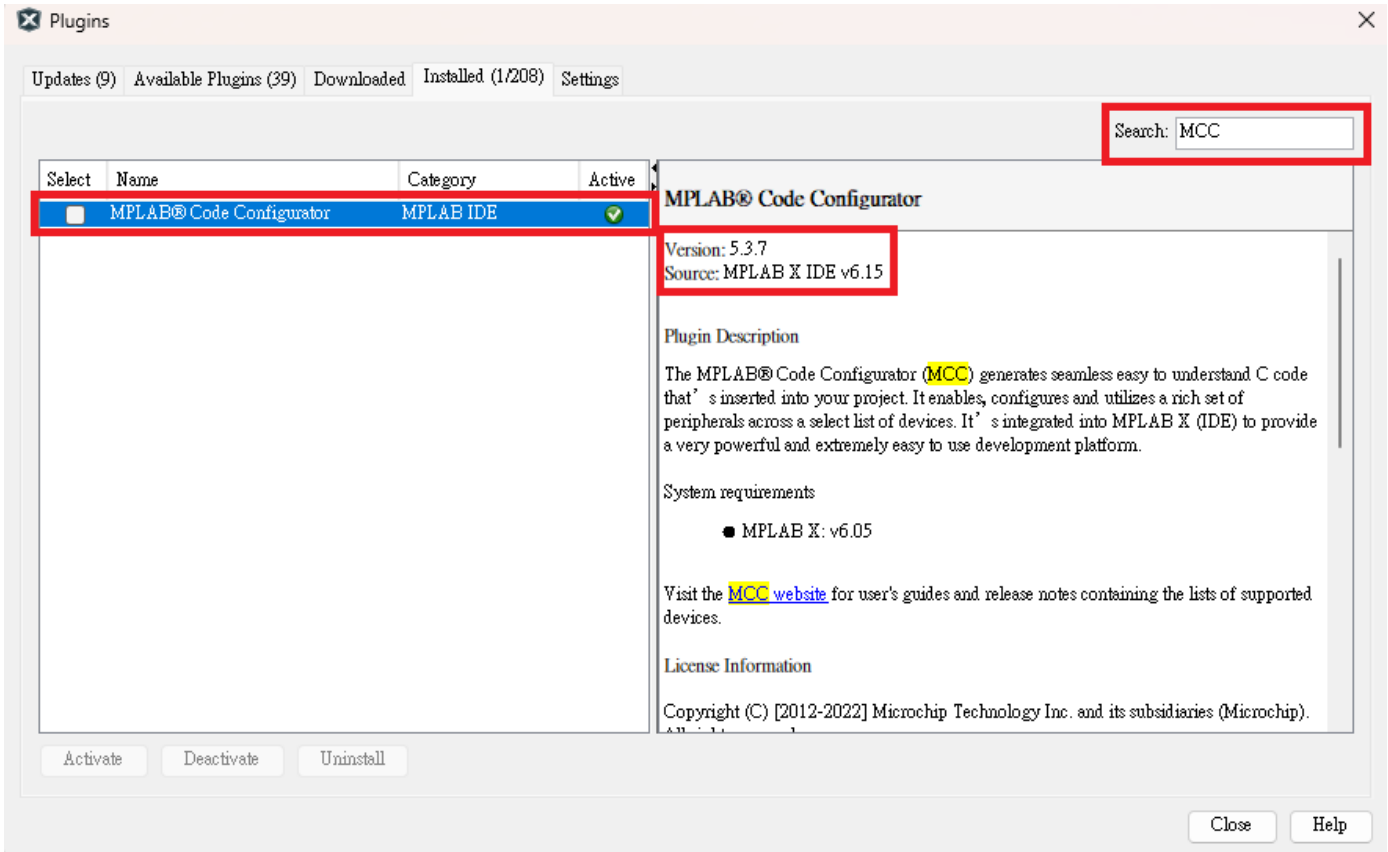
- **Tools -> Plugins**



環境設置

MPLAB X IDE/ XC16 Compiler/ MCC

- **Search : MCC -> Install and Update**



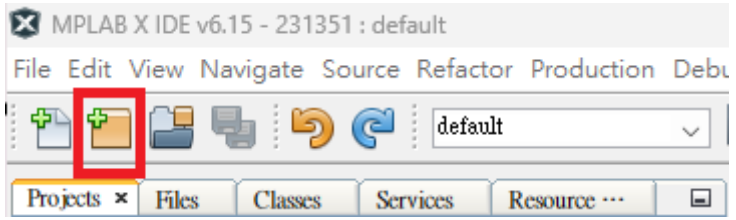
Start Dual Core

Main & Secondary Core

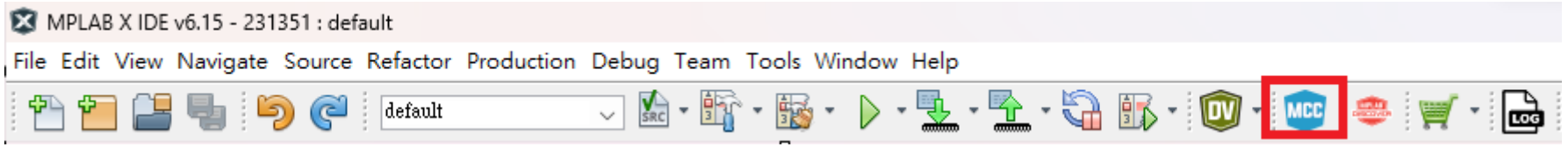
Start Dual Core

Main Core

- 先建Main Project



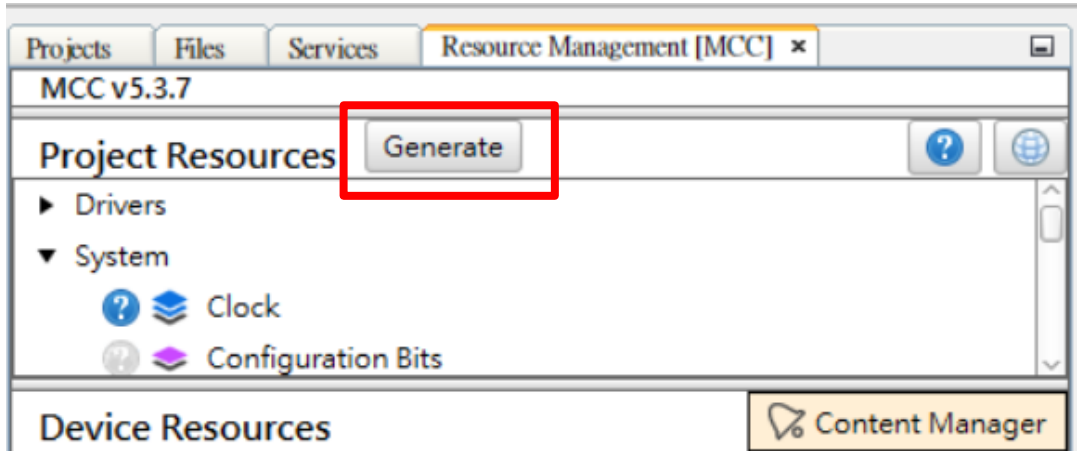
- 打開MCC(Melody)



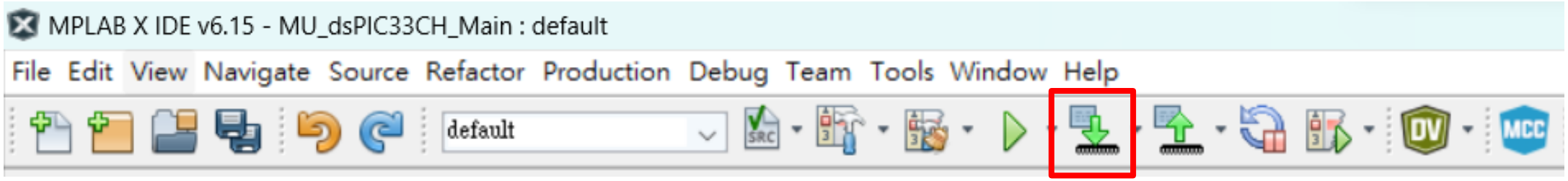
Start Dual Core

Main Core

- **MCC Gen code**



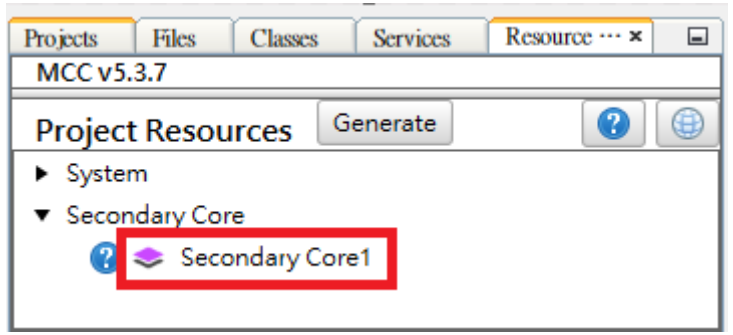
- 燒錄空程式到板子，確保燒錄沒問題



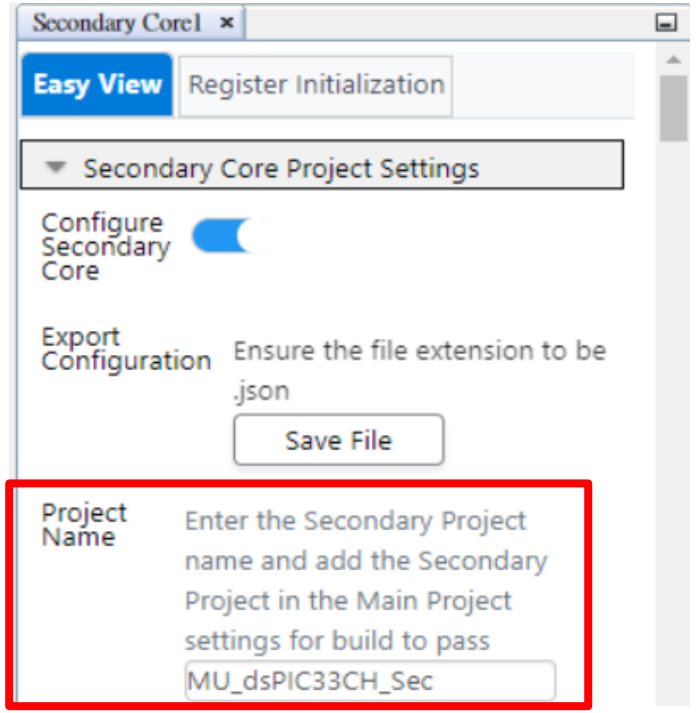
Start Dual Core

Main Core

- MCC左上的視窗找到Secondary Core1



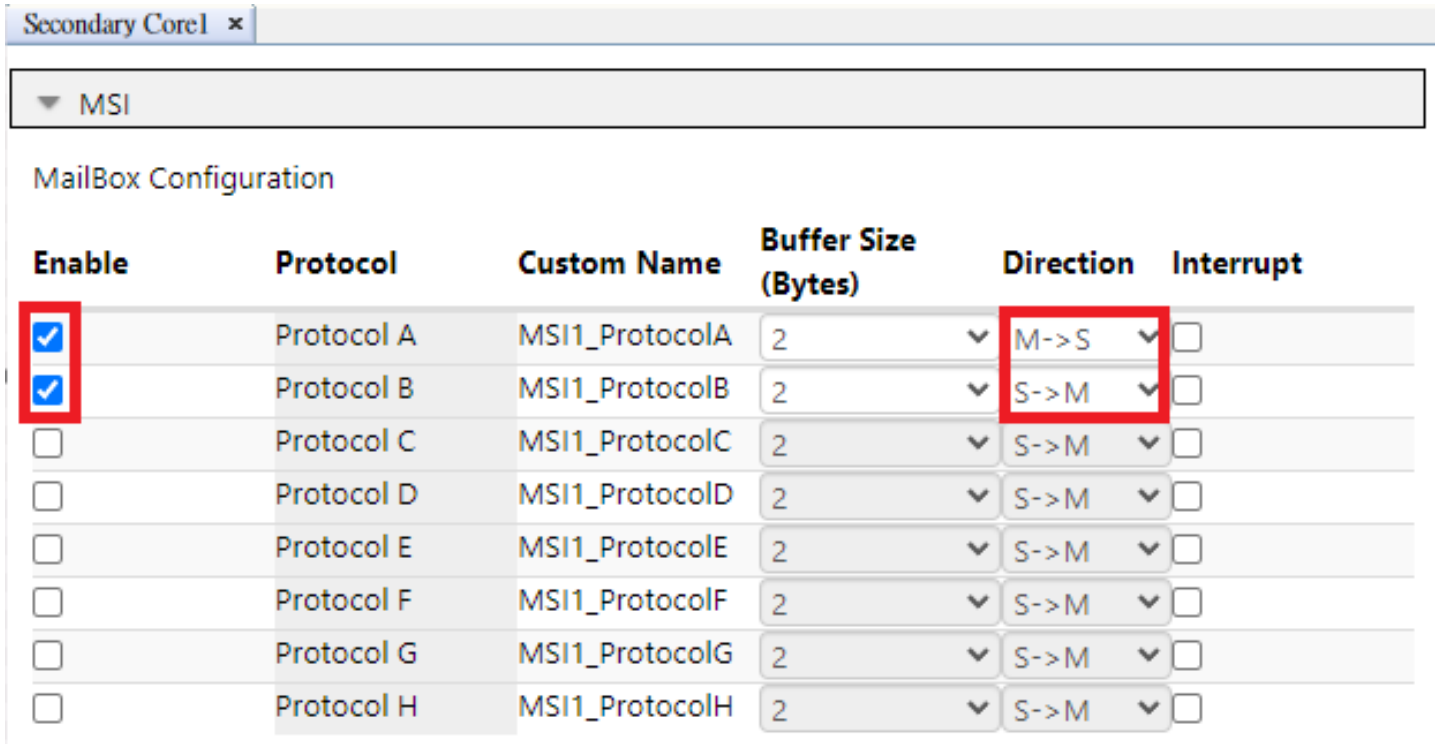
- 填Secondary Project的名稱



Start Dual Core

Main Core

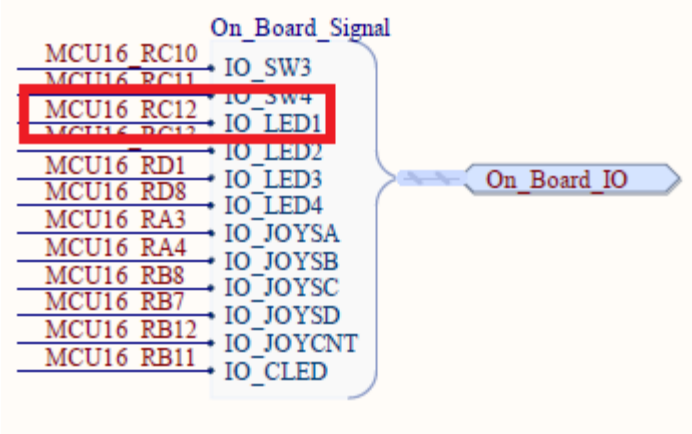
- 勾PA選M->S 以及 勾PB選S->M



Start Dual Core

Main Core

- **LED1，使用RC12作為輸出**



Output | Notifications [MCC] | Pin Grid View

Package:	TQFP48	Pin No:	8	9	10	11	12	21	22	25	26	27	33	34	35	36	37	45	46	47	48	1	2	7	15	16	20	38	39	17	24	28	29	40	41	3
			PORTA					PORTB																PORTC													
Module	Function	Direction	0	1	2	3	4	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7	8	9	10	11	12
Clock	CLKO	output																																		
	REFI	input																																		
	REFO	output																																		
ICD	PGCx	input																																		
	PGDx	input																																		
	GPIO	input																																		
Pins	GPIO	output																																		

MICROCHIP

Start Dual Core

Main Core

- **Pins -> RC12以LED1命名，並勾選Start High**

Pins ×

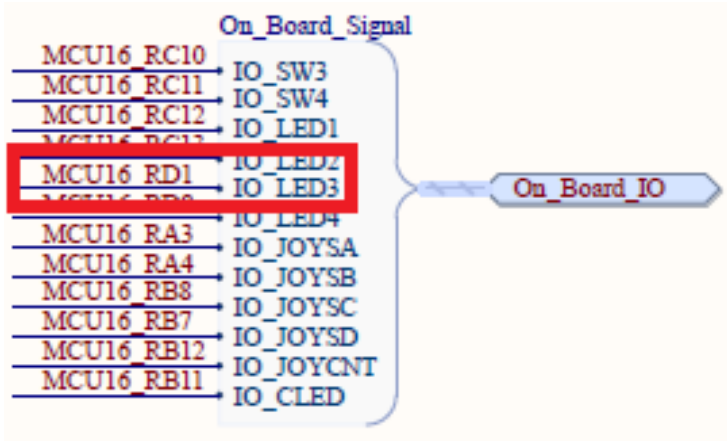
Location	Pin Name	Module	Function	Direction	Custom Name	Analog	Start High
3	RC12	Pins	GPIO	output	LED1		☑

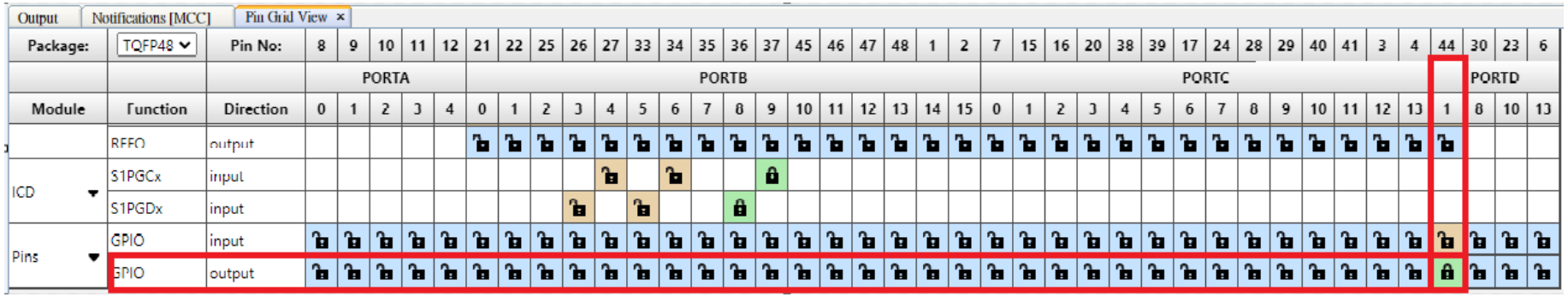
MICROCHIP

Start Dual Core

Main Core

- **LED3，使用RD1作為輸出**

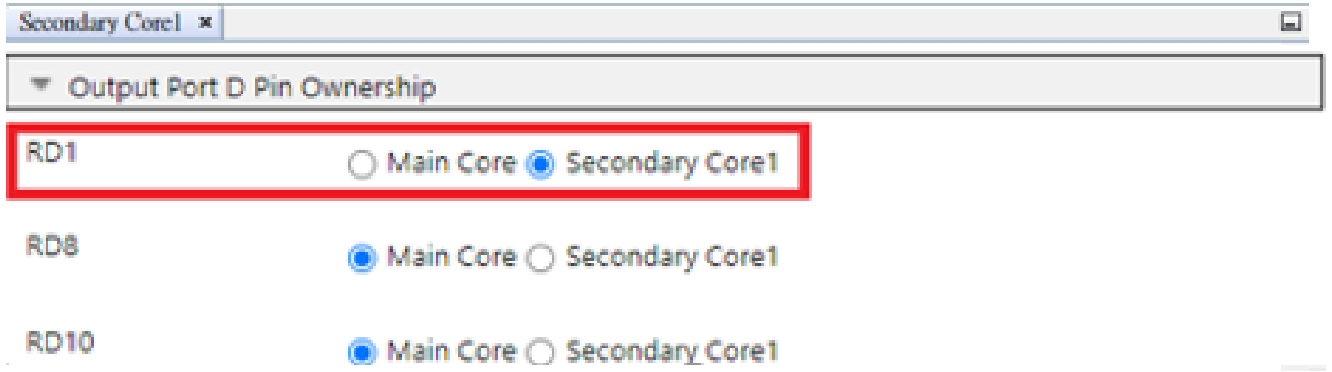




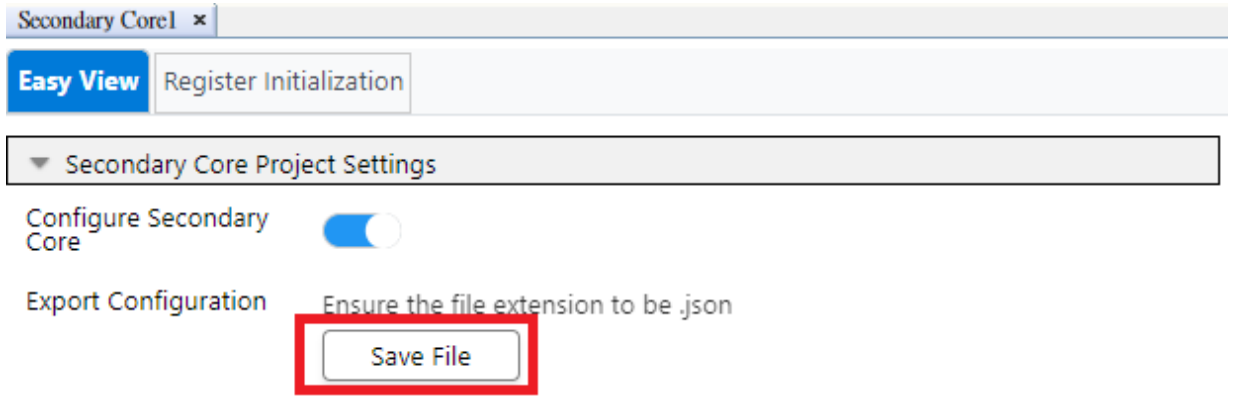
Start Dual Core

Main Core

- **MCC(MU_dsPIC33CH_Main)設定使用RD1作為Secondary Core的輸出**



- **最後Save File 以json 的形式儲存在Main Project的.X資料夾中**



Start Dual Core

Main Core

- **Gen Code後相關的.h檔**

```
#include "mcc_generated_files/system/system.h"
#include "mcc_generated_files/secondary_core/sec_core1.h"
#include "mcc_generated_files/system/pins.h"
```

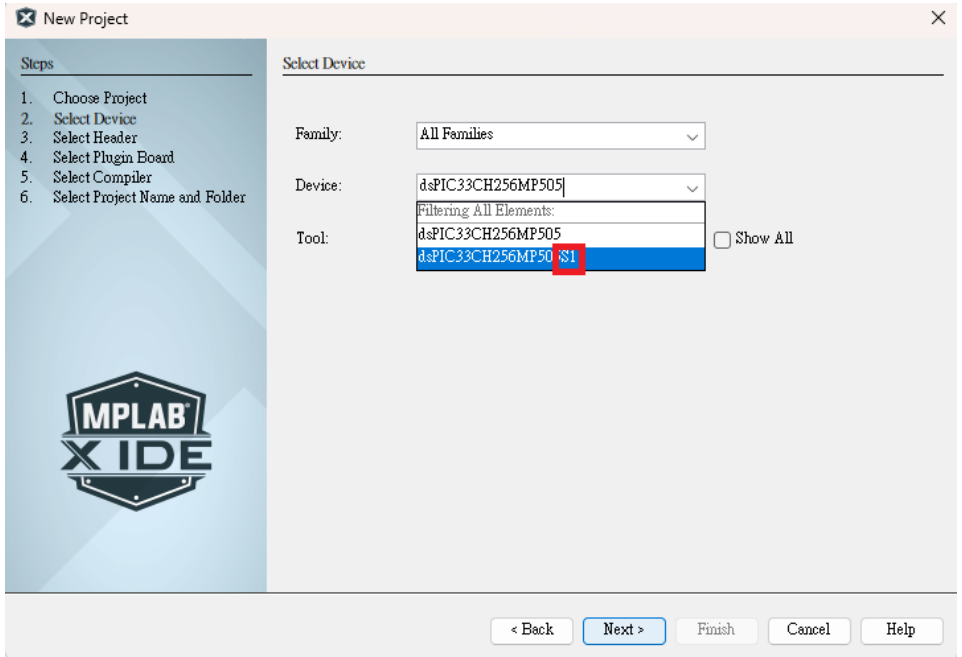
- 主程式

```
SYSTEM_Initialize();
SEC_CORE1_Initialize();
SEC_CORE1_Program();
SEC_CORE1_Start();
```

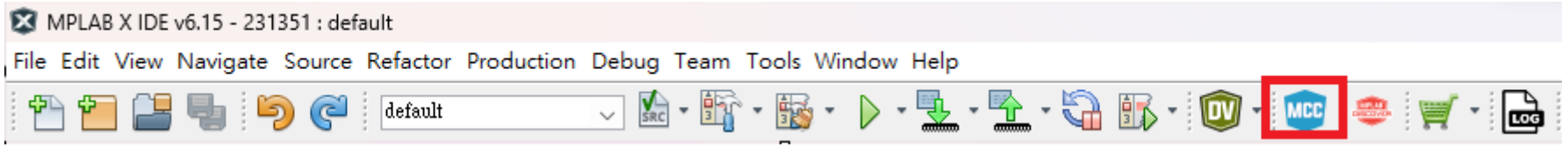
Start Dual Core

Secondary Core

- 建一個Secondary Project(S1)



- 打開MCC(Melody)



Start Dual Core

Secondary Core

- **Pins -> RD1用LED3命名 -> Start high**

Pins ×

Location	Pin Name	Module	Function	Direction	Custom Name	Analog	Start High
44	RD1	Pins	GPIO	output	LED3		☑

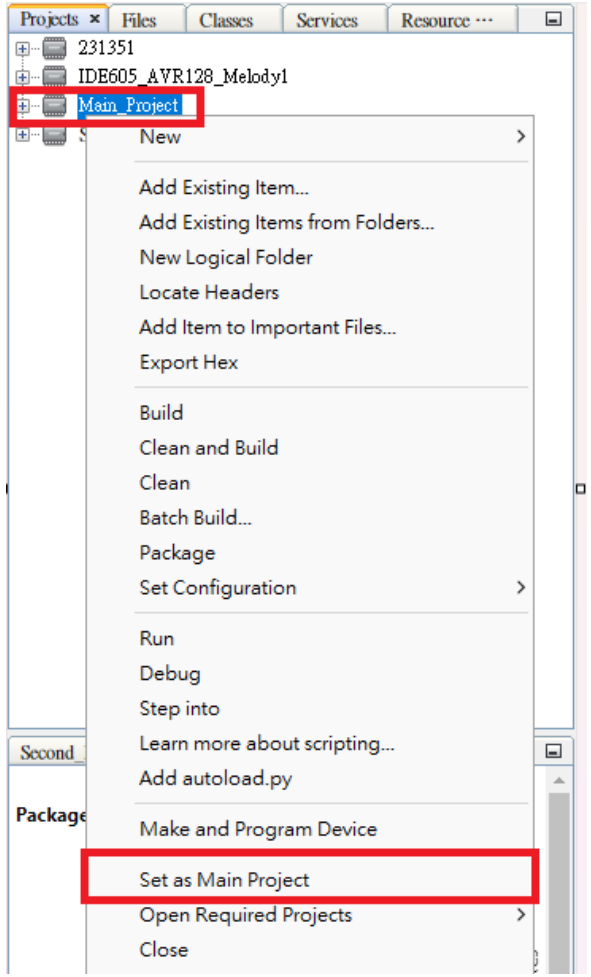
- **Gen Code後相關的.h檔**

```
#include "mcc_generated_files/system/system.h"
#include "mcc_generated_files/secondary_core/sec_core1.h"
#include "mcc_generated_files/system/pins.h"
```

Start Dual Core

Secondary Core

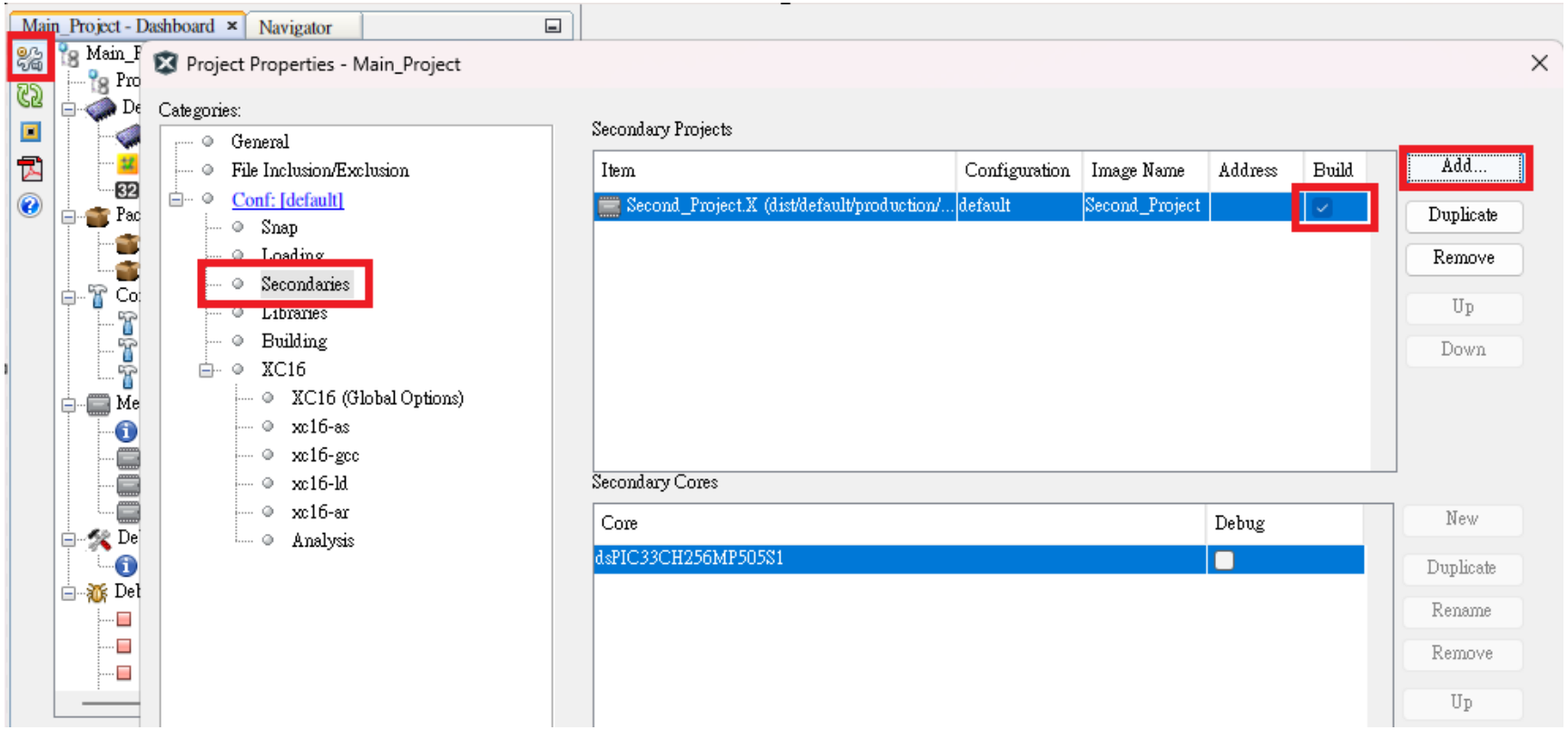
- **Secondary Project Gen Code和Build後不燒錄**
- **把MU_dsPIC33CH_Main設為Main project**



Start Dual Core

Secondary Core

- **MU_dsPIC33CH_Main的Project Properties把Secondaries加入Secondary project並且勾選Build**



Start Dual Core

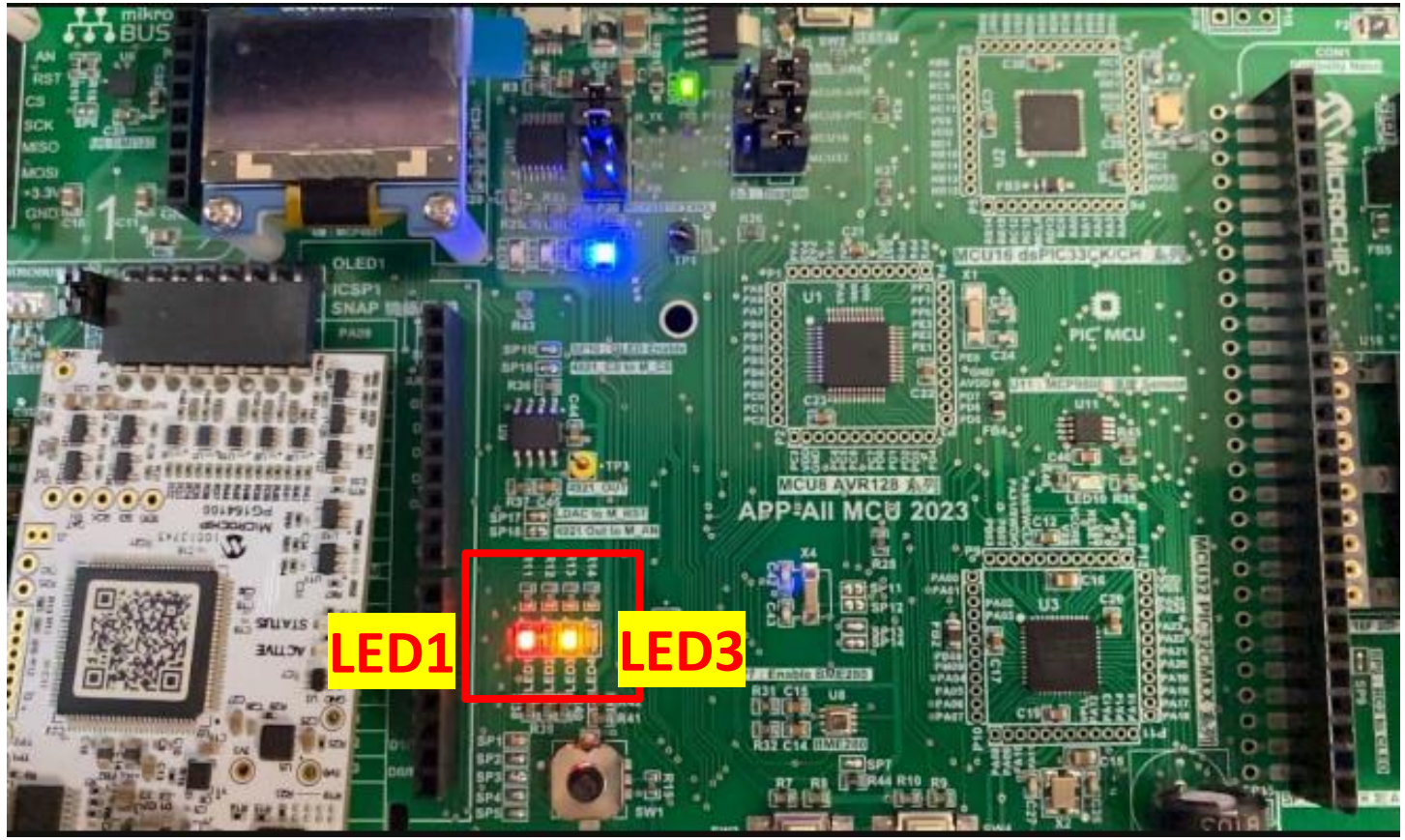
Main Core

- 之後我們只要燒錄主核的程式，副核會連帶一起燒錄，副核只需要做**Gen Code** 以及**Clean and Build**即可

Start Dual Core

Main Core

- LED燈



Timer & Interrupt

Main & Secondary Core

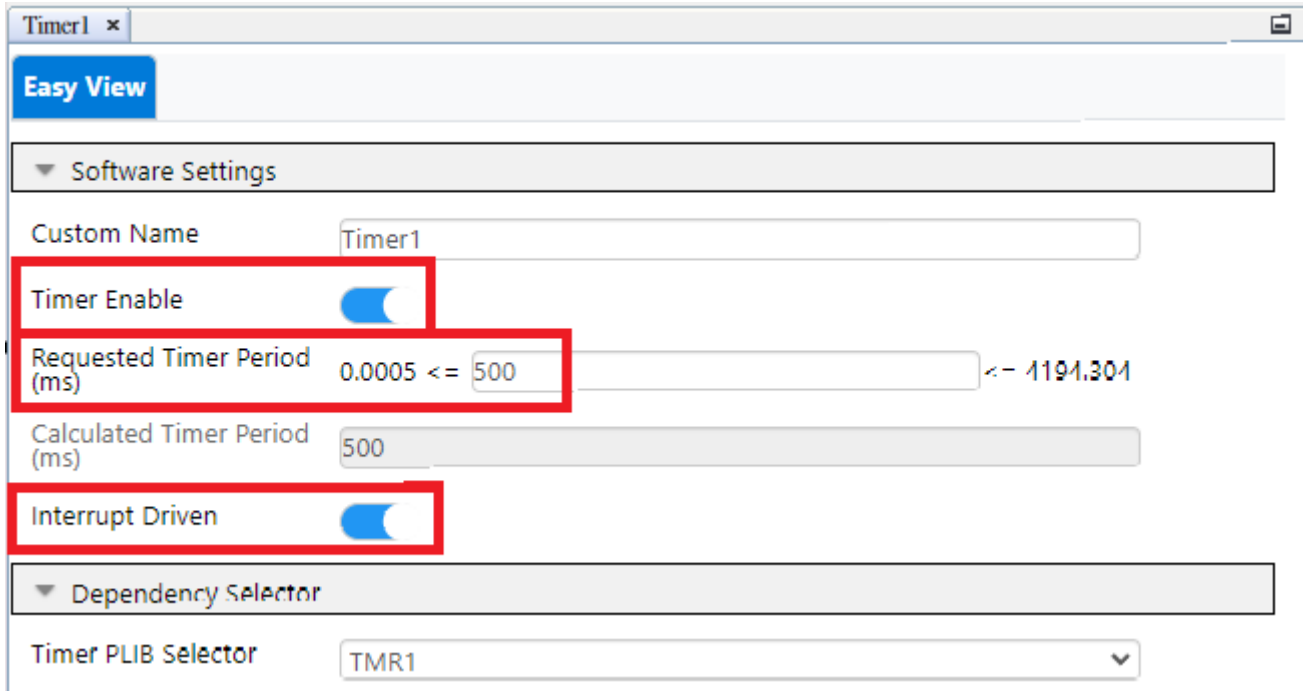
MICROCHIP

Timer & Interrupt

Main Core

- **MCC->Drivers->Add Timer**

- **設定每500ms進一次中斷**



Timer & Interrupt

Main Core

- **MCC Gen Code後相關的.h檔**

```
#include "mcc_generated_files/system/pins.h"
#include "mcc_generated_files/timer/tmr1.h"
```

- **主程式用中斷的Callback Function去呼叫Main_Interrupt 這個Function**

```
TMR1_TimeoutCallbackRegister(Main_Interrupt);
```

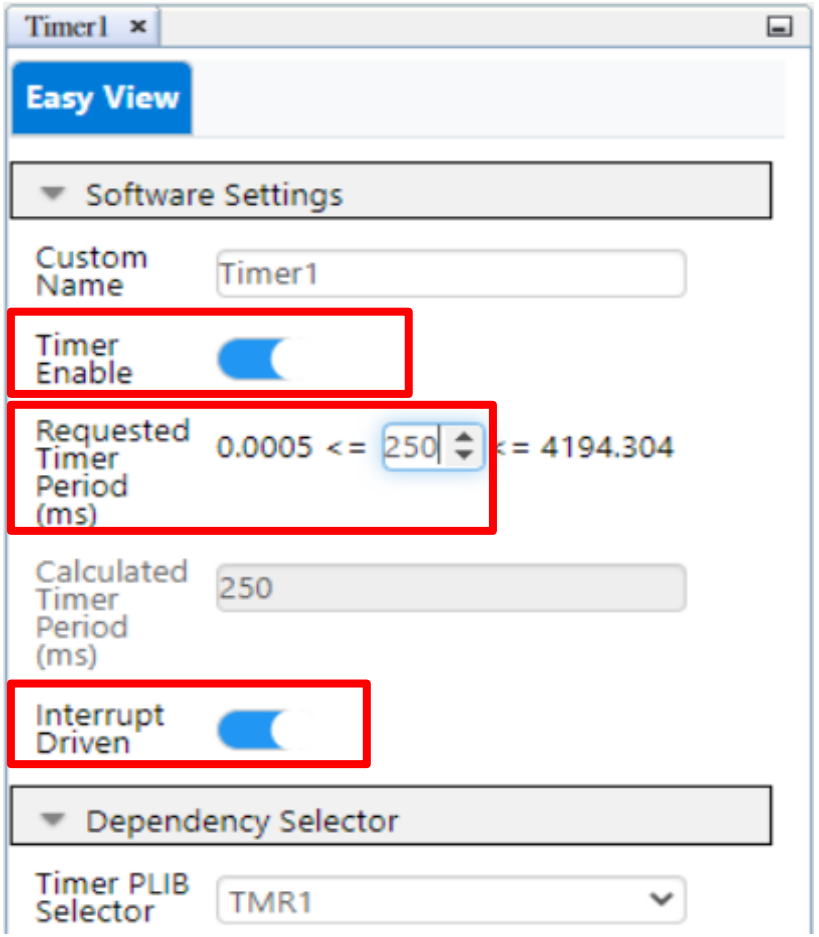
```
void Main_Interrupt(void)
{
    LED1_Toggle();
}
```

Timer & Interrupt

Secondary Core

- **MCC->Drivers->Add Timer**

- **設定每250ms進一次中斷**



Timer & Interrupt

Secondary Core

- **MCC Gen Code後相關的.h檔**

```
#include "mcc_generated_files/system/pins.h"
#include "mcc_generated_files/timer/tmr1.h"
```

- **主程式用中斷的Callback Function去呼叫Sec_Interrupt 這個Function**

```
TMR1_TimeoutCallbackRegister(Sec_Interrupt);
```

```
void Sec_Interrupt(void)
{
    LED3_Toggle();
}
```

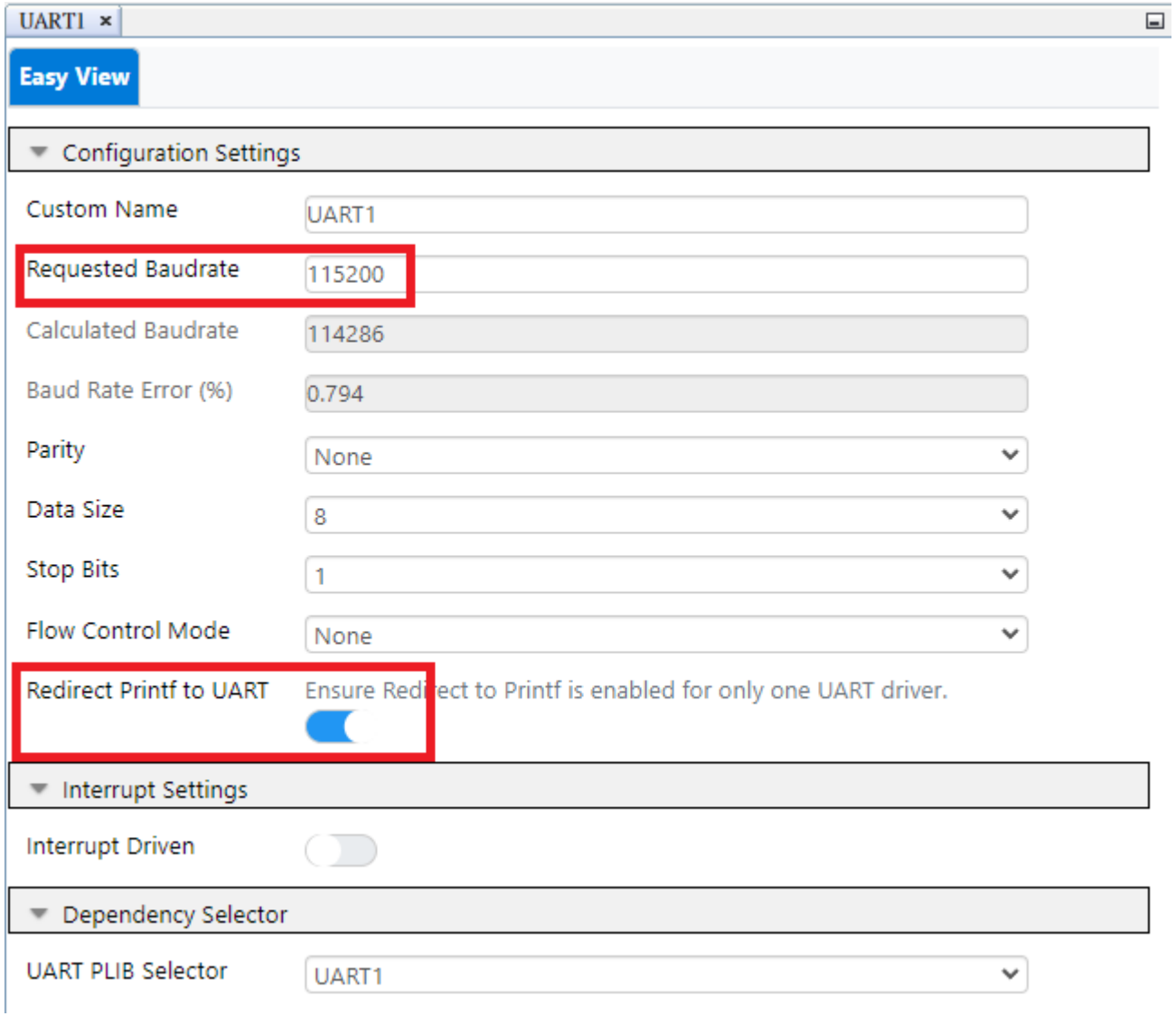
UART

Main Core

UART

Main Core

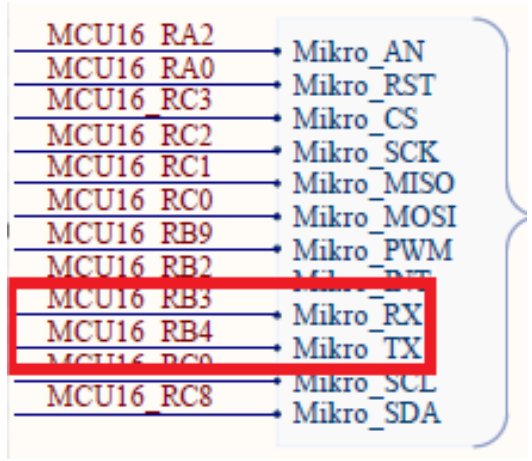
- MCC->Drivers->Add UART
- Baudrate設定115200
- 允許UART使用printf



UART

Main Core

- 設定UART的RB3為輸入，RB4為輸出



Output | Notifications [MCC] | Pin Grid View ×

Package:	TQFP48	Pin No:	8	9	10	11	12	21	22	25	26	27	33	34	35	36	37	45	46	47	48	1	2	7	15	16	20	38	39	17	24	28	29	40	41	3	4	44	30	23	6
			PORTA										PORTB											PORTC															PORTD			
Module	Function	Direction	0	1	2	3	4	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7	8	9	10	11	12	13	1	8	10	13
UART1	U1TX	output																																							
	U1RX	input																																							

MICROCHIP

UART

Main Core

- **MCC Gen Code 後相關的.h檔，以及手動 include <stdio.h>**

```
#include "mcc_generated_files/uart/uart1.h"
#include <stdio.h>
```

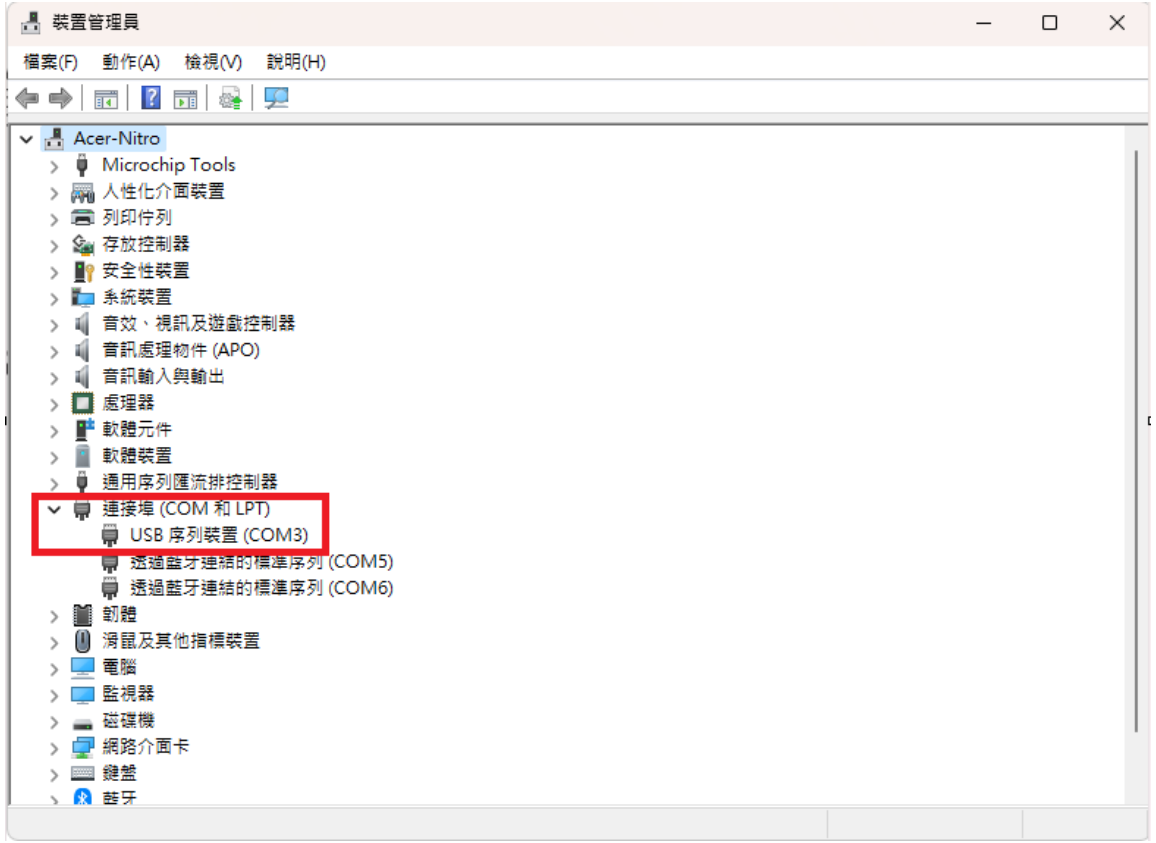
- **主程式 - While(1) Loop**

```
while(1)
{
    printf("Hello World \r\n");
}
```

UART

Main Core

- **Windows裝置管理員 -> 確認COM Port**



UART

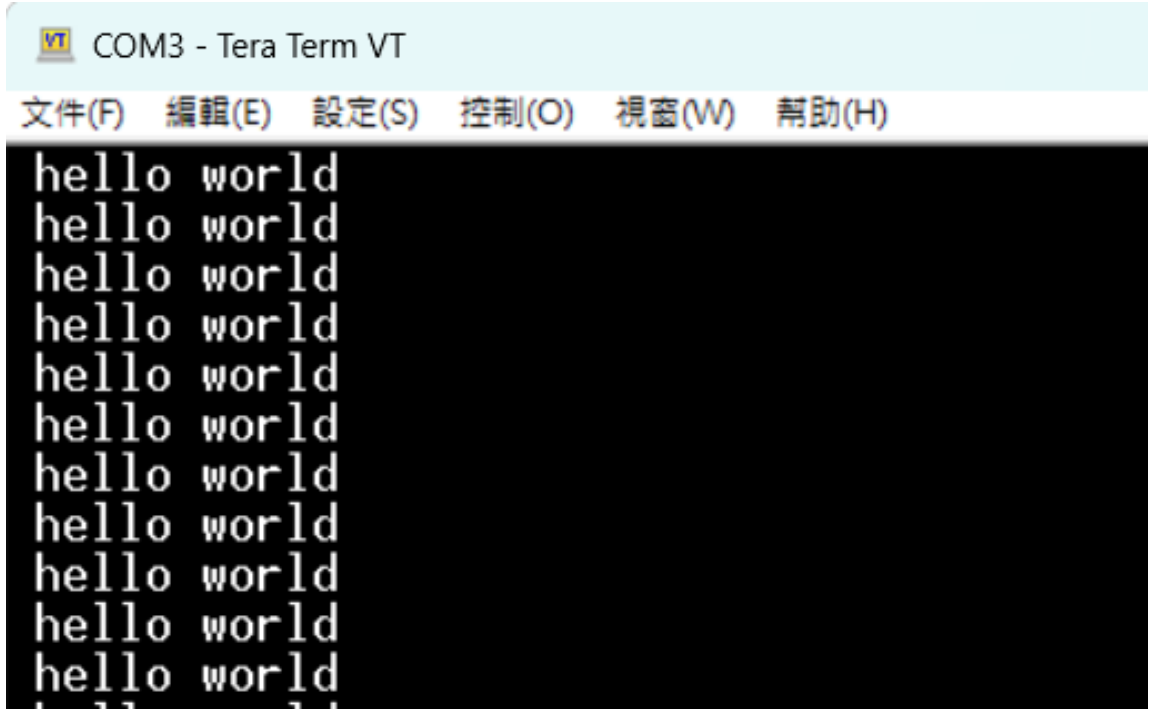
Main Core

- **Tera term -> COMx -> Boudrate：115200**

UART

Main Core

- **Terminal**



MICROCHIP

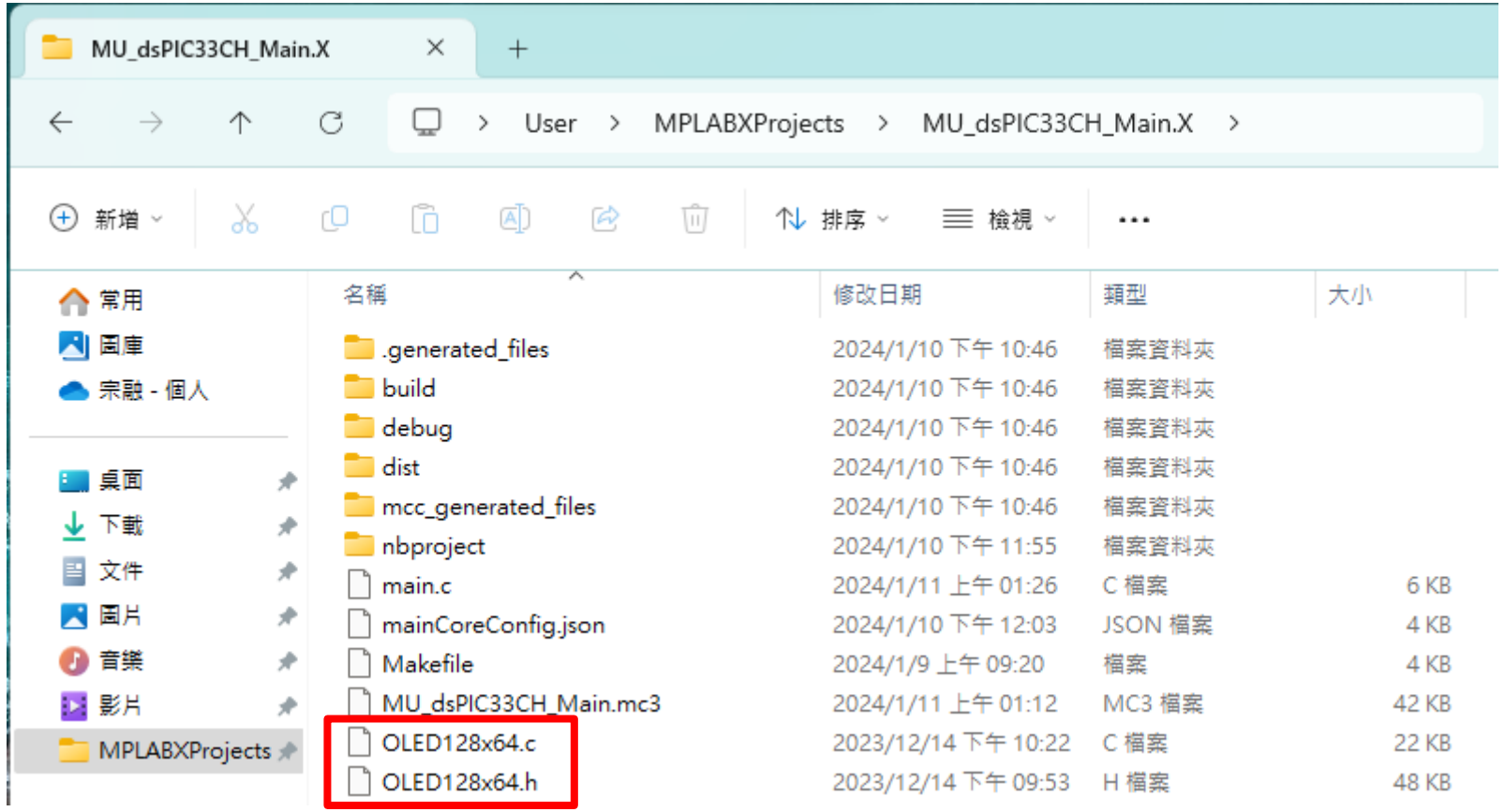
I2C(Temperature Sensor)

Main Core

I2C(Temperature Sensor)

Main Core

- 把附件提供的**OLED**的兩個檔案加入到**Main Project**資料夾底下

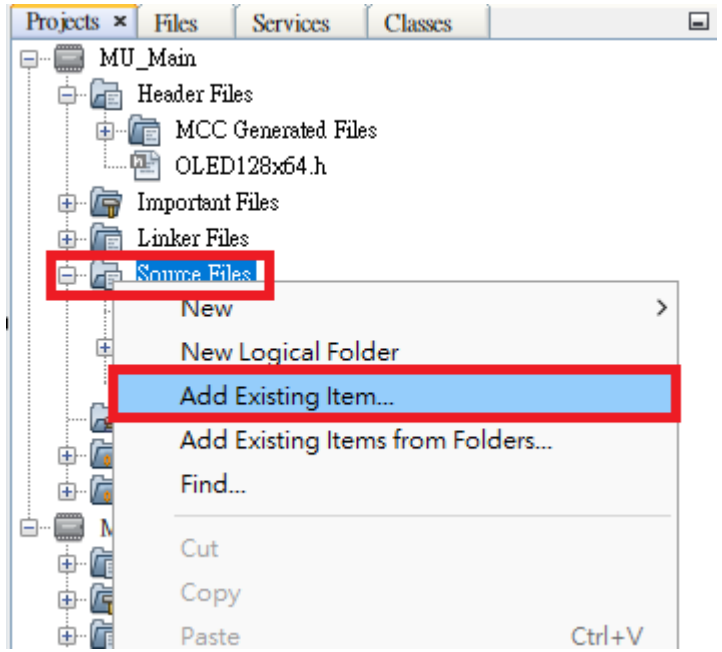


I2C(Temperature Sensor)

Main Core

- 在Main Project底下的Source Files按右鍵

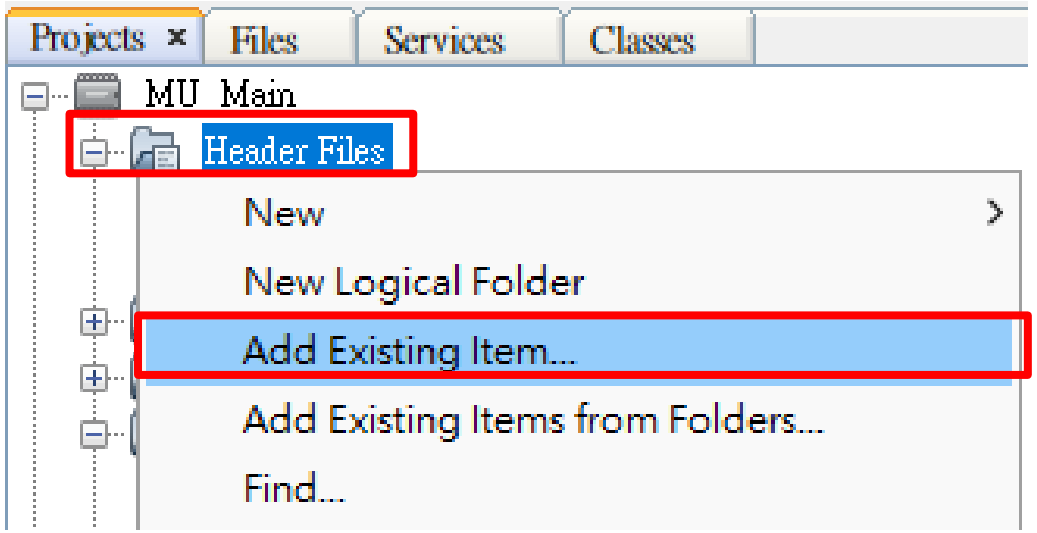
- 選擇 Add Existing Item



I2C(Temperature Sensor)

Main Core

- 在Main Project底下的Header Files按右鍵
- 選擇 Add Existing Item

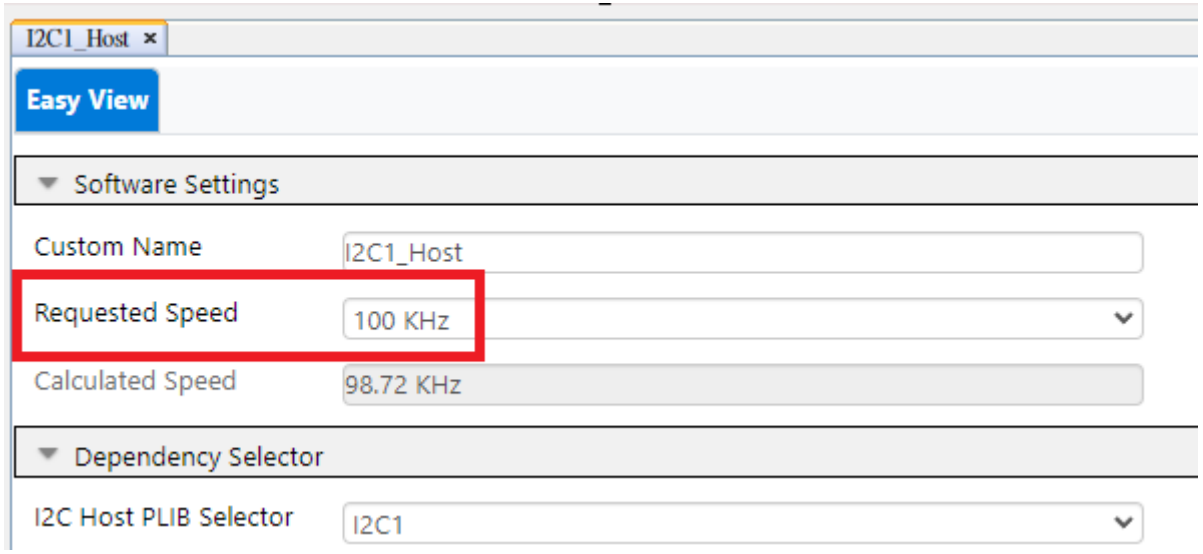


I2C(Temperature Sensor)

Main Core

- **MCC -> Drivers -> I2C_HOST**

- **Requested Speed = 100KHz**



I2C(Temperature Sensor)

Main Core

- **MCC Gen Code相關的.h檔，以及手動添加<string.h>、”OLED128x64”**

```
#include "mcc_generated_files/i2c_host/i2c1.h"
#include <stdio.h>
#include <string.h>
#include "OLED128x64.h"
```

- **Define Temperature Sensor Address**

```
#define MCP9808_ADDR     0x1f //Temp. Sensor
```

- 宣告陣列及變數

```
volatile unsigned char   I2C_Wbuffer[16];
volatile unsigned char   I2C_Rbuffer[16];
unsigned int             MCP9808Value = 0 ;
unsigned char            ASCII_Buffer[20];
```

I2C(Temperature Sensor)

Main Core

- 主程式

```
OLED_Init();
OLED_CLS();
```

```
I2C_Wbuffer[0] = 0x01 ;
I2C_Wbuffer[1] = 0x00 ;
I2C_Wbuffer[2] = 0x04 ;
I2C1_Write(MCP9808_ADDR,I2C_Wbuffer,2);
while ( I2C1_IsBusy()) ;

I2C_Wbuffer[0] = 0x02 ;
I2C_Wbuffer[1] = 0x05 ;
I2C_Wbuffer[2] = 0xa0 ;
I2C1_Write(MCP9808_ADDR,I2C_Wbuffer,3);
while ( I2C1_IsBusy()) ;

I2C_Wbuffer[0] = 0x03 ;
I2C_Wbuffer[1] = 0x00 ;
I2C_Wbuffer[2] = 0x10 ;
I2C1_Write(MCP9808_ADDR,I2C_Wbuffer,3);
while ( I2C1_IsBusy()) ;

I2C_Wbuffer[0] = 0x04 ;
I2C_Wbuffer[1] = 0x05 ;
I2C_Wbuffer[2] = 0xa0 ;
I2C1_Write(MCP9808_ADDR,I2C_Wbuffer,3);
while ( I2C1_IsBusy()) ;
```

```
OLED_Put8x16Str(0,0,"dsPIC33CH Melody");
OLED_Put8x16Str(0,4,"Temp:");
```


MICROCHIP

I2C(Temperature Sensor)

Main Core

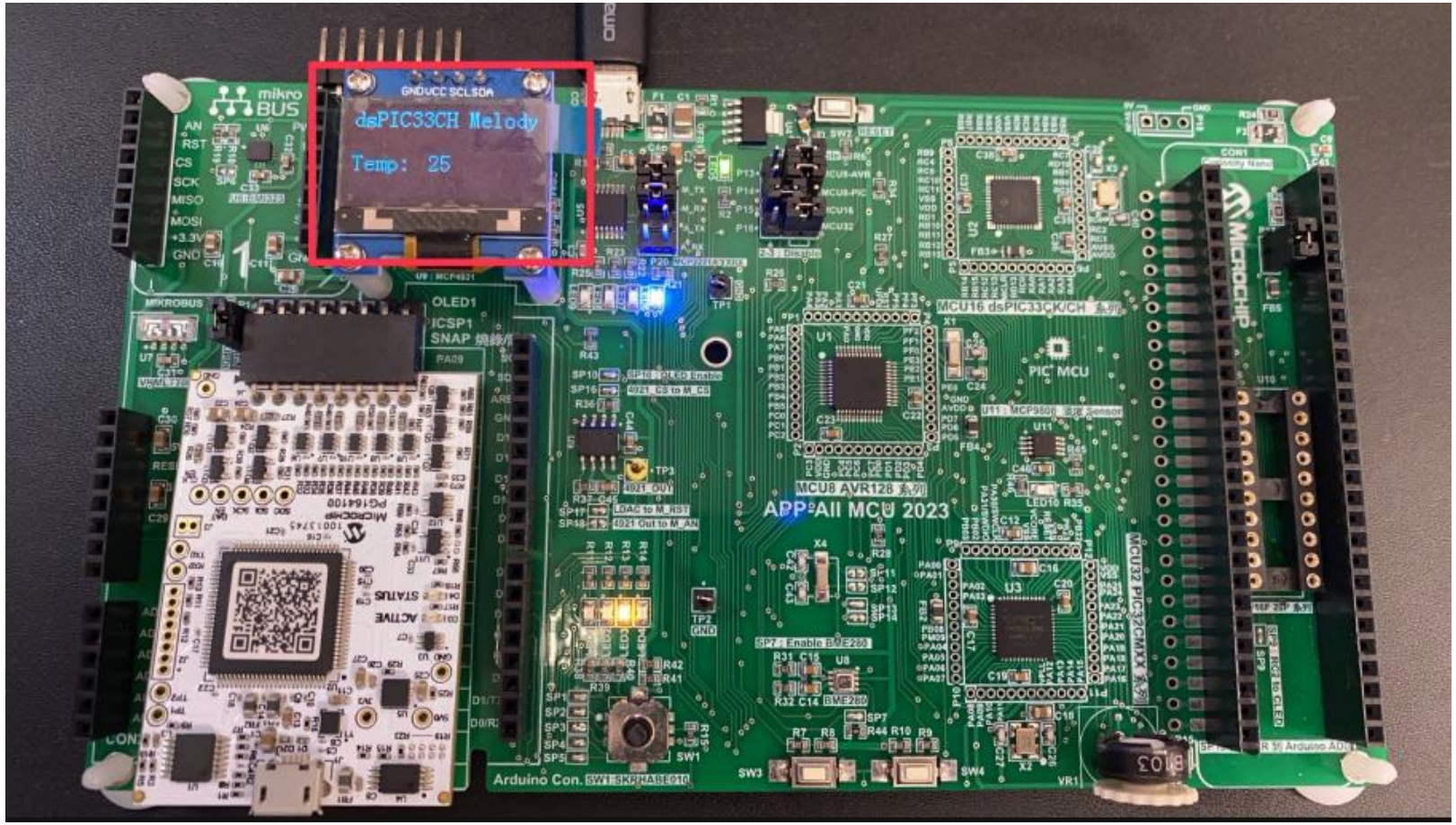
- 主程式 - **While(1) Loop**

```
I2C_Wbuffer[0] = 5 ;
I2C1_WriteRead(MCP9808_ADDR,I2C_Wbuffer,1,I2C_Rbuffer,2);
while ( I2C1_IsBusy()) ;

MCP9808Value = (unsigned int)(I2C_Rbuffer[0] * 256) + I2C_Rbuffer[1];
MCP9808Value  = MCP9808Value >> 4 ;
MCP9808Value &= 0x0fff ;
sprintf (ASCII_Buffer,"%3d",MCP9808Value) ;
OLED_Put8x16ASCII(44,4,3,(uint8_t*)ASCII_Buffer) ;
```

I2C(Temperature Sensor)

Main Core

- **LCD Display**



PWM

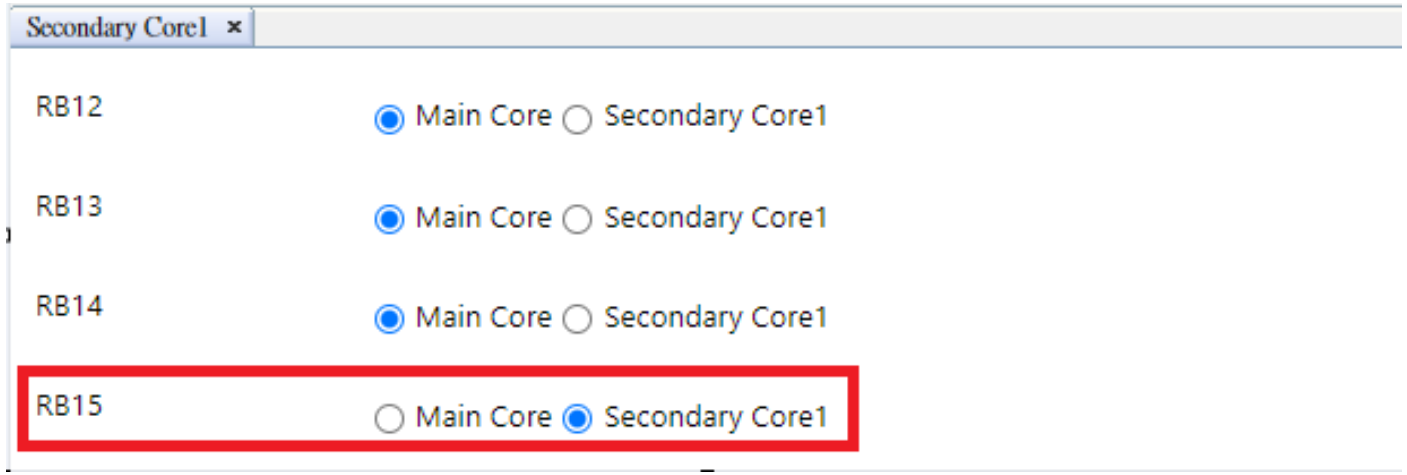
Secondary Core

MICROCHIP

PWM

Secondary Core

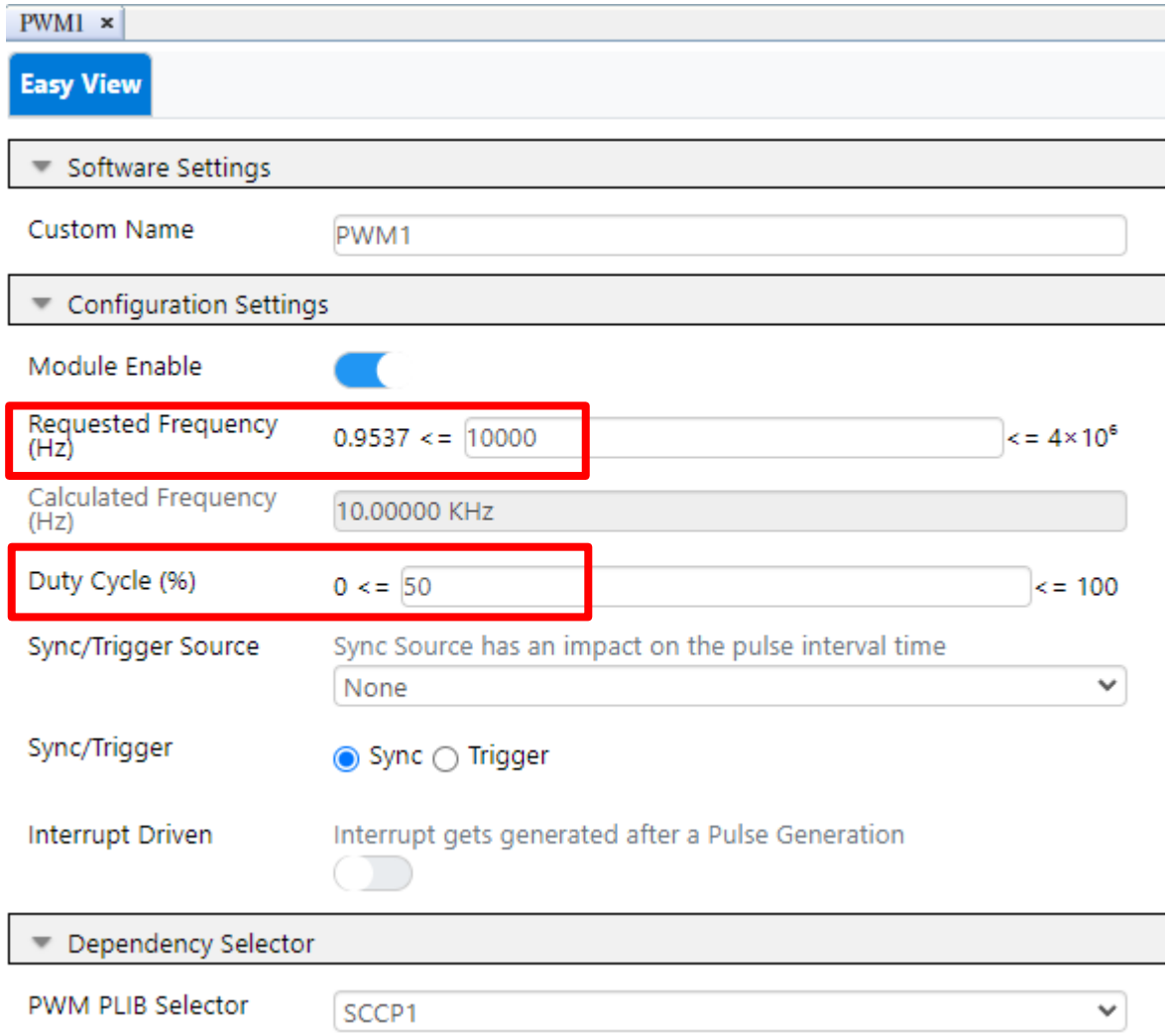
- **MCC(MU_dsPIC33CH_Main)設定使用RB15作為PWM的輸出**



PWM

Secondary Core

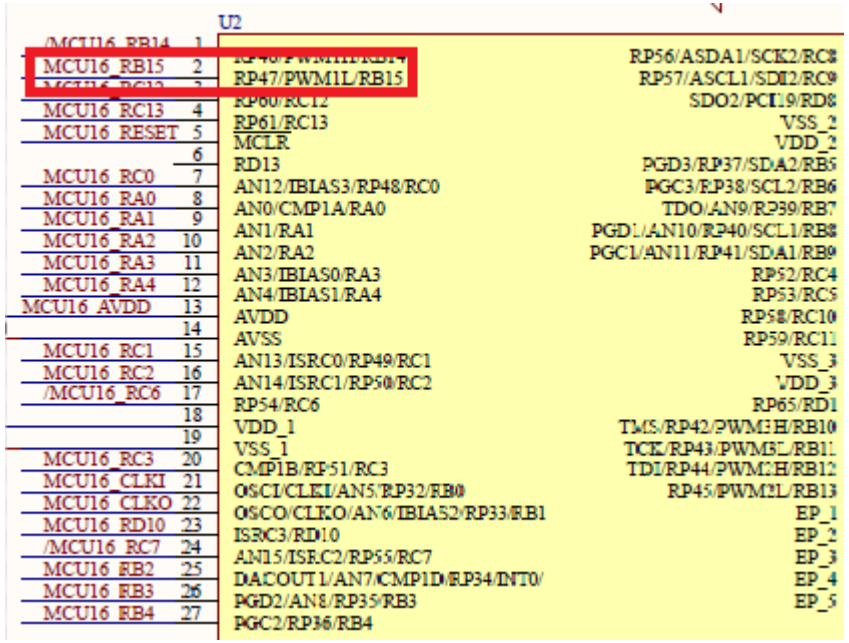
- MCC -> Drivers -> Add PWM
- Requested Frequency = 10KHz
- Duty Cycle = 50%

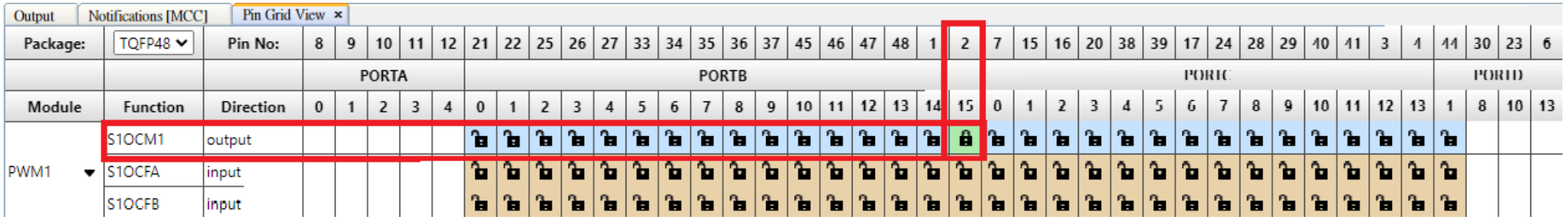


PWM

Secondary Core

- 設**RB15**為輸出腳位





PWM

Secondary Core

- 用示波器勾出**RB15**的波型

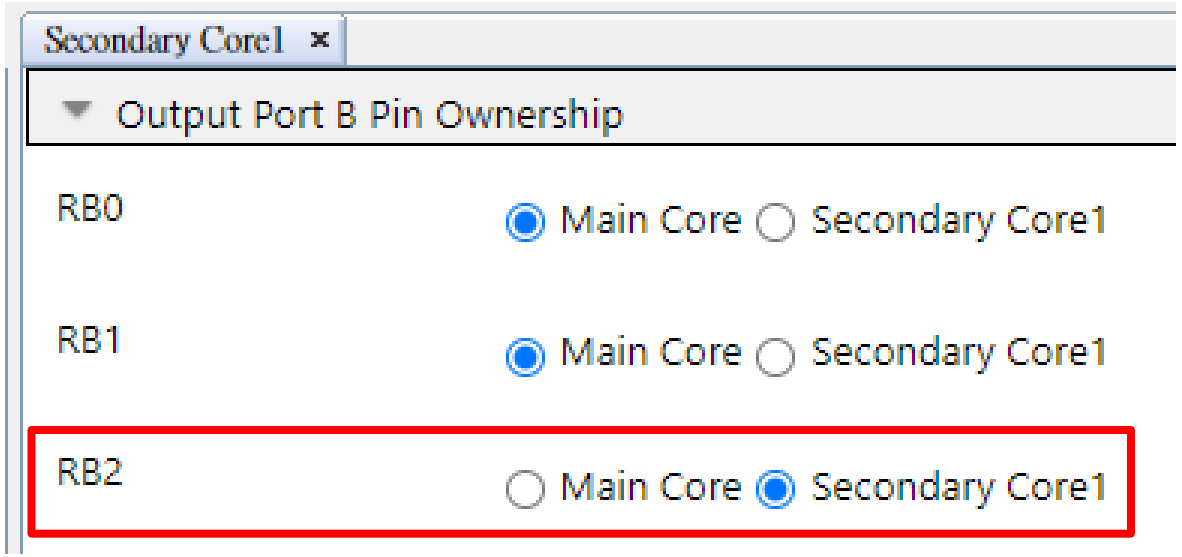
DAC(弦波)

Secondary Core

DAC(弦波)

Secondary Core

- **MCC(MU_dsPIC33CH_Main)設定使用RB15作為Secondary的輸出**

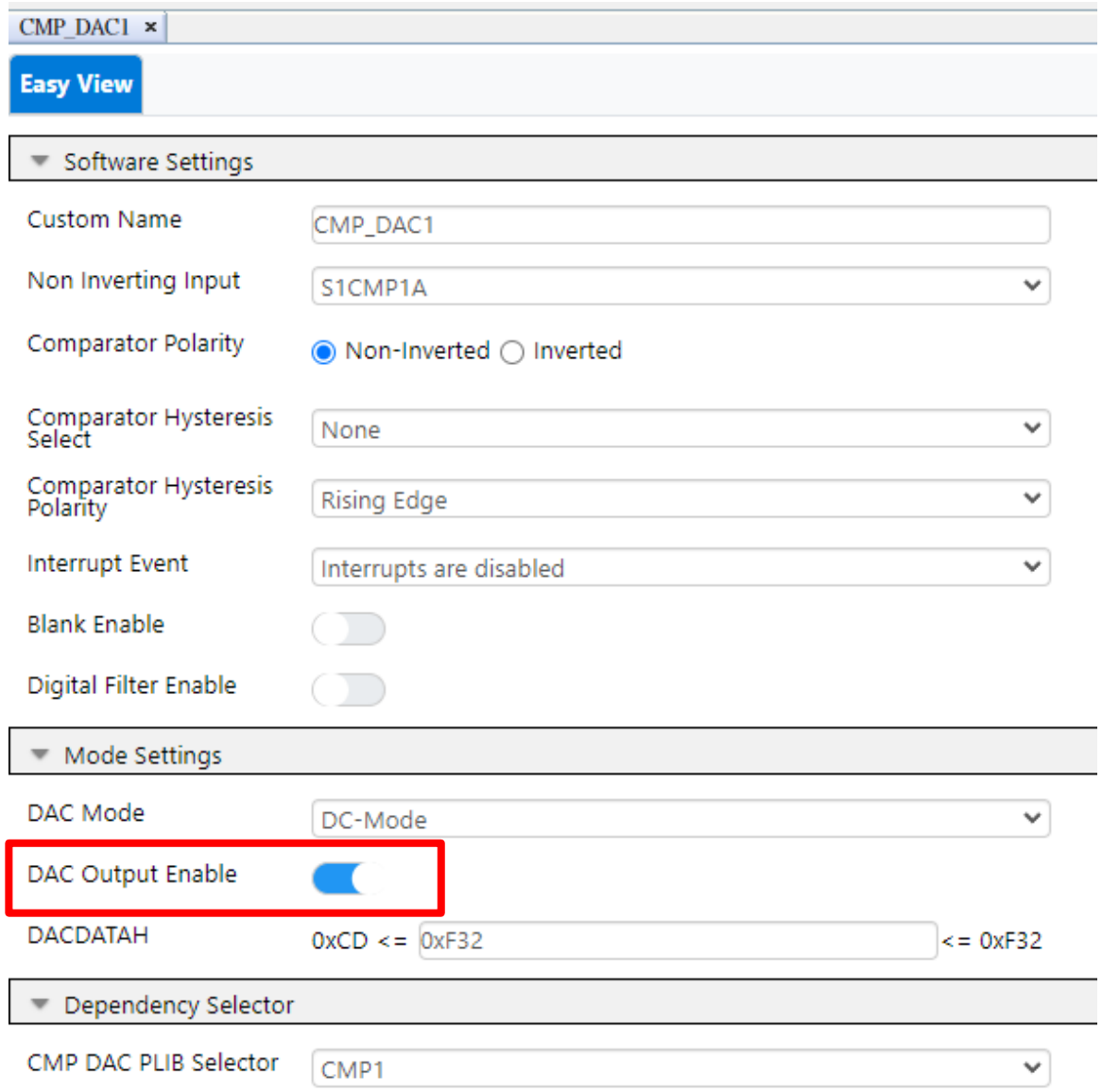


DAC(弦波)

Secondary Core

- **MCC -> Drivers -> Add DAC**

- **DAC Output Enable**



DAC(弦波)

Secondary Core

- 設RB2為輸出腳位

/MCU16_RB14	1	RP46/PWM1H/RB14	RP56/ASDA1/SCK2/RC8
MCU16_RB15	2	RP47/PWM1L/RB15	RP57/ASCL1/SDI2/RC9
MCU16_RC12	3	RP60/RC12	SDO2/PCI19/RD8
MCU16_RC13	4	RP61/RC13	VSS_2
MCU16_RESET	5	MCLR	VDD_2
	6	RD13	PGD3/RP37/SDA2/RB5
MCU16_RC0	7	AN12/IBIAS3/RP48/RC0	PGC3/RP38/SCL2/RB6
MCU16_RA0	8	AN0/CMP1A/RA0	TDO/AN9/RP39/RB7
MCU16_RA1	9	AN1/RA1	PGD1/AN10/RP40/SCL1/RB8
MCU16_RA2	10	AN2/RA2	PGC1/AN11/RP41/SDA1/RB9
MCU16_RA3	11	AN3/IBIAS0/RA3	RP52/RC4
MCU16_RA4	12	AN4/IBIAS1/RA4	RP53/RC5
MCU16_AVDD	13	AVDD	RP58/RC10
	14	AVSS	RP59/RC11
MCU16_RC1	15	AN13/ISRC0/RP49/RC1	VSS_3
MCU16_RC2	16	AN14/ISRC1/RP50/RC2	VDD_3
/MCU16_RC6	17	RP54/RC6	RP65/RD1
	18	VDD_1	TMS/RP42/PWM3H/RB10
	19	VSS_1	TCK/RP43/PWM3L/RB11
MCU16_RC3	20	CMP1B/RP51/RC3	TDI/RP44/PWM2H/RB12
MCU16_CLKI	21	OSCI/CLKI/AN5/RP32/RB0	RP45/PWM2L/RB13
MCU16_CLKO	22	OSCO/CLKO/AN6/IBIAS2/RP33/RB1	EP_1
MCU16_RD10	23	ISRC3/RD10	EP_2
/MCU16_RC7	24		EP_3
MCU16_RB2	25	DACOUT1/AN7/CMP1D/RP34/INT0/	EP_4
	26	PGD2/AN8/RP35/RB3	EP_5
MCU16_RB4	27	PGC2/RP36/RB4	

Output | Notifications | Notifications [MCC] | Pin Grid View ×

Package:	TQFP48	Pin No:	8	9	10	11	12	21	22	25	26	27	33	34	35	36	37	45	46	47	48	1	2	7	15	16	20	38	39	17	24	28	29	40	41	3	4	44	30	23	6
	S1OCFB	input	PORTA								PORTB													PORTC															PORTD			
Module	Function	Direction	0	1	2	3	4	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7	8	9	10	11	12	13	1	8	10	13
CMP_DAC1	S1CMP1B	input																																							
	S1CMP1D	input																																							
	S1CMP1	output																																							
	DACOUT	output																																							

MICROCHIP

DAC(弦波)

Secondary Core

- **Gen Code後相關的.h檔**

```
#include "mcc_generated_files/cmp/cmp1.h"
#include "mcc_generated_files/pwm/sccp1.h"
```

- 宣告變數及陣列

```
float Pwm_sin_value = 0;
float DAC_data = 0;
int DAC_data_int = 0;
int Middle_of_DAC = 0;
```

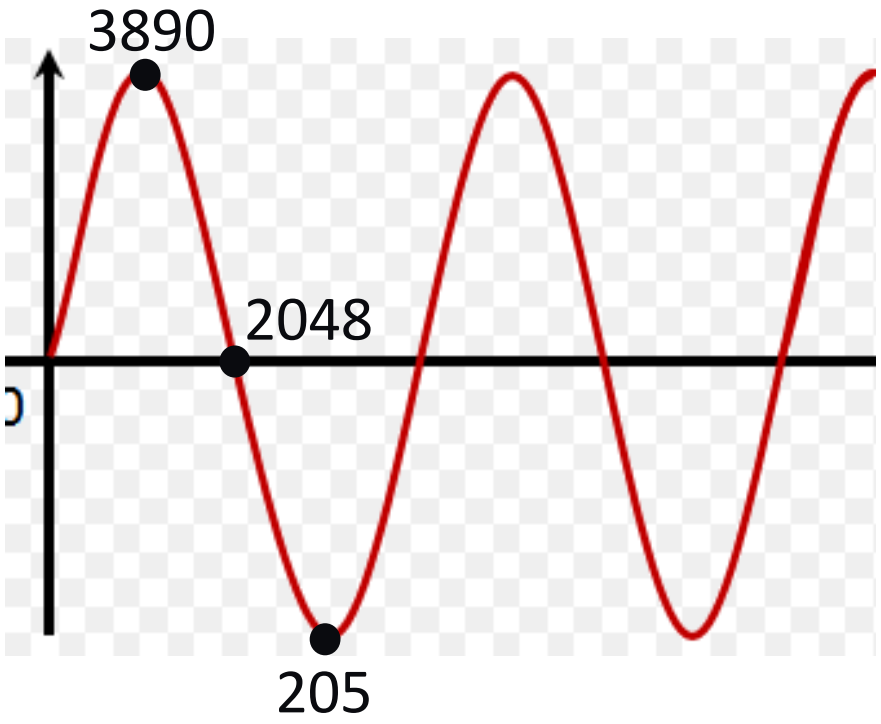
```
float sine_table[180] =
{
    0.0000, 0.0175, 0.0349, 0.0523, 0.0698, 0.0872, 0.1045, 0.1219, 0.1392, 0.1564,
    0.1736, 0.1908, 0.2079, 0.2250, 0.2419, 0.2588, 0.2756, 0.2924, 0.3090, 0.3256,
    0.3420, 0.3584, 0.3746, 0.3907, 0.4067, 0.4226, 0.4384, 0.4540, 0.4695, 0.4848,
    0.5000, 0.5150, 0.5299, 0.5446, 0.5592, 0.5736, 0.5878, 0.6018, 0.6157, 0.6293,
    0.6428, 0.6561, 0.6691, 0.6820, 0.6947, 0.7071, 0.7193, 0.7314, 0.7431, 0.7547,
    0.7660, 0.7771, 0.7880, 0.7986, 0.8090, 0.8192, 0.8290, 0.8387, 0.8480, 0.8572,
    0.8660, 0.8746, 0.8829, 0.8910, 0.8988, 0.9063, 0.9135, 0.9205, 0.9272, 0.9336,
    0.9397, 0.9455, 0.9511, 0.9563, 0.9613, 0.9659, 0.9703, 0.9744, 0.9781, 0.9816,
    0.9848, 0.9877, 0.9903, 0.9925, 0.9945, 0.9962, 0.9976, 0.9986, 0.9994, 0.9998,
    0.9998, 0.9994, 0.9986, 0.9976, 0.9962, 0.9945, 0.9925, 0.9903, 0.9877, 0.9848,
    0.9816, 0.9781, 0.9744, 0.9703, 0.9659, 0.9613, 0.9563, 0.9511, 0.9455, 0.9397,
    0.9336, 0.9272, 0.9205, 0.9135, 0.9063, 0.8988, 0.8910, 0.8829, 0.8746, 0.8660,
    0.8572, 0.8480, 0.8387, 0.8290, 0.8192, 0.8090, 0.7986, 0.7880, 0.7771, 0.7660,
    0.7547, 0.7431, 0.7314, 0.7193, 0.7071, 0.6947, 0.6820, 0.6691, 0.6561, 0.6428,
    0.6293, 0.6157, 0.6018, 0.5878, 0.5736, 0.5592, 0.5446, 0.5299, 0.5150, 0.5000,
    0.4848, 0.4695, 0.4540, 0.4384, 0.4226, 0.4067, 0.3907, 0.3746, 0.3584, 0.3420,
    0.3256, 0.3090, 0.2924, 0.2756, 0.2588, 0.2419, 0.2250, 0.2079, 0.1908, 0.1736,
    0.1564, 0.1392, 0.1219, 0.1045, 0.0872, 0.0698, 0.0523, 0.0349, 0.0175, 0.0000,
};
```


MICROCHIP

DAC(弦波)

Secondary Core

- 主程式

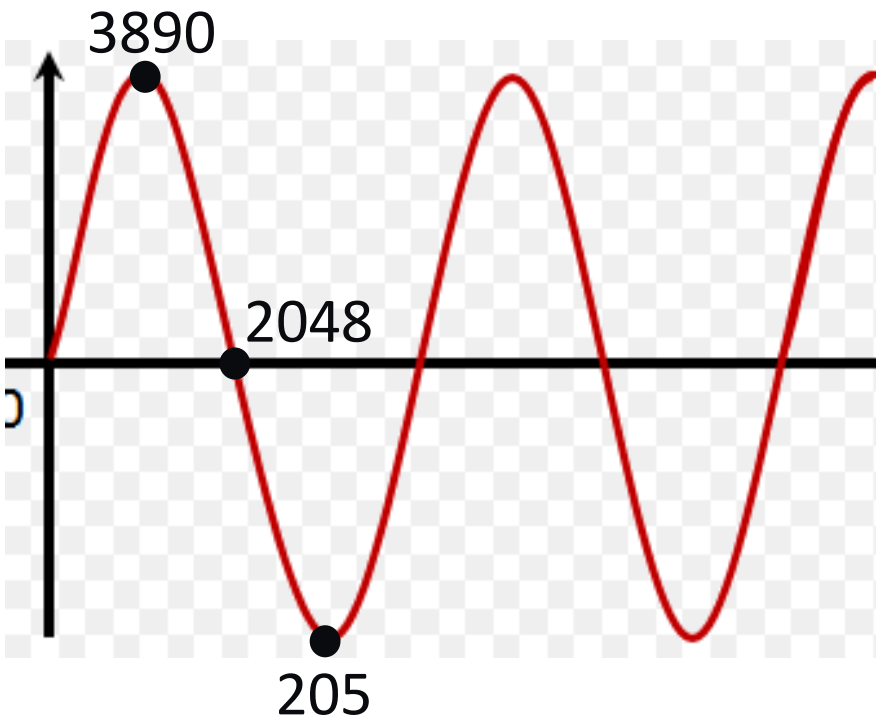



```
Middle_of_DAC = (3890-205)/2;
```

DAC(弦波)

Secondary Core

- 主程式 – **While(1) Loop**




```
//sin波
for (int m=0; m<180; ++m)
{
    Pwm_sin_value = sine_table[m] * 100;
    SCCP1_PWM_DutyCycleSet(Pwm_sin_value);
    DAC_data = (((Middle_of_DAC + 1) / 100) * Pwm_sin_value) + 2048;
    DAC_data_int = (int)DAC_data;
    CMP1_DACDataWrite(DAC_data_int);
}

for (int n=0; n<180; ++n)
{
    Pwm_sin_value = sine_table[n] * 100;
    SCCP1_PWM_DutyCycleSet(Pwm_sin_value);
    DAC_data = ((Middle_of_DAC / 100) * (100-Pwm_sin_value)) + 205;
    DAC_data_int = (int)DAC_data;
    CMP1_DACDataWrite(DAC_data_int);
}
```

DAC(弦波)

Secondary Core

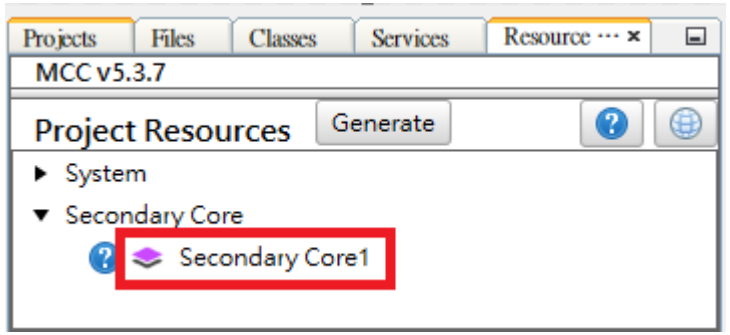
- 用示波器勾**RB2**的波型

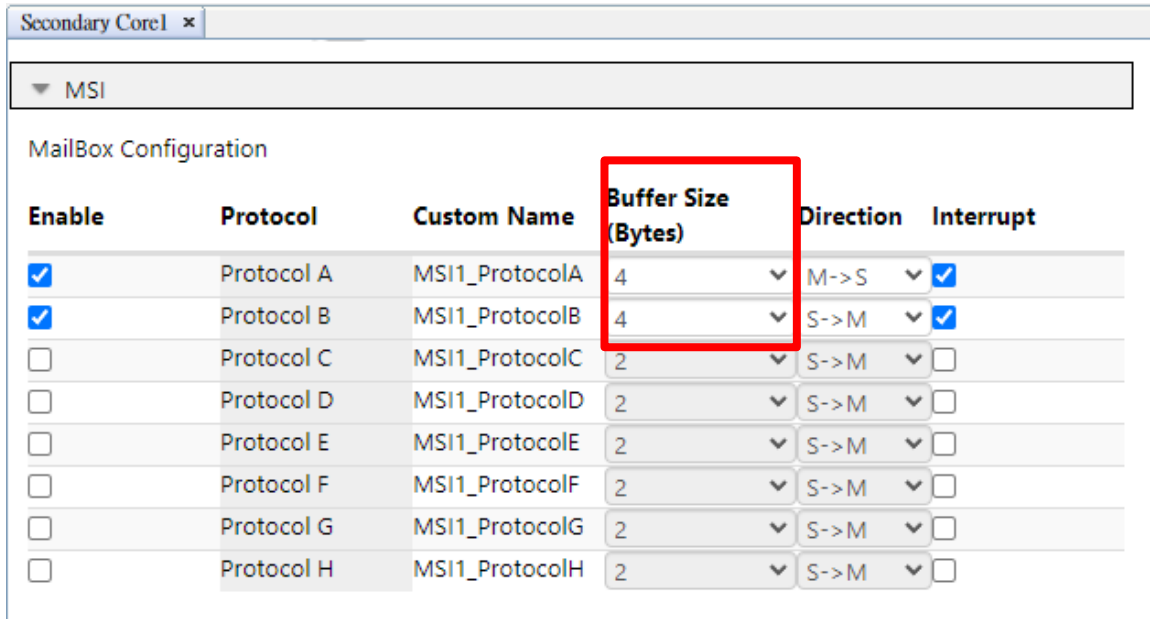
主副核功能的整合

Main & Secondary Core

主副核功能的整合
Main Core

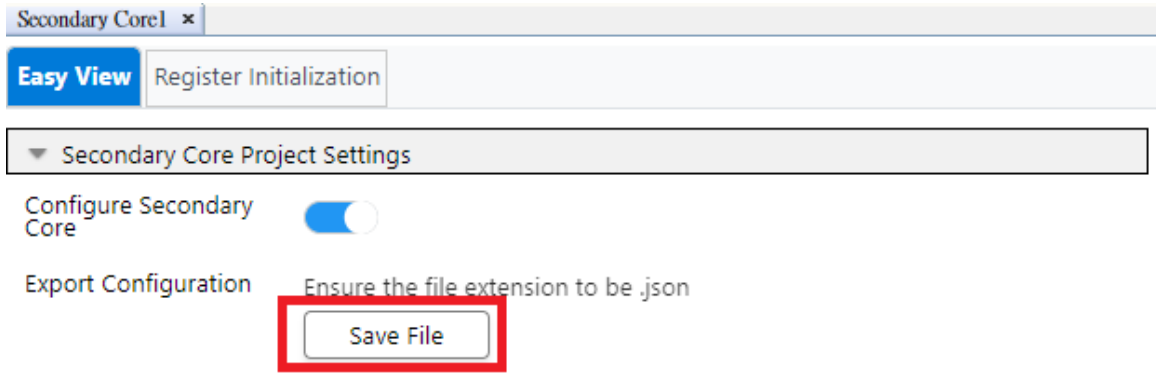
- **MCC -> Secondary Core -> Buffere Size 改成4**





主副核功能的整合
Main Core

- **Save File 以json 的形式儲存在Main Project的資料夾中**



- 宣告陣列 `unsigned int UART_Duty[2];`

- **Define一個mailbox 做讀跟寫的測試常數** `#define DATA_UNDER_TEST 0xAAAA`

主副核功能的整合
Main Core

- 主程式

```
dataSend[0] =  DATA_UNDER_TEST;          //Initializing to known value.
dataReceive[0] = 0;

//Mailbox write
SEC_CORE1_ProtocolWrite(MSI1_ProtocolA, (uint16_t*)dataSend);

//Issue interrupt to secondary
SEC_CORE1_InterruptRequestGenerate();
while(!SEC_CORE1_IsInterruptRequestAcknowledged());
SEC_CORE1_InterruptRequestComplete();
while(SEC_CORE1_IsInterruptRequestAcknowledged());

//Wait for interrupt from secondary
while(!SEC_CORE1_IsInterruptRequested());
SEC_CORE1_InterruptRequestAcknowledge();
while(SEC_CORE1_IsInterruptRequested());
SEC_CORE1_InterruptRequestAcknowledgeComplete();

//Mailbox read
SEC_CORE1_ProtocolRead(MSI1_ProtocolB, (uint16_t*)dataReceive);
```

```
//Glow LED on data match
if(memcmp(dataSend, dataReceive, sizeof(dataSend)) == 0)
{
    LED1_SetHigh();
}
else
{
    LED1_SetLow();
}
```


MICROCHIP

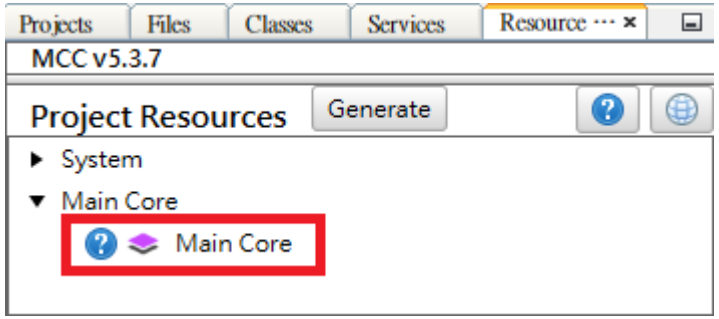
主副核功能的整合
Main Core

- 主程式-while(1)

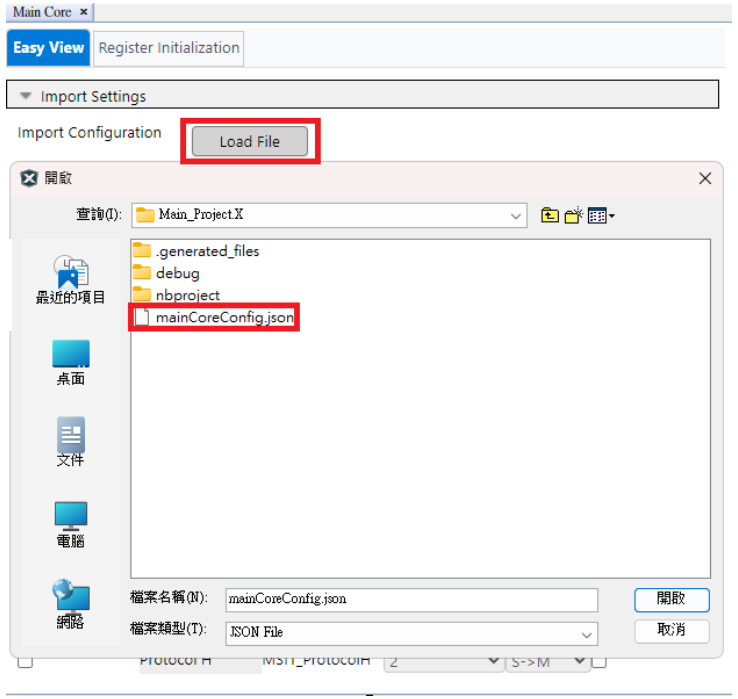
```
UART_Duty[0] = UART1_Read();
UART_Duty[1] = UART1_Read();

printf(" DutyCycle = %c%c\r\n", (char)UART_Duty[0], (char)UART_Duty[1]);

dataSend[0] = (UART_Duty[0]-48);
dataSend[1] = (UART_Duty[1]-48);

SEC_CORE1_ProtocolWrite(MSI1_ProtocolA, (uint16_t*)dataSend);
printf(" mailbox = %d%d\r\n", MSI1MBX0D, MSI1MBX1D);
```

主副核功能的整合
Secondary Core

- 打開MCC 左上方視窗選MainSlave1



- **Load File 選之前儲存在Main Project資料夾裡面的.json檔 & Gen Code**



Dual Core Communicate

Secondary Core

- Define一個mailbox 做讀跟寫的測試常數 `#define DATA_UNDER_TEST 0xAAAA`

- 宣告變數和陣列

```
const struct MAIN_CORE_INTERFACE *mainCore = &MSIInterface;
uint16_t dataReceive[MSI1_ProtocolA_SIZE];    //for Protocol size refer main_core_types.h file
uint16_t dataSend[MSI1_ProtocolB_SIZE];
```

主副核功能的整合
Secondary Core

- 主程式

```
//Wait for interrupt from Main core
while(!MAIN_CORE_IsInterruptRequested());
MAIN_CORE_InterruptRequestAcknowledge();
while(MAIN_CORE_IsInterruptRequested());
MAIN_CORE_InterruptRequestAcknowledgeComplete();

//Mailbox read
MAIN_CORE_ProtocolRead(MSI1_ProtocolA, (uint16_t*)dataReceive);
//Copy the received data for retransmission
memcpy(dataSend, dataReceive, sizeof(dataReceive));
//Retransmits Mailbox
MAIN_CORE_ProtocolWrite(MSI1_ProtocolB, (uint16_t*)dataSend);

//Issue interrupt to Main core
MAIN_CORE_InterruptRequestGenerate();
while(!MAIN_CORE_IsInterruptRequestAcknowledged());
MAIN_CORE_InterruptRequestComplete();
while(MAIN_CORE_IsInterruptRequestAcknowledged());
```

```
if(dataReceive[0] == DATA_UNDER_TEST)
{
    LED3_SetHigh();
}
else
{
    LED3_SetLow();
}
```

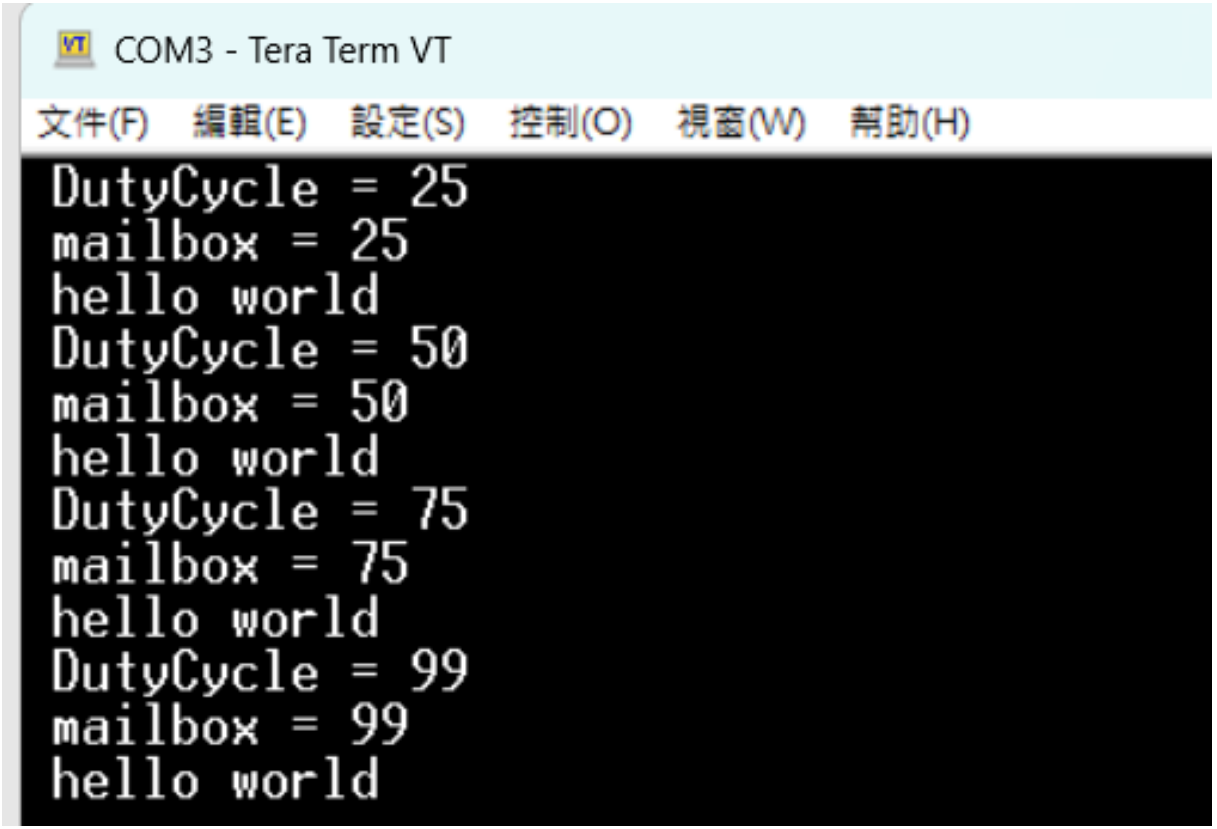
主副核功能的整合
Secondary Core

- 主程式-while(1) loop

```
MAIN_CORE_ProtocolRead(MSI1_ProtocolA, (uint16_t*)dataReceive);
SCCP1_PWM_DutyCycleSet((405 * (dataReceive[0]*10)+dataReceive[1]) / 100);
```

主副核功能的整合
Secondary Core

- **Terminal**



MICROCHIP

本課程到此結束

謝謝各位！

MICROCHIP