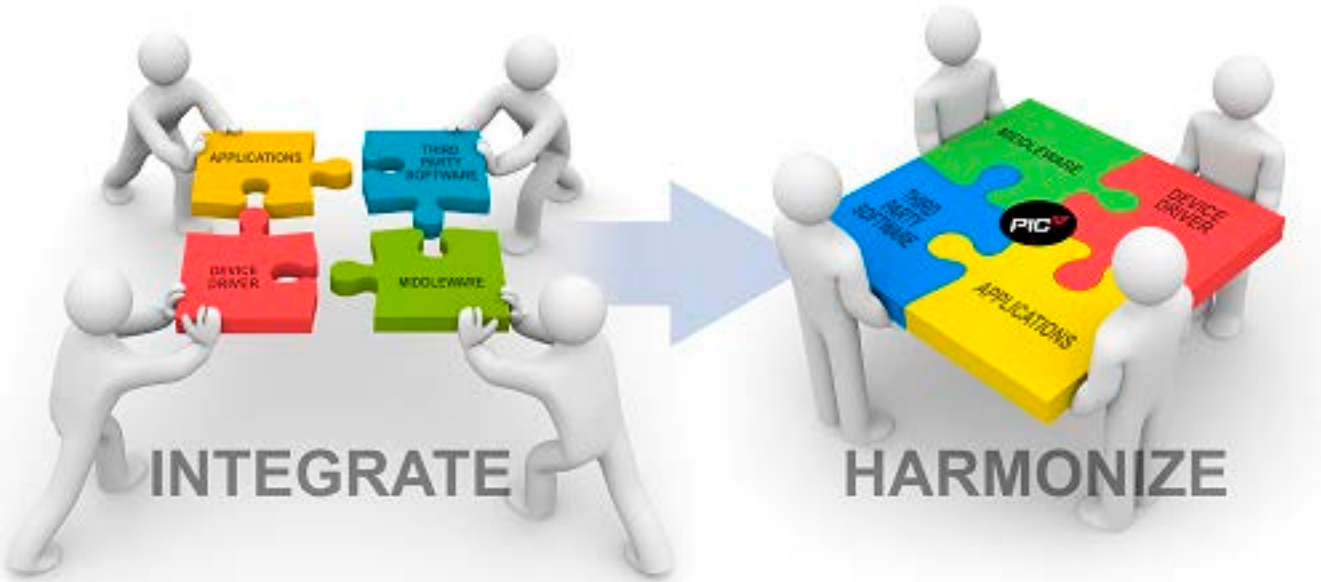
MICROCHIP

PIC32MX470 Curiosirty Board
&
APP043

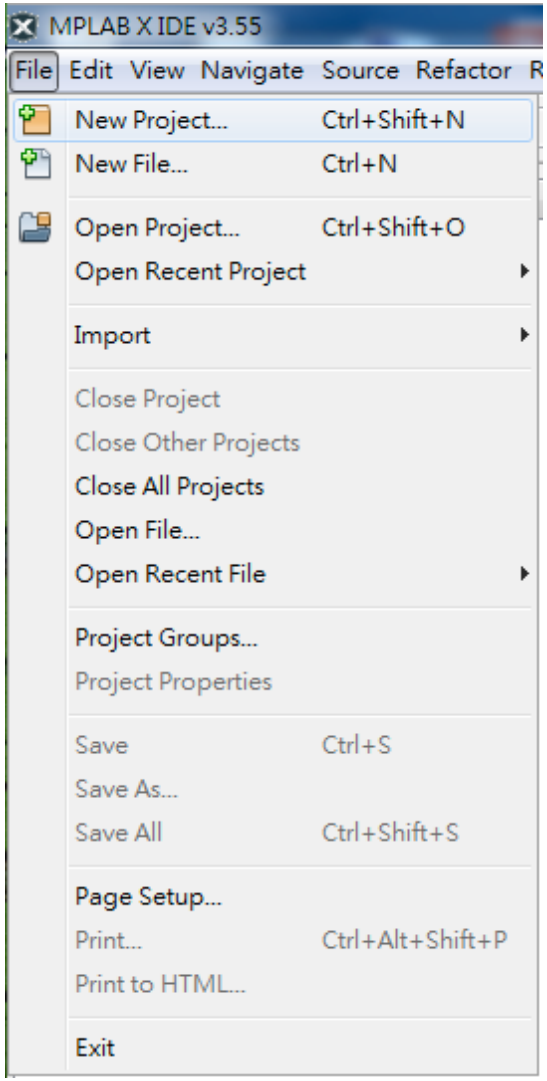


MPLAB® Harmony

系統開發環境

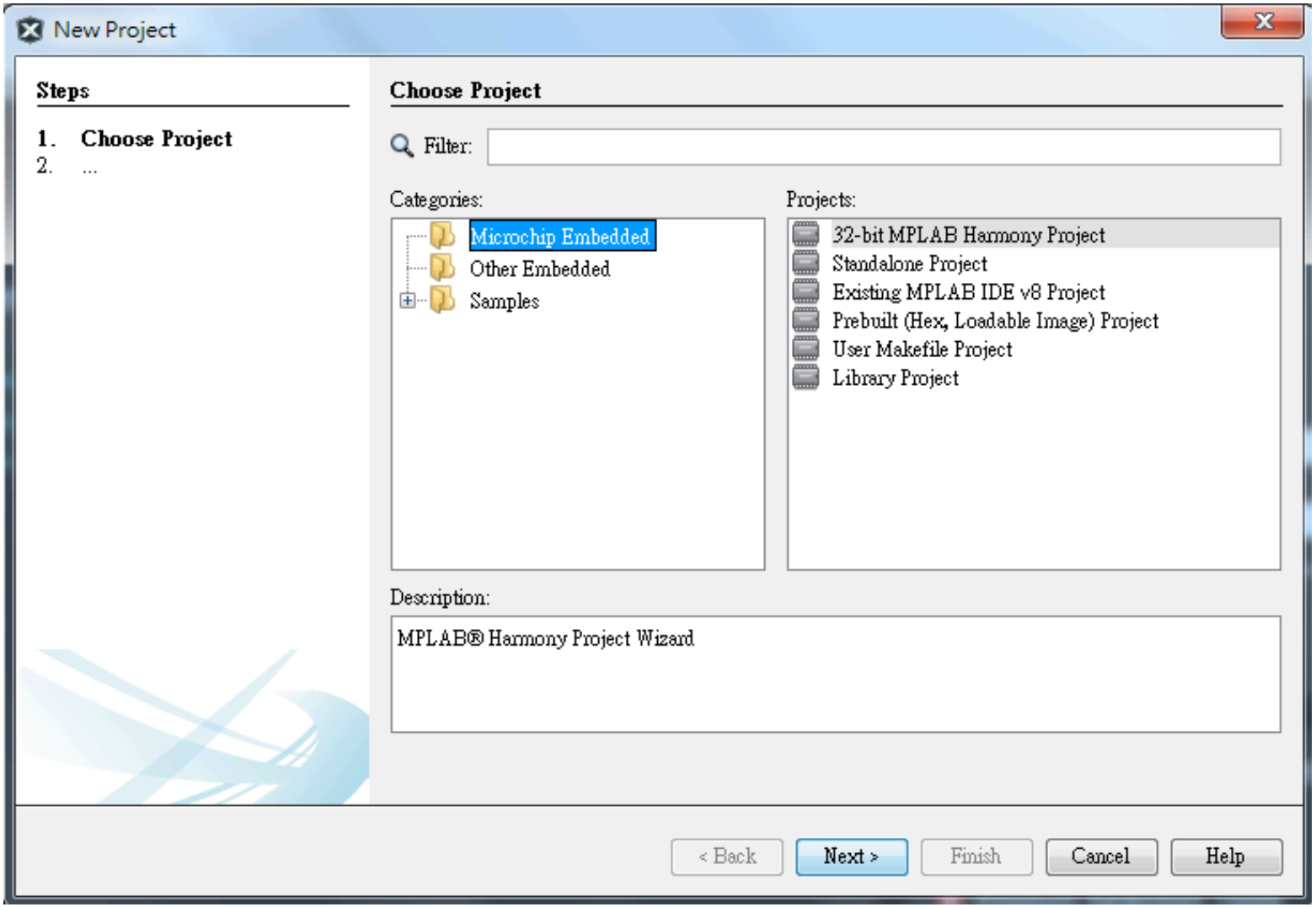
- **MCU 32 Harmony 開發環境**
  - MPLBA X v3.55
  - XC32 v1.42
- **MCU32 Curiosity 開發板**
- **APP043**
- 參考資料
  - PIC32MX330/350/370/430/450/470（60001185F）

建立 Harmony 專案
Step 1：開啟新的專案

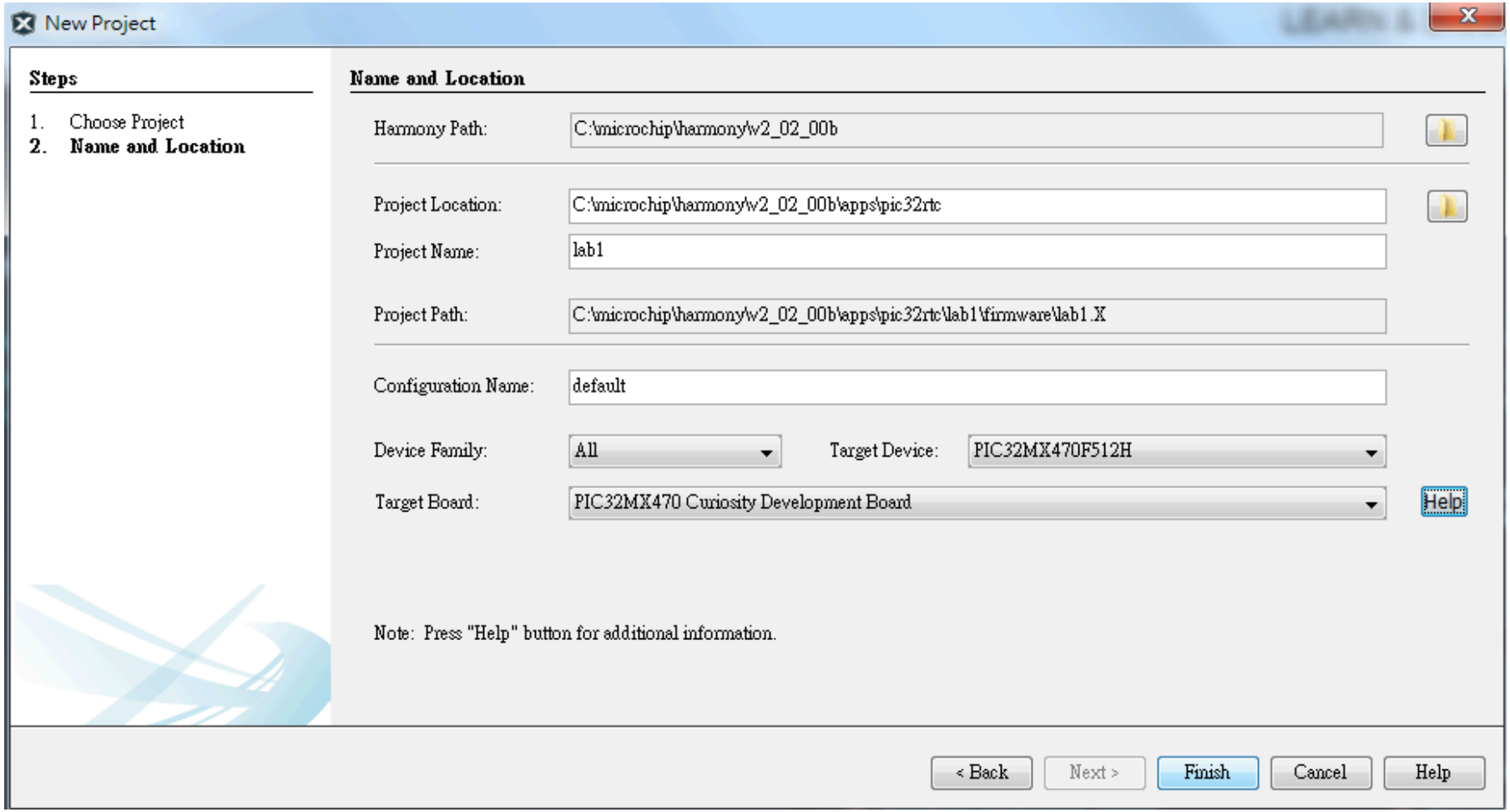


建立 Harmony 專案

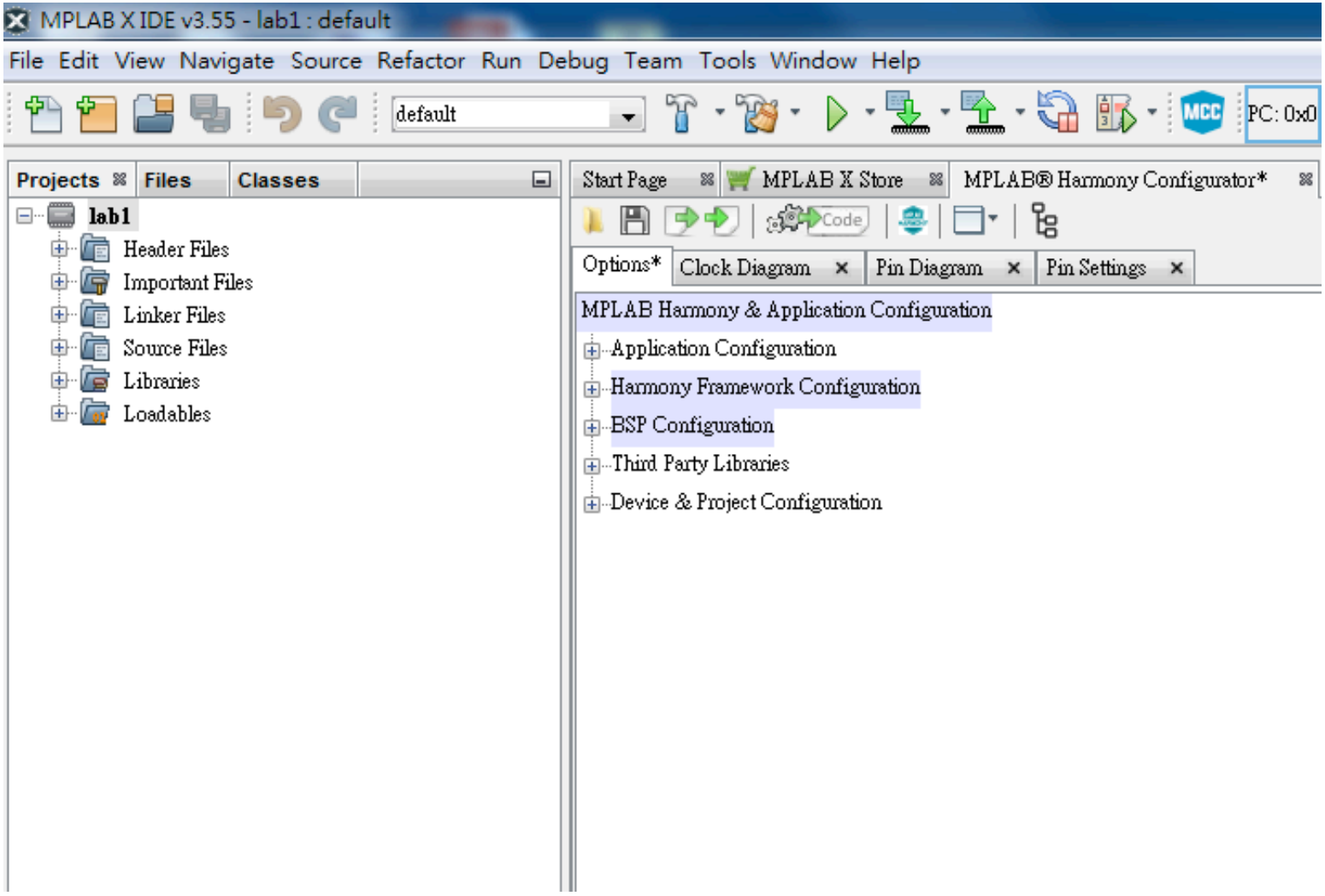
Step 2：選擇專案型別 “32-bit MPLAB Harmony Project”



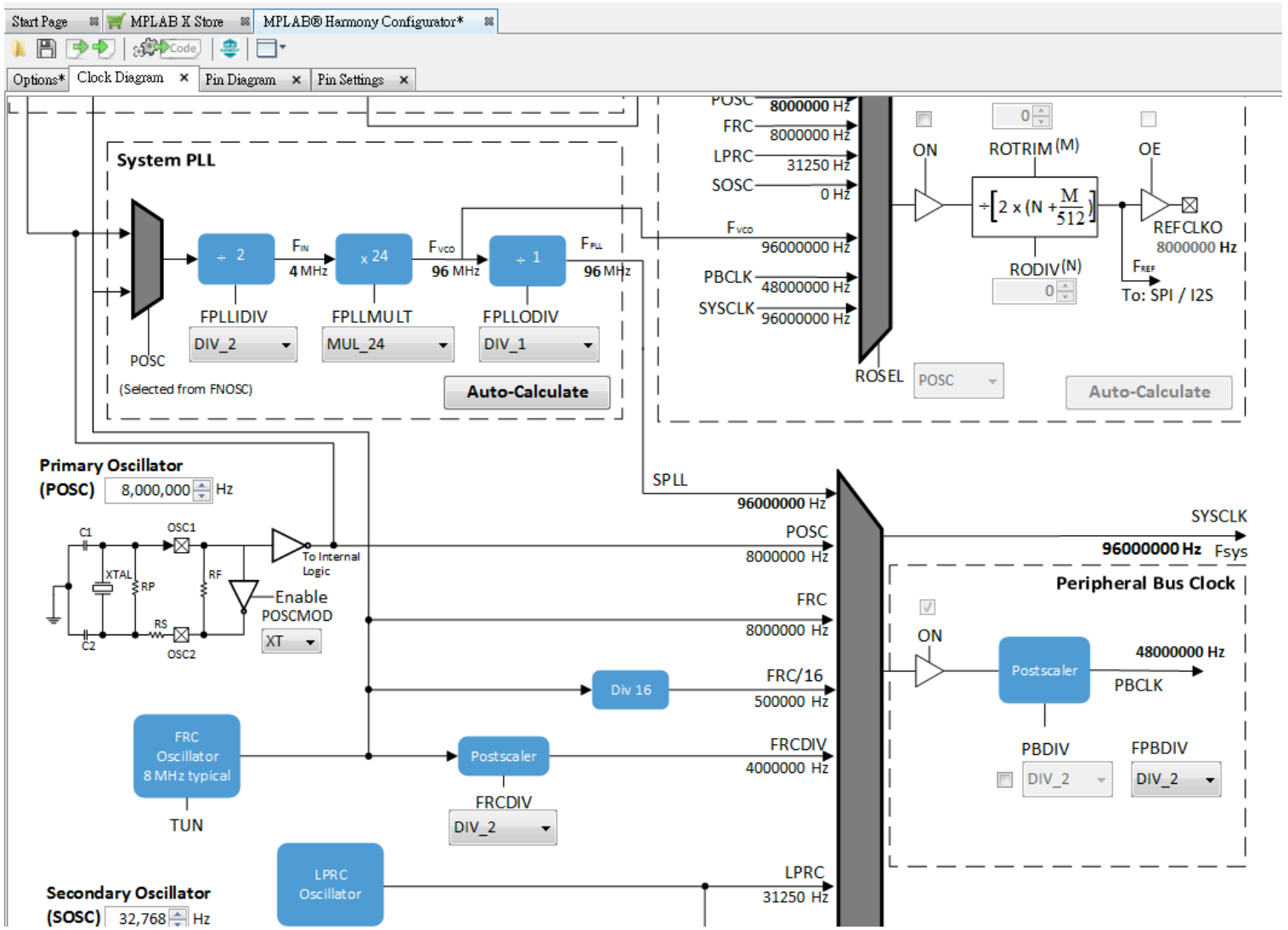
建立 Harmony 專案
Step 3：設定Harmony版本、專案名稱、路徑、MCU32



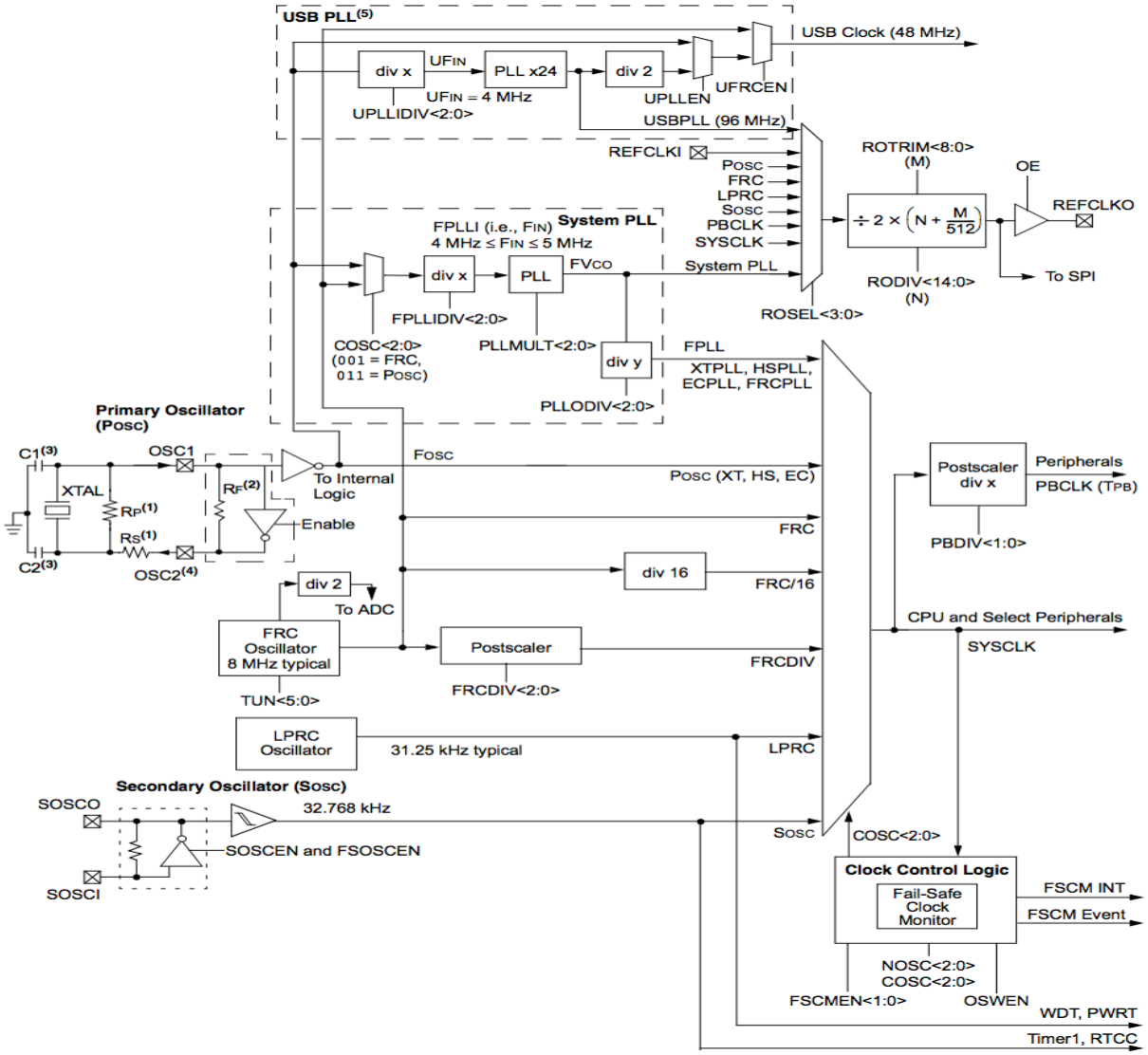
建立 Harmony 專案
Step 4：成功建立Harmony專案框架



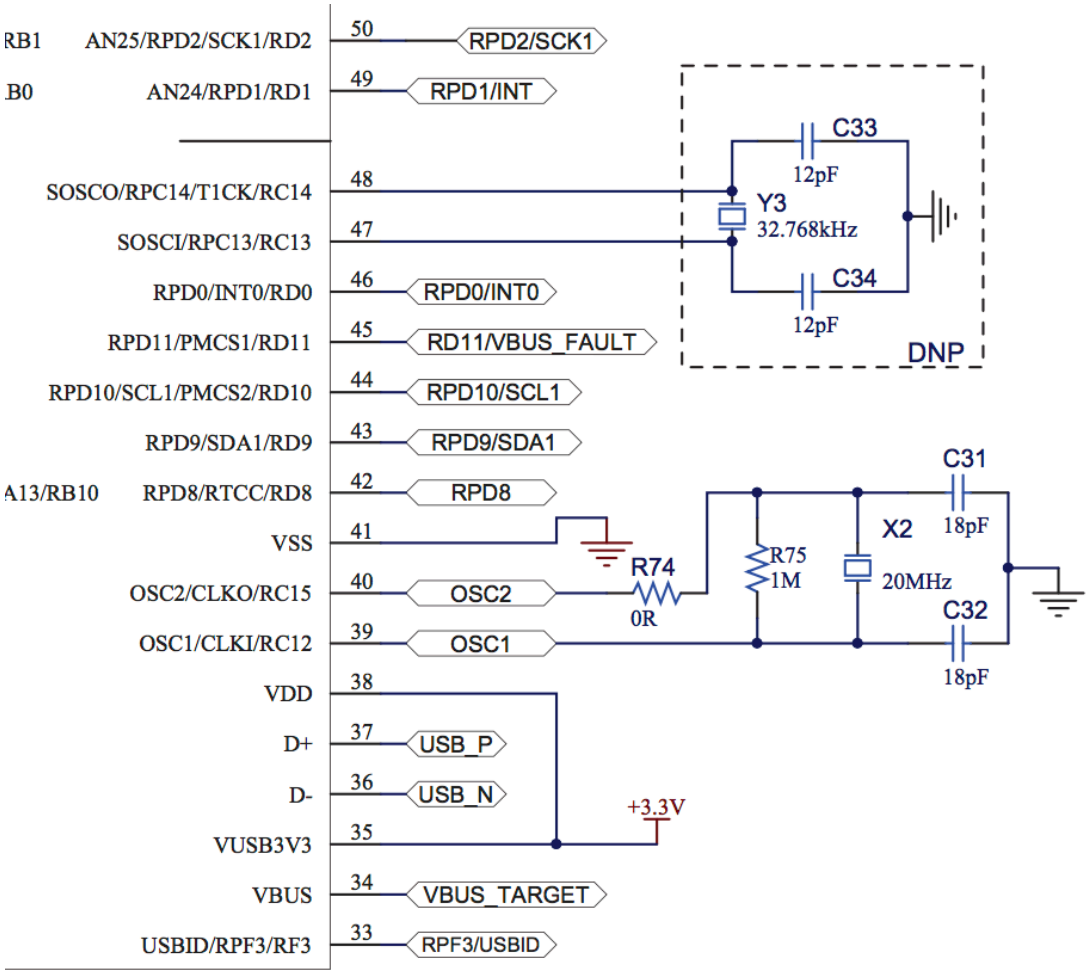
建立 Harmony 專案
Step 5：MHC – Clock Diagram



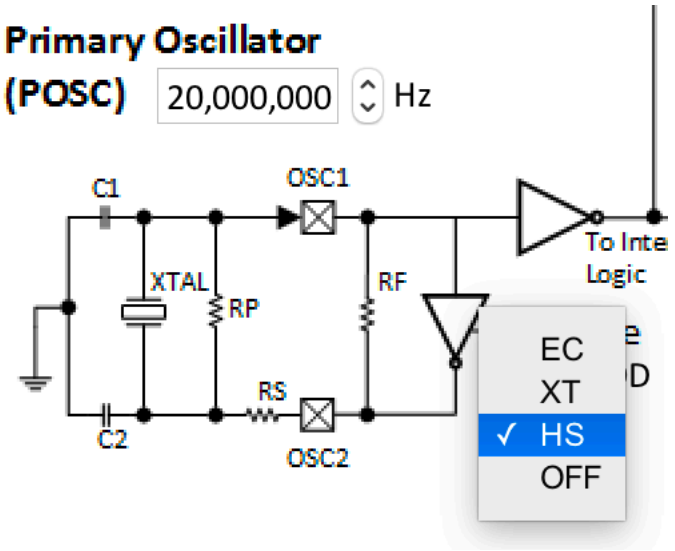
建立 Harmony 專案
Step 5：PIC32MX470 CLK 方塊圖

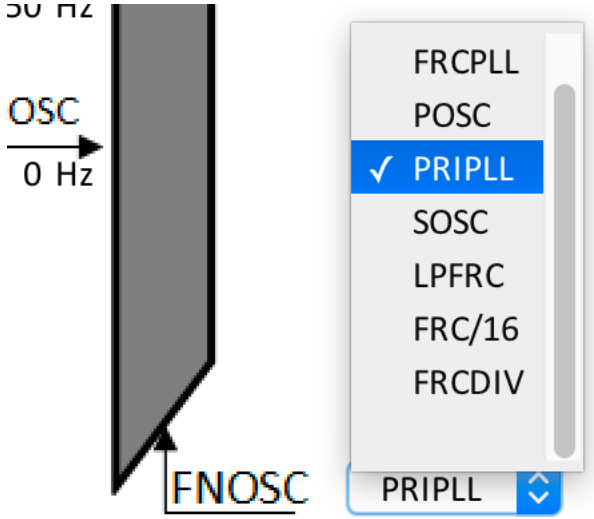


建立 Harmony 專案
Step 5：PIC32 Curiosity board - CLK

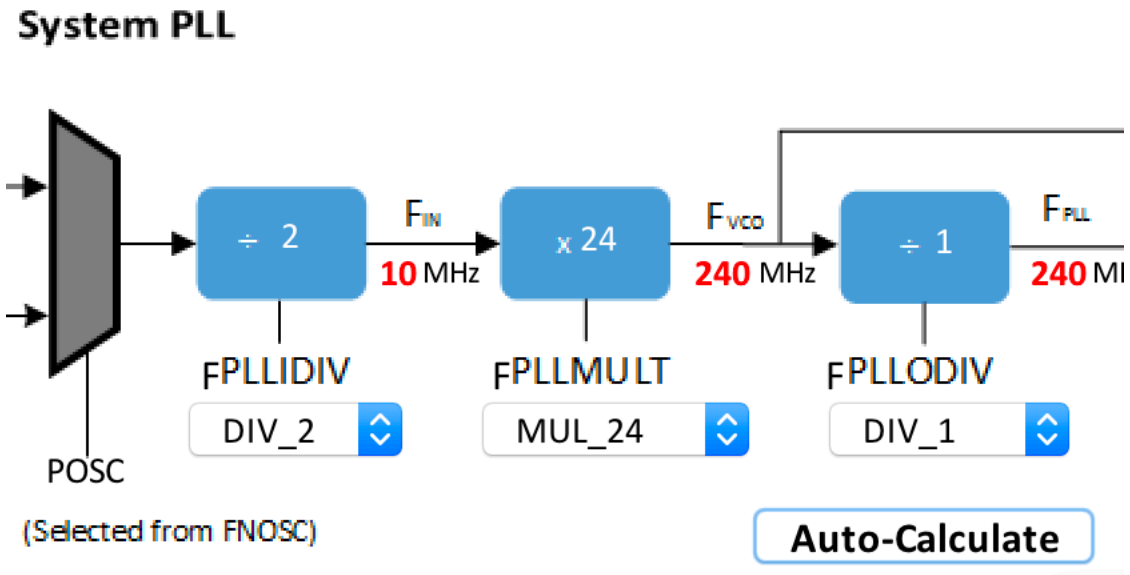


建立 Harmony 專案
Step 5：選擇震盪頻率類別、系統時脈來源

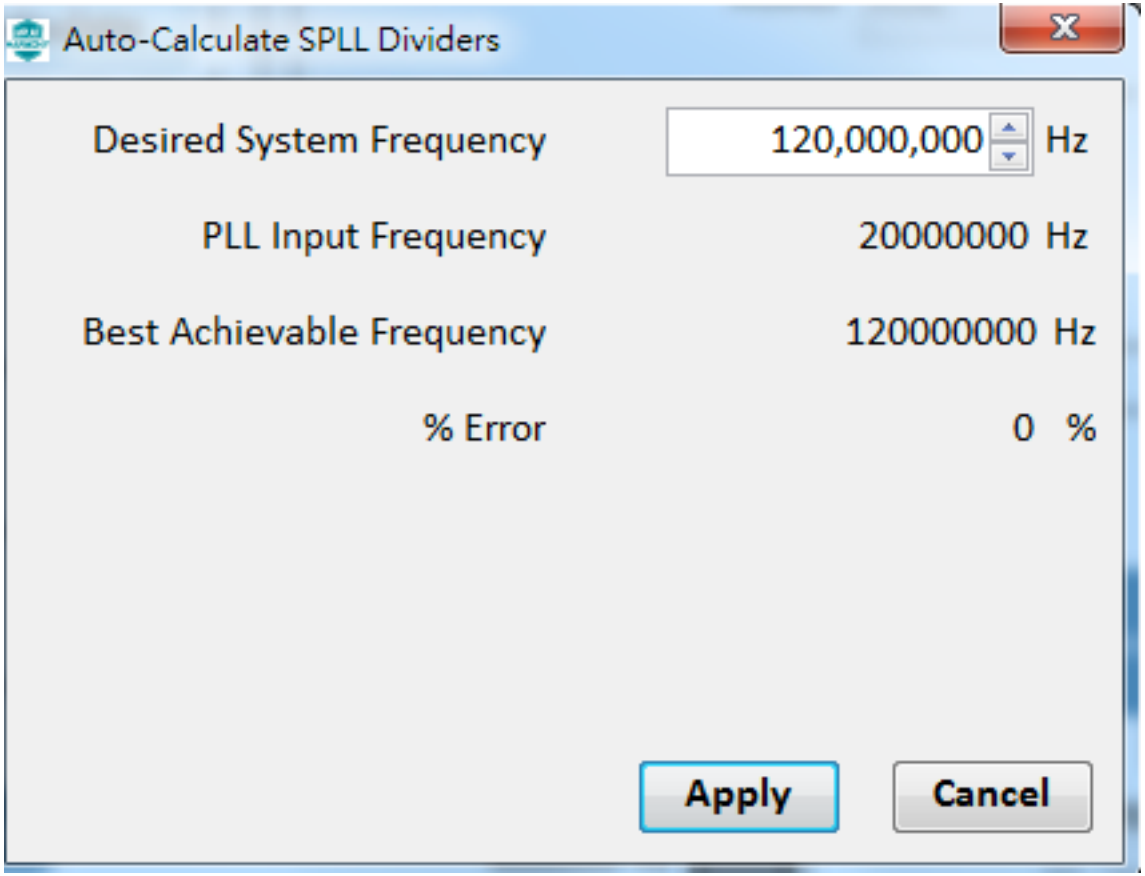




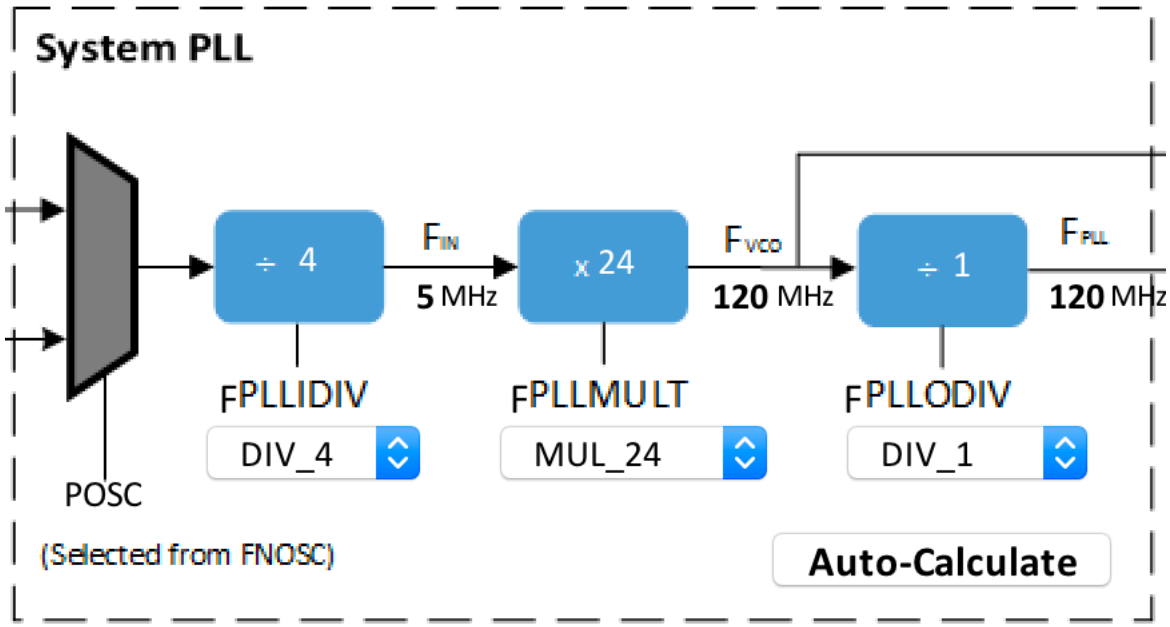
建立 Harmony 專案
Step 6：設定系統工作頻率



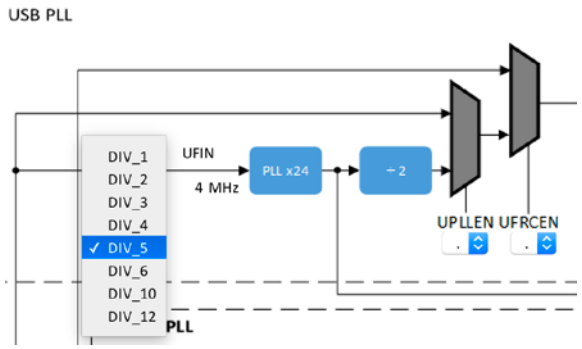
建立 Harmony 專案
Step 7：設定系統工作頻率為120MHz

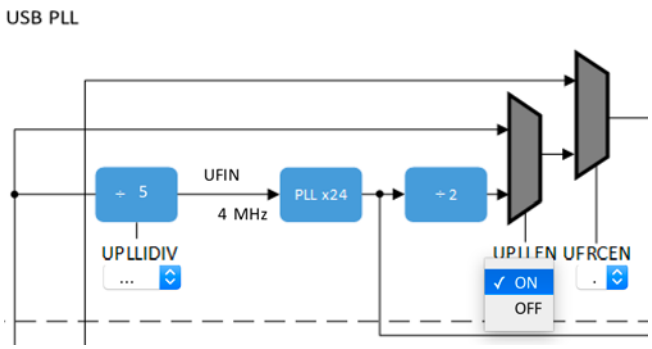


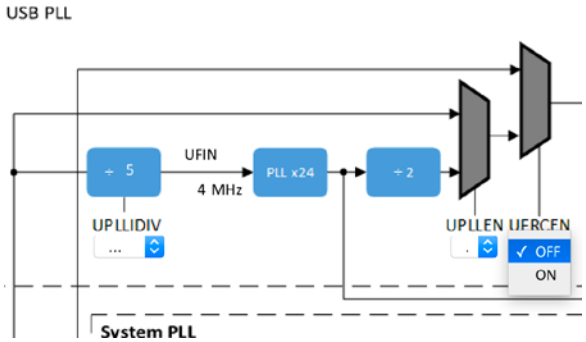
建立 Harmony 專案
Step 8：系統工作頻率為120MHz

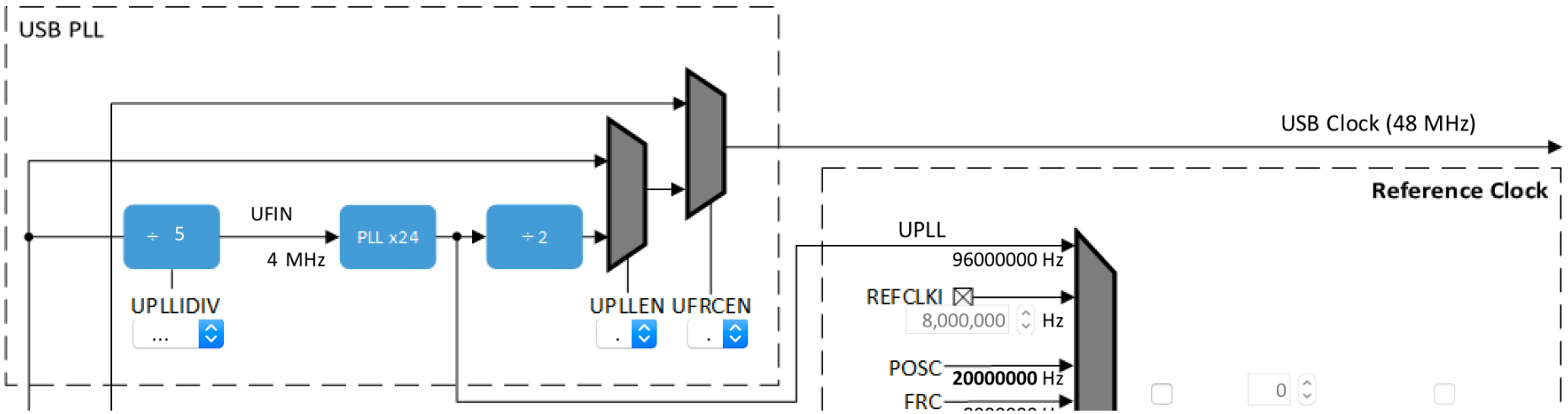


建立 Harmony 專案
Step 9：設定USB工作頻率為48MHz

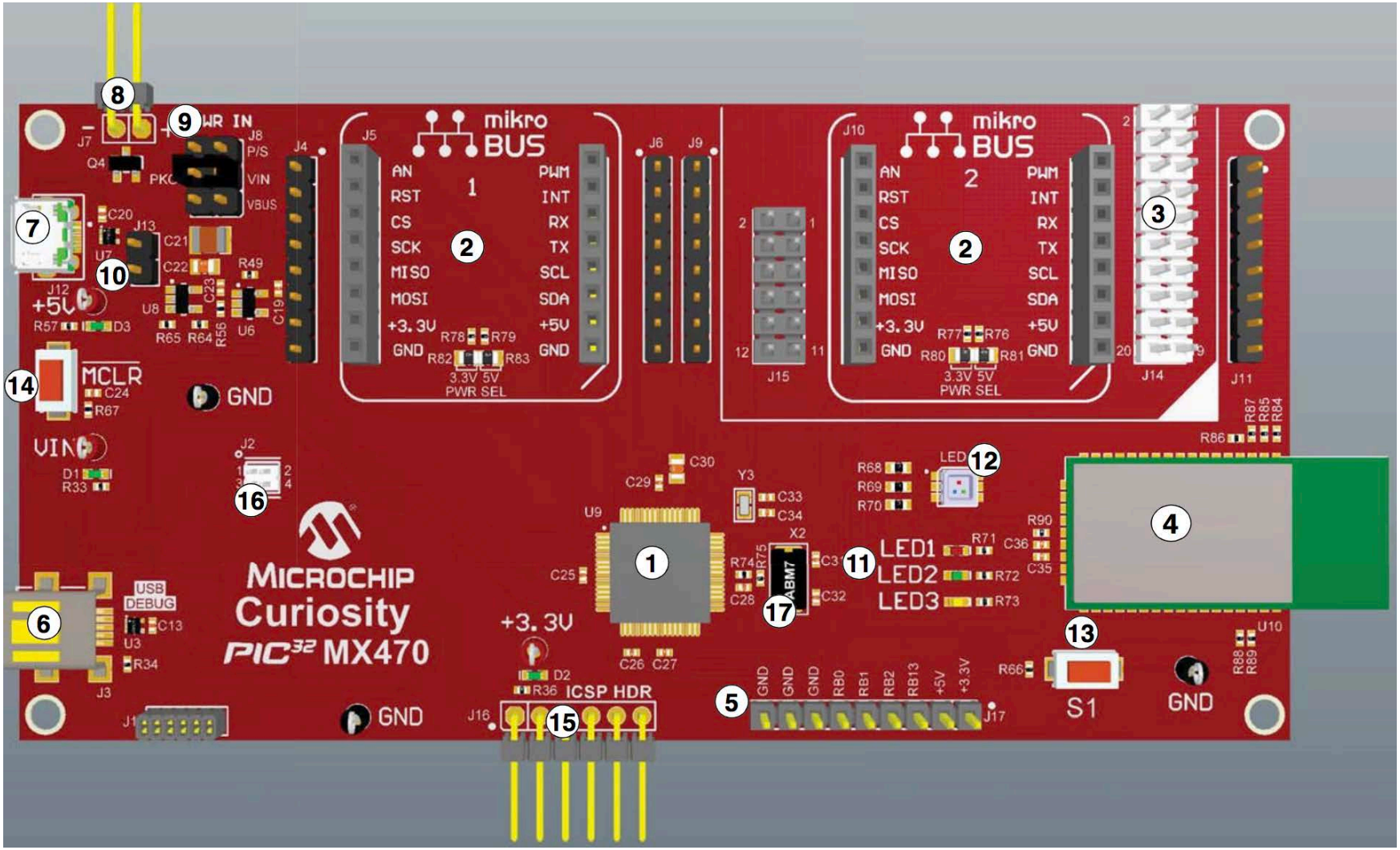




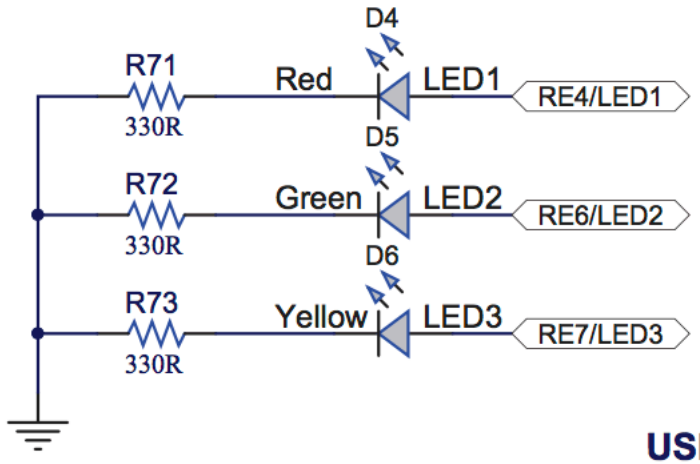




建立 Harmony 專案
Step 10：PIC32 Curiosity board

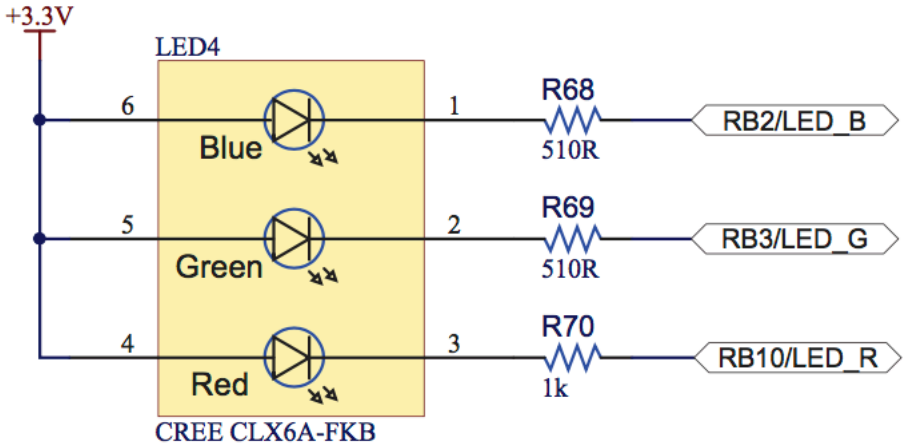


建立 Harmony 專案
Step 10：PIC32 Curiosity board -> GPIO 設定

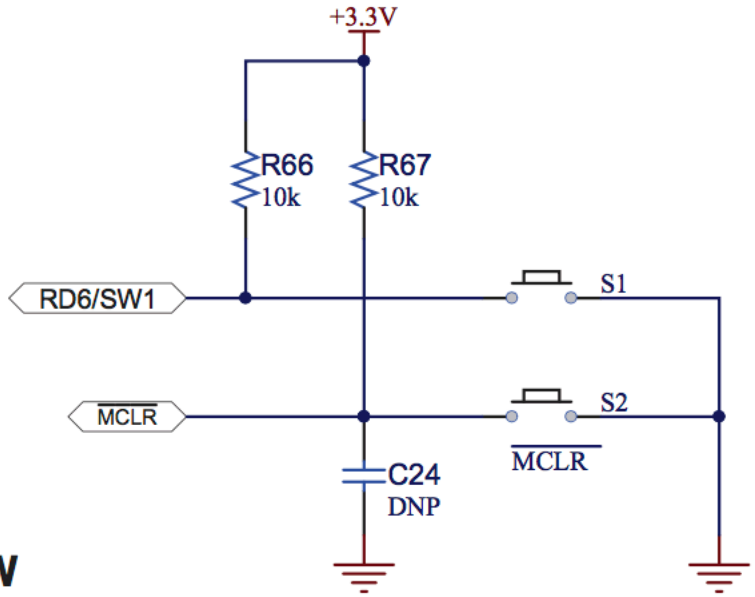


USER LEDs

CREE LED



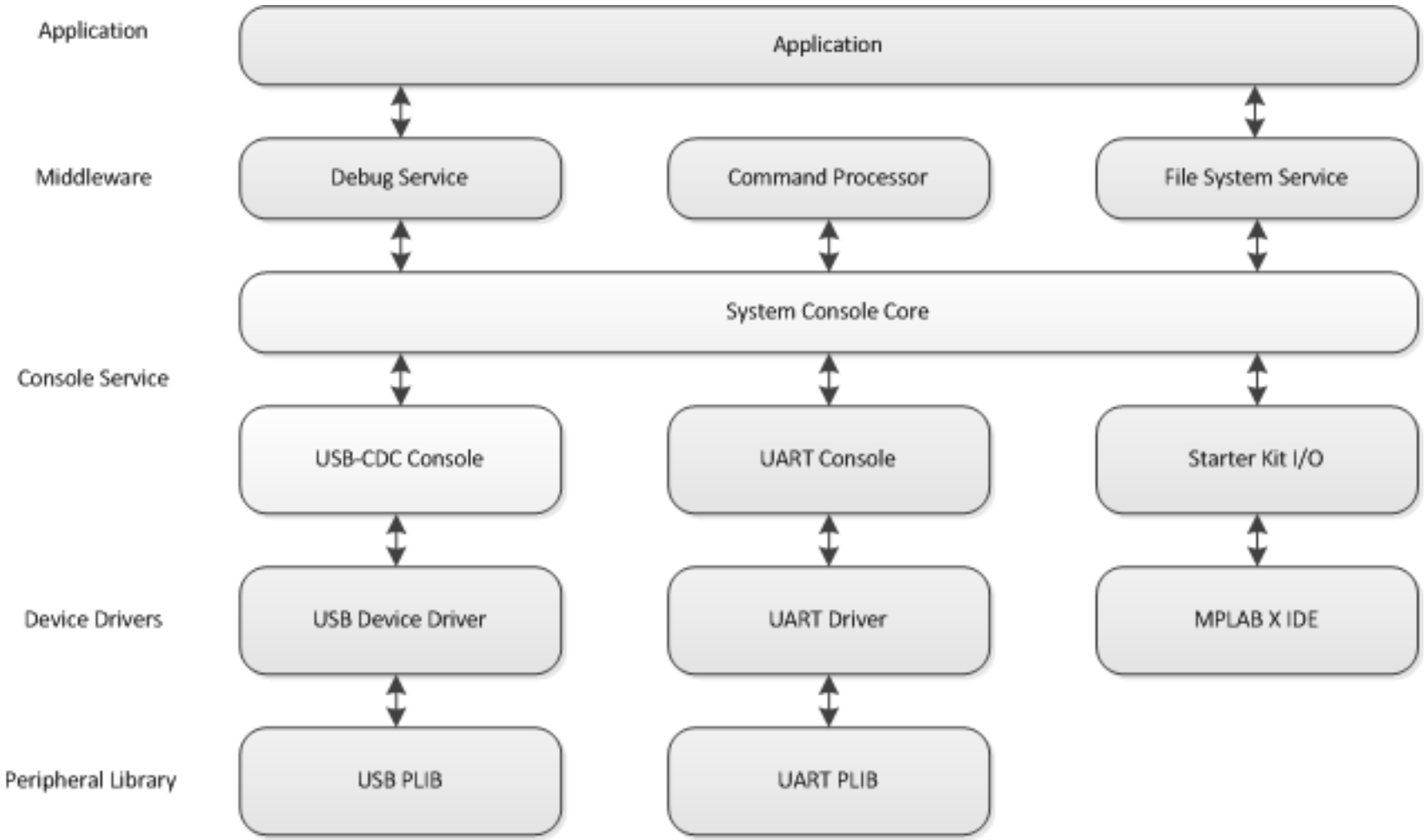
SW



建立 Harmony 專案
Step 10：MHC – Pin Settings -> GPIO 設定

Pin Number	Pin ID	Voltage Tolerance	Name	Function	Direction (TRIS)	Latch (LAT)	Open Drain (ODC)	Mode (ANSEL)	Change Notification (CNEN)	Pull Up (CNPU)	Pull Down (CNPD)
1	RE5			Available	In	n/a	☐	Analog	☐	☐	☐
2	RE6		BSP_LED_2	LED_AH	Out	Low	☐	Digital	☐	☐	☐
3	RE7		BSP_LED_3	LED_AH	Out	Low	☐	Digital	☐	☐	☐
4	RG6			Available	In	n/a	☐	Analog	☐	☐	☐
5	RG7			Available	In	n/a	☐	Analog	☐	☐	☐
6	RG8			Available	In	n/a	☐	Analog	☐	☐	☐
7	MCLR	5V			In	n/a	☐	Digital	☐	☐	☐
8	RG9			Available	In	n/a	☐	Analog	☐	☐	☐
9	VSS				In	n/a	☐	Digital	☐	☐	☐
10	VDD				In	n/a	☐	Digital	☐	☐	☐
11	RB5		USB_VBUS_SWITCH	VBUS	Out	Low	☐	Digital	☐	☐	☐
12	RB4			Available	In	n/a	☐	Analog	☐	☐	☐
13	RB3			Available	In	n/a	☐	Analog	☐	☐	☐
14	RB2			Available	In	n/a	☐	Analog	☐	☐	☐
15	RB1		BSP_BM64_WAKEUP	GPIO_OUT	Out	Low	☐	Digital	☐	☐	☐
16	RB0		BSP_BM64_RST	GPIO_OUT	Out	Low	☐	Digital	☐	☐	☐
54	RD6	5V	BSP_SWITCH_1	SWITCH	In	Low	☐	Digital	☐	☑	☐
55	RD7	5V	BSP_STBY_RST	GPIO_OUT	Out	Low	☐	Digital	☐	☐	☐
56	VCAP				In	n/a	☐	Digital	☐	☐	☐
57	VDD				In	n/a	☐	Digital	☐	☐	☐
58	RF0	5V		Available	In	n/a	☐	Digital	☐	☐	☐
59	RF1	5V		Available	In	n/a	☐	Digital	☐	☐	☐
60	RE0	5V	BSP_RGB_LED_RED	LED_AL	Out	Low	☐	Digital	☐	☐	☐
61	RE1	5V	BSP_RGB_LED_GREEN	LED_AL	Out	Low	☐	Digital	☐	☐	☐
62	RE2		BSP_RGB_LED_BLUE	LED_AL	Out	Low	☐	Digital	☐	☐	☐
63	RE3	5V		Available	In	n/a	☐	Digital	☐	☐	☐
64	RE4		BSP_LED_1	LED_AH	Out	Low	☐	Digital	☐	☐	☐

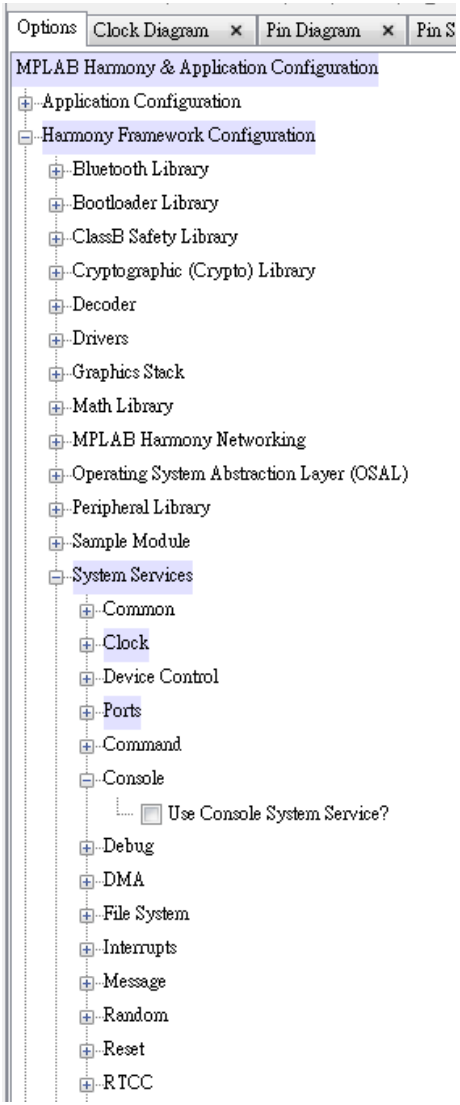
Console Service Software Abstraction Block Diagram

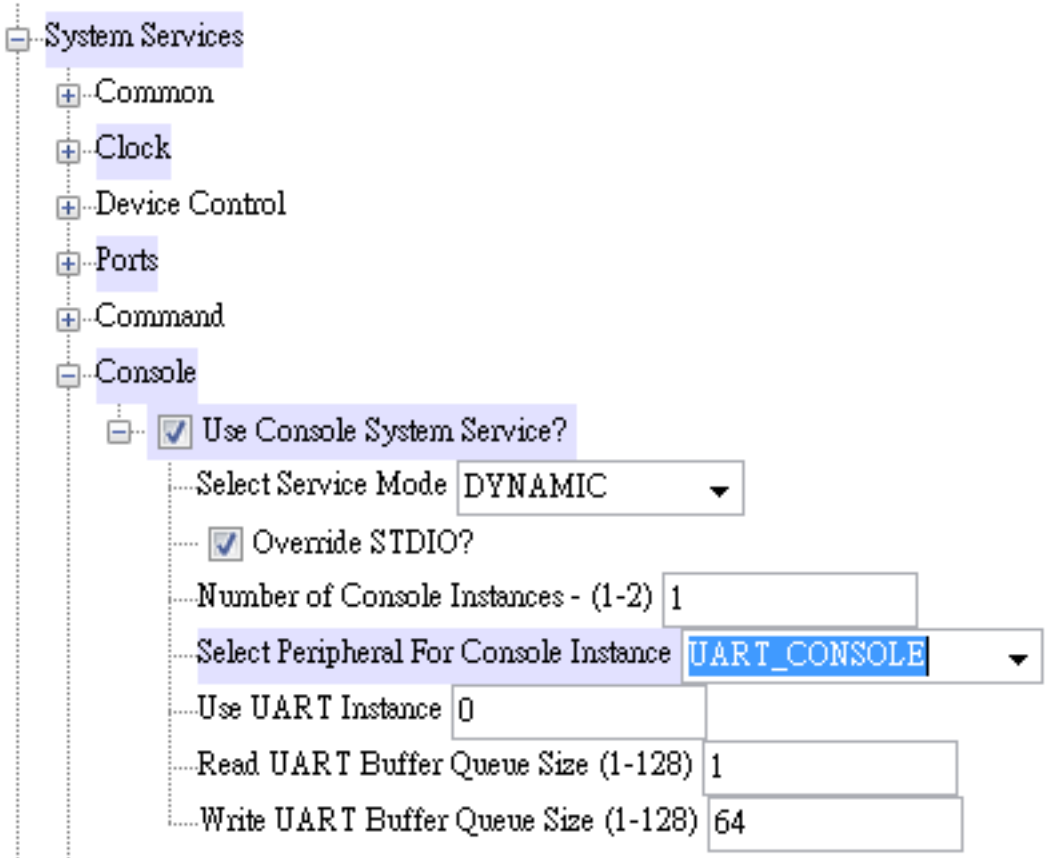


Generate “Console system service”

Step 1：Harmony Framework Configuration

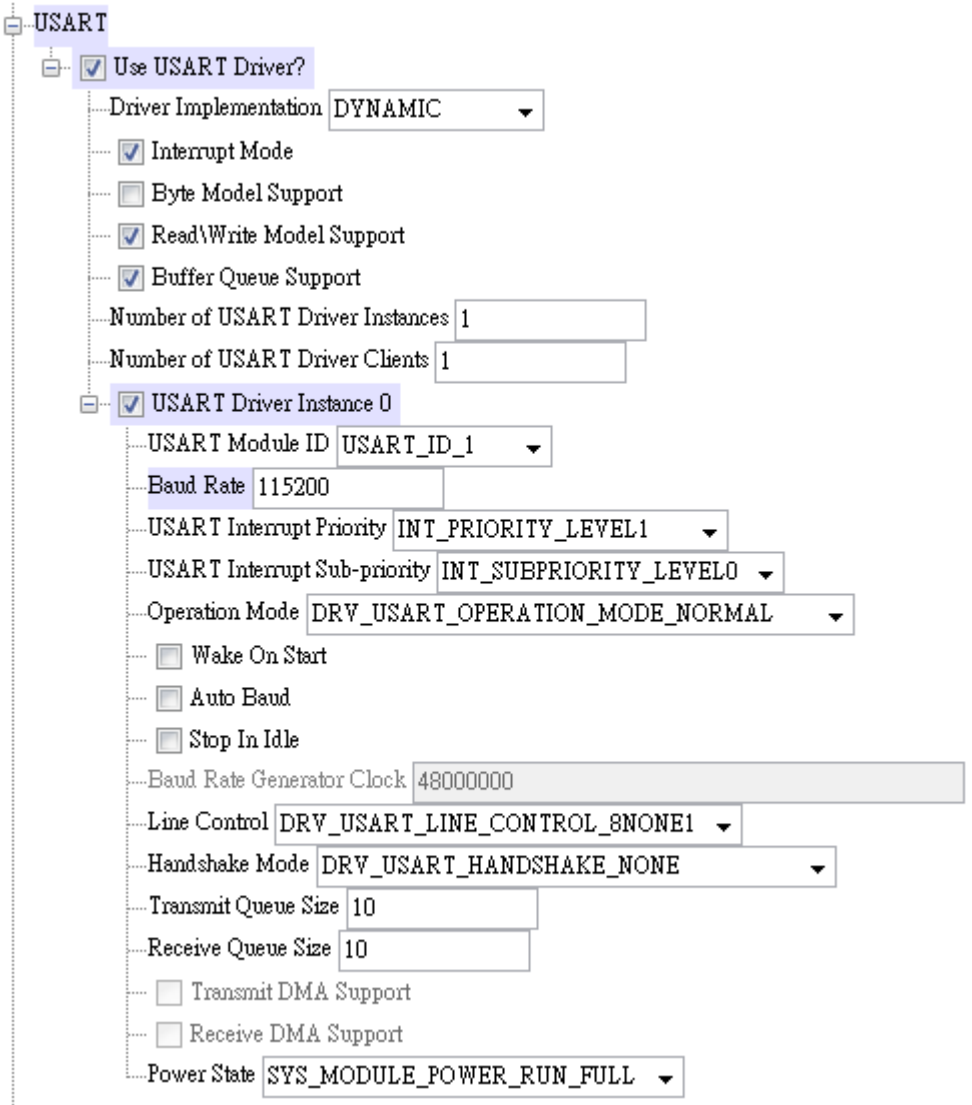
-> System Services -> Console





Generate “Console system service”

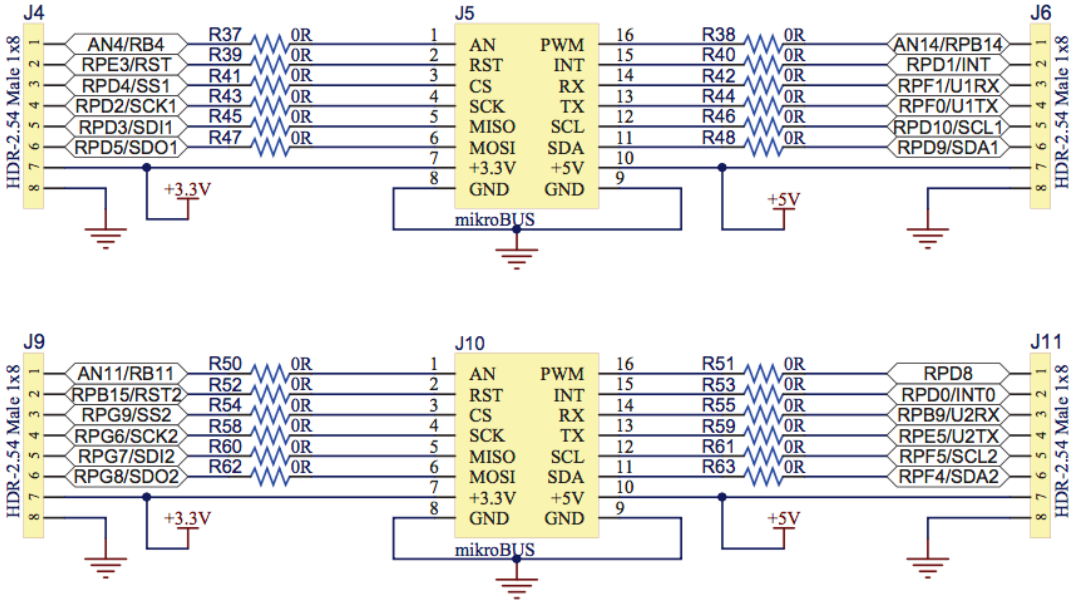
Step 2：Harmony Framework Configuration -> Drivers -> USART



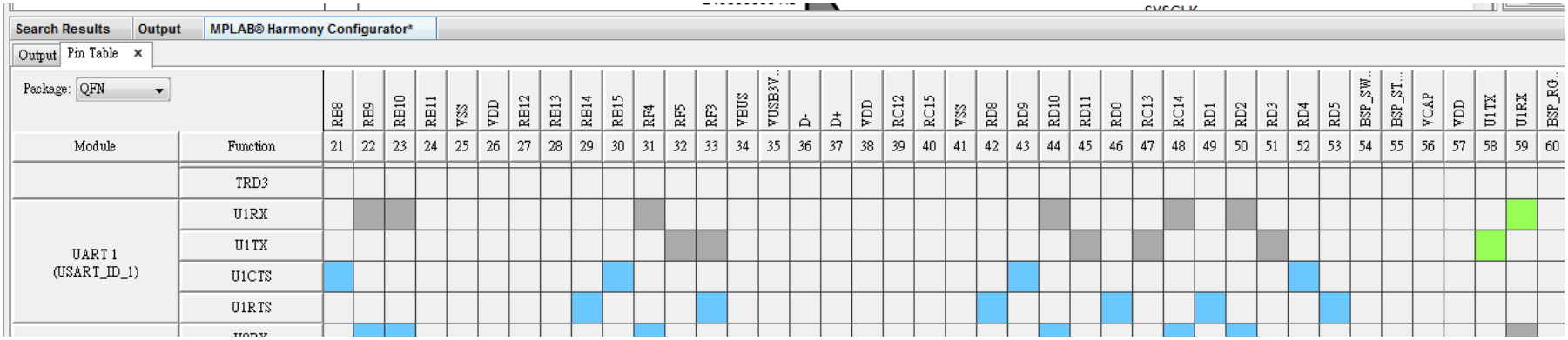
Generate “Console system service”

Step 5：Harmony Framework Configuration

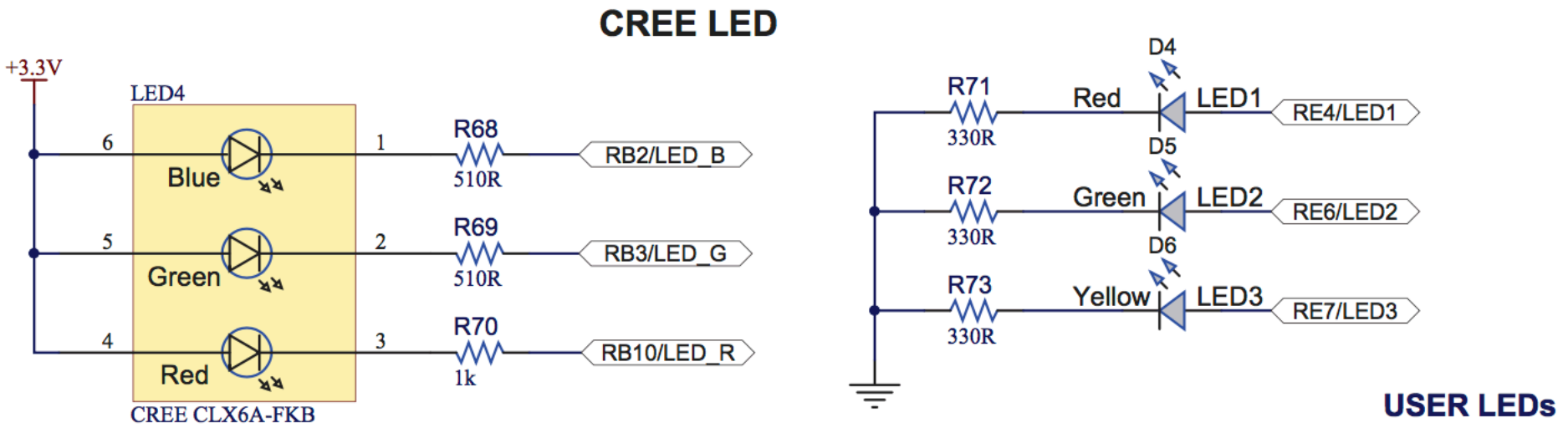
-> Pin Table



MikroBUS HDRs



LED 點亮及關閉
Step 3：檢查電路及BSP的設定



LED 點亮及關閉
Step 1：bsp.c 的函式

```
void BSP_LEDOn(BSP_LED led)
{
    if(led_active_level_map[led] == BSP_LED_ACTIVE_HIGH)
    {
        PLIB_PORTS_PinSet( PORTS_ID_0, led_port_channel_map[led], led_port_bit_pos_map[led] );
    }
    else
    {
        PLIB_PORTS_PinClear( PORTS_ID_0, led_port_channel_map[led], led_port_bit_pos_map[led] );
    }
}

void BSP_LEDOff(BSP_LED led)
{
    if(led_active_level_map[led] == BSP_LED_ACTIVE_HIGH)
    {
        PLIB_PORTS_PinClear( PORTS_ID_0, led_port_channel_map[led], led_port_bit_pos_map[led] );
    }
    else
    {
        PLIB_PORTS_PinSet( PORTS_ID_0, led_port_channel_map[led], led_port_bit_pos_map[led] );
    }
}
```

LED 點亮及關閉
Step 1：bsp.h 的函式

```
typedef enum
{
    BSP_LED_2 = 0,
    BSP_LED_3 = 1,
    BSP_RGB_LED_GREEN = 2,
    BSP_RGB_LED_BLUE = 3,
    BSP_RGB_LED_RED = 4,
    BSP_LED_1 = 5
} BSP_LED;
```

```
static const PORTS_BIT_POS led_port_bit_pos_map[] =
{
    PORTS_BIT_POS_6,
    PORTS_BIT_POS_7,
    PORTS_BIT_POS_3,
    PORTS_BIT_POS_2,
    PORTS_BIT_POS_10,
    PORTS_BIT_POS_4
};
```

LED 點亮及關閉
Step 1：main.c 加入 BSP_LEDOn() / BSP_LEDOff()

```
#include <stddef.h>                     // Defines NULL
#include <stdbool.h>                    // Defines true
#include <stdlib.h>                     // Defines EXIT_FAILURE
#include "system/common/sys_module.h"   // SYS function prototypes
#include "bsp.h"

int main ( void )
{
    /* Initialize all MPLAB Harmony modules, including application(s). */
    SYS_Initialize ( NULL );

    while ( true )
    {
        /* Maintain state machines of all polled MPLAB Harmony modules. */
        BSP_LEDOn(BSP_LED_1);
        BSP_LEDOn(BSP_LED_2);
        BSP_LEDOn(BSP_LED_3);
        BSP_LEDOff(BSP_LED_1);
        BSP_LEDOff(BSP_LED_2);
        BSP_LEDOff(BSP_LED_3);

        BSP_LEDOn(BSP_RGB_LED_RED);
        BSP_LEDOn(BSP_RGB_LED_GREEN);
        BSP_LEDOn(BSP_RGB_LED_BLUE);
        BSP_LEDOff(BSP_RGB_LED_RED);
        BSP_LEDOff(BSP_RGB_LED_GREEN);
        BSP_LEDOff(BSP_RGB_LED_BLUE);

        SYS_Tasks ( );
    }
    /* Execution should not come here during normal operation */
    return ( EXIT_FAILURE );
}
```

Generate “Console system service”

Step 2：app.c -> APP_Tasks

```
ssize_t nr;
char myBuffer[] = "message";
```

```
case APP_STATE_SERVICE_TASKS:
{
    nr = SYS_CONSOLE_Write( SYS_CONSOLE_INDEX_0, STDOUT_FILENO, myBuffer, strlen(myBuffer) );
    if (nr != strlen(myBuffer))
    {
        // Handle error
    }
    appData.state = APP_STATE_IDLE;
    break;
}
case APP_STATE_IDLE:

    break;
```

Generate “Console system service”

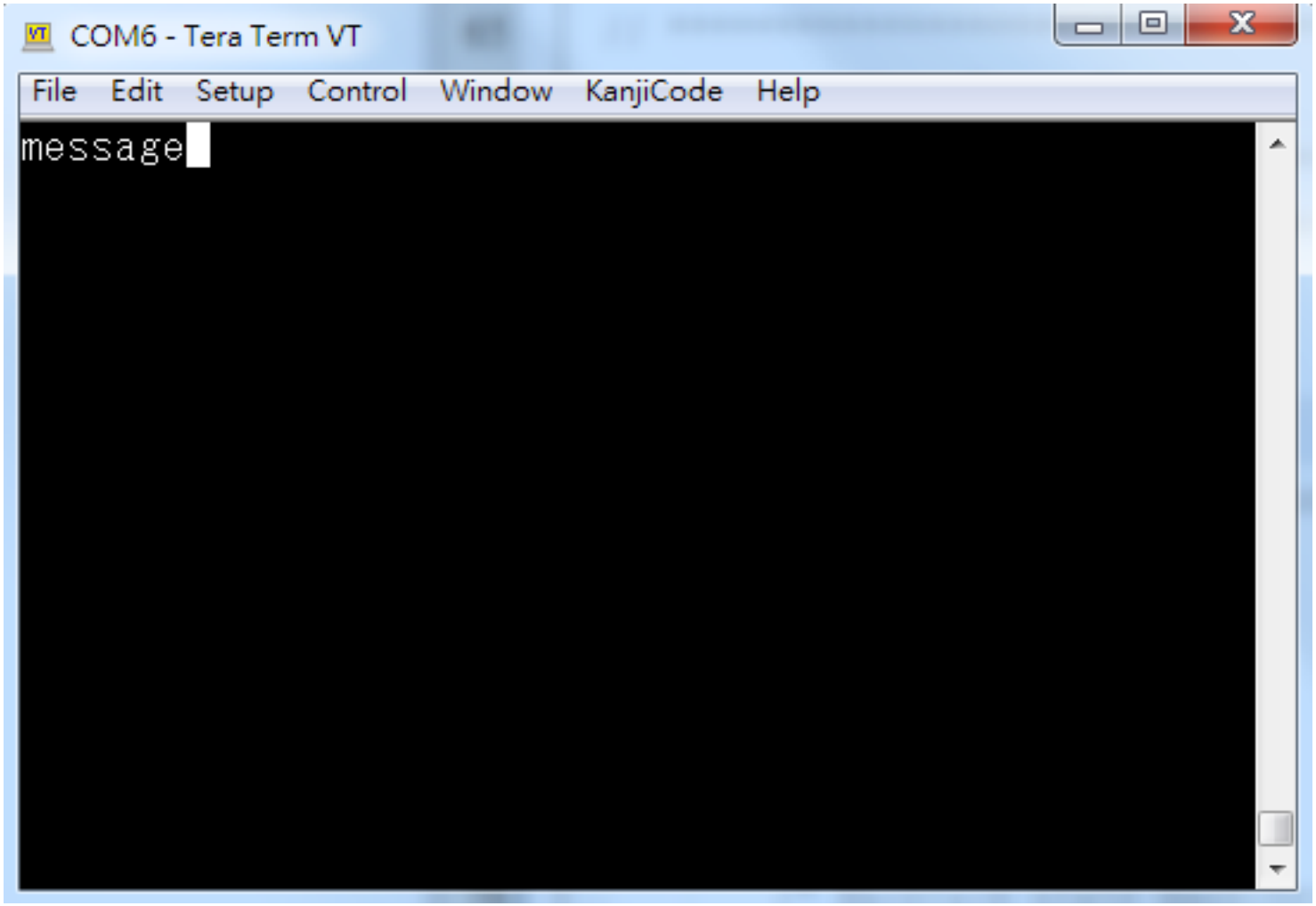
Step 2：app.h -> APP_STATES

```
typedef enum
{
    /* Application's state machine's initial state. */
    APP_STATE_INIT=0,
    APP_STATE_SERVICE_TASKS,
    APP_STATE_IDLE,
    /* TODO: Define states used by the application state machine. */

} APP_STATES;
```

Generate “Console system service”

Step 8：編譯、執行



Thank You

Any Questions?