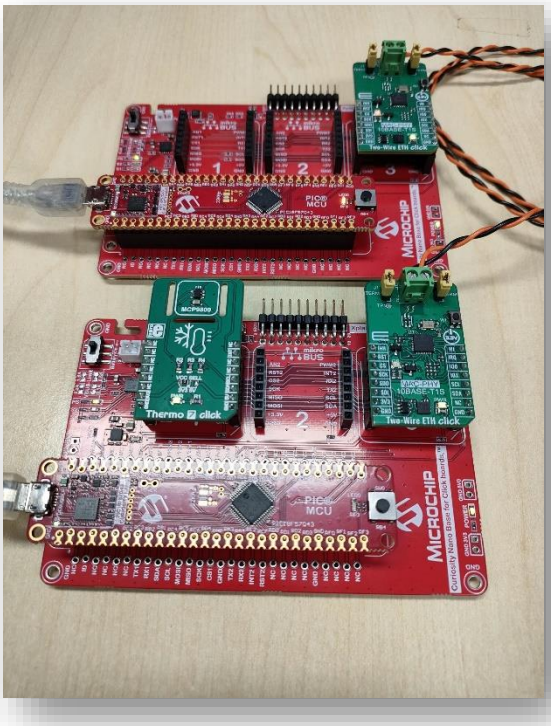
8-bit 10Base-T1S Project Example

Using LAN8651B Two Wire ETH & Thermo 7 Click



Objective

Demonstrate the use of 10Base-T1S using 8-bit MCU's

Hardware

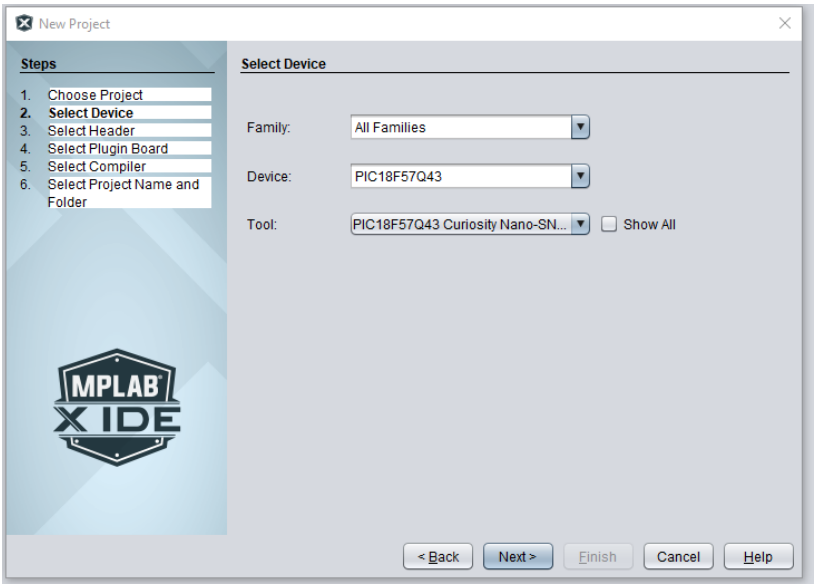
- 2x Two-Wire ETH Click Boards LAN8651B
- 2x Curiosity Nano Base Boards AC164162
- 2x PIC18F57Q43 Curiosity Nanos DM164150
- At least 1x Thermo 7 Click

Software

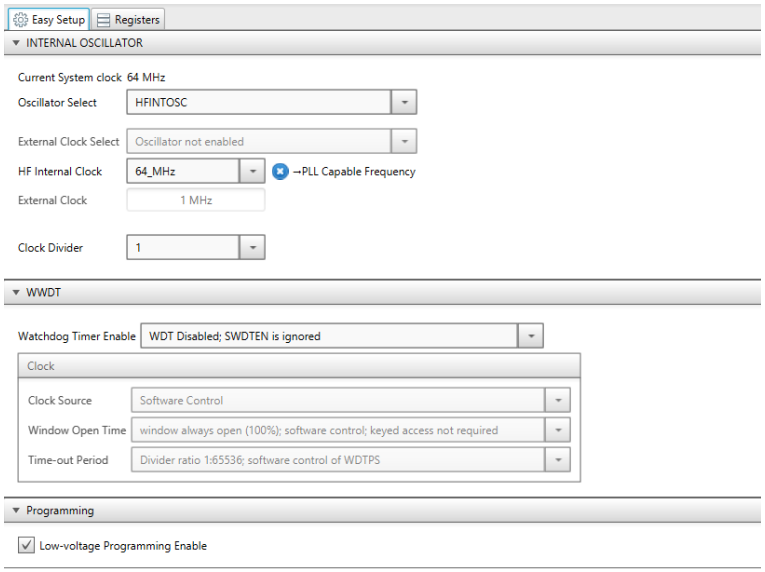
- **MPLABX v6.05**
- **TC6 Library**
  https://github.com/MicrochipTech/oa-tc6-lib
- **Harmony 3 + MCC**

Project Setup in MCC

1. **Create a new project in MPLABX:**
   - Select Device
   - Select Compiler
   - Name Project

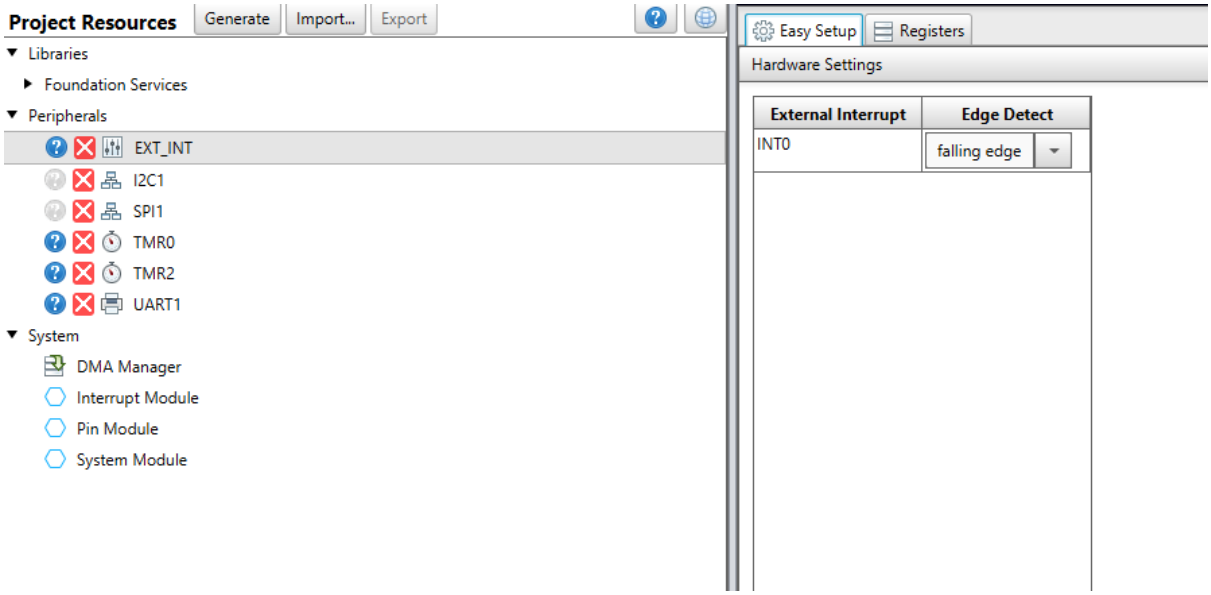


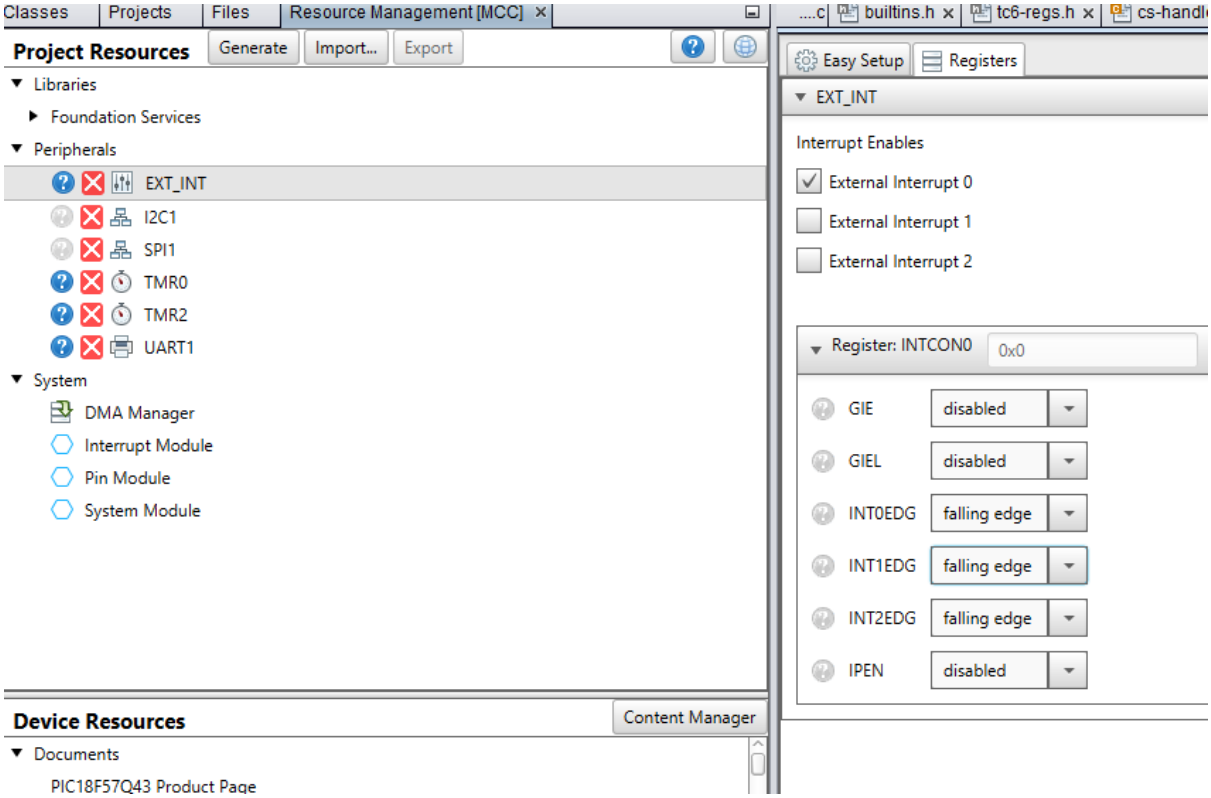
2. **System Module:**
   - HFINTOSC
   - 64Mhz Internal Clock
   - Low-voltage Programming



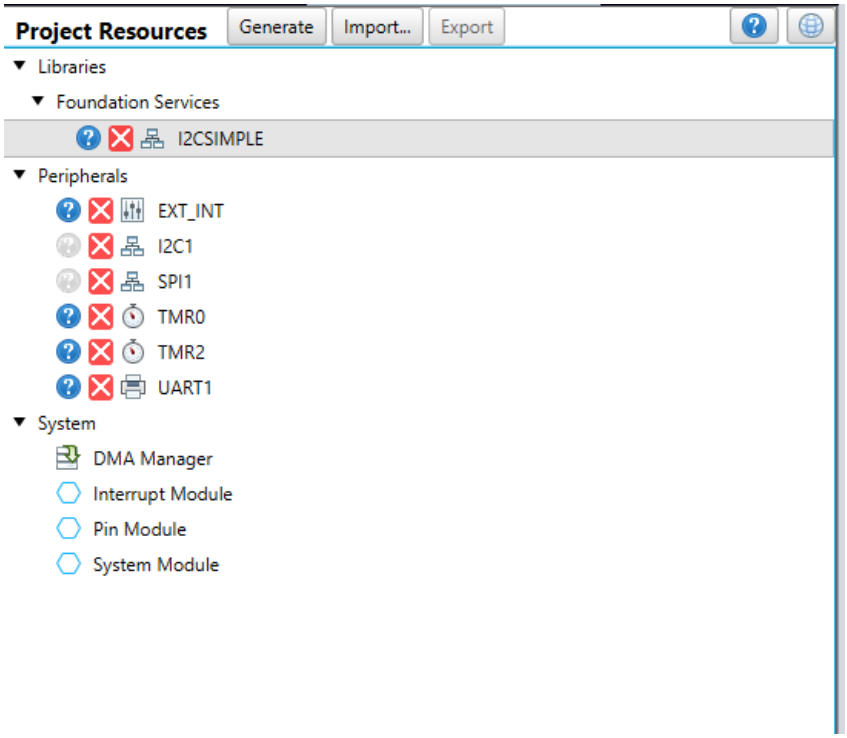
3. Add EXT_INT to Project

- Enable External Interrupt 0 (INT0)
- Edge Detect: Falling edge

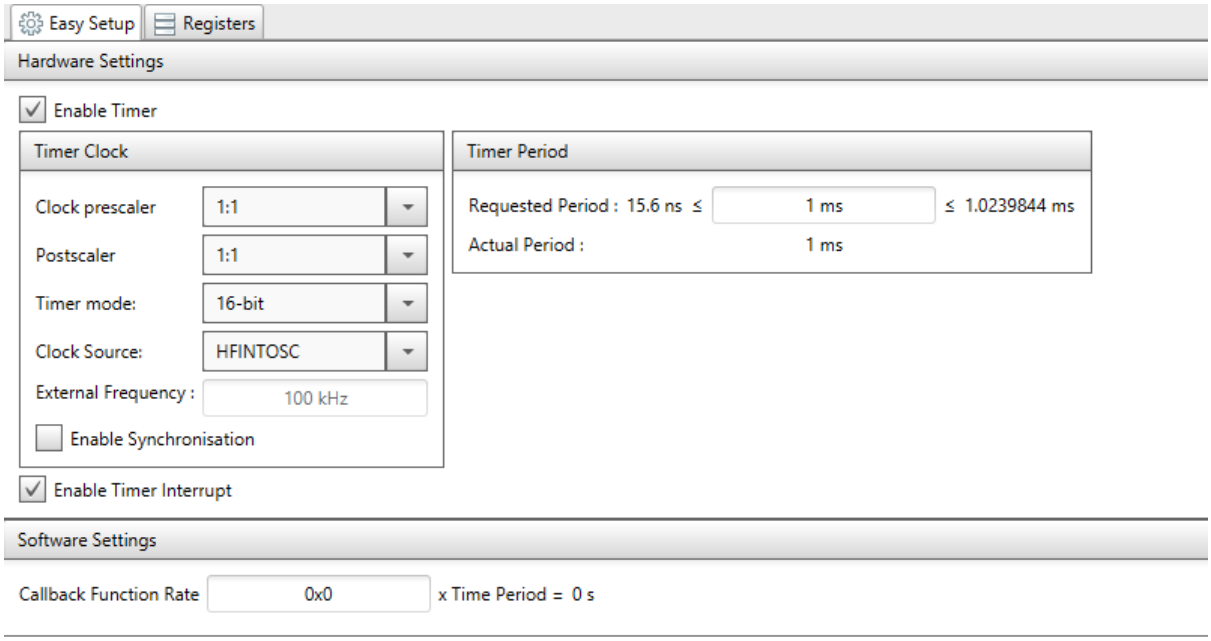




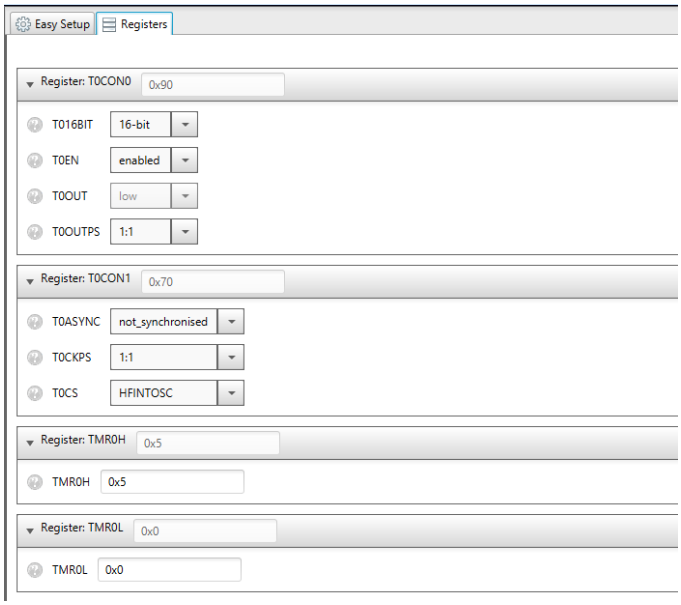
4. **Add to project.**
    - SPI1
    - TMR0
    - TMR2
    - UART1
    - I2C1 & I2CSIMPLE



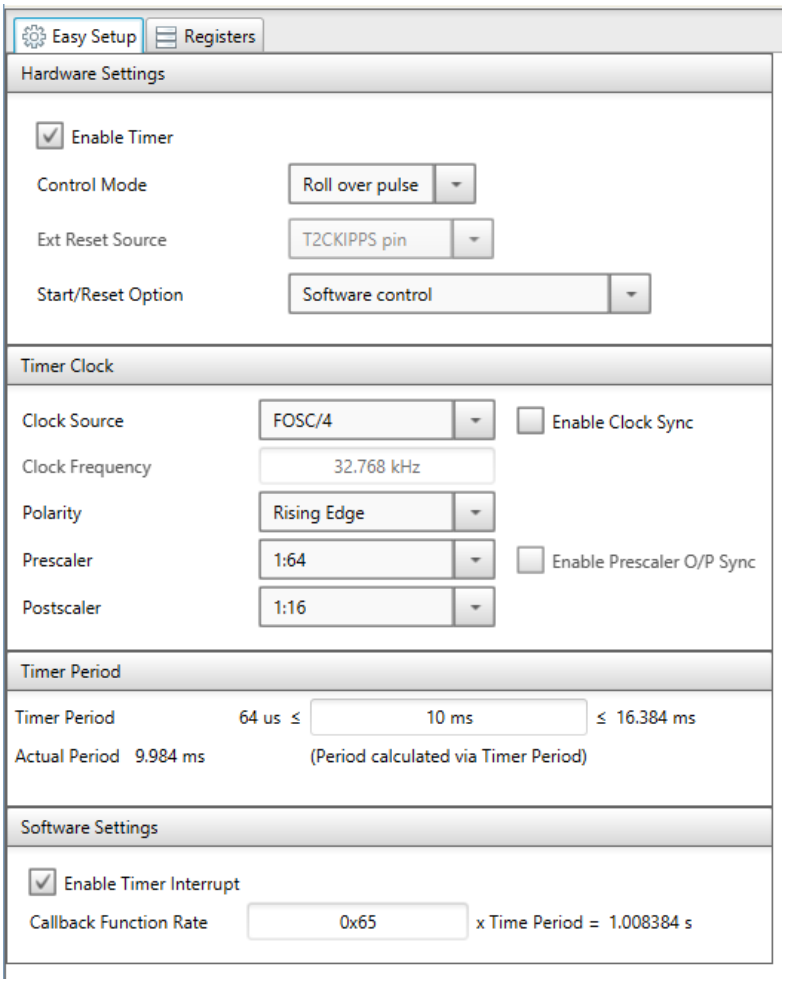
5. **Setup Timer0**
    - Choose settings as image below



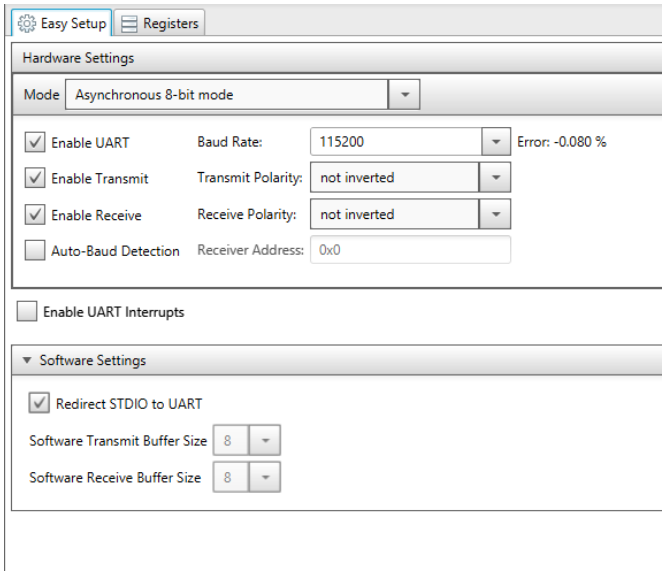
- Configure TMR0 registers:
    - T0CON0 = 0x90
    - T0CON1 = 0x70
    - TMROH = 0x05
    - TMROL = 0x00



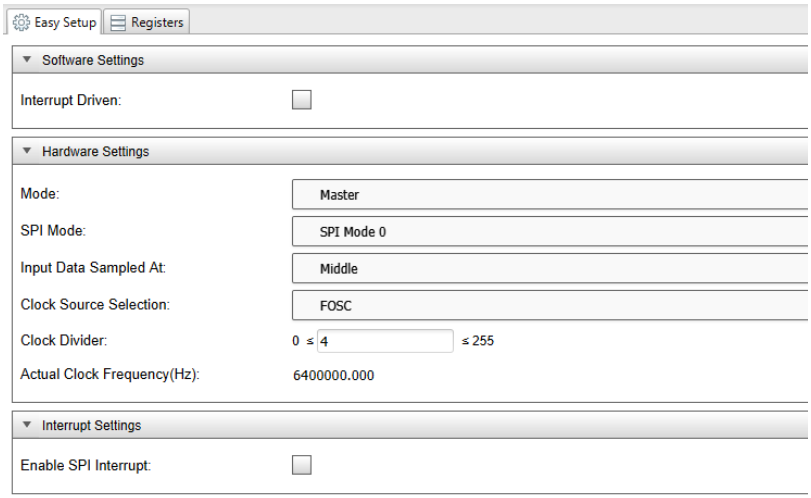
6. **Setup Timer 2**
    - The Interrupt for the temperature checking can be adjusted for the time using the callback function rate.



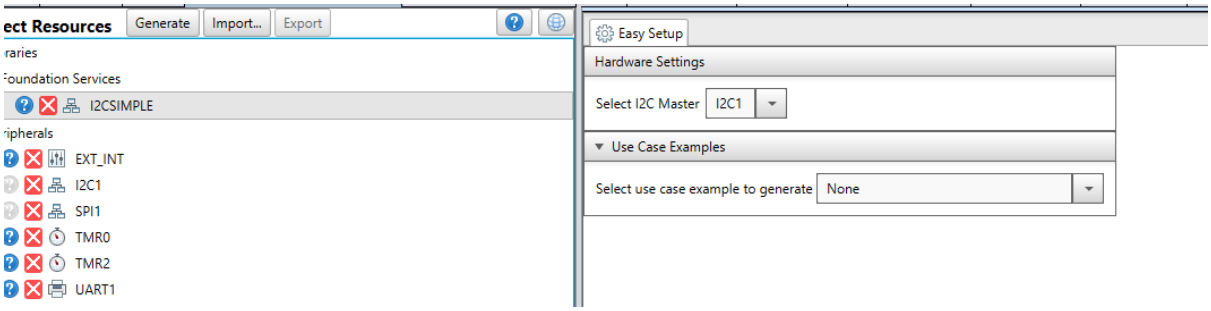
7. **Setup UART**
    - Baud Rate 115200
    - Enable Transmit and Receive
    - Do not enable UART interrupts
    - Redirect STDIO



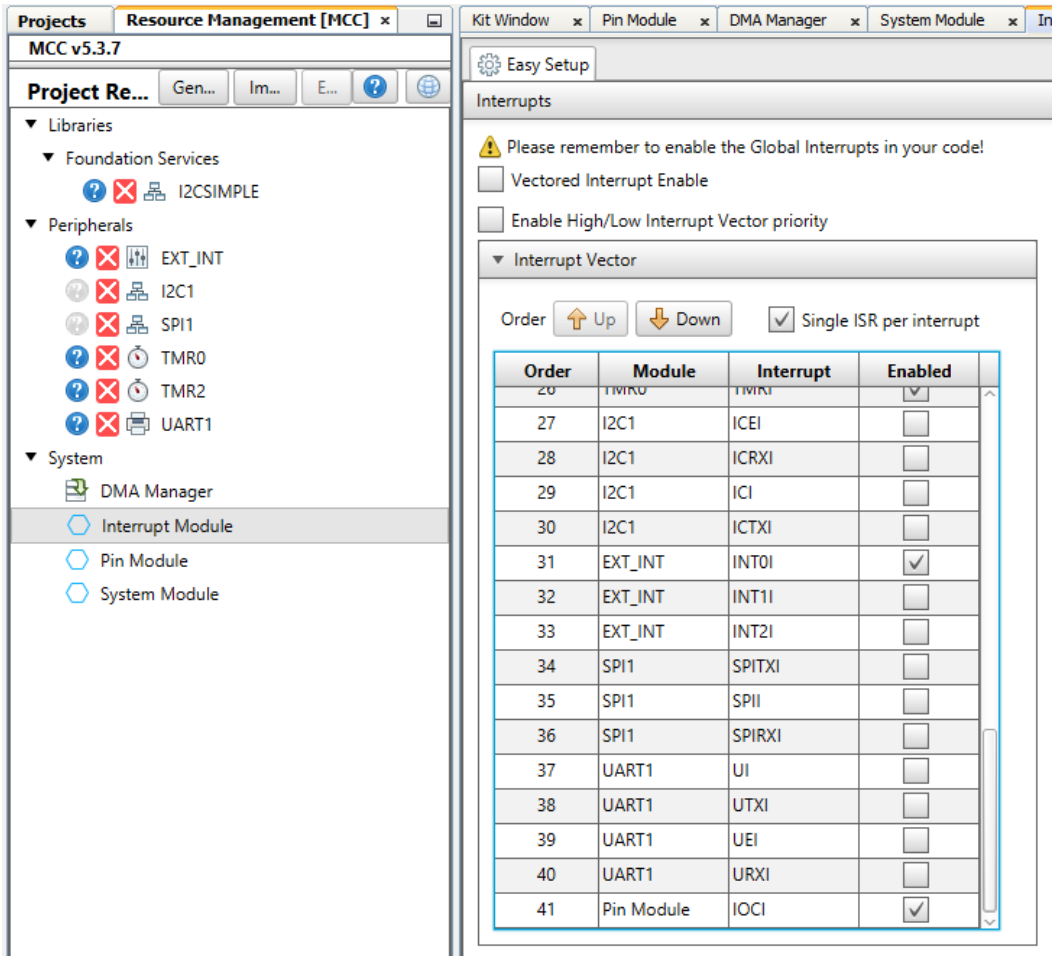
8. **Setup SPI as below**



9. **Setup i2c module**
    - Leave as default

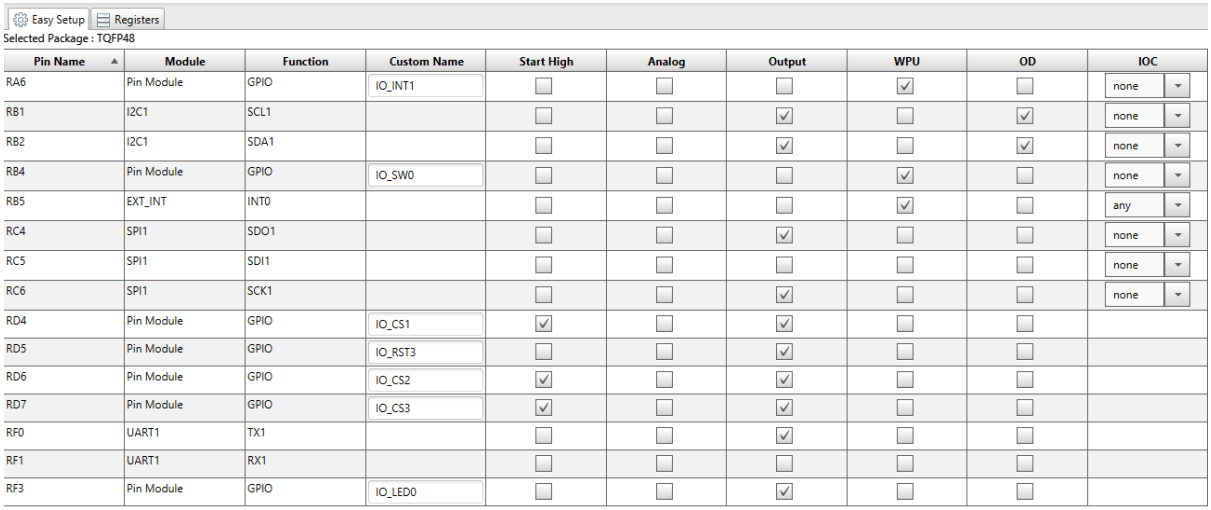


10. **Interrupt Module**
    - Enable INT0I
    - Enable TMRI
    - Enable TMR2
    - Enable IOCI



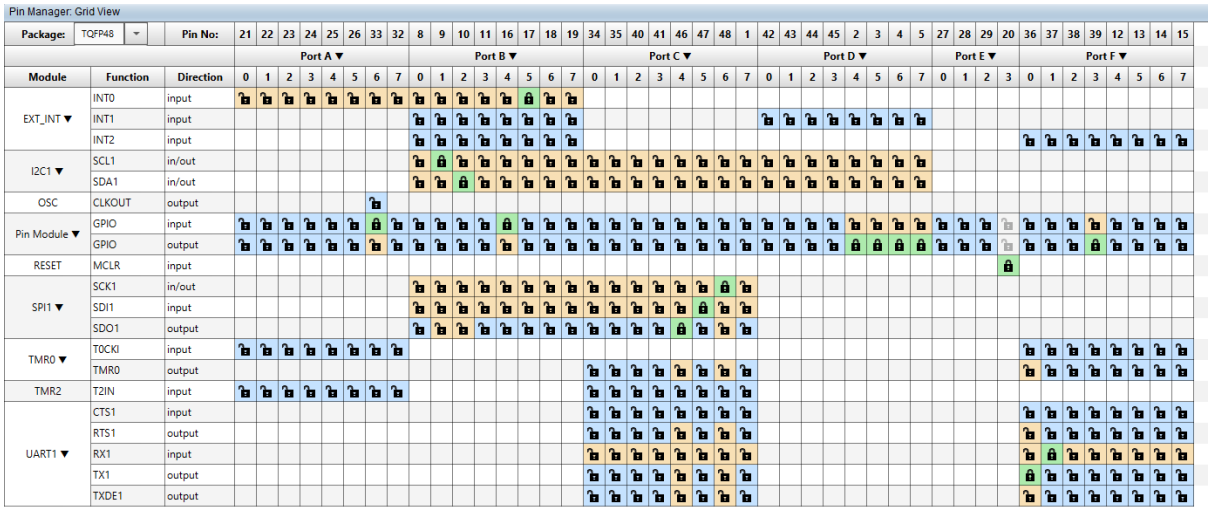
11. Pin Module

- Setup as below

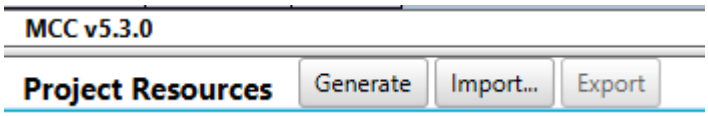


12. Pin Manager

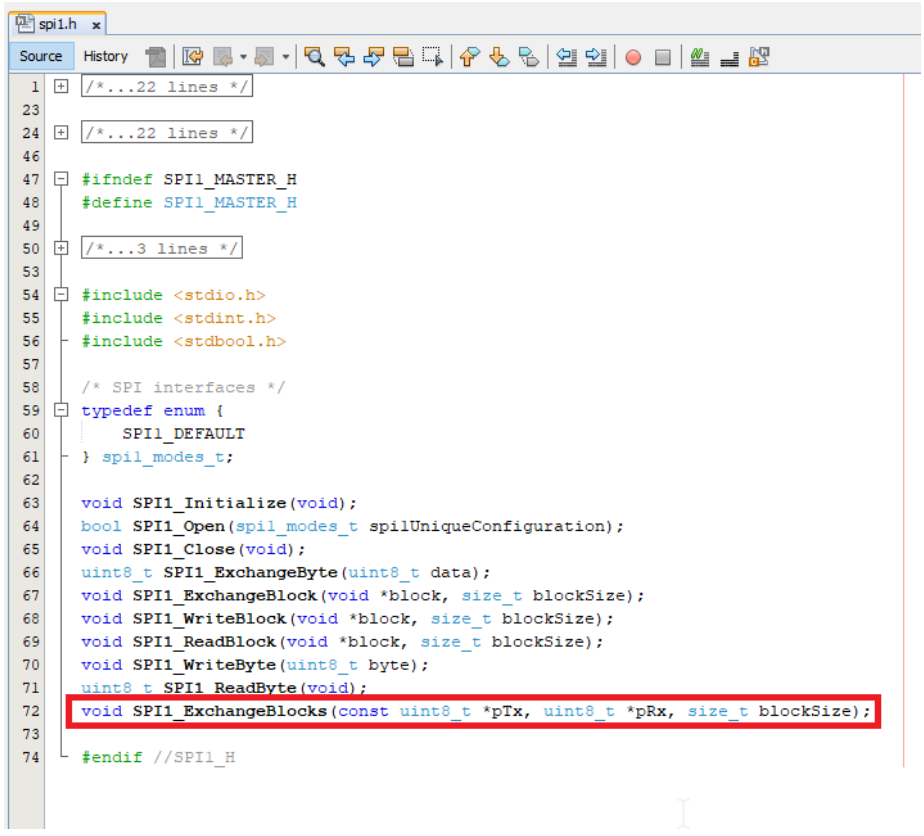
- Setup as below



13. Generate the Code in MCC



- Add the following function to spi1.c and the function declaration in spi1.h




```
void SPI1_ExchangeBlocks(const uint8_t *pTx, uint8_t *pRx, size_t blockSize);
```

spi1.c

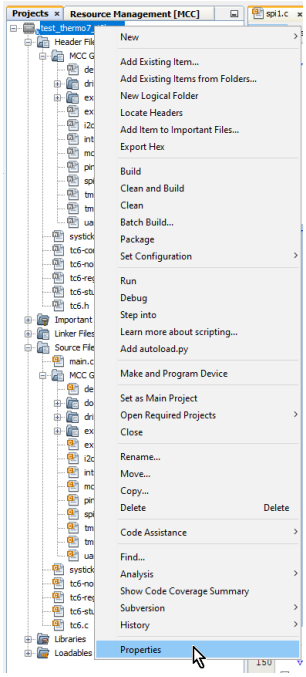
Source | History

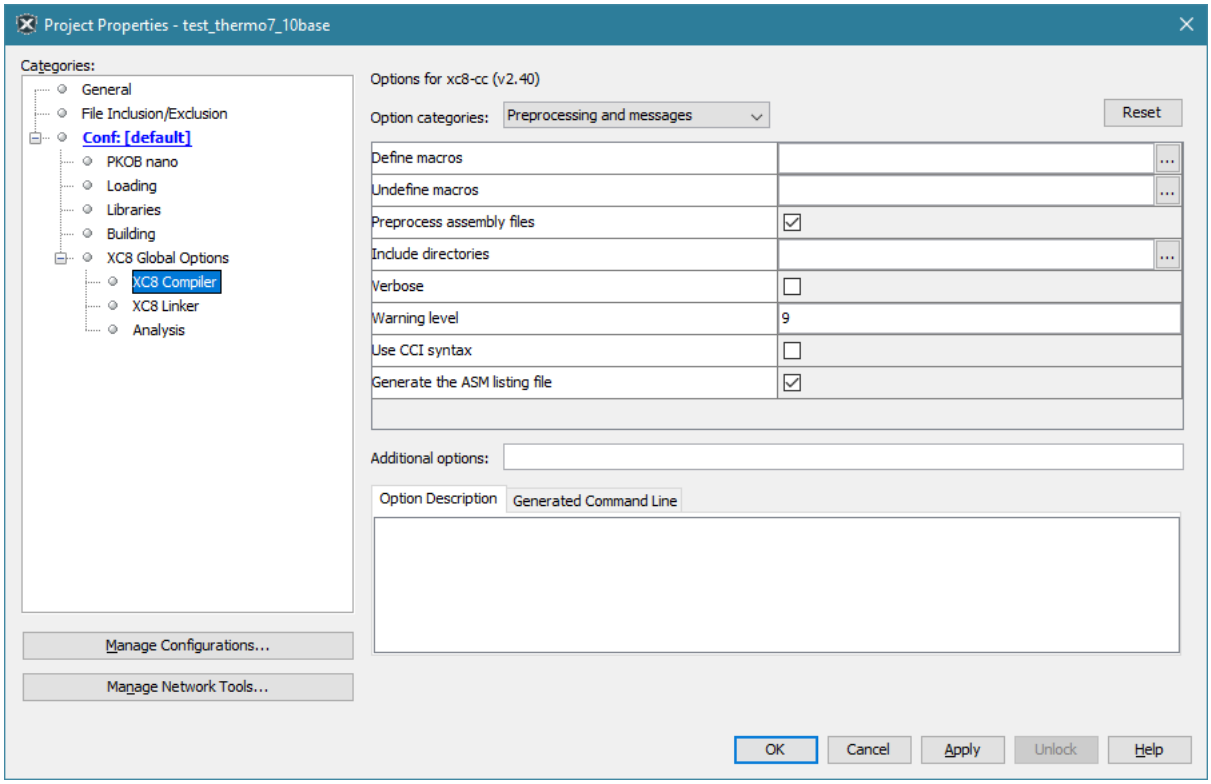
```
106
107  void SPI1_ExchangeBlock(void *block, size_t blockSize)
108  {
109      uint8_t *data = block;
110      while(blockSize--)
111      {
112          SPI1TCNTL = 1;
113          SPI1TXB = *data;
114          while(!PIR3bits.SPI1RXIF);
115          *data++ = SPI1RXB;
116      }
117  }
118
119  void SPI1_ExchangeBlocks(const uint8_t *pTx, uint8_t *pRx, size_t blockSize)
120  {
121      size_t i;
122      if (pTx && pRx) {
123          for (i = 0; i < blockSize; i++)
124          {
125              SPI1TCNTL = 1;
126              SPI1TXB = pTx[i];
127              while(!PIR3bits.SPI1RXIF);
128              pRx[i] = SPI1RXB;
129          }
130      } else if (pTx) {
131          for (i = 0; i < blockSize; i++)
132          {
133              SPI1TCNTL = 1;
134              SPI1TXB = pTx[i];
135              while(!PIR3bits.SPI1RXIF);
136              (void)SPI1RXB;
137          }
138      } else if (pRx) {
139          for (i = 0; i < blockSize; i++)
140          {
141              SPI1TCNTL = 1;
142              SPI1TXB = 0xFF;
143              while(!PIR3bits.SPI1RXIF);
144              pRx[i] = SPI1RXB;
145          }
146      }
147  }
```

```
void SPI1_ExchangeBlocks(const uint8_t *pTx, uint8_t *pRx, size_t blockSize)
{
    size_t i;
    if (pTx && pRx) {
        for (i = 0; i < blockSize; i++)
        {
            SPI1TCNTL = 1;
            SPI1TXB = pTx[i];
            while(!PIR3bits.SPI1RXIF);
            pRx[i] = SPI1RXB;
        }
    } else if (pTx) {
        for (i = 0; i < blockSize; i++)
        {
            SPI1TCNTL = 1;
            SPI1TXB = pTx[i];
            while(!PIR3bits.SPI1RXIF);
            (void)SPI1RXB;
        }
    } else if (pRx) {
        for (i = 0; i < blockSize; i++)
        {
            SPI1TCNTL = 1;
            SPI1TXB = 0xFF;
            while(!PIR3bits.SPI1RXIF);
            pRx[i] = SPI1RXB;
        }
    }
}
```

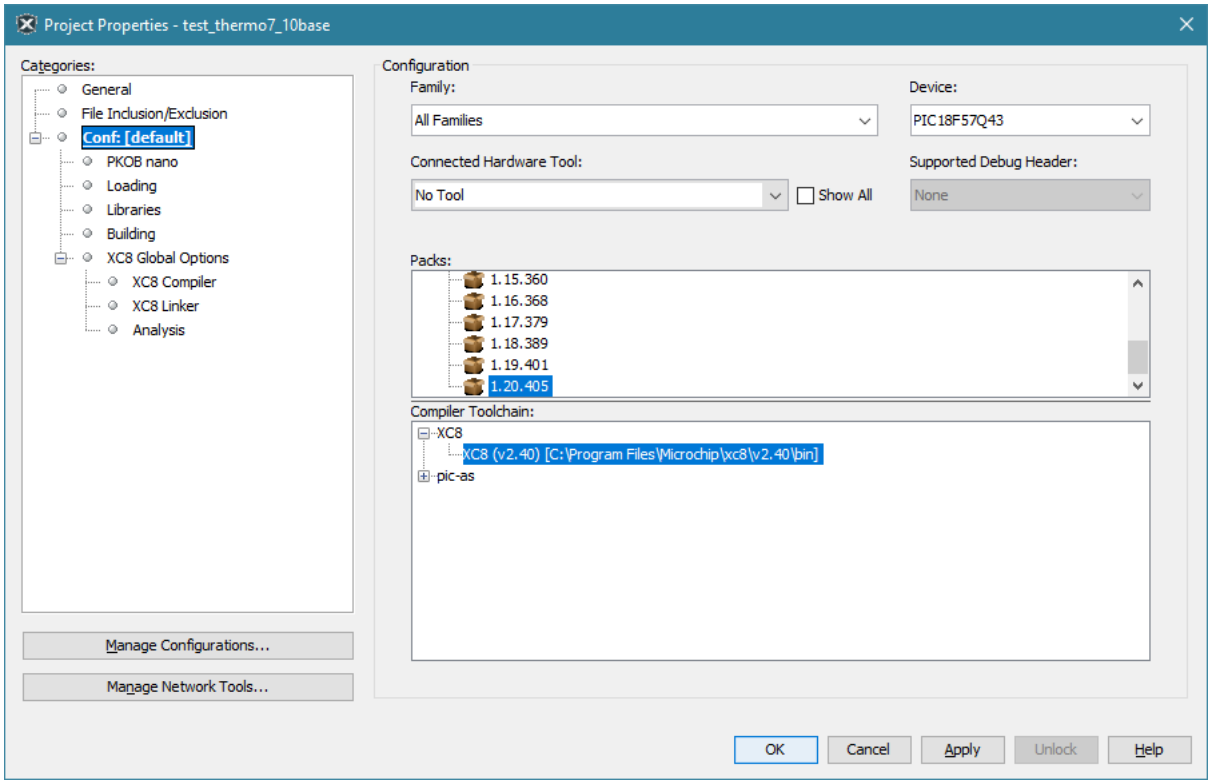
14. Project Properties setup

- Manage Configurations

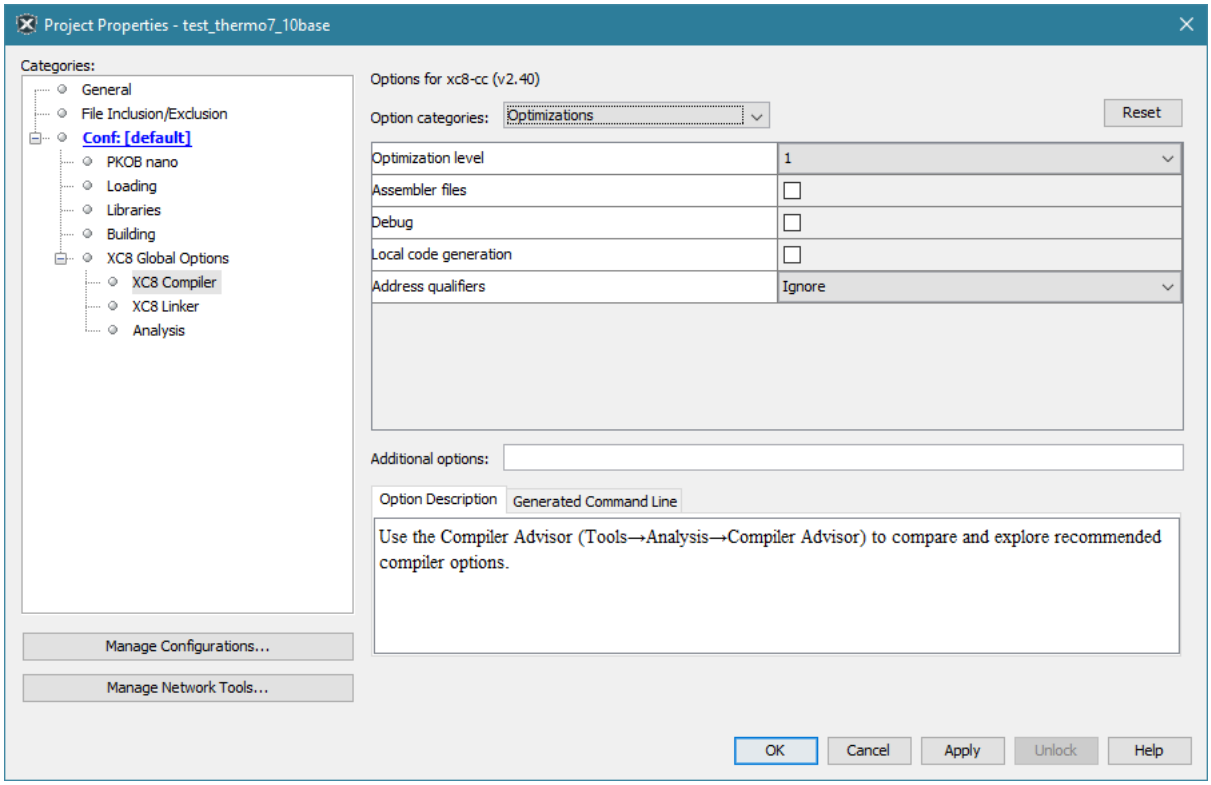




- Select the following Pack & Compiler

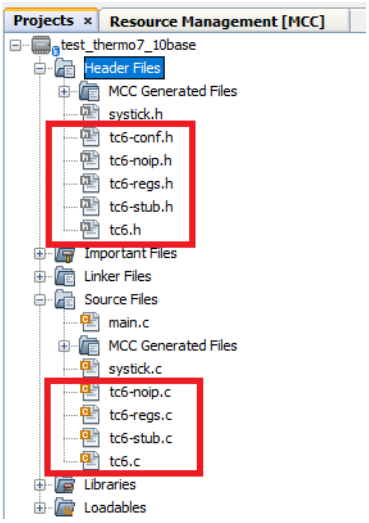


- Optimizations



15. Add TC6 Library and application Code to the Project

- **Go to [https://github.com/MicrochipTech/oa-tc6-lib and download the tc6](https://github.com/MicrochipTech/oa-tc6-lib) library into the project.**



16. Main Code Explanation

- The following is the adjustable T1S setup

```
#ifndef BOARD_INSTANCE
#define BOARD_INSTANCE              (0)
#endif
#define T1S_PLCA_ENABLE             (true)
#define T1S_PLCA_NODE_ID            (BOARD_INSTANCE)
#define T1S_PLCA_NODE_COUNT         (8)
#define T1S_PLCA_BURST_COUNT        (0)
#define T1S_PLCA_BURST_TIMER        (0x80)
#define MAC_PROMISCUOUS_MODE        (false)
#define MAC_TX_CUT_THROUGH          (false)
#define MAC_RX_CUT_THROUGH          (false)

#define DELAY_LED                   (256)
#define UDP_PAYLOAD_OFFSET          (42)
#define UDP_TARGET_PORT             (34505)
```

- MAC Address

```
97    static const uint8_t m_mac[] = { 0x00, 0x80, 0xC2, 0x00, 0x01, T1S_PLCA_NODE_ID }; // MAC = 00:80:C2:00:01: ID
```

- UDP Packet, will use index 42 & 43 to send the temp data High & Low bytes.

```
static uint8_t m_udpPacket[] = {
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x70, 0xb3, /* ......p. */
    0xd5, 0x77, 0xf0, BOARD_INSTANCE, 0x08, 0x00, 0x45, 0x00, /* .w.*..E. */
    0x00, 0x22, 0x37, 0x57, 0x00, 0x00, 0x80, 0x11, /* ."7W.... */
    0x00, 0x00, 0xc0, 0xa8, 0x00, 0xde, 0xc0, 0xa8, /* .F...... */
    0x00, 0xff, 0x86, 0xc9, 0x86, 0xc9, 0x00, 0x0e, /* ........ */
    0x00, 0x00, 0x01, 0xCD, 0xCD, 0xCD, 0xCD, 0xCD  /* f....... */
};
```

- This function reads the thermo7 temperature and stores the value in index 42 & 43 of the UDP packet as above. The thermo7 I2C slave address is 0x48. The configuration register address is 0x00 to read ambient temperature from the thermo7.

```
static void send_thermo7_temp(void)
{
    uint8_t addr = 0x00;
    uint16_t val = UINT16_MAX;
    val = i2c_read2ByteRegister(MCP9800_SLAVE_ADDR, addr);
    m_udpPacket[UDP_PAYLOAD_OFFSET + 0] = (uint8_t)(val >>8);
    m_udpPacket[UDP_PAYLOAD_OFFSET + 1] = (uint8_t)val;
}
```

- In the main, after initialisation, Timer2's interrupt handler is set, and a flag in the defaultInterruptHandler will be set for when to read the temperature so that it's not so fast.
- Memcpy, copies index six from the mac address into the UDP packet array at index six

```
void main(void)
{
    SYSTEM_Initialize();
    MCP9800_Initialize();
    TMR2_SetInterruptHandler(TMR2_DefaultInterruptHandler);
    INTERRUPT_GlobalInterruptEnable();
    SysTick_Init();

    memcpy(&m_udpPacket[6], m_mac, 6);
    m_udpPacket[0x1D] = 100 + T1S_PLCA_NODE_ID; /* Last Byte of IP address */
    m_udpPacket[0x24] = (UDP_TARGET_PORT >> 8) & 0xFF;
    m_udpPacket[0x25] = UDP_TARGET_PORT & 0xFF;
```

- In TMR2.c

```
/* Public variable */
bool time_to_read_temp = false;

void TMR2_DefaultInterruptHandler(void){
    // add your TMR2 interrupt custom code
    // or set custom function using TMR2_SetInterruptHandler()
    time_to_read_temp = true;
}
```

- In TMR2.h

```
extern bool time_to_read_temp;
```

- In the while(true) loop, the TC6 library is getting services and the if statement does cyclic read the temperature (via the `send_thermo7_temp()` function) and the UDP packet is sent with `TC6NoIP_SendEthernetPacket()` function.

```
while (true)
{
    uint32_t now = SysTick_GetMillis();
    TC6NoIP_Service();

    if(time_to_read_temp) {  // read the temperature every second

       send_thermo7_temp();  //read temperature and store the data into UDP buffer
       TC6NoIP_SendEthernetPacket(0, m_udpPacket, sizeof(m_udpPacket));

       time_to_read_temp = false; // reset flag
    }
```

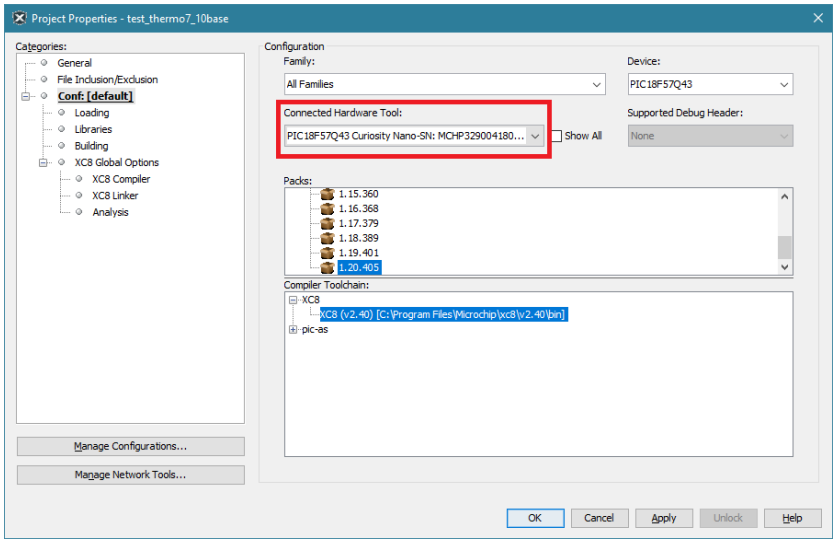
- On the other board, the Ethernet Receive function takes the High and Low bytes of the temperature value and converts to a float and prints the value.

```
void TC6NoIP_CB_OnEthernetReceive(int8_t idx, const uint8_t *pRx, uint16_t len)
{
    if ((0x11 /* UDP Protocol */ == pRx[0x17]) && (len >= (UDP_PAYLOAD_OFFSET + 2))) {
        bool error = false;
        uint16_t targetPort = ((uint16_t)pRx[0x24] << 8) | (uint16_t)pRx[0x25];
        if (UDP_TARGET_PORT == targetPort) {

            uint32_t now = SysTick_GetMillis();
            uint8_t remoteNode = pRx[0xB];
            uint8_t dataH = pRx[UDP_PAYLOAD_OFFSET + 0];
            uint8_t dataL = pRx[UDP_PAYLOAD_OFFSET + 1];
            uint16_t data = ((uint16_t)dataH << 8) | (dataL);
            if ((0 == data) || (UINT16_MAX == data)) {
                error = true;
            } else {
                float temp = (float)data;
                temp /= CONVERSION_FACTOR;

                if((temp < MCP9800_MAX_NEG_VALUE) || (temp > MCP9800_MAX_POS_VALUE)) {
                    error = true;
                } else {
                    printf("[%ld]Temperature from Node %d: %.3f C \r\n", now, remoteNode, temp);
                }
            }
            if (error) {
                printf("[%ld]Temperature from Node %d invalid. Is Thermo7 click missing?\r\n", now, remoteNode);
            }
        }
    }
}
```

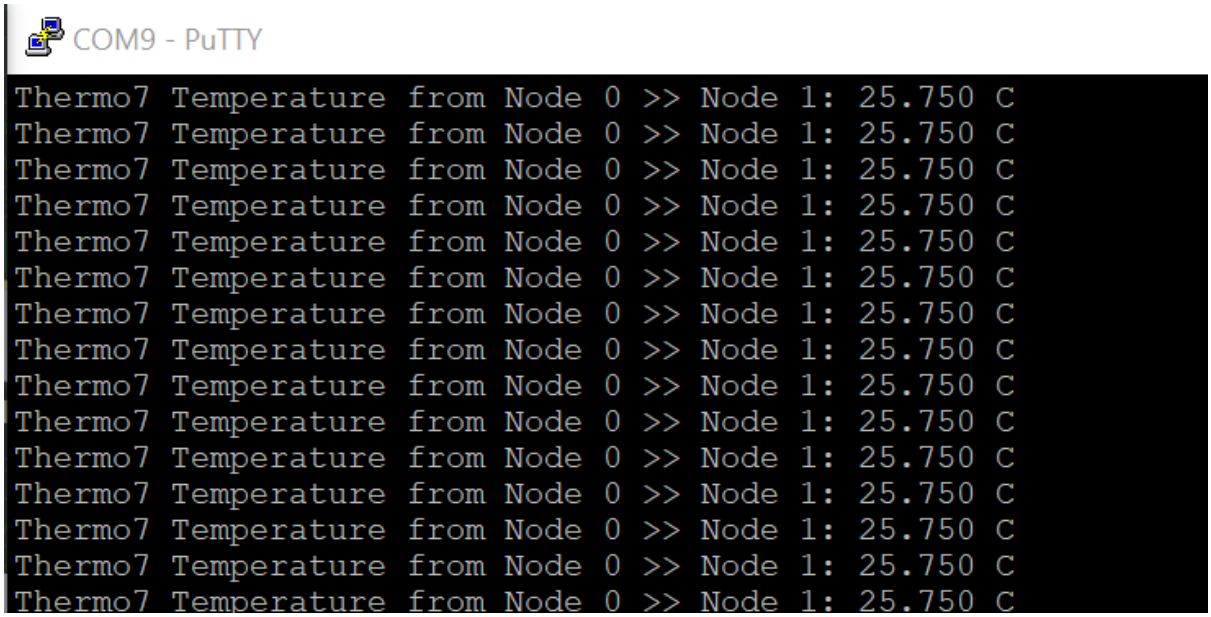
- Both boards can be programmed using the same code, however the BOARD_INSTANCE must be 0 for one board and 1 for the other, with a maximum of 8 nodes.
- The receive function also checks if the thermo7 click is disconnected by checking the temperature. Failure case is data equals to 0 or UINT16_MAX.

17. Chose the right board in the project properties:

- If both boards are connected, the correct programmer can be chosen in the project properties:



18. Output


COM9 - PuTTY
Thermo7 Temperature from Node 0 >> Node 1: 25.750 C
Thermo7 Temperature from Node 0 >> Node 1: 25.750 C
Thermo7 Temperature from Node 0 >> Node 1: 25.750 C
Thermo7 Temperature from Node 0 >> Node 1: 25.750 C
Thermo7 Temperature from Node 0 >> Node 1: 25.750 C
Thermo7 Temperature from Node 0 >> Node 1: 25.750 C
Thermo7 Temperature from Node 0 >> Node 1: 25.750 C
Thermo7 Temperature from Node 0 >> Node 1: 25.750 C
Thermo7 Temperature from Node 0 >> Node 1: 25.750 C
Thermo7 Temperature from Node 0 >> Node 1: 25.750 C
Thermo7 Temperature from Node 0 >> Node 1: 25.750 C
Thermo7 Temperature from Node 0 >> Node 1: 25.750 C
Thermo7 Temperature from Node 0 >> Node 1: 25.750 C
Thermo7 Temperature from Node 0 >> Node 1: 25.750 C