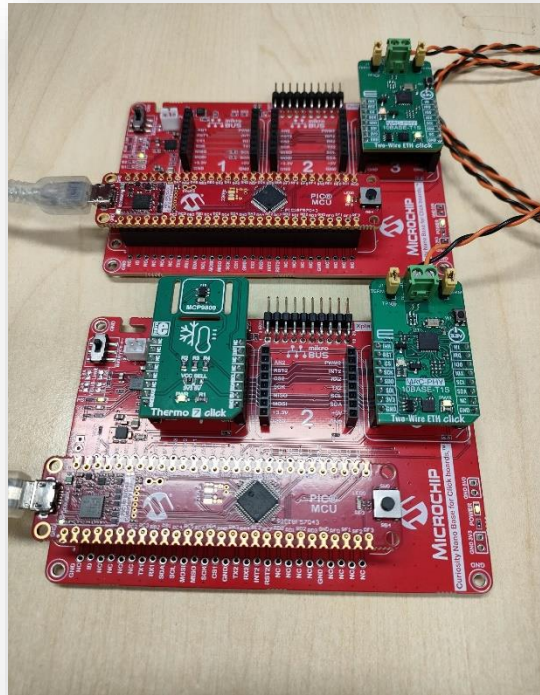




8-bit 10Base-T1S Project Example

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

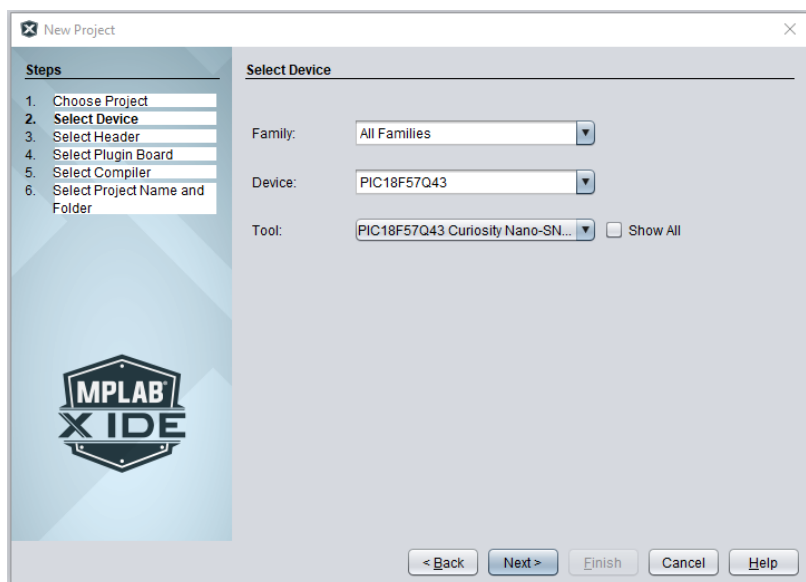
Software

- **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

Easy Setup

Registers

INTERNAL OSCILLATOR

Current System clock 64 MHz

Oscillator Select

HFINTOSC

External Clock Select

Oscillator not enabled

HF Internal Clock

64_MHz

→PLL Capable Frequency

External Clock

1 MHz

Clock Divider

1

WWDTClock

Watchdog Timer Enable

WDT Disabled; SWDTEN is ignored

Clock Source

Software Control

Window Open Time

window always open (100%); software control; keyed access not required

Time-out Period

Divider ratio 1:65536; software control of WDTPS

Programming

☒ Low-voltage Programming Enable

3. Add EXT_INT to Project

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

Project Resources

Generate

Import...

Export

?

?

Libraries

Foundation Services

Peripherals

EXT_INT

I2C1

SPI1

TMR0

TMR2

UART1

System

DMA Manager

Interrupt Module

Pin Module

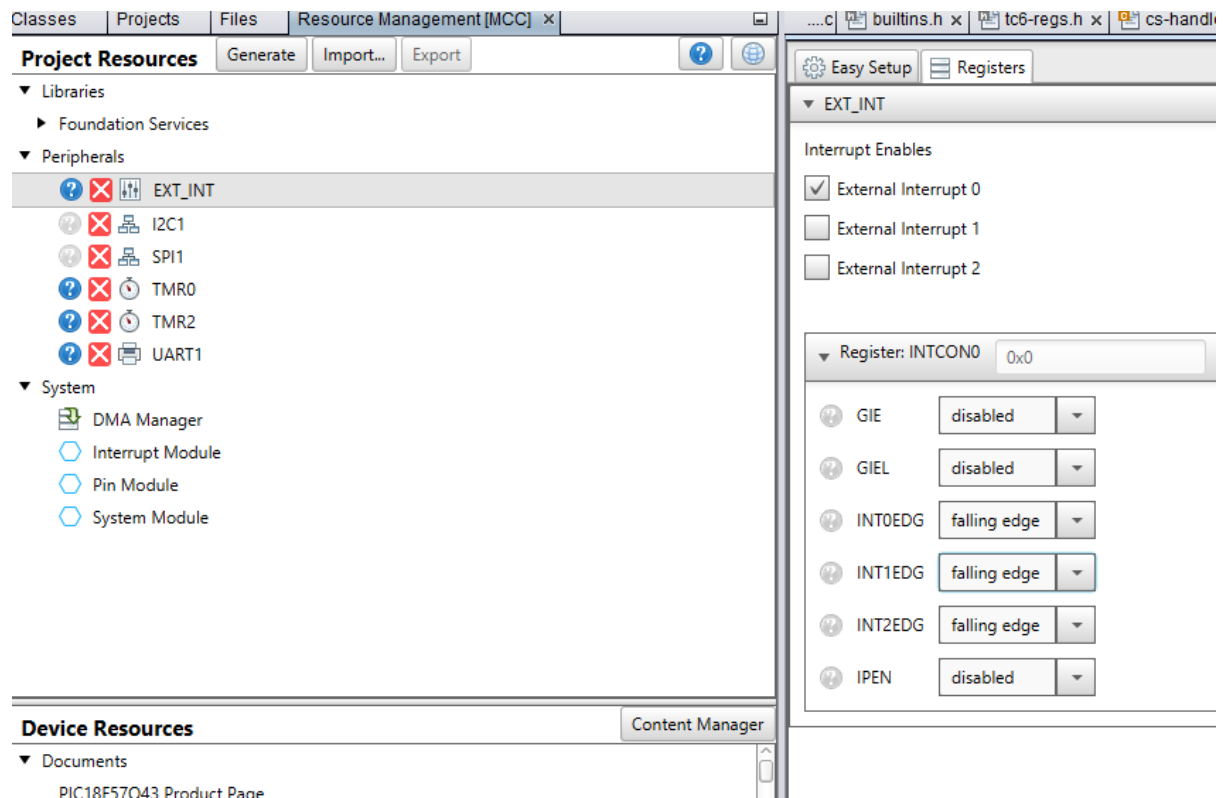
System Module

Easy Setup

Registers

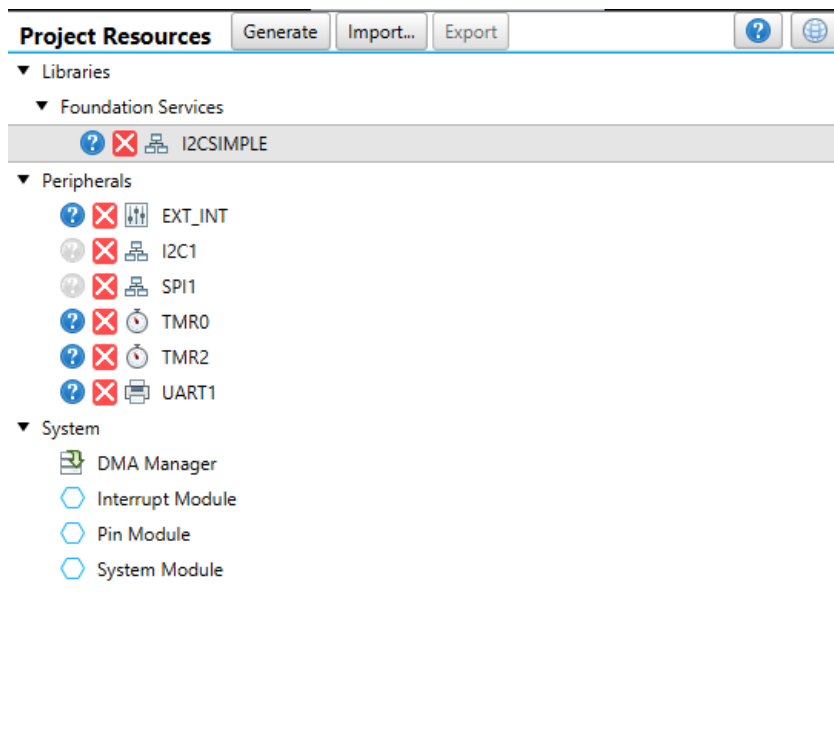
Hardware Settings

External Interrupt	Edge Detect
INT0	falling edge



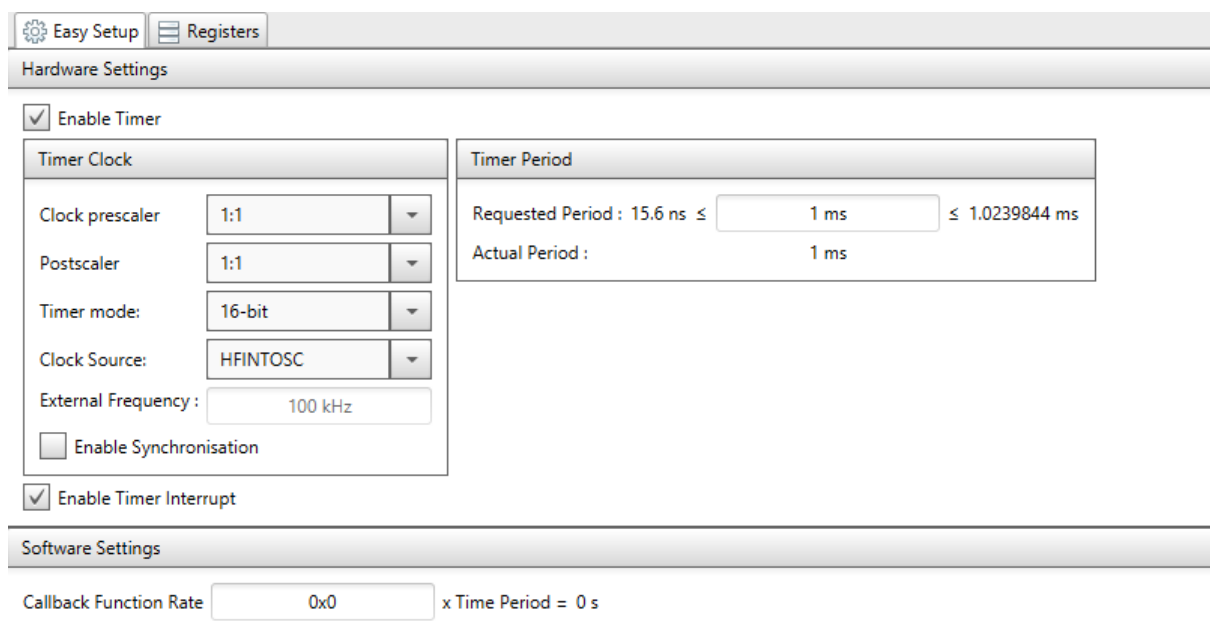
4. Add to project.

- SPI1
- TMR0
- TMR2
- UART1
- I2C1 & I2CSIMPLE



5. Setup Timer0

- Choose settings as image below



- Configure TMR0 registers:
 - T0CON0 = 0x90
 - T0CON1 = 0x70
 - TMR0H = 0x05
 - TMR0L = 0x00

The screenshot shows the 'Easy Setup' window with the 'Registers' tab selected. It displays the configuration for four TMR0 registers: T0CON0, T0CON1, TMR0H, and TMR0L.

Register	Value	Field	Value
Register: T0CON0	0x90	T016BIT	16-bit
		T0EN	enabled
		T0OUT	low
		T0OUTPS	1:1
Register: T0CON1	0x70	T0ASYNC	not_synchronised
		T0CKPS	1:1
		T0CS	HFINTOSC
Register: TMR0H	0x5	TMR0H	0x5
Register: TMR0L	0x0	TMR0L	0x0

6. Setup Timer 2

- The Interrupt for the temperature checking can be adjusted for the time using the callback function rate.

Easy Setup

Registers

Hardware Settings

☒ Enable Timer

Control Mode

Roll over pulse

Ext Reset Source

T2CKIPPS pin

Start/Reset Option

Software control

Timer Clock

Clock Source

FOSC/4

☐ Enable Clock Sync

Clock Frequency

32.768 kHz

Polarity

Rising Edge

Prescaler

1:64

☐ Enable Prescaler O/P Sync

Postscaler

1:16

Timer Period

Timer Period

64 us

10 ms

16.384 ms

Actual Period

9.984 ms

(Period calculated via Timer Period)

Software Settings

☒ Enable Timer Interrupt

Callback Function Rate

0x65

x Time Period = 1.008384 s

7. Setup UART

- Baud Rate 115200
- Enable Transmit and Receive
- Do not enable UART interrupts
- Redirect STDIO

Easy Setup

Registers

Hardware Settings

Mode

Asynchronous 8-bit mode

☒ Enable UART

Baud Rate:

115200

Error: -0.080 %

☒ Enable Transmit

Transmit Polarity:

not inverted

☒ Enable Receive

Receive Polarity:

not inverted

☐ Auto-Baud Detection

Receiver Address:

0x0

☐ Enable UART Interrupts

Software Settings

☒ Redirect STDIO to UART

Software Transmit Buffer Size

8

Software Receive Buffer Size

8

8. Setup SPI as below

Easy Setup	
Registers	
▼ Software Settings	
Interrupt Driven:	<input type="checkbox"/>
▼ Hardware Settings	
Mode:	Master
SPI Mode:	SPI Mode 0
Input Data Sampled At:	Middle
Clock Source Selection:	FOSC
Clock Divider:	0 ≤ 4 ≤ 255
Actual Clock Frequency(Hz):	6400000.000
▼ Interrupt Settings	
Enable SPI Interrupt:	<input type="checkbox"/>

9. Setup i2c module

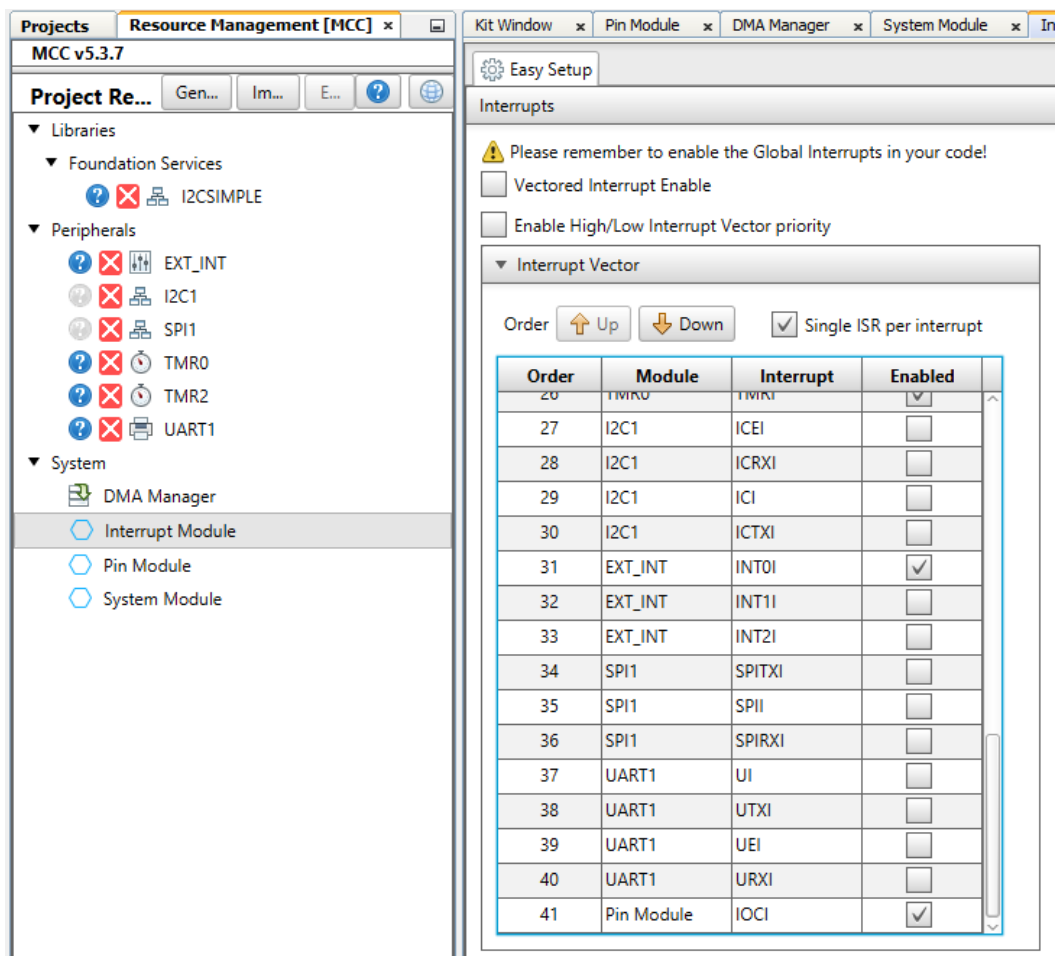
- Leave as default

ect Resources	Generate	Import...	Export	?	!
Foundation Services					
I2CSIMPLE					
Peripherals					
EXT_INT					
I2C1					
SPI1					
TMR0					
TMR2					
UART1					

Easy Setup	
Hardware Settings	
Select I2C Master	I2C1
▼ Use Case Examples	
Select use case example to generate	None

10. Interrupt Module

- Enable INT0I
- Enable TMRI
- Enable TMR2
- Enable IOCI



11. Pin Module

- Setup as below

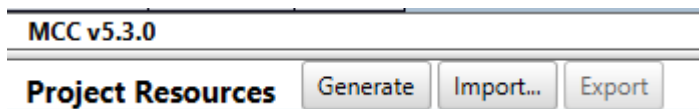
<div> <div>Easy Setup</div> <div>Registers</div> </div> <div>Selected Package : TQFP48</div>										
Pin Name	Module	Function	Custom Name	Start High	Analog	Output	WPU	OD	IOC	
RA6	Pin Module	GPIO	IO_INT1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	none	
RB1	I2C1	SCL1		<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	none	
RB2	I2C1	SDA1		<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	none	
RB4	Pin Module	GPIO	IO_SW0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	none	
RB5	EXT_INT	INT0		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	any	
RC4	SPI1	SDO1		<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	none	
RC5	SPI1	SDI1		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	none	
RC6	SPI1	SCK1		<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	none	
RD4	Pin Module	GPIO	IO_CS1	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
RD5	Pin Module	GPIO	IO_RST3	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
RD6	Pin Module	GPIO	IO_CS2	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
RD7	Pin Module	GPIO	IO_CS3	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
RF0	UART1	TX1		<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
RF1	UART1	RX1		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
RF3	Pin Module	GPIO	IO_LED0	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

12. Pin Manager

- Setup as below

Pin Manager: Grid View																																																
Package:	TQFP48		Pin No:		21	22	23	24	25	26	33	32	8	9	10	11	16	17	18	19	34	35	40	41	46	47	48	1	42	43	44	45	2	3	4	5	27	28	29	20	36	37	38	39	12	13	14	15
					Port A ▼							Port B ▼							Port C ▼							Port D ▼							Port E ▼			Port F ▼												
Module	Function	Direction	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	0	1	2	3	4	5	6	7		
EXT_INT ▼	INT0	input																																														
	INT1	input																																														
	INT2	input																																														
I2C1 ▼	SCL1	in/out																																														
	SDA1	in/out																																														
OSC	CLKOUT	output																																														
Pin Module ▼	GPIO	input																																														
	GPIO	output																																														
RESET	MCLR	input																																														
SPI1 ▼	SCK1	in/out																																														
	SDI1	input																																														
	SDO1	output																																														
TMR0 ▼	TOCKI	input																																														
	TMR0	output																																														
TMR2	T2IN	input																																														
UART1 ▼	CTS1	input																																														
	RTS1	output																																														
	RX1	input																																														
	TX1	output																																														
	TXDE1	output																																														

13. Generate the Code in MCC



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

```

1  /*...22 lines */
23
24  /*...22 lines */
46
47  #ifndef SPI1_MASTER_H
48  #define SPI1_MASTER_H
49
50  /*...3 lines */
53
54  #include <stdio.h>
55  #include <stdint.h>
56  #include <stdbool.h>
57
58  /* SPI interfaces */
59  typedef enum {
60      SPI1_DEFAULT
61  } spi1_modes_t;
62
63  void SPI1_Initialize(void);
64  bool SPI1_Open(spi1_modes_t spi1UniqueConfiguration);
65  void SPI1_Close(void);
66  uint8_t SPI1_ExchangeByte(uint8_t data);
67  void SPI1_ExchangeBlock(void *block, size_t blockSize);
68  void SPI1_WriteBlock(void *block, size_t blockSize);
69  void SPI1_ReadBlock(void *block, size_t blockSize);
70  void SPI1_WriteByte(uint8_t byte);
71  uint8_t SPI1_ReadByte(void);
72  void SPI1_ExchangeBlocks(const uint8_t *pTx, uint8_t *pRx, size_t blockSize);
73
74  #endif //SPI1_H

```

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

```
spi1.c x
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 }
```

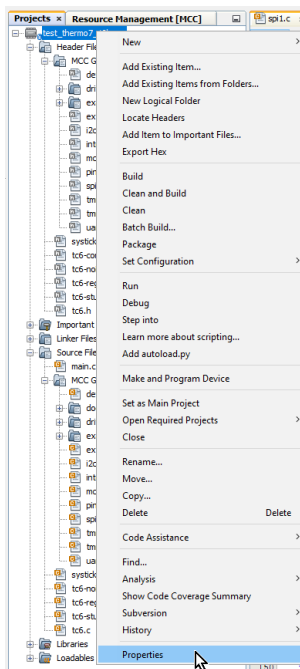
```

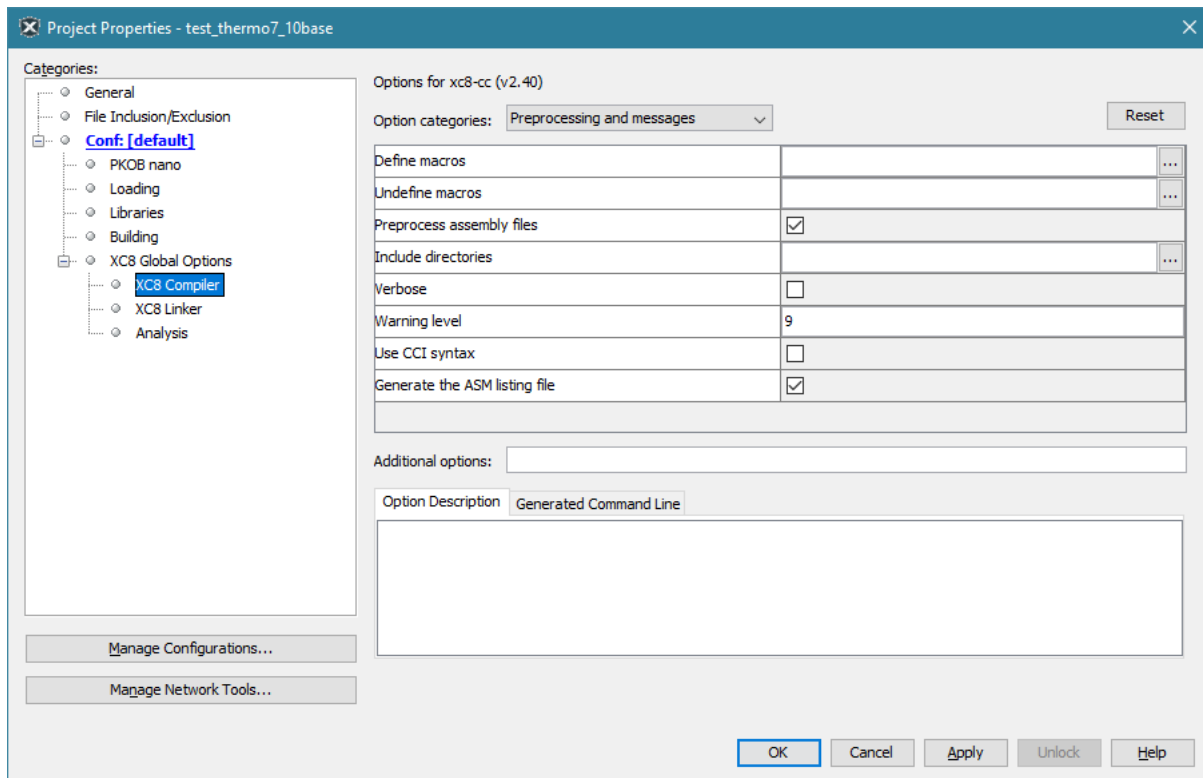
void SPI1_ExchangeBlocks(const uint8_t *pTx, uint8_t *pRx, size_t blockSize)
{
    size_t i;
    if (pTx && pRx) {
        for (i = 0; i < blockSize; i++)
        {
            SPI1TCNTL = 1;
            SPI1TXB = pTx[i];
            while(!PIR3bits.SPI1RXIF);
            pRx[i] = SPI1RXB;
        }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

```

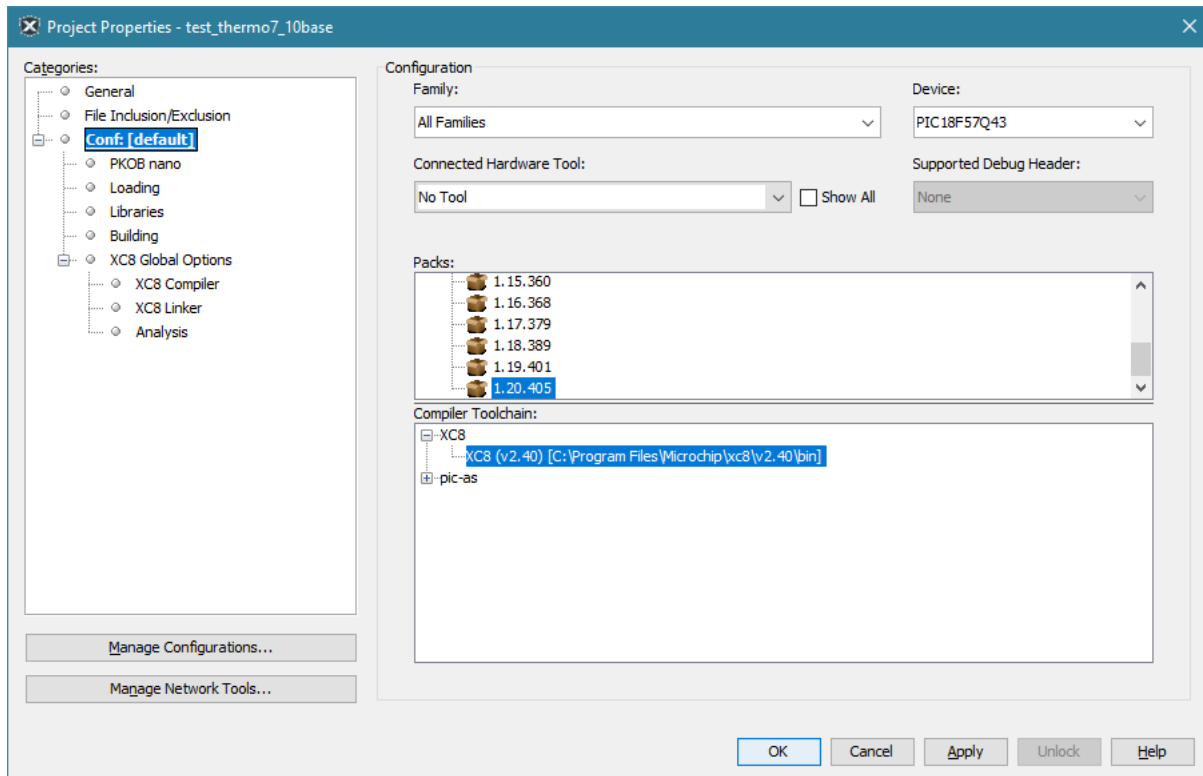
14. Project Properties setup

- Manage Configurations

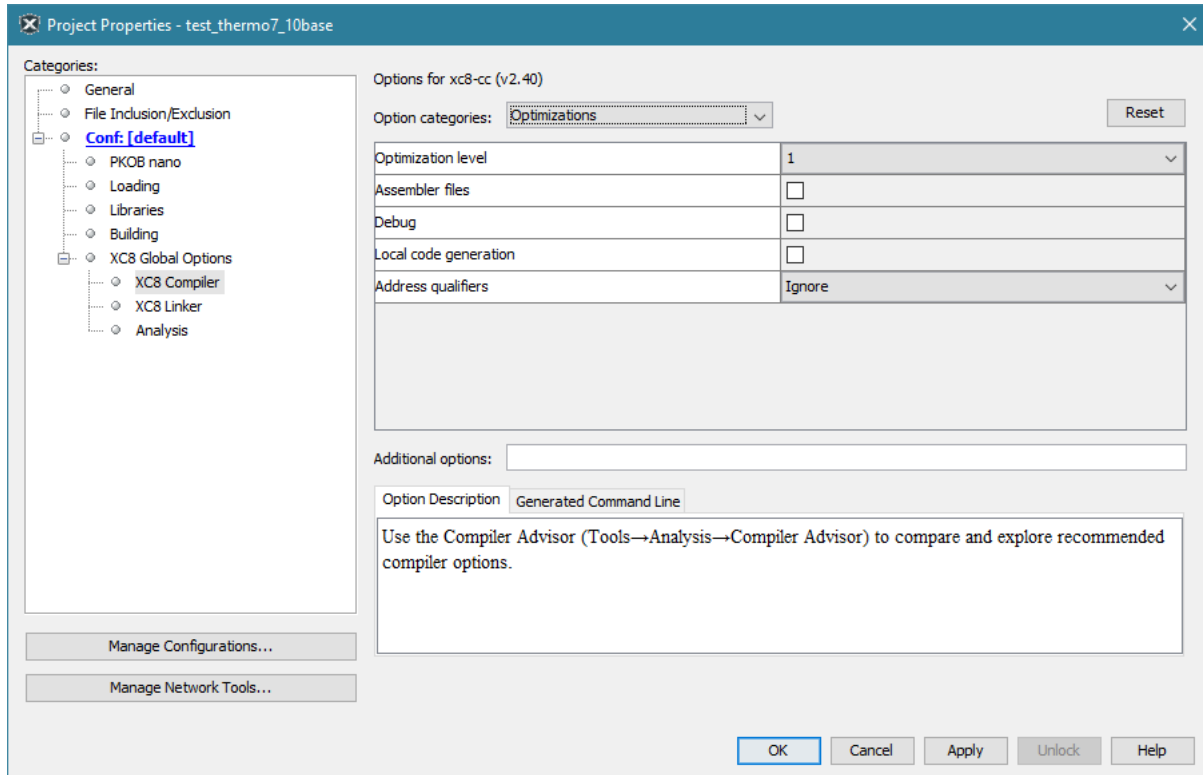




- Select the following Pack & Compiler

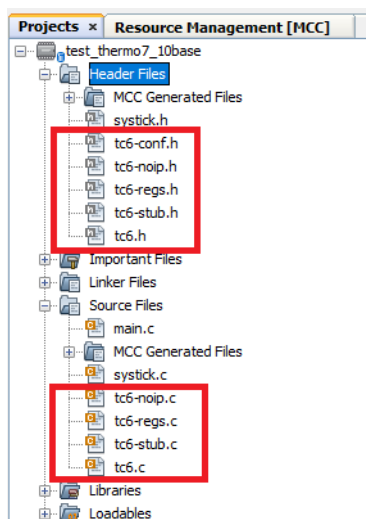


- Optimizations



15. Add TC6 Library and application Code to the Project

- Go to <https://github.com/MicrochipTech/oa-tc6-lib> and download the tc6 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) {

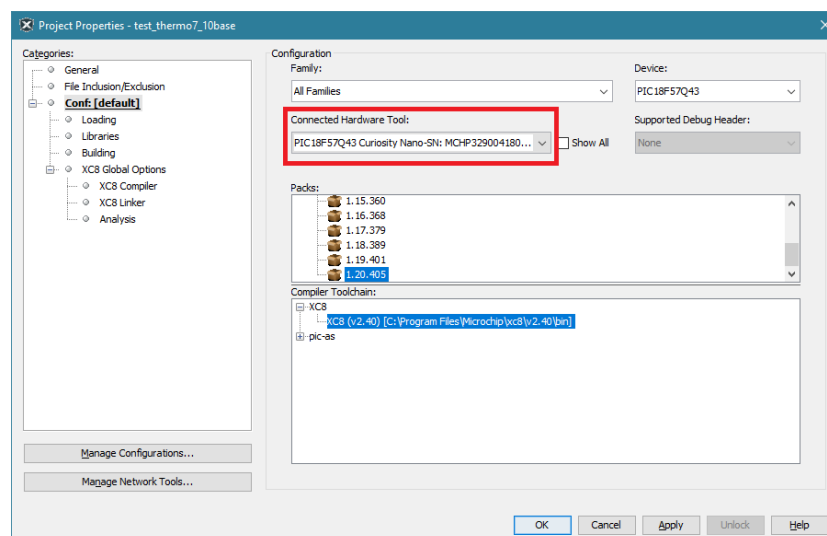
            uint32_t now = SysTick_GetMillis();
            uint8_t remoteNode = pRx[0xB];
            uint8_t dataH = pRx[UDP_PAYLOAD_OFFSET + 0];
            uint8_t dataL = pRx[UDP_PAYLOAD_OFFSET + 1];
            uint16_t data = ((uint16_t)dataH << 8) | (dataL);
            if ((0 == data) || (UINT16_MAX == 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[illegible]