Creating the First Application on PIC32CX SG Microcontrollers Using MPLAB Harmony v3 with MPLAB Code Configurator (MCC)

TB3345

Introduction

MPLAB® Harmony v3 is a software development framework consisting of compatible and interoperable modules that include peripheral libraries (PLIBs), drivers, system services, middleware, and third-party libraries. The MPLAB Code Configurator (MCC) is a GUI-based tool that provides an easy way to enable and configure various MPLAB Harmony modules. The MCC is a plug-in to the MPLAB X Integrated Development Environment (IDE).

This document describes how to create a simple application on a Arm® Cortex-M4F based PIC32CX SG Microcontroller using the MCC with MPLAB Harmony v3 modules. The objective of this Application is to toggle an LED on a timeout basis and print the LED toggling rate and temperature reading periodically on a serial console. For this demonstration, the following MPLAB Harmony v3 modules are used and configured using the MCC:

- The PORT pin to toggle the LED.
- Real-Time Clock (RTC) PLIB to periodically sample the LED toggling rate and temperature reading.
- Two instances of the External Interrupt Controller (EIC) PLIB: one to change the toggling rate when there is a switch press event and another one to decide what needs to be printed on the Serial Console; temperature reading or LED toggling rate.
- SERCOM (configured as I^2C) PLIB to read the temperature from an on-board temperature sensor.
- SERCOM (configured as USART) and DMA PLIBs to print the LED toggling rate and temperature reading on a COM (serial console) port terminal application running on a PC.

1. Creating First Application on the PIC32CX SG61 MCU

The following software and hardware tools are used for this demonstration:

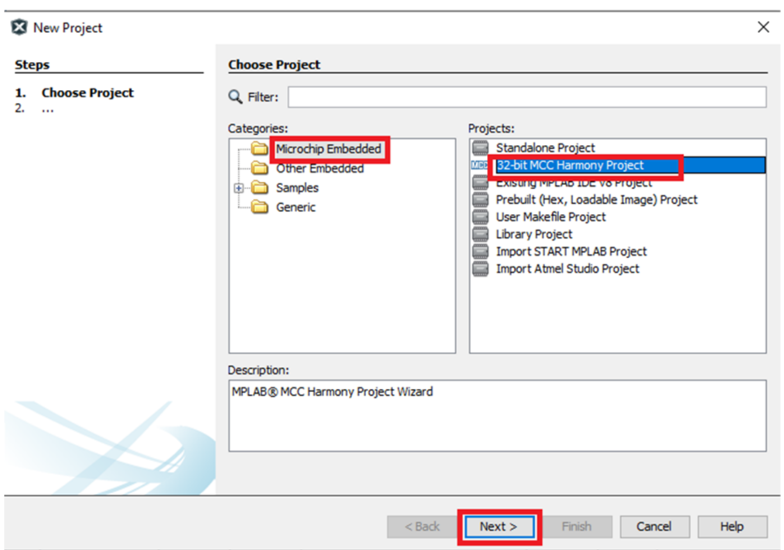
- MPLAB X IDE v6.15
- MPLAB Code Configurator (MCC) Plug-in v5.3.7
- MPLAB XC32 Compiler v4.30
- MPLAB Harmony v3 repositories:
  - csp v3.18.0
  - dev_packs v3.18.0
- PIC32CX SG61 Curiosity Ultra Evaluation Board

Note: The updated versions of the above listed tools can also be used to create the application, and users are not restricted to the usage of the older versions.

To create an MPLAB Harmony v3-based project, follow these steps:

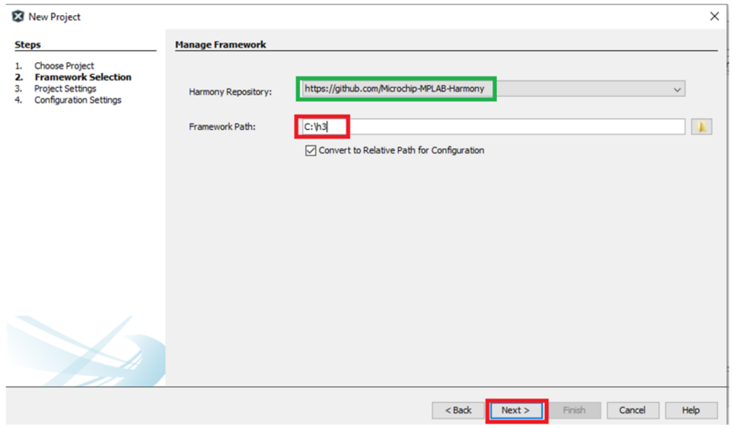
1. From the Start Menu launch MPLAB X IDE.
2. Once MPLAB X IDE is open, on the File Menu click **New Project** or click on the new project icon.
3. In the New Project window, under Steps navigation pane, select **Choose Project**.
4. In the right Choose Project properties page, under Categories select **Microchip Embedded**, and under Projects select **32-bit MCC Harmony Project**.

Figure 1-1. Choose Project



5. Click **Next**.
6. In the left navigation pane, select **Framework Selection** and in the right Manage Framework properites page, enter these details:
   a. Harmony Repository: Enter the path *https://github.com/Microchip-MPLAB-Harmony*.
   b. Framework Path: Enter `C:\h3` (path to the folder in which the MPLAB Harmony v3 packages are downloaded).

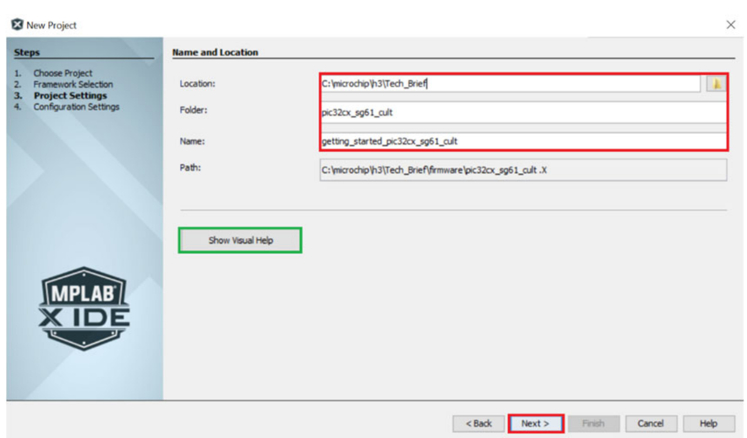
Figure 1-2. Framework Selection



Note: For this demonstration application, the following MPLAB Harmony v3 packages are required: `dev_packs` and `csp`. The MCC Content Manager simplifies the downloading of the MPLAB Harmony v3 packages. If these packages are not downloaded, refer to the MPLAB® Code Configurator Content Manager for MPLAB Harmony v3 Projects video to download it.

7. Click **Next.**
8. In the left navigation pane, select **Project Settings** and in the Name and Locations properties page, enter these details:
   Location: Enter *C:\microchip\h3\tech_brief* (Indicates the path to the root folder of the new project. All project files will be placed in this folder. The project location can be any valid path).
   Folder: Enter `pic32cx_sg61_cult` (Indicates the name of the MPLAB X IDE .X folder to create a `pic32cx_sg61_cult.X` folder).
   Name: Enter `getting_started_pic32cx_sg61_cult` (Indicates the name of the project that will be shown in MPLAB X IDE to set the project's name).
   Path: Read-only content (Automatically updates when users make changes to the above entries).
   Note: This project can also be created for the PIC32CX SG41 Curiosity Ultra Evaluation Board by following the similar steps to create and configure the project.

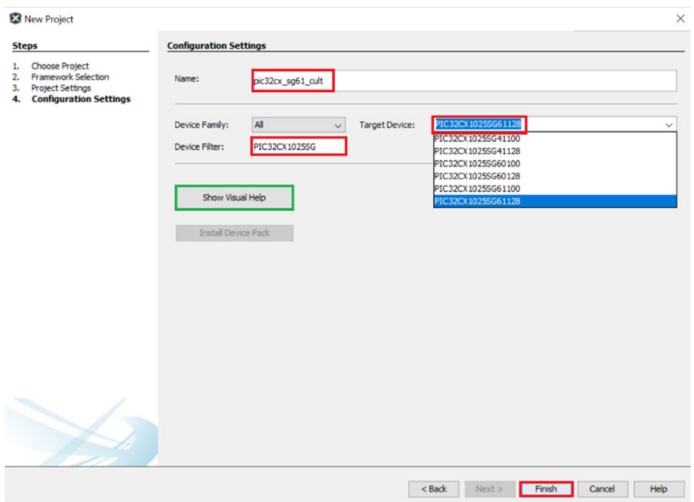
Figure 1-3. Project Settings



Note: Click on the **Show Visual Help** button to open a contextual help window for a detailed description of the various fields in the Project Settings.

9. Click **Next**.
10. In the left navigation pane, select **Configuration Settings** and in the right Configuration Settings properties page, enter these details:
    - **Name:** Enter `pic32cx_sg61_cult`.
    - **Device Family:** All.
    - **Device Filter:** Enter PIC32CX1025SG.
    - **Target Device:** In the drop-down item list select PIC32CX1025SG61128 for creating the project on the PIC32CX SG61 Curiosity Ultra Evaluation Board (The Device Filter entry will be reflected under the Target Device).

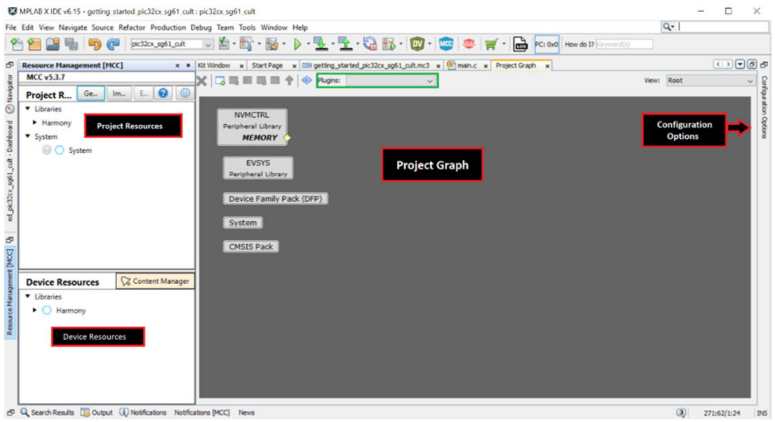
Figure 1-4. Configuration Settings



Note: Click on the **Show Visual Help** button to open a contextual help window for a detailed description of various fields in the Configuration Settings.

11. Click **Finish** to launch the MCC.
12. Before launching the MCC, the Configuration Database Setup window will be displayed, where the Device Family Pack (DFP) and Cortex® Microcontroller Software Interface Standard (CMSIS) path can be changed if required. For this demonstration, the default settings are used.
13. The MCC plug-in will open in a new window as shown in the following figure:

Figure 1-5. MPLAB Code Configurator Window

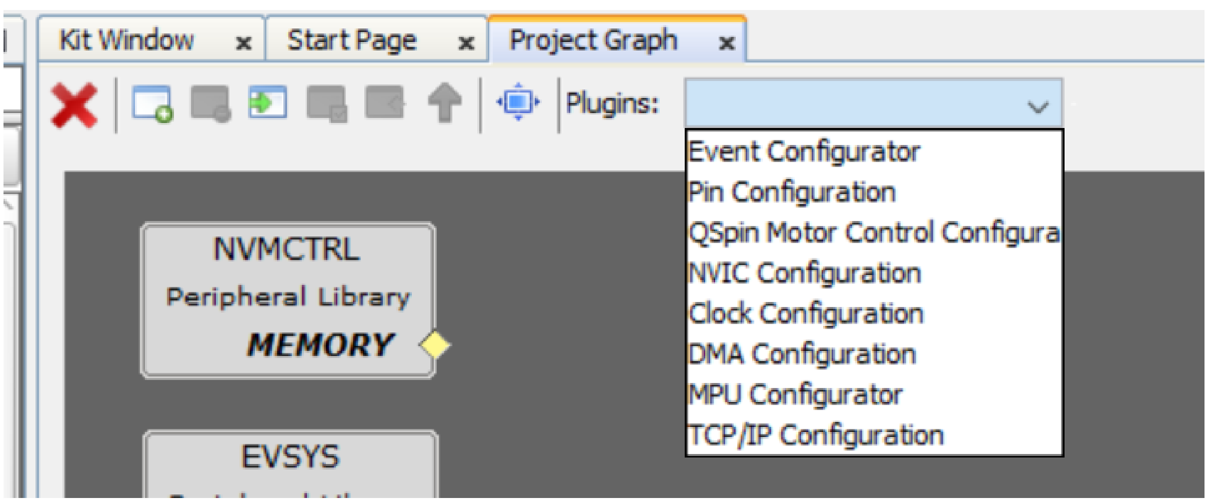


1.1 Adding and Configuring MPLAB Harmony Components

To add and configure the MPLAB Harmony components using the MCC, follow these steps:

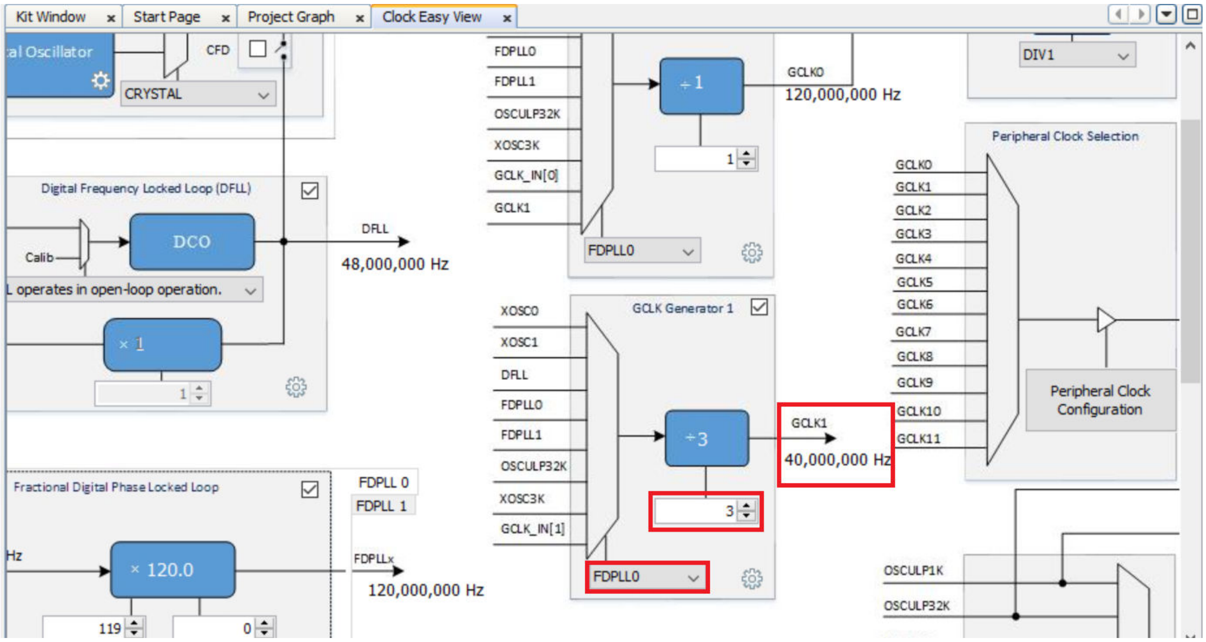
1. In the MCC window, from the Plugins drop-down list, select the required Configuration Window.

Figure 1-6. MPLAB Code Configurator - Plugins



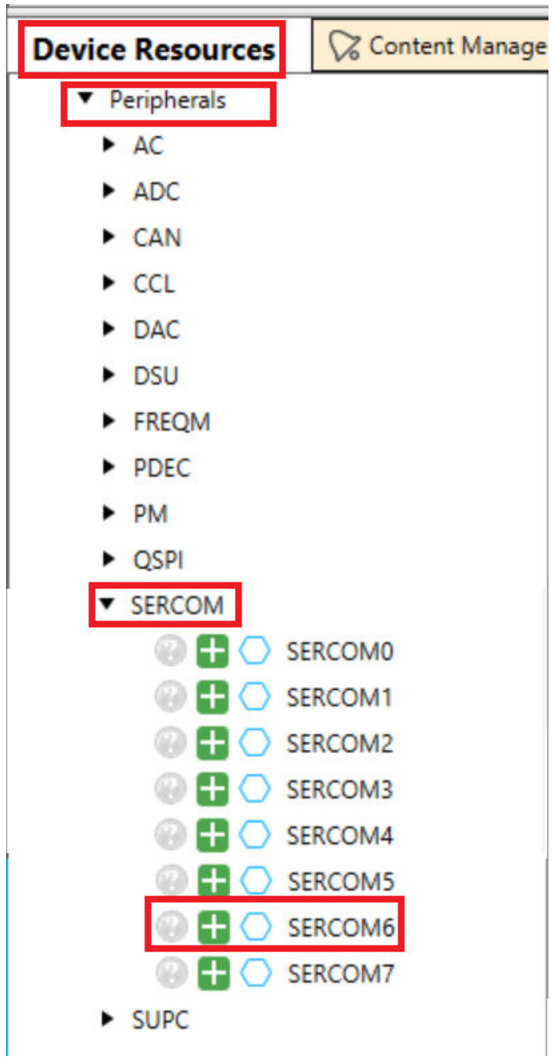
2. Select Clock Configuration in the drop-down list to open the Clock Easy View window, and verify that the Main Clock is set to 120 MHz.
Note: Make sure to make the following modification for GCLK Generator 1.

Figure 1-7. MPLAB Code Configurator - GCLK Generator 1



3. Under Device Resources, select *Peripherals* > *SERCOM* > *SERCOM6* and observe that the SERCOM6 Peripheral Library block is added in the Project Graph Window.

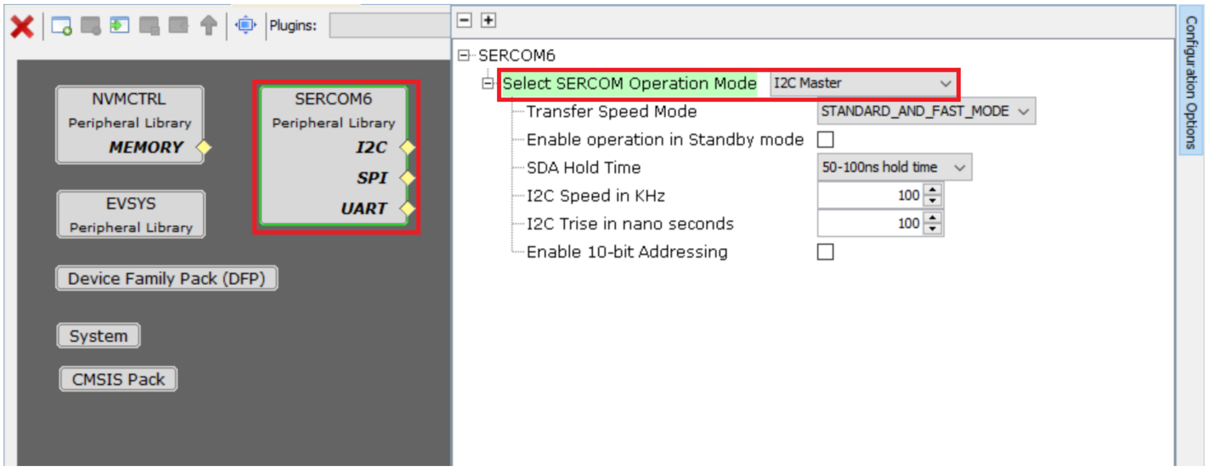
Figure 1-8. MPLAB Code Configurator - Selection of Peripherals



Note: Similarly all peripherals can be selected under *Device Resources* > *Peripherals*.

4. In the left pane, select the SERCOM6 Peripheral Library in the Project Graph. In the Configuration Options right pane, configure it as follows to read the temperature from the on-board temperature sensor of the evaluation board.

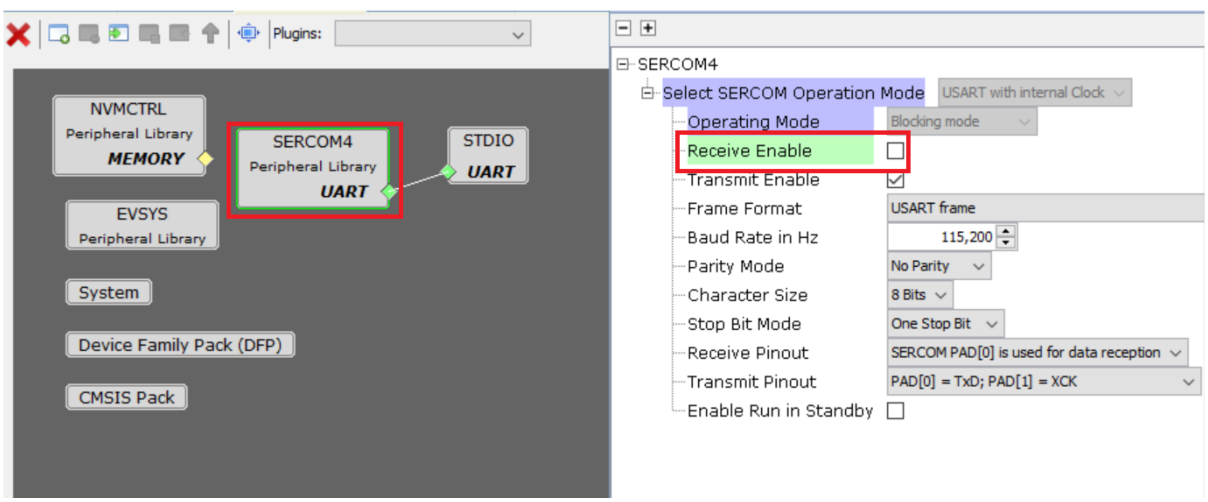
Figure 1-9. MPLAB Code Configurator - SERCOM6 Configuration



Note: The SCL clock is not configured for 100 kHz, tune the GCLK to achieve 100 kHz.

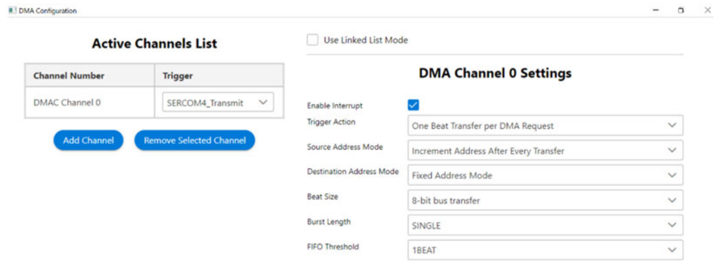
5. Under Device Resources, select *Peripherals* > *SERCOM* > *SERCOM4* and observe that the *SERCOM4* Peripheral Library block is added in the Project Graph Window.
6. In the left pane Project Graph, select **SERCOM4 Peripheral Library** and right-click on the **UART**, and then under consumers select **STDIO**. This establishes a connection between the STDIO and SERCOM4 as a UART. In the Configuration Options right pane, configure it as shown in the figure below to print the data on a Serial Console at 115200 baud rate. Clear the Receive Enable check box or change the Receive Pinout to PAD1.
Note: PAD0 is configured for TX and RX.

Figure 1-10. MPLAB Code Configurator - SERCOM4 Configuration



7. From the Plugins drop-down list select **DMA Configuration**, and configure **DMA Channel 0** to transmit the application buffer to the USART TX register. The DMA transfers one byte from the user buffer to the USART transmit buffer on each trigger.

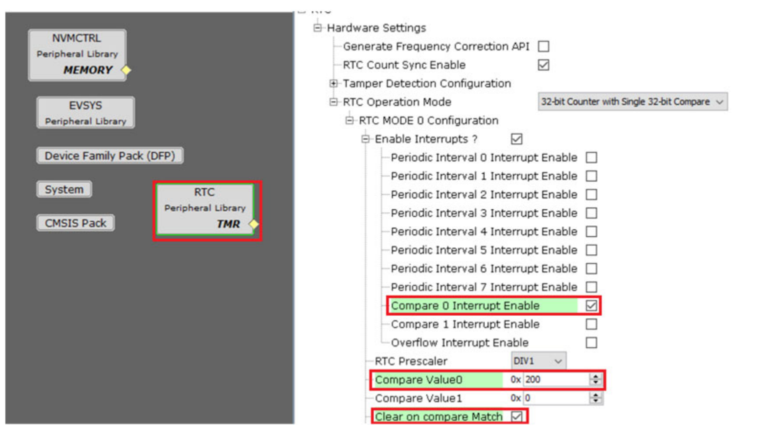
Figure 1-11. MPLAB Code Configurator - DMA Configuration



Note: Both the SERCOM4 as USART and the DMA Peripheral Libraries obtain the LED toggling rate and temperature reading from the application and print the data on a serial console running on a PC.

8. Under Device Resources, select *Peripherals* > *RTC* > *RTC* and observe that the RTC Peripheral Library block is added in the Project Graph Window to generate a compare interrupt every 500 milliseconds.

Figure 1-12. MPLAB Code Configurator - RTC PLIB Configuration



Note: The Compare Value is set as 0x200. This compare value generates an RTC compare interrupt every 500 milliseconds.

RTC clock = 1024 Hz

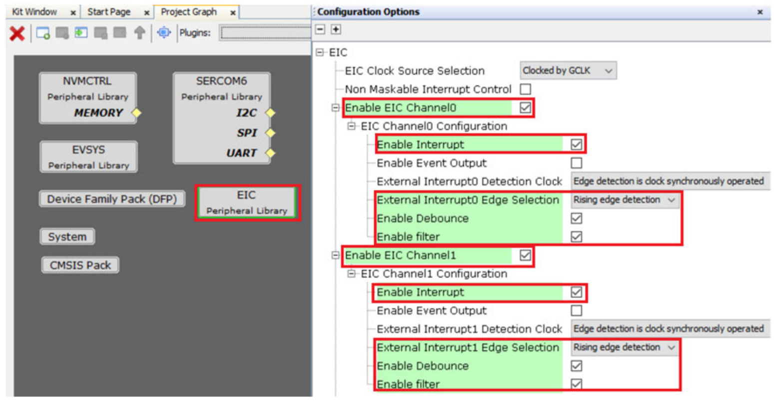
RTC Prescaler = 1

Required Interrupt rate = 500 ms

Therefore, Compare Value = (500/1000) x 1024 = 512 (i.e., 0x200).

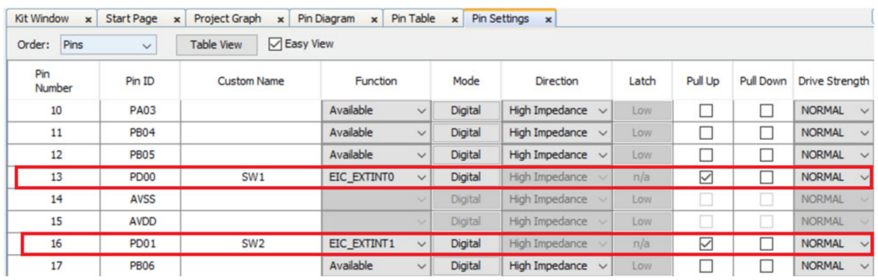
9. Under Device Resources, select *Peripherals* > *EIC* > *EIC* and observe that the EIC Peripheral Library block is added in the Project Graph Window, and enable interrupts for switch press events.

Figure 1-13. MPLAB Code Configurator - EIC PLIB Configuration



10. Open the Pin Configuration Window from the Plugins drop-down list and configure required pins as follows:

Figure 1-14. Pin Settings Window - EIC Pin Configuration

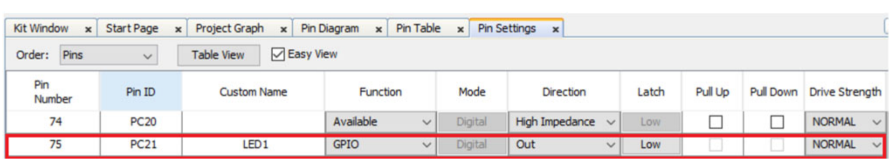


Pin Number	Pin ID	Custom Name	Function	Mode	Direction	Latch	Pull Up	Pull Down	Drive Strength
10	PA03		Available	Digital	High Impedance	Low	☐	☐	NORMAL
11	PB04		Available	Digital	High Impedance	Low	☐	☐	NORMAL
12	PB05		Available	Digital	High Impedance	Low	☐	☐	NORMAL
13	PD00	SW1	EIC_EXTINT0	Digital	High Impedance	n/a	☑	☐	NORMAL
14	AVSS			Digital	High Impedance	Low	☐	☐	NORMAL
15	AVDD			Digital	High Impedance	Low	☐	☐	NORMAL
16	PD01	SW2	EIC_EXTINT1	Digital	High Impedance	n/a	☑	☐	NORMAL
17	PB06		Available	Digital	High Impedance	Low	☐	☐	NORMAL

Figure 1-15. Pin Settings Window - SERCOM Pin Configuration

41	PB12		SERCOM4_PAD0	Digital	High Impedance	n/a	☐	☐	NORMAL
42	PB13		SERCOM4_PAD1	Digital	High Impedance	n/a	☐	☐	NORMAL
43	PB14		Available	Digital	High Impedance	Low	☐	☐	NORMAL
44	PB15		Available	Digital	High Impedance	Low	☐	☐	NORMAL
45	VSS			Digital	High Impedance	Low	☐	☐	NORMAL
46	VDD			Digital	High Impedance	Low	☐	☐	NORMAL
47	PD08		SERCOM6_PAD1	Digital	High Impedance	n/a	☐	☐	NORMAL
48	PD09		SERCOM6_PAD0	Digital	High Impedance	n/a	☐	☐	NORMAL

Figure 1-16. Pin Settings Window - LED Pin Configuration

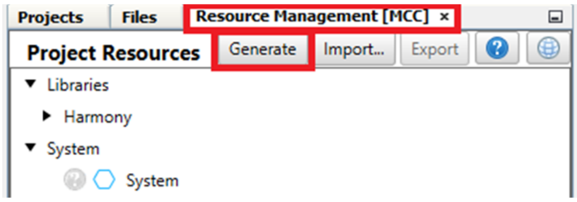


Pin Number	Pin ID	Custom Name	Function	Mode	Direction	Latch	Pull Up	Pull Down	Drive Strength
74	PC20		Available	Digital	High Impedance	Low	☐	☐	NORMAL
75	PC21	LED1	GPIO	Digital	Out	Low	☐	☐	NORMAL

1.2 Generating Code

After configuring the peripherals, click **Resource Management [MCC]** and then click on the **Generate** tab.

Figure 1-17. Generation of Code



Note: The generated code will add files and folders to the 32-bit MCC Harmony v3 project. In the generated code, notice the Peripheral Library files generated for Real-Time Clock (RTC), External Interrupt Controller (EIC), PORT peripherals, SERCOM4 (as Universal Synchronous Asynchronous Receiver Transmitter (USART)), Direct Memory Access (DMA) peripherals, and SERCOM6 (as I^2C PLIB). The MCC also generates the `main.c` file.

Note: The MCC provides an option to change the generated file name, and if this option is not used, by default, the file name `main.c` is generated.

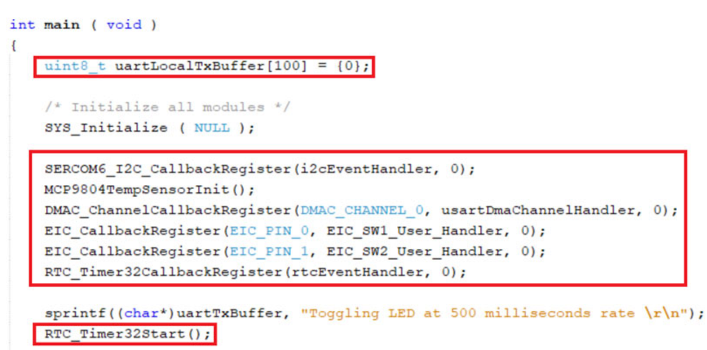
1.3 Adding Application Logic to the Project

To develop and run the application, follow these steps:

1. Open the `main.c` file of the project and add the following application logic:

```
//Declare and Define the array inside main() function:
uint8_t uartLocalTxBuffer[100] = {0};
//Register the callback event handlers:
SERCOM6_I2C_CallbackRegister(i2cEventHandler, 0);
MCP9804TempSensorInit();
DMAC_ChannelCallbackRegister(DMAC_CHANNEL_0,usartDmaChannelHandler, 0);
EIC_CallbackRegister(EIC_PIN_0, EIC_SW1_User_Handler, 0);
EIC_CallbackRegister(EIC_PIN_1, EIC_SW2_User_Handler, 0);
RTC_Timer32CallbackRegister(rtcEventHandler, 0);

sprintf((char*)uartTxBuffer, "Toggling LED at 500 milliseconds rate \r\n");
//Start the Timer:
    RTC_Timer32Start();
```

Figure 1-18. Adding Application Logic to Register Callback Event Handlers



2. Implement the registered callback event handlers for peripherals by adding the following code:

```
static void EIC_SW1_User_Handler(uintptr_t context)
{
    if(SW1_Get() == SWITCH_PRESSED_STATE)
    {
        changeTempSamplingRate = true;
```

```
    }
}
```

```
static void EIC_SW2_User_Handler(uintptr_t context)
{
    if(SW2_Get() == SWITCH_PRESSED_STATE)
    {
        if(false == startTemperatureReading)
        {
            startTemperatureReading = true;
            sprintf((char*)uartTxBuffer, "************* Printing Temperature
*************\r\n");
            TemperatureReadStartMsgLen = strlen((const char*)uartTxBuffer);
        }
        else
        {
            startTemperatureReading = false;
            sprintf((char*)uartTxBuffer, "************* Printing Toggling LED rate
*************\r\n");
            TemperatureReadStartMsgLen = strlen((const char*)uartTxBuffer);
        }
    }
}
```

```
static void rtcEventHandler (RTC_TIMER32_INT_MASK intCause, uintptr_t context)
{
    if (intCause & RTC_MODE0_INTENSET_CMP0_Msk)
    {
        isRTCExpired    = true;
    }
}
```

```
static void usartDmaChannelHandler(DMAC_TRANSFER_EVENT event, uintptr_t contextHandle)
{
    if (event == DMAC_TRANSFER_EVENT_COMPLETE)
    {
        isUARTTxComplete = true;
    }
}
```

```
static void i2cEventHandler(uintptr_t contextHandle)
{
    if (SERCOM6_I2C_ErrorGet() == SERCOM_I2C_ERROR_NONE)
    {
        isTemperatureRead = true;
    }
}
```

3. According to the status of the *isRTCExpired* and *isUARTTxComplete* flags, the LED1 is toggled at a default rate of 500 ms. These flags are handled by the *rtcEventHandler* and the *usartDmaChannelHandler* when the RTC Timer expires, and when the UART completes the transfer of data. To change the toggling rate, if the user presses the SW1 switch, the toggling rate changes to 1 second, 2 second, 4 second, and back to 500 millisecond with subsequent switch press events. The *EIC_SW1_User_Handler* will be responsible for changing the toggling rate when the user presses the SW1 switch on the board.

Figure 1-19. Application Logic to Print LED Toggling Rate

Inside the while loop, add the following code to toggle the LED at a default rate of 500 ms:

```
while(false == startTemperatureReading)
        {
            if ((isRTCExpired == true) && (true == isUARTTxComplete))
            {
                isRTCExpired = false;
                isUARTTxComplete = false;
                LED1_Toggle();
                sprintf((char*)(uartTxBuffer + 
TemperatureReadStartMsgLen), "Toggling LED at %s rate \r\n", 
&timeouts[(uint8_t)tempSampleRate][0]);
                TemperatureReadStartMsgLen = 0;
                DMAC_ChannelTransfer(DMAC_CHANNEL_0, uartTxBuffer, \
                    (const void *)&(SERCOM4_REGS->USART_INT.SERCOM_DATA), \
                    strlen((const char*)uartTxBuffer));
            }
```

Add the following code immediately after adding the above code to change the toggling rate when there is a switch press event:

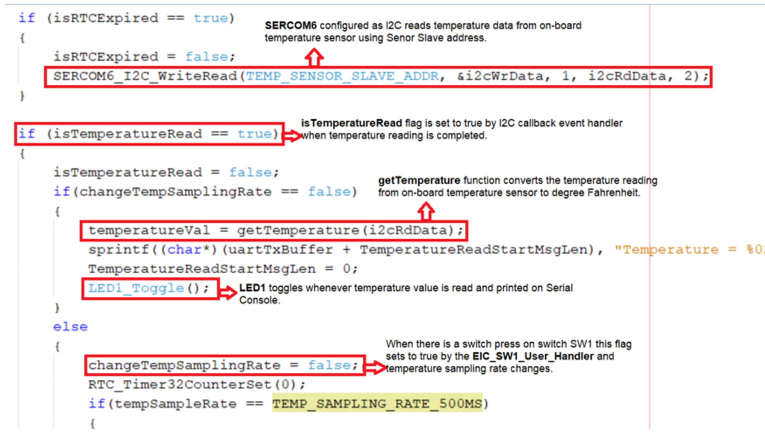
```
if(changeTempSamplingRate == true)
            {
                changeTempSamplingRate = false;
                if(tempSampleRate == TEMP_SAMPLING_RATE_500MS)
                {
                    tempSampleRate = TEMP_SAMPLING_RATE_1S;
                    RTC_Timer32Compare0Set(PERIOD_1S);
                }
                else if(tempSampleRate == TEMP_SAMPLING_RATE_1S)
                {
                    tempSampleRate = TEMP_SAMPLING_RATE_2S;
                    RTC_Timer32Compare0Set(PERIOD_2S);
                }
                else if(tempSampleRate == TEMP_SAMPLING_RATE_2S)
                {
                    tempSampleRate = TEMP_SAMPLING_RATE_4S;
                    RTC_Timer32Compare0Set(PERIOD_4S);
                }
                else if(tempSampleRate == TEMP_SAMPLING_RATE_4S)
                {
                   tempSampleRate = TEMP_SAMPLING_RATE_500MS;
                   RTC_Timer32Compare0Set(PERIOD_500MS);
                }
                else
                {
                    ;
                }
                RTC_Timer32CounterSet(0);
                sprintf((char*)uartLocalTxBuffer, "LED Toggling rate is changed to
```

```
%s\r\n", &timeouts[(uint8_t)tempSampleRate][0]);
                DMAC_ChannelTransfer(DMAC_CHANNEL_0, uartLocalTxBuffer, \
                    (const void *)&(SERCOM4_REGS->USART_INT.SERCOM_DATA), \
                    strlen((const char*)uartLocalTxBuffer));
            }
        }
```

4. Inside the while loop if the *startTemperatureReading* flag status is true, then temperature reading is printed on the serial console, otherwise the LED toggling rate will be printed. The status of this flag is controlled by the *EIC_SW2_User_Handler* when there is a SW2 switch press event. Add the logic to read and print the temperature reading from the temperature sensor:

```
if (isRTCExpired == true)
        {
            isRTCExpired = false;
            SERCOM6_I2C_WriteRead(TEMP_SENSOR_SLAVE_ADDR, &i2cWrData, 1, i2cRdData, 2);
        }

        if (isTemperatureRead == true)
        {
            isTemperatureRead = false;
            if(changeTempSamplingRate == false)
            {
                temperatureVal = getTemperature(i2cRdData);
                sprintf((char*)(uartTxBuffer + TemperatureReadStartMsgLen), "Temperature =
%02d F\r\n", (int)temperatureVal);
                TemperatureReadStartMsgLen = 0;
                LED1_Toggle();
            }
            else
            {
                changeTempSamplingRate = false;
                RTC_Timer32CounterSet(0);
                if(tempSampleRate == TEMP_SAMPLING_RATE_500MS)
                {
                    tempSampleRate = TEMP_SAMPLING_RATE_1S;
                    sprintf((char*)uartTxBuffer, "Sampling Temperature every 1 second
\r\n");
                    RTC_Timer32Compare0Set(PERIOD_1S);
                }
                else if(tempSampleRate == TEMP_SAMPLING_RATE_1S)
                {
                    tempSampleRate = TEMP_SAMPLING_RATE_2S;
                    sprintf((char*)uartTxBuffer, "Sampling Temperature every 2 seconds
\r\n");
                    RTC_Timer32Compare0Set(PERIOD_2S);
                }
                else if(tempSampleRate == TEMP_SAMPLING_RATE_2S)
                {
                    tempSampleRate = TEMP_SAMPLING_RATE_4S;
                    sprintf((char*)uartTxBuffer, "Sampling Temperature every 4 seconds
\r\n");
                    RTC_Timer32Compare0Set(PERIOD_4S);
                }
                else if(tempSampleRate == TEMP_SAMPLING_RATE_4S)
                {
                   tempSampleRate = TEMP_SAMPLING_RATE_500MS;
                   sprintf((char*)uartTxBuffer, "Sampling Temperature every 500 ms \r\n");
                   RTC_Timer32Compare0Set(PERIOD_500MS);
                }
                else
                {
                    ;
                }
                RTC_Timer32Start();
            }
            DMAC_ChannelTransfer(DMAC_CHANNEL_0, uartTxBuffer, \
                (const void *)&(SERCOM4_REGS->USART_INT.SERCOM_DATA), \
                strlen((const char*)uartTxBuffer));
        }
```

Figure 1-20. Application Logic to Print Temperature Reading



5. Add the following code to include the necessary header files, and define the macros for different RTC compare values:

```
#include <stdio.h>
#include <stddef.h>                        // Defines NULL
#include <stdbool.h>                       // Defines true
#include <stdlib.h>                        // Defines EXIT_FAILURE
#include <string.h>
#include "definitions.h"                   // SYS function prototypes

#define TEMP_SENSOR_SLAVE_ADDR             0x18
#define TEMP_SENSOR_REG_ADDR               0x05

#define SWITCH_PRESSED_STATE               1   // Active HIGH switch

/* RTC Time period match values for input clock of 1 KHz */
#define PERIOD_500MS                       512
#define PERIOD_1S                          1024
#define PERIOD_2S                          2048
#define PERIOD_4S                          4096
```

This code declares various flags whose status is monitored and changed by event handlers in the application. It has various declarations and definitions of arrays used to read the data from the temperature sensor and print to the console.

```
typedef enum
{
    TEMP_SAMPLING_RATE_500MS = 0,
    TEMP_SAMPLING_RATE_1S = 1,
    TEMP_SAMPLING_RATE_2S = 2,
    TEMP_SAMPLING_RATE_4S = 3,
} TEMP_SAMPLING_RATE;

static TEMP_SAMPLING_RATE tempSampleRate = TEMP_SAMPLING_RATE_500MS;
static volatile bool changeTempSamplingRate = false;
static volatile bool startTemperatureReading = false;
static volatile uint8_t TemperatureReadStartMsgLen = 0x00;
static volatile bool isTemperatureRead = false;
static volatile bool isRTCExpired = false;
static volatile bool isUARTTxComplete = true;
static uint8_t temperatureVal = 0;
static uint8_t i2cWrData = TEMP_SENSOR_REG_ADDR;
static uint8_t i2cRdData[2] = {0};
static const char timeouts[4][20] = {"500 milliSeconds", "1 Second",  "2 Seconds",  "4
Seconds"};
static uint8_t uartTxBuffer[100] = {0};
```

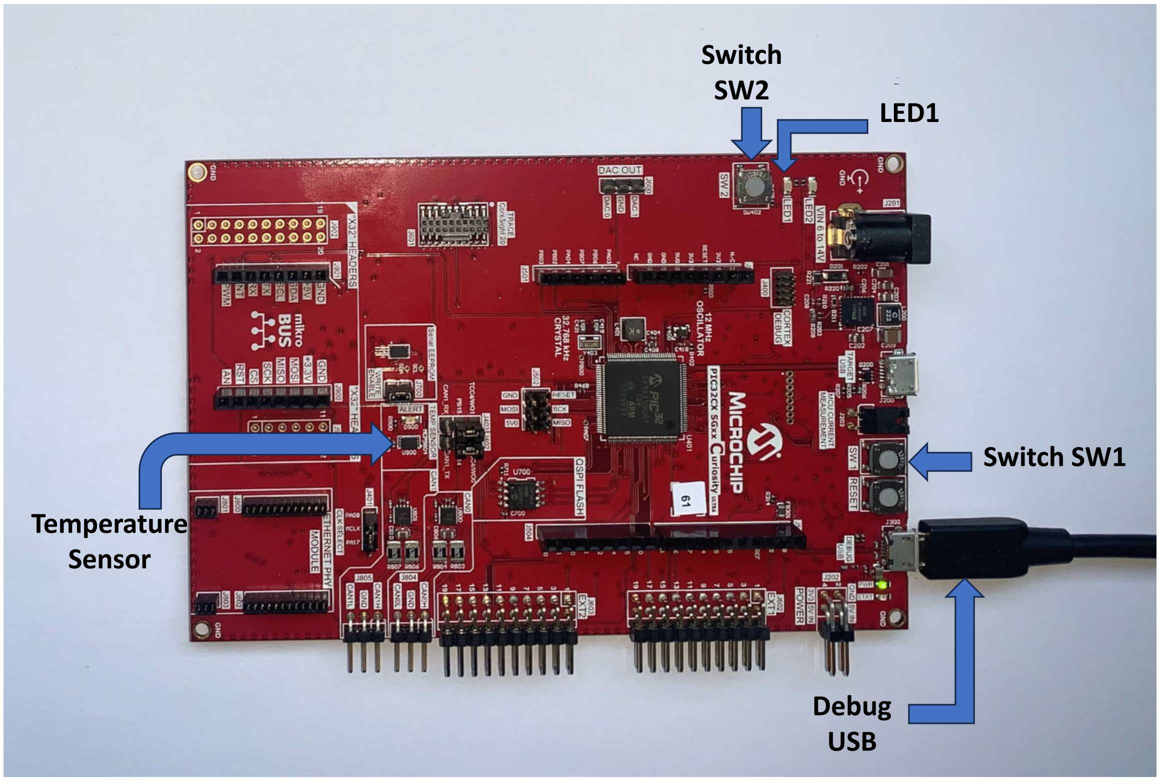
6. Add the functions to initialize the temperature sensor and the function to convert the temperature reading to degrees Fahrenheit.

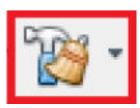
```
static void MCP9804TempSensorInit(void)
{
    uint8_t config[3] = {0};
    config[0] = 0x01;
    config[1] = 0x00;
    config[2] = 0x00;

    SERCOM6_I2C_Write(TEMP_SENSOR_SLAVE_ADDR, config, 3);

    while (isTemperatureRead != true);
    isTemperatureRead = false;

    config[0] = 0x08;
    config[1] = 0x03;
    SERCOM6_I2C_Write(TEMP_SENSOR_SLAVE_ADDR, config, 2);

    while (isTemperatureRead != true);
    isTemperatureRead = false;
}

static uint8_t getTemperature(uint8_t* rawTempValue)
{

    int temp = ((rawTempValue[0] & 0x1F) * 256 + rawTempValue[1]);
    if(temp > 4095)
    {
        temp -= 8192;
    }
    float cTemp = temp * 0.0625;
    float fTemp = cTemp * 1.8 + 32;
    return (uint8_t)fTemp;
}
```

1.4 Building and Programming the Application

1. The PIC32CX SG61 Curiosity Ultra Evaluation Board supports debugging using an Embedded Debugger (EDBG). Connect the "Type-A male to micro-B" USB cable to the micro-B debug USB port on the PIC32CX SG61 Curiosity Ultra Evaluation Board to power and debug the PIC32CX SG61 Curiosity Ultra Evaluation Board.

Figure 1-21. Hardware Setup

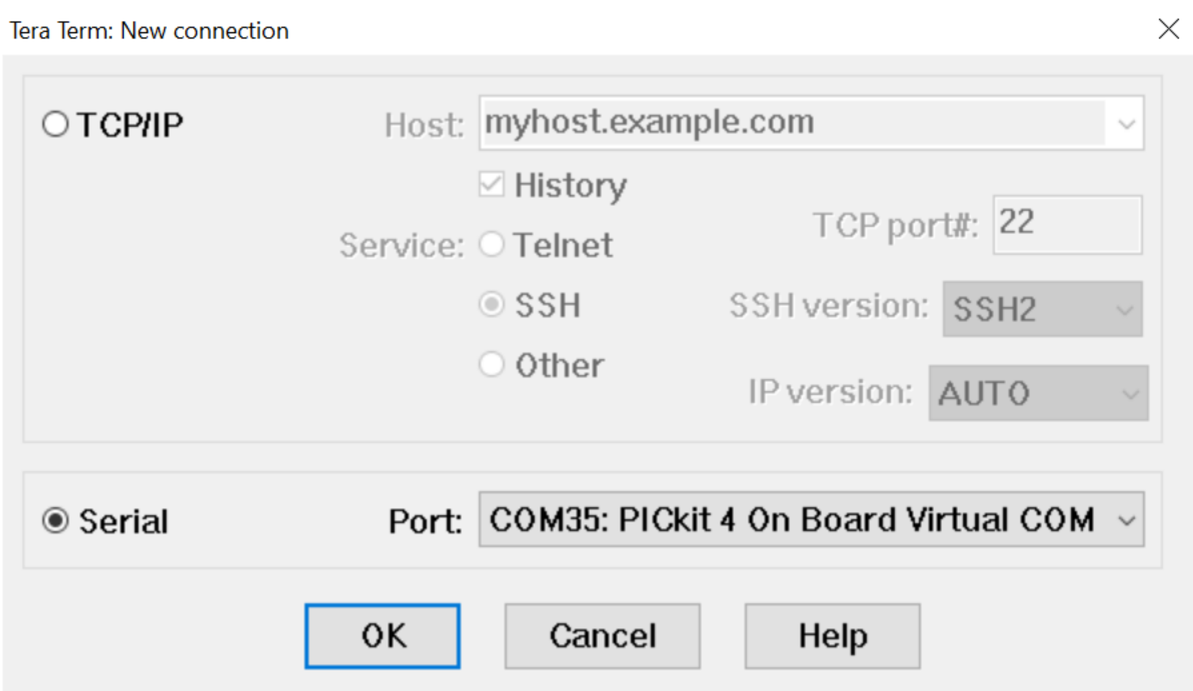


2. Set `getting_started_pic32cx_sg61_cult` as the main project, and from Project Properties select the latest compiler version (v4.30).
3. To clean and build the project, click  (the Build icon).
4. To program the application, click  (the Program icon).

1.5 Observing the Output on the Board and Serial Terminal

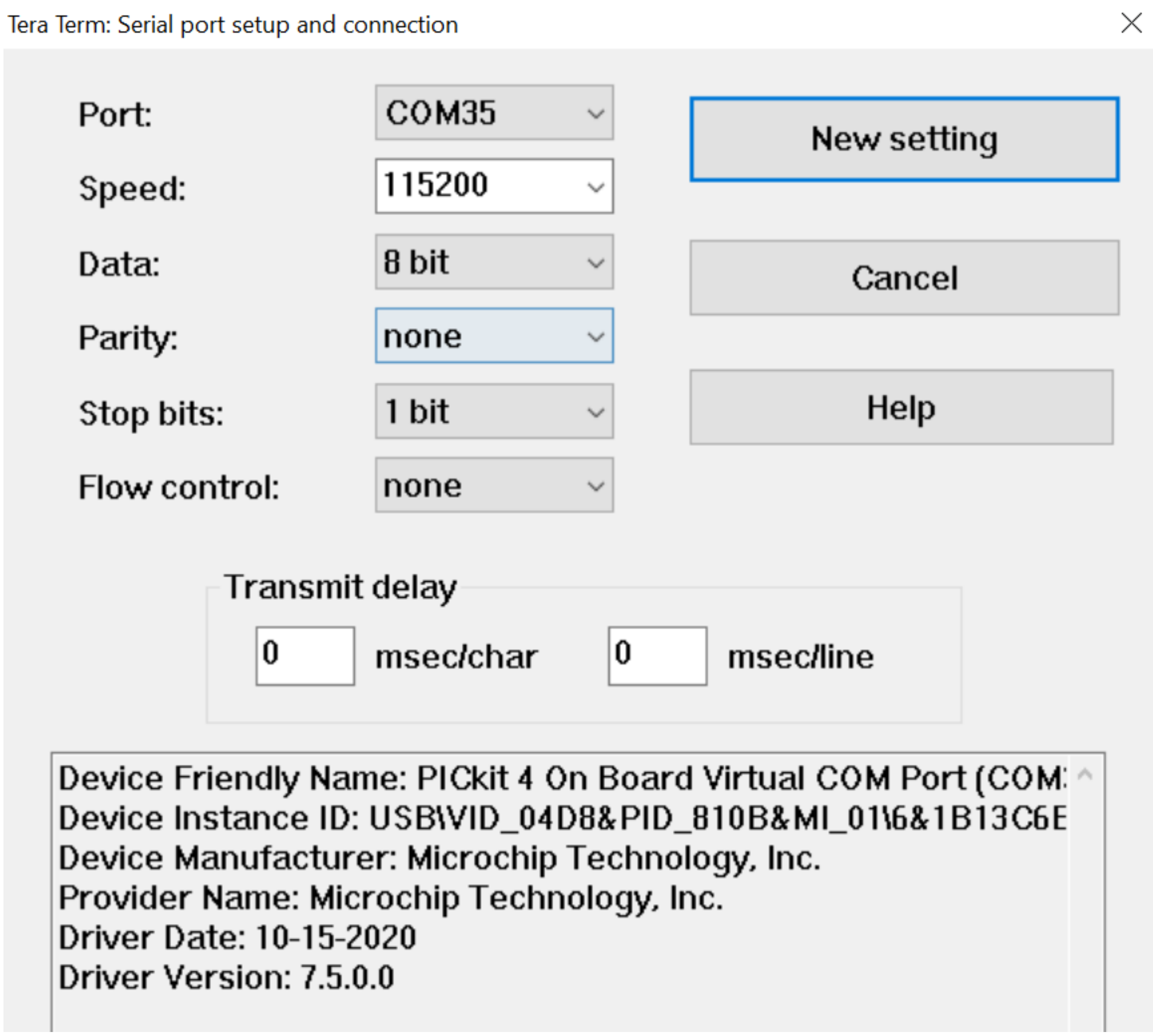
1. After building the application and completing the programming, open the Tera Term tool on the PC.

Figure 1-22. Selection of Serial Port



2. Select the Serial Port and set the baud rate as 115200.

Figure 1-23. Setting the Baud Rate



3. Press the Reset button on the PIC32CX SG61 Curiosity Ultra Evaluation Board. The LED will toggle at 500 millisecond by default and with every subsequent SW1 switch press, the LED toggling rate will change to 1 second, 2 second, and 4 second.
4. Press the SW2 switch on the PIC32CX SG61 Curiosity Ultra Evaluation Board to read and print the temperature on the Serial Terminal application running on the PC.
5. Press the SW1 switch on the PIC32CX SG61 Curiosity Ultra Evaluation Board to change the periodicity of the temperature values displayed on the serial console.

Figure 1-24. LED Toggling Rate and Temperature Reading on Serial Terminal

```
Toggling LED at 500 milliSeconds rate
Toggling LED at 500 milliSeconds rate
Toggling LED at 500 milliSeconds rate
Toggling LED at 500 milliSeconds rate
LED Toggling rate is changed to 1 Second
Toggling LED at 1 Second rate
Toggling LED at 1 Second rate
Toggling LED at 1 Second rate
LED Toggling rate is changed to 2 Seconds
Toggling LED at 2 Seconds rate
Toggling LED at 2 Seconds rate
LED Toggling rate is changed to 4 Seconds
Toggling LED at 4 Seconds rate
************* Printing Temperature *************
Temperature = 84 F
Temperature = 87 F
Temperature = 87 F
Sampling Temperature every 500 ms
Temperature = 87 F
Temperature = 87 F
Temperature = 87 F
Temperature = 87 F
Temperature = 87 F
Temperature = 87 F
Temperature = 87 F
Temperature = 87 F
Temperature = 87 F
Temperature = 87 F
Temperature = 87 F
```

The LED toggling rate on the Serial Terminal changes with every subsequent switch press. The same change is observed in the toggling rate of LED1 on the evaluation board.

2. Resources

- For additional information on MPLAB Harmony v3, refer to the Microchip web site: https://www.microchip.com/mplab/mplab-harmony and microchipdeveloper.com/harmony3:start
- For more information on various applications, refer to: github.com/Microchip-MPLAB-Harmony/reference_apps
- For the example application, refer “Getting Started Application with PIC32CX SG61 Curiosity Ultra Evaluation Board” under the “Software” heading: www.microchip.com/en-us/development-tool/ev09h35a
- PIC32CX SG61 Curiosity Ultra Evaluation Board

3. Revision History

Revision A - November 2023

This is the initial release of this document.

Microchip Information

The Microchip Website

Microchip provides online support via our website at www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip product is strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is "unbreakable". Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.

Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

THIS INFORMATION IS PROVIDED BY MICROCHIP "AS IS". MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION.

Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, Adaptec, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, CryptoMemory, CryptoRF, dsPIC, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

AgileSwitch, ClockWorks, The Embedded Control Solutions Company, EtherSynch, Flashtec, Hyper Speed Control, HyperLight Load, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, TimeCesium, TimeHub, TimePictra, TimeProvider, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, Augmented Switching, BlueSky, BodyCom, Clockstudio, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, Espresso T1S, EtherGREEN, EyeOpen, GridTime, IdealBridge, IGaT, In-Circuit Serial Programming, ICSP, INICnet, Intelligent Paralleling, IntelliMOS, Inter-Chip Connectivity, JitterBlocker, Knob-on-Display, MarginLink, maxCrypto, maxView, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, mSiC, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, Power MOS IV, Power MOS 7, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, RTAX, RTG4, SAM-ICE, Serial Quad I/O, simpleMAP, SimpliPHY, SmartBuffer, SmartHLS, SMART-I.S., storClad, SQI, SuperSwitcher, SuperSwitcher II, Switchtec, SynchroPHY, Total Endurance, Trusted Time, TSHARC, Turing, USBCheck, VariSense, VectorBlox, VeriPHY, ViewSpan, WiperLock, XpressConnect, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, and Symmcom are registered trademarks of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

ISBN: 978-1-6683-3558-1

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: www.microchip.com/support Web Address: www.microchip.com **Atlanta** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **Austin, TX** Tel: 512-257-3370 **Boston** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **Chicago** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4485-5910 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-72400 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560
Dallas Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **Detroit** Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983 **Indianapolis** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000	**China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-72884388 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654

Worldwide Sales and Service

..........continued

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
San Jose, CA Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078			**UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820