Creating the First Application on PIC32CM JH01 Microcontrollers Using MPLAB Harmony v3 with MPLAB Code Configurator (MCC)

TB3353

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).

The PIC32CM JH family of microcontrollers (MCUs) is the next generation of the popular SAM C21 family of Arm® Cortex®-M0+ based MCUs. The PIC32CM JH family of MCUs delivers a variety of popular features plus extended memory options up to 512 KB of Flash and 64 KB to help create designs that need functional safety, enhanced touch, or security. This family pin is compatible with the SAM C21 family in the 32-pin, 48-pin and 64-pin packages.

This document explains how to create a simple application on a Cortex-M0+ based PIC32CM JH01 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. For this demonstration, the following MPLAB Harmony v3 modules are used and configured using the MCC:

- The PORT pin to toggle LED.
- Real-Time Clock (RTC) PLIB to periodically sample LED toggling rate.
- External Interrupt Controller (EIC) PLIB to change the toggling rate when there is a switch press event.
- Serial Communication Interfaces (SERCOM (SERCOM configured as USART)) and DMA PLIBs to print LED toggling rate on COM port terminal application (Serial Console) running on a computer.

1. Creating the First Application on the PIC32CM JH01 MCU

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

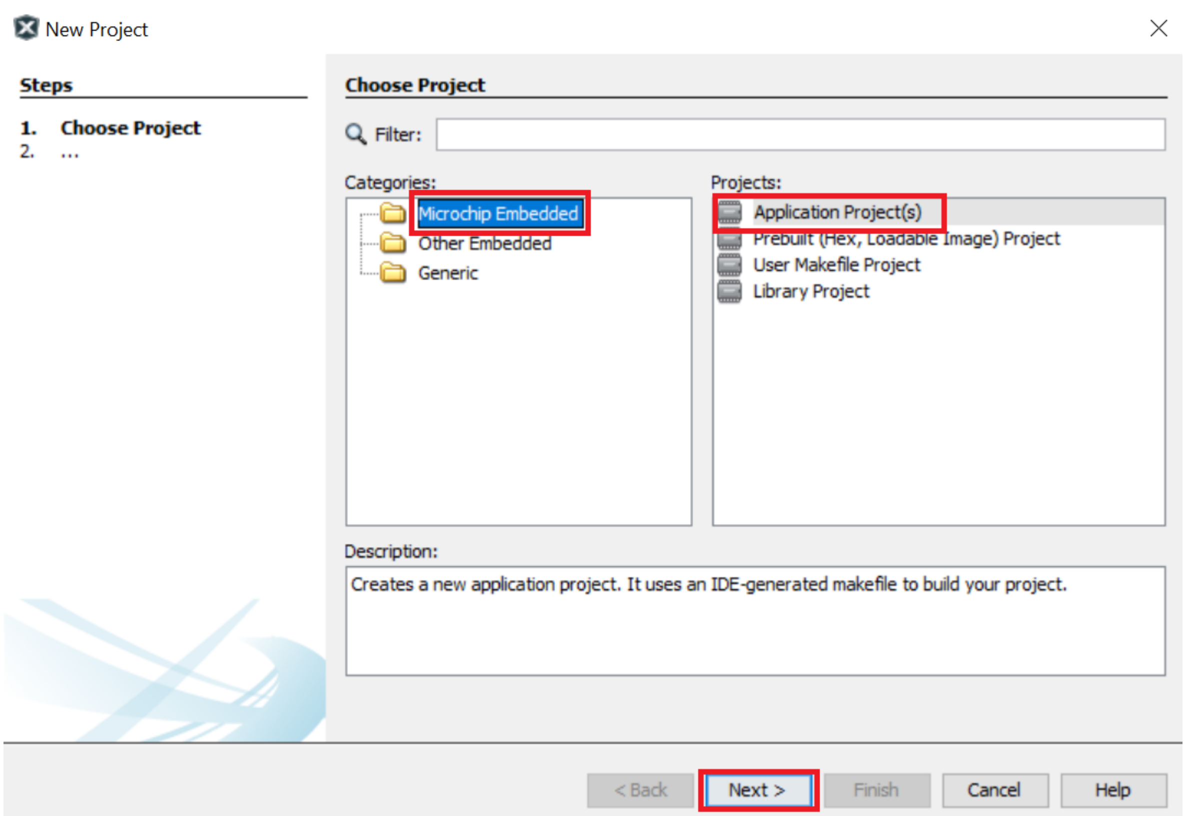
- MPLAB X IDE v6.20
- MPLAB Code Configurator Plug-in v5.5.0
- MPLAB XC32 Compiler v4.35
- MPLAB Harmony v3 repository: csp v3.18.2
- PIC32CM JH01 Curiosity Pro Evaluation Kit

Note: The latest versions of these tools can also be used to develop the application.

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

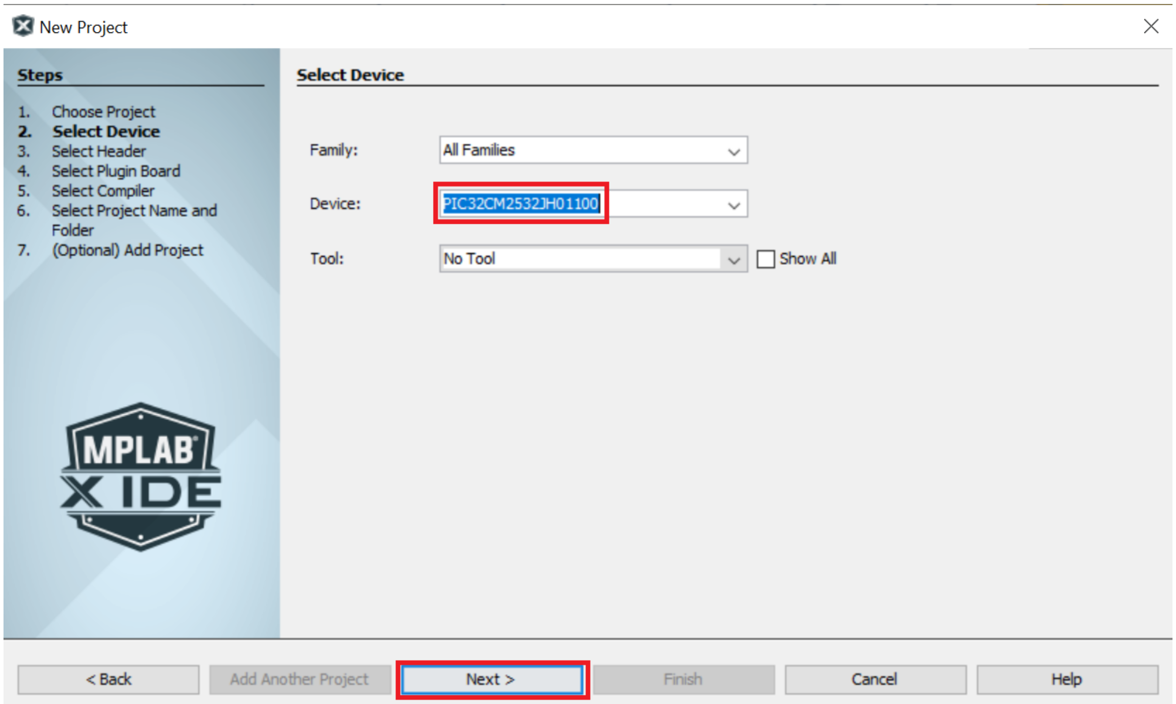
1. On the **Start** menu, launch MPLAB X IDE.
2. On the **File** menu, click **New Project** or click the *New Project* icon.
3. In the New Project window, in the left Navigation bar, under Steps click **Choose Project**.

Figure 1-1. Choose Project



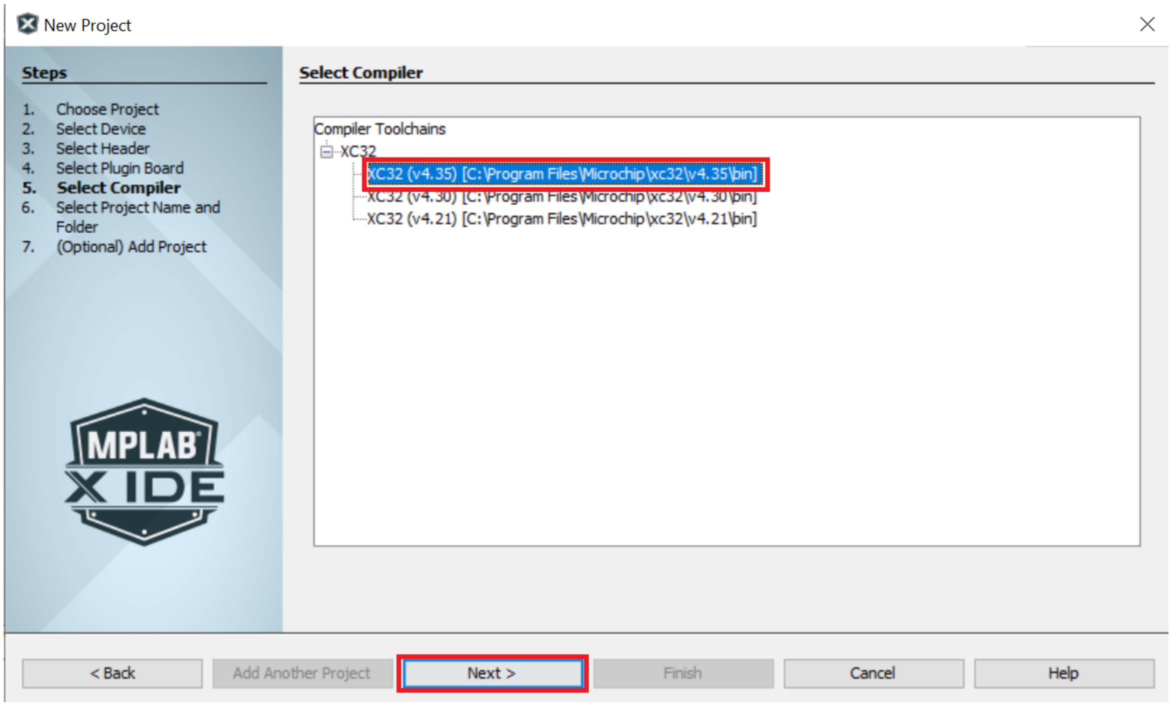
4. In the Choose Project property page:
   a. Categories: Selelct Micorchip Embedded.
   b. Project: Select Application Project(s).
5. Click **Next**.
6. In the left Navigation bar, under Steps click **Select Device**.

Figure 1-2. Selecting the Device



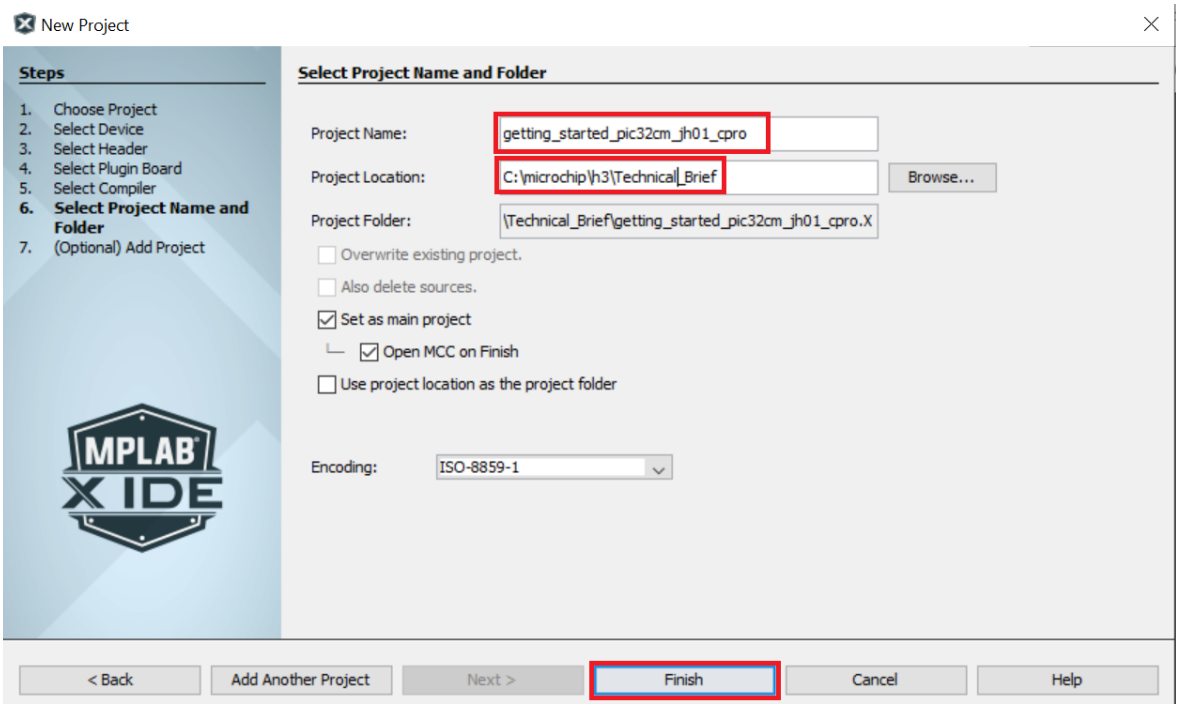
7. In the Select Device property page, in the **Device** box, type or select the device PIC32CM5164JH01100.
8. Click **Next**.
9. In the left Navigation bar, under Steps click **Select Compiler**.
10. In the Select Compiler property page, expand XC32 list of options, and then select **XC32 (v4.35)**.

Figure 1-3. Compiler Selection



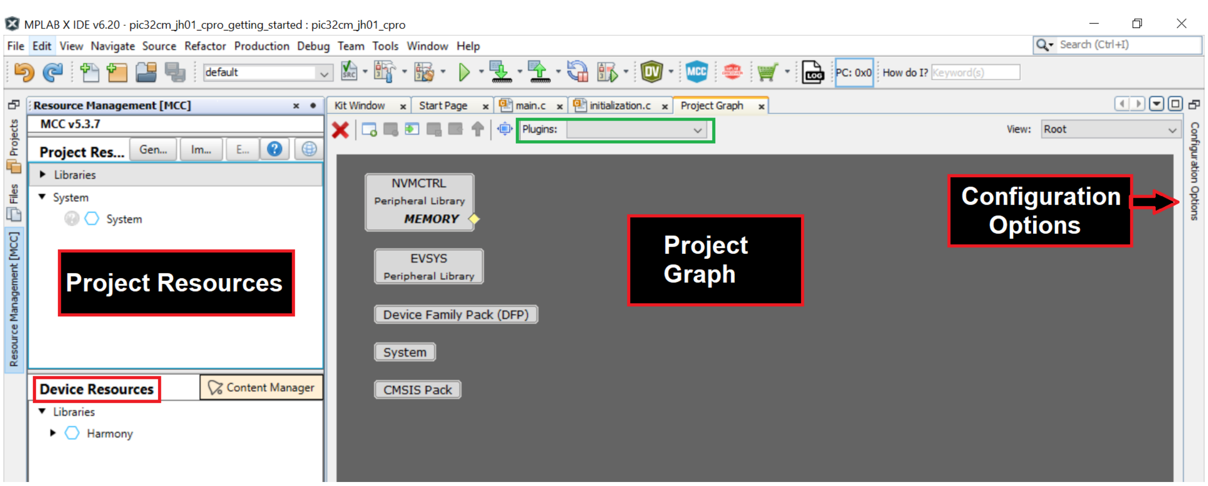
11. Click **Next**.
12. In the left Navigation bar, click **Select Project Name and Folder**.
13. In the Select Project Name and Folder property page:
    a. Project Name: Enter `getting_started_pic32cm_jh01_cpro`.
    b. Project Location: Click the **Browse** button and choose C:\microchip\h3\Tech_Brief.

Figure 1-4. Project Creation



14. Click **Finish**.
15. 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.
16. The MCC plug-in will open in a new window as shown in the following figure.

Figure 1-5. MPLAB Code Configurator Window

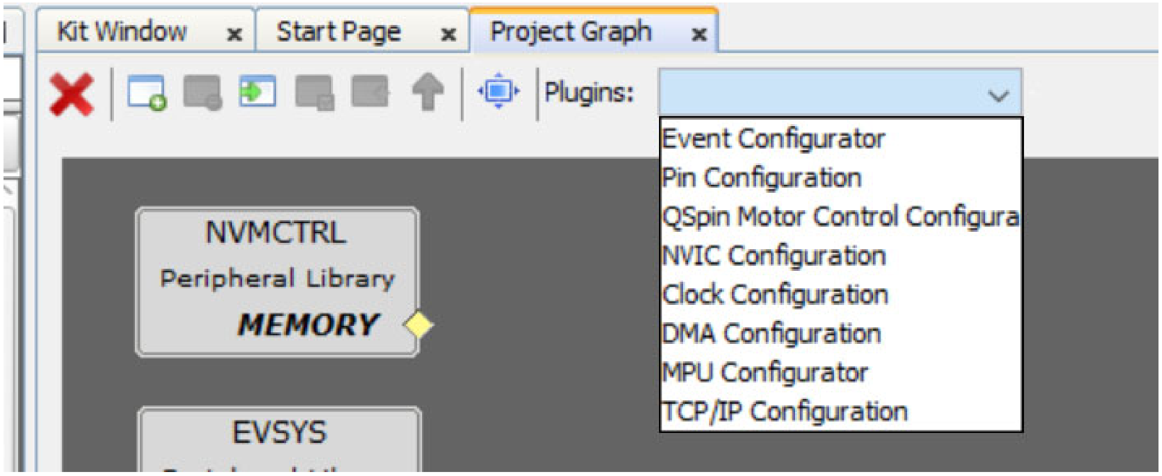


1.1 Adding and Configuring the MPLAB Harmony Components

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

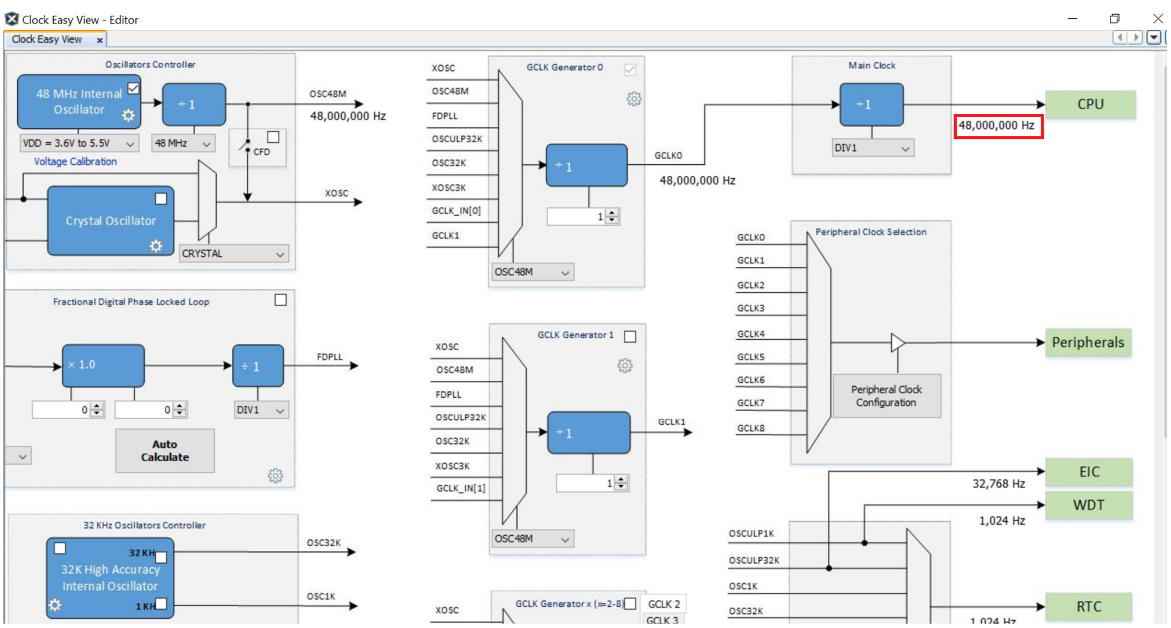
1. In the MCC window, click **Project Graph**.

Figure 1-6. MPLAB Code Configurator



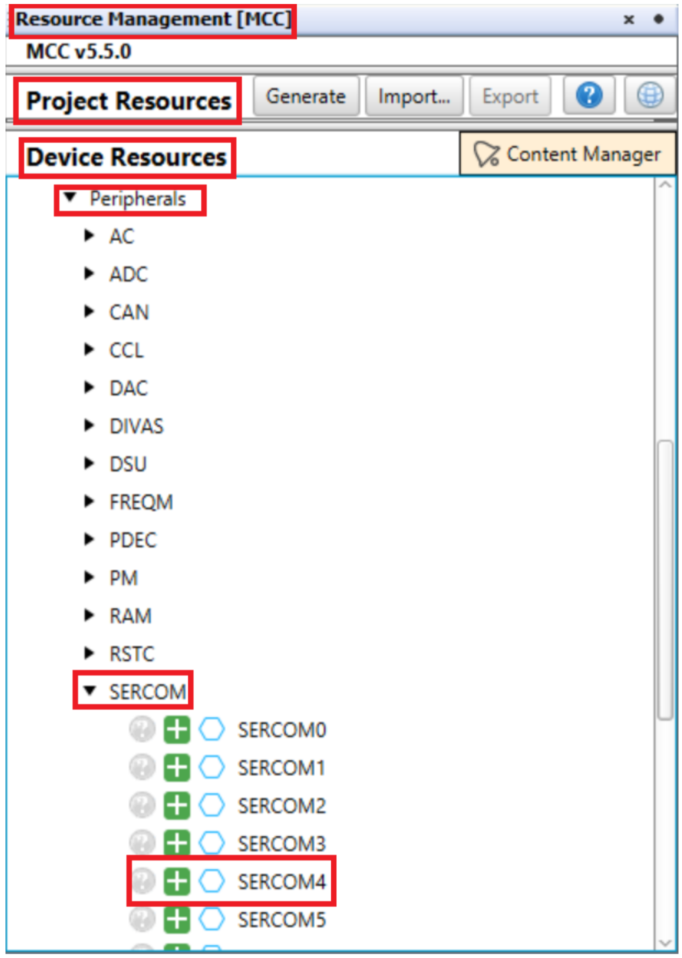
2. In the **Plugins** drop-down list, select **Clock Configuration**. The **Clock Easy View** window will be displayed, verify that the Main Clock is set to 48 MHz.
   Note: Make sure to make the following modification for the GCLK Generator 1.

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



3. Under **Device Resources**, expand the list of options *Harmony* > *Peripherals* > *SERCOM*.
4. Click **SERCOM4** and observe that the SERCOM4 Peripheral Library block is added in the Project Graph window.

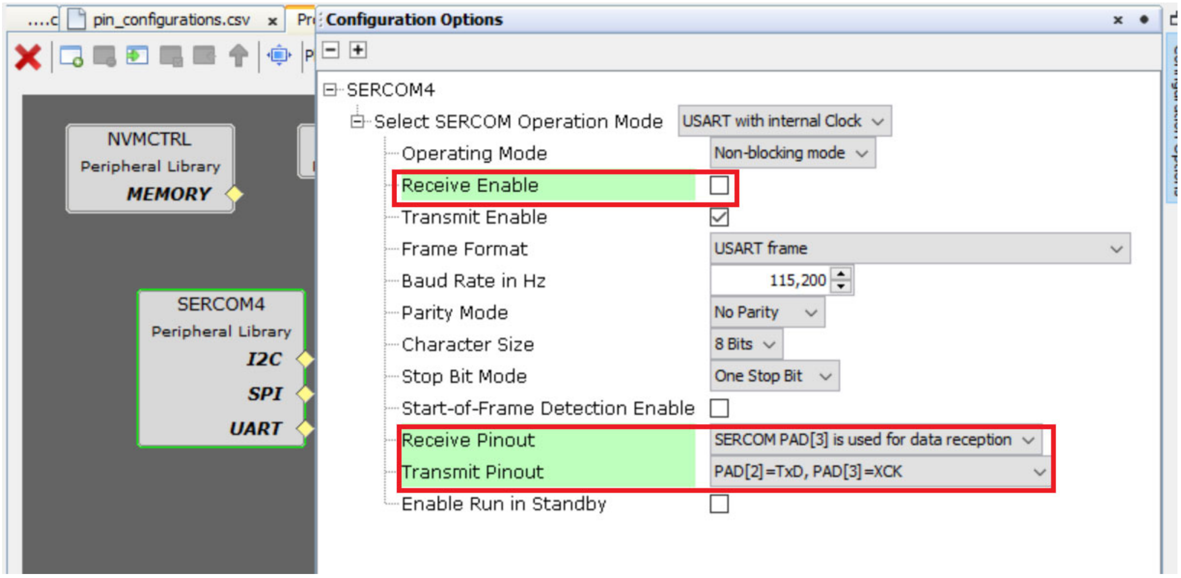
Figure 1-8. MPLAB Code Configurator - Selection of Peripherals



Note: Users can also select other peripherals under Device Resources, Harmony > Peripherals.

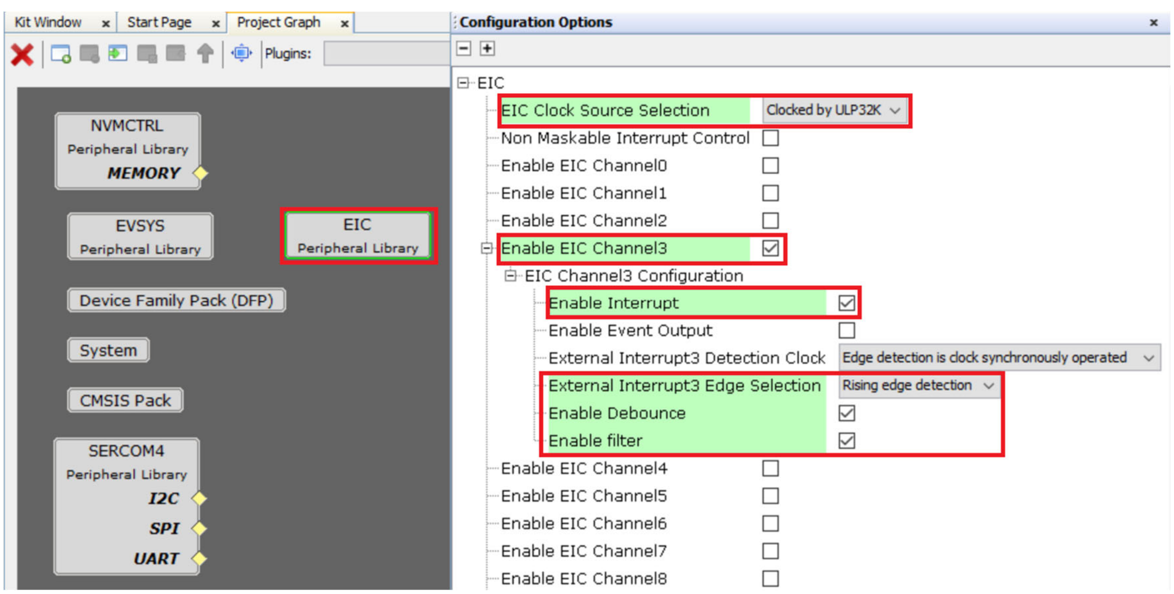
5. In the Project Graph window, in the left Navigation bar, select **SERCOM4 Peripheral Library**. In the right **Configuration Options** property page, configure it as follows to print the data on the Serial Console at 115200 baud rate.

Figure 1-9. MPLAB Code Configurator - SERCOM4 Configuration



6. Under **Device Resources** expand the list of options *Harmony* > *Peripherals* > *EIC*. Click **EIC** and observe that the EIC Peripheral Library block is added in the Project Graph window.
7. In the right Configuration Options property page, select the **Enable Interrupt** check box and the **Enable EIC Channel3** check box for the switch press event.

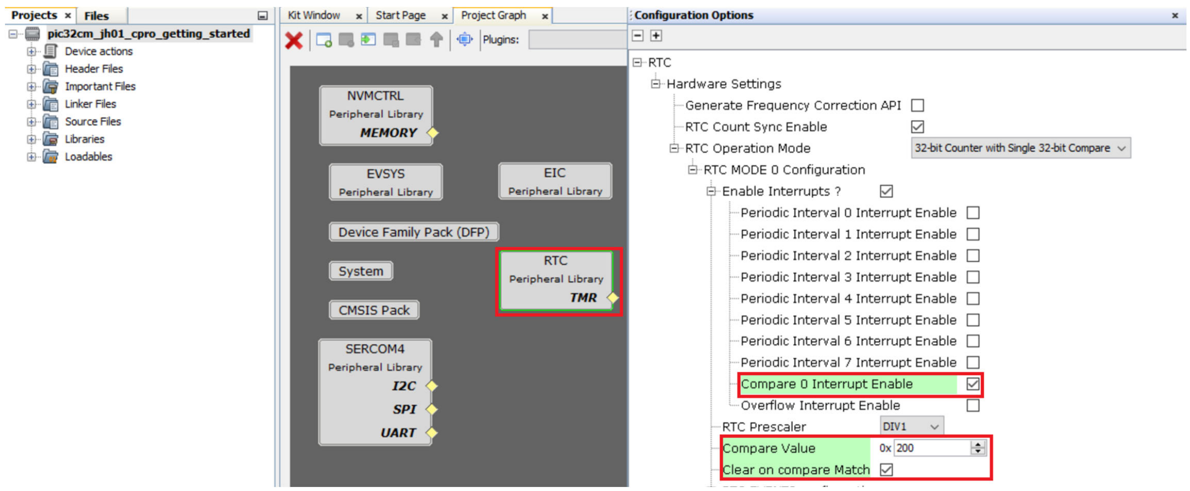
Figure 1-10. MPLAB Code Configurator - EIC Configuration



8. Under **Device Resources** expand the list of options *Harmony* > *Peripherals* > *RTC*. Click **RTC** and observe that the RTC Peripheral Library block is added in the Project Graph window to generate a compare interrupt every 500 milliseconds. Slect the **Compate 0 Interrupt Enable** check box.

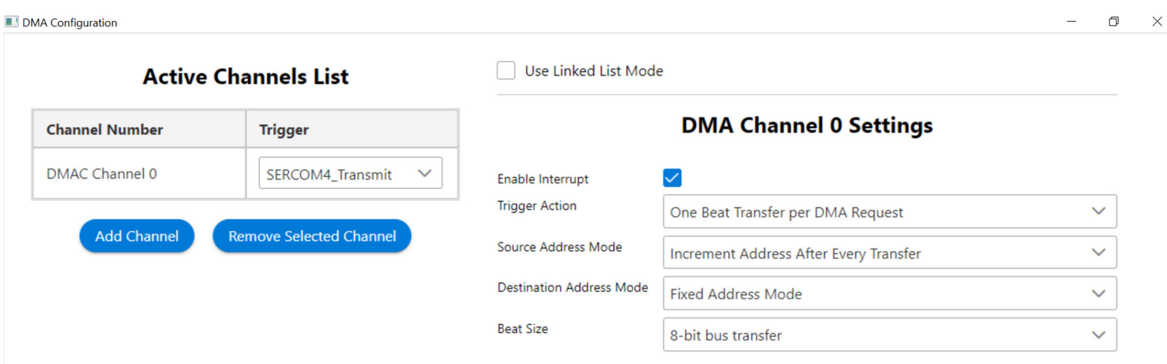
Note: The Compare Value is set as 0x200. This compare value generates an RTC compare interrupt every 500 milliseconds.
RTC clock = 1024 Hz
RTC Prescaler = 1
Required Interrupt rate = 500 ms
Hence, Compare Value = (500/1000) x 1024 = 512 (i.e., 0x200)

Figure 1-11. MPLAB Code Configurator - RTC PLIB Configuration



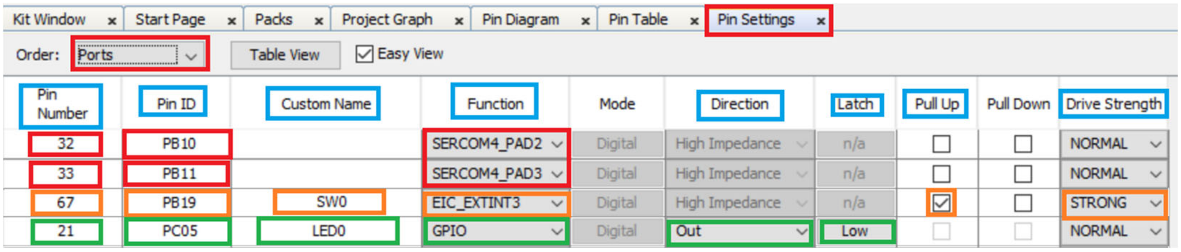
9. From the Plugins drop-down list, select **Add Channel** and then select **DMA Configuration**. Configure the DMA Channel 0 to transmit application buffer to the USART TX register. The DMA transfers one byte from the user buffer to the USART transmit buffer on each trigger.

Figure 1-12. MPLAB Code Configurator - DMA Configuration



10. From the Plugins drop-down list select **Pin Configuration** and then click **Pin Settings**.
11. In the **Order** box, type or select Ports. Build configurations according to the application as indicated below. Change the Custom Name of the pin IDs, PC05 and PB19, as shown in the following figure.

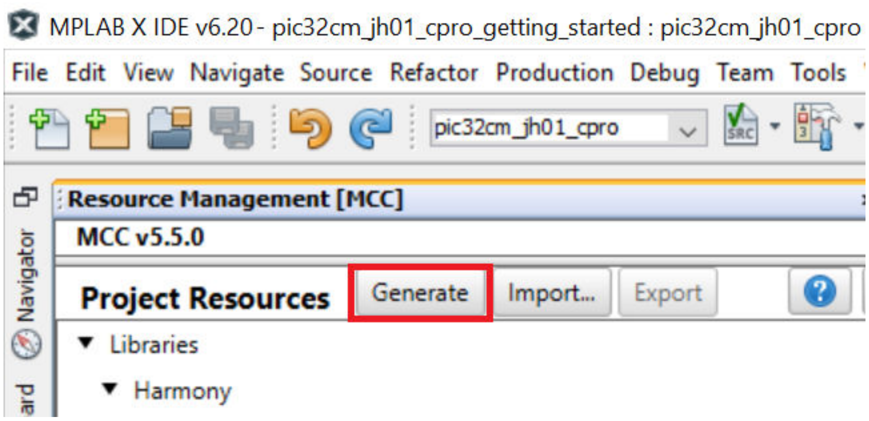
Figure 1-13. Pin Settings Window - SERCOM, EIC and LED Pin Configuration



1.2 Generating the Code

After configuring the peripherals, as shown in the following figure click **Generate** under **Resource Management [MCC]**.

Figure 1-14. Code Generation



Notes:

1. The generated code will add files and folders to the 32-bit MCC Harmony project. In the generated code, notice the Peripheral Library files generated for the Real-Time Clock (RTC), External Interrupt Controller (EIC), PORT peripherals, SERCOM4 (as Universal Synchronous Asynchronous Receiver Transmitter (USART)), and the Direct Memory Access (DMA) peripherals. The MCC also generates the `main.c` file.
2. The MCC provides an option to change the generated file name. By default, the file name `main.c` is generated if a name is not assigned.

1.3 Adding Application Logic to the Project

1. To develop and run the Application, use the following steps:
   Open the `main.c` file of the project and add the following application logic. Add the following code in the `main()` function:

```
uint8_t uartLocalTxBuffer[100] = {0};

    /* Initialize all modules */
    DMAC_ChannelCallbackRegister(DMAC_CHANNEL_0, uartDmaChannelHandler_Tx, 0);
    RTC_Timer32CallbackRegister(rtcEventHandler, 0);
    EIC_CallbackRegister(EIC_PIN_3, SW0_userHandler, 0);

    sprintf((char*)uartTxBuffer, "************* Printing Toggling LED rate
*************\r\n");
    DMAC_ChannelTransfer(DMAC_CHANNEL_0, uartTxBuffer, \
        (const void *)&(SERCOM4_REGS->USART_INT.SERCOM_DATA), \
        strlen((const char*)uartTxBuffer));
```

```
    /* Start the timer */
    RTC_Timer32Start();
```

Figure 1-15. Adding Application Logic to Register Callback Event Handlers

```
int main ( void )
{
    uint8_t uartLocalTxBuffer[100] = {0};

    /* Initialize all modules */
    SYS_Initialize ( NULL );
    DMAC_ChannelCallbackRegister(DMAC_CHANNEL_0, uartDmaChannelHandler_Tx,
    RTC_Timer32CallbackRegister(rtcEventHandler, 0);
    EIC_CallbackRegister(EIC_PIN_3, SW0_userHandler, 0);

    sprintf((char*)uartTxBuffer, "************* Printing Toggling LED rate
    DMAC_ChannelTransfer(DMAC_CHANNEL_0, uartTxBuffer, \
        (const void *)&(SERCOM4_REGS->USART_INT.SERCOM_DATA), \
        strlen((const char*)uartTxBuffer));

    /* Start the timer */
    RTC_Timer32Start();
```

2. Implement the registered callback event handlers for the peripherals by adding the following code before the `main()` function.

```
static void SW0_userHandler(uintptr_t context)
{
        changeTempSamplingRate = true;
}
```

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

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

3. According to the status of the `isRTCExpired` and `isUARTTxComplete` flags (These flags are handled by `rtcEventHandler` and `uartDmaChannelHandler_Tx` event handlers when RTC Timer expires and when UART completes the transfer of data.), the LED0 is toggled at a default rate of 500 ms. To change the toggling rate, if the user presses the SW0 switch, the toggling rate changes to 1s, 2s, and 4s and back to 500 ms with subsequent switch press events. `SW0_userHandler` will be responsible for changing the toggling rate when the user presses the SW0 switch on the board.

Inside the while loop, delete `SYS_Tasks()` and add the following code to toggle the LED at a default rate of 500 ms.

```
if ((isRTCExpired == true) && (true == isUARTTxComplete))
            {
                isRTCExpired = false;
                isUARTTxComplete = false;
                LED0_Toggle();
                sprintf((char*)(uartTxBuffer), "Toggling LED at %s rate \r\n",
&timeouts[(uint8_t)tempSampleRate][0]);
                DMAC_ChannelTransfer(DMAC_CHANNEL_0, uartTxBuffer, \
                    (const void *)&(SERCOM4_REGS->USART_INT.SERCOM_DATA), \
                    strlen((const char*)uartTxBuffer));
            }
```

4. 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_Timer32CompareSet(PERIOD_1S);
                }
                else if(tempSampleRate == TEMP_SAMPLING_RATE_1S)
                {
                    tempSampleRate = TEMP_SAMPLING_RATE_2S;
                    RTC_Timer32CompareSet(PERIOD_2S);
                }
                else if(tempSampleRate == TEMP_SAMPLING_RATE_2S)
                {
                    tempSampleRate = TEMP_SAMPLING_RATE_4S;
                    RTC_Timer32CompareSet(PERIOD_4S);
                }
                else if(tempSampleRate == TEMP_SAMPLING_RATE_4S)
                {
                   tempSampleRate = TEMP_SAMPLING_RATE_500MS;
                   RTC_Timer32CompareSet(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));
            }
```

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

```
#include <stdio.h>
              // SYS function prototypes

/* Timer Counter Time period match values for input clock of 4096 Hz */
#define PERIOD_500MS                    512
#define PERIOD_1S                       1024
#define PERIOD_2S                       2048
#define PERIOD_4S                       4096

#define TX_BUFFER_SIZE 100
```

This code declares the various flags whose status is monitored and changed by the event handlers in the application. It has various declarations and definitions of arrays used to print the LED toggling rate on the console.

```
static volatile bool isRTCExpired = false;
static volatile bool changeTempSamplingRate = false;
static volatile bool isUARTTxComplete = true;
```

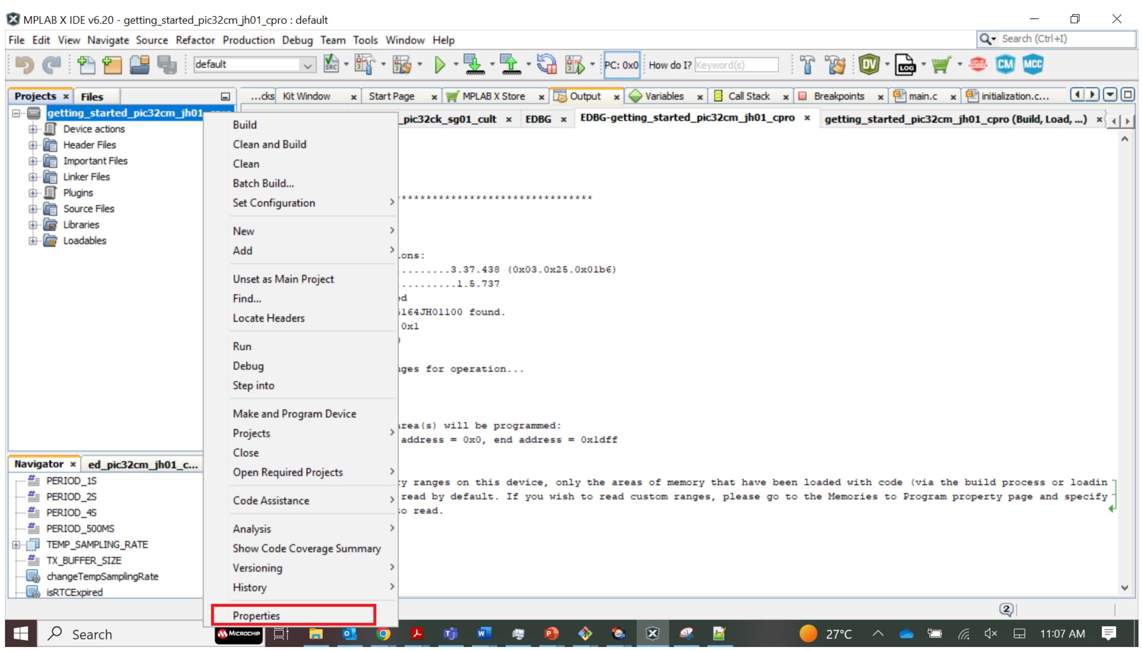
```
static volatile bool isUARTRxComplete = false;
static uint8_t uartTxBuffer[TX_BUFFER_SIZE] = {0};
typedef enum
{
    TEMP_SAMPLING_RATE_500MS = 0,
    TEMP_SAMPLING_RATE_1S = 1,
    TEMP_SAMPLING_RATE_2S = 2,
    TEMP_SAMPLING_RATE_4S = 3,
} TEMP_SAMPLING_RATE;

static TEMP_SAMPLING_RATE tempSampleRate = TEMP_SAMPLING_RATE_500MS;
static const char timeouts[4][20] = {"500 milliSeconds", "1 Second",  "2 Seconds",  "4
Seconds"};
```

1.4 Building and Programming the Application

1. The PIC32CM JH01 Curiosity Pro Evaluation Kit 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 PIC32CM JH01 Curiosity Pro Evaluation Kit to power and debug the PIC32CM JH01 Curiosity Pro Evaluation Kit.
2. Ensure that the compiler optimization is set to 1, to check that follow these steps:
   a. Right-click on the project *getting_started_pic32cm_jh01_cpro*, a shortcut menu appears. Click **Properties.**

Figure 1-16. Project Properties



   b. In the **Project Properties** window, for Option categores select **Optimization** and for the Optimation level from the item list select **1**.
   c. Click **OK** to close the **Project Properties** window.

Figure 1-17. Optimization level

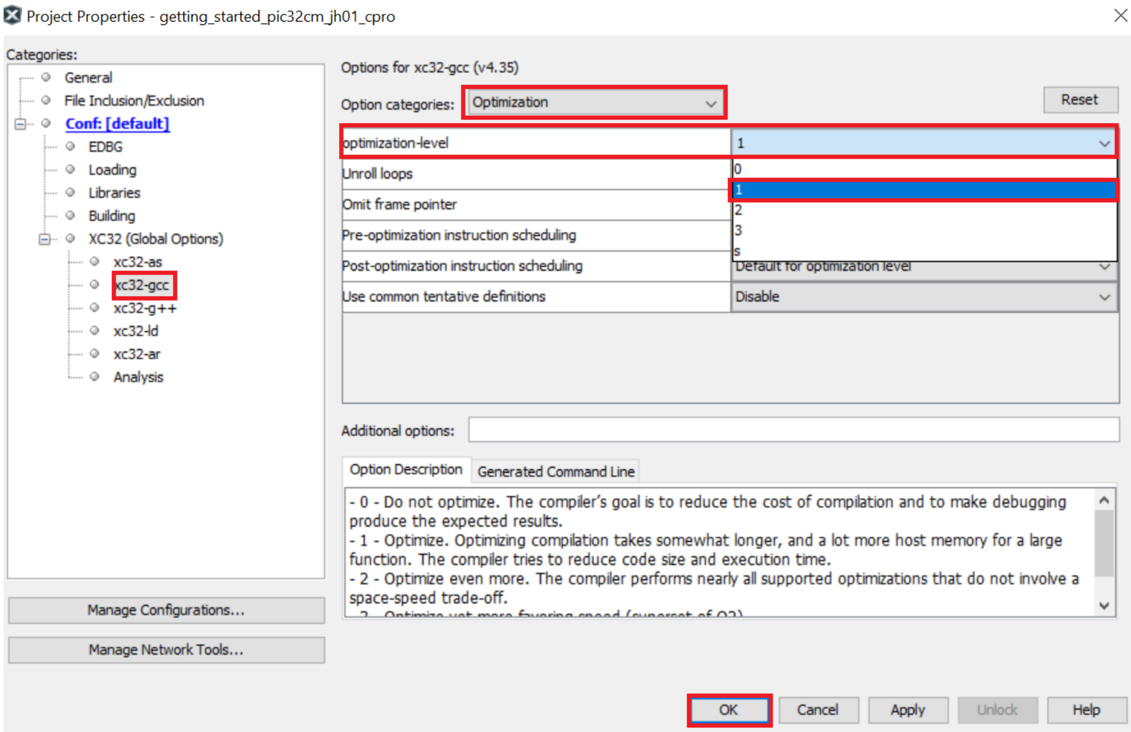
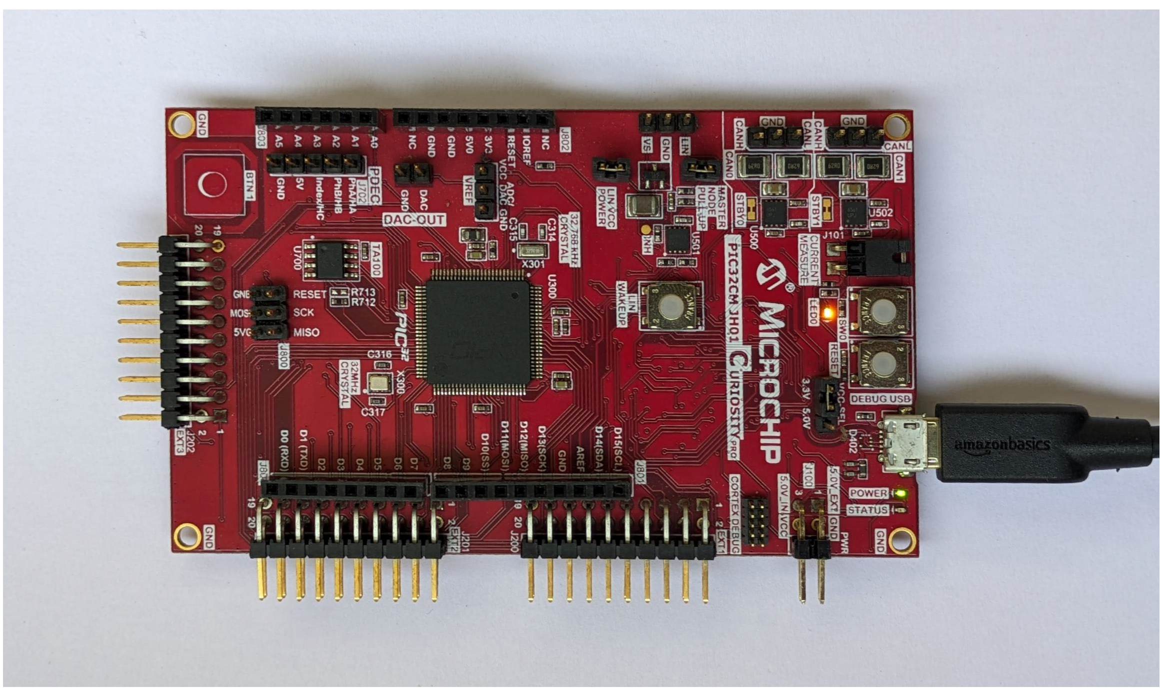


Figure 1-18. Hardware Setup



3. Set `getting_started_pic32cm_jh01_cpro` as the main project, and from Project Properties, select the latest compiler version (v4.35). Clean and build your project by clicking the highlighted icon.

Figure 1-19. Clean and Build

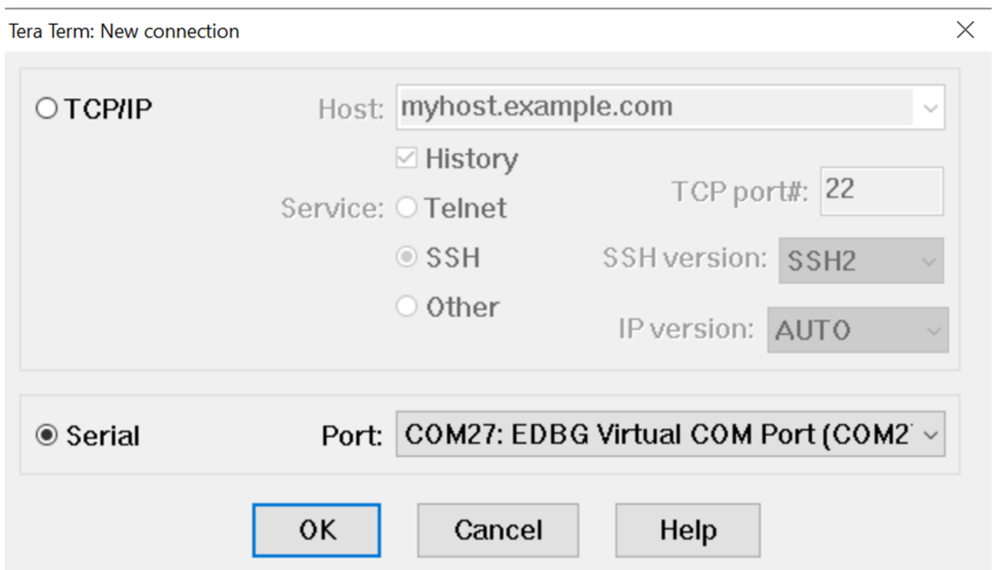
4. Program the application by clicking the highlighted icon.

Figure 1-20. Program the Device

1.5 Observing the Output on the Board and Serial Terminal

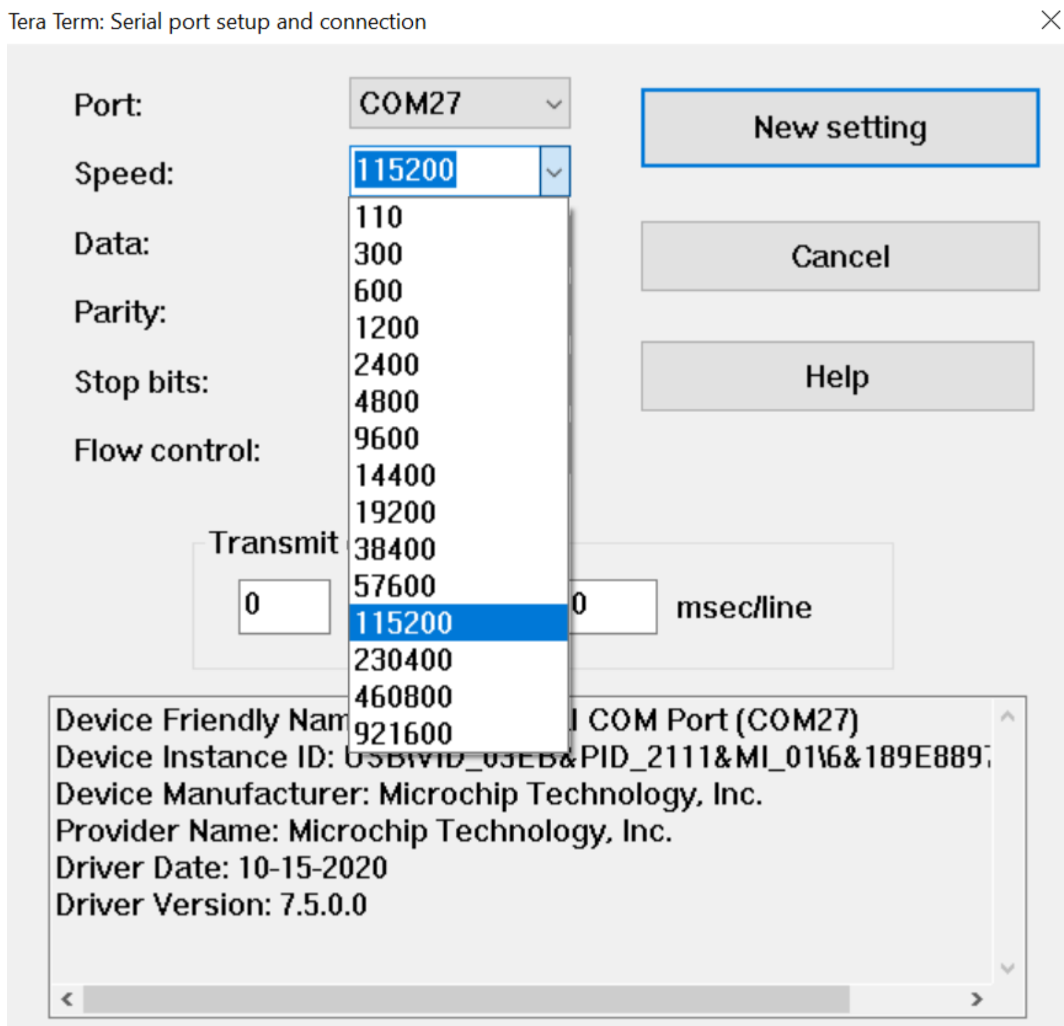
1. Press the **Start** button to open Tera Term or PuTTY terminal application on the computer from the **Start** menu.
2. Select the required Serial Port and then click **OK**.

Figure 1-21. Selection of Serial COM Port



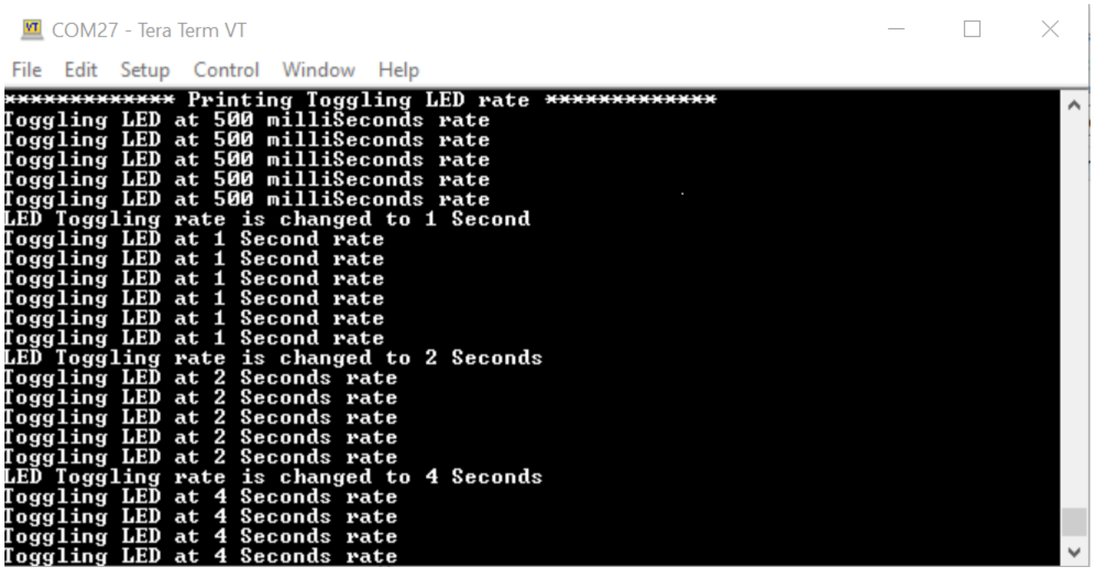
3. In the **Tera Term Serial port setup and connection** dialog box, in the **Speed** box type or select the baud rate to **115200**.

Figure 1-22. Setting the Baud Rate



4. An LED (LED0) on the PIC32CM JH01 Curiosity Pro Development Evaluation Kit toggles on a timeout basis and the default periodicity of the timeout is 500 milliseconds.
5. The LED toggling rate is displayed on the Serial Terminal.
6. Press the switch SW0 on the PIC32CM JH01 Curiosity Pro Evaluation Kit to change the default periodicity of the timeout to one second.
7. Every subsequent pressing of the switch SW0 on the PIC32CM JH01 Curiosity Pro Evaluation Kit changes the periodicity of the timeout to 2 seconds, 4 seconds, 500 milliseconds, and back to 1 second in cyclic order.

Figure 1-23. LED Toggling Rate on a Serial Terminal



While the LED toggling rate on the Serial Terminal changes with every subsequent switch press, observe the same change in the toggling rate of the LED0 on the evaluation kit.

2. Resources

- PIC32CM JH01 Curiosity Pro Evaluation Kit
- 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 to “Getting Started Application with PIC32CM JH01 Curiosity Pro Development Board” under the “Software” heading.
  www.microchip.com/en-us/development-tool/ev81x90a
- For additional info about 32-bit Microcontroller Collaterals and Solutions, refer to:
  ww1.microchip.com/downloads/aemDocuments/documents/MCU32/ProductDocuments/ReferenceManuals/32-bit-Microcontroller-Collateral-and-Solutions-Reference-Guide-DS70005534.pdf

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

Trademarks

ISBN: 978-1-6683-4212-1

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