Creating the First Application on PIC32CZ CAx Microcontrollers Using MPLAB Harmony v3 with MPLAB Code Configurator (MCC)

TB3348

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 explains how to create a simple application on an Arm® Cortex®-M7 based PIC32CZ CAx 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 on the serial console. For this demonstration, the following MPLAB Harmony v3 modules are used and configured using the MCC:

- PORT Pin to toggle LED0.
- Real-Time Clock (RTC) PLIB to periodically sample the LED0 toggling rate.
- External Interrupt Controller (EIC) PLIB to change the toggling rate when there is a switch press event.
- Serial Communication Interfaces ((SERCOM) configured as USART) and Direct Memory Access (DMA) PLIBs to print the LED0 toggling rate on a COM (serial) port terminal application running on a PC.
- Port Pins (USART pins as they are responsible for printing the data on the terminal after obtaining the values) to communicate with the serial terminal.

1. Creating First Application on the PIC32CZ CA90 MCU

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

- MPLAB X IDE v6.15
- MPLAB Code Configurator (MCC) Plug-in v5.4.1
- MPLAB XC32 Compiler v4.35
- MPLAB Harmony v3 repositories:
  – MPLAB Harmony v3 Chip Support Package (CSP) v3.18.1
  – MPLAB Harmony v3 dev_packs v3.18.1
- PIC32CZ CA90 Curiosity Ultra Development Board

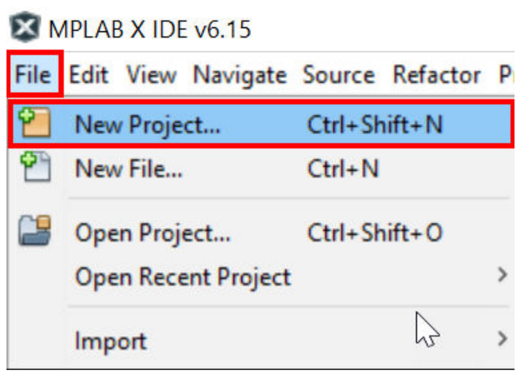
Note: Updated versions of the listed tools can also be used to create applications; users are not restricted to using the older versions.

1.1 Creating an MPLAB Harmony v3-Based Project

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

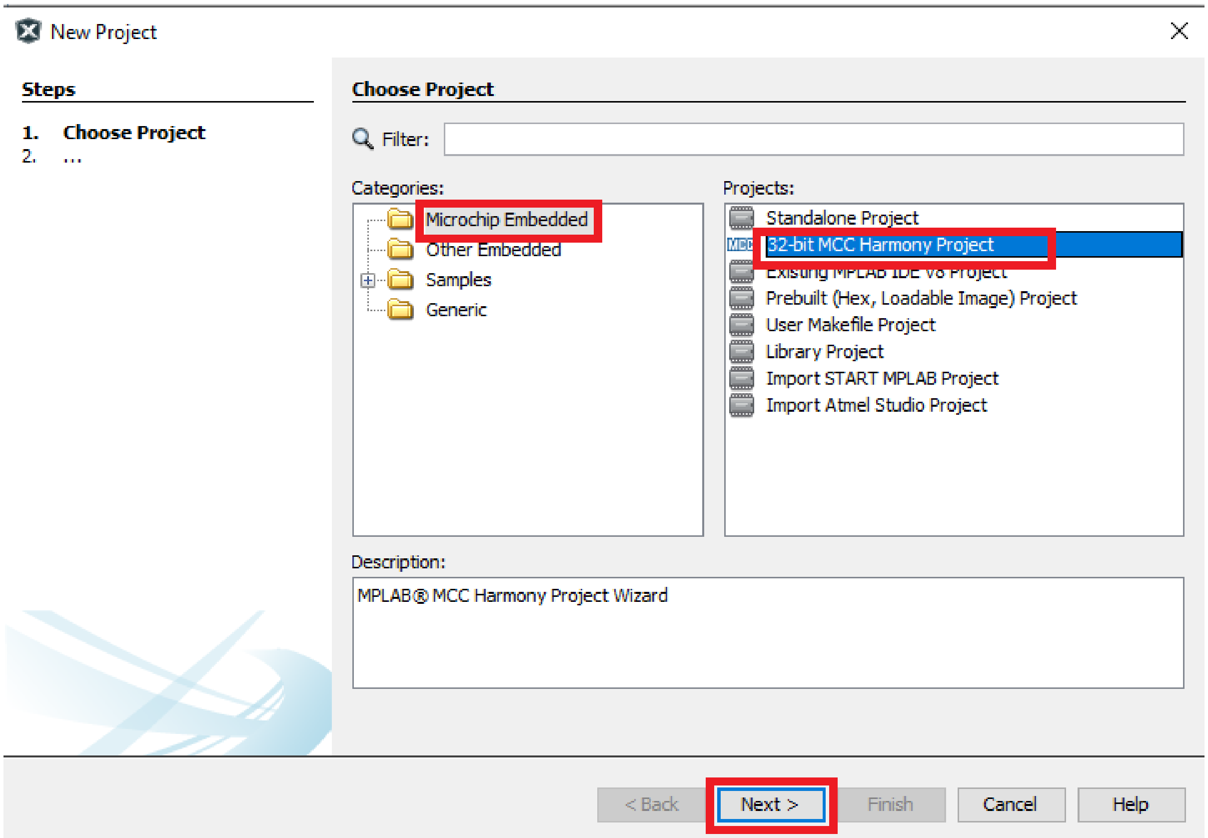
1. From the Start menu, launch MPLAB X IDE.
2. In MPLAB X IDE, from the File menu, select **New Project**.

Figure 1-1. New Project



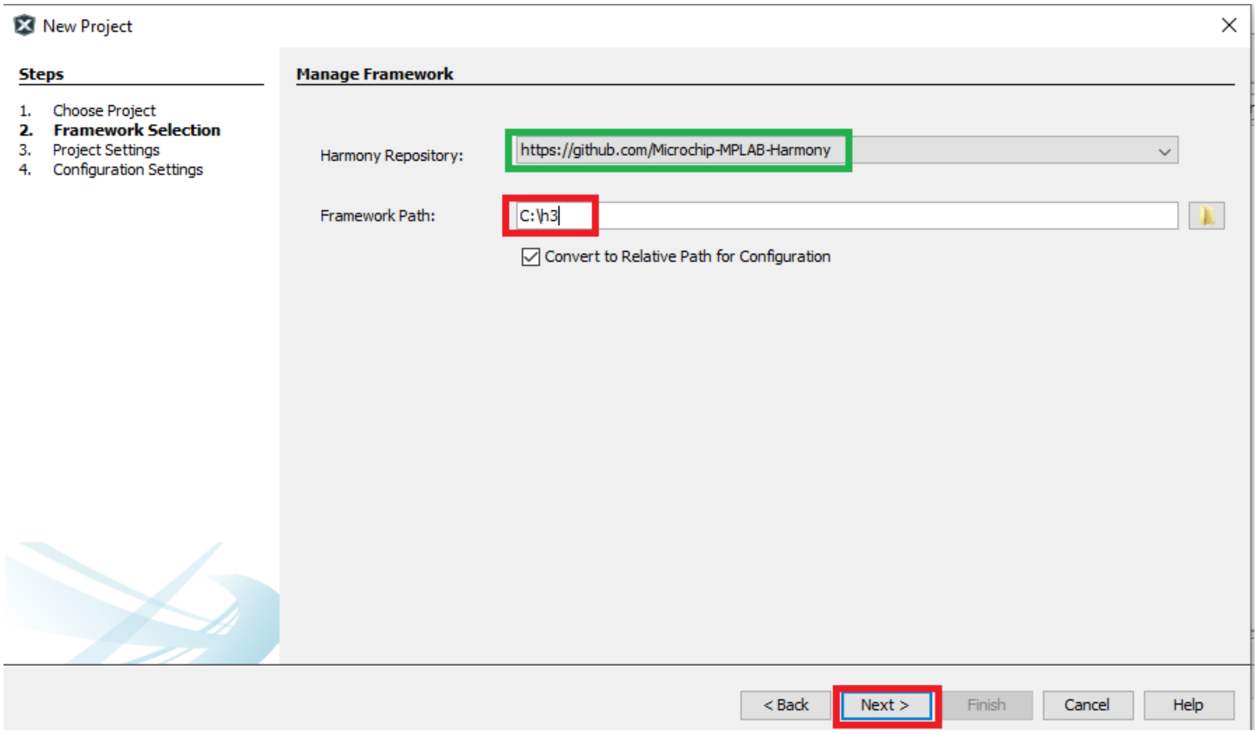
3. In the New Project window, under Steps navigation pane, select **Choose Project**. In the right Choose Project property section, under Categories, select **Microchip Embedded** and under Projects select **32-bit MCC Harmony Project**.
Note: If the 32-bit MCC Harmony Project is not available, before continuing with the demonstration install the MPLAB Code Configurator plug-in from the Tools Menu selecting *Plugins* > *Available Plugins*. For additional information, refer to the MPLAB Code Configurator overview at https://www.microchip.com/en-us/tools-resources/configure/mplab-code-configurator.

Figure 1-2. Creating an MPLAB Harmony v3-Based Project - Chose Project



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

Figure 1-3. Creating MPLAB Harmony v3-Based Project - Framework Selection

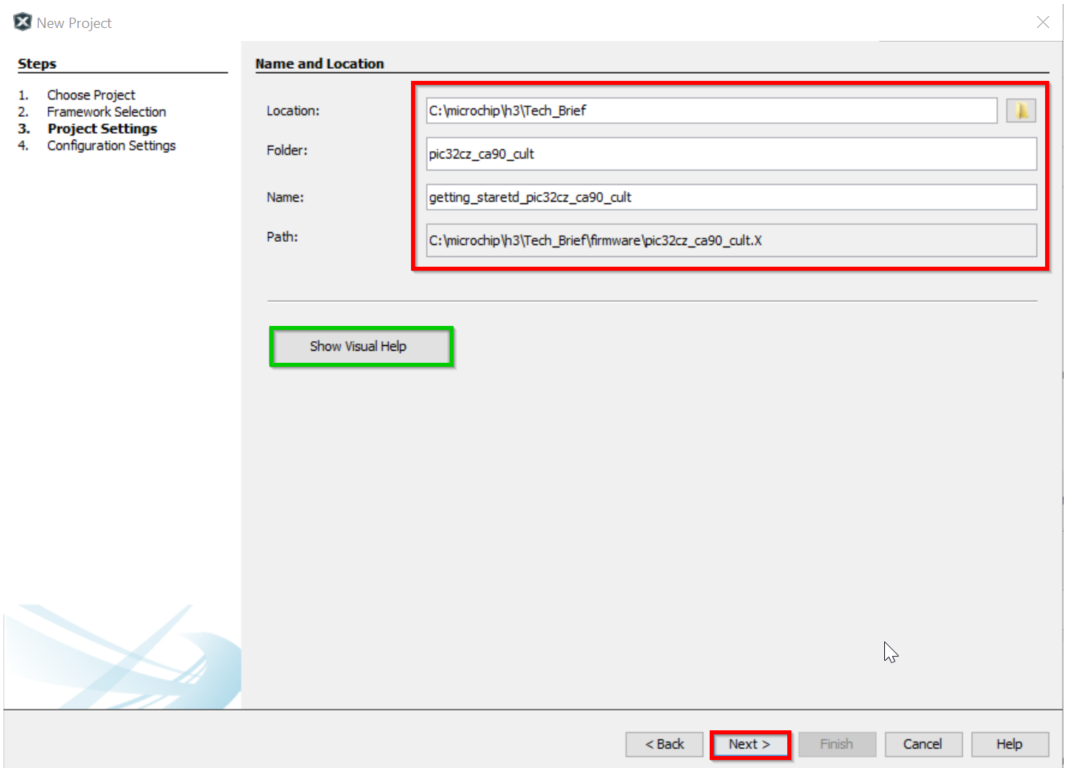


Note: For this demonstration application, the MPLAB Harmony v3 package, `csp`, is required. The MPLAB Harmony v3 Content Manager tool simplifies the downloading of the MPLAB Harmony v3 package, and if the package is not downloaded, use the MPLAB Harmony v3 Content Manager tool to download it.

6. Click **Next**.
7. Select **Project Settings** and in the Name and Location property section enter these details:
   a. Location: 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, for example: `C:\microchip\h3\Tech_Brief`.
   b. Folder: Indicates the name of the MPLAB X IDE folder. Enter pic32cz_ca90_cult to create the `pic32cz_ca90_cult.X` folder.
   c. Name: Indicates the name of the project that will be shown in MPLAB X IDE. Enter `getting_started_pic32cz_ca90_cult`.
   d. Path: This information will be updated automatically as and when users make changes to other fields.

Note: This project can also be created for the PIC32CZ CA80 Curiosity Ultra Development Board by following the same steps to create and configure the project.

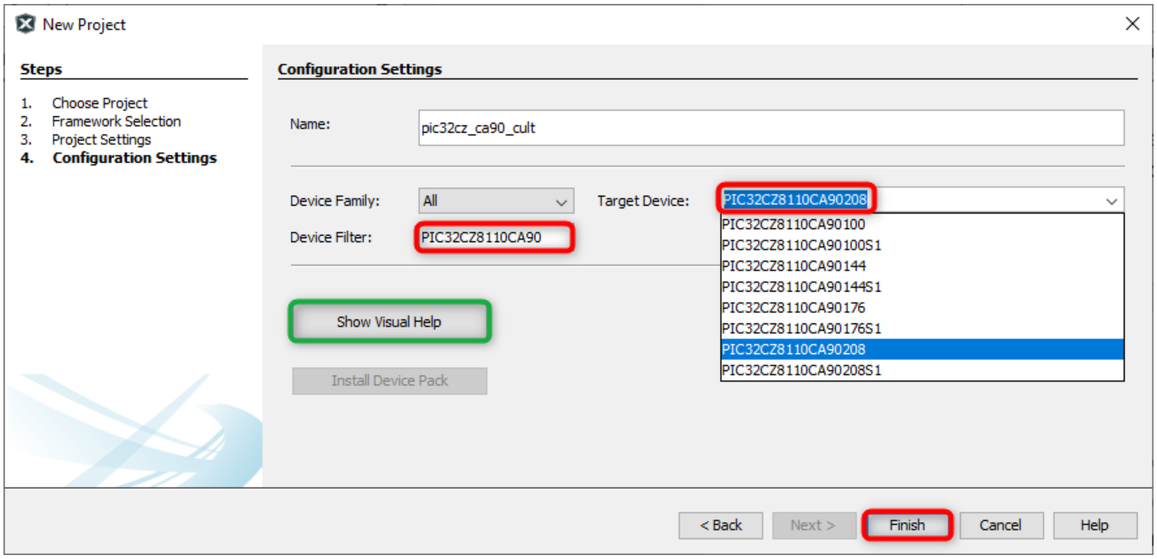
Figure 1-4. Creating an MPLAB Harmony v3-Based Project - Project Settings



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

8. Click **Next**.
9. Select **Configuration Settings** and in the Configuration Settings, enter these details:
   1. Name: Enter `pic32cz_ca90_cult`.
   2. Device Filter: Enter PIC32CZ8110CA90.
   3. Target Device: The Device Filter details will be refected here, select PIC32CZ8110CA90208 for creating the project on the PIC32CZ CA90 Curiosity Ultra Development Board.

Figure 1-5. Creating an MPLAB Harmony v3 Based Project - Configuration Settings

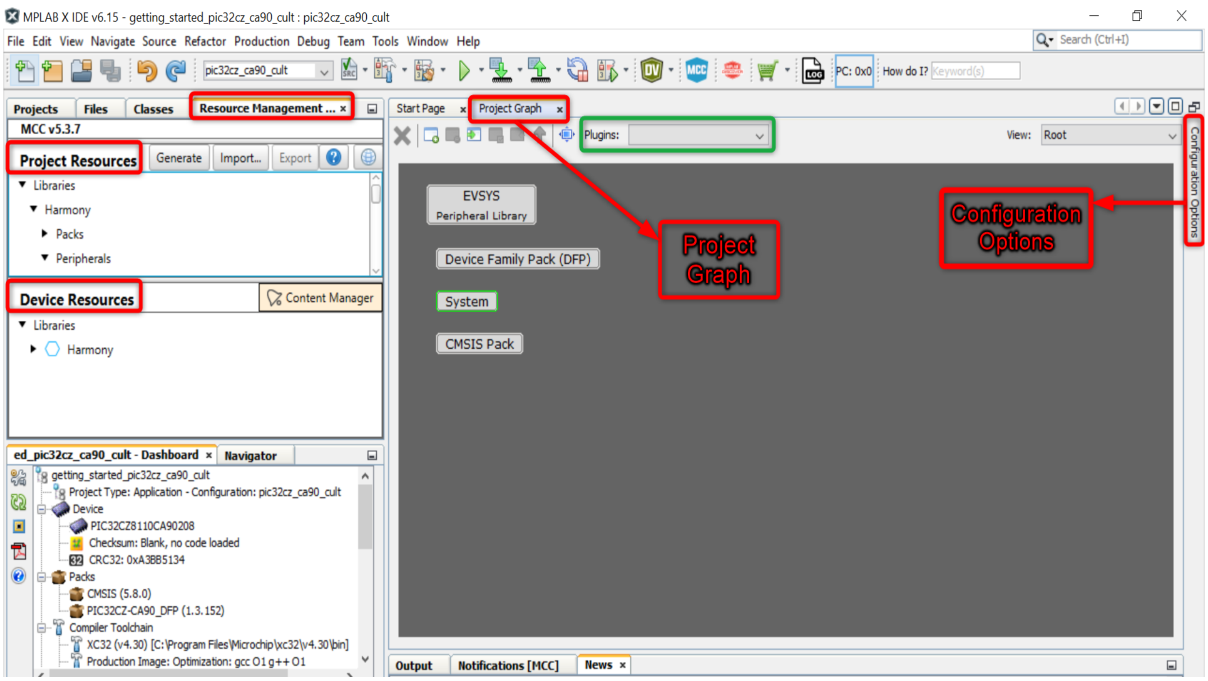


Note: Click the **Show Visual Help** button to open a contextual help window for a detailed description of various fields in the Configuration Settings.

Note: To create a project on the PIC32CZ CA90 Curiosity Ultra Development Board, in the Device Filter box enter PIC32CZ8110CA90208, and click **Finish**. The following configurations will work for both of the PIC32CZ CA90 Curiosity Ultra Development Board and PIC32CZ CA80 Curiosity Ultra Development Board. Users need to change the Target Device name to PIC32CZ8110CA80208 for the PIC32CZ CA80 Curiosity Ultra Development Board

10. Click **Finish** to launch MCC.
    Note: By default, the project will be set as the main project while launching MCC.
11. Before launching 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.
12. The MCC plugin will open in a new window.

Figure 1-6. MPLAB Code Configurator Window

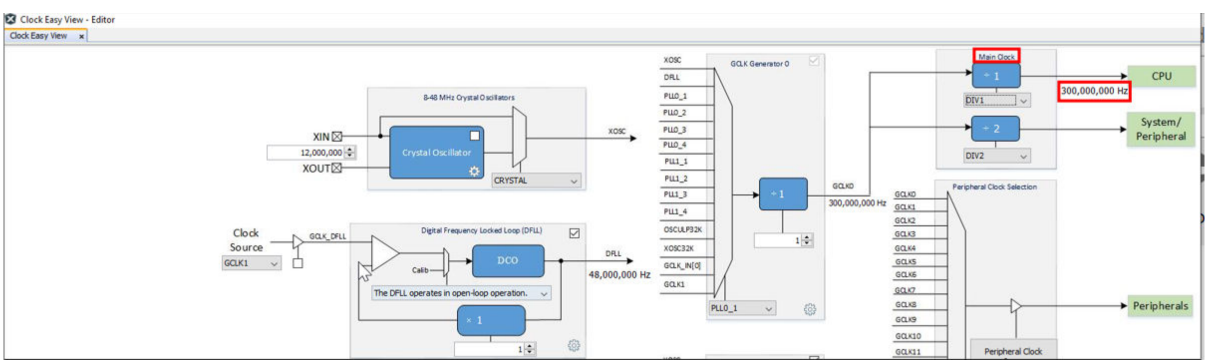


1.2 Adding and Configuring MPLAB Harmony v3 Components Using MCC

To add and configure MPLAB Harmony V3 components using MCC, follow these steps:

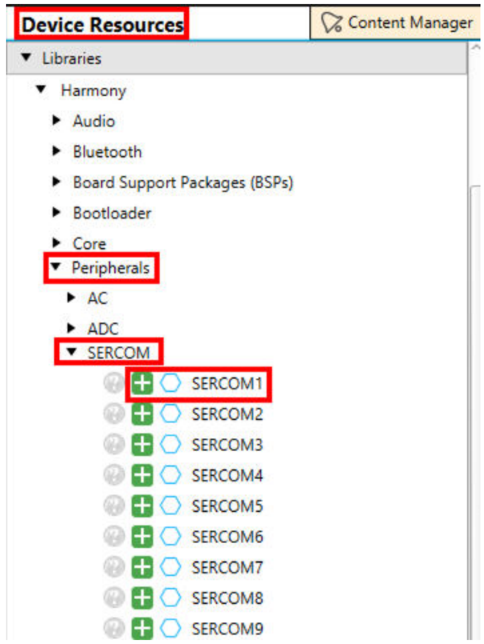
1. To launch Clock Easy View in MPLAB X IDE, click **Project Graph** and in the Plugins drop-down list, select **Clock Configuration**. The Clock Easy View will be displayed within the MCC window.
2. Scroll to the right and verify that the Main Clock is set to 300 MHz.

Figure 1-7. MPLAB Code Configurator - Clock Easy View window



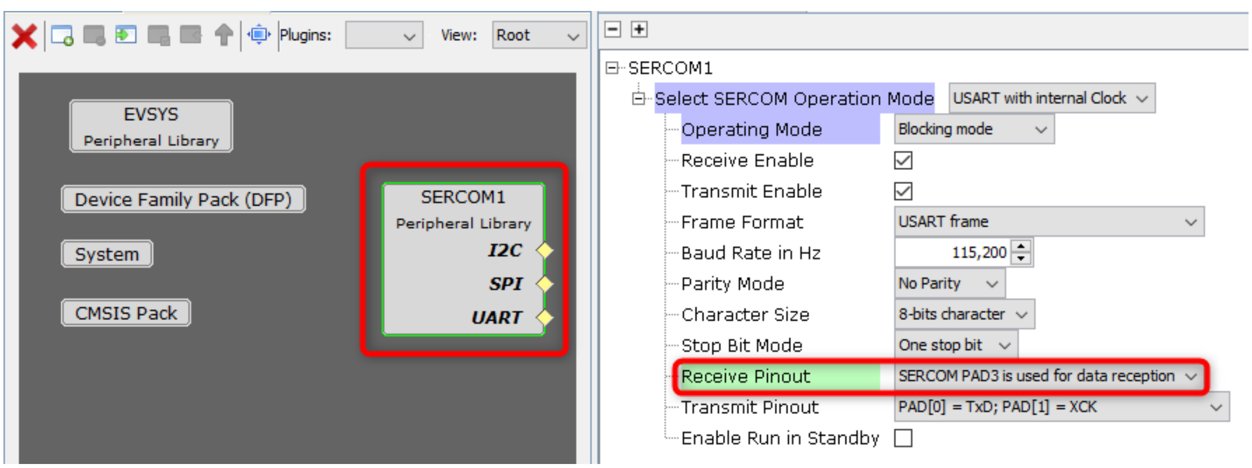
3. Under the left Device Resources navigation pane, click and expand *Libraries* > *Harmony* > *Peripherals* > *SERCOM* , and then click **SERCOM1**. Notice the SERCOM1 Peripheral Library is added in the Project Graph.

Figure 1-8. MPLAB Code Configurator - Device Resources



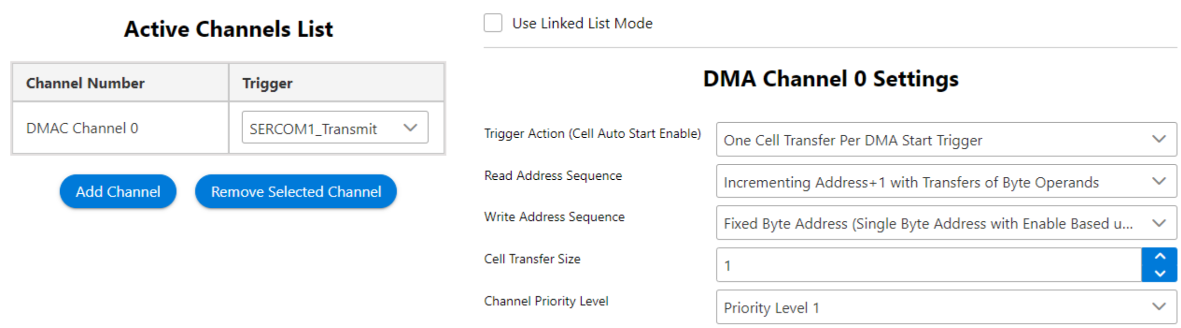
4. Select the **SERCOM1 Peripheral Library**, and in the Configuration Options property section configure to print the LED0 toggling rate on the Serial Console.
   a. To view the Receive Pinout, select **SERCOM PAD[3]** from the drop-down list and retain the remaining parameters with default setting as shown in the figure below:

Figure 1-9. MPLAB Code Configurator - SERCOM1 Configuration



5. From the Plugins drop-down list select **DMA Configuration**. 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.

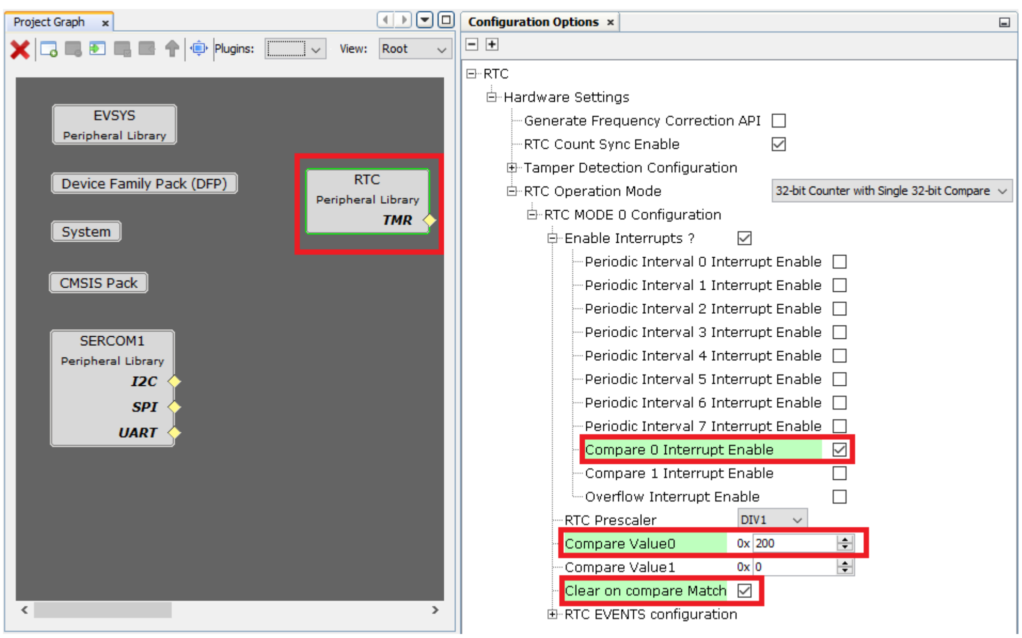
Figure 1-10. MPLAB Code Configurator - DMA Configuration



Note: The SERCOM1 and DMA are configured to obtain the LED0 toggling rate from the application and to print the LED0 toggling rate on a Serial Console running on a PC.

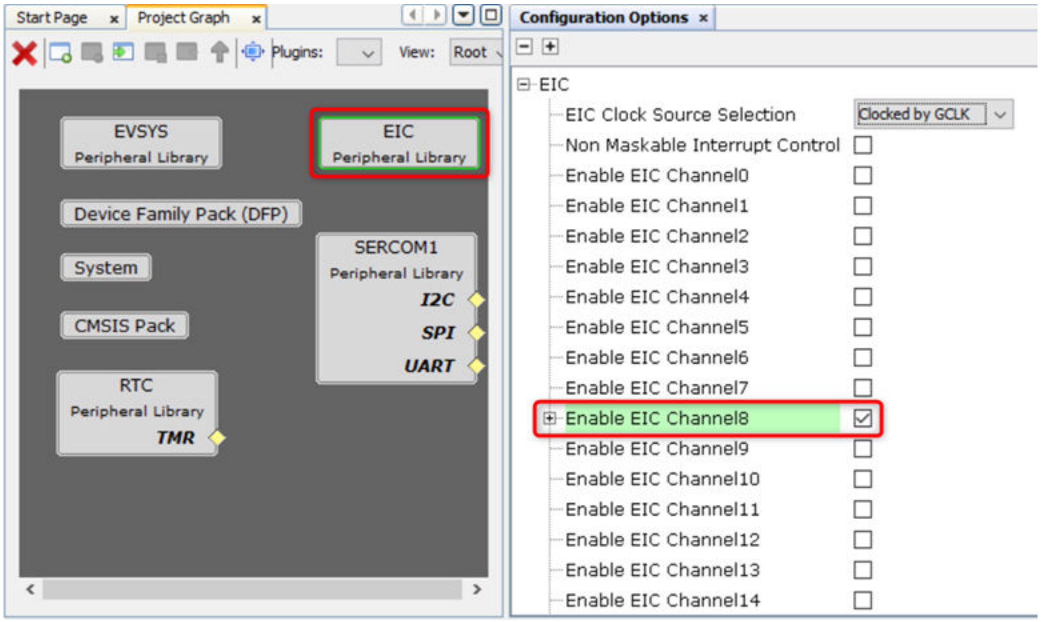
6. Under left **Device Resources** navigation pane, click and expand *Libraries* > *Harmony* > *Peripherals* and then click **RTC**. Notice the RTC Peripheral Library block is added in the Project Graph.
7. Select the **RTC Peripheral Library**, and in the Configuration Options property section configure as shown in the following figure (i.e., to generate a compare interrupt every 500 milliseconds).

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



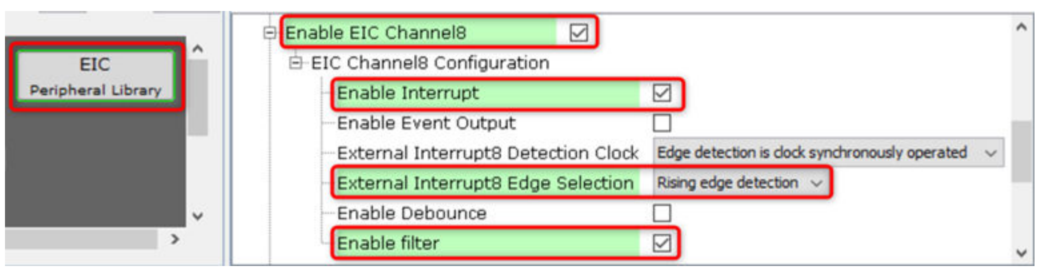
8. Under **Device Resources**, click and expand *Libraries* > *Harmony* > *Peripherals* and then click **EIC.** Notice the EIC Peripheral Library block is added in the Project Graph.
9. Select the **EIC Peripheral Library**, and in the Configuration Options property section configure as shown in the following figure.

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



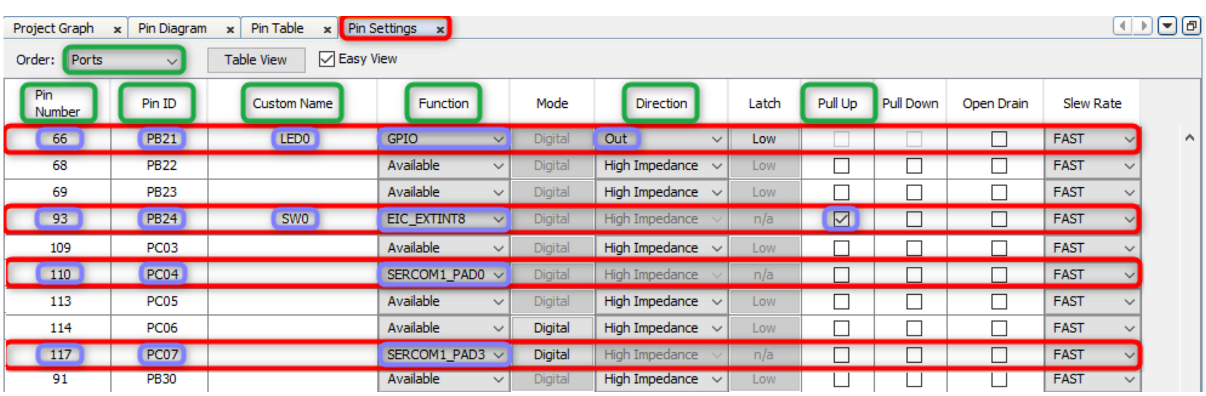
a. Select and expand the **Enable EIC Channel8** and make the following configurations.

Figure 1-13. MPLAB Code Configurator - EIC PLIB Configuration Channel 8



10. From the Plugins drop-down list, select **Pin Configuration**, and then click **Pin Settings**.
11. In the Order box, select **Ports**, and then configure according to the application requirements.
12. Change the Custom Name of these two pins: Pin ID PB21 and PB24. Refer to the Pin Configuration image below for configuring GPIO, SERCOM1, and EIC_EXTINT8.

Figure 1-14. MCC Pin Configuration



Project Graph | Pin Diagram | Pin Table | Pin Settings

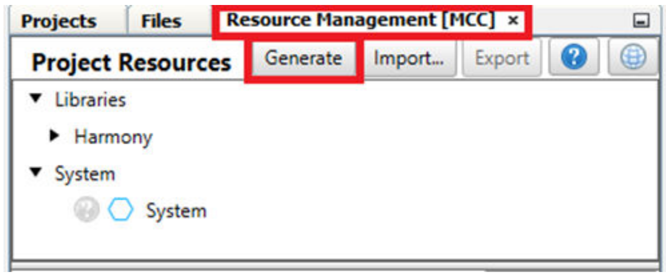
Order: Ports | Table View | Easy View

Pin Number	Pin ID	Custom Name	Function	Mode	Direction	Latch	Pull Up	Pull Down	Open Drain	Slew Rate
66	PB21	LED0	GPIO	Digital	Out	Low				FAST
68	PB22		Available	Digital	High Impedance	Low				FAST
69	PB23		Available	Digital	High Impedance	Low				FAST
93	PB24	SW0	EIC_EXTINT8	Digital	High Impedance	n/a	☑			FAST
109	PC03		Available	Digital	High Impedance	Low				FAST
110	PC04		SERCOM1_PAD0	Digital	High Impedance	n/a				FAST
113	PC05		Available	Digital	High Impedance	Low				FAST
114	PC06		Available	Digital	High Impedance	Low				FAST
117	PC07		SERCOM1_PAD3	Digital	High Impedance	n/a				FAST
91	PB30		Available	Digital	High Impedance	Low				FAST

1.3 Generating The Code

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

Figure 1-15. Generation of Code



Note: The generated code will add files and folders to the 32-bit MCC Harmony project. In the generated code, notice the Peripheral Library files that are generated for the Real-Time Clock (RTC), External Interrupt Controller (EIC), PORT, SERCOM1 (as USART), and DMA peripherals are added in the project. The MCC also generates the `main.c` files in the project.

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.

1.4 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 initialization of the buffer and application logic.

```
uint8_t uartLocalTxBuffer[100] = {0};
    RTC_Timer32CallbackRegister(rtcEventHandler, 0);
    EIC_CallbackRegister(EIC_PIN_8, SW0_eventHandler, 0);
    DMA_ChannelCallbackRegister(DMA_CHANNEL_0, UARTDmaChannelHandler, 0);
```

Figure 1-16. Adding Application Logic to Register Callback Event Handlers

a. In the `main()` function add the following code to register callback event handlers. Add the line of code to call the `RTC_Timer32Start()` function after registering the callback event handlers.

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

```
static void SW0_eventHandler(uintptr_t context)
{
    changeTempSamplingRate = true;
}
```

```
static void rtcEventHandler (RTC_TIMER32_INT_MASK intCause, uintptr_t context)
{
    if (intCause & RTC_TIMER32_INT_MASK_CMP0)
    {
        isRTCTimerExpired = true;
    }
}

static void UARTDmaChannelHandler(DMA_TRANSFER_EVENT event, uintptr_t contextHandle)
{
    if (event == DMA_TRANSFER_EVENT_BLOCK_TRANSFER_COMPLETE)
    {
        isUARTTxComplete = true;
    }
}
```

2. Declare the pre-processor directives, macros, local variables, and flags.

```
#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
#include "device_cache.h"

/* 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

static volatile bool isRTCTimerExpired = false;
static volatile bool changeTempSamplingRate = false;
static volatile bool isUARTTxComplete = true;
static uint8_t __attribute__ ((aligned (16))) uartTxBuffer[100] = {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"};
```

3. Add the application logic when both flags have detected the completion of the event i.e., `(isRTCExpired)` and `(isUSARTTxComplete)` in the `while()` super loop.

```
       while ( true )
    {
        if ((isRTCTimerExpired == true) && (true == isUARTTxComplete))
            {
                isRTCTimerExpired = false;
                isUARTTxComplete = false;
                LED0_Toggle();
                sprintf((char*)(uartTxBuffer), "Toggling LED at %s rate 
\r\n",
                &timeouts[(uint8_t)tempSampleRate][0]);
                DCACHE_CLEAN_BY_ADDR(uartTxBuffer, sizeof(uartTxBuffer));
                DMA_ChannelTransfer(DMA_CHANNEL_0, uartTxBuffer, \
                    (const void *)&(SERCOM1_REGS->USART_INT.SERCOM_DATA), \
                    strlen((const char*)uartTxBuffer));
             }
```

Figure 1-17. Add the Application Logic

```
while ( true )
{
    if ((isRTCTimerExpired == true) && (true == isUARTTxComplete))
        {
            isRTCTimerExpired = false;
            isUARTTxComplete = false;
            LED0_Toggle();
            sprintf((char*)(uartTxBuffer), "Toggling LED at %s rate \r\n", &timeouts[(uint8_t)tempSampleRate][0]);
            DCACHE_CLEAN_BY_ADDR(uartTxBuffer, sizeof(uartTxBuffer));
            DMA_ChannelTransfer(DMA_CHANNEL_0, uartTxBuffer, \
                (const void *)&(SERCOM1_REGS->USART_INT.SERCOM_DATA), \
                strlen((const char*)uartTxBuffer));
        }
```

4. Add the application logic for toggling the LED0 at different rates of 500 ms, 1s, 2s, and 4s whenever there is a switch press on the board by the user, and DMA transfer logic is in the `main()` function of the `main.c` file.

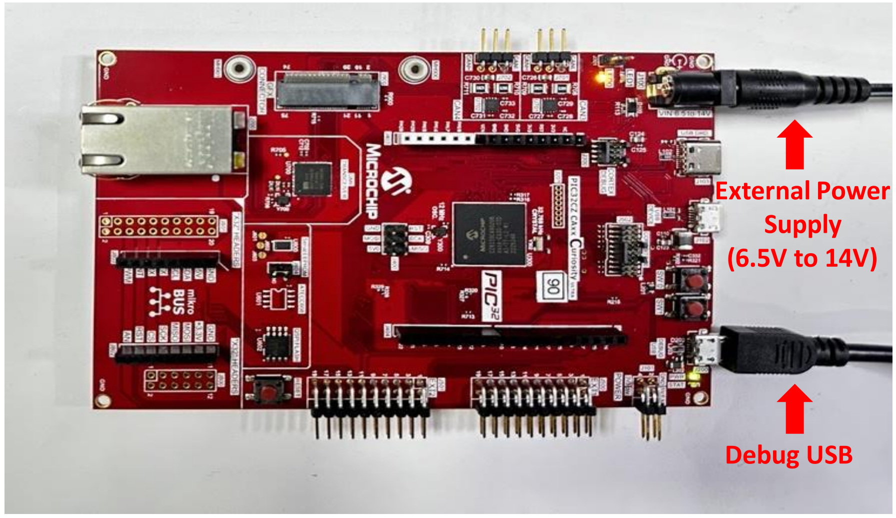
```
        /* Maintain state machines of all polled MPLAB Harmony modules. */
             if(changeTempSamplingRate == true)
            {
                changeTempSamplingRate = false;
                if(tempSampleRate == TEMP_SAMPLING_RATE_500MS)
                {
                    tempSampleRate = 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]);
                DCACHE_CLEAN_BY_ADDR(uartLocalTxBuffer, sizeof(uartLocalTxBuffer));
                DMA_ChannelTransfer(DMA_CHANNEL_0, uartLocalTxBuffer, \
                    (const void *)&(SERCOM1_REGS->USART_INT.SERCOM_DATA), \
                    strlen((const char*)uartLocalTxBuffer));
            }

    }
    /* Execution should not come here during normal operation */

    return ( EXIT_FAILURE );
}
```

1.5 Building and Programing The Application

To build and program the applicatin, follow these steps:

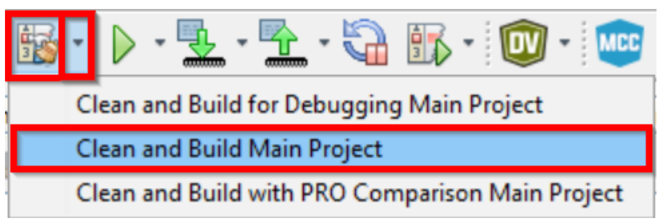
1. The PIC32CZ CA90 Curiosity Ultra Development Board supports an on-board debugger (PKoB4) for debugging. Connect the Type-A male to micro-B USB cable to the micro-B DEBUG USB port to power and debug the PIC32CZ CA90 Curiosity Ultra Development Board. Connect the External Power Supply (6.5v to 14v) to the J100 connector.

Figure 1-18. Hardware Setup



2. To set the project as the main project, from Project Properties select the latest compiler version (v4.35).
3. To clean and build the project, click on the highlited icon or from the drop-down item list select **Clean and Build Main Project**.

Figure 1-19. Clean and Build



4. Program the application by clicking the highlighted icon below.

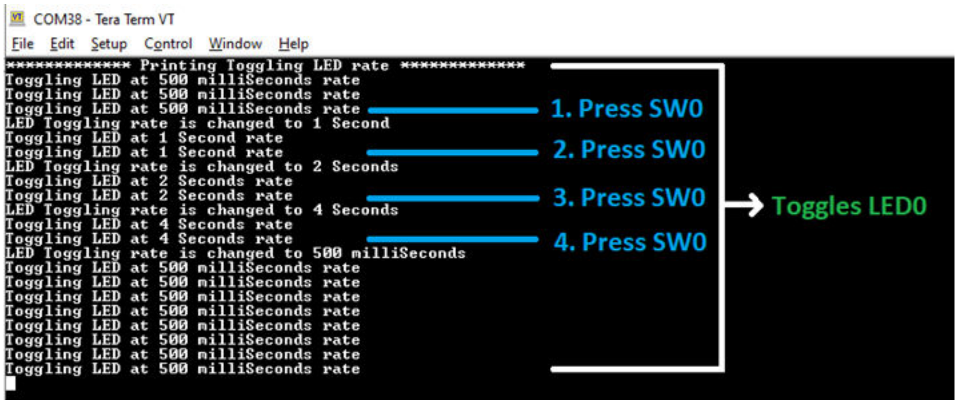
Figure 1-20. Program the Device

1.6 Observing the Output on the Board and Serial Terminal

To observe the output on the board and Serial Terminal, follow these steps:

1. After building and programing the application, open the Tera Term tool on the PC.
2. Select the Serial Port and set the baud rate as 115200.
3. Press the Reset button on the PIC32CZ CA90 Curiosity Ultra Development Board. The LED0 will toggle at 500 ms by default, and with every subsequent SW0 switch press the LED0 toggling rate will change to 1s, 2s, and 4s.

Figure 1-21. LED0 Toggling Rate on the Serial Terminal



While the LED0 toggling rate on the Serial Terminal changes with every subsequent switch press, observe the corresponding changes in the toggling rate of the LED0 on the development 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/xwiki/bin/view/software-tools/harmony/
- For more information on various applications, refer to: github.com/Microchip-MPLAB-Harmony/reference_apps
- The PIC32CZ CA90 Curiosity Ultra Evaluation Board details can be found here: www.microchip.com/en-us/development-tool/EA58X56A
- For the example application, refer to “Getting Started Extended Application on PIC32CZ CA90 Curiosity Ultra Development Board” under “Software” https://www.microchip.com/en-us/development-tool/EA58X56A
- 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

3. Revision History

Revision A - January 2024

This is the initial released version 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.

ISBN: 978-1-6683-3845-2

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			**UK - Wokingham**
Tel: 408-735-9110			Tel: 44-118-921-5800
Tel: 408-436-4270			Fax: 44-118-921-5820
Canada - Toronto			
Tel: 905-695-1980			
Fax: 905-695-2078			