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

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 graphical user interface (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 an Arm® Cortex®-M23 based PIC32CM LSx Microcontroller using the MCC with MPLAB Harmony v3 modules. This application demonstrates the TrustZone®-based security feature on PIC32CM LSx Microcontrollers. The application consists of two projects which details about Secure and Non-Secure modes on PIC32CM LSx Microcontrollers. These two projects offer security isolation between the trusted and non-trusted resources in the device. 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 as Secure and Non-Secure.

The Secure modules include:

- Secure PORT Pin to toggle LED (by default, all the PORT pins are configured as Secure).
- Secure Real-Time Clock (RTC) PLIB to periodically sample LED toggling rate.
- Secure External Interrupt Controller (EIC) PLIB to change the toggling rate when a switch press event occurs.

The Non-Secure modules include:

- Non-Secure SERCOM (configured as USART) and Non-Secure Direct Memory Access (DMA) PLIBs to print the LED toggling rate on a COM (serial) port terminal application running on a PC.
- Non-Secure Port Pins (USART pins are responsible for printing the data on the Terminal after obtaining the values from Secure Application) to communicate with the Serial Terminal.

Table of Contents

1. Creating the First Trust Zone Application on PIC32CM LS60 MCU

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

- MPLAB X IDE v6.20
- MCC Plug-in v5.5.0
- MPLAB XC32 v4.35
- MPLAB Harmony v3 repository: csp v3.18.2
- PIC32CM LS60 Curiosity Pro Evaluation Kit

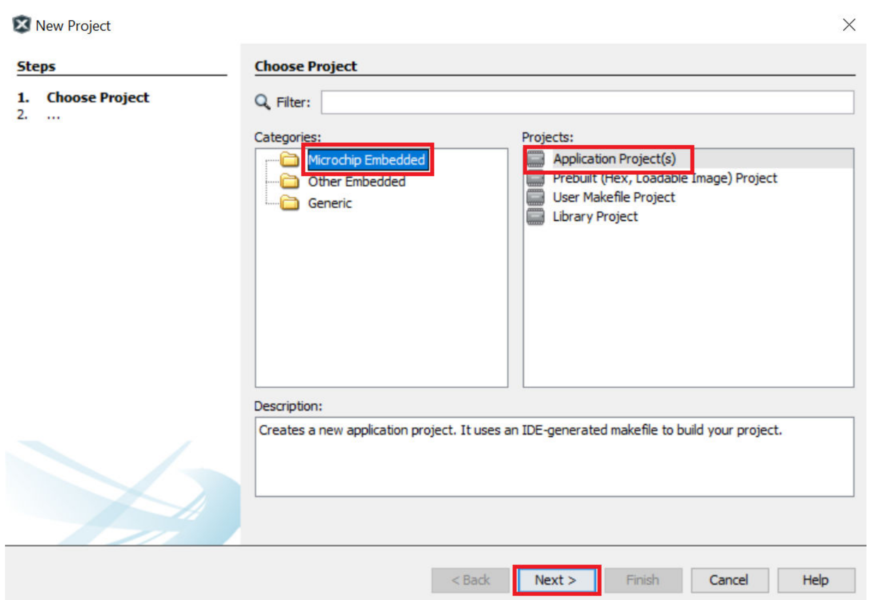
Note: Updated versions of the above listed hardware tools can also be used to create the application.

1.1 Creating an MPLAB Harmony v3-based Project

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

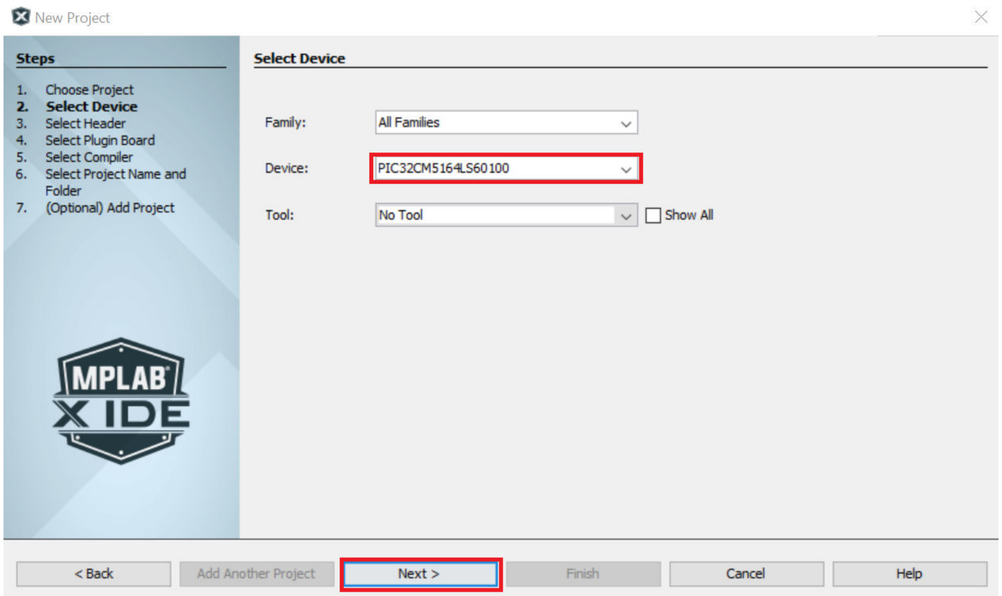
1. On the Start menu, launch MPLAB X IDE.
2. In MPLAB X IDE, on the **File** menu, click **New Project** or click on the *New Project* icon.
3. In the New Project window, in the left navigation bar, under Steps click **Choose Project**.
4. In the right Choose Project property page:
   a. Categories: Select **Microchip Embedded**.
   b. Projects: Select **Application Projects**.

Figure 1-1. Choose Project



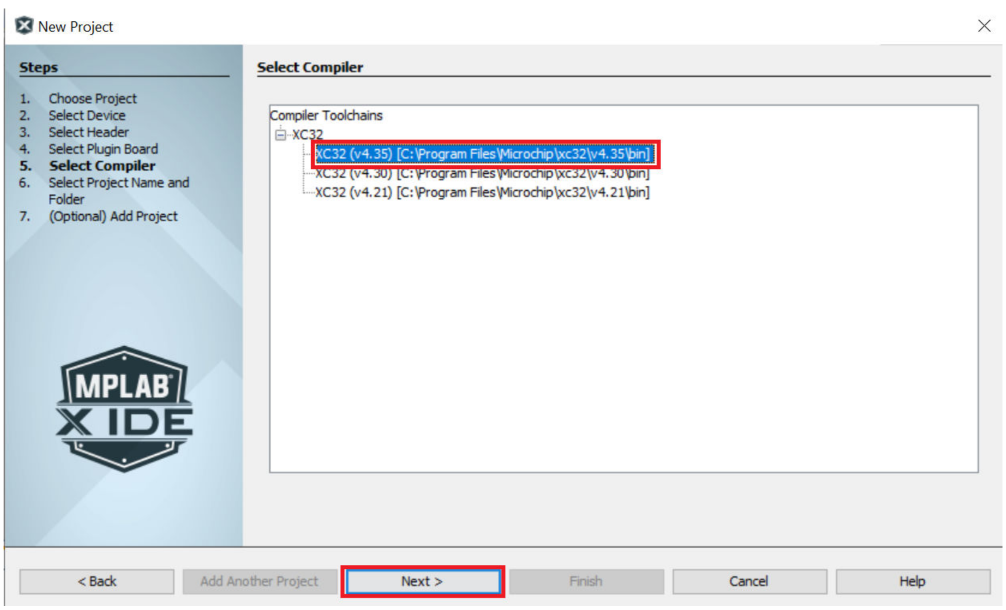
5. Click **Next.**
6. In the left navigation bar, click **Select Device**.
7. In the Select Device property page, in the Device box, type or select the device PIC32CM5164LS60100.

Figure 1-2. Select Device



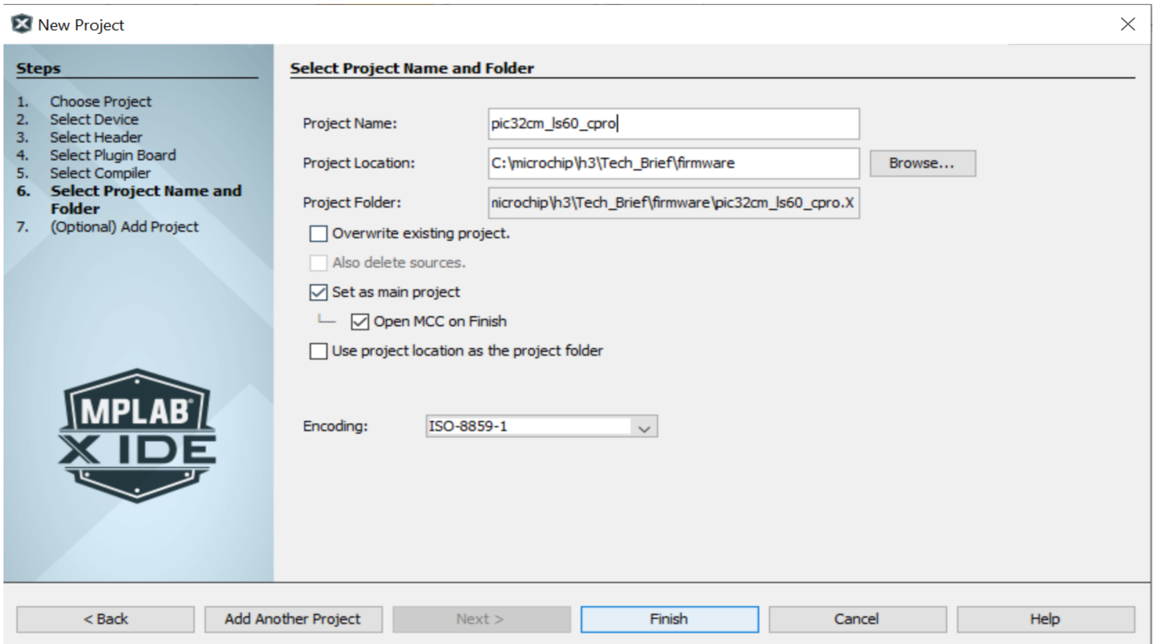
8. Click **Next**.
9. In the left navigation bar, click **Select Compiler**.
10. In the Select Compiler property page, click and expand XC32 list of options and then select the Compiler Toolchain as shown below.

Figure 1-3. Select Compiler



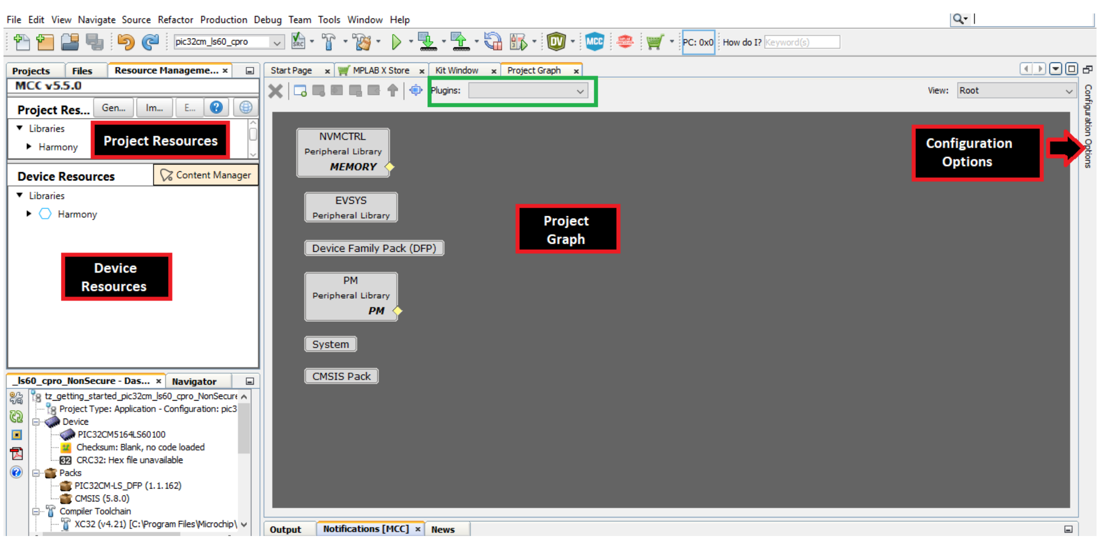
11. Click **Next**.
12. In the left navigation bar, click **Select Project Name and Folder**.
13. In the right Select Project Name and Folder property page:
    - Project Name: Enter *pic32cm_ls60_cpro*.
    - Project Location: Click the **Browse** button and choose *C:\microchip\h3\Tech_Brief\firmware*.

Figure 1-4. Select Project Name and Folder



14. Click **Finish** to launch MCC.
    Note: By default, Non-Secure project will be set as the main project while launching MCC.
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



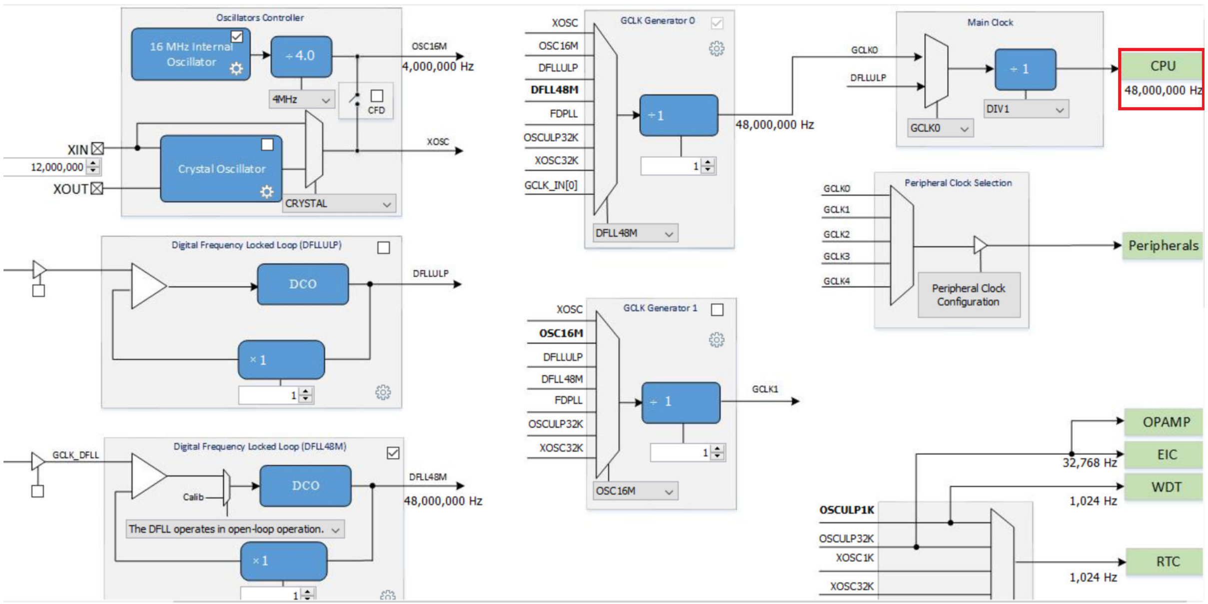
1.2 Adding and Configuring the MPLAB Harmony Components

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

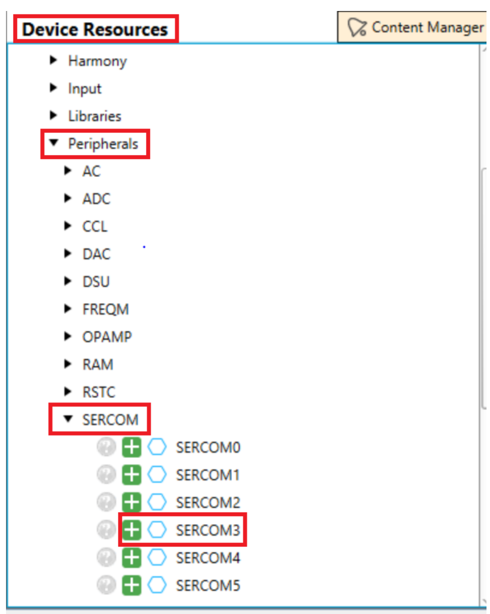
1. In the MCC window, click **Project Graph**.
2. In the **Plugins** drop-down list (highlighted green in the Figure 1.5), select **Clock Configuration** .

3. This opens the Clock Easy View window, which embedded within MCC. Verify that the Main Clock is set to 48 MHz.

Figure 1-6. Clock Easy View Window



4. In MCC, under the Device Resources section, click and expand the list of options *Harmony* > *Peripherals* > *SERCOM* .
5. Click **SERCOM3** to add. Observe that the *SERCOM3* Peripheral Library block is added in the Project Graph window.

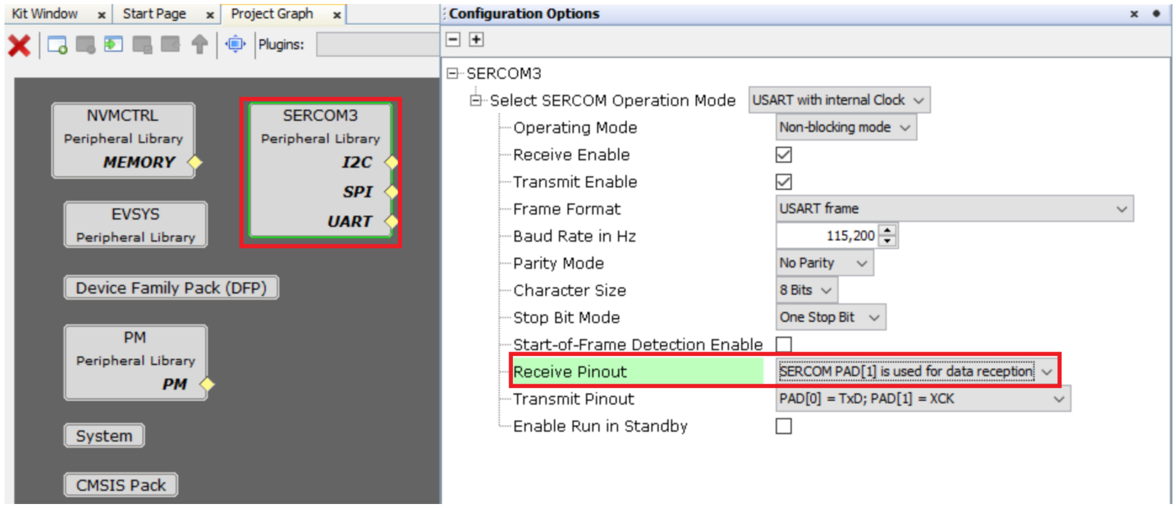


Note: Similarly users can select and add all peripherals, available under *Device Resources* > *Harmony* > *Peripherals*.

6. In the Project Graph window, in the left navigation bar, select **SERCOM3 Peripheral Library**.
7. In the right Configuration Options property page, configure it as follows to print the LED toggling rate on the Serial Console.

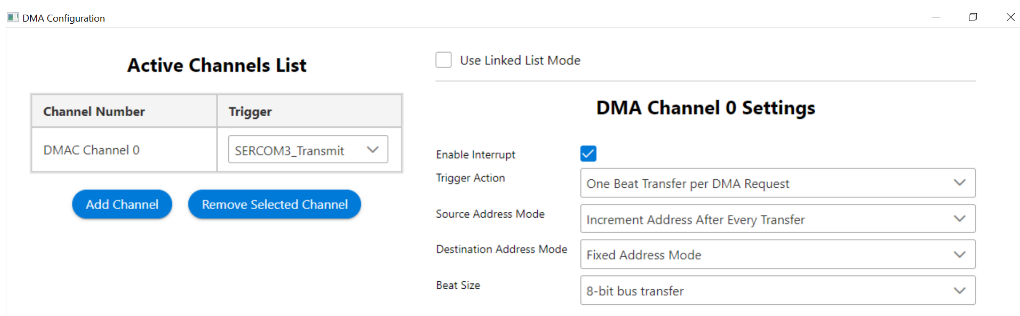
a. For Receive Pinout, from the drop-down list, select SERCOM PAD[1] and leave the remaining parameters with the default setting as shown in the figure below.

Figure 1-7. MPLAB Code Configurator – SERCOM3 Configuration Window



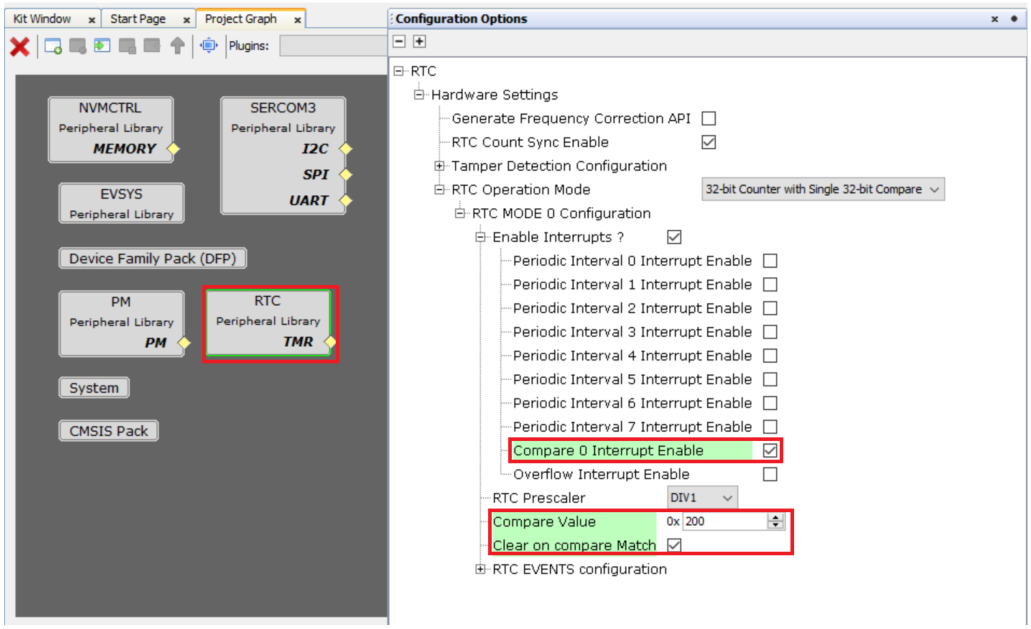
8. From the **Plugins** item list, select **DMA Configuration**,
9. In the DMA Configuratin dialogue box, click **Add Channel** and configure DMA Channel 0 to transmit application buffer to the USART TX register. The DMA transfers one byte from the user buffer to the USART transmit buffer on each trigger.

Figure 1-8. DMA Configuration



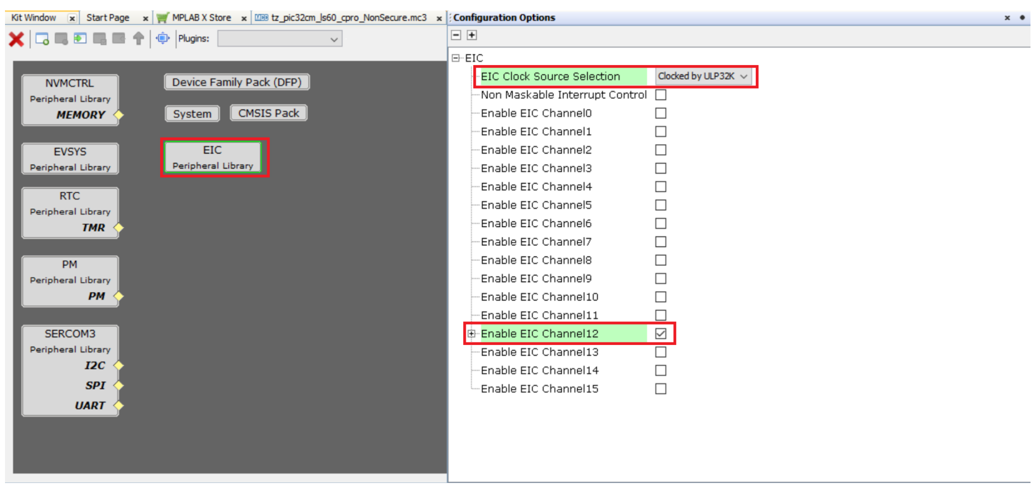
10. Under Device Resources, click and expand *Harmony* > *Peripherals* > *RTC*. Click on the **RTC** to add, and then observe that the *RTC Peripheral Library* block is added in the Project Graph window to generate a compare interrupt every 500 milliseconds.

Figure 1-9. RTC PLIB Configuration



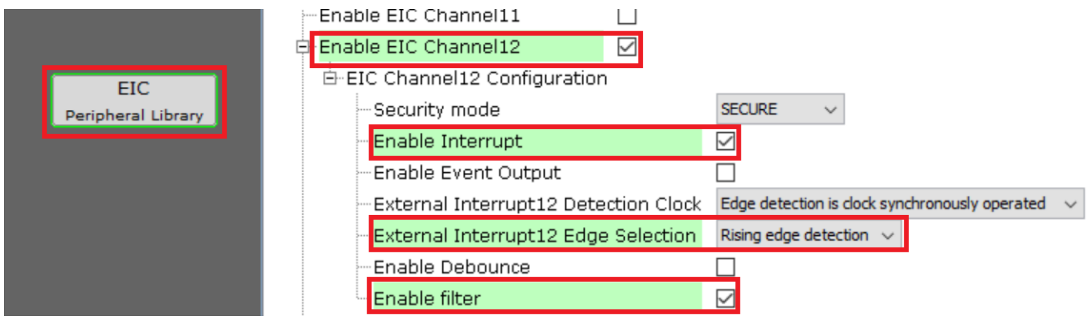
11. Under the Device Resources section, click and expand *Harmony* > *Peripherals* > *EIC.*
12. Click on the **EIC** to add, and then observe that the *EIC Peripheral Library* block is added in the Project Graph window.

Figure 1-10. Enable EIC Channel 12



13. Click on the **EIC Peripheral Library**, and then expand *Enable EIC Channel 12* and configure by selecting the options as shown below.

Figure 1-11. MPLAB Code Configurator – EIC PLIB Configuration



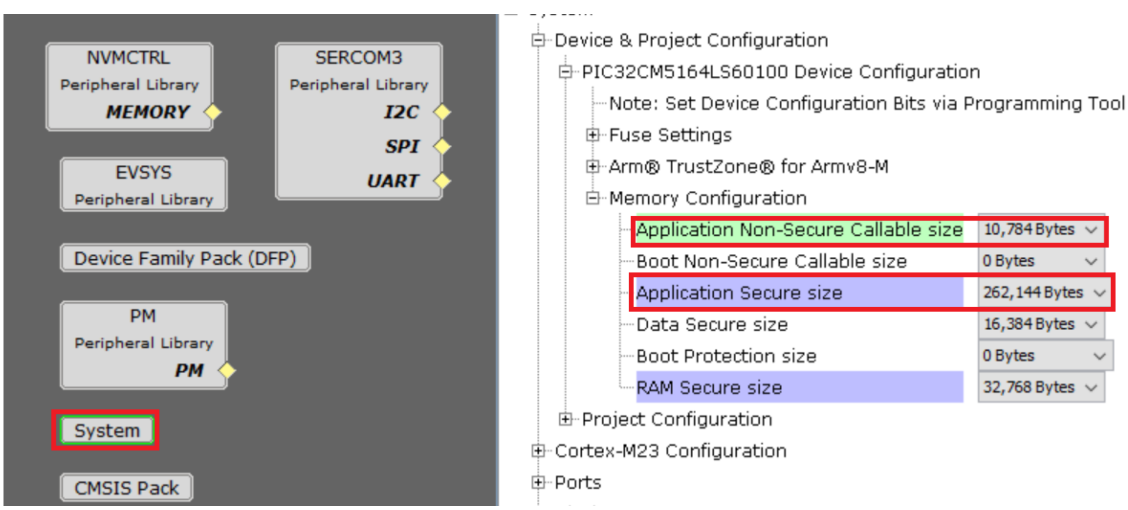
14. Select **Arm TrustZone for Armv8-M** from the **Plugins** list, and then check *Memory Configuration for Secure and Non-Secure* regions of the application.

Figure 1-12. Arm TrustZone Memory Configuration Window

Note: The markers can be used to configure the memory if there are any changes needed in the memory configuration. It is recommended not to change the default configuration for this application.

The memory can be configured using a Tree view interface by selecting **System** and following the highlighted options in the below screenshot.

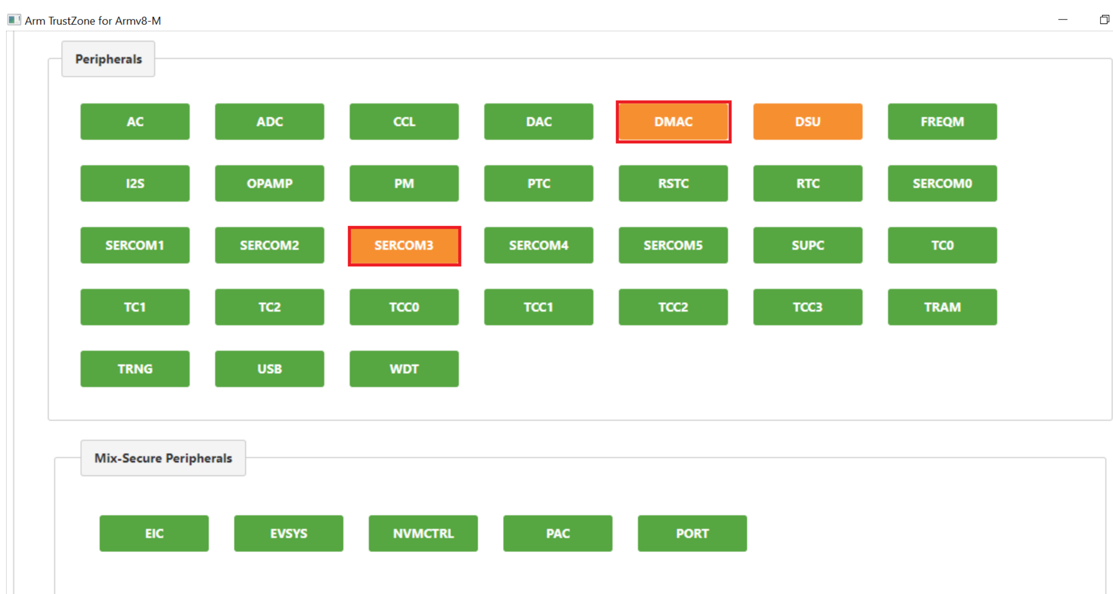
Figure 1-13. Memory Configuration using System



The Application Non-Secure Callable (ANSC) memory size has been increased to accommodate memory space if users required to add code in future.

15. Under **plugin** list select **Arm TrustZone for Armv8-M**, and then open *Peripheral Configuration* and select **SERCOM3** and **DMAC** boxes as Non-Secure peripherals. Upon selection the box color will change from green to orange.

Figure 1-14. SERCOM3 and DMAC Peripheral Configuration



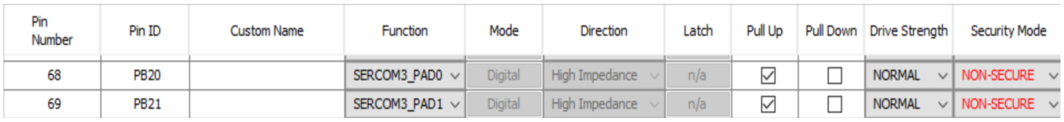
Note: The SERCOM3 and DMA peripheral libraries are configured as Non-Secure peripherals. These libraries obtain the LED toggling rate from the Secure application through NSCs (Non-Secure Callable) APIs to print the LED toggling rate on a Serial Console running on a PC.

16. Open NVIC Configuration from **Plugins** list and make DMAC_0 Channel as Non-Secure.

Figure 1-15. NVIC Configuration (DMAC_0 as Non-Secure)

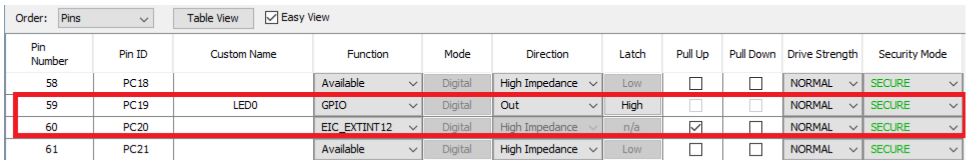
17. Open the Pin Configuration window from the **Plugins** list and configure the required pins as shown below:

Figure 1-16. SERCOM3 Pin Configuration



Pin Number	Pin ID	Custom Name	Function	Mode	Direction	Latch	Pull Up	Pull Down	Drive Strength	Security Mode
68	PB20		SERCOM3_PAD0	Digital	High Impedance	n/a	☑	☐	NORMAL	NON-SECURE
69	PB21		SERCOM3_PAD1	Digital	High Impedance	n/a	☑	☐	NORMAL	NON-SECURE

Figure 1-17. Switch and LED Pin Configuration



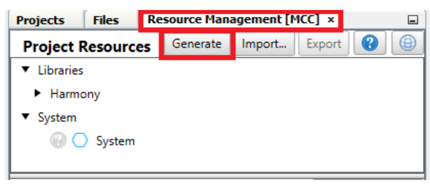
Order: Pins | Table View | ☑ Easy View

Pin Number	Pin ID	Custom Name	Function	Mode	Direction	Latch	Pull Up	Pull Down	Drive Strength	Security Mode
58	PC18		Available	Digital	High Impedance	Low	☐	☐	NORMAL	SECURE
59	PC19	LED0	GPIO	Digital	Out	High	☐	☐	NORMAL	SECURE
60	PC20		EIC_EXTINT12	Digital	High Impedance	n/a	☑	☐	NORMAL	SECURE
61	PC21		Available	Digital	High Impedance	Low	☐	☐	NORMAL	SECURE

1.3 Generating the Code

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

Figure 1-18. 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 generated for Real-Time Clock (RTC), External Interrupt Controller (EIC), and PORT peripherals in the Secure project while SERCOM3 (as USART), DMA peripherals in the Non-Secure project. MCC also generates *main.c* files in both Secure and Non-Secure Projects.

Note: *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 Secure Project

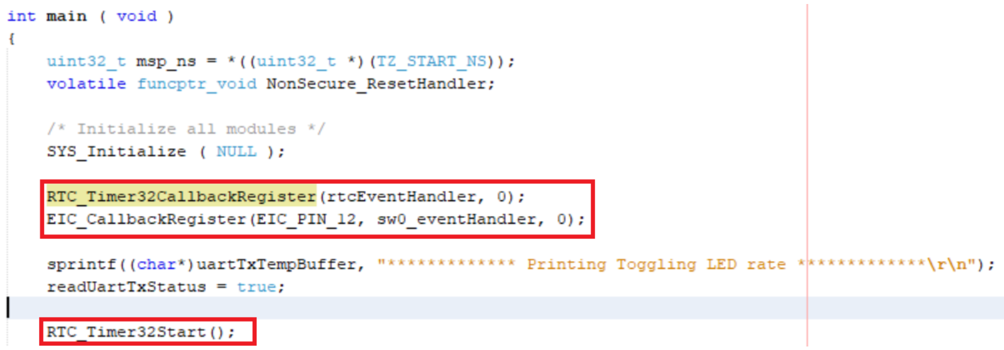
To develop and run the application, follow these steps:

1. Open the *main.c* file of the Secure project and add the following application logic. Add the following code to the register RTC event handlers for a 500 ms compare event. `RTC_Timer32CallbackRegister (rtcEventHandler, 0);` and add the following code to the EIC callback event handler for the switch press event `EIC_CallbackRegister (EIC_PIN_12, sw0_eventHandler, 0);` in the `main()` function and `SYS_Initialize (NULL)` function below:

```
RTC_Timer32CallbackRegister(rtcEventHandler, 0);
EIC_CallbackRegister(EIC_PIN_12, sw0_eventHandler, 0);
sprintf((char*)uartTxTempBuffer,
"************* Printing Toggling LED rate
*************\r\n");
    readUartTxStatus = true;
```

2. Add the line of code to call the `RTC_Timer32Start();` function after registering the callback event handlers.

Figure 1-19. Adding Application Logic to Register Callback Event Handlers



3. Implement the registered callback event handlers for Secure peripherals by adding the following code outside the `main()` function.

```
static void sw0_eventHandler(uintptr_t context)
{
    changeSamplingRate = true;
}

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

4. In the `secureApp()` function, add the application logic of toggling LED at different rates of 500 ms, 1s, 2s, and 4s whenever there is a switch press on the board by the user before the `main()` function in Secure project.
Note: Add this function outside the `main()` function.

```
void secureApp(void)
{
    /* Basic Functionality: Demonstrates an LED toggle, i.e. LED0 toggles when
     * the switch SW0 is pressed on a timeout basis and prints the LED toggling
     * rate on the serial terminal.*/
        if (printLedToggleRate == true)
        {
            memset((char*)uartTxTempBuffer, 0x00, 100);
            sprintf((char*)uartTxTempBuffer, "************* Printing Toggling LED rate
*************\r\n");
            printLedToggleRate = false;
            readUartTxStatus = true;
        }
        if (isRTCTimerExpired == true)
        {
            isRTCTimerExpired = false;
            memset((char*)uartTxTempBuffer, 0x00, 100);
            sprintf((char*)uartTxTempBuffer, "Toggling LED at %s rate \r\n",
&timeouts[(uint8_t)tempSampleRate][0]);
            LED0_Toggle();
            readUartTxStatus = true;
        }
        if(changeSamplingRate == true)
        {
            changeSamplingRate = false;
            if(tempSampleRate == SAMPLING_RATE_500MS)
            {
                tempSampleRate = SAMPLING_RATE_1S;
                RTC_Timer32CompareSet(PERIOD_1S);
            }
            else if(tempSampleRate == SAMPLING_RATE_1S)
            {
                tempSampleRate = SAMPLING_RATE_2S;
                RTC_Timer32CompareSet(PERIOD_2S);
            }
            else if(tempSampleRate == SAMPLING_RATE_2S)
```

```
            {
                tempSampleRate = SAMPLING_RATE_4S;
                RTC_Timer32CompareSet(PERIOD_4S);
            }
            else if(tempSampleRate == SAMPLING_RATE_4S)
            {
               tempSampleRate = SAMPLING_RATE_500MS;
               RTC_Timer32CompareSet(PERIOD_500MS);
            }
            else
            {
                ;
            }
            RTC_Timer32CounterSet(0);
            sprintf((char*)uartTxTempBuffer, "LED Toggling rate is changed to %s\r\n",
&timeouts[(uint8_t)tempSampleRate][0]);
            readUartTxStatus = true;
        }
}
```

Add the following code snippet to include the necessary header files, and define the macros for different RTC compare values.
Note: Add this logic at start of the file to include necessary files which have definitions of functions used in the file.

```
#include <stdio.h>
#include <string.h>

#define PERIOD_500MS                           512
#define PERIOD_1S                              1024
#define PERIOD_2S                              2048
#define PERIOD_4S                              4096
```

Add the following code snippet after including the header files.

The following codes declare various flags whose status is monitored and changed by event handlers in the application.

```
static volatile bool isRTCTimerExpired = false;
static volatile bool changeSamplingRate = false;
static volatile bool printLedToggleRate = false;
static const char timeouts[4][20] = {"500 milliSeconds", "1 Second",  "2 Seconds",  "4
Seconds"};

volatile bool readUartTxStatus = false;

uint8_t uartTxTempBuffer[100] = {0};

typedef enum
{
    SAMPLING_RATE_500MS = 0,
    SAMPLING_RATE_1S    = 1,
    SAMPLING_RATE_2S    = 2,
    SAMPLING_RATE_4S    = 3,
} SAMPLING_RATE;

static SAMPLING_RATE tempSampleRate = SAMPLING_RATE_500MS;
```

5. In the `nonsecure_entry.c` file, availble under *Source Files* > *trustZone*, implement the Non-Secure callables below to access and request the Secure application from the Non-Secure application.
Note: Delete the generated template code and add the following code.

```
bool __attribute__((cmse_nonsecure_entry)) readUartTxData(uint8_t *lcluartTxBuffer)
{
    bool localSecureUartStatus = readUartTxStatus;
    if(localSecureUartStatus == true)
    {
        memset((char*)lcluartTxBuffer, 0x00, 100);
        memcpy(lcluartTxBuffer, uartTxTempBuffer, strlen((const char
*)&uartTxTempBuffer[0]));
        readUartTxStatus   = false;
    }
```

```
    return (localSecureUartStatus);
}

void __attribute__((cmse_nonsecure_entry)) secureAppEntry(void)
{
    secureApp();
}
```

Add the following code snippet to include necessary header files and extern the variables and prototype of the *SecureApp()* function whose implementation takes place in the Secure `main.c` file.

```
#include <stdint.h>
#include <stdio.h>
#include <stddef.h>                      // Defines NULL
#include <stdbool.h>                     // Defines true
#include <stdlib.h>                      // Defines EXIT_FAILURE
#include <string.h>
extern uint8_t uartTxTempBuffer[];
extern volatile bool readUartTxStatus;
extern void secureApp(void);
```

1.5 Adding Application Code to Non-Secure Project

1. In the `main()` function below `SYS_Initialize()` add the following code to register callback event handlers.

```
DMAC_ChannelCallbackRegister(DMAC_CHANNEL_0, usartTxDmaChannelHandler, 0);
```

2. Implement the registered callback event handler before the `main()` function.

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

3. Remove `SYS_Tasks( );` function call and replace with the below code to enter the Secure functionality from this Non-Secure application inside the `while` loop.

```
    while ( true )
    {
        if (readUartTxData(nonSecureUartTxBuffer) == true)
        {
        /* Maintain state machines of all polled MPLAB Harmony modules. */
            DMAC_ChannelTransfer(DMAC_CHANNEL_0, nonSecureUartTxBuffer, \
                    (const void *)&(SERCOM3_REGS->USART_INT.SERCOM_DATA), \
                    strlen((const char*)nonSecureUartTxBuffer));
        }
        secureAppEntry();
    }
```

Figure 1-20. Adding Application Logic to Enter Secure Functionality from Non-Secure App

```
int main ( void )
{
    /* Initialize all modules */
    SYS_Initialize ( NULL );

    DMAC_ChannelCallbackRegister(DMAC_CHANNEL_0, usartTxDmaChannelHandler, 0);
    while ( true )
    {
        if (readUartTxData(nonSecureUartTxBuffer) == true)
        {
        /* Maintain state machines of all polled MPLAB Harmony modules. */
            DMAC_ChannelTransfer(DMAC_CHANNEL_0, nonSecureUartTxBuffer, \
                    (const void *)&(SERCOM3_REGS->USART_INT.SERCOM_DATA), \
                    strlen((const char*)nonSecureUartTxBuffer));
        }
        secureAppEntry();
    }
```

Add the following code snippet to include the header files and declaration of variables used in the Non-Secure `main.c` file.

```
#include <stdio.h>
#include <string.h>
#include "trustZone/nonsecure_entry.h"

static volatile bool isUSARTTxComplete = false;
static volatile bool isUSARTRxComplete = false;
static uint8_t nonSecureUartTxBuffer[100] = {0};
```

4. In the `nonsecure_entry.h` file, available under *Header Files* > *trustZone*, add the following code by declaring NSCs with extern keyword to access and request the Secure application from the Non-Secure application.
   Note: Delete the generated template code and add the following code.

```
extern bool readUartTxData(uint8_t *lcluartTxBuffer);
extern void secureAppEntry(void);
```

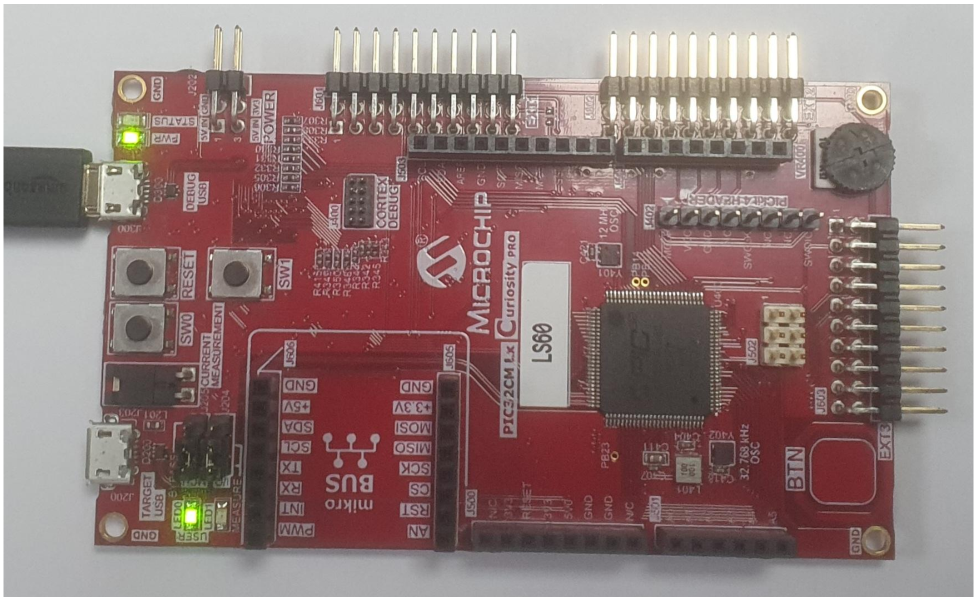
Figure 1-21. Global NSCs to Access and Request Secure App from Non-Secure App

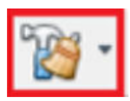
```
#ifndef NONSECURE_ENTRY_H_
#define NONSECURE_ENTRY_H_

/* Non-secure callable functions */
extern bool readUartTxData(uint8_t *lcluartTxBuffer);
extern void secureAppEntry(void);

#endif /* NONSECURE_ENTRY_H_ */
```

1.6 Building and Programming the Application

1. The PIC32CM LS60 Curiosity Pro Evaluation Kit supports Embedded Debugger (EDBG) for debugging. Connect the 'Type-A male to micro-B' USB cable to the micro-B debug USB port on the PIC32CM LS60 Curiosity Pro Evaluation Kit to power and debug.

Figure 1-22. Hardware Setup

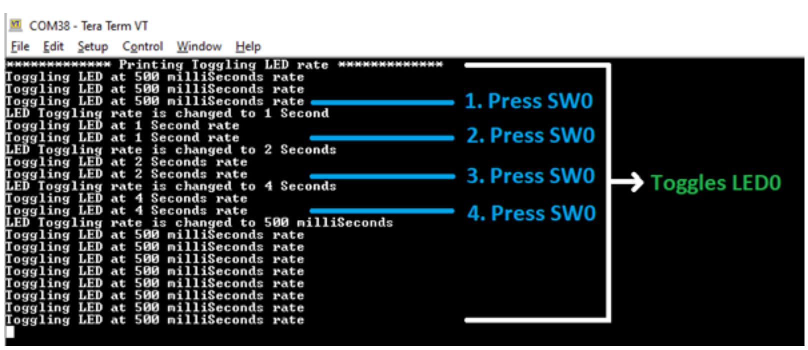


2. Setup the Non-Secure project as the main project, and from the Project Properties select the latest compiler version (v4.35).
3. Clean and build the project by clicking  (the Clean and Build icon).
4. Program the application by clicking  (the Program the Device icon).

1.7 Observing the Output on Board and Serial Terminal

1. When the application builds and completes programming, open the *Tera Term* tool on the PC. Select **Serial Port** and set the *baud rate* as 115200.
2. Press the *RESET* button on the PIC32CM LS60 Curiosity Pro Evaluation Kit, and the LED will toggle at 500 ms by default, and with every subsequent *SW0* switch press, the LED toggling rate will change to 1s, 2s, and 4s.

Figure 1-23. LED Toggling Rate on Serial Terminal



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

2. References

- 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 a more detailed insight into this project, refer to Microchip Developer Help on the YouTube channel: www.youtube.com/watch?v=5w0JYHnSzPM
- For more information on various applications, refer to https://github.com/Microchip-MPLAB-Harmony/reference_apps
- Secure Boot on PIC32CM LS60 Curiosity Pro Evaluation Kit, refer to https://microchipdeveloper.com/harmony3:secure-boot-application-on-pic32cm-ls60
- PIC32CM LS60 Curiosity Pro Evaluation Kit
- 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 B - April 2024

The following updates were incorporated for this revision:

Document	Minor changes of format.
Introduction	Separated Secure and Non-Secure modules characteristics.
1. Creating the First Trust Zone Application on PIC32CM LS60 MCU	Updated version of software tools. Updated steps in section 1.1. Creating an MPLAB Harmony v3-based Project. Updated Figure 1-1, Figure 1-2, Figure 1-3, 1.4. Adding Application Logic to Secure Project and Figure 1-5. Updated steps in section 1.2. Adding and Configuring the MPLAB Harmony Components. Added Figure 1-6, Figure 1-13, Figure 1-14 and Figure 1-15. Updated Figure 1-7, Figure 1-9 and Figure 1-12. Updated steps in section 1.4. Adding Application Logic to Secure Project. Updated steps in section 1.5. Adding Application Code to Non-Secure Project. Updated Figure 1-20.
2. References	Added new references.

Revision A - September 2023

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: 978-1-6683-4278-7

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