Getting started with MCC and Soteria-G3

User guide

Rev 1.2	Jan 31, 2023

Microchip Technology Ltd.

TABLE OF CONTENTS

1 Introduction

1.1 Purpose

This document provides details on how to use MCC with CEC173x part and use Soteria secure-boot solution.

1.2 Scope

The scope of this document is limited to providing the user with a high-level overview of MCC, Soteria-G3 and getting started with using Soteria-G3 in CEC173x part.

1.3 References

MPLAB MCC getting started: **https://microchipdeveloper.com/mcc:start**

1.4 Pre-requisites

IDE	MPLABX IDE v6.05
DFP	v1.8.258
Debugger (only in case of debugging)	ICD4 or PICKit4
Compiler	XC32 v4.21
Device	CEC1736_S0_2ZW
Development board	EV19K07A development board with, (a) CEC173x internal Flash pre-programmed binary, and (b) external Flash modules with pre-programmed AP_FW binaries
Harmony3 Core	v1.1.5
Utilities	MicrochipTech/cec173x_soteria_utilities (github.com)

1.5 Assumptions and Dependencies

The user is expected to have a fair idea of using MCC with any other Microchip micro-controllers.

1.6 Glossary of Terms and Acronyms

Term/Acronym	**Meaning/Expansion**

AP	Application Processor
API	Application Programming Interface
BSP	Board Support Package
CoT	Chain-Of-Trust
ECIA	Embedded Controller Interrupt Aggregator
GPIO	General Purpose Input Output
HAL	Hardware Abstraction Layer
Hex	Hexadecimal
IRQ	Interrupt Request
MCC	Microchip Code Configurator
OEM	Original Equipment Manufacturer
PLIB	Peripheral Library
RoT	Root-Of-Trust
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver and Transmitter

2 What is Soteria?

Soteria-G3 is a firmware design executed on the CEC173x family of devices. It can be used in conjunction with any application processor (AP) that boots out of an external SPI flash device to extend the Root-of-Trust (RoT) and enforce a secure boot process in the system.

Soteria-G3 uses the CEC173x immutable secure bootloader, implemented in ROM, as the system RoT. The CEC173x secure bootloader loads, decrypts and authenticates the embedded controller firmware from the external (or internal) SPI Flash device. The validated Soteria-G3 that runs on the CEC173x is designed to subsequently authenticate the application processor firmware (AP_FW) located in the same SPI Flash component and up to three additional SPI Flash components.

Soteria-G3 prevents the system from booting unless the AP_FW stored in the external SPI Flash is authentic code signed by the original equipment manufacturer (OEM). It offers security features to authenticate the SPI Flash image in the external SPI Flash device.

The validated AP_FW that runs on the application processor can utilize crypto resources in the CEC173x to authenticate other code in the system, thereby extending the Chain-of-Trust (CoT) to ensure that all code running in the system is authorized.

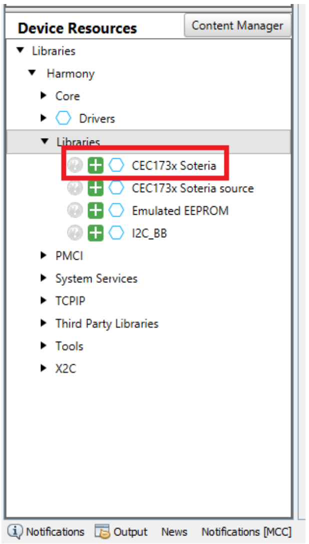
Soteria-G3 also supports secure firmware updates, which can authenticate updates to both AP_FW and Soteria-G3 in the system.

3 Setting up an MCC project with Soteria library

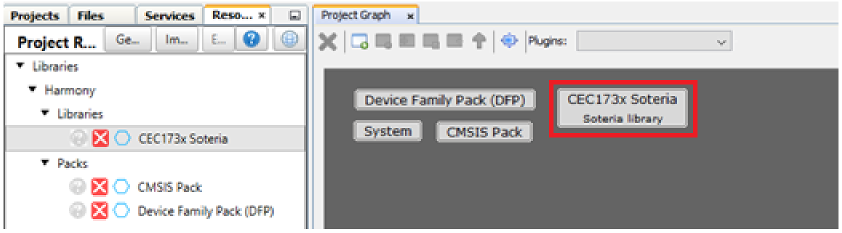
3.1 Creating Project and adding components

3.1.1 SG3 library component

1. Create a new ***"32-bit MCC Harmony Project"*** and select ***"CEC1736_S0_2ZW"*** as the target device
2. Select and download ***"cec173x_soteria_lib"*** component from MCC content manager
3. To add Soteria as a library into the created application project, ***"double click"*** on ***"CEC173x Soteria"*** componenet which can be found under ***"Libraries → Harmony → Libraries → CEC173x Soteria"*** under ***"Device Resources"*** window as shown below

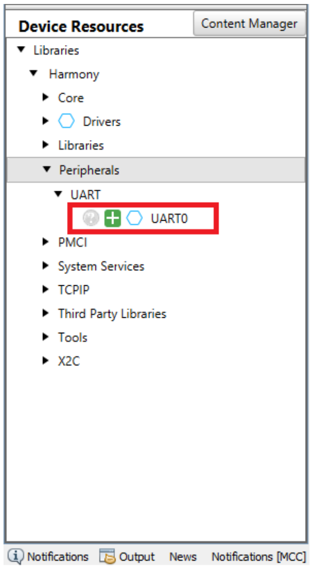


4. The Soteria library component should get added in the ***"Project Graph"*** and ***"Project Resources"*** as shown below

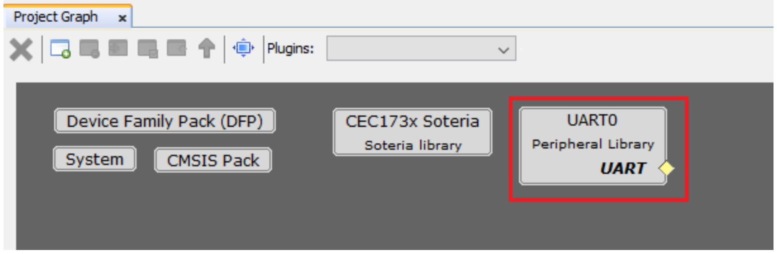


3.1.2 UART peripheral component

1. To add UART peripheral into the created application project, ***"double click"*** on ***"UART0"*** componenet which can be found under ***"Peripherals → UART → UART0"*** under ***"Device Resources"*** window as shown below

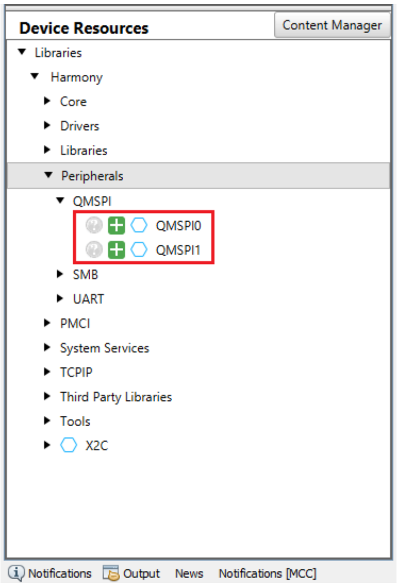


2. The UART peripheral component should get added in the ***"Project Graph"*** and ***"Project Resources"*** as shown below

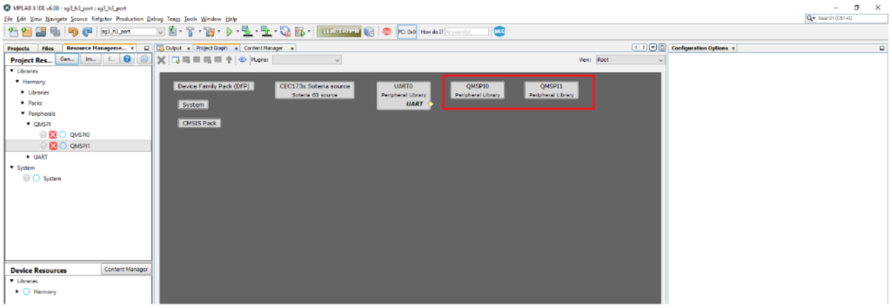


3.1.3 QMSPI peripheral component

1. To add **QMSPI** peripheral into the created application project, ***"double click"*** on **"QMSPI0"** which can be found under ***"Peripherals → QMSPI → QMSPI0"*** under ***"Device Resources"*** window as shown below

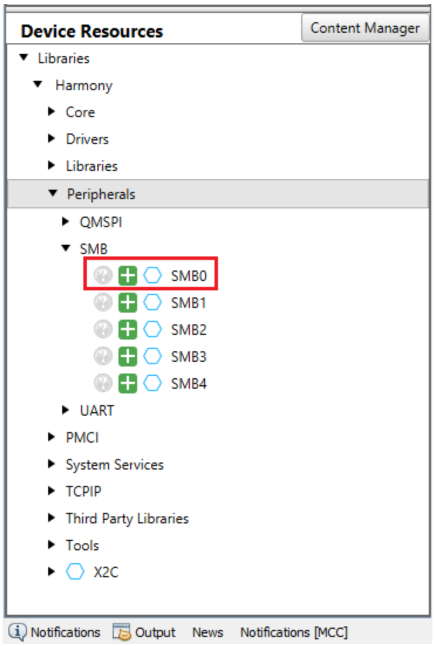


2. Follow similar steps mentioned in step #7, add “***QMSPI1***”, which is located below “***QMSPI0”***
3. The QMSPI peripheral components should get added in the ***“Project Graph”*** and ***“Project Resources”*** as shown below

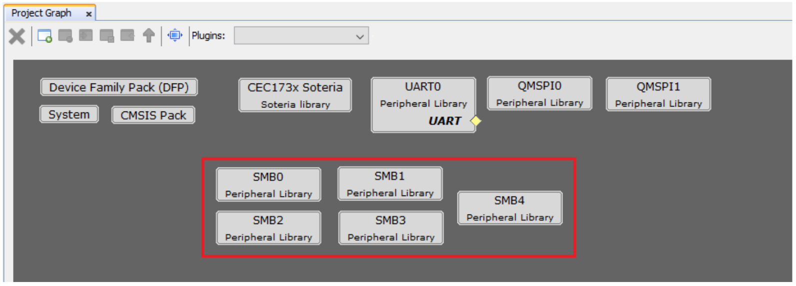


3.1.4 SMBUS peripheral component

1. To add SMB peripheral into the created application project, ***“double click”*** on ***“SMB0”*** which can be found under ***“Peripherals → SMB → SMB0”*** under ***“Device Resources”*** window as shown below

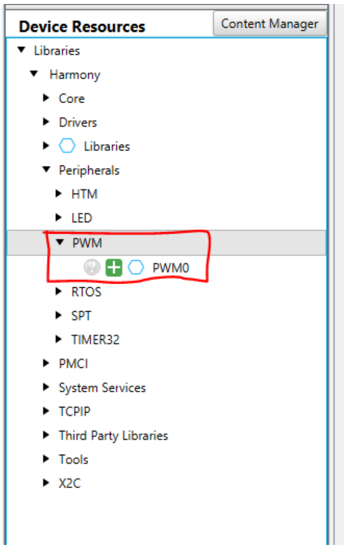


2. Similarly, add ***SMB1***, ***SMB2***, ***SMB3***, ***SMB4*** found under “***Peripherals → SMB”*** under ***“Device Resources”*** window
3. The ***SMB*** peripheral components should get added in the ***“Project Graph”*** and ***“Project Resources”*** as shown below

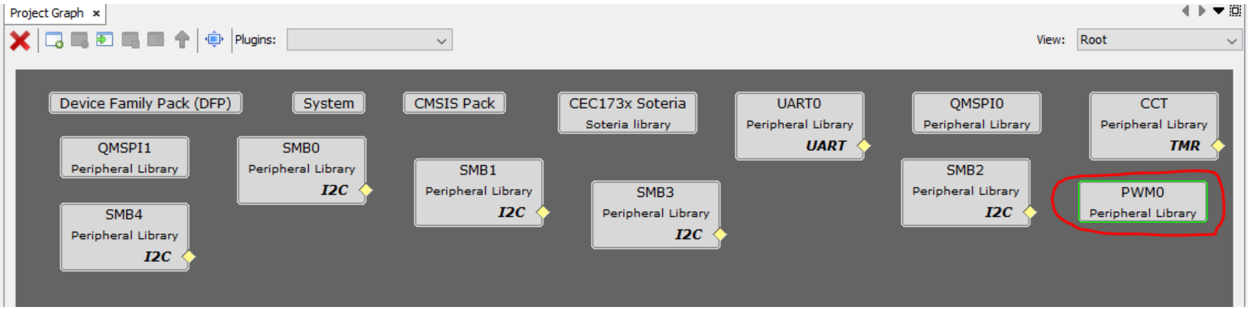


3.1.5 PWM peripheral component

1. To add ***PWM*** peripheral into the created application project, ***"double click"*** on **"PWM0"** which can be found under ***"Peripherals → PWM → PWM0"*** under ***"Device Resources"*** window as shown below

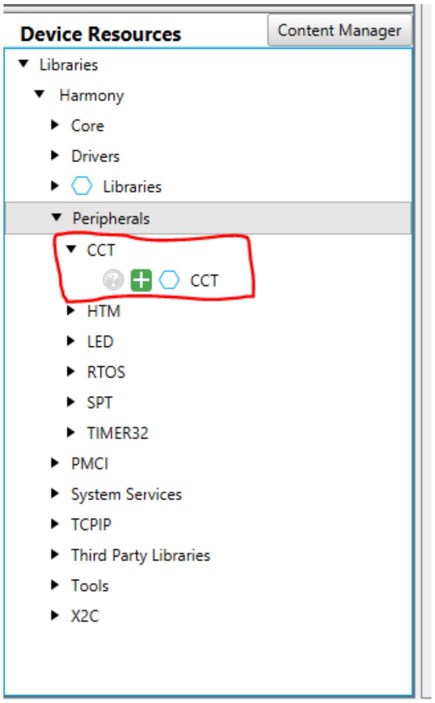


2. The ***PWM*** peripheral components should get added in the ***"Project Graph"*** and ***"Project Resources"*** as shown below

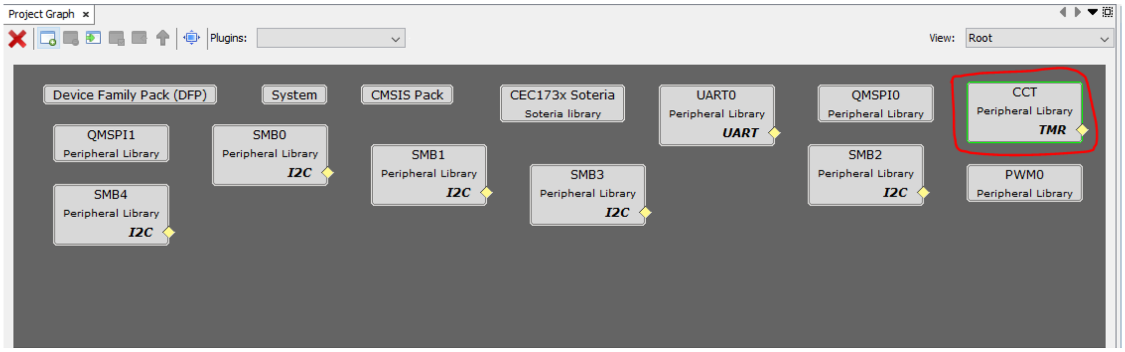


3.1.6 Capture and compare timer peripheral component

3. To add ***CCT*** peripheral into the created application project, ***"double click"*** on ***"CCT"*** which can be found under ***"Peripherals → CCT → CCT"*** under ***"Device Resources"*** window as shown below



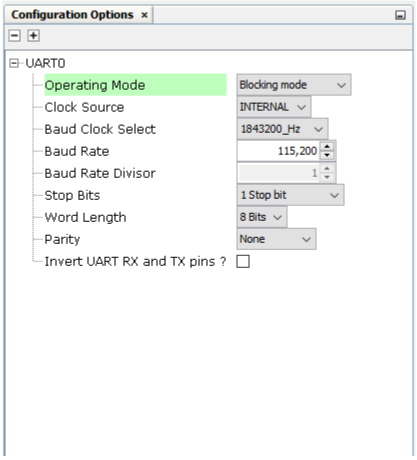
4. The ***CCT*** peripheral components should get added in the ***"Project Graph"*** and ***"Project Resources"*** as shown below



3.2 Configuring peripheral components

3.2.1 UART peripheral component

1. Change the UART0 configuration as shown in the below image



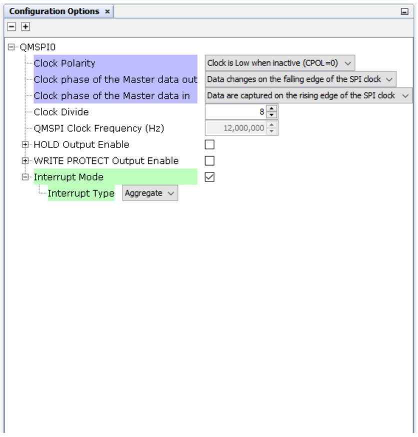
Configuration Options

UART0

Operating Mode	Blocking mode
Clock Source	INTERNAL
Baud Clock Select	1843200_Hz
Baud Rate	115,200
Baud Rate Divisor	1
Stop Bits	1 Stop bit
Word Length	8 Bits
Parity	None
Invert UART RX and TX pins ?	□

3.2.2 QMSPI peripheral component

1. Change ***QMSPI0*** and ***QMSPI1*** configurations as shown in the below image



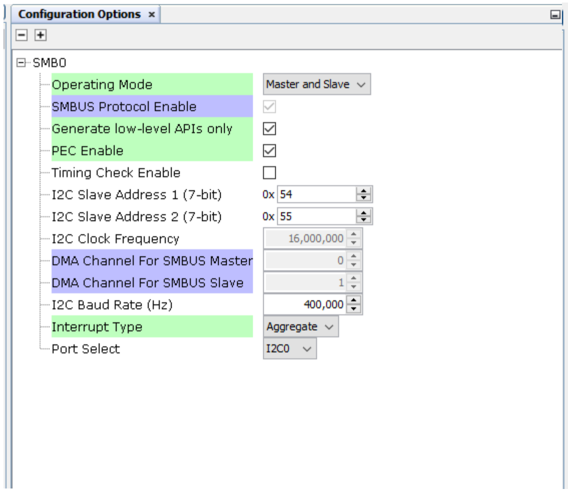
Configuration Options

QMSPI0

Clock Polarity	Clock is Low when inactive (CPOL=0)
Clock phase of the Master data out	Data changes on the falling edge of the SPI clock
Clock phase of the Master data in	Data are captured on the rising edge of the SPI clock
Clock Divide	8
QMSPI Clock Frequency (Hz)	12,000,000
HOLD Output Enable	□
WRITE PROTECT Output Enable	□
Interrupt Mode	☑
Interrupt Type	Aggregate

3.2.3 SMBUS peripheral component

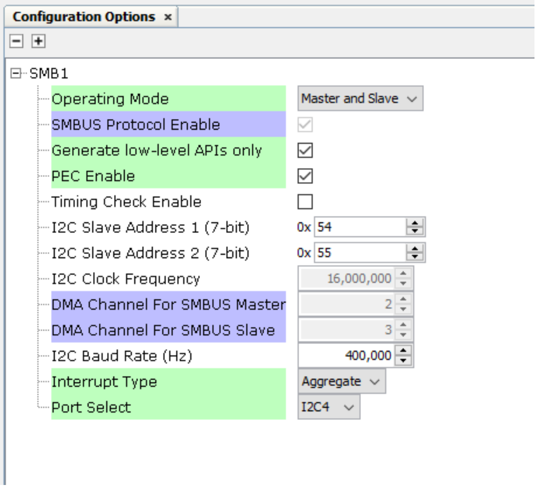
1. Change ***SMB0***, ***SMB1***, ***SMB2***, ***SMB3*** and ***SMB4*** configurations as shown in the below image



Configuration Options

SMB0

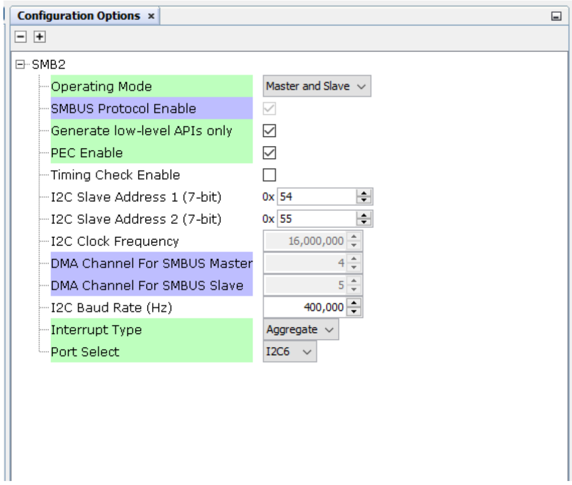
Operating Mode	Master and Slave
SMBUS Protocol Enable	☑
Generate low-level APIs only	☑
PEC Enable	☑
Timing Check Enable	□
I2C Slave Address 1 (7-bit)	0x 54
I2C Slave Address 2 (7-bit)	0x 55
I2C Clock Frequency	16,000,000
DMA Channel For SMBUS Master	0
DMA Channel For SMBUS Slave	1
I2C Baud Rate (Hz)	400,000
Interrupt Type	Aggregate
Port Select	I2C0



Configuration Options

SMB1

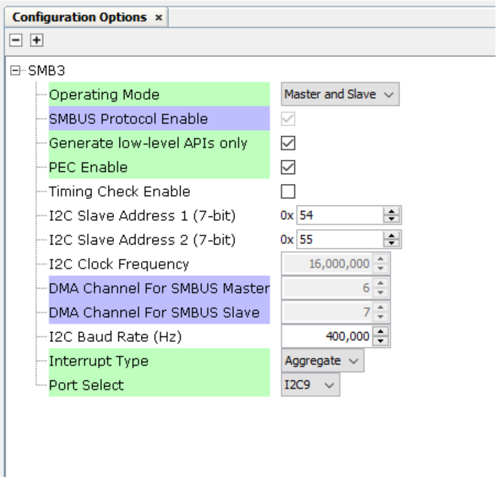
Operating Mode	Master and Slave
SMBUS Protocol Enable	☑
Generate low-level APIs only	☑
PEC Enable	☑
Timing Check Enable	□
I2C Slave Address 1 (7-bit)	0x 54
I2C Slave Address 2 (7-bit)	0x 55
I2C Clock Frequency	16,000,000
DMA Channel For SMBUS Master	2
DMA Channel For SMBUS Slave	3
I2C Baud Rate (Hz)	400,000
Interrupt Type	Aggregate
Port Select	I2C4



Configuration Options ×

SMB2

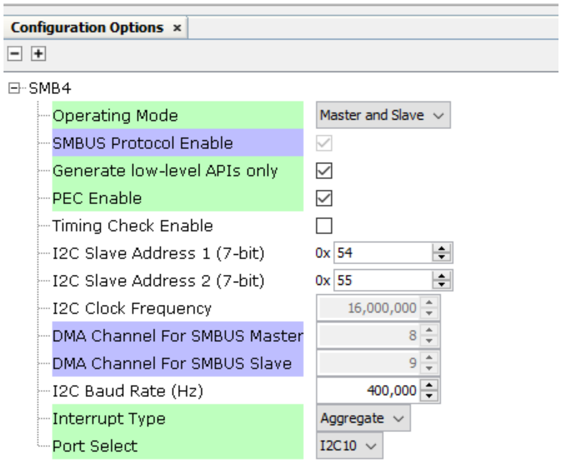
Operating Mode	Master and Slave
SMBUS Protocol Enable	☑
Generate low-level APIs only	☑
PEC Enable	☑
Timing Check Enable	□
I2C Slave Address 1 (7-bit)	0x 54
I2C Slave Address 2 (7-bit)	0x 55
I2C Clock Frequency	16,000,000
DMA Channel For SMBUS Master	4
DMA Channel For SMBUS Slave	5
I2C Baud Rate (Hz)	400,000
Interrupt Type	Aggregate
Port Select	I2C6



Configuration Options ×

SMB3

Operating Mode	Master and Slave
SMBUS Protocol Enable	☑
Generate low-level APIs only	☑
PEC Enable	☑
Timing Check Enable	□
I2C Slave Address 1 (7-bit)	0x 54
I2C Slave Address 2 (7-bit)	0x 55
I2C Clock Frequency	16,000,000
DMA Channel For SMBUS Master	6
DMA Channel For SMBUS Slave	7
I2C Baud Rate (Hz)	400,000
Interrupt Type	Aggregate
Port Select	I2C9



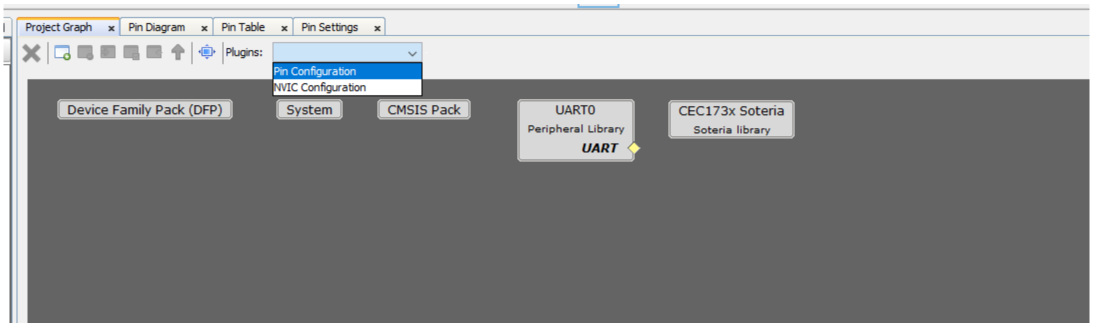
Configuration Options ×

SMB4

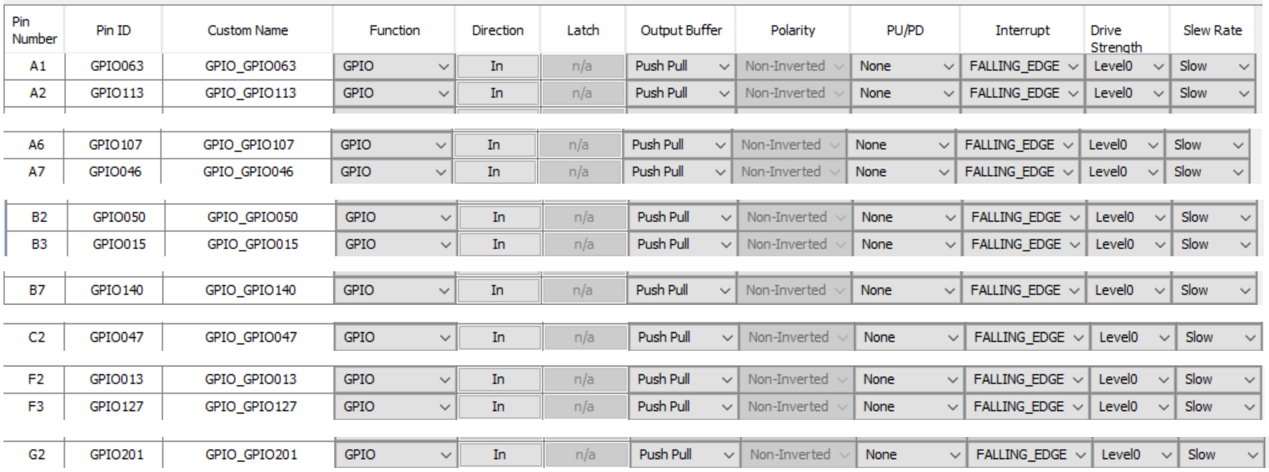
Operating Mode	Master and Slave
SMBUS Protocol Enable	☑
Generate low-level APIs only	☑
PEC Enable	☑
Timing Check Enable	□
I2C Slave Address 1 (7-bit)	0x 54
I2C Slave Address 2 (7-bit)	0x 55
I2C Clock Frequency	16,000,000
DMA Channel For SMBUS Master	8
DMA Channel For SMBUS Slave	9
I2C Baud Rate (Hz)	400,000
Interrupt Type	Aggregate
Port Select	I2C10

3.2.4 GPIO peripheral component

1. Goto ***"Plugins -> Pin Configuration"*** located in the project graph as shown in the below image



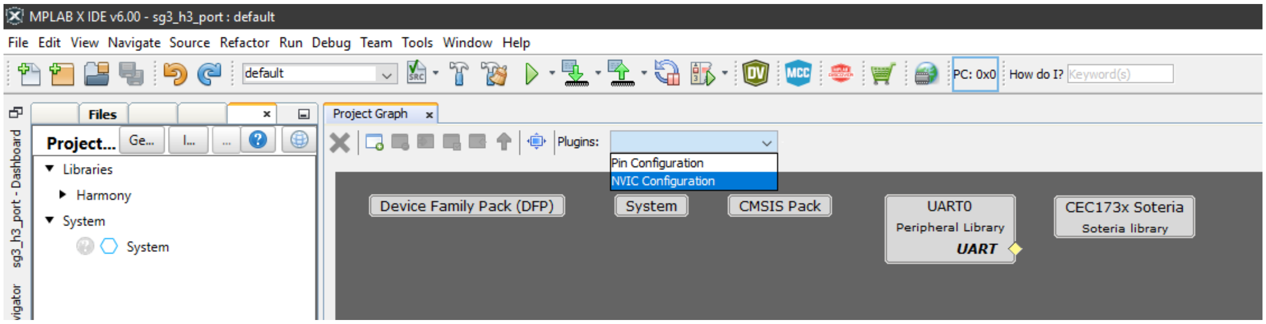
2. Change the pin configurations as shown in the below image



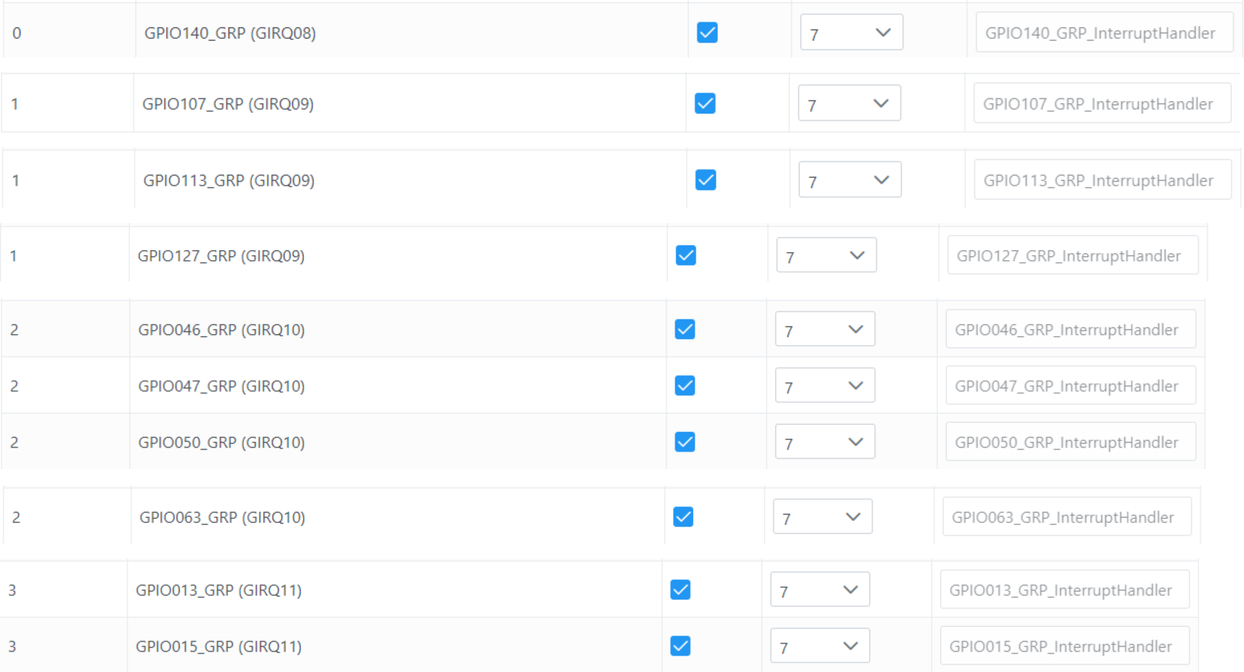
Pin Number	Pin ID	Custom Name	Function	Direction	Latch	Output Buffer	Polarity	PU/PD	Interrupt	Drive Strength	Slew Rate
A1	GPIO063	GPIO_GPIO063	GPIO	In	n/a	Push Pull	Non-Inverted	None	FALLING_EDGE	Level0	Slow
A2	GPIO113	GPIO_GPIO113	GPIO	In	n/a	Push Pull	Non-Inverted	None	FALLING_EDGE	Level0	Slow
A6	GPIO107	GPIO_GPIO107	GPIO	In	n/a	Push Pull	Non-Inverted	None	FALLING_EDGE	Level0	Slow
A7	GPIO046	GPIO_GPIO046	GPIO	In	n/a	Push Pull	Non-Inverted	None	FALLING_EDGE	Level0	Slow
B2	GPIO050	GPIO_GPIO050	GPIO	In	n/a	Push Pull	Non-Inverted	None	FALLING_EDGE	Level0	Slow
B3	GPIO015	GPIO_GPIO015	GPIO	In	n/a	Push Pull	Non-Inverted	None	FALLING_EDGE	Level0	Slow
B7	GPIO140	GPIO_GPIO140	GPIO	In	n/a	Push Pull	Non-Inverted	None	FALLING_EDGE	Level0	Slow
C2	GPIO047	GPIO_GPIO047	GPIO	In	n/a	Push Pull	Non-Inverted	None	FALLING_EDGE	Level0	Slow
F2	GPIO013	GPIO_GPIO013	GPIO	In	n/a	Push Pull	Non-Inverted	None	FALLING_EDGE	Level0	Slow
F3	GPIO127	GPIO_GPIO127	GPIO	In	n/a	Push Pull	Non-Inverted	None	FALLING_EDGE	Level0	Slow
G2	GPIO201	GPIO_GPIO201	GPIO	In	n/a	Push Pull	Non-Inverted	None	FALLING_EDGE	Level0	Slow

3.2.5 NVIC peripheral component

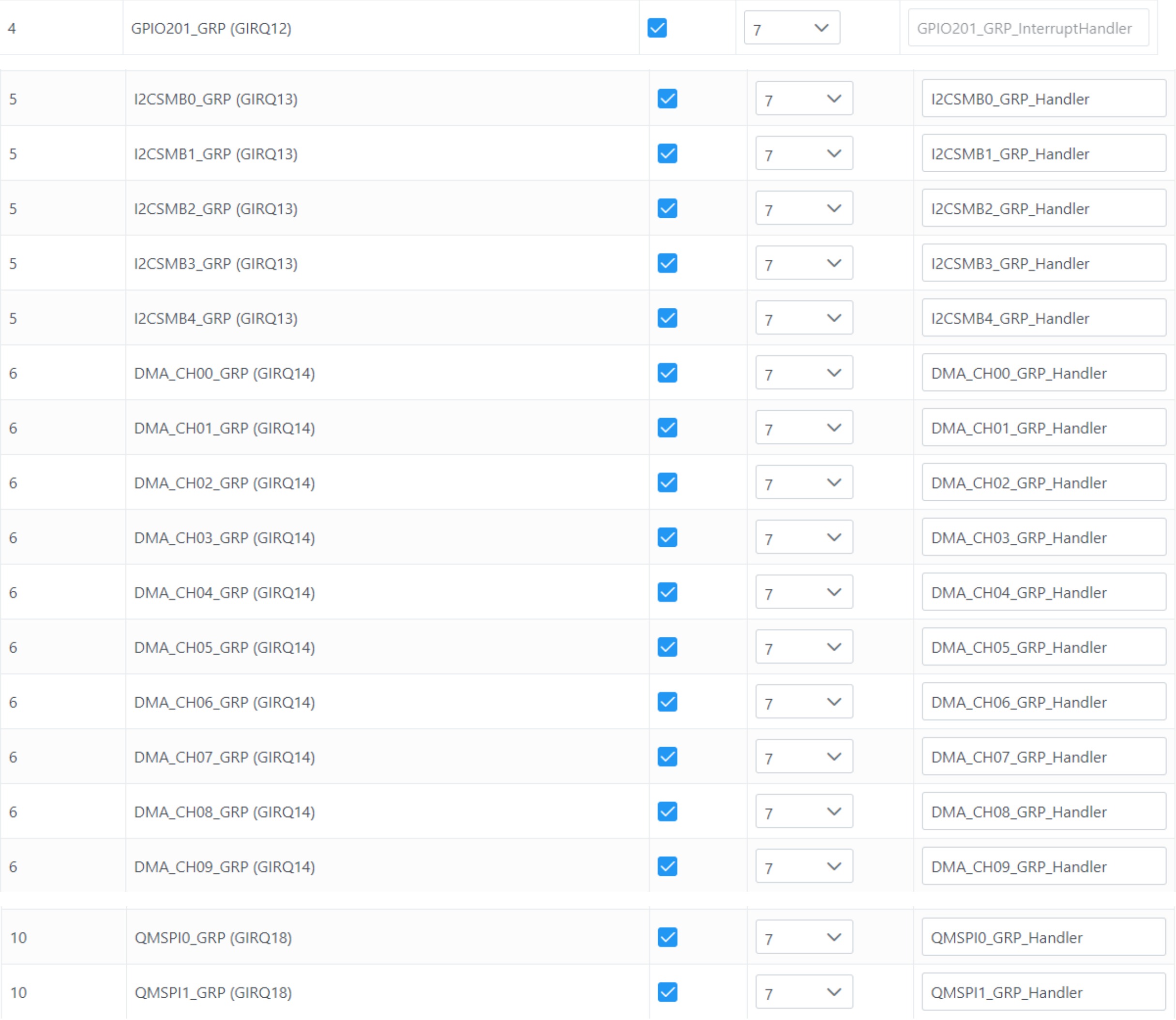
1. Goto ***"Plugins -> NVIC Configuration"*** located in the project graph as shown in the below image



2. Change the interrupt configurations as shown in the below image



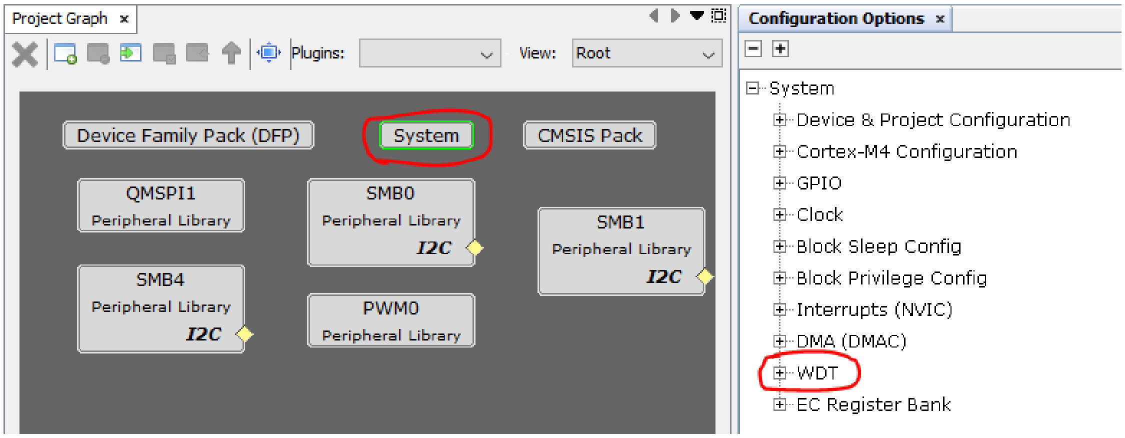
0	GPIO140_GRP (GIRQ08)	☑	7	GPIO140_GRP_InterruptHandler
1	GPIO107_GRP (GIRQ09)	☑	7	GPIO107_GRP_InterruptHandler
1	GPIO113_GRP (GIRQ09)	☑	7	GPIO113_GRP_InterruptHandler
1	GPIO127_GRP (GIRQ09)	☑	7	GPIO127_GRP_InterruptHandler
2	GPIO046_GRP (GIRQ10)	☑	7	GPIO046_GRP_InterruptHandler
2	GPIO047_GRP (GIRQ10)	☑	7	GPIO047_GRP_InterruptHandler
2	GPIO050_GRP (GIRQ10)	☑	7	GPIO050_GRP_InterruptHandler
2	GPIO063_GRP (GIRQ10)	☑	7	GPIO063_GRP_InterruptHandler
3	GPIO013_GRP (GIRQ11)	☑	7	GPIO013_GRP_InterruptHandler
3	GPIO015_GRP (GIRQ11)	☑	7	GPIO015_GRP_InterruptHandler



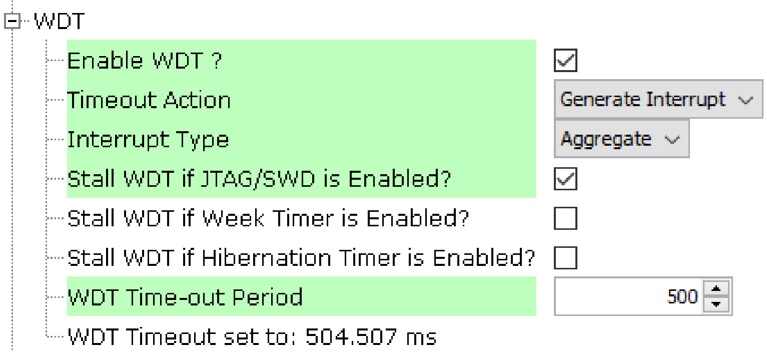
4	GPIO201_GRP (GIRQ12)	☑	7	GPIO201_GRP_InterruptHandler
5	I2CSMB0_GRP (GIRQ13)	☑	7	I2CSMB0_GRP_Handler
5	I2CSMB1_GRP (GIRQ13)	☑	7	I2CSMB1_GRP_Handler
5	I2CSMB2_GRP (GIRQ13)	☑	7	I2CSMB2_GRP_Handler
5	I2CSMB3_GRP (GIRQ13)	☑	7	I2CSMB3_GRP_Handler
5	I2CSMB4_GRP (GIRQ13)	☑	7	I2CSMB4_GRP_Handler
6	DMA_CH00_GRP (GIRQ14)	☑	7	DMA_CH00_GRP_Handler
6	DMA_CH01_GRP (GIRQ14)	☑	7	DMA_CH01_GRP_Handler
6	DMA_CH02_GRP (GIRQ14)	☑	7	DMA_CH02_GRP_Handler
6	DMA_CH03_GRP (GIRQ14)	☑	7	DMA_CH03_GRP_Handler
6	DMA_CH04_GRP (GIRQ14)	☑	7	DMA_CH04_GRP_Handler
6	DMA_CH05_GRP (GIRQ14)	☑	7	DMA_CH05_GRP_Handler
6	DMA_CH06_GRP (GIRQ14)	☑	7	DMA_CH06_GRP_Handler
6	DMA_CH07_GRP (GIRQ14)	☑	7	DMA_CH07_GRP_Handler
6	DMA_CH08_GRP (GIRQ14)	☑	7	DMA_CH08_GRP_Handler
6	DMA_CH09_GRP (GIRQ14)	☑	7	DMA_CH09_GRP_Handler
10	QMSPI0_GRP (GIRQ18)	☑	7	QMSPI0_GRP_Handler
10	QMSPI1_GRP (GIRQ18)	☑	7	QMSPI1_GRP_Handler

3.2.6 WDT peripheral component

1. Goto ***"System-> WDT"*** as shown in the image below



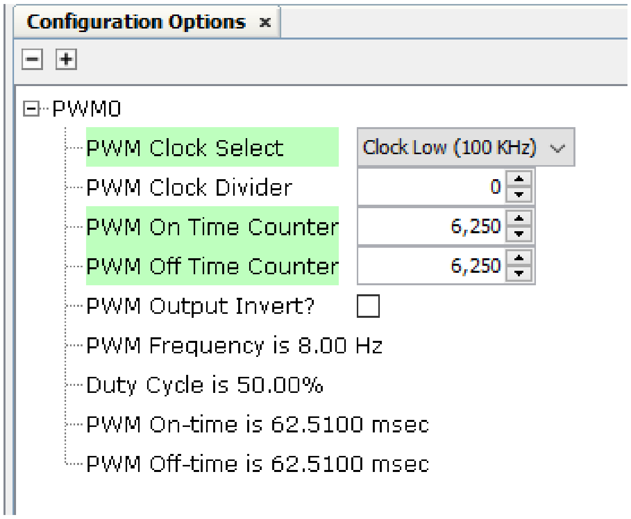
2. Change ***WDT*** configuration as shown in the below image



WDT
- Enable WDT ? ☑
- Timeout Action: Generate Interrupt
- Interrupt Type: Aggregate
- Stall WDT if JTAG/SWD is Enabled? ☑
- Stall WDT if Week Timer is Enabled? □
- Stall WDT if Hibernation Timer is Enabled? □
- WDT Time-out Period: 500
- WDT Timeout set to: 504.507 ms

3.2.7 PWM peripheral component

1. Change ***PWM0*** configuration as shown in the below image

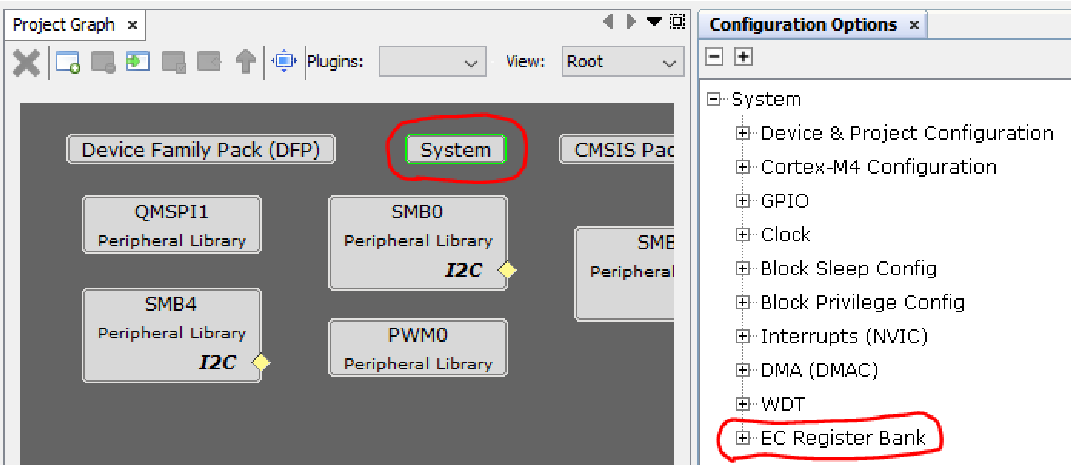


Configuration Options

PWM0
- PWM Clock Select: Clock Low (100 KHz)
- PWM Clock Divider: 0
- PWM On Time Counter: 6,250
- PWM Off Time Counter: 6,250
- PWM Output Invert? □
- PWM Frequency is 8.00 Hz
- Duty Cycle is 50.00%
- PWM On-time is 62.5100 msec
- PWM Off-time is 62.5100 msec

3.2.8 VTR monitor peripheral component

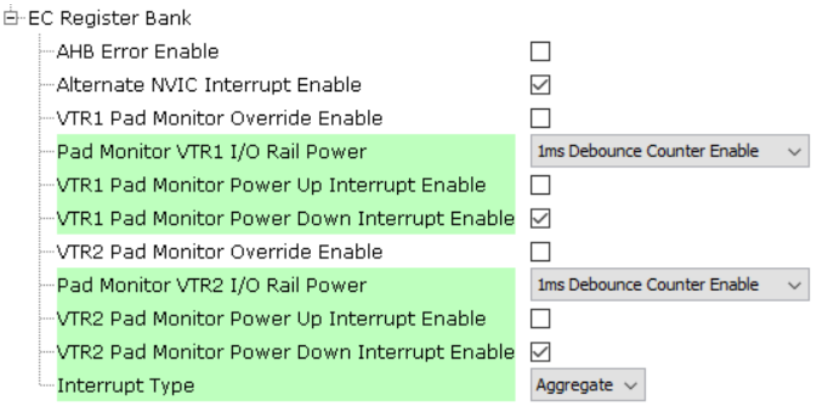
1. Goto ***“System-> EC Register Bank”*** as shown in the image below



Configuration Options

System
- Device & Project Configuration
- Cortex-M4 Configuration
- GPIO
- Clock
- Block Sleep Config
- Block Privilege Config
- Interrupts (NVIC)
- DMA (DMAC)
- WDT
- EC Register Bank

2. Change ***VTR monitor*** configuration as shown in the below image

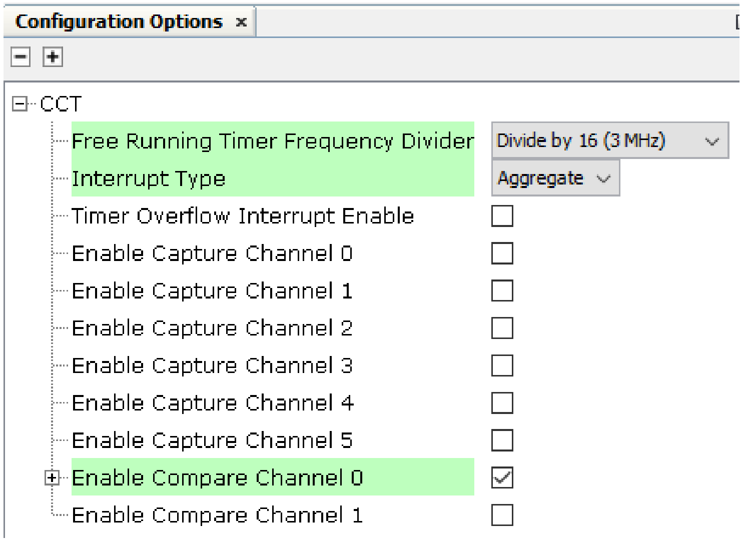


EC Register Bank

AHB Error Enable	□
Alternate NVIC Interrupt Enable	☑
VTR1 Pad Monitor Override Enable	□
Pad Monitor VTR1 I/O Rail Power	1ms Debounce Counter Enable
VTR1 Pad Monitor Power Up Interrupt Enable	□
VTR1 Pad Monitor Power Down Interrupt Enable	☑
VTR2 Pad Monitor Override Enable	□
Pad Monitor VTR2 I/O Rail Power	1ms Debounce Counter Enable
VTR2 Pad Monitor Power Up Interrupt Enable	□
VTR2 Pad Monitor Power Down Interrupt Enable	☑
Interrupt Type	Aggregate

3.2.9 Capture and compare timer peripheral component

1. Change ***Capture and Compare Timer*** configuration as shown in the below image

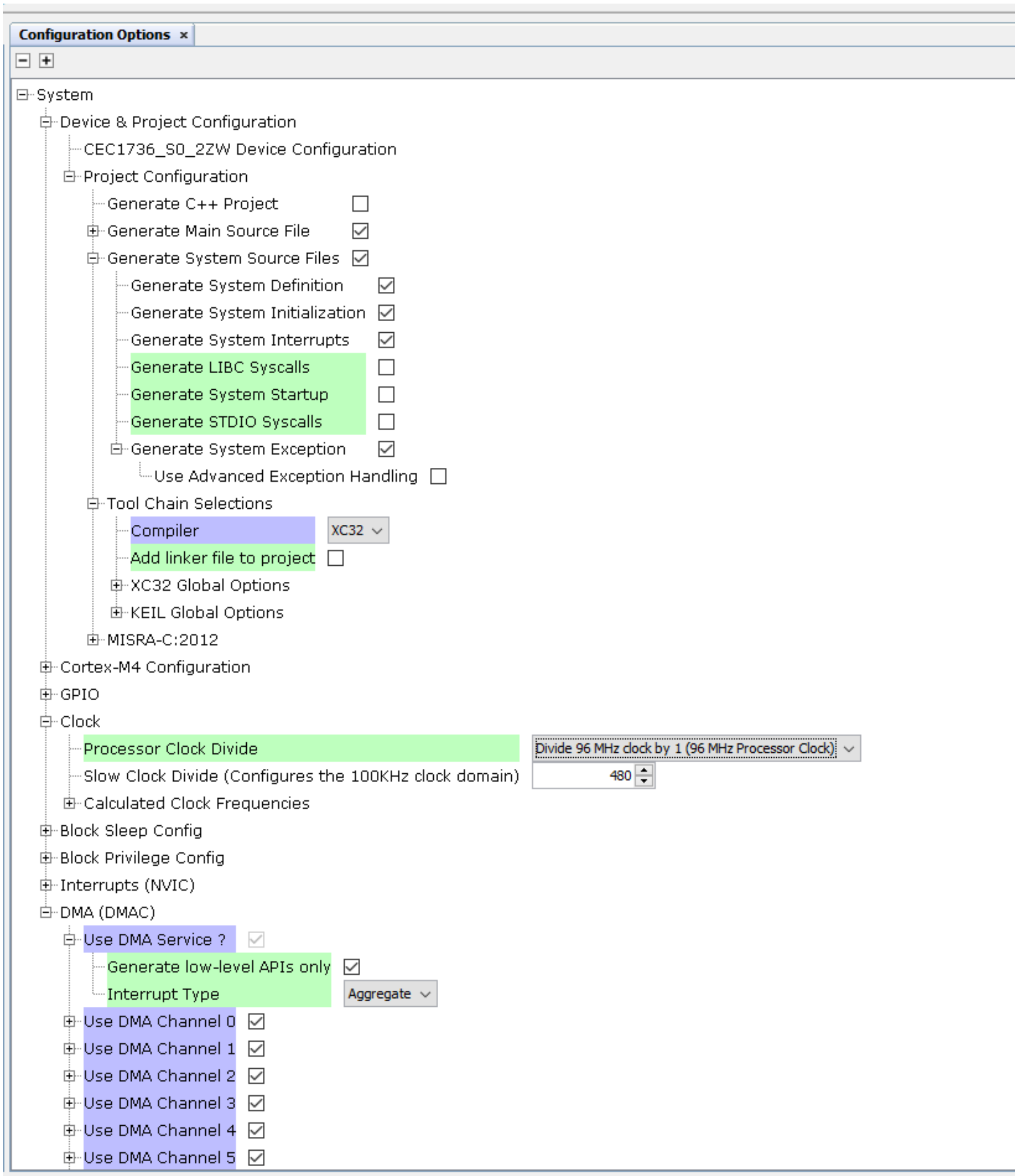


CCT

Free Running Timer Frequency Divider	Divide by 16 (3 MHz)
Interrupt Type	Aggregate
Timer Overflow Interrupt Enable	□
Enable Capture Channel 0	□
Enable Capture Channel 1	□
Enable Capture Channel 2	□
Enable Capture Channel 3	□
Enable Capture Channel 4	□
Enable Capture Channel 5	□
Enable Compare Channel 0	☑
Enable Compare Channel 1	□

3.3 Configuring project settings

1. Select the project configurations as shown in the below image

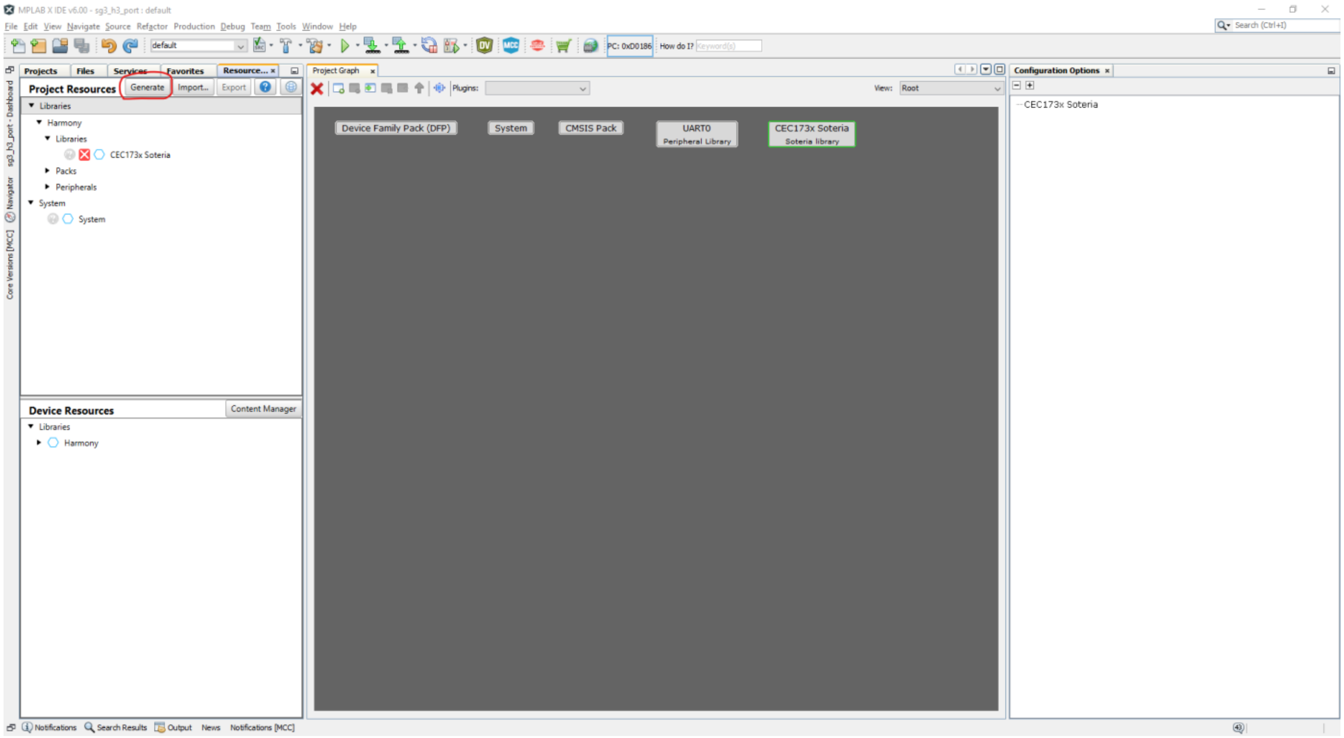


Configuration Options ×

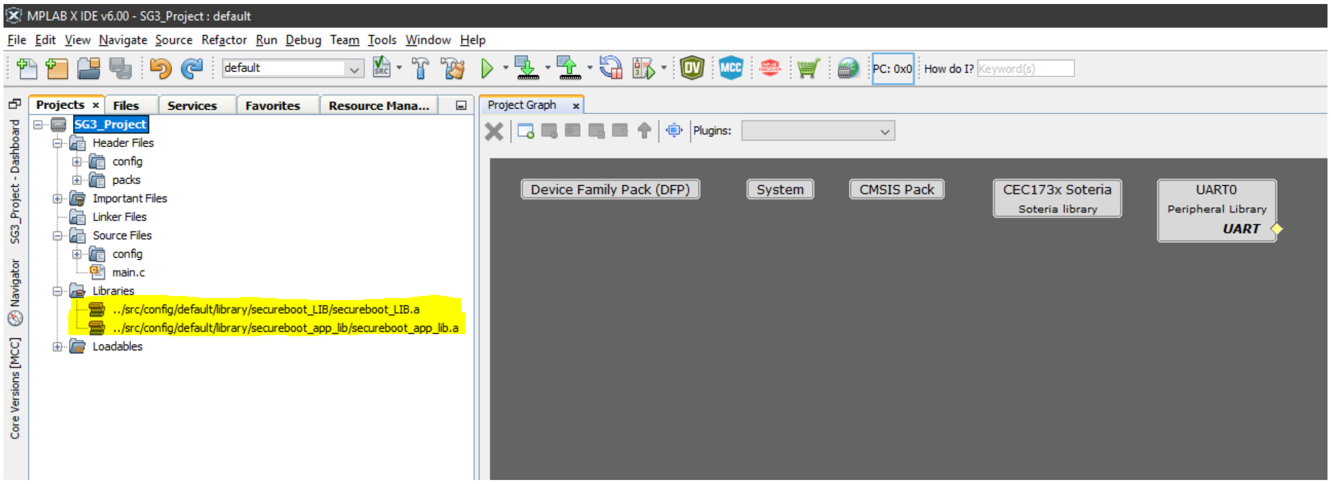
- System
  - Device & Project Configuration
    - CEC1736_S0_2ZW Device Configuration
    - Project Configuration
      - Generate C++ Project □
      - Generate Main Source File ☑
      - Generate System Source Files ☑
        - Generate System Definition ☑
        - Generate System Initialization ☑
        - Generate System Interrupts ☑
        - Generate LIBC Syscalls □
        - Generate System Startup □
        - Generate STDIO Syscalls □
        - Generate System Exception ☑
          - Use Advanced Exception Handling □
      - Tool Chain Selections
        - Compiler: XC32
        - Add linker file to project □
        - XC32 Global Options
        - KEIL Global Options
      - MISRA-C:2012
  - Cortex-M4 Configuration
  - GPIO
  - Clock
    - Processor Clock Divide: Divide 96 MHz clock by 1 (96 MHz Processor Clock)
    - Slow Clock Divide (Configures the 100KHz clock domain): 480
    - Calculated Clock Frequencies
  - Block Sleep Config
  - Block Privilege Config
  - Interrupts (NVIC)
  - DMA (DMAC)
    - Use DMA Service ? ☑
      - Generate low-level APIs only ☑
      - Interrupt Type: Aggregate
    - Use DMA Channel 0 ☑
    - Use DMA Channel 1 ☑
    - Use DMA Channel 2 ☑
    - Use DMA Channel 3 ☑
    - Use DMA Channel 4 ☑
    - Use DMA Channel 5 ☑

3.4 Code generation

1. Click on the ***"Generate"*** button located under ***"Project Resources"*** window and wait for the code generation to complete

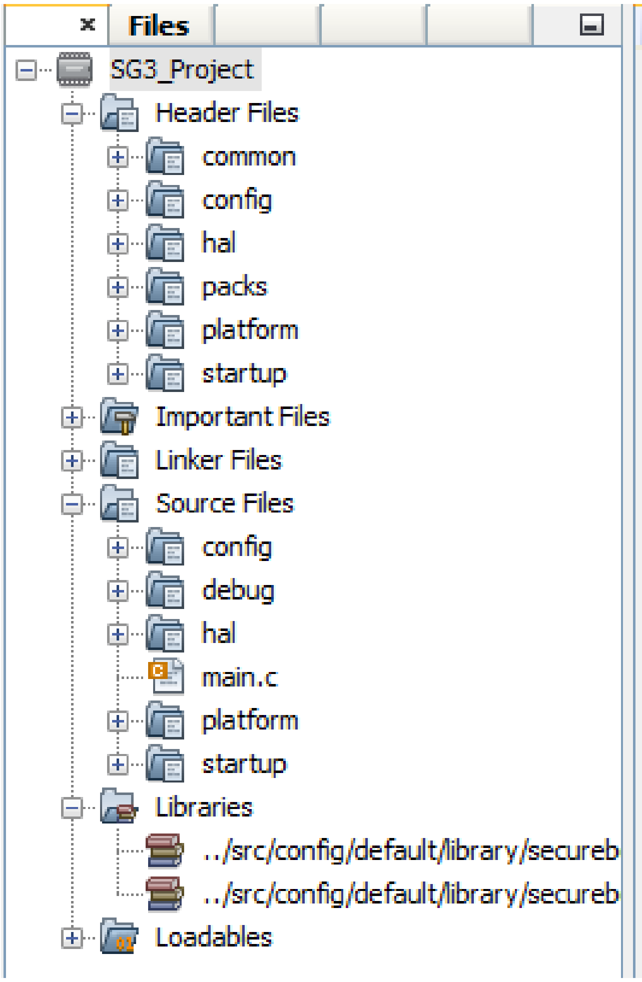


2. Once the code generation is complete, the Soteria can be located under the ***"Libraries"*** logical folder of the current project as shown below

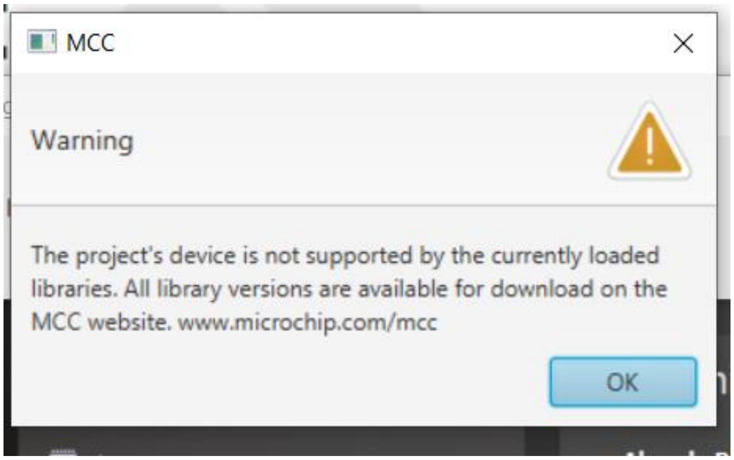


../src/config/default/library/secureboot_LIB/secureboot_LIB.a
../src/config/default/library/secureboot_app_lib/secureboot_app_lib.a

3. Once the code generation is complete, the project structure should look like the image below



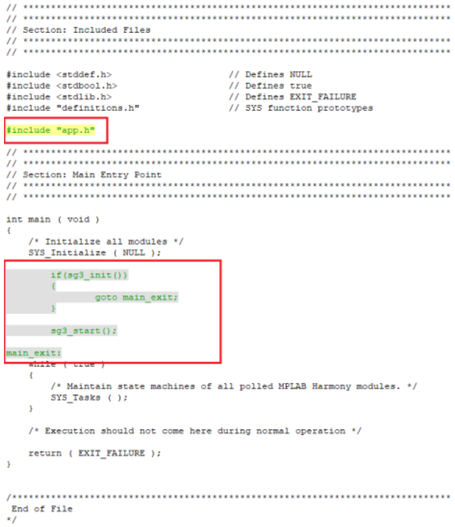
4. If you get the below error during the project creation process, then navigate to ***"Tools -> Options -> Plugins Tab -> MPLAB Code Configurator x.x"*** as shown in Step #2 under ***Section 4.1*** of this document and re-set the path to the Harmony Framework with the same value again



Warning

The project's device is not supported by the currently loaded libraries. All library versions are available for download on the MCC website. www.microchip.com/mcc

5. Include the file ***"common.h"*** in the ***main.c*** file of this project
6. To run the SG3 application, the application's main function should call the functions described in Section 6.1.2 as shown below



```
// *****************************************************************************
// *****************************************************************************
// Section: Included Files
// *****************************************************************************
// *****************************************************************************

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

#include "app.h"

// *****************************************************************************
// *****************************************************************************
// Section: Main Entry Point
// *****************************************************************************
// *****************************************************************************

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

        if(sg3_init())
        {
                goto main_exit;
        }

        sg3_start();

main_exit:
    while ( true )
    {
        /* Maintain state machines of all polled MPLAB Harmony modules. */
        SYS_Tasks ( );
    }

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

    return ( EXIT_FAILURE );
}


/*******************************************************************************
 End of File
*/
```

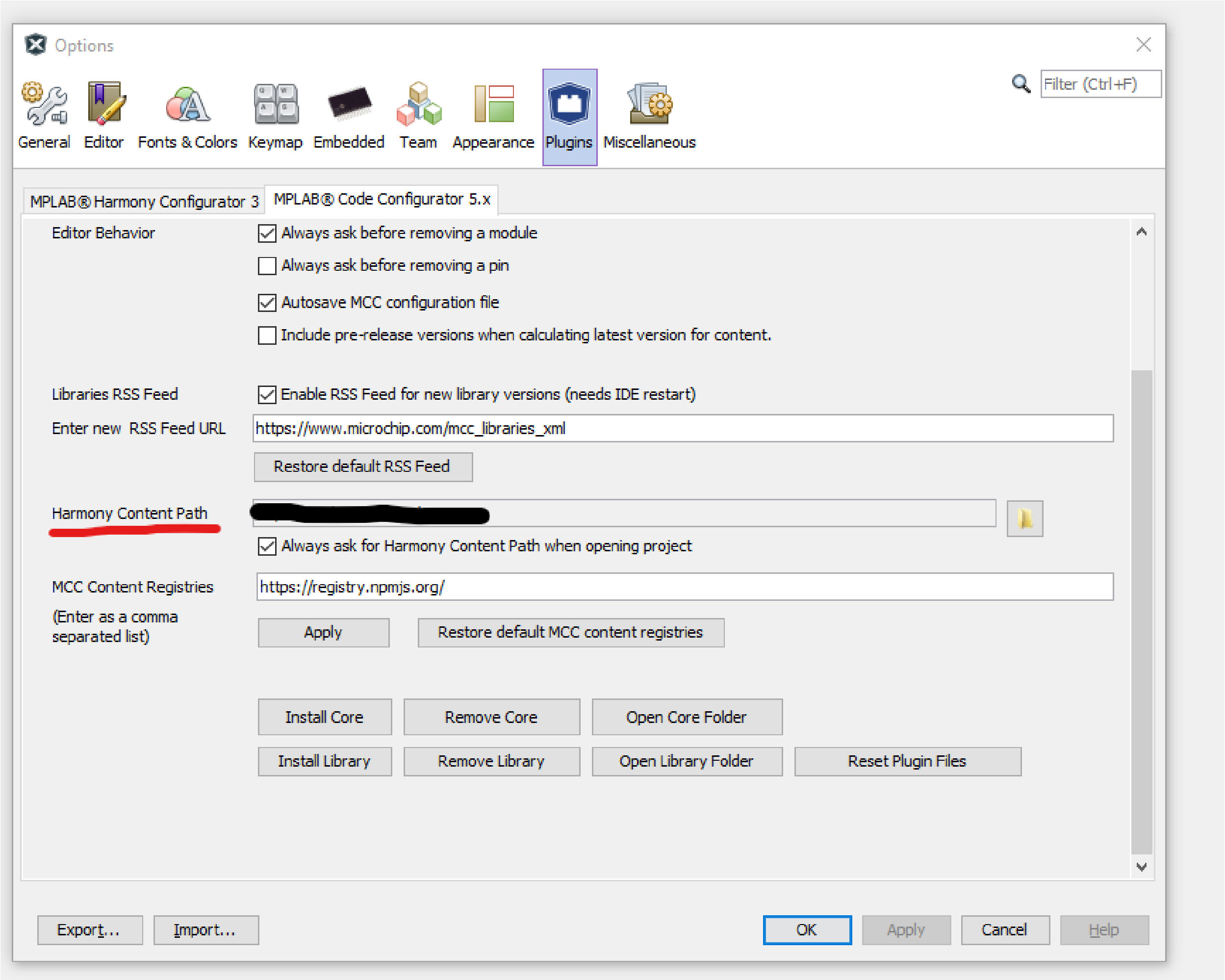
7. Refer to Section 6 and Section 8 to understand the usage of the available API functions and OEM tasks

4 Soteria-G3 sample library project

To ease the process of creating a Soteria-G3 project from scratch, a sample project has already been created, which can be found under “***HarmonyFrameworkPath/cec173x_soteria_lib/apps/sg3_h3_port/”***

4.1 Opening SG3 sample library project

1. From the MCC content manger, select the component ***“cec173x_soteria_lib”*** and download it
2. Locate the ***“MCC Content Path”*** by navigating to ***“Tools -> Options -> Plugins Tab -> MPLAB Code Configurator x.x”*** tab as shown below



Editor Behavior
☑ Always ask before removing a module
☐ Always ask before removing a pin
☑ Autosave MCC configuration file
☐ Include pre-release versions when calculating latest version for content.

Libraries RSS Feed ☑ Enable RSS Feed for new library versions (needs IDE restart)

Enter new RSS Feed URL https://www.microchip.com/mcc_libraries_xml

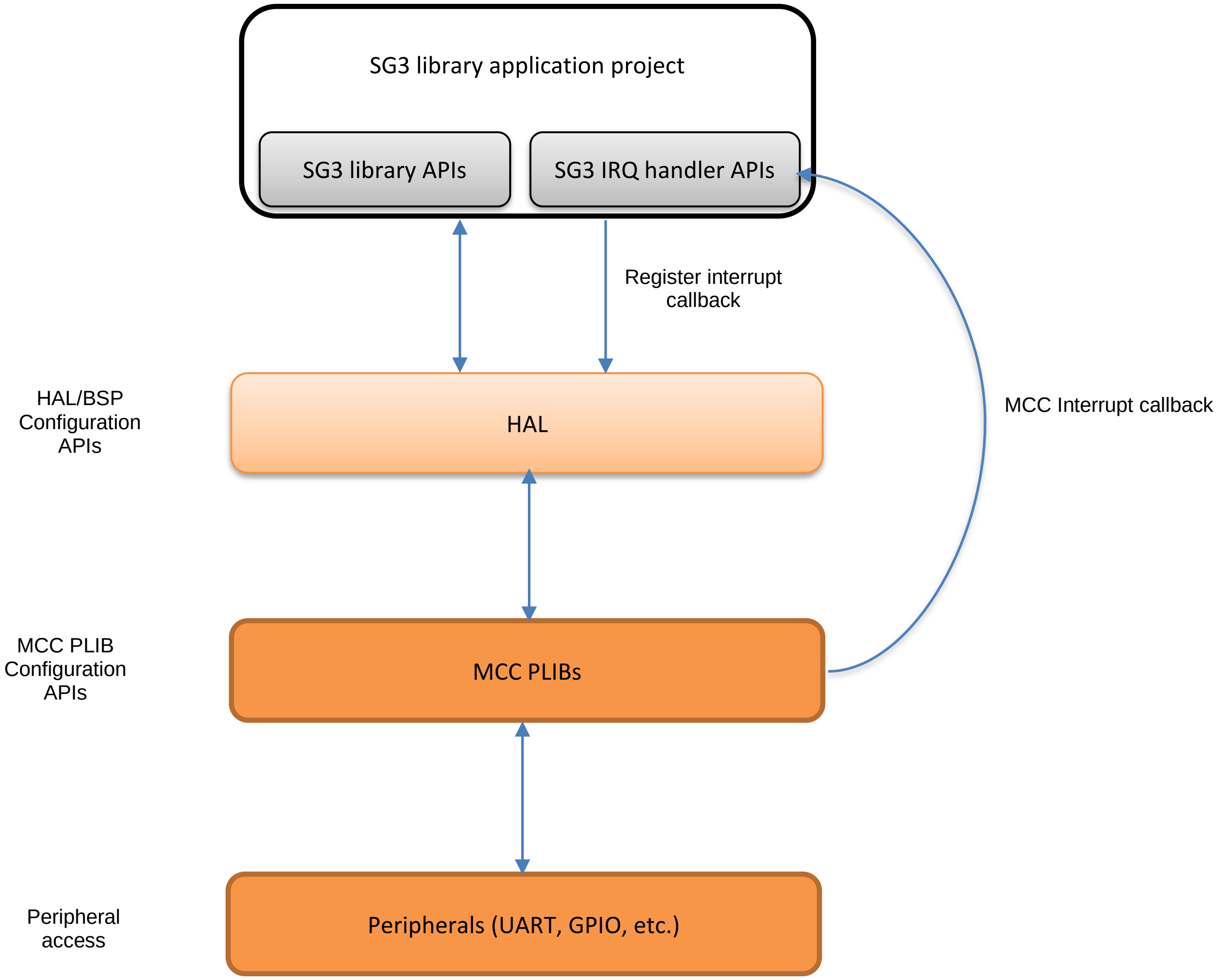
Harmony Content Path [redacted]
☑ Always ask for Harmony Content Path when opening project

MCC Content Registries (Enter as a comma separated list) https://registry.npmjs.org/

3. Navigate to this location to find the folder ***“cec173x_soteria_lib/apps/sg3_h3_port/firmware/”*** which contains the SG3 application project for this device
4. Open the ***“sg3_h3_port”*** sample application project in MPLABX

5. Users can get started with developing an application by using the application task functions of this project as mentioned in ***Section 8*** of this document

4.2 High level design


SG3 library application project
SG3 library APIs
SG3 IRQ handler APIs
Register interrupt callback
MCC Interrupt callback
HAL/BSP Configuration APIs
HAL
MCC PLIB Configuration APIs
MCC PLIBs
Peripheral access
Peripherals (UART, GPIO, etc.)

5 Soteria-G3 library project structure

common/debug/	APIs for UART debugging
common/include/	1. APIs for working with GPIO and ECIA blocks 2. Common file inclusions for use by application 3. Linker script
config/	MCC generated PLIB files
hal/	Hardware Abstraction Layer APIs (not to be used unless an API is not present in ahb_api_mpu.h)
kernel/	SG3 APIs for application use
oem/	Functions and definitions for adding user code
packs/	MCC generated device specific files (not for application use)
platform/	1. Application specific configurations 2. Interrupt handling routines
startup/	Device startup file

6 Soteria-G3 library APIs

6.1.1 UART debugging

6.1.1.1 Formatted printing to UART

Function prototype:

```
void tracex(const char *fmt, ...);
```

Description:

The function usage is like the ***printf*** function of stdio

Inputs:

Same as ***printf*** function of stdio

Outputs:

None

6.1.1.2 ISR safe formatted printing to UART

Function prototype:

```
void tracex_from_ISR(const char *fmt, ...);
```

Description:

This function is an ISR safe equivalent of ***tracex***

Inputs:

Same as **printf** function of stdio

Outputs:

None

6.1.1.3 Hex dump to UART

Function prototype:

```
void print_buf(uint8_t *buf, uint32_t len);
```

Description:

Prints hexadecimal values inside a buffer of user defined length

Inputs:

Input Parameter	Description
buf	Pointer to a user defined allocated buffer which contains
len	Length of the user defined allocated buffer

Outputs:

None

6.1.2 Soteria-G3 specific APIs

6.1.2.1 Soteria-G3 firmware initialization

Function prototype:

```
int sg3_init(void);
```

Description:

Initializes the Soteria-G3 firmware application

Inputs:

None

Outputs:

Input Parameter	Description
0	Soteria-G3 initialization succeeded
-1	Soteria-G3 initialization failed

6.1.2.2 Start Soteria-G3 firmware operation

Function prototype:

```
void sg3_start(void);
```

Description:

Runs the Soteria-G3 firmware application

Note:

Inputs:

None

Outputs:

None

6.1.3 Peripheral access

To access peripherals from OEM functions, please refer to the API functions provided in file “***ahb_api_mpu.h”*** located under “***cec173x_soteria_lib /apps/sg3_h3_port”*** sample SG3 project. SG3 design constraints does not allow accessing these peripherals directly using MCC generated APIs.

6.1.4 ROM API access

The OEM tasks, oem1, oem2, oem3 can access ROM APIs listed in ***“rom_api_mpu.h”*** located under “***cec173x_soteria_lib/apps/sg3_h3_port***”.

To access ROM APIs, based on task add ROM API permission by calling “***oem_task1_rom_api_permission_add***”, ***“oem_task2_rom_api_permission_add”***, ***“oem_task3_rom_api_permission_add”,*** with argument having the bit set corresponding to the ROM API, refer ***“di_permissions_rom_apis_tbl0”***, ***“di_permissions_rom_apis_tbl_1”.***

Each task can access only 8 ROM APIs listed.

Example:

OEM1 can add permission to access table 0 ROM APIs eg : BROM version by calling:

oem_task1_rom_api_permission_add(DI_PERMISSION_ROM_API_ROM_VER_TBL0);

Followed by:

MPU_API_rom_ver();

Similarly, one another new function call is required to access table 1 ROM APIs eg: efuse byte read, Efuse write byte after which the operation can be performed.

oem_task1_rom_api_permission_add(DI_PERMISSION_ROM_API_EFUSE_BYTE_READ_TBL1 | *DI_PERMISSION_ROM_API_EFUSE_BYTE_WRITE_TBL1);*

Followed by:

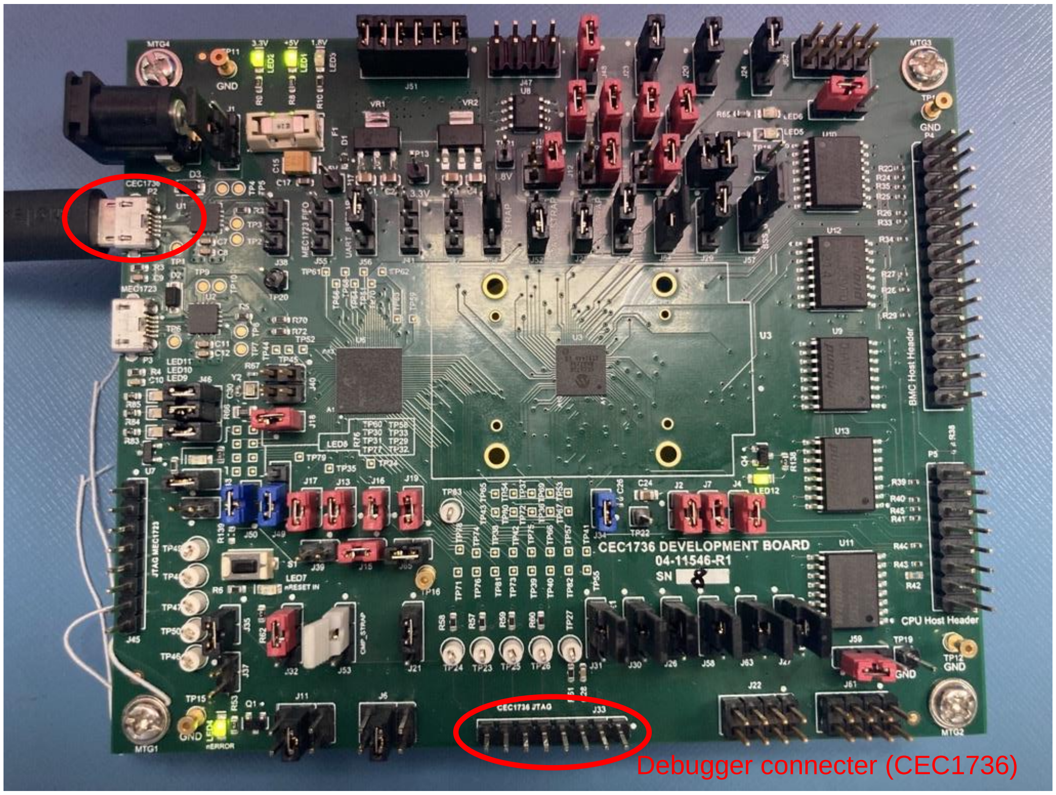
MPU_API_efuse_byte_read(byte_index, out_data);

MPU_API_efuse_byte_write(byte_index, out_data);

7 Soteria user interaction and feedback

7.1 Debugging

1. Connect a micro-USB cable to the P2 connector on the development board
2. Connect the debugger to the J33 connector on the development board



Power and serial port connector (CEC1736)

Debugger connecter (CEC1736)

3. Open the ***“sg3_h3_port”*** sample Soteria project using MPLABX IDE (Refer ***Section 4.1***)
4. Clean and build the project by selecting ***“Clean and Build”*** option from the project context menu
5. Start a debug session of this project by selecting the ***“Debug”*** option from the project context menu
6. Click on “Run” from the “Debug” context menu
7. Open “PuTTY” or any other serial port application with the following settings
    a. Baud rate: 115200
    b. Stop bits: 1
    c. Flow control: Off
    d. Parity: None
8. The UART output from SG3 can be observed on the serial port application

7.2 On board LEDs

State	Observation
Authenticating AP images	Blink rate = 2Hz Pattern = None
Authentication completed and no error detected	Blink rate = 0.5Hz Pattern = None
Authentication completed and non-fatal error detected	Blink rate = 1Hz Pattern = 2
Authentication completed and fatal error detected	Blink rate = 1Hz Pattern = 1
Executing recovery sequence	Blink rate = 4Hz Pattern = None
Authentication completed post recovery and no error detected	Blink rate = 1Hz Pattern = None

LED12 behavior

State	AP0 critical image	AP1 critical image	LED5	LED6
Authenticating AP images	No failure	No failure	Off	Off
	Image failure	No failure	Blink rate = 1Hz Pattern = None	Off
	No failure	Image failure	Off	Blink rate = 1Hz Pattern = None
	Image failure	Image failure	Blink rate = 1Hz Pattern = None	Blink rate = 1Hz Pattern = None

Executing recovery sequence	Recover image	No recovery	Blink rate = 4Hz Pattern = None	Off
	No recovery	Recover image	Off	Blink rate = 4Hz Pattern = None
	Recover image	Recover image	Blink rate = 4Hz Pattern = None	Blink rate = 4Hz Pattern = None
Authentication completed and error detected	Non-fatal error	No failure	Blink rate = 1Hz Pattern = None	Off
	No Failure	Non-fatal error	Off	Blink rate = 1Hz Pattern = None
	Non-fatal error	Non-fatal error	Blink rate = 1Hz Pattern = None	Blink rate = 1Hz Pattern = None
	No failure	Fatal error	Off	Blink rate = 1Hz Pattern = 2
	Non-fatal error	Fatal error	Blink rate = 1Hz Pattern = None	Blink rate = 1Hz Pattern = 2
	Fatal error	X	Blink rate = 1Hz Pattern = 1	Blink rate = 1Hz Pattern = 1

Authentication completed and no error detected	Pass	Pass	Off	Off
Authentication completed post recovery	Image recovered	No image recovered	Blink rate = 1Hz Pattern = None	Off
	No image recovered	Image recovered	Off	Blink rate = 1Hz Pattern = None
	Image recovered	Image recovered	Blink rate = 1Hz Pattern = None	Blink rate = 1Hz Pattern = None

LED5 and LED6 behavior

Blink patterns:

1. Blink – Blink – Off – Off <repeat>
2. Blink – Off – Off <repeat>

8 Application tasks for debugging

Soteria provides OEM task functions for user to play around with various features of the application project.

There are three functions provided to the user to get started with Soteria.

- oem_task1_function ()
- oem_task2_function ()
- oem_task3_function ()

The user can add his own code inside these functions to evaluate the capabilities and features of Soteria and CEC173x secure-boot controller.

Please refer to the sample Soteria application project present in "***cec173x_soteria_lib/apps/sg3_h3_port"*** for reference. The OEM task functions can be located under "***src/oem/oem_task1"***, "***src/oem/oem_task2"*** and "***src/oem/oem_task3"*** directories.

9 Revision History

Name	Revision Level	Date	Section	Remarks
Shreyas Kannan	0.1	March 29, 2022	1	Initial draft
Shreyas Kannan	0.2	March 30, 2022	2, 3, 4, 5, 6	Updated
Shreyas Kannan	0.3	April 1, 2022	2, 3, 4, 5, 6	Updated
Shreyas Kannan	0.4	April 5, 2022	1.4, 1.6, 2, 4, 5	Updated
Shreyas Kannan	0.5	April 6, 2022	6.2	Updated
Shreyas Kannan	0.6	April 7, 2022	4, 7	Updated
Shreyas Kannan	1.0	May 18, 2022	3, 4, 6, 8	Updated
Shreyas Kannan	1.1	Jan 03, 2023	3,4,6	Updated
Shreyas Kannan	1.2	Jan 31, 2023	3	Updated