Getting started with MCC and Soteria-G3

User guide

Rev 0.6	April 7, 2022

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 v6.00 or higher
DFP	v1.5.139 or higher
Debugger (only in case of debugging)	ICD4 or PICKit4
Compiler	XC32 v2.50
Board	CEC1736 development board with, 1. CEC1736 internal flash pre-programmed binary 2. External flash modules with pre-programmed AP_FW binaries

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
OEM	Original Equipment Manufacturer
AP	Application Processor
SG3	Soteria Generation 3
MCC	Microchip Code Configurator
EC_FW	Embedded Controller Firmware
SPI	Serial Peripheral Interface
CoT	Chain Of Trust
HAL	Hardware Abstraction Layer
PLIB	Peripheral LIBrary
API	Application Programming Interface
GPIO	General Purpose Input Output
ECIA	Embedded Controller Interrupt Aggregator
IRQ	Interrupt ReQuest
BSP	Board Support Package
UART	Universal Asynchronous Receiver and Transmitter
Hex	Hexadecimal

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 and enforce a secure boot process in the system.

Soteria-G3 uses the CEC173x immutable secure bootloader, implemented in ROM, as the system Root-of-Trust (RoT). The CEC173x secure bootloader loads, decrypts and authenticates the embedded controller firmware (EC_FW) from the external (or) internal SPI Flash. The validated EC_FW 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 OEM. It offers security features to authenticate the SPI Flash image in the external SPI flash device.

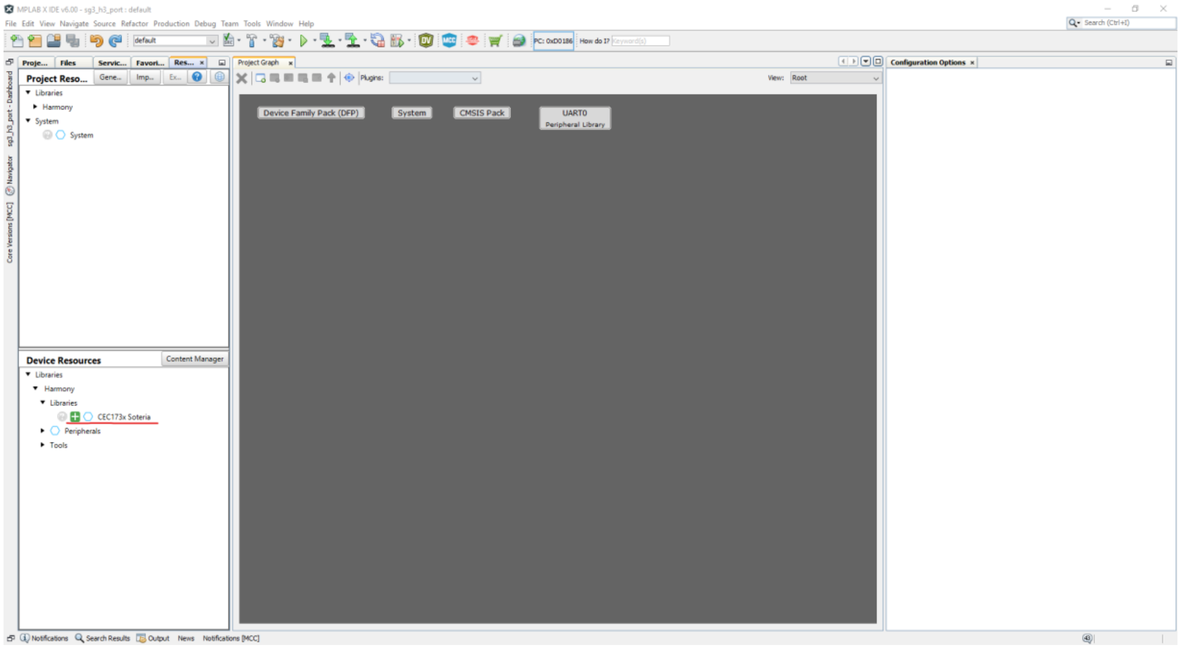
The validated AP_FW that runs on the application processor can utilize crypto resources in the CEC173x to authenticate other code in the system, thereby extending the Chain-of-Trust (CoT) to ensure that all code running in the system is authorized.

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

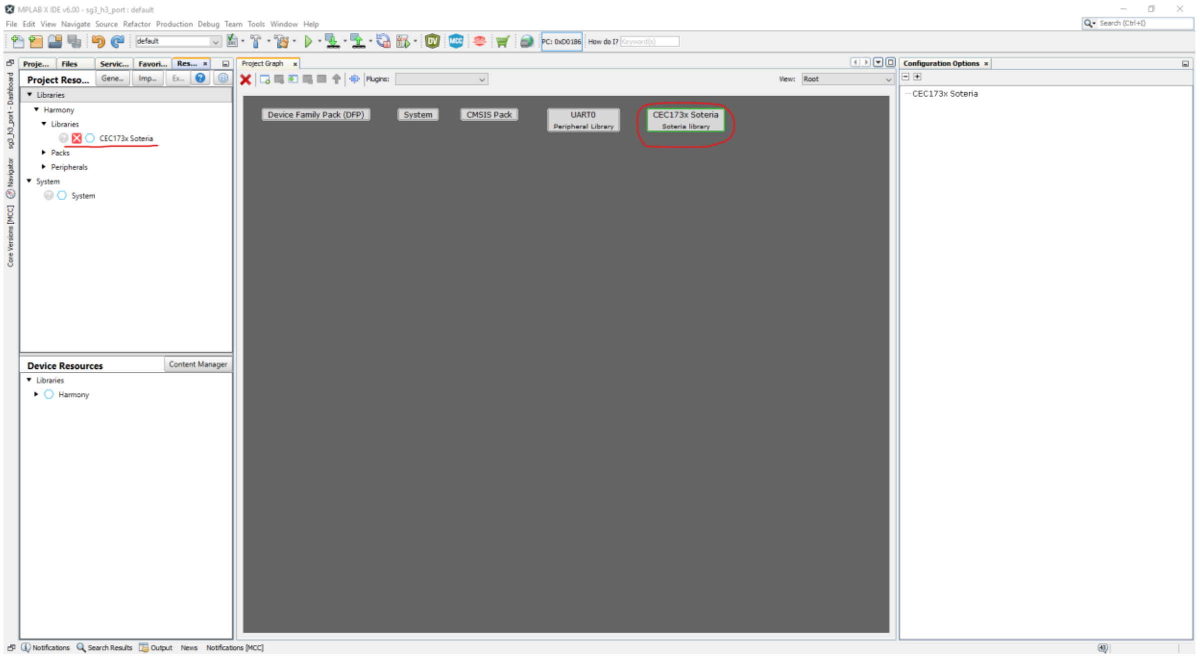
3 Setting up an MCC project with Soteria library

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

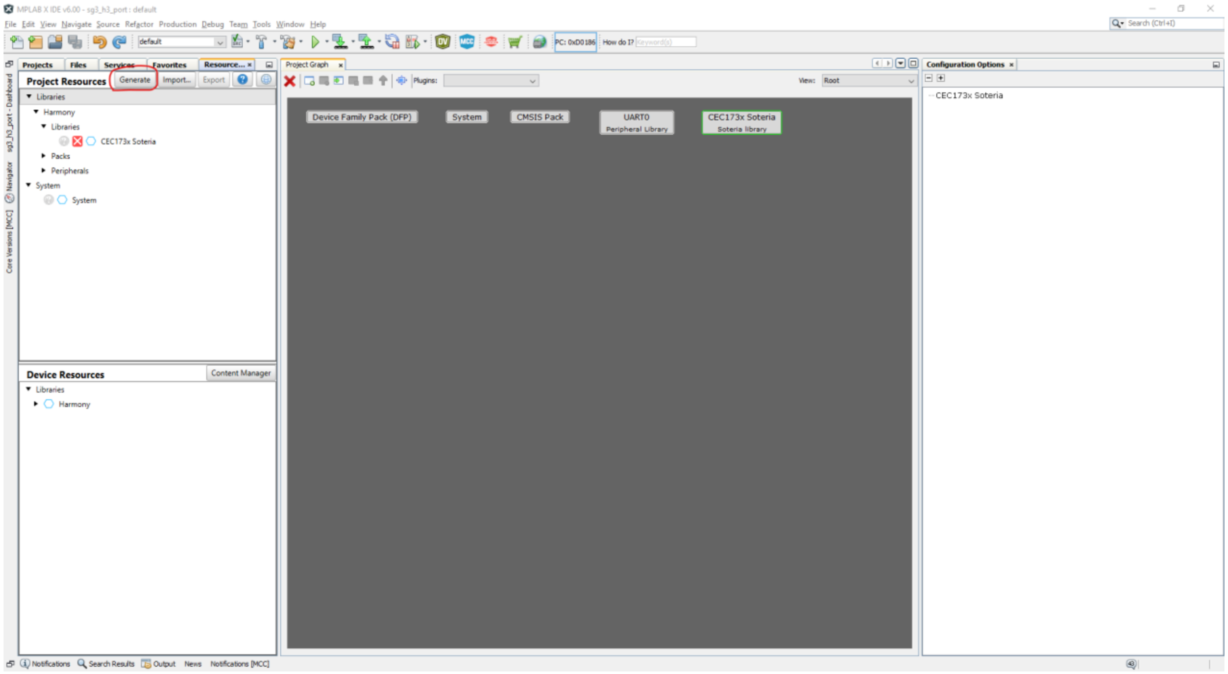
3. To add Soteria as a library into the created application project, ***"double click"*** on ***"CEC173x Soteria"*** componenet which can be found under ***"Libraries → Harmony → Libraries → CEC173x Soteria"*** under ***"Device Resources"*** window as shown below



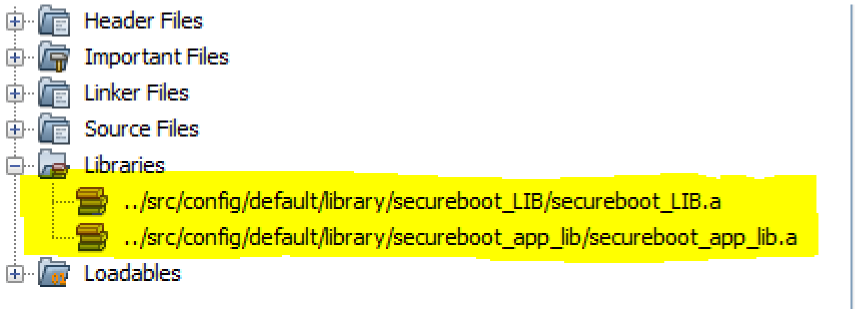
4. The Soteria library component should get added in the ***"Project Graph"*** and ***"Project Resources"*** as shown below



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



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

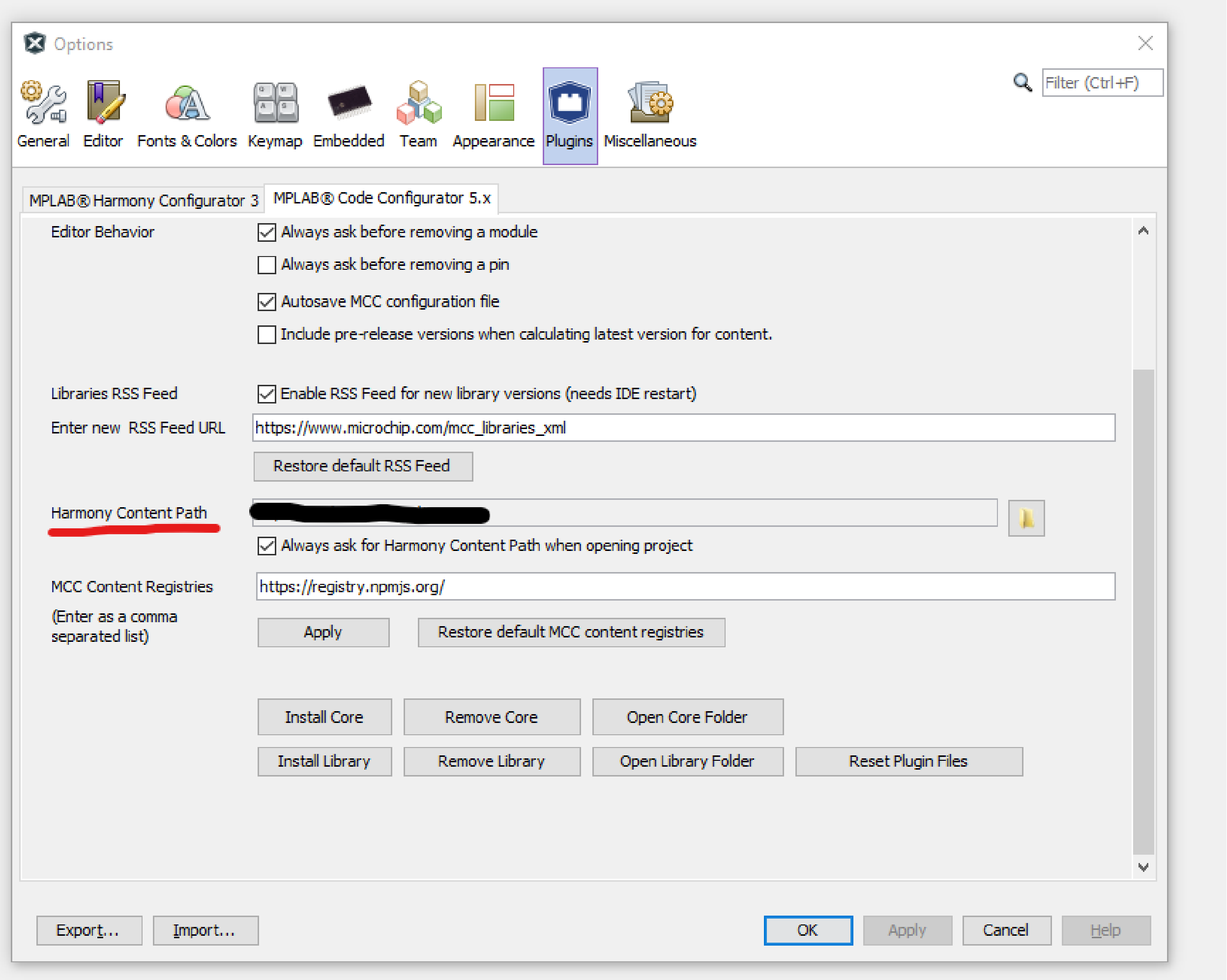


Header Files
Important Files
Linker Files
Source Files
Libraries
../src/config/default/library/secureboot_LIB/secureboot_LIB.a
../src/config/default/library/secureboot_app_lib/secureboot_app_lib.a
Loadables

4 Soteria-G3 sample library project

4.1 Opening SG3 sample library project

1. From the MCC content manger, locate the component ***“core_apps_cec173x”*** and select it for download
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)

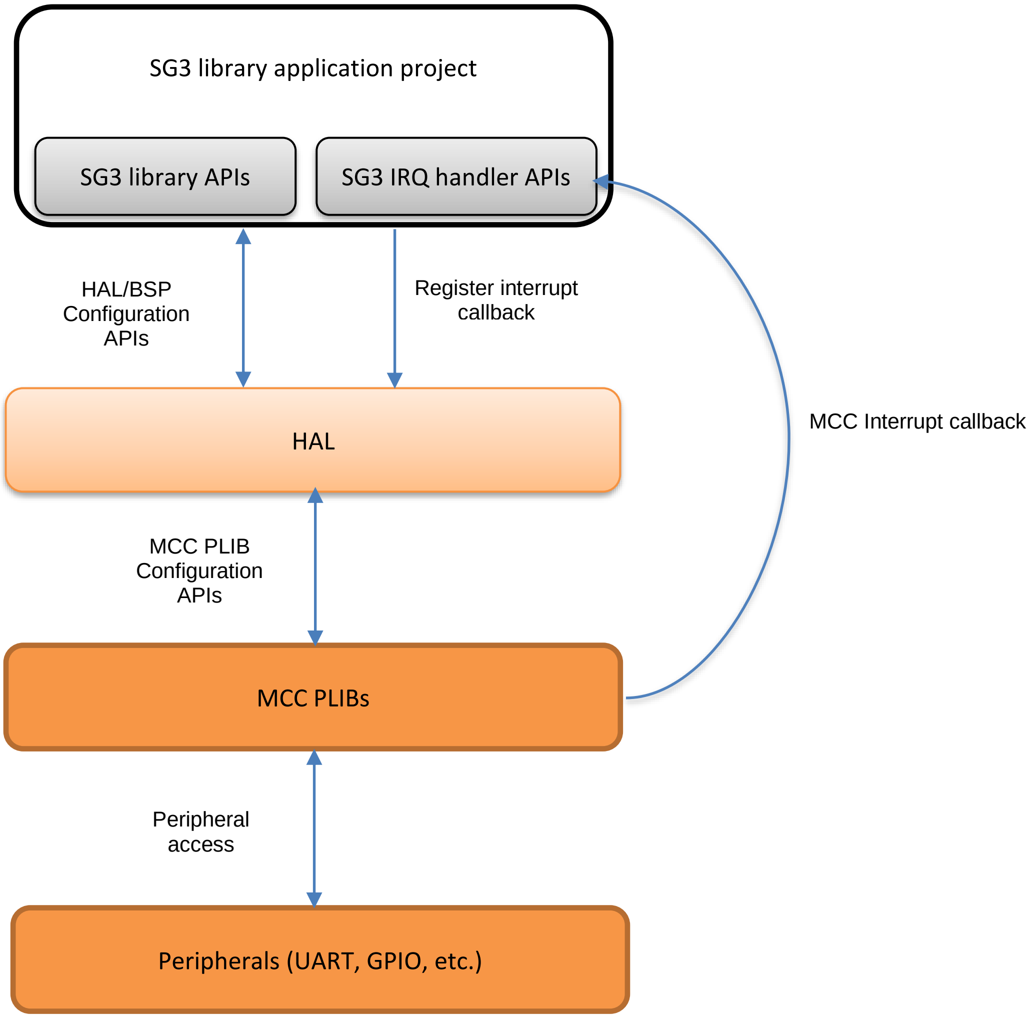
Enter new RSS Feed URL https://www.microchip.com/mcc_libraries_xml

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 ***“core_apps_cec173x/apps/”*** which contains all sample applications 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 as mentioned in ***Section 8*** of this document

4.2 High level design


SG3 library application project
SG3 library APIs
SG3 IRQ handler APIs
HAL/BSP Configuration APIs
Register interrupt callback
HAL
MCC Interrupt callback
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

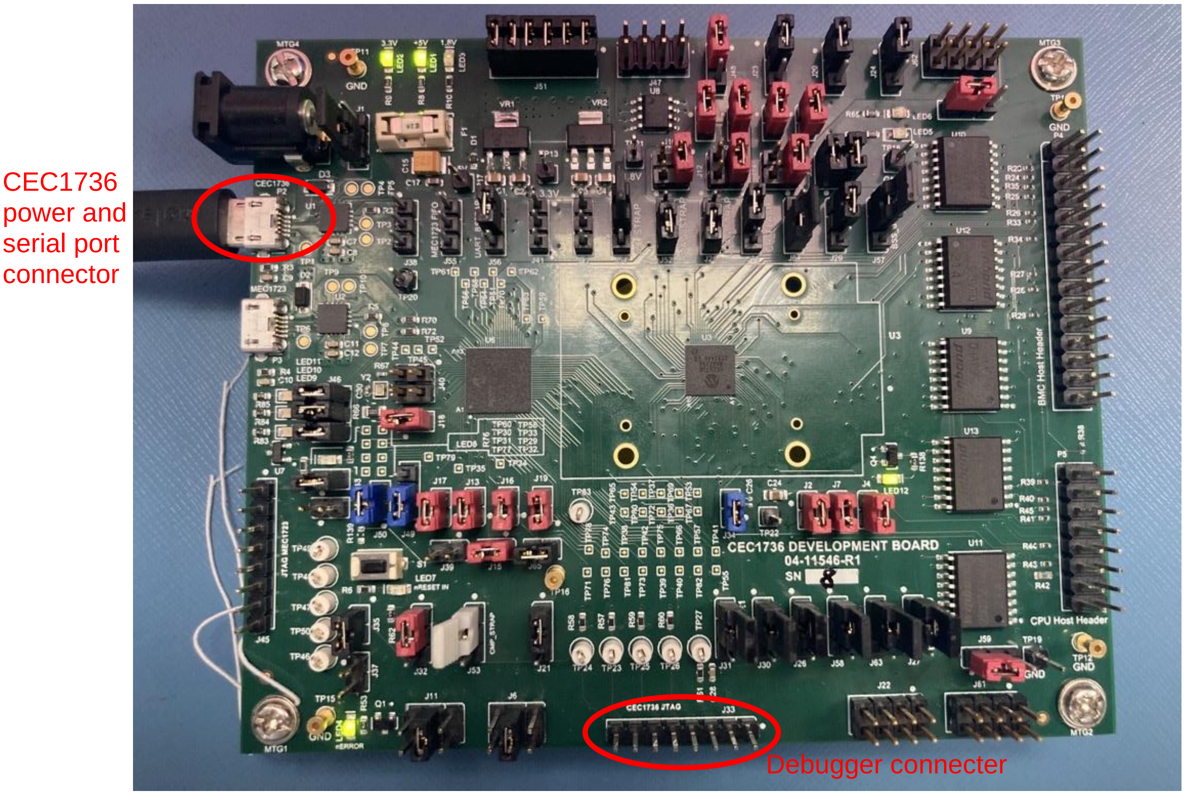
6.1.3 GPIO and ECIA peripheral access

To configure the GPIO and ECIA peripherals from OEM functions, please refer to the file ***ahb_api_mpu.h*** present in ***core_apps_cec173x/apps/sg3_h3_port*** sample SG3 project. Accessing these peripherals directly using MCC generated APIs is not allowed because of software design constraints.

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


CEC1736 power and serial port connector
Debugger connecter
CEC1736 DEVELOPMENT BOARD
04-11546-R1
CEC1736 JTAG
J33

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 ***core_apps_cec173x/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