Application Note

MSPM0 Design Flow Guide

Eason Zhou, Zoey Wei, Helic Chi, and Janz Bai

ABSTRACT

The application note details steps on how to develop MSPM0 MCUs. Related materials and instructions are provided.

Table of Contents

List of Figures

List of Tables

Trademarks

LaunchPad™, Code Composer Studio™, SimpleLink™, C2000™, and TIVA™ are trademarks of Texas Instruments.
Arm® and Cortex® are registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere.
All trademarks are the property of their respective owners.

1 Overview

The document provides steps for projects to develop an MSPM0 MCU. A list of related documents and step-by-step instructions are provided. For common questions that developers can encounter, see Section 7.

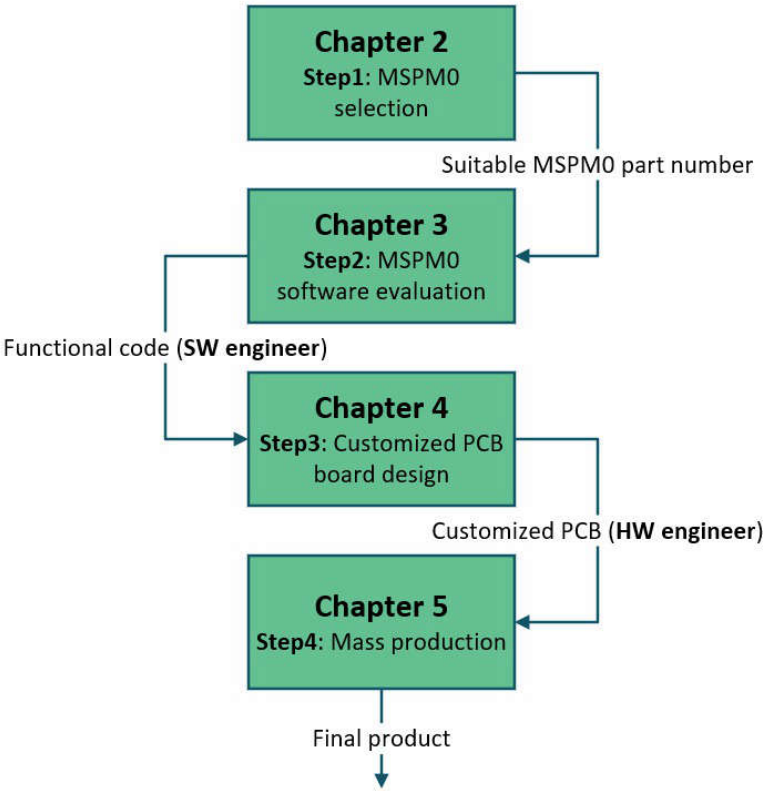


Figure 1-1. MSPM0 Design Flow

2 Step 1: MSPM0 Selection

This step discusses how to find an MSPM0 orderable number.

Visit the Arm Cortex-M0+ MCUs product page to view the list of MSPM0 devices. After navigating to this page, use the filters on the left to perform an initial screening based on MCU peripheral requirements, or directly navigate to the device page using the search box on the left side of the page.

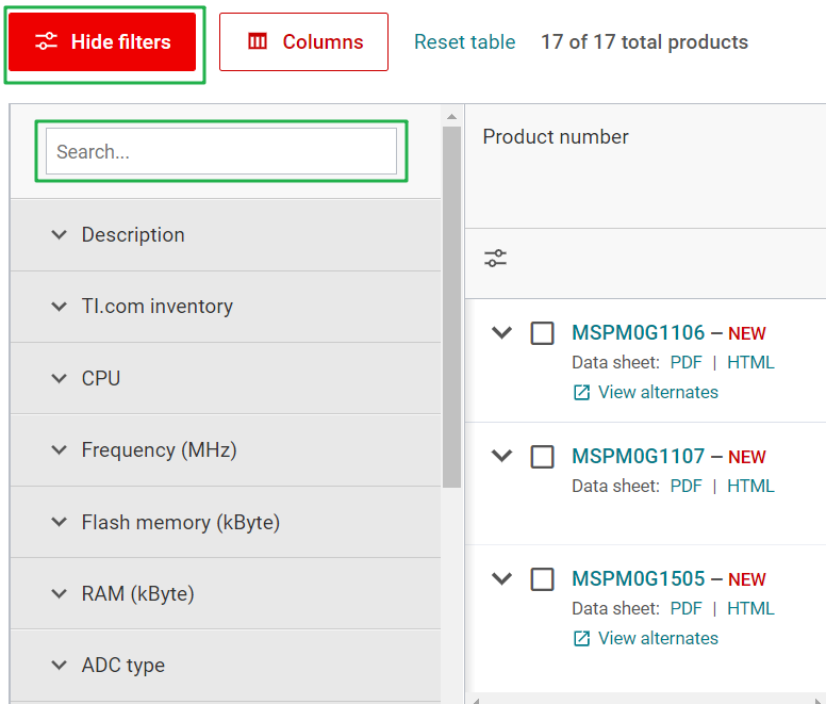


Figure 2-1. MSPM0 Device List

After navigating to the device page, more specification or functional details for a specific product are available. The key documents are the data sheet, technical reference manual (TRM), and errata. The device-specific data sheet introduces the parameters and functional data information for the MSPM0. The device-specific TRM introduces the application method and characteristics of a MSPM0 device. The device-specific errata shows descriptions of MSPM0 related series or versions.

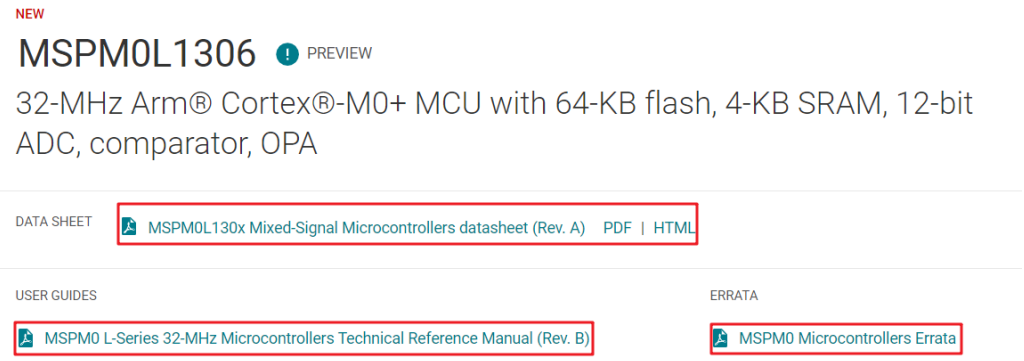


Figure 2-2. MSPM0 Important Document List

Figure 2-3 shows the *Device Comparison* table in a device-specific data sheet. A user can compare different part numbers using this table.

5 Device Comparison

Table 5-1. Device Comparison

DEVICE NAME (1) (2)	FLASH / SRAM (KB)	QUAL(3)	ADC CH.	COMP	OPA	GPAMP	UART/I2C/SPI	TIMG	GPIOs	5-V TOL. IO	PACKAGE [BODY SIZE] (4)
MSPM0L1306xRHB	64 / 4	T/S	10	1	2	1	2 / 2 / 1	4	28	2	32 VQFN [5 mm × 5 mm] (5)
MSM0L1305xRHB	32 / 4										
MSM0L1304xRHB	16 / 2										
MSPM0L1306xDGS28	64 / 4	T/S	10	1	2	1	2 / 2 / 1	4	24	2	28 VSSOP [7.1 mm × 3 mm]
MSPM0L1305xDGS28	32 / 4										
MSPM0L1304xDGS28	16 / 2										
MSPM0L1346xDGS28	64 / 4	T	9						22		
MSPM0L1345xDGS28	32 / 4										

Figure 2-3. Device Comparison Table

See the *Ordering and Quality* page on the device page to view the orderable part number and the reference price.

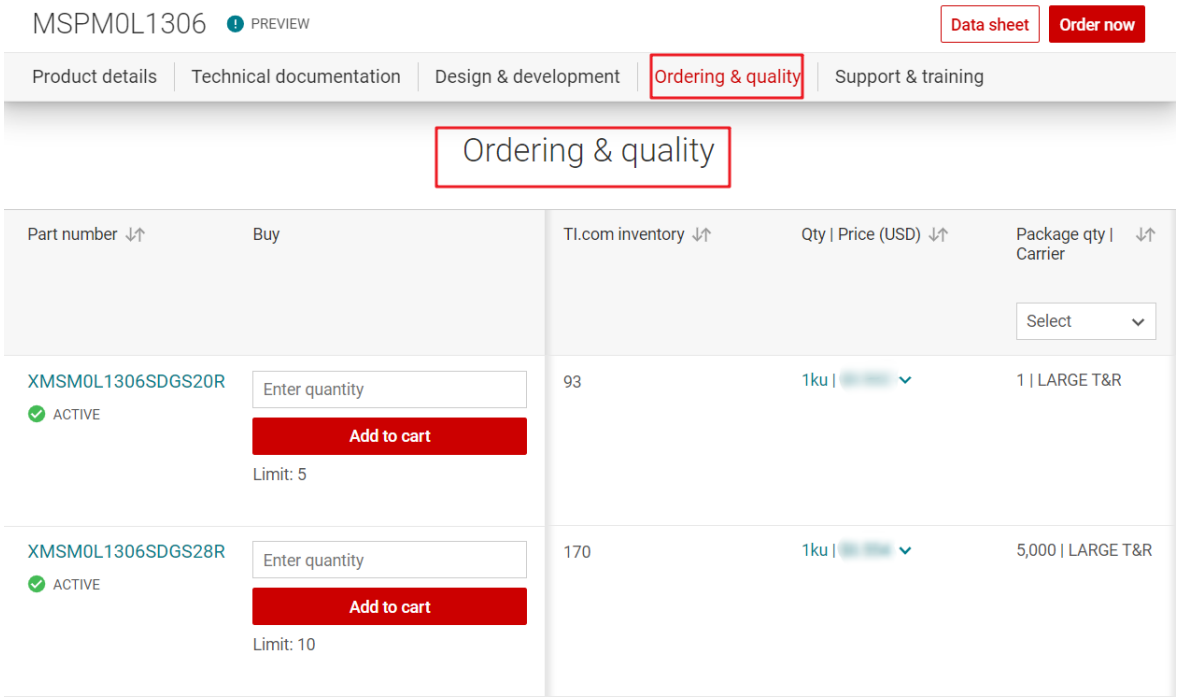


Figure 2-4. Ordering and Quality Part View

3 Step 2: MSPM0 Evaluation

This step shows how to set up a hardware and software evaluation environment for MSPM0. For step-by-step instructions based on CCS and LaunchPad, see Section 8.1.

Table 3-1 lists a summary of all the required components in an MSPM0 development chain. Devices are described individually in the following sections.

Table 3-1. MSPM0 Development Chain

IDE	SysConfig (Code Generator GUI)	SDK	Debugger	Hardware
CCS with SysConfig integrated		MSPM0 SDK	Launchpad with XDS110 On-Board	
Keil	Standalone SysConfig		XDS110	Customized board
IAR			J-Link	

3.1 Hardware Setup

3.1.1 Debugger Selection

This section summarizes different debuggers that support MSPM0 devices. The XDS110 debuggers are owned by TI, which support more functions, as compared to general debuggers. For more details about XDS110 debuggers, see Section 5.3.

Table 3-2. MSPM0 Debugger Comparison

Features	XDS110 (TMDSEMU110-U)	XDS110 On-Board	J-Link
cJTAG (SBW)	√	√	√
BSL tool	√	√	
Backchannel UART	√	√	
Power supply	1.8 - 3.6V	3.3 - 5V	5V
IDE	CCS, IAR, Keil	CCS, IAR, Keil	CCS, IAR, Keil

3.1.2 LaunchPad Introduction

TI recommends to start MSPM0 development with LaunchPad™. Figure 3-1 shows an overview of the LaunchPad. The LaunchPad contains the MCU and a XDS110 debugger. A user can use a debugger such as a J-Link to debug the MCU after removing the jumpers.

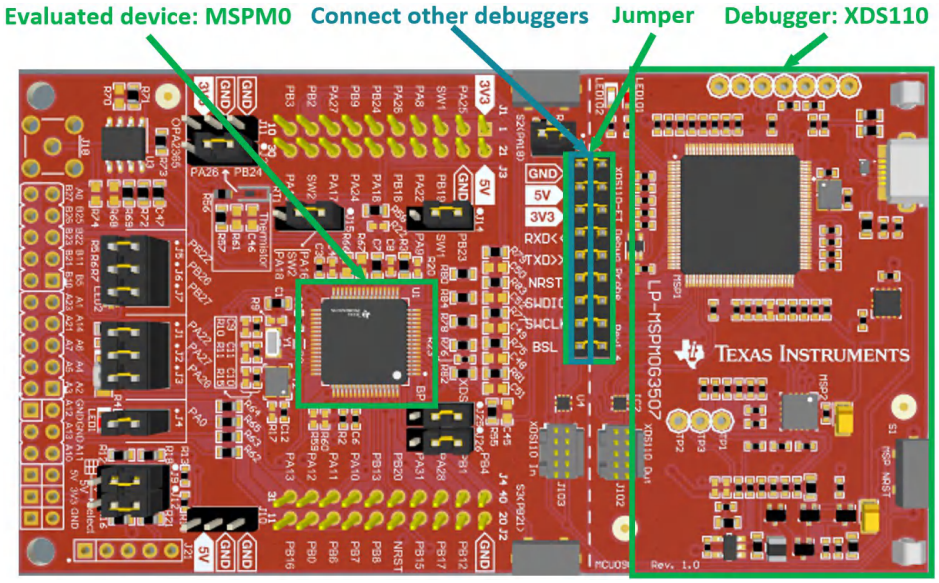


Figure 3-1. MSPM0G3507 LaunchPad

The following links show orderable LaunchPad devices and related user's guides.

- LP-MSPM0L1306 landing page
- LP-MSPM0G3507 landing page
- LP-MSPM0C1104 landing page

- *MSPM0L1306 LaunchPad Development Kit User's Guide*
- *MSPM0G3507 LaunchPad Development Kit User's Guide*
- *MSPM0C1104 LaunchPad Development Kit User's Guide*

A real launchpad setup condition is shown in Figure 3-1, which can be debugged and powered with a USB port.

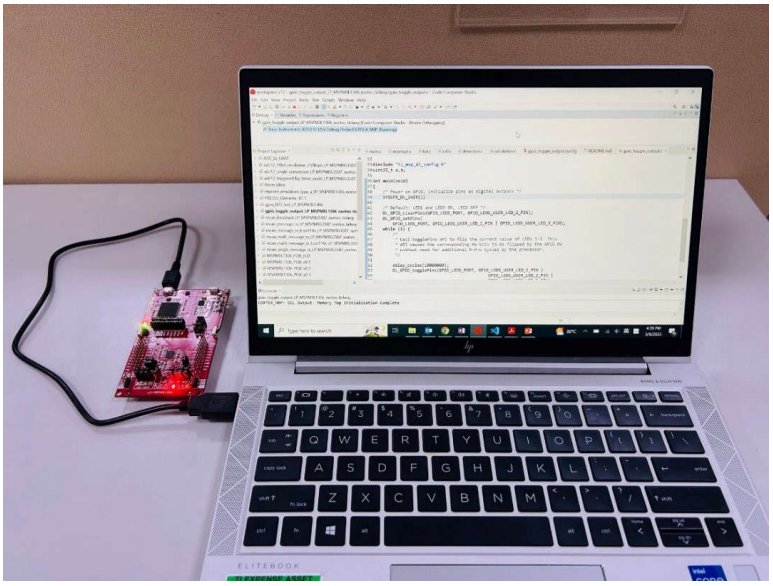


Figure 3-2. Launchpad Setup View

3.2 MSPM0-SDK Setup

The MSPM0-SDK provides the ultimate collection of software, tools, and documentation to accelerate the development of applications for the MSPM0 MCU platform, providing a consistent and cohesive experience with a wide variety of drivers, libraries, and examples under a single software package.

3.2.1 MSPM0-SDK Installation

This section details steps to install MSPM0-SDK. After installation, the default SDK directory path is: *C:\ti\mspm0_sdk_x_xx_xx_xx*.

1. Before downloading, a myTI account is required. Register for a myTI account here.
2. Download the latest MSPM0-SDK from the product page. Click *Download options*, select the operating system, and click the file name to start downloading.

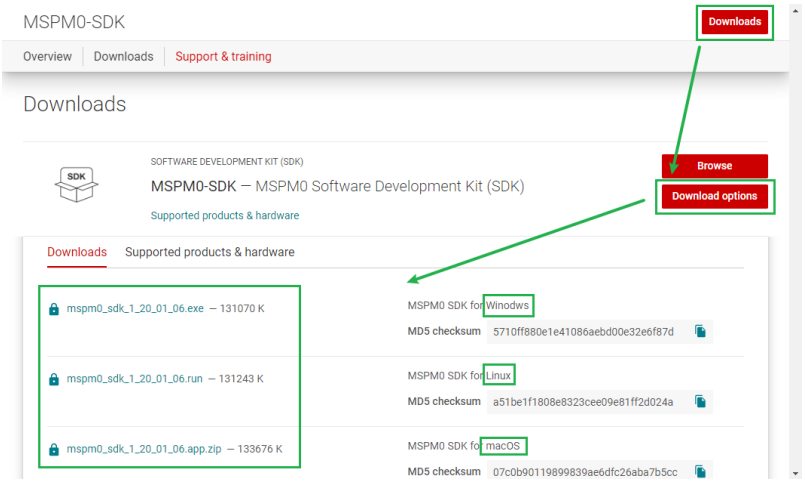


Figure 3-3. MSPM0-SDK Download

3. After downloading, follow the steps in Figure 3-4 to finish installation.

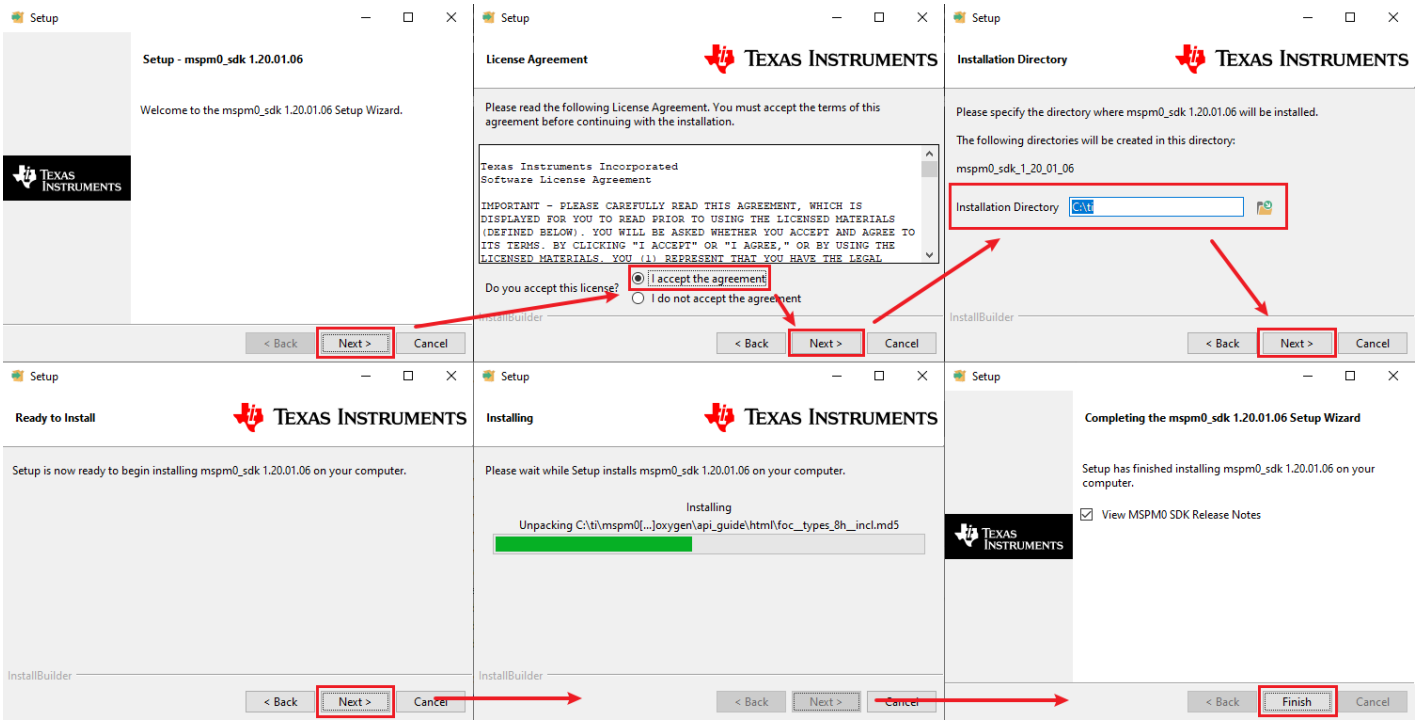


Figure 3-4. MSPM0-SDK Install Step-by-Step

3.2.2 MSPM0-SDK Introduction

There are five folders in the SDK install directory, listed in Figure 3-5. This section provides a brief introduction to all the folders.

- Docs folder: Contains all the documentation for SDK.
- Examples folder: Contains all the examples for reference, which can be used to provide a reference and starting point to accelerate application development. For more details, see the MSPM0-SDK Example Guide.
- Kernel folder: Built files for RTOS and nortos, which is included in the example project and accelerates the speed of the project build.
- Source folder: Contains all the source code for TI and third party libraries.
- Tool folder: Contains all the tools related to SDK, such as sysconfig support files, BSL GUI, and metrology GUI.

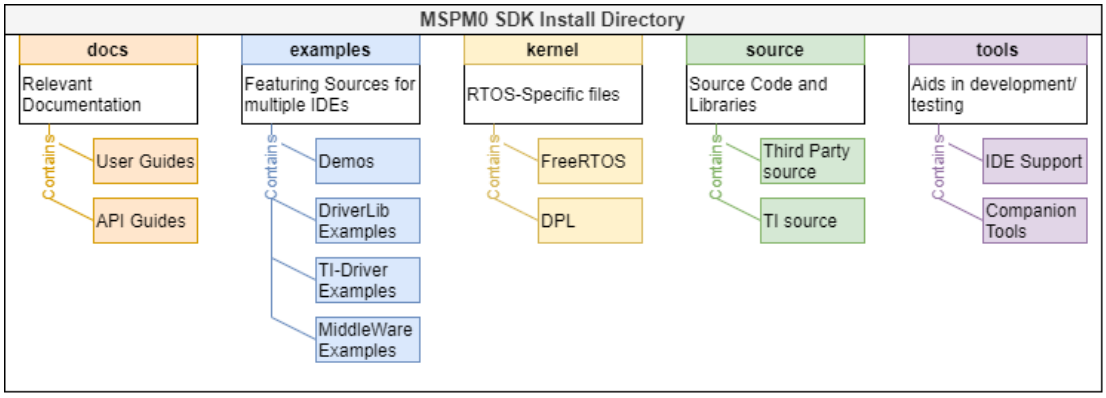


Figure 3-5. MSPM0-SDK Structure

The most important folders are example and document folders. Figure 3-6 shows the related addresses of the folders under the MSPM0-SDK directory.

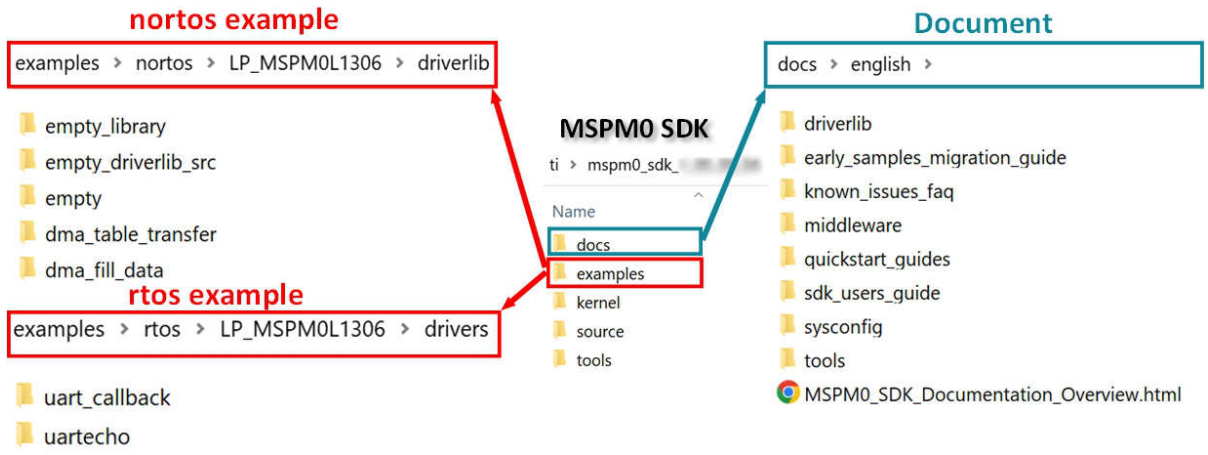


Figure 3-6. MSPM0-SDK Example

3.2.2.1 Documents Folder Introduction

This section lists all the documents in MSPM0-SDK. This is based on version 1_20_01_06.

MSPM0 SDK Documentation:

- Release Notes: Lists all the contents of the MSPM0-SDK and release notes.
- Quick Start Guides: Provides step-by-step instructions to get started quickly using MSPM0 with Code Composer Studio™ (CCS) Theia, CCS, IAR or Keil.
- MSPM0 SDK User's Guide: Homepage of MSPM0-SDK. Provide navigation to MSPM0-SDK example guide and SDK overview.
- Manifest: Lists all the contents in SDK and every installation file path for each component.
- Early Samples Migration Guide: Describes the recommended tool versions that support production samples and provide migration guidelines for applications using DriverLib and SysConfig configuration files.

DriverLib Documentation:

- DriverLib Guide: Provides a software layer to the programmer to facilitate a higher level of programming compared to direct register access.

TI Drivers Documentation:

- TI Drivers Overview: TI Drivers is a collective of peripheral drivers for TI's MSPM0 portfolio. The drivers are centered around a portable application programming interface (API) which enables seamless migration across the MSPM0-SDK portfolio. Unless specifically stated otherwise, TI Drivers are designed to be thread safe and work seamlessly inside of a real-time operating system (RTOS) application.

Middleware Documentation (Libraries and protocol stacks for different applications):

- Middleware Main Folder
- Secure Booting and Updating
- Brushed Motor Control Library
- DALI Library
- Diagnostic Library
- EEPROM Emulation Library
- Energy Metrology Library
- GUI Composer Library
- Hall Sensored Trap Motor Control Library
- IQMath Library
- LIN Library
- Sensorless FOC Motor Control Library
- SENT Library
- SMBBus Library
- Stepper Motor Control Library
- PMBus Library

Third Party Documentation:

- CMSIS DSP: Texas Instruments supports Arm® Cortex® Microcontroller Software Interface Standard (CMSIS), a standardized hardware abstraction layer for the Cortex-M processor series.
- IO-Link: Digital interfaces such as IO-Link on the sensor and actuator level offer advantages when maintenance and repair is required in addition to providing seamless communication and improved interoperability.
- Zephyr: Texas Instruments has started development to support Zephyr as a real-time operating option for MSPM0 devices.

MSPM0 Tools Documentation:

- IDEs and Compilers: MSPM0 supports IDEs: Code Compose Studio (CCS), IAR Embedded Workbench for Arm, Arm Keil MDK. For the toolchain, MSPM0 supports both TI Arm Clang Compiler and Arm GCC Toolchain.
- Code Generation: MSPM0 supports SysConfig.

Debugging and Programmings Tools:

- XDS-110: The Texas Instruments XDS110 is a new class of debug probe (emulator) for TI embedded processors.
- MSP-GANG: The MSP Gang Programmer (MSP-GANG) is a device programmer that supports MSPM0 and all variants of MSP430 and MSP432.
- UniFlash: UniFlash is a standalone tool used to program on-chip flash memory on TI MCUs and on-board flash memory for Sitara processors. To access the quick start guide, click here.
- BSL Host: MSPM0 devices are shipped with a highly customizable ROM-based bootloader that supports universal asynchronous receiver/transmitter (UART) and inter-integrated circuit (I2C) communication by default. For more information, see the *MSPM0 Bootloader (BSL) Implementation*.
- MSPM0 Factory Reset GUI Tool: The Debug Subsystem Mailbox (DSSM) can be used to perform a device mass erase, perform a factory reset, and send a password to unlock the SWD interface.
- Elprotronic: Elprotronic offers multiple hardware and software programming tools supporting MSPM0 in addition to Texas Instruments' MSP430 and MSP432, SimpleLink™ (CC), C2000™, and TIVA™-C MCUs. Elprotronic supports MSPM0 include the MSP-GANG, FlashPro-ARM, and GangPro-ARM.
- Segger: SEGGER J-Link debug probes are the most widely used line of debug probes available today. For more details, see Using Segger programmers with MSPM0.
- PEmicro: PEmicro Multilink and Multilink FX debug probes offer an affordable and compact method for TI MSPM0 development, and allow debugging and programming to be accomplished simply and efficiently.
- Lauterbach: MSPM0 is supported by all Arm debug tools. Generally used for Cortex-M controllers, the preferred tool is the µTrace for Cortex-M.

3.2.2.2 Examples Folder Introduction

TI manufactures a LaunchPad for one MSPM0 sub family with a superset MSPM0 on board. The same example code can be reused across this MSPM0 sub family. The nortos example is under the address *mspm0_sdk_x_x_x_x* \ *examples* \ *nortos* \ *LP_MSPM0xxxx* and the RTOS example is under the address *mspm0_sdk_x_x_x_x* \ *examples* \ *RTOS* \ *LP_MSPM0xxxx*. This section shows a brief introduction for some key example types.

- RTOS Folder:
  - Drivers: Examples uses kernel functionality and provide higher-level hardware operation based on TI Drivers. For Driver Porting Layer (DPL), the DPL abstracts the drivers, allowing for migration between different RTOS kernels or No-RTOS. For POSIX layer, the layer abstracts RTOS functionality, allowing for migration to new kernels.
- Nortos Folder:
  - DriverLib: Simple modular examples showing MSPM0 functionality, consisting of low-level drivers with the highest optimization.
  - Middleware: Designs for different applications, with libraries and protocol stacks, including automotive, appliances, building automation, and so on. For a list of supported middleware, see MSPM0-SDK Document Overview.
  - Demos: Integrated ready-to-use demos, such as driver code examples to work with TI analog devices.

For reference, examples under *Drivers* and *DriverLib* supports all the platforms listed in Table 3-3. Examples under other folders at least support the CCS platform.

Table 3-3. MSPM0 Example Coverage

Supported by SDK	Platform 1		Platform 2	Platform 3
IDE	CCS		Keil	IAR
Compilers	TI Arm-Clang	GNU Arm (GCC)	Arm and Keil Compiler	IAR Arm compiler
RTOS	FreeRTOS			
Code examples	DriverLib and TI Drivers			

In the RTOS example level, the most important folder is the *Drivers* folder that demos the peripheral control based on TI Drivers.

In the nortos example level, the most important folder is the *DriverLib* folder, which contains the peripheral example code based on DriverLib. In nortos examples, there are four empty examples for users to build projects. The differences are listed in Table 3-4.

Table 3-4. Empty Project Description

Example	Type	Language	Use SysConfig	Library Files in Project
empty	Project	C	Yes	No
empty_cpp	Project	C++	Yes	No
empty_library	Static library (.lib file in *Debug* folder after debugging)	C	No	No
empty_driverlib_src (Suggested)	Project	C	Yes	Yes

For a MSPM0 peripheral quick start, please see MSPM0 Academy as well, which delivers training modules for various topics in the MSP MCU portfolio.

3.3 SysConfig Setup

SysConfig is a collection of graphical utilities for configuring pins, peripherals, and other components. SysConfig helps manage, expose, and resolve conflicts visually so that a user has more time to create differentiated applications. The output of the tool includes the C header and code files that can be used with MSPM0-SDK examples or used to configure custom software.

3.3.1 SysConfig Installation

If a user selects CCS as the IDE platform, this section can be ignored, as SysConfig is already integrated.

If a user selects Keil or IAR as the IDE platform, download the standalone SysConfig configuration tool and follow the steps to finish the installation, as shown in Figure 3-7.

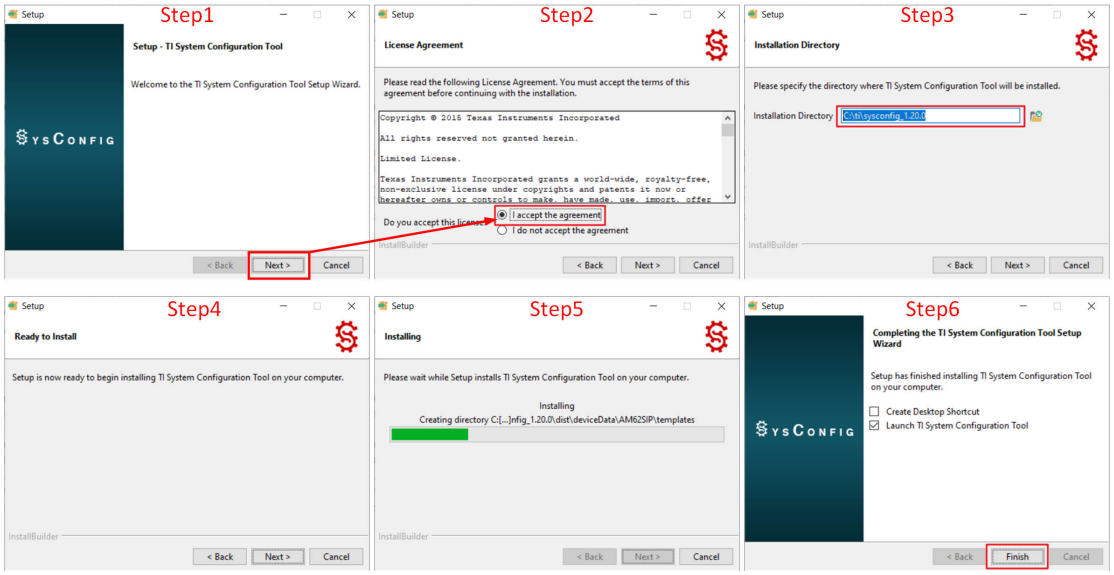


Figure 3-7. SysConfig Install

3.3.2 SysConfig Introduction

This section is a simple introduction on how to use SysConfig. Additional sections further introduce how to use SysConfig with IDE in Section 3.4.

- Add the required peripherals in *Peripheral Usage*.
- Set the parameters in *Peripheral* setting, paired with the device-specific technical reference manual.
- After debugging, the peripheral can generate C code directly.

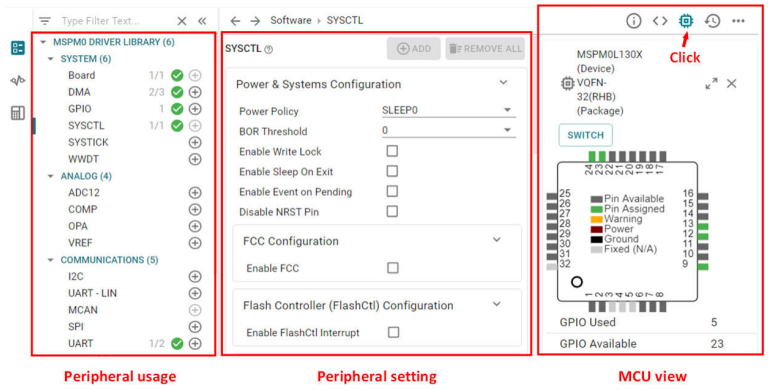


Figure 3-8. MSPM0 SysConfig

The next section introduces the components in SysConfig, which is abstracted from Using SysConfig with MSPM0.

3.3.2.1 Basic Concept

This section introduces SysConfig function blocks and basic operation.

As shown in Figure 3-9, the basic view is shown after SysConfig is opened. SysConfig has two function blocks: the peripheral usage block, which is used to show the added peripherals and the peripheral setting menu entrance. Second is the peripheral setting, which is used to configure the MCU peripherals.

After clicking the buttons on the right side of the screen, the user can open more windows. Generated files are shown after the project build. The user can click the files individually to know the changes after doing new settings on SysConfig. The MCU view is used to view the pin assignment and pin resources, which is also an entrance for MSPM0 migration.

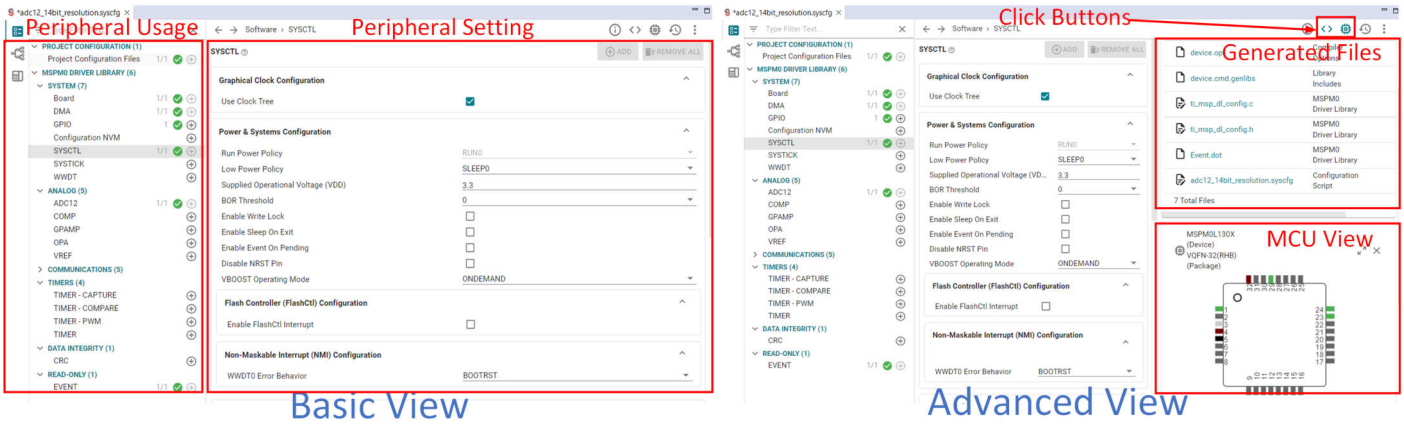


Figure 3-9. SysConfig View

The basic operations of SysConfig, includes adding peripherals, removing peripherals and referring the peripheral or function descriptions. As SysConfig is a low level MSPM0 peripheral setting GUI, see the technical reference manual or the peripheral examples to obtain a better understanding.

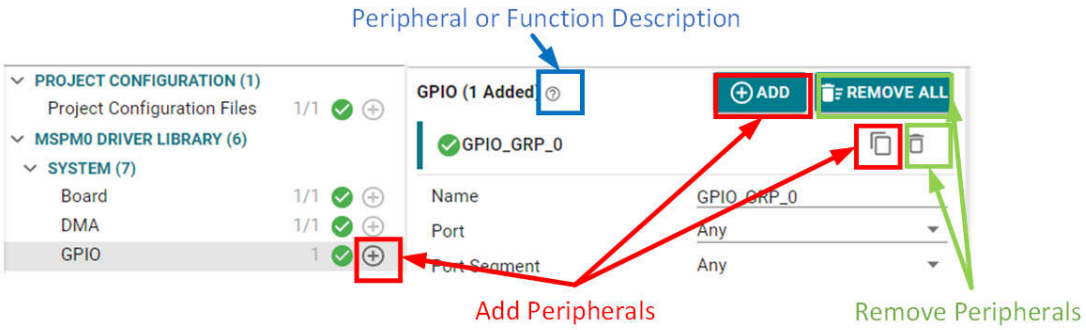


Figure 3-10. Basic Operations

3.3.2.2 Project Configuration View

Here is the project configuration. The configuration influences the total MCU project setting. This section is an introduction to some important features.

- *File Generation*: After you enable all the selection box, the project related files are auto generated by SysConfig. We suggest you to keep them under selection.
- *Include Libraries*: This shows all the libraries included in the SDK. After the selection box is enabled, the related library is included into the project automatically.
- *Select Device*: As the SDK examples is for the LP, after the MCU is migrated, this setting can be changed.

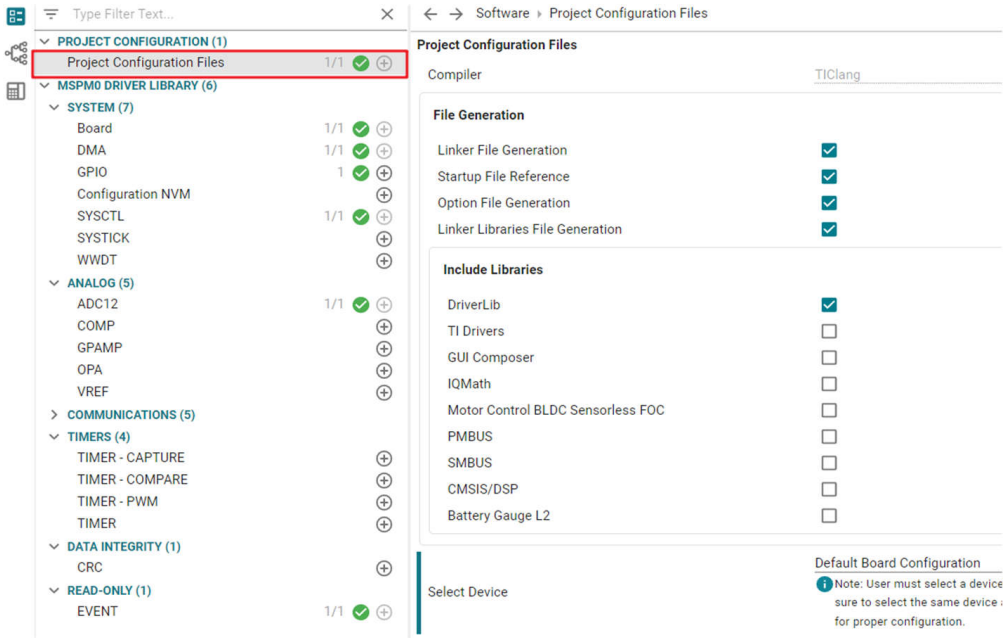


Figure 3-11. Project Configuration

3.3.2.3 Board View

Board view is used to configure the total MCU configuration.

- *Debug Configuration*: For some MSPM0s, the configuration reuses the debug port as peripheral functions. This is the SWD disabled entrance.
- *Global Pin Configuration*:
  - Enable Global Fast-Wake: This reduces the wake-up time sourced from any GPIO port.
  - Generate Peripherals and Pin Assignments File: After enabling, a peripherals and pin assignments file is generated in the *Debug* folder, as shown in Figure 3-12.

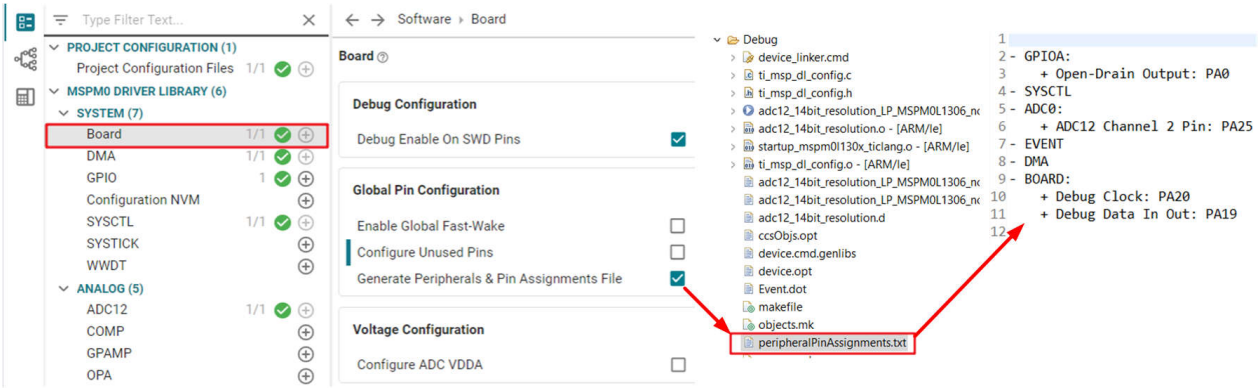


Figure 3-12. Board View

3.3.2.4 NONMAIN View

The NVM (NONMAIN) is used to configure the MSPM0 protected area related to boot configuration, security, and bootloader. With the incorrect program in NONMAIN, MSPM0 breaks. That is why the configuration risks must be accepted before performing configurations. As this function is for high level users, for details, please refer to MSPM0 NONMAIN FLASH Operation Guide.

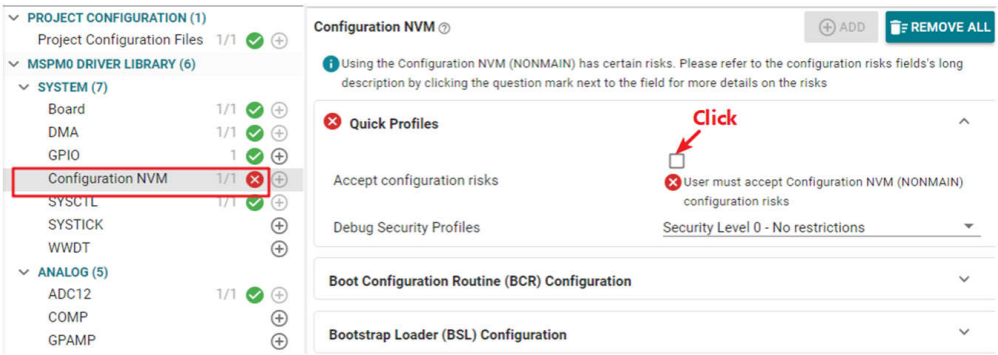


Figure 3-13. NONMAIN View

3.3.2.5 SYSCTL View

SYSCTL is used to configure MCU power, clock, and reset modules. The basic view is menus view. This section introduces the main configurations.

- Power and Systems Configuration:
  - Low power policy: Sets the low-power level for MSPM0.
  - Disable NRST pin: For some MSPM0 devices, the NRST pin can be reused as peripheral functions. This is the NRST pin disabled entrance.

The second view is clock tree view. The clock tree feature allows the user to configure the clocking of a device graphically rather than using SYSCTL menus, which can be found by clicking the signal icon near the top left corner of SysConfig. At the bottom left of the clock tree view, a user can locate all of the used clocks. For a detailed configuration on every clock source, click the icons as shown in Figure 3-14.

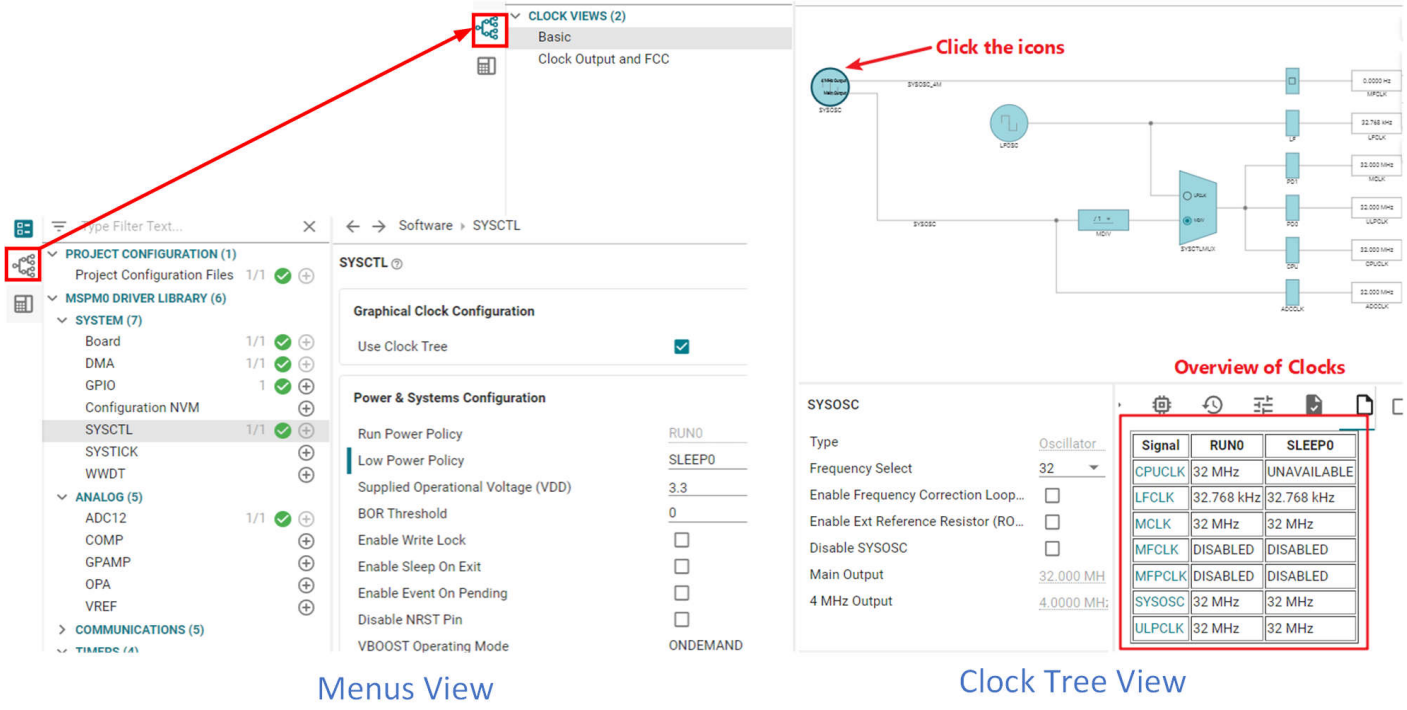


Figure 3-14. SYSCTL View

3.3.2.6 Peripherals Setup

This section introduces peripheral settings, as shown in Figure 3-15. Open the software module description by selecting the module before adding the description. The description includes an overview of the functionality of the module. For more information, see the device data sheet or technical reference manual.

A peripheral configuration is a combination of these configurations:

- Basic configuration: Basic peripheral configuration
- Advanced configuration: Advanced peripheral configuration
- Interrupts configuration: Enable or disable MCU interrupt
- Event configuration: Peripheral to peripheral trigger configuration
- Pin configuration: Enables pullup or pulldown resistors
- PinMux: Selects the pin input or output for the selected functions

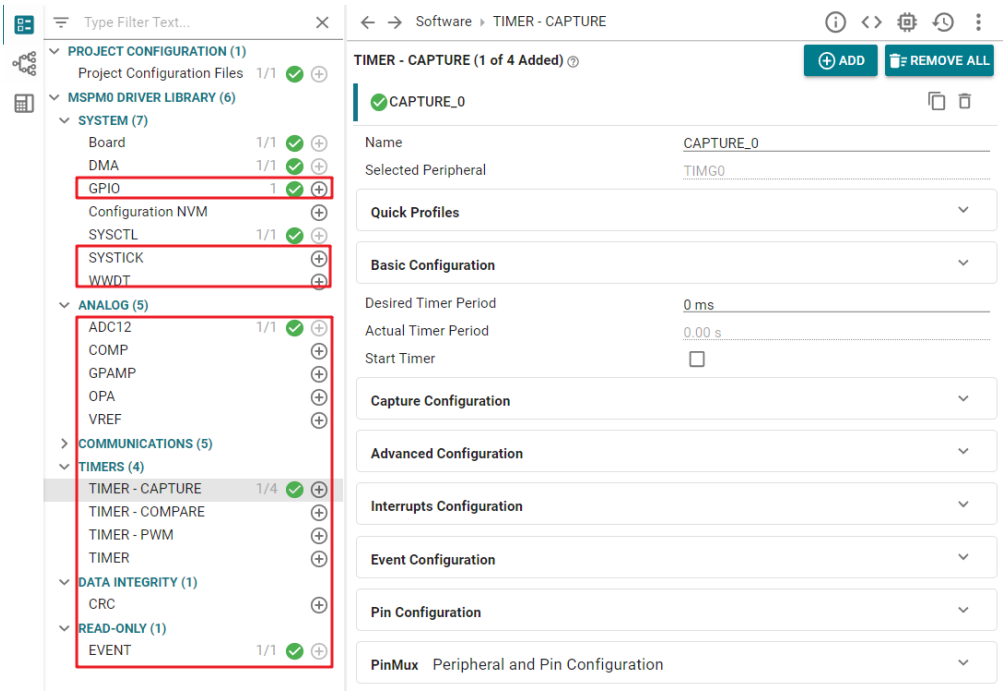


Figure 3-15. Peripherals View

3.4 IDE Quick Start

The MSPM0 series supports three IDEs to develop. CCS is recommended as a preferred option, as this is TI's IDE, which is compatible with MSPM0. The three different types of IDEs are listed and compared in Table 3-5.

Table 3-5. MSPM0 Supported IDEs Overview

IDEs	CCS	IAR	Keil
License	Free	Paid	Paid
Compiler	TI Arm Clang GCC	IAR C/C++ Compiler™ for Arm	Arm Compiler Version 6
Disk size	3.44G(ccs1220)	6.33G(Arm 8.50.4)	2.5G (µVision V5.37.0)
XDS110	Supported	Supported	Supported
J-Link	Supported	Supported	Supported
EnergyTrace	Supported	No	No
MISRA-C	No	Supported	No
Security	No	Supported	No
ULINKplus	No	No	Supported
Function safety	No	Supported	Supported

The following links provide the related guides for different IDEs. All the content in this part is abstracted from these guides.

- Quick Start Guides
- CCS IDE Guide for MPSM0
- IAR IDE Guide for MSPM0
- Keil IDE Guide for MSPM0

3.4.1 CCS Quick Start

3.4.1.1 CCS Installation

This section details steps and tips for CCS installation. Remember to save CCS at the address and the default installation place that is suggested.

1. Download CCS (12.2 version or above), start installation, and keep pressing *Next*.

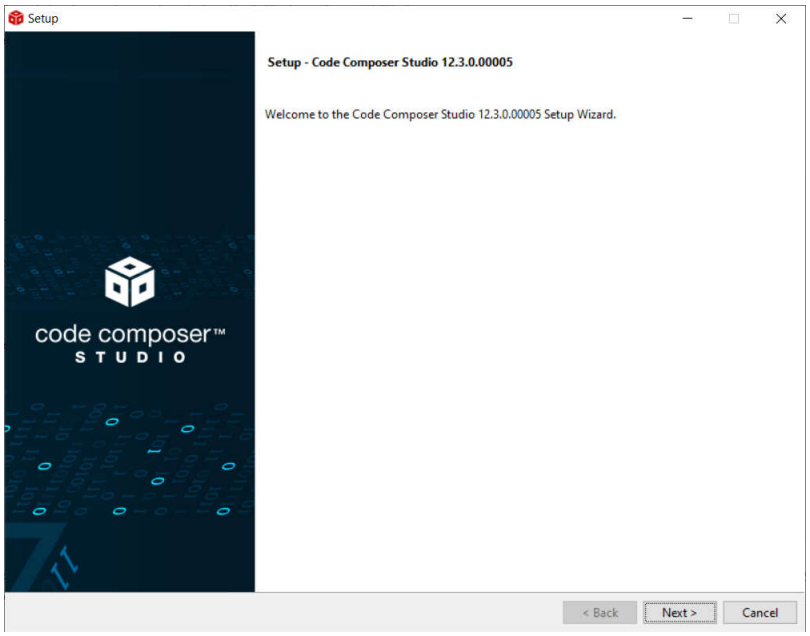


Figure 3-16. CCS Installation

2. Select MSPM0 support component.

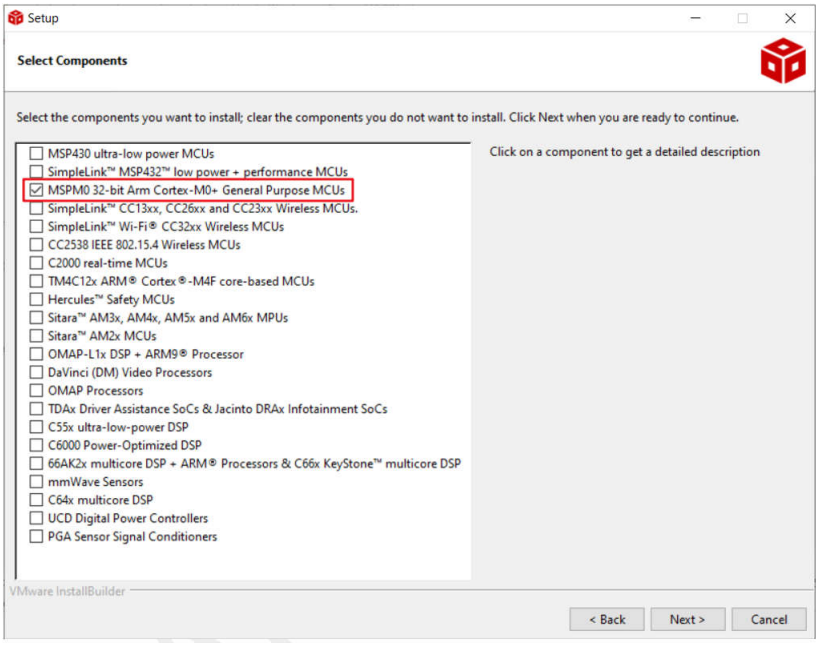


Figure 3-17. MSPM0 Support Selection

3. Select J-link if required.

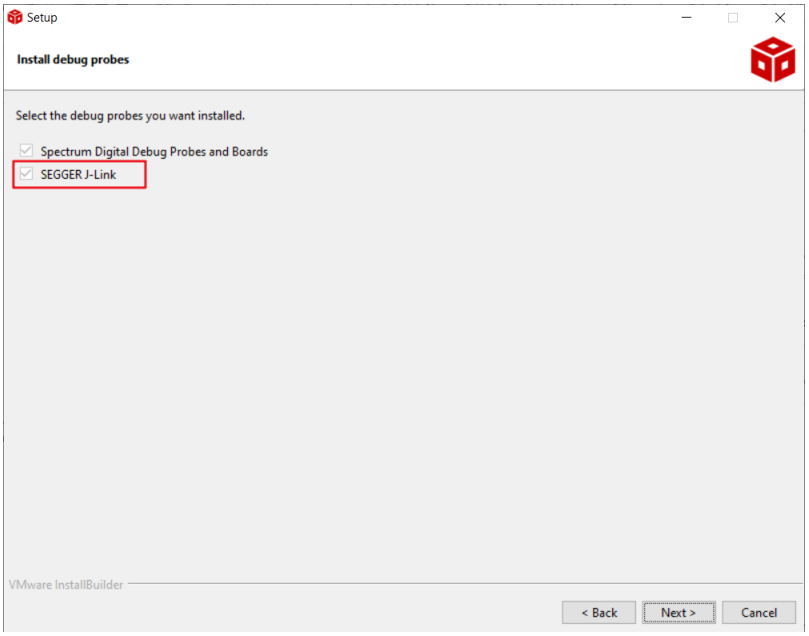


Figure 3-18. J-Link Selection

3.4.1.2 Environment Setup

If the user installs CCS and SDK at the default address folder: (*C:\ti*), then the related SDK and SysConfig is loaded automatically when an example is imported. The environment setup chapter can be skipped.

3.4.1.2.1 SDK Support Setup

For SDK introduction and installation, see Section 3.2.2. If CCS and SDK are installed at the customized address, use the following steps so that CCS loads SDK successfully.

1. Select *Window → Preferences.*
2. As the SDK 2.1.0.03 is installed in the C:\ drive, add *C:* as the product discovery path.
3. Refresh the *Discovered products* window. The SDK 2.1.0.03 is recognized automatically.
4. Click the *Apply and Close* button. The new imported project loads the SDK automatically.

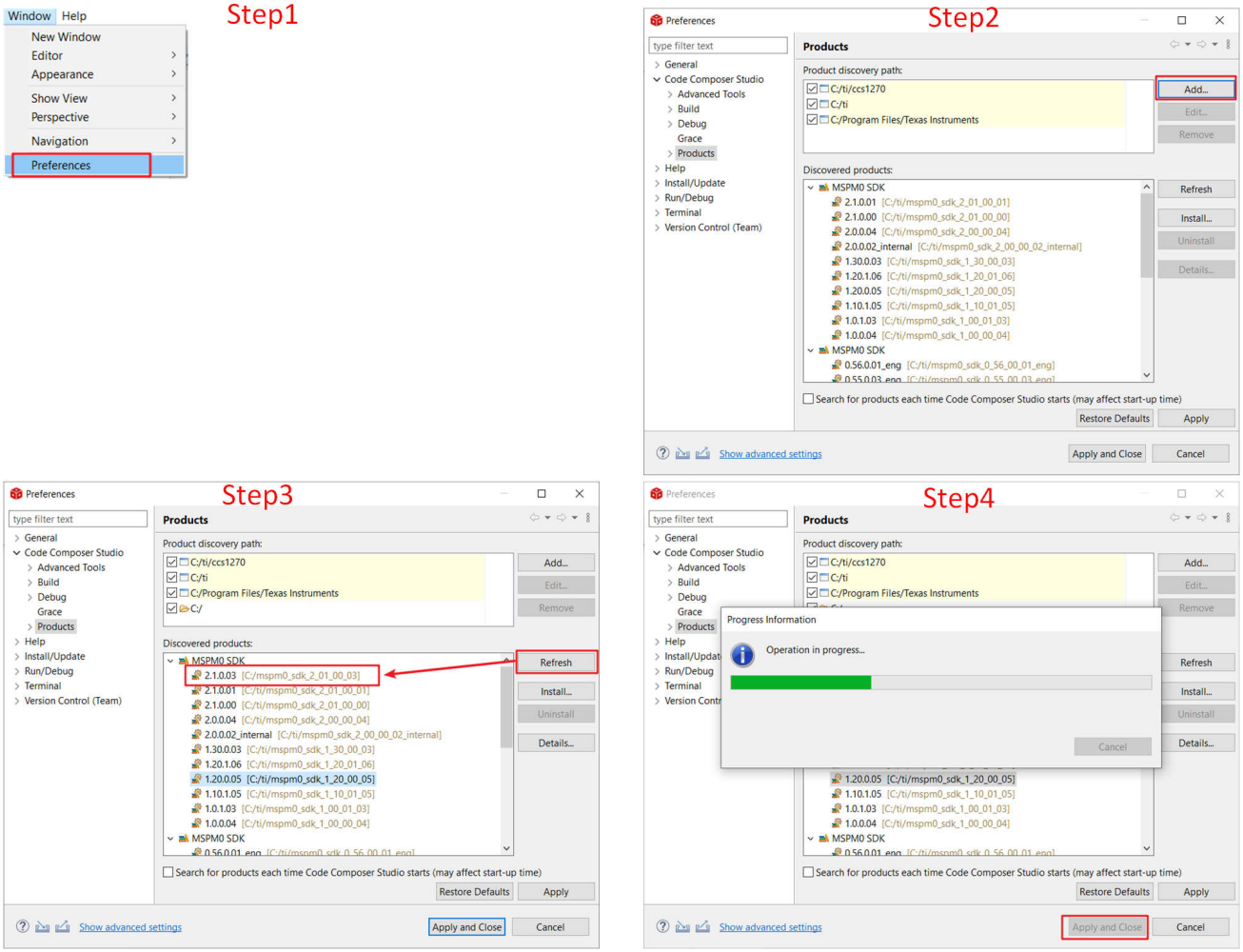


Figure 3-19. Load SDK Product

After an example is imported, follow the steps to select the desired version of the SDK. The steps can also be used when the user wants to migrate an example from an older version of SDK to a newer version.

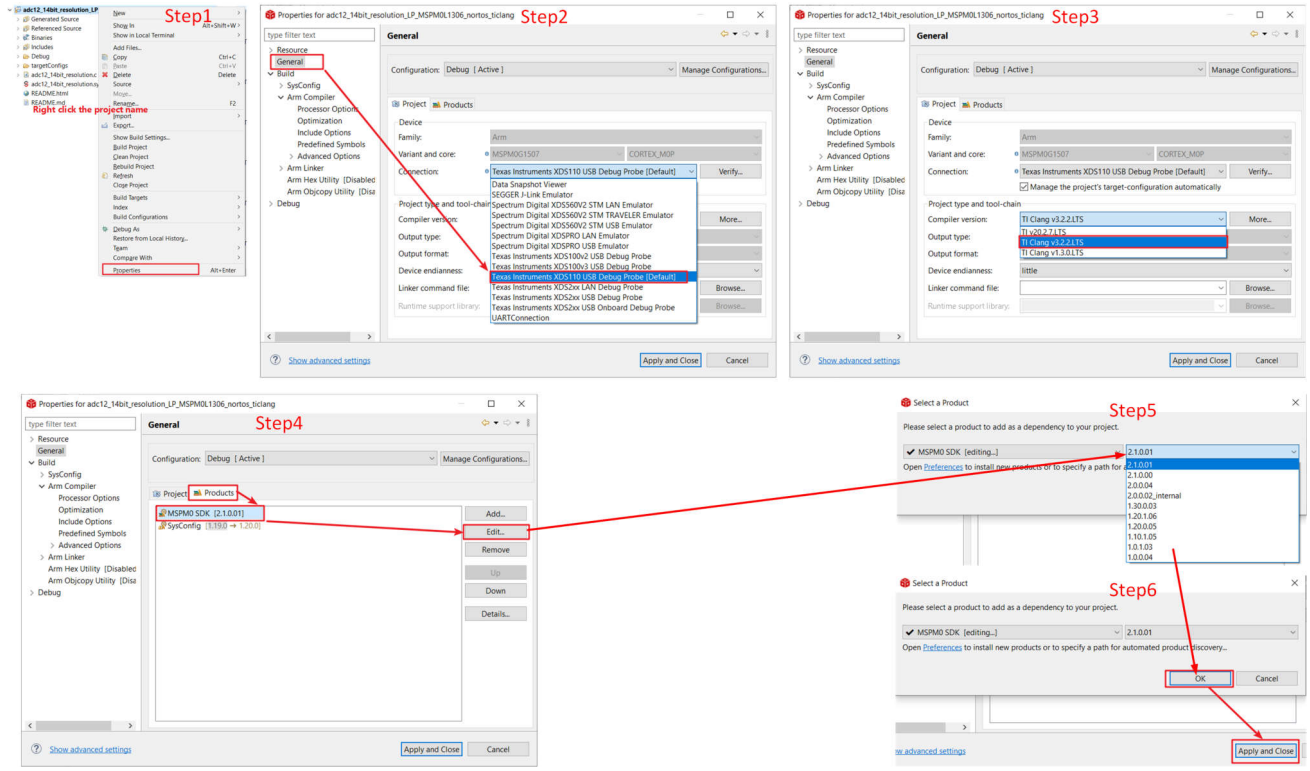


Figure 3-20. Select or Change SDK Version

3.4.1.2.2 SysConfig Support Setup

As the SysConfig is installed with CCS, no further work is required. However, there can be some version compatibility problems on old CCS projects or when a new CCS version is installed that the opened SysConfig reports errors. See SDK installation steps in Section 3.4.1.2.1 to migrate to a different SysConfig version. For SysConfig introduction and installation, see Section 3.3.

3.4.1.3 Import a SDK Example

Open CCS. The workspace is the address to copy an imported project to.

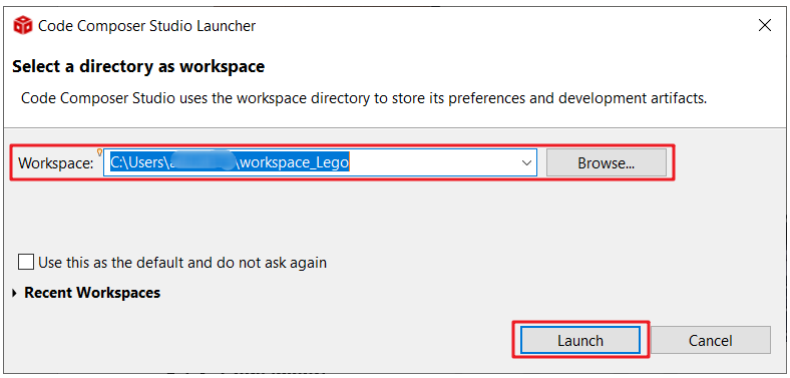


Figure 3-21. Select CCS Workspace

Import an example with the TI-Clang compiler from the installed SDK.

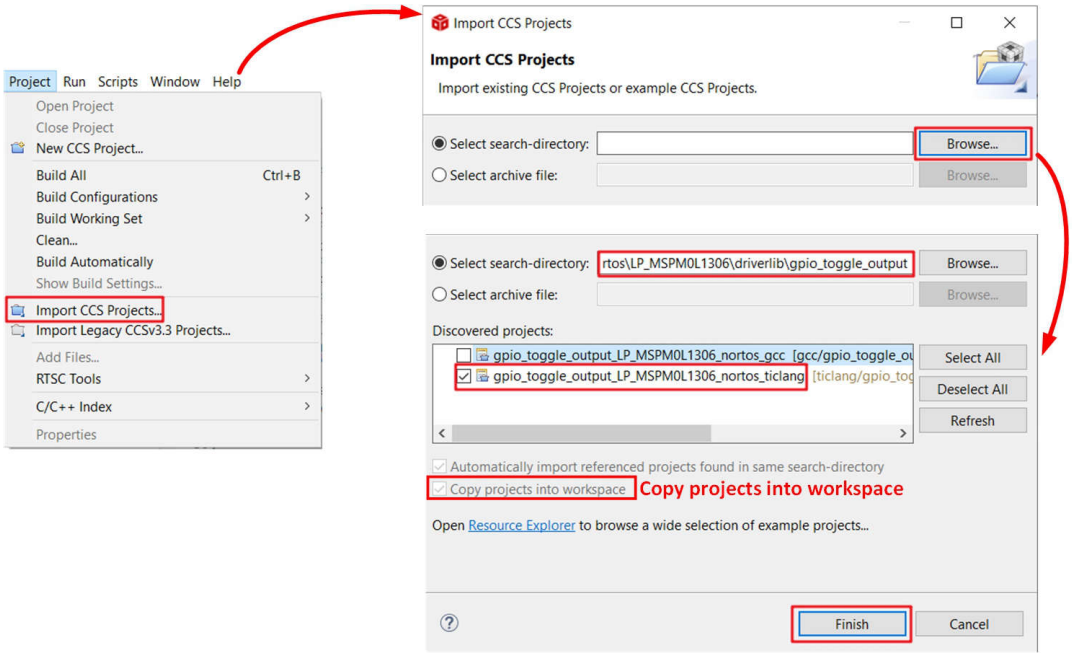


Figure 3-22. Import Project

Here is the view of the imported project. The most important files are in red. This section shows a brief introduction.

- Sysconfig generated code: Click the *Build* button, the SysConfig generates the code under the *Debug\syscfg* folder.
- .map file: In the *Debug* folder, refer to the .map file to find out more about the memory usage condition.
- Main function .c file: Includes the main function in the file.
- .cmd file: Define the MCU memory allocation. In the latest CCS, the user can select to allow SysConfig to generate the allocation automatically.
- SysConfig: GUI tool to generate the peripheral setting code.

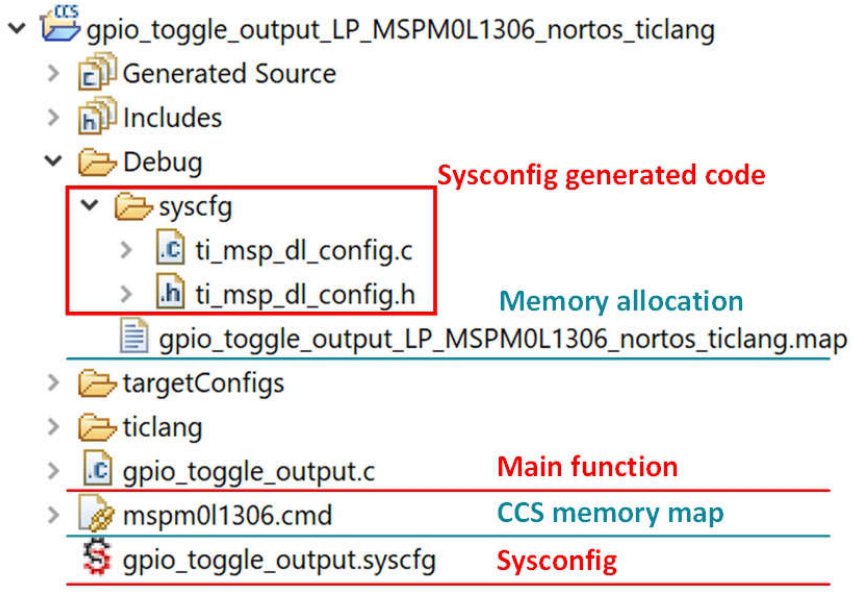


Figure 3-23. CCS Project Overview

3.4.1.4 Example Download and Debug

The default debugger selection is XDS110. To select J-Link, right click the *Project->Properties* and follow the steps to select J-Link.

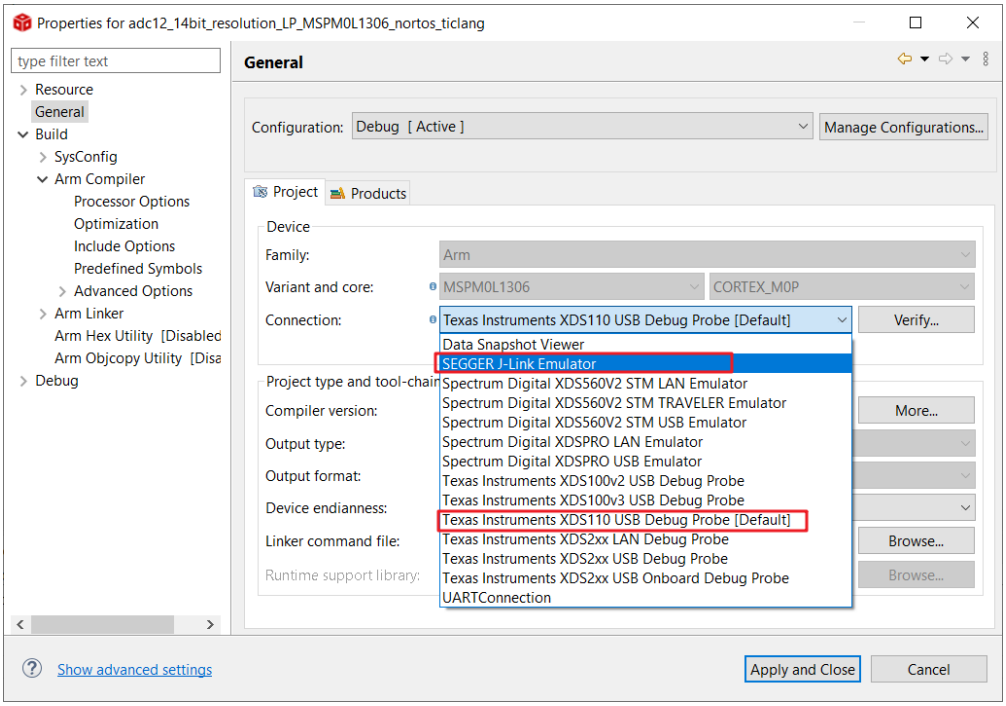


Figure 3-24. Change Debugger Selection

Start debug by click the *Build* button at the top. After that, the window automatically moves from the CCS edit view to CCS debug view. After the MCU enters debug mode, click the *Run* button to enable the code running.

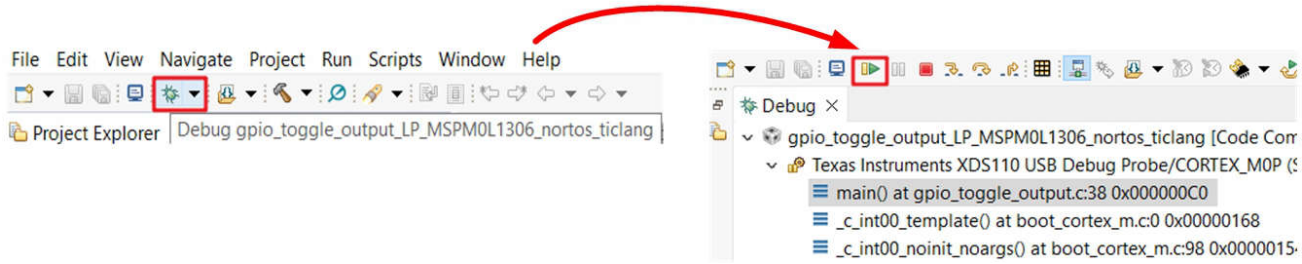


Figure 3-25. Debug Code

This section is a quick introduction to CCS functions. The commonly used functions and meanings are shown in Figure 3-26.

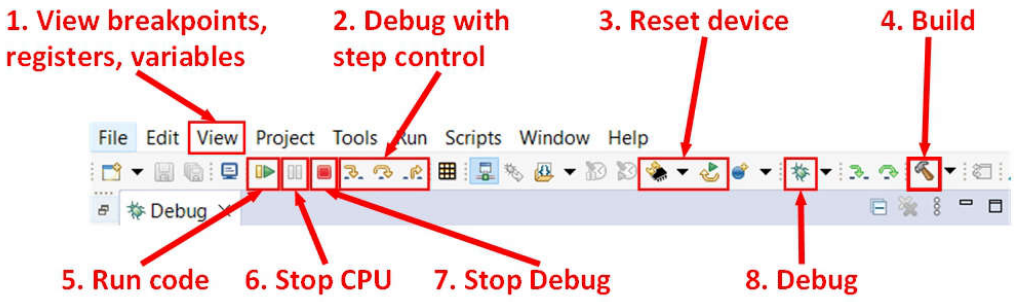


Figure 3-26. Commonly Used Debug Functions

3.4.1.5 Migrating Between MSPM0 Derivatives

Project migration in this scope means updating relevant project configuration files and settings that are specific to the derivative, including linker files, startup files, and included libraries. To facilitate project migration, SysConfig generates project configuration files by default, which can be controlled through the project configuration module.

Here are the migration steps based on CCS:

1. In SysConfig, enable the device view and click on *SWITCH*.
2. Select *New Values* for the *Device*, *Package*, and *CCS Launch Device* to migrate the project configuration to a new device, and then click *CONFIRM* **.**
3. After confirming the new device values, SysConfig highlights an error on the project configuration module. The user must select the new device in the *Select Device* options. Make sure the device selection matches what was selected for *CCS Launch Device* in the previous step.
4. Note that SysConfig highlights any conflicts with the migration, such as unavailable pins and peripherals. Fix any conflicts as needed, and save all the changes to the SysConfig configuration script. Migration is now complete and the user can build a project for the new target device.

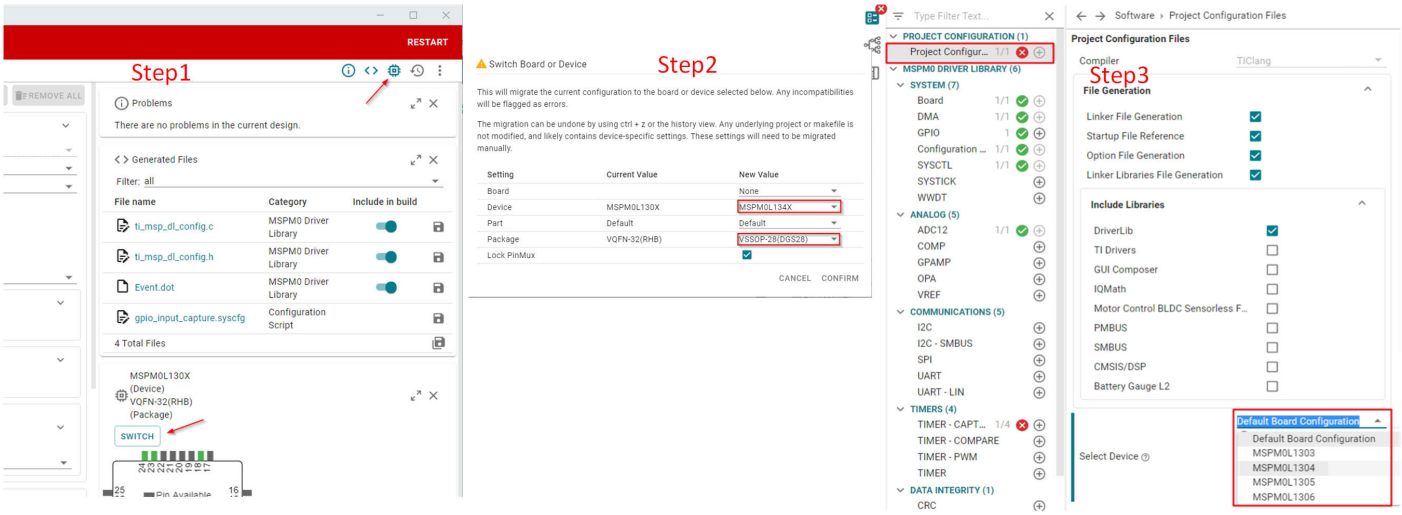


Figure 3-27. Migrating Between MSPM0 Derivatives

3.4.1.6 Generate Hex Files

CCS includes utilities which can be used to generate output objects in multiple formats for use with programming tools. The following steps explain how to enable the hex files using the hex utility which is integrated into CCS.

1. Right-click on a project and select *Properties*. Select *Build → Arm Hex Utility* and select *Enable Arm Hex Utility*.
2. Select *Output Format Options*. The common selections are *Bin*, *Hex*, and TI_TXT format. Select the desired output format options.
3. If the Intel *HEX* format is selected, **one additional step** is required to specify the memory and ROM width as parameters. Select a memory and ROM width of 8in *Properties → Arm Hex Utility → General Options*.
4. After clicking the *Build* button, the hex file generates in the debug folder.

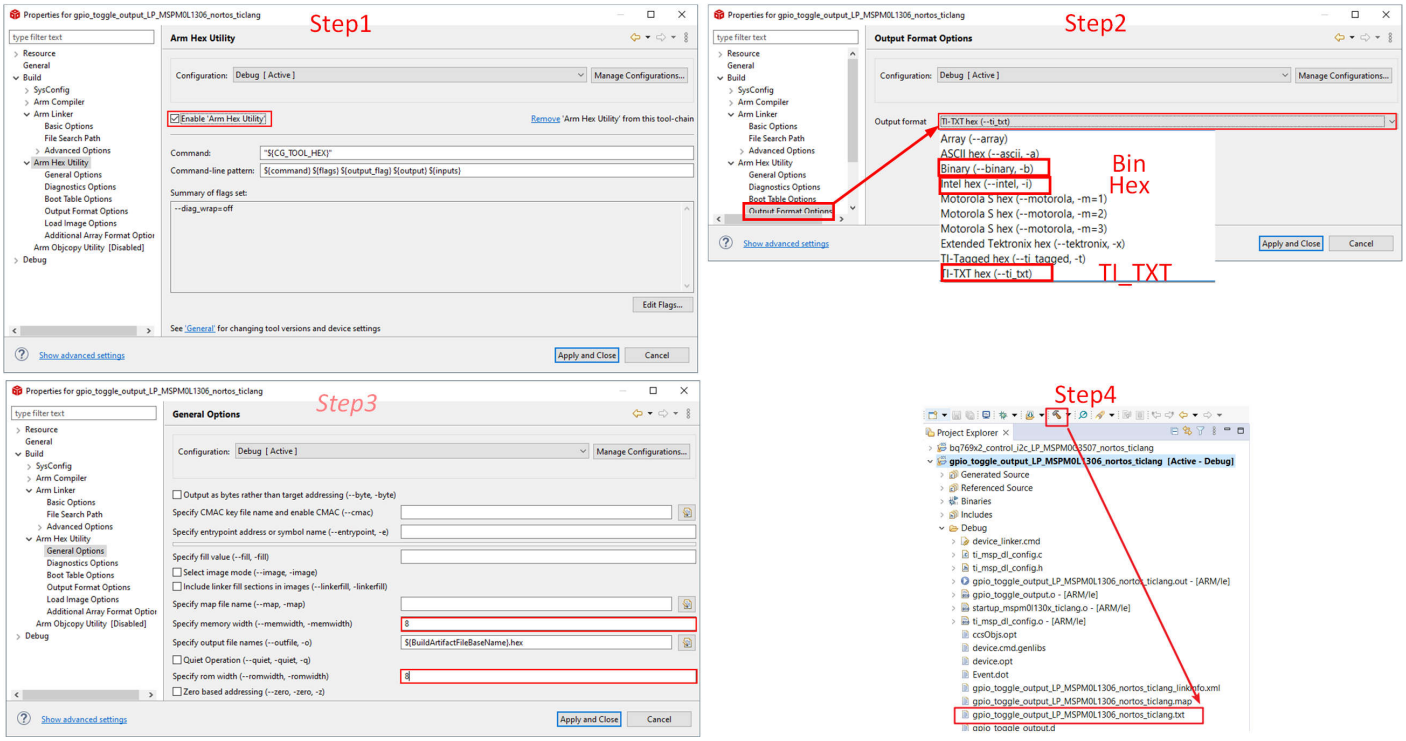


Figure 3-28. Generate Hex File

3.4.1.7 Program NONMAIN

If changes are made on the bootloader or MCU security settings by configuring the NONMAIN as shown in Section 3.3.2.4, enable the NONMAIN erase in the CCS setting as well, as shown in Figure 3-29. Otherwise, keep the default settings.

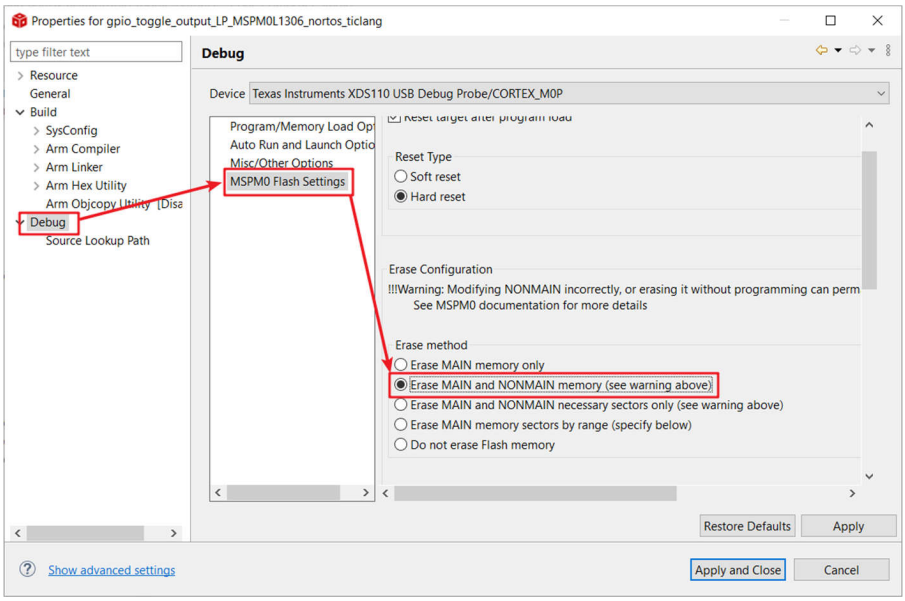


Figure 3-29. Programming NONMAIN

Note

Extreme care must be taken when erasing and programming NONMAIN. If done incorrectly, like losing connection in NONMAIN programming, the device is locked in a permanently unrecoverable state.

3.4.2 IAR Quick Start

TI recommends an IAR Embedded Workbench version higher than Arm 9.32.x. The less recent versions do not support MSPM0.

3.4.2.1 Environment Setup

3.4.2.1.1 SDK Support Setup

In IAR, users must add the latest MSPM0 SDK version. This step only has to be done once, or when the SDK is updated. In IAR, users must add the latest MSPM0 SDK version. This step only has to be done once, or when the SDK is updated.

- Step 1: In IAR, click on *Tools* → *Configure Custom Argument Variables*.
- Step 2: Click the *Global* tab, and then *Import*.
- Step 3: Navigate to your SDK folder into *<MSPM0_SDK_INSTALL_DIR>/tools/iar/* and open *MSPM0_SDK.custom_argvars*.
- Step 4: The SDK variables is now installed in IAR. Click *OK* to close the window.

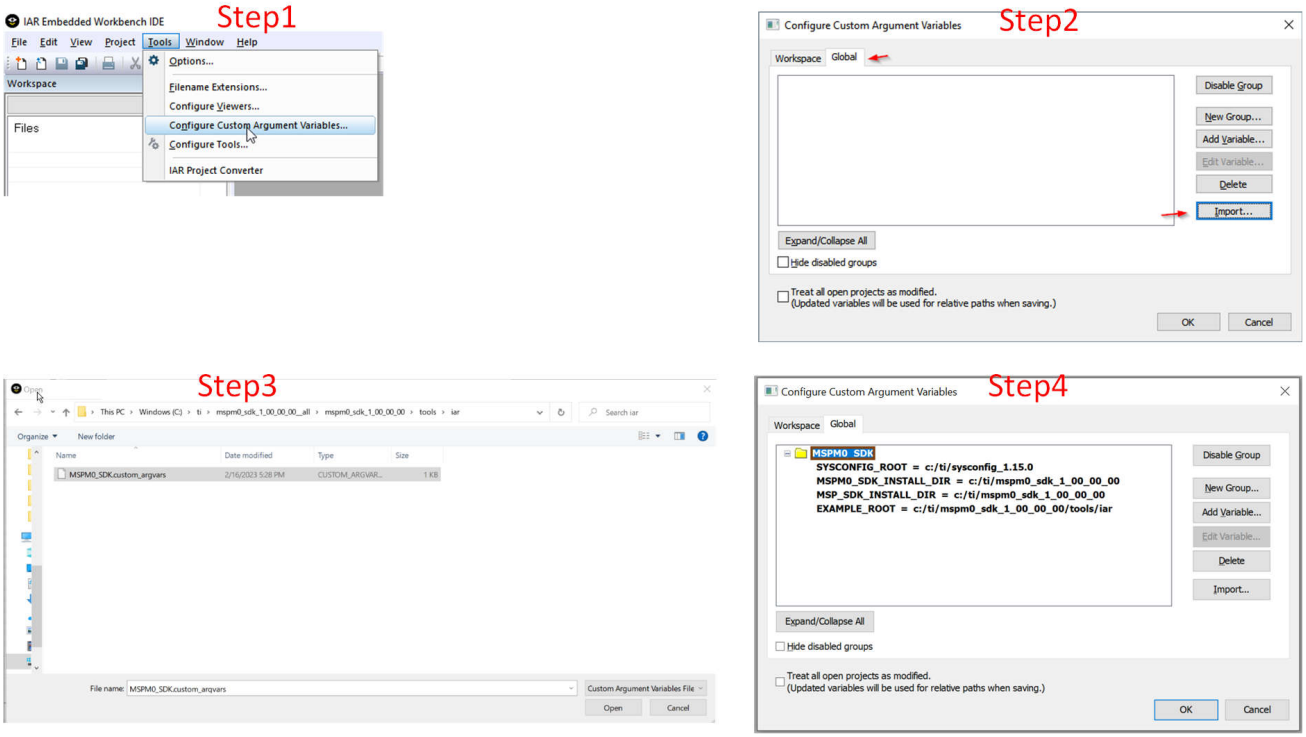


Figure 3-30. Add MSPM0 SDK to IAR

Note

Make sure the MSPM0 SDK path and SysConfig path matches the location and version needed for this SDK release. If an earlier version of the SDK is installed, then make sure to update the path to the current version. If the SysConfig path installed is incorrect or pointing to an older version, then modify the version.

3.4.2.1.2 SysConfig Support Setup

The SDK includes a preliminary version of SysConfig metadata which can be used to evaluate the user experience of MSPM0 SDK.

1. In IAR, select *Tools* → *Configure Viewers* from the menu.
2. Click *Import.*
3. Navigate to your SDK folder into *<MSPM0_SDK_INSTALL_DIR>/tools/iar/* and open *sysconfig_iar_setup.xml*.
4. The standalone SysConfig is associated to .syscfg files. Click *OK* to close window.
5. Double-check that the *SYSCONFIG_ROOT* Custom Argument Variable is correctly pointing to the SysConfig folder.

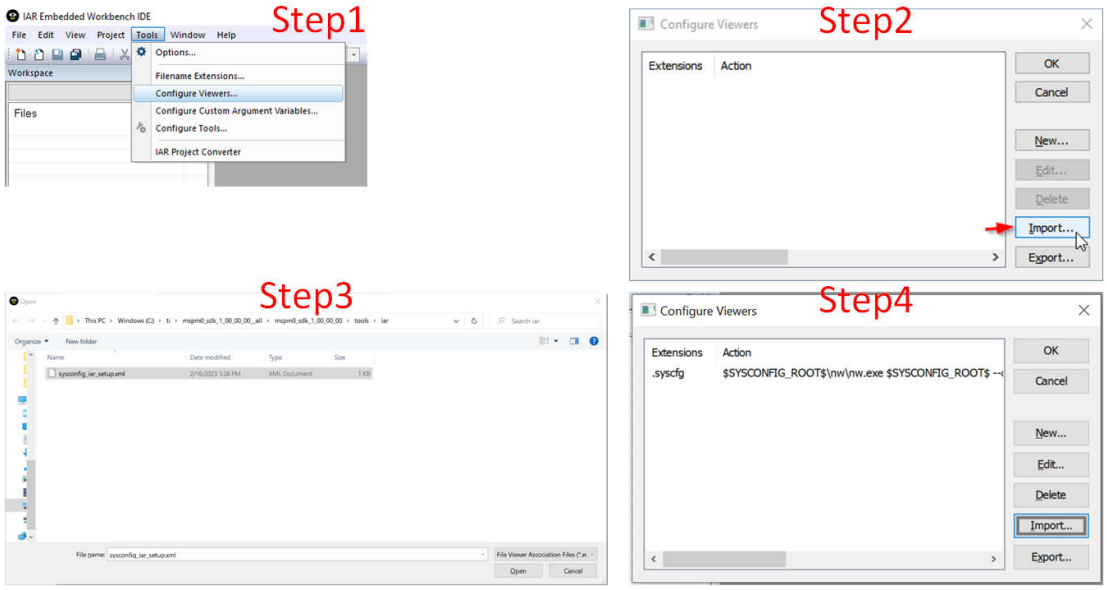


Figure 3-31. Install SysConfig for MSPM0

3.4.2.2 Import a SDK Example

Here are the steps to import an IAR code example from SDK:

1. In IAR, select *File* → *Open Workspace* from the menu.
2. Navigate to an IAR folder in SDK example at <MSPM0_SDK_INSTALL_DIR>/examples/ and open the .eww workspace file.
3. Click *OK* on the message.
4. Select a folder to install the example.

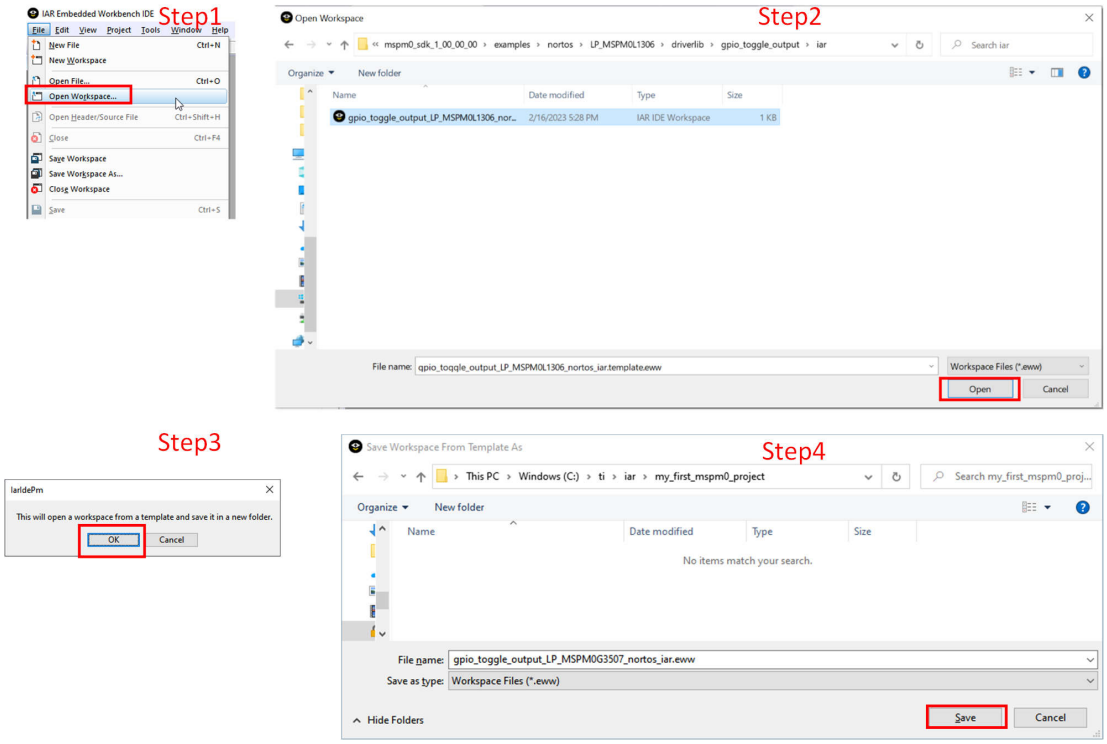


Figure 3-32. Import a SDK Example

This is a simple instruction to use SysConfig with IAR.

1. Double-click on the .syscfg file in your project.
2. This opens SysConfig and allows users to configure peripherals, IO pins, and other settings.
3. Save the changes and switch back to IAR EWARM. Build the code example. The Files in the SysConfig Generate Files folder is updated.

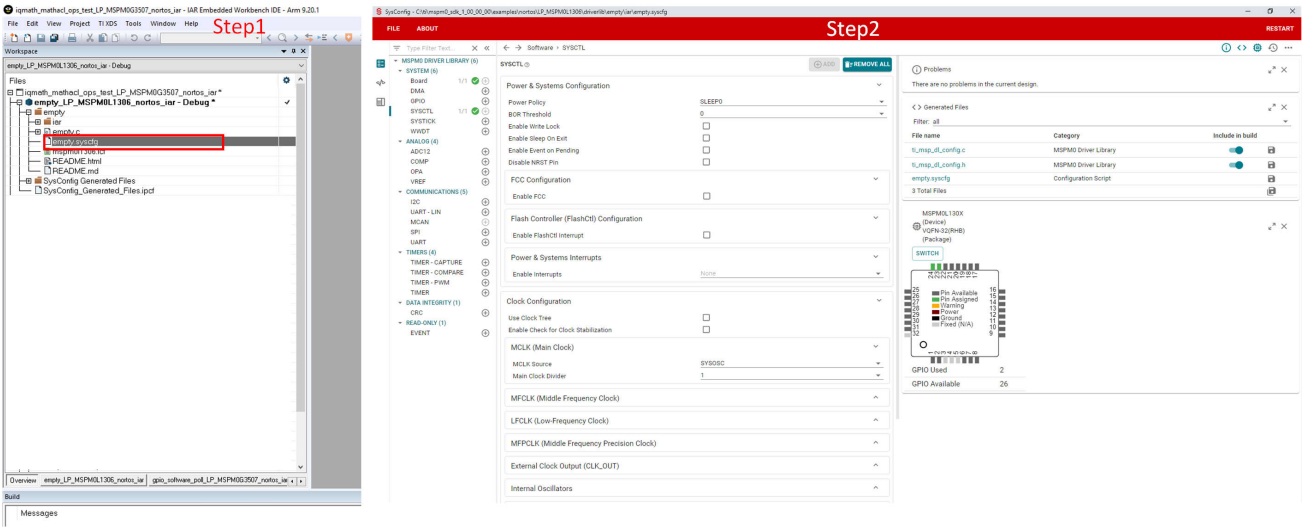


Figure 3-33. Use SysConfig With IAR

3.4.2.3 Example Download and Debug

Follow the steps below to build the example under IAR:

1. To build the example, right-click in the project and select *Make*. Note that SysConfig projects automatically generates files in the *SysConfig Generated Files* folder.
2. Click the *Download and Debug* button to download the code.
3. Now, start to debug the code.

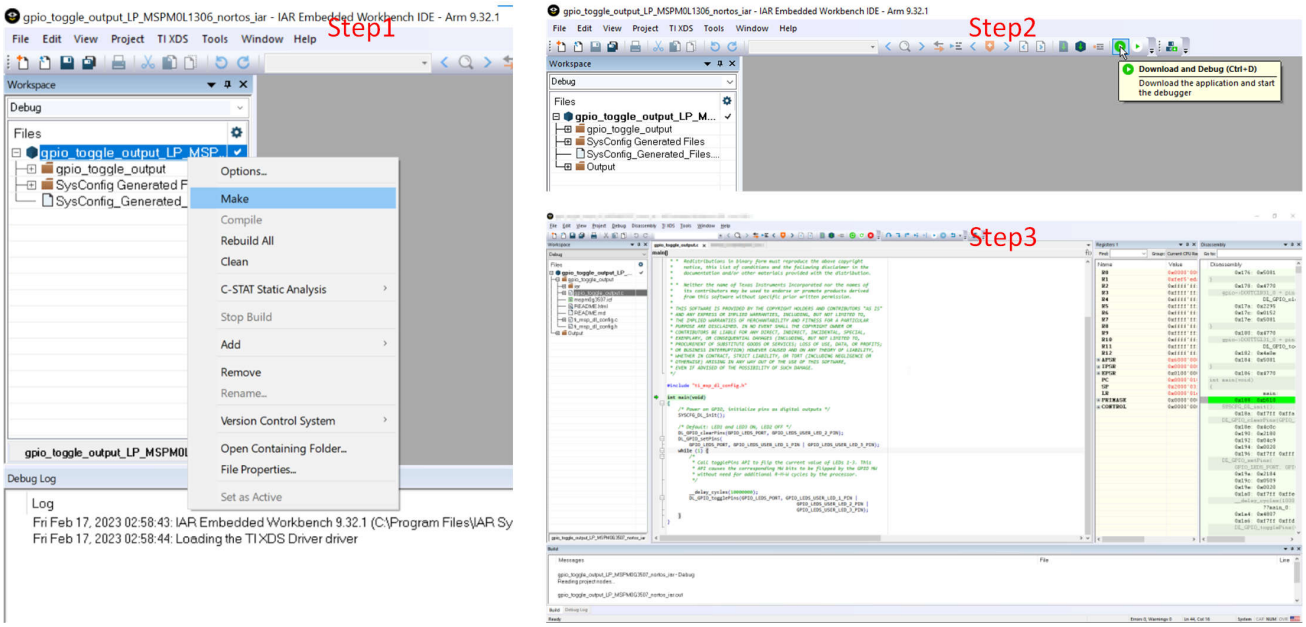


Figure 3-34. Download and Debug

3.4.2.4 Migrating Between MSPM0 Derivatives

SysConfig allows for an easier migration between MSPM0 derivatives. However some manual modifications are required on IAR. Here are the instructions:

1. In SysConfig, enable the Device View and click on *SWITCH*.
2. Select the corresponding options for the new MSPM0 device and click *CONFIRM*. Note that SysConfig highlights any conflicts with the migration, such as unavailable pins and peripherals. Fix any conflicts as needed.
3. In the project options, select *General Options* → *Target* → *Device*. Select the MSPM0 device.
4. In the project options, select *C/C++ Compiler* → *Preprocessor* → *Defined symbols*. Add the device definition as per the device selected.

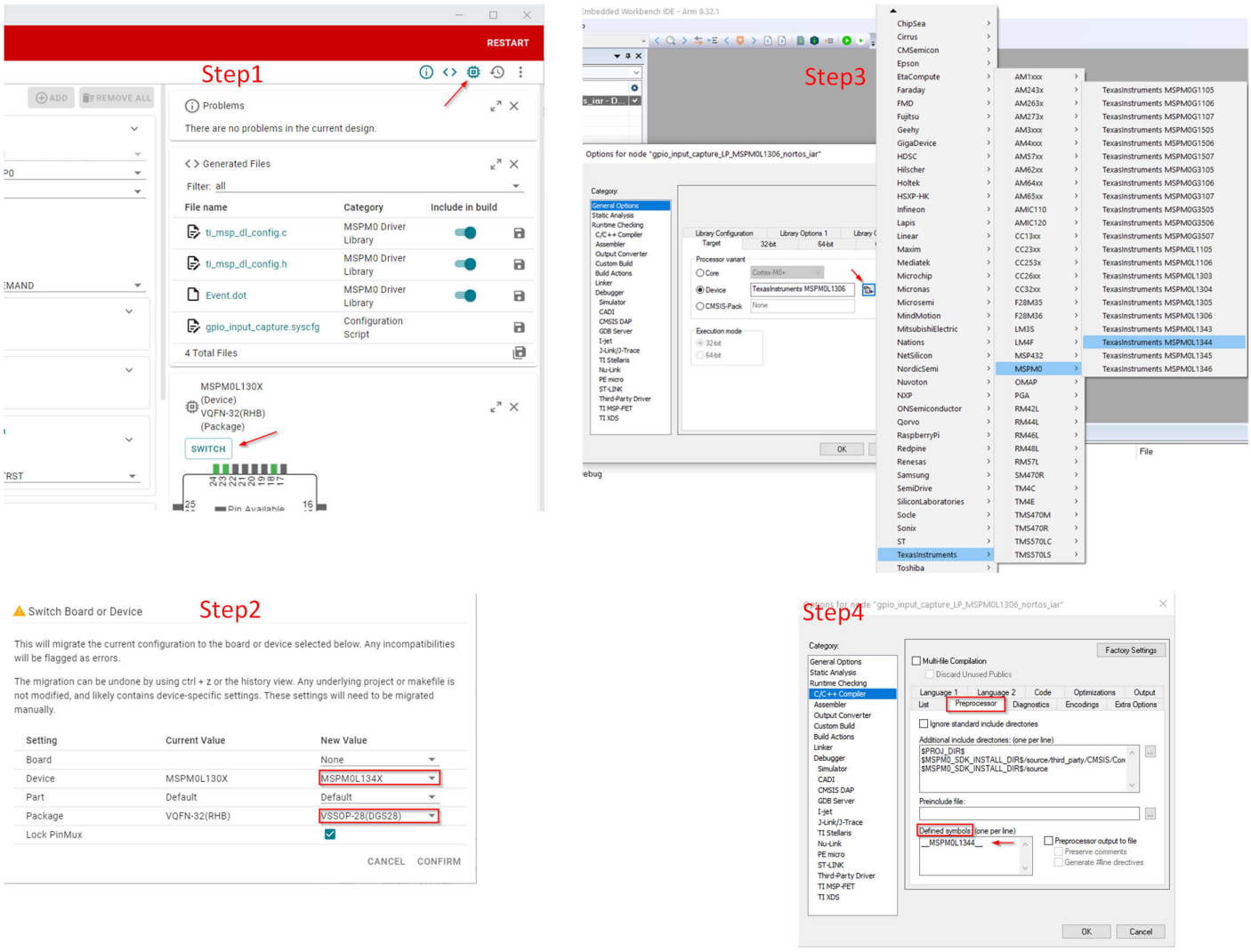


Figure 3-35. Migrating Between MSPM0 Derivatives

3.4.2.5 Generate Hex Files

Here is the instruction to generate hex files in IAR. Click *Project* → *Options* → *Output Converter* → *Generate additional output* → *Output format* → *Texas Instruments TI-TXT*. Intel Hex or other formats also can be selected.

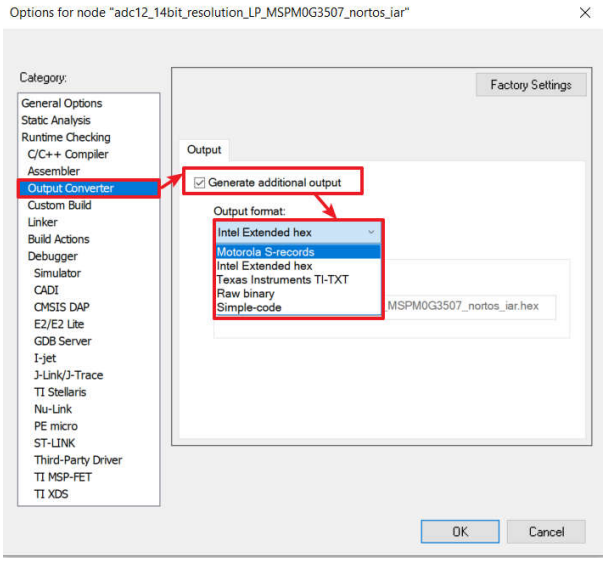


Figure 3-36. Generate Hex Files

3.4.2.6 Program NONMAIN

If users do the changes on Bootloader or MCU security setting by configuring the NONMAIN, then users need to enable the NONMAIN Erase in the IAR setting, as shown in Section 3.3.2.4. Follow the steps below, otherwise, please keep the default:

1. Click *Options* → *Debugger* → *Download* → *Override default .board file* → *Edit*. Select the 2nd element and then click Okay.
2. Add *--non_main_erase* as an extra parameter.

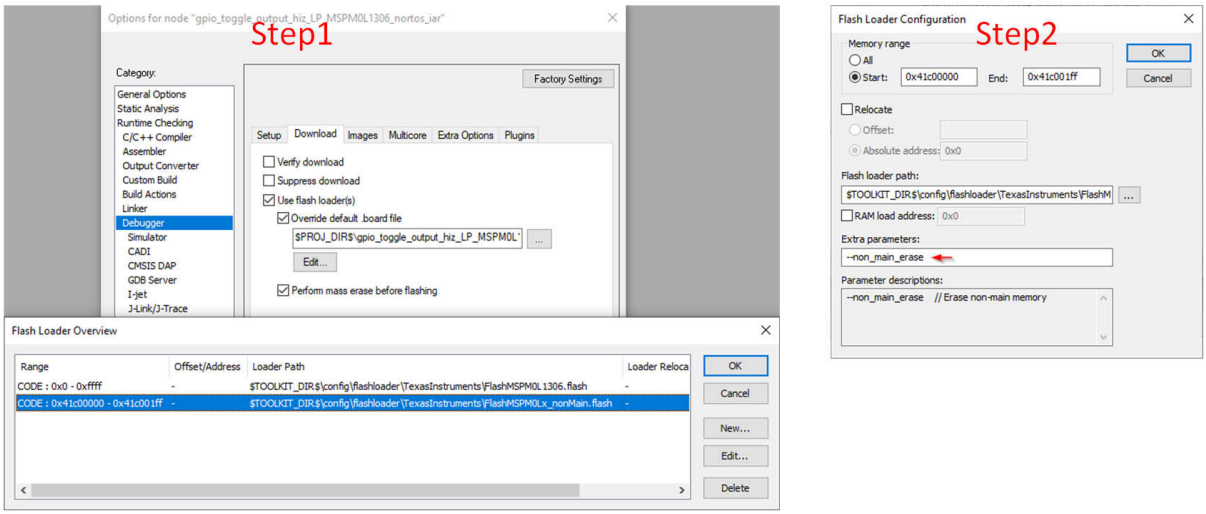


Figure 3-37. Program NONMAIN

Note

Extreme care needs to be taken when erasing and programming NONMAIN. If done incorrectly like losing connection in NONMAIN programming, then the device becomes locked in a permanently unrecoverable state.

3.4.3 Keil Quick Start

3.4.3.1 Environment Setup

Unlike the IAR, this is OK to use old version, Keil, however, remember to update the MSPM0 CMSIS-Pack.

3.4.3.1.1 MSPM0 CMSIS-Pack Setup

The Pack installer needs to be installed first before the MSPM0 is developed. Here are the steps to update MSPM0 CMSIS-Pack:

1. In µVision, open *Pack Installer* through quick guide or select *Project* → *Manage* → *Pack Installer*.

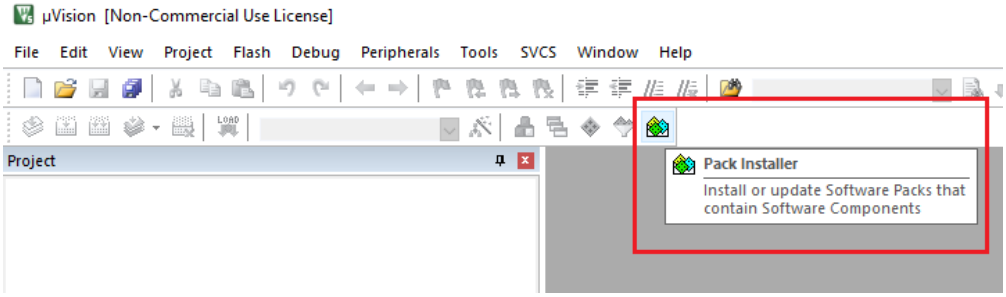


Figure 3-38. Open Pack Installer

2. In Pack Installer, search MSPM0 on the left side in the search text box. Then, the corresponding MSPM0 family is shown on the screen.

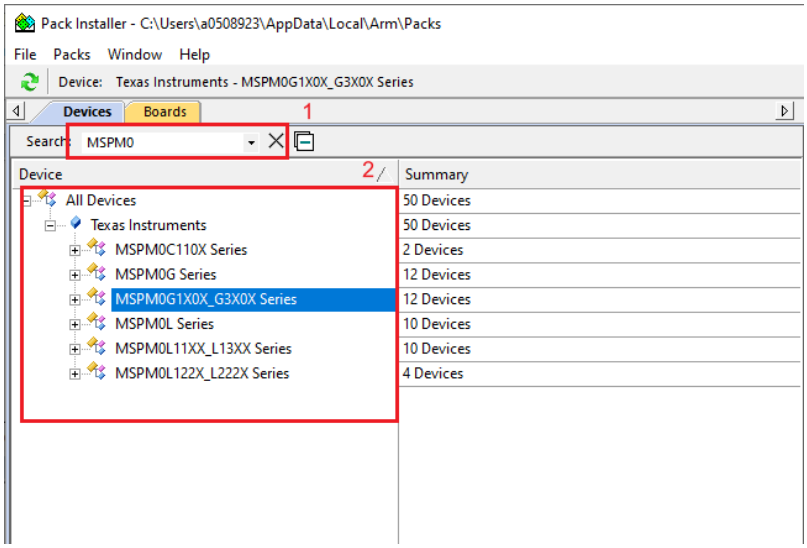


Figure 3-39. Search Device

3. Select the device to install a pack. Then on the right side, install the device-specific pack.

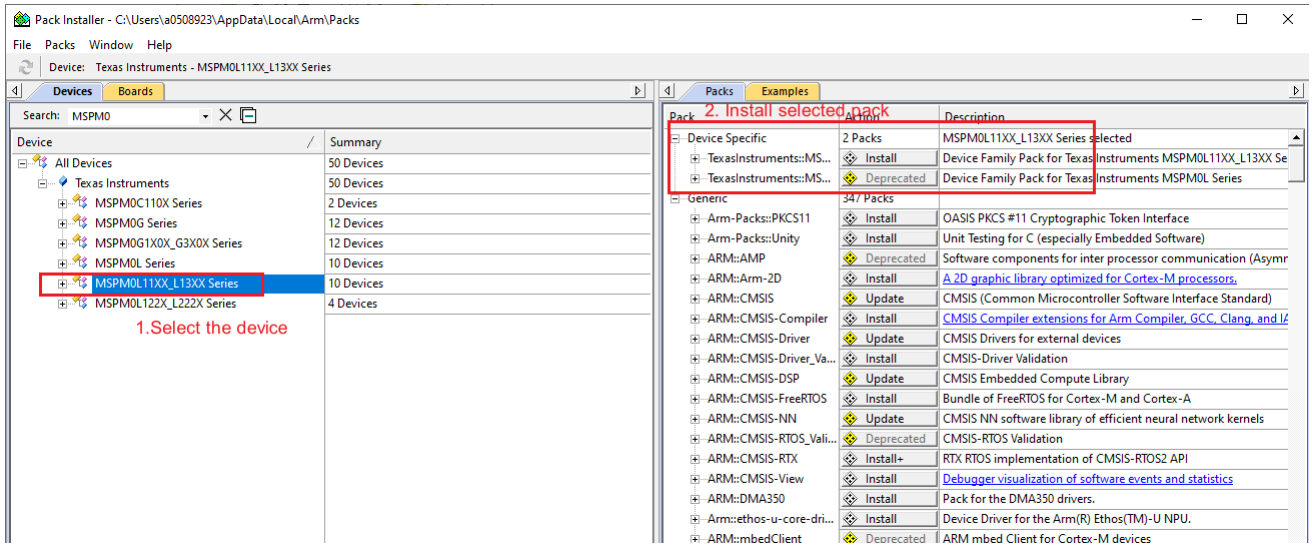


Figure 3-40. Install Device Pack

4. After approving the license terms, the pack is successfully installed.

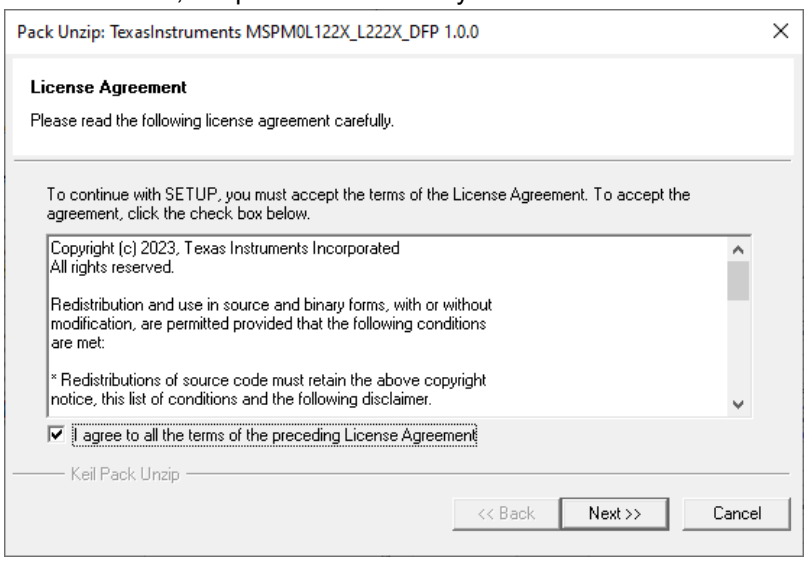


Figure 3-41. Approve the License

3.4.3.1.2 Sysconfig Support Setup

If SysConfig is required, follow the steps below to enable use. Make sure that SysConfig and SDK are installed ahead. Here, we use SDK v1.30 and SysConfig v1.19 as an example.

1. Navigate to the SDK folder (*...\ti\mspm0_sdk_x_xx_xx_xx\tools\keil*). Edit SysConfig path in *syscfg.bat* to match the downloaded standalone SysConfig address.

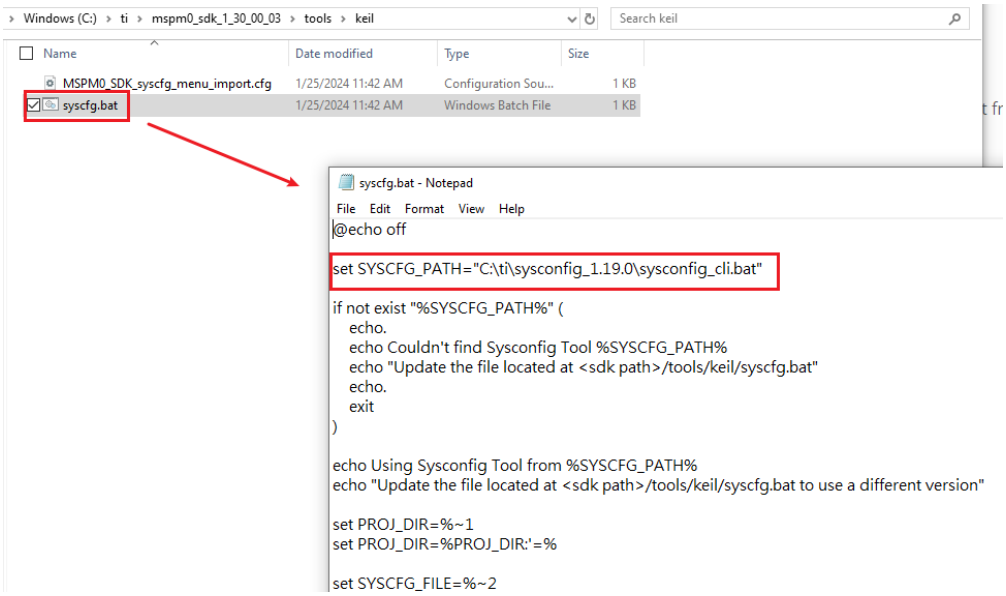


Figure 3-42. Edit syscfg.bat

2. In the same folder, open another file for editing. Modify the SysConfig and SDK versions and paths.

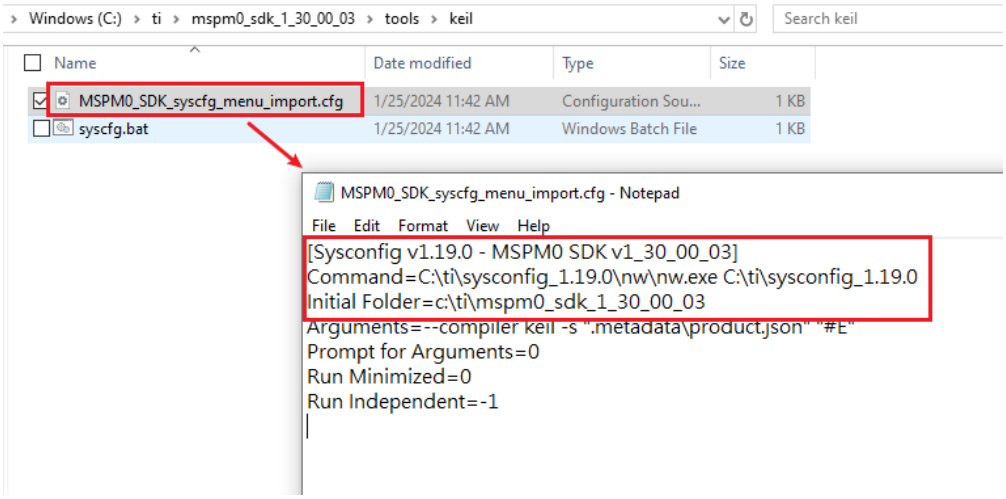


Figure 3-43. Edit MSPM0_SDK_syscfg_menu_import.cfg

3. In Keil, select *Tools* → *Customize Tools Menu* from the menu.

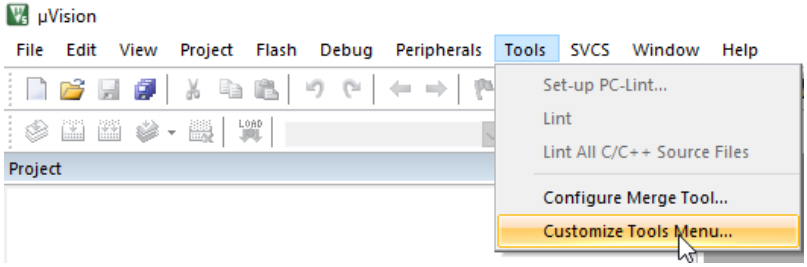


Figure 3-44. Keil Customize Tools

4. Import MSPM0_SDK_syscfg_menu_import.cfg file into the *Customize Tools Menu*.

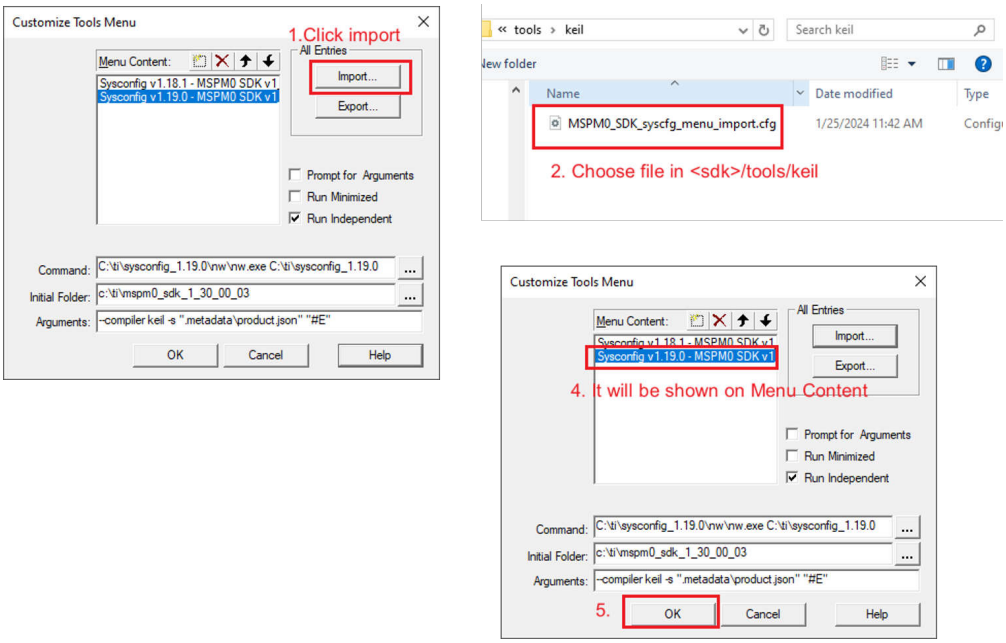


Figure 3-45. Import MSPM0_SDK_syscfg_menu_import.cfg File

5. The SysConfig entrance now appears on the menu. You can use SysConfig for MSPM0 development on Keil.

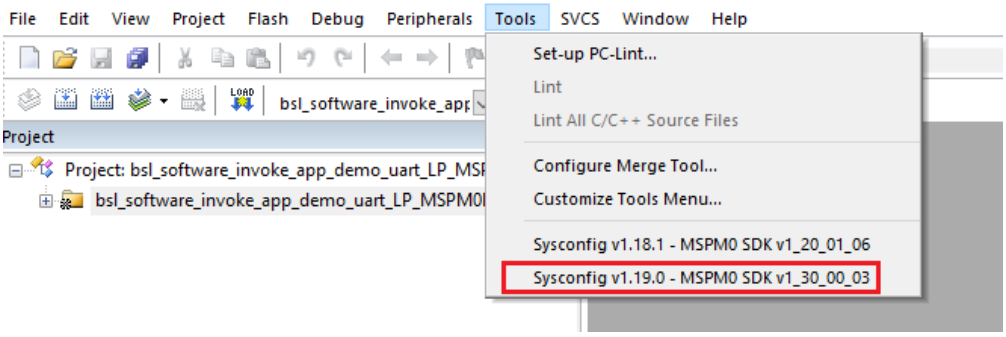


Figure 3-46. Finish SysConfig Setup

3.4.3.2 Import a SDK Example

Here is the guide that explains how to import a MSPM0 SDK example into Keil:

1. In Keil, select *Project → Open Project*.

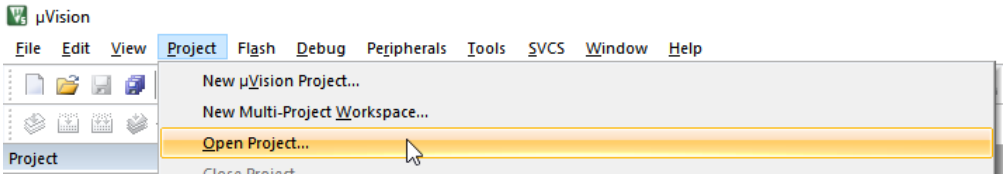


Figure 3-47. Open Project

2. Select a demo project from SDK. For the nortos example, use the .uvprojx project file. For the RTOS example, use .the uvmpw work space file. An example is shown in Figure 3-48.

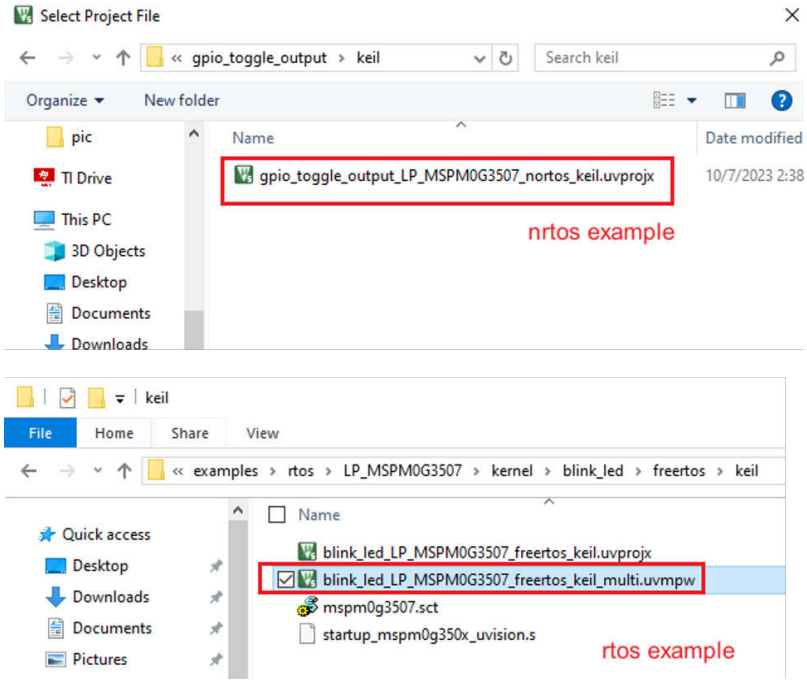


Figure 3-48. Select Keil Project

3. To open the .syscfg file, double click the .syscfg file. Then, select *Tools → Sysconfig v1.19.0 - MSPM0 SDK v1_30_00_03*. The .syscfg file opens in a separate window.

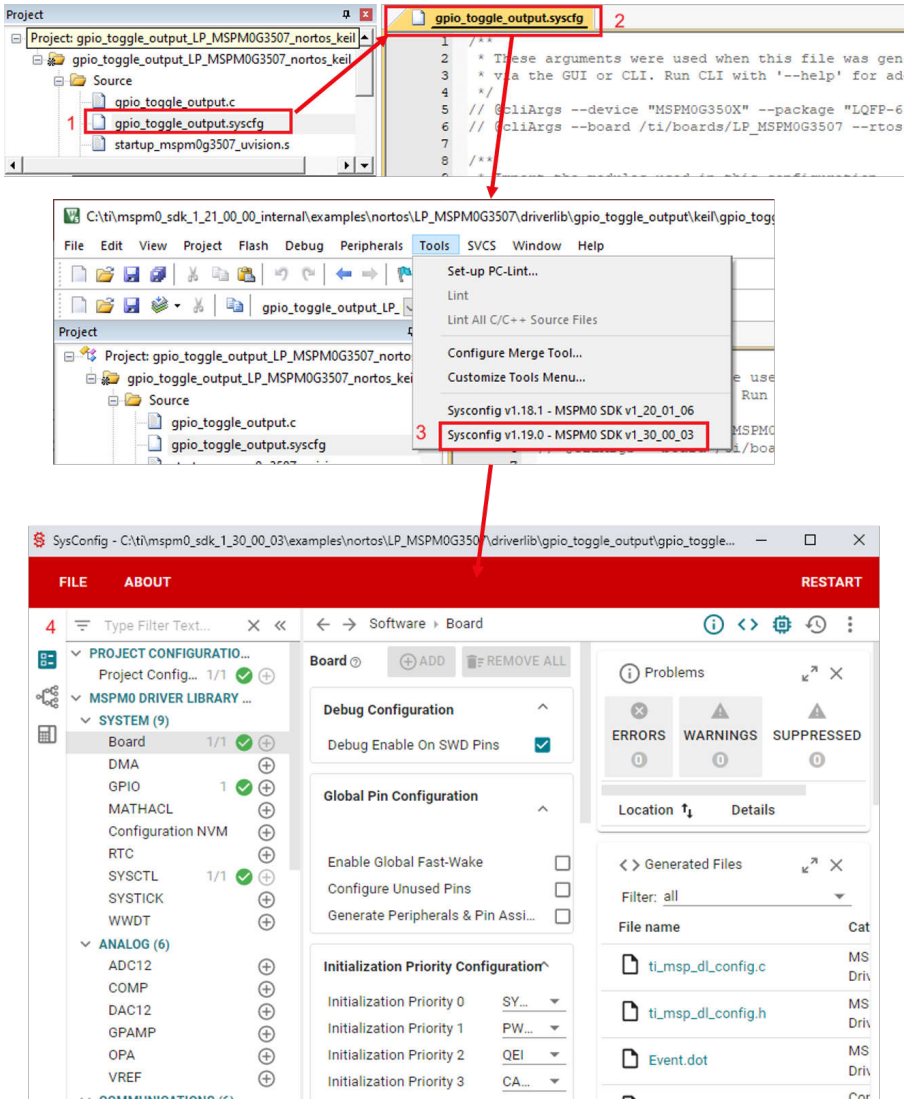


Figure 3-49. Open .syscfg file

3.4.3.3 Example Download and Debug

Here is the guide that explains how to download the code into MSPM0 based on Keil:

1. Right-click project files, then select *open options for target*

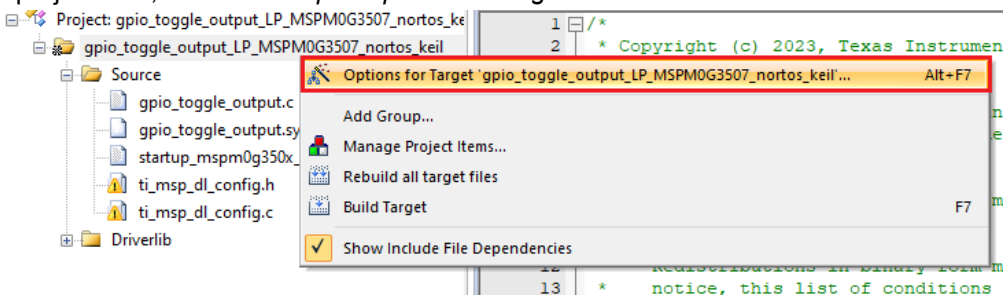


Figure 3-50. Open Options for Target

2. Select a debugger from the *Target Options* window. To use XDS-110, select *CMSIS-DAP Debugger*. If J-Link is required, then select *J-LINK/J-TRACE Cortex*.

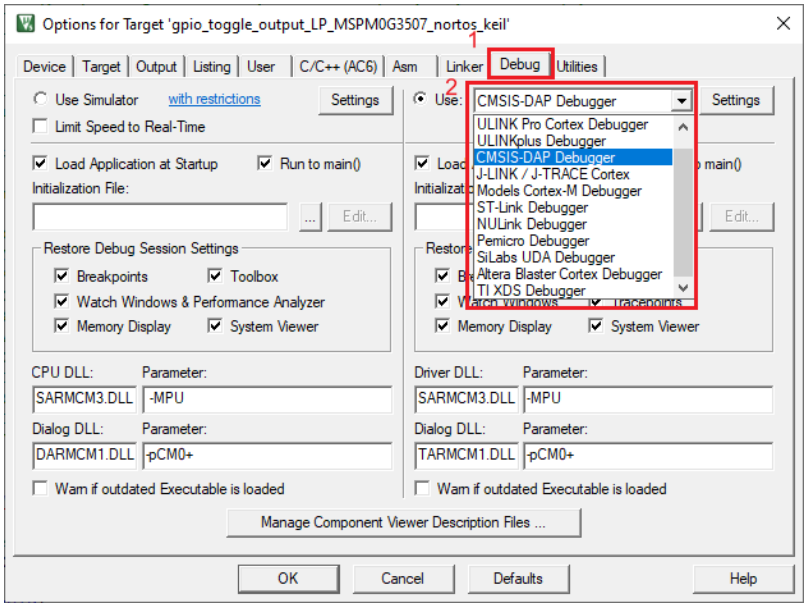


Figure 3-51. Select the Debug Pane

3. Click on the *Settings* button. On the *Debug* tab, make sure the settings match with Figure 3-52 and Figure 3-53.

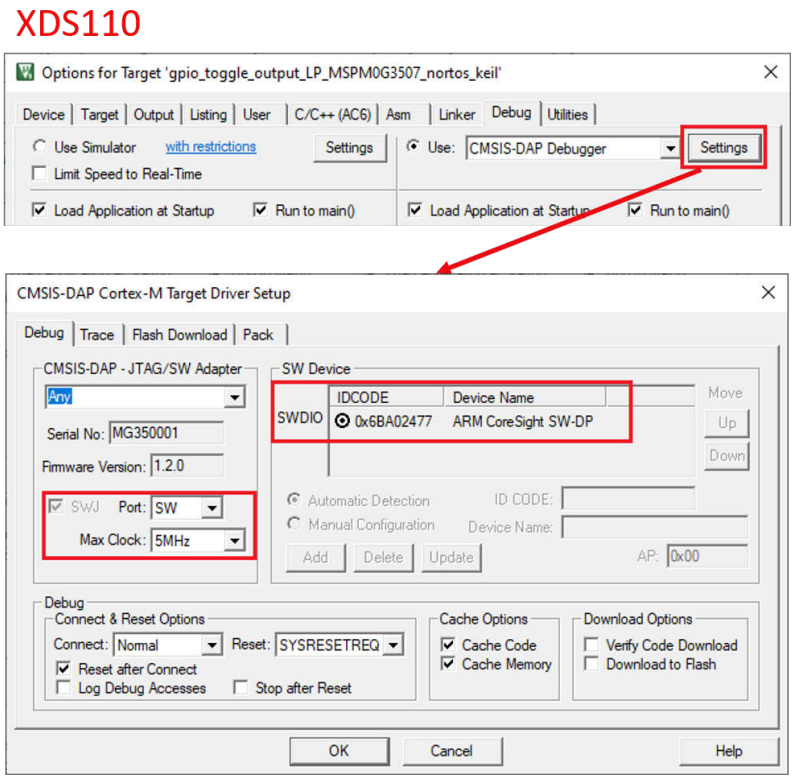


Figure 3-52. Check the Setting of XDS110 Probe

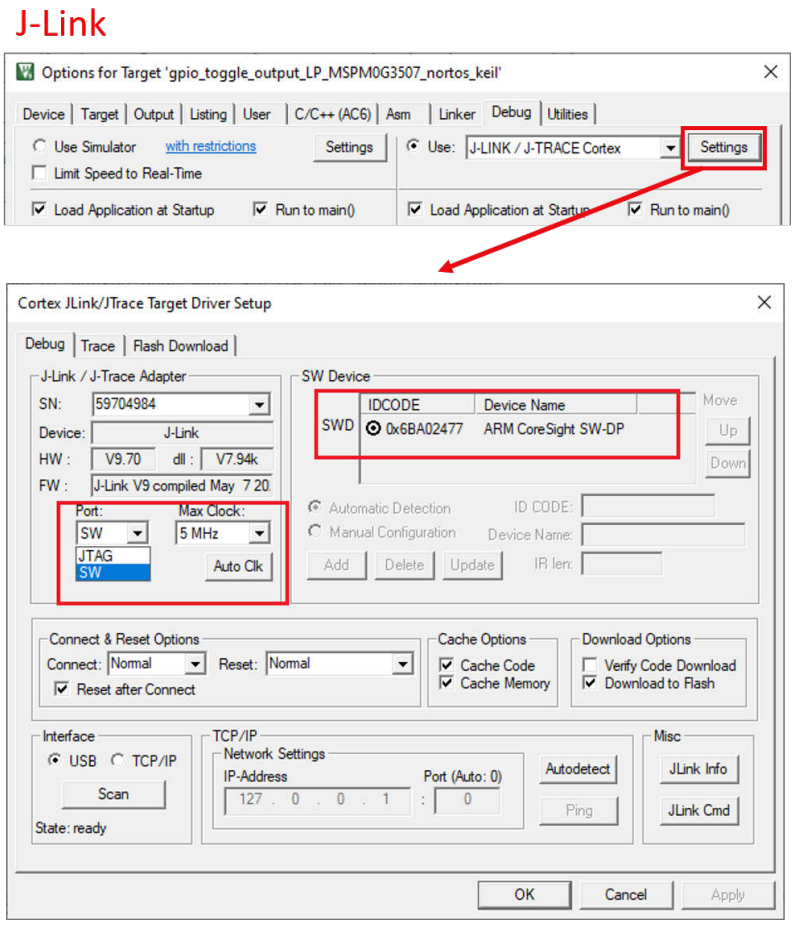


Figure 3-53. Check the Setting of J-Link Probe

4. Click on the *Flash Download* tab and check whether the description matches Figure 3-54. If this does not match, then click on the *Add* button and select the corresponding MSPM0 MAIN option. The device type is *On-chip Flash*. At last select *Reset and Run*.

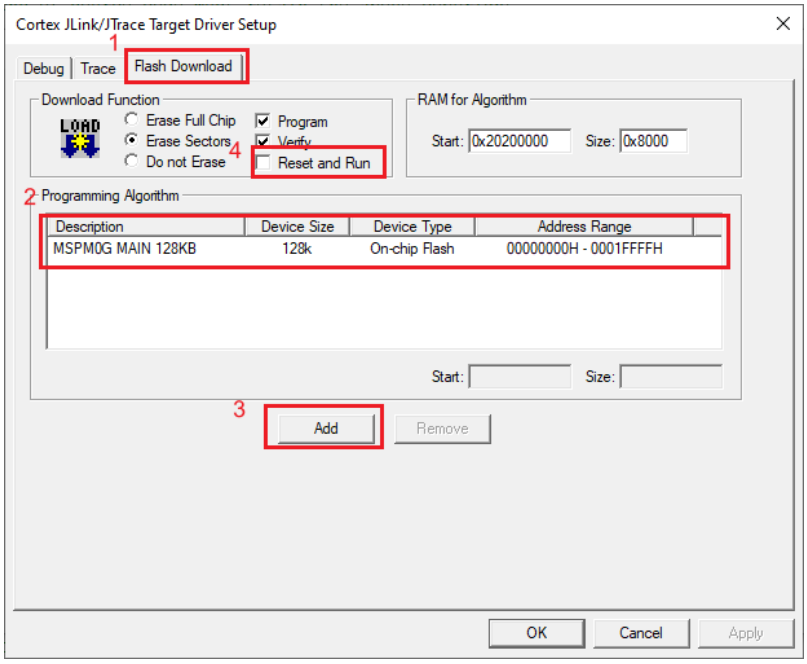


Figure 3-54. Flash Download Setting

5. Click the *Build* button to build the project, then click the *Load* button.

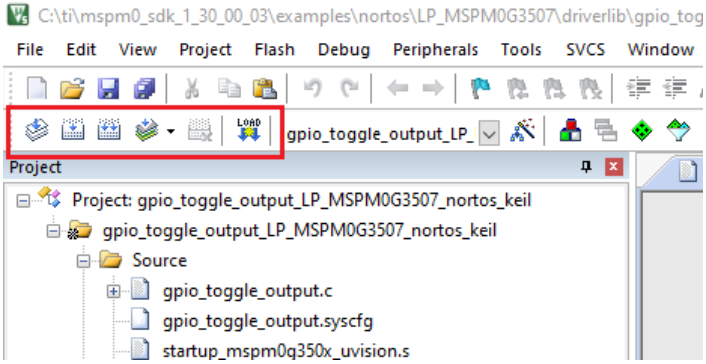


Figure 3-55. Download Project

6. To build the FreeRTOS supported example, select *Project* → *Batch Setup* and select all the project targets for the build. Next, select *Batch Build* to build all the projects in the workspace.

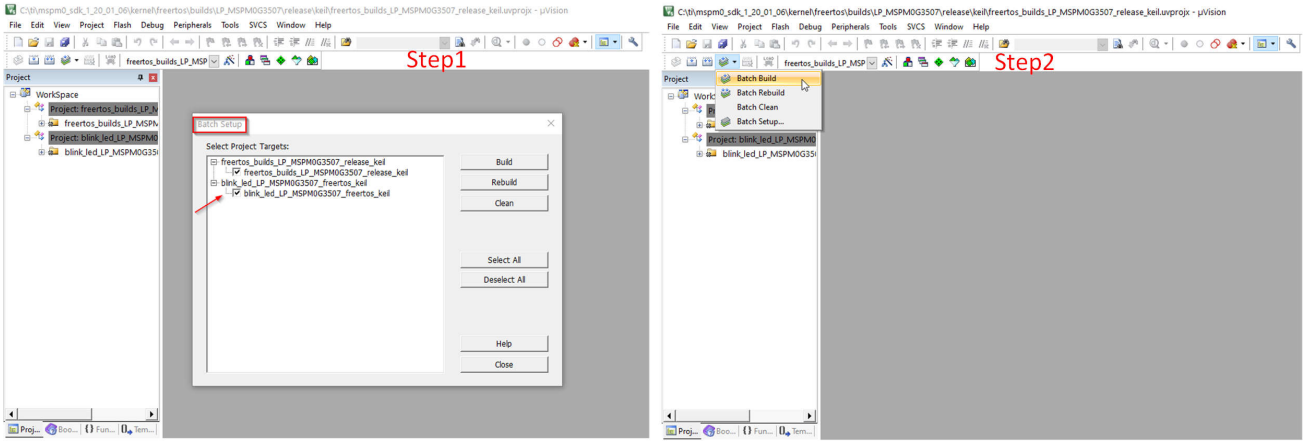


Figure 3-56. Build RTOS Example Under Keil

3.4.3.4 Migrating Between MSPM0 Derivatives

SysConfig allows for an easier migration between MSPM0 derivatives. However some manual modifications are required on Keil. Follow the steps below:

1. In SysConfig, enable the Device View and click on *SWITCH*.
2. Select the corresponding options for the new MSPM0 device and click *CONFIRM*. Note that SysConfig highlights any conflicts with the migration, such as unavailable pins and peripherals. Fix any conflicts as needed.
3. In the Keil IDE, open the *Device* tab in project options, and select the new MSPM0 derivative.
4. Update the device definition by selecting *C/C++ (AC6)* → *Preprocessor Symbols* → *Define*. Add the device definition as per the device selected.
5. Update the linker file in *Linker* → *Scatter File*. The MSPM0 SDK includes default files for all MSPM0 derivatives at *<sdk>\source\ti\devices\msp\m0p\linker_files\keil*.
6. Add the startup file of the new derivative to the project and remove existing one. The MSPM0 SDK includes default files for all MSPM0 derivatives at *<sdk>\source\ti\devices\msp\m0p\startup_system_files\keil*.

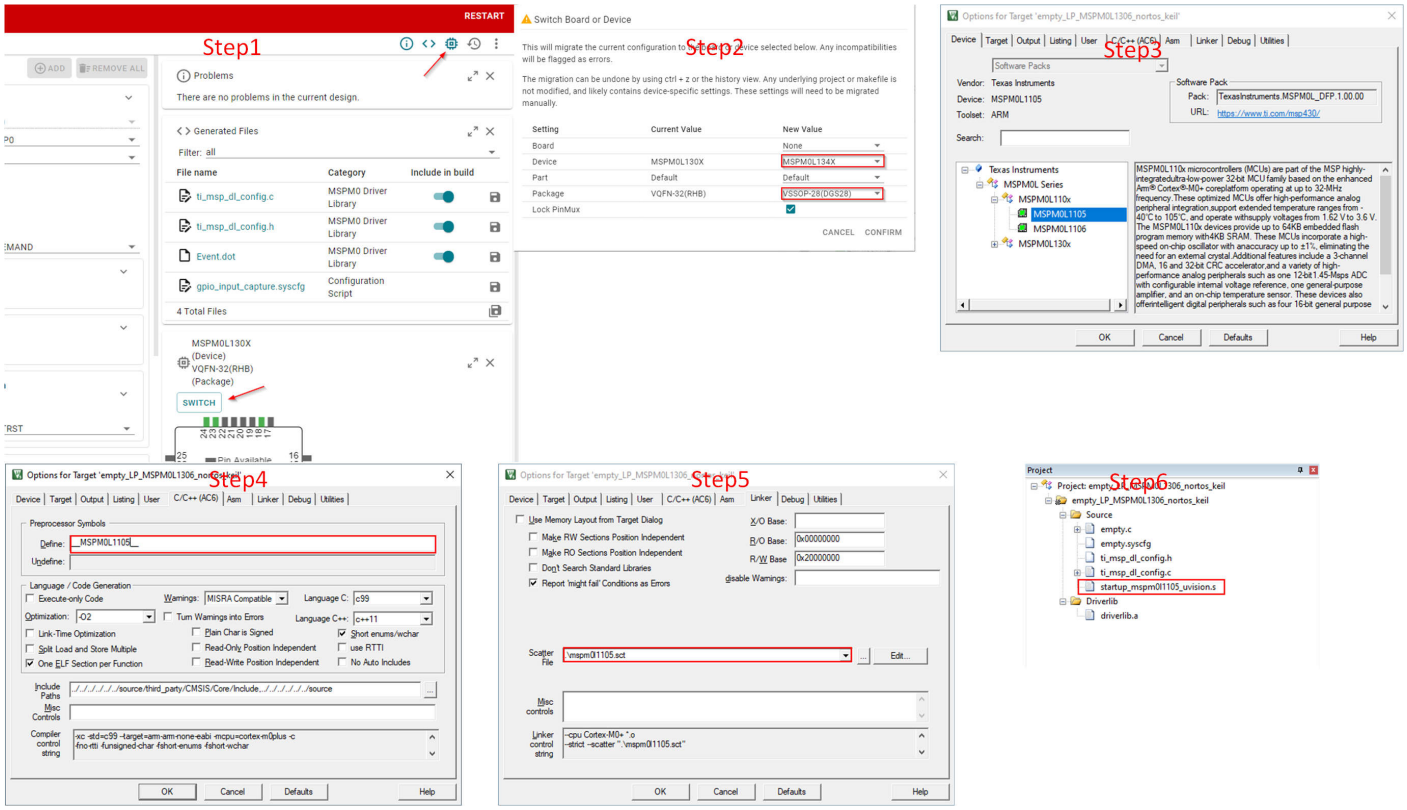


Figure 3-57. Migrating Between MSPM0 Derivatives

3.4.3.5 Generate Hex Files

Here is the instruction to generate hex files in Keil. Click *Project* → *Options* → *Output* → *Create Hex File* → *OK*. You can select the paths through click *Select Folder for Objects* to locate the HEX file. The default path is the object folder under project file.

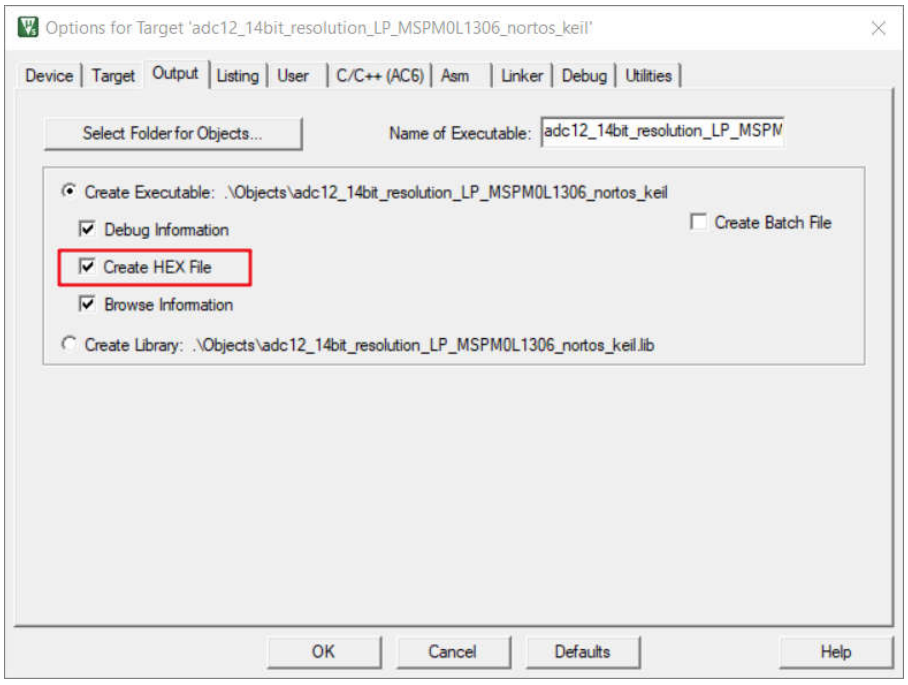


Figure 3-58. Generate Hex Files

3.4.3.6 Program NONMAIN

If users make the changes on Bootloader or MCU security setting by configuring the NONMAIN, as shown in Section 3.3.2.4, then users need to enable the NONMAIN Erase in the IAR setting as well. Follow the steps below, otherwise and keep the default:

1. Click *Options* → *Debug* → *Settings* → *Flash Download*.
2. Add the NONMAIN programming algorithm, and then click *OK*.

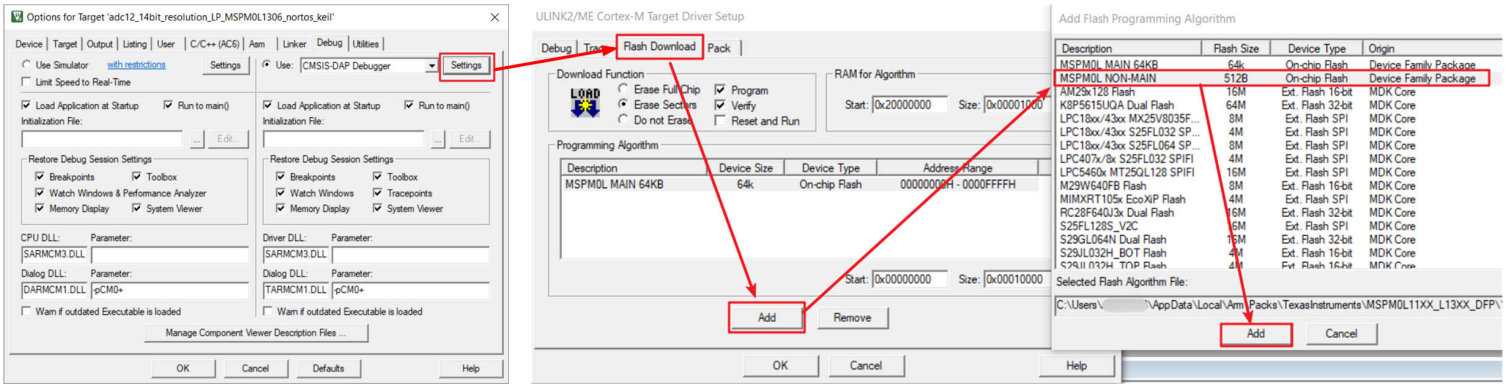


Figure 3-59. Program NONMAIN

4 Step 3: Hardware Design

4.1 Obtaining a MSPM0 Package

To obtain a MSPM0 package, use the Ultra Librarian tool on TI.com, as shown in Figure 4-1. For detailed instructions, see Section 8.

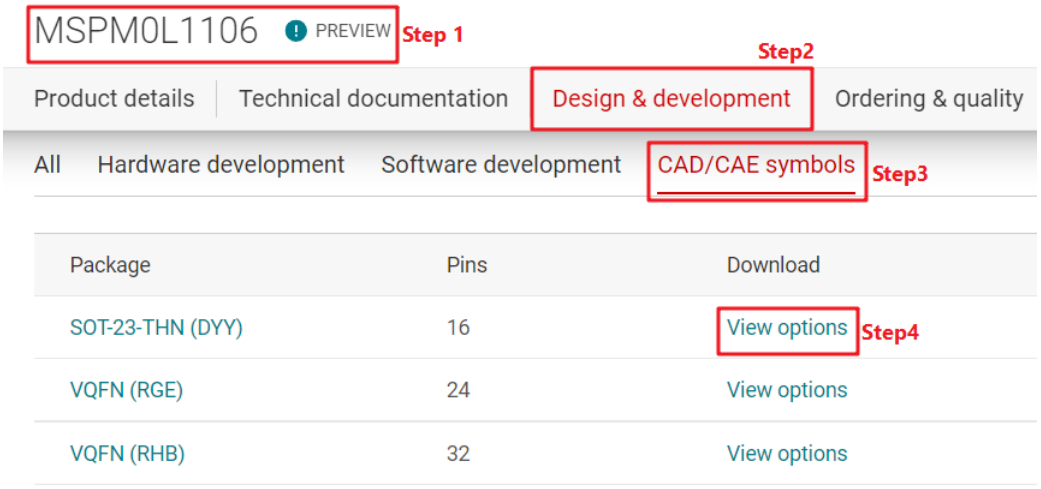


Figure 4-1. Ultra Librarian Tool Entrance

4.2 Fix Pin Functions

TI recommends hardware engineers use the *Peripherals and Pin Assignments File* to fix the pin functions with assistance from a software engineer by following the instructions in Figure 4-2.

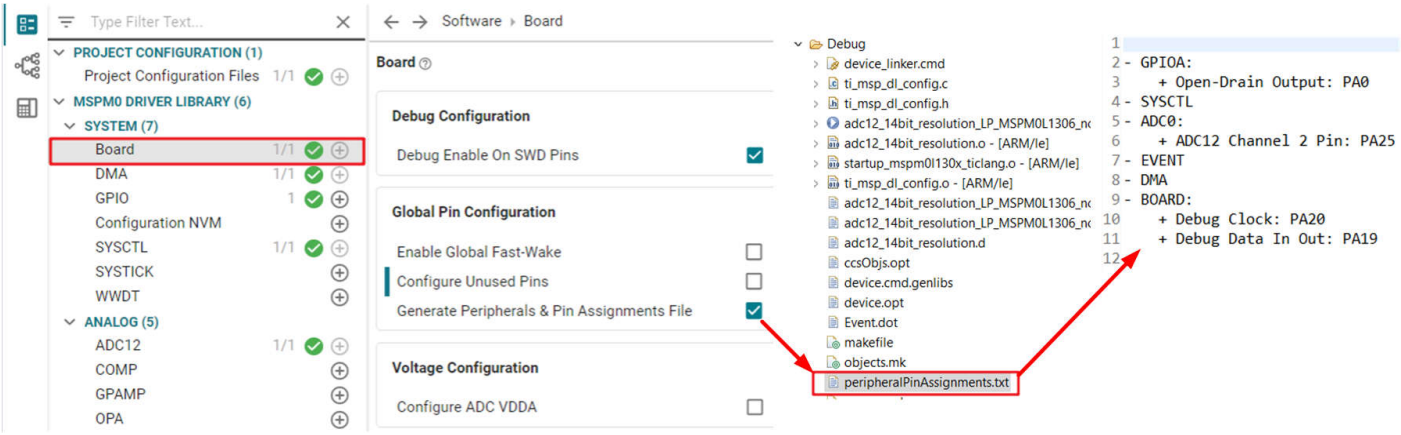


Figure 4-2. Generate Peripherals and Pin Assignments File

4.3 Schematic and PCB Generation

The minimum requirements (power, reset, and Vcore) with suggested values for MSPM0 hardware setup are shown in Figure 4-3.

- Power pin: TI recommends adding 10uF and 0.1uF capacitors, which are used to remove AC noise on the power rail.
- Reset pin: TI recommends adding a 47kR pullup resistor and a 10nF pulldown resistor. This makes sure that the MSPM0 releases from reset, after the power rail is stabilized. For some MSPM0 devices, the reset pin can be reused with another function, like I2C or UART. TI recommends reducing the resistor and capacitor, such as using a 2.2kR pullup resistor and 10pF pulldown capacitor.
- Vcore pin: This pin is used to stabilize the CPU voltage. For some MSPM0 devices, this pin is not included. If the pin is included, connect the pin to a 0.47uF capacitor.

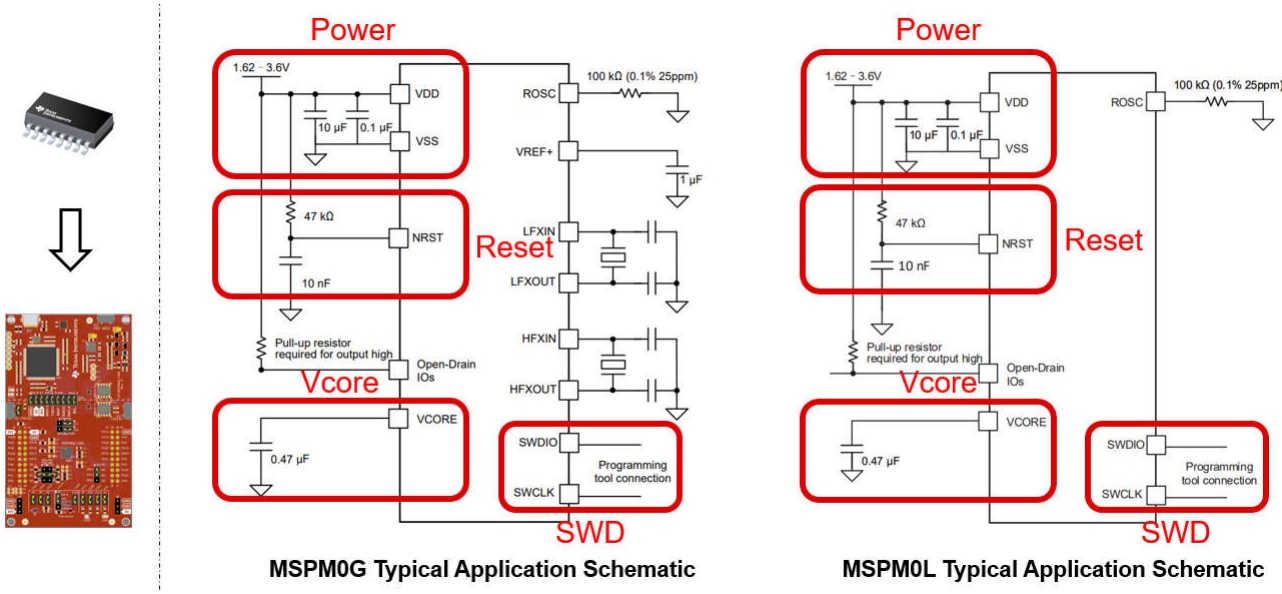


Figure 4-3. MSPM0 Minimum System

Other considerations when drawing a schematic file are listed in Figure 4-4.

- ROSC Pin: If users want to reach accurate high frequency clock with internal SYSOSC, then 0.1% resistor is suggested. Some low-cost devices cannot have this function.

- VREF+/VREF- Pin:
  - If using an internal reference,then the G series require a 1uF capacitor between VREF+ and VREF- to support 4Msps ADC. For L or C series, the capacitor is not required, as the ADC speed is only support 200Ksps with internal Vref.
  - If using an external reference, then all the MSPM0 devices require a 1uF capacitor between VREF+ and VREF-.
- Open-Drain IO: Open-Drain IO cannot output high voltage from the MCU side, so external pullup resistors are required, such as a 4.7kR capacitor.
- NRST: If reusing the reset pin as GPIO, then the pullup resistor and the pulldown capacitor are still required. This makes sure that the MCU is released from reset state after the power is stable.
- PA18: PA18 is the invoke pin to enter bootloader. Make sure this pin is not float or pullup. Otherwise, a user can change and disable the invoke pin in sysconfig, as shown in Section 7.3.

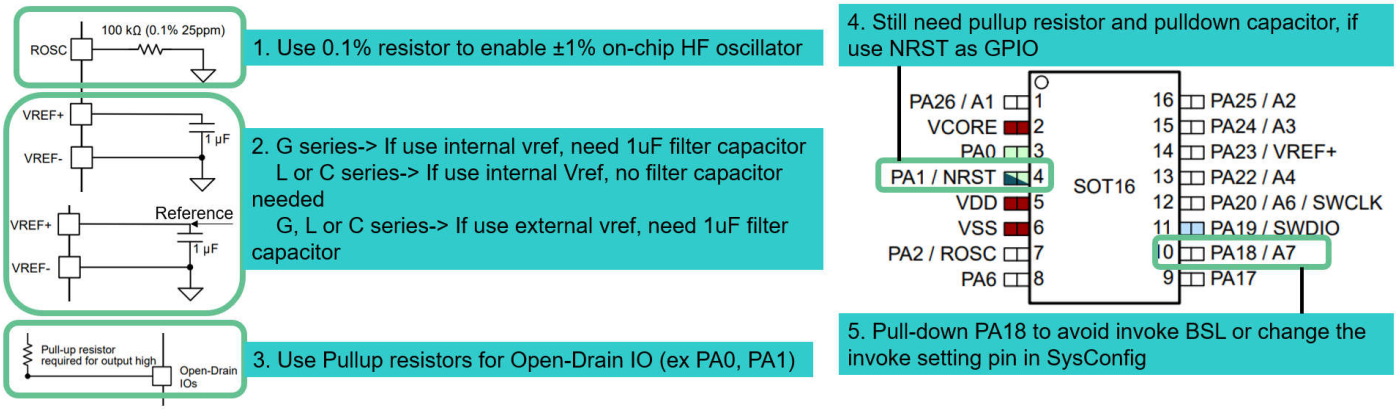


Figure 4-4. MSPM0 Schematic

For further information about schematics or PCB design references, see the following links.

- *MSPM0 L-Series MCUs Hardware Development Guide*
- *MSPM0 G-Series MCUs Hardware Development Guide*
- Device-specific MSPM0 Launchpad EVM user's guide
- Device-specific MSPM0 data sheet

5 Step 4: Mass Production

An overview of the program software and tools is shown in Figure 5-1. The available interface is JTAG (SWD) and Bootloader (BSL). For J-Link only supports SWD. For XDS110 and MSP-GANG supports SWD and Bootloader over UART.

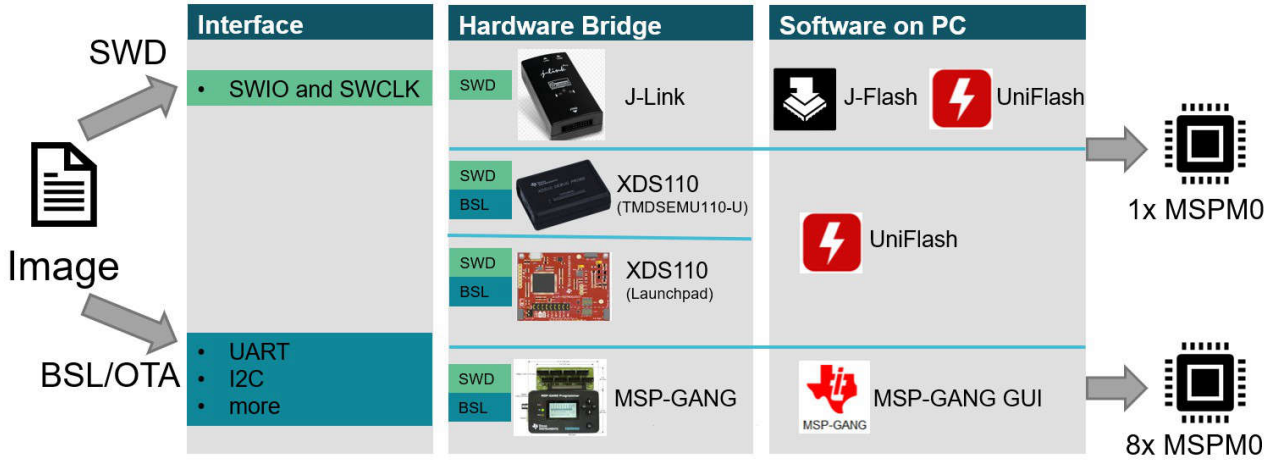


Figure 5-1. Program Software and Tools

For more implementation about bootloader, see MSPM0 Bootloader (BSL) Implementation. For more production programming tools, see E2E page.

5.1 Generate Production Image

Table 5-1 lists different types of image generated by different IDEs. For the step by step generation guidance, see Section 3.4.

Table 5-1. Product File Generated by IDE

IDE	TI_TXT (.txt)	Intel hex (.hex)	bin (.bin)	Step by Step Guidance
CCS	Y	Y	Y	Link
IAR	Y	Y	Y	Link
Keil	N	Y	N	Link

5.2 Program Software

5.2.1 Uniflash Quick Start

This section describes how to install the UniFlash tool with TI's MSPM0 devices. See the UniFlash Quick Start Guide for more information.

5.2.1.1 Program Through SWD

The debugging interface such as XDS110 can be used by UniFlash to program the device. The needed hardware pins are SWDIO, SWCLK, 3V3 and GND. Follow the steps below:

1. Follow the steps to select the debugger (either XDS110 or J-Link). Then click *Start* to start program.
2. If NONMAIN must change, change the erase setting before programming. If this is not required, keep the default option.
3. Select the image and start to program by clicking *Load Image*.
4. Using the *Memory* tab, UniFlash can also inspect the flash memory of the device simply by selecting *Read Target Device*.

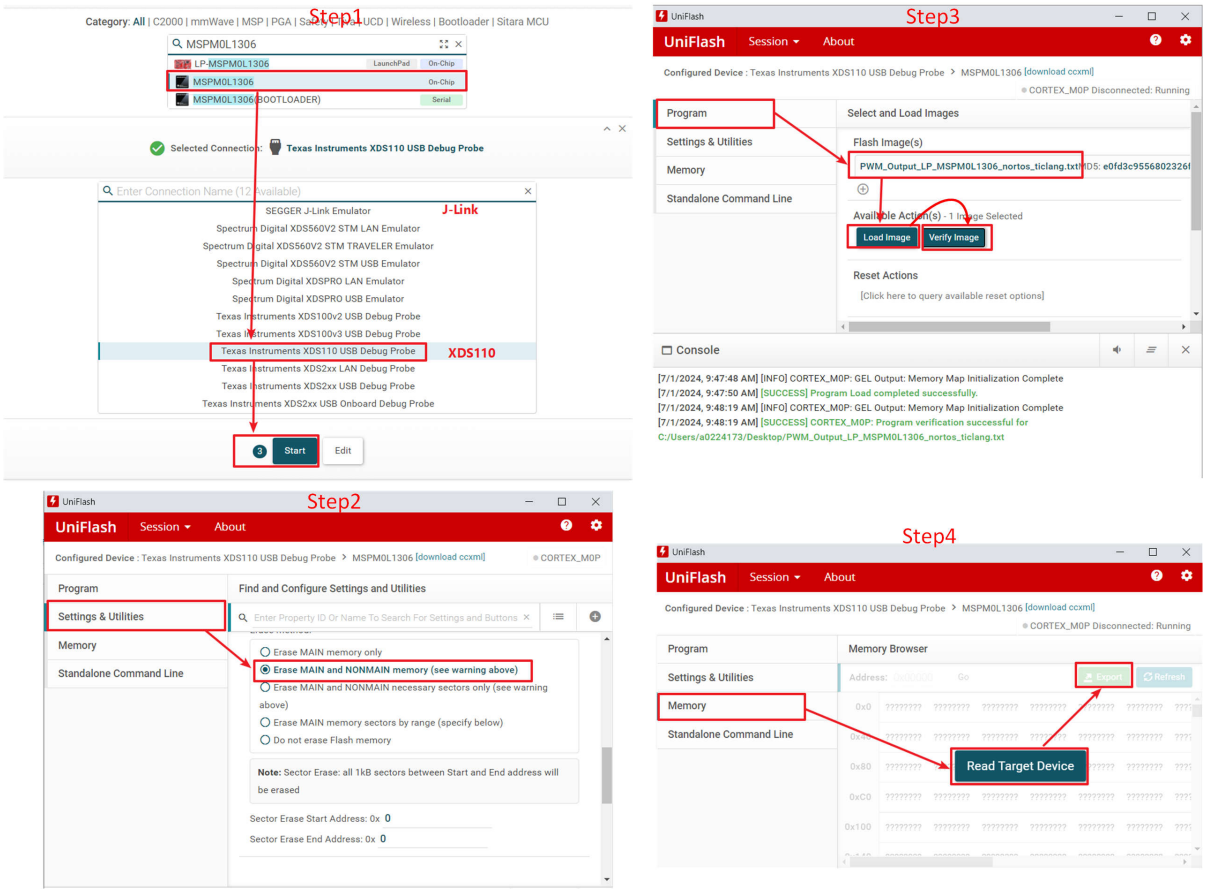
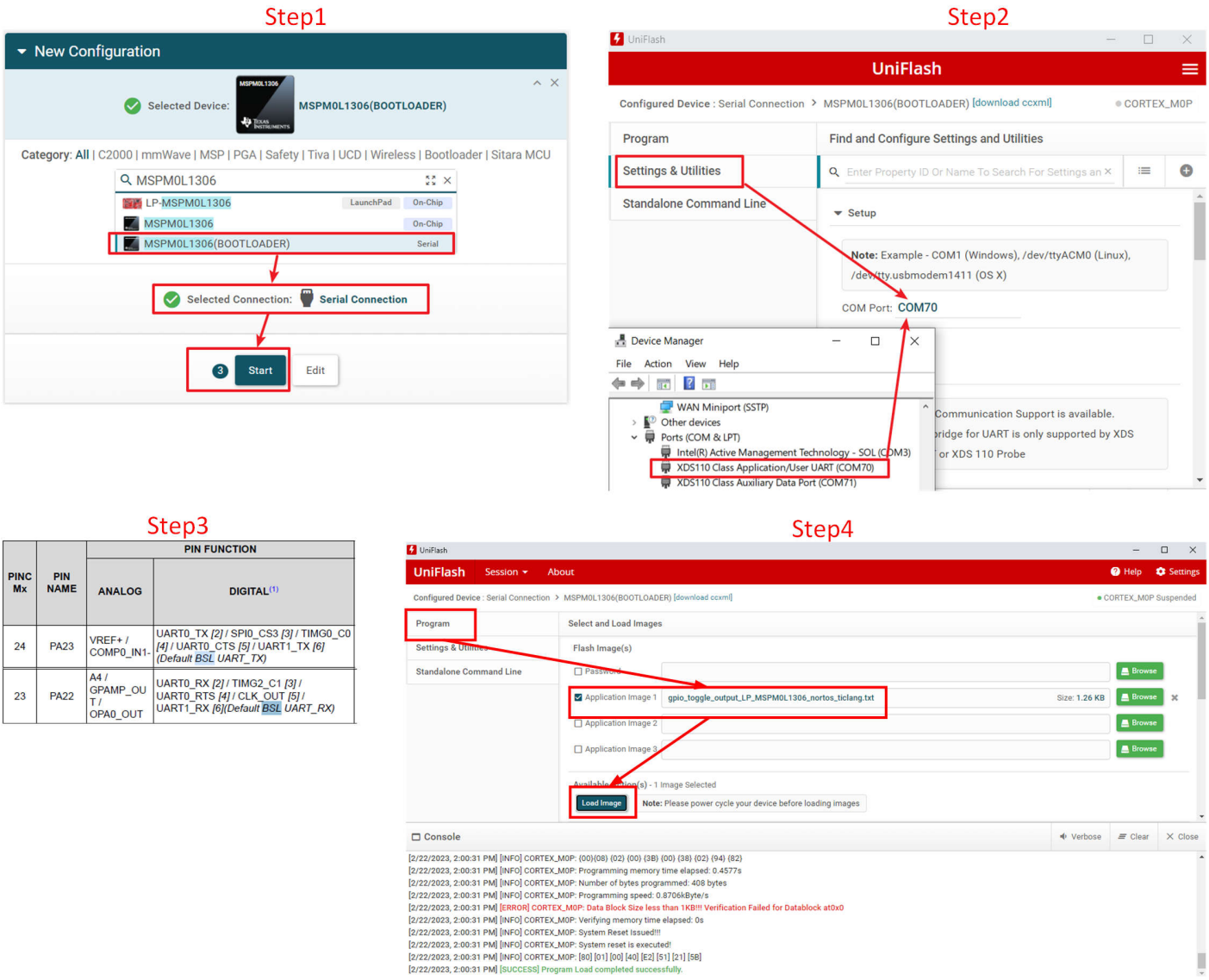


Figure 5-2. Program Through SWD

5.2.1.2 Program Through Bootloader

Here are the steps to program MSPM0 through bootloader using Uniflash. The required hardware pins are TX, RX, 3V3, GND and invoke pins.

1. Search the device name and select the bootloader option for the device.
2. Check the COM port by referring to the device manager.
3. Check the UART Bootloader port by referring to the data sheet.
4. Finish the hardware connection (RX, TX, 3V3, GND, Invoke) and start program.



PINC Mx	PIN NAME	PIN FUNCTION	
		ANALOG	DIGITAL(1)
24	PA23	VREF+ / COMP0_IN1-	UART0_TX *[2]* / SPI0_CS3 *[3]* / TIMG0_C0 *[4]* / UART0_CTS *[5]* / UART1_TX *[6]* *(Default BSL UART_TX)*
23	PA22	A4 / GPAMP_OUT / OPA0_OUT	UART0_RX *[2]* / TIMG2_C1 *[3]* / UART0_RTS *[4]* / CLK_OUT *[5]* / UART1_RX *[6](Default BSL UART_RX)*

Figure 5-3. Program Through Bootloader

5.2.1.3 Program Through CMD Line Interface

For this requirement, see this E2E thread.

5.2.2 JFlash Quick Start

This instruction is based on J-Flash V7.92n. TI recommends using the latest J-Flash version, which supports all the latest versions of MSPM0. Use the following steps to program MSPM0 with J-Flash:

1. Click *New project.*
2. Select the related MSPM0 part number.
3. Select the desired programming memory. If NONMAIN does not need to change, deselect *NONMAIN memory*.
4. Click *Connect device* and click *Production Programming*.
5. A confirmation screen appears.

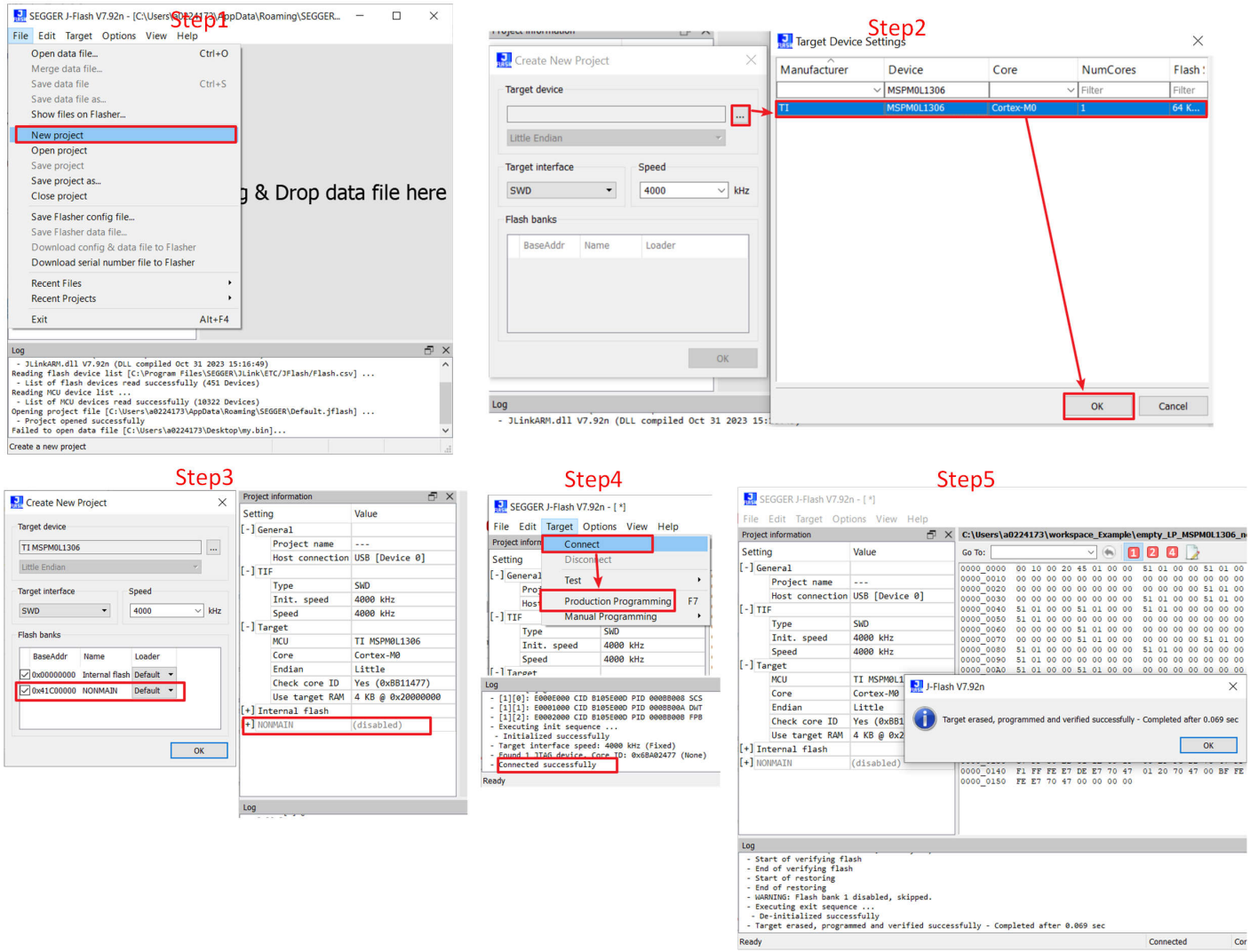


Figure 5-4. J-Flash Quick Start

5.2.3 MSP-GANG GUI Quick Start

The MSP Gang Programmer (MSP-GANG) is a device programmer that supports MSPM0L110x, MSPM0L130x, MSPM0G150x, MSPM0G350x. **At this time, any MSPM0 devices not stated above is not supported by the MSP-Gang Programmer.** Please refer to E2E page for alternative production programming tools.

For the quick start guidance for MSP-GANG, refer to Section 8.3.

5.3 Program Hardware

Due to J-Link is commonly used, this section focuses on the XDS110 debugger. For more production programming tools, see E2E page.

There are four different types of XDS110 debuggers available. The summary table is listed Table 5-2.

Table 5-2. XDS110 Debugger Summary

Support Features	XDS110	XDS110 On Board		
	TMDSEMU110-U	MSPM0 LaunchPad	LP-XDS110	LP-XDS110ET
JTAG	Yes	No	Yes	Yes
SBW	Yes	Yes	Yes	Yes
EnergyTrace	Yes	Rely on type	No	Yes
MSPM0 bootloader	Yes	Rely on type	No	No
Comment	Highest Performance	Cheapest	Easy to use	Easy to use

With the TMDSEMU110-U device, the pin that is used is shown in Figure 5-5. When using for bootloader, GPIOOUT0 must connect to the MCU reset pin. GPIOOUT1 must connect to the MCU invoke pin (PA18).

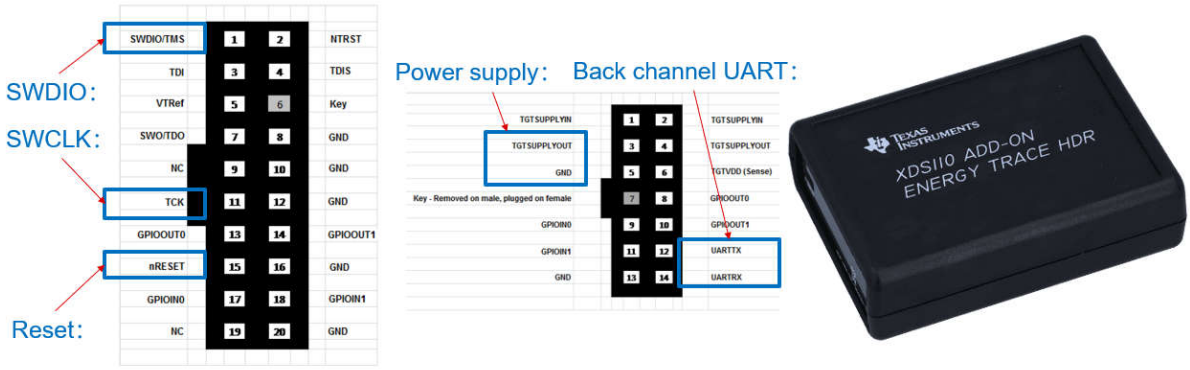


Figure 5-5. Pin Connection of TMDSEMU110-U

For XDS110 on LaunchPad, the basic programming functions are intact compared to the TMDSEMU110-U. The board is shown in Figure 5-6. The cheapest XDS110 on LaunchPad is LP-MSPM0C1104. However, LP-MSPM0C1104 only supports SBW and there is no EnergyTrace or bootloader function.

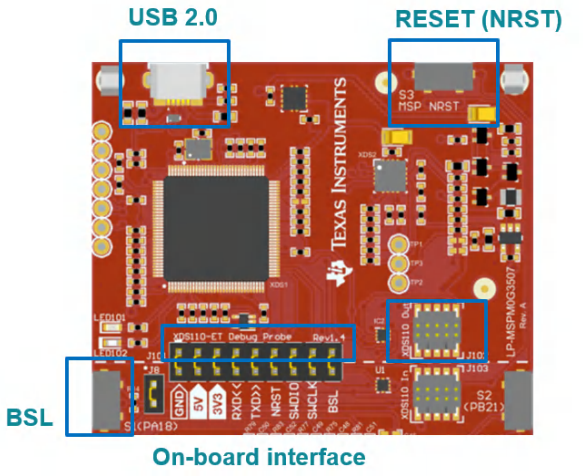


Figure 5-6. XDS110 Onboard

LP-XDS110 and LP-XDS110ET are similar with XDS110 on a LaunchPad. The difference lies on that one has EnergyTrace function and the other does not. The pin assignment is shown in Figure 5-7.

For LP-XDS110 and LP-XDS110ET, the level shift function is enabled by changing the jumper at the left bottom of the board. The support voltage range is from 1.2V to 3.6V.

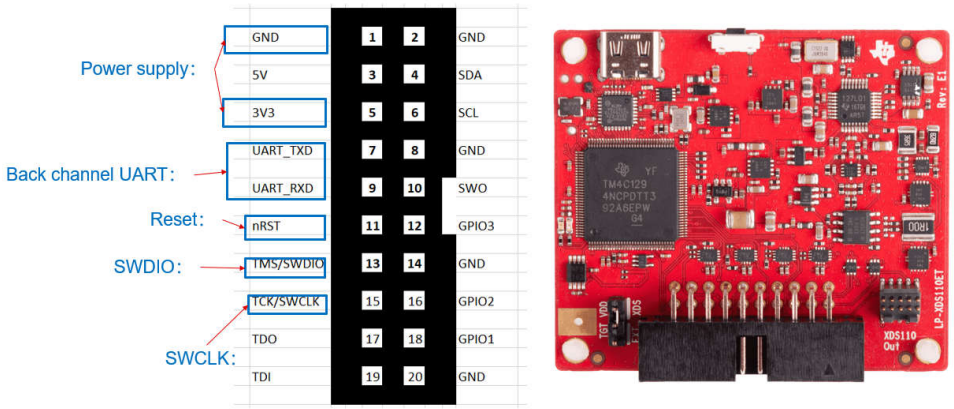


Figure 5-7. LP-XDS110ET

6 Step 5: Quality and Reliability Instructions

TI is committed to delivering high quality and reliable semiconductor designs that meet our customers' needs. Our holistic approach to quality permeates every aspect of the company's supply chain from process technology and design through manufacturing, packaging, test and delivery.

6.1 Quality and Reliability Material Entrance

This is the landing page for Quality & reliability. The following are the common used tools and links under that page:

- Qualification summary★: Used to search reliability data of related devices. Representative data summary of the material sets, processes, and manufacturing sites used by the device family.
- Reliability testing: Listed the various types of testing that TI conducts for reliability of the products.
- Customer returns: The *Customer Returns* page provides detailed guidelines for returning material to TI.
- DPPM/FIT/MTBF estimator: The DPPM/FIT/MTBF estimator search tool allows you to find generic data based on technology groupings to estimate these typical questions and shows conditions under which the rates were derived.
- Ongoing reliability monitoring: The search tool of ongoing reliability monitor (ORM) program provides the quarterly ORM report by wafer fab process or device package family.
- Packaging: This website allows users to find package considerations including package size, SMT recommendations, reliability, and performance expectations.

6.2 Failure Information Collection and Analysis Guidance

Doing failure analysis needs to collect as much technical background information as possible to narrow down the scope of analysis and accelerate the analysis speed. If users meet any device failure on MSPM0, then collect the information as below, and connect TI through Customer returns page or the Regional CQE and Sales supporting your product or business.

Device name (TI Part Number, including package designator):

- Example: MSPM0L1306SRGER

Failure rate (purchased vs. customer failed units):

- Example: Failure rate: 5% (Total tested qty: 2000, Failed qty: 100)

Detection place (field return, production, incoming …):

- Example: Board level function test

Schematic of the application:

- Example: Schematic of the MCU part, with detailed description to every input and output signals

Detailed device level failure description

- Example: MCU PA1 can't output high voltage

This is an introduction to the common methods to collected the failure information.

- Method 1: ABA swap test. In some failure conditions, this is hard to judge whether the issue is caused from the device or the relativity between the device with the total system. ABA swap test is a good method to solve this problem and provide more information for further analysis. Here are the steps to do ABA swap test: Remove the suspected component (A) from the original failing board. Replace the suspected component (A) with a known good component (B) and check if the original board now works properly. Mount the suspected component (A) to a known good board and see if the same failure occurs on the good board.
- Method 2: Check MCU current consumption under the low power mode. Some device failure is caused from EOS (Electrostatic Overstress). This can also be detected after setting the MCU to a low power mode, like standby mode. As the EOS causes additional leakage current, users find the current consumption is above the data sheet spec.
- Method 3: Pin impedance check. Some EOS (Electrostatic Overstress) is purely happened at I/Os, and using pin impedance check can easily catch this failure to give more information to TI. A formal method is to use the curve trace analysis (CT). However, end users can also use a multimeter to do the roughly evaluation.

Users can choose to detect the IO resistance with or without powering the device. The resistance of a GPIO in high impedance state needs to be MΩ level. However, this is safer to test the device with power, because the multimeter applies a current to the detected note, which can cause damage to the shutdown device.

- Method 4: Find a smallest system or code example. Some failure happens with the typical application and typical code project. Through comparison method, removing the unrelated hardware setup and software code step-by-step can gradually narrow down the scope of analysis. The best result is that the problem is purely related to the device and a simplest code example. With that, TI can carry the further failure analysis faster.

7 Common Questions

This section lists some common questions for users to search. For further questions, search the device-specific data sheet, technical reference manual, or E2E. TI engineers provide response in 24 hours on this online support platform.

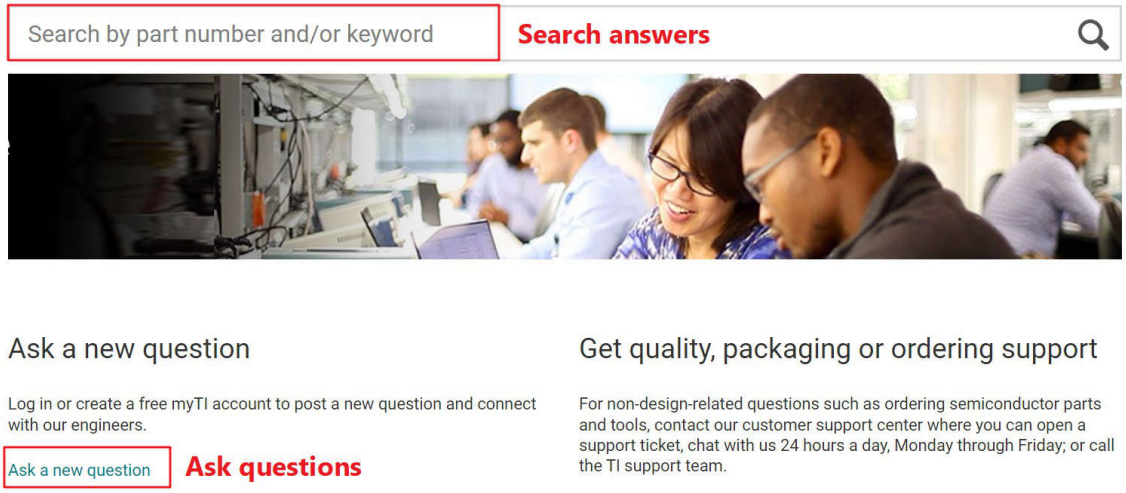


Figure 7-1. E2E Online

7.1 MSPM0 Program Failure

If the program failure is met for the first time, then check these items one by one:

1. Install the latest IDE or programming software tools at the English path. The default install path is suggested. For install instructions, please see the related chapter in this note.
2. Plug in the debugger and check whether the debugger is found by the computer. Check for computer limitations if the debugger does not show like Figure 7-2.

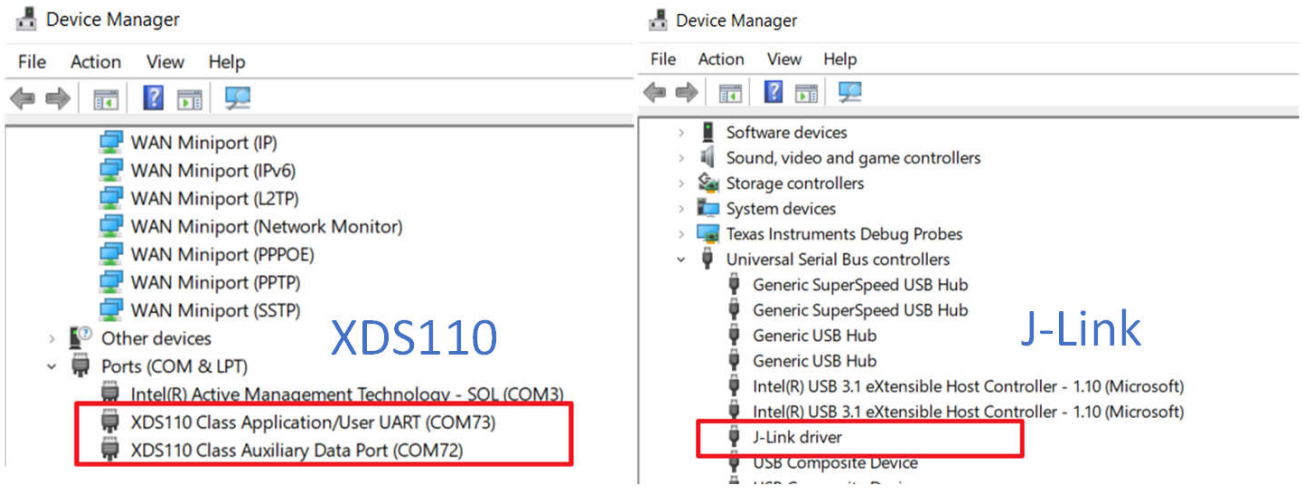


Figure 7-2. Device Manager View

3. Try to program with MSPM0 Launchpad to check whether the PC environment setting is OK.

4. For your customized board, check the schematic by referring to Section 4.3. Pay attention to the Vcc, Vcore and reset pin setting.
5. Then, check the connection between the debugger and the MSPM0. Users can use multimeter to directly check the signal path at debugger side by referring to Section 5.3, and at MCU pin side by referring to the related data sheet.
6. Check the power supply on the board. Remember the power output of the debugger has limitations and the output voltage can only be 3V3. An additional power supply can be needed.
7. Use oscilloscope to check the signal wave on SWDIO and SWCLK, especially when the wire is very long. Please make sure the signal establishment time is enough.

If the program failure is met for the second time and the device can be programmed before, then refer to Section 7.2.

7.2 Unlock MCU

MSPM0 can experience SWD connection issues when going into STOP, STANDBY, or SHUTDOWN mode. The effect of this limitation depends on the IDE and debugger implementation. Please use the tools with the latest versions, shown in Table 7-1. For more details, please refer to the Debugging in Low Power Modes chapter in the MSPM0 SDK Known Issues and FAQ.

Table 7-1. Tools Suggested Version

Keil CMSIS Pack	IAR IDE	CCS IDE	J_Link
MSPM0L11XX_L13XX_DFP: 1.3.1+ MSPM0G1X0X_G3X0X_DFP: 1.3.1+ MSPM0C110X_DFP: 1.1.1+ MSPS003FX_DFP:1.1.0+ MSPM0L122X_L222X_DFP:1.1.0+	9.60.1+	12.80+	V 8.10+

MSPM0 can also lose connection after downloading a wrong code, and CCS reports errors when programming a new code. An example is shown in Figure 7-3.

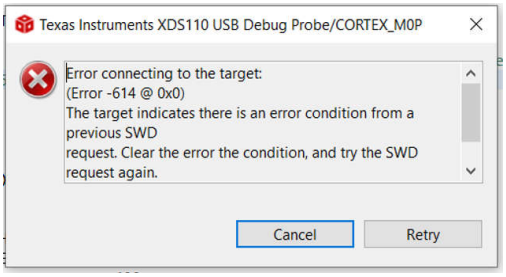


Figure 7-3. CCS Error

The Debug Subsystem Mailbox (DSSM) enables a debug probe to pass messages to the boot ROM of an MSPM0 device through the SWD interface. There are four unlock commands that you can choose in tools. The brief introduction is in Table 7-2. **DSSM Factory Reset is recommended**, which the reset level is higher than DSSM Mass Erase.

Table 7-2. Unlock Commands

Unlock Commands	Hardware Connection With Debugger	Reset Pin Control	Command Influence
DSSM Factory Reset Manual	3v3, GND, SWDIO, SWCLK, Reset	End users	Erase main flash and reset NONMAIN flash
DSSM Factory Reset Auto		Debugger	
DSSM Mass Erase Manual		End users	Erase main flash
DSSM Mass Erase Auto		Debugger	

The suggestion on the provided three unlock methods is shown in Table 7-3. An important note is that the unlock method only supports XDS110 and does not support J-Link currently.

Table 7-3. Unlock Method Selection

Unlock Method	Support Debugger	When to Choose
Factory Reset GUI Tool	XDS110	Internet connection is available
Uniflash	XDS110	Internet connection is unavailable
CCS	XDS110	Use CCS as the development IDE

7.2.1 Unlock Through Factory Reset GUI Tool

The MSPM0 Factory Reset GUI tool is a standalone tool used to gain debug access or recover an MSPM0 device using this interface. This tool is available free of charge. Follow the steps to reset the MSPM0.

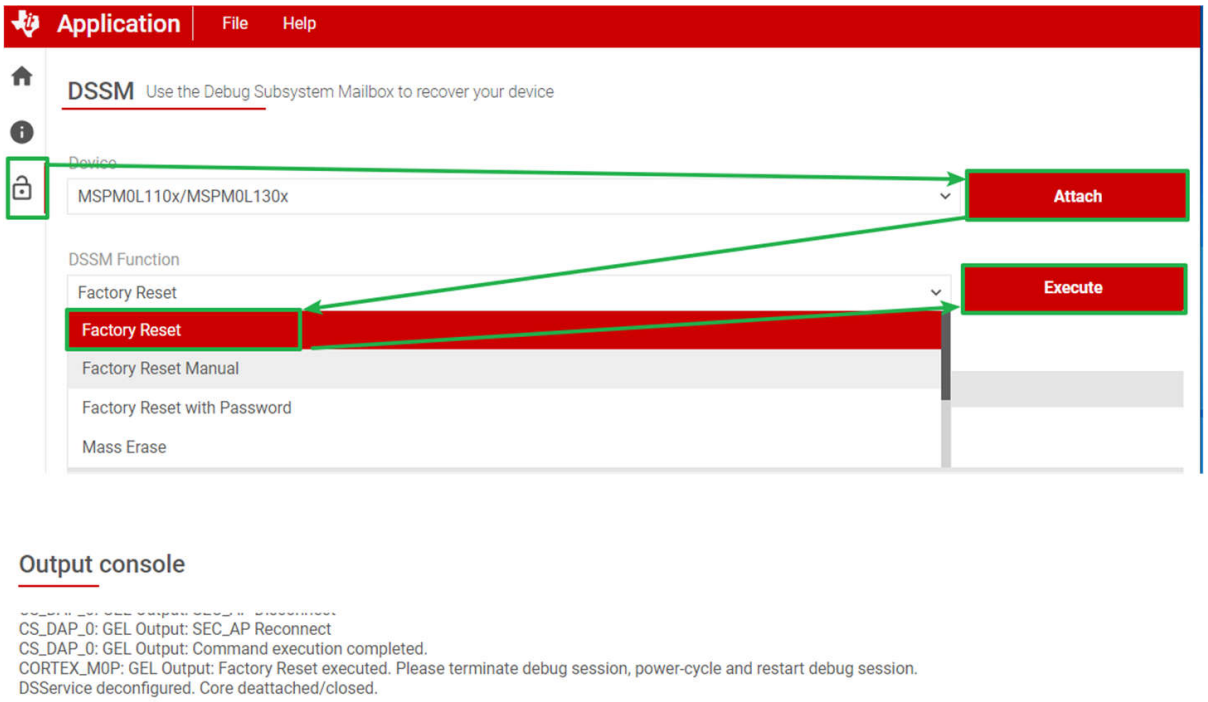


Figure 7-4. Unlock Through GUI

7.2.2 Unlock Through Uniflash

Uniflash above Version: 8.7.0.4818 also supports to unlock MSPM0. First, follow the steps to connect the MSPM0 with Uniflash, as shown in Section 5.2.1.1. Then, follow the instructions to unlock MSPM0 in Figure 7-5.

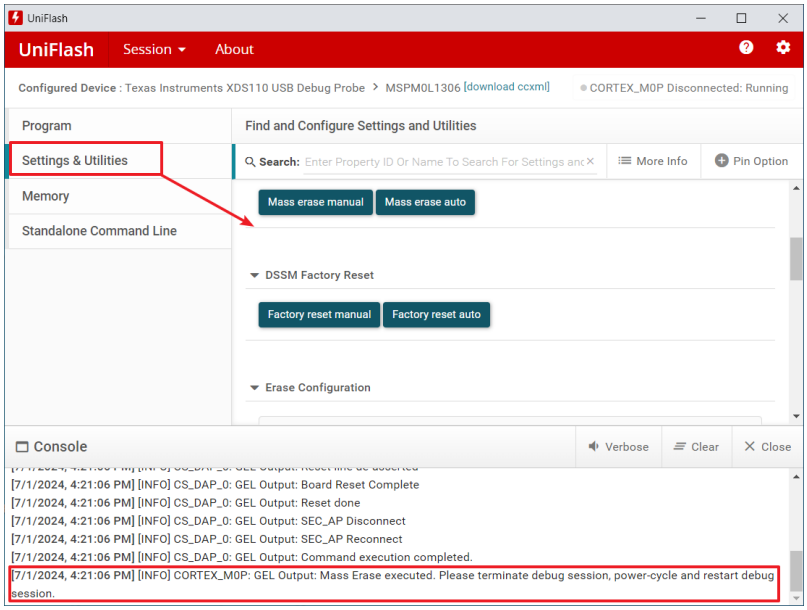


Figure 7-5. Unlock Through Uniflash

7.2.3 Unlock Through CCS

Here are the steps to unlock MSPM0 through CCS:

1. On the CCS menu, select *View* → *Target Configurations*. On the *Target Configurations* window, right-click the .ccxml of an active project and select *Launch Selected Configuration*.
2. Click *Debug Probe*, and select *Scripts* → *MSPM0xxxx_Commands*.
3. If you choose manual command, you need to reset the device according to the command in the consule. After that, you can repower the device. If you choose auto command, you can repower the device following the instruction.

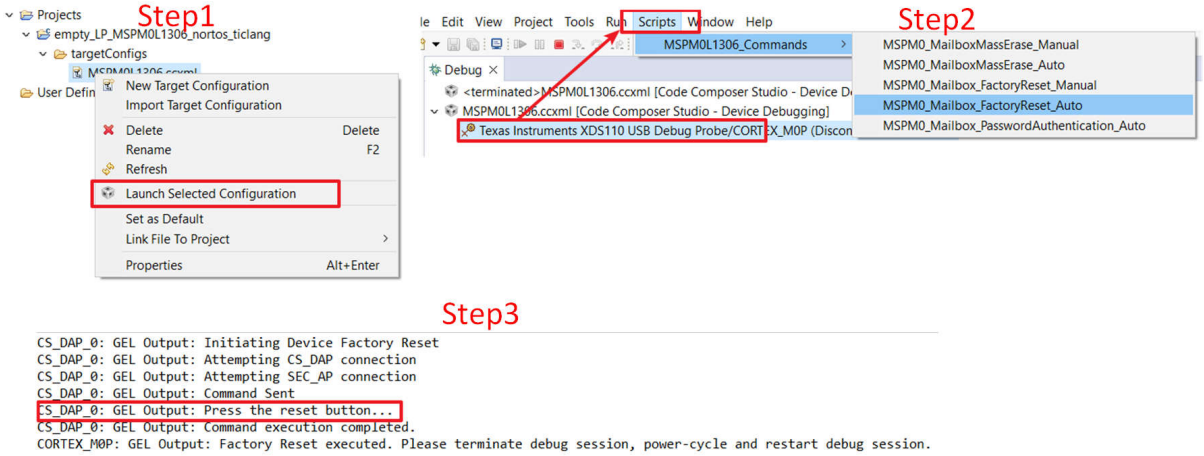


Figure 7-6. Unlock Through CCS

7.3 MCU Performs Differently in Debug and Free Run

MSPM0 performs differently in debug and free run. Check the setting on PA18. The device enters the Bootloader in free run mode after MSPM0 is reset or repower, when PA18 input is **pulled to a high level** or **affected by noise with this pin floating**. If you meet this problem and PA18 cannot be pulled to a low level with an external resistor, you can follow the steps in Figure 7-7 to disable BSL or change the invoke pin assignment. As these settings need to change NOMAIN, please refer to the Program NONMAIN chapter for the related IDE in Section 3.4.

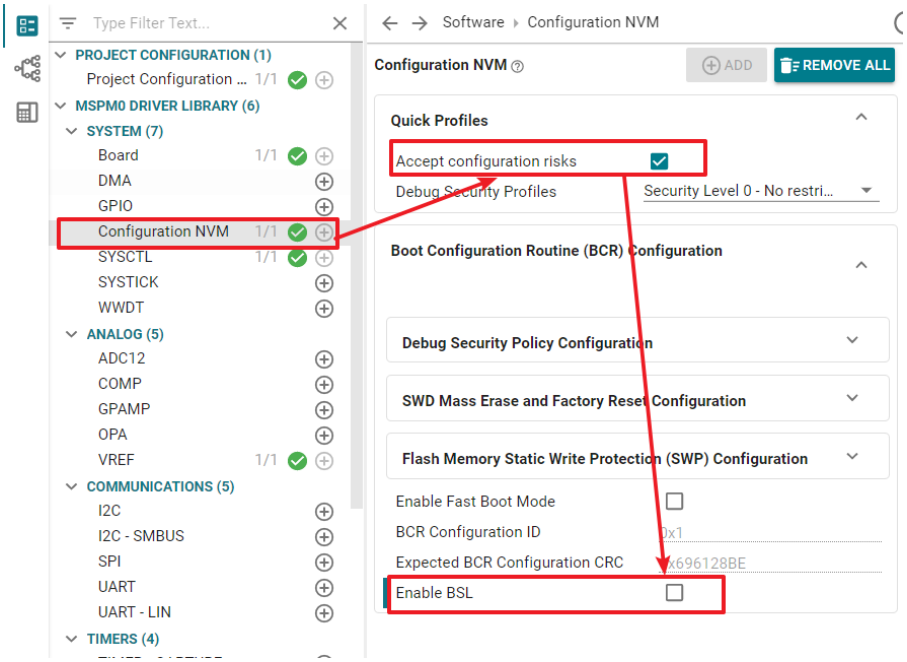


Figure 7-7. Disable BSL

7.4 BSL Related Questions

For questions about how to use bootloader, see MSPM0 Bootloader (BSL) Implementation. This provides an overview of bootloader implementation and step-by-step instructions.

For questions about bootloader protocol and the spec, see the *MSPM0 Bootloader User's Guide*.

7.5 Set SWD Password

The SWD interface can be configured to be disabled, enabled, or enabled with a 128-bit password by writing the BOOTCFG0 and SWDPW registers in NONMAIN. See the device Technical Reference Manual for more information about NONMAIN and SWD password. You can follow the steps to add password on SWD.

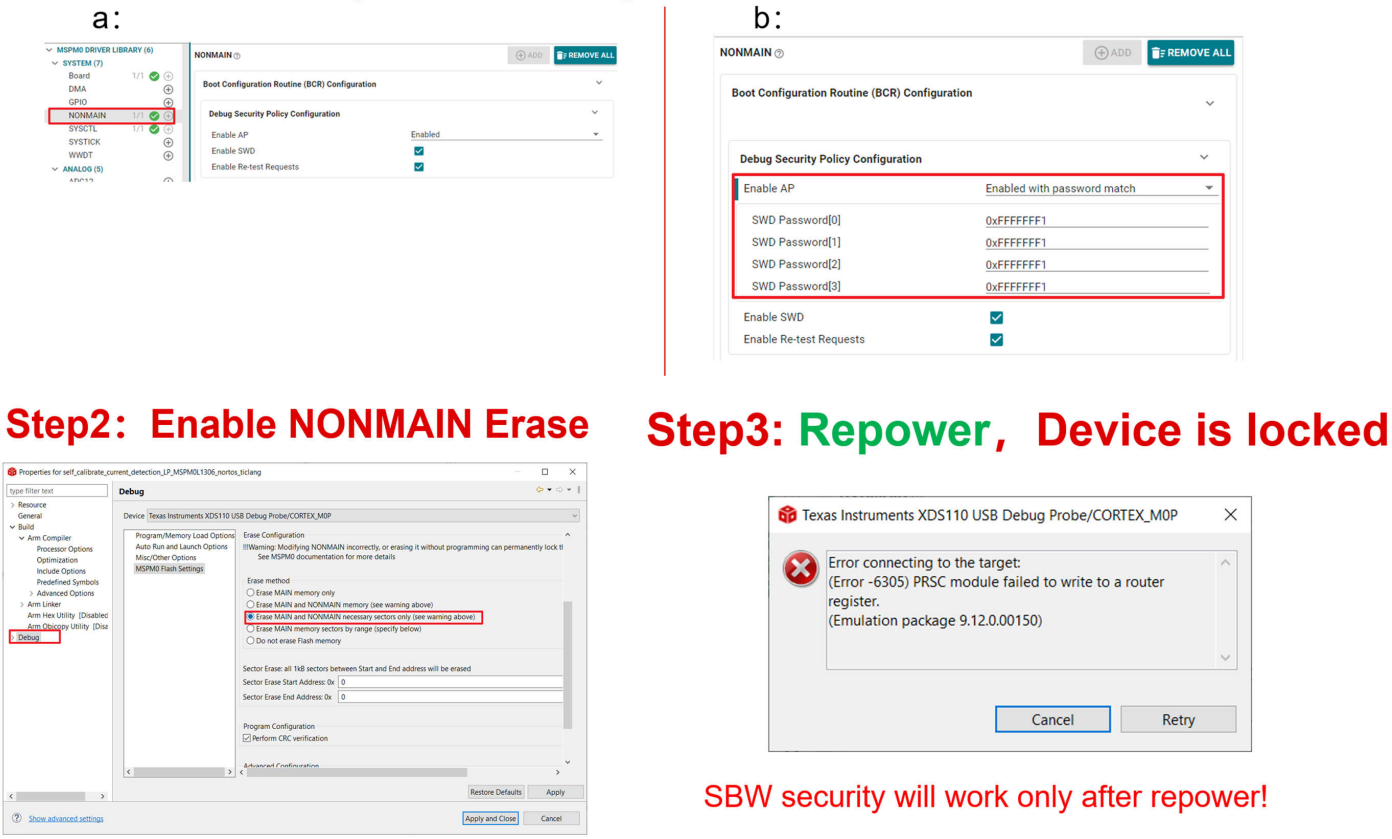


Figure 7-8. Enable SWD Password

Here are the steps to reprogram MSPM0 with the password. This action doesn't erase NONMAIN, so the password remains active unless NONMAIN is modified.

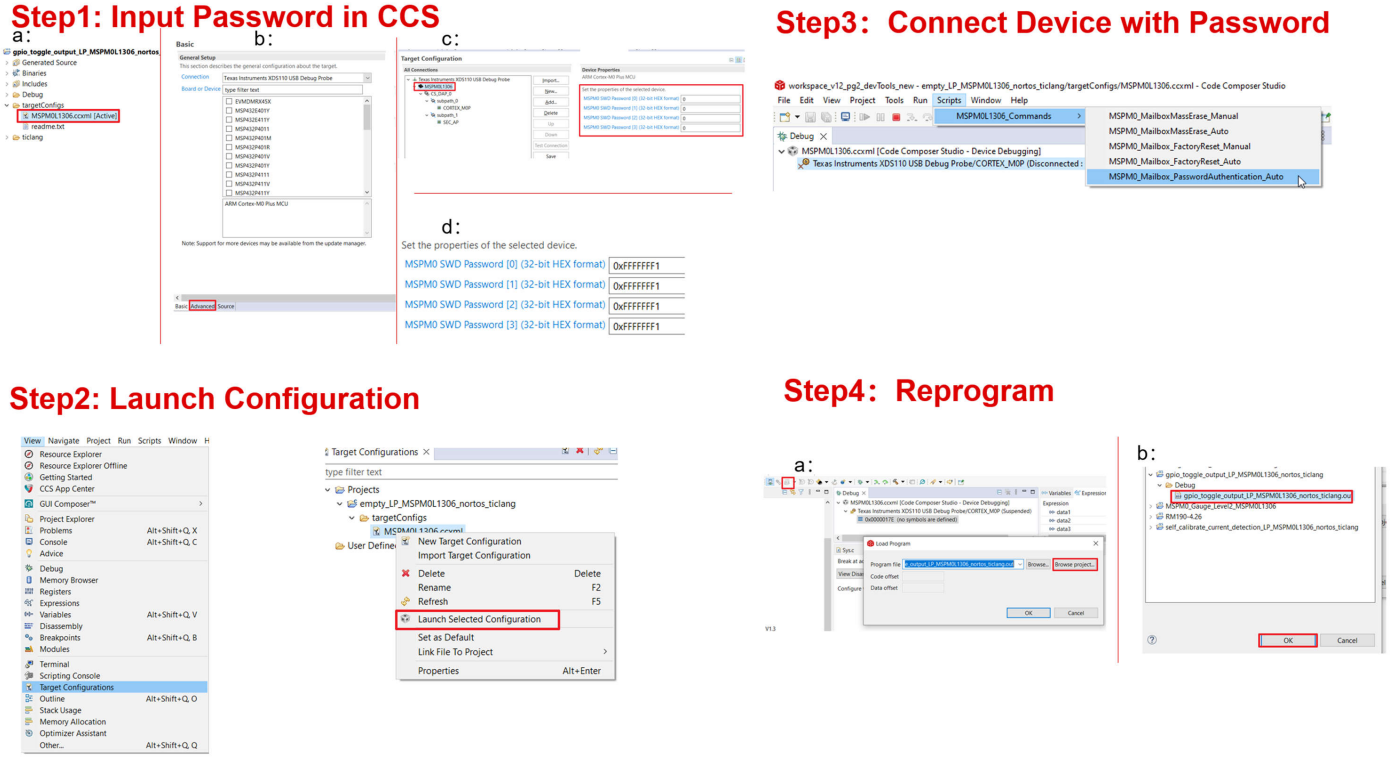


Figure 7-9. Reprogram Device

7.6 CCS Common Questions

In this part, some common questions met in CCS are introduced. Here are some additional documents for your reference when meeting questions with TI's complier, linker or IDE:

- *ARM Assembly Language Tools User's Guide*
- *ARM Optimizing C/C++ Compiler User's Guide*
- TI Arm Clang Compiler Tools User's Guide

7.6.1 Setting Breakpoints at Wanted Places

The default SDK example is with optimization level 2. The code size is smaller. However, this causes a mismatch of the C code and the assembly code and breakpoint cannot be added at the certain C code line. To solve this issue, choose the optimization from level 2 to level 0.

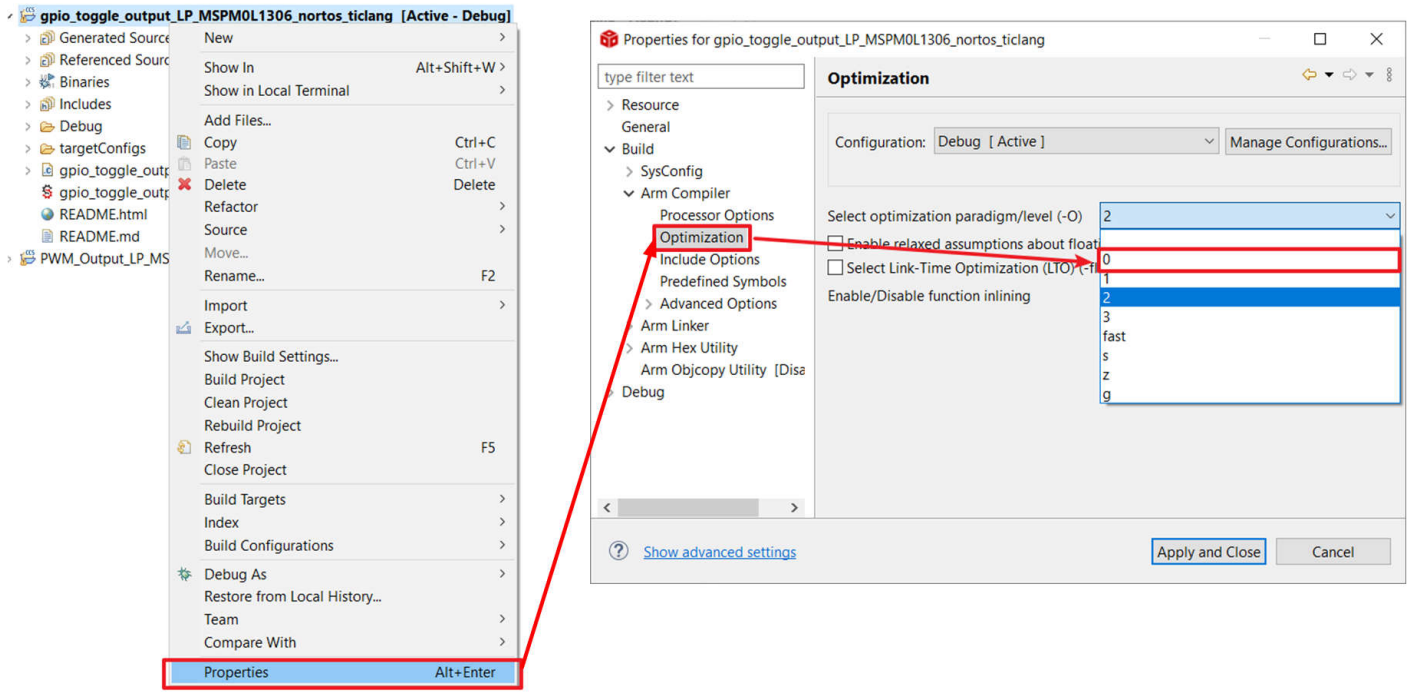


Figure 7-10. Change Optimization Level

7.6.2 Discovered Projects Become Gary

CCS has a workspace concept. If the workspace contains a project with the same name, when a new project is imported, the new project cannot be selected. This problem occurs when the project is deleted without deleting the copy in the workspace. An example is shown in Figure 7-11.

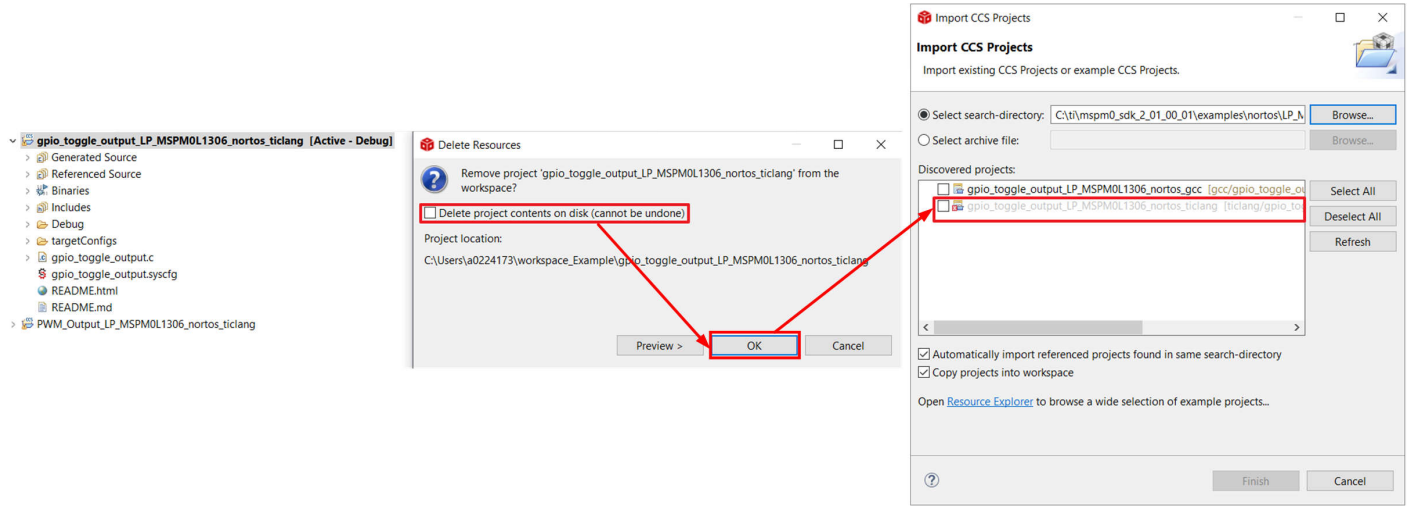


Figure 7-11. Project Cannot be Selected

Follow these steps to solve this problem:

1. Open the workspace address.
2. Copy the workspace address and navigate to the project folder.
3. Remove the duplicated project.
4. Reimport the project.

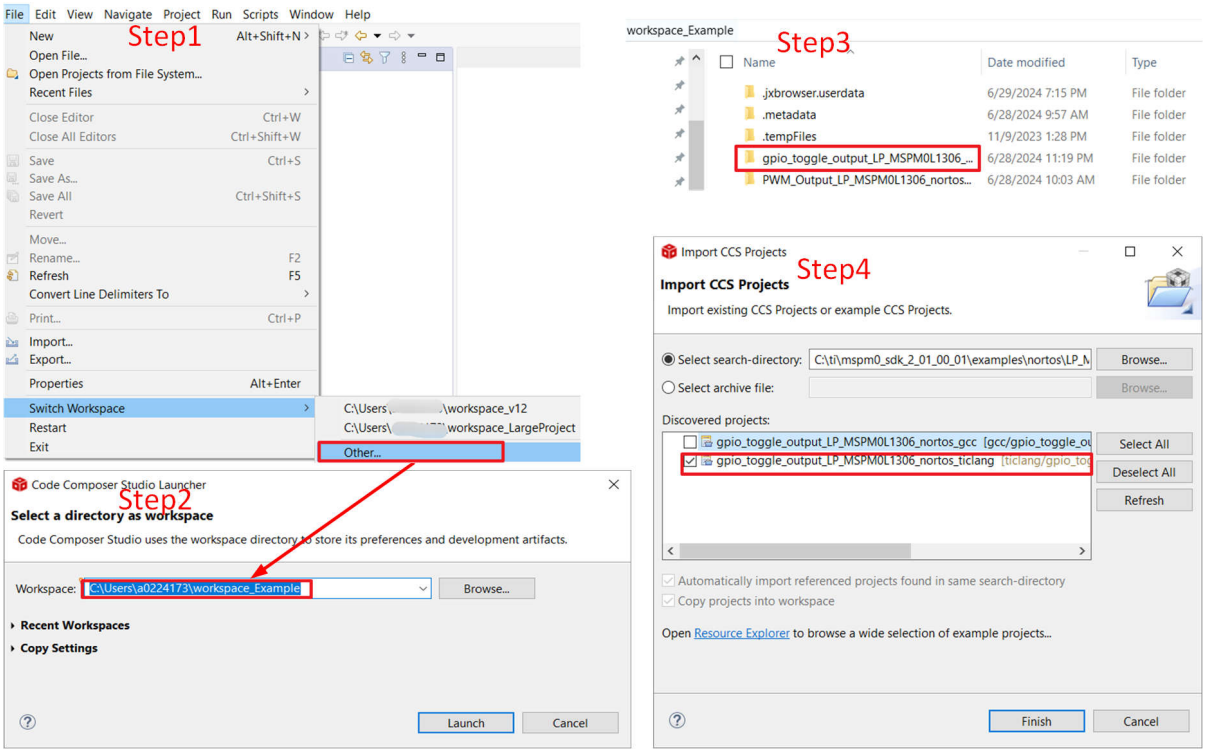


Figure 7-12. Remove Project With the Same Name

7.6.3 CCS Cannot Locate .h File

Some users find that after debug that CCS produces an error that the .h file cannot be located, which is already in the project. The reason is that CCS does not include the .c and .h files in the imported folder and users need to add the link. Follow the steps to add the folder into include address:

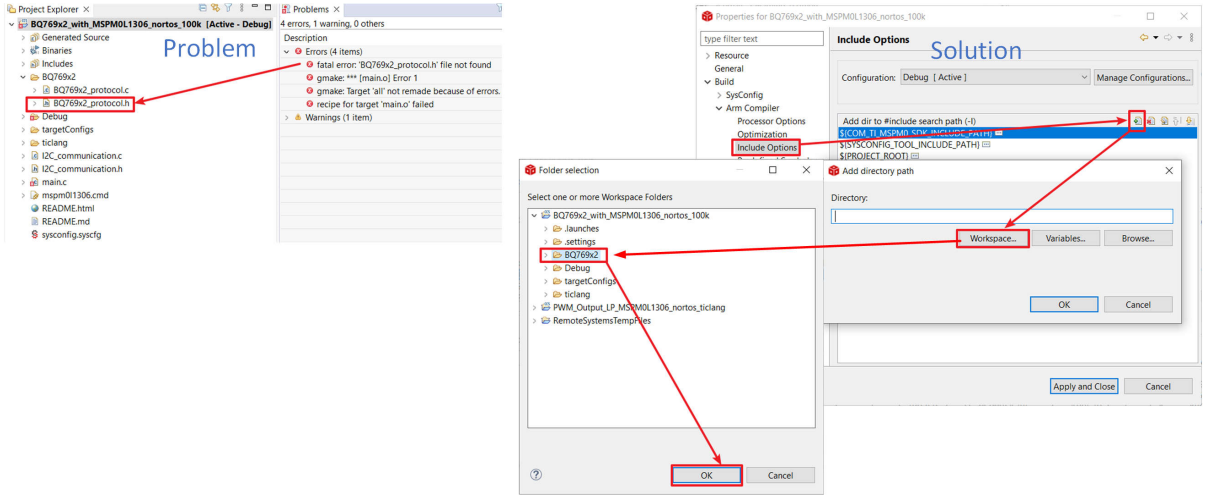


Figure 7-13. Cannot Locate .h File

7.6.4 Install Arm GCC

The MSPM0 SDK includes examples supporting both TI Arm Clang and GCC; however, TI Arm Clang is installed by default in CCS, while GCC is not. Here are the steps to enable CCS support Gcc.

1. GCC can be installed by selecting *Help → Install GCC ARM Compiler Tools*
2. Select the version to install. CCS and the MSPM0-SDK only include and support some versions of the toolchain.
3. If the installation was successful, click on *Window → Preferences*, then *Code Composer Studio → Build → Compilers* to see a list of the compilers installed in CCS.

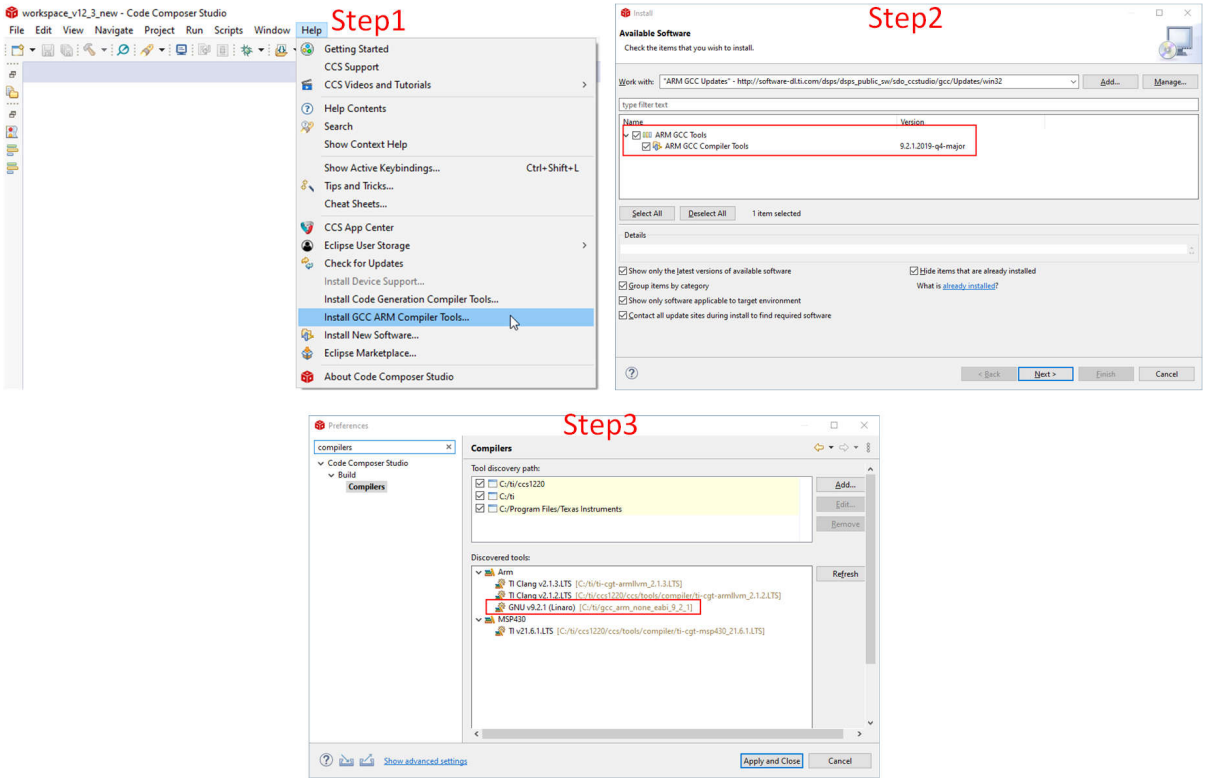


Figure 7-14. Install Arm Gcc

7.6.5 Device is Not Connected After Entering Debug

Sometimes after entering debug mode, there can be an error that the MCU is not connected and the PC does not jump to main function automatically. This can happen when the project name changes and the IDE cannot find the correct debug information.

For this condition, delete the debug folder. Right click the project and enter *Properties*, follow the steps in Figure 7-15 to restore the default debug setting.

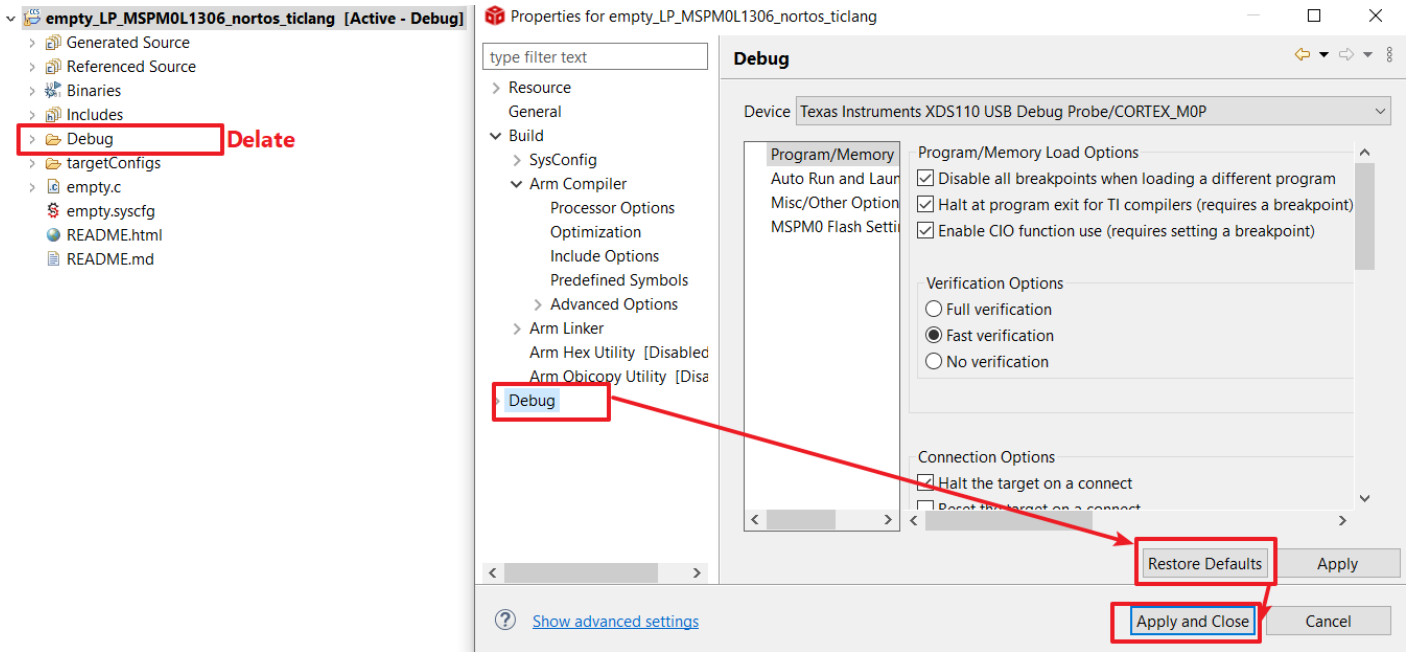


Figure 7-15. Restore Default Debug Setting

7.6.6 Erase the Wanted Memory

Some users want to realize the customized memory erasing. The *Erase Method* entrance is shown in Figure 7-16. The default setting is option 1 and this erases the total memory. When users want to erase the NONMAIN additionally, select option 2. When users want to keep some memory range not to be erased, choose option 3 or option 4 with the declared memory range.

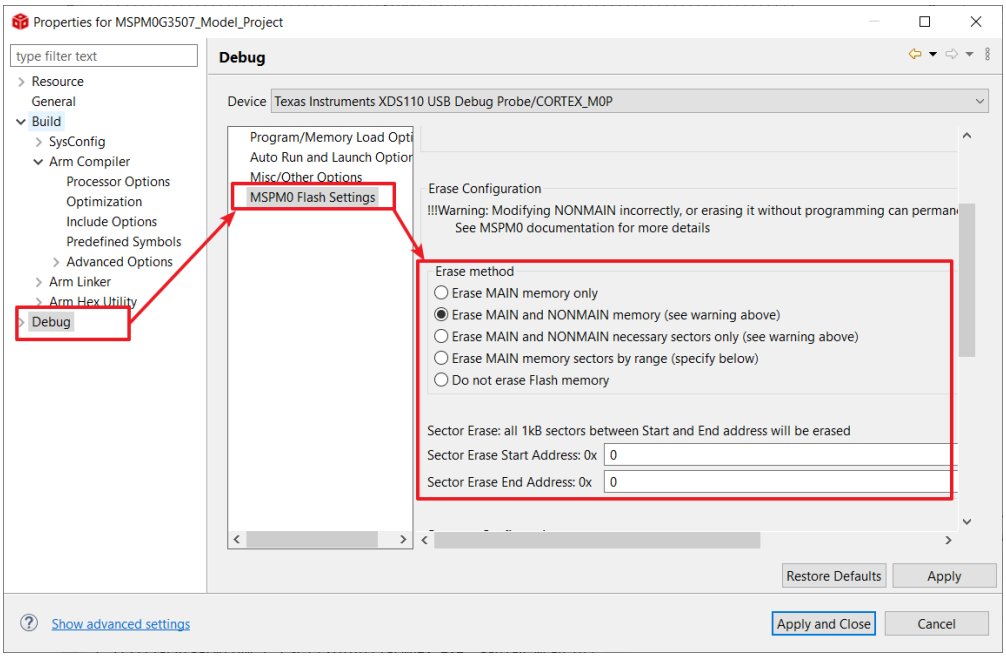


Figure 7-16. Erase the Wanted Memory

7.6.7 Output Data Log From CCS

Some users want to output the data from CCS to desktop for further analysis. The easy way is to save the data in a global array ahead and then flow the instructions bellow to output the data as an excel.

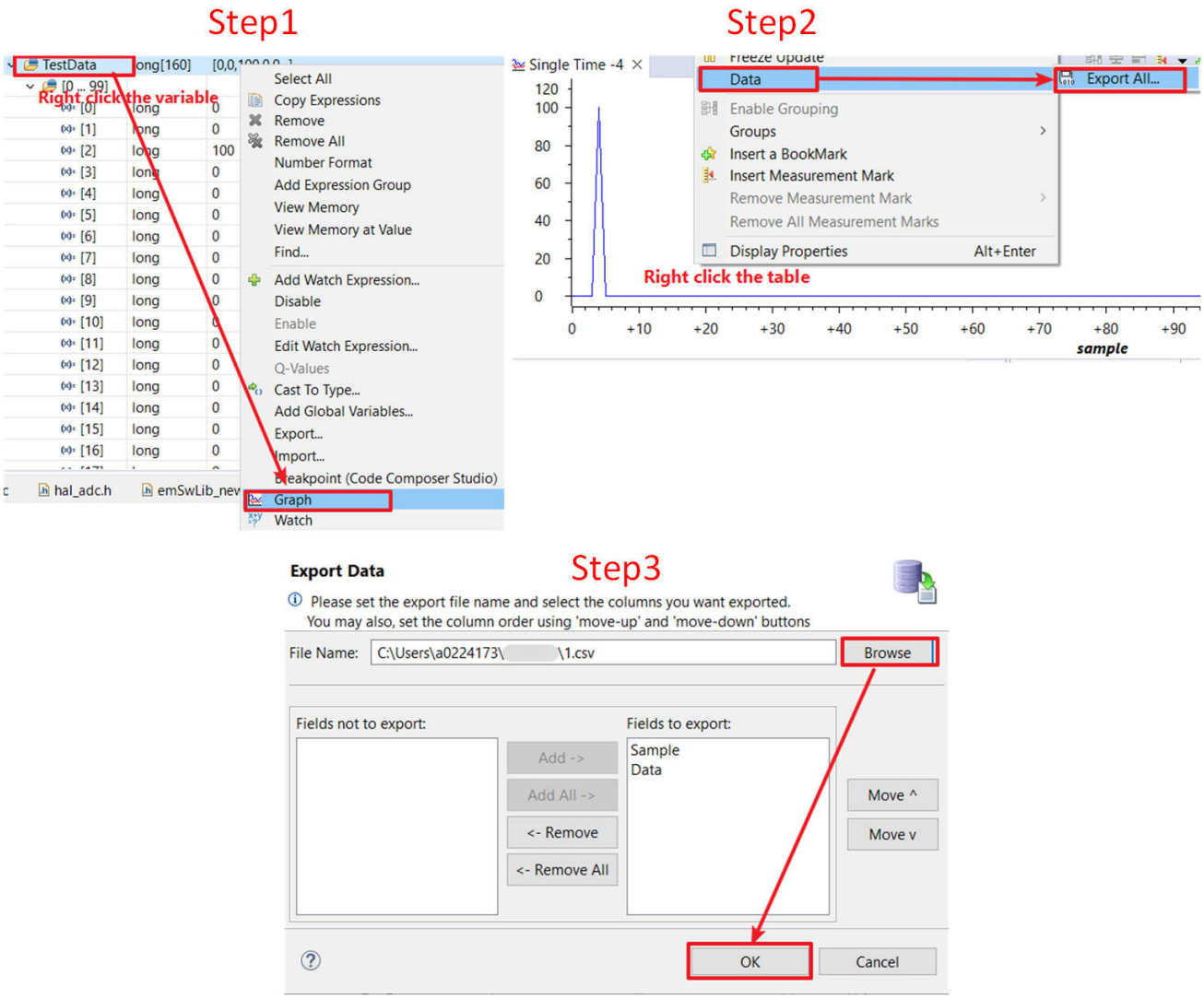


Figure 7-17. Output Data Log From CCS

7.7 Keil Common Questions

7.7.1 Copy Keil Example Out of SDK

If example code is copied out of SDK and compiled directly, then there are errors. The root cause lies on the SDK and SysConfig address setting in the code example. To solve this problem, see Figure 7-18.

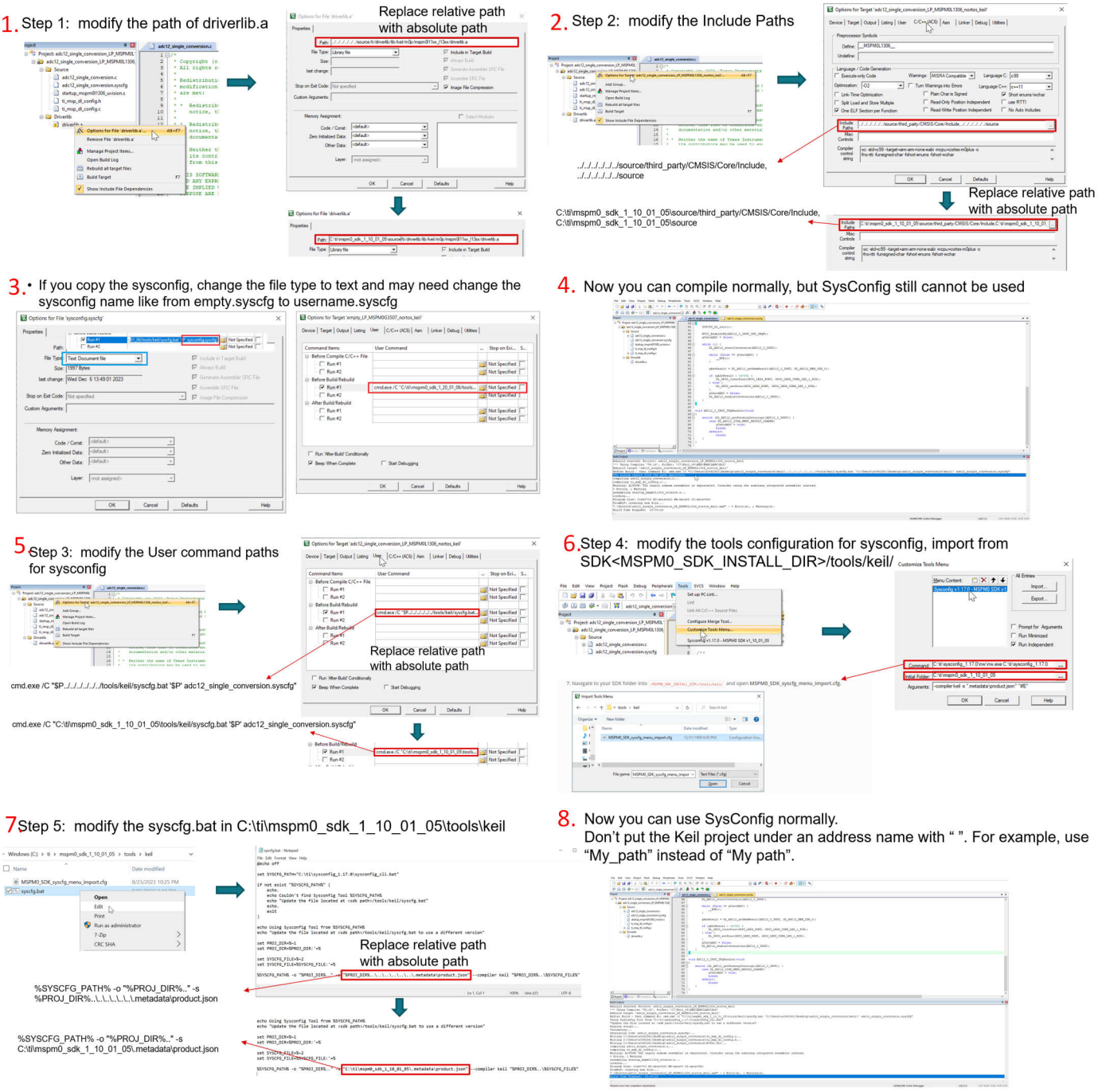


Figure 7-18. Copy Keil Example Out of SDK

8 Additional Information

8.1 Light an LED and CCS Quick Introduction

This section discusses how to light an LED based on CCS from start to finish. A short description of CCS is also provided to users with instructions on how to use the tool.

8.1.A Install CCS and SDK

Here are the important steps and tips for CCS installation.

1. Download CCS (**above 12.2 version**) and stat installation, and keep pressing next.

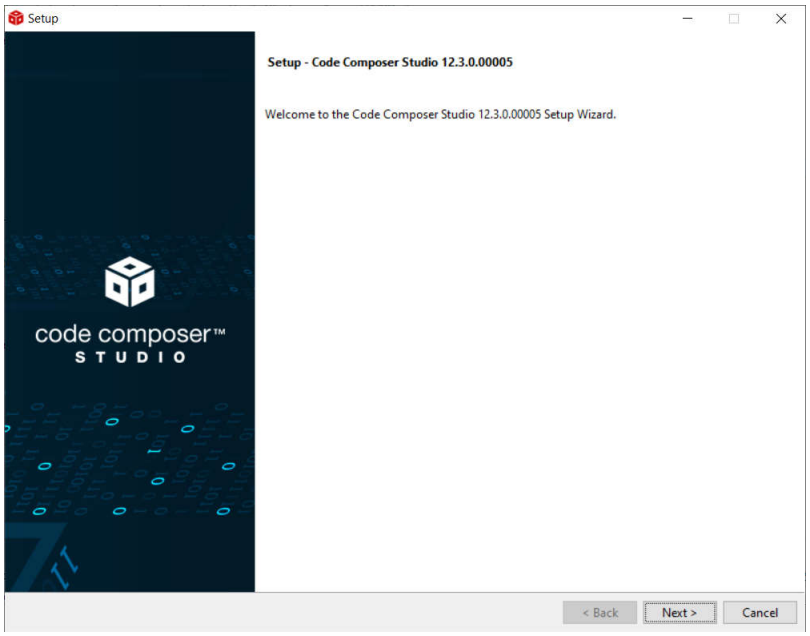


Figure A-1. CCS Installation

2. Select MSPM0 support component.

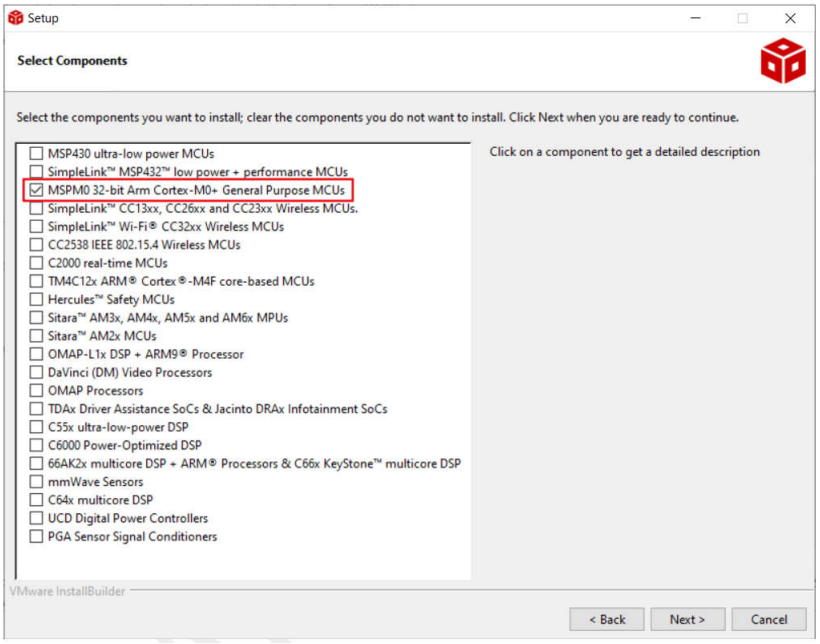


Figure A-2. MSPM0 Support Selection

3. Select J-link if needed.

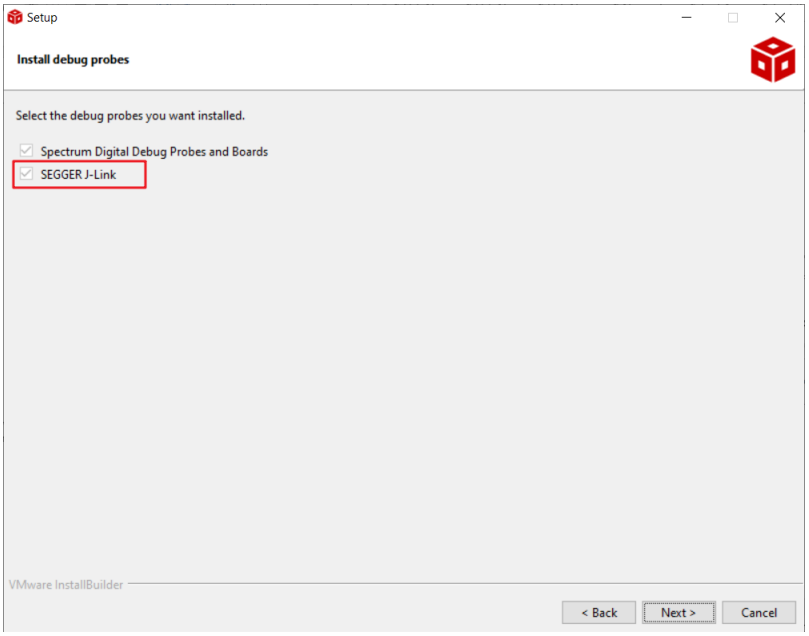


Figure A-3. J-Link Selection

4. Install MSPM0 SDK.

8.1.B Hardware Setup

Get a launchpad and plug in the computer.

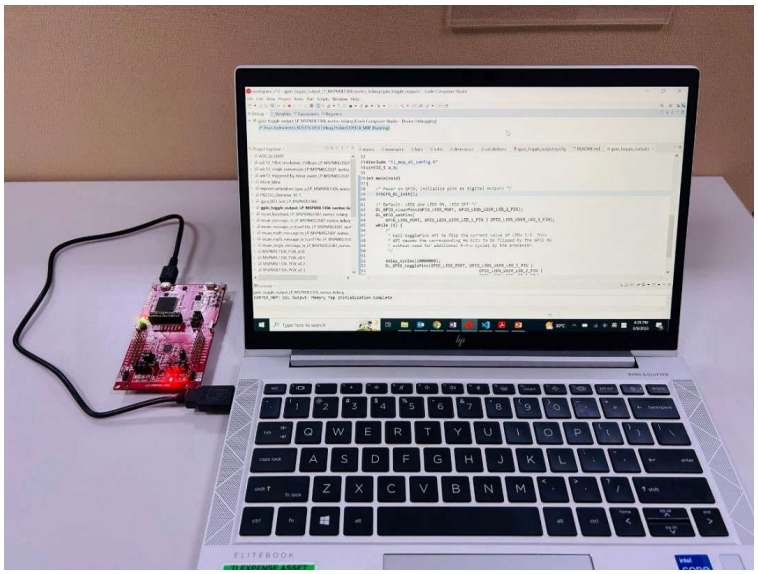


Figure 8-4. Hardware Setup

8.1.C Code Import

1. Open CCS. The workspace means the address where to copy the imported project.

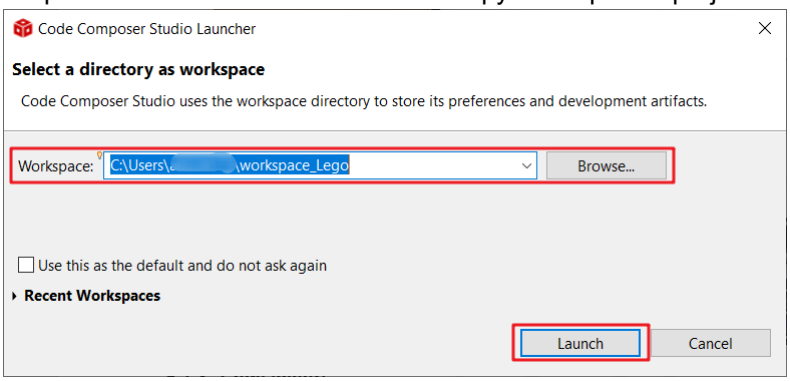


Figure 8-5. Choose CCS Workspace

2. Import the general-purpose input/output (GPIO) toggle project with the TI-Clang compiler.

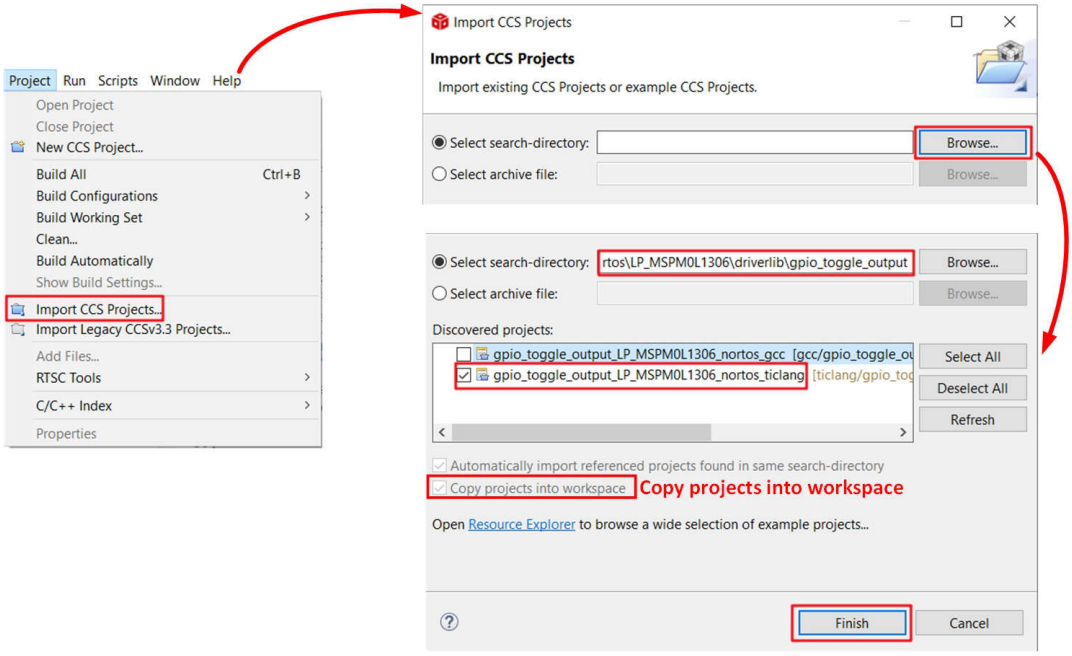


Figure 8-6. Import Project

3. If the project cannot be imported, then delete the same name project under workspace.

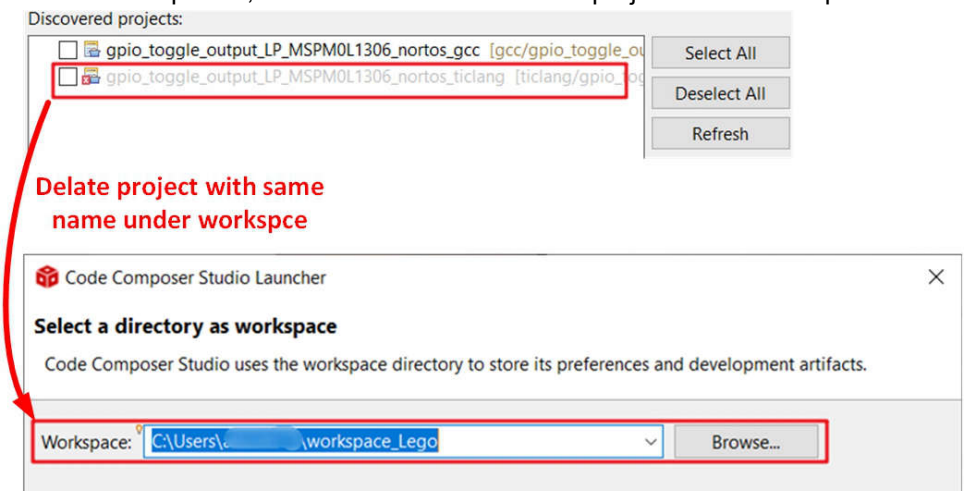


Figure 8-7. Remove Duplicated Project

8.1.D Debug and CCS Quick Introduction

1. Start debug, then you can see GPIO toggle on the LP.

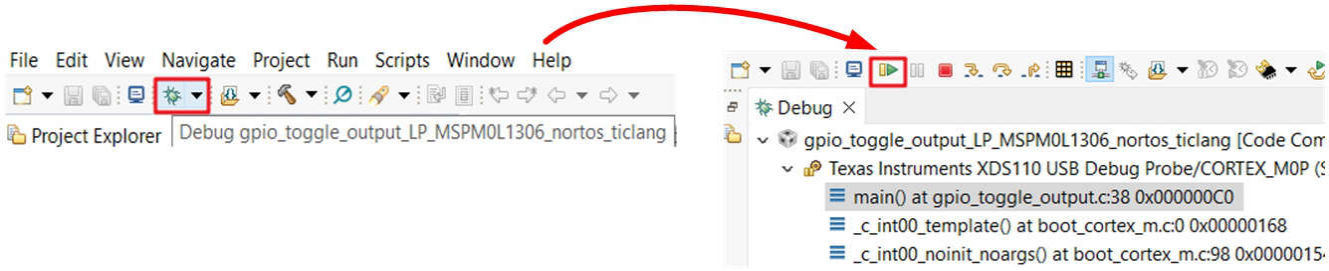


Figure 8-8. Debug Code

2. Here we give a quick introduction to CCS functions.
   a. Project properties common used settings:

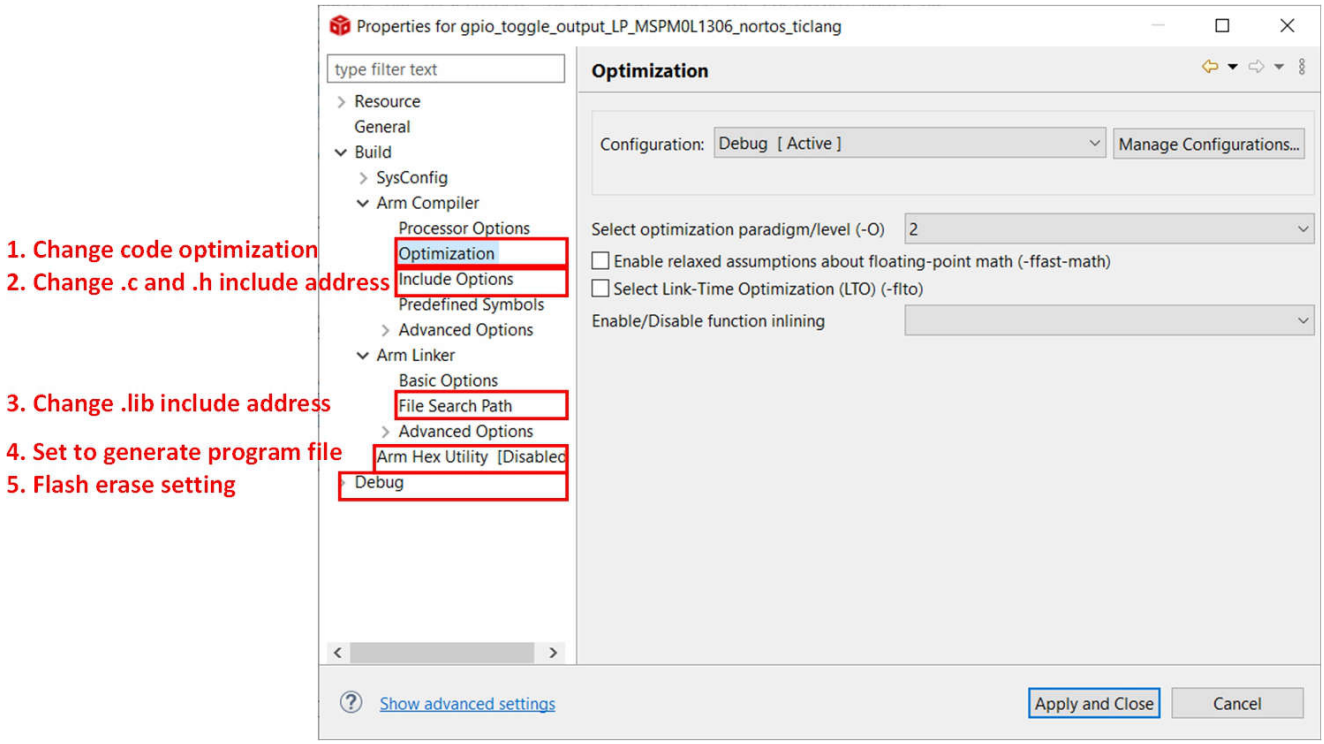


Figure 8-9. Common Used Project Settings

   b. Debug common used functions.

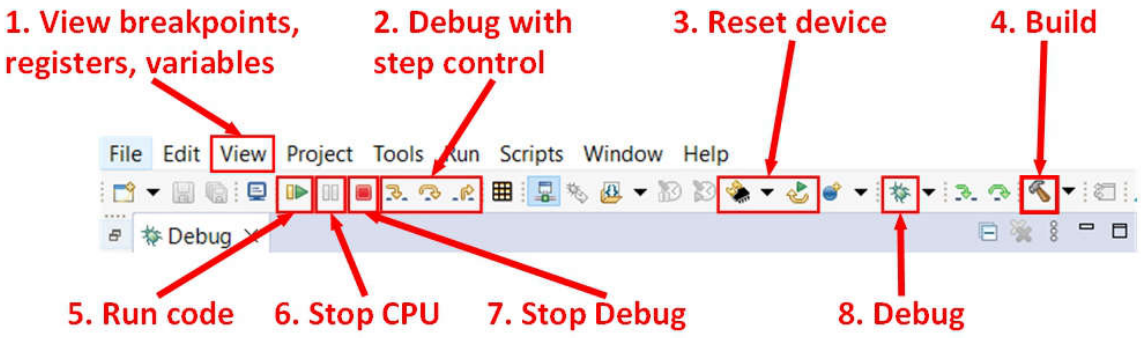


Figure 8-10. Common Used Debug Functions

8.2 Steps to Generate the PCB Library

1. Go to the start page of the Ultra Librarian tool under the MSPM0 device page using the steps shown in Figure 8-11.

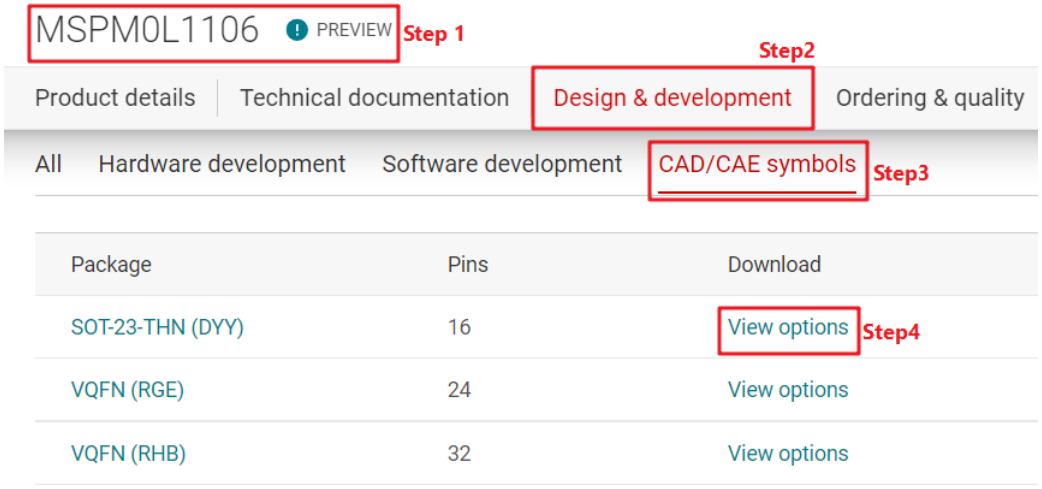


Figure 8-11. Ultra Librarian Tool Start Page

2. Select the desired CAD format and pin ordering to obtain the Altium design library file.

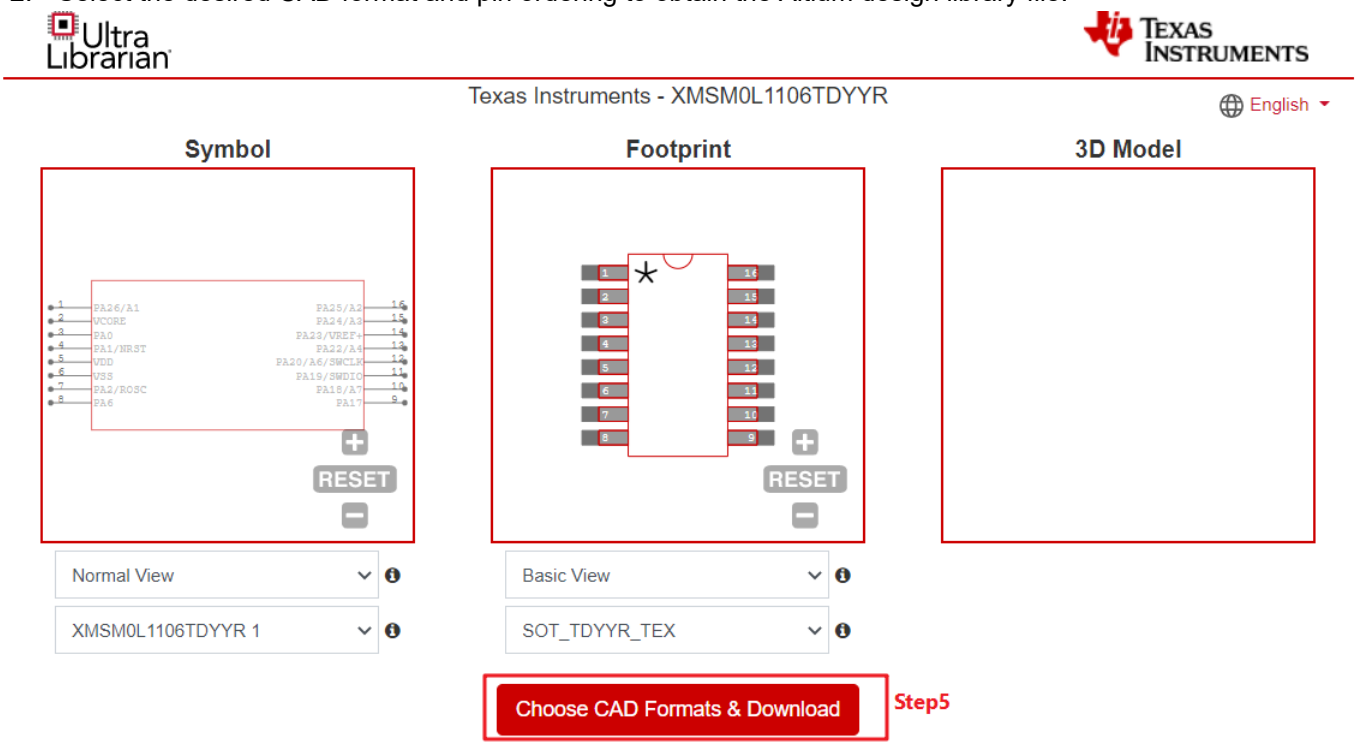


Figure 8-12. Ultra Librarian Tool Device Selection

3. The Altium Designer library file is used as an example.

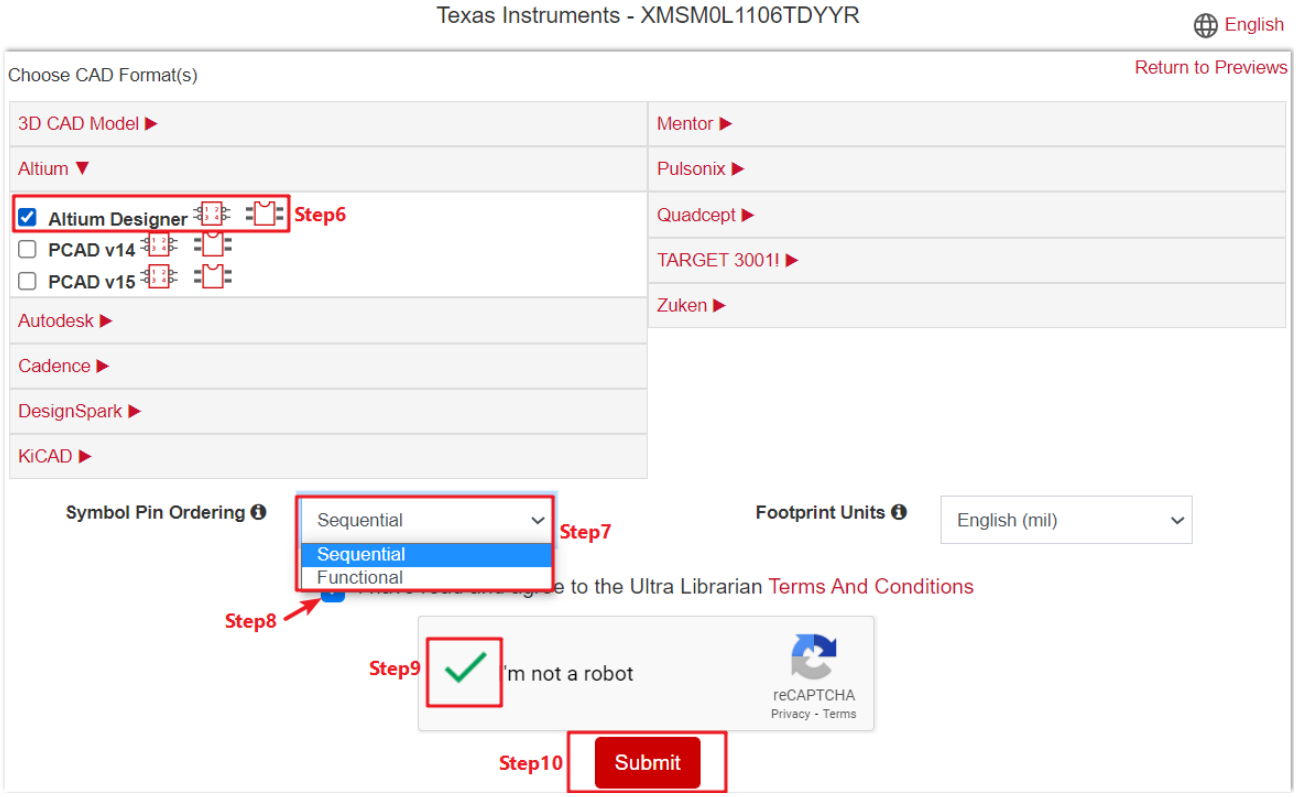


Figure 8-13. Ultra Librarian Tool CAD Download

4. Run the *Altium Designer* script as shown in Figure 8-14.

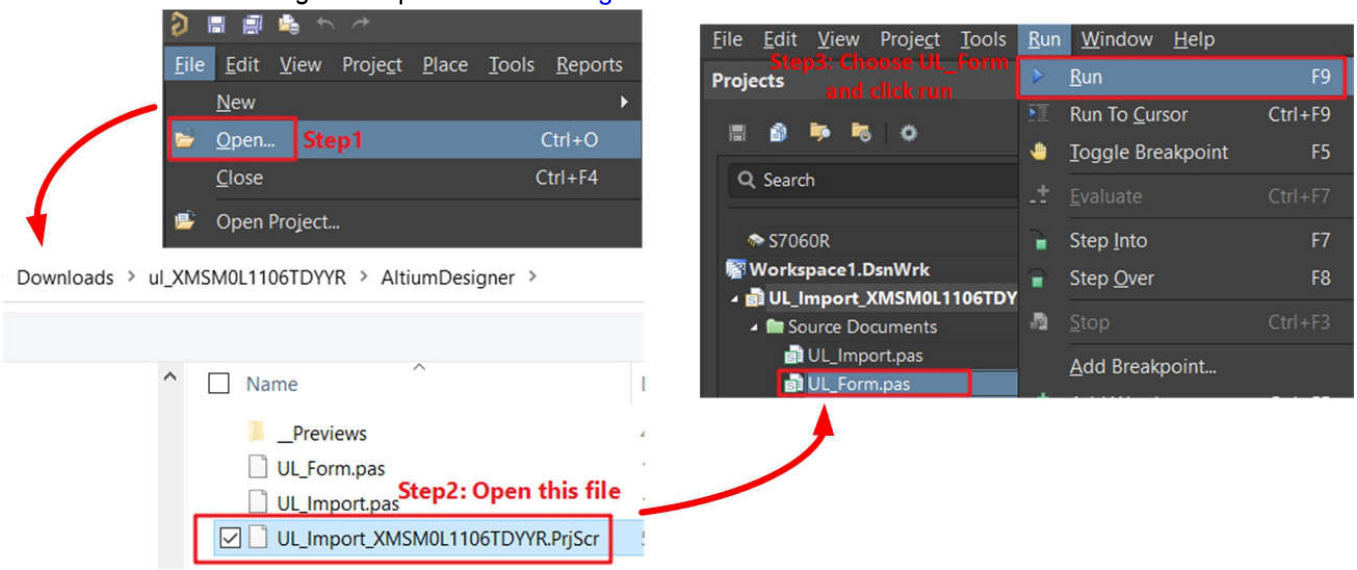


Figure 8-14. Run Altium Designer Script

5. Generate the PCB library and schematic library as shown in Figure 8-15.

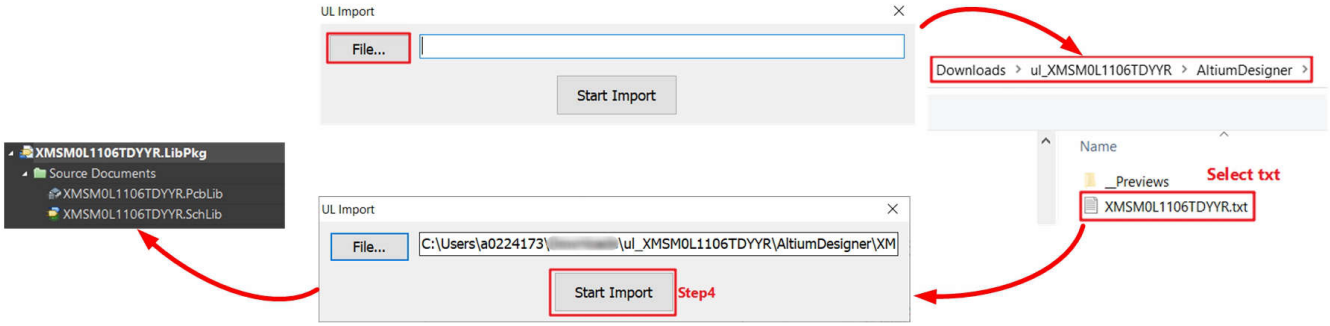


Figure 8-15. Generate Library

6. Select the correct footprint under *PCB Library*.

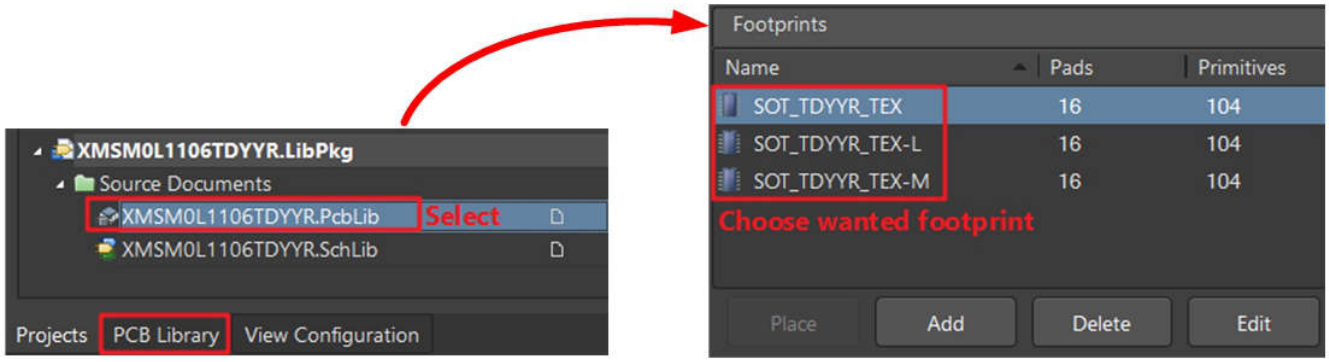


Figure 8-16. Select Footprint

7. Import the PCB library and schematic library.

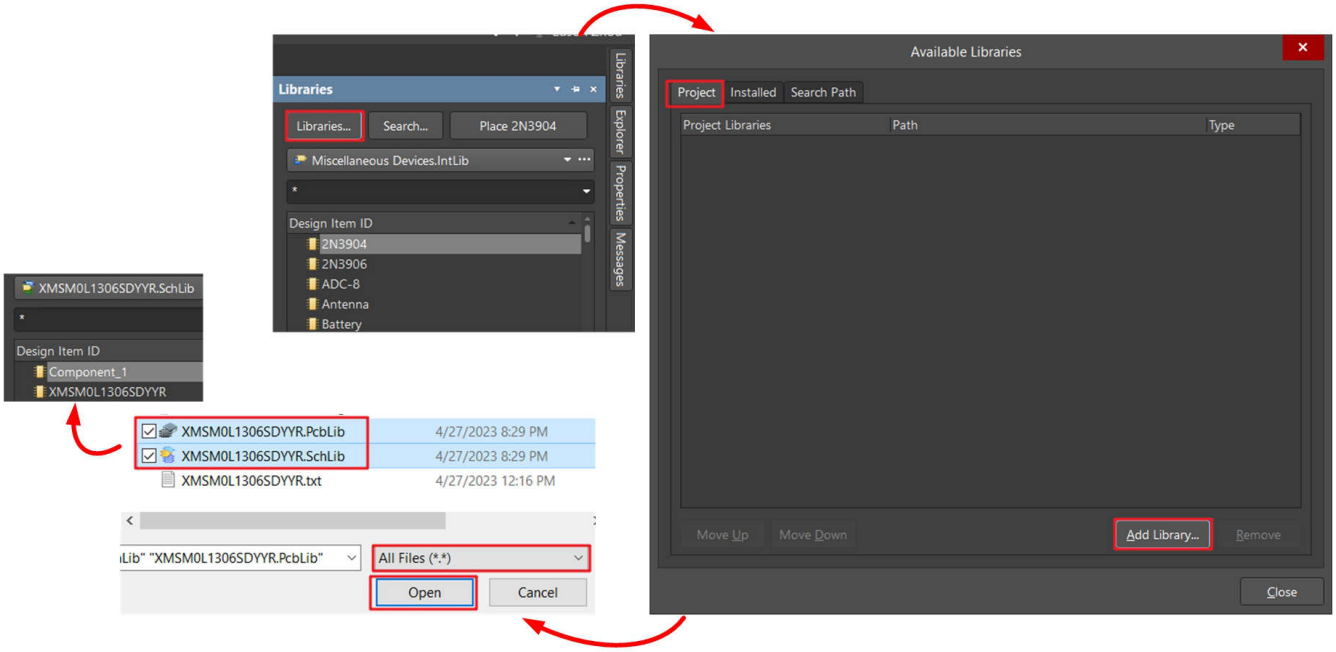


Figure 8-17. Import Library

8.3 MSP-GANG Quick Introduction

This section shows how to use MSP-GANG to the MSPM0 offline program. The section shows how to use MSP-GANG with a GUI to program an MSPM0 device.

1. Finish the pin connection used for software as shown in Figure 8-18.

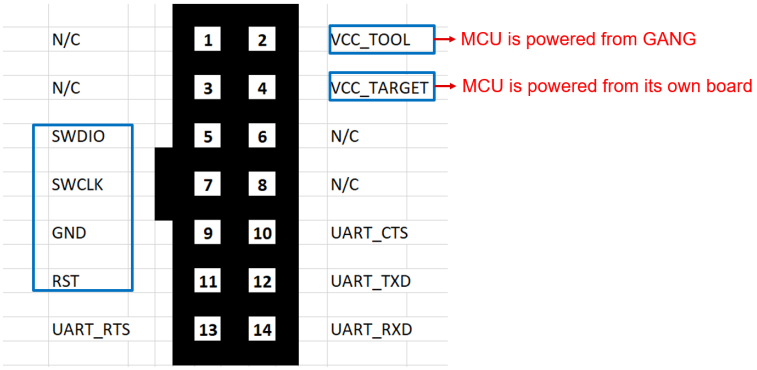


Figure 8-18. MSP-GANG Pin Assignment

2. After the hardware setup is finished, follow the programming steps. For Step 2, see Section 5.1 to generate the code file. For Step 4, the enabled target is related to the hardware port used, which is labeled numerically next to the port.

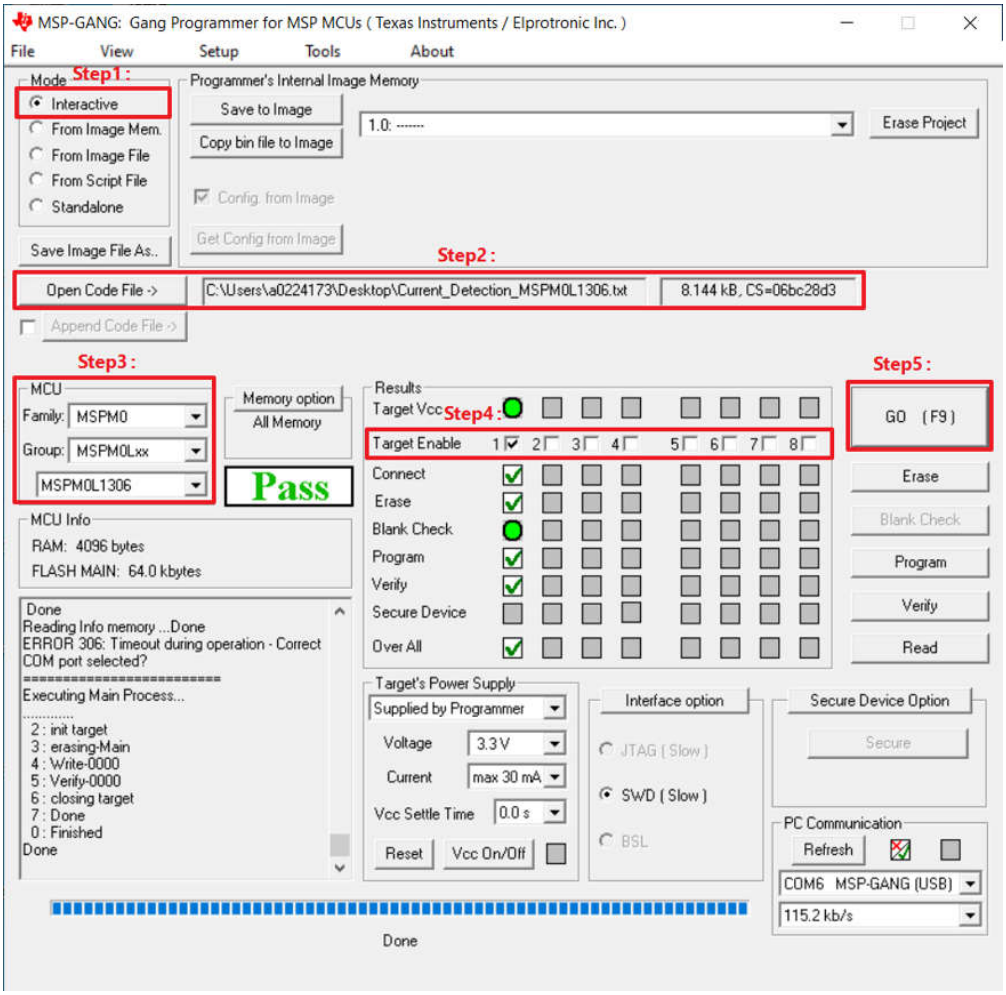


Figure 8-19. Download Code Using MSP-GANG With GUI

3. To change the code file in the non-main (SWD and BSL configure flash area), enable this function first.

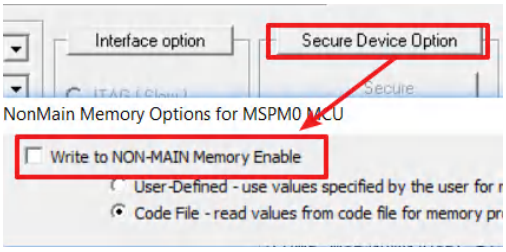


Figure 8-20. Enable Non-Main Programming

4. Save the code file and settings into MSP-GANG. Assign a project name to this image. Then click *Save to Image* as shown in Figure 8-21.

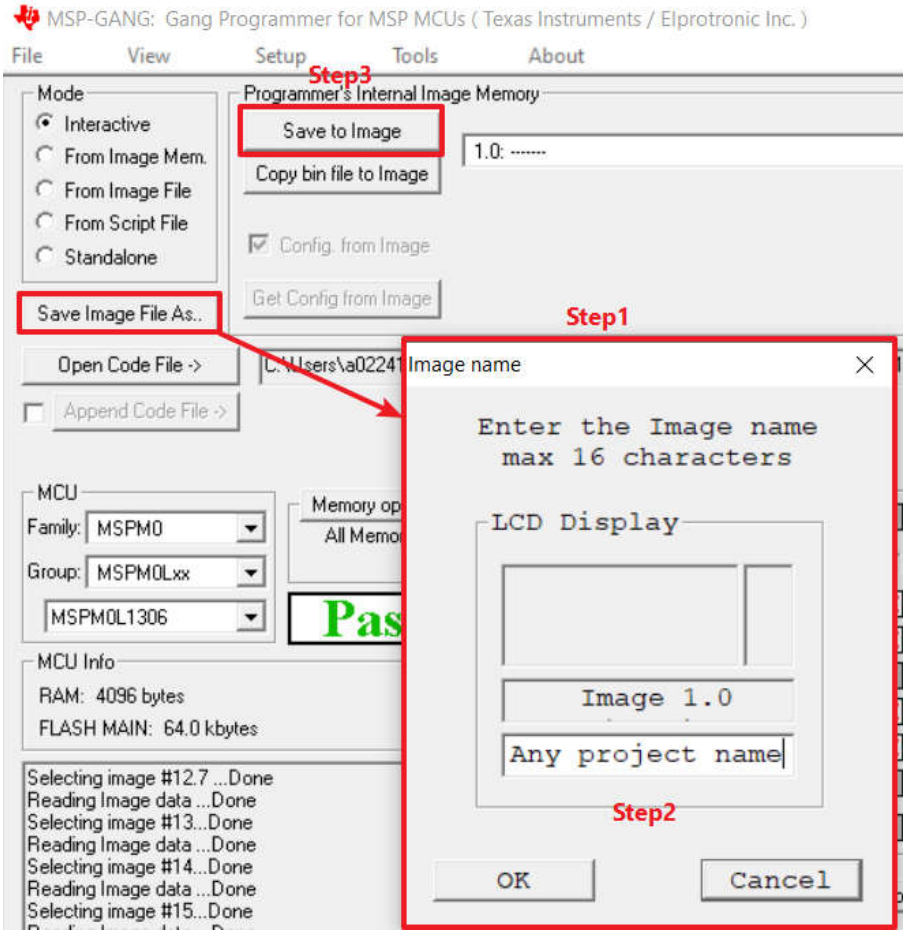


Figure 8-21. Generate and Save Image

5. Change the mode to standalone or directly close the GUI.

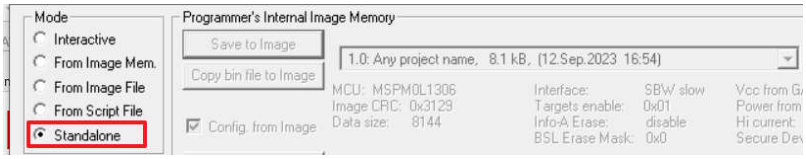


Figure 8-22. Change Mode

6. If only one image is saved in the MSP-GANG, click *Go* to do the programming. If more than one image is saved, switch to the correct image first.

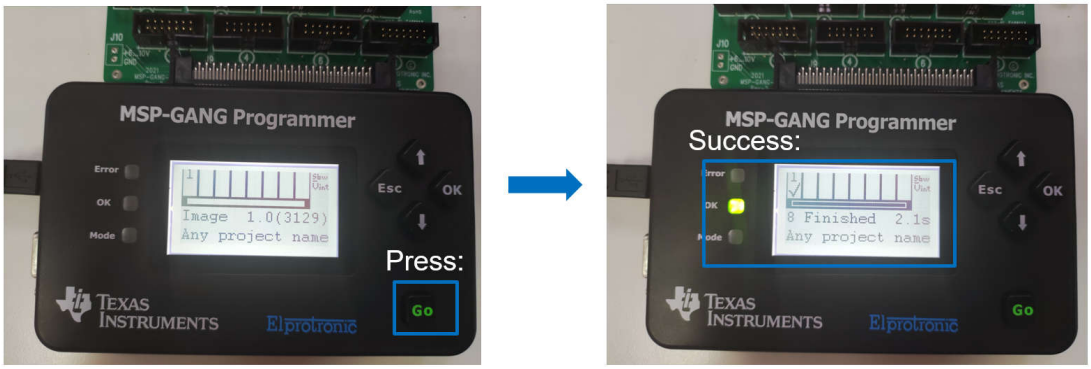


Figure 8-23. Offline Programming

9 Summary

This document is a good start for the MSPM0 development and provides an overview of MSPM0 ecosystem and step-by-step instructions. Users are also provided with clear processes and image explanations. In addition to basic knowledge, the document also lists references and further reading materials for users to refer to further. TI recommends this document for users to quickly handle MSPM0 development work and overcome common obstacles.

Revision History

Changes from Revision C (July 2024) to Revision D (October 2024)	Page
• Add *Quality and Reliability Instructions* section	54
• Add more information to unlock MSPM0 in *Unlock MCU* section	56
• Add a new common question for CCS in *Output Data Log From CCS* section	66

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265