User's Guide

Migration Guide From STM8 to MSPM0

ABSTRACT

This application note assists with migrating from the STMicroelectronics STM8L and STM8S platform to the Texas Instrument MSPM0L and MSPM0C MCU ecosystem. This document introduces the MSPM0 development and tool ecosystem, core architecture, peripheral considerations, and software development kit. The intent is to highlight the differences between the two families and to leverage existing knowledge of the STM8 development environment to quickly ramp with the MSPM0 series of MCUs.

Table of Contents

List of Figures

List of Tables

Trademarks

LaunchPad™, Code Composer Studio™, EnergyTrace™, and BoosterPack™ are trademarks of Texas Instruments.
Arm® and Cortex® are registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere.
µVision® is a registered trademark of Arm Limited.
All trademarks are the property of their respective owners.

1 MSPM0 Portfolio Overview

1.1 Introduction

MSPM0 microcontrollers (MCUs) products are part of the MSP highly-integrated ultra-low power 32-bit MCU family based on the enhanced Arm® Cortex®-M0+ 32-bit core platform operating. These cost-optimized MCUs offer high-performance analog peripheral integration, support extended temperature ranges, and offer small footprint packages. The TI MSPM0 family of low-power MCUs consists of devices with varying degrees of analog and digital integration allowing engineers to find the MCU that meets the requirements of a project. The MSPM0 MCU family combines the Arm Cortex-M0+ platform with an ultra-low-power system architecture, allowing system designers to increase performance while reducing energy consumption.

The MSPM0 MCUs offer a competitive alternative to the STM8 MCUs. This application note assists with migration from STM8 MCUs to MSPM0 MCUs by comparing device features and ecosystems.

1.2 Portfolio Comparison of STM8 MCUs to MSPM0 MCUs

Table 1-1. Comparison of the TI MSPM0Lx, MSPM0Cx and STM8Lx, STM8Sx

		ST Micro STM8 L Series	ST Micro STM8 S Series	TI MSPM0 Lx Series	TI MSPM0 Cx Series	TI MSPM0 Hx Series
Core		STM8 CPU core		Arm Cortex-M0+		
Max frequency		16MHz	24MHz	32MHz	24MHz, 32MHz(1)	32MHz
Supply voltage		1.6-3.6 V	2.95-5.5 V	1.62-3.6 V	1.62-3.6 V	4.5-5.5 V(2)
Max. temperature		-40-125°C	-40 to 125°C	-40 to 125°C	-40 to 125°C	-40 to 125°C
Memory		64KB to 2KB	128KB to 4KB	64KB to 8KB	16KB to 8KB	64KB to 32KB
RAM		Up to 4KB	Up to 6KB	Up to 4KB	Up to 8KB	Up to 8KB
GPIO (max)		41	38	28	45	45
RTC		Yes	No	Yes	Yes (C1103 and C1104 does not support)	Yes
Analog	ADC	Up to 12-bit x 28-ch	Up to 10-bit x 16-ch	1x 1.68-Msps 12-bit ADC(16-ch)	1x 1.5-Msps 12-bit ADC(Up to 27-ch)	1x 1.5-Msps 12-bit ADC(Up to 27-ch)
	DAC	Up to 12-bit x 2-ch	none	8-bit	8-bit (C1103 and C1104 does not support)	none
	comparator	3µs propagation delay	none	1x high-speed	1x high-speed (C1103 and C1104 does not support)	none
Communication	UART	1 Mbit/s, up to 3 UARTs	1 Mbit/s, up to 2 UARTs	2x 4 Mbit/s	2x 4 Mbit/s	3x 4 Mbit/s
	I2C	1, 100 and 400 Kbit/s	1, 100 and 400 Kbit/s	2, up to 1 Mbit/s	1, up to 1 Mbit/s	2, up to 1 Mbit/s
	SPI	10 Mbit/s	10 Mbit/s	16 Mbit/s	12 Mbit/s	16 Mbit/s
	CAN	none	1Mbit/s, up to 3 mailboxes	None	None	None
	LIN	UART support		UART support		
Other key peripherals / features		LCD driver Beeper Touch sensing (STM8L CT library) DMA IR interface	Beeper Touch sensing (STM8S RC library)	2x op amps LCD (L2228) DMA	Smallest QFN package (2x2) and BGA package(0.861x1.6), 0.5/0.65 mm pitch packages, Pin-compatible with industry DMA	5V power, DMA, Window watchdog timer
Timer number		3, 4, 5	3/4	7	5 (C1103 and C1104 support 3 timers)	4
Pin count		Up to 68	Up to 68	16-80 pins	8-48 pins	20-48 pins

Table 1-1. Comparison of the TI MSPM0Lx, MSPM0Cx and STM8Lx, STM8Sx (continued)

	ST Micro STM8 L Series	ST Micro STM8 S Series	TI MSPM0 Lx Series	TI MSPM0 Cx Series	TI MSPM0 Hx Series
Low power	Active: 1.6mA at 16MHz Halt: 0.3µA	Active: 1.8mA at 16MHz Halt: 5µA	Active: 71µA/MHz Standby: 1µA	Active: 100µA/MHz Standby: 5µA	N/A

(1) MSPM0C1103 and MSPM0C1104: 24MHz, and MSPM0C1105 and MSPM0C1106: 32MHz
(2) The first device in H series is 4.5 - 5.5V, and supports wide power supply in the future.

2 Ecosystem And Migration

MSPM0 MCUs are supported by an extensive hardware and software ecosystem with reference designs and code examples to get designs started quickly. MSPM0 MCUs are also supported by online resources, trainings with MSP Academy, and online support through the TI E2E™ support forums.

2.1 Ecosystem Comparison

Table 2-1. Ecosystem Comparison

Feature	STM8 Devices	MSPM0 Devices
Code source	Standard peripheral library (collection of embedded software drivers and examples) firmware packages	MSPM0-SDK (DriverLib, Middleware, RTOS, Code example)
IDE	STVD, IDEA, IAR-EWSTM8, iSYS-WinIDEA, Arduino IDE	CCS, IAR, Keil
Software Configuration	STM8CubeMX	SysConfig
Flash programming tool	FLASHER-STM8 STVP(STM8)	UniFlash
Programmer	FlashRunner	MSP-GANG, C-GANG
Debugger	ST-Link v2, ic5000	XDS110, J-LINK
Hardware	STM8-SO8-DISCO, STM8S-DISCOVERY, STM8SVLDISCOVERY, STM8L-DISCOVERY, NUCLEO-8S208RB, NUCLEO-8L152R8	LP-MSPM0G3507 LaunchPad, LP-MSPM0L1306 LaunchPad, LP-MSPM0C1104 LaunchPad, LP-MSPM0C1106 LaunchPad, LP-MSPM0H3216 LaunchPad

Figure 2-1 shows the overview of the MSPM0 ecosystem.

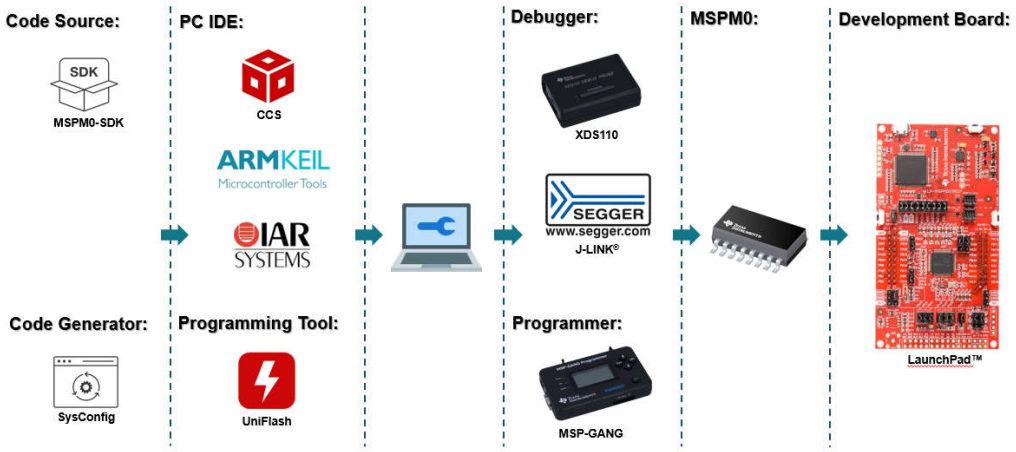


Figure 2-1. MSPM0 Ecosystem Overview

2.1.1 MSPM0 Software Development Kit (MSPM0 SDK)

The MSPM0 SDK is packaged with a wide selection of code examples to enable engineers to develop applications on Texas Instruments' MSPM0+ microcontroller devices. Examples are provided to demonstrate the use of each functional area on every supported device and are a starting point for your own projects. Figure 2-2 illustrates a structure of MSPM0 SDK.

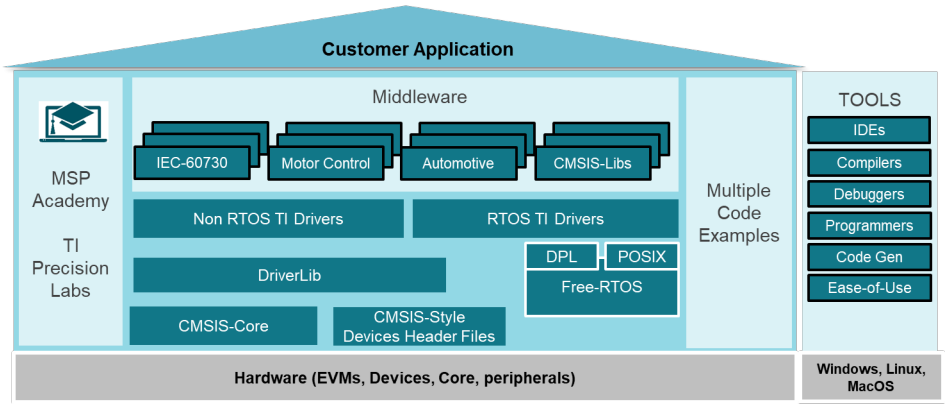


Figure 2-2. MSPM0 SDK Structure

The MSPM0 SDK can be downloaded from MSPM0-SDK Support software | TI.com. There are four folders included in MSPM0 SDK:

Example: The examples folder is divided into RTOS and non-RTOS subfolders (currently only non-RTOS is supported). These folders contain examples for each LaunchPad™ and are organized based on function with lower-level Driverlib examples, higher-level *TI Drivers* examples, and examples for *middleware* such as GUI Composer, LIN, IQMath, and others.

Docs: Includes all relevant documentation including user's guides and API guides.

Source: Source code and libraries for all drivers and middleware.

Tools: Set of tools to aid in the development and/or testing of MSPM0 applications.

As for STM8, ST povides a standard peripheral library for free, RTOS, bootloader and so on. And STM8 also has a large of Miidleware and application fields, like touch-sensing, motor control, Lin library, which MSPM0 almost covers.

Most MSPM0 examples support SysConfig to simplify the device configuration and accelerate software development.

Other reference document are shown below:

- MSPM0 SDK User Guide
- MSPM0 Tools Guide
- Driverlib API Guide

2.1.2 IDE Supported By MSPM0

An integrated development environment (IDE) is a software application that helps programmers develop software code efficiently, which normally includes editor, compiler, debugger and so on.

The typical IDE of STM8 is STVD provided by STMicroelectronics, which can download sample code and has an easy-to-use Eclipse code editor. STVD only has an assembly compiler, not a C compiler, so users need to install an additional C compiler, the Cosmic Tool. Cosmic has launched a compiler for STM8. Codes up to 32KB can be used for free. As a result, STM8 users tend to develop a project through IAR, with which MSPM0 also works.

As for TI, Code Composer Studio™ IDE (CCS) is highly recommended, which supports TI's microcontroller (MCU) and embedded processor portfolios. Specifically, CCS comprises a series of tools used to develop and debug embedded applications including an optimizing C/C++ compiler, source code editor, project build environment, debugger, profiler and many other features. CCS is also completely free to use and is available as both.

The differences and similarities between the two IDEs are shown in Table 2-2.

Table 2-2. Comparison Between CCS and STVD

IDEs	CCS	STVD
License	Free	Free
Compiler	TI Arm Clang, GCC	Cosmic, raisonance
Current consumption integrated in IDE	EnergyTrace	Not supported (supported by STM8CubeMX)
Peripherals' API function assistance	Not supported	Not supported
Display language	English	English
Convert file	Hex file, binary file, Motorola S-record file, Ti_txt file	Hex file, binary file, Motorola S-record file
Generate code GUI	SysConfig	STM8CubeMX

CCS integrates MSPM0 device configuration and auto-code generation from SysConfig as well as MSPM0 code examples and academy trainings in the integrated TI Resource explorer. What's more, CCS offers an all-in-one development tool experience.

In addition to CCS, MSPM0 devices are also supported in industry-standard IDEs listed in Table 2-3.

- CCS: https://www.ti.com/tool/CCSTUDIO
- IAR: https://www.iar.com/
- Keil: https://www.keil.com/

Table 2-3. 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	0.88G (ccs20.1.1)	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 use of CCS and some of features can be seen in Section 2.2.2.2. Other reference materials are shown as follows:

- CCS quick start guide
- CCS
- CCS training videos
- CCS user's guide
- IAR quick start guide
- IAR training videos
- IAR user's guide
- Keil quick start guide
- Keil training videos

- Keil getting started

2.1.3 SysConfig

Similar to STM8CubeMX, SysConfig is an intuitive and comprehensive collection of graphical utilities for configuring pins, peripherals, radios, subsystems, and other components, which can be seen in Figure 2-3. SysConfig helps manage, expose, and resolve conflicts visually so that you have more time to create differentiated applications. The tool output includes C header and code files that can be used with MSPM0 SDK examples or used to configure custom software. SysConfig is integrated into CCS but can also be used with Keil and IAR.

SysConfig can be downloaded at the following URL: SYSCONFIG IDE, configuration, compiler or debugger | TI.com.

SysConfig can run without an IDE. The standalone version can be used for code generation and to evaluate the capabilities of the device, but is not capable of running an example.

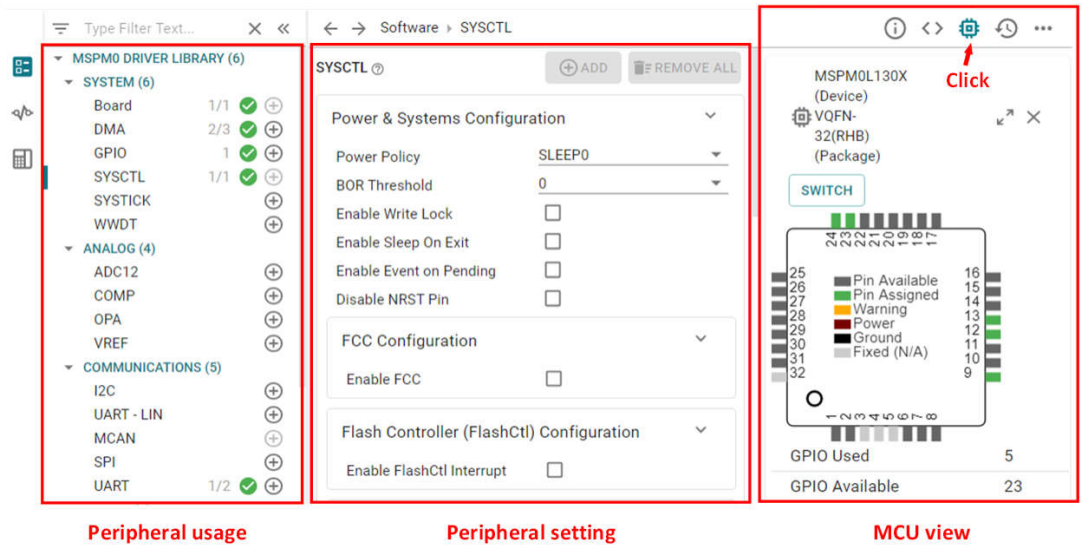


Figure 2-3. MSPM0 SysConfig

Here are the same and different features between SysConfig and STM8CubeMX:

- Both allow creating, saving and importing previously saved projects. STM8CubeMX projects comes with an .ioc8 file that can be saved anywhere, next to other .ioc8 files. But STM8CubeMX does not support C code generation, which is different from SysConfig and can extend development time.
- Both allow easy pinout and clock tree configuration, which can be seen from Chip view.
- STM8CubeMX support power consumption calculation, which, for MSPM0, can be realized in CCS IDE.
- SysConfig supports more comprehensive configuration capabilities. Different from STM8CubeMX,which can only config pinout or simple configuration for peripheral, SysConfig provides more specific and detailed initialization configurations. As shown in Figure 2-4, SysConfig can configure the type of GPIO as well as features such as internal pull-up or pull-down resistors.
- STM8CubeMX comes with an updater mechanism that can be configured for automatic or on-demand check for updates, which supports STM8CubeMX self-updates. SysConfig does not support that and users can download newest version from SysConfig download.

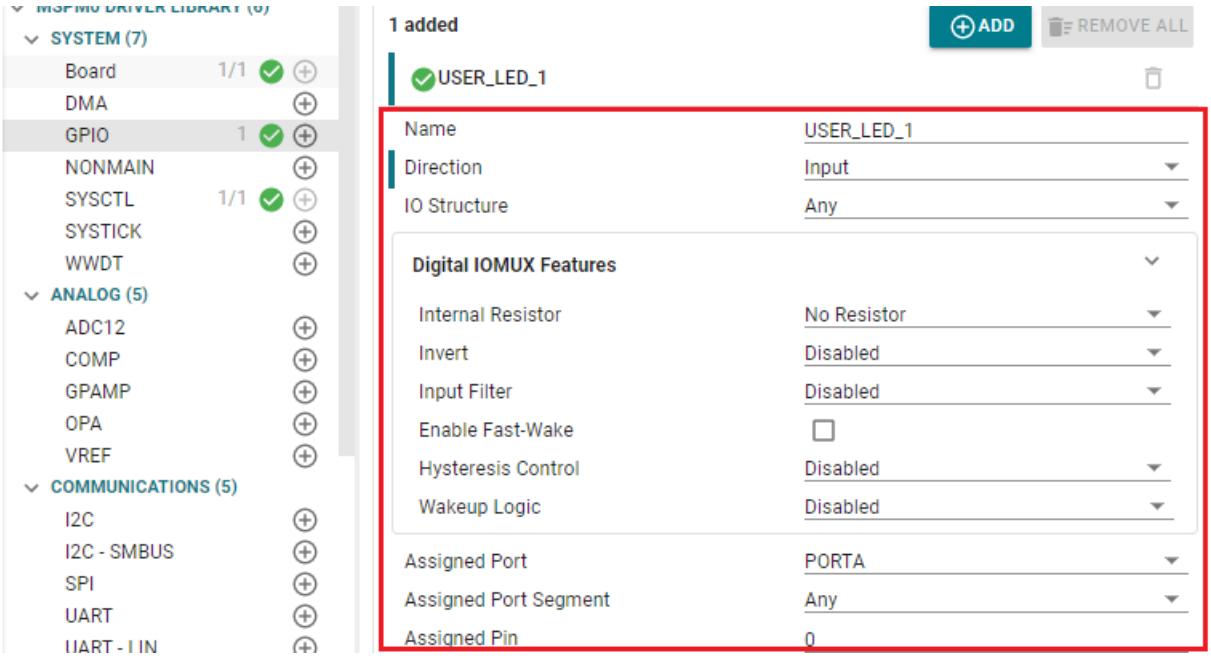


Figure 2-4. GPIO Configuration in SysConfig

2.1.4 Debug Tools

STM support single wire interface module (SWIM) and debug module (DM). In-circuit debugging mode or in-circuit programming mode are managed through a single wire hardware interface featuring ultrafast memory programming. Coupled with an in-circuit debugging module, this also offers a non-intrusive emulation mode, making the in-circuit debugger extremely powerful, close in performance to a full-featured emulator. The typical debugger of STM8 is ST-LINK. The SWIM and JTAG, serial wire debugging (SWD) interfaces are used to communicate with any STM8 microcontroller located on an application board.

For MSPM0, the debug subsystem (DEBUGSS) interfaces the serial wire debug (SWD) two-wire physical interface to multiple debug functions within the device. MSPM0 devices support debugging of processor execution, the device state, and the power state (via EnergyTrace technology). For more details on the connection of the debugger, see Figure 2-5.

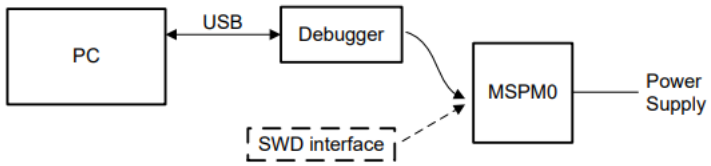


Figure 2-5. MSPM0 Debugging

MSPM0 support XDS110 and J-Link debugger for standard serial wire debug.

The Texas Instruments XDS110 is for TI embedded processors. XDS110 connects to the target board using a TI 20-pin connector (adapters are available for TI 14-pin and Arm 10-pin and 20-pin connectors) and to the host PC using USB2.0 High Speed (480Mbps). The XDS110 supports a wider variety of standards (IEEE1149.1, IEEE1149.7, SWD) in a single unit. All XDS debug probes support Core and System Trace in all Arm and DSP processors that feature an Embedded Trace Buffer (ETB). For details, see XDS110 Debug Probe.

J-Link debug probes are the most popular choice for optimizing the debugging and flash programming experience. Benefit from record-breaking flash loaders, up to 3-MiB/s RAM download speed and the ability to set an unlimited number of breakpoints in the flash memory of MCUs. J-Link also supports a wide range of CPUs and architectures included CortexM0+. For details, see the J-Link Debug Probes page.

Table 2-4 shows a different feature summery between XDS110 and J-LINK debugger supporting MSPM0.

Table 2-4. MSPM0 Debugger Compare

Features	XDS110	XDS110 OB (1)	J-Link
cJTAG (SBW)	√	√	√
BSL (2) tool	√	√	
Backchannel UART	√	√	2.5G (µVision V5.37.0)
Power supply	1.8 - 3.6V	3.3, 5V	5V
IDE (3): CCS	√	√	√
IDE: 3rd party (4)	IAR, Keil	IAR, Keil	IAR, Keil

(1) XDS110 OB means XDS110 onboard.
(2) BSL means bootstrap loader.
(3) IDE means Integrated Development Environment.
(4) 3rd party includes IAR and Keil.

2.1.5 LaunchPad

Like STM8, MSPM0 also has corresponding LaunchPad development kits to support rapid development.

LaunchPad kits are easy-to-use EVMs that contain everything needed to start developing on the MSPM0. This includes an onboard debug probe for programming, debugging, and measuring power consumption with EnergyTrace™ technology. MSPM0 LaunchPad kits also feature onboard buttons, LEDs, and temperature sensors among other circuitry. Rapid prototyping is simplified by the 40-pin BoosterPack™ plug-in module headers, which support a wide range of available BoosterPack plug-in modules. You can quickly add features like wireless connectivity, graphical displays, environmental sensing, and more.

Figure 2-6 illustrates the LaunchPad overview. which contains the MCU and a XDS110 debugger. You can also use other debuggers like J-Link to debug the MCU after removing the jumpers.

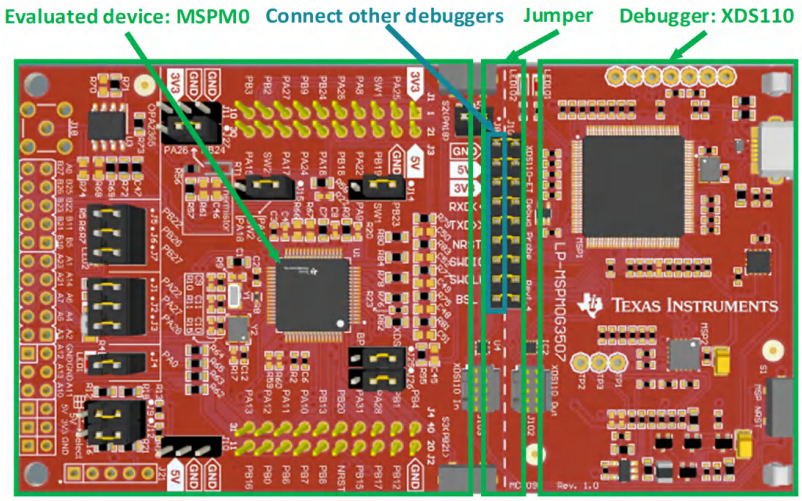


Figure 2-6. MSPM0G3507 Launchpad Overview

Jumper isolation block contains Power(GND,5V,3.3V), UART(RXD, TXD), reset pin, arm debug channel(SWDIO,SWCLK) and BSL.

In addition to jumper caps, users can burn using the standard Arm Cortex 10 pins connector (as shown in Figure 2-7) which is located on the right side of the Launchpad. The Cortex Debug Connector supports JTAG

debug, Serial Wire debug and Serial Wire Viewer (via SWO connection when Serial Wire debug mode is used) operations.

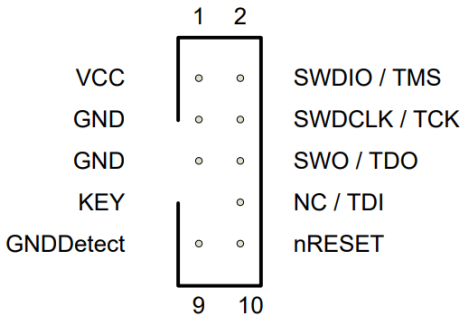


Figure 2-7. Arm Cortex 10-Pin Definition

Figure 2-8 shows some feature function of MSPM0G3507 Launchpad.

The lower sides of the Launchpad are the connectors of the booster pack, which are used to plug in specific functional modules directly and quickly build prototypes. In addition to this, it's also possible to use DuPont wire alone to lead out for quick use. The Launchpad has a user-defined button on each side, a temperature sensor, a light sensor, a monochrome LED, and an RGB LED underneath.

- LP-MSPM0G3507 LaunchPad development kit LP-MSPM0G3507 Evaluation board | TI.com
- LP-MSPM0L1306 LaunchPad development kit LP-MSPM0L1306 Evaluation board | TI.com
- LP-MSPM0C1104 LaunchPad development kit LP-MSPM0C1106 Evaluation board | TI.com

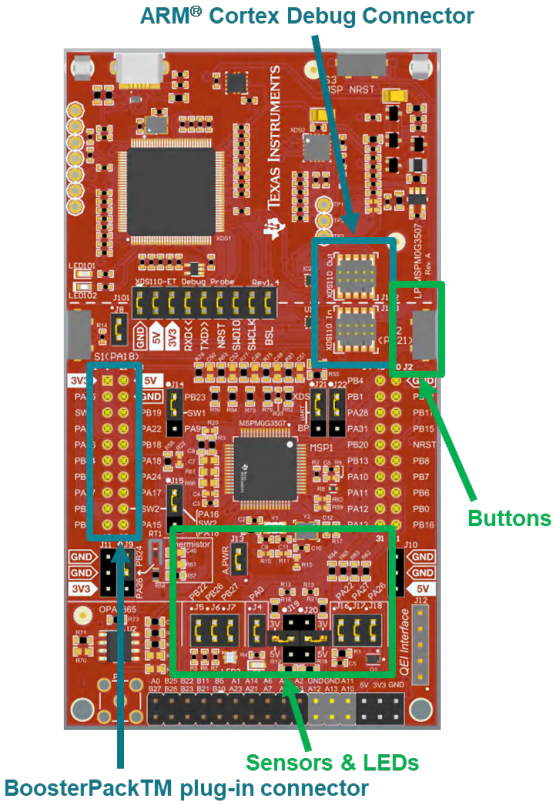


Figure 2-8. MSPM0G3507 Launchpad Feature Function

2.2 Migration Process

For smooth migration to MSPM0, the detailed process is written in a flow as shown in Figure 2-9. Each step is described in detail and examples are given in the following sections.

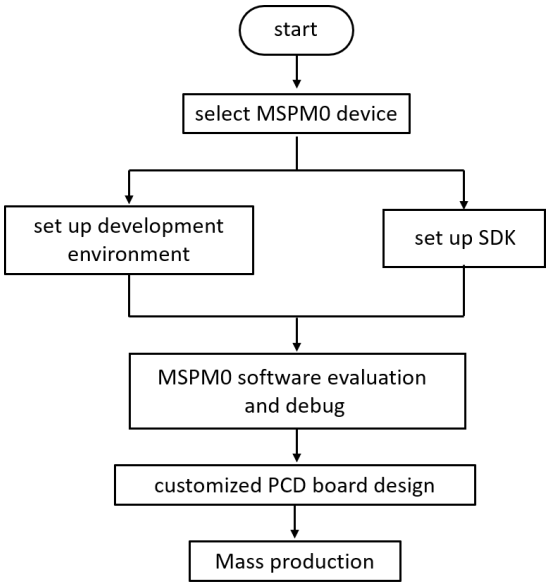


Figure 2-9. MSPM0 Migration Flowchart

2.2.1 Step 1: Choose The Right MSPM0 MCU

The first step of migration is to choose the correct MSPM0 device for the application. Figure 2-10 shows the portfolio of MSPM0 C, L and G series family, which can be seen in official website of TI. Both portfolios all distinguish the device according to memory and packaging, which makes selecting a device easier.

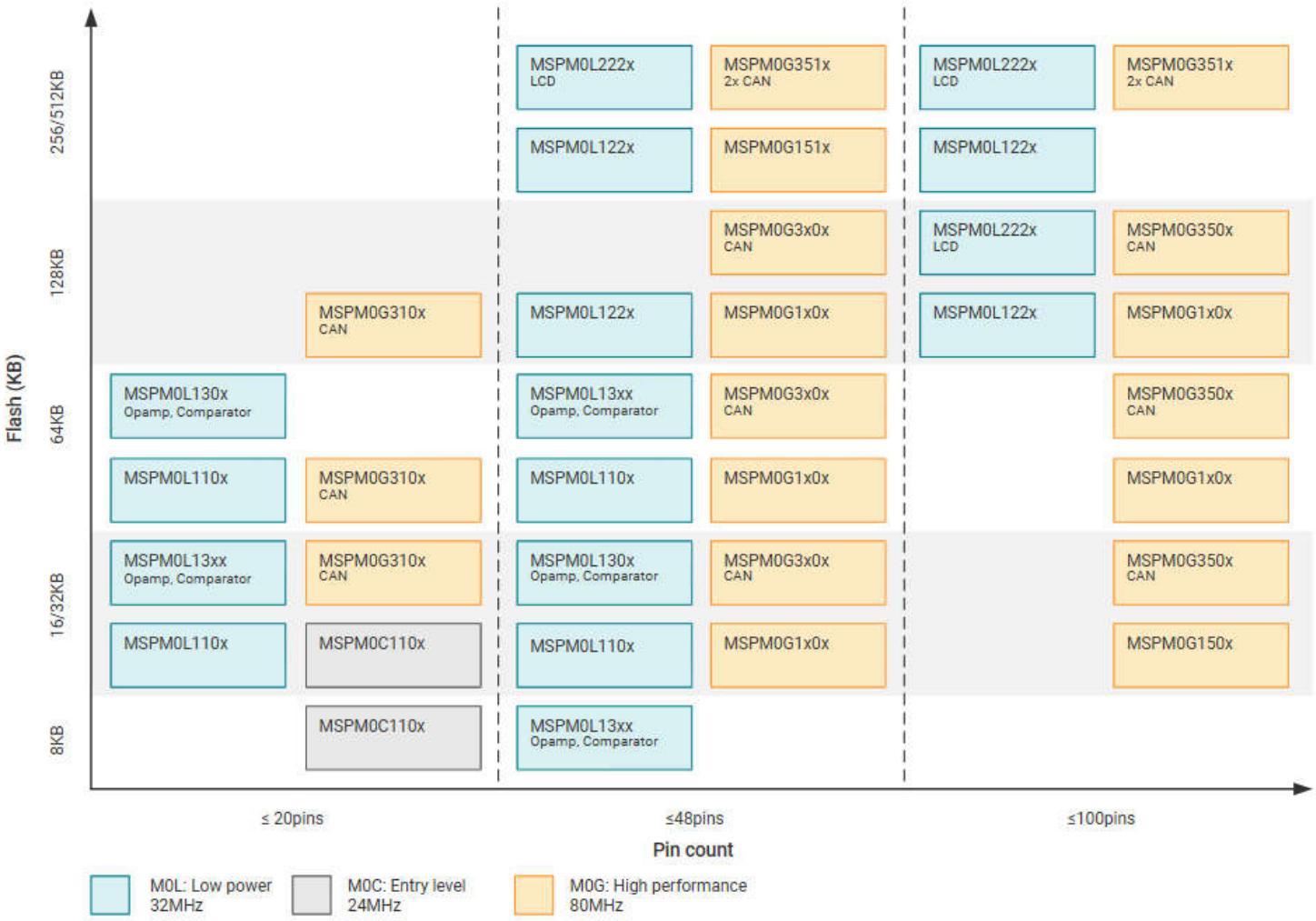


Figure 2-10. Portfolio of MSPM0C, MSPM0L and MSPM0G Series

To narrow down to a specific device, the product selection tool plays an import role. In this link, users use the filter on the left to do initial screening based on the MCU peripheral requirement. For example, to filter out the MCU that meet the UART number, users can directly use the filter tool to select, and the qualified MCU devices pops up on the right, as shown in the Figure 2-11. Users can directly go to the device page through the left search text box for detailed information of corresponding device.

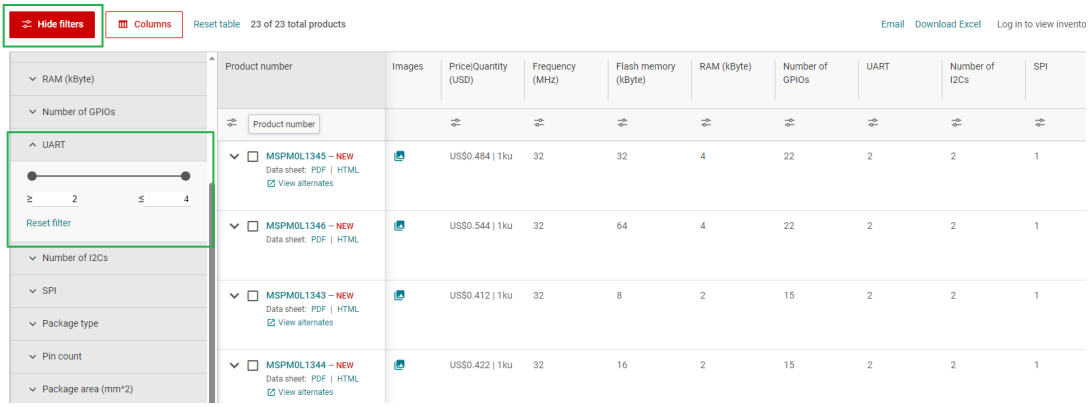


Figure 2-11. MSPM0 Product Selection Tool

On the device page, the key documents like the data sheet, Technical Reference Manual (TRM) and Errata can be found and downloaded easily, as shown in Figure 2-12. The device-specific data sheet introduces the parameters and functional data information of dedicated MSPM0. The device-specific TRM introduces the application method and characteristics of a series MSPM0. The device-specific errata introduces the description of MSPM0 related series or versions.

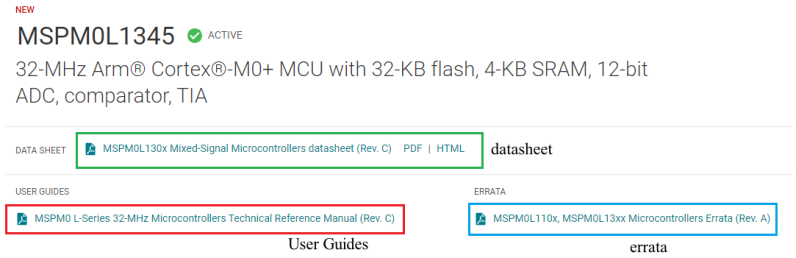


Figure 2-12. MSPM0 Important Document List

As shown in Figure 2-13, the website also lists the relevant technical documents on MSPM0, the most common being application notes.

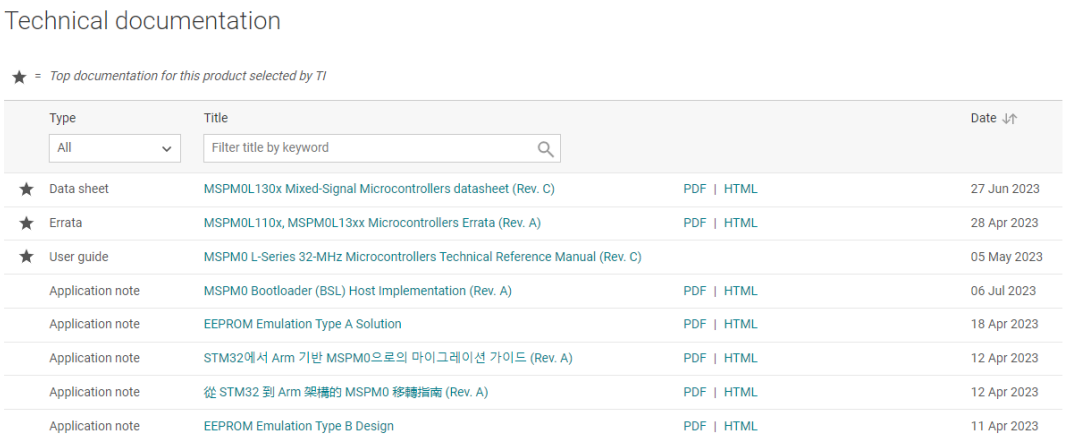


Figure 2-13. MSPM0 Relevant Technical Documentation List

After completing the selection, users can check the price and other information through ordering and quality, as shown in Figure 2-14.

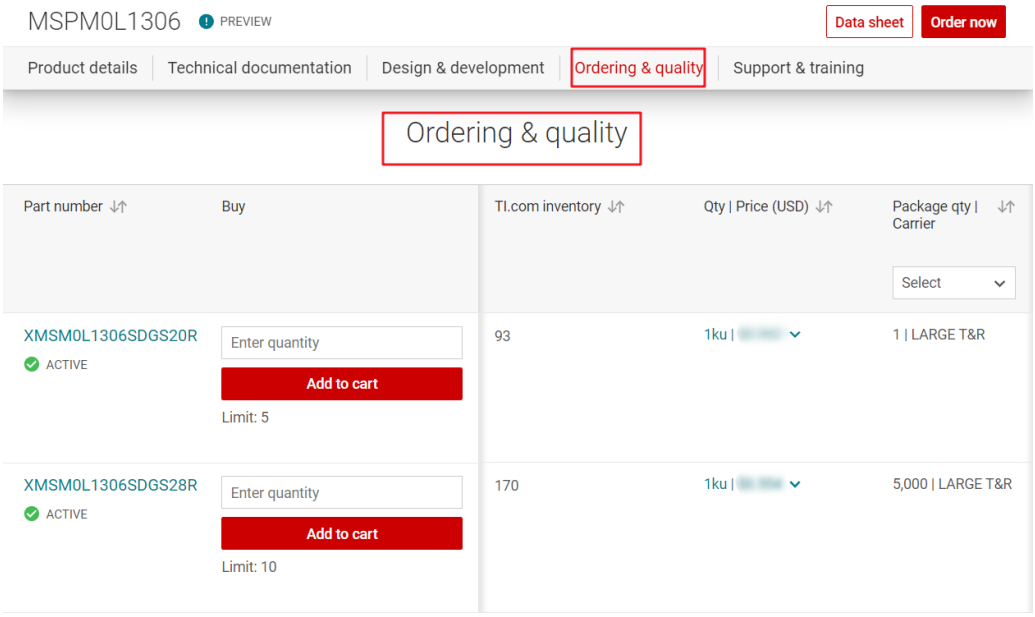


Figure 2-14. Ordering and Quality Part View

2.2.2 Step 2. Set Up IDE And Quick Introduction of CCS

2.2.2.1 Set Up IDE

TI's CCS is the chosen IDE.

1. Click the link to download and then start installation. In installation process, keep pressing next.

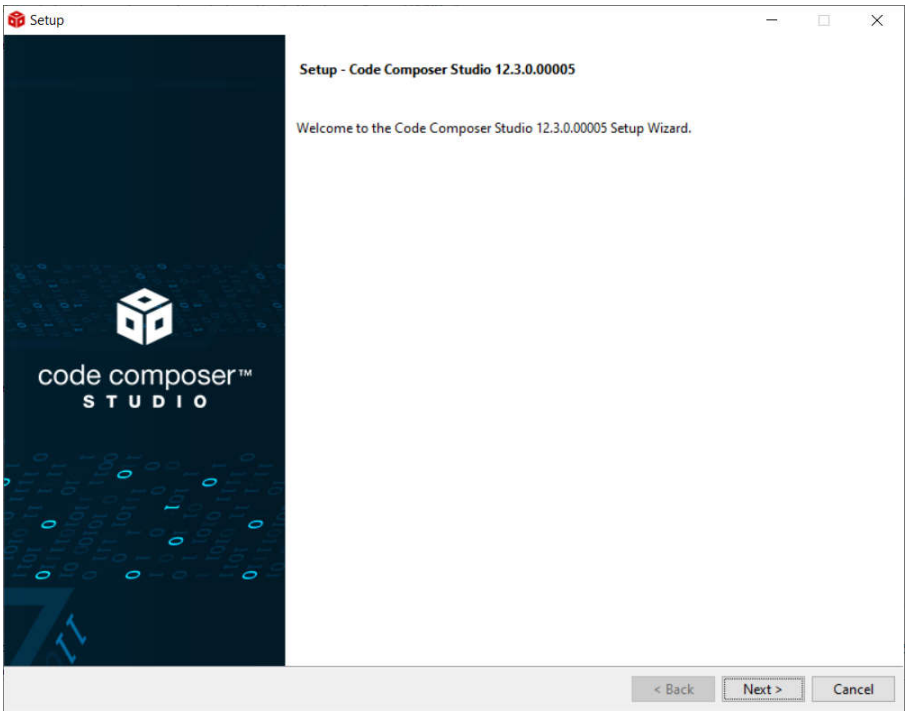


Figure 2-15. CCS Installation

2. Select MSPM0 support component.

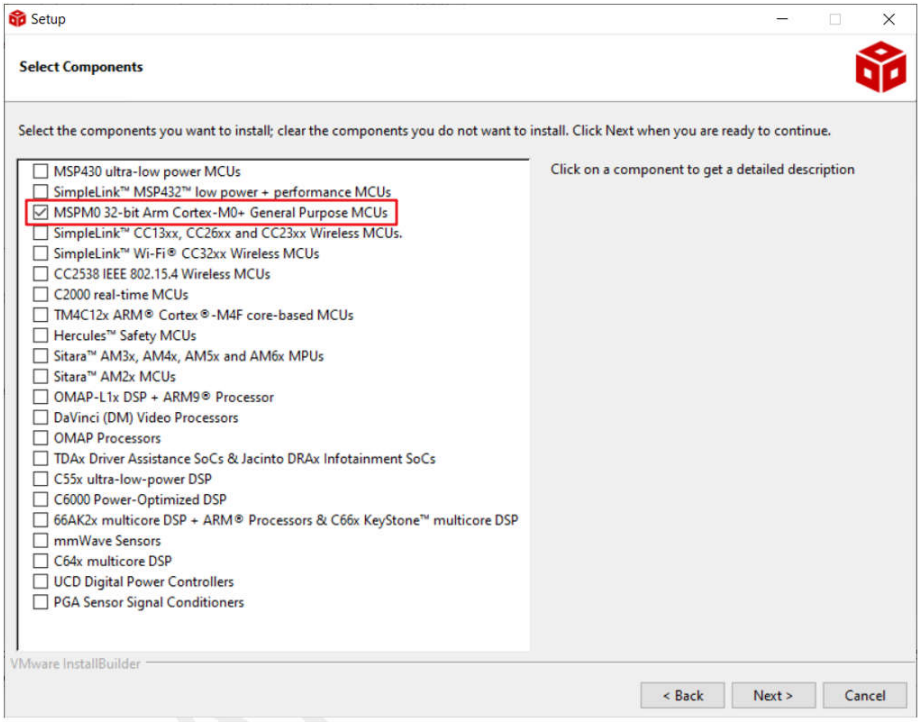


Figure 2-16. CCS Installation- MSPM0 Support Selection

3. Select J-link, if needed.

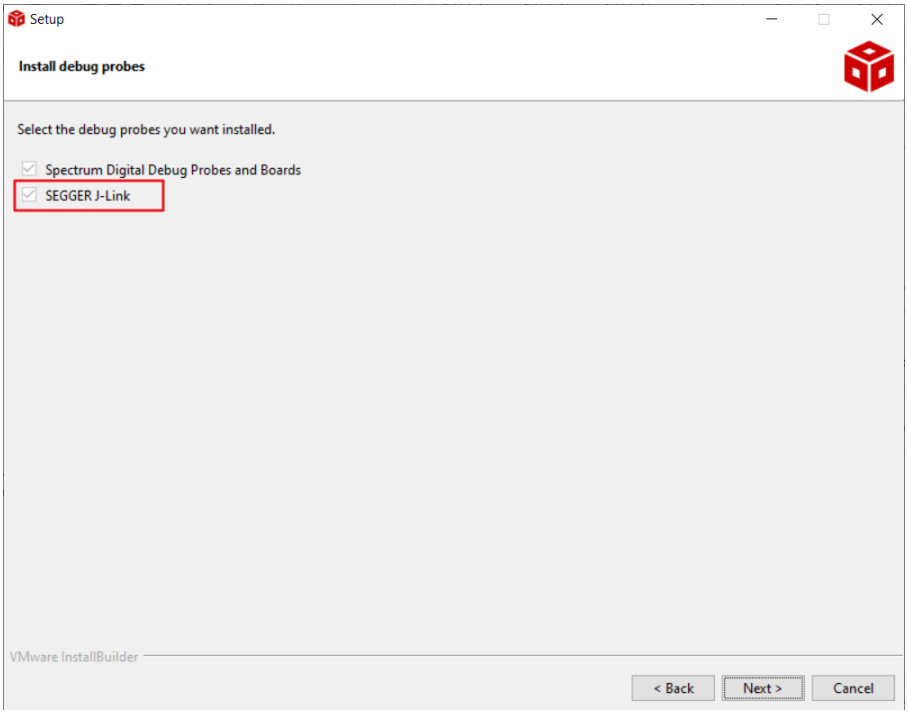


Figure 2-17. CCS Installation- J-Link Download Selection

4. Finish CCS download.

2.2.2.2 Quick Introduction of CCS

1. Launch a new workspace. The workspace means the address where to copy an imported project.

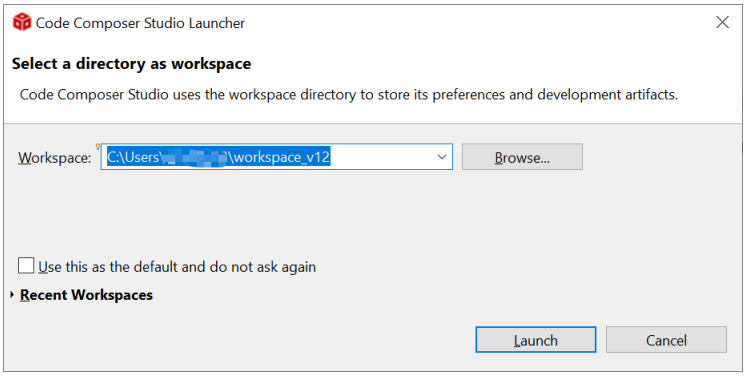


Figure 2-18. CCS Launch Workspace

2. If users want to create a new project, then go to file--> new-->CCS project. There are two important items that need to be done. The first one is to choose MSPM0 device and the other one is to choose the connection, as shown in Figure 2-19. Then, the program can be created after project name is added and press finish button. TI recommends to find the MSPM0 SDK example, which introduces how to use CCS in Section 2.2.4.

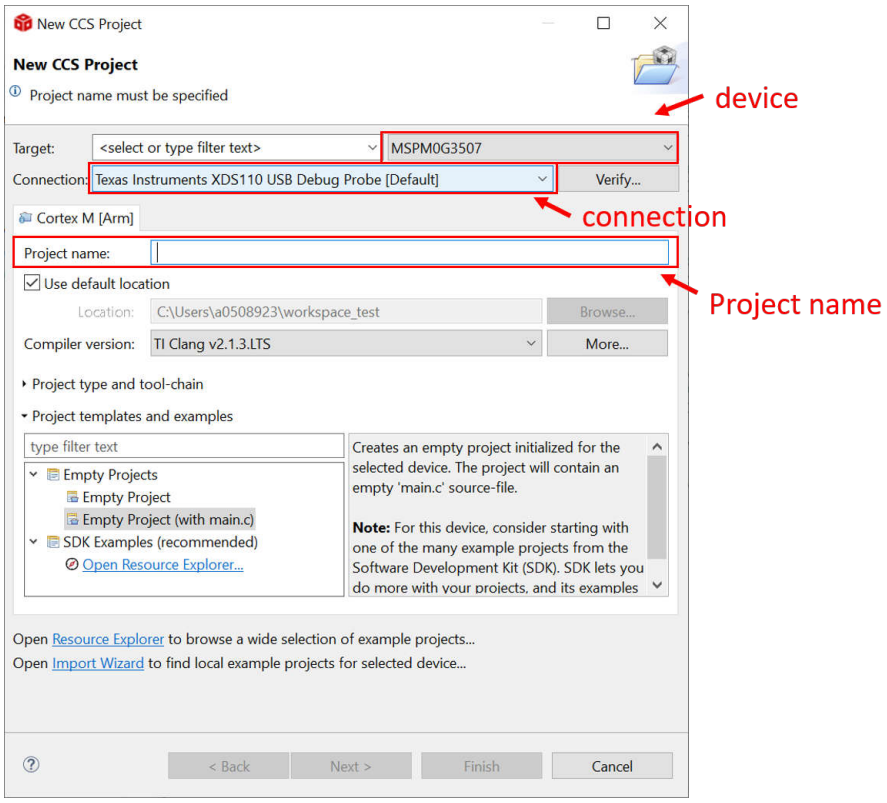


Figure 2-19. Create a New Project in CCS

3. Figure 2-20 and Figure 2-21 show a quick introduction to CCS functions.

Shortcut key functions：

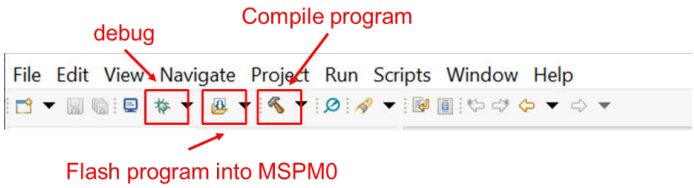


Figure 2-20. Commonly Used Function

Debug functions:

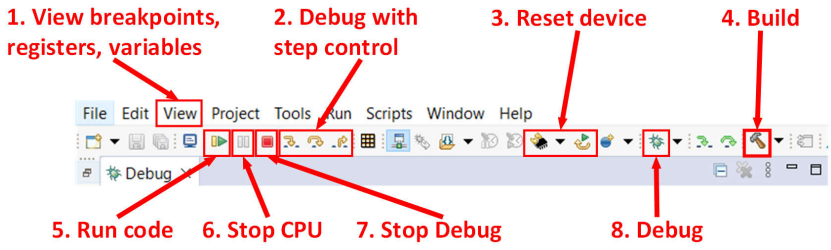


Figure 2-21. Commonly Used Debug Functions

Project properties common used settings:

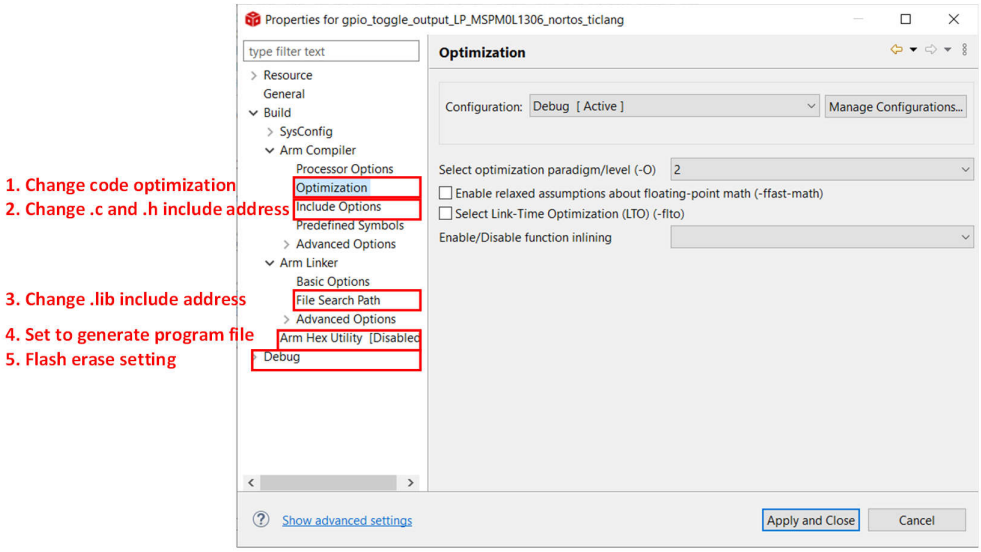


Figure 2-22. Commonly Used Project Settings

For detailed information, refer to Code Composer Studio IDE Version 12.4+ for MSPM0 MCUs .

2.2.3 Step 3. Set Up MSPM0 SDK And Quick Introduction of MSPM0 SDK

STM8 gives a "Standard Peripheral Library", enabling developers to easily exploit all the functions of the STM8 microcontrollers to address a wide range of applications. And TI also give the similar support.

2.2.3.1 Set Up MSPM0 SDK

1. Click the link to download MSPM0 SDK.

Figure 2-23. MSPM0 SDK Download

2. Press next to install SDK.

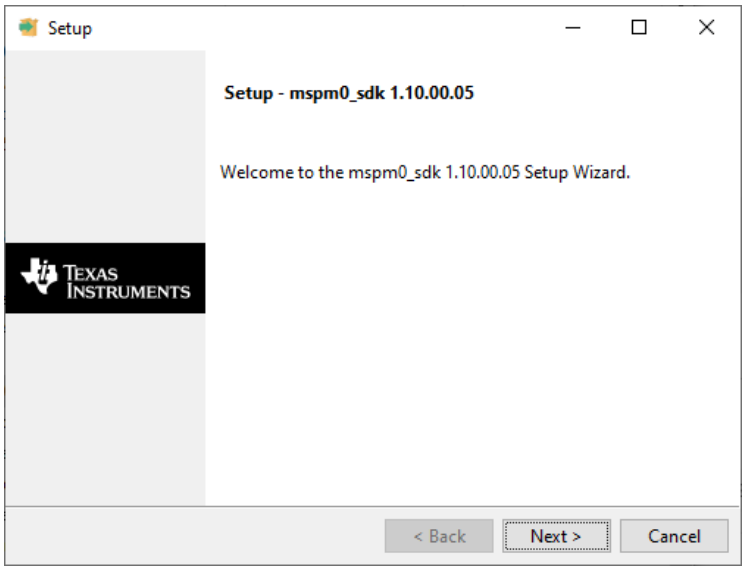


Figure 2-24. MSPM0 SDK Installation

3. Finish MSPM0 SDK download.

2.2.3.2 Quick Introduction of SDK

When downloading is finished, the file content is shown in the SDK folder, which is *c:/ti/mspm0_sdk_xxx* by default, as shown in Figure 2-25. Among which, the mostly used folders are examples folder and docs folder.

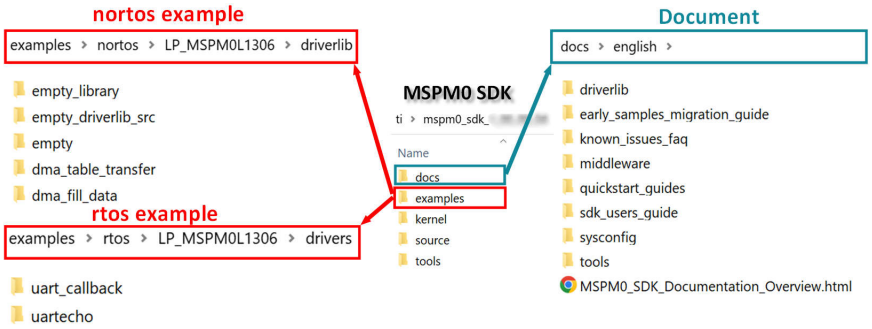


Figure 2-25. MSPM0 SDK Fold

Table 2-5 shows a summary of example coverage.

Table 2-5. MSPM0 Example Coverage

Supported by SDK	Platform			
IDE	CCS		Keil	IAR
Compilers	TI Arm-Clang	GUN Arm	Arm/Keil Compiler	IAR Arm compiler
RTOS	FreeRTOS			
Code examples	Driverlib, TI Dirvers(drivers)			

In nortos examples, users can also find three empty projects for users to build a project.Table 2-6 shows the differences.

Table 2-6. Empty Project Description

Example	Use Sysconfig	Include Library Files Into Project
empty	Yes	No
empty_library	No	Yes
empty_driverlib_src (Suggested)	Yes	Yes

As for docs folder, the structure and the important documents are shown in Figure 2-26.

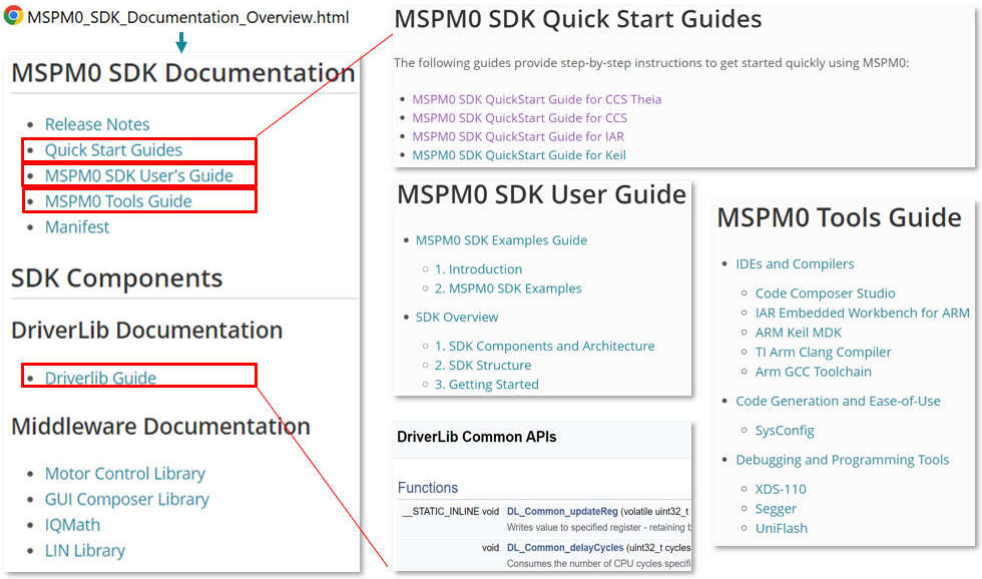


Figure 2-26. Document Overview

2.2.4 Step 4: Software Evaluation

The following are steps to port the example into CCS.

1. Select *Project*, and then import CCS Projects from the menu.

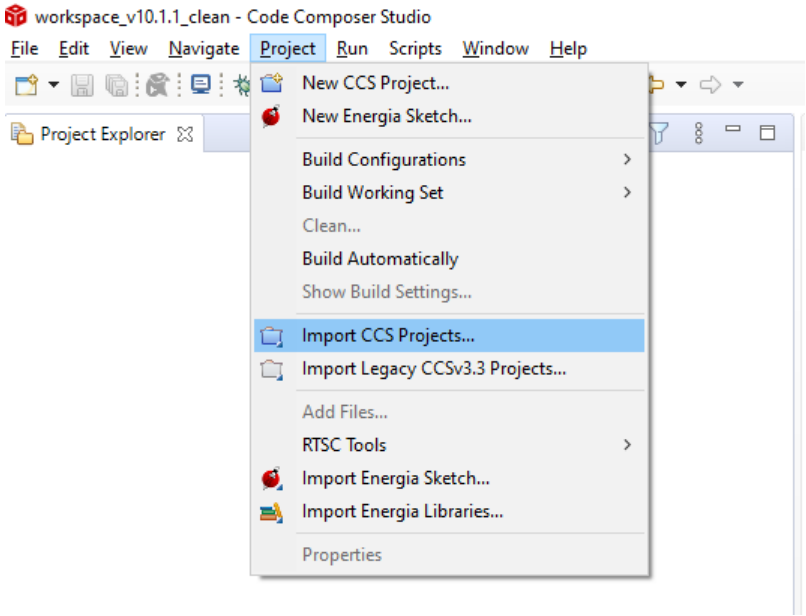


Figure 2-27. Import CCS Projects

2. Choose the program from SDK. Take the MSPM0L1306, for example.

   \mspm0_sdk_1_10_00_05\examples\nortos\LP_MSPM0L1306\driverlib

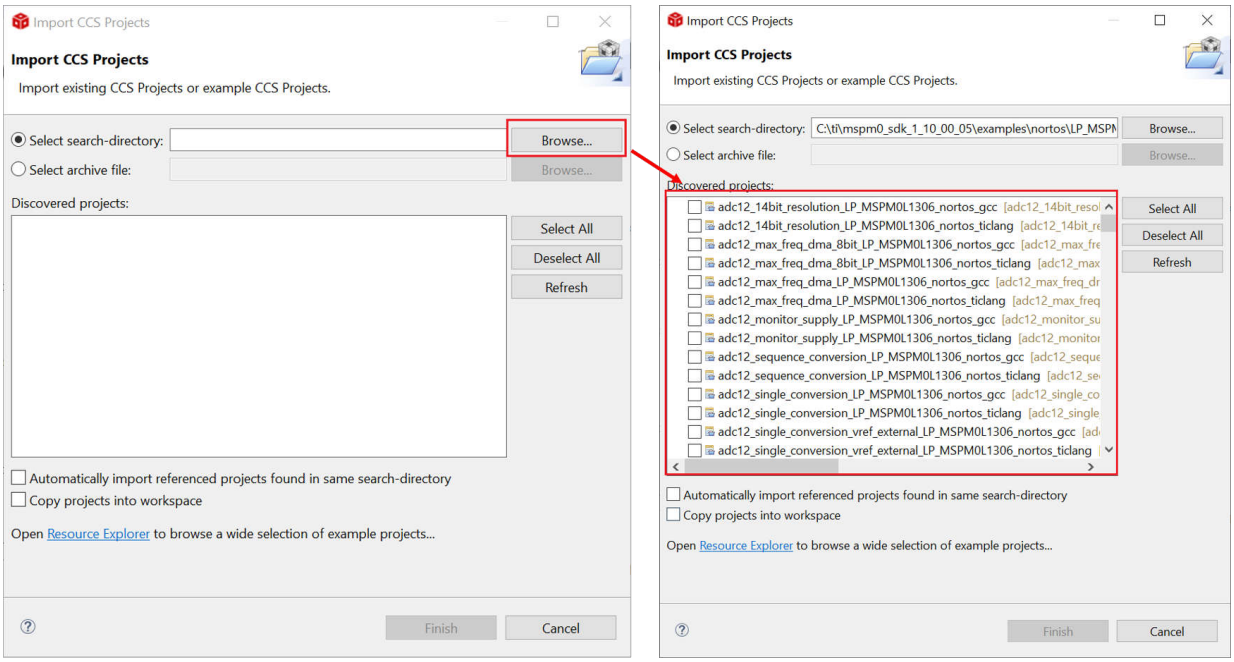


Figure 2-28. Choose Program From SDK

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

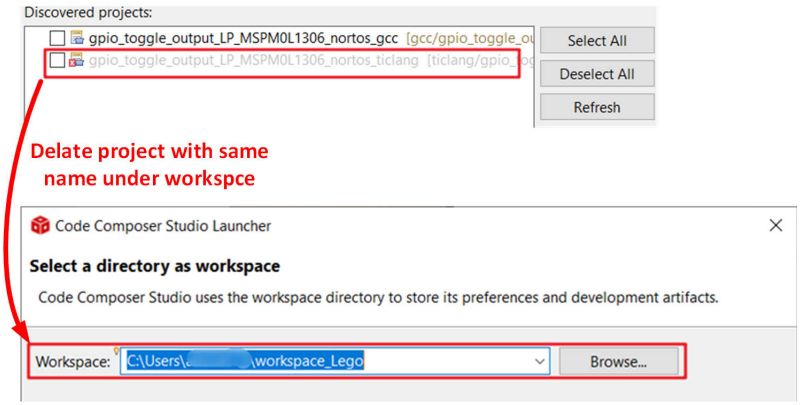


Figure 2-29. Remove Duplicated Project

3. After importation, there is a project on the left, and a REAME.md automatically opens. TI recommends for users to read the README.md file first, which contains the purpose of this example and the hardware configuration.

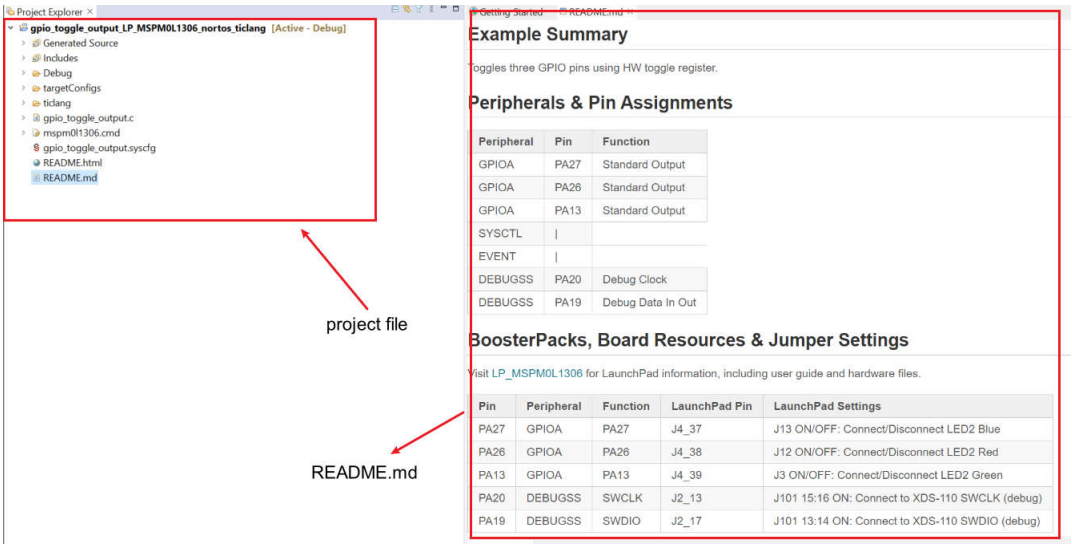


Figure 2-30. Project and README.md

4. Figure 2-31 shows the most important files in the project.

Figure 2-31. CCS Project Overview

5. Similar to STM development, TI also supports chip view. Double-click .syscfg file to reach to SysConfig, where users allowed to configure the required peripherals through a graphical interface. And, TI recommends to use the MCU view of SysConfig to help users fix the pin function first with a software engineer, which is similar to MCU, MPU Package in STM8CubeMAX.
6. Based on the code and SysConfig example, users can polish the project or modify with a device-specific TRM or application note released on Ti.com.
7. If users want to add third-party libraries, then follow the steps below. First, add relevant file into the project, as shown in Figure 2-32.

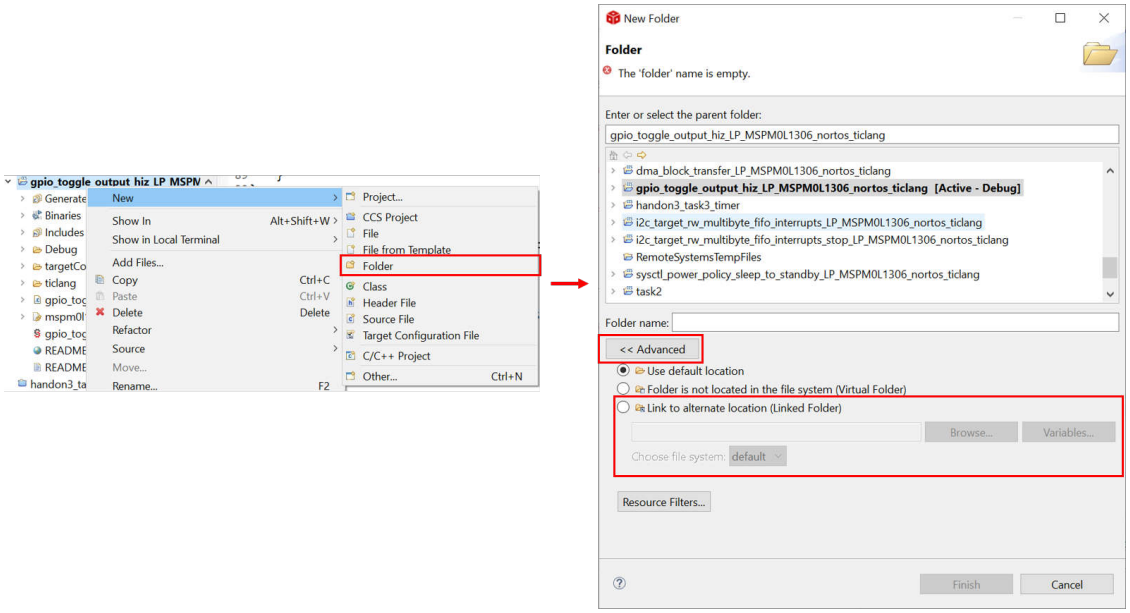


Figure 2-32. Add Relevant File

Then, other steps need to be done to tell the compiler to add header files.

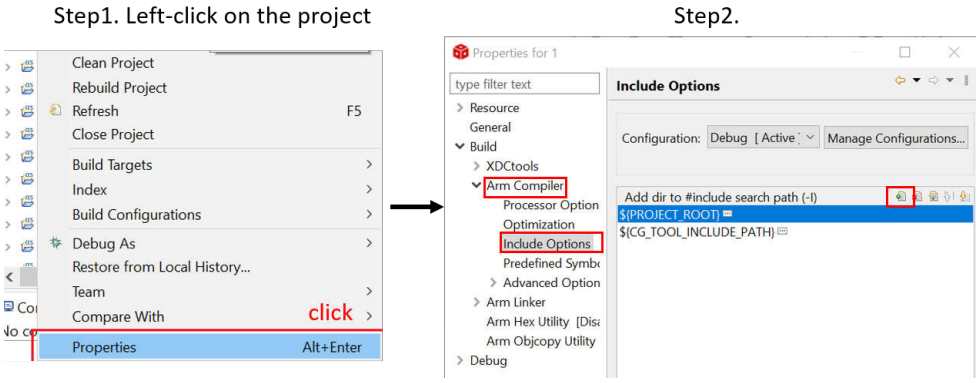


Figure 2-33. Include Options Set

8. As users finish evaluating the software, click the *build* icon in the main toolbar, as shown in Figure 2-34. The appearance of *Build Finished* means the compilation is successful.

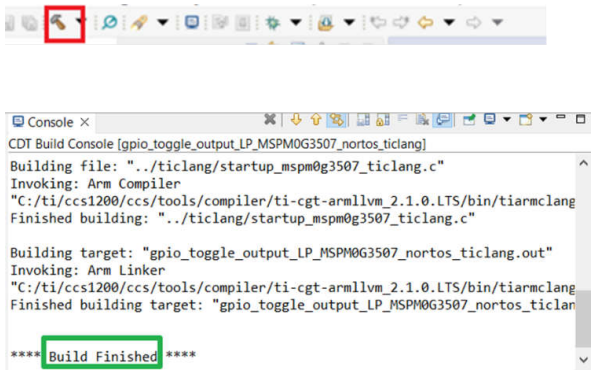


Figure 2-34. Successful Build

2.2.5 Step 5. PCB Board Design

1. To get the design package, go through the Ti.com and enter the specific device page. Take MSPM0L1304, for example.
2. Click Design and development --> CAD/CAE symbols, select different package models to download according to your needs.

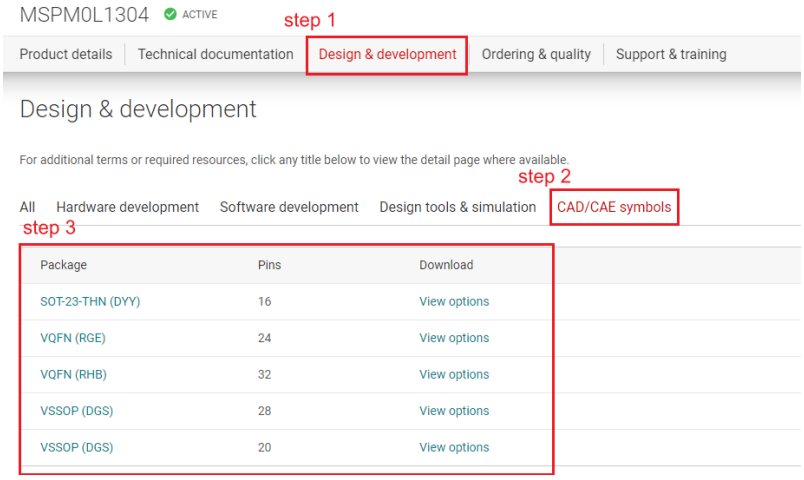


Figure 2-35. Ultra Librarian Tool Entrance

3. As for MSPM0 hardware design into your own board. Figure 2-36 shows a sample minimal system design.

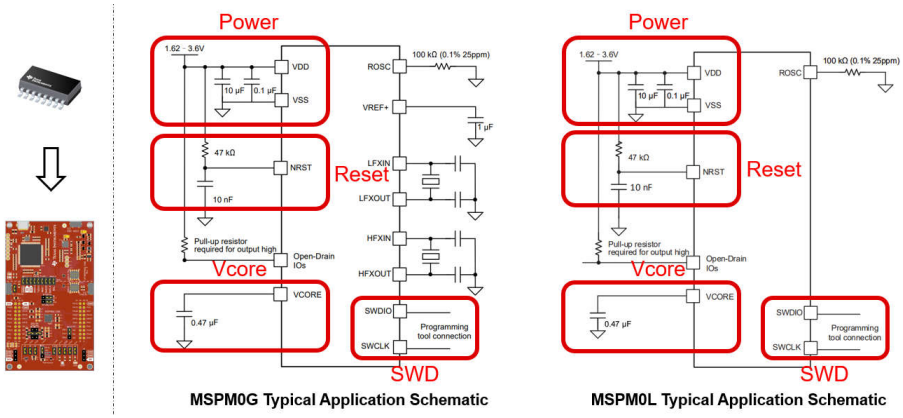


Figure 2-36. MSPM0 Minimum System

For minimal systems, the following points need to be noted:

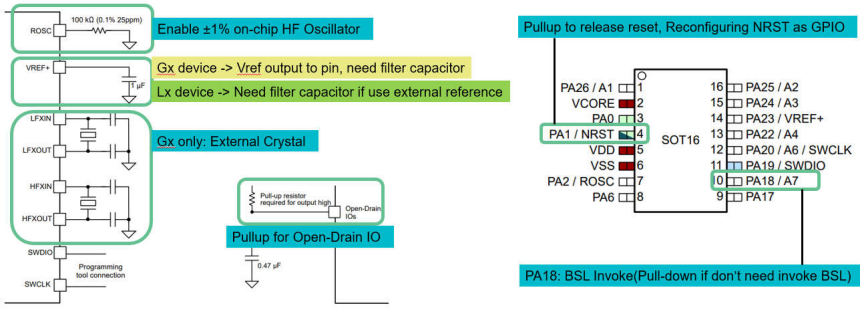


Figure 2-37. MSPM0 Minimum System Attention

For more detailed information about hardware development, see the following:

- MSPM0 G-Series MCUs Hardware Development Guide
- MSPM0 L-Series MCUs Hardware Development Guide

2.2.6 Step 6. Mass Production

1. Generate production files(.bin/.txt/…) through CCS.

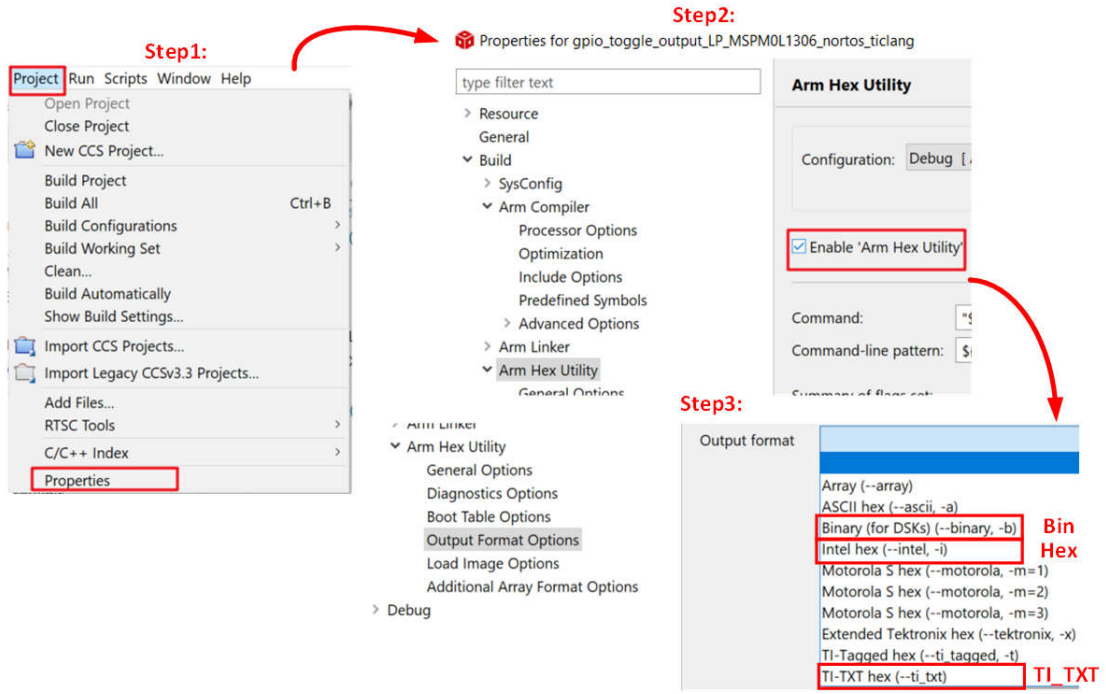


Figure 2-38. Create Program Files

2. Choose the programmers/debuggers to program up MSP device.

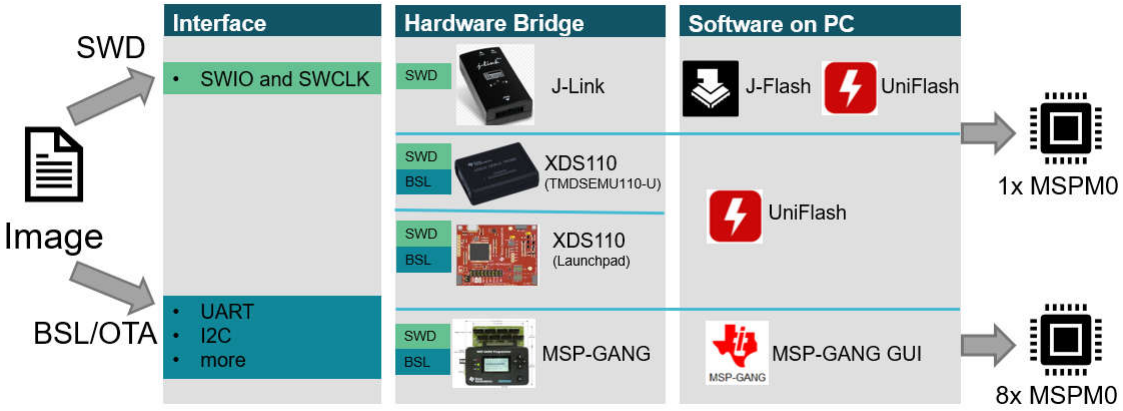


Figure 2-39. Program Software and Tool

For the using of MSP-GANG and J-LINK,please refer to: MSPM0 Design Flow Guide.

For more information about debugging , see: Debugging and Programming Tools guide.

2.3 Example

The MSPM0 design flow is shown below. This example aims to use PWM to drive LED.

1. Choose the right MSPM0 MCU and select hardware and order an EVM, here we use Launchpad MSPM0L1306.
2. Set up CCS and SDK, detail can be seen in Section 2.2.
3. Code import.

   When the environment is ready, you can import code into CCS. As for this example, a timer is used to control PWM. The first thing to do is understand any differences between the timer modules between STM8 and MSPM0, and choose the similar example in SDK of MSPM0.

   The closet example in the SDK is probably *timx_timer_mode_pwm_center_stop*. Once a similar example is found, open CCS and import the code example by going to Project --> Import CCS Projects... and navigate to the MSPM0 SDK example folder.

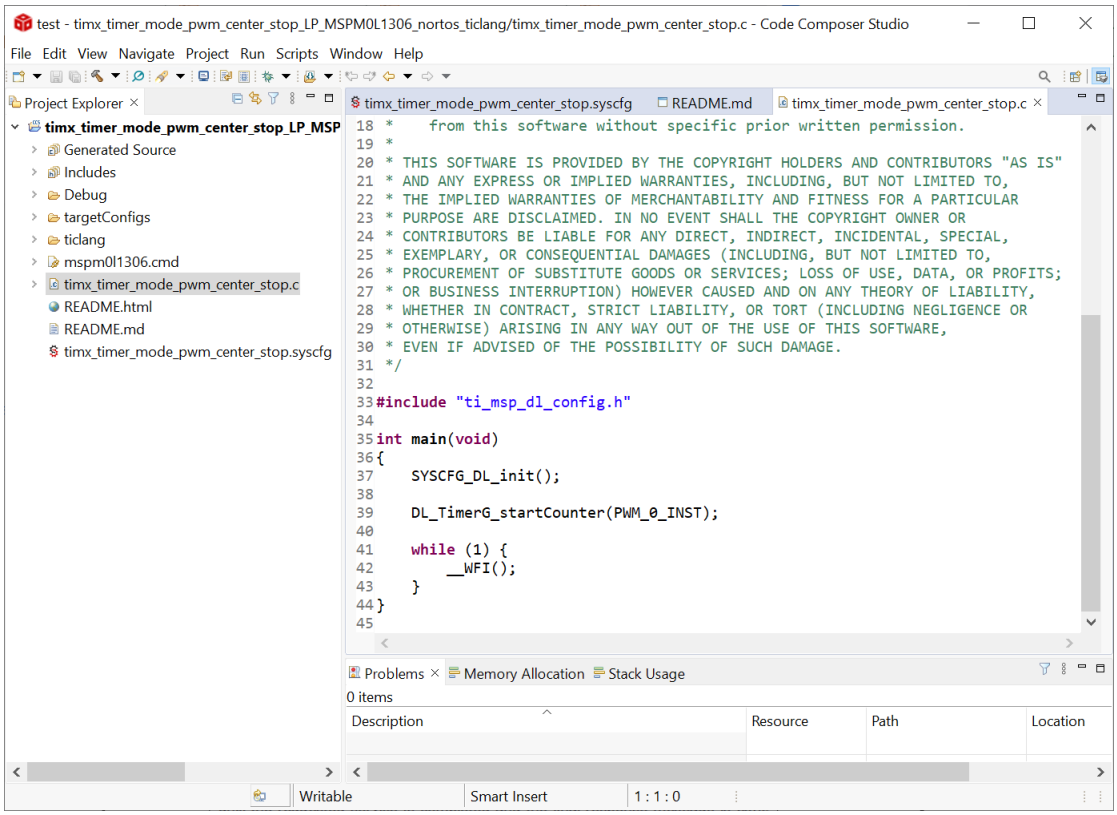


Figure 2-40. Code Example File

4. Modify project.

   To see the SysConfig configuration, open the .syscfg file. Select TIMER-PWM section to generate PWM, as shown in Figure 2-41. Check the PWM's clock configuration, like self frequency and duty cycle. In this case, PWM frequency is 2.7Hz and 75% duty cycle. You can change duty cycle easily through typing 50% in desired duty cycle, and then Counter compare Value changes automatically.

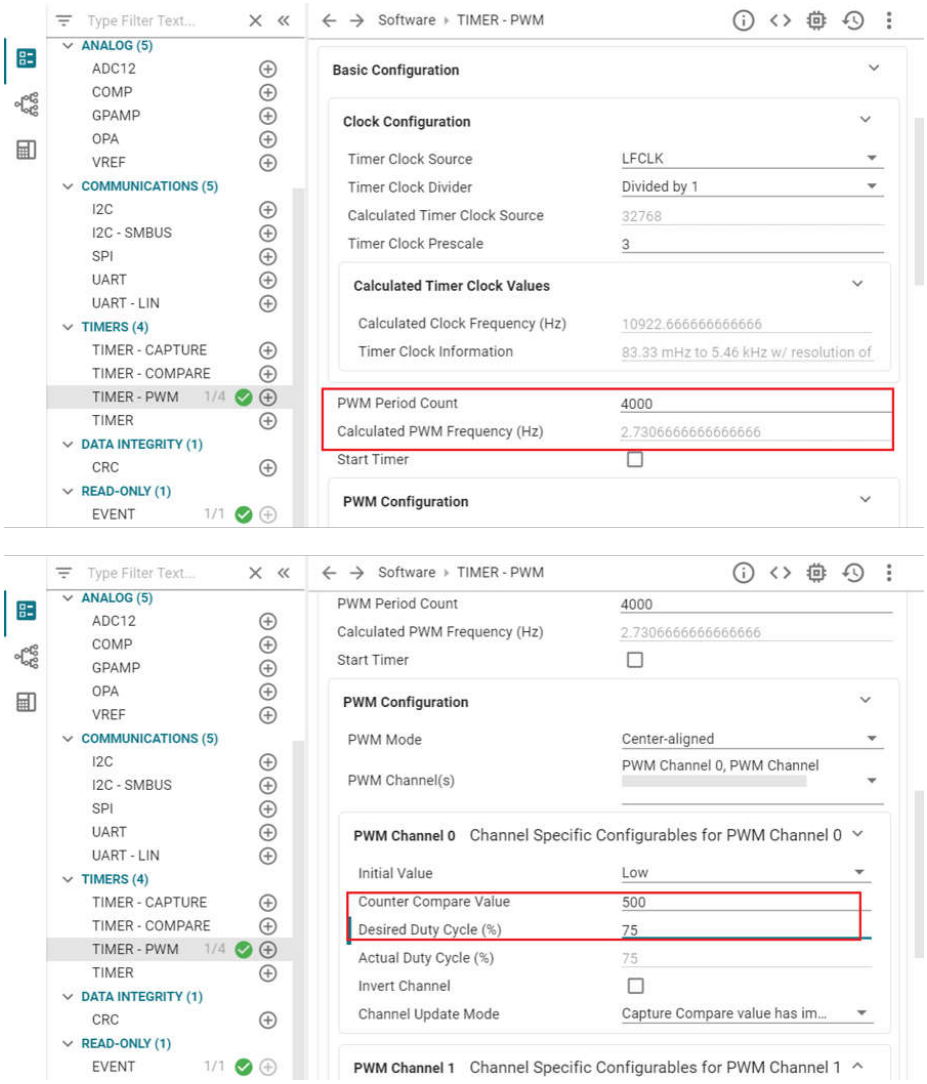


Figure 2-41. PWM Configuration in SysConfig

To further elaborate on each feature module, you can click "?" next to each item.

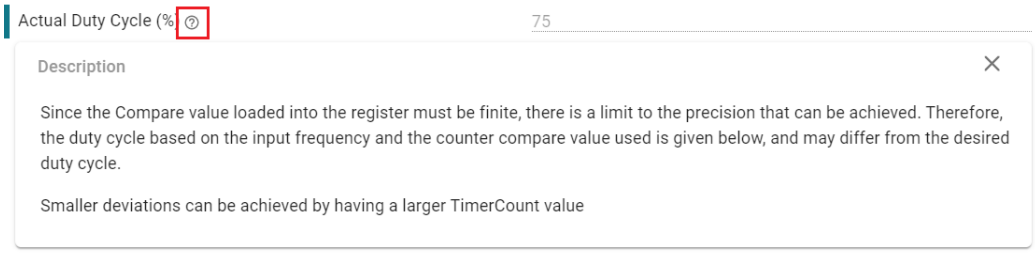


Figure 2-42. To Get Detailed Information of Each Item

Also check the rest feature of TIMER-POWER module and pins being used by clicking the chip icon in the top right and checking the highlighted pins for the PWM.

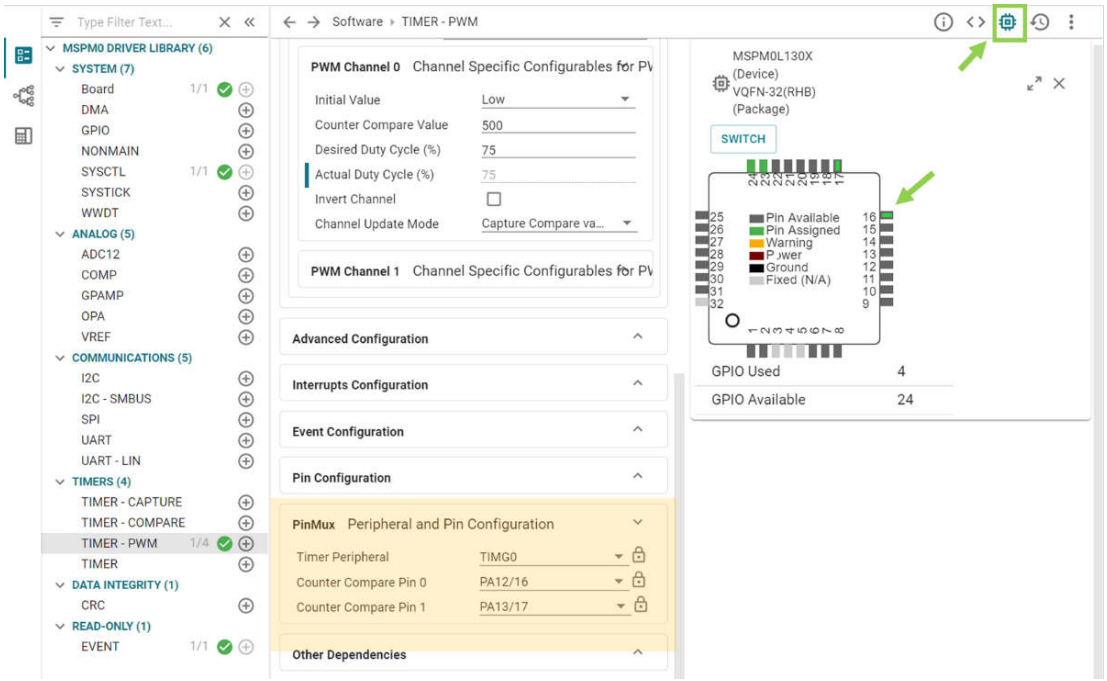


Figure 2-43. Pins Configuration

When the project is saved and rebuilt, SysConfig updates the files in Figure 2-44. At this point, the example hardware configuration has been modified to match the full functionality of the original software being ported.

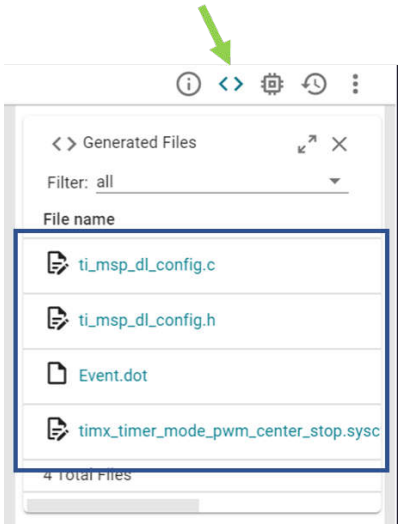


Figure 2-44. The Files SysConfig Updates

The only remaining effort is to check application-level software. This example generates PWM waves like SDK code, so there is no need to change the .c file.

5. Hardware setup.

   Get LaunchPad plugged into the computer. According to pins configurations, use DuPont cables to connect the PA12 to the LED pins.

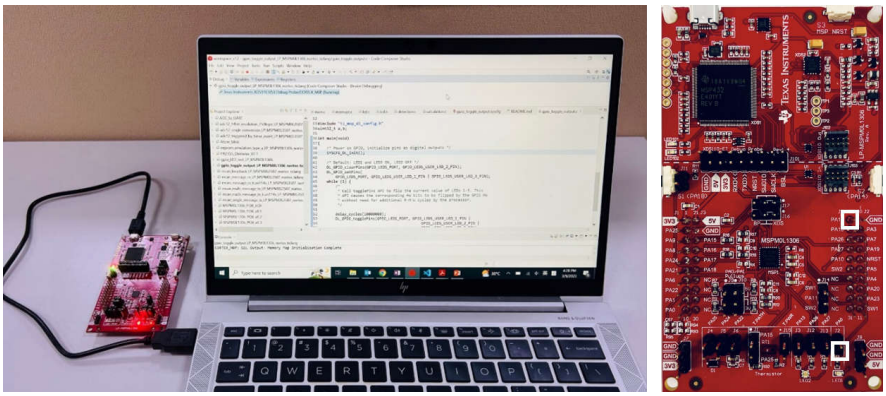


Figure 2-45. Hardware Setup

6. Debug and verify.

   Start debug, by clicking debug icon. And you can set breakpoint by double-click the space before the line number or adding one line code *__BKPT();*

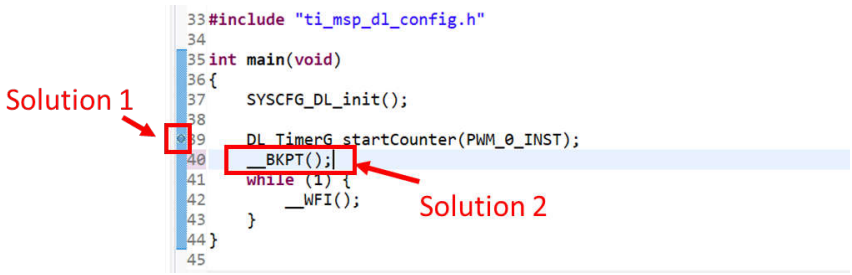


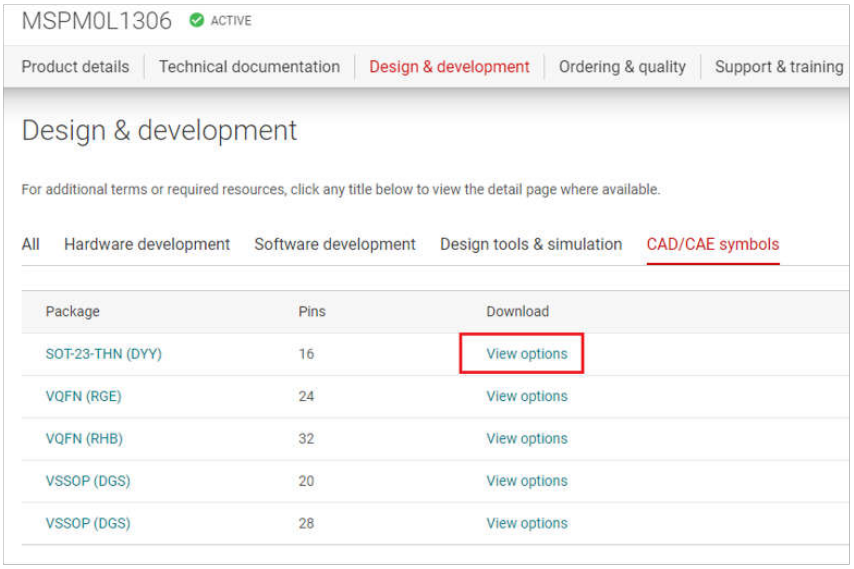
Figure 2-46. Add Breakpoint Solutions

   Try to use debug functions (detailed can be seen in Section 2.2.2.2) and verify the feasibility of the procedure. While debugging, LED can be toggled as code is running step by step.

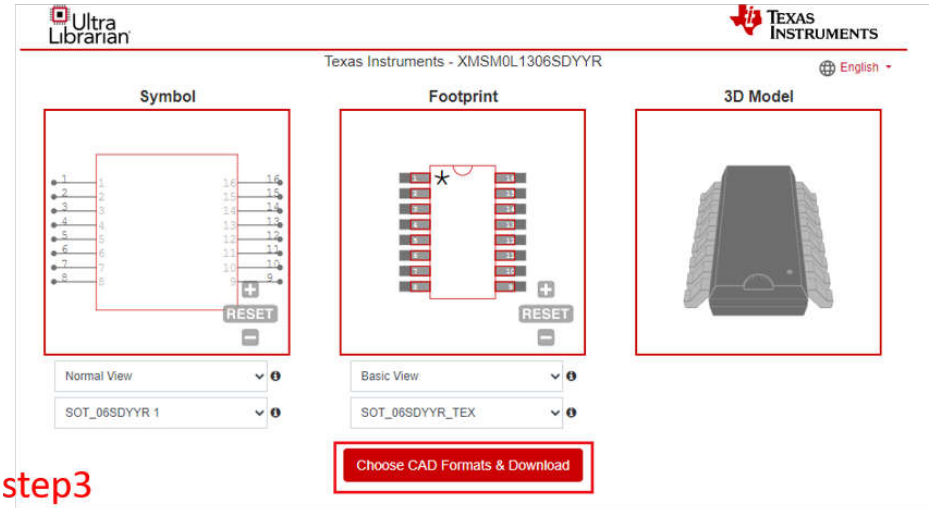
7. Generate PCB library and import to Altium Design.

   The specific steps are shown in Figure 2-47. Go to the entrance of Ultra Librarian tool under MSPM0 device page (detailed can be seen in Section 2.2.5). Clink *View options*. Select your wanted CAD format and Pin ordering, then you can get the Altium design lib file.

step1



step2



step3

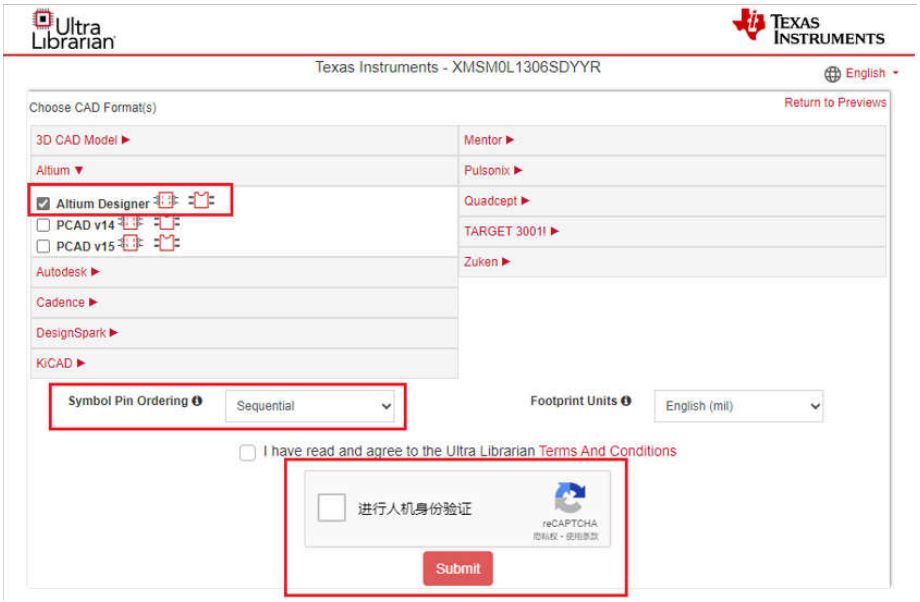


Figure 2-47. Ultra Librarian Tool Download

As the lib have been downloaded, the next step is to run Altium Designer script and generate PCB lib and schematic library, as shown in Figure 2-48.

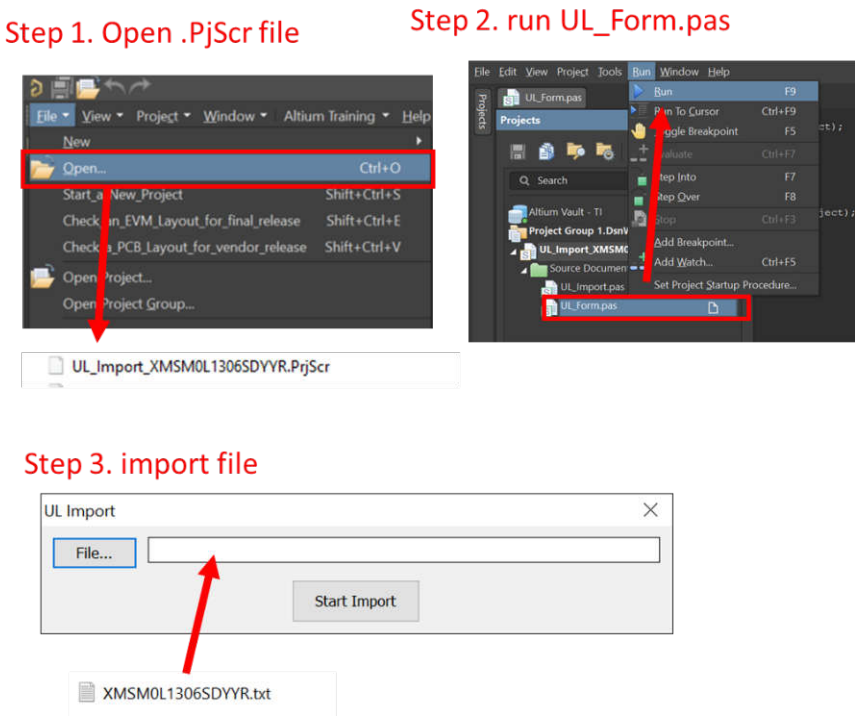


Figure 2-48. Run Altium Designer Script

After completing the steps, the following new files are going to generated in the same source folder.

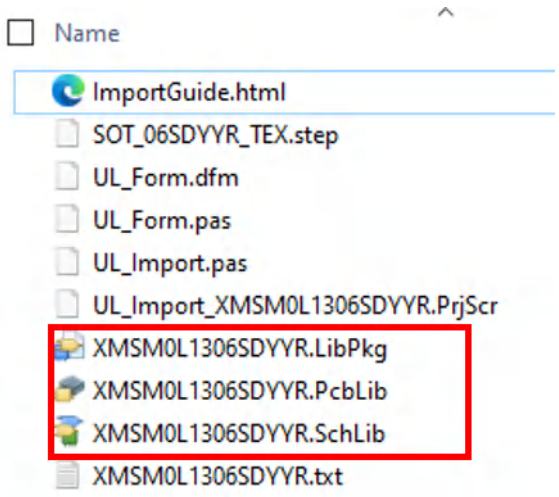


Figure 2-49. PCB Library and Schematic File

The final step is to import them into your AD lib, as shown in Figure 2-50. And based on this, a schematic and PCB can be designed.

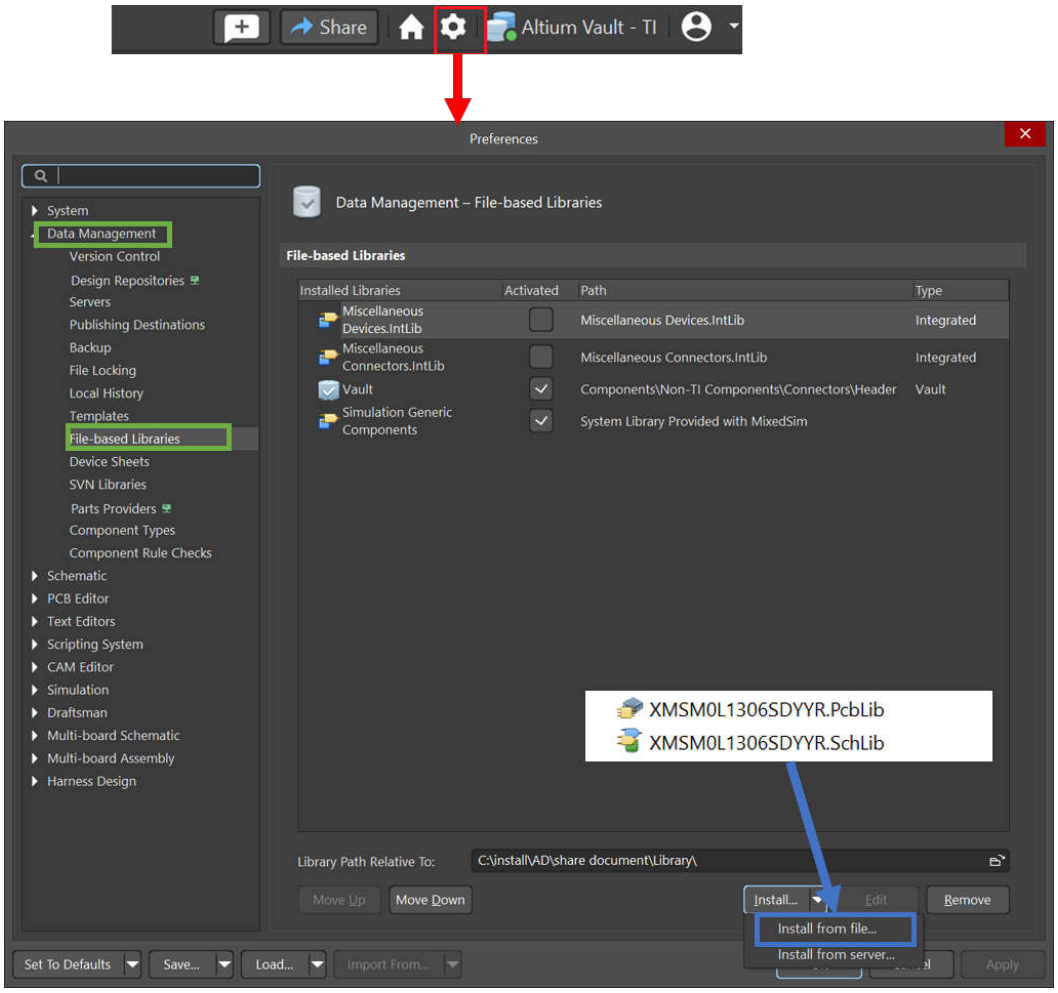


Figure 2-50. Import library

8. Design in MSPM0.
9. Mass production.

3 Core Architecture Comparison

3.1 CPU

The MSPM0 family is based on the ARM Cortex M0+ CPU core architecture. The STM8 family is based on the STM8 CPU core architecture. Table 3-1 gives an overview of the general features of the CPU in the MSPM0 family compared to the STM8.

Table 3-1. Comparison of CPU Feature Sets

Features	STM8L and STM8S	MSPM0C, MSPM0L and MSPM0H
Architecture	Enhanced STM8 CPU core	Arm Cortex M0+
Data bus width	8-bit	32-bit
Instruction set	Complex Instruction Set	Reduced Instruction Set
Number of instructions	80	56
Multiplication instruction	MUL (8 by 8)	MULS (32 by 32)
Division instruction	DIV (16 by 8), DIVW (16 by 16)	MATHACL supports 32-bit division(1)
Pipeline	3-stage	2-stage
Operating Freq (Max)	16MHz or 24MHz(2)	24MHz or 32MHz(3)
DMA	Yes	Yes
Coremark/MHz	unavailable(4)	2.39(5)

(1) MSPM0Gxx Series have math accelerator (MATHACL) which can improve the speed of 32-bit division.
(2) The max operating frequency of STM8Lxx is 16MHz and the max operating frequency of STM8Sxx is 24MHz.
(3) The max operating frequency of MSPM0Cxx is 24MHz and the max operating frequency of MSPM0Lxx is 32MHz.
(4) The coremark of STM8 is unavailable on st.com and eembc.com.
(5) The coremark score is obtained through the *Arm Cortex-m0+ Processor Data sheet* given by ARM official website.

3.2 Embedded Memory Comparison

3.2.1 Flash and EEPROM Features

The MSPM0 and STM8 family of MCUs feature nonvolatile flash and EEPROM memory used for storing executable program code and application data. Table 3-2 shows the features of flash and EEPROM. Note that not all the devices have all features. For details, refer to the device-specific data sheet.

Table 3-2. Features of FLASH and EEPROM

Features	STM8L and STM8S	MSPM0L, MSPM0C and MSPM0H	
Flash memory	STM8Lxx ranges 2 KB to 64 KB STM8Sxx ranges 4 KB to 128 KB	MSPM0Lxx ranges 8 KB to 64 KB MSPM0Cxx 8 KB or 64 KBMSPM0Hxx 32KB to 64KB	
EEPROM	up to 2 KB	EEPROM emulation using Flash	
Memory organization	Block size (64 B/128 B) Page size (a set of blocks)	Word line (128B) Sector size (1 KB) Bank size (variable): Device up to 256 KB-1bank	
Wait states	0 (f_{CPU} < 16MHz) 1 (f_{CPU} > 16MHz)	0 (MCLK, CPUCLK < 24MHz) 1 (MCLK, CPUCLK < 48MHz)	
Single word size	32 bits	64 bits (72 bits with ECC)	
Programming mode	Byte, single flash word, block	Resolution	Single flash word, 32-, 16-, or 8-bit
		Multi-word	2, 4, or 8 words (up to 64 bytes)
Erase	Block erase	Sector erase Bank erase (up to 256 KB)	
Error code correction	Supported	Supported	
Write Protection	Yes	Yes, static and dynamic	
Read Protection	Yes	Yes	

Table 3-2. Features of FLASH and EEPROM (continued)

Features	STM8L and STM8S		MSPM0L, MSPM0C and MSPM0H
Erase and Write Cycles	Program memory	100	100k (lower 32 KB) or 10k (above 32 KB)
	Data memory	100k	

In addition to the flash memory features listed in the previous table, the MSPM0 flash also has the following features:

- In-circuit program and erase supported across the entire supply voltage range.
- Interprogramming voltage generation.

3.2.2 Flash and EEPROM Organization

The Flash and EEPROM memory is further mapped into one or more logical memory regions and assigned system address space for use by the application.

3.2.2.1 Flash and EEPROM Regions

Table 3-3 shows the flash and EEPROM regions of STM8 devices and MSPM0 devices.

Table 3-3. The Flash and EEPROM Regions

STM8L & STM8S		MSPM0L, MSPM0C and MSPM0H	
Main program area	Application code.	**MAIN**	Application code and data.
Opetion bytes	Configuration of device hardware features and memory protection.	**NONMAIN**	BCR configuration.
			BSL configuration.
Proprietary code area(PCODE) (1)	Protecting proprietary software libqraries used to drive peripherals.	**FACTORY**	Device ID, and other parameters.
User boot area (UBC)(2)	The reset and the interrupt vectors.		
Data EEPROM	Application data.	**DATA**(3)	Data or EEPROM emulation.

(1) The PCODE is available only in low, medium+ and high-density devices.
(2) STM8Sx03xx, STM8S001xx, STM8L101xx and STM8L001xx don't have embedded bootloader (no ROM bootloader is implemented inside the microcontroller). When using these devices, the user has to write his own bootloader code and save his own bootloader in the UBC program area.
(3) MSPM0 devices with one bank implement the FACTORY, NONMAIN, and MAIN regions on BANK0 (the only bank present), and the DATA region is not available. MSPM0 devices with multiple banks also implement FACTORY, NONMAIN, and MAIN regions on BANK0, but include additional banks (BANK1 through BANK4) that can implement MAIN or DATA regions.

3.2.2.2 NONMAIN Memory of MSPM0

The NONMAIN flash memory contains the configuration registers used by the BCR and BSL to boot the device, such as the FLASHSWP0 and FLASHSWP1 (static write protection policy). The region is not used for any other purpose. The BCR and BSL both have configuration policies that can be left at the default values (as is typical during development and evaluation) or modified for specific purposes (as is typical during production programming) by altering the values programmed into the NONMAIN flash region.

3.2.3 Embedded SRAM

The MSPM0 and STM8 family of MCUs feature SRAM used for storing application data. Table 3-4 shows the comparison of SRAM features. For details, see the device-specific data sheet.

Table 3-4. Comparison of SRAM Features

Features	STM8L and STM8S	MSPM0L, MSPM0C and MSPM0H
SRAM memory	STM8L Value line: 1 KB to 4 KB STM8L101: 1.5 KB STM8Sxx: 1 KB to 6 KB	MSPM0Lxx: 2 KB to 4 KB MSPM0Cxx: 1 KB or 8KB MSPM0Hxx: 8KB
Parity check	Not supported	MSPM0Lxx: supported(1), MSPM0Cxx: not supported, MSPM0Hxx: not supported
ECC	Not supported	MSPM0Lxx: supported(2), MSPM0Cxx: not supported, MSPM0Hxx: not supported
Write protection (RAM guard)	Not supported	Supported

(1) Only MSPM0Lx22x and MSPM0L111x support.
(2) Only MSPM0Lx22x supports.

MSPM0 MCUs include low-power high-performance SRAM with zero wait state access across the supported CPU frequency range of the device. SRAM can be used for storing information such as the call stack, heap, and global data, in addition code. The SRAM content is fully retained in run, sleep, stop and standby operating modes, but is lost in shutdown mode. A write protection mechanism is provided to allow the application to dynamically write protect the lower 32 KB of SRAM with 1 KB resolution. On devices with less than 32 KB of SRAM, write protection is provided for the entire SRAM. Write protection is useful when placing executable code into SRAM as write protection provides a level of protection against unintentional overwrites of code by either the CPU or DMA. Placing code in SRAM can improve performance of critical loops by enabling zero wait state operation and lower power consumption.

3.3 Power UP and Reset Summary and Comparison

Both STM8 devices and MSPM0 devices have the minimum operating voltage and have modules in place to make sure that the device starts up properly by holding the device or portions of the device in a reset state. Table 3-5 shows a comparison on how this is done between the two families and what modules control the power up process and reset across the families.

Table 3-5. Summary and Comparison of Power Up

STM8		MSPM0	
Power-On Reset (POR)	Rise detection: V_{DD}>V_{POR}, POR state is released, BOR starts to work.	**Power-On Reset (POR)**	Rise detection: V_{DD}>POR+, POR state is released, and bandgap reference and BOR is started Fall detection: V_{DD}<POR-, device is held in POR state
Power-Down Reset (PDR)	Fall detection: V_{DD}<V_{PDR}, PDR keeps the device under reset.		
Brownout Reset (BOR)(1)	Rise detection: V_{DD}>V_{BOR+}, BOR state is released, device continues the boot process. Fall detection: V_{DD}<V_{BOR-}, BOR state is generated. BOR has five different levels to be selected.	**Brownout Reset (BOR)- 0 level**(2)	Rise detection: V_{DD}>BOR0+, Device continues the boot process, and PMU is started Fall detection: V_{DD}<BOR0-, Device is held in BOR state.
		Brownout Reset (BOR)- 1 to 3 level(2)	Fall detection: 1) V_{DD}<BORx- (x=1, 2, 3), an interrupt request is generated, and the BOR circuit automatically switches the BOR threshold level to BOR0. 2) V_{DD}<BOR0-, Device is held in BOR state

Table 3-5. Summary and Comparison of Power Up (continued)

STM8		MSPM0	
Programmable voltage detector (PVD)(3)	Rise detection: V_{DD}>V_{PVD}, a PVD event is generated. Fall detection: V_{DD}<V_{PVD}, a PVD event is generated. PVD has seven different levels to be selected.	**N/A**	N/A
RTC Reset	RTC and associated registers are reset through system reset or power-on reset.	**RTC Reset**	RTC and associated clocks are reset through BOOTRST, BOR, or POR

(1) During startup, the BOR need be configured to BOR Level 0 to make sure the minimum operating voltage. BOR is always active at power-on, keeping the MCU under reset till the application operating threshold is reached. If the BOR is disabled at power-down, the reset threshold is V_{PDR} to make sure a VDD min.
(2) There are four selectable BOR threshold levels (BOR0-BOR3). During startup, the BOR threshold is always BOR0 (the lowest value) to make the device always starts at the specified V_{DD} minimum. After boot, software can optionally re-configure the BOR circuit to use a different (higher) threshold level.
(3) STM8L001xx, STM8L101xx and STM8S series microcontrollers does not have the PVD.

Figure 3-1 shows the MSPM0 Reset function. MSPM0 devices have five reset levels: Power-on reset (POR), Brownout reset (BOR), Boot reset (BOOTRST), System reset (SYSRST) and CPU reset (CPURST).

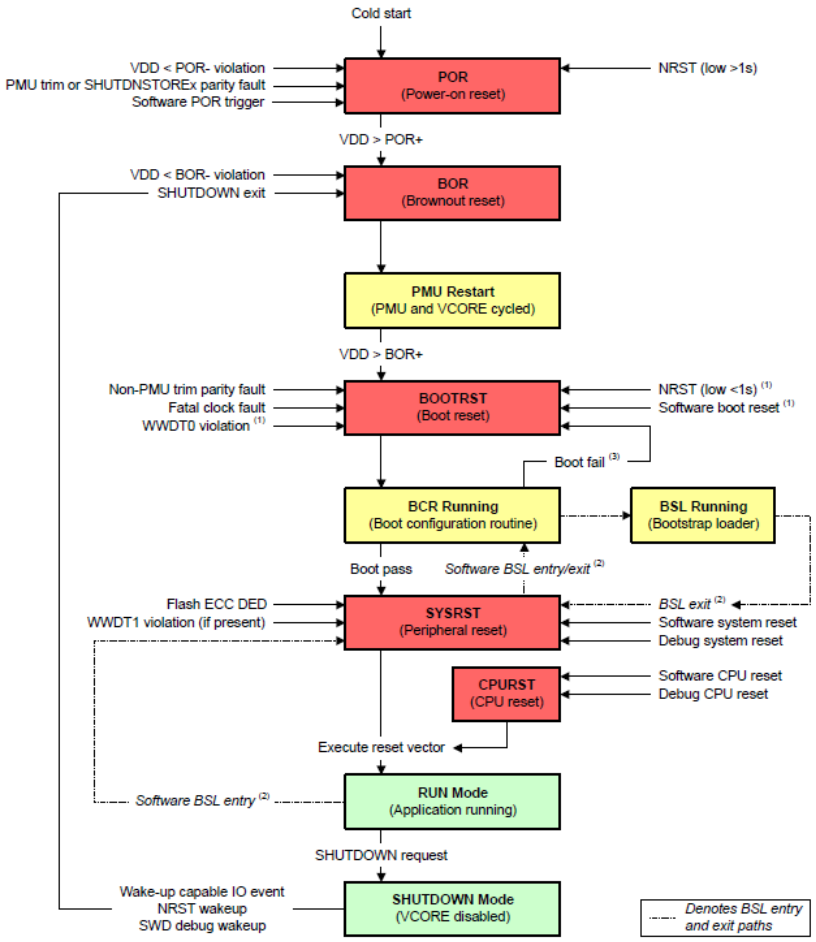


Figure 3-1. MSP Reset Function

3.4 Clocks Summary and Comparison

3.4.1 Oscillators

STM8 and MSPM0 devices have many types of clock sources including both internal and external for low system cost and low power consumption. Table 3-6 lists the different clock sources in STM8 and MSPM0 devices. Note that not all the devices have all types clock sources. For details, refer to the device-specific data sheet.

Table 3-6. Oscillator Comparison

Type		STM8L	STM8S	MSPM0L	MSPM0C	MSPM0H
Internal oscillators	**High Speed**	HSI: internal 16MHz RC oscillator	HSI: internal 16MHz RC oscillator	SYSOSC: internal oscillator from 4MHz to 32MHz	SYSOSC : internal 24/32 MHz oscillator	SYSOSC : internal 32MHz oscillator
	Low Speed	LSI: internal 38kHz RC oscillator	LSI: internal 128kHz RC oscillator	LFOSC: internal 32kHz oscillator	LFOSC: internal 32kHz oscillator	LFOSC: internal 32kHz oscillator
External oscillators	**High Speed**	HSE: 1-16 MHz external crystal	HSE: 1-24 MHz external crystal	4MHz to 32MHz(1)	4MHz to 32MHz(1)	Not available
	Low Speed	LSE: 32.768kHz external crystal	Not available	32kHz(1)	32kHz(1)	Not available

(1) Only MSPM0Lx22x, MSPM0C1105 and MSPM0C11056 supported.

3.4.2 Clock Signal Comparison

Different clock signals can be divided to source other clocks and be distributed across the multitude of peripherals.

Table 3-7. Clock Signal Comparison

Clock Description		STM8L Clock	STM8S Clock	MSPM0L, MSPM0C and MSPM0H Clock
External digital clock input	**High frequency**	External source: up to 16MHz(1)	HSE Ext: up to 24MHz(1)	Lx22x and C1106: supports 32MHz
	Low frequency	External source: 32.768kHz(1)	N/A	Lx22x and C1106: supports 32KHz
High-frequency sources for main clock		HSI, HSE	f_{HSE}, f_{HSIDIV}	SYSOSC
Low-frequency sources for main clock		LSI, LSE	f_{LSI}	LFCLK (fixed 32kHz)
Main system clock		SYSCLK, f_{MASTER}	f_{MASTER}	MCLK, ULPCLK (BUSCLK)(2)
Source CPU		SYSCLK, f_{MASTER}	f_{CPU}	CPUCLK
Clock for most peripheral hardware		PCLK (SYSCLK), f_{MASTER}	f_{MASTER}	MCLK, ULPCLK(2)
Peripheral specific clock		BEEPCLK, IWDGCLK, RTCCLK, f_{LSI}, $f_{HSI/2}$ (3)	N/A	ADCCLK
Fixed frequency clock		N/A	N/A	MFCLK: 4MHz, synchronized to MCLK, ULPCLK

(1) HSE crystal and LSE crystal need be switched off when external clock sources are used. STM8L001xx and STM8L101xx families don't support external digital clock input.
(2) The MCLK is the main system clock for PD1 and the ULPCLK, derived from MCLK, is the main system clock for PD0. PD1 (power domain 1) contains the CPU subsystem, memory interfaces, and high-speed peripherals. PD0 (power domain 0) contains the low-speed low-power peripherals.
(3) The f_{LSI} of STM8L001xx and STM8L101xx families is just used to source AWU, BEEP, SWIM, IWDG. The $f_{HSI/2}$ is just used to source SWIM.

Table 3-8. Peripheral Clock Sources

Peripheral	STM8L, STM8S	MSPM0L, MSPM0C, MSPM0H
UART/USART	SYSCLK, f_{MASTER}	SYSCLK, MFCLK, LFCLK
SPI	SYSCLK, f_{MASTER}	SYSCLK, MFCLK, LFCLK
I2C	SYSCLK, f_{MASTER}	BUSCLK, MFCLK
ADC	PCLK or PCLK/2(1), f_{ADC} (f_{MASTER} divided by 2 to 18)	ADCCLK (sourced by ULPCLK or SYSOSC)
TIMERS	SYSCLK, f_{MASTER}, $f_{MASTER/DIV}$	BUSCLK, MFCLK, LFCLK
COMPARATOR	PCLK, f_{MASTER} (2)	BUSCLK
WATCHDOG	LSI, SYSCLK, f_{CPU} (3)	LFCLK

(1) STM8L001xx and STM8L101xx microcontroller families do not have ADC.
(2) STM8S series microcontrollers do not have COMPARATOR.
(3) LSI is used to source Independent watchdog (IWDG). SYSCLK or f_{CPU} are used to source window watchdog (WWDG).

3.5 MSPM0 Operating Modes Summary and Comparison

MSPM0 MCUs provide five main operating modes (power modes) to allow for optimization of the device power consumption based on application requirements. In order of decreasing power, the modes are: RUN, SLEEP, STOP, STANDY, and SHUTDOWN. The CPU is active executing code in RUN mode. Peripheral interrupt events can wake the device from SLEEP, STOP, or STANDBY mode to the RUN mode. SHUTDOWN mode completely disables the internal core regulator to minimize power consumption, and wake is only possible via NRST, SWD, or a logic level match on certain IOs. RUN, SLEEP, STOP, and STANDBY modes also include several configurable policy options (for example, RUN.x) for balancing performance with power consumption.

To further balance performance and power consumption, MSPM0 devices implement two power domains: PD1 (for the CPU, memories, and high-performance peripherals), and PD0 (for low speed, low power peripherals). PD1 is always powered in RUN and SLEEP modes, but is disabled in all other modes. PD0 is always powered in RUN, SLEEP, STOP, and STANDBY modes. PD1 and PD0 are both disabled in SHUTDOWN mode.

3.5.1 Operating Modes Comparison

Table 3-9 gives a brief comparison between STM8 and MSPM0 devices.

Table 3-9. Operating Modes Comparison Between STM8 and MSPM0 Devices

STM8		MSPM0		
Operation Mode	**Description**	**Operation Mode**		**Description**
Run mode	CPU and peripherals run normally after a system or power reset.	**RUN**	0	MCLK and CPUCLK run from a fast clock source (SYSOSC)
Low power run mode	CPU and peripherals run with a low speed oscillator (LSI or LSE). All interrupts must be masked.		1	MCLK and CPUCLK run from LFCLK (at 32kHz).
			2	
Wait mode	CPU operation stops. Oscillator remains enable. Selected peripherals keep running. Wait mode is entered from Run mode by executing a WFI or WFE instruction.	**SLEEP**	0	CPU operation stops. SYSOSC remains enable. LFOSC remains enable. MCLK run from a fast clock source (SYSOSC).
			1	CPU operation stops. SYSOSC remains enable. LFOSC remains enable. MCLK run from LFCLK.
Low power wait mode	CPU operation stops. Low speed oscillator remains enable. Selected peripherals keep running. This mode is entered when executing a Wait for event in low power run mode. All interrupts must be masked.		2	CPU operation stops. SYSOSC remains disable. LFOSC remains enable. MCLK run from LFCLK.

Table 3-9. Operating Modes Comparison Between STM8 and MSPM0 Devices (continued)

STM8		MSPM0		
Operation Mode	**Description**	**Operation Mode**		**Description**
Active-halt mode(STM8S)	CPU operation stops. Oscillators are disable except LSI or HSE. Almost all the peripherals are stopped except AWU. the MVR regulator is powered on.	**STOP**[2]	0	CPU operation stops. Status of SYSOSC is retained[1]. LFOSC remains enable. ULPCLK is limited to 4MHz. PD0 is enabled and PD1 is disabled. And analog peripherals such as ADC can operate.
			1	Same as STOP0, with the SYSOSC and ULPCLK gear shifted to 4MHz.
Active-halt mode with MVR auto power off(STM8S)	CPU operation stops. Oscillators are disable except LSI only. Almost all the peripherals are stopped except AWU. the MVR regulator is powered off.		2	CPU operation stops. SYSOSC is disabled. ULPCLK runs at 32kHz. PD0 is enabled and PD1 is disabled.
Active-halt mode (not STM8S families)	CPU operation stops. Oscillators are disable except LSI or LSE. Almost all the peripherals are stopped except RTC, AWU, and so forth. The voltage regulator is at low power mode.	**STANDBY**	0	CPU operation stops. SYSOSC is disable. All PD0 peripherals receive the ULPCLK and LFCLK.
Halt mode	CPU operation stops. Oscillators are disable[3]. Almost all the peripherals are stopped. The voltage regulator is at low power mode[3].		1	Similar to STANDBY0, with only TIMG0/1 receiving ULPCLK or LFCLK.
N/A	N/A	**SHUTDOWN**	No clocks are available and device is shut down.	

(1) If STOP0 is entered from RUN1 (SYSOSC enabled but MCLK sourced from LFCLK), then SYSOSC remains enabled as in RUN1. If STOP0 is entered from RUN2 (SYSOSC disabled and MCLK sourced from LFCLK), then SYSOSC remains disabled as in RUN2.
(2) MSPM0C devices don't have the STOP1 mode.
(3) LSI oscillators of STM8L001xx and STM8L101xx devices are enabled on halt mode if IWDG is activated and *no watchdog in Halt* option is disabled. Only BEEP and IWDG keep running on halt mode if activated and *no watchdog in Halt* option disabled.

STM8L05xx devices have five low power modes: Wait mode, Low power run mode, Low power wait mode, Active-halt mode and Halt mode. STM8L001xx and STM8L101xx devices have three low power modes: Wait mode, Active-halt mode and Halt mode. STM8 series have four low power modes: Wait mode, Active-halt mode, Active-halt with MVR auto power off, and Halt mode.

3.5.2 MSPM0 Capabilities in Lower Modes

MSPM0 peripherals or peripheral modes can be limited in availability or operating speed in lower power operating modes. For specific details, refer to the *Supported Functionality by Operating Mode* table found in the MSPM0 device-specific data sheet. Two relevant data sheets are listed as example below:

- *MSPM0L134x, MSPM0L130x Mixed-Signal Microcontrollers Data Sheet*
- *MSPM0C110x, MSPS003 Mixed-Signal Microcontrollers Data Sheet*

An additional capability of the MSPM0 devices is the ability for some peripherals to perform an asynchronous fast clock request. This allows MSPM0 device to be in a lower power mode where a peripheral is not active, but still allow a peripheral to be triggered or activated. When an asynchronous fast clock request happens, the MSPM0 device has the ability to quickly ramp up an internal oscillator to a higher speed or temporarily go into a higher operating mode to process the impending action. This allows for fast wake up of the CPU from timers, comparator, GPIO; receive SPI, UART, and I2C; or trigger DMA transfers and ADC conversions, while sleeping in the lowest power modes. For specific details on implementation of asynchronous clock requests and peripheral support and purpose, refer to the corresponding chapter in the MSPM0 device-specific TRMs.

- *MSPM0 L-Series 32MHz Microcontrollers Technical Reference Manual*
- *MSPM0 C-Series 24MHz Microcontrollers Technical Reference Manual*

3.5.3 Entering Lower-Power Modes

The MSPM0 devices go into a lower-power mode when executing the wait for event, **_WFE( )**, or wait for interrupt, **_WFI( )**, instruction. The low-power mode is determined by the current power policy settings. The device power policy is set by a driver library function. The following function call sets that power policy to Standby 0.

DL_SYSCTL_setPowerPolicySTANDBY0 ( );

STANDBY0 can be replaced with the operating mode of choice. For a full list of driverlib APIs that govern power policy, see this section of the MSPM0 SDK DriverLib API guide. Also, see the following code examples that demonstrate entering different operating modes. Similar examples are available for every MSPM0 device.

3.5.4 Low-Power Mode Code Examples

Navigate to the SDK installation and find low-power mode code examples in examples > nortos > LP name > driverlib.

3.6 Interrupts and Events Comparison

3.6.1 Interrupts and Exceptions

The MSPM0 and STM8 both register and map interrupt and exception vectors depending on the device's available peripherals. A summary and comparison of the interrupt vectors for each family of devices is included in Table 3-10.

Table 3-10. Interrupts Comparison

Features	STM8L and STM8S	MSPM0L, MSPM0C and MSPM0H
Interrupt Types	Peripheral interrupts: determined by the particular devices	Peripheral interrupts: NVIC of MSPM0L supports up to 19 peripheral interrupt vectorsNVIC of MSPM0C supports up to 23 peripheral interrupt vectorsMSPM0H supports up to 22 peripheral interrupt vectors(2)
	External interrupts: STM8L value line has 11 vectorsSTM8L101x has 10 vectorsSTM8S has 5 vectors(1)	
	Non-maskable interrupts: RESET, TRAP (software interrupt), TLI (top level hardware interrupt)(3)	Reset, Hard Fault, SVCall, PendSV, SysTick
		NMI: software trigger, hardware error signal from SYSCTL
Priority Level	The hardware priority level: IRQ number of interrupt mapping	The default priority level: NVIC Number(4)
	Non-maskable interrupts are considered as having the highest software priority	System exceptions (Reset, NMI, Hard Fault) have fixed priority levels of -3, -2, and -1
	The maskable interrupts have 4 software priority levels: 0 (main), 1, 2, and 3 (software priority disabled)	The peripheral interrupts have 4 programmable priority levels: 0, 64, 128, 192
Priority Set	ITC_SPRx register: used to define the software priority of each interrupt vector(5) CCR register: used to load the software priority of the current interrupt request automatically(6)	IPRx registers in the NVIC: used to set the peripheral interrupt priority level
Interrupt mask	The corresponding interrupt enable bit is set in the peripheral control register	IMASK register in the peripheral side: used to configure which interrupt conditions propagate into an event(7)
		ISER and ICER register in the NVIC: used to enable or disable the peripheral interrupts

(1) To generate an external interrupt, the corresponding GPIO port must be configured in input mode with interrupts enable.
(2) In addition to the NVIC, interrupt grouping modules (INT_GROUP0 and INT_GROUP1) can be present on a MSPM0 device to enable interfacing of more peripheral interrupts to the NVIC. And the external interrupts / GPIO interrupts are in INT_GROUP1 module.
(3) Only the STM8S devices support the top level hardware interrupt (TLI).
(4) The NVIC number indicates the relative interrupt priority if multiple NVIC interrupts have the same programmable priority.
(5) Writing 10 (priority level 0) to VECTxSPR[1:0] is forbidden. If 10 is written, the previous value is kept and the interrupt priority remains unchanged.
(6) Non-maskable interrupt sources are processed regardless of the state of bits I1 and I0 of the CCR register.
(7) The event handler and related management registers of MSPM0 are shown in Event Handler of MSPM0.

For MSPM0 devices, a lower value of priority for an interrupt or exception is given higher precedence over interrupts with a higher priority value. When the processor is currently handling an interrupt, the processor can

only be preempted by an interrupt with high priority. For STM8 devices, a higher value of priority for an interrupt or exception is given higher precedence over interrupts with a lower priority value. And STM8 devices feature two interrupt management modes: concurrent mode and nested mode. For details, see the device-specific data sheet.

3.6.1.1 Interrupt Management of MSPM0

MSPM0 devices set the priority level of each peripheral interrupt source through the IPRx registers in the NVIC, and mask/unmask a peripheral interrupt source through ISER and ICER register in the NVIC. Each peripheral interrupt contains kinds of interrupt conditions. For example, as a peripheral interrupt source, UARTx has multiple interrupt conditions such as transmit interrupt and receive interrupt, and so forth. And the interrupt conditions are managed by six standard registers in the peripheral side. Figure 3-2 shows the peripheral interrupt hierarchy.

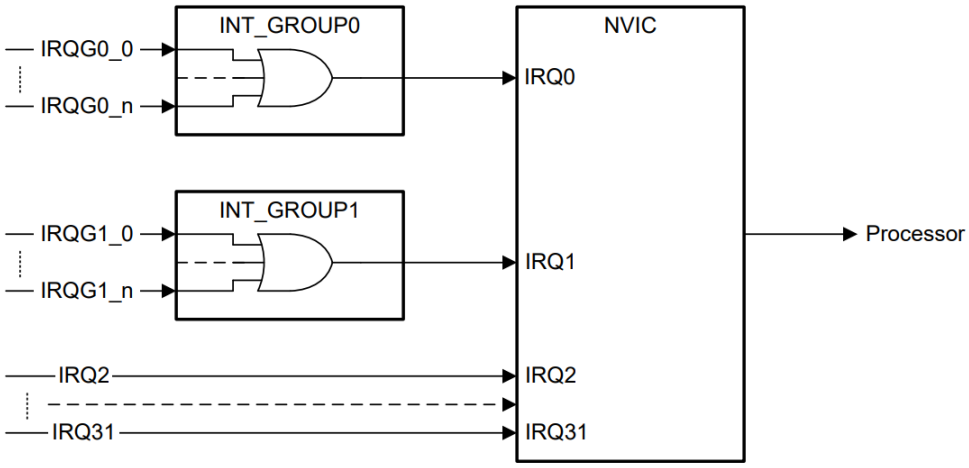


Figure 3-2. Peripheral Interrupt Hierarchy of MSPM0

3.6.1.2 Interrupt Controller (ITC) of STM8

Figure 3-3 shows the interrupt processing flowchart of STM8. If the interrupt mask bits I0 and I1 are set within an interrupt service routine (ISR) with the instruction SIM, removal of the interrupt mask with RIM causes the software priority to be set to level 0.The interrupt service routine need to end with the IRET instruction which causes the content of the saved registers to be recovered from the stack. As a consequence of the IRET instruction, bit I1 and I0 are restored from the stack and the program execution resumes.

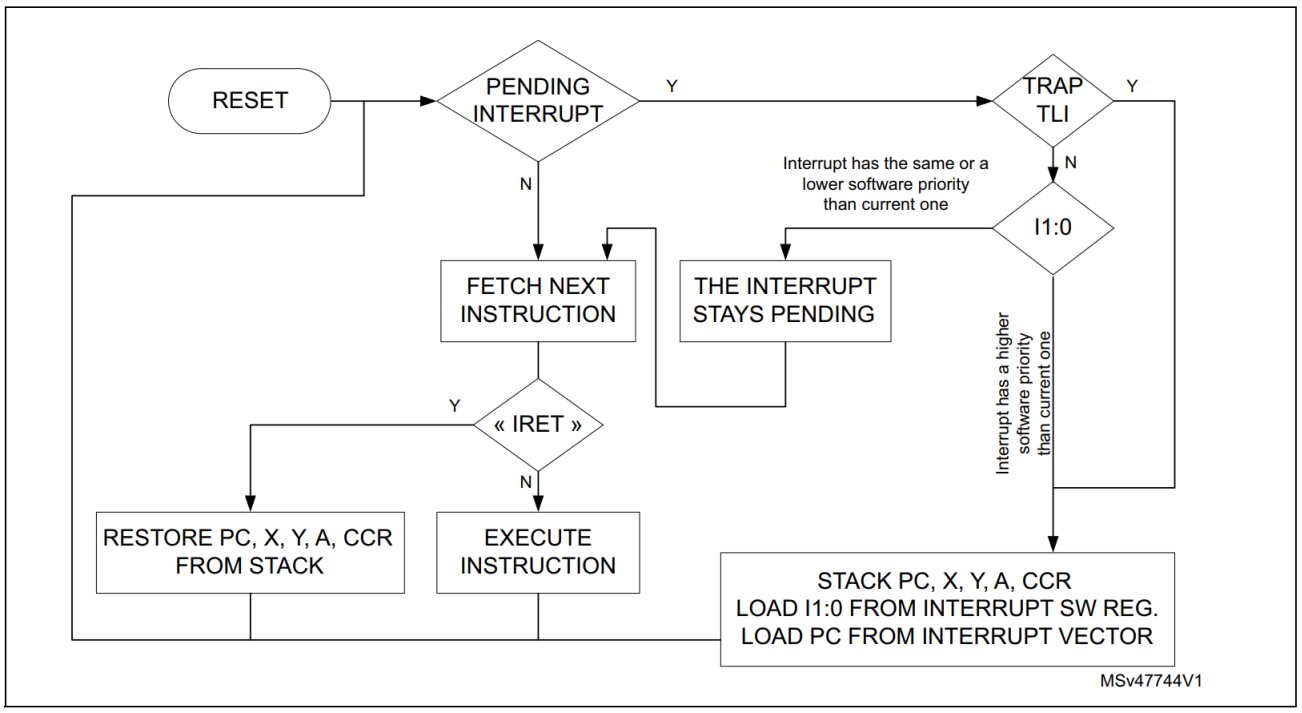


Figure 3-3. Interrupt Processing Flowchart

3.6.2 Event Handler of MSPM0

MSPM0 MCUs have an event manager that transfers digital events from one entity to another. The event manager implements event transfer through a defined set of event publishers (generators) and subscribers (receivers) that are interconnected through an event fabric containing a combination of static and programmable routes. The event manger can also perform handshaking with the power management and clock unit (PMCU), to make sure that the necessary clock and power domain are present for triggered event actions to take place.

Events that are transferred by the event manager include:

- Peripheral event transferred to the CPU as an interrupt request (IRQ)
- Peripheral event transferred to the DMA as a DMA trigger
- Peripheral event transferred to another peripheral to directly trigger an action in hardware

The event manager connects event publishers to event subscribers through an event fabric. There are three types of event fabric: CPU interrupt (fixed event route), DMA route, and generic route. For example, Figure 3-4 shows the generic route.

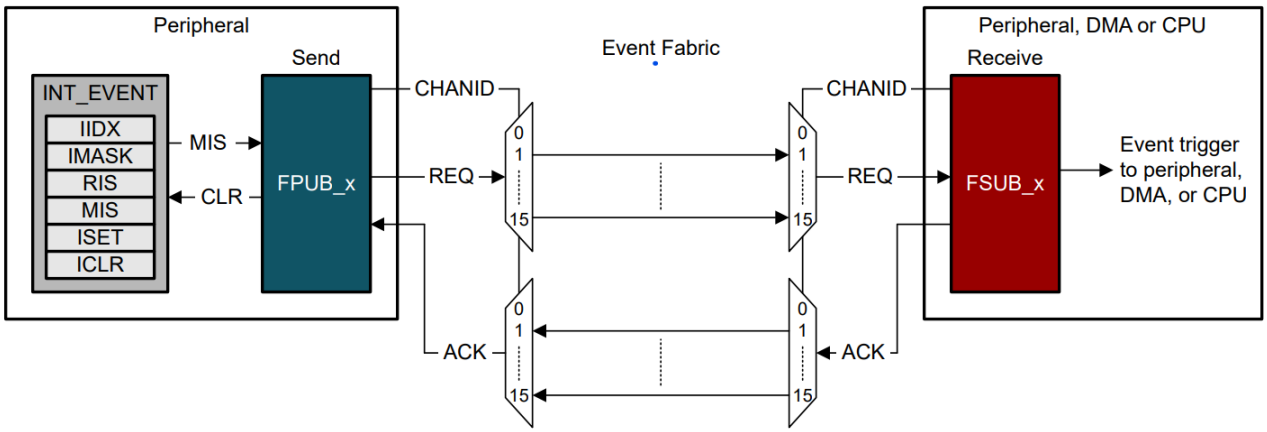


Figure 3-4. Generic Event Route

The event management register set contains six standard registers: RIS, IMASK, MIS, ISET, ICLR, and IIDX. And the event registers are interconnected as shown in Figure 3-5. Once unmasked, a pending interrupt is indicated in both the RIS and MIS registers, and an event is generated. In the case of a CPU interrupt with a CPU interrupt event route, a read of the IIDX register clears the highest priority pending interrupt in the RIS and MIS registers and return the index of the highest priority pending interrupt to application software.

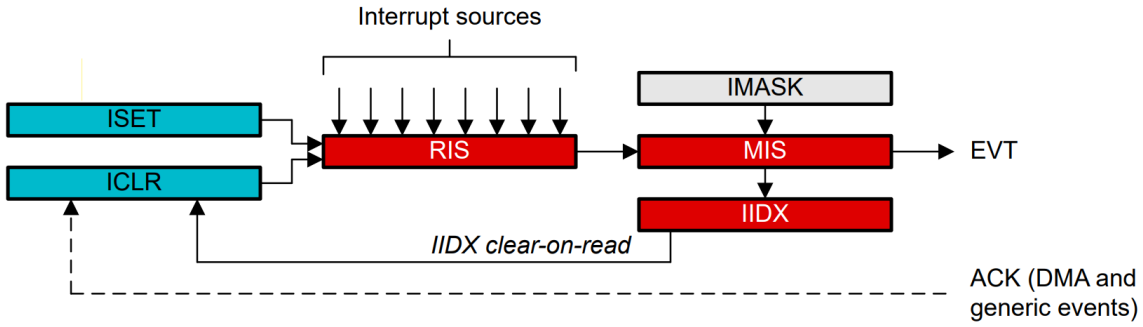


Figure 3-5. Event Management Register Relationship

Figure 3-6 shows the event map. Different peripherals are routed through different event fabrics to achieve different event transitions. For more details on the use of the event handler in MSPM0, see the *Event* section of the MSPM0 L-Series 32MHz Microcontrollers Technical Reference Manual technical reference manual, or the MSPM0 C-Series 24MHz Microcontrollers technical reference manual.

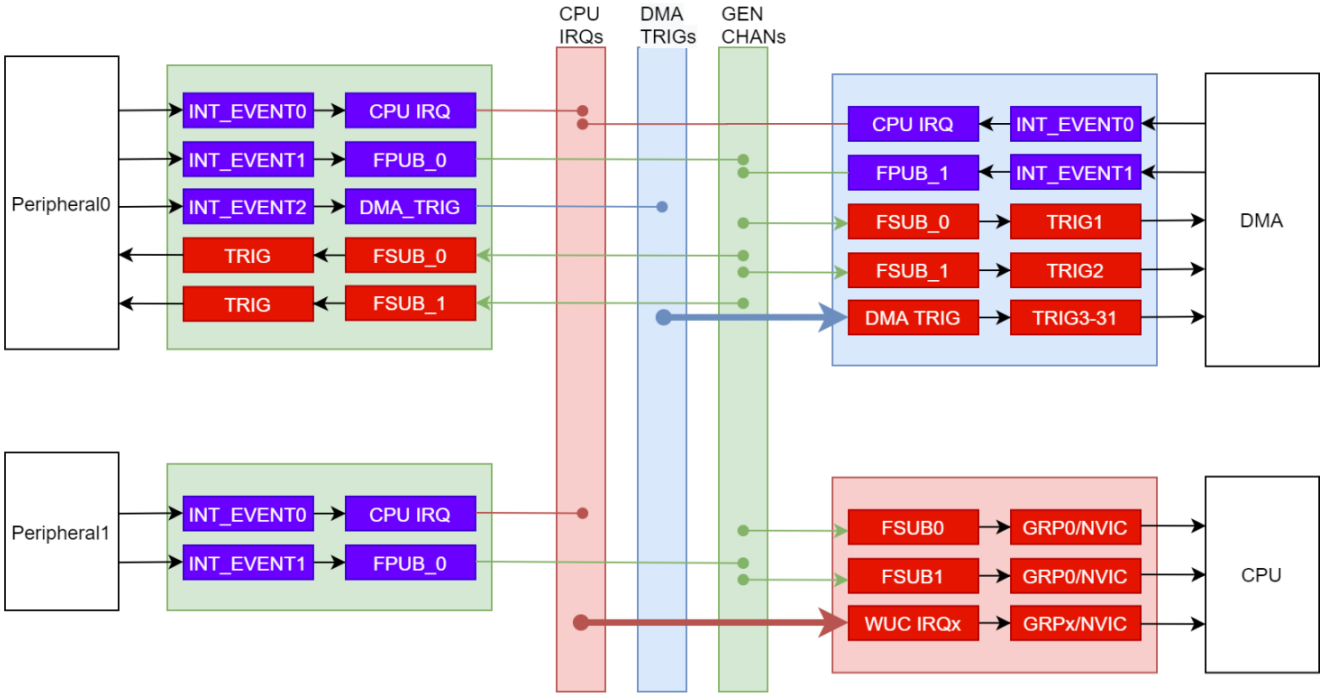


Figure 3-6. MSPM0 Event and Interrupt Handling

3.6.3 Event Management Comparison

MSPM0 and STM8 have different event management structures and features. MSPM0 MCUs use Event handler to manage different events. And STM8 MCUs use different controllers to manage different events. The comparison of Event Handler of MSPM0 and event management of STM8 is shown in Table 3-11.

Table 3-11. Event Management Comparison

Features		STM8L and STM8S	MSPM0L, MSPM0C and MSPM0H
Publisher		Peripheral	Peripheral
Subscriber		CPU, DMA trigger, peripheral	CPU, DMA trigger, peripheral
Management Style	**peripheral → CPU**	Interrupt controller (ITC)	Event manager
	peripheral → DMA	DMA controller(1)	
	peripheral → peripheral	Trigger Controller(2)	
Route Type		Point-to-point (1:1)	Point-to-point
			Point-to-two (splitter)(3)

(1) Not all STM8 devices have the DMA controller. STM8L001xx family, STM8L101xx family and STM8S series don't support the DMA function.
(2) The trigger controller of TIMx can generate trigger output signal (TRGO) to trigger other TIM timers, ADC and DAC.
(3) The generic route channels can be configured with one subscriber (1:1) or two subscribers (1:2 splitter route), depending on which channel is selected.

3.7 Debug and Programming Comparison

3.7.1 Debug Mode Comparison

The Arm SWD 2-wire JTAG port is the main debug and programming interface for MSPM0 devices. This interface is typically used during application development, and during production programming.

Unlike MSPM0 devices, STM8 devices do not have SWD 2-wire JTAG port. The single wire hardware interface (SWIM) featuring ultrafast memory programming is the main debug and programming interface for STM8 devices. SWIM pin can be used as a standard I/O with some restrictions if you also want to use the SWIM pin for debug. The most secure way is to provide on the PCB a strap option.

3.7.2 Programming Mode Comparison

The bootstrap loader (BSL) programming interface is an alternative programming interface to the ARM SWD and STM8 SWIM. This interface offers programming capabilities only, and typically is utilized through a standard embedded communication interface. This allows for firmware updates through existing connections to other embedded devices in system or external ports. Although programming updates is the main purpose of this interface, BSL can also be utilized for initial production programming as well.

3.7.2.1 Bootstrap Loader (BSL) Programming Options

Both MSPM0 and STM8 devices support the BSL programming interface. Table 3-12 shows a comparison of the different options and features between MSPM0 and STM8 device families.

Table 3-12. BSL Feature Comparison

BSL Features	STM8L and STM8S	MSPM0L	MSPM0C	MSPM0H
Embedded BSL code storage	ROM (1)	ROM (2)	Not supported	Not supported
Customizable	No	Configurable invoke pin, COMM pin and plug-in feature	No	No
Secondary BSL code storage	UBC program area (1)	Main Flash (2)	Main Flash	Main Flash
BSL started on blank device	Yes	Yes	N/A	N/A
Auto detection of programming interface	Yes	Yes	N/A	N/A
Security	Readout protection (ROP); Commands checksum	Secure boot options; CRC protections AES256 with key store; TRNG	CRC; Firewall; IP protection	CRC; Firewall; IP protection
Invoke methods	Check whether BSL option bytes are 0x55AA or whether program memory is virgin	Invoke pin high at BOOTRST; SW entry	Invoke pin high at BOOTRST; SW entry	Invoke pin high at BOOTRST; SW entry
Interfaces Supported				
UART	Yes	Yes	Secondary BSL	Secondary BSL
I2C	Not supported	Yes	Secondary BSL	Secondary BSL
SPI	Yes	Custom plug-in needed	Secondary BSL	Secondary BSL
CAN	Yes (3)	Plug-in planned (4)	Secondary BSL	Secondary BSL

(1) STM8Sx03xx, STM8S001xx, STM8L101xx and STM8L001xx devices don't have the embedded BSL (no ROM BSL is implemented inside the microcontroller). When using these devices, the user has to write his own BSL code and save his own BSL code in the UBC program area.
(2) MSPM0C devices don't have the ROM BSL code. When using these devices, the user has to write his own BSL code including customized plugin interface and BSL core and save his own code in the main Flash.
(3) The CAN peripheral of STM8 devices can only be used if an external clock (8MHz, 16MHz, or 24MHz) is present.
(4) Only on selected devices.

4 Digital Peripheral Comparison

4.1 General-Purpose I/O (GPIO, IOMUX)

MSPM0 GPIO functionality covers almost all the features provided by STM8S and STML series. STM8 uses the term pin functions and port function to refer to all the functionality responsible for managing the device pins, generating interrupts, and so forth. Here is the description of MSPM0 GPIO and IOMUX function:

- MSPM0 GPIO refers to the hardware capable of reading and writing IO, generating interrupts, and so forth.
- MSPM0 IOMUX refers to the hardware responsible for connecting different internal digital peripherals to a pin. IOMUX services many different digital peripherals including, but not limited to, GPIO.

Together MSPM0 GPIO and IOMUX cover the same functionality as STM8 GPIO. Additionally, MSPM0 offers functionality not available in STM8L and STM8S series devices such as DMA connectivity, controllable input filtering and event capabilities.

Table 4-1. GPIO Feature Comparison

Feature	STM8S and STM8L	MSPM0L, MSPM0L, MSPM0H
Output modes	Push-pull Open drain	Push-pull Open drain with pulldown Hi-Z
Input modes	Pull-up Floating Analog	Floating Pullup or pulldown Analog
GPIO speed selection	Speed selection for each I/O Speed0 up to 2MHz Speed up to 10MHz	MSPM0 offers Standard IO (SDIO) on all IO pins. MSPM0 High-Speed IO (HSIO) is available on select pins. SDIO and HSIO all up to 32MHz at VDD ≥ 2.7V, and HSIO up to 24MHz at VDD ≥ 1.71V
Atomic bit set and reset	Yes	Yes
Alternate functions	Use ODR, IDR and DDR register	Use IOMUX
Fast toggle	At least every two clocks	Toggle pins every clock cycle
Wake-up	External interrupts	GPIO pin state change
GPIO controlled by DMA	No	*Only available on MSPM0*
User controlled input filtering to reject glitches less than 1, 3, or 8 ULPCLK periods	No	*Only available on MSPM0*
User controllable input hysteresis	No	*Only available on MSPM0*

GPIO Code Examples: information about GPIO code examples can be found in the MSPM0 SDK examples guide.

4.2 Universal Asynchronous Receiver-Transmitter (UART)

STM8S series and MSPM0 both offer peripherals to perform asynchronous (clockless) communication. while STM8L series has the USART (universal synchronous asynchronous receiver transmitter) to offer a flexible means of full-duplex data exchange with external equipment.

Table 4-2. UART Standard Feature Comparison

Feature	STM8S and STML	MSPM0L, MSPM0L, MSPM0H
Data direction	LSB-first	MSB-fiirst or LSB-first
Hardware flow control	flow	Yes
Configurable stop bits	1, 2	1, 2
Tx, Rx FIFO depth	2	4
Multiprocessor	Yes	Yes
Active in low-power mode	Yes (only in wait mode)	Yes (all low-power)
Single-wire half duplex communication	Yes	Yes(1)
Wakeup from low-power mode	Yes	Yes
Parity	Even, odd	Even, odd
Communication using DMA	Yes(2)	Yes

(1) Requires reconfiguration of the peripheral between transmission and reception.
(2) Only STM8L Value Line has the DMA.

Table 4-3. UART Advanced Feature Comparison

Feature	STMS and STML	MSPM0L, MSPM0L, MSPM0H
Synchronous mode	Yes(1)	No
Data length	8, 9	5, 6, 7, 8
Oversampling	No	16x 8x 3x
LIN HW support	Yes(2)	Yes
DALI HW support	No	Yes
IrDA HW support	Yes	Yes
Manchester code HW support	No	Yes
Smart card mode (ISO7816)	Yes	Yes
External driver enable	No	Yes
Glitch filter	No	Yes

(1) For STM8S, only UART1, UART2 and UART4 have transmitter clock output for synchronous communication. STM8L has the UASART module to communicate synchronously.
(2) The hardware module of STM8 LIN module has more comprehensive functions than MSPM0, such as: LIN break and delimiter generation, header errors detection, and so forth.

UART Code Examples: information about UART code examples can be found in the MSPM0 SDK examples guide.

4.3 Serial Peripheral Interface (SPI)

MSPM0 and STM8 both support serial peripheral interface (SPI). Overall, MSPM0 and STM8 SPI support is comparable with the difference listed in Table 4-4.

Table 4-4. SPI Feature Comparison

Feature	STM8S and STM8L	MSPM0L, MSPM0L and MSPM0H
Operation wires	SCK, MOSI, MISO, NSS	SCLK, PICO, POCI, CSx
Controller or peripheral operation	Yes	Yes
Multiple controller capability	Yes	No
Data order	MSB-first or LSB-first	MSB-first or LSB-first
Data bit width (controller mode)	Not mentioned	4 to 16 bit
Data bit width (peripheral mode)		7 to 16 bit
Maximum speed	10MHz	16MHz
Simplex transfers (unidirectional data line)	Yes	Yes
Hardware chip select management	Yes (1 peripheral)	Yes (4 peripherals)
Phase control of I/O clock	Yes	Yes
SPI format support	Motorola	Motorola, TI, MICROWIRE
Hardware CRC	Yes (STM8S)	No, MSPM0 offers SPI parity mode
Low power mode	Wait mode	Sleep mode
TX FIFO depth	1 (buffer)	4
RX FIFO depth	1(buffer)	4

SPI Code Examples: information about SPI code examples can be found in the MSPM0 SDK examples guide.

4.4 Interintegrated Circuit Interface (I2C)

MSPM0 and STM8 both support I2C peripherals. Overall MSPM0 and STM8 I2C support is comparable with notable difference outlined in Table 4-5.

Table 4-5. I2C Feature Comparison

Feature	STM8S and STM8L	MSPM0L, MSPM0C and MSPM0H
Controller and target modes	Yes	Yes
Multi-controller capability	Yes	Yes
Maximum transfer rate	400 kHz(Fast speed)	1Mbps (Fast-mode Plus)
Addressing mode	7 bit or 10 bit	7 bit
Address number (Target mode)	2 addresses	2 addresses
General call	Yes	Yes
Event management	Yes	Yes
PEC management	Yes	Yes
Clock stretching	Yes	Yes
Wakeup function (low-power mode)	Yes	Yes
Software reset	Yes	Yes
FIFO, buffer	1 byte	TX: 8 byte
		RX: 8 byte
DMA	Yes	Yes
Programmable analog and digital noise filters	Yes	Yes

I2C Code Examples: information about I2C code examples can be found in the MSPM0 SDK examples guide.

4.5 Timers (TIMGx, TIMAx)

STM8 and MSPM0 both offer various timers. MSPM0 offers timers with varying features that support use cases from low power monitoring to advanced motor control.

Table 4-6. Timer Naming

STM8L		STM8S		MSPM0L		MSPM0C		MSPM0H	
Timer Name	Abbreviated Name	Timer Name	Abbreviated Name	Timer Name	Abbreviated Name	Timer Name	Abbreviated Name	Timer Name	Abbreviated Name
		Advanced control	TIM1	Advanced control	TIMA0(Only in MSPM0Lx22x)	Advanced control	TIMA0	Advanced control	TIMA0
General purpose	TIM2,3	General purpose	TIM2,3,5	General purpose	TIMG0,1,2,4,5,8,12	General purpose	TIMG1,2,8,14	General purpose	TIMG1,2,8,14
Basic	TIM4	Basic	TIM4,6						

Table 4-7. Timer Feature Comparison

Feature	STM8L	STM8S	MSPM0L, MSPM0C and MSPM0H
Resolution	8, 16 bit	8, 16 bit	16 bit
PWM	Yes	Yes	Yes
Capture	Yes	Yes	Yes
Compare	Yes	Yes	Yes
Repetition counter	No	Yes	Yes
One-shot	Yes	Yes	Yes
Up down count functionality	Yes	Yes	Yes
Low power modes	Yes	Yes	Yes
QEI support	No	No	Yes
Programmable prescalar	Yes	Yes	Yes
Shadow register mode	Yes	Yes	Yes
Events, interrupt	Yes	Yes	Yes
Auto reload functionality	Yes	Yes	Yes
Fault handling	Yes	Yes	Yes

Table 4-8. Timer Module Replacement

STM8	MSPM0 Equivalent	Reasoning
TIM1	TIMA0	Advanced control, 16-bit resolution, up/down counter, repeat counter
TIM2, 3, 5	TIMG0-11, TIMG14	16-bit resolution, General purpose,capture/ compare function
TIM4, 6	Any(1)	Basic timer

Table 4-9. Timer Use-Case Comparisons

Feature	STM8L and STM8S	MSPM0L, MSPM0C and MSPM0H
PWM	TIM1TIM2, 3, 5	All timers
Capture	TIM1TIM2, 3, 5	All timers
Compare	TIM1TIM2, 3, 5	All timers
One-shot	TIM1TIM2, 3, 5	All timers
Prescaler	All timers	All timers
Synchronization	All timers	All timers

Timer Code Examples: information about timer code examples can be found in the MSPM0 SDK examples guide.

4.6 Windowed Watchdog Timer (WWDT)

STM8 and MSPM0 both offer Window Watchdog Timers. The window watchdog timer (WWDT) initiates a system reset when the application fails to check-in during a specified window in time.

Table 4-10. WWDT Naming

STM8	MSPM0
Independent watchdog (IWDG) **Window watchdog (WWDG)**(1)	Windowed watchdog timer (WWDT)

(1) Only STM8L has the WWDG.

Table 4-11. WDT Feature Comparison

Feature	STM8L and STM8S	MSPM0L, MSPM0C and MSPM0H
Window mode	Yes	Yes
Interval timer mode	No	Yes
LFCLK source	Yes	Yes
Interrupts	Yes	Yes
Counter resolution	8bit(IWDG) 7 bit(WWDG)	25 bit
Clock divider	Yes	Yes

WWDT Code Examples: information about WWDT code examples can be found in the MSPM0 SDK examples guide.

5 Analog Peripheral Comparison

5.1 Analog-to-Digital Converter (ADC)

STM8 and MSPM0 both offer ADC peripherals to convert analog signals to a digital equivalent. For STM8, STM8L001XX and STM8L101XX do not have ADC module. STM8S series and the rest of STM8L series offer 10-bit or 12-bit ADC, respectively. For MSPM0, both device series feature a 12-bit ADC. Table 5-1 and Table 5-2 compare the different features and modes of the ADCs.

Table 5-1. Feature Set Comparison

Feature	STM8S	STM8L	MSPM0
Resolution (Bits)	10	12	12, 10, 8
Conversion Rate	0.43 Msps	1 Msps	1.68 Msps
Hardware Averaging	No	No	Yes
FIFO	No	No	Yes
ADC reference (V)	Internal: 1.224	Internal: 1.48, VDD	Internal: 1.4, 2.5, VDD
	External: 2.75V ≤ V_{REF} ≤ VDDA	External: 2.4V ≤ V_{REF} ≤ VDDA	External: 1.4 ≤ V_{REF} ≤ V_{DD} (1)
Operating power modes	Wait and low power wait	Wait	Run, sleep, stop, standby(2)
Auto power down	No	No	Yes
External input channels	Up to 16	Up to 28	MSPM0L: up to 16, MSPM0C: up to 27, MSPM0H: up to 27
Internal input channels	Temperature Sensor Internal reference voltage	Temperature Sensor Internal reference voltage	Temperature sensor Supply monitoring, analog signal chain
DMA support	No	Yes	Yes
ADC window comparator unit	No	No	Yes
Number of ADCs	Up to 1	Up to 1	Up to 1

(1) Only MSPM0C1105 and MSPM0C1106 in C series support external reference.
(2) ADC can be triggered in standby mode, which changes the operating mode.

Table 5-2. Conversion Modes

STM8	MSPM0	Comments
Single conversion mode	Single channel single conversion	ADC samples and converts a single channel once
Single scan mode	Sequence of channels conversion	ADC samples a sequence of channels and converts once.
Continuous and buffered continuous modes	Repeat single channel conversion	The one selected channel is repeatedly sampled and converted
Continuous scan mode	Repeat sequence of channels conversion	The group of channels are sampled and converted repeatedly

ADC Code Examples: information about ADC code examples can be found in the MSPM0 SDK examples guide.

5.2 Comparator (COMP)

For STM8, the STML value line has comparators, but S series and the rest of L series do not. As for MSPM0, L series offers integrated comparators as optional peripherals and C series devices do not. In both families of devices comparators are denoted as COMPx, where the *x* final character refers to the specific comparator module being considered. The comparator modules can take inputs from various internal and external sources, and can be used to trigger changes in power mode or control PWM signals. A summary of how the MSPM0 and STM8 comparator modules compare feature-by-feature is included in Table 5-3.

Table 5-3. COMP Feature Set Comparison

Feature	STM8L Value line	MSPM0L and MSPM0C[3]
Available comparators	Up to 2	Up to 1
Output routing	External I/Os	Multiplexed I/O pins
	TIM1 BRK or OCREFCLR inputs TIM2/TIM3 input capture 2	-
	Interrupt or event interface	Interrupt or event interface
Positive input	External I/Os	Multiplexed I/O pins OPA1 output DAC8[1] output
Negative input	Internal reference voltage Internal reference voltage submultiple (1/4, 1/2, 3/4) DAC output One of three external I/Os	Multiplexed I/O pins internal temperature sensor OPA0 output DAC8 output
Programmable hysteresis	No	None, 10mV, 20mV, 30mV
		Other values from 0V to V_{DD} using DAC8
Register lock	No	Yes, some COMP registers (writes require key)
Window comparator configuration	Yes	No (single COMP)
Input short mode	No	Yes
Operating modes	optimum speed, consumption ratio	High speed, low power
Output filtering	No	Blanking filter
		Adjustable analog filter
Output polarity control	No	Yes
Interrupts	Rising edge	Rising edge
	Falling edge	Falling edge
	Both edges	Output ready
Exchange inputs (2)	No	Yes

(1) The 8-bit DAC is integrated in COMP.
(2) When enable exchange inputs mode, the input signals of comparator positive and negative terminals are exchanged. Additionally, the output signal from the comparator is inverted too.
(3) MSPM0H currently does not have COMP.

COMP Code Examples: information about COMP code examples can be found in the MSPM0 SDK examples guide.

5.3 Voltage References (VREF)

The STM8 and MSPM0 both have internal references which can be used to supply a reference voltage to internal peripherals and can output internal references to external peripherals. As for STM8, only STM8L value line has the internal reference.

Table 5-4. VREF Feature Set Comparison

Feature	STM8S	STM8L	MSPM0
Internal reference (V)	1.8, 1.2	1.22	1.4, 2.5
External reference (V)	2.4 ≤ V_{REFP} ≤V_{DDA} V	2.4 ≤ V_{REFP} ≤V_{DDA} V	1.4 ≤ V_{REF} ≤ V_{DD} V(1)
Output internal reference	No	Yes	Yes
Internally connect to ADC	Yes	Yes	Yes
Internally connect to DAC	N/A	Yes	N/A
Internally connect to COMP	N/A	Yes	Yes
Internally connect to OPA	N/A	N/A	MSPM0L supported; MSPM0C and H do not supported

(1) MSPM0C1103 and MSPM0C1104 does not support external references.

For the MSPM0 VREF, you must enable the power bit, PWREN Bit0 (ENABLE).

VREF Code Examples: code examples that use VREF can be found in the MSPM0 SDK examples guide.

6 Summary

The goal of this application note is to help engineers migrate from STM8 devices to MSPM0. The differences between STM8 devices and MSPM0 in terms of ecosystem, CPU, analog and digital peripherals are compared in this document. The application note also describes in detail how to quickly get started with MSPM0 through examples to help users speed up the development progress.

7 References

- Texas Instruments, MSPM0 SDK User Guide
- Texas Instruments, MSPM0 Tools Guide
- Texas Instruments, Driverlib API Guide
- Texas Instruments, Code Composer Studio (CCS)
- Texas Instruments, MSPM0 L-Series 32MHz Microcontrollers Technical Reference Manual, technical reference manual
- Texas Instruments, MSPM0 C-Series 24MHz Microcontrollers, technical reference manual
- IAR, IAR website
- Keil, Keil website
- Texas Instruments, CCS quick start guide
- Texas Instruments, CCS training videos
- Texas Instruments, CCS user's guide
- Texas Instruments, IAR quick start guide
- Texas Instruments, IAR training videos
- IAR, IAR user's guide
- Keil, Keil quick start guide
- Keil, Keil training videos
- Keil, Keil getting started
- Texas Instruments, SYSCONFIG IDE
- Texas Instruments, MSPM0 SysConfig Guide
- Texas Instruments, XDS110 Debug Probe
- Texas Instruments, J-Link Debug Probes page
- Texas Instruments, LP-MSPM0G3507 LaunchPad development kit
- Texas Instruments, LP-MSPM0L1306 LaunchPad development kit
- Texas Instruments, LP-MSPM0C1104 LaunchPad development kit

8 Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from Revision * (December 2023) to Revision A (May 2025)	Page
• Updated the MSPM0H series to the comparison with STM8 device throughout the document	4
• Updated the MSPM0C series specs throughout the document	4
• Updated the new ecosystem, including IDE version and MSP flashing tools	5

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