

# Migration Guide From Renesas RL78 to Arm-Based MSPM0



Helic Chi, Zoey Wei, and Janz Bai

## ABSTRACT

This application note assists with migrating from the Renesas RL78 platform to the Texas Instrument MSPM0 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 RL78 development environment to quickly ramp with the MSPM0 series of MCUs.

## Table of Contents

<b>1 MSPM0 Portfolio Overview</b> .....	4
1.1 Introduction.....	4
1.2 Portfolio Comparison of Renesas RL78 MCUs to MSPM0 MCUs.....	4
<b>2 Ecosystem And Migration</b> .....	5
2.1 Ecosystem Comparison.....	5
2.2 Migration Process.....	13
2.3 Example.....	27
<b>3 Core Architecture Comparison</b> .....	35
3.1 CPU.....	35
3.2 Embedded Memory Comparison.....	35
3.3 Power UP and Reset Summary and Comparison.....	37
3.4 Clocks Summary and Comparison.....	39
3.5 MSPM0 Operating Modes Summary and Comparison.....	40
3.6 Interrupts and Events Comparison.....	42
3.7 Debug and Programming Comparison.....	47
<b>4 Digital Peripheral Comparison</b> .....	48
4.1 General-Purpose I/O (GPIO, IOMUX).....	48
4.2 Universal Asynchronous Receiver-Transmitter (UART).....	49
4.3 Serial Peripheral Interface (SPI).....	49
4.4 Inter-Integrated Circuit (I2C).....	50
4.5 Timers (TIMGx, TIMAx).....	51
4.6 Windowed Watchdog Timer (WWDT).....	52
4.7 Real-Time Clock (RTC).....	52
<b>5 Analog Peripheral Comparison</b> .....	53
5.1 Analog-to-Digital Converter (ADC).....	53
5.2 Comparator (COMP).....	54
5.3 Digital-to-Analog Converter (DAC).....	55
5.4 Operational Amplifier (OPA).....	55
5.5 Voltage References (VREF).....	56

## List of Figures

Figure 2-1. MSPM0 Ecosystem Overview.....	6
Figure 2-2. MSPM0 SDK Structure.....	6
Figure 2-3. RL78 Software Development Environment.....	7
Figure 2-4. MSPM0 SysConfig.....	9
Figure 2-5. Peripheral List Comparison.....	10
Figure 2-6. Comparison of Interrupt Settings.....	10

Figure 2-7. MSPM0 Debugging.....	11
Figure 2-8. MSPM0G3507 Launchpad Overview.....	12
Figure 2-9. Arm Cortex 10-Pin Definition.....	12
Figure 2-10. MSPM0G3507 Launchpad Feature Function.....	13
Figure 2-11. MSPM0 Migration Flowchart.....	13
Figure 2-12. Portfolio of MSPM0L and MSPM0G Series.....	14
Figure 2-13. Portfolio of MSPM0C series.....	14
Figure 2-14. MSPM0 Product Selection Tool.....	15
Figure 2-15. MSPM0 Important Document List.....	15
Figure 2-16. MSPM0 Relevant Technical Documentation List.....	15
Figure 2-17. Ordering and Quality Part View.....	16
Figure 2-18. CCS Installation.....	16
Figure 2-19. CCS Installation- MSPM0 Support Selection.....	17
Figure 2-20. CCS Installation- J-Link Download Selection.....	17
Figure 2-21. CCS Launch Workspace.....	18
Figure 2-22. Create a New Project in CCS.....	18
Figure 2-23. Commonly Used Function.....	19
Figure 2-24. Commonly Used Debug Functions.....	19
Figure 2-25. Commonly Used Project Settings.....	19
Figure 2-26. MSPM0 SDK Download.....	20
Figure 2-27. MSPM0 SDK Installation.....	20
Figure 2-28. MSPM0 SDK Fold.....	20
Figure 2-29. Document Overview.....	21
Figure 2-30. import CCS Projects.....	22
Figure 2-31. Choose Program From SDK.....	22
Figure 2-32. Remove Duplicated Project.....	23
Figure 2-33. Project and README.md.....	23
Figure 2-34. CCS Project Overview.....	24
Figure 2-35. MCU View in Smart Configuration and SysconfigSysConfig.....	24
Figure 2-36. Add Relevant File.....	25
Figure 2-37. Include Options Set.....	25
Figure 2-38. Successful Build.....	25
Figure 2-39. Ultra Librarian Tool Entrance.....	26
Figure 2-40. MSPM0 Minimum System.....	26
Figure 2-41. MSPM0 Minimum System Attention.....	26
Figure 2-42. Create Program Files.....	27
Figure 2-43. Program Software and Tool.....	27
Figure 2-44. Code Example File.....	28
Figure 2-45. PWM Configuration in SysConfig.....	29
Figure 2-46. To Get Detailed Information of Each Item.....	29
Figure 2-47. Pins Configuration.....	30
Figure 2-48. The files SysConfig Updates.....	30
Figure 2-49. Hardware Setup.....	31
Figure 2-50. Add Breakpoint Solutions.....	31
Figure 2-51. Ultra Librarian Tool Download.....	32
Figure 2-52. Run Altium Designer Script.....	33
Figure 2-53. PCB Library and Schematic File.....	33
Figure 2-54. Import library.....	34
Figure 3-1. MSP Reset Function.....	38
Figure 3-2. Internal Maskable Interrupt Hierarchy of RL78.....	43
Figure 3-3. Peripheral Interrupt Hierarchy of MSPM0.....	44
Figure 3-4. Generic Event Route.....	44
Figure 3-5. Event Management Register Relationship.....	45
Figure 3-6. MSPM0 Event and Interrupt Handling.....	45
Figure 3-7. RL78 Event Link Controller.....	46
Figure 3-8. RL78 Event and Interrupt Handling.....	46

### List of Tables

Table 1-1. Comparison of the TI MSPM0Gx/Lx /Cx and Renesas RL78 Series.....	4
Table 2-1. Ecosystem Comparison.....	5
Table 2-2. Software Ecosystems.....	7
Table 2-3. Comparison Between CCS and e <sup>2</sup> Studio.....	8

Table 2-4. MSPM0 supported IDEs Overview.....	8
Table 2-5. MSPM0 debugger compare.....	11
Table 2-6. MSPM0 Example Coverage.....	21
Table 2-7. Empty Project Description.....	21
Table 3-1. Comparison of CPU Feature Sets.....	35
Table 3-2. Comparison of Flash Features.....	35
Table 3-3. The Comparison of Flash Memory Regions.....	36
Table 3-4. Comparison of SRAM Features.....	37
Table 3-5. Summary and Comparison of Power Up.....	37
Table 3-6. Oscillator Comparison.....	39
Table 3-7. Oscillators in MSPM0 MCUs.....	39
Table 3-8. Clock Signal Comparison.....	39
Table 3-9. Peripheral Clock Sources.....	40
Table 3-10. Operating Modes Comparison Between RL78 and MSPM0 Devices.....	41
Table 3-11. Interrupts Comparison.....	42
Table 3-12. Event Management Comparison.....	47
Table 3-13. Programming Mode Comparison.....	47
Table 4-1. GPIO Feature Comparison.....	48
Table 4-2. UART Feature Comparison.....	49
Table 4-3. SPI Feature Comparison.....	49
Table 4-4. I2C Feature Comparison.....	50
Table 4-5. Timer Naming.....	51
Table 4-6. Timer Feature Comparison.....	51
Table 4-7. Timer Module Replacement.....	51
Table 4-8. Timer Use-Case Comparisons.....	51
Table 4-9. WWDT Naming.....	52
Table 4-10. WDT Feature Comparison.....	52
Table 4-11. RTC Feature Comparison.....	52
Table 5-1. Feature Set Comparison.....	53
Table 5-2. Conversion Modes.....	53
Table 5-3. COMP Feature Set Comparison.....	54
Table 5-4. DAC Feature Set Comparison.....	55
Table 5-5. OPA Feature Set Comparison.....	55
Table 5-6. VREF Feature Set Comparison.....	56

## Trademarks

LaunchPad™, 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.

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 their project's needs. 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 Renesas RL78. This application note assists with migration from RL78 MCUs to MSPM0 MCUs by comparing device features and ecosystems.

## 1.2 Portfolio Comparison of Renesas RL78 MCUs to MSPM0 MCUs

**Table 1-1. Comparison of the TI MSPM0Gx/Lx /Cx and Renesas RL78 Series**

		Renesas RL78 G Series	Renesas RL78 L Series	Renesas RL78 I Series	Renesas RL78 F Series	TI MSPM0 Gx Series	TI MSPM0 Lx Series	TI MSPM0 Cx Series
<b>Core</b>		RL78 CPU core				Arm Cortex-M0+		
<b>Frequency</b>		16/20/24/32 MHz	24 MHz	24/32 MHz	24/32/40 MHz	80 MHz	32 MHz	24 MHz
<b>Supply Voltage</b>		1.6/1.8/2/2.7-5.5 V, 1.6-3.6 V	1.6-5.5 V 1.8-3.6/5.5 V	1.7/1.9/2.4/2.7-5.5 V, 1.6-3.6 V	2.7-5.5 V, 1.8-5.5 V	1.62-3.6 V	1.62-3.6 V	1.62-3.6 V
<b>Temperature</b>		-40-125°C, -25-75°C	-40-85°C, -40-125°C	-40-85°C, -40-105°C, -40-125°C	-40-105°C, -40-125°C, -40-150°C	-40-125°C	-40-125°C	-40-125°C
<b>Memory</b>		768KB to 1KB	256KB to 8KB	512KB to 8KB	512KB to 8KB	128KB to 32KB	64KB to 8KB	16KB to 8KB
<b>RAM</b>		Up to 144KB	Up to 32KB	Up to 32KB	Up to 4KB	Up to 32KB	Up to 4KB	1KB
<b>GPIO (max)</b>		130	79	76	130	60	28	18
<b>Analog</b>	ADC	Up to 12-bit x 28-ch	Up to 12-bit x 14-ch	Up to 12-bit x 17-ch	Up to 12-bit x 25-ch	2x 4-Msps 12-bit	1x 1-Msps 12-bit ADC	1x 1-Msps 12-bit ADC (10-ch)
	DAC	Up to 10-bit x 2-ch	Up to 12-bit x 2-ch	12-bit x 1-ch (RL/11E)	8-bit x 1-ch	12-bit	8-bit	none
	comparator	Up to 2-ch	Up to 2-ch	Up to 2-ch	Up to 1-ch	3x high-speed	1x high-speed	none
<b>Communication (max number)</b>	UART	4	4	4	5	4	2	1
	I2C	10	5	4	5	2(Fast)	2(Fast)	1
	SPI	0	4	1	4	2	1	1
	CAN	0	0	0	2	1(CAN-FD)	0	0
	LIN	1 (UART support)			3 (UART support)	1 (UART support)		
<b>Other key peripherals / features</b>		Intern boost LCD USB(RL/G1A) 1-ch PGA Bluetooth(RL/G1D) 1% oscillator	Intern boost LCD USB(RL78/L1C) 3 - ch AMP	LCD 1-ch PGA 3-ch AMP USB VBAT, sigma-delta AFE	MATHACL, ASIL-B, 150°C	2x op amps CAN-FD, USB, Fast4Msps sim-sam ADCs, Math acceleration	2x op amps LCD(L2228)	Smallest QFN package (2x2), 0.5/0.65 mm pitch packages, Pin-compatible with industry
<b>Timer number</b>		1/2/4/5	1/2/3	1/2/5	1/2	4	7	4
<b>Pin count</b>		16-128 pins	32-100 pins	20-100 pins	20-144 pins	20-100 pins	16-80 pins	8-48 pins

**Table 1-1. Comparison of the TI MSPM0Gx/Lx /Cx and Renesas RL78 Series (continued)**

	Renesas RL78 G Series	Renesas RL78 L Series	Renesas RL78 I Series	Renesas RL78 F Series	TI MSPM0 Gx Series	TI MSPM0 Lx Series	TI MSPM0 Cx Series
<b>Security</b>	CRC, RNG, AES library, SHA hash function library, RSA library	CRC, AES GCM	CRC,	CRC	CRC, TRNG, AES256	CRC	CRC
<b>Low power <sup>(1)</sup></b>	Active: Low to 37.5 $\mu$ A/MHz Stop:Low to 0.2 $\mu$ A	Active: Low to 66 $\mu$ A/MHz Stop:Low to 0.23 $\mu$ A	Active: Low to 96 $\mu$ A/MHz Stop:Low to 0.23 $\mu$ A	(Not mentioned)	Active: 85 $\mu$ A/MHz Standby: 1.5 $\mu$ A	Active: 71 $\mu$ A/MHz Standby: 1 $\mu$ A	Active: 100 $\mu$ A/MHz Standby: 5 $\mu$ A

(1) RL78 Stop mode is similar to MSPM0 Shut down mode(CPU, Clock, Peripherals are shut down)

Some performance comparisons of the RL78 and MSPM0 are provided here. Details can be found in the following sections.

## 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	RL78 Devices	MSPM0 Devices
Code source	Middleware Renesas Flash Driver/Self-programming library Drivers OS	MSPM0-SDK(DriverLib, Middleware, RTOS, Code example)
IDE	e <sup>2</sup> studio CS+ CC-RL IAR	CCS IAR Keil
Software Configuration	Smart Configurator	SysConfig
Flash programming tool	Renesas Flash Programmer	UniFlash
Programmer	PG-FP6	MSP-GANG
Debugger	E2 Emulator E2 Emulator Lite EZ-CUBE	XDS110 J-LINK
Hardware	Fast Prototyping Board Target Board Starter kits	LP-MSPM0G3507 launchpad LP-MSPM0L1306 launchpadLP-MSPM0C1104 launchpad

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

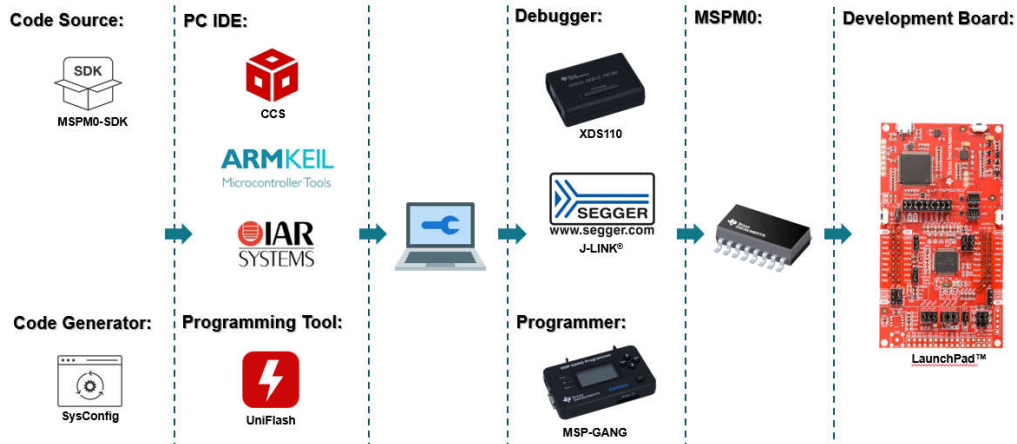


Figure 2-1. MSPM0 Ecosystem Overview

### 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.

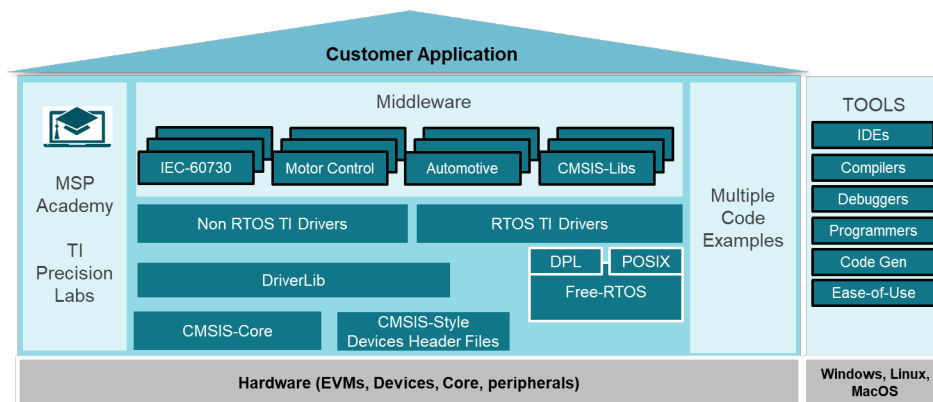


Figure 2-2. MSPM0 SDK Structure

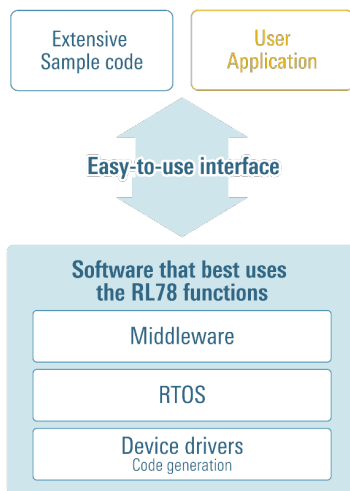
The MSPM0 SDK can be downloaded from [MSPM0-SDK Support software | TI.com](https://www.ti.com/tool/mspm0-sdk). There are four folders included in MSPM0 SDK:

**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. Though the Renesas RL78 support large middle and extensive sample code such as DSP, USB driver, but there is no package for code development program and RL78 has no example code, which means user needs to create new project and set up configurations such as the debugger from scratch. In contrast, MSPM0 SDK integrates all source code with middleware and Driver lib for easy development. The example code helps customers get started quickly and learn more about MCU peripherals.



**Figure 2-3. RL78 Software Development Environment**

**Table 2-2. Software Ecosystems**

Feature	RL78 Software	MSPM0 SDK
Register-level code	No	No
Driver library	Yes	Yes
Middleware	Yes	Yes
Self-programming	Yes	No
Out of box code	No	Yes
Free RTOS	Yes	Yes

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 The 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 RL78 is e<sup>2</sup>studio, which can download sample code and has an easy-to-use Eclipse code editor. As for TI, Code Composer Studio IDE (CCS) is highly recommended, which supports TI's microcontroller (MCU) and embedded processor portfolios. As CCS is also an Eclipse-based IDE, it's easier for users to get started. 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. Also, CCS is completely free to use and is available as both

**Table 2-3. Comparison Between CCS and e<sup>2</sup> Studio**

IDEs	CCS	e <sup>2</sup> studio
License	Free	Free
Compiler	TI Arm Clang / GCC	CC-RL/ LLVM
Current Consumption integrated in IDE	EnergyTrace	Renesas QE
Peripherals' API function assistance	not support	support
Display language	English	English Japanese Chinese
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	Smart Configuration

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 2-4](#).

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

**Table 2-4. MSPM0 supported IDEs Overview**

IDEs	CCS(Eclipse)	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 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 Smart Configuration, SysConfig is an intuitive and comprehensive collection of graphical utilities for configuring pins, peripherals, radios, subsystems, and other components. SysConfig helps manage, expose, and resolve conflicts visually so that you have more time to create differentiated applications. The tool's 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](https://www.ti.com/tool/SYSCONFIG).

Besides, 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.

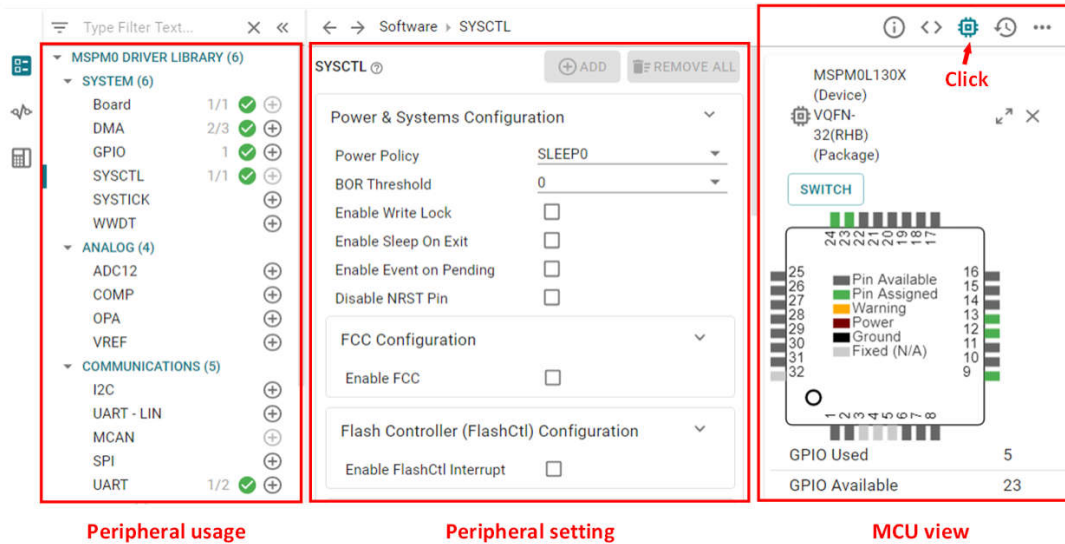


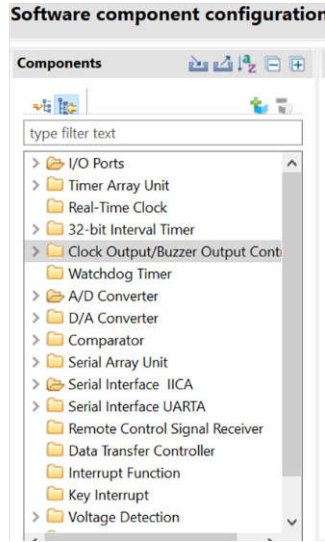
Figure 2-4. MSPM0 SysConfig

For details, see the [MSPM0 SysConfig Guide](#).

Compared to Smart Configuration, SysConfig has the following advantages:

- The classification of peripherals in SysConfig is clearer, the overall interface has high readability, and the human-computer interaction interface is great, as shown in [Figure 2-5](#).
- SysConfig has a hardware schematic display and detailed description of each peripheral, and a detailed functional description of any configuration of the GUI interface.
- Smart Configuration interrupts and pin settings are in a separate module, which is cluttered and not conducive to development. TI's SysConfig can be configured directly in specific peripheral modules, which makes development easy and clear. The comparison of interrupt settings is shown in [Figure 2-6](#). SysConfig can realize multiple peripheral linkage configurations, such as Event configuration in ADC module.
- When you configure the specific functions of a peripheral, SysConfig can show the code changes in real time, and cannot be seen in Smart configuration immediately.

Smart Configuration



Sysconfig

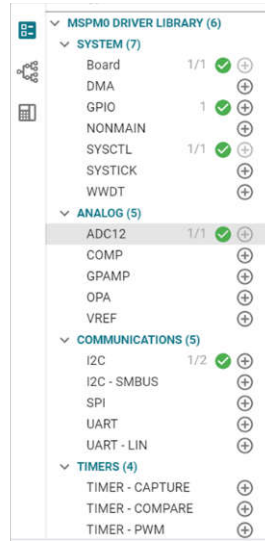


Figure 2-5. Peripheral List Comparison

Smart Configuration

Interrupt vectors

Vector Number	Vector Table Address	Interrupt	Interrupt request source	Peripheral	Priority	Status	Bit
> 15	00022H	INTSR0/INTTM01H			Level 3 (low)	N	
> 16	00024H	INTST1/INTCS10/INTTRIC10			Level 3 (low)	N	
> 17	00026H	INTSR1/INTCS11/INTTRIC11			Level 3 (low)	N	
> 18	00028H	INTSR1/INTTM03H			Level 3 (low)	N	
> 19	0002AH	INTIICAO	End of IICAO communication	IICA	Level 3 (low)	Used	N
> 20	0002CH	INTSR0/INTCS01/INTTRIC01			Level 3 (low)	N	
> 21	0002EH	INTTM01	End of timer channel 01 count or capture (at 16-bit/lowe...	TAU0	Level 3 (low)	N	
> 22	00029H	INTTM02	End of timer channel 02 count or capture	TAU0	Level 3 (low)	N	
> 23	00023H	INTTM03	End of timer channel 03 count or capture (at 16-bit/lowe...	TAU0	Level 3 (low)	N	
> 24	00034H	INTA0	End of A/D conversion	ADC	Level 3 (low)	Used	N
> 25	00036H	INTRIC	Fixed-cycle signal of real-time clock/alarm match detection	RTC	Level 3 (low)	N	
> 26	00038H	INTITL	Interval signal of 32-bit interval timer detection	ITL	Level 3 (low)	N	
> 27	0003AH	INTR	Key return signal detection	KR	Level 3 (low)	N	
> 31	00042H	INTTM04	End of timer channel 04 count or capture	TAU0	Level 3 (low)	N	
> 32	00044H	INTTM05	End of timer channel 05 count or capture	TAU0	Level 3 (low)	N	
> 33	00046H	INTTM06	End of timer channel 06 count or capture	TAU0	Level 3 (low)	N	
> 34	00048H	INTTM07	End of timer channel 07 count or capture	TAU0	Level 3 (low)	N	

Sysconfig

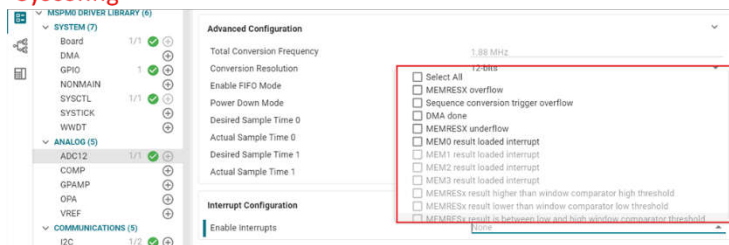
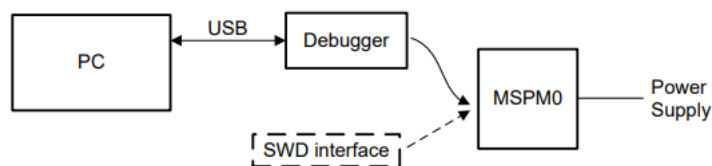


Figure 2-6. Comparison of Interrupt Settings

2.1.4 Debug Tools

For RL78, the TOOL0 pin is used for manipulation via a dedicated single-line UART to interface between the debugger (like E2 emulator or E2 emulator Lite) and the RL78/F23. The typical debugger tools for RL78 is the E2 Emulator, which supports the measuring current consumption, monitoring point and setting External Trigger input/output.

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 2-7.



**Figure 2-7. 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 (480 Mbps). 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](#).

Here is a different feature summery between XDS110 and J-LINK debugger supporting MSPM0.

**Table 2-5. MSPM0 debugger compare**

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

- (1) XDS110 OB means XDS110 on-board.
- (2) BSL means bootsrap loader.
- (3) IDE means Integrated Development Environment.
- (4) 3rd party includes IAR/Keil.

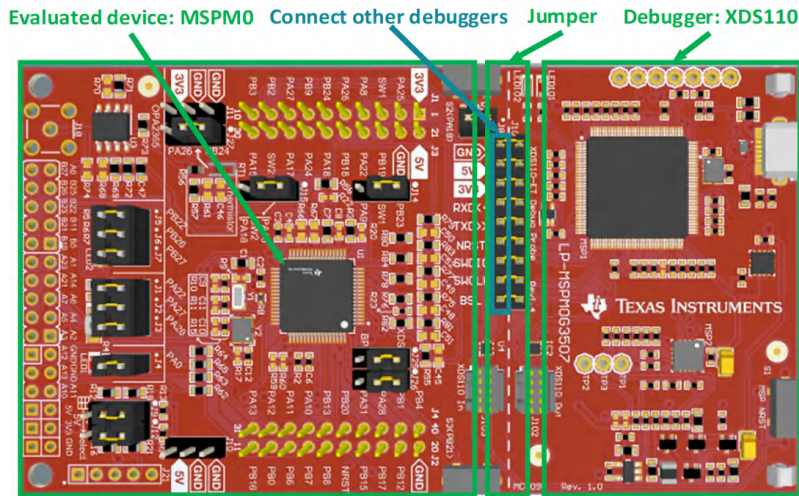
### 2.1.5 LaunchPad

Different from RL78 that has a Starter Kit, FPB and Target Board, LaunchPad development kits are the only evaluation modules for the MSPM0.

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.

- [LP-MSPM0G3507 LaunchPad development kit](#)
- [LP-MSPM0L1306 LaunchPad development kit](#)

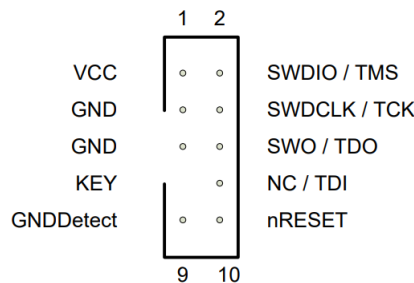
Figure 2-8 illustrates the LaunchPad overview, which contains the MCU and a XDS110 debugger. You can also use other debuggers like J-Link to debug the MCU after removing the jumpers.



**Figure 2-8. 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, it is possible to burn using the standard Arm Cortex 10 pins connector (as shown in Figure 2-9 ) 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 2-9. Arm Cortex 10-Pin Definition**

Figure 2-10 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](#)

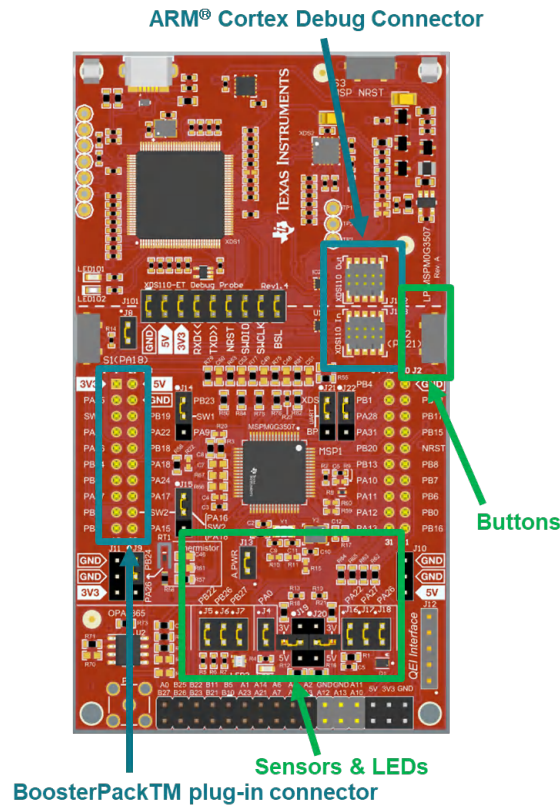


Figure 2-10. MSPM0G3507 Launchpad Feature Function

## 2.2 Migration Process

For smooth migration to MSPM0, the detailed process is written in a flow as shown in Figure 2-11. Each step is described in detail and examples are given in the following sections.

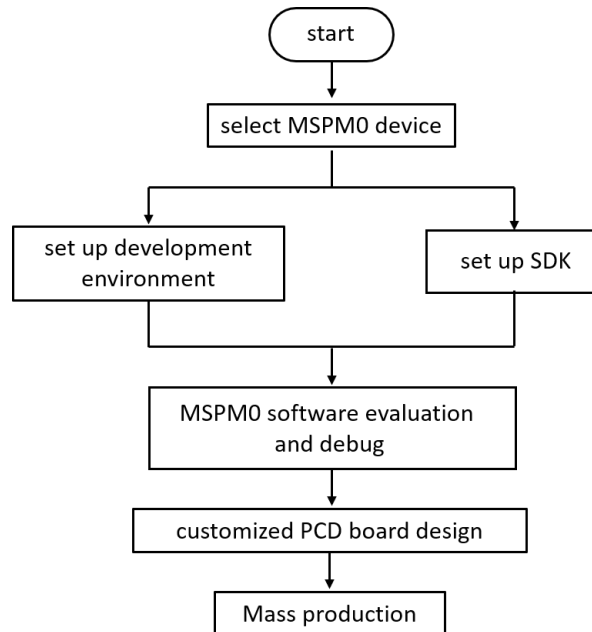


Figure 2-11. MSPM0 Migration Flowchart

### 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 2-12 shows the portfolio of MSPM0 L and G series family, which can be seen in official website of TI. And Figure 2-13 shows the portfolio of MSPM0 C series. Both portfolios all distinguish the device according to memory and packaging, which makes it easier to have a simple selection.

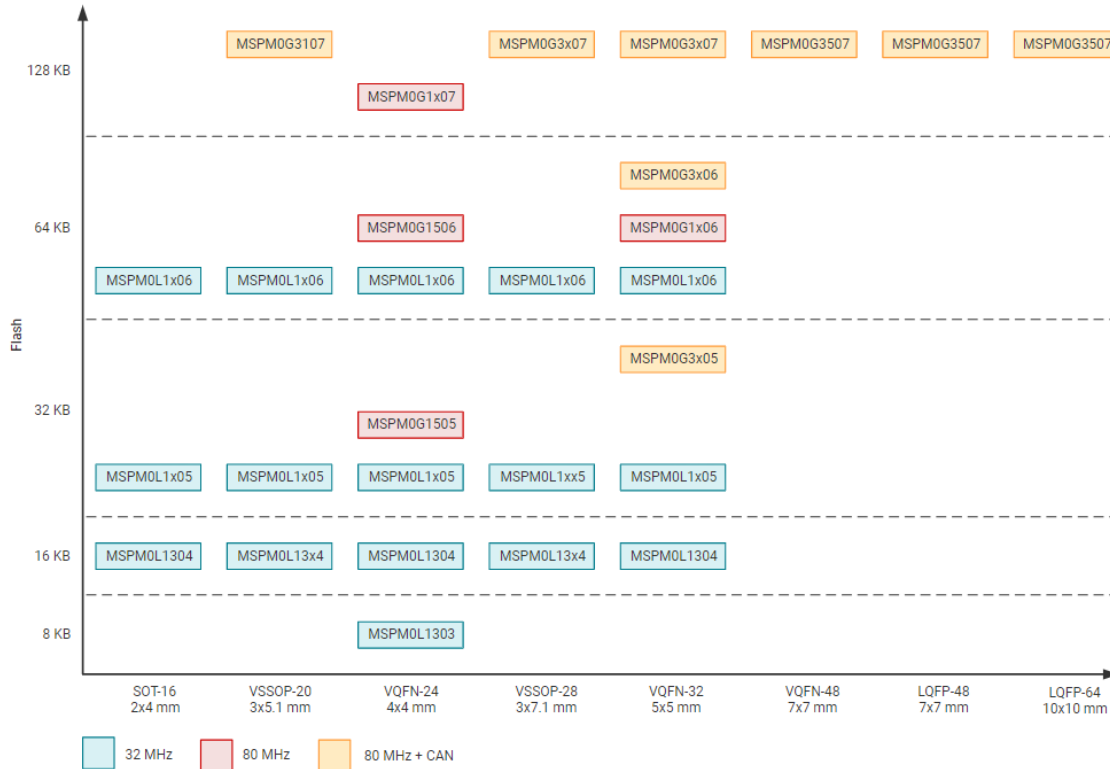


Figure 2-12. Portfolio of MSPM0L and MSPM0G Series

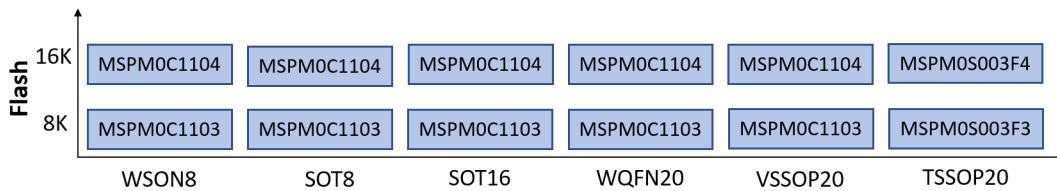


Figure 2-13. Portfolio of MSPM0C series

To narrow down to a specific device, the [product selection tool](#) plays an import role. In this link, you can use the filter on the left to do Initial screening based on your MCU peripheral requirement. For example, to filter out the MCU that meet the UART number, you can directly use the filter tool to select, and the qualified MCU devices will pop up on the right, as shown in the Figure 2-14, and you can directly go to the device page through the left search text box for detailed information of corresponding device.



Hide filters Columns Reset table 23 of 23 total products Email Download Excel Log in to view invento

Product number	Images	Price/Quantity (USD)	Frequency (MHz)	Flash memory (kByte)	RAM (kByte)	Number of GPIOs	UART	Number of I2Cs	SPI
<input type="checkbox"/> MSPM0L1345 - NEW Data sheet: PDF   HTML View alternates		US\$0.484   1ku	32	32	4	22	2	2	1
<input type="checkbox"/> MSPM0L1346 - NEW Data sheet: PDF   HTML		US\$0.544   1ku	32	64	4	22	2	2	1
<input type="checkbox"/> MSPM0L1343 - NEW Data sheet: PDF   HTML View alternates		US\$0.412   1ku	32	8	2	15	2	2	1
<input type="checkbox"/> MSPM0L1344 - NEW Data sheet: PDF   HTML View alternates		US\$0.422   1ku	32	16	2	15	2	2	1

Figure 2-14. 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 2-15. 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 ecorrigendum description of MSPM0 related series or versions.

NEW  
MSPM0L1345 ACTIVE

32-MHz Arm® Cortex®-M0+ MCU with 32-KB flash, 4-KB SRAM, 12-bit ADC, comparator, TIA

---

DATA SHEET MSPM0L130x Mixed-Signal Microcontrollers datasheet (Rev. C) PDF | HTML datasheet

---

USER GUIDES MSPM0 L-Series 32-MHz Microcontrollers Technical Reference Manual (Rev. C) MSPM0L110x, MSPM0L13xx Microcontrollers Errata (Rev. A)

User Guides errata

Figure 2-15. MSPM0 Important Document List

The website also lists the relevant technical documents on MSPM0, the most common being application notes.

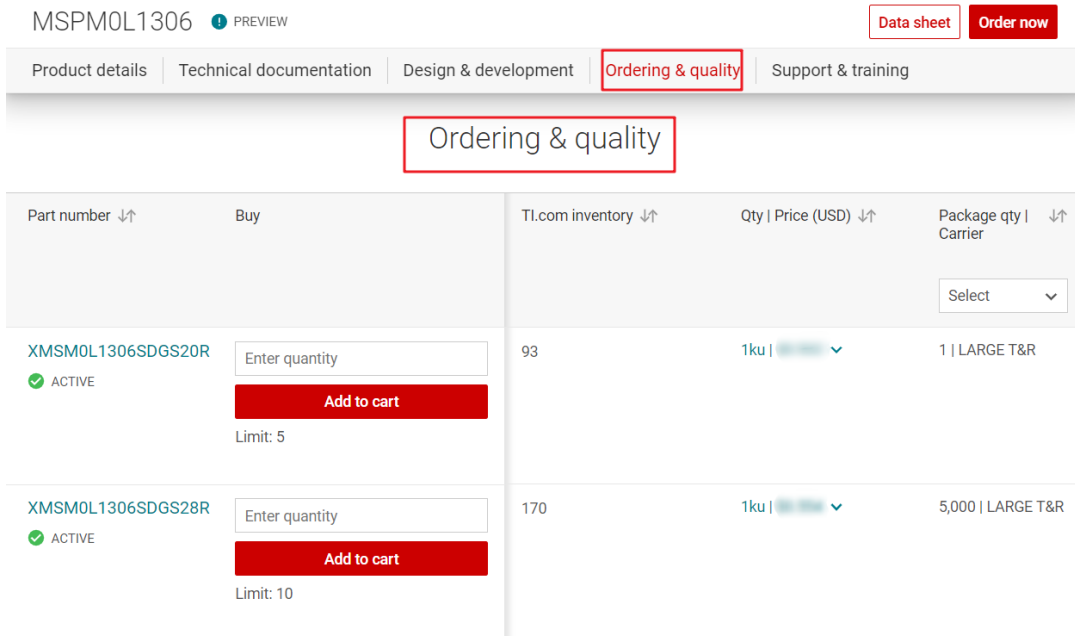
Technical documentation

★ = Top documentation for this product selected by TI

Type	Title	Date
★ Data sheet	MSPM0L130x Mixed-Signal Microcontrollers datasheet (Rev. C) PDF   HTML	27 Jun 2023
★ Errata	MSPM0L110x, MSPM0L13xx Microcontrollers Errata (Rev. A) PDF   HTML	28 Apr 2023
★ User guide	MSPM0 L-Series 32-MHz Microcontrollers Technical Reference Manual (Rev. C)	05 May 2023
Application note	MSPM0 Bootloader (BSL) Host Implementation (Rev. A) PDF   HTML	06 Jul 2023
Application note	EEPROM Emulation Type A Solution PDF   HTML	18 Apr 2023
Application note	STM32에서 Arm 기반 MSPM0으로의 마이그레이션 가이드 (Rev. A) PDF   HTML	12 Apr 2023
Application note	從 STM32 到 Arm 架構的 MSPM0 移轉指南 (Rev. A) PDF   HTML	12 Apr 2023
Application note	EEPROM Emulation Type B Design PDF   HTML	11 Apr 2023

Figure 2-16. MSPM0 Relevant Technical Documentation List

After completing the selection, you can check the price and other information through ordering and quality, as shown in [Figure 2-17](#).



MSPM0L1306 PREVIEW Data sheet Order now

Product details | Technical documentation | Design & development | **Ordering & quality** | Support & training

**Ordering & quality**

Part number ↓↑	Buy	TI.com inventory ↓↑	Qty   Price (USD) ↓↑	Package qty   Carrier ↓↑
XMSM0L1306SDGS20R ACTIVE	<input type="text" value="Enter quantity"/> <span>Add to cart</span> Limit: 5	93	1ku   <span>Price</span>	1   LARGE T&R
XMSM0L1306SDGS28R ACTIVE	<input type="text" value="Enter quantity"/> <span>Add to cart</span> Limit: 10	170	1ku   <span>Price</span>	5,000   LARGE T&R

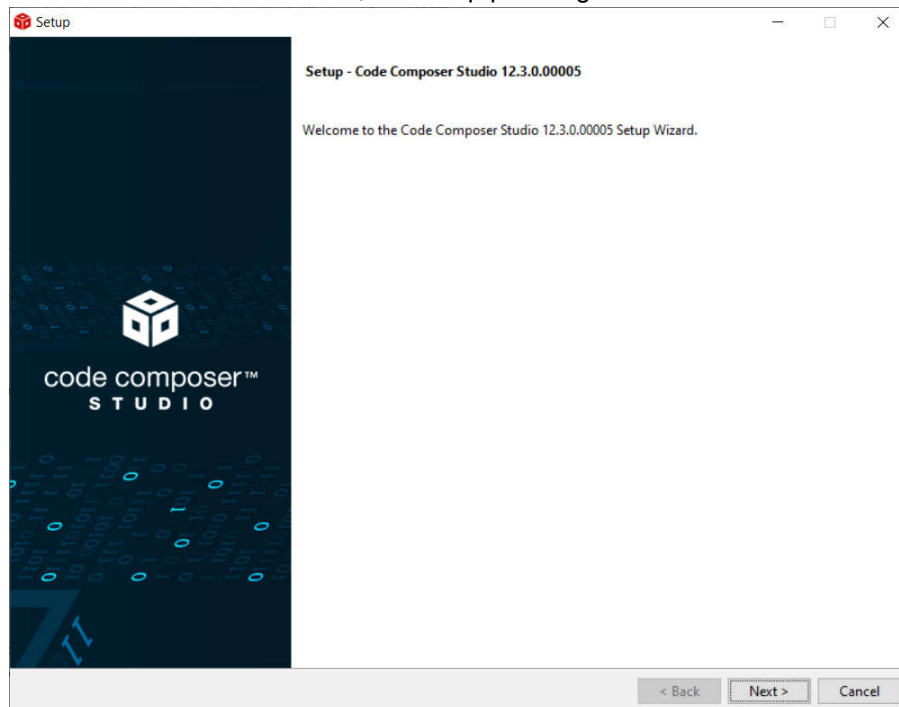
**Figure 2-17. Ordering and Quality Part View**

## 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 download link and start installation, and keep pressing next.



**Figure 2-18. CCS Installation**



2. Select MSPM0 support component.

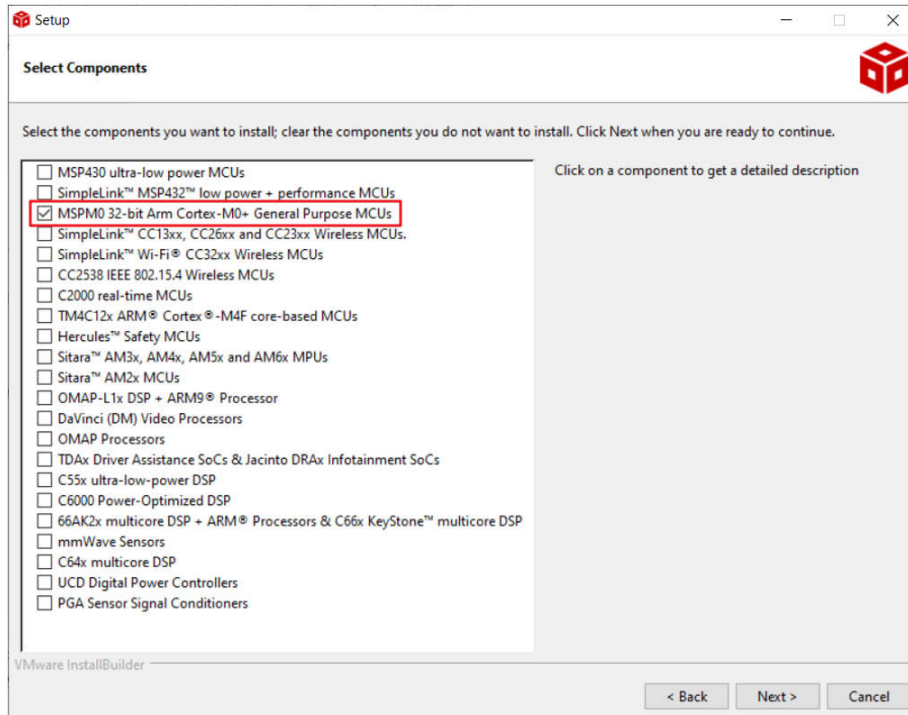


Figure 2-19. CCS Installation- MSPM0 Support Selection

3. Select J-link, if needed.

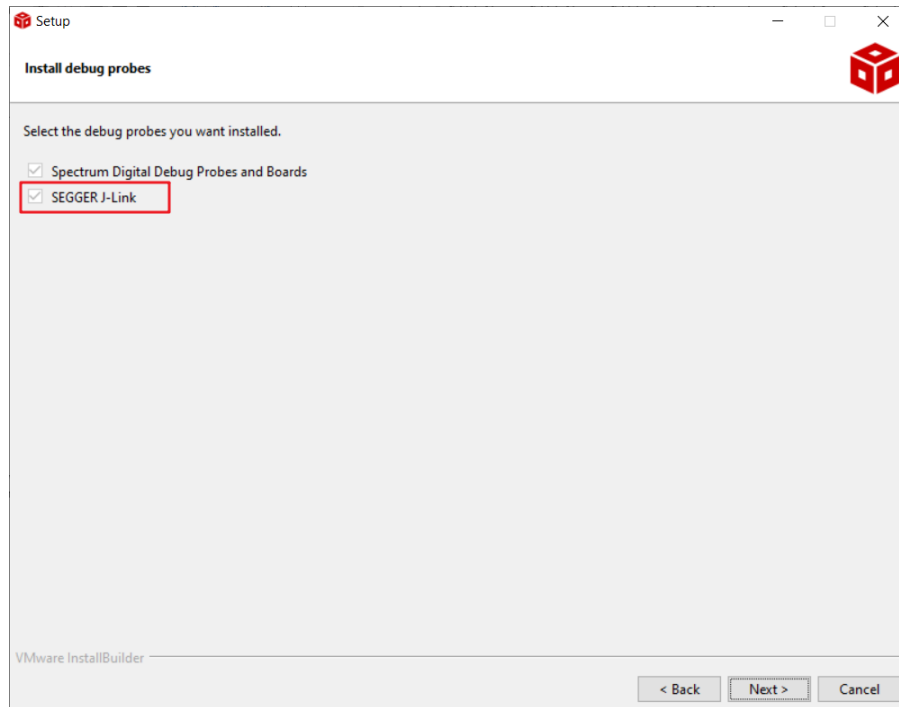
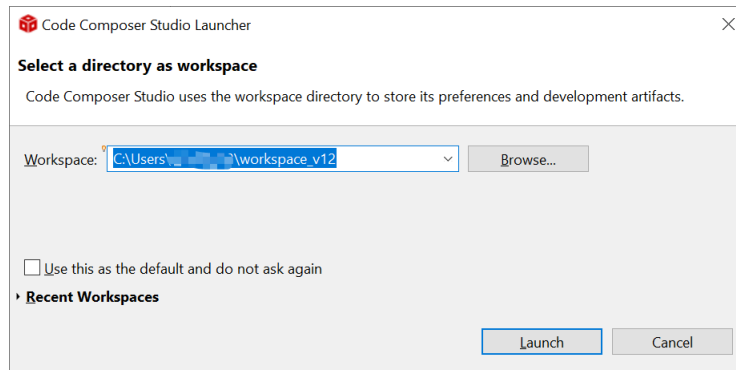


Figure 2-20. CCS Installation- J-Link Download Selection

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 your imported project. The step in CCS is same as that in e<sup>2</sup>studio.

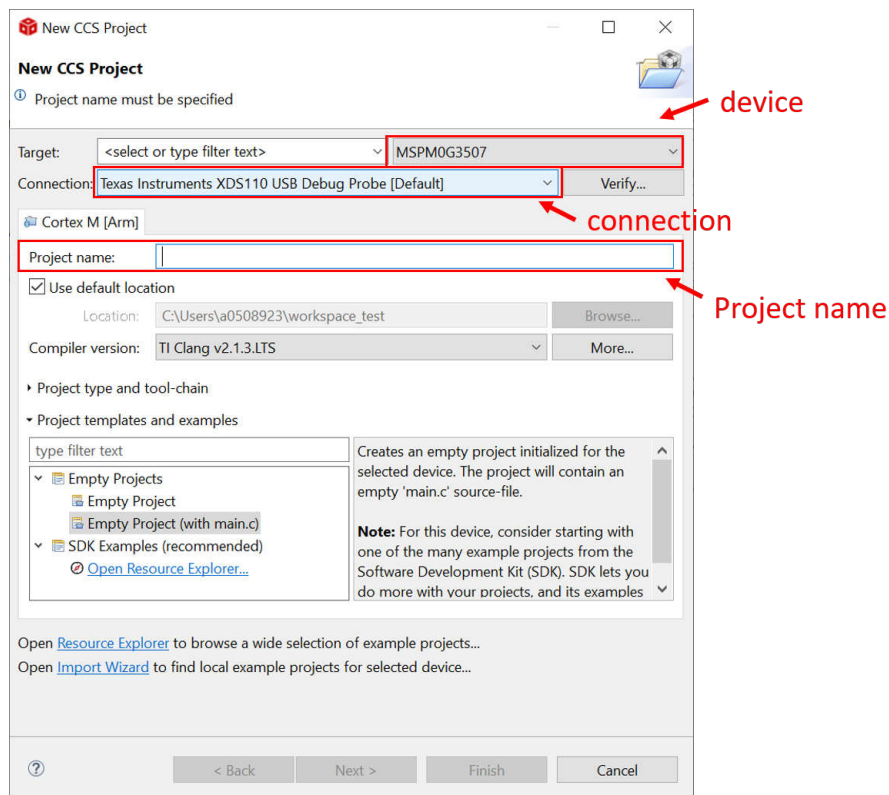


**Figure 2-21. CCS Launch Workspace**

2. If you want to create a new project, 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 2-22. And then, the program can be created after project name is added and press finish button.

This is similar to the creation of new project in e<sup>2</sup>studio , which requires selecting device, tool chain, and program name.

It is recommended to find the MSPM0 SDK example, which introduces how to use CCS in Section 2.2.4.



**Figure 2-22. Create a New Project in CCS**

3. Figure 2-23 shows a quick introduction to CCS functions.

Shortcut key functions :

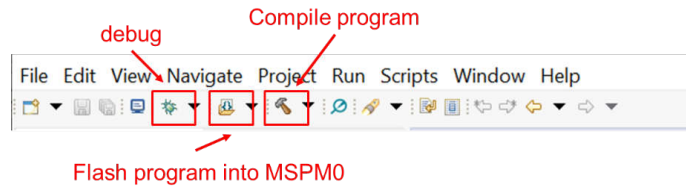


Figure 2-23. Commonly Used Function

Debug functions:

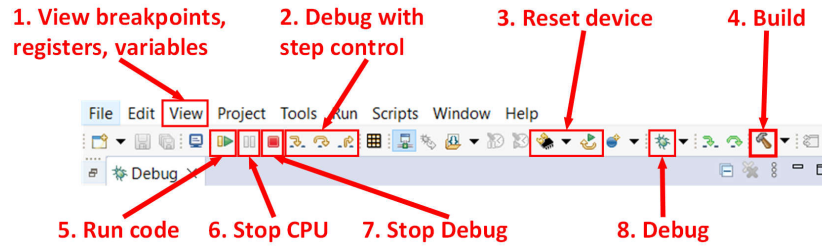


Figure 2-24. Commonly Used Debug Functions

Project properties common used settings:

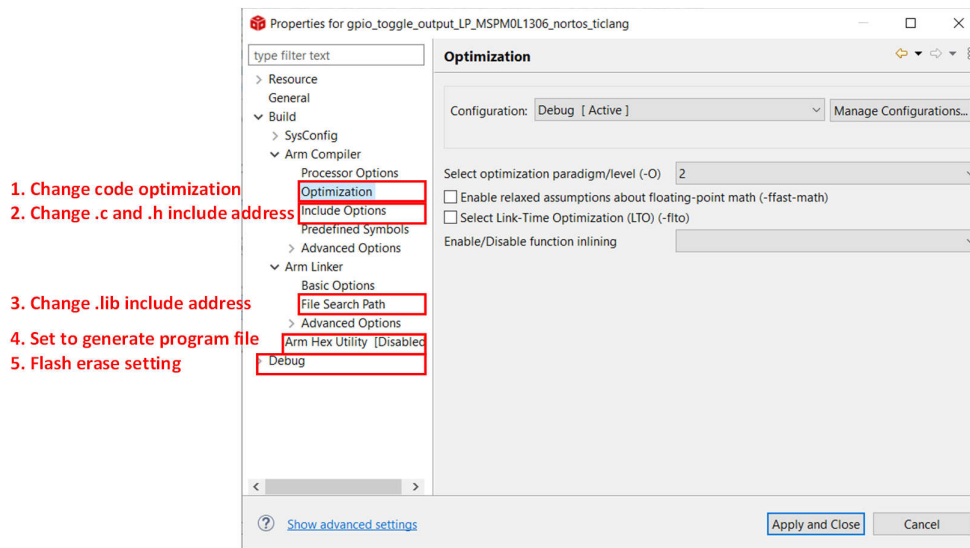


Figure 2-25. Commonly Used Project Settings

For detailed information, see : [Code Composer Studio IDE Version 12.4+ for MSPM0 MCUs — Code Composer Studio IDE for MSPM0 MCUs 1.0 documentation](#)

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

As for RL78, there is no software package and users are only allowed to search example code from IDE’s “Developer Assistance” or download a specific sample code from the website. But TI has a supported software development kit that makes it easy to develop.

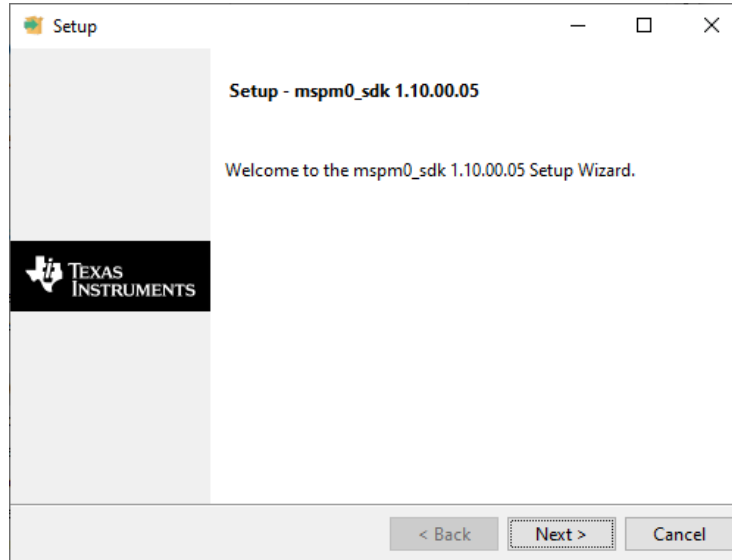
### 2.2.3.1 Set Up MSPM0 SDK

1. Click the link to download [MSPM0 SDK](#).



**Figure 2-26. MSPM0 SDK Download**

2. Press next to install SDK.

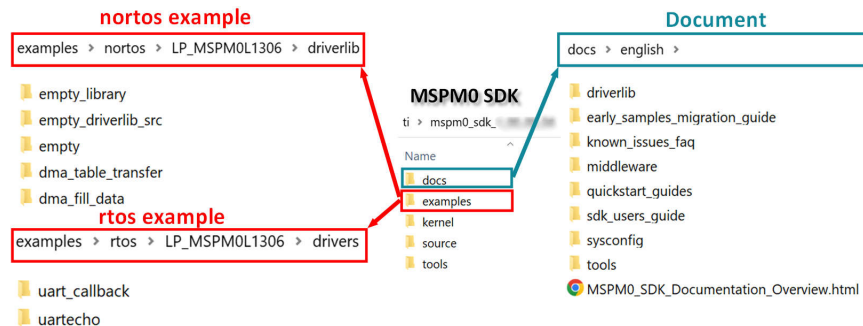


**Figure 2-27. MSPM0 SDK Installation**

3. Finish MSPM0 SDK download.

### 2.2.3.2 Quick Introduction of SDK

When you have finished downloading, the file content is shown in the SDK folder, which is "c:/ti/mspm0\_sdk\_xxx" by default, as shown in [Figure 2-28](#), among which the mostly used folders are examples folder and docs folder.



**Figure 2-28. MSPM0 SDK Fold**

Table 2-6 shows a summary of example coverage.

**Table 2-6. MSPM0 Example Coverage**

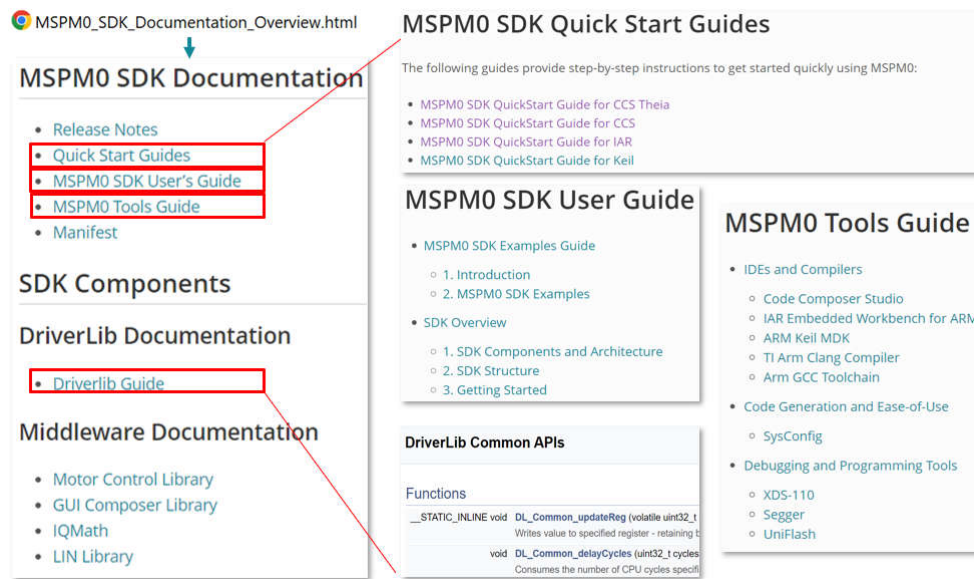
Supported by SDK	Platform			
IDE	CCS		Keil	IAR
Compilers	TI Arm-Clang	GUN Arm	Arm/Keil Compiler	IAR Arm compiler
RTOS	FreeRTOS			
Code examples	Driverlib/TI Drivers(drivers)			

In nortos examples, you can also find three empty projects for users to build their own project. Table 2-7 the differences.

**Table 2-7. 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 2-29.

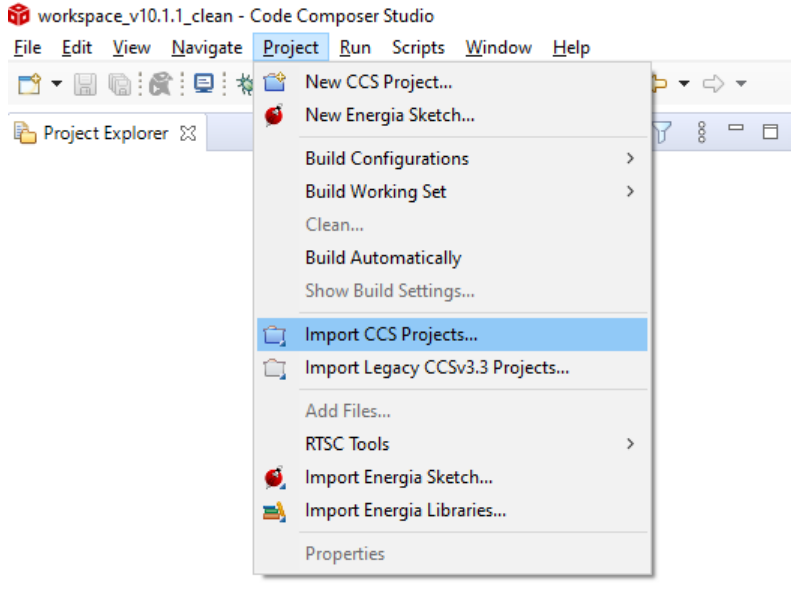


**Figure 2-29. Document Overview**

### 2.2.4 Step 4. Software Evaluation

Here are some simple steps to port the example into CCS.

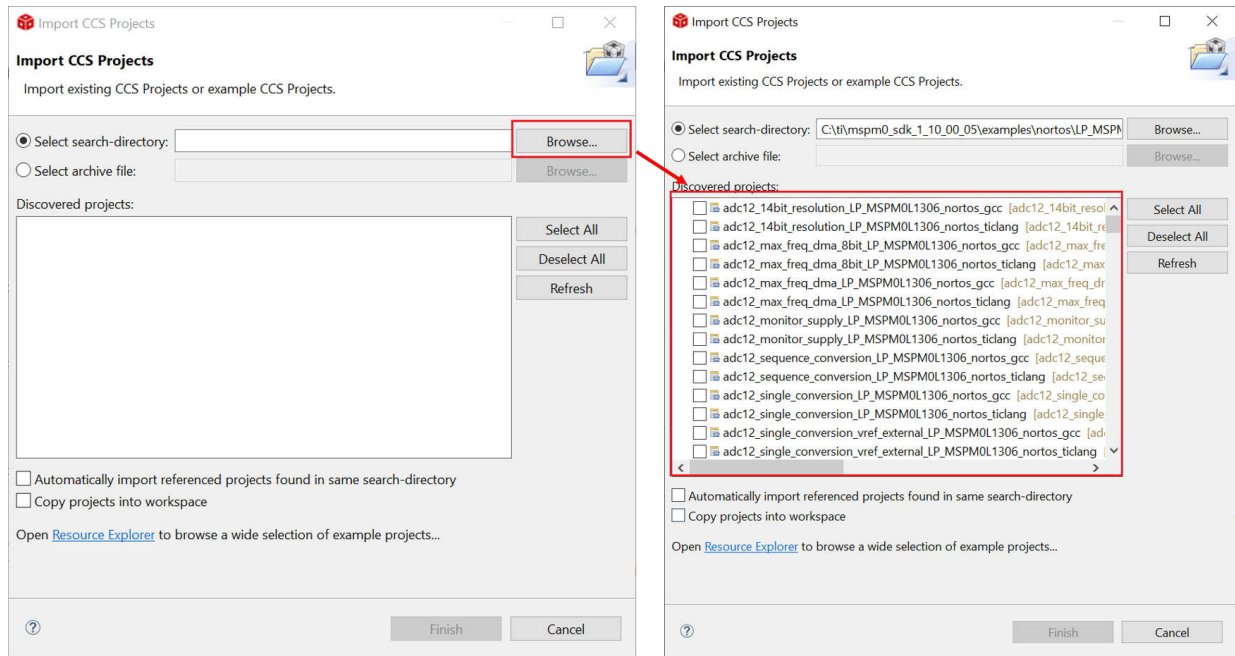
1. Select Project, and then import CCS Projects from the menu.



**Figure 2-30. import CCS Projects**

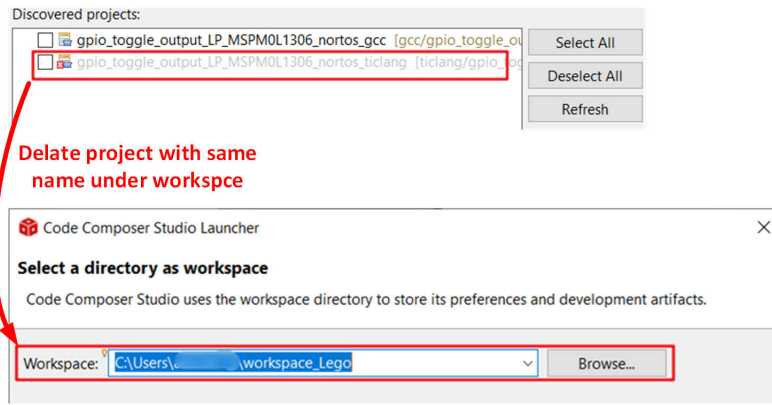
2. Choose the program from SDK. Take the MSPM0L1306, for example.

`\mspm0_sdk_1_10_00_05\examples\nortos\LP_MSPM0L1306\driverlib`



**Figure 2-31. Choose Program From SDK**

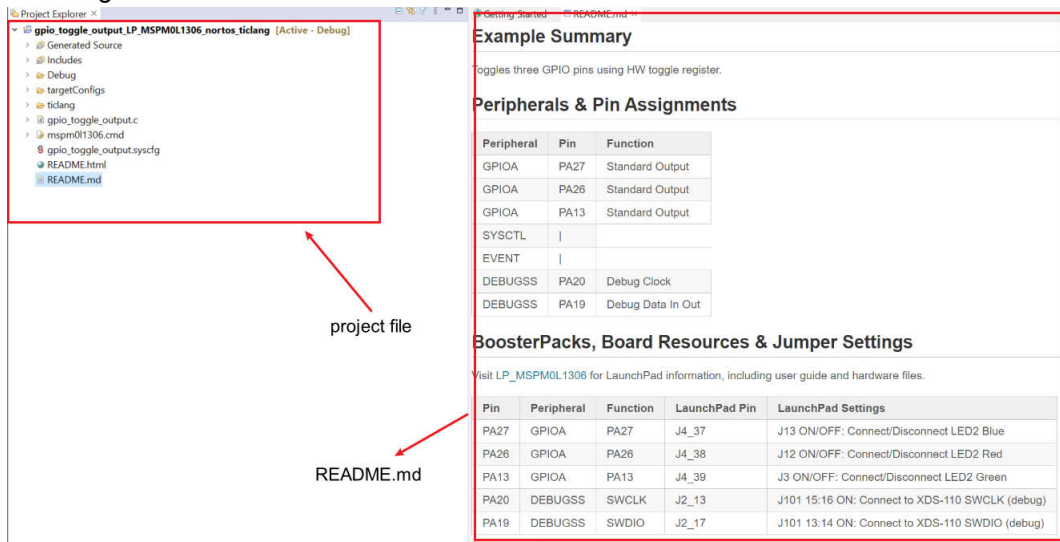
If the file cannot be imported, delete the same name project under workspace.



Delete project with same name under workspace

**Figure 2-32. Remove Duplicated Project**

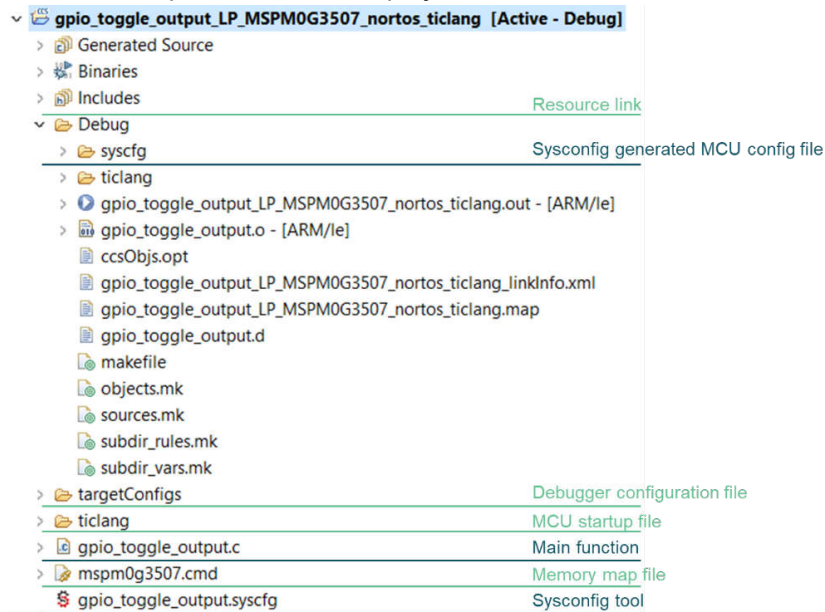
- After importation, there is going to be a project lying on the left, and a README.md will automatically open. It is recommended that you read the README.md file first, which contains the purpose of this example and the hardware configuration.



**Figure 2-33. Project and README.md**

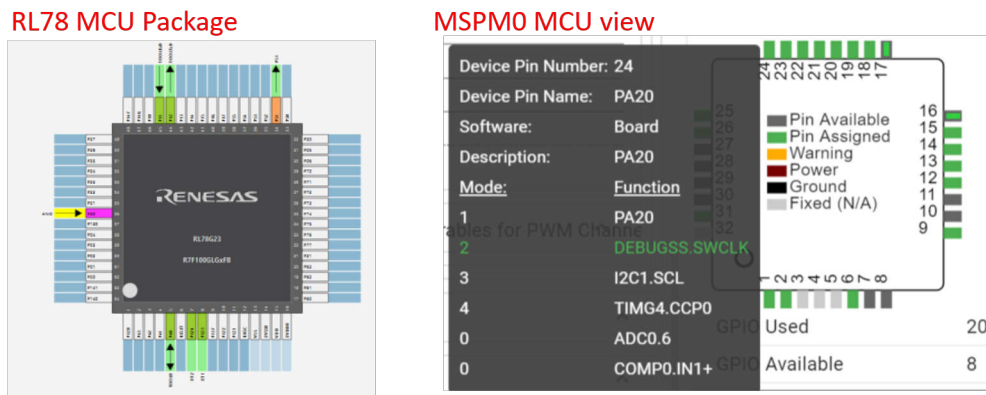


4. Figure 2-34 shows the most important files in the project.



**Figure 2-34. CCS Project Overview**

5. Just like in RL78 development, double-click the .sfcg file to reach the Smart Configuration interface, double click .syscfg file, you can reach to SysConfig, where you are allowed to configure the required peripherals through a graphical interface. And, it is suggested to use the MCU view of SysConfig to help you fix the pin function first with software engineer, which is similar to MCU/MPU Package in e<sup>2</sup>studio.



**Figure 2-35. MCU View in Smart Configuration and SysconfigSysConfig**

6. Based on the code and SysConfig example, you are able to polish the project or modify them with device-specific TRM or application note released on Ti.com.
7. If you want to add third-party libraries, you can follow the steps below. First, you have to add relevant file into your project, as shown if Figure 2-36.



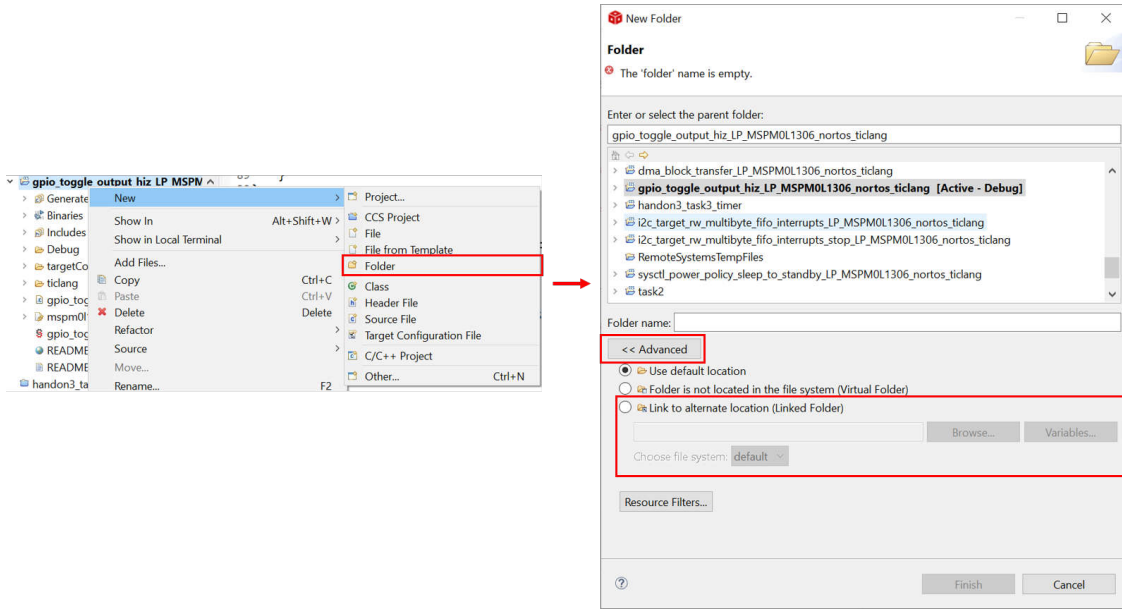


Figure 2-36. Add Relevant File

Then, other steps need to be done in order to tell compiler that you have add header files.

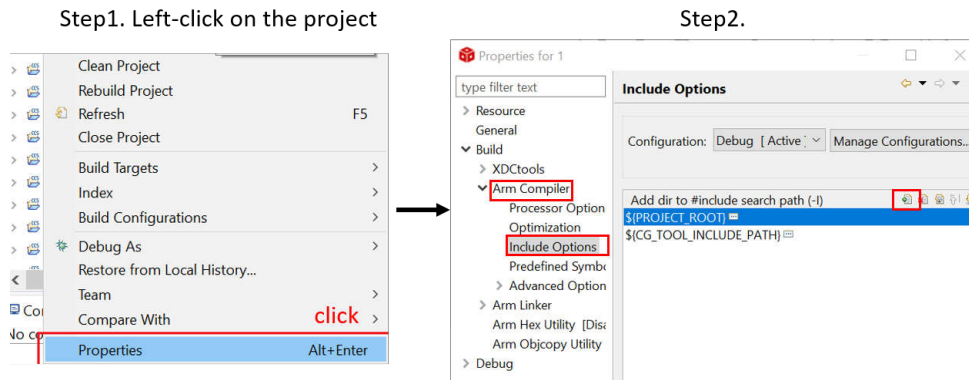


Figure 2-37. Include Options Set

- As you finish evaluating the software, click “build” icon in the main toolbar, as shown in Figure 2-38. The appearance of “Build Finished” shows your successful compilation.

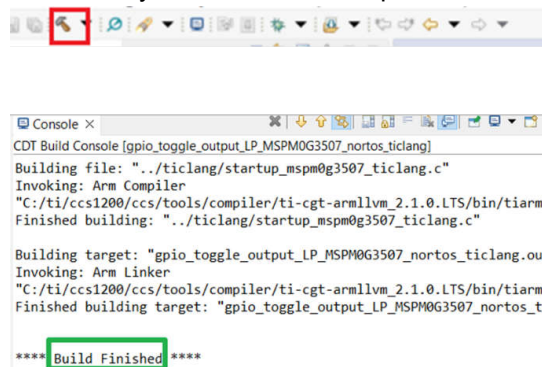
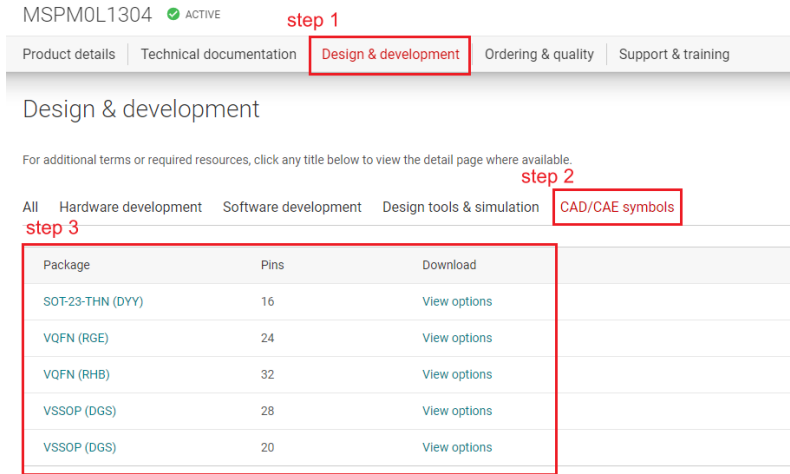


Figure 2-38. Successful Build

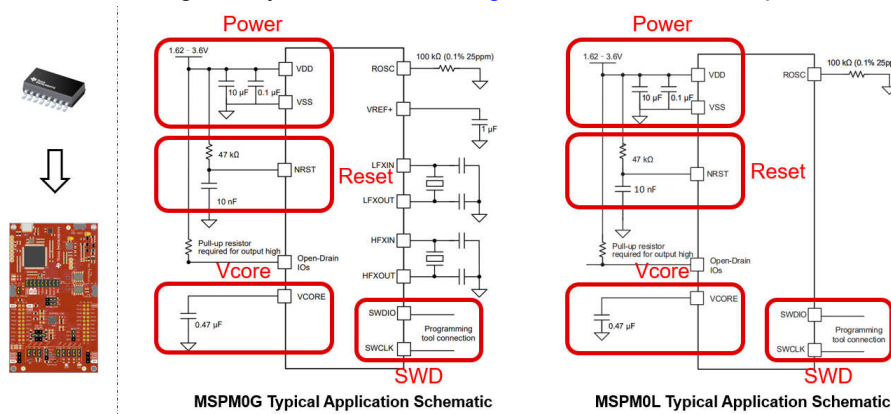
### 2.2.5 Step 5. PCB Board Design

1. To get the design package, go through the [Ti.com](https://www.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.



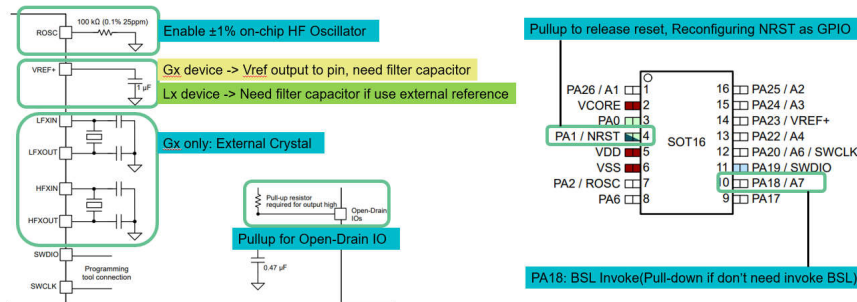
**Figure 2-39. Ultra Librarian Tool Entrance**

3. As for MSPM0 hardware design into your own board. [Figure 2-40](#) shows a sample minimal system design.



**Figure 2-40. MSPM0 Minimum System**

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



**Figure 2-41. 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.

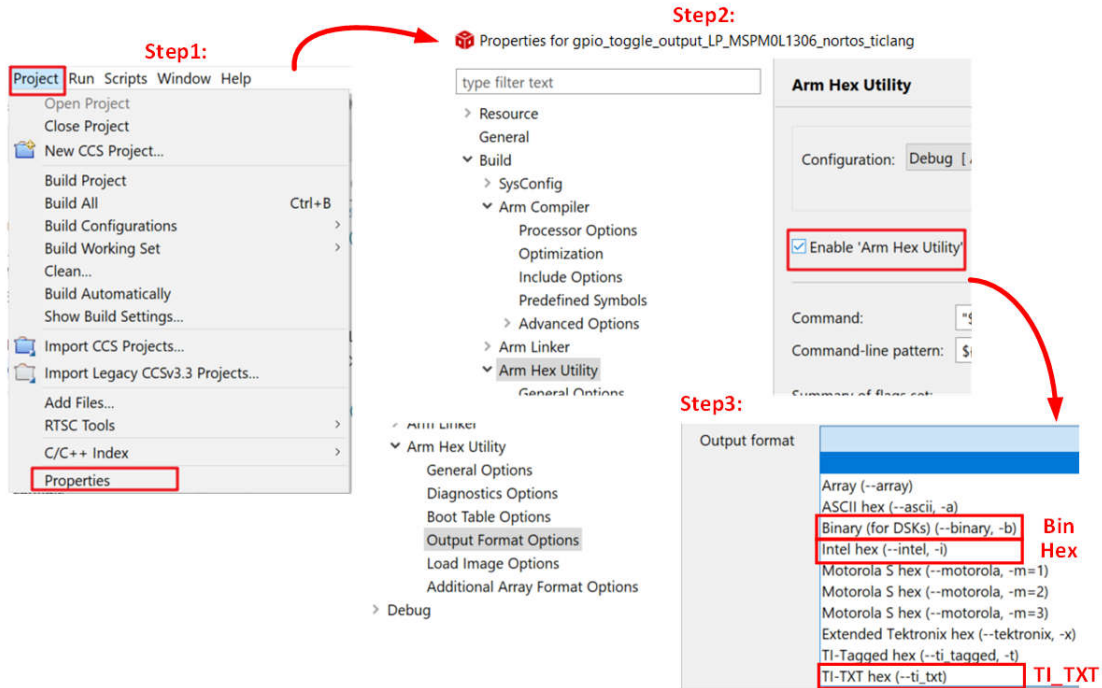


Figure 2-42. Create Program Files

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

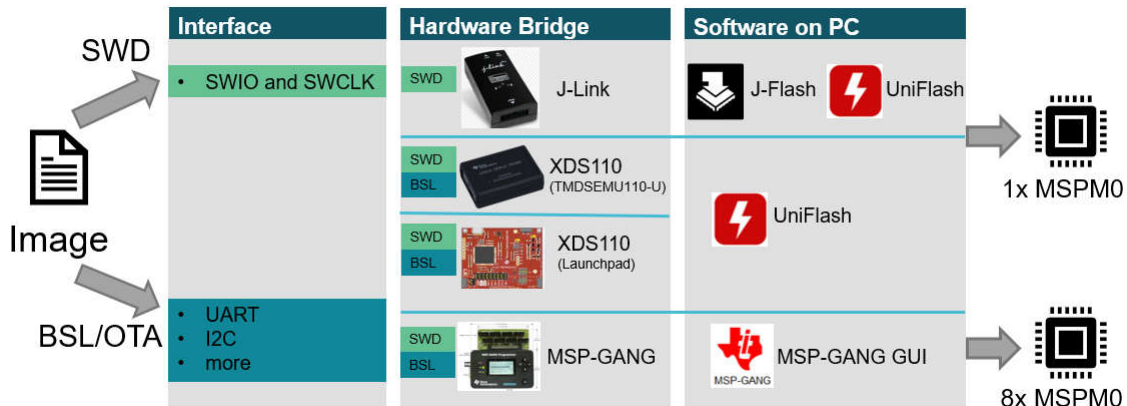


Figure 2-43. Program Software and Tool

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

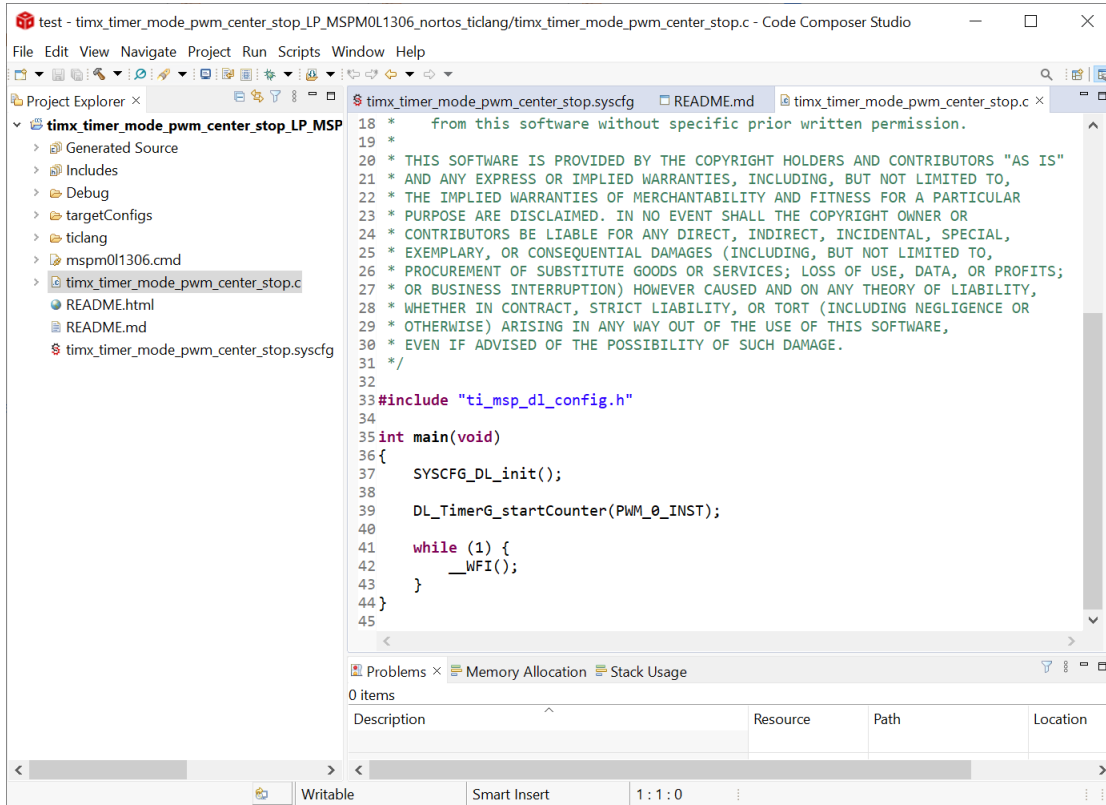
The MSPM0 design flow is shown below. This example aims to use PWM to drive LED.

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 RL78 and MSPM0, and choose the similar example in SDK.

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 it to the MSPM0 SDK example folder.



```

18 *   from this software without specific prior written permission.
19 *
20 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
21 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
22 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
23 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
24 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
25 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
26 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
27 * OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
28 * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
29 * OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
30 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
31 */
32
33#include "ti_msp_dl_config.h"
34
35int main(void)
36{
37    SYSCFG_DL_init();
38
39    DL_TimerG_startCounter(PWM_0_INST);
40
41    while (1) {
42        __WFI();
43    }
44}
45
    
```

**Figure 2-44. 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 2-45. 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.

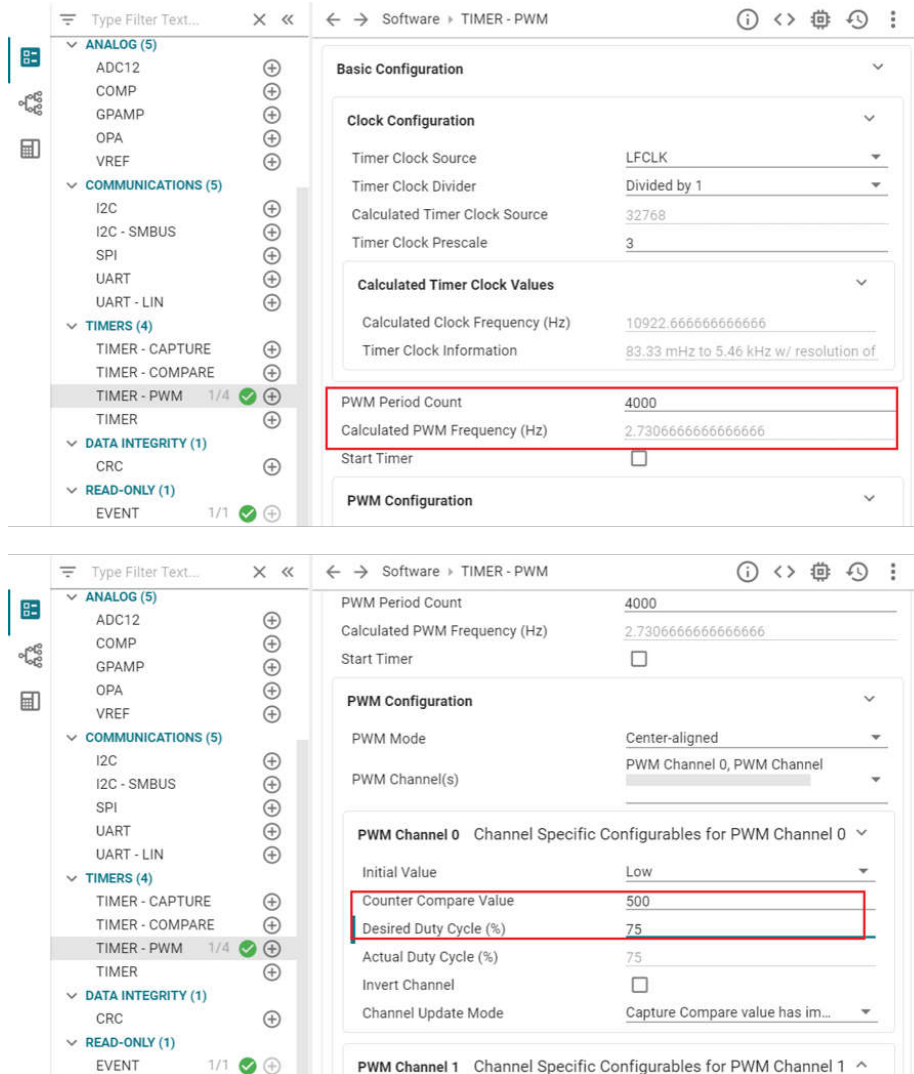


Figure 2-45. PWM Configuration in SysConfig

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

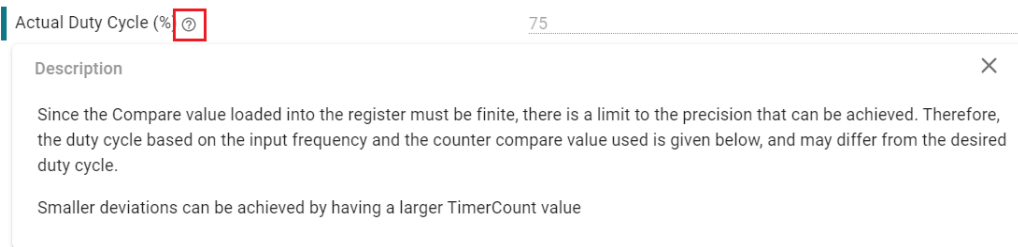


Figure 2-46. To Get Detailed Information of Each Item

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.

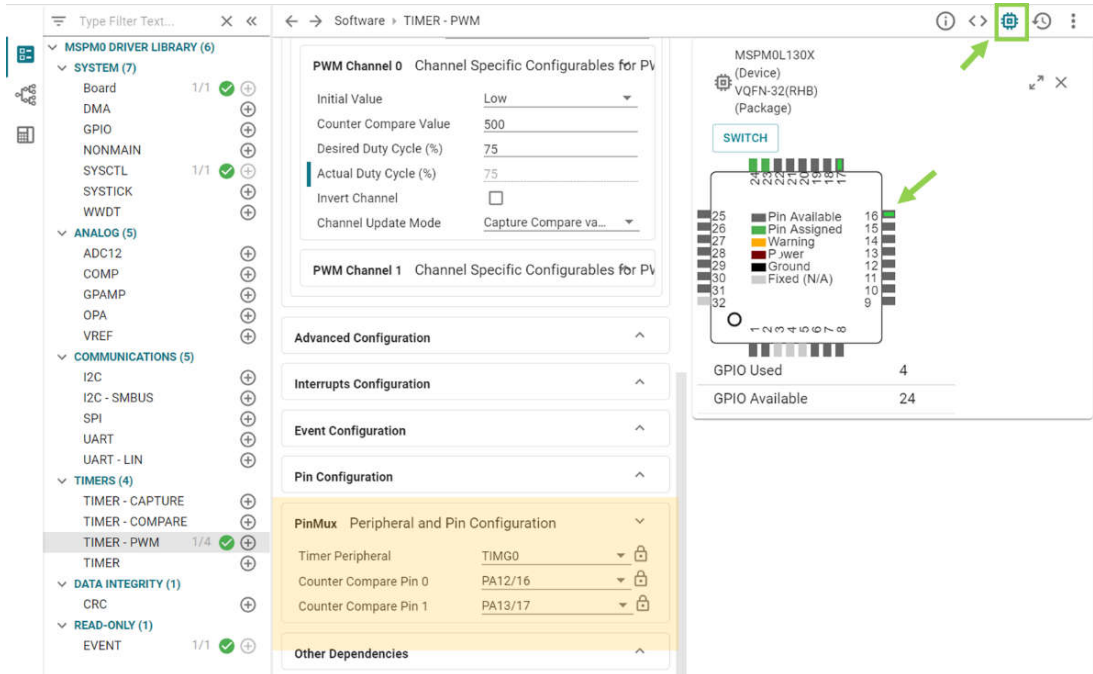


Figure 2-47. Pins Configuration

When the project is saved and rebuilt, SysConfig updates the files in Figure 2-48. At this point, the example hardware configuration has been modified to match the full functionality of the original software being ported.

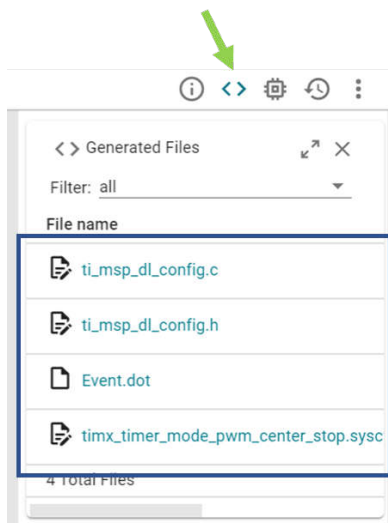


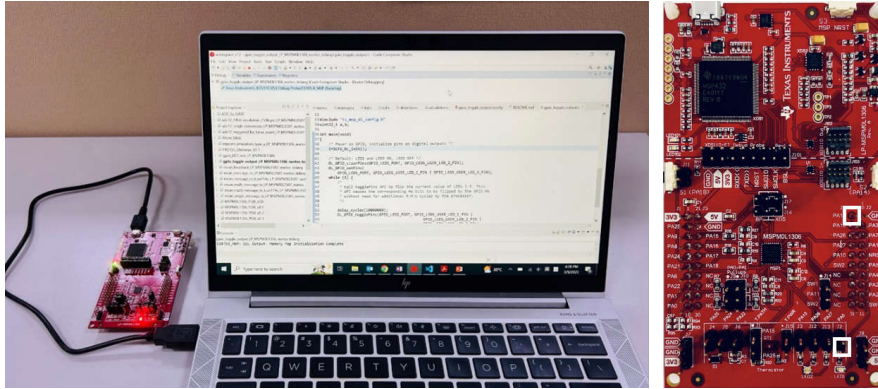
Figure 2-48. The files SysConfig Updates

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 2-49. 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();`

```

33#include "ti_msp_dl_config.h"
34
35int main(void)
36{
37    SYSCFG_DL_init();
38
39    DL_TimerG_startCounter(PWM_0_INST);
40    __BKPT();
41    while (1) {
42        __WFI();
43    }
44}
45

```

*Solution 1* (points to line 39)      *Solution 2* (points to line 40)

**Figure 2-50. 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 2-51](#). Go to the entrance of Ultra Librarian tool under MSPM0 device page (detailed can be seen in [Section 2.2.5](#)). Click *View options*. Select your wanted CAD format and Pin ordering, then you can get the Altium design lib file.

### step1

MSPM0L1306 ✔ ACTIVE

Product details | Technical documentation | **Design & development** | Ordering & quality | Support & training

## Design & development

For additional terms or required resources, click any title below to view the detail page where available.

All | Hardware development | Software development | Design tools & simulation | **CAD/CAE symbols**

Package	Pins	Download
SOT-23-THN (DYY)	16	<a href="#">View options</a>
VQFN (RGE)	24	<a href="#">View options</a>
VQFN (RHB)	32	<a href="#">View options</a>
VSSOP (DGS)	20	<a href="#">View options</a>
VSSOP (DGS)	28	<a href="#">View options</a>

### step2

Ultra Librarian | Texas Instruments - XMSM0L1306SDYYR | English

**Symbol**

Normal View

SOT\_06SDYYR 1

**Footprint**

Basic View

SOT\_06SDYYR\_TEX

**3D Model**

Choose CAD Formats & Download

### step3

Ultra Librarian | Texas Instruments - XMSM0L1306SDYYR | English

Choose CAD Format(s) Return to Previews

**3D CAD Model**

Altium

Altium Designer

PCAD v14

PCAD v15

Autodesk

Cadence

DesignSpark

KiCAD

Mentor

Pulsonix

Quadcept

TARGET 30011

Zuken

Symbol Pin Ordering: Sequential

Footprint Units: English (mil)

I have read and agree to the Ultra Librarian Terms And Conditions

进行人机身份验证

reCAPTCHA

Submit

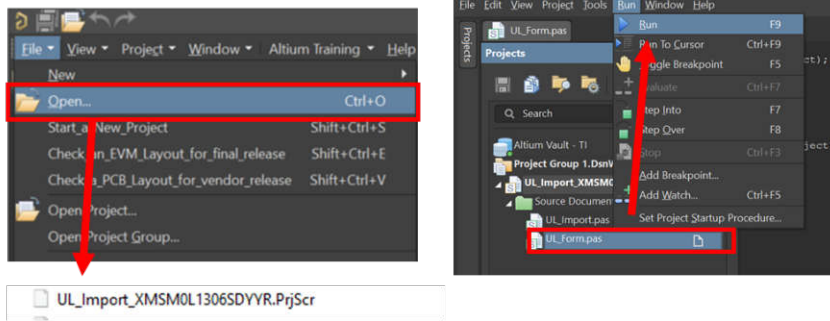
**Figure 2-51. Ultra Librarian Tool Download**



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 2-52.

Step 1. Open .PjScr file

Step 2. run UL\_Form.pas



Step 3. import file

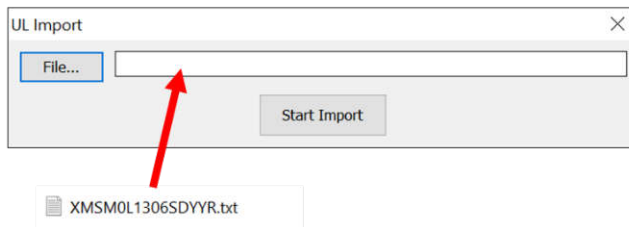


Figure 2-52. Run Altium Designer Script

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

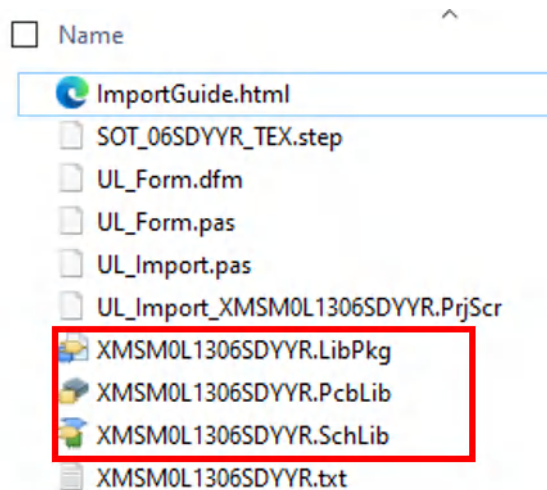
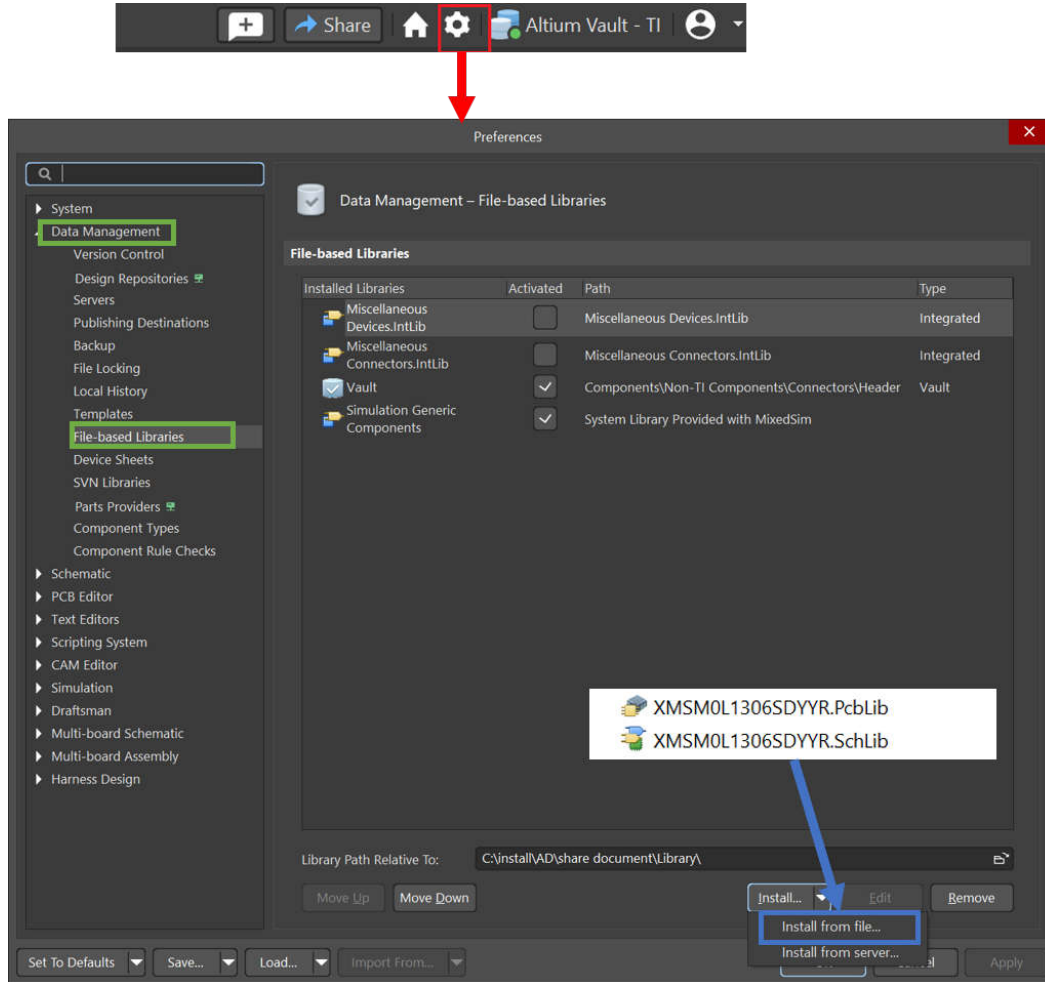


Figure 2-53. PCB Library and Schematic File

The final step is to import them into your AD lib, as shown in [Figure 2-54](#). And based on this, a schematic and PCB can be designed.



**Figure 2-54. 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 RL78 family is based on their own RL78 CPU core architecture. [Table 3-1](#) gives an overview of the general features of the CPU in the MSPM0 family compared to the RL78.

**Table 3-1. Comparison of CPU Feature Sets**

Features	RL78	MSPM0G	MSPM0L	MSPM0C
Architecture	Private RL78 core	Arm Cortex M0+	ARM Cortex M0+	Arm Cortex M0+
Instruction set	CISC	RISC	RISC	RISC
Pipeline	3-stage	2-stage	2-stage	2-stage
Operating Freq (Max)	40 MHz	80 MHz	32 MHz	24 MHz
DMA	Yes	Yes	Yes	Yes
Coremark/MHz	1.5952 <sup>(1)</sup>	2.39 <sup>(2)</sup>	2.39 <sup>(2)</sup>	2.39 <sup>(2)</sup>

(1) The score of RL78G23 is obtained under 32 MHz operating frequency and Renesas CC-RL V1.12 Compiler.

(2) The 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 Features

The MSPM0 and RL78 family of MCUs feature nonvolatile flash memory used for storing executable program code and application data. [Table 3-2](#) shows the comparison of flash features.

**Table 3-2. Comparison of Flash Features**

Features	RL78		MSPM0
Flash memory	Program Flash	RL78Gxx ranges 1 KB to 768 KB RL78Lxx ranges 8 KB to 256 KB RL78Ixx, RL78Hxx range 8 KB to 512 KB RL78Fxx ranges 8 KB to 512 KB	MSPM0Gxx ranges 32 KB to 128 KB MSPM0Lxx ranges 8 KB to 64 KB MSPM0Cxx 8 KB or 16 KB
	Data Flash	RL78Gxx ranges 0 to 8 KB RL78Lxx ranges 2 KB to 8 KB RL78Ixx, RL78Hxx range 0 KB to 4 KB RL78Fxx ranges 4 KB to 16 KB	
Single flash size	Program Flash	32 bits	64 bits
	Data Flash	32 bits or 8 bits	
Memory organization	Block size (512 B or 1 KB) Bank size (variable) Most devices-2banks I1C devices (512 KB)-3banks <sup>(1)</sup>		Sector size (1 KB) Bank size (variable) Device up to 256 KB-1bank Device with>256 KB-2banks
Access	8 bits or 16 bits		Single flash word (64 bits) or multiple words
Program mode	Program Flash	Single flash word (32 bits)	Single flash word (64 bits) or multiple words
	Data Flash	Single flash word (32 bits or 8 bits)	
Erase	Block erase		Sector erase Bank erase (up to 256 KB)
Error code correction	Supported (RL78F23, F24)		Supported
Write Protection	Yes		Yes, static and dynamic
Read Protection	Yes		Yes

**Table 3-2. Comparison of Flash Features (continued)**

Features	RL78	MSPM0
Cycles	1000k (TYP.)	100k (lower 32 KB) or 10k (above 32 KB)

- (1) Most RL78 devices have two banks (one code bank and one data bank), and some RL7811C devices whose code flash memory is 512 KB have three banks (two code banks and one data bank).

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.
- Inter programming voltage generation.

### 3.2.2 Flash Organization

The Flash memory is organized into one or more banks, and the memory in each bank 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 Memory Regions

Table 3-3 shows the flash memory regions of RL78 devices and MSPM0 devices.

**Table 3-3. The Comparison of Flash Memory Regions**

RL78 (code Flash)		MSPM0	
Program area		MAIN	Application code and data
Boot cluster <sup>(1)</sup>	Small Program area	NONMAIN	BCR configuration
	On-chip debug security ID setting		BSL configuration
	Flash serial programming security ID		
	Option byte		
	CALLT table	FACTORY	Device ID, and so forth
	Vector table	DATA <sup>(2)</sup>	Data or EEPROM emulation

- (1) Some RL78 devices which have two boot clusters (cluster 0 and cluster 1) can implement the boot swapping function.  
 (2) 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 their 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.

#### 3.2.2.3 Flash Memory Registers of RL78

The Flash memory registers in the special function register area (SFR and 2nd SFR) are used to control flash memory programming. For example, the FLPMC register controls enable or disable programming of code and data flash memory and the DFLCTL register is used to enable or disable accessing to the data flash.

### 3.2.3 Embedded SRAM

The MSPM0 and RL78 family of MCUs feature SRAM used for storing application data.

**Table 3-4. Comparison of SRAM Features**

Features	RL78	MSPM0
<b>SRAM memory</b> <sup>(1)</sup>	RL78Gxx ranges 0.1 KB to 48 KB RL78Lxx ranges 1 KB to 16 KB RL78Ixx, RL78Hxx range 0.7 KB to 32 KB RL78Fxx ranges 0.5 KB to 32 KB	MSPM0Gxx 16 KB to 32 KB MSPM0Lxx 2 KB to 4 KB MSPM0Cxx 1 KB
<b>Parity check</b>	Supported	MSPM0Gxx: supported MSPM0Lxx: supported MSPM0Cxx: not supported
<b>ECC</b>	Supported (RL78F13, F14, F15, F23, F24)	MSPM0Gxx: supported MSPM0Lxx: supported MSPM0Cxx: not supported
<b>Write protection (RAM guard)</b>	Yes	Yes

(1) A specific area in SRAM of RL78 is used for General-purpose registers.

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 RL78 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**

RL78		MSPM0	
POR (Power-On Reset Circuit) <sup>(1)</sup>	Rise detection: $V_{DD} > V_{POR}$ , POR reset signal is released Fall detection: $V_{DD} < V_{PDR}$ , POR reset signal is generated	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
LVD (Voltage Detector)-Reset mode	Rise detection: $V_{DD} > V_{LVD}$ , LVD reset signal is released Fall detection: $V_{DD} < V_{LVD}$ , LVD reset signal is generated	Brownout Reset (BOR)- 0 level <sup>(2)</sup>	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.
LVD (Voltage Detector)-Interrupt and reset mode	Rise detection: $V_{DD} > V_{LVDH}$ , LVD reset signal is released Fall detection: 1) $V_{DD} < V_{LVDH}$ : an interrupt request signal is generated 2) $V_{DD} < V_{LVDL}$ : LVD reset signal is generated	Brownout Reset (BOR)- 1 to 3 level <sup>(2)</sup>	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

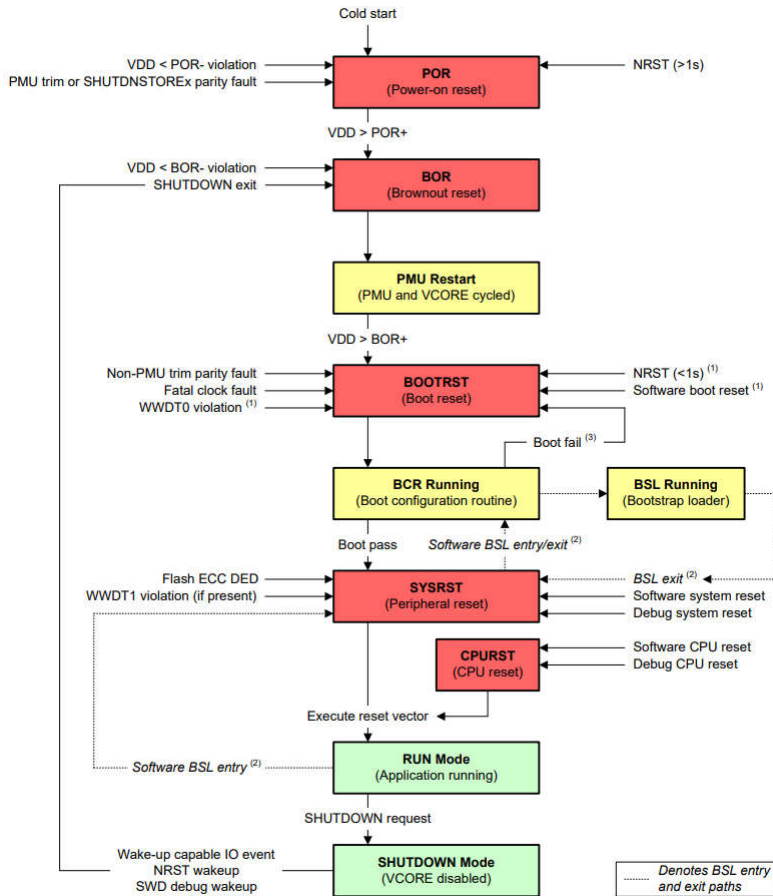
**Table 3-5. Summary and Comparison of Power Up (continued)**

RL78		MSPM0	
LVD (Voltage Detector)-Interrupt mode	Rise detection: $V_{DD} > V_{LVD}$ . LVD reset signal is released After the LVD reset is released, an interrupt request signal is generated when $V_{DD} > V_{LVD}$ or $V_{DD} < V_{LVD}$	N/A	N/A
RTCPOR (RTC Power-on Reset)	Reset of the RTC and XT1 oscillator by comparison of supply voltage of the RTCPOR circuit and detection voltage	RTC and associated clocks are reset through BOOTRST, BOR, or POR	

- (1) Some RL78 devices have SPOR (Selectable Power-On Reset Circuit) whose detection level for the power supply detection can be selected by using the option byte.
- (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.

The relationship between various voltage thresholds of RL78 is:  $V_{PDR} < V_{POR} < \text{Low limit of operation voltage} < V_{LVLDL} < V_{LVLDH}$ . The relationship between various voltage thresholds of MSPM0 is:  $POR- < POR+ < BOR0- < BOR0+$ , and  $BOR0+$  is the specified  $V_{DD}$  minimum to enable correct operation of internal circuits.

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

RL78 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 RL78 and MSPM0 devices. Note that not all the devices have all types clock sources. For details, see the device-specific data sheet.

**Table 3-6. Oscillator Comparison**

Type	RL78	MSPM0L	MSPM0G	MSPM0C
<b>Internal oscillators</b>	$f_{HOCO}/f_{IH}$ : internal high-speed oscillator up to 40 MHz, which can source the PLL for 80 MHz	SYSOSC: internal oscillator from 4 MHz to 32 MHz	SYSOSC: internal oscillator from 4 MHz to 32 MHz, which can source the PLL for 80 MHz	SYSOSC : internal 24 MHz oscillator
	$f_{IM}$ : internal middle-speed oscillator up to 4 MHz			
	$f_{IL}$ : internal low-speed oscillator (15 kHz or 32.768 kHz)	LFOSC: internal 32 kHz low-frequency oscillator	LFOSC: internal 32 kHz low-frequency oscillator	LFOSC: internal 32 kHz low-frequency oscillator
	$f_{WDT}$ : internal oscillator for WDT (15 kHz)	Not available	Not available	Not available
<b>External oscillators</b>	$f_X$ : high-frequency oscillator up to 20 MHz which can source the PLL for 80 MHz	Not available	HFXT: external high frequency oscillator from 4 MHz to 48 MHz, which can source the PLL for 80 MHz	Not available
	$f_{XT}$ : low-frequency oscillator (32.768 kHz or 38.4 kHz)	Not available	LFXT: external 32 kHz low-frequency oscillator	Not available

##### 3.4.1.1 MSPM0 Oscillators

MSPM0 devices have many types of clock sources including both internal and external for low cost and low consumption. Table 3-7 lists the clock sources in MSPM0 devices. Note that not all the devices have all type clock sources. For details, see the device-specific data sheet.

**Table 3-7. Oscillators in MSPM0 MCUs**

Type	Clock Sources	Description
<b>Internal</b>	SYSOSC	System oscillator (4 or 32 MHz factory-trimmed frequencies, 16 or 24 MHz user-trimmed frequencies)
	LFSOSC	Low frequency oscillator (32 kHz typical frequency)
	SYSPLL <sup>(1)</sup>	System PLL with programmable frequency
<b>External</b>	LFXT	Low-frequency low-power crystal oscillator (32 kHz typical frequency)
	HFXT	High-frequency crystal oscillator (4 to 48 MHz typical frequency)

(1) Only the MSPM0Gx devices have the SYSPLL oscillator.

#### 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 3-8. Clock Signal Comparison**

Clock Description	RL78 Clock	MSPM0 Clock
<b>External digital clock input</b>	<b>High frequency</b>	EXCLK ( $f_{EX}$ )
	<b>Low frequency</b>	EXCLKS ( $f_{EXS}$ )
<b>High-frequency external clock</b>	$f_{MX}$	HFCLK
<b>Low-frequency external clock</b>	$f_{SUB}$ <sup>(1)</sup>	Selection of LFCLK_IN and LFXT
<b>PLL circuit output clock</b>	$f_{PLL}$	SYSPLLCLK0, SYSPLLCLK1, SYSPLLCLK2x <sup>(2)</sup>



**Table 3-8. Clock Signal Comparison (continued)**

Clock Description	RL78 Clock	MSPM0 Clock
Main system clock	$f_{\text{MAIN}}$ <sup>(3)</sup>	MCLK, ULPCLK <sup>(4)</sup> (BUSCLK)
High-frequency clock for CPU/peripherals	$f_{\text{MAIN}}$ or $f_{\text{MP}/n}$ <sup>(5)</sup>	Selection of HSCLK <sup>(6)</sup> and SYSOSC
Low-frequency clock for CPU/peripherals	$f_{\text{SL}}$	LFCLK (fixed 32 kHz)
Source CPU	$f_{\text{CLK}}$	CPUCLK
Clock for most peripheral hardware	$f_{\text{CLK}}$	MCLK, ULPCLK
Available clock for high-speed peripherals	$f_{\text{MX}}$ , $f_{\text{IH}}$ , $f_{\text{MAIN}}$ , $f_{\text{PLL}}$ , $f_{\text{MP}}$ , $f_{\text{CLK}}$	MCLK
Available clock for low-speed low-power peripherals	$f_{\text{SUB}}$ , $f_{\text{IL}}$ , $f_{\text{SL}}$ , $f_{\text{CLK}}$	ULPCLK
Fixed frequency clock	N/A	MFCLK: 4 MHz, synchronized to MCLK MFPCLK: 4 MHz

- (1)  $f_{\text{SUB}}$  is the subsystem clock which can be sourced from low-frequency external oscillator ( $f_{\text{XT}}$ ) or the low-frequency external digital clock input (EXCLKS).
- (2) SYSPLLCLK2x is twice the speed of the output of the PLL and can be divided down.
- (3) The main system clock of RL78 is sourced from  $f_{\text{MX}}$  or  $f_{\text{HOCO}}/f_{\text{IH}}$ .
- (4) The main system clock of MSPM0 is sourced from LFCLK, HSCLK, or SYSOSC. The MCLK is the main system clock for PD1 and the ULPCLK, derived from MCLK, is the main system clock for PD0.
- (5) The  $f_{\text{MP}}$  is the main system/PLL selection clock, and n can be selected as 1, 2, 4,..... Most RL78 devices select  $f_{\text{MAIN}}$  as the high-frequency clock for CPU/peripherals, and some RL78 devices select  $f_{\text{MP}}/n$  as the high-frequency clock for CPU/peripherals.
- (6) The HSCLK is sourced from SYSPLL or HFCLK.

**Table 3-9. Peripheral Clock Sources**

Peripheral	RL78	MSPM0
Real-time clock (RTC)	$f_{\text{IH}}$ , $f_{\text{IL}}$ , $f_{\text{MX}}$ , $f_{\text{SUB}}$ , $f_{\text{CLK}}$	LFCLK (LFOSC, LFXT)
UART	$f_{\text{CLK}}$	BUSCLK, MFCLK, LFCLK
SPI/CSI (simplified SPI)	$f_{\text{CLK}}$	BUSCLK, MFCLK, LFCLK
I2C	$f_{\text{CLK}}$	BUSCLK, MFCLK
CAN	$f_{\text{MX}}$ , $f_{\text{MP}}$ , $f_{\text{CLK}}$	PLLCLK1, HFCLK
ADC	$f_{\text{CLK}}$	ULPCLK, HFCLK, SYSOSC
TIMERS	$f_{\text{HOCO}}$ , $f_{\text{IL}}$ , $f_{\text{MX}}$ , $f_{\text{SL}}$ , $f_{\text{PLL}}$ , $f_{\text{MP}}$ , $f_{\text{CLK}}$ , $f_{\text{TMKB2}}$ <sup>(1)</sup>	BUSCLK, MFCLK, LFCLK
COMPARATOR	$f_{\text{PLL}}$ , $f_{\text{CLK}}$	ULPCLK

- (1)  $f_{\text{TMKB2}}$  clock is only used to source the 16-bit timers of RL78 MCU.

### 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, STANDBY, 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 3-10 gives a brief comparison between RL78 and MSPM0 devices.

**Table 3-10. Operating Modes Comparison Between RL78 and MSPM0 Devices**

RL78			MSPM0		
Operation Mode	Description		Operation Mode	Description	
MAIN RUN	CPU operates on main system clock <sup>(1)</sup>	CPU, clock, and peripherals work	RUN	0	MCLK and CPUCLK run from a fast clock source (SYSOSC, HFCLK, or SYSPLL)
	CPU operates on subsystem clock	CPU, clock, and peripherals work		1	MCLK and CPUCLK run from LFCLK (at 32 kHz).
				2	
HALT	CPU operates on main system clock <sup>(1)</sup>	CPU operation stops. Operation of main system clock continues. Status of subsystem clock is retained. Most peripheral functions can operate.	SLEEP	0	CPU operation stops. SYSOSC remains enable and other high-speed oscillators are optional. Low-speed oscillators remains enable. MCLK run from a fast clock source.
	N/A	N/A		1	CPU operation stops. SYSOSC remains enable and other high-speed oscillators are disable. Low-speed oscillators remains enable. MCLK run from LFCLK.
	CPU operates on subsystem clock	CPU operation stops. Operation of main system clock stops. Operation of subsystem clock continues. Most peripheral functions can operate.		2	CPU operation stops. High-speed oscillators are disable. Low-speed oscillators remains enable. MCLK run from LFCLK.
SNOOZE <sup>(2), (3)</sup>	CPU operation stops. $f_{HOCO}/f_{IH}$ operation starts, $f_X, f_{EX}$ and $f_{PLL}$ operation stop. The status of subsystem clock while used in the STOP mode continues. Peripheral functions such as ADC, UART or CSI can operate without operating the CPU.		STOP	0	CPU operation stops. Status of SYSOSC is retained. Other high-speed oscillators are disable. Low-speed oscillators remains enable. ULPCLK is limited to 4 MHz. 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 4 MHz
	N/A			2	CPU operation stops. High-speed oscillators are disable. ULPCLK runs at 32 kHz. PD0 is enabled and PD1 is disabled. The use of ADC is not supported.
N/A	N/A		STANDBY	0	CPU operation stops. High-speed oscillators are disable. All PD0 peripherals receive the ULPCLK and LFCLK. ADC is not supported.
				1	Similar to STANDBY0, with only TIMG0/1 receiving ULPCLK or LFCLK.
STOP <sup>(3)</sup>	CPU operation stops. Operation of main system clock stops. The status of subsystem clock before STOP mode was set is retained. The whole system stops.		SHUTDOWN	No clocks are available and device is shut down.	

(1) CPU can operate on  $f_{IH}/f_{HOCO}$ ,  $f_X$ ,  $f_{EX}$  or  $f_{PLL}$ .

(2) The SNOOZE mode can only be specified when the high-speed on-chip oscillator is selected for the CPU/peripheral hardware clock ( $f_{CLK}$ ).

(3) The SNOOZE mode can only be specified for CSI, UART and the A/D converter, and so forth. In the case of CSI or UART data reception, an A/D conversion request by the timer trigger signal, and so forth, MCU is gear shifted from the STOP mode to the SNOOZE mode. Then the CSI or UART data is received without operating the CPU, A/D conversion is performed, and so forth.

### 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, see the “Supported Functionality by Operating Mode” table found in the MSPM0 device-specific data sheet, for example:

- [MSPM0G350x Mixed-Signal Microcontrollers data sheet](#)
- [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 and/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, and RTC; 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 as well as peripheral support and purpose, see the appropriate chapter in the MSPM0 device-specific TRMs.

- [MSPM0 G-Series 80-MHz Microcontrollers Technical Reference Manual](#)
- [MSPM0 L-Series 32-MHz Microcontrollers Technical Reference Manual](#)
- [MSPM0 C-Series 24-MHz 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 RL78 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 3-11](#). 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 programmable priority.

**Table 3-11. Interrupts Comparison**

Features	RL78	MSPM0x
Interrupt Types	Maskable: determined by devices and divided into internal interrupt and external interrupt.	Peripheral interrupts: NVIC supports up to 32 native peripheral interrupt sources <sup>(1)</sup> .
	Reset: determined by devices	Reset, NMI, Hard Fault, SVCcall, PendSV, SysTick
Priority Level	The default priority level: determined by devices <sup>(2)</sup>	The default priority level: NVIC Number <sup>(3)</sup>
	The maskable interrupts have 4 programmable priority levels: 0, 1, 2, 3	System exceptions (Reset, NMI, Hard Fault) have fixed priority levels of -3, -2, and -1
		The peripheral interrupts have 4 programmable priority levels: 0, 64, 128, 192

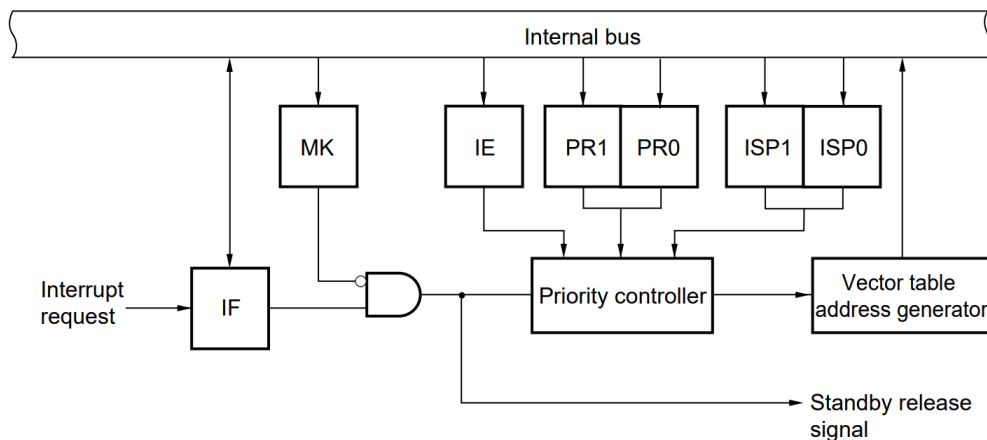
**Table 3-11. Interrupts Comparison (continued)**

Features	RL78	MSPM0x
Priority Set	PR0xy and PR1xy registers: used to set the maskable interrupt priority level	IPRx registers in the NVIC: used to set the peripheral interrupt priority level
Interrupt mask	MKxy registers: used to enable/disable the corresponding maskable interrupt	IMASK register in the peripheral side: used to configure which interrupt conditions propagate into an event <sup>(4)</sup>
		ISER and ICER register in the NVIC: used to enable or disable the peripheral interrupts

- (1) 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 than 32 peripheral interrupts to the NVIC.
- (2) The default priority indicates the relative interrupt priority if multiple maskable interrupts have the same programmable priority.
- (3) The NVIC number indicates the relative interrupt priority if multiple NVIC interrupts have the same programmable priority.
- (4) The event handler and related management registers of MSPM0 are shown in [Section 3.6.2](#).

**3.6.1.1 Interrupt Management of RL78**

RL78 devices set the priority level of each interrupt condition through the PR0xy and PR1xy registers and enable/disable an interrupt condition through the MKxy registers. For example, [Figure 3-2](#) shows the internal maskable interrupt hierarchy of RL78. Before each interrupt request is acknowledged, the EI instruction must always be issued to set IE=1 to enable interrupt request acknowledgment.



**Figure 3-2. Internal Maskable Interrupt Hierarchy of RL78**

**3.6.1.2 Interrupt Management of MSPM0**

Unlike RL78 devices, 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 3-3 shows the peripheral interrupt hierarchy.

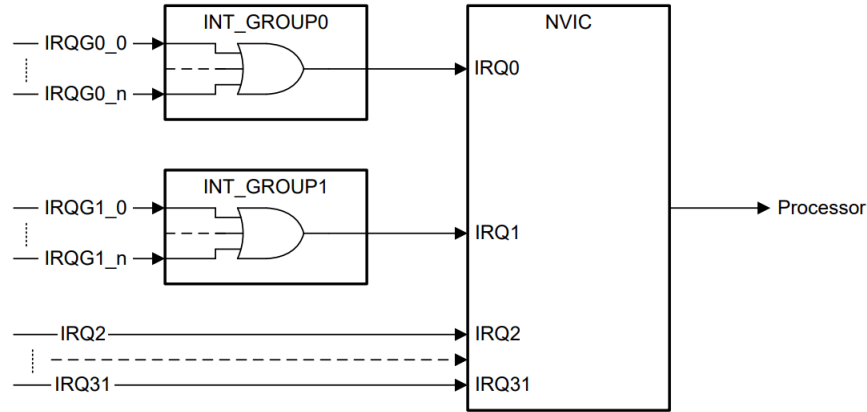


Figure 3-3. Peripheral Interrupt Hierarchy of MSPM0

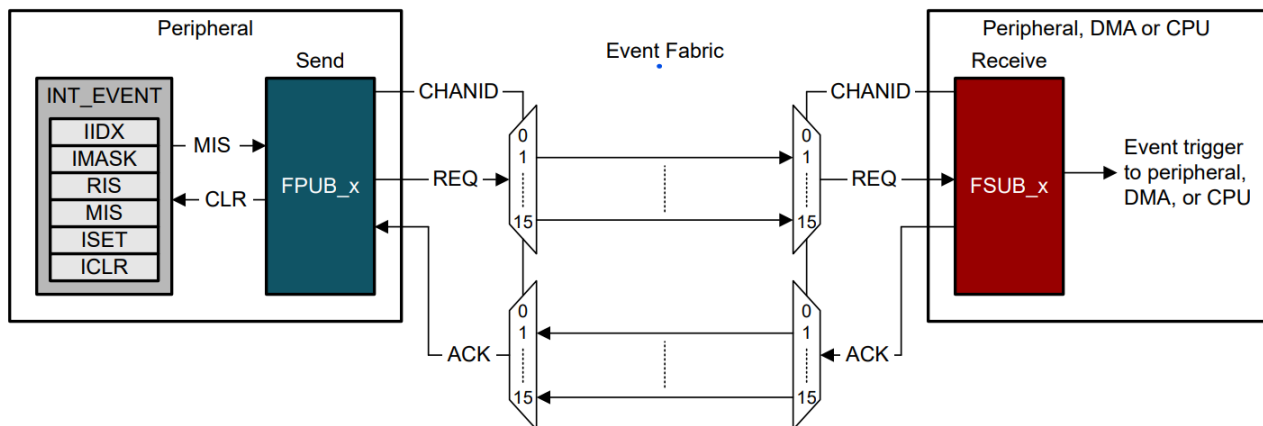
### 3.6.2 Event Handler of MSPM0

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 manager 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.



The event management register set contains 6 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 3-4. Generic Event Route

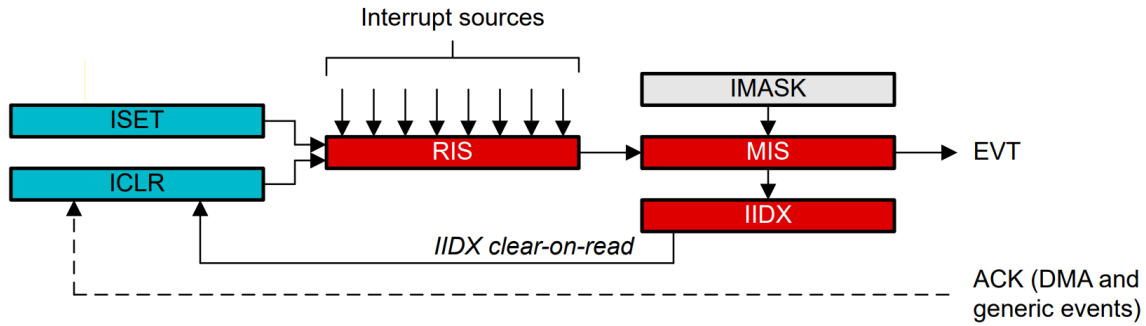


Figure 3-5. Event Management Register Relationship

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 [MSPM0G technical reference manual](#), the [MSPM0L technical reference manual](#), or the [MSPM0C technical reference manual](#).

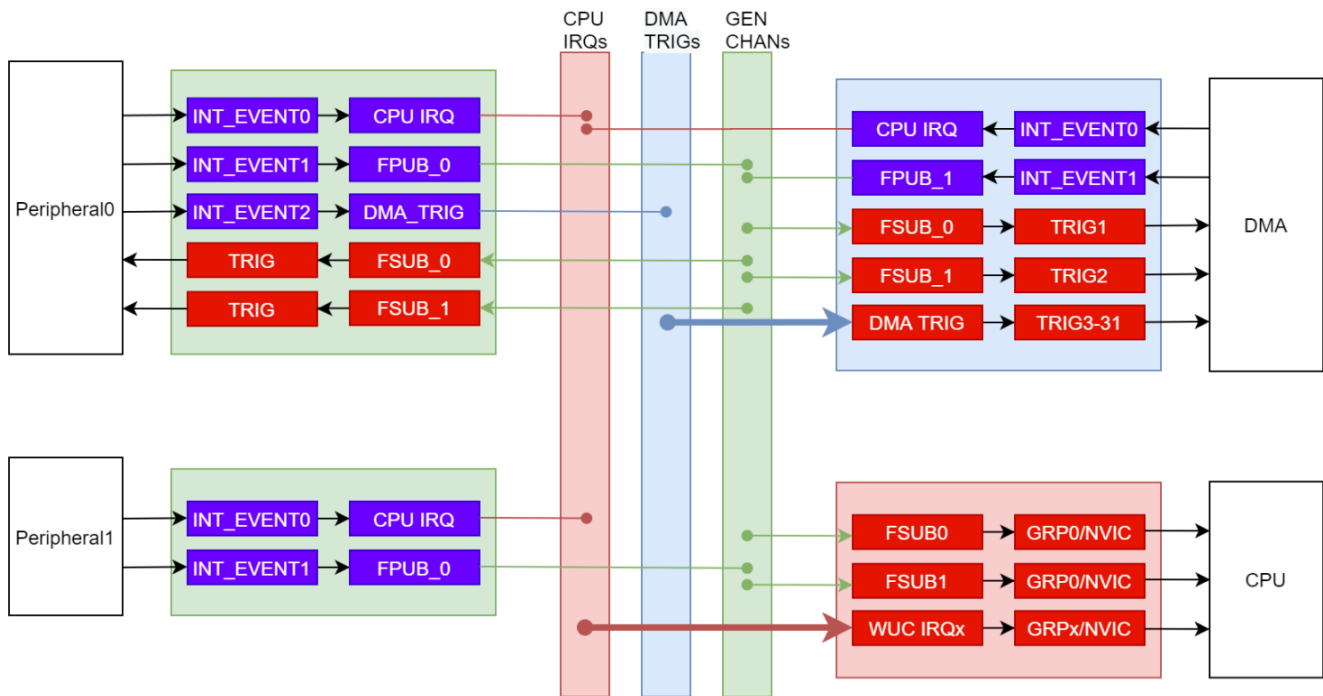
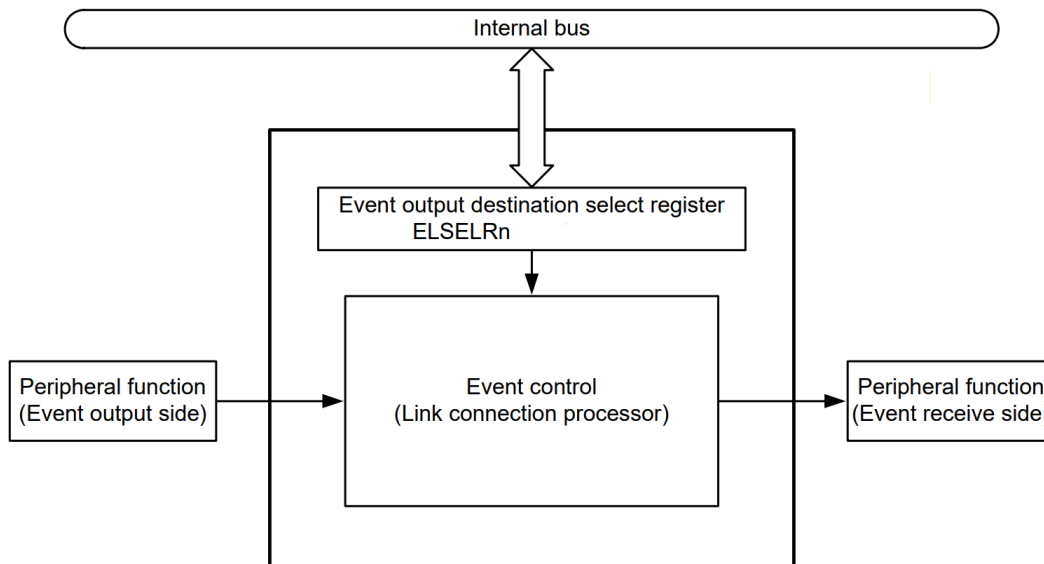


Figure 3-6. MSPM0 Event and Interrupt Handling

### 3.6.3 Event Link Controller (ELC) of RL78

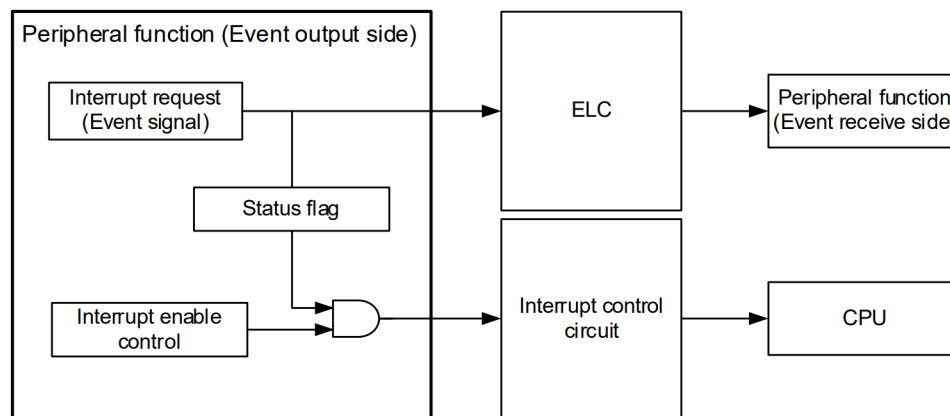
Some RL78 MCUs have an event link controller (ELC) that mutually connects (links) events output from each peripheral function. By linking events, RL78 MCUs can coordinate operation between peripheral functions directly without going through the CPU.

Figure 3-7 shows the ELC block diagram. The ELSELN register (some devices use the ELISELN and ELOSELN registers) links each event signal to an operation of an event-receiving peripheral function (link destination peripheral function) after reception. Different ELSELN registers represent different event generator and the values set to ELSELN registers determine the operation of link destination peripheral functions.



**Figure 3-7. RL78 Event Link Controller**

Figure 3-8 shows the relationship between interrupt handling and ELC. The path for using an event signal generated by a peripheral function as an interrupt request to the interrupt control circuit is independent from the path for using the signal as an ELC event. Therefore, each event signal can be used as an event signal for operation of an event-receiving peripheral function, regardless of interrupt control.



**Figure 3-8. RL78 Event and Interrupt Handling**

### 3.6.4 Event Management Comparison

RL78 and MSPM0 have different event management structures and features. The comparison of [Section 3.6.2](#) and [Section 3.6.3](#) is shown in [Table 3-12](#).

**Table 3-12. Event Management Comparison**

Features		RL78	MSPM0
Publisher		Peripheral	peripheral
Subscriber		CPU, DMA trigger, peripheral	CPU, DMA trigger, peripheral
Management Style	peripheral → CPU	Interrupt control circuit	Event manager
	peripheral → DMA	DMA controller <sup>(1)</sup>	
	peripheral → peripheral	Event Link Controller <sup>(1)</sup>	
Route Type		point-to-point (1:1)	point-to-point
			point-to-two (splitter) <sup>(2)</sup>

(1) Not all RL78 devices have both DMA controller and Event Link Controller and Some RL78 devices have neither A nor B.

(2) 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 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, RL78 devices do not have SWD 2-wire JTAG port. RL78 devices use the dedicated flash memory programmer (PG-FP5/6, E1/E2/E2 Lite/E20 on-chip debugging emulator) for debugging and programming. And communication between the programmer and the RL78 microcontroller is established by serial communication using the TOOL0 pin via a dedicated single-line UART of the RL78 microcontroller.

### 3.7.2 Programming Mode Comparison

#### 3.7.2.1 Bootstrap Loader (BSL) Programming of MSPM0

The bootstrap loader (BSL) programming interface is an alternative programming interface to the Arm SWD. 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, the BSL programming interface can also be utilized for initial production programming as well.

#### 3.7.2.2 Serial Programming (Using External Device) of RL78

The serial programming mode of RL78 allows for firmware updates by using the RL78 microcontroller and an external device (a microcontroller or ASIC) connected to a UART. Processing to write data to or delete data from the RL78 microcontroller by using an external device is performed on-board. Off-board writing is not possible. [Table 3-13](#) shows a comparison of the programming mode between MSPM0 and RL78 device families.

**Table 3-13. Programming Mode Comparison**

Programming Features	RL78	MSPM0
Type	Serial programming using external device	Bootstrap loader
Security	Memory security and access restriction options <sup>(1)</sup>	Secure boot options: CRC protections
Customizable	No	Yes, configurable invoke pin and plug-in feature
Invoke methods	TOOL0 pin high after reset release	1 pin high at BOOTRST, SW entry



**Table 3-13. Programming Mode Comparison (continued)**

Programming Features	RL78	MSPM0
<b>Command</b>	Erase, Write, Getting information, Security, and so forth. <sup>(1)</sup>	Connection, Unlock, Erase, Write, Memory read back, Factory Reset, Get info, and so forth.
<b>Password protection</b>	No	Yes
<b>Interfaces supported</b>	Dedicated UART	UART, I2C, SPI (Custom plug-in needed), Can (Plug-in planned)

(1) The block erase, write commands, and rewriting boot cluster are enable by the default setting. Disabling block erase, disabling write and disabling rewriting boot cluster can be performed using the Security Set command.

In addition, RL78 devices supports a self-programming mode that can be used to rewrite the Flash memory via a user program. Because this mode allows a user application to rewrite the flash memory by using the RL78 microcontroller self-programming library.

## 4 Digital Peripheral Comparison

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

MSPM0 GPIO functionality covers all the features provided by RL78G. RL78 uses the term Pin Functions and Port Function to refer to all the functionality responsible for managing the device pins, generating interrupts, etc. 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 RL78 Port Function and Pin Function. Additionally, MSPM0 offers functionality not available in RL78 devices such as DMA connectivity, controllable input filtering and event capabilities.

**Table 4-1. GPIO Feature Comparison**

Feature	RL78	MSPM0
<b>Output modes</b>	Pullup Open drain with N-ch	Push-pull with pullup or pulldown Open drain with pulldown Hi-Z
<b>Input modes</b>	Pullup Input threshold level CMOS or TTL input buffer Analog	Floating Pullup or pulldown Analog
<b>GPIO speed selection</b>	No	MSPM0 offers Standard IO (SDIO) on all IO pins. MSPM0 High-Speed IO (HSIO) is available on select pins.
<b>High-drive GPIO</b>	Depending on the Port and Chip type, the maximum is approximately 56 mA @Vdd=5 V, and 13.3 mA @Vdd=3.3 V	Approximately 20 mA @Vdd=3.3V, called High Drive IO (HDIO)
<b>Atomic bit set and reset</b>	Yes	Yes
<b>Alternate functions</b>	Use PIOR Register	Use IOMUX
<b>Fast toggle</b>	No	MSPM0 can toggle pins every clock cycle
<b>Wake-up</b>	No	GPIO pin state change
<b>GPIO controlled by DMA</b>	No	<i>Only available on MSPM0</i>
<b>User controlled input filtering to reject glitches less than 1, 3, or 8 ULPCLK periods</b>	No	<i>Only available on MSPM0</i>
<b>User controllable input hysteresis</b>	No	<i>Only available on MSPM0</i>

## GPIO Code Examples

Information about GPIO code examples can be found in the [MSPM0 SDK examples guide](#).

## 4.2 Universal Asynchronous Receiver-Transmitter (UART)

RL78 and MSPM0 both offer peripherals to perform asynchronous (clockless) communication. For the RL78's UART, the standard features are in the Serial Array Unit (SAU) peripherals use as UART, the advanced features are in the SAU, or stand-alone UART peripherals such as LIN-UART, DALI/UART, and so forth. And for MSPM0, the standard features are named main, and the advanced features are named extend. Additionally, MSPM0 offers functionality not available in RL78 devices such as IrDA hardware support, Smart card mode, and Hardware flow control and so forth.

**Table 4-2. UART Feature Comparison**

Feature	RL78	MSPM0
Data direction	MSB or LSE	Equivalent
Continuous communication using DMA	Yes	Yes
Data phase	Yes (reverse output)	No
Reception in SNOOZE mode	Yes	Yes, active in all low-power mode.
Single-wire half duplex communication	Yes	Yes <sup>(1)</sup>
Data length	5, 7, 8, 9, 16 <sup>(2)</sup>	5, 6, 7, 8
LIN HW support	Yes	Yes
DALI HW support	Yes	Yes
IrDA HW support	No	Yes
Manchester Code HW support	No	Yes
Smart card mode (ISO7816)	No	Yes
Wakeup from low-power mode	Yes (LIN)	Yes
Auto baud rate detection	Yes (LIN)	No
External Driver enable	No	Yes
Hardware flow control	No	Yes
Multiprocessor	No	Yes
Tx/Rx FIFO Depth	No	4

(1) Requires reconfiguration of the peripheral between transmission and reception

(2) Data length 5 bits is available to UARTx, 9 bits is available to UARTx, and 16 bits is only available to some devices.

## 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 RL78 both support serial peripheral interface (SPI). For the RL78's SPI, SPI function is in the SAU (Serial Array Unit) peripherals used as CSI (Clocked Serial Interface). Besides, MSPM0 uses **Controller** and **Peripheral** to represent the communication parties of the SPI. Overall, MSPM0 and RL78 SPI support is comparable with the difference listed in [Table 4-3](#).

### Note

For RL78, different devices provide different SPI support levels, which are called SPI and simplified SPI.

**Table 4-3. SPI Feature Comparison**

Feature	RL78	MSPM0
Operation wires	SCK, SI, SO	SCLK, PICO, POCI, CSx
Controller or peripheral operation	Yes	Yes
Data bit width (controller mode)	7 to 16 bit <sup>(1)</sup>	4 to 16 bit
Data bit width (peripheral mode)		7 to 16 bit

**Table 4-3. SPI Feature Comparison (continued)**

Feature	RL78	MSPM0
Maximum speed	16 MHz (CSI00 only, Controller)	MSPM0L: 16 MHz
	8 MHz (Else, Controller) 5.33 MHz (Else, Peripheral)	MSPM0G: 32 MHz
Simplex transfers (unidirectional data line)	Yes	Yes
Hardware chip select management	No	Yes (4 peripherals)
Phase control of I/O clock	Yes	Yes
Transfer direction setting with MSB-first or LSB-first shifting	Yes	Yes
SPI format support	Motorola	Motorola, TI, MICROWIRE
Hardware CRC	No	No, MSPM0 offers SPI parity mode
TX FIFO depth	No	4
RX FIFO depth	No	4

(1) The supported data bit length varies depending on the device and operation mode.

### SPI Code Examples

Information about SPI code examples can be found in the [MSPM0 SDK examples guide](#).

### 4.4 Inter-Integrated Circuit (I2C)

MSPM0 and RL78 both support I2C peripherals. For RL78's I2C, basic function is provided in Serial Array Unit (SAU) peripherals use as simplified I2C, the advanced function is provided in IICA peripheral. And for MSPM0, I2C both basic and advance function is provided in I2C peripheral. MSPM0 uses **Controller** and **Target** to represent the both sides of communication . Overall MSPM0 and RL78 I2C support is comparable with notable difference outlined in the following table.

**Table 4-4. I2C Feature Comparison**

Feature	RL78	MSPM0
Controller and target modes	Yes (IICA)	Yes
Multi-controller capability	Yes (IICA)	Yes
Maximum transfer rate	1 MHz	Equivalent (Fast-mode Plus)
Addressing mode	7 bit	7 bit
Address number (Target mode)	1 address	2 addresses
Event management	No	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	No	Yes
Programmable analog and digital noise filters	No	Yes

### I2C Code Examples

Information about I2C code examples can be found in the [MSPM0 SDK examples guide](#).

## 4.5 Timers (TIMGx, TIMAx)

RL78 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 4-5. Timer Naming**

RL78		MSPM0	
Timer Name	Abbreviated Name	Timer Name	Abbreviated Name
		Advanced control	TIMA0
General purpose	TIMER RJ, RD, RG, RX	General purpose	TIMG0-11
Time interval generate	Interval Timer	High resolution	TIMG12
Basic	Timer Array Unit (TAU)		

**Table 4-6. Timer Feature Comparison**

Feature	RL78 Timers	MSPM0G Timers	MSPM0L Timers
Resolution	8, 12, 16, 32 bit	16 bit, 32 bit	16 bit
PWM	Yes	Yes	Yes
Capture	Yes	Yes	Yes
Compare	Yes	Yes	Yes
One-shot	Yes	Yes	Yes
Up down count functionality	Yes	Yes	Yes
Power Modes	Yes	Yes	Yes
QEI support	No	Yes	No
Programmable pre-scalar	Yes	Yes	Yes
Shadow register mode	No	Yes	Yes
Events/Interrupt	Yes	Yes	Yes
Auto reload functionality	Yes	Yes	Yes
Fault handling	No	Yes	Yes

**Table 4-7. Timer Module Replacement**

RL78 Timer	MSPM0 Equivalent	Reasoning
TAU	Any	General purpose, 16-bit resolution
32-bit Interval Timer	TIMG12	32-bit resolution
TIMER RJ, RD, RG, RX	TIMA, TIMG0-11	Advance purpose, PWM, Capture, Compare

**Table 4-8. Timer Use-Case Comparisons**

Feature	RL78 Timer	MSPM0 Timer
PWM	TIMER RJ, RD, RG	All timers have edge aligned or center aligned options
Capture	TIMER RD, RG, RX	All timers
Compare	TIMER RD	All timers
One-shot	All timers	All timers
Prescaler	4-bit prescaler	8-bit prescaler
Synchronization	TAU	All timers have this capability

### Timer Code Examples

Information about timer code examples can be found in the [MSPM0 SDK examples guide](#).

## 4.6 Windowed Watchdog Timer (WWDT)

RL78 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 4-9. WWDT Naming**

RL78	MSPM0
Watchdog timer (WDT)	Windowed watchdog timer (WWDT)

**Table 4-10. WDT Feature Comparison**

Feature	RL78	MSPM0G
Window mode	Yes	Yes
Interval timer mode	No	Yes
LFCLK source	Yes	Yes
Interrupts	Yes	Yes
Counter resolution	17 bit	25 bit
Clock divider	No	Yes

### WWDT Code Examples

Information about WWDT code examples can be found in the [MSPM0 SDK examples guide](#).

## 4.7 Real-Time Clock (RTC)

RL78 and MSPM0<sup>1</sup> both offer a real-time clock (RTC). The real-time clock (RTC) module provides time tracking for the application, with counters for seconds, minutes, hours, day of the week, day of the month, and year, in selectable binary or binary-coded decimal format.

**Table 4-11. RTC Feature Comparison**

Feature	RL78	MSPM0G
Power modes	Yes	Yes
Binary coded format	No	Yes
Years count up	99 years	199 years
Leap year correction	Yes	Yes
Number of customizable alarms	1	2
Internal and External crystal	Yes	Yes
Offset calibration	Yes	Yes
Interrupts	Yes	Yes

### RTC Code Examples

Information about RTC code examples can be found in the [MSPM0 SDK examples guide](#).

<sup>1</sup> Only MSPM0G devices support RTC.

## 5 Analog Peripheral Comparison

### 5.1 Analog-to-Digital Converter (ADC)

RL78 and MSPM0 both offer ADC peripherals to convert analog signals to a digital equivalent. Both device families feature a 12-bit ADC. In addition, RL78 I series offers 24-bit delta-sigma ADC, which is not in this discussion, and MSP430F676x can be considered. [Table 5-1](#) and [Table 5-2](#) compare the different features and modes of the ADCs.

**Table 5-1. Feature Set Comparison**

Feature	RL78	MSPM0G	MSPM0L
Resolution (Bits)	12, 10, 8	12, 10, 8	12, 10, 8
Conversion Rate (Msps)	0.888	4	1.4
Oversampling (Bits)	N/A	14	N/A
Hardware Oversampling	16x	128x	N/A
FIFO	No	Yes	Yes
ADC Reference (V)	Internal: 1.48, VDD	Internal: 1.4, 2.5, VDD	Internal: 1.4, 2.5, VDD
	External: $2.4 \leq V_{REFP} \leq V_{DD} \leq 5.5V$ $V_{REFM} = V_{SS} \text{ or } V_{REFM}$	External: $1.4 \leq V_{REF} \leq V_{DD}$	External: $1.4 \leq V_{REF} \leq V_{DD}$
Operating Power Modes	Run, SNOOZE	Run, Sleep, Stop, Standby <sup>(1)</sup>	Run, Sleep, Stop, Standby <sup>(1)</sup>
Auto Power Down	Yes	Yes	Yes
External Input Channels <sup>(2)</sup>	Up to 31	Up to 16	Up to 16
Internal Input Channels	Temperature Sensor, Internal Reference Voltage, Touch Sensor Capacitance	Temperature Sensor, Supply Monitoring, Analog Signal Chain	Temperature Sensor, Supply Monitoring, Analog Signal Chain
DMA Support	Yes (DTC/DMA)	Yes (DMA)	Yes (DMA)
ADC Window Comparator Unit	Yes (ADxL)	Yes	Yes
Simultaneous Sampling	No	Yes	No
Number of ADCs <sup>(3)</sup>	Up to 1	Up to 2	Up to 1

(1) ADC can be triggered in standby mode, which changes the operating mode.

(2) The number of external input channels varies per device.

(3) The number of ADCs varies per device.

**Table 5-2. Conversion Modes**

RL78 <sup>(1)</sup>	MSPM0	Comments
Select Mode, One-shot Conversion Mode / Single Scan Mode	Single Channel Single Conversion	ADC samples and converts a single channel once
Scan Mode, One-shot Conversion Mode / Single Scan Mode	Sequence of Channels Conversion	ADC samples a sequence of channels and converts once.
Select Mode, Sequential Conversion Mode / Continuous Scan Mode	Repeat Single Channel Conversion	Repeat single channel continuously samples and converts one channel
Scan Mode, Sequential Conversion Mode / Continuous Scan Mode	Repeat Sequence of Channels Conversion	Samples and converts a sequence of channels then repeats the same sequence

(1) The name of RL78 ADC conversion mode varies per device, here is two types of naming used '/' split.

#### ADC Code Examples

Information about ADC code examples can be found in the [MSPM0 SDK examples guide](#).

## 5.2 Comparator (COMP)

The RL78 and MSPM0 family of parts both offer integrated comparators as optional peripherals on some devices. In RL78, comparator is denoted as CMP, COMP, or COMPARATOR x, while in MSPM0 as COMPx. In RL78 G1F family, these x are numbered 0-1, and in MSPM0 family these x are numbered 0-2. In RL78 family, G1F series is used in BLDC motor, and have advance feature comparator, while the other series of RL78 mainly have a basic feature comparator. The comparator modules can take inputs from various internal and external sources, and can be used to trigger changes in power mode or truncate/control PWM signals. A summary of how the MSPM0 and RL78 comparator modules compare feature-by-feature is included in [Table 5-3](#).

**Table 5-3. COMP Feature Set Comparison**

Feature	RL78	MSPM0G	MSPM0L
<b>Available comparators</b>	Up to 2	Up to 3	Up to 1
<b>Output routing</b>	External	Multiplexed I/O Pins	Multiplexed I/O Pins
	Event linker controller	Interrupt/Event Interface	Interrupt/Event Interface
<b>Positive input</b>	External 4 Analog pins input	Multiplexed I/O Pins	Multiplexed I/O Pins
		DAC12 output <sup>(1)</sup>	DAC8 output
		DAC8 output	
	PGA output Comparator 0	Internal $V_{REF}$ : 1.4 V and 2.5 V	/
<b>Negative input</b>	External analog pin input	Multiplexed I/O pins	Multiplexed I/O Pins
		Internal temperature sensor	Internal temperature sensor
	Internal $V_{REF}$ : 1.45 V	Internal $V_{REF}$ : 1.4 V and 2.5 V	DAC8 output
	8 bit DAC Comparator 1	DAC8 output	OPA0 <sup>(3)</sup> output
		OPA0 output <sup>(3)</sup>	
<b>Programmable hysteresis</b>	None, 10 mV, 20 mV, 30 mV	None, 10 mV, 20 mV, 30 mV	None, 10 mV, 20 mV, 30 mV
		Other values from 0 V to $V_{REF}/V_{DD}$ using DAC8	Other values from 0 V to $V_{DD}$ using DAC8
<b>Register lock</b>	No	Yes, some COMP registers (writes require key)	Yes, some COMP registers (writes require key)
<b>Window comparator configuration</b>	Timer window with TAU0	Yes	No (single COMP)
<b>Input short mode</b>	No	Yes	Yes
<b>Operating modes</b>	Run	High speed, low power	High speed, low power
<b>Fast PWM shutdowns</b>	Yes	Yes (through TIMA fault handler)	No
<b>Output filtering</b>	Elimination digital filter (3 cycles)	Blanking filter	Blanking filter
		Adjustable analog filter	Adjustable analog filter
<b>Output polarity control</b>	Yes	Yes	Yes
<b>Interrupts</b>	Rising edge	Rising edge	Rising edge
	Falling edge	Falling edge	Falling edge
	Both edges	Output ready	Output ready
<b>Exchange inputs mode <sup>(4)</sup></b>	No	Yes	Yes

(1) Only on devices with DAC12 peripheral

(2) Only on devices with OPA1 peripheral

(3) Only on devices with OPA0 peripheral

(4) 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.

### COMP Code Examples

Information about COMP code examples can be found in the [MSPM0 SDK examples guide](#).



### 5.3 Digital-to-Analog Converter (DAC)

The RL78 and MSPM0 family of parts both offer 12-bit, 8-bit DAC peripherals to perform digital to analog conversion for various applications. In RL78 documentation, 12-bit DAC is referred to as the 12-bit D/A converter and 8-bit DAC is referred to as the DAC, D/A Converter. The 12-bit D/A converter is only available on the RL78 I1E and L1A families of devices. In MSPM0, the 12-bit DAC peripheral is referred to as the DAC12. This differentiates the DAC12 from the 8-bits DACs that are available for use with each comparator peripheral included in a given MSPM0 device. Those additional 8-bit DACs are covered in the comparator section of this document. This DAC12 peripheral is only available on the MSPM0G family of devices.

The features of the 12-bit DAC peripherals for the RL78 and MSPM0G are summarized in [Table 5-4](#).

**Table 5-4. DAC Feature Set Comparison**

Feature	RL78	MSPM0
Resolution	12 bits	12 bits (11 ENOB)
Output rate	33 kSPS	1 MSPS
Output channels	2 <sup>(1)</sup>	1 <sup>(2)</sup>
Data formats	12-bit right aligned, 12-bit left aligned	8-bit right aligned, 12-bit right aligned, two's complement or straight binary
DMA integration	Yes (DTC)	Yes
Output routing	External Pins	External Pins Internal peripheral connections: OPA IN+, COMP IN+, ADC0
Internal reference voltage	Yes, 1.48 V	Yes, 2.5 V or 1.4 V
External reference voltage	Yes	Yes
FIFO	No	Yes
Output buffer	No	Yes
Configurable output offset	No	Yes
Self-calibration mode	No	Yes
Trigger sources	Event link	Internal dedicated sample time generator, DMA interrupts/events, FIFO threshold interrupts/events, two hardware triggers (available from event fabric)

- (1) Available only on L1A device with 12 bits resolution.
- (2) Dual DAC channels are planned for future MSPM0G devices.

#### DAC12 Code Examples

Information about DAC12 code examples can be found in the [MSPM0 SDK examples guide](#).

### 5.4 Operational Amplifier (OPA)

The RL78 family of devices offer a series of devices having different types of amplifier. There are PGA and Amplifier Unit that contains three OPAs and 1 PGA in RL78. The MSPM0 OPA modules are completely flexible, and can individually, or in combination, replace many discrete amplifiers in sensing or control applications. The differences between MSPM0 OPA modules and RL78 Amplifier Unit are included in [Table 5-5](#).

**Table 5-5. OPA Feature Set Comparison**

Feature	RL78	MSPM0
GBW (low-power mode)	1.5 kHz (PGA), 60 kHz (OPA)	1 MHz (low-power mode)
GBW (standard mode)	67 kHz (PGA), 1.7 MHz (OPA)	6 MHz (standard mode)
Amplifier configurations or types	General-purpose Amplifier	General-purpose mode
	Buffer mode	Equivalent
	PGA peripheral	PGA mode (inverting or noninverting)
	Differential amplifier mode	Equivalent
	No	Cascade amplifier mode

**Table 5-5. OPA Feature Set Comparison (continued)**

Feature	RL78	MSPM0
Input/output routing	External pin routing	External pin routing
	Internal connections to DAC and ADC	Internal connections to ADC and COMP modules
Fault detection	No	Burnout current source (BCS)
Chopper stabilization	No	Standard (selectable chopping frequency)
		ADC assisted chop
		Disabled
Reference voltages	Internal bias voltage 1.0 V	Internal VREF (MSPM0G devices only)
	DAC12 (H1D)	DAC12 (MSPM0G devices only)
	DAC8 (H1D)	DAC8 (devices with COMP module only)

### OPA Code Examples

Information about OPA code examples can be found in the [MSPM0 SDK examples guide](#).

### 5.5 Voltage References (VREF)

The RL78 and MSPM0 both have internal references which can be used to supply a reference voltage to internal peripherals and only MSPM0 can output internal references to external peripherals.

**Table 5-6. VREF Feature Set Comparison**

Feature	RL78	MSPM0G	MSPM0L
Internal Reference (V)	1.45, SBIAS	1.4, 2.5	1.4, 2.5
External Reference (V)	$2.4 \leq V_{REFP} \leq 5.5 \text{ V}$	External: $1.4 \leq V_{REF} \leq V_{DD}$	External: $1.4 \leq V_{REF} \leq V_{DD}$
Output Internal Reference	Yes	Yes	Yes
Internally Connect to ADC	Yes	Yes	Yes
Internally Connect to DAC	Yes	Yes	No
Internally Connect to COMP	Yes	Yes	No
Internally Connect to OPA	Yes	Yes	No

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](#).

## 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](https://www.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  
Copyright © 2023, Texas Instruments Incorporated