Application Note

Migration Guide From MSP430™ MCUs to MSPM0 MCUs

ABSTRACT

The MSP430™ devices have nearly 30 years of history as TI's classic microcontroller. Many customers have used MSP430 MCUs on different projects and are still using them today. The new generation of MSPM0 products adopts the Arm® Cortex®-M0+ core, which has more function-rich peripheral resources and a smaller package. With the development of new products or upgrades to old products, you might have the need to use the newer MSP devices. This application note describes migration of software from MSP430 MCUs to MSPM0 MCUs.

Table of Contents

Trademarks

MSP430™, Code Composer Studio™, and LaunchPad™ are trademarks of Texas Instruments.
Arm® and Cortex® are registered trademarks of Arm Limited.
All trademarks are the property of their respective owners.

1 Software Porting Flow

Figure 1-1 shows the steps of software migration from MSP430 to MSPM0 MCUs. The following sections describe differences between the development environments, software, and peripherals.

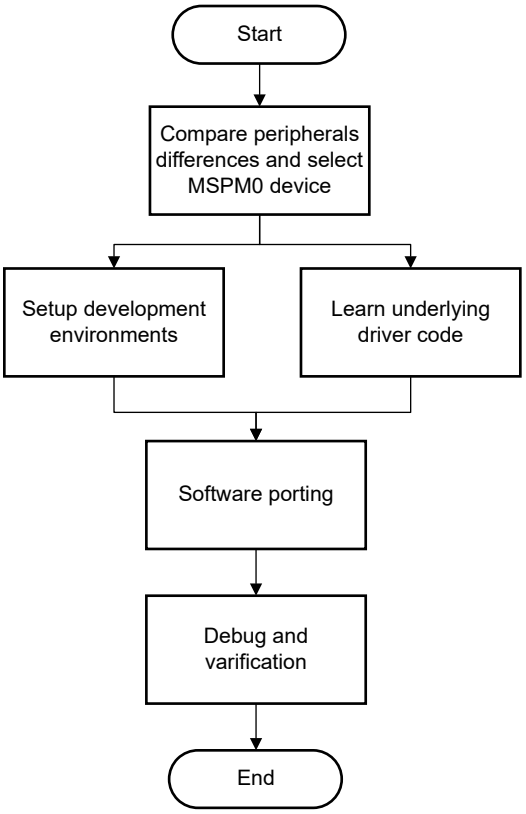

Figure 1-1. Software Migration Workflow

2 Development Environments

2.1 Integrated Development Environments (IDEs)

The Code Composer Studio™ integrated development environment (IDE) (CCS) supports all devices in the TI Microcontroller and Embedded Processors portfolio. In addition to CCS, MSP430 and MSPM0 devices are supported by other IDEs (see Table 2-1).

Table 2-1. IDE Support

IDE	MSP430	MSPM0
CCS	Yes	Yes
IAR	Yes	Yes
Keil	No	Yes

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

2.2 Software Ecosystems

Table 2-2. Comparison of Software Ecosystems

Feature	MSP430Ware	MSPM0 SDK
Register-level code	Yes	No
Driver library	Yes	Yes
Middleware	Yes	Yes
Out of box code	Yes	Yes
Free RTOS	No	Yes

2.2.1 MSP430 Software Support Package: MSP430Ware

MSP430Ware is a collection of resources that help users to effectively create and build MSP430 code. These resources support all MSP430 microcontrollers (MCUs). In addition to example code for all peripherals, this complete collection of design resources includes a wide selection of highly abstracted software libraries. In particular, the MSP430 Driver Library is an essential library to help software developers leverage convenient APIs to control low-level and intricate hardware peripherals, making the resulting code much easier to read and maintain. MSP430Ware also has a cloud version.

2.2.1.1 Register-Level Example Code

The 16-bit MSP430 MCUs support core speeds up to 25 MHz. Register-level example code operates the registers directly to make the project faster and smaller (see Figure 2-1 for an example). The code examples for the MSP430 MCUs cover all peripheral functions of the devices.

```
/******************************************************************************
*
*                       MSP430 CODE EXAMPLE DISCLAIMER
*
* MSP430 code examples are self-contained low-level programs that typically
* demonstrate a single peripheral function or device feature in a highly
* concise manner. For this the code may rely on the device's power-on default
* register values and settings such as the clock configuration and care must
* be taken when combining code from several examples to avoid potential side
* effects. Also see www.ti.com/grace for a GUI- and www.ti.com/msp430ware
* for an API functional library-approach to peripheral configuration.
*
* --/COPYRIGHT--*/
//******************************************************************************
//   MSP430F665x Demo - Software Toggle P1.0
//
//   Description: Toggle P1.0 by xor'ing P1.0 inside of a software loop.
//   ACLK = 32.768kHz, MCLK = SMCLK = default DCO~1MHz
//
//                MSP430F665x
//             -----------------
//         /|\|                 |
//          | |                 |
//          --|RST              |
//            |                 |
//            |             P1.0|-->LED
//
//   P. Thanigai
//   Texas Instruments Inc.
//   May 2012
//   Built with IAR Embedded Workbench Version: 5.40 & CCS V5.2
//******************************************************************************

#include <msp430.h>

int main(void)
{
  volatile unsigned int i;

  WDTCTL = WDTPW | WDTHOLD;                 // Stop WDT
  P1DIR |= BIT0;                            // P1.0 set as output

  while(1)                                  // continuous loop
  {
    P1OUT ^= BIT0;                          // XOR P1.0
    for(i=20000;i>0;i--);                   // Delay
  }
}
```

Figure 2-1. MSP430 Register-Level Example Code

2.2.1.2 Driver Library

This peripheral driver library allows application development at an API level instead at the device register level. The APIs allow developers to concentrate on the applications instead of the nuances of the particular MSP430 device in use. For more information, see the *MSP430Ware Peripheral Driver Library API* and *MSP430Ware Peripheral Driver Library User's Guides* in CCS or TI Resource Explorer.

2.2.1.3 Middleware

For some specific application scenarios, the middleware example projects provide reference code to help users to design more conveniently. For example, the IQmath Library helps to develop faster and more complex calculations, and the USB Developer's Package helps to develop USB applications. For details, refer to the middleware of each device in TI Resource Explorer. Figure 2-2 shows the location of a typical middleware example.

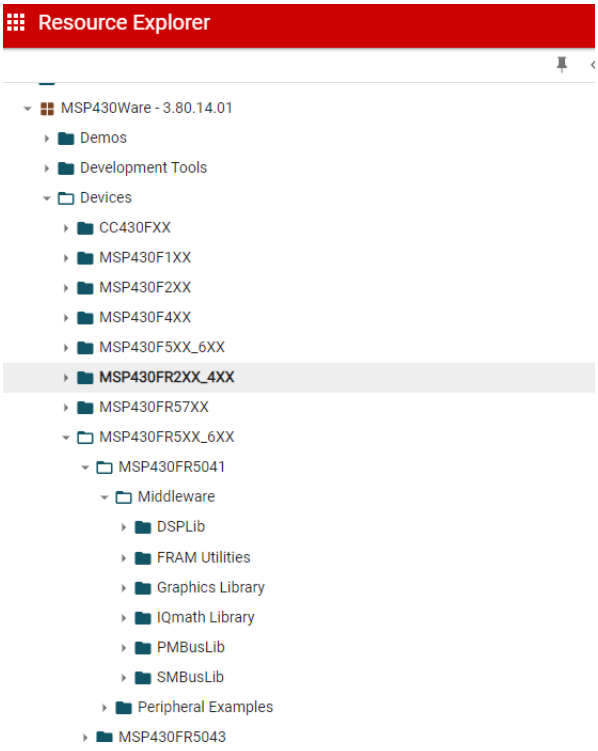

Figure 2-2. MSP430 Middleware

2.2.2 MSPM0 Software Support Package: MSPM0SDK

The MSPM0 SDK delivers components that help to develop applications on Texas Instruments MSPM0+ microcontrollers. The MSPM0 SDK comprises multiple software components and examples of how to use these components together. In addition, examples demonstrate the use of each functional area and each supported device as a starting point for your own projects. Figure 2-3 shows a typical 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™ development kit 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. For details, see the *MSPM0 SDK User's Guide*.

Some examples support SysConfig to simplify device configuration and accelerate software development. See each example for more details.

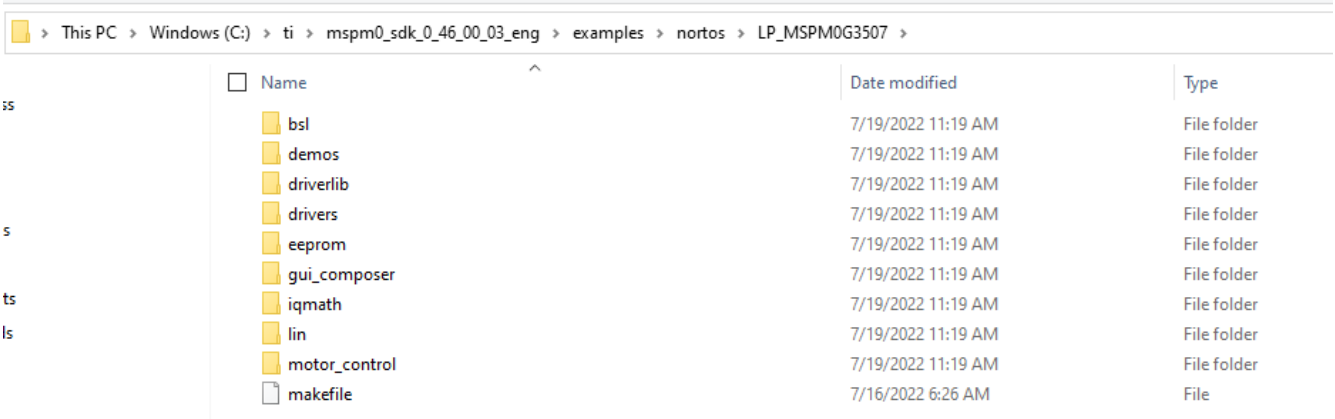

Figure 2-3. MSPM0 SDK

2.3 SysConfig for MSPM0 MCUs

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 output includes C header and code files that can be used with MSPM0 SDK examples or used to configure custom software. For details, see the *MSPM0 SysConfig Guide*.

2.3.1 Standalone SysConfig

Standalone SysConfig (see Figure 2-4) can be downloaded and installed on your PC to generate header and code files to configure your project.

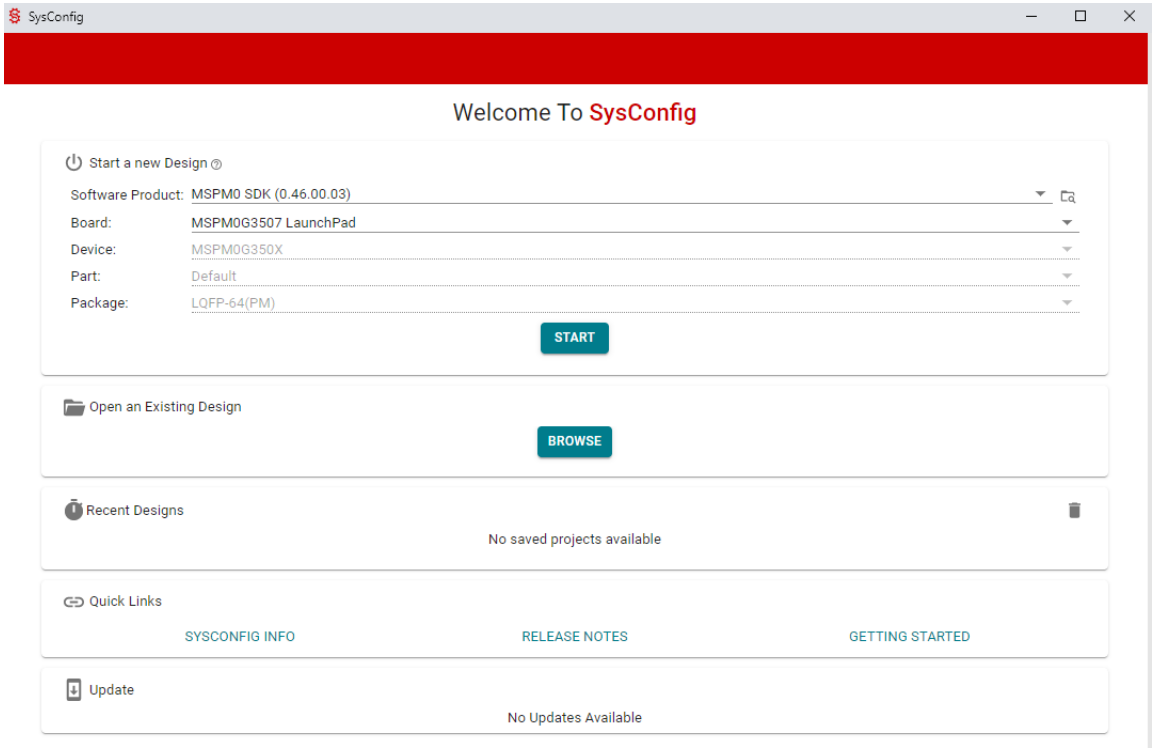

Figure 2-4. Standalone SysConfig

2.3.2 CCS-Integrated SysConfig

Many examples in the MSPM0 SDK include a SysConfig file with the extension *.syscfg*. This SysConfig project file saves the initialization of systems such as clock configuration, low-power mode rules, ADC configuration, I2C, SPI, and UART configuration, and other peripheral configuration. These project files can be changed in CCS and kept with the project. If you want to use the same configuration for another project, the configuration can be easily imported to the new project.

2.3.3 Example of a SysConfig Project

In the toggle output example project shown in Figure 2-5, the configuration file is included in the project. When the project is built, CCS generates the configuration header and code files based on the configuration file. In this project, the GPIOs and system clock are configured. For further development, open the SysConfig file in CCS and modify the peripherals as needed. The header and code files are updated when the project is built.

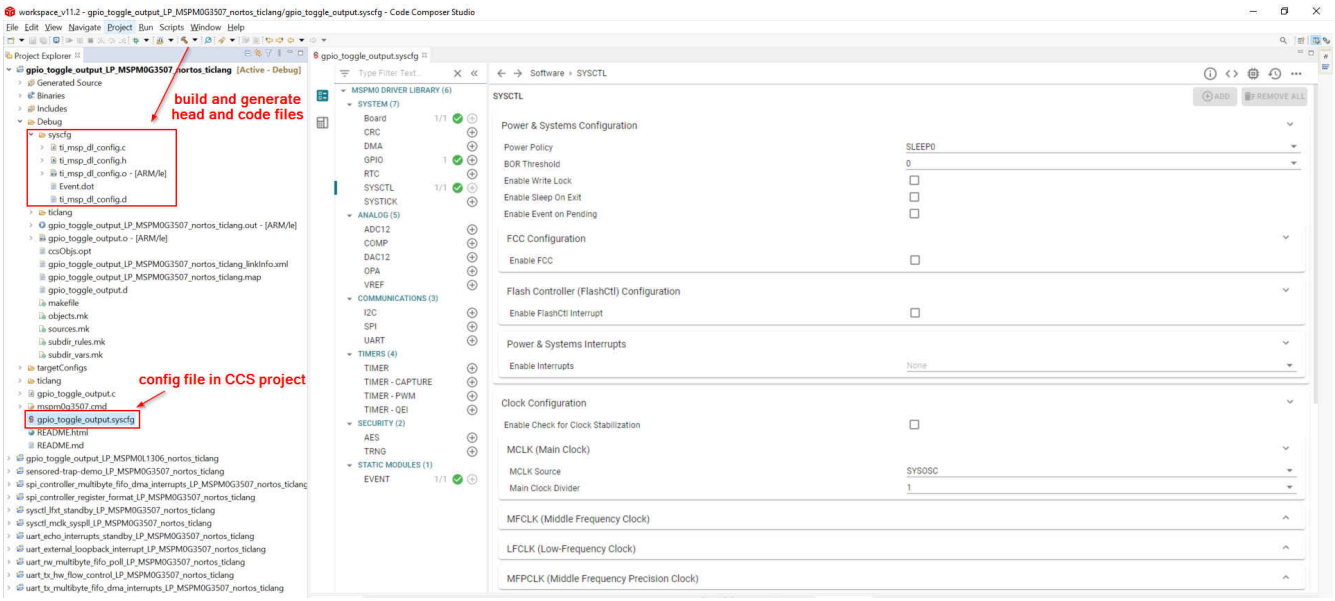

Figure 2-5. SysConfig Project With CCS

2.4 MSP430 and MSPM0 Projects

The following examples are of a PWM output. In MSP430 register-level code, each bit in the configuration registers must be set to the relevant values for the clock control register, GPIO register, and timer register. The code is difficult to read and modify.

```
//           MSP430FR5994
//         ---------------
//     /|\|               |
//      | |               |
//      --|RST            |
//        |               |
//        |     P1.2/TA1.1|--> CCR1 - 75% PWM
//        |     P1.3/TA1.2|--> CCR2 - 25% PWM
//
//   William Goh
//   Texas Instruments Inc.
//   October 2015
//   Built with IAR Embedded Workbench V6.30 & Code Composer Studio V6.1
//******************************************************************************

#include <msp430.h>

int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;               // Stop WDT

    // Configure GPIO
    P1DIR |= BIT2 | BIT3;                   // P1.2 and P1.3 output
    P1SEL0 |= BIT2 | BIT3;                  // P1.2 and P1.3 options select
    P1SEL1 &= ~(BIT2 | BIT3);

    // Disable the GPIO power-on default high-impedance mode to activate
    // previously configured port settings
    PM5CTL0 &= ~LOCKLPM5;

    CSCTL0_H = CSKEY_H;                     // Unlock CS registers
    CSCTL1 = DCOFSEL_6;                     // Set DCO to 8MHz
    CSCTL2 = SELA__VLOCLK | SELS__DCOCLK | SELM__DCOCLK;// Set ACLK = VLO; SMCLK = DCO/8
    CSCTL3 = DIVA__8 | DIVS__8 | DIVM__8;   // Set all dividers
    CSCTL0_H = 0;

    TA1CCR0 = 1000-1;                       // PWM Period
    TA1CCTL1 = OUTMOD_7;                    // CCR1 reset/set
    TA1CCR1 = 750;                          // CCR1 PWM duty cycle
    TA1CCTL2 = OUTMOD_7;                    // CCR2 reset/set
    TA1CCR2 = 250;                          // CCR2 PWM duty cycle
    TA1CTL = TASSEL__SMCLK | MC__UP | TACLR;// SMCLK, up mode, clear TAR

    __bis_SR_register(LPM0_bits);           // Enter LPM0
    __no_operation();                       // For debugger
}
```

Figure 2-6. MSP430 PWM Output Example

To configure the same function in an MSPM0 device, select the options in the SysConfig tool, which generates the configuration file. Select the clock and input the count number for the PWM frequency to be calculated by the tool. Input the duty cycle you want and then the compare number is generated by tools. All used GPIO function are configured by SysConfig. Using SysConfig helps to design and initial your system.

The SysConfig output is based on the SDK using driverlib. The APIs are more readable compared to the MSP430 register level example code above.

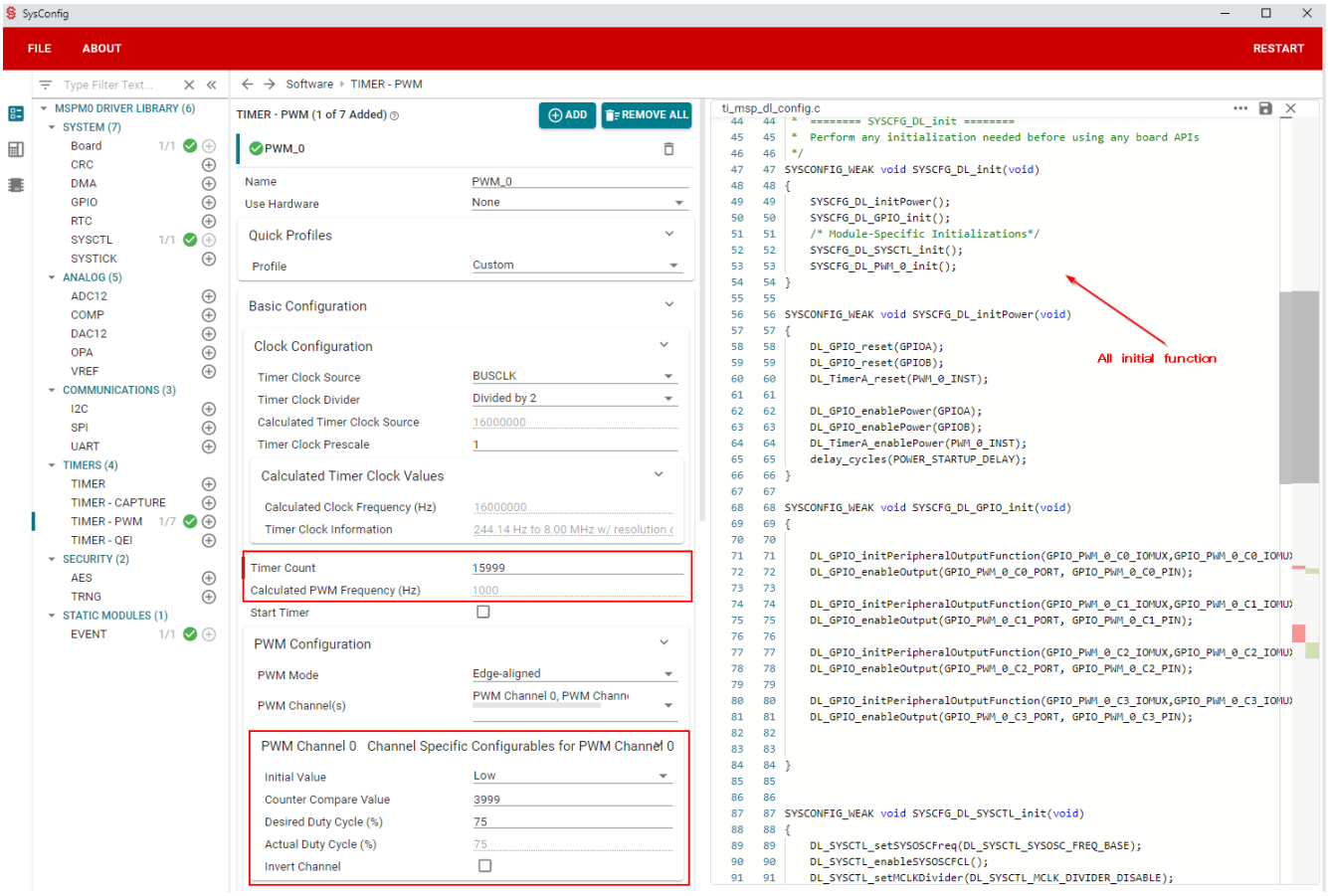

Figure 2-7. MSPM0 PWM Output Example

2.5 Debugger Interfaces

Different from the MSP430 devices, the MSPM0 devices are equipped with an Arm Cortex-M0+ core, so the supported debuggers are also different. Table 2-3 lists the differences between those two families.

Table 2-3. Debugger Interfaces

Tool Type	MSP430	MSPM0
Development tool	MSP-FET	XDS110/J-Link
Production tool	MSP-GANG	MSP-GANG

2.5.1 MSP430 Debugger

The MSP-FET (see Figure 2-8) is a powerful emulation development tool that helps users to quickly begin development on MSP430 devices. The MSP-FET supports programming and real-time debugging over both JTAG and SBW interfaces. Furthermore, the MSP-FET also provides a Backchannel UART connection between a computer USB interface and the MSP UART. This affords the MSP programmer a convenient method for communicating serially between the MSP and a terminal running on the computer. It also supports loading programs (often called firmware) to the MSP target using the BSL (bootstrap loader) through the UART and I2C communication protocols. For details, see the *MSP Debuggers User's Guide*.

Figure 2-8. MSPFET

2.5.1.1 MSPFET Connection Interface

The USB interface connects the MSP-FET to the computer, while the 14-pin connector provides access to the MSP debug emulation port, which uses either a standard JTAG interface or the pin-saving Spy-Bi-Wire (2-wire JTAG) protocol. These two protocols have different speeds.

Table 2-4. Speed of SBW and JTAG Communication for MSP430 MCUs

Interface	Slow	Medium	Fast
SBW	200 kHz	400 kHz	600 kHz
JTAG 4-wire MSP430	1 MHz	4 MHz	8 MHz

Figure 2-9 shows the connections between the 14-pin FET interface module connector and the target device required to support in-system programming and debugging for 4-wire JTAG communication. Figure 2-10 shows the connections for 2-wire JTAG mode (Spy-Bi-Wire). The 4-wire JTAG mode is supported on most MSP430 devices, except devices with low pin counts (for example, MSP430G2230). The 2-wire JTAG mode is supported on selected devices only. See the *Code Composer Studio IDE for MSP430 MCUs User's Guide* or *IAR Embedded Workbench IDE for MSP430 MCUs User's Guide* for information on which interface method can be used on which device.

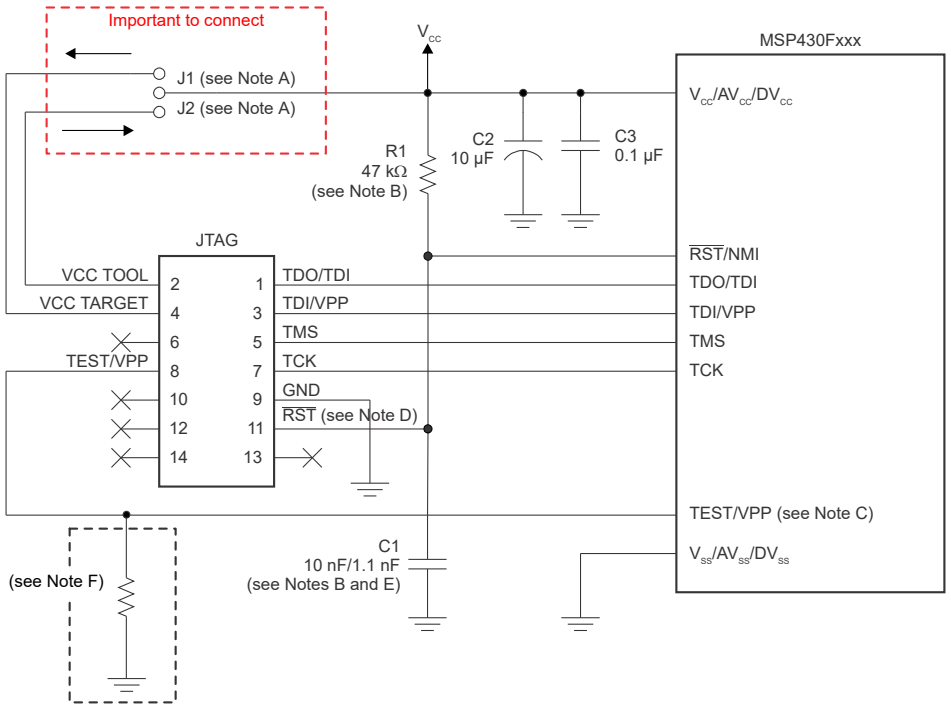

Figure 2-9. Signal Connections for 4-Wire JTAG Communication

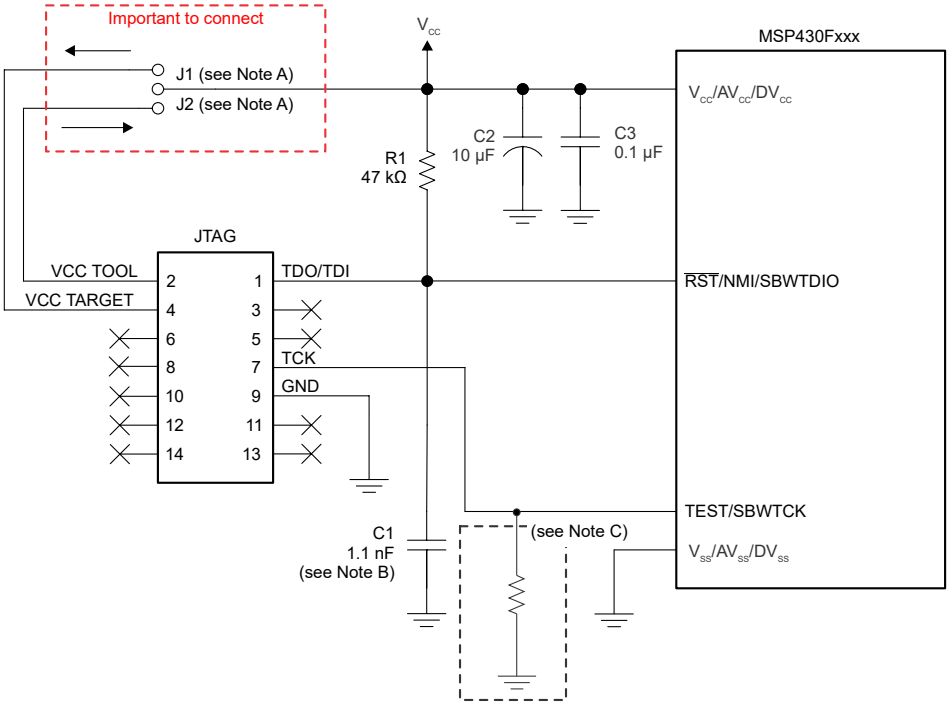

Figure 2-10. Signal Connections for 2-Wire JTAG Communication (Spy-Bi-Wire)

Note

On some Spy-Bi-Wire capable MSP430 devices, TEST/SBWTCK is very sensitive to rising signal edges that can cause the test logic to enter a state where an entry sequence (either 2-wire or 4-wire) is not recognized correctly and JTAG access stays disabled. Unintentional edges on SBWTCK can occur when the JTAG connector is connected to the target device.

2.5.2 MSPM0 Debugger

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). The connection of debugger see figure 1-9.

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. These probes benefit from record-breaking flashloaders, 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 including Cortex-M0+. For details, visit the SEGGER J-Link Debug Probes page.

Figure 2-11 shows a high-level diagram of the major functional areas and interfaces of the XDS110 probe to MSPM0 target.

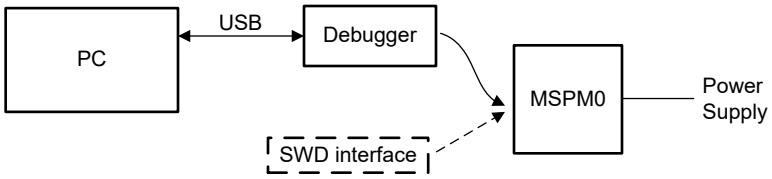

Figure 2-11. XDS110 Probe to MSPM0 Target

2.5.2.1 MSPM0 Debug Port Pins and Pinout

The debug port uses the SWCLK and SWDIO pins, which have internal pulldown and pullup resistors (see Figure 2-12). The MSPM0 MCU family is offered in various packages with different numbers of available pins. For details, see the device-specific data sheet.

Table 2-5. MSPM0 SWD Interface

Device Signal	Direction	SWD Function
SWCLK	Input	Serial wire clock from debug probe
SWDIO	Input/Output	Bidirectional (shared) serial wire data

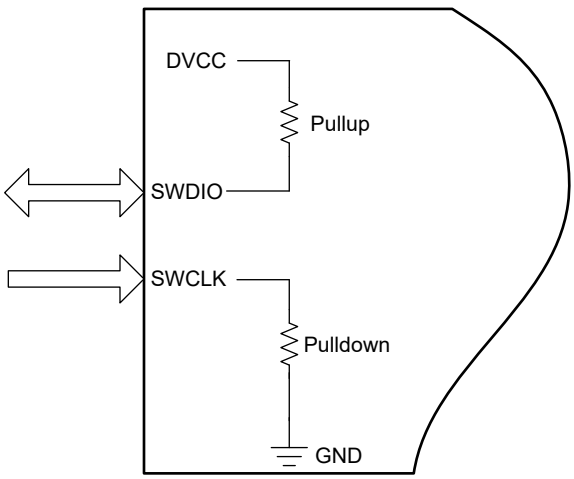

Figure 2-12. MSPM0 SWD Connection

Figure 2-13 shows the connection between the MSPM0 SWD debug port and the standard 10-pin JTAG connector.

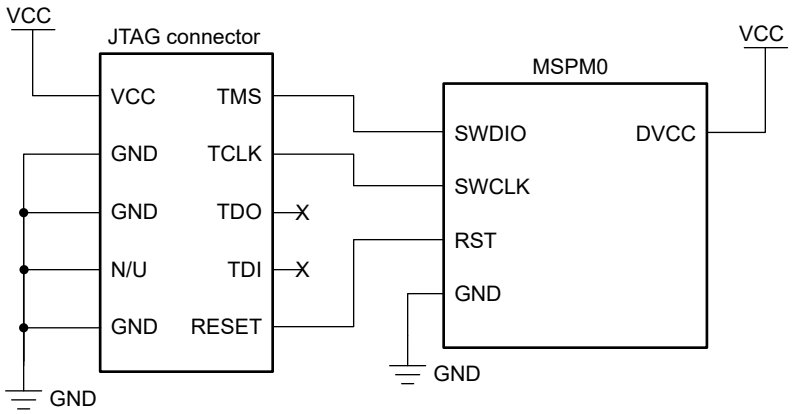

Figure 2-13. MSPM0 Connection to 10-Pin JTAG

3 Migration Considerations

3.1 Peripherals

Table 3-1 compares the peripherals of the MSP device families. See the device-specific data sheet to find the peripherals that are supported on each individual device.

Table 3-1. Peripherals Comparison

Peripheral	MSP430	MSPM0L	MSPM0G
CPU	Up to 25 MHz	Up to 32 MHz	Up to 80 MHz
SAR ADC	8-, 10-, or 12-bit SAR ADC (depends on devices)	12-bit 1-Msps SAR ADC	14-bit 4-Msps SAR ADC oversampling (4-Msps ADC)
Sigma-Delta ADC	16- or 24-bit SD ADC	NA	NA
COMP	Supported	Window compare	Window compare
OPA, GPAMP	GBW up to 2.8M	Burnout circuit GBW up to 6M	Burnout circuit GBW up to 6M
DAC	8- or 12-bit DAC	8-bit DAC	12-bit DAC
UART	Support IrDA	Supports LIN, IrDA, ISO7816, RS485	Supports LIN, IrDA, ISO7816, RS485
REF	1.5 V, 2.0 V, 2.5 V	Internal 1.4 V or 2.5 V	Internal 1.4 V or 2.5 V
SPI	Up to 8 MHz	Up to 16 MHz, supports TI and Motorola modes, Command Line mode	Up to 32 MHz, supports TI and Motorola modes, Command Line mode
I2C	100 kHz	Up to Fm+ 1 MHz, supports hardware SMBus and PMBus	Up to Fm+ 1 MHz, supports hardware SMBus and PMBus
CAN-FD	Not available	Not available	supported
CRC	CRC16-CCITT	CRC16-CCITT, CRC32-ISO3309	CRC16-CCITT, CRC32-ISO3309
AES	AES 128-bit data with 128-, 192-, or 256-bit key	AES 128-bit data with 128-, 192-, or 256-bit key	AES 128-bit data with 128-, 192-, or 256-bit key
TRNG	Not available	Not available	32-bit true random output
TIMER	General timer functions, PWM, or capture and compare	General timer functions, PWM, or capture and compare	General timer functions, PWM with edge- or center-align, complementary, and dead-band function
WWDT	1	1	2
RTC	Supported	Not available	Supported
DEBUG	MSPFET: JTAG (4-wire) or SBW	XDS110 or J-link: SWD	XDS110 or J-link: SWD
LCD	Supported	Not available	Not available
USS	Supported	Not available	Not available
CapTIvate technology	Supported	Not available	Not available
USB	Supported	Not available	Not available

3.2 System Clocks

3.2.1 Oscillators

MSP430 and MSPM0 devices have many types clock sources form both internal and external for low system cost and low power consumption. Table 3-2 lists the different clock sources in MSP430 and MSPM0 devices. Note that not all the devices have all types clock sources. For details, see the device-specific data sheet.

Table 3-2. Oscillator Comparison

Type	MSP430	MSPM0L	MSPM0G
Internal oscillators	DCOCLK: internal high- speed oscillator up to 25 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
	REFOCLK: internal 32-kHz oscillator, which can source the FLL	LFOSC: internal 32-kHz low-frequency oscillator	LFOSC: internal 32-kHz low-frequency oscillator
	VLOCKL: internal 10-kHz very low-power oscillator	Not available	Not available
External oscillators	XT1CLK: High-frequency oscillator or low-frequency oscillator (32 kHz). Depends on device.	Not available	LFXT: external low-frequency oscillator
	XT2CLK: High-frequency oscillator. Depends on device.	Not available	HFXT: external high-frequency oscillator

3.2.1.1 MSPM0 Oscillators

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

Table 3-3. Oscillators in MSPM0 MCUs

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

3.2.2 Clock Signals

Different clock signals are supported in different low-power modes to reduce system power consumption. The following sections describe the differences in the clock signals of the MSP430 and MSPM0 MCUs.

Table 3-4. Clock Signal Comparison

Clock	MSP430	MSPM0L	MSPM0G
MCLK	Source CPU and some digital peripherals	Main system clock for PD1 bus and peripherals	Main system clock for PD1 bus and peripherals
CPUCLK		CPU clock derived from MCLK	CPU clock derived from MCLK
SMCLK	Subsystem main clock for peripherals working independently from CPU	Main system clock for PD0 peripherals and PD0 bus, derived from MCLK	Main system clock for PD0 peripherals and PD0 bus, derived from MCLK
ACLK	Auxiliary clock 32 kHz	Fixed 32-kHz clock	Fixed 32-kHz clock
MFCLK		Fixed 4 MHz synchronized to MCLK	Fixed 4 MHz synchronized to MCLK
MFPCLK		Fixed 4 MHz	Fixed 4 MHz

3.3 Operation Modes

All MSP devices have multiple low-power modes. Table 3-5 lists the operating modes of the MSP430 and MSPM0 MCUs.

Table 3-5. Operation Mode Comparison

MSP430		MSPM0L and MSPM0G	
Operation Mode	Description	Operation Mode	Description
Active	CPU, clock, and peripherals work	RUN	CPU, clock, and peripherals work
LPM0	Active with CPU clock down	SLEEP	RUN with CPU clock down

Table 3-5. Operation Mode Comparison (continued)

MSP430		MSPM0L and MSPM0G	
Operation Mode	**Description**	**Operation Mode**	**Description**
LPM2, LPM3	ACLK clock enabled, some peripherals optional	STOP	PD0 enabled and PD1 disabled. Available clock: MFCLK or LFCLK
		STANDBY	PD0 enabled and PD1 disabled. Available clock: LFCLK
LPM4	All clocks off, some peripherals optional	SHUTDOWN	Device shut down
LPM3.5	Device shut down with RTC		
LPM4.5	Device shut down		

3.4 Nonvolatile Memory (NVM)

3.4.1 MSPM0 Memory Protection Unit

MSPM0 MCUs support a memory protection unit to check all memory accesses made by the processor against a set of access permission policies that are defined by the programmer.

The MPU is configured through memory-mapped registers in the system private peripheral bus (PPB) region. For more detailed information on the MPU register configuration, see the MPU section of the *Cortex-M0+ Devices Generic User Guide*.

3.4.2 MSP430 FRAM and MSPM0 Flash

FRAM is a nonvolatile memory that reads and writes like standard SRAM. FRAM is available on MSP430FRxx devices.

Table 3-6. MSP430 FRAM and MSPM0 Flash

Feature	MSP430 FRAM	MSPM0 Flash
Access	Word or byte write access	Single flash word (64 bits) or multiple words
ECC	Supported	Supported (MSPM0G devices)
Protection	Supported	Supported
Cycles	10^{15}	100k (lower 32KB) or 10k (above 32KB)

FRAM has speed limitation. The system clock for CPU can exceed the FRAM access and cycle time requirements. For these scenarios, a wait-state generator mechanism is implemented. The maximum FRAM memory access speed is 8 MHz. If the MCLK is operating faster than 8 MHz and FRAM access is required, wait states are necessary for reliable FRAM access. When using MCLK ≥ 8 MHz, configure the FRAM wait states in software before configuring the MCLK frequency.

3.4.3 MSP430 Flash and MSPM0 Flash

Table 3-7 compares the key features of MSP430 flash devices and MSPM0 flash devices. Details refer to specific user guide.

Table 3-7. MSP430 Flash and MSPM0 Flash

Feature	MSP430 Flash	MSPM0 Flash
Voltage generation	Internal	Internal
Program mode	Byte, word (2 bytes), and long (4 bytes)	Single flash word (64 bits) or multiple words
Erase	Segment, bank, mass	Sector, bank, mass
ECC	NA	Supported
Cycles	100k	100k (lower 32KB) or 10k (above 32KB)

3.5 Event and Interrupt Handling

In MSP430 MCUs, the interrupt priorities are fixed and defined by the arrangement of the modules in the connection chain as shown in Figure 3-1. There are three types of interrupts: system reset, (non)maskable, and maskable.

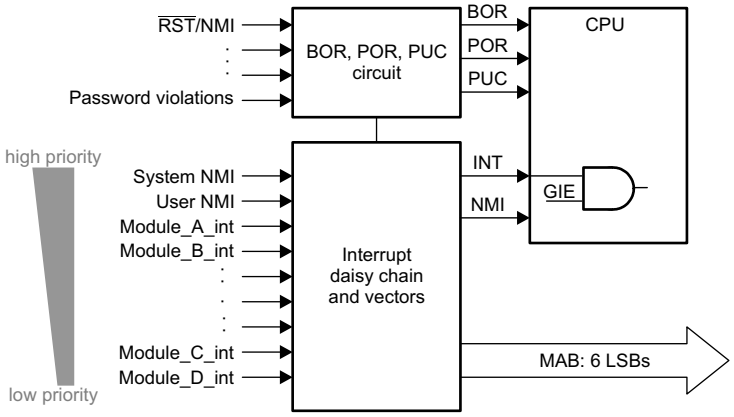

Figure 3-1. MSP430 Event and Interrupt Handling

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.

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: static event routes, DMA event routes, and generic event routes.

Figure 3-2 shows the event map. Different peripherals are routed through different event fabrics to achieve different event transitions. For more details, see the device technical reference manual.

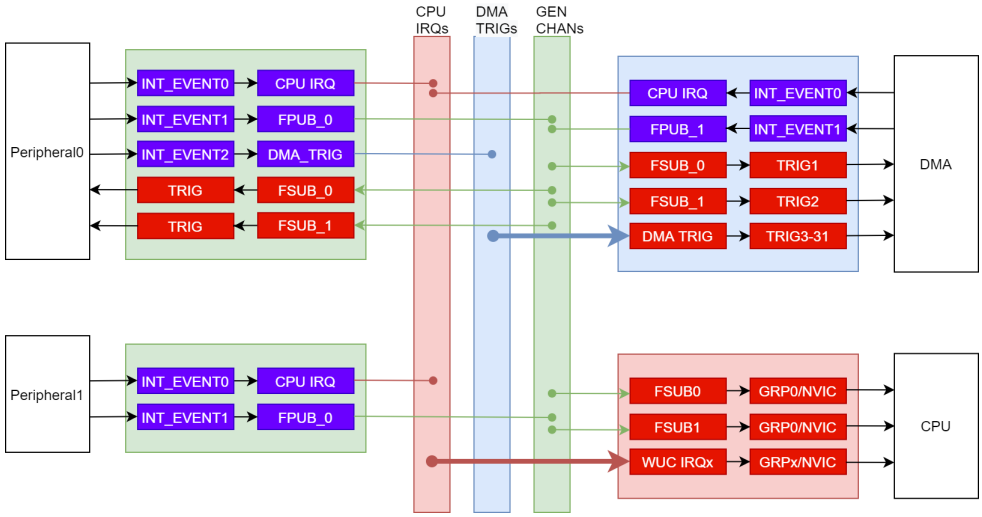

Figure 3-2. MSPM0 Event and Interrupt Handling

3.6 Reset Levels

MSP430 MCUs support three reset levels: POR, BOR and POC. Each is triggered by specific events as described in the device family user guides. The relationship between these trigger signals are:

- A POR is always generated when a BOR is generated, but a BOR is not generated by a POR.
- A PUC is always generated when a POR is generated, but a POR is not generated by a PUC.

MSPM0 MCUs have five reset levels:

1. Power-on reset (POR)
2. Brownout reset (BOR)
3. Boot reset (BOOTRST)
4. System reset (SYSRST)
5. CPU reset (CPURST)

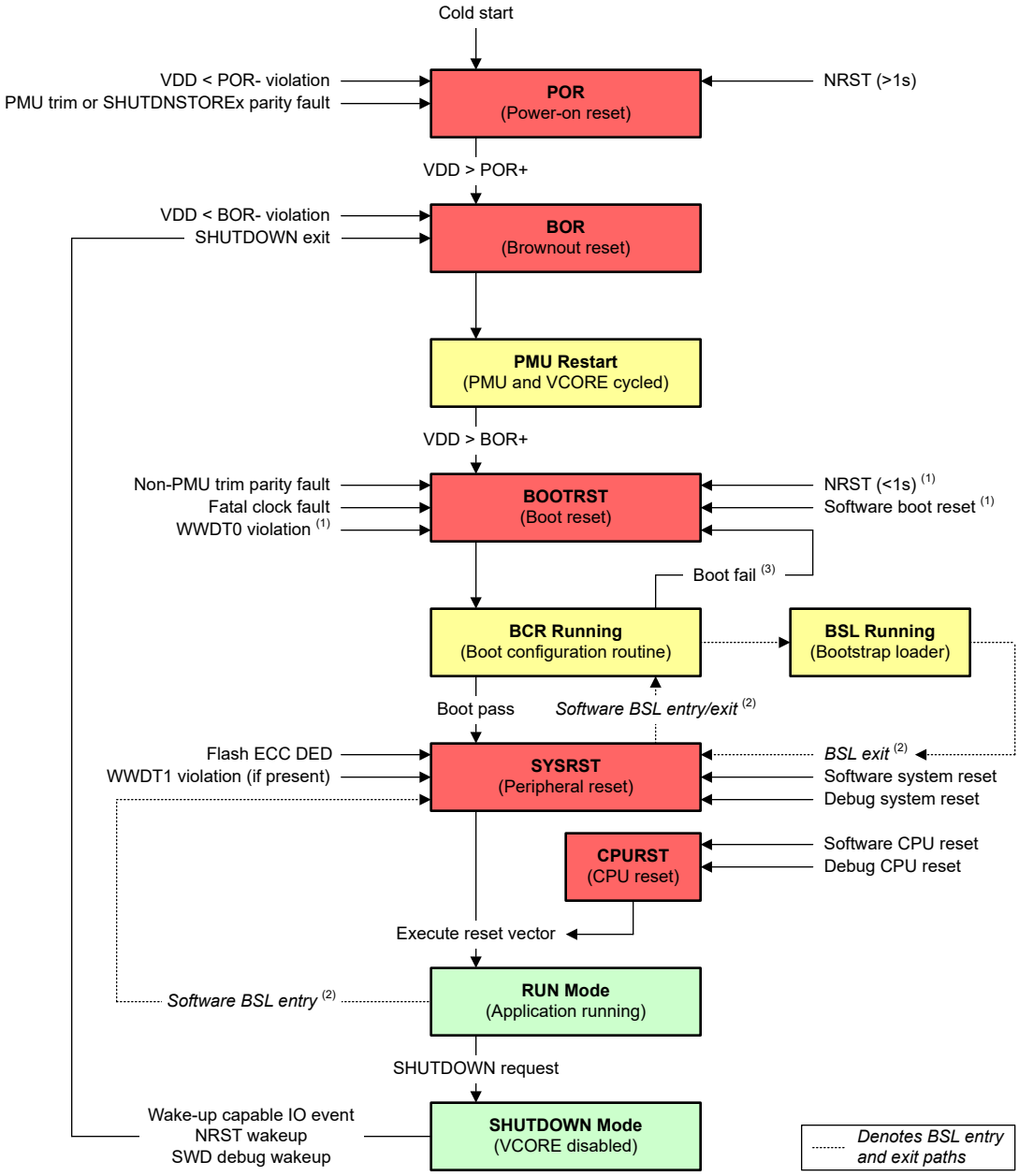

Figure 3-3. MSPM0 Resets

3.7 GPIOs and Pin Multiplexing

In MSP430 MCUs, the pins are set to digital I/Os by default with multiplexed functions like ADC or DAC output, CapTIvate I/O, or eUSCI. The digital I/O features include:

- Independently programmable I/Os
- Any combination of input or output
- Individually configurable P1 and P2 interrupts. Some devices may include additional port interrupts.
- Independent input and output data registers

- Individually configurable pullup or pulldown resistors

In MSPM0 devices, all pins are set to analog functions by default.

The IOMUX manages the configuration of the digital IO. Key functions configured by IOMUX include:

- Selection of which peripheral is muxed to each digital IO pin (for example, a GPIO or UART peripheral)
- Digital input path configuration
  - Hysteresis control
  - Input path enable or disable
  - Input logic inversion control
- Digital output path configuration
  - Drive strength control
  - Output connection enable or disable
  - Output logic inversion (control shared with input logic inversion)
  - Logic-high to High-Z output conversion (for open-drain style interfaces)
  - Retention of "last state" when a peripheral connected to an IO is disabled
- Wakeup configuration (for wakeup from SHUTDOWN mode)
  - Wakeup compare level
  - Wakeup enable or disable
- Pullup or pulldown resistor control

3.8 Communication Interfaces

The performance of the communication interface of MSPM0 MCUs is better than that of MSP430 MCUs. There can be richer applications on MSPM0 MCUs.

3.8.1 SPI

The SPI module provides a standardized serial interface to transfer data between MSPM0 devices and other external devices using SPI protocols (such as sensors, memory, ADCs, or DACs).

The SPI modules have the following features:

- Configurable as a controller or a peripheral
- Programmable clock bit rate and prescaler
- Separate transmit (TX) and receive (RX) first-in first-out buffers (FIFOs)
- Programmable data frame size from 4 bits to 16 bits (controller mode)
- Programmable data frame size from 7 bits to 16 bits (peripheral mode)
- Interrupts for transmit and receive FIFOs, overrun and timeout interrupts, and DMA done
- Programmable SPI mode support Motorola SPI, MICROWIRE, or Texas Instruments format
- Direct memory access controller interface (DMA):
  - Separate channels for transmit and receive
  - Transfer complete interrupt

3.8.2 I2C

The I2C module provides a standardized serial interface to transfer data between MSP devices and other external I2C devices (such as a sensors, memory, or DACs). The I2C peripheral provides bidirectional data transfer through a two-wire serial bus consisting of a data (SDA) and clock (SCL) line. The I2C bus interfaces to external I2C devices such as serial memory (RAMs and ROMs), networking devices, LCDs, tone generators, and so on. The I2C bus can also be used for system testing and diagnostic purposes in product development and manufacturing. This I2C peripheral can both transmit to and receive from other I2C devices on the bus.

The controller includes I2C modules with the following features:

- Devices on the I2C bus can be designated as either a controller or a target.
  - Supports both transmitting and receiving data as either a controller or target with 7-bit addressing
- Support four I2C modes
  - Controller transmit
  - Controller receive
  - Target transmit

  - Target receive
- Supported transmission speeds: Standard (100 kbps), FastMode (400 kbps), FastMode+(1 Mbps)
- FIFOs for receive and transmit data, 8 bytes
- Glitch suppression
- Independent controller and target interrupt generation
- Controller operation with arbitration, clock synchronization, multi-controller support
- Hardware support for SMBus
  - Clock low timeout interrupt
  - Dual target address capability
  - Quick command capability
- Hardware support for DMA with separate channels for transmit and receive

3.8.3 UART

This interface can be used transfer data between a MSPM0 device and another device with serial asynchronous communication protocols like LIN (local interconnection network), ISO7816 (Smart card protocol), IrDA (infrared data association), hardware flow control (CTS/RTS) and multiprocessor communications are supported.

The UART controller different features in MSPM0:

- Separated transmit and receive 4 depth FIFOs reduce CPU interrupt service loading
- Supports DMA data transfer
- Support loop back mode operation
- Support hardware flow control
- Support 9-bit multi-drop configuration
- Protocols supported:
  - Local interconnect network (LIN) support
  - DALI
  - IrDA
  - ISO/IEC 7816 Smart card
  - RS485
  - Manchester coding
  - Idle-Line Multiprocessor

3.8.4 CAN FD

Unlike MSP430, CAN control is integrated on MSPM0 devices. Controller Area Network (CAN) is a serial communications protocol that efficiently supports distributed real-time control with a high level of reliability. CAN has high immunity to electrical interference and the ability to detect various type of errors. In CAN, many short messages are broadcast to the entire network, which provides data consistency in every node of the system.

The MCAN module supports both classic CAN and CAN FD (CAN with flexible data rate) protocols. The CAN FD feature allows higher throughput and increased payload per data frame. Classic CAN and CAN FD devices may coexist on the same network without any conflict provided that partial network transceivers, which can detect and ignore CAN FD without generating bus errors, are used by the classic CAN devices. The MCAN module is compliant to ISO 11898-1:2015.

The details of CAN controller and its features. Refer to the MSPM0 technical reference manual.

3.9 BSL

Table 3-8 compares the MSP430 and MSPM0 BSL implementations.

Table 3-8. BSL Comparison

			MSP430									MSP432	MSPM0
			F20xx, G2xx0, G2xx1, G2xx2, I20xx	F1xx, F2xx, F4xx, G2xx3	F5xx, F6xx		FR5xx, FR6xx		FR231x, FR242x, FR243x FR25xx, FR263x	FR215x, FR235x, FR247x, FR267x	FR20xx, FR21xx, FR41xx	P4xx	M0Gxxx M0Lxxx
					Non-USB	USB	Factory	Crypto-Boot-loader					
General	BSL memory type		No BSL	ROM	Flash	Flash	ROM	FRAM	ROM	ROM	ROM	Flash	ROM
	BSL memory size		N/A	1 KB	2 KB	2 KB	2 KB	4 KB	3 KB	3 KB	1 KB	8 KB	5K
	User configuration									✓		✓	✓
	UART			✓	✓		✓	✓	✓	✓	✓	✓	✓
	I2C				✓		✓	✓	✓	✓		✓	✓
	SPI											✓	
	USB					✓							
	Flash based interface plugin supported												✓
Invoke mechanism	Entry sequence on I/Os	Sequence on TEST/RST		✓	✓		✓	✓	✓	✓	✓		
		PUR pin tied to VUSB				✓							
		Sequence on defined I/O						✓				✓	✓
	Empty reset vector invokes BSL					✓		✓	✓	✓		✓	✓
	Calling BSL from software application			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Tools Support	Hardware	MSP-BSL 'Rocket'		✓	✓		✓	✓	✓	✓	✓	✓	
		MSP-FET			✓		✓	✓	✓	✓	✓	✓	(1)
		USB cable				✓							
		USB-to-Serial Converter		✓									✓
		XDS110											✓
	Software	BSL Scripter			✓	✓	✓	✓	✓	✓	✓	✓	
		BSLDEMO		✓									
		MSPM0 BSL GUI											✓
Security	Password protection (bytes)			32	32	32	32		32	32	32	256	32
	Mass erase on incorrect password			✓	✓	✓	✓		✓	✓	✓	✓	✓(2)
	BSL payload encryption							✓				✓	
	Update of IP protected regions through boot code											✓	
	Authenticated encryption							✓					
	Additional security							✓					

(1) Can use the USB to UART or I2C channel to communicate with MSPM0 device by BSL.
(2) Three password failures trigger security actions that include factory reset.

3.10 Analog Peripherals

3.10.1 SAR ADC

Table 3-9 compares the MSP430 and MSPM0 SAR ADC implementations.

Table 3-9. ADC Comparison

Feature	MSP430	MSPM0L	MSPM0G
Resolution	10 or 12 bit	10 or 12b bit	10 or 12 bit
Sample rate	200 ksps	1 Msps	12 bit 4 Msps
FIFO mode	Not supported	FIFO	FIFO
Internal REF	1.5 V, 2.0 V, 2.5V	1.4 V, 2.5 V	1.4 V, 2.5 V
Oversampling	Not supported	Supported	Supported
Window comparator	Supported on some devices	Supported	Supported
Simultaneous sample	Not supported	Not supported	Supported

3.10.1.1 Simultaneous Sampling

Some applications, such as current and voltage sensing, require measurement from multiple analog signals at the exact same time. In these circumstances, using multiple ADCs on a single MCU to perform simultaneous sampling is a requirement. Any device with multiple ADC peripherals in the MSPM0xx platform supports simultaneous sampling.

3.10.1.2 Window Comparator

There is one window comparator unit available in the ADC which can be used to check if the input signal is within predefined threshold values set by software. Both MSPM0L and MSPM0G devices support this feature. The ADC result that goes into MEMRES or FIFO is what gets checked against the threshold values of the window comparator.

Based on the comparison it can generate 3 interrupt conditions:

1. LOWIFG – Conversion result is below the Low threshold (WCLOW)
2. HIGHIFG – Conversion result is above the High threshold (WCHIGH)
3. INIFG – Conversion result is in between or equal to the Low and High thresholds

3.10.2 COMP

MSPM0 devices have a high-speed comparator that has better performance than MSP430 devices.

Table 3-10. COMP Comparison

	MSP430	MSPM0L	MSPM0G
Multiple input sources	Yes	Yes	Yes
Software LPF for COMP	Yes	Yes	Yes
Reference Voltage	VREF and VDD	VDD	VREF and VDD
Internal DAC	6 bit	8 bit	8 bit
Propagation delay(high-speed)	120 ns to 1 µs, depends on device	40 ns	40 ns
Window compare mode	No	No	Yes
Voltage hysteresis	Programmable	Programmable	Programmable

3.10.2.1 Window Compare Mode

Some MSPM0G MCUs have more than one comparator. The purpose of window comparator mode is to monitor the input signal within a specified threshold range defined by lower and upper thresholds.

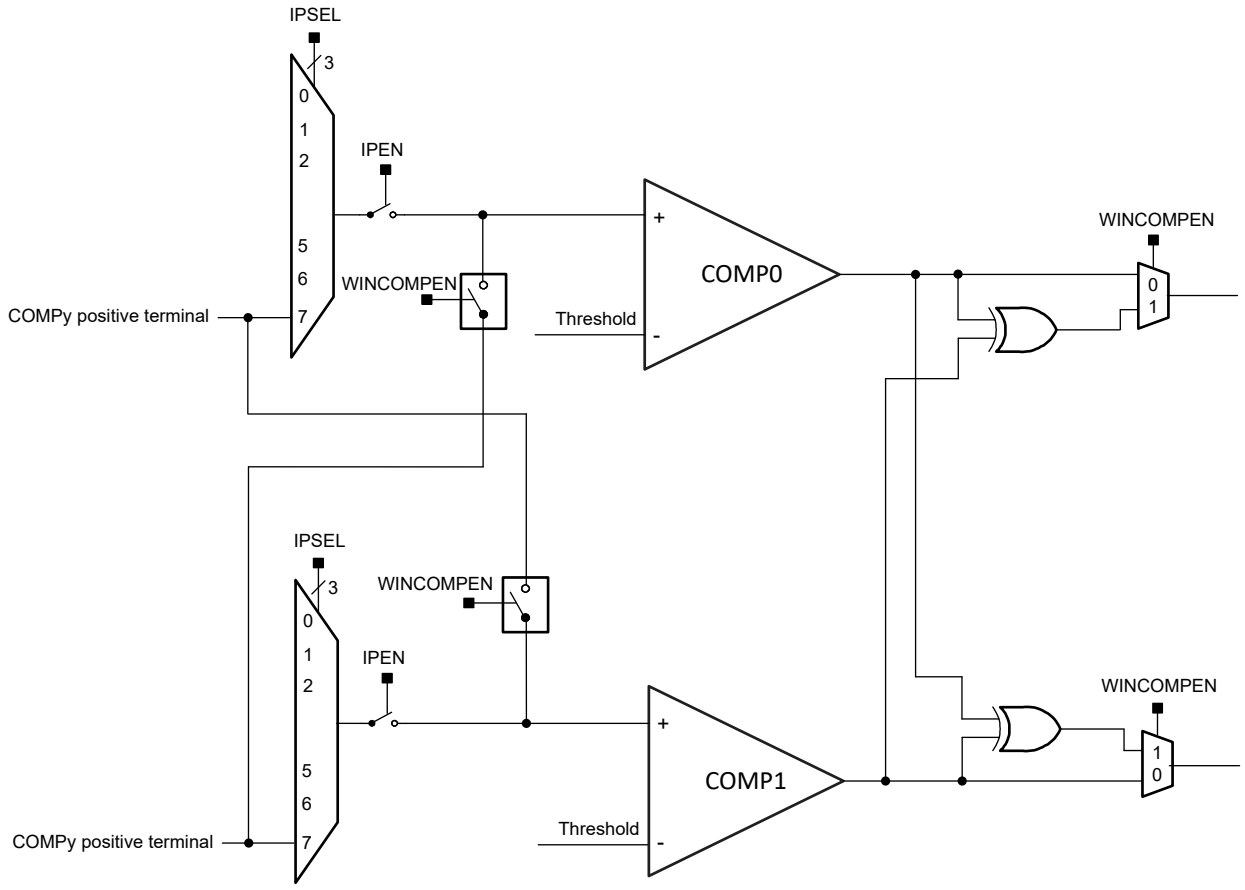

Figure 3-4. Window Compare Mode Block Diagram

3.10.3 OPA

MSP430 and MSPM0 MCUs are rich in analog peripherals. These peripherals have excellent ability to handle mixed signals which benefit by internal analog and digital IP especially OPA.

Table 3-11 compares the different between MSP430 and MSPM0 op amps.

Table 3-11. Op Amp Comparison

	MSP430	MSPM0L, MSPM0G	
	SAC-OA	OPA	GPAMP
GBW	2.8 MHz	6 MHz	350 kHz
PGA	33x	32x	NA
Interaction	Yes	Yes	Yes
Burnout detection	No	Yes	No

Figure 3-5 shows sample circuits for the MSPM0 OPA in typical use cases.

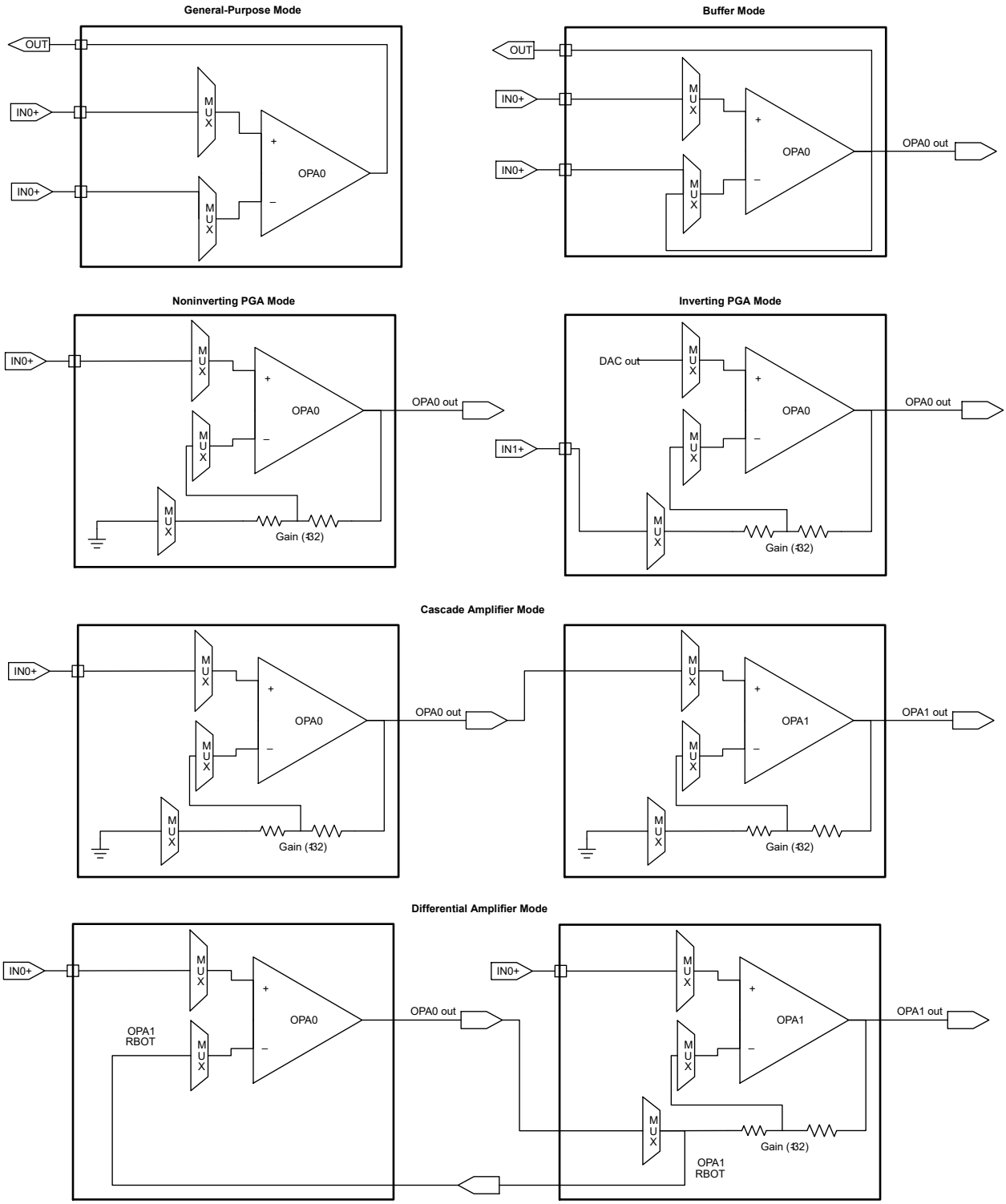

Figure 3-5. Typical Use Cases for Op Amps

3.11 Timers

Table 3-12 compares the MSPM0 timers (TIMA, TIMH, TIMG) with the MSP430 timers (Timer_A, Timer_B).

Table 3-12. Timer Comparison

	MSP430		MSPM0L and MSPM0G		
	Timer_A	**Timer_B**	**TIMA**	**TIMG**	**TIMH**
Counter	16 bit	16 bit	16 bit	16 bit	24 bit
Count mode	Up or Up-Down	Up or Up-Down	Down or Up-Down	Down or Up-Down	Down or Up-Down
Compare mode	Yes	Yes	Yes	Yes	Yes
PWM output	Yes	Yes	Yes	Yes	Yes
Synchronize	No	No	Yes	Yes	Yes
Cross trigger	No	No	Yes	Yes	Yes
Complementary PWM with programming deadband	No	No	Yes	No	No
QEI	No	No	No	Yes	No

MSPM0 timers support advanced feature like complementary PWM, deadband, and cross trigger that can be used for motor control and other complicated PWM control. See the family TRMs and application notes for more information.

3.12 Hardware Design Guide

For more information on hardware design considerations, refer to these documents:

MSPM0 L-Series MCUs Hardware Development Guide

MSPM0 G-Series MCUs Hardware Development Guide

These guides describe the considerations for hardware development with MSPM0 MCUs, including detailed hardware design information for power supplies, reset circuitry, clocks, debugger connections, key analog peripherals, communication interfaces, GPIOs, and board layout guidance.

4 Revision History

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

Changes from Revision * (December 2022) to Revision A (March 2023)	**Page**
• Updated links and images throughout	1

IMPORTANT NOTICE AND DISCLAIMER

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

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

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

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

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

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