Application Note

Migration Guide From STM32® to Arm®-Based MSPM0

Owen Li

ABSTRACT

This application note assists with migrating from the STMicroelectronics STM32® platform to the Texas Instruments MSPM0 MCU ecosystem. This guide introduces the MSPM0 development and tool ecosystem, core architecture, peripheral considerations, and software development kit. The intent is highlight the differences between the two families and to leverage existing knowledge of the STM32 ecosystem to quickly ramp with the MSPM0 series of MCUs.

Table of Contents

1 MSPM0 Portfolio Overview....................3
1.1 Introduction....................3
1.2 Portfolio Comparison of STM32 MCUs to MSPM0 MCUs....................3
1.3 Pin to Pin Comparison of STM32 MCUs to MSPM0 MCUs....................4
2 Ecosystem and Migration....................5
2.1 Software Ecosystem Comparison....................5
2.2 Hardware Ecosystem....................6
2.3 Debug Tools....................7
2.4 Migration Process....................8
2.5 Migration and Porting Example....................9
3 Core Architecture Comparison....................17
3.1 CPU....................17
3.2 Embedded Memory Comparison....................17
3.3 Power Up and Reset Summary and Comparison....................19
3.4 Clocks Summary and Comparison....................21
3.5 MSPM0 Operating Modes Summary and Comparison....................23
3.6 Interrupt and Events Comparison....................24
3.7 Debug and Programming Comparison....................26
4 Digital Peripheral Comparison....................28
4.1 General-Purpose I/O (GPIO, IOMUX)....................28
4.2 Universal Asynchronous Receiver-Transmitter (UART)....................29
4.3 Serial Peripheral Interface (SPI)....................29
4.4 I²C....................30
4.5 Timers (TIMGx, TIMAx)....................31
4.6 Windowed Watchdog Timer (WWDT)....................32
4.7 Real-Time Clock (RTC)....................32
5 Analog Peripheral Comparison....................34
5.1 Analog-to-Digital Converter (ADC)....................34
5.2 Comparator (COMP)....................35
5.3 Digital-to-Analog Converter (DAC)....................36
5.4 Operational Amplifier (OPA)....................37
5.5 Voltage References (VREF)....................37
6 Summary....................38
7 References....................38
8 Revision History....................39

List of Figures

Figure 2-1. MSPM0 SysConfig....................6
Figure 2-2. LP-MSPM0G3507 LaunchPad Development Kit....................7

Figure 2-3. MSPM0 Debugging........8
Figure 2-4. MSPM0 Migration Flowchart........8
Figure 2-5. Code Composer Studio IDE........9
Figure 2-6. uart_echo_interrupts_standby Example........11
Figure 2-7. Power Mode Configuration........12
Figure 2-8. UART Configuration........13
Figure 2-9. GPIO Configuration........14
Figure 2-10. Changes to Application Code........15
Figure 2-11. 16
Figure 3-1. MSPM0 Reset Levels........21
Figure 3-2. Generic Event Route........26

List of Tables

Table 1-1. Comparison of the TI MSPM0Gx, MSPM0Lx, MSPM0Cx, MSPM0Hx and STM32G0, STM32F0, STM32C0 Series........3
Table 1-2. Pin to Pin Compatibility Between STM and MSPM0........4
Table 2-1. STM32 Software Tool Equivalents for MSPM0........5
Table 2-2. MSPM0 Supported IDEs........5
Table 3-1. Comparison of CPU Feature Sets........17
Table 3-2. Comparison of Flash Feature........17
Table 3-3. Comparison of SRAM Features........19
Table 3-4. Comparison of Power-up........20
Table 3-5. Comparison of Reset Domains........20
Table 3-6. Oscillator Comparisons........22
Table 3-7. Clock Comparison........22
Table 3-8. Peripheral Clock Sources........22
Table 3-9. Operating Modes Comparison Between STM32G0 and MSPM0 Devices........23
Table 3-10. Interrupt Comparison........24
Table 3-11. Arm SWD JTAG Feature Comparison........27
Table 3-12. BSL Feature Comparison........27
Table 4-1. GPIO Feature Comparison........28
Table 4-2. UART Naming Differences Between STM32G0 and MSPM0........29
Table 4-3. UART Advanced Feature Set Comparison........29
Table 4-4. UART Standard Feature Set Comparison........29
Table 4-5. SPI Feature Comparison........30
Table 4-6. I^2C Feature Comparison........30
Table 4-7. Timer Naming........31
Table 4-8. Timer Feature Comparison........31
Table 4-9. Timer Module Replacement........31
Table 4-10. Timer Use-Case Comparisons........32
Table 4-11. WWDT Naming........32
Table 4-12. WDT Feature Comparison........32
Table 4-13. RTC Feature Comparison........33
Table 5-1. Feature Set Comparison........34
Table 5-2. Conversion Modes........34
Table 5-3. COMP Feature Set Comparison........35
Table 5-4. DAC Feature Set Comparison........36
Table 5-5. MSPM0 OPA Feature Set........37
Table 5-6. Feature Set Comparison........37
Table 5-7. Control Bit Comparison........38

Trademarks

MSP430™, TI E2E™, Code Composer Studio™, LaunchPad™, EnergyTrace™, and BoosterPack™ are trademarks of Texas Instruments.
STM32® is a registered trademark of STMicroelectronics International N.V.
Arm® and Cortex® are registered trademarks of Arm Limited.
All trademarks are the property of their respective owners.

1 MSPM0 Portfolio Overview

1.1 Introduction

The MSP430™ MCUs have nearly 30 years of history as TI's classic microcontroller. The latest generation introduces the MSPM0 family. MSPM0 microcontrollers (MCUs) 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. 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 project requirements. The MSPM0 MCU family combines the Arm Cortex-M0+ platform with a ultra-low-power system architecture, allowing system designers to increase performance while reducing energy consumption.

The MSPM0 MCUs offer a competitive alternative to the STM32 MCUs. This application note assists with migration from STM32 MCUs to MSPM0 MCUs by comparing device features and ecosystems.

1.2 Portfolio Comparison of STM32 MCUs to MSPM0 MCUs

Table 1-1. Comparison of the TI MSPM0Gx, MSPM0Lx, MSPM0Cx, MSPM0Hx and STM32G0, STM32F0, STM32C0 Series

	ST Micro STM32G0 Series	ST Micro STM32F0 Series	ST Micro STM32C0 Series	TI MSPM0 MSPM0Gx Series	TI MSPM0 MSPM0Lx Series	TI MSPM0 MSPM0Cx Series	TI MSPM0 MSPM0Hx Series
Core or Frequency	CM0+ , 64 MHz	CM0, 48 MHz	CM0+, 48MHz	CM0+, 80 MHz	CM0+, 32 MHz	CM0+, 24-32MHz	CM0+ , 32MHz
Supply voltage	1.7V to 3.6V	2V to 3.6V	2V to 3.6V	1.62V to 3.6V	1.62V to 3.6V	1.62V to 3.6V	4.5V to 5.5V
Temperature	-40°C to 125°C	-40°C to 105°C	-40°C to 125°C	-40°C to 125°C	-40°C to 125°C	-40°C to 125°C	-40°C to 125°C
Memory	512KB to 16KB	256KB to 16KB	256KB to 16KB	512KB to 32KB	256KB to 8KB	64KB to 8KB	64KB to 32KB
RAM	Up to 144KB	Up to 32KB	Up to 32KB	Up to 128KB	Up to 32KB	Up to 8KB	Up to 8KB
GPIO (max)	90	88	61	94	73	45	45
Analog	1x 2.5-Msps 12-bit ADC 1x 12-bit DAC 3x comparators	1x 1-Msps 12-bit ADC 1x 12-bit DAC 2x comparators	1x 2.5-Msps 12-bit ADC	2x 4-Msps 12-bit ADC 1x 12-bit DAC 3x high-speed comparators 2x op amps	1x 1.68-Msps 12-bit ADC 1x high-speed comparator 1x general purpose amp 2x op amps	1x 1.68-Msps 12-bit ADC 1x high-speed comparator	1x 1.6-Msps 12-bit ADC
Communication (max)	3x SPI 3x I2C Fast+ 6x UART (LIN) 2x CAN-FD 1x USB	2x SPI 2x I2C Fast+ 8x UART (LIN) 1x CAN	2x SPI 2x I2C Fast+ 4x UART (LIN) 1x CAN-FD	3x SPI 3x I2C Fast+ 7x UART 2x UART (LIN) 2x CAN-FD	2x SPI 3x I2C Fast+ 5x UART 2x UART (LIN)	1x SPI 2x I2C Fast+ 3x UART 1x UART (LIN)	1x SPI 2x I2C Fast+ 3x UART 1x UART (LIN)
Timers	8	4	10	7	5	4	5
Advance Timers	Yes (1x)	Yes (1x)	Yes (1x)	Yes (2x)	Yes (1x)	Yes (1x)	Yes (1x)
Hardware Accelerator	N/A	N/A	N/A	MATHACL	N/A	N/A	N/A
Security	CRC, TRNG, AES256	CRC	CRC	CRC, TRNG, AES256	CRC, TRNG, AES256	CRC	CRC
Low power	Active: 100µA/MHz Standby (RTC): 1.5µA	Active: 281µA/MHz Standby (RTC): 2.5µA	Active: 67.5µA/MHz Standby (RTC):1.4µA	Active: 123µA/MHz Standby (RTC): 1.7µA	Active: 106µA/MHz Standby: 1.1µA	Active: 100µA/MHz Standby: 2.5µA	Active: 125µA/MHz Standby: 3.6µA

1.3 Pin to Pin Comparison of STM32 MCUs to MSPM0 MCUs

The MSPM0 MCUs also offer pin to pin compatibility to the STM32 MCUs that can be found in the following table.

Table 1-2. Pin to Pin Compatibility Between STM and MSPM0

STM Family	P2P MSPM0 MCU	MSP Family	Package
STM8S003F3P6	MSPS003F3SPW20R MSPS003F4SPW30R	MSPM0C1103/4	20-TSSOP
STM32C031C6	MSP32C031C6SPTR	MSPM0C1105/6	48-LQFP
STM32G031C6 STM32G031C8	MSP32G031C6SPTR MSP32G031C8SPTR	MSPM0C1105/6	48-LQFP
STM32C031K6	MSP32C031K6SVFCR	MSPM0C1105/6	32-LQFP
STM32G031K6 STM32G031K8	MSP32G031K6SVFCR MSP32G031K8SVFCR	MSPM0C1105/6	32-LQFP

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 Software Ecosystem Comparison

Table 2-1. STM32 Software Tool Equivalents for MSPM0

	STM32	MSPM0
IDE	CubeIDE	Code Composer Studio™ IDE (CCS)
Software Configuration	CubeMX	SysConfig
Stand-alone programming	CubeProgrammer	UniFlash
Display/Demo GUI Editor	CubeMonitor	GuiComposer

2.1.1 MSPM0 Software Development Kit (MSPM0 SDK)

The MSPM0 SDK delivers software APIs, examples, documentation, and libraries that help engineers quickly 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. Additionally, interactive MSP Academy trainings are included in the MSPM0 SDK to provide a guided learning path.

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 categories such as lower-level DriverLib examples, higher-level TI Drivers examples, and examples for middleware such as GUI Composer, LIN, IQMath, and others. For details, refer to the *MSPM0 SDK User's Guide*.

2.1.2 CubeIDE vs Code Composer Studio IDE (CCS)

Code Composer Studio IDE (CCS) is TI's equivalent of STM32's CubeIDE. CCS is a free Eclipse-based IDE that supports TI's microcontroller (MCU) and embedded processor portfolios. CCS comprises a suite of tools used to develop and debug embedded applications including an optimizing C/C++ compiler, source code editor, project build environment, debugger, profiler and many other features. CCS is available as both a desktop or cloud-based IDE.

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. CCS offers an all-in-one development tool experience.

In addition to CCS, MSPM0 devices are also supported in industry-standard IDEs listed in the following table.

Table 2-2. MSPM0 Supported IDEs

IDE	MSPM0
CCS	✓
IAR	✓
Keil	✓

2.1.3 CubeMX vs SysConfig

SysConfig is an intuitive and comprehensive collection of graphical utilities for configuring pins, peripherals, radios, subsystems, and other components. This is TI's equivalent of STM32 CubeMX. SysConfig helps manage, expose, and resolve conflicts visually so that you have more time to create differentiated applications. The tool output includes C header and code files that can be used with MSPM0 SDK examples or used to configure custom software. SysConfig is integrated into CCS but can also be used as a standalone program.

For details, refer to the MSPM0 SysConfig Guide.

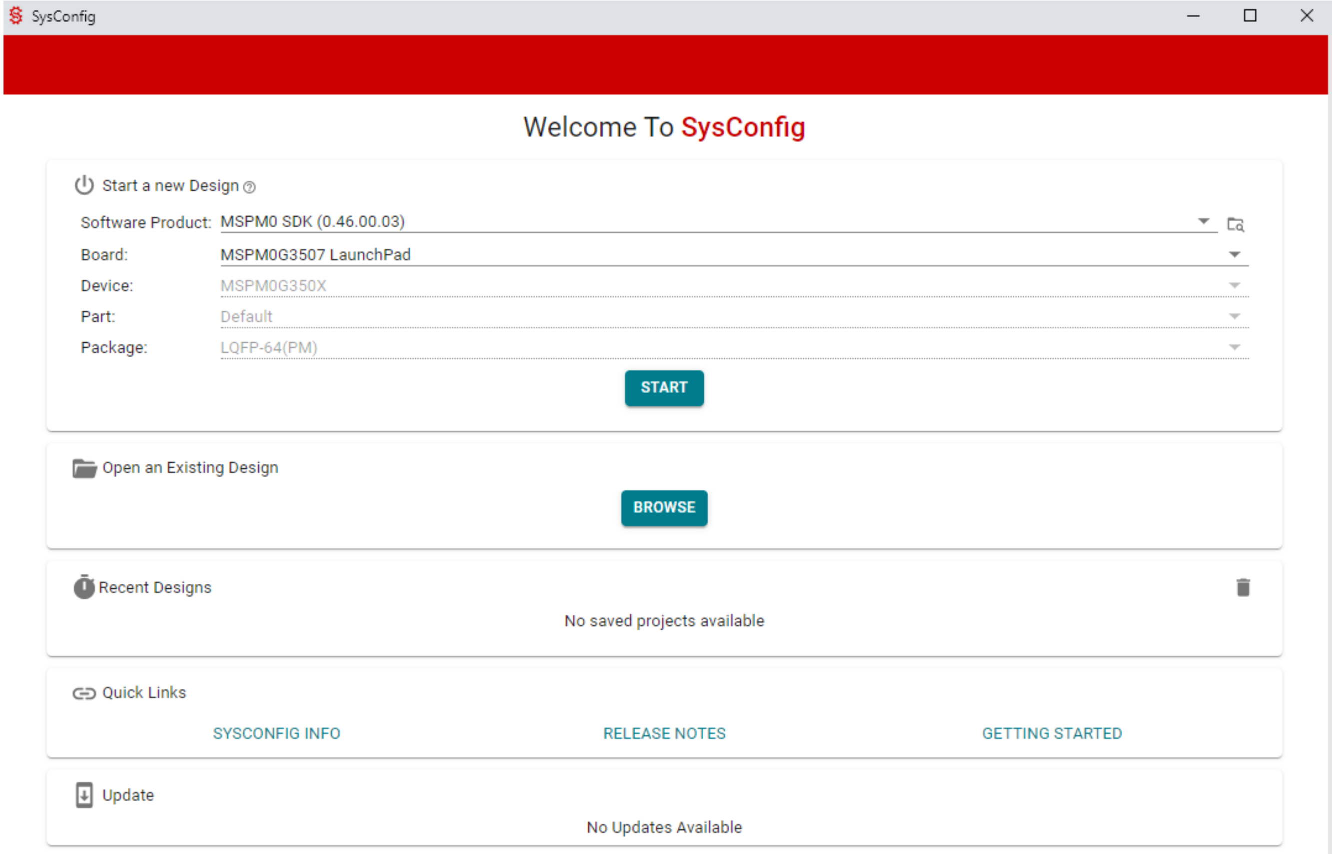


Figure 2-1. MSPM0 SysConfig

2.2 Hardware Ecosystem

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. Add features like wireless connectivity, graphical displays, environmental sensing, and more.

- LP-MSPM0G3507 LaunchPad development kit
- LP-MSPM0G3519 LaunchPad development kit
- LP-MSPM0L1117 LaunchPad development kit
- LP-MSPM0L1306 LaunchPad development kit
- LP-MSPM0L2228 LaunchPad development kit
- LP-MSPM0C1104 LaunchPad development kit
- LP-MSPM0C1106 LaunchPad development kit
- LP-MSPM0H3216 LaunchPad development kit

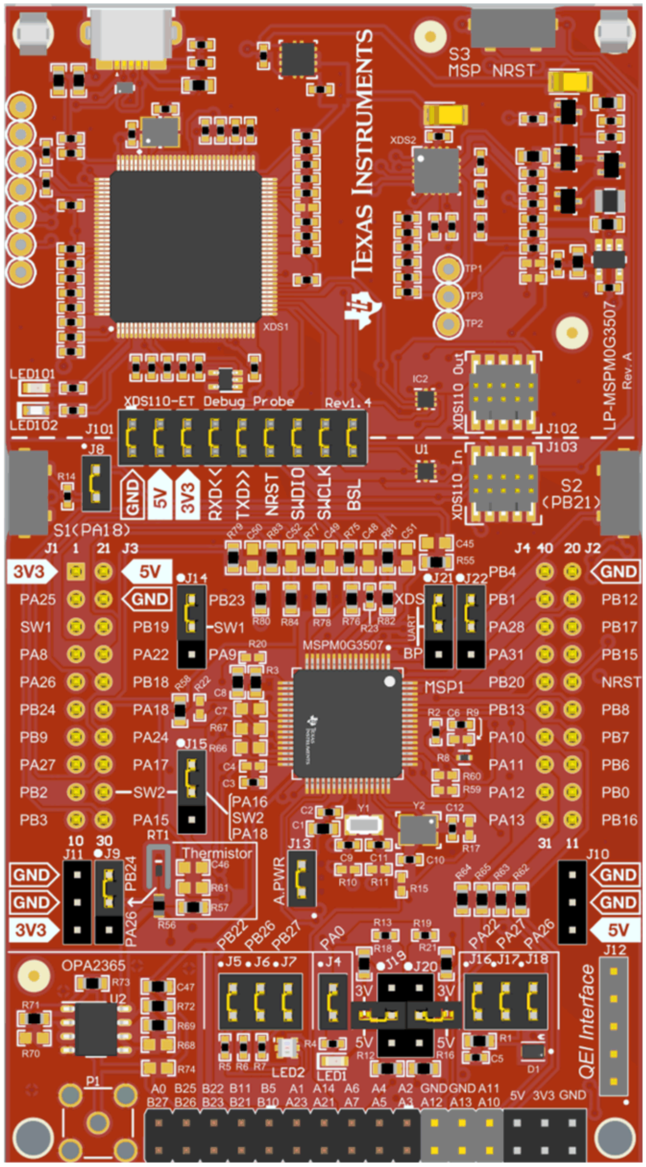


Figure 2-2. LP-MSPM0G3507 LaunchPad Development Kit

2.3 Debug Tools

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 (using EnergyTrace technology). Figure 2-3 shows the connection of the debugger.

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

The Texas Instruments' XDS110 is designed for TI embedded processors. XDS110 connects to the target board through a TI 20-pin connector (with multiple adapters for TI 14-pin and Arm 10-pin and Arm 20-pin) and to the host PC through USB2.0 High Speed (480 Mbps). XDS110 supports a wider variety of standards (IEEE1149.1, IEEE1149.7, SWD) in a single pod. All XDS debug probes support Core and System Trace in all Arm and DSP processors that feature an Embedded Trace Buffer (ETB). For details, refer to 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, refer to Segger J-Link Debug Probes page.

Figure 2-3 shows a high-level diagram of the major functional areas and interfaces of the XDS110 probe to MSPM0 target.

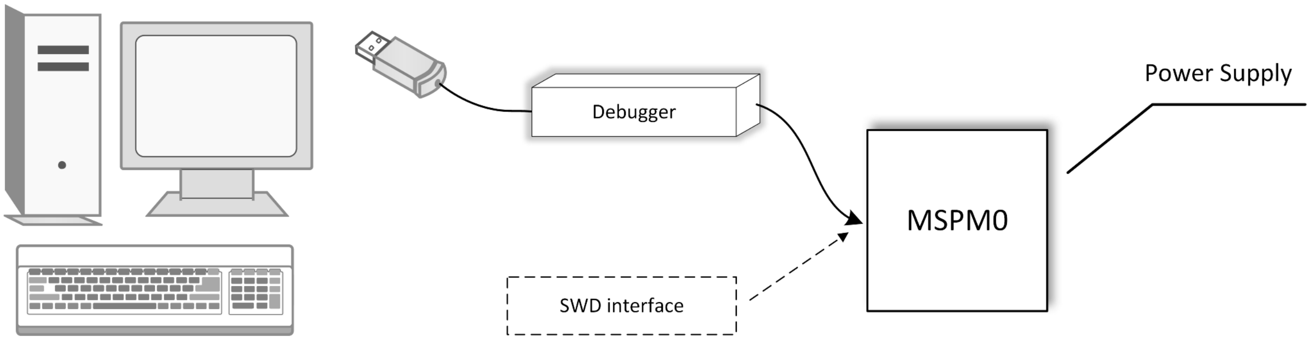


Figure 2-3. MSPM0 Debugging

2.4 Migration Process

The first step in migrating is to review the portfolio and choose the best MSPM0 MCU. After an MSPM0 MCU has been selected, choose a development kit. Development kits include a LaunchPad kit available for purchase and design files for a Target-Socket Board. TI also provides a free MSPM0 Software Development Kit (SDK), which is available as a component of Code Composer Studio IDE desktop and cloud version within the TI Resource Explorer. Use the peripheral sections of this application note for help with porting software from STM32 to MSPM0. Finally, once the software ported, download and debug the application with our debugging tools.

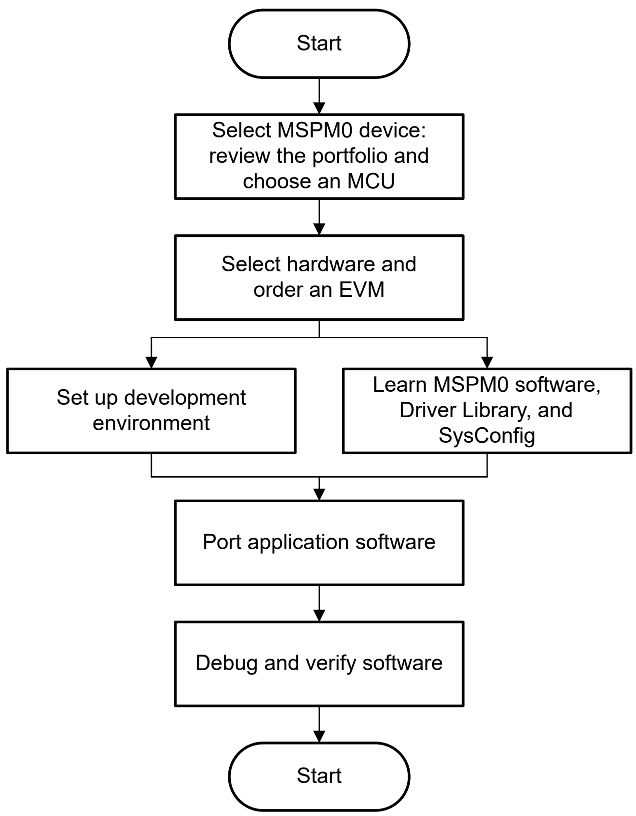


Figure 2-4. MSPM0 Migration Flowchart

2.5 Migration and Porting Example

To become more familiar with the TI ecosystem and explain how to best get started with MSPM0, this section describes the step-by-step migration process of a basic application.

To demonstrate the process of porting from STM32 to MSPM0, this description includes the steps to port a basic low-power UART monitor application from an STM32G0x to a MSPM0 device using an existing ST UART example as the starting point.

1. *Choose the right MSPM0 MCU*

   The first step of migration is to choose the correct MSPM0 device for the application. To do this, the portfolio section of this guide can be used to choose a MSPM0 family. To narrow down to a specific device using the product selection tool. STM32G0 and MSPM0 share the M0+ core, but features such as memory size, power, and key peripherals must also be considered. MSPM0 also offers many pin-to-pin scalable options, providing the ability to easily scale to larger or smaller memory devices without changing anything else in the system. For purposes of this example, the MSPM0G3507 ws chosen as the best fit for this application.
2. *Select hardware and order an EVM*

   Using an evaluation module (EVM) can expedite the migration process. For the MSPM0 MCUs, a LaunchPad kit is the easiest hardware to begin on. LaunchPad kits are easy-to-use because these come with a built-in programmer and are designed to enable rapid development. The MSPM0G3507 has a LaunchPad development kit (LP-MSPM0G3507) that can be used for porting the software.
3. *Setup software IDE and SDK*

   Before the software can be ported, a software development environment must be chosen and setup. Section 2.1 shows all of the IDEs supported by MSPM0. The migration and porting process is similar for any IDE that is chosen. Make sure the latest version of the MSPM0 SDK is used. For this example, TI's CCS is the chosen IDE.

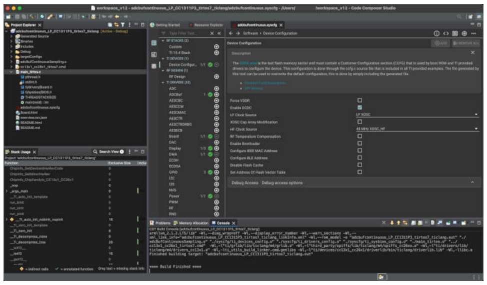


Figure 2-5. Code Composer Studio IDE

4. *Software porting*

   When the environment is ready, start using the MSPM0 SDK. As mentioned, the MSPM0 SDK is similar to the STM32Cube software package. The MSPM0 SDK offers different layers for software development. MSPM0 TI Drivers operate at a similar level to STM32Cube HAL, while MSPM0 DriverLib is comparable to the STM32Cube low-level drivers. Most MSPM0 users find DriverLib level software is the best fit for the applications, so most MSPM0 software examples are also DriverLib based. This example uses DriverLib.

   One option when porting a project is to try to replace each section of code with equivalent MSPM0 DriverLib APIs, but this is not generally the easiest path. Generally, understand the application code being ported. Then, start with the closest MSPM0 example project and modify to match the original code functionality. This process is going to be shown below using a low-power UART example from STM32CubeG0. For more complex projects using many peripherals, this process is typically repeated for each peripheral.

a. *Understand the application*

The following description is from the example project from STM32CubeG0 named 'LPUART_WakeUpFromStop_Init'.

```
@par Example Description

Configuration of GPIO and LPUART peripherals to allow characters received on LPUART_RX pin
to wake up the MCU from low-power mode. This example is based on the LPUART LL API. The
peripheral initialization uses LL initialization function to demonstrate LL init usage.

LPUART Peripheral is configured in asynchronous mode (9600 bauds, 8 data bit, 1 start bit, 1
stop bit, no parity).
No HW flow control is used.
LPUART Clock is based on HSI.

Example execution:
After startup from reset and system configuration, LED3 is blinking quickly during 3 sec,
then MCU enters "Stop 0" mode (LED3 off). On first character reception by the LPUART from
PC Com port (ex: using HyperTerminal) after "Stop 0" Mode period, MCU wakes up from "Stop 0"
Mode.

Received character value is checked :
- On a specific value ('S' or 's'), LED3 is turned On and program ends.
- If different from 'S' or 's', program performs a quick LED3 blinks during 3   sec and
enters again "Stop 0" mode, waiting for next character to wake up.
```

The first step is to understand the main settings for the MCU. This is generally clock speeds and power policies. In this example, the general clock frequency is not specified because the only important setting is that the UART works in low power Stop0 mode. This states the low power UART clock is based on the 'HIS' or high-speed internal oscillator meaning there is no external crystal being used. The UART runs at 9600 baud, 8 data bits, 1 start and stop bit, no parity. No hardware flow control is used. The application side checks for an 'S' or 's' to be received and blinks an LED.

b. *Find the closest MSPM0 example*

Next step is to understand any differences between the UART modules for STM32G0 and MSPM0 and then find the closest example in the MSPM0 SDK. This is easily accomplished by referring to the UART section in Section 4. This section highlights differences between the UART modules and links to the UART-related MSPM0 SDK code examples. The closest example in the SDK for this example is probably uart_echo_interrupts_standby where the "UART RX/TX echos using interrupts while device is in STANDBY mode".

This MSPM0 example is similar, but not identical to the being ported. This example is going to standby mode, which is a lower power mode than Stop mode. The UART communication settings must be checked as well as which GPIOs are being used. Finally, the application layer of monitoring for a specific character must be added.

c. *Import and modify the example*

Once a similar example is found, Open CCS and import the code example by going to *Project > Import CCS Projects...* and navigate to the MSPM0 SDK example folder. Import the example. Here is the *uart_echo_interrupts_standby* example imported. This is a SysConfig project, so the main C file is simple. The project first calls the SysConfig driverlib initialization which is a function autogenerated by SysConfig to configures the device. Then, this enables the UART interrupt. Finally, this goes to sleep waiting for any UART transaction. If this receives a UART transaction, then this echos the data right back and wakes up.

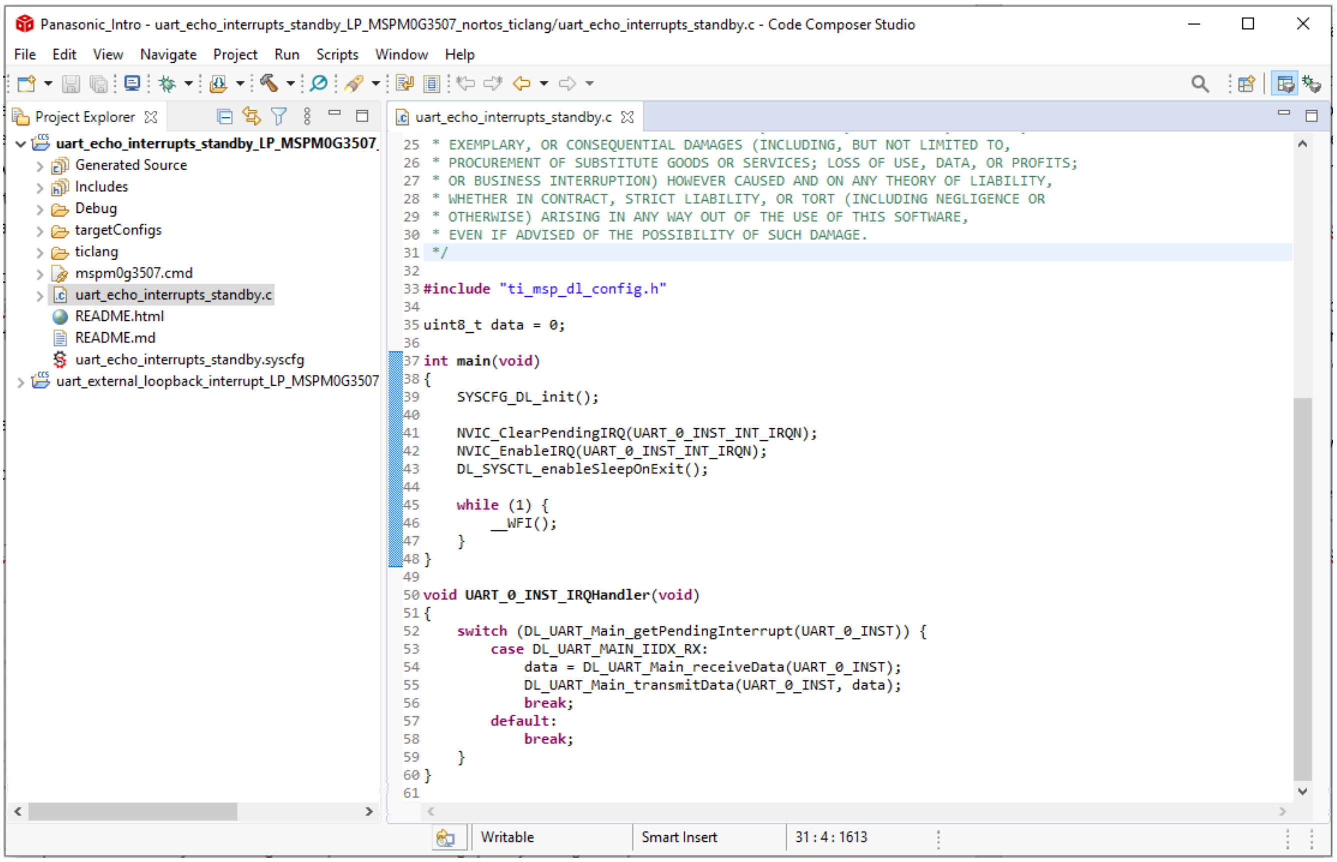


Figure 2-6. uart_echo_interrupts_standby Example

To see the SysConfig configuration, open the .syscfg file, which opens on the SYSCTL tab by default. For detailed guide on using SysConfig, see the SysConfig Guide in the in the MSPM0 SDK.

The first thing to note is the power policy. This MSPM0 example is using Standby0 mode but the goal is to use Stop0 mode. By clicking the drop-down list, the correct low-power mode can be chosen. All of the clocks and oscillators can be configured on this tab as well, but are fine for now.

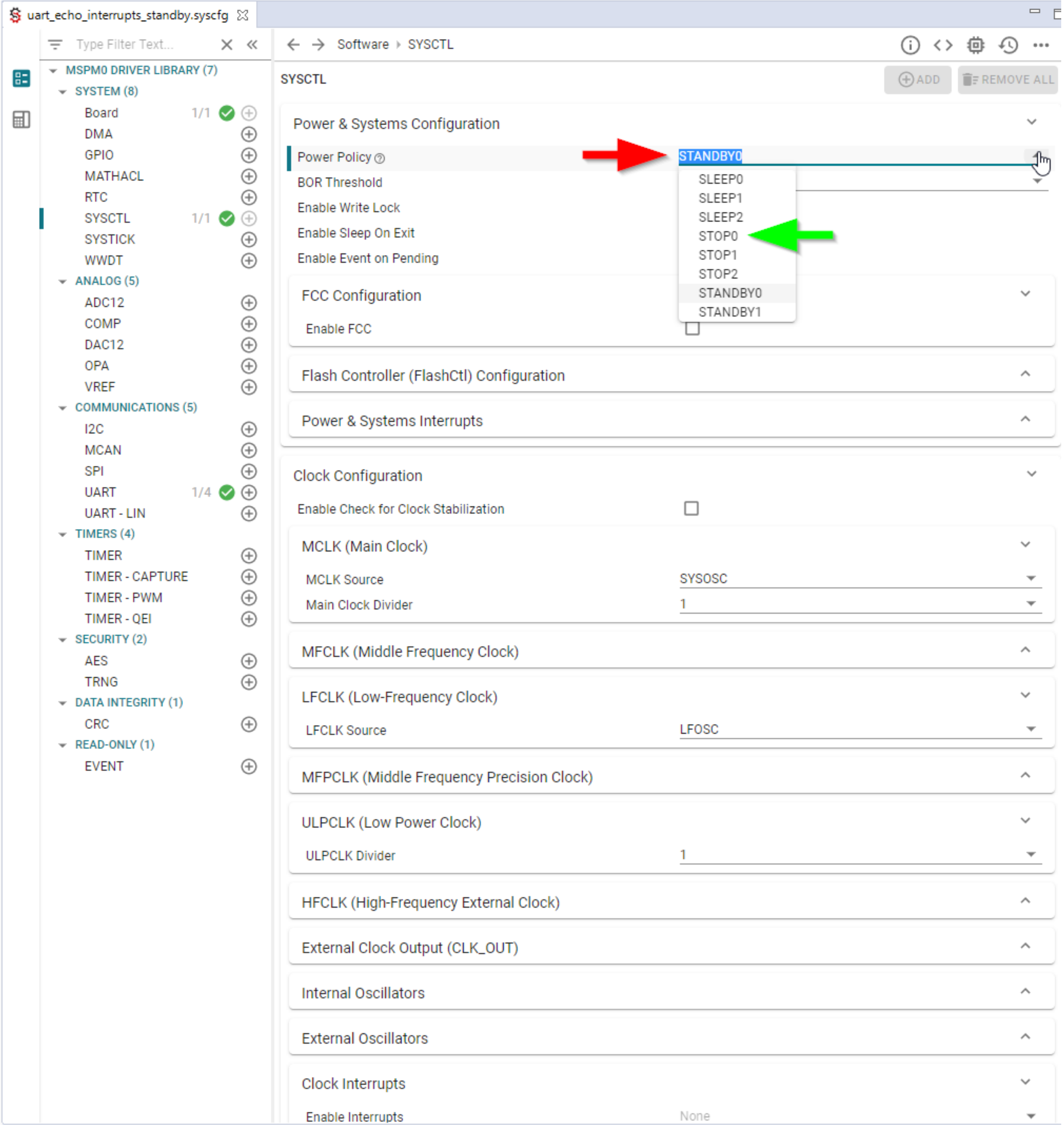


Figure 2-7. Power Mode Configuration

Next, check the UART communication settings on the UART tab (see Figure 2-8). In this case, the baud rate is already set to 9600 and the rest of communication settings are correct. The receive interrupt is already enabled and used in the main program. Also, check the UART module and pins being used by clicking the chip icon in the top right and checking the highlighted pins for the UART. Nothing here needs to be changed as these are already connected to the MSPM0G3507 LaunchPad kit backchannel UART.

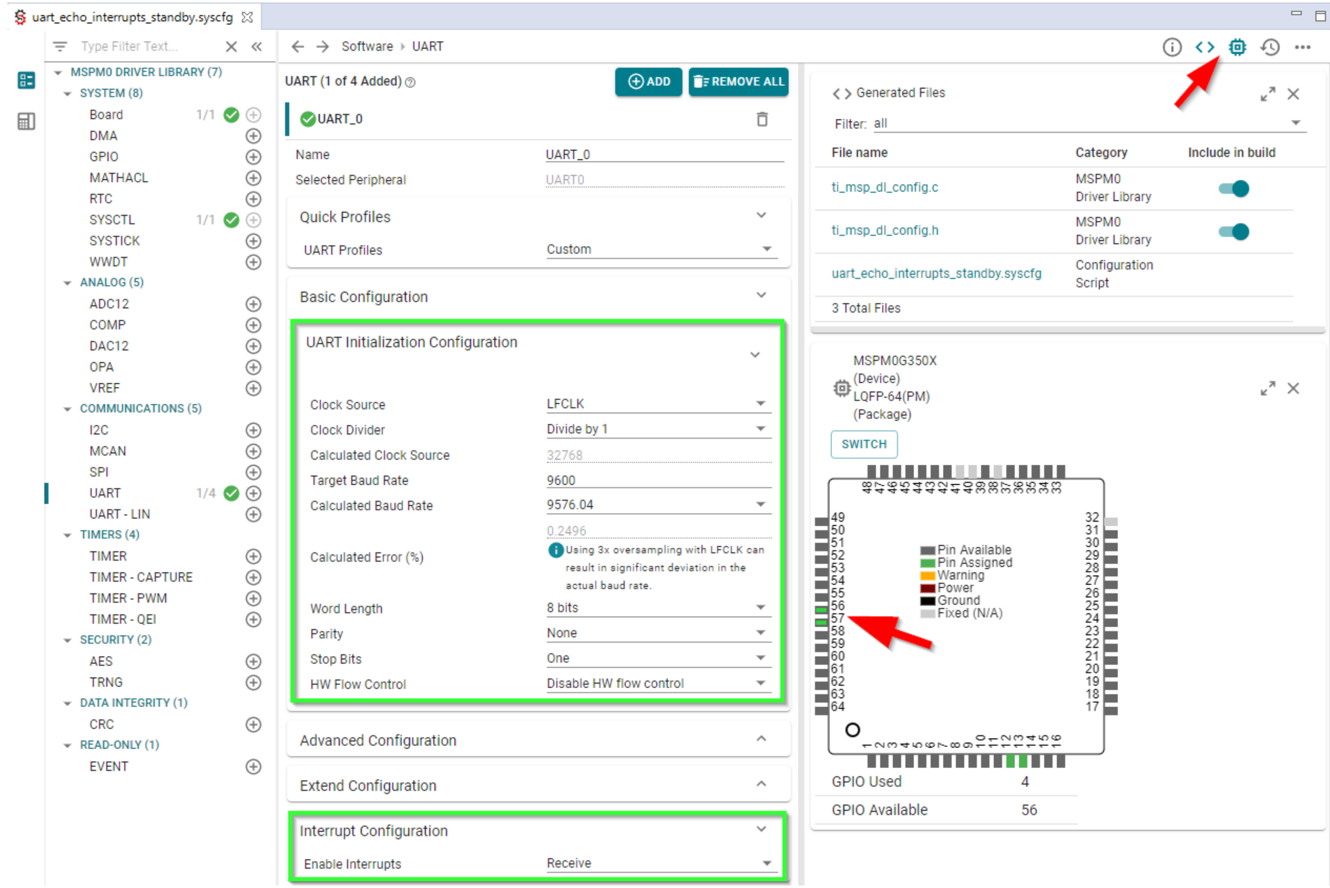


Figure 2-8. UART Configuration

This example currently has no GPIOs configured for driving an LED, but the configuration can be added easily (see Figure 2-9). A GPIO can be added by using the +ADD button at the top of the page. The GPIO port and pin can be named, in this case *LED* and *RED*, respectively. This GPIO is set as an output and then put on Port A pin 0 (PA0.) On the LaunchPad kit, this GPIO is tied to a simple red LED.

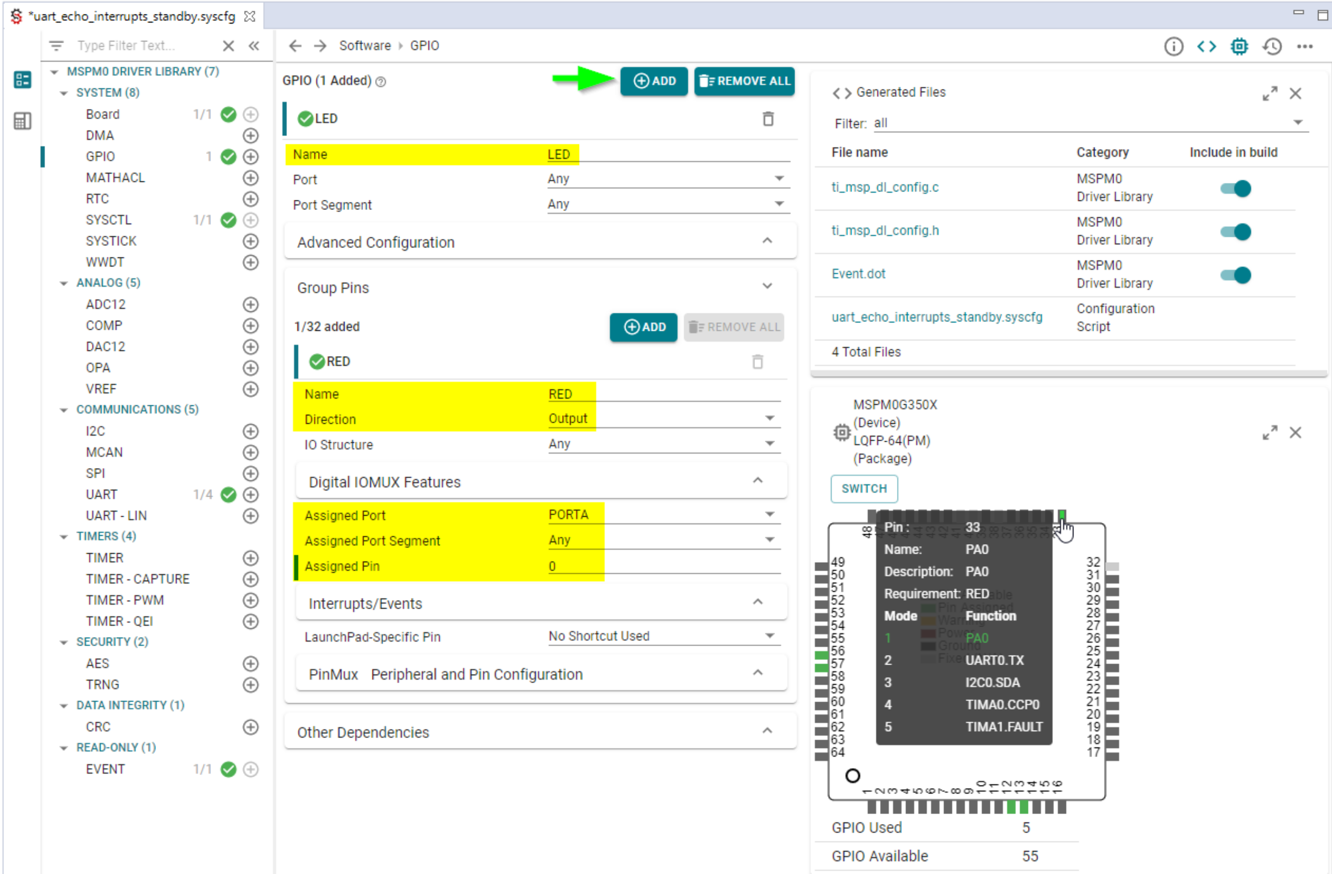


Figure 2-9. GPIO Configuration

When the project is saved and rebuilt, SysConfig updates the ti_msp_dl_config.c and ti_msp_dl_config.h files for the example. At this point, the example hardware configuration has been modified to match the full functionality of the original software being ported. The only remaining effort is application-level software to check the incoming UART bytes and toggle the LED. This is accomplished by moving over a small amount of code into the main C file.

*uart_echo_interrupts_standby.c | uart_echo_interrupts_standby.syscfg

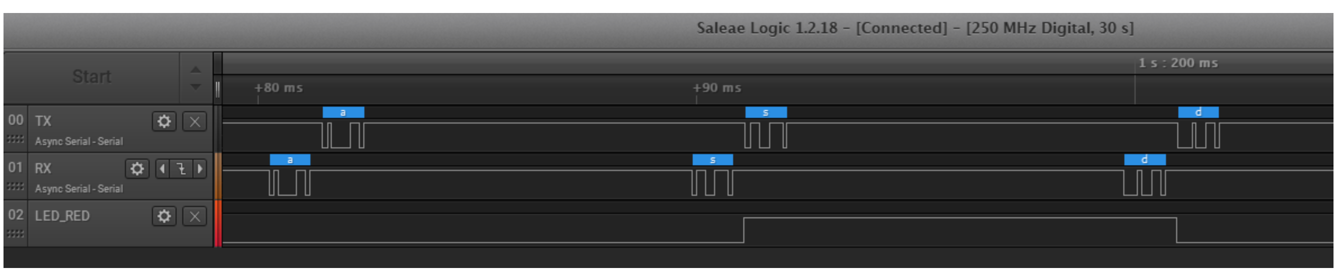
```c
 */

#include "ti_msp_dl_config.h"

uint8_t data = 0;

int main(void)
{
    SYSCFG_DL_init();

    DL_GPIO_clearPins(LED_PORT, LED_RED_PIN);

    NVIC_ClearPendingIRQ(UART_0_INST_INT_IRQN);
    NVIC_EnableIRQ(UART_0_INST_INT_IRQN);
//    DL_SYSCTL_enableSleepOnExit();
    DL_SYSCTL_disableSleepOnExit();

    while (1) {
        __WFI();

        if((data == 'S')||( data == 's')){
            DL_GPIO_setPins(LED_PORT, LED_RED_PIN);

        }else{
            DL_GPIO_clearPins(LED_PORT, LED_RED_PIN);
        }
    }
}

void UART_0_INST_IRQHandler(void)
{
    switch (DL_UART_Main_getPendingInterrupt(UART_0_INST)) {
        case DL_UART_MAIN_IIDX_RX:
            data = DL_UART_Main_receiveData(UART_0_INST);
            DL_UART_Main_transmitData(UART_0_INST, data);
            break;
        default:
            break;
    }
}
```

Figure 2-10. Changes to Application Code

Two changes are made to the application code. First, DL_SYSCTL_disablesSleepOnExit() is used so that the MSPM0 wakes briefly on each UART RX. Next, a simple check of the UART RX data is added. If an 'S' or 's' is received, then the red LED is turned on. Anything else means this is turned off.

5. *Debug and verify*

   The following figures are captures from an logic analyzer showing the UART communication at 9600 baud and the red LED being turned on and off correctly. The code is echoing every UART character but only turning on the LED when the correct character is received.



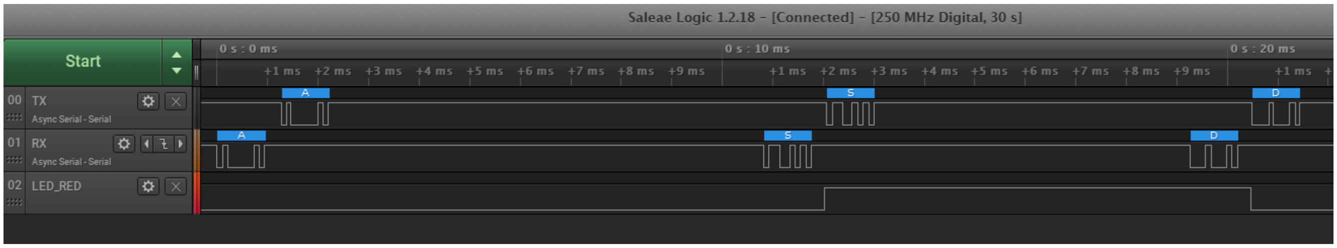


Figure 2-11.

The software has successfully been ported. If this was just the first peripheral of many, continue to repeat this process and use SysConfig to combine each block.

3 Core Architecture Comparison

3.1 CPU

The STM32G0 and MSPM0 family of parts are both based on the Arm Cortex M0+ CPU core architecture and instruction set. Table 3-1 gives a high-level overview of the general features of the CPUs in the MSPM0G and MSPM0L families compared to the STM32G0. Interrupts and Exceptions provides a comparison of the interrupts and exceptions and how these are mapped in the Nested Vectored Interrupt Controller (NVIC) peripheral included in the M0 architecture for each device.

Table 3-1. Comparison of CPU Feature Sets

Feature	STM32G0	MSPM0G	MSPM0L	MSPM0C	MSPM0H
Architecture	Arm Cortex-M0+	Arm Cortex-M0+	Arm Cortex-M0+	Arm Cortex-M0+	Arm Cortex-M0+
Maximum MCLK	64MHz	80MHz	32MHz	24 or 32MHz	32MHz
Processor trace capabilities	No	Yes, integrated micro trace buffer	No	No	No
Memory protection unit (MPU)	Yes	Yes	Yes	Yes	Yes
System timer (SYSTICK)	Yes	Yes - 24 bit	Yes - 24 bit	Yes - 24 bit	Yes
Hardware Accelerator	CORDIC/FMAC	MATHACL	No	No	No
Hardware breakpoint / watchpoints	4 / 2	4 / 2	4 / 2	2 / 1	4 / 2
Boot routine storage	Flash (system memory)	ROM	ROM	N/A	ROM
Bootstrap loader storage	Flash (system memory)	ROM	ROM	N/A	ROM
Bootloader interface support(1) (2)	UART, I2C, SPI, USB, FDCAN	UART, I2C, user extendable	UART, I2C, user extendable	UART, I2C, user extendable	UART, I2C, user extendable
DMA	Yes	Yes	Yes	Yes	Yes

(1) Refer to the device-specific data sheet for availability.
(2) Other interfaces to be made available in later device releases.

3.2 Embedded Memory Comparison

3.2.1 Flash Features

The MSPM0 and STM32G0 family of MCUs feature nonvolatile flash memory used for storing executable program code and application data.

Table 3-2. Comparison of Flash Feature

Features	STM32G0	MSPM0
Flash memory	STM32G0B1xx, G0C1xx (up to 512KB) STM32G071xx, G081xx (up to 128KB) STM32G031xx, G041xx, G051xx, G061xx (up to 64KB)	MSPM0Gxx ranges 512KB to 32KB MSPM0Lxx ranges 256KB to 8KB MSPM0Cxx ranges 64KB to 8KB MSPM0Hxx ranges 64KB to 8KB
Memory organization	1 bank – devices up to 128KB 2 banks – devices with >128KB	1 bank – devices up to 256KB 2 banks – devices with >256KB
Flash wait states	0 (HCLK ≤ 24 MHz) 1 (HCLK ≤ 48 MHz) 2 (HCLK ≤ 64 MHz)	0 (MCLK, CPUCLK ≤ 24 MHz) 1 (MCLK, CPUCLK ≤ 48 MHz) 2 (MCLK, CPUCLK ≤ 80 MHz)
Flash word size	64 bits plus 8 ECC bits	64 bits plus 8 ECC bits
Programming resolution	Single word size	Single word, 32-, 16-, or 8-bit (byte)
Multi-word programming	32 words (256 bytes)	2, 4, or 8 words (up to 64 bytes)

Table 3-2. Comparison of Flash Feature (continued)

Features	STM32G0	MSPM0
Erase	Page size = 2KB Bank erase (single bank) Mass erase (all banks)	Sector size = 1KB Bank erase (up to 256KB)
Write protection	Yes (2 write protection areas per bank)	Yes, static and dynamic
Read protection	Yes	Yes
Flash memory read operations	64-bit flash word size plus 8 ECC bits	Same – if optional ECC is present
Flash memory write operations	64-bit flash word size plus 8 ECC bits	Same – if optional ECC is present
Error code correction (ECC)	8 bits for 64 bits	Same – if optional ECC is present
Securable memory area	Yes, main memory	No
Info memory	Yes	Yes (NONMAIN)
OTP data region	1KB	No
Prefetch	Yes	Yes
CPU instruction cache	Two 64-bit cache lines (16 bytes) 4x 32-bit instructions or 8x 16-bit instructions	MSPM0Gx: Four 64-bit cache lines (32 bytes) 8x 32-bit instructions or 16x 16-bit instructions
		MSPM0Lx/Cx/Hx: Two 64-bit cache lines 4x 32-bit instructions or 8x 16-bit instructions

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

- In-circuit program and erase supported across the entire supply voltage range
- Internal programming voltage generation
- Support for EEPROM emulation with up to 100 000 program/erase cycles on the lower 32KB of the flash memory, with up to 10 000 program/erase cycles on the remaining flash memory (devices with 32KB support 100 000 cycles on the entire flash memory)

3.2.2 Flash Organization

The flash memory is used for storing application code and data, the device boot configuration, and parameters that are preprogrammed by TI from the factory. 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.

Memory Banks

Most MSPM0 devices implement a single flash bank (BANK0). On devices with a single flash bank, an ongoing program/erase operation stalls all read requests to the flash memory until the operation has completed and the flash controller has released control of the bank. On devices with more than one flash bank, a program/erase operation on a bank also stalls read requests issued to the bank that is executing the program/erase operation but does not stall read requests issued to another bank. Therefore, the presence of multiple banks enables application cases such as:

- Dual-image firmware updates (an application can execute code out of one flash bank while a second image is programmed to a second symmetrical flash bank without stalling the application execution)
- EEPROM emulation (an application can execute code out of one flash bank while a second flash bank is used for writing data without stalling the application execution)

Flash Memory Regions

The memory within each bank is mapped to one or more logical regions based upon the functions that the memory in each bank supports. There are four regions:

- FACTORY – Device Id and other parameters
- NONMAIN – Device boot configuration (BCR and BSL)
- MAIN – Application code and data
- DATA – Data or EEPROM emulation

Devices with one bank implement the FACTORY, NONMAIN, and MAIN regions on BANK0 (the only bank present), and the data region is not available. 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.

NONMAIN Memory

The NONMAIN is a dedicated region of flash memory that stores the configuration data used by the BCR and BSL to boot the device. The region is not used for any other purpose. The BCR and BSL both have configuration policies that can be left at the default values (as is typical during development and evaluation) or modified for specific purposes (as is typical during production programming) by altering the values programmed into the NONMAIN flash region.

3.2.3 Embedded SRAM

The MSPM0 and STM32G0 family of MCUs feature SRAM used for storing application data.

Table 3-3. Comparison of SRAM Features

Feature	STM32G0	MSPM0
SRAM memory	STM32G0B1xx, G0C1xx: 144KB (128KB with SRAM parity enabled) STM32G071xx, G081xx: 36KB (32KB with SRAM parity enabled) STM32G051xx, G061xx: 18KB (16KB with SRAM parity enabled) STM32G031xx, G041xx: 8KB (8KB with SRAM parity enabled) Zero wait states	MSPM0Gxx: 128KB to 16KB MSPM0Lxx: 32KB to 2KB MSPM0Cxx: 16KB to 1KB MSPM0Hxx: 8KB Select devices include SRAM parity and ECC. See device data sheet for details
Zero wait states at maximum CPU clock frequency	Yes	Yes
Access resolution	Byte, half-word (16-bits) or full word (32-bits)	Byte, half-word (16-bits) or full word (32-bits)
Parity check	Yes	Yes

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 volatile information such as the call stack, heap, and global data, in addition to 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 32KB of SRAM with 1KB resolution. On devices with less than 32KB of SRAM, write protection is provided for the entire SRAM. Write protection is useful when placing executable code into SRAM as this 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

Similar to STM32G0 devices, MSPM0 devices have a 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 3-4 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 3-4. Comparison of Power-up

STM32G0 Devices		MSPM0 Devices	
Modules governing power up and resets	PWR (power) and RCC (Reset and Clock Control) modules	Module governing power up and resets	PMCU (Power Management and Clock Unit)
Voltage-Level Based Resets			
POR (Power-On Reset)	Complete device reset. First level voltage release for power up. Lowest voltage level for power down.	POR (Power-On Reset)	Complete device reset. First level voltage release for power up. Lowest voltage level for power down.
BOR (Brownout Reset) with configurable levels	Sometimes programmable. Set voltage level that releases reset state on power up, or resets device on power down.	Configurable BOR (Brownout Reset)	Can be configured as a reset or interrupt, with different voltage thresholds, combining the functionality of the STM32G0 BOR and PVD.
PVD (Programmable Voltage Detector)	Configurable voltage monitor that can provide interrupts.		

STM32G0 defines different reset domains, while MSPM0 devices have different levels of reset states. For MSPM0 devices, the reset levels have a set order, and when a level is triggered, all subsequent levels are reset until the device is released into RUN mode. Table 3-5 gives a brief description and comparison between STM32G0 reset domains and MSPM0 reset states. Figure 3-1 shows the relationship between all of the MSPM0 reset states.

Table 3-5. Comparison of Reset Domains

STM32G0 Reset Domains		MSPM0 Reset States(1)	
Power reset domain	Typical triggers are POR, BOR, and exits from standby or shutdown modes. All registers reset except those outside VCORE domain.	POR	Typical triggers: POR voltage levels, SW trigger, NRST held low for >1s. Resets shutdown memory, re-enables NRST and SWD, triggers BOR
		BOR	Typical triggers: POR or BOR voltage level, exit from shutdown mode. Resets PMU, VCORE, and associated logic. Triggers BOOTRST.
No exact equivalent. Boot configuration is read on the fourth clock cycle of SYSCLK after a reset.		Boot reset (BOOTRST)	Typical triggers: BOR or software trigger, fatal clock failure, NRST held low for <1 s. Executes boot configuration routine. Resets majority of core logic and registers, including RTC, clock, and IO configurations.(2) SRAM power cycled and lost. Triggers SYSRST.
System reset domain	System reset sets all registers to the reset values except the reset flags in the clock control and status register (RCC_CSR) and the registers in the RTC domain.	System reset (SYSRST)	Typical triggers: BOOTRST, BSL entry or exit, watchdog timer, software trigger, debug subsystem. Resets CPU state and all peripherals except RTC, LFCLK, LFXT, and SYSOSC frequency correction loop. Device enters RUN mode on exit.
No equivalent		CPU-only reset (CPURST)	Software and debug subsystem triggers only. Resets CPU logic only. Peripheral state are not affected.
RTC domain	Triggered by software or VDD or VBAT power on, if both supplies where previously been powered off. Resets only the LSE oscillator, RTC, backup registers and RCC RTC domain control register.	RTC and associated clocks are reset through BOOTRST, BOR, or POR.(2)	

(1) Not all reset triggers described. Refer to the PMCU chapter of the device TRM for all available reset triggers.
(2) If BOOTRST cause was through NRST or software trigger, RTC, LFCLK, and LFXT or LFLCK_IN configurations and IOMUX settings are NOT reset to allow RTC to maintain operation through external reset.

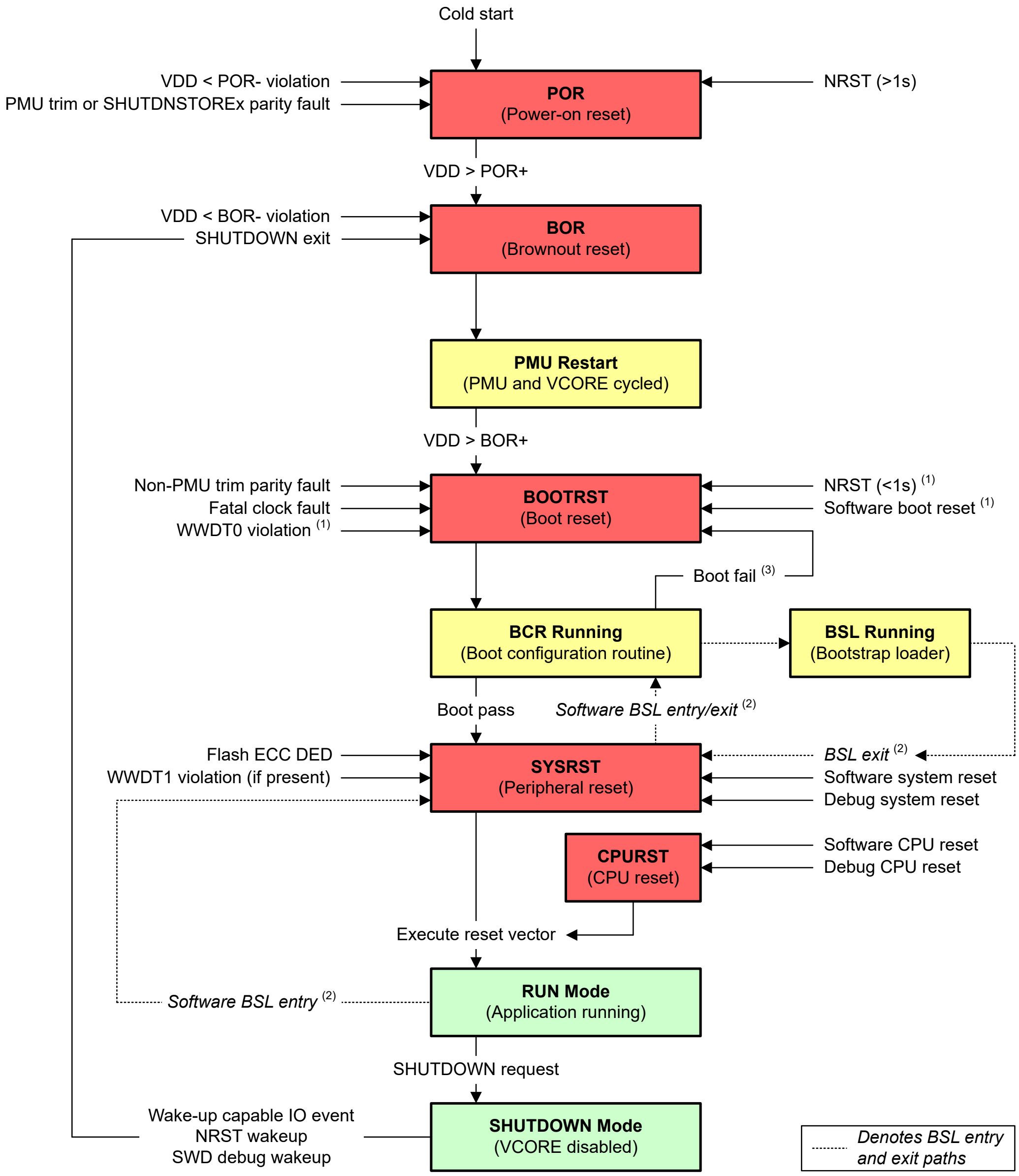


Figure 3-1. MSPM0 Reset Levels

3.4 Clocks Summary and Comparison

STM32G and MSPM0 contain internal oscillators which source primary clocks. The clocks can be divided to source other clocks and be distributed across the multitude of peripherals.

Table 3-6. Oscillator Comparisons

STM32G0 Oscillators	MSPM0 Oscillators
HSI16RC 16MHz	SYSOSC[(1)]
HSI48RC 48MHz	SYSOSC
LSI RC 32kHz	LFOSC
HSE OSC 4-48 MHz	HFXT
LSE OSC 32kHz	LFXT
I2S_CLKIN	HFCLK_IN (Digital Clock)

(1) SYSOSC is programmable to be 32MHz, 24MHz, 16MHz, or 4MHz.

Table 3-7. Clock Comparison

STM32G Clock	MSPM0 Clock
HSISYS	N/A
PLLPCLK	SYSPLLCLK1
PLLQCLK	SYSPLLCLK1
PLLRCLK	SYSPLLCLK0
N/A	SYSPLLCLK2x[(1)]
SYSCLK	BUSCLK[(2)]
HCLK	MCLK
HCLK8	CPUCLK
PCLK	BUSCLK
TIMPCLK	BUSCLK
LPTIMx_IN	LFCLK_IN

(1) SYSPLLCLK2x is twice the speed of the output of the PLL module and can be divided down.
(2) BUSCLK depends on the Power Domain. For Power Domain 0, BUSCLK is ULPCLK. For Power Domain 1, BUSCLK is MCLK.

Table 3-8. Peripheral Clock Sources

Peripheral	STM32G Clock Source	MSPM0 Clock Source
RTC	LSI, LSE, HSE/32	LFCLK (LFOSC, LFXT)
UART	PCLK, LSE, HSI16, SYSCLK	BUSCLK, MFCLK, LFCLK
SPI	NEED TO FIND	BUSCLK, MFCLK, LFCLK
I2C	PCLK, HSI16, SYSCLK	BUSCLK, MFCLK
ADC	HSI16, SYSCLK, PLLPCLK	ULPCLK, HFCLK, SYSOSC
CAN	PCLK, HSE, PLLQCLK	PLLCLK1, HFCLK
TIMERS	PCLK, TIMPCLK, PLLQCLK	BUSCLK, MFCLK, LFCLK
LPTIM 1/2 (TIMG0/1)	PCLK, LSI, LSE, HSI16, LPTIMX_IN	LFCLK, ULPCLK, LFCLK_IN
RNG	HSI48, PLLQCLK, HSI16/8, SYSCLK	MCLK

The TRM for each device family has a clock tree to help visualize the clock system. Sysconfig can assist with the options for clock division and sourcing for peripherals.

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.

Operating Modes Comparison

STM32G0 devices have similar operating modes. The table below gives a brief comparison between STM32G0 and MSPM0 devices.

Table 3-9. Operating Modes Comparison Between STM32G0 and MSPM0 Devices

STM32G0			MSPM0		
Mode		**Description**	**Mode**	**Description**	
Run		Full clocking and peripherals available	Run	0	Full clocking and peripherals available
LP RUN		CPU limited to 2 MHz		1	SYSOSC at set frequency; CPUCLK and MCLK limit to 32 kHz
				2	SYSOSC disabled; CPUCLK and MCLK limit to 32 kHz
Sleep		CPU not clocked	Sleep	0	CPU not clocked
LP Sleep		Same as LP RUN; but CPU not clocked		1	Same as Run1, but CPU not clocked
				2	Same as Run2, but CPU not clocked
Stop	0	VCORE domain clocks disabled	Stop	0	Sleep 0 + PD1 disabled
				1	Sleep 1 + SYSOSC gear shifted to 4 MHz
	1	Stop 0 + main power regulator off		2	Sleep 2 + ULPCLK limited to 32 kHz
Standby		Lowest power with BOR capability; RTC available; register settings lost.	Standby	0	Lowest power with BOR capability; all PD0 peripherals can receive ULPCLK and LFCLK at 32 kHz; RTC available with RTCCLK
				1	Only TIMG0 and TIMG1 can receive ULPCLK or LFCLK at 32 kHz; RTC available with RTCCLK
Shutdown		No clocks or BOR. Core regulation off. RTC domain can still be active. Exit triggers Reset.	Shutdown	No clocks, BOR, or RTC. Core regulation off. PD1 And PD0 disabled. Exit triggers reset level BOR.	

MSPM0 Capabilities in Lower Power Modes

As seen in Table 3-9, 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

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, refer to the appropriate chapter in the MSPM0 TRMs.

MSPM0 G-Series 80MHz Microcontrollers Technical Reference Manual

MSPM0 L-Series 32MHz Microcontrollers Technical Reference Manual

Entering Lower-Power Modes

Like STM32G0 devices, 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.

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 Interrupt and Events Comparison

Interrupts and Exceptions

The MSPM0 and STM32G0 both register and map interrupt and exception vectors depending on the device available peripherals. A summary and comparison of the interrupt vectors for each family of devices is included in Table 3-10. A lower value of priority for an interrupt or exception is given higher precedence over interrupts with a higher priority value. For some of these vectors the priority is user-selectable, and for others, this is fixed.

In the MSPM0 and STM32G0, exceptions such as NMI, reset, and hard fault handlers are given negative priority values to indicate that these always have the highest precedence over peripheral interrupts. For peripherals with selectable interrupt priorities, up to four programmable priority levels are available on both families of devices.

Table 3-10. Interrupt Comparison

NVIC Number	STM32G0		MSPM0	
	Interrupt/Exception	Priority	Interrupt/Exception	Priority
-	Reset	Fixed: -3	Reset	Fixed: -3
-	NMI Handler	Fixed: -2	NMI Handler	Fixed: -2
-	Hard Fault Handler	Fixed: -1	Hard Fault Handler	Fixed: -1

Table 3-10. Interrupt Comparison (continued)

NVIC Number	STM32G0		MSPM0	
	Interrupt/Exception	Priority	Interrupt/Exception	Priority
-	SVCall Handler	Selectable	SVCall Handler	Selectable
-	PendSV	Selectable	PendSV	Selectable
-	SysTick	Selectable	SysTick	Selectable
0	Window Watchdog Interrupt	Selectable	INT_GROUP0: WWDT0/1, DEBUGSS, FLASHCTL, WUC FSUBx, and SYSCTL	Selectable
1	Power Voltage Detector Interrupt	Selectable	INT_GROUP1: GPIOA/B/C, COMP0/1/2, TRNG(1)	Selectable
2	RTC and Timestamp	Selectable	TIMG8(1)	Selectable
3	Flash Global Interrupt	Selectable	UART3(1)	Selectable
4	RCC Global Interrupt	Selectable	ADC0	Selectable
5	EXTI0 and EXTI1 interrupt	Selectable	ADC1(1)	Selectable
6	EXTI2 and EXTI3 interrupt	Selectable	CANFD0(1)	Selectable
7	EXTI4-EXTI15 interrupt	Selectable	DAC0(1)	Selectable
8	UCPD1/UCPD2/USB	Selectable	TIMG9(1)	Selectable
9	DMA1 Channel 1	Selectable	SPI0	Selectable
10	DMA1 Channel 2 and 3	Selectable	SPI1(1)	Selectable
11	DMA1 Channel 4-6, and DMA2 Channel 1-5	Selectable	SPI2(1)	Selectable
12	ADC and Comparator	Selectable	CANFD1(1)	Selectable
13	Timer 1 (TIM1), Break, Update, Trigger, and Commutation	Selectable	UART1(1)	Selectable
14	TIM1 Capture Compare	Selectable	UART4(1)	Selectable
15	TIM2 global interrupts	Selectable	UART0	Selectable
16	TIM3 and TIM4 global interrupts	Selectable	TIMG0(1)	Selectable
17	TIM6, LPTIM1, and DAC interrupts	Selectable	TIMG6(1)	Selectable
18	TIM6 and LPTIM2 global interrupts	Selectable	TIMA0(1)	Selectable
19	TIM14 global interrupts	Selectable	TIMA1	Selectable
20	TIM15 global interrupts	Selectable	TIMG7(1)	Selectable
21	TIM16 and FDCAN0 global interrupts	Selectable	TIMG12(1)	Selectable
22	TIM17 and FDCAN1 global interrupts	Selectable	TIMG14(1)	Selectable
23	12C1 global interrupts	Selectable	UART5(1)	Selectable
24	I2C2 and I2C3 global interrupts	Selectable	I2C0	Selectable
25	SPI1 global interrupts	Selectable	I2C1	Selectable
26	SPI2 and SPI3 global interrupts	Selectable	I2C2(1)	Selectable
27	USART1 global interrupts	Selectable	UART7(1)	Selectable
28	USART2 and LPUART2 global interrupts	Selectable	AES(1)	Selectable
29	USART 3-6 and LPUART1 global interrupts	Selectable	UART6(1)	Selectable
30	CEC global interrupts	Selectable	RTC(1)	Selectable
31	AES and RNG global interrupts	Selectable	DMA	Selectable

(1) Listed interrupts are specific to the MSPM0G3519. Check the device-specific data sheet for exact specifications.

Event Handler and EXTI (Extended Interrupt and Event Controller)

The MSPM0 devices include a dedicated event manager peripheral, which extends the concept of the NVIC to allow digital events from a peripheral to be transferred to the CPU as interrupts, to the DMA as a trigger, or to another peripheral to trigger a hardware action. 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.

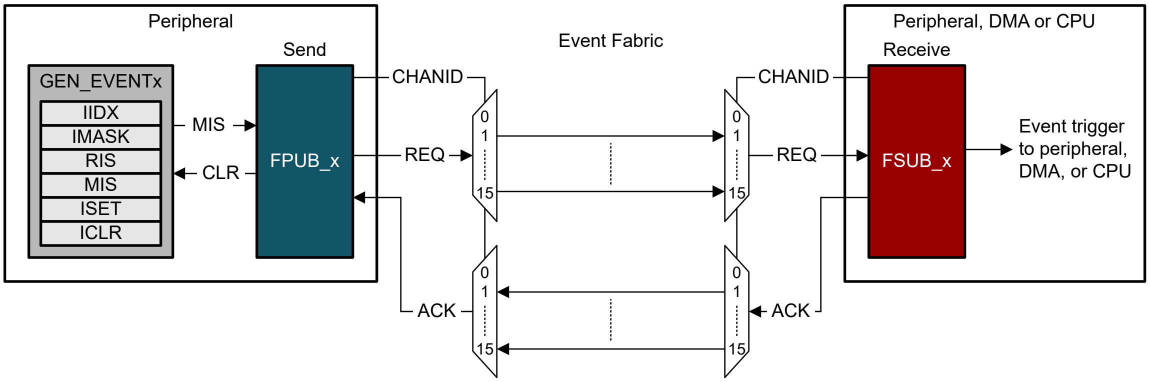


Figure 3-2. Generic Event Route

In the MSPM0 event manager, the peripheral that generates the event is known as a publisher, and the peripheral, DMA, or CPU that acts based on the publisher is known as the subscriber. The potential combinations of available publisher and subscriber are extremely flexible and can be used when migrating software to replace functionality previously handled by interrupt vectors and the CPU, to bypass the CPU entirely. For example, an I^2C-to-UART bridge can previously have triggered a UART transmission upon receipt of an I^2C STOP, using an ISR to set a flag, or load the UART TX buffer directly. With the MSPM0 Event handler, the I^2C transaction complete event can trigger the DMA to load the UART TX buffer directly, and therefore eliminate the need for any action by the CPU.

See the Events section of the MSPM0G technical reference manual or the MSPM0L technical reference manual to get more details on the use of the event handler in MSPM0.

Not to be confused with the MSPM0 event handler, the STM32G0 family of devices implement an extended interrupt and event controller (EXTI), which allows for the system wake from STOP mode through configurable events from IOs or peripherals. The wakeup features of the STM32G0 EXTI can be best replicated in MSPM0 using the IO wakeup features (see the IOMUX section of the MSPM0 technical reference manuals) and GPIO FastWake (see the GPIO section of the MSPM0 technical reference manuals). If the wakeup is for a single action, the Event handler peripheral is capable of requesting necessary PMCU resources for a peripheral operation to occur, and returning to the applicable low power mode after.

3.7 Debug and Programming Comparison

The Arm SWD 2-wire JTAG port is the main debug and programming interface for both MSPM0 and STM32G0 devices. This interface is typically used during application development, and during production programming. Table 3-11 compares the features between the two device families. For additional information about security features of the MSPM0 debug interface, see the *Cybersecurity Enablers in MSPM0 MCUs* application note.

Table 3-11. Arm SWD JTAG Feature Comparison

	STM32G0	MSPM0
Debug port	Arm SWD port (2-wire)	Arm SWD port (2-wire)
Break Point Unit (BPU)	4 hardware breakpoints	4 hardware breakpoints
Data Watch Unit (DWT)	2 watchpoints	2 watchpoints
Micro-Trace Buffer (MTB)	No	MTB support with 4 trace packets(1)
Low-power debug support	Yes	Yes
EnergyTrace support	No	EnergyTrace+ support (CPU states with power profiling)
Peripheral run support during debug	Yes	Yes
Debug interface locking	Can temporarily block debug read access	Can permanently disable debug capabilities, or can lock with password

(1) MSPM0Gxxxx devices only

Bootstrap Loader (BSL) Programming Options

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 are the main purpose of this interface, the updates can also be utilized for initial production programming as well. Table 3-12 shows a comparison of the different options and features between MSPM0 and STM32G0 device families.

Table 3-12. BSL Feature Comparison

BSL Features	STM32G0	MSPM0
BSL started on blank device	Yes	Yes
Auto detection of programming interface	Yes	Yes
Security	Memory security and access restriction options	Secure boot options; CRC protections
Customizable	No	Yes, configurable invoke pin and plug-in feature
Invoke methods	Pattern(1) involving up to 2 pins and device register settings at RESET, SW entry	1 pin high at BOOTRST, SW entry
Interfaces Supported		
UART	Yes	Yes
I2C	Yes	Yes
SPI	Yes(2)	Custom plug-in needed
CAN	Yes(2)	Plug-in planned(2)
USB	Yes(2)	No MSPM0 device with USB capability at this time.

(1) Pattern option availability is device dependent.
(2) Only on select devices

4 Digital Peripheral Comparison

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

MSPM0 GPIO functionality covers virtually all the features provided by STM32G0 GPIO. STM32G0 uses the term GPIO to refer to all the functionality responsible for managing the device pins. However, MSPM0 uses a slightly different nomenclature, namely:

- MSPM0 GPIO refers to the hardware capable of reading and writing IO, generating interrupts, etc.
- 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 STM32G0 GPIO. Additionally, MSPM0 offers functionality not available in STM32G0 devices such as DMA connectivity, controllable input filtering and event capabilities.

Table 4-1. GPIO Feature Comparison

Feature	STM32G0	MSPM0
Output modes	Push-pull Open drain with pullup or pulldown	Equivalent
GPIO speed selection	Speed selection for each I/O	Similar MSPM0 offers Standard IO (SDIO) on all IO pins. SDIO is comparable or better than STM GPIO speed=01. MSPM0 High-Speed IO (HSIO) is available on select pins. HSIO is comparable to STM GPIO speed=10.
High-drive GPIO	Approximately 20mA	Equivalent, called High Drive IO (HDIO)
Input modes	Floating Pullup or pulldown Analog	Equivalent
Atomic bit set and reset	Yes	Equivalent
GPIO locking	Register locking mechanism	*No MSPM0 equivalent*
Alternate functions	Selection register	Equivalent MSPM0 uses IOMUX
Fast toggle	Can change every two clocks	Equivalent, MSPM0 can toggle pins every clock cycle
Wake-up	GPIO pin state change	Equivalent
GPIO controlled by DMA	No	*Only available on MSPM0*
User controlled input filtering to reject glitches less than 1, 3, or 8 ULPCLK periods	No	*Only available on MSPM0*
User controllable input hysteresis	No	*Only available on MSPM0*

GPIO Code Examples

Information about GPIO code examples can be found in the MSPM0 SDK examples guide.

4.2 Universal Asynchronous Receiver-Transmitter (UART)

STM32G0 and MSPM0 both offer peripherals to perform asynchronous (clockless) communication. These UART peripherals come in two variants, one with standard features and one with advanced features. The naming differences are shown in Table 4-2.

Table 4-2. UART Naming Differences Between STM32G0 and MSPM0

	STM32G0 Naming	MSPM0 Naming
Standard features	Basic	Main
Advanced features	Full	Extend

Table 4-3. UART Advanced Feature Set Comparison

Feature	STM32G0 USART Full feature Set	MSPM0 UART Extend Feature Set
Hardware flow control	Yes	Yes
Continuous communication using DMA	Yes	Yes
Multiprocessor	Yes	Yes
Synchronous mode	Yes	No
Smart card mode (ISO7816)	Yes	Yes
Single-wire half duplex communication	Yes	Yes(1)
IrDA HW support	Yes	Yes
LIN HW support	Yes	Yes
DALI HW support	No	Yes
Manchester Code HW support	No	Yes
Wakeup from low-power mode	Yes	Yes
Auto baud rate detection	Yes	No
Driver enable	Yes	Yes
Data length	7, 8, 9	5, 6, 7, 8
Tx/Rx FIFO Depth	8	4

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

Table 4-4. UART Standard Feature Set Comparison

Feature	STM32G0 USART Basic Feature Set	MSPM0 UART Main Feature Set
Hardware flow control	Yes	Yes
Continuous communication using DMA	Yes	Yes
Multiprocessor	Yes	Yes
Synchronous mode	Yes	No
Single-wire half duplex communication	Yes	Yes(1)
Wakeup from low-power mode	No	Yes
Driver enable	Yes	Yes
Data length	7, 8, 9	5, 6, 7, 8
Tx/Rx FIFO Depth	None	4

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

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 STM32G0 both support serial peripheral interface (SPI). Overall, MSPM0 and STM32G0 SPI support is comparable with the difference listed in Table 4-5.

Table 4-5. SPI Feature Comparison

Feature	STM32G0x	MSPM0
Controller or peripheral operation	Yes	Yes
Data bit width (controller mode)	4 to 16 bit	4 to 16 bit
Data bit width (peripheral mode)	4 to 16 bit	7 to 16 bit
Maximum speed	32MHz	MSPM0G: 32MHz MSPM0L: 16MHz MSPM0C: 16MHz MSPM0H: 16MHz
Full-duplex transfers	Yes	Yes
Half-duplex transfer (bidirectional data line)	Yes	No
Simplex transfers (unidirectional data line)	Yes	Yes
Multiple controller capability	Yes	No
Hardware chip select management	Yes (1 peripheral)	Yes (4 peripherals)
Programmable clock polarity and phase	Yes	Yes
Programmable data order with MSB-first or LSB-first shifting	Yes	Yes
SPI format support	Motorola, TI	Motorola, TI, MICROWIRE
Hardware CRC	Yes	No, MSPM0 offers SPI parity mode
TX FIFO depth	Depends on data size	4
RX FIFO depth	Depends on data size	4

SPI Code Examples

Information about SPI code examples can be found in the MSPM0 SDK examples guide

4.4 I²C

MSPM0 and STM32G0 both support I²C. Overall MSPM0 and STM32G0 I²C support is comparable with notable difference outlined in the following table.

Table 4-6. I²C Feature Comparison

Feature	STM32G0	MSPM0
Controller and target modes	Yes	Yes
Multi-controller capability	Yes	Yes
Standard-mode (up to 100kHz)	Yes	Yes
Fast-mode (up to 400kHz)	Yes	Yes
Fast-mode Plus (up to 1MHz)	Yes	Yes
Addressing mode	7, 10 bit	7,10 bit
Peripheral addresses	2 addresses and 1 configurable mask	2 addresses
General call	Yes	Yes
Programmable setup and hold times	Yes	No
Event management	Yes	Yes
Clock stretching	Yes	Yes
Software reset	Yes	Yes
FIFO/Buffer	1 byte	TX: 8 byte RX: 8 byte
DMA	Yes	Yes
Programmable analog and digital noise filters	Yes	Yes

I²C Code Examples

Information about I2C code examples can be found in the MSPM0 SDK examples guide.

4.5 Timers (TIMGx, TIMAx)

STM32G0 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 4-7. Timer Naming

STM32G0		MSPM0	
Timer Name	**Abbreviated Name**	**Timer Name**	**Abbreviated Name**
Advanced control	TIM1	Advanced control	TIMA0-1
General-purpose	TIM2-4, TIM14/-17	General purpose	TIMG0-11,14
		High resolution	TIMG12-13
Basic	TIM6/7		
Low power	LPTIM		

Table 4-8. Timer Feature Comparison

Feature	STM32G0 Timers	MSPM0G Timers	MSPM0L Timers	MSPM0C Timers	MSPM0H Timers
Resolution	16 bit, 32 bit	16 bit, 32 bit	16 bit, 32 bit	16 bit	16 bit
PWM	Yes	Yes	Yes	Yes	Yes
Capture	Yes	Yes	Yes	Yes	Yes
Compare	Yes	Yes	Yes	Yes	Yes
One-shot	Yes	Yes	Yes	Yes	Yes
Up down count functionality	Yes	Yes	Yes	Yes	Yes
Power Modes	Yes	Yes	Yes	Yes	Yes
QEI support	Yes	Yes	No	Yes	Yes
Programmable pre-scalar	Yes	Yes	Yes	Yes	Yes
Shadow register mode	Yes	Yes	Yes	Yes	Yes
Events or interrupt	Yes	Yes	Yes	Yes	Yes
Fault Event Mechanism	Yes	Yes	Yes	Yes	Yes
Auto reload functionality	Yes	Yes	Yes	Yes	Yes

Table 4-9. Timer Module Replacement

STM32G0 Timer	MSPM0 Equivalent	Reasoning
TIM1	TIMA, TIMG8-12	Advanced control, Both 16-bit resolution, QEI support
TIM2	TIMG12-13	32-bit resolution
TIM3, TIM4	TIMG0-7,14	General purpose, 16-bit resolution
TIM6, TIM7	Any	Basic timer
TIM14	Any	Same functionality as TIM3/4
TIM15, TIM16, TIM17	Any	General purpose
LPTIM	Any timer in PD0	LPTIM sources LFCLK, PD0 – low power mode in MSPM0

Table 4-10. Timer Use-Case Comparisons

Feature	STM32G0 Timer	MSPM0 Timer
PWM	TIM1-4 have edge and center aligned options, TIM6-7 do not have PWM functionality. TIM15-17 only edge aligned option.	All timers have edge aligned or center aligned options
Capture	No major differences	No major differences
Compare	No major differences	No major differences
One-shot	No major differences	No major differences
Prescaler	16-bit prescaler, besides LPTIM (3-bit prescaler)	8-bit prescaler
Synchronization	TIM1-4, TIM15	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)

STM32G0 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 4-11. WWDT Naming

Key	STM32G0	MSPM0
Name	Independent watchdog timer, windowed watchdog timer	Windowed watchdog timer
Abbreviated name (same order)	IWDG, WWDG	WWDT

Table 4-12. WDT Feature Comparison

Feature	STM32G0	MSPM0
Window mode	Yes	Yes
Interval timer mode	Yes	Yes
LFCLK source	Yes	Yes
Interrupts	Yes	Yes
Counter resolution	7 bit	25 bit
Clock divider	WWDG no, IWDG yes	Yes

WWDT code examples

Information about WWDT code examples can be found in the MSPM0 SDK examples guide.

4.7 Real-Time Clock (RTC)

STM32G0 and MSPM0 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 4-13. RTC Feature Comparison

Feature	STM32G0	MSPM0
Power modes	Yes	Yes
Binary coded format	Yes	Yes
Leap year correction	Yes	Yes
Number of customizable alarms	2	2
Internal and External crystal	Yes	Yes
Crystal offset calibration	Yes	Yes
Prescaler blocks	Yes	Yes
Interrupts	Yes	Yes

RTC Code Examples

Information about RTC code examples can be found in the MSPM0 SDK examples guide.

5 Analog Peripheral Comparison

5.1 Analog-to-Digital Converter (ADC)

STM32G0 and MSPM0 both offer ADC peripherals to convert analog signals to a digital equivalent. Both device families feature a 12-bit ADC. The following tables compare the different features and modes of the ADCs.

Table 5-1. Feature Set Comparison

Feature	STM32G0	MSPM0
Resolution (Bits)	12	12/10/8
Conversion Rate (Msps)	2.5	MSPM0Gx: 4
		MSPM0Lx: 1.68
		MSPM0Cx: 1.68
		MSPM0Hx: 1.6
Oversampling (Bits)	16	14
Hardware Oversampling	256x	128x
FIFO	No	Yes
ADC Reference (V)	Internal: 2.048, 2.5	Internal: 1.4, 2.5, VDD
	When $V_{DD} < 2$ External: $V_{REF} = V_{DD}$	External: $1.4 \le V_{REF} \le V_{DD}$
	When $V_{DD} \ge 2$ External: $2 \le V_{REF} \le V_{DD}$	
Operating Power Modes	Run, Sleep	Run, Sleep, Stop, Standby[1]
Auto Power Down	Yes	Yes
External Input Channels[2]	Up to 16	MSPM0Gx/Cx/Hx: Up to 27
		MSPM0Lx: Up to 26
Internal Input Channels	Temperature Sensor, VREF, VBAT	Temperature Sensor, Supply Monitoring, Analog Signal Chain
DMA Support	Yes	Yes
ADC Window Comparator Unit	No	Yes
Simultaneous Sampling	No	Yes (MSPM0Gx only)
Number of ADCs[3]	Up to 1	MSPM0Gx: Up to 2
		MSPM0Lx/Cx/Hx: 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

STM32G0	MSPM0	Comments
Single Conversion Mode	Single Channel Single Conversion	ADC samples and converts a single channel once
Scan a Sequence of Channels	Sequence of Channels Conversion	ADC samples a sequence of channels and converts once.
Continuous Conversion Mode	Repeat Single Channel Conversion	Repeat single channel continuously samples and converts one channel
	Repeat Sequence of Channels Conversion	Samples and converts a sequence of channels then repeats the same sequence

Table 5-2. Conversion Modes (continued)

STM32G0	MSPM0	Comments
Discontinuous Mode	Repeat Sequence of Channels Conversion	Samples and converts a discontinuous set of channels. This can be done on MSPM0 by mapping the MEMCTRLx to different channels.

ADC code examples

Information about ADC code examples can be found in the MSPM0 SDK examples guide.

5.2 Comparator (COMP)

The STM32G0 and MSPM0 family of parts both offer integrated comparators as optional peripherals on some devices. In both families of devices these are denoted as COMPx, where the 'x' final character refers to the specific comparator module being considered. In the STM32G0 family these are numbered 1-3, and in the MSPM0 family these are numbered 0-2. The comparator modules can both provide a windowed comparator functionality in devices with more than 1 comparator, 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 STM32G0 comparator modules compare feature-by-feature is included in Table 5-3.

Note

MSPM0H does not currently support comparators, but this can change in future roadmap devices.

Table 5-3. COMP Feature Set Comparison

Feature	SMT32G0	MSPM0G	MSPM0L	MSPM0C
Available comparators	Up to 3	Up to 3	1	1
Output routing	Multiplexed I/O Pins	Multiplexed I/O Pins	Multiplexed I/O Pins	Multiplexed I/O Pins
	EXTI Interrupt	Interrupt/Event Interface	Interrupt/Event Interface	Interrupt/Event Interface
Noninverting input sources	Multiplexed I/O Pins	Multiplexed I/O Pins	Multiplexed I/O Pins	Multiplexed I/O Pins
		DAC12 output[1]	DAC8 output	
		DAC8 output	OPA1 Output[2]	DAC8 output
		Internal V_{REF}: 1.4V and 2.5V		
Inverting input sources	Multiplexed I/O Pins	Multiplexed I/O pins	Multiplexed I/O Pins	Multiplexed I/O Pins
	DAC Channels 1 and 2	Internal temperature sensor	Internal temperature sensor	
	Internal V_{REF}: 2.048 V and 2.5 V	Internal V_{REF}: 1.4 V and 2.5 V	DAC8 output	DAC8 output
	Buffered V_{REF} Divider including: ¼V_{REF}, ½V_{REF}, and ¾V_{REF}	DAC8 output	OPA0[3] output	
		OPA0 output [3]		
Programmable hysteresis	None, 10mV, 20mV, 30mV	None, 10mV, 20mV, 30mV	None, 10mV, 20mV, 30mV	None, 10mV, 20mV, 30mV
		Other values from 0V to V_{REF}/V_{DD} using DAC8	Other values from 0V to V_{DD} using DAC8	Other values from 0V to V_{DD} using DAC8
Register lock	Yes, all COMP registers (disabled on device reset)	Yes, some COMP registers (writes require key)	Yes, some COMP registers (writes require key)	Yes, some COMP registers (writes require key)
Window comparator configuration	Yes	Yes	No (single COMP)	No (single COMP)
Input short mode	No	Yes	Yes	Yes
Operating modes	High speed, medium speed	High speed, low power	High speed, low power	High speed, low power
Fast PWM shutdowns	Yes	Yes (through TIMA fault handler)	Yes (through TIMA fault handler)	Yes (through TIMA fault handler)

Table 5-3. COMP Feature Set Comparison (continued)

Feature	SMT32G0	MSPM0G	MSPM0L	MSPM0C
Output filtering	Blanking filter	Blanking filter	Blanking filter	Adjustable analog filter
		Adjustable analog filter	Adjustable analog filter	
Output polarity control	Yes	Yes	Yes	Yes
Interrupts	Rising edge	Rising edge	Rising edge	Rising edge
	Falling edge	Falling edge	Falling edge	Falling edge
	Both edges	Output ready	Output ready	Output ready
Exchange inputs mode	No	Yes	Yes	Yes

(1) Only on devices with DAC12 peripheral
(2) Only on devices with OPA1 peripheral
(3) Only on devices with OPA0 peripheral

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 STM32G0 and MSPM0 family of parts both offer 12-bit DAC peripherals to perform digital to analog conversion for various applications. In the STM32G0 documentation, this peripheral is referred to just as the DAC. In the MSPM0 Technical Reference Manual, the MSPM0 series data sheets, and the MSPM0 SDK, the 12-bit DAC peripheral is referred to as the DAC12. This differentiates the DAC12 from the 8-bits DACs which 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 STM32G0 and MSPM0G are summarized in Table 5-4.

Table 5-4. DAC Feature Set Comparison

Feature	STM32G0	MSPM0
Resolution	12 bits (11.4 to 11.5 ENOB)	12 bits (11 ENOB)
Output rate	1 MSPS	1 MSPS
Output channels	2 [1]	1 [2]
Data formats	8-bit right aligned, 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	Yes
Output routing	External Pins	External Pins
	Internal peripheral connections: COMP IN-, ADC	Internal peripheral connections: OPA IN+, COMP IN+, ADC0
Internal reference voltage	Yes, 2.5V or 2.048V	Yes, 2.5V or 1.4V
External reference voltage	Yes	Yes
FIFO	No	Yes
Output buffer	Yes	Yes
Configurable output offset	Yes	Yes
Self-calibration mode	Yes	Yes
Automatic waveform generation	Noise wave, triangle wave	No
Sample and hold mode	Yes	No
Trigger sources	External pin, internal timer signals, DAC hold clock, DMA underrun	Internal dedicated sample time generator, DMA interrupts/events, FIFO threshold interrupts/events, 2 hardware triggers (available from event fabric)

(1) Available only on some devices.
(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 STM32G0 family of devices does not offer an integrated Operational Amplifier (OPA) peripheral, but when migrating from the STM32G0 to MSPM0 family, you can make use of the MSPM0 internal OPAs to replace external discrete devices, or to buffer internal signals as necessary. The MSPM0 OPA modules are completely flexible, and can individually, or in combination, replace many discrete amplifiers in sensing or control applications. The primary features of the MSPM0 OPA modules are included in Table 5-5, and examples of common OPA configurations you can recreate are included in OPA code examples

Table 5-5. MSPM0 OPA Feature Set

Feature	MSPM0 Implementation
Input type	Rail to rail (can be enabled or disabled)
Gain bandwidth	1MHz (low-power mode)
	6MHz (standard mode)
Amplifier configurations	General-purpose mode
	Buffer mode
	PGA mode (inverting or noninverting)
	Differential amplifier mode
	Cascade amplifier mode
Input/output routing	External pin routing
	Internal connections to ADC and COMP modules
Fault detection	Burnout current source (BCS)
Chopper stabilization	Standard (selectable chopping frequency)
	ADC assisted chop
	Disabled
Reference voltages	Internal VREF (MSPM0G devices only)
	DAC12 (MSPM0G devices only)
	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 STM32G0x and MSPM0 both have internal references which can be used to supply a reference voltage to internal peripherals and output to external peripherals.

Table 5-6. Feature Set Comparison

Feature	STM32G0	MSPM0G	MSPM0L/C	MSPM0H
Internal Reference (V)	2.048, 2.5	1.4, 2.5	1.4, 2.5	4.05
External Reference (V)	When $V_{DD} < 2$, $V_{REF} = V_{DD}$	External: $1.4 \le V_{REF} \le V_{DD}$	External: $1.4 \le V_{REF} \le V_{DD}$	External: $4.05 \le V_{REF} \le V_{DD}$
	When $V_{DD} \ge 2$, $2 \le V_{REF} \le V_{DD}$			
Output Internal Reference	Yes	Yes	Yes	Yes
Internally Connect to ADC	Yes	Yes	Yes	Yes
Internally Connect to DAC	Yes	Yes	No	No

Table 5-6. Feature Set Comparison (continued)

Feature	STM32G0	MSPM0G	MSPM0L/C	MSPM0H
Internally Connect to COMP	No	Yes	No	No
Internally Connect to OPA	N/A	Yes	No	No

Table 5-7. Control Bit Comparison

STM32G0x VREFBUF Bits	MSPM0 Equivalent
VREFBUF Bit3 (VRR)	CTL1 Bit0 (READY)
VREFBUF Bit2 (VRS)	CTL0 Bit7 (BUFCONFIG)
VREFBUF Bit1 (HIZ)	N/A
VREFBUF Bit0 (ENVR)	CTL0 Bit0 (ENABLE)
	For sample and hold mode: CTL0 Bit8 (SHMODE)

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

VREF code examples

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

6 Summary

The goal of this application note is to help engineers migrate from STM32 devices to MSPM0. The differences between STM32 devices and MSPM0 in terms of ecosystem, CPU, analog and digital peripherals are compared in this document. The application note also describes in detail how to quickly get started with MSPM0 through examples to help users speed up the development progress.

7 References

- Texas Instruments, MSPM0 MCUs Quick Reference Guide
- Texas Instruments, MSPM0 SDK User Guide
- Texas Instruments, MSPM0 Tools Guide
- Texas Instruments, Driverlib API Guide
- Texas Instruments, Code Composer Studio (CCS)
- Texas Instruments, MSPM0 G-Series 80MHz Microcontrollers Technical Reference Manual, technical reference manual
- Texas Instruments, MSPM0 L-Series 32MHz Microcontrollers Technical Reference Manual, technical reference manual
- Texas Instruments, MSPM0 C-Series 24MHz Microcontrollers Technical Reference Manual, technical reference manual
- Texas Instruments, MSPM0 H-Series 32MHz Microcontrollers Technical Reference Manual, technical reference manual
- IAR, IAR website
- Keil, Keil website
- Texas Instruments, CCS quick start guide
- Texas Instruments, CCS training videos
- Texas Instruments, CCS user’s guide
- Texas Instruments, IAR quick start guide
- Texas Instruments, IAR training videos
- IAR, IAR user’s guide
- Keil, Keil quick start guide
- Keil, Keil training videos
- Keil, Keil getting started
- Texas Instruments, SYSCONFIG IDE
- Texas Instruments, MSPM0 SysConfig Guide
- Texas Instruments, XDS110 Debug Probe

- Texas Instruments, J-Link Debug Probes page
- Texas Instruments, LP-MSPM0G3507 LaunchPad development kit
- Texas Instruments, LP-MSPM0G3519 LaunchPad development kit
- Texas Instruments, LP-MSPM0L1117 LaunchPad development kit
- Texas Instruments, LP-MSPM0L1306 LaunchPad development kit
- Texas Instruments, LP-MSPM0L2228 LaunchPad development kit
- Texas Instruments, LP-MSPM0C1104 LaunchPad development kit
- Texas Instruments, LP-MSPM0C1106 LaunchPad development kit
- Texas Instruments, LP-MSPM0H3216 LaunchPad development kit

8 Revision History

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

Changes from Revision A (March 2025) to Revision B (August 2025) **Page**

- Updated entire document with MSPM0C and MSPM0H information.. 1
- Updated MSPM0G and MSPM0L specifications. Added MSPM0C and MSPM0H columns. Added column for STM32C0...3
- Added *Pin to Pin Compatibility Between STM and MSPM0* table to showcase pin-to-pin options.................... 4
- Updated *LP-MSPM0G3507 LaunchPad Development Kit* figure... 6
- Added all current LaunchPads™ and hyperlinks..6
- Updated the *Comparison of Flash Feature* table to include the most recent information on MSPM0G and MSPM0L...17
- Updated the *Comparison of SRAM Features* table to include the most recent information on MSPM0G and MSPM0L...19
- Updated *Interrupt Comparison* table to reflect the interrupts for MSPM0G3519..24
- Added timers to *Timer Naming* table and *Timer Module Replacement* table..31
- Added *References* section.. 38