Application Note

STM32®에서 Arm® 기반 MSPM0으로의 마이그레이션 가이드

추상

본 애플리케이션 노트는 STMicroelectronics STM32® 플랫폼에서 텍사스 인스트루먼트 MSPM0 MCU 에코시스템으로의 마이그레이션을 지원합니다. 본 가이드에서는 MSPM0 개발 및 툴 에코시스템, 코어 아키텍처, 주변 장치 고려 사항 및 SDK(소프트웨어 개발 키트)를 소개합니다. 본 가이드의 의도는 두 제품군 간의 차이점을 강조하고 STM32 에코시스템에 대한 기존 지식을 활용해 MCU의 MSPM0 시리즈로 빠르게 옮겨갈 수 있도록 하는 것입니다.

목차

상표

1 MSPM0 포트폴리오 개요

1.1 머리말

MSP430™ MCU는 TI의 클래식 마이크로컨트롤러로 거의 30년 가까운 역사를 가지고 있습니다. 이제 최신 세대인 MSPM0 제품군을 소개합니다. MSPM0 마이크로컨트롤러(MCU)는 향상된 Arm® Cortex®-M0+ 32비트 코어 플랫폼에 기반을 둔 MSP의 고집적 초저전력 32비트 MCU 제품군에 속합니다. 비용 최적화된 이 MCU 제품은 고성능 아날로그 주변 장치 결합을 제공하며, 확장된 온도 범위를 지원하고, 작은 풋프린트 패키지를 제공합니다. 저전력 MCU로 구성된 TI MSPM0 제품군은 아날로그 및 디지털 통합을 지원하는 장치들로 구성되어 있어 엔지니어가 프로젝트의 요구 사항에 맞는 MCU를 찾을 수 있습니다. MSPM0 MCU 제품군은 Arm Cortex-M0+ 플랫폼과 초저전력 시스템 아키텍처를 결합해 시스템 설계자가 성능을 높이고 에너지 소비를 줄일 수 있도록 도와줍니다.

MSPM0 MCU는 STM32 MCU에 대한 경쟁력 있는 대안을 제공합니다. 이 애플리케이션 노트는 장치 기능과 에코시스템을 비교하여 STM32 MCU에서 MSPM0 MCU로 마이그레이션하는 데 도움을 드립니다.

1.2 STM32 MCU vs. MSPM0 MCU 포트폴리오 비교

표 1-1. TI MSPM0Gx/Lx vs. STM32G0/F0 시리즈 비교

	ST Micro STM32G0 시리즈	ST Micro STM32F0 시리즈	TI MSPM0 MSPM0Gx 시리즈	TI MSPM0 MSPM0Lx 시리즈
코어/주파수	CM0+/64MHz	CM0/48MHz	CM0+/80MHz	CM0+/32MHz
전원 전압	1.7 V~3.6 V	2 V~3.6 V	1.62 V~3.6 V	1.62 V~3.6 V
온도	-40°C~125°C	-40°C ~ 105°C	-40°C~125°C	-40°C~125°C
메모리	512KB~16KB	256KB~16KB	128KB~32KB	64KB~8KB
RAM	최대 144KB	최대 32KB	최대 32KB	최대 4KB
GPIO(최대)	90	88	60	28
아날로그	1x 2.5Msps 12비트 ADC 1x 12비트 DAC 3x 콤퍼레이터	1x 1Msps 12비트 ADC 1x 12비트 DAC 2x 콤퍼레이터	2x 4Msps 12비트 ADC 1x 12비트 DAC 3x 고속 콤퍼레이터 2x 연산 증폭기	1x 1Msps 12비트 ADC 1x 고속 콤퍼레이터 2x 연산 증폭기
통신(최대)	3x SPI 3x I^2C Fast+ 6x UART(LIN) 2x CAN-FD 1x USB	2x SPI 2x I^2C Fast+ 8x UART(LIN) 1x CAN	2x SPI 2x I^2C Fast+ 4x UART(LIN) 1x CAN-FD	1x SPI 2x I^2C Fast+ 2x UART(LIN)
타이머	8	4	7	4
어드밴스 타이머	예(1)	예(1)	예(3x)	아니요
하드웨어 액셀러레이터	해당 없음	해당 없음	옵션	해당 없음
보안	CRC, TRNG, AES256	CRC	CRC, TRNG, AES256	CRC
저전력	활성: 100 µA/MHz 대기(RTC): 1.5 µA	활성: 281 µA/MHz 대기(RTC): 2.5 µA	활성: 85 µA/MHz 대기(RTC): 1.5 µA	활성: 85 µA/MHz 대기: 1.5 µA

2 에코시스템과 마이그레이션

MSPM0 MCU는 신속하게 설계를 시작할 수 있도록 레퍼런스 설계와 코드 예제를 갖춘 광범위한 하드웨어 및 소프트웨어 에코시스템이 지원합니다. MSPM0 MCU에 대해서도 온라인 리소스, MSP 아카데미를 통한 교육, TI E2E™ 지원 포럼을 통한 온라인 지원이 제공됩니다.

2.1 소프트웨어 에코시스템 비교

표 2-1. MSPM0에 해당하는 STM32 소프트웨어 툴

	STM32	MSPM0
IDE	CubeIDE	Code Composer Studio™ IDE(CCS)
소프트웨어 구성	CubeMX	SysConfig
독립형 프로그래밍	CubeProgrammer	UniFlash
디스플레이/데모 GUI 편집기	CubeMonitor	GuiComposer

2.1.1 MSPM0 소프트웨어 개발 키트(MSPM0 SDK)

MSPM0 SDK는 엔지니어가 텍사스 인스트루먼트 MSPM0+ 마이크로컨트롤러 장치를 사용하여 애플리케이션을 신속하게 개발할 수 있도록 돕는 소프트웨어 API, 예제, 문서 및 라이브러리를 제공합니다. 예제는 모든 지원 장치에서 각 기능 영역의 사용 방법을 보여주기 위해 제공되며, 자체 프로젝트에서 시작점으로 사용할 수 있습니다. 또한 MSPM0 SDK에는 대화형 MSP 아카데미 교육 도 포함되어 있어 가이드가 있는 학습 경로를 제공합니다.

예제 폴더는 RTOS와 비 RTOS 하위 폴더로 나뉩니다(지금은 비 RTOS만 지원하고 있습니다). 이 폴더에는 각 LaunchPad ™ 개발 키트에 대한 예제가 포함되어 있으며, 하위 수준 DriverLib 예제, 상위 수준 TI 드라이버 예제 및 GUI Composer, LIN, IQMath 등과 같은 미들웨어 예제가 포함되어 있습니다. 자세한 사항은 *MSPM0 SDK 사용 설명서*를 참조하세요.

2.1.2 CubeIDE vs. CCS(Code Composer Studio IDE)

CCS(Code Composer Studio IDE)는 TI에서 STM32's CubeIDE에 해당됩니다. CCS는 TI의 MCU(마이크로컨트롤러)와 임베디드 프로세서 포트폴리오를 지원하는 Eclipse 기반 IDE입니다. CCS는 최적화 C/C++ 컴파일러, 소스 코드, 편집기, 프로젝트 빌드 환경, 디버거, 프로파일러 및 그 외 다양한 기능을 포함하는 임베디드 애플리케이션을 개발 및 디버깅하는 데 사용되는 툴 패키지로 구성되어 있습니다. CCS는 데스크톱과 클라우드 기반 IDE로 사용할 수 있습니다.

CCS는 MSPM0 장치 구성과 SysConfig의 자동 코드 생성, 일체형 TI 리소스 익스플로러의 MSPM0 코드 예제와 아카데미 교육을 하나로 결합합니다. CCS는 올인원 개발 도구 경험을 제공합니다.

CCS 외에도, MSPM0 장치는 아래 표에 나와 있는 산업 표준 IDE에서도 지원합니다.

표 2-2. MSPM0 지원 IDE

IDE	MSPM0
CCS	✓
IAR	✓
케일	✓

2.1.3 CubeMX vs. SysConfig

SysConfig는 핀, 주변 장치, 무선 장치, 서브시스템 및 기타 구성 요소를 구성하는 데 사용되는 직관적이고 포괄적인 그래픽 유틸리티 모음입니다. 이 제품은 TI에서 STM32 CubeMX에 해당됩니다. SYSConfig를 사용하면 충돌을 시각적으로 관리, 노출 및 해결하여 차별화된 애플리케이션을 만들 수 있는 시간을 더 많이 확보할 수 있습니다. 이 도구의 출력에는 MSPM0 SDK 예제에서 사용하거나 사용자 지정 소프트웨어를 구성하는 데 사용할 수 있는 C 헤더와 코드 파일이 포함됩니다. SysConfig은 CCS에 결합되어 있지만 독립적인 프로그램으로도 사용할 수 있습니다.

자세한 사항은 MSPM0 SysConfig 가이드를 참조하세요.

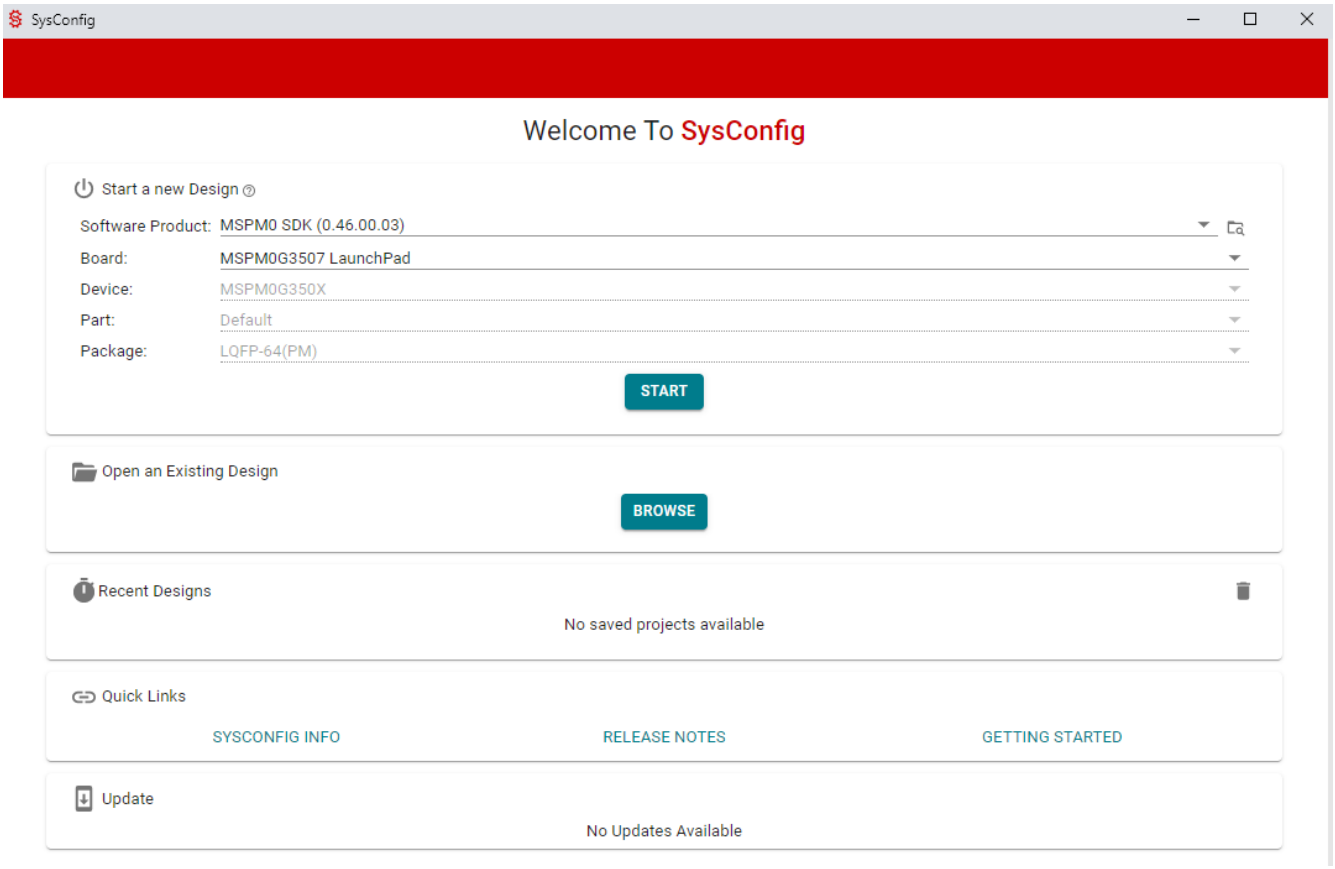

그림 2-1. MSPM0 SysConfig

2.2 하드웨어 에코시스템

LaunchPad 개발 키트는 유일한 MSPM0용 평가 모듈입니다. LaunchPad 키트는 MSPM0에서 개발을 시작하는 데 필요한 모든 것이 포함되어 있어 쉽게 사용할 수 있는 EVM입니다. 여기에는 프로그래밍, 디버깅 및 EnergyTrace™ 기술을 이용한 전력 소비량 측정을 위한 온보드 디버그 프로브가 포함되어 있습니다. 또한 MSPM0 LaunchPad 키트에는 여러 회로 중 특히 온보드 버튼, LED 및 온도 센서가 포함되어 있습니다. 다양한 BoosterPack 플러그인 모듈을 지원하는 40핀 BoosterPack™ 플러그인 모듈 헤더를 사용하면 빠르고 간편하게 프로토타이핑을 수행할 수 있습니다. 무선 연결, 그래픽 디스플레이, 환경 센서 등과 같은 기능을 빠르게 추가할 수 있습니다.

- LP-MSPM0G3507 LaunchPad 개발 키트
- LP-MSPM0L1306 LaunchPad 개발 키트

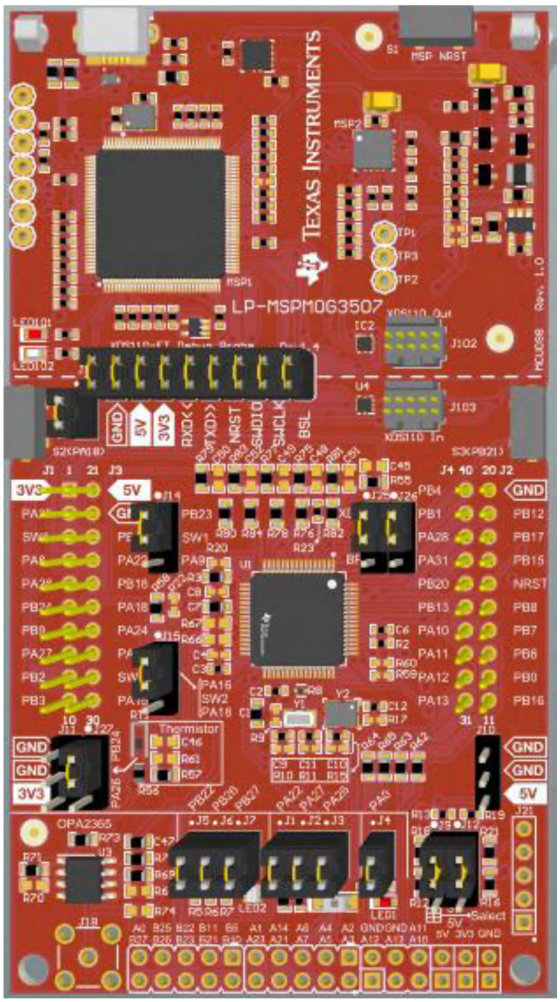

그림 2-2. LP-MSPM0G3507 LaunchPad 개발 키트

2.3 디버그 툴

디버그 서브시스템(DEBUGSS)은 SWD(Serial Wire Debug) 2선 물리적 인터페이스를 디바이스 내의 여러 디버그 기능에 인터페이스합니다. MSPM0 장치는 프로세서 실행, 장치 상태 및 전원 상태(EnergyTrace 기술 사용)의 디버깅을 지원합니다. 그림 2-3는 디버그의 연결을 보여줍니다.

MSPM0는 표준 직렬 와이어 디버그용으로 XDS110과 J-Link 디버거를 지원합니다.

텍사스 인스트루먼트 XDS110은 TI 임베디드 프로세서용으로 설계한 제품입니다. XDS110은 TI 20핀 커넥터를 통해 대상 보드에 연결(TI 14핀 및 Arm 10핀과 Arm 20핀용 복수 어댑터 사용)되고 USB2.0 고속(480Mbps)을 통해 호스트 PC에 연결됩니다. 단일 포드에서 다양한 표준(IEEE1149.1, IEEE1149.7, SWD)을 지원합니다. 모든 XDS 디버그 프로브는 ETB(Embedded Trace Buffer)가 포함되어 있는 모든 Arm 및 DSP 프로세서에서 코어(Core) 및 시스템 트레이스(System Trace)를 지원합니다. 자세한 사항은 XDS110 디버그 프로브를 참조하세요.

J-Link 디버그 프로브는 디버깅 및 플래시 프로그래밍 경험을 최적화하는 데 가장 널리 사용됩니다. 기록적인 플래시 로더의 이점을 누리세요. 최대 3MiB/s RAM의 다운로드 속도와 MCU 플래시 메모리에서 중단점을 무제한 설정할 수 있습니다. 또한 J-Link는 CortexM0+를 포함해 다양한 CPU와 아키텍처를 지원합니다. 자세한 사항은 Segger J-Link 디버그 프로브 페이지를 참조하세요.

그림 2-3에서 XDS110 프로브의 주요 기능 영역 및 MSPM0 대상 인터페이스에 대한 개략적 다이어그램을 확인할 수 있습니다.

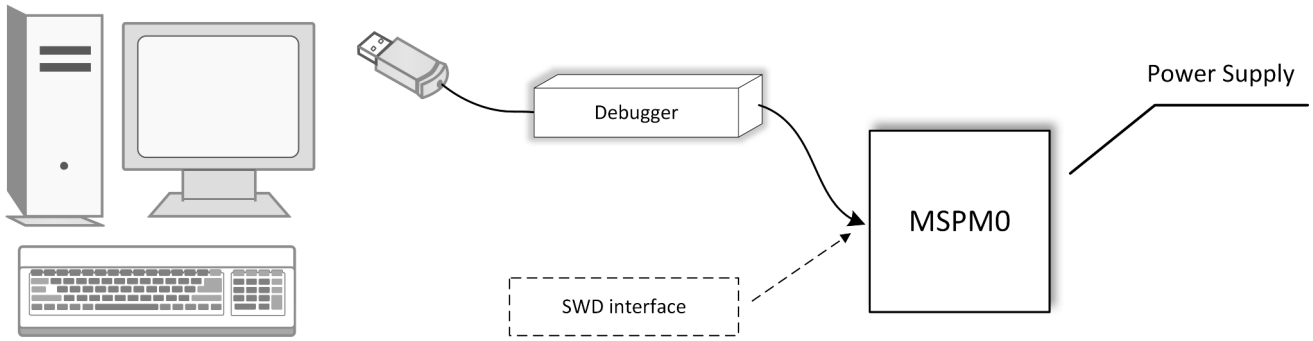

그림 2-3. MSPM0 디버깅

2.4 마이그레이션 프로세스

마이그레이션의 첫 단계는 포트폴리오를 검토하고 가장 적합한 MSPM0 MCU를 선택하는 것입니다. MSPM0 MCU를 선택하고 나면 개발 키트를 선택합니다. 개발 키트에는 구매 가능한 LaunchPad 키트와 대상 소켓 보드(Target-Socket Board)를 위한 설계 파일이 포함되어 있습니다. 또한 TI는 IT Resource Explorer 내에 Code Composer Studio IDE 데스크톱과 클라우드 버전의 구성 요소로 포함되어 있는 MSPM0 SDK(소프트웨어 개발 키트)가 무료로 제공됩니다. STM32에서 MSPM0로 소프트웨어를 포팅하는 데 도움이 되는 정보는 이 애플리케이션 노트의 주변 장치 섹션을 참조하세요. 마지막으로, 소프트웨어 포팅이 완료되면 TI 디버깅 툴을 사용하여 애플리케이션을 다운로드하고 디버깅합니다.

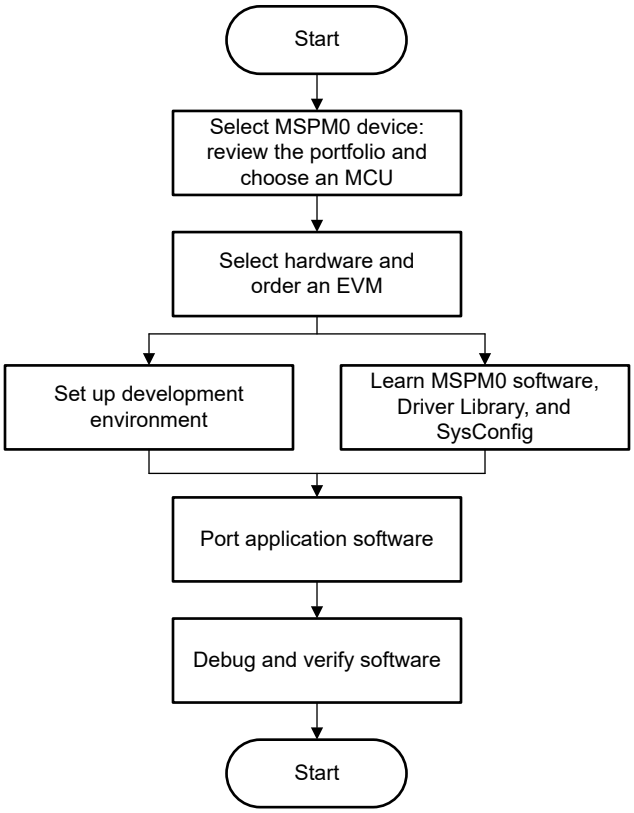

그림 2-4. MSPM0 마이그레이션 순서도

2.5 마이그레이션 및 포팅 예제

이 섹션에서는 TI 에코시스템과 보다 친숙해지고 MSPM0을 가장 잘 시작하는 방법을 설명하기 위해 기본 애플리케이션의 마이그레이션 과정을 단계별로 설명합니다.

STM32에서 MSPM0로 포팅하는 과정을 보여주기 위해, 이 설명에는 기존 ST UART 예제를 출발점으로 사용해 기본 저전력 UART 모니터 애플리케이션을 STM32G0x에서 MSPM0 장치로 포팅하는 과정이 포함되어 있습니다.

1단계. 적절한 MSPM0 MCU를 선택하세요

마이그레이션의 첫 번째 단계는 애플리케이션에 적합한 MSPM0 장치를 선택하는 것입니다. 이를 위해 본 가이드의 포트폴리오 섹션을 사용해 MSPM0 제품군을 선택할 수 있습니다. 제품 선택 도구를 사용해 특정 장치로 범위를 좁힙니다. STM32G0와 MSPM0은 M0+ 코어를 공유하지만 메모리 크기, 전력 및 주요 주변 장치 등의 기능도 고려해야 합니다. 또한 MSPM0는 다수의 핀 대 핀 확장 가능 옵션을 제공하기 때문에 시스템의 다른 사항을 변경하지 않고도 더 크거나 작은 메모리 장치로 쉽게 확장할 수 있습니다.

이 예제에서는 MSPM0G3507을 이 애플리케이션에 가장 적합한 것으로 선택했습니다.

2단계. 하드웨어를 선택하고 EVM을 주문합니다

EVM(평가 모듈)을 사용하면 마이그레이션 절차를 가속화할 수 있습니다. MSPM0 MCU의 경우 LaunchPad 키트는 가장 쉽게 시작할 수 있는 하드웨어입니다. LaunchPad 키트에는 프로그래머가 내장되어 있고 신속한 개발이 가능하도록 설계되어 있기 때문에 사용하기 쉽습니다.

MSPM0G3507에는 소프트웨어 포팅에 사용할 수 있는 LaunchPad 개발 키트(LP-MSPM0G3507)가 들어 있습니다.

3단계. 소프트웨어 IDE 및 SDK를 설정합니다

소프트웨어를 포팅하려면 먼저 소프트웨어 개발 환경을 선택하고 설정해야 합니다. 섹션 2.1에는 MSPM0가 지원하는 IDE가 모두 나와 있습니다. 마이그레이션 및 포팅 절차는 어느 IDE를 선택하든 비슷합니다. MSPM0 SDK 최신 버전을 사용해야 합니다.

이 예제에서는 TI의 CCS를 IDE로 선택했습니다.

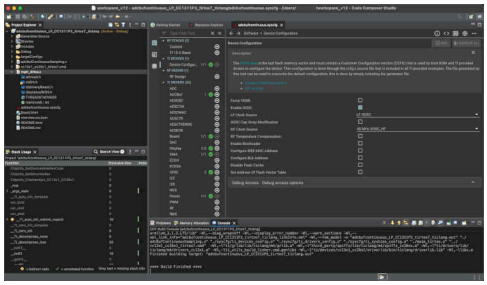

그림 2-5. Code Composer Studio IDE

4단계. 소프트웨어 포팅

환경이 준비되면 MSPM0 SDK를 사용하기 시작합니다. 앞서 언급했듯이 MSPM0 SDK는 STM32Cube 소프트웨어 패키지와 유사합니다. MSPM0 SDK에는 소프트웨어 개발을 위한 여러 계층이 포함되어 있습니다. MSPM0 TI 드라이버는 STM32Cube HAL과 유사한 수준에서 작동하는 반면, MSPM0 DriverLib은 STM32Cube 낮은 수준 드라이버와 비슷합니다. 대부분의 MSPM0 사용자는 DriverLib 레벨 소프트웨어가 자신의 애플리케이션에 가장 적합하다고 판단하기 때문에 대부분의 MSPM0 소프트웨어 예제도 DriverLib 기반입니다. 이 예제에서는 DriverLib을 사용합니다.

프로젝트를 포팅할 때 한 가지 옵션은 각 코드 섹션을 그에 상응하는 MSPM0 DriverLib API로 교체하는 것이지만, 이는 일반적으로 가장 쉬운 경로는 아닙니다. 보통 먼저 포팅 대상 응용 프로그램 코드를 이해하는 것이 가장 좋습니다. 그런 다음 가장 가까운 MSPM0 예제 프로젝트로 시작하여 원래 코드 기능에 맞게 수정합니다. 이 프로세스는 STM32CubeG0의 저전력 UART 예제를 사용하여 아래에 표시됩니다. 다수의 주변 장치를 사용하는 보다 복잡한 프로젝트의 경우 일반적으로 각 주변 장치마다 이 프로세스가 반복됩니다.

4a 단계: 애플리케이션을 이해합니다

다음 설명은 'LPUART_WakeUpFromStop_Init'이라는 STM32CubeG0의 예제 프로젝트에서 가져온 것입니다.

```
@par 예제 설명

LPUART_RX 핀에 수신된 문자가 저전력 모드에서 MCU를 활성화할 수 있도록 GPIO 및 LPUART 주변 장치 구성. 이 예제는
LPUART LL API를 기반으로 합니다. 주변 장치 초기화는 LL 초기화 기능을 사용하여 LL 초기화 사용을 시연합니다.

LPUART 주변 장치는 비동기 모드(9600 보드, 8 데이터 비트, 1 시작 비트, 1 정지 비트, 패리티 없음)로 구성됩니다.
하드웨어 흐름 제어는 사용하지 않습니다.
LPUART 클록은 HSI를 기준으로 합니다.

예제 실행:
재설정 및 시스템 구성에서 시작한 후 LED3이 3초 동안 빠르게 깜박인 다음 MCU가 "정지 0" 모드(LED3 꺼짐)로 들어갑니
다. "중지 0" 모드 기간 후 PC COM 포트에서 LPUART의 첫 번째 문자를 수신(예: HyperTerminal 사용)하면 MCU가 "중
지 0" 모드에서 깨어납니다.

받은 문자 값을 확인합니다.
- 특정 값('S' 또는 's')에서 LED3이 켜지고 프로그램이 종료됩니다.
- 'S' 또는 's'와 다른 경우, 프로그램은 3초 동안 LED3을 빠르게 깜박인 후 다시 "Stop 0" 모드로 들어가 다음 문자가
깨우기를 기다립니다.
```

첫 번째 단계는 MCU의 주요 설정을 이해하는 것입니다. 보통 클록 속도와 전원 정책이 주요 설정입니다. 이 예제에서는 유일하게 중요한 설정은 UART가 저전력 Stop0 모드에서 작동한다는 것이기 때문에 일반 클록 주파수가 지정되지 않았습니다. 저전력 UART 클록이 'HIS' 또는 고속 내부 오실레이터를 기반으로 한다고 나와 있는데, 이는 외부 크리스탈이 사용되고 있지 않다는 뜻입니다. UART는 9,600 보드, 8 데이터 비트, 1 시작 및 정지 비트, 패리티 없음으로 실행됩니다. 하드웨어 흐름 제어가 사용되지 않습니다. 애플리케이션 측에서 수신될 'S' 또는 's'가 있는지 확인하고 LED가 깜박입니다.

4b 단계: 가장 가까운 MSPM0 예제를 찾습니다

다음 단계는 STM32G0과 MSPM0에 대한 UART 모듈의 차이점을 이해하고 MSPM0 SDK에서 가장 근접한 예제를 찾는 것입니다. 이 단계는 섹션 4에서 UART 섹션을 참조하면 쉽게 완료할 수 있습니다. 이 섹션에서는 UART 모듈의 차이점과 UART 관련 MSPM0 SDK 코드 예제 링크에 대해 설명합니다. SDK에서 이 예제에 가장 가까운 예제는 아마도 "장치가 STANDBY 모드에 있는 동안 인터럽트를 사용하는 UART RX/TX 에코"의 uart_echo_interrupts_standby일 것입니다.

이 MSPM0 예제는 비슷하지만 포팅되는 것과 동일하지는 않습니다. 이 예제는 대기 모드로 전환하는 것이며, 대기 모드는 중지 모드보다 낮은 전력 모드입니다. UART 통신 설정과 어느 GPIO를 사용 중인지 반드시 확인해야 합니다. 마지막으로 특정 문자에 대한 모니터링의 애플리케이션 계층을 추가해야 합니다.

4c 단계: 예제를 가져와 수정합니다

비슷한 예제를 찾으면, CCS를 열고 **프로젝트 > CCS 프로젝트 가져오기...**를 차례로 선택해 코드 예제를 가져온 다음 MSPM0 SDK의 예제 폴더로 들어갑니다. 예제를 가져오기합니다. 여기 보이는 것이 가져오기한 *uart_echo_interrupts_standby* 예제입니다. 이것은 SysConfig 프로젝트이기 때문에 메인 C 파일이 간단합니다. 먼저 SysConfig가 자동 생성한 함수인 SysConfig driverlib 초기화를 호출하여 장치를 구성합니다. 그런 다음 UART 인터럽트를 활성화합니다. 마지막으로 UART 트랜잭션을 기다리는 대기 상태로 들어갑니다. UART 트랜잭션을 수신하면 데이터를 즉시 다시 에코하여 활성화합니다.

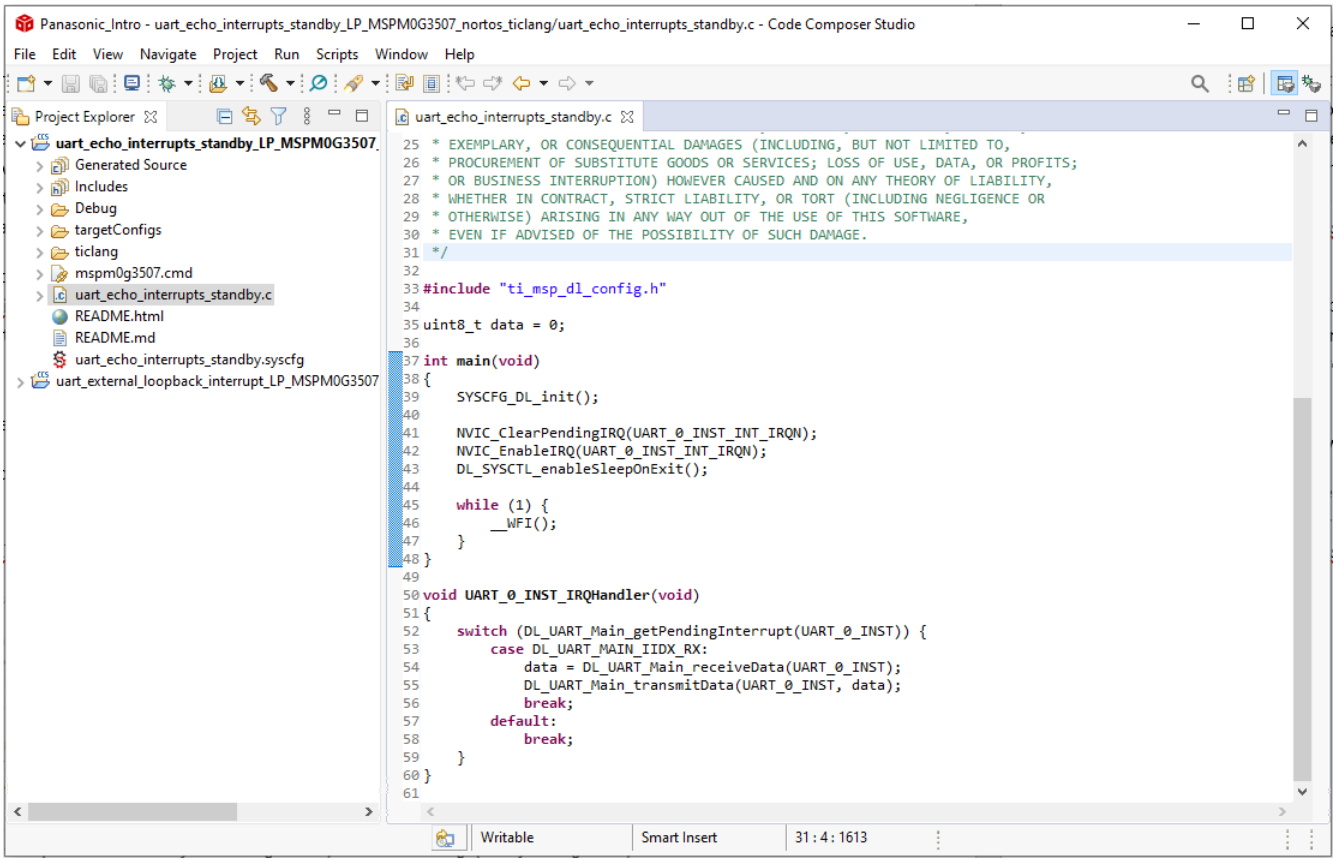

그림 2-6. uart_echo_interrupts_standby 예제

SysConfig 구성을 보려면 기본 설정상 SYSCTL 탭에서 열리는 .syscfg 파일을 엽니다. SysConfig 사용에 대한 자세한 지침은 MSPM0 SDK의 SysConfig 가이드를 참조하십시오.

가장 먼저 주목할 사항은 전원 정책입니다. 이 MSPM0 예제에서는 Standby0 모드를 사용하지만 Stop0 모드를 사용하는 것이 목표입니다. 드롭다운 목록을 클릭해 올바른 저전력 모드를 선택할 수 있습니다. 이 탭에서 클록과 오실레이터를 모두 구성할 수 있지만 지금 당장은 괜찮습니다.

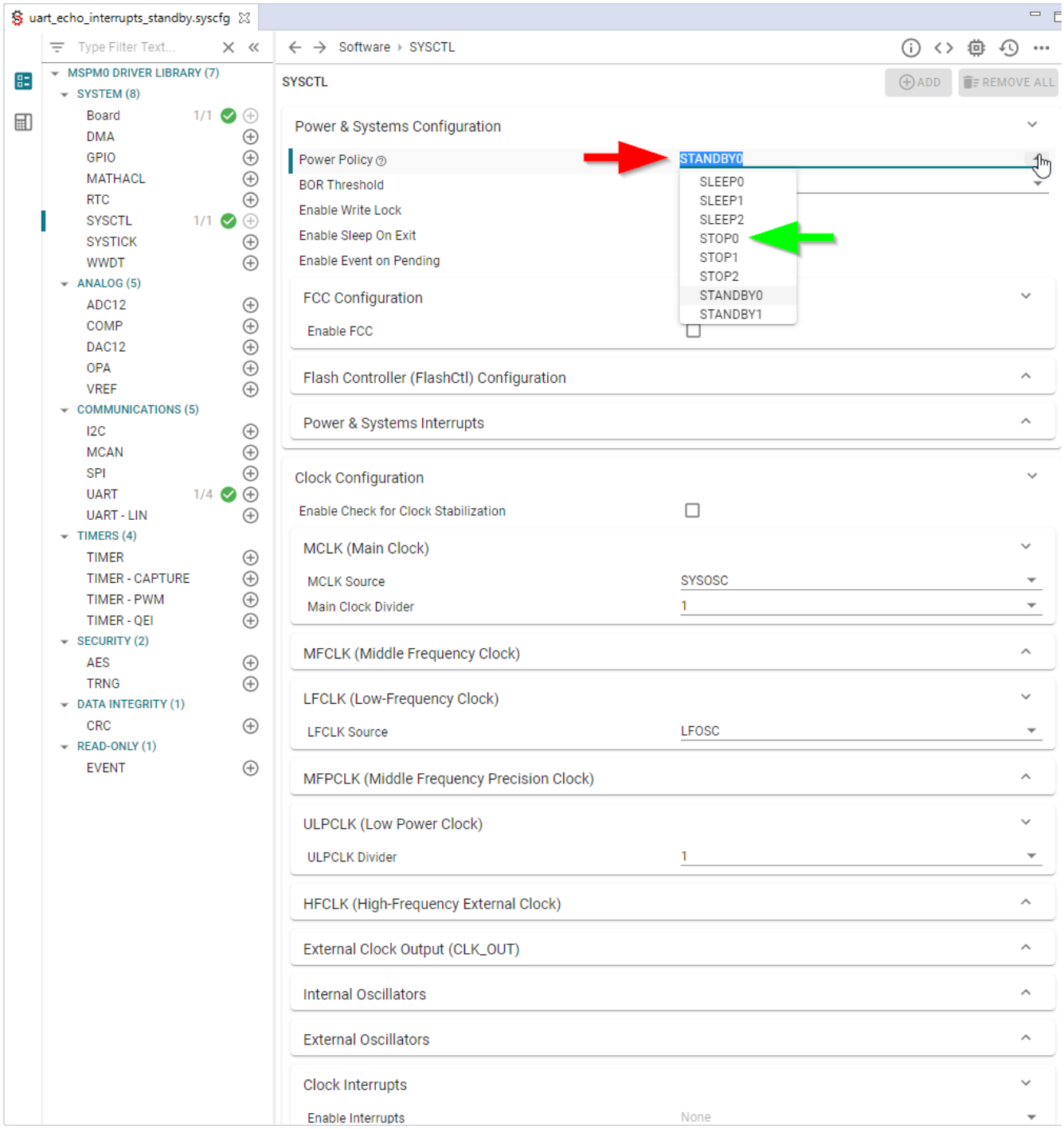

그림 2-7. 전원 모드 구성

그런 다음 UART 탭에서 UART 통신 설정을 확인합니다(그림 2-8 참조). 이 경우 보드율은 이미 9,600으로 설정되어 있고 나머지 통신 설정은 정확합니다. 수신 인터럽트는 이미 활성화되어 메인 프로그램에서 사용되고 있습니다. 또한 오른쪽 상단에서 칩 아이콘을 클릭하고 강조표시된 UART에 핀을 확인해 사용 중인 UART 모듈과 핀을 점검합니다. MSPM0G3507 LaunchPad 키트의 백채널 UART에 이미 연결되어 있기 때문에 여기서는 아무것도 변경할 필요가 없습니다.

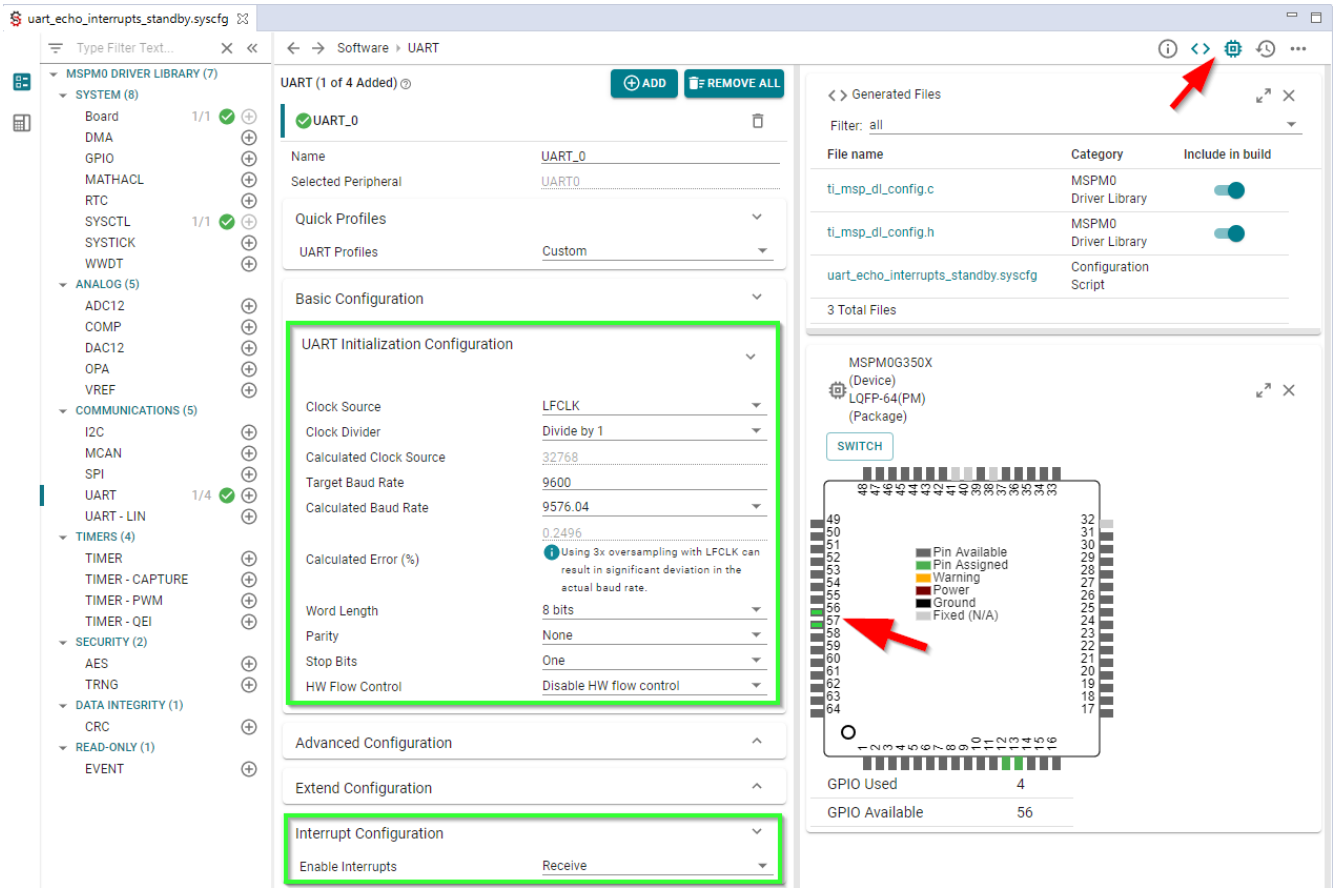

그림 2-8. UART 구성

이 예제에는 현재 LED 구동을 위해 구성된 GPIO가 없지만 구성을 쉽게 추가할 수 있습니다(그림 2-9 참조). GPIO는 페이지 상단에서 **+ADD** 버튼을 클릭해 추가할 수 있습니다. GPIO 포트와 핀은 이름을 지정할 수 있으며, 이 경우 각각 'LED'와 'RED'로 지정되어 있습니다. 이 GPIO는 출력으로 설정된 다음 포트 A 핀 0(PA0)에 배치됩니다. LaunchPad 키트에서 이 GPIO는 단순한 빨간색 LED에 연결됩니다.

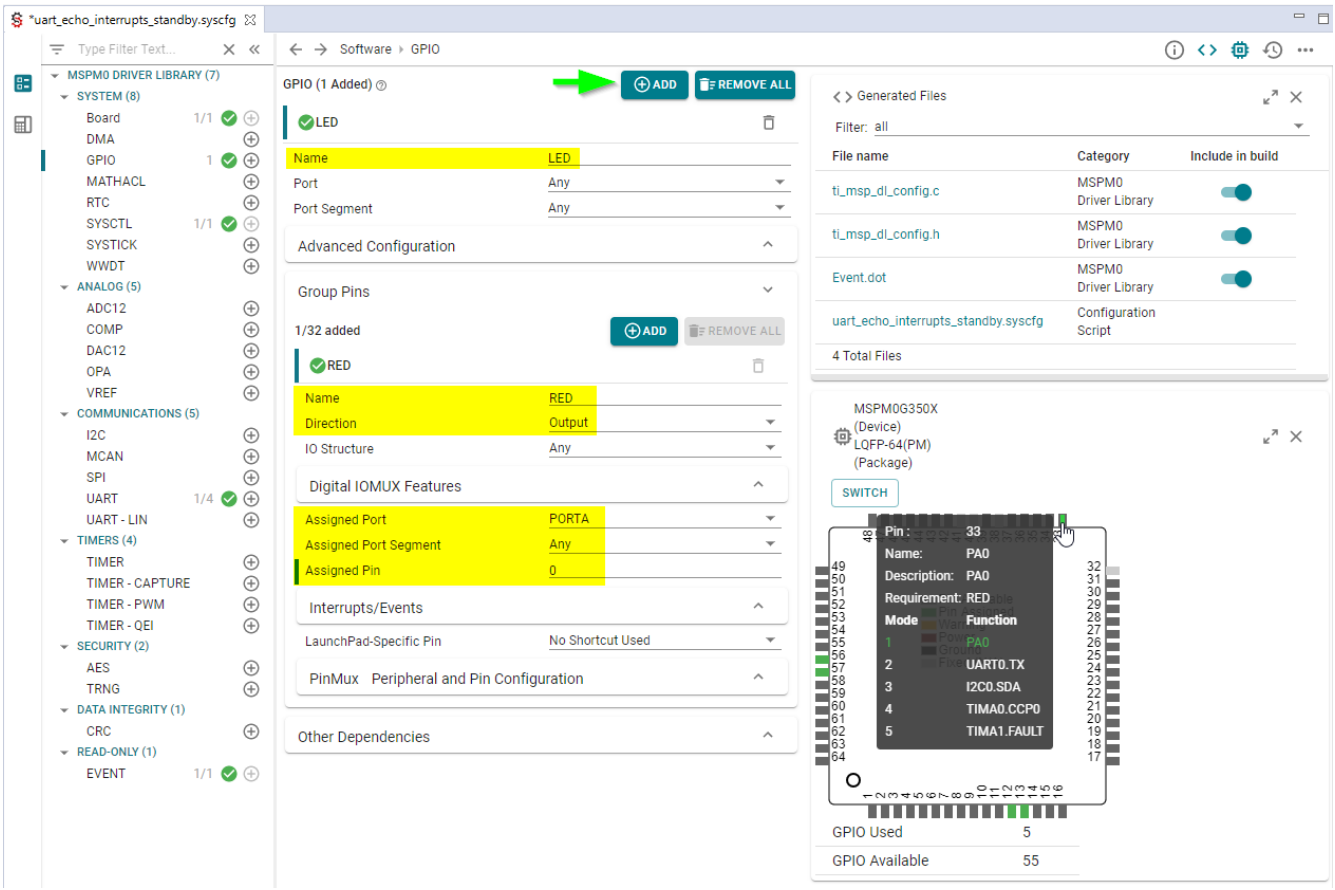

그림 2-9. GPIO 구성

프로젝트를 저장하고 재빌드하면 SysConfig가 예제를 위해 ti_MSP_DL_config.c 및 ti_msp_dl_config.h 파일을 업데이트합니다. 이제 예제 하드웨어 구성이 변경되어 포팅되는 원래 소프트웨어의 전체 기능과 일치합니다. 이제 유일하게 남은 일은 애플리케이션 수준 소프트웨어에서 입력되는 UART 바이트를 확인하고 LED를 전환하는 것입니다. 그렇게 하려면 소량의 코드를 메인 C 파일로 이동하면 됩니다.

*uart_echo_interrupts_standby.c uart_echo_interrupts_standby.syscfg

```
*/

#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;
    }
}
```

그림 2-10. 애플리케이션 코드 변경 사항

애플리케이션 코드에서는 두 가지 변경되었습니다. 먼저, DL_sysctl_disablesSleepOnExit()를 사용해 MSPM0가 각 UART RX에서 잠시 동안 활성화되도록 합니다. 다음, UART RX 데이터에 대한 간단한 검사가 추가되고, 'S' 또는 's'가 수신되면 빨간색 LED가 켜집니다. 그런 다음 꺼집니다.

5단계: 디버그 및 검증

다음 그림은 9,600 보드에서 UART 통신, 빨간색 LED가 올바르게 켜지고 꺼지는 것을 보여주는 논리 분석기 화면 캡처입니다. 코드는 모든 UART 문자를 에코하지만 올바른 문자가 수신된 경우에만 LED를 켭니다.

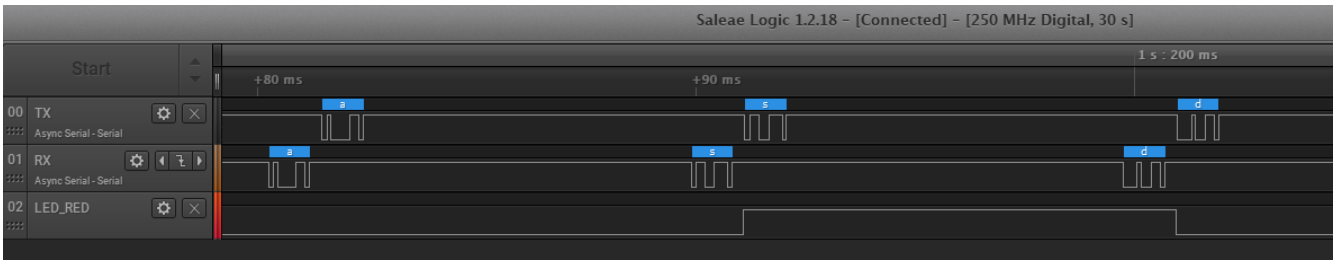

그림 2-11.

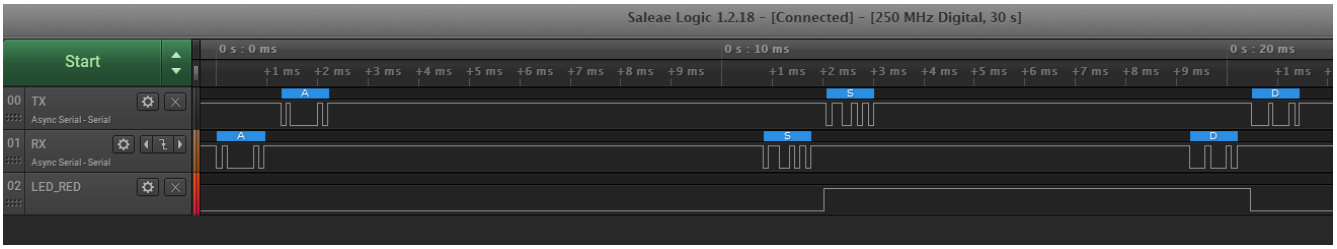

그림 2-12.

소프트웨어가 성공적으로 포팅되었습니다! 그 외에도 주변 장치가 더 있는 경우 주변 장치마다 이 과정을 반복하고 SysConfig를 사용해 각 블록을 결합하면 됩니다.

3 코어 아키텍처 비교

3.1 CPU

STM32G0 및 MSPM0 부품 제품군은 모두 Arm Cortex® M0+ CPU 코어 아키텍처 및 명령어 세트를 기반으로 합니다. 아래 표는 MSPM0G 및 MSPM0L 제품군에서 CPU의 일반적인 기능을 STM32G0와 비교한 대략적인 개요입니다. 인터럽트와 예외는 인터럽트와 예외를 비교한 것으로, 각 장치별로 M0 아키텍처 내에 포함되어 있는 NVIC(Nested Vectored Interrupt Controller) 주변 장치에서 어떻게 매핑되어 있는지 보여줍니다.

표 3-1. CPU 기능 세트 비교

주요 기능	STM32G0	MSPM0G	MSPM0L
아키텍처	Arm Cortex-M0+	Arm Cortex-M0+	Arm Cortex-M0+
최대 MCLK	64 MHz	80 MHz	32 MHz
CPU 명령 캐시	2x64비트 라인(16바이트)	4X64비트 라인(32바이트)	2x64비트 라인(16바이트)
프로세서 추적 기능	아니요	예, 일체형 마이크로 트레이스 버퍼	아니요
MPU(메모리 보호 유닛)	예	예	아니요
SYSTICK(시스템 타이머)	예	예 - 24비트	예 - 24비트
NVM 프리페치	예	예	예
하드웨어 곱셈	예	예	예
하드웨어 중단점/감시점	4/2	4/2	4/2
부팅 루틴 스토리지	플래시(시스템 메모리)	ROM	ROM
부트스트랩 로더 스토리지	플래시(시스템 메모리)	ROM	ROM
부트로더 인터페이스 지원(1) (2)	UART, I2C, SPI, USB, FDCAN	UART, I2C, 사용자 확장 가능	UART, I2C, 사용자 확장 가능
DMA	예	예	예

(1) 가용성은 장치별 데이터시트를 참조하세요.
(2) 기타 인터페이스는 이후 장치 릴리스에서 제공됩니다.

3.2 임베디드 메모리 비교

3.2.1 플래시 기능

MCU의 MSPM0 및 STM32G0 제품군에는 실행 가능한 프로그램 코드와 애플리케이션 데이터 저장에 사용되는 비휘발성 플래시 메모리가 포함되어 있습니다.

표 3-2. 플래시 기능 비교

특징	STM32G0	MSPM0
플래시 메모리	STM32G0B1xx, G0C1xx(최대 512KB) STM32G071xx, G081xx(최대 128KB) STM32G031xx, G041xx, G051xx, G061xx(최대 64KB)	MSPM0Gx x 범위는 128KB~32KB입니다 MSPM0Lxx 범위는 64KB~8KB입니다
메모리 구성	뱅크 1개 - 장치 최대 128KB 뱅크 2개 - >128KB 장치	뱅크 1개 - 장치 최대 256KB 뱅크 2개 - >256KB 장치
플래시 대기 상태	0(HCLK ≤ 24MHz) 1(HCLK ≤ 48MHz) 2(HCLK ≤ 64MHz)	0(MCLK, CPUCLK ≤ 24MHz) 1(MCLK, CPUCLK ≤ 48MHz) 2(MCLK, CPUCLK ≤ 80MHz)
플래시 단어 크기	64비트 + 8 ECC 비트	동일
프로그래밍 해상도	단일 단어 크기	단일 단어, 32, 16 또는 8비트(바이트)
다중 단어 프로그래밍	32단어(256바이트)	2, 4 또는 8단어(최대 64바이트)
삭제	페이지 크기 = 2KB 뱅크 삭제(단일 뱅크) 대량 삭제(모든 뱅크)	섹터 크기 = 1KB 뱅크 삭제(최대 256KB)
쓰기 보호	예(뱅크당 쓰기 보호 영역 2개)	예, 정적 및 동적
읽기 보호	예	예

표 3-2. 플래시 기능 비교 (continued)

특징	STM32G0	MSPM0
플래시 메모리 읽기 작업	64비트 플래시 단어 크기 + 8 ECC 비트	동일 - 선택 사항 ECC가 있는 경우
플래시 메모리 쓰기 작업	64비트 플래시 단어 크기 + 8 ECC 비트	동일 - 선택 사항 ECC가 있는 경우
ECC(오류 코드 수정)	64비트에 대해 8비트	동일
보안 가능 메모리 영역	예, 메인 메모리	아니요
정보 메모리	예	예(NONMAIN)
OTP 데이터 영역	1KB	아니요
프리페치	예	예
CPU 명령 캐시	64비트 캐시 라인 2개(16바이트) 4x 32비트 명령 또는 8x 16비트 명령	64비트 캐시 라인 4개(32바이트) 8x 32비트 명령 또는 16x 16비트 명령

앞의 표에 나와 있는 플래시 메모리 기능 외에, MSPM0 플래시 메모리에는 다음 기능도 있습니다.

- 전체 공급 전압 범위에 걸쳐 회로 내 프로그램 및 삭제 지원
- 내부 프로그래밍 전압 생성
- 플래시 메모리 하위 32KB에서 프로그램/삭제 주기 최대 10만 회까지, 나머지 플래시 메모리에서 프로그램/삭제 주기 최대 1만 회까지 EEPROM 에뮬레이션 지원(32KB 장치의 경우 전체 플래시 메모리에서 주기 10만회 지원)

3.2.2 플래시 구성

플래시 메모리는 애플리케이션 코드와 데이터, 장치 부팅 구성, 공장 출하 시 TI가 프로그래밍하는 매개 변수 저장 용도로 사용됩니다. 플래시 메모리는 1개 이상의 뱅크로 구성되며, 각 뱅크 내 메모리는 다시 1개 이상의 논리 메모리 지역으로 매핑되어 애플리케이션이 사용할 수 있도록 시스템 주소 공간에 할당됩니다.

메모리 뱅크

대부분의 MSPM0 장치는 단일 플래시 뱅크(BANK0)를 구현합니다. 단일 플래시 뱅크 장치에서는 프로그램/삭제 작업이 지속적으로 이루어지면서 작업이 완료되고 플래시 컨트롤러가 뱅크에 대한 제어를 풀 때까지 플래시 메모리에 대한 모든 읽기 요청을 보류합니다. 플래시 뱅크가 여러 개 있는 장치의 경우에도 뱅크에 프로그램/삭제 작업이 그 프로그램/삭제 작업을 실행하고 있는 뱅크에 대해 이루어지는 읽기 요청을 보류하지만 다른 뱅크에 대한 요청은 보류하지 않습니다. 따라서 뱅크가 여러 개 있는 경우 다음과 같은 애플리케이션 케이스가 가능합니다.

- 듀얼 이미지 펌웨어 업데이트(애플리케이션은 첫 번째 플래시 뱅크에서 코드를 실행하는 동안 두 번째 이미지는 애플리케이션 실행 보류 없이 두 번째 대칭적 플래시 뱅크로 프로그램됩니다)
- EEPROM 에뮬레이션(애플리케이션이 첫 번째 플래시 뱅크에서 코드를 실행하는 동안 애플리케이션 실행 보류 없이 두 번째 플래시 뱅크를 사용해 데이터 쓰기를 진행합니다)

플래시 메모리 영역

각 뱅크 내 메모리는 각 뱅크 내 메모리가 지원하는 기능에 따라 1개 이상의 논리적 영역으로 매핑됩니다. 모두 4개의 영역이 있습니다.

- FACTORY(공장) - 장치 ID와 그 외 매개 변수
- NONMAIN - 장치 부팅 구성(BCR 및 BSL)
- MAIN - 애플리케이션 코드 및 데이터
- DATA - 데이터 또는 EEPROM 에뮬레이션

뱅크가 하나뿐인 장치의 경우 BANK0(유일한 뱅크)에서 FACTORY, NONMAIN 및 MAIN 영역을 구현하며 데이터 영역은 제공되지 않습니다. 뱅크가 여러 개인 장치에서도 BANK0에서 FACTORY, NONMAIN 및 MAIN 영역을 구현하지만 나머지 뱅크(BANK1~BANK4)에서 MAIN 또는 DATA 영역을 구현할 수 있습니다.

NONMAIN 메모리

NONMAIN은 장치 부팅을 위해 BCR과 BSL이 사용하는 구성 데이터를 저장하는 플래시 메모리 전용 영역입니다. 이 영역은 다른 용도로는 사용되지 않습니다. BCR과 BSL은 모두 기본값으로 남겨 두거나(보통 개발 및 평가 단계에서 하듯이)

NONMAIN 플래시 영역으로 프로그램된 값을 변경해 특정 목적으로 수정(보통 프로덕션 프로그래밍 단계에서 하듯이)할 수 있는 구성 정책을 갖고 있습니다.

3.2.3 임베디드 SRAM

MCU 제품군 MSPM0 및 STM32G0에서는 애플리케이션 데이터 저장 용도로 SRAM을 사용합니다.

표 3-3. SRAM 기능 비교

주요 기능	STM32G0	MSPM0
SRAM 메모리	STM32G0B1xx, G0C1xx: 144KB(SRAM 패리티 활성화된 경우 128KB) STM32G071xx, G081xx: 36KB(SRAM 패리티 활성화된 경우 32KB) STM32G051xx, G061xx: 18KB(SRAM 패리티 활성화된 경우 16KB) STM32G031xx, G041xx: 8KB(SRAM 패리티 활성화된 경우 8KB) 제로 대기 상태	MSPM0Gxx: 32KB~16KB MSPM0Lxx: 4KB~2KB 제로 대기 상태 일부 장치에는 SRAM 패리티와 ECC가 포함됩니다. 자세한 사항은 장치 데이터 시트를 참조하세요
최대 CPU 클록 주파수에서 제로 대기 상태	예	예
액세스 해석	바이트, 하프워드(16비트) 또는 풀워드(32비트)	바이트, 하프워드(16비트) 또는 풀워드(32비트)
패리티 검사	예	예

MSPM0 MCU에는 장치의 지원 대상 CPU 주파수 범위에 걸쳐 제로 대기 상태 액세스를 갖는 저전력 고성능 SRAM이 포함되어 있습니다. SRAM은 코드 외에 호출 스택, 힙 및 글로벌 데이터 같은 휘발성 정보를 저장하는 용도로 사용할 수 있습니다. SRAM 콘텐츠는 실행, 절전, 중지 및 대기 작동 모드에서 완전히 유지되지만 종료 모드에서는 삭제됩니다. 애플리케이션이 1KB 해상도로 SRAM의 하위 32KB를 동적으로 쓰기 보호할 수 있도록 쓰기 보호 메커니즘이 제공됩니다. SRAM이 32KB 미만인 장치에서는 전체 SRAM에 대해 쓰기 보고가 제공됩니다. 쓰기 보호는 CPU 또는 DMA가 의도치 않게 코드를 덮어쓰기하는 경우에 대비해 일정 수준의 보호를 제공하기 때문에 SRAM에 실행 가능한 코드를 넣을 때 유용합니다. SRAM에 코드를 삽입하면 제로 대기 상태 작동을 활성화하고 전력 소비를 줄여 필수 루프의 성능을 개선할 수 있습니다.

3.3 전원 켜기 및 재설정 요약 및 비교

STM32G0 장치와 마찬가지로 MSPM0 장치는 최소 작동 전압을 제공하며, 장치 또는 장치의 일부를 재설정 상태로 유지하여 장치가 제대로 시동되도록 하는 모듈을 갖추고 있습니다. 표 3-4 에서는 두 제품군 간에 이러한 작업이 어떻게 수행되는지, 그리고 여러 제품군에 걸쳐 어느 모듈이 전원 켜기 프로세스 및 재설정을 제어하는지 보여줍니다.

표 3-4. 전원 켜기 비교

STM32G0 장치		MSPM0 장치	
전원 켜기 및 재설정을 담당하는 모듈	PWR(전원) 및 RCC(재설정 및 클록 제어) 모듈	전원 켜기 및 재설정을 담당하는 모듈	PMCU(전원 관리 및 클록 장치)
전압 수준에 따른 재설정			
POR(Power-On Reset)	전체 장치 재설정. 전원 켜기를 위한 1단계 전압 해제. 전원 차단을 위한 최저 전압 수준.	POR(Power-On Reset)	전체 장치 재설정. 전원 켜기를 위한 1단계 전압 해제. 전원 차단을 위한 최저 전압 수준.
구성 가능 BOR(브라운아웃 재설정)	경우에 따라 프로그래밍 가능. 전원을 켤 때 재설정 상태를 해제하거나 전원을 끌 때 장치를 재설정하는 전압 레벨을 설정.	구성 가능한 BOR(브라운아웃 재설정)	STM32G0 BOR 및 PVD 기능을 결합한 다양한 전압 임계값을 사용하여 재설정 또는 인터럽트로 구성 가능.
PVD(프로그래머블 전압 감지기)	인터럽트를 제공할 수 있는 구성 가능한 전압 모니터.		

STM32G0는 다양한 재설정 도메인을 정의하지만 MSPM0 장치에는 재설정 상태 수준이 여러 가지 있습니다. MSPM0 장치의 경우 재설정 수준에 정해진 순서가 있으며, 특정 수준이 트리거되면 이후 수준들은 모두 장치가 RUN 모드로 해제될 때까지 재설정됩니다. 표 3-5에서는 간단한 설명과 함께 STM32G0 재설정 도메인과 MSPM0 재설정 상태를 비교합니다. 그림 3-1는 모든 MSPM0 재설정 상태 간의 관계를 보여줍니다.

표 3-5. 재설정 도메인 비교

STM32G0 재설정 도메인		MSPM0 재설정 상태[1]	
전원 재설정 도메인	보통 트리거는 POR, BOR, 그리고 대기 또는 종료 모드에서 나가기입니다. VCORE 도메인을 벗어나는 경우를 제외하고 모든 레지스터는 재설정됩니다.	POR	일반적인 트리거: POR 전압 수준, SW 트리거, NRST는 1초 미만 동안 낮게 유지. 종료 메모리 재설정, NRST 및 SWD 재활성화, BOR 트리거
		BOR	일반적인 트리거: POR 또는 BOR 전압 수준, 종료 모드에서 나가기. PMU, VCORE 및 연결 논리 재설정. BOOTRST 트리거.
정확한 상응 아님. 부팅 구성은 재설정 후 SYSCLK의 네 번째 클록 사이클에서 읽습니다.		BOOTRST(부팅 리셋)	일반적인 트리거: BOR 또는 소프트웨어 트리거, 치명적인 클록 장애, 1초 미만 동안 NRST를 낮은 상태로 유지. 부팅 구성 루틴 실행. RTC, 클록 및 IO 구성을 포함한 대부분의 코어 논리 및 레지스터를 리셋합니다.[2] SRAM 전력 사이클링 후 상실되었습니다. SYSRST를 트리거합니다.
시스템 재설정 도메인	시스템 재설정은 클록 제어 및 상태 레지스터(RCC_CSR)와 RTC 도메인의 레지스터의 재설정 플래그를 제외하고 모든 레지스터를 재설정 값으로 설정합니다.	SYSRST(시스템 리셋)	일반적인 트리거: BOOTRST, BSL 진입 또는 종료, 워치독 타이머, 소프트웨어 트리거, 디버그 하위 시스템. CPU 상태, 그리고 RTC, LFCLK, LFXT 및 SYSOSC 주파수 수정 루프를 제외한 모든 주변 장치를 재설정합니다. 종료 시 장치가 RUN 모드로 들어갑니다.
상응 없음		CPURST(CPU 한정 재설정)	소프트웨어 및 디버그 하위 시스템 트리거만 해당. CPU 논리만 재설정. 주변 장치 상태는 영향을 받지 않습니다.
RTC 도메인	이전에 두 전원이 모두 꺼진 경우 소프트웨어 또는 VDD 또는 VBAT 전원이 켜지면서 트리거됩니다. LSE 오실레이터, RTC, 백업 레지스터 및 RCC RTC 도메인 제어 레지스터만 재설정합니다.	RTC 및 관련 클록은 BOOTRST, BOR 또는 POR를 통해 재설정됩니다.[2]	

(1) 일부 재설정 트리거만 설명합니다. 사용 가능한 모든 재설정 트리거에 대한 설명은 장치 TRM의 PMCU 장을 참조하세요.
(2) BOOTRST 원인이 NRST 또는 소프트웨어 트리거를 통한 것인 경우, RTC, LFCLK 및 LFXT/LFCK_IN 구성과 IOMUX 설정은 외부 재설정을 통해 RTC가 작동을 유지할 수 있도록 재설정되지 않습니다.

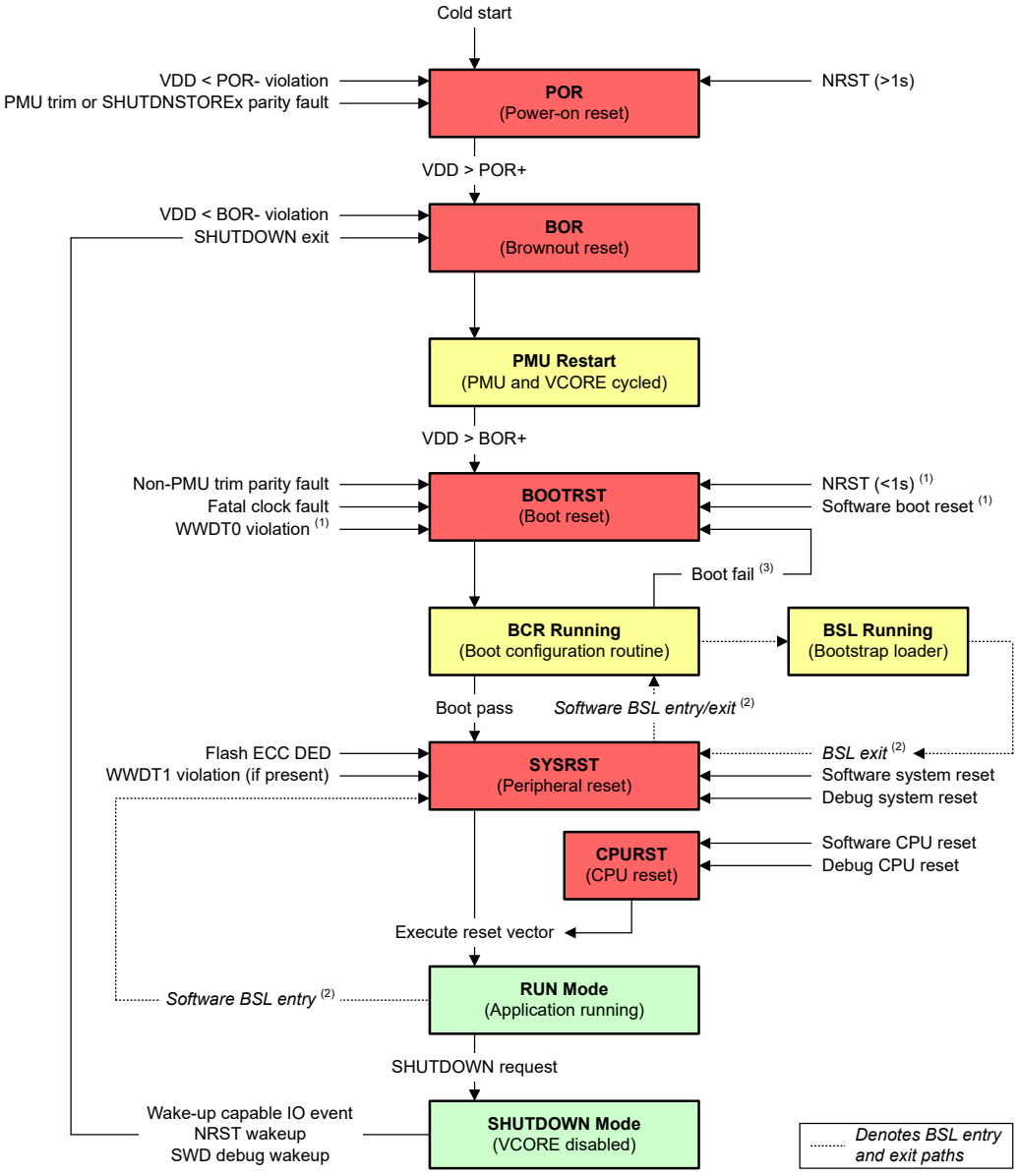

그림 3-1. MSPM0 재설정 수준

3.4 클록 요약 및 비교

STM32G와 MSPM0에는 1차 클록을 소싱하는 내부 오실레이터가 포함되어 있습니다. 클록들은 나눠서 다른 클록을 소싱하도록 하고 여러 주변 장치로 분산시킬 수 있습니다.

표 3-6. 오실레이터 비교

STM32G0 오실레이터	MSPM0 오실레이터
HSI16RC 16MHz	SYSOSC(1)
HSI48RC 48MHz	SYSOSC
LSI RC 32kHz	LFOSC
HSE OSC 4~48MHz	HFXT
LSE OSC 32kHz	LFXT
I2S_CLKIN	HFCLK_IN(디지털 클록)

(1) SYSOSC는 32MHz, 24MHz, 16MHz 또는 4MHz로 프로그래밍할 수 있습니다.

표 3-7. 클록 비교

STM32G 클록	MSPM0 클록
HSISYS	해당 없음
PLLPCLK	SYSPLLCLK1
PLLQCLK	SYSPLLCLK1
PLLRCLK	SYSPLLCLK0
해당 없음	SYSPLLCLK2x(1)
SYSCLK	BUSCLK(2)
HCLK	MCLK
HCLK8	CPUCLK
합니다.	BUSCLK
TIMPCLK	BUSCLK
LPTIMx_IN	LFCLK_IN

(1) SYSPLLCLK2x는 PLL 모듈 출력 속도의 두 배이며 분배할 수 있습니다.
(2) BUSCLK은 전원 도메인에 따라 다릅니다. 전원 도메인 0의 경우 BUSCLK는 ULPCLK입니다. 전원 도메인 1의 경우 BUSCLK은 MCLK입니다.

표 3-8. 주변 장치 클록 소스

주변 장치	STM32G 클록 소스	MSPM0 클록 소스
RTC	LSI, LSE, HSE/32	LFCLK(LFOSC, LFXT)
UART	PCLK, LSE, HSI16, SYSCLK	BUSCLK, MFCLK, LFCLK
SPI	찾아야 함	BUSCLK, MFCLK, LFCLK
I2C	PCLK, HSI16, SYSCLK	BUSCLK, MFCLK
ADC	HSI16, SYSCLK, PLLPCLK	ULPCLK, HFCLK, SYSOSC
CAN	PCLK, HSE, PLLQCLK	PLLCLK1, HFCLK
타이머	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

각 장치 제품군의 TRM에는 클록 트리가 있어서 클록 시스템을 시각화하는 데 유용합니다. Sysconfig는 주변 장치에 대한 클록 분할 및 소싱 옵션을 지원할 수 있습니다.

3.5 MSPM0 작동 모드 요약 및 비교

MSPM0 MCU는 애플리케이션 요구 사항에 따라 장치 전력 소비를 최적화할 수 있도록 5가지 주요 작동 모드(전력 모드)를 제공합니다. 사용 전력이 감소하는 순서대로 다음과 같은 모드가 있습니다. 실행, 설전, 중지, 대기, 종료(셧다운). CPU는 실행 모드에서 활성 상태로 코드를 실행합니다. 주변 장치 인터럽트 이벤트는 장치를 절전, 중지 또는 대기 모드에서 실행 모드로 깨울 수 있습니다. 종료(셧다운) 모드는 전력 소비를 최소화하기 위해 내부 코어 레귤레이터를 완전히 비활성화하며, 일부 IO의 NRST, SWD, 또는 논리 수준 일치를 통해서만 깨우는 것이 가능합니다. 또한, 실행, 절전, 중지 및 대기 모드에는 전력 소비량과 성능 간 균형을 맞추기 위해 구성 가능한 여러 정책 옵션(예: RUN.x)이 포함되어 있습니다.

MSPM0 장치는 성능과 전력 소비 간 균형을 더욱 향상시키기 위해 2개의 전원 도메인을 구현합니다. PD1(CPU, 메모리 및 고성능 주변 장치용)과 PD0(저속 저전력 주변 장치용)입니다. PD1은 항상 실행 및 절전 모드에서 켜지지만 다른 모드에서는 모두 비활성화됩니다. PD0은 실행, 절전, 중지 및 대기 모드에서 항상 전원이 공급됩니다. PD1과 PD0은 모두 종료 모드에서 비활성화됩니다.

작동 모드 비교

STM32G0 장치의 작동 모드도 비슷합니다. 아래 표는 STM32G0와 MSPM0 장치를 간단히 비교해서 보여줍니다.

표 3-9. STM32G0와 MSPM0 장치 간 작동 모드 비교

STM32G0			MSPM0		
모드		**설명**	**모드**	**설명**	
실행		완전한 클로킹 및 주변 장치를 사용할 수 있습니다	실행	0	완전한 클로킹 및 주변 장치를 사용할 수 있습니다
LP RUN		CPU는 2MHz로 제한됩니다		1	설정 주파수에서 SYSOSC, CPUCLK 및 MCLK 32kHz로 제한
				2	SYSOSC 비활성화, CPUCLK 및 MCLK 32 kHz로 제한
절전		CPU 클록되지 않음	절전	0	CPU 클록되지 않음
LP 절전		LP RUN과 동일하지만 CPU가 클록되지 않습니다		1	Run1과 동일하지만 CPU가 클록되지 않습니다
				2	Run2와 동일하지만 CPU가 클록되지 않습니다
중지	0	VCORE 도메인 클록이 비활성화됩니다	중지	0	Sleep 0 + PD1 비활성화
				1	Sleep 1 + SYSOSC 기어가 4MHz로 전환
	1	Stop 0 + 주 전원 레귤레이터 꺼짐		2	Sleep 2 + ULPCLK 32kHz로 제한
Standby		BOR 기능으로 최저 전력, RTC 사용 가능, 레지스터 설정 손실.	Standby	0	BOR 기능으로 최저 전력, 모든 PD0 주변 장치는 32kHz에서 ULPCLK 및 LFCLK를 수신할 수 있고, RTCCLK 로 RTC를 사용할 수 있습니다
				1	TIMG0 및 TIMG1만 32kHz에서 ULPCLK 또는 LFCLK를 수신할 수 있으며, RTCCLK 로 RTC를 사용할 수 있습니다
셧다운		클록 또는 BOR이 없습니다. 코어 레귤레이션 꺼짐. RTC 도메인은 여전히 활성 상태일 수 있습니다. 종료 트리거 재설정.	셧다운	클록, BOR 또는 RTC가 없습니다. 코어 레귤레이션 꺼짐. PD1 및 PD0 비활성화. 종료가 재설정 수준 BOR을 트리거합니다.	

저전력 모드에서 MSPM0 기능

표 3-9에서 볼 수 있듯이 MSPM0 주변 장치 또는 주변 장치 모드는 더 낮은 전력 운영 모드에서 가용성 또는 작동 속도를 제한할 수 있습니다. 자세한 사항은 MSPM0 장치별 데이터시트에 있는 "작동 모드별 지원 기능" 표를 참조하십시오. 예를 들어:

MSPM0G350x 혼합 신호 마이크로컨트롤러 데이터 시트

MSPM0L134x, MSPM0L130x 혼합 신호 마이크로컨트롤러 데이터 시트

MSPM0 장치의 추가 기능은 일부 주변 장치가 비동기식 고속 클록 요청을 수행할 수 있다는 것입니다. 따라서 MSPM0 장치는 주변 장치가 비활성 상태일 때 더 낮은 전력 모드로 구동하면서도 주변 장치가 트리거 또는 활성화되도록 할 수 있습니다. 비동기 고속 클록 요청이 있을 경우 MSPM0 장치는 내부 오실레이터를 빠르게 가속시키거나 일시적으로 더 높은 작동 모드로 전환하여 임박한 작업을 처리할 수 있습니다. 따라서 최저 전력 모드에서 절전 상태를 유지하면서 타이머, 콤퍼레이터, GPIO 및 RTC에서 CPU를 빠르게 깨울 수 있으며, SPI, UART 및 I2C를 수신하거나 DMA 전송 및 ADC 변환을 트리거할 수 있습니다. 비동기 클록 요청 구현에 대한 자세한 내용과 주변 장치 지원 및 목적은 MSPM0 TRM에서 해당 장을 참조하십시오.

MSPM0 G 시리즈 80MHz 마이크로컨트롤러 기술 레퍼런스 매뉴얼

MSPM0 L 시리즈 32MHz 마이크로컨트롤러 기술 레퍼런스 매뉴얼

저전력 모드 진입

STM32G0 장치와 마찬가지로 MSPM0 장치는 이벤트 대기, `__WFE();`, 또는 인터럽트 대기, `__WFI();`, 명령어를 실행할 때 저전력 모드로 들어갑니다. 저전력 모드는 현재 전원 정책 설정에 따라 결정됩니다. 장치 전원 정책은 드라이버 라이브러리 기능에 의해 설정됩니다. 다음 함수 호출은 해당 전원 정책을 Standby 0으로 설정합니다.

```
DL_SYSCTL_setPowerPolicySTANDBY0();
```

STANDBY0는 원하는 작동 모드로 대체할 수 있습니다. 전력 정책에 적용되는 driverlib API 전체 목록은 MSPM0 SDK DriverLib API 가이드를 참조하세요. 또한 다른 작동 모드로 들어가는 방법을 보여 주는 다음 코드 예제도 참조하십시오. 모든 MSPM0 장치에 대해 유사한 예제가 제공됩니다.

저전력 모드 코드 예제

SDK 설치 섹션으로 이동해 예제 > nortos > LP 이름 > driverlib으로 들어가 저전력 모드 코드 예제를 찾아보세요

3.6 인터럽트 및 이벤트 비교

인터럽트 및 예외

MSPM0와 STM32G0는 장치의 사용 가능한 주변 장치에 따라 인터럽트와 예외 벡터를 모두 레지스터하고 매핑합니다. 표 3-10에 보시면 각 장치 제품군별로 인터럽트 벡터에 대한 요약과 비교 정보가 나와 있습니다. 인터럽트 또는 예외에 대한 우선 순위 값이 낮을수록 우선 순위 값이 높은 인터럽트에 비해 높은 우선 순위가 부여됩니다. 이러한 벡터의 경우 우선 순위는 사용자가 선택할 수 있고, 다른 벡터의 경우 고정되어 있습니다.

MSPM0와 STM32G0에서는 NMI, 리셋, 하드 폴트 처리기 등의 예외 사항에 음의 우선 순위 값이 부여되어 항상 주변 인터럽트보다 우선 순위가 가장 높음을 나타냅니다. 인터럽트 우선 순위를 선택할 수 있는 주변 장치의 경우 양쪽 장치 제품군에서 최대 4개의 프로그래머블 우선 순위 수준을 사용할 수 있습니다.

표 3-10. 인터럽트 비교

NVIC 번호	STM32G0		MSPM0x	
	인터럽트/예외	우선 순위	인터럽트/예외	우선 순위
-	리셋	고정: -3	리셋	고정: -3
-	NMI 처리기	고정: -2	NMI 처리기	고정: -2
-	하드 폴트 처리기	고정: -1	하드 폴트 처리기	고정: -1
-	SVCall 처리기	선택 가능	SVCall 처리기	선택 가능
-	PendSV	선택 가능	PendSV	선택 가능
-	SysTick	선택 가능	SysTick	선택 가능
0	윈도우 워치독 인터럽트	선택 가능	INT_GROUP0: WWDT0, DEBUGSS, FLASHCTL, WUC FSUBx 및 SYSCTL	선택 가능
1	전원 전압 감지기 인터럽트	선택 가능	INT_GROUP1: GPIO0 및 COMP0	선택 가능
2	RTC 및 타임스탬프	선택 가능	타이머 G1(TIMG1)	선택 가능
3	플래시 전역 인터럽트	선택 가능	UART3(1)	선택 가능
4	RCC 전역 인터럽트	선택 가능	ADC0	선택 가능
5	EXTI0 및 EXTI1 인터럽트	선택 가능	ADC1(1)	선택 가능
6	EXTI2 및 EXTI3 인터럽트	선택 가능	CANFD0(1)	선택 가능
7	EXTI4-EXTI15 인터럽트	선택 가능	DAC0(1)	선택 가능
8	UCPD1/UCPD2/USB	선택 가능	예약됨	선택 가능
9	DMA1 채널 1	선택 가능	SPI0	선택 가능
10	DMA1 채널 2 및 3	선택 가능	SPI1(1)	선택 가능
11	DMA1 채널 4-6 및 DMA2 채널 1-5	선택 가능	예약됨	선택 가능
12	ADC 및 콤퍼레이터	선택 가능	예약됨	선택 가능
13	타이머 1(TIM1), 브레이크, 업데이트, 트리거 및 정류	선택 가능	UART1	선택 가능
14	TIM1 캡처 비교	선택 가능	UART2(1)	선택 가능
15	TIM2 전역 인터럽트	선택 가능	UART0	선택 가능
16	TIM3 및 TIM4 전역 인터럽트	선택 가능	TIMG0	선택 가능
17	TIM6, LPTIM1 및 DAC 인터럽트	선택 가능	TIMG10(1)	선택 가능
18	TIM6 및 LPTIM2 전역 인터럽트	선택 가능	TIMA0(1)	선택 가능
19	TIM14 전역 인터럽트	선택 가능	TIMA1	선택 가능
20	TIM15 전역 인터럽트	선택 가능	TIMA2(2)	선택 가능

표 3-10. 인터럽트 비교 (continued)

NVIC 번호	STM32G0		MSPM0x	
	인터럽트/예외	우선 순위	인터럽트/예외	우선 순위
21	TIM16 및 FDCAN0 전역 인터럽트	선택 가능	TIMH0(1)	선택 가능
22	TIM17 및 FDCAN1 전역 인터럽트	선택 가능	예약됨	선택 가능
23	I2C1 전역 인터럽트	선택 가능	예약됨	선택 가능
24	I2C2 및 I2C3 전역 인터럽트	선택 가능	I2C0	선택 가능
25	SPI1 전역 인터럽트	선택 가능	I2C1	선택 가능
26	SPI2 및 SPI3 전역 인터럽트	선택 가능	예약됨	선택 가능
27	USART1 전역 인터럽트	선택 가능	예약됨	선택 가능
28	USART2 및 LPUART2 전역 인터럽트	선택 가능	AES(1)	선택 가능
29	USART 3-6 및 LPUART1 전역 인터럽트	선택 가능	예약됨	선택 가능
30	CEC 전역 인터럽트	선택 가능	RTC(1)	선택 가능
31	AES 및 RNG 전역 인터럽트	선택 가능	DMA	선택 가능

(1) MSPM0G 장치 제품군에서만 사용 가능합니다.
(2) MSPM0L 장치 제품군에서 TIMG4

이벤트 처리기 및 EXTI(Extended Interrupt and Event Controller)

MSPM0 장치에는 NVIC 개념을 확장하여 주변 장치의 디지털 이벤트를 인터럽트로 CPU, DMA, 또는 다른 주변 장치로 전송하여 하드웨어 동작을 트리거할 수 있게 해 주는 전용 이벤트 관리자 주변 장치가 포함되어 있습니다. 또한 이벤트 관리자는 전원 관리 및 클록 장치(PMCU)와 핸드셰이크를 수행하여 트리거된 이벤트 작업이 진행되는 데 필요한 클록 및 전원 도메인이 있는지 확인할 수 있습니다.

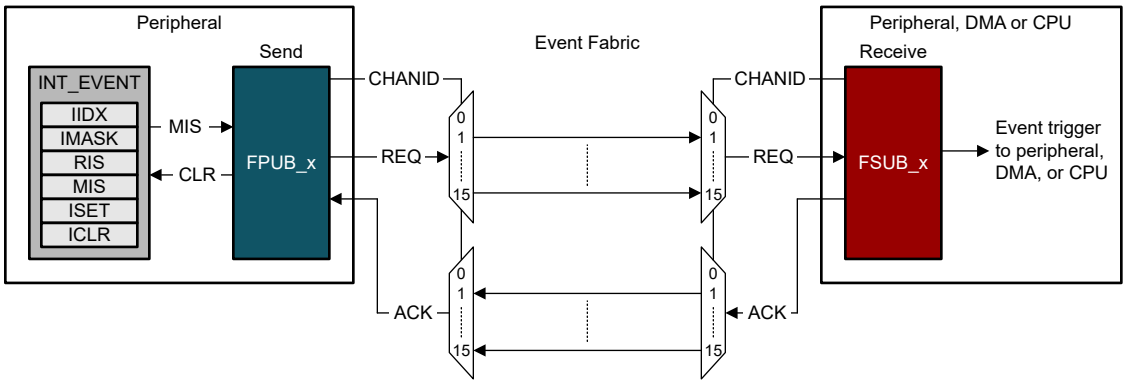

그림 3-2. 일반 이벤트 경로

MSPM0 이벤트 관리자에서 이벤트를 생성하는 주변 장치를 게시자라고 하며, 게시자를 기반으로 작동하는 주변 장치, DMA 또는 CPU를 가입자라고 합니다. 사용 가능한 게시자와 가입자의 잠재적인 조합은 매우 유연하기 때문에, 소프트웨어를 마이그레이션할 때 이전에 인터럽트 벡터와 CPU에 의해 처리되었던 기능을 대체해 CPU를 완전히 우회할 수 있습니다. 예를 들어, I^2C-UART 브리지는 이전에 I^2C STOP 수신 시 ISR을 사용하여 플래그를 설정하거나 UART TX 버퍼를 직접 로드할 때 UART 전송을 트리거했을 수 있습니다. MSPM0 이벤트 처리기를 사용하면 I^2C 트랜잭션 완료 이벤트가 DMA를 트리거하여 UART TX 버퍼를 직접 로드하므로 CPU의 작업이 필요 없습니다.

MSPM0G에서 이벤트 처리기 사용에 대한 자세한 내용은 MSPM0G 기술 참조 설명서 또는 MSPM0L 기술 참조 설명서의 이벤트 섹션을 참조하십시오.

MSPM0 이벤트 처리기와 혼동하지 않도록 STM32G0 제품군은 EXTI(Extended Interrupt and Event Controller)를 구현하며, 이를 이용해 iOS 또는 주변 장치의 구성 가능한 이벤트를 통해 중단 모드에서 시스템을 다시 켤 수 있습니다. STM32G0 EXTI의 활성화 기능은 IO 활성화 기능(MSPM0 기술 참조 매뉴얼의 IOMUX 섹션 참조) 및 GPIO FastWake(MSPM0 기술 참조 매뉴얼의 GPIO 섹션 참조)를 사용하면 MSPM0에서 가장 그대로 잘 재현할 수 있습니다. 활성화가 단일 동작을 위한 것인 경우, 이벤트 처리기 주변 장치는 주변 장치 작동 발생에 필요한 PMCU 리소스를 요청할 수 있으며, 이후 해당 저전력 모드로 돌아갑니다.

3.7 디버그 및 프로그래밍 비교

Arm SWD 2선 JTAG 포트는 MSPM0와 STM32G0 장치에서 모두 주요 디버그 및 프로그래밍 인터페이스입니다. 이 인터페이스는 보통 애플리케이션 개발 및 프로덕션 프로그래밍 중에 사용됩니다. 표 3-11에서는 이 2개의 장치 제품군 기능을 비교합니다. MSPM0 디버그 인터페이스에 관한 자세한 정보는 *MSPM0 MCU의 사이버 보안 인에이블러* 애플리케이션 노트를 참조하세요.

표 3-11. Arm SWD JTAG 기능 비교

	STM32G0	MSPM0
디버그 포트	Arm SWD 포트(2선)	Arm SWD 포트(2선)
BPU(Break Point Unit)	하드웨어 중단점 4개	하드웨어 중단점 4개
DWT(Data Watch Unit)	감시점 2개	감시점 2개
MTB(Micro-Trace Buffer)	아니요	4개의 추적 패킷을 이용한 MTB 지원(1)
저전력 디버그 지원	예	예
EnergyTrace 지원	아니요	EnergyTrace+ 지원(전원 프로파일링을 통한 CPU 상태)
디버그 중 주변 장치 실행 지원	예	예
디버그 인터페이스 잠금	디버그 읽기 액세스를 일시적으로 차단할 수 있습니다	디버그 기능을 영구적으로 비활성하거나 암호로 잠글 수 있습니다

(1) MSPM0Gxxxx 장치만 해당

BSL(Bootstrap Loader) 프로그래밍 옵션

BSL(Bootstrap Loader) 프로그래밍 인터페이스는 Arm SWD를 대체하는 프로그래밍 인터페이스입니다. 이 인터페이스는 프로그래밍 기능만을 제공하며, 보통 표준 임베디드 통신 인터페이스를 통해 활용합니다. 이를 이용하면 시스템 또는 외부 포트에서 다른 임베디드 장치에 대한 기존 연결을 통해 펌웨어를 업데이트할 수 있습니다. 프로그래밍 업데이트가 이 인터페이스의 주요 목적이지만 초기 프로덕션 프로그래밍 용도로도 활용할 수 있습니다. 표 3-12에는 MSPM0와 STM32G0 장치 제품군 간의 여러 다른 옵션과 기능을 비교한 정보가 나와 있습니다.

표 3-12. BSL 기능 비교

BSL 기능	STM32G0	MSPM0
빈 장치에서 시작된 BSL	예	예
프로그래밍 인터페이스 자동 감지	예	예
보안	메모리 보안 및 액세스 제한 옵션	보안 부팅 옵션, CRC 보호
사용자 지정 가능	아니요	예, 구성 가능 호출 핀과 플러그인 기능
메소드 호출	RESET, SW 입력 시 핀 최대 2개와 장치 등록 설정 사용 패턴(1)	BOOTRST, SW 입력 시 핀 1개 High
지원 인터페이스		
UART	예	예
I2C	예	예
SPI	예(2)	사용자 정의 플러그인 필요
CAN	예(2)	계획된 플러그인(2)
USB	예(2)	현재 USB 기능을 갖춘 MSPM0 장치가 없습니다.

(1) 패턴 옵션 이용 가능 여부는 장치에 따라 다릅니다.
(2) 일부 장치에서만 가능

4 디지털 주변 장치 비교

4.1 범용 I/O(GPIO, IOMUX)

MSPM0 GPIO 기능에는 사실상 STM32G0 GPIO가 제공하는 모든 기능이 포함됩니다. STM32G0는 장치 핀 관리를 맡고 있는 모든 기능을 지칭하기 위해 GPIO라는 용어를 사용합니다. 하지만 MSPM0는 다음과 같이 약간 다른 명명법을 사용합니다.

- MSPM0 GPIO는 읽기와 쓰기 IO, 인터럽트 생성 등이 가능한 하드웨어를 가리킵니다.
- MSPM0 IOMUX는 여러 다른 내부 디지털 주변 장치를 핀에 연결하는 일을 맡고 있는 하드웨어를 가리킵니다. IOMUX는 예를 들어 GPIO 같은 다양한 디지털 주변 장치를 지원합니다.

MSPM0 GPIO와 IOMUX는 함께 STM32G0 GPIO와 같은 기능을 담당합니다. 또한, MSPM0는 DMA 연결, 제어 가능한 입력 필터링 및 이벤트 기능 등 STM32G0 장치에서 제공되지 않는 기능도 제공합니다.

표 4-1. GPIO 기능 비교

주요 기능	STM32G0	MSPM0G 및 MSPM0L
출력 모드	푸시-풀 풀업 또는 풀다운이 있는 오픈 드레인	상응
GPIO 속도 선택	각 I/O에 대한 속도 선택	유사 MSPM0는 모든 IO 핀에서 표준 IO(SDIO)를 제공합니다. SDIO는 STM GPIO 속도=01과 같거나 더 빠릅니다. 일부 핀의 경우 MSPM0 HSIO(고속 IO)가 제공됩니다. HSIO는 STM GPIO 속도=10과 같습니다.
고출력 드라이브 GPIO	약 20mA	상응, 고출력 드라이브 IO(HDIO)라고 함
입력 모드	부동 풀업 또는 풀다운 아날로그	상응
원자 비트 설정 및 재설정	예	상응
GPIO 잠금	레지스터 잠금 메커니즘	*MSPM0 상응 없음*
대체 기능	선택 레지스터	상응 MSPM0은 IOMUX 사용
빠른 토글 전환	클록 2회마다 변경 가능	상응, MSPM0는 매 클록 주기마다 핀을 토글할 수 있습니다
활성화	GPIO 핀 상태 변경	상응
DMA가 GPIO를 제어	아니요	*MSPM0에서만 사용 가능*
1, 3 또는 8 ULPCLK 기간 미만의 사소한 문제를 거부하기 위해 사용자가 제어하는 입력 필터링	아니요	*MSPM0에서만 사용 가능*
사용자 제어 가능 입력 이력 현상	아니요	*MSPM0에서만 사용 가능*

GPIO 코드 예제

GPIO 코드 예제에 대한 정보는 MSPM0 SDK 예제 가이드에서 확인할 수 있습니다.

4.2 UART(범용 비동기 리시버 트랜스미터)

STM32G0와 MSPM0는 모두 비동기(클록 없음) 통신을 수행하는 주변 장치를 제공합니다. 이러한 UART 주변 장치에는 표준 기능을 갖춘 것과 고급 기능을 갖춘 것 등 두 가지가 있습니다. 표 4-2에 보면 이름이 어떻게 다른지 알 수 있습니다.

표 4-2. STM32G0와 MSPM0 간의 UART 이름 지정 차이

	STM32G0 이름 지정	MSPM0 이름 지정
표준 기능	기본	메인
고급 기능	완전	확장

표 4-3. UART 고급 기능 세트 비교

주요 기능	STM32G0 USART 전체 기능 세트	MSPM0L 및 MSPM0G UART 확장 기능 세트
하드웨어 흐름 제어	예	예
DMA를 이용한 연속 통신	예	예
다중 프로세서	예	예
동기 모드	예	아니요
스마트 카드 모드(ISO7816)	예	예
단일 회선 하프 듀플렉스 통신	예	예(1)
IrDA 하드웨어 지원	예	예
LIN 하드웨어 지원	예	예
DALI 하드웨어 지원	아니요	예
맨체스터 코드 하드웨어 지원	아니요	예
저전력 모드에서 깨우기	예	예
자동 보드율 감지	예	아니요
드라이버 활성화	예	예
데이터 길이	7, 8, 9	5, 6, 7, 8
Tx/Rx FIFO 깊이	8	4

(1) 전송과 수신 간의 주변 장치 재구성 필요

표 4-4. UART 표준 기능 세트 비교

주요 기능	STM32G0 USART 기본 기능 세트	MSPM0 UART 주요 기능 세트
하드웨어 흐름 제어	예	예
DMA를 이용한 연속 통신	예	예
다중 프로세서	예	예
동기 모드	예	아니요
단일 회선 하프 듀플렉스 통신	예	예(1)
저전력 모드에서 깨우기	아니요	예
드라이버 활성화	예	예
데이터 길이	7, 8, 9	5, 6, 7, 8
Tx/Rx FIFO 깊이	없음	4

(1) 전송과 수신 간의 주변 장치 재구성 필요

UART 코드 예제

UART 코드 예제에 관한 정보는 MSPM0 SDK 예제 가이드에서 확인할 수 있습니다.

4.3 SPI(직렬 주변기기 인터페이스)

MSPM0와 STM32G0는 모두 SPI(직렬 주변 장치 인터페이스)를 지원합니다. 전체적으로, MSPM0와 STM32G0 SPI 지원은 비슷하며, 차이점은 표 4-5에 나와 있습니다.

표 4-5. SPI 기능 비교

주요 기능	STM32G0x	MSPM0L 및 MSPM0G
컨트롤러 또는 주변 장치 작동	예	예
데이터 비트 폭(컨트롤러 모드)	4~16비트	4~16비트
데이터 비트 폭(주변 장치 모드)	4~16비트	7~16비트
최대 속도	32 MHz	MSPM0L: 16 MHz
		MSPM0G: 32 MHz
풀 듀플렉스 전송	예	예
하프 듀플렉스 전송(양방향 데이터 회선)	예	아니요
심플렉스 전송(단방향 데이터 회선)	예	예
다중 컨트롤러 기능	예	아니요
하드웨어 칩 선택 관리	예(주변 장치 1개)	예(주변 장치 4개)
프로그래머블 클록 극성 및 위상	예	예
데이터 순서 프로그래밍 가능, MSB 우선 또는 LSB 우선 시프팅	예	예
SPI 형식 지원	모토로라, TI	모토로라, TI, MICROWIRE
하드웨어 CRC	예	아니오, MSPM0은 SPI 패리티 모드 제공
TX FIFO 깊이	데이터 크기에 따라 다름	4
RX FIFO 깊이	데이터 크기에 따라 다름	4

SPI 코드 예제

SPI 코드 예제에 관한 정보는 MSPM0 SDK 예제 가이드에서 확인할 수 있습니다.

4.4 I²C

MSPM0와 STM32G0는 모두 I²C를 지원합니다. 전체적인 MSPM0 및 STM32G0 I²C 지원은 비슷하고, 눈에 띄는 차이점은 아래 표에 나와 있습니다.

표 4-6. I²C 기능 비교

주요 기능	STM32G0	MSPM0L 및 MSPM0G
컨트롤러 및 대상 모드	예	예
다중 컨트롤러 기능	예	예
표준 모드(최대 100kHz)	예	예
고속 모드(최대 400kHz)	예	예
고속 모드 플러스(최대 1MHz)	예	예
주소 지정 모드	7, 10비트	7 bit
주변 장치 주소	주소 2개와 구성 가능한 마스크 1개	주소 2개
전체 호출	예	예
프로그래머 가능한 설정 및 유지 시간	예	아니요
이벤트 관리	예	예
클록 스트레칭	예	예
소프트웨어 리셋	예	예
FIFO/버퍼	1바이트	TX: 8바이트
		RX: 8바이트
DMA	예	예
프로그래밍 가능한 아날로그 및 디지털 노이즈 필터	예	예

I²C 코드 예제

I2C 코드 예제에 관한 정보는 MSPM0 SDK 예제 가이드에서 확인할 수 있습니다.

4.5 타이머(TIMGx, TIMAx)

STM32G0와 MSPM0는 모두 다양한 타이머를 제공합니다. MSPM0는 저전력 모니터링부터 고급 모터 제어까지 유스 케이스를 지원하는 다양한 기능을 갖춘 타이머를 제공합니다.

표 4-7. 타이머 이름 지정

STM32G0		MSPM0	
타이머 이름	축약형 이름	타이머 이름	축약형 이름
고급 제어	TIM1	고급 제어	TIMA0
범용	TIM2-4, TIM14/-17	범용	TIMG0-11
		고해상도	TIMG12
기본	TIM6/7		
저전력	LPTIM		

표 4-8. 타이머 기능 비교

주요 기능	STM32G0 타이머	MSPM0G 타이머	MSPM0L 타이머
분해능	16비트, 32비트	16비트, 32비트	16비트
PWM	예	예	예
캡처	예	예	예
비교	예	예	예
원샷	예	예	예
업다운 카운트 기능	예	예	예
전원 모드	예	예	예
QEI 지원	예	예	아니요
프로그래밍 가능한 프리스케일러	예	예	예
섀도우 레지스터 모드	예	예	예
이벤트/인터럽트	예	예	예
오류 이벤트 메커니즘	예	예	아니요
자동 재로드 기능	예	예	예

표 4-9. 타이머 모듈 교체

STM32G0 타이머	MSPM0 상응	추론
TIM1	TIMA, TIMG8-12	고급 제어, 양쪽 16비트 분해능, QEI 지원
TIM2	TIMG12	32비트 분해능
TIM3/4	TIMG0-7	범용, 16비트 분해능
TIM6/7	모두	기본 타이머
TIM14	모두	TIM3/4와 동일한 기능
TIM15/16/17	모두	범용
LPTIM	PD0의 모든 타이머	LPTIM은 LFCLK 소싱, PD0 – MSPM0에서 저전력 모드

표 4-10. 타이머 유스 케이스 비교

주요 기능	STM32G0 타이머	MSPM0 타이머
PWM	TIM1-4에는 에지와 센터 정렬 옵션이 있으며, TIM6-7에는 PWM 기능이 없습니다. TIM15-17에지 정렬 옵션만.	모든 타이머에는 에지 정렬 또는 센터 정렬 옵션이 있습니다
캡처	큰 차이는 없습니다	큰 차이는 없습니다
비교	큰 차이는 없습니다	큰 차이는 없습니다
원샷	큰 차이는 없습니다	큰 차이는 없습니다
프리스케일러	16비트 프리스케일러, LPTIM(3비트 프리스케일러) 외	8비트 프리스케일러
동기화	TIM1-4, TIM15	모든 타이머에는 이 기능이 있습니다

타이머 코드 예제

타이머 코드 예제에 관한 정보는 MSPM0 SDK 예제 가이드에서 확인할 수 있습니다.

4.6 WWDT(윈도우 워치독 타이머)

STM32G0와 MSPM0는 모두 WWDT(윈도우 워치독 타이머)를 제공합니다. WWDT(윈도우 워치독 타이머)는 지정된 시간 이내에 애플리케이션이 체크인에 실패하면 시스템 재설정을 개시합니다.

표 4-11. WWDT 이름 지정

키	STM32G0	MSPM0
이름	독립형 워치독 타이머, 윈도우형 워치독 타이머	WWDT(윈도우형 워치독 타이머)
축약형 이름(동일 순서)	IWDG, WWDG	WWDT

표 4-12. WDT 기능 비교

주요 기능	STM32G0	MSPM0G	MSPM0L
윈도우 모드	예	예	예
간격 타이머 모드	예	예	예
LFCLK 소스	예	예	예
인터럽트	예	예	예
카운터 해상도	7 bit	25 bit	25 bit
클록 분할기	WWDG 아니오, IWDG 예	예	예

WWDT 코드 예제

WWDT 코드 예제에 관한 정보는 MSPM0 SDK 예제 가이드에서 확인할 수 있습니다.

4.7 실시간 클록(RTC)

STM32G0 및 MSPM0[1] 양쪽 모두 RTC(실시간 클록)를 제공합니다. RTC(실시간 클록) 모듈은 초, 분, 시간, 요일, 날짜, 연도 등의 카운터를 이용해 선택할 수 있는 2진 또는 2진 코드 10진수 형식으로 애플리케이션에 대한 시간 추적 기능을 제공합니다.

표 4-13. RTC 기능 비교

주요 기능	STM32G0	MSPM0G
전원 모드	예	예
2진 코드 형식	예	예
윤년 보정	예	예
사용자 지정 가능한 알람 개수	2	2
내부 및 외부 크리스털	예	예
크리스털 오프셋 보정	예	예
프리스케일러 블록	예	예
인터럽트	예	예

RTC 코드 예제

RTC 코드 예제에 관한 정보는 MSPM0 SDK 예제 가이드에서 확인할 수 있습니다.

[1] RTC를 지원하는 것은 MSPM0G 장치뿐입니다.

5 아날로그 주변 장치 비교

5.1 ADC(아날로그-디지털 컨버터)

STM32G0와 MSPM0은 모두 아날로그 신호를 디지털 신호로 변환하기 위한 ADC 주변 장치를 제공합니다. 두 장치 제품군에 모두 12비트 ADC가 포함되어 있습니다. 다음은 ADC의 서로 다른 기능과 모드를 비교한 표입니다.

표 5-1. 기능 세트 비교

주요 기능	STM32G0	MSPM0G	MSPM0L
해상도(bit)	12	12	12
변환 속도(Msps)	2.5	4	1.4
오버샘플링(비트)	16	14	해당 없음
하드웨어 오버샘플링	256배	128배	해당 없음
FIFO	아니요	예	예
ADC 레퍼런스(V)	내부: 2.048, 2.5	내부: 1.4, 2.5, VDD	내부: 1.4, 2.5, VDD
	V_{DD} < 2일 때 외부: $V_{REF} = V_{DD}$	외부: $1.4 \le V_{REF} \le V_{DD}$	외부: $1.4 \le V_{REF} \le V_{DD}$
	$V_{DD} \ge 2$일 때 외부: $2 \le V_{REF} \le V_{DD}$		
작동 전력 모드	실행, 절전	실행, 절전, 중지, 대기(1)	실행, 절전, 중지, 대기(1)
자동 전원 차단	예	예	예
외부 입력 채널(2)	최대 16	최대 16	최대 16
내부 입력 채널	온도 센서, VREF, VBAT	온도 센서, 공급 모니터링, 아날로그 신호 체인	온도 센서, 공급 모니터링, 아날로그 신호 체인
DMA 지원	예	예	예
ADC 윈도우 콤퍼레이터 유닛	아니요	예	예
동시 샘플링	아니요	예	아니요
ADC 개수(3)	최대 1	최대 2	최대 1

(1) ADC는 대기 모드에서 작동될 수 있으며, 그렇게 되면 작동 모드가 바뀝니다.
(2) 외부 입력 채널 수는 장치별로 다릅니다.
(3) ADC 개수는 장치별로 다릅니다.

표 5-2. 변환 모드

STM32G0	MSPM0	기타 의견
단일 변환 모드	단일 채널 단일 변환	ADC는 단일 채널을 한 번 샘플링하고 변환합니다
채널 시퀀스를 스캔합니다	채널 시퀀스 변환	ADC는 채널 시퀀스를 샘플링하고 한 번 변환합니다.
연속 변환 모드	단일 채널 변환을 반복합니다	단일 채널을 반복하여 한 채널을 연속적으로 샘플링하고 변환합니다
	채널 시퀀스 변환 반복	채널 시퀀스를 샘플링하고 변환한 후 동일한 시퀀스를 반복합니다
불연속 모드	채널 시퀀스 변환 반복	불연속 채널 세트를 샘플링하고 변환합니다. 이 작업은 MEMCTRLx를 다른 채널에 매핑하는 방법으로 MSPM0에서 수행할 수 있습니다.

ADC 코드 예제

ADC 코드 예제에 관한 정보는 MSPM0 SDK 예제 가이드에서 확인할 수 있습니다.

5.2 콤퍼레이터(COMP)

STM32G0과 MSPM0 부품 제품군은 모두 일부 장치에서 선택 사항 주변 장치로 일체형 콤퍼레이터를 제공합니다. 양쪽 장치 제품군에서 모두 COMPx로 표시되는데, 여기서 마지막 문자 'x'는 고려 대상인 구체적인 콤퍼레이터 모듈을 나타냅니다. STM32G0 제품군에서는 여기에 1~3 사이 숫자가 사용되고, MSPM0 제품군에서는 0~2 사이 숫자가 사용됩니다. 콤퍼레이터 모듈은 모두 콤퍼레이터가 2개 이상인 장치에서 윈도우 콤퍼레이터 기능을 제공하고, 다양한 내부 및 외부 소스에서 입력을 받을 수 있고, 전원 모드에서 변경을 트리거하거나 PWM 신호를 중단/제어하는 데 사용할 수 있습니다. 표 5-3에 보시면 MSPM0 및 STM32G0 콤퍼레이터 모듈 간 기능별 비교 요약 정보가 나와 있습니다.

표 5-3. COMP 기능 세트 비교

주요 기능	SMT32G0	MSPM0G	MSPM0L
사용 가능한 콤퍼레이터	최대 3	최대 3	최대 1
출력 라우팅	멀티플렉스 I/O 핀	멀티플렉스 I/O 핀	멀티플렉스 I/O 핀
	EXTI 인터럽트	인터럽트/이벤트 인터페이스	인터럽트/이벤트 인터페이스
비반전 입력 소스	멀티플렉스 I/O 핀	멀티플렉스 I/O 핀	멀티플렉스 I/O 핀
		DAC12 출력(1)	DAC8 출력
		DAC8 출력	OPA1 출력(2)
		내부 V_{REF}: 1.4V 및 2.5V	
		OPA1 출력(2)	
반전 입력 소스	멀티플렉스 I/O 핀	멀티플렉스 I/O 핀	멀티플렉스 I/O 핀
	DAC 채널 1과 2	내부 온도 센서	내부 온도 센서
	내부 V_{REF}: 2.048V 및 2.5V	내부 V_{REF}: 1.4V 및 2.5V	DAC8 출력
	버퍼된 V_{REF} 분배기, 예: ¼V_{REF}, ½V_{REF} 및 ¾V_{REF}	DAC8 출력	OPA0(3) 출력
		OPA0 출력 (3)	
프로그래머블 이력	없음, 10mV, 20mV, 30mV	없음, 10mV, 20mV, 30mV	없음, 10mV, 20mV, 30mV
		DAC8을 사용하는 0V~V_{REF}/V_{DD}의 다른 값	DAC8을 사용하는 0V~V_{DD}의 다른 값
레지스터 잠금	예, 전체 COMP 레지스터(장치 재설정 시 비활성화됨)	예, 일부 COMP 레지스터(쓰기에 키 필요)	예, 일부 COMP 레지스터(쓰기에 키 필요)
윈도우 콤퍼레이터 구성	예	예	아니오(단일 COMP)
입력 단락 모드	아니요	예	예
작동 모드	고속, 중간 속도	고속, 저전력	고속, 저전력
빠른 PWM 차단	예	예(TIMA 오류 처리기를 통해)	아니요
출력 필터링	블랭킹 필터	블랭킹 필터	블랭킹 필터
		조정식 아날로그 필터	조정식 아날로그 필터
출력 극성 제어	예	예	예
인터럽트	상승 에지	상승 에지	상승 에지
	하강 에지	하강 에지	하강 에지
	양쪽 에지	출력 준비 완료	출력 준비 완료
교환 입력 모드	아니요	예	예

(1) DAC12 주변 장치가 있는 장치에서만
(2) OPA1 주변 장치가 있는 장치에서만
(3) OPA0 주변 장치가 있는 장치에서만

COMP 코드 예제

COMP 코드 예제에 관한 정보는 MSPM0 SDK 예제 가이드에서 확인할 수 있습니다.

5.3 DAC(디지털-아날로그 컨버터)

STM32G0 및 MSPM0 부품 제품군에는 다양한 애플리케이션 목적으로 디지털-아날로그 변환을 수행할 수 있는 12비트 DAC 주변 장치가 포함되어 있습니다. STM32G0 설명서에서는 이 주변 장치를 단순히 DAC라고 부릅니다. MSPM0 기술 레퍼런스 매뉴얼, MSPM0 시리즈 데이터 시트, MSPM0 SDK에서는 12비트 DAC 주변 장치를 DAC12라고 부릅니다. 이는 DAC12를 MSPM0 장치에 포함되어 있는 각 콤퍼레이터 주변 장치에서 사용할 수 있는 8비트 DAC와 구별하기 위한 것입니다. 추가적인 8비트 DAC는 이 문서의 콤퍼레이터 섹션에서 다룹니다. 이 DAC12 주변 장치는 MSPM0G 장치 제품군에서만 제공됩니다.

STM32G0 및 MSPM0G용 12비트 DAC 주변 장치의 기능은 표 5-4에 요약되어 있습니다.

표 5-4. DAC 기능 세트 비교

주요 기능	STM32G0	MSPM0
분해능	12비트(11.4~11.5 ENOB)	12비트(11 ENOB)
출력 속도	1MSPS	1MSPS
출력 채널	2 (1)	1 (2)
데이터 형식	8비트 오른쪽 정렬, 12비트 오른쪽 정렬, 12비트 왼쪽 정렬	8비트 오른쪽 정렬, 12비트 오른쪽 정렬, 2의 보수 또는 직선 이진
DMA 결합	예	예
출력 라우팅	외부 핀	외부 핀
	내부 주변 장치 연결: COMP IN-, ADC	내부 주변 장치 연결: OPA IN+, COMP IN+, ADC0
내부 레퍼런스 전압	예, 2.5V 또는 2.048V	예, 2.5V 또는 1.4V
외부 레퍼런스 전압	예	예
FIFO	아니요	예
출력 버퍼	예	예
구성 가능 출력 오프셋	예	예
자체 보정 모드	예	예
자동 파형 생성	노이즈파, 삼각파	아니요
샘플 앤 홀드 모드	예	아니요
트리거 소스	외부 핀, 내부 타이머 신호, DAC 홀드 클록, DMA 언더런	내부 전용 샘플 타임 생성기, DMA 인터럽트/이벤트, FIFO 임계값 인터럽트/이벤트, 하드웨어 트리거 2개(이벤트 패브릭에서 사용 가능)

(1) 일부 장치에서만 사용 가능.
(2) 듀얼 DAC 채널은 미래 MSPM0G 장치용도록 계획된 것입니다.

DAC12 코드 예제

DAC12 코드 예제에 관한 정보는 MSPM0 SDK 예제 가이드에서 확인할 수 있습니다.

5.4 OPA(연산 증폭기)

STM32G0 제품군은 OPA(Integrated Operational Amplifier) 주변 장치를 제공하지 않지만, STM32G0에서 MSPM0 제품군으로 마이그레이션할 때 MSPM0 내부 OPA를 사용하여 외부 개별 장치를 교체하거나 필요에 따라 내부 신호를 버퍼링할 수 있습니다. MSPOPA M0 모듈은 매우 유연해 감지 또는 제어 애플리케이션에서 다수의 개별 증폭기를 개별적으로, 또는 조합하여 대체할 수 있습니다. MSPM0 OPA 모듈의 주요 기능은 표 5-5에 소개되어 있고, 사용자가 생성할 수 있는 흔히 사용되는 OPA 구성 예제는 OPA 코드 예제에 나와 있습니다.

표 5-5. MSPM0 OPA 기능 세트

주요 기능	MSPM0 구현
입력 유형	레일 투 레일(활성화 또는 비활성화 가능)
게인 대역폭	1MHz(저전력 모드)
	6MHz(표준 모드)

표 5-5. MSPM0 OPA 기능 세트 (continued)

주요 기능	MSPM0 구현
증폭기 구성	범용 모드
	버퍼 모드
	PGA 모드(인버팅 또는 비인버팅)
	차동 증폭기 모드
	캐스케이드 증폭기 모드
입력/출력 라우팅	외부 핀 라우팅
	ADC 및 COMP 모듈에 대한 내부 연결
오류 감지	BCS(번아웃 전류 소스)
초퍼 안정화	표준(선택 가능 초핑 빈도)
	ADC 지원 초핑
	비활성화됨
레퍼런스 전압	내부 VREF(MSPM0G 장치만 해당)
	DAC12(MSPM0G 장치만 해당)
	DAC8(COMP 모듈이 있는 장치만 해당)

OPA 코드 예제

OPA 코드 예제에 관한 정보는 MSPM0 SDK 예제 가이드에서 확인할 수 있습니다.

5.5 VREF(전압 레퍼런스)

STM32G0x와 MSPM0에는 모두 내부 주변 장치에 레퍼런스 전압을 공급하고 외부 주변 장치에 출력하는 데 사용할 수 있는 내부 레퍼런스가 있습니다.

표 5-6. 기능 세트 비교

주요 기능	STM32G0	MSPM0G	MSPM0L
내부 레퍼런스(V)	2.048, 2.5	1.4, 2.5	1.4, 2.5
외부 레퍼런스(V)	V_{DD} < 2일 때 $V_{REF} = V_{DD}$	외부: 1.4 ≤ V_{REF} ≤ V_{DD}	외부: 1.4 ≤ V_{REF} ≤ V_{DD}
	V_{DD} ≥ 2일 때, 2 ≤ V_{REF} ≤ V_{DD}		
출력 내부 레퍼런스	예	예	예
내부적으로 ADC에 연결	예	예	예
내부적으로 ADC에 연결	예	예	아니요
내부적으로 COMP에 연결	아니요	예	아니요
내부적으로 OPA에 연결	해당 없음	예	아니요

표 5-7. 제어 비트 비교

STM32G0x VREFBUF 비트	MSPM0 상응
VREFBUF Bit3(VRR)	CTL1 Bit0(READY)
VREFBUF Bit2(VRS)	CTL0 Bit7(BUFCONFIG)
VREFBUF Bit1(HIZ)	해당 없음
VREFBUF Bit0(ENVR)	CTL0 Bit0(ENABLE)
	샘플 앤 홀드 모드: CTL0 Bit8(SHMODE)

MSPM0 VREF의 경우 전원 비트인 PWREN Bit0(ENABLE)을 활성화해야 합니다.

VREF 코드 예제

VREF를 사용하는 코드 예제는 MSPM0 SDK 예제 가이드에서 확인할 수 있습니다.

6 개정 내역

참고: 이전 개정판의 페이지 번호는 현재 버전의 페이지 번호와 다를 수 있습니다

날짜	개정	참고
2023년 3월	A	첫 번째 공개 릴리스

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