# AN0918.1: MCU Series 0 to EFM32GG1x/ TG11 Compatibility and Migration Guide

This porting guide is targeted at migrating an existing design from MCU Series 0 to MCU Series 1. Both hardware and software migration needs to be considered.

The core and peripherals of MCU Series 1 devices are based on the existing MCU Series 0 with better performance and lower current consumption.

This document will describe which aspects are enhanced in the peripherals common between MCU Series 0 and MCU Series 1. Details for all of the new peripherals of MCU Series 1 can be found in the reference manual, and it is recommended to review the available example code for assistance and recommendations.

All peripherals in the MCU Series 0 and MCU Series 1 devices are described in general terms. Not all modules are present in all devices, and the feature set for each device might vary. Such differences, including pinout, are covered in the device-specific data sheets.

**KEY POINTS**

- MCU Series 1 have commonalities and enhancements from MCU Series 0 peripherals.
- Software and hardware migration must both be considered when porting from an MCU Series 0 device to MCU Series 1 device.
- The MCU Series 1 devices are software compatible with the existing MCU Series 0 devices, so only minor changes are required for common peripherals.
- Refer to the example code for specific recommendations and assistance.

# 1. Device Compatibility

This application note supports multiple device families, and some functionality is different depending on the device.

MCU Series 0 consists of:

- EFM32 Gecko (EFM32G)
- EFM32 Giant Gecko (EFM32GG)
- EFM32 Wonder Gecko (EFM32WG)
- EFM32 Leopard Gecko (EFM32LG)
- EFM32 Tiny Gecko (EFM32TG)
- EFM32 Zero Gecko (EFM32ZG)
- EFM32 Happy Gecko (EFM32HG)

Wireless MCU Series 0 consists of:

- EZR32 Wonder Gecko (EZR32WG)
- EZR32 Leopard Gecko (EZR32LG)
- EZR32 Happy Gecko (EZR32HG)

MCU Series 1 consists of:

- EFM32 Giant Gecko 11 (EFM32GG11)
- EFM32 Tiny Gecko 11 (EFM32TG11)
- EFM32 Giant Gecko 12 (EFM32GG12)

## 2. Compatibility Overview

Four factors must be considered when porting a design from MCU Series 0 to MCU Series 1: pin compatibility, hardware compatibility, software compatibility, and peripheral compatibility.

### 2.1 Pins and Hardware

MCU Series 1 devices without DC-DC converter are footprint compatible with select MCU Series 0 devices.

More information on footprint and hardware compatibility between MCU Series 0 and MCU Series 1 can be found in 6. Hardware Migration.

### 2.2 Software and Peripherals

Software compatibility between MCU Series 0 is maintained using emlib and emdrv, which are software libraries built upon the CMSIS (Cortex Microcontroller Software Interface Standard) layer defined by ARM. These devices are not binary compatible, meaning code compiled for MCU Series 0 will not work after being downloaded to MCU Series 1. However, if the software is written using the emlib or emdrv modules, then the application code should not need to change in most cases when recompiling for the new MCU Series 1 target.

**Note:** There are some small exceptions to full software compatibility across MCU Series 0 and MCU Series 1. For example, wake-up pins and GPIO drive strength are implemented slightly differently on these parts, so the emlib functions have changed slightly to accommodate these differences. Wherever possible, these details have been abstracted away by the emlib and emdrv modules. See 5.1 Peripheral Support Library (emlib) and energyAware Drivers (emdrv) for more information on compatibility between MCU Series 0 and MCU Series 1. Consult the [**SDK Documentation**] under the [**Getting Started**] tab in Simplicity Studio for more information on the emlib and emdrv modules.

The emlib and emdrv modules provide abstraction layers that make peripheral initialization and usage simple and easy. Version 5.6.1.0 or above of the emlib and emdrv modules support the following peripherals across MCU Series 0 and MCU Series 1:

**Table 2.1. The emlib and emdrv Support for MCU Series 0 and MCU Series 1**

| Peripherals Supported by emlib | | | | | | | |
|---|---|---|---|---|---|---|---|
| ACMP | ADC | AES[1] | BURTC | CAN | CMU | CORE | CRYOTIMER |
| CRYPTO[1] | CSEN | DAC | DBG | DMA | EBI | EMU | GPCRC |
| GPIO | I2C | IDAC | INT | LCD | LDMA | LESENSE | LETIMER |
| LEUART | MPU | MSC | OPAMP | PCNT | PDM[2] | PRS | QSPI |
| RMU | RTC | RTCC | SYSTEM | TIMER | USART | VCMP | VDAC |
| WDOG | | | | | | | |

**Note:**

1. For AES and CRYPTO, use the mbedTLS library.
2. PDM is available only on EFM32GG12 devices. This peripheral is not currently supported in v5.6.1.0 but will be supported in higher versions.

| The emdrv Modules | | | | | | | |
|---|---|---|---|---|---|---|---|
| DMADRV | EZRADIODRV | GPIOINTERRUPT | NVM | NVM3 | RTCDRV | SLEEP | SPIDRV |
| TEMPDRV | UARTDRV | USTIMER | — | — | — | — | — |

Since the emlib and emdrv modules are common across MCU Series 0 and MCU Series 1, the look and feel of the software development experience is familiar. In other words, developers experienced with the MCU Series 0 products will already know how to construct software and use peripherals on the MCU Series 1 products. In addition, existing MCU Series 0 designs can be quickly ported to new products to take advantage of new capabilities available on the MCU Series 1 by utilizing the common code base between the families.

For systems that have software written without the use of emdrv, there are two methods to migrate a design from MCU Series 0 to MCU Series 1:

1. Elevate the code to the level of emdrv to take advantage of the hardware abstraction provided by this layer.
2. Use the information in this document to migrate the code to the peripherals featured on the MCU Series 1 products.

More information on software migration can be found in 5. Software Migration, and more information on the peripheral commonalities and differences can be found in 4. Peripherals Common Between MCU Series 0 and MCU Series 1.

# 3. System Overview

## 3.1 Core and Memory

This section compares the core and memory of MCU Series 0 with MCU Series 1.

**Table 3.1. Core and Memory**

| MCU Series 0 | EFM32GG11 | EFM32TG11 | EFM32GG12 |
|---|---|---|---|
| **Core** | | | |
| ARM Cortex M0+, M3 and M4 with FPU | ARM Cortex M4 with FPU | ARM Cortex M0+ | ARM Cortex M4 with FPU |
| **Debug Interface**[1] | | | |
| The 2-pin serial-wire debug (SWD) interface. | The 2-pin serial-wire debug (SWD) interface or a 4-pin Joint Test Action Group (JTAG) interface. | The 2-pin serial-wire debug (SWD) interface or a 4-pin Joint Test Action Group (JTAG) interface. | The 2-pin serial-wire debug (SWD) interface or a 4-pin Joint Test Action Group (JTAG) interface. |
| **DMA Controller (DMA)**[2] | | | |
| ARM µDMA Controller | Linked DMA Controller (LDMA) | Linked DMA Controller (LDMA) | Linked DMA Controller (LDMA) |
| **Flash Program Memory**[3] | | | |
| 4 - 1024 kB | 1024 - 2048 kB | 64 - 128 kB | 512 - 1024 kB |
| **RAM Memory**[4] | | | |
| 2 - 128 kB | 384 - 512 kB | 32 kB | 192 kB |

**Note:**

1. Debug interface (SWD, JTAG, and ETM) for each device might vary, such differences are covered in the device-specific data sheets.
2. LDMA is completely a new design, it is more flexible and higher performance.

   Number of DMA channels for each device might vary, such differences are covered in the device-specific data sheets. The dmadrv module can also be used to assist with family differences.
3. Flash program memory for each device might vary, such differences are covered in the device-specific data sheets.
4. RAM memory for each device might vary, such differences are covered in the device-specific data sheets.

### 3.2 Peripherals in MCU Series 0 only

This section describes the peripherals of MCU Series 0 that are not available in MCU Series 1.

Refer to 5.1 Peripheral Support Library (emlib) and energyAware Drivers (emdrv) and on how to migrate these MCU Series 0 peripherals to MCU Series 1.

**Table 3.2. Peripherals in MCU Series 0 only**

| MCU Series 0 |
| --- |
| **Energy Management** |
| Voltage Comparator (VCMP) |
| **Security** |
| Advanced Encryption Standard Accelerator (AES) |
| **Timers and Triggers** |
| Backup Real Time Counter (BURTC) |

### 3.3 New Peripherals in MCU Series 1

This section describes the new peripherals that are available in MCU Series 1.

**Table 3.3. New Peripherals in MCU Series 1**

| MCU Series 1 |
|---|
| **Analog Interfaces** |
| Analog Port (APORT) |
| Capacitive Sense Module (CSEN) |
| **Clock Management** |
| High Frequency RC Oscillator (HFRCO) with Digital Phase-Locked Loop (DPLL) |
| **Energy Management** |
| DC-DC Converter[1] |
| Voltage/Temp Monitor |
| **Security** |
| Crypto Accelerator (CRYPTO) |
| General Purpose Cyclic Redundancy Check (GPCRC) |
| Security Management Unit (SMU) |
| True Random Number Generator (TRNG) |
| **Serial Interfaces** |
| 10/100 Ethernet[2] |
| CAN |
| Quad-SPI[3] |
| SD/MMC/SDIO[4] |
| PDM[5] |
| **Timers and Triggers** |
| Ultra Low Energy Timer/Counter (CRYOTIMER) |
| Real Time Counter and Calendar (RTCC) |
| 32 bit General Purpose Timer (WTIMER) |

**Note:**
1. Some OPNs within the MCU Series 1 do not have the DC-DC converter.
2. 10/100 Ethernet is not available in EFM32TG11 and EFM32GG12.
3. Quad-SPI is not available in EFM32TG11.
4. SD/MMC/SDIO is not available in EFM32TG11
5. PDM is not available in EFM32TG11 and EFM32GG11

# 4. Peripherals Common Between MCU Series 0 and MCU Series 1

## 4.1 Overview

The table below summaries the maximum capabilities of common peripherals between MCU Series 0 and MCU Series 1.

**Table 4.1. Common Peripherals between MCU Series 0 and MCU Series 1**

| | MCU Series 0 | EFM32GG11 | EFM32TG11 | EFM32GG12 |
|---|---|---|---|---|
| DMA Channel | 12 | 24 | 8 | 12 |
| PRS Channel | 12 | 24 | 8 | 16 |
| I2C | 2 | 3 | 2 | 2 |
| LEUART | 2 | 2 | 1 | 2 |
| UART | 2 | 2 | 0 | 2 |
| USART | 3 | 6 | 4 | 5 |
| USB | Y | Y | N | Y |
| LESENSE | Y | Y | Y | Y |
| LETIMER | 1 | 2 | 1 | 2 |
| PCNT | 3 | 3 | 1 | 3 |
| RTC | 1 | 1 | 0 | 1 |
| TIMER | 4 | 7 | 2 | 4 |
| WDOG | 1 | 2 | 1 | 2 |
| ACMP | 2 | 4 | 2 | 3 |
| ADC | 1 | 2 | 1 | 2 |
| DAC/VDAC | 1 | 1 | 1 | 1 |
| IDAC | 1 | 1 | 0 | 1 |
| LCD | 8x36 | 8x36 | 8x32 | 8x36 |
| OPAMP | 3 | 4 | 4 | 4 |
| EBI | Y | Y | N | Y |
| GPIO | 93 (BGA120) | 144 (BGA192) | 67 (TQFP80/QFN80) | 95 (BGA120) |

## 4.2 Core and Memory

### 4.2.1 Debug (DBG)

The major changes are JTAG support and AAP lock.

**Table 4.2. DBG**

| MCU Series 0 | MCU Series 1 | Notes |
|---|---|---|
| **Enhancements** | | |
| Hardware Debug support through a 2-pin serial-wire debug (SWD) interface. | Hardware debug support through a 2-pin serial-wire debug (SWD) interface or a 4-pin Joint Test Action Group (JTAG) interface. | The debug pins can be enabled and disabled through GPIO_ROUTEPEN.<br>If enabling the JTAG pins, the part must be power cycled to enable a SWD debug session. |
| Debug Lock only. | Debug Lock (DLW in lockbits page) and AAP Lock (ALW in lockbits page). | The AAP_CMD register is locked by AAP Lock and this process is irreversible. The user can no longer access the AAP_CMD register to issue a mass erase to the FLASH in order to gain entry to the system via the debugger. |
| **New Features** | | |
| — | The system bus can be stalled by AAP_CTRL register. | Use the SYSBUSSTALL bit in AAP_CTRL register. |
| — | The CRCREQ command (AAP_CRCCMD register) initiates a CRC calculation on a given Flash Page. | The CRC is only available on the Main, User Data, and Lock Bit pages.<br>It is highly recommended that the system bus is stalled before any CRCREQ commands are issued. |
| **Limitations** | | |
| — | — | — |

### 4.2.2 Memory and Bus System

The major changes are the RAM segments, ECC protection, and single-cycle bit access for peripherals.

**Table 4.3. Memory and Bus System**

| MCU Series 0 | MCU Series 1 | Notes |
| --- | --- | --- |
| **Enhancements** | | |
| Single RAM segment. | The SRAM memory is split into different AHB slaves, each having an individual bus connection. | This enables simultaneous access to different RAM segments, e.g. if the core is accessing one RAM segment, the DMA can access another RAM segment without any bus contention. |
| **New Features** | | |
| — | Peripheral Bit Set and Clear – single cycle bit set and clear to peripherals' registers. | Dedicate inline function in em_bus.h.<br>`__STATIC_INLINE void BUS_RegBitWrite(volatile uint32_t* const addr, uint32_t bit, uint32_t val)`<br>Peripherals that do not support Bit Set and Bit Clear are EMU, RMU, ETH, SDIO, CAN0, CAN1, QSPI0, CRYOTIMER, USB and TRNG0. |
| — | Up to 256 KB RAM with ECC (SEC-DED) protection. | Add MSC_ECCCTRL, MSC_RAMECCADDR and MSC_RAM1ECCADDR registers.[1] |
| **Limitations** | | |
| — | RAM wait state at voltage scaling level 2 for EFM32GG11 and EFM32GG12:<br>• HFCLK > 38 MHz WS1<br>RAM wait state at voltage scaling level 0:<br>• HFCLK <= 16 MHz WS0<br>• 16 MHz < HFCLK <= 20 MHz WS1 | The RAM wait state is enabled by RAMxWSEN field in the MSC_RAMCTRL register.<br>[1] |

**Note:**

1. SRAM can support the max bus frequency at both voltage scaling levels 2 and 0 in EFM32TG11. No waits required with this device. There is no ECC protection in the EFM32TG11 device.

### 4.2.3 Memory System Controller (MSC)

The major change is the addition of a dedicated page for the bootloader and AAP lock.

**Table 4.4. MSC**

| MCU Series 0 | MCU Series 1 | Notes |
|---|---|---|
| **Enhancements** | | |
| Configure MSC_TIMEBASE register for flash erase and write operations. | Timing configuration is not required for flash erase and write operations. | — |
| Bus fault only on access to unmapped code and system space. | Bus fault on different scenarios. | The different bus fault responses are enabled by fields in the MSC_CTRL register. |
| **New Features** | | |
| Bootloader is placed on Main Page. | Bootloader can be placed on dedicated page at address 0x0FE10000 (Upto 32 KB)[1]. | The system is configured to boot from bootloader at address 0x0FE10000 automatically after system reset.<br>User can bypass the bootloader by clear bit 1 in Config Lock Word 0 (CLW0) in word 122 of lockbit (LB) page. |
| — | An additional atomic Read-clear operation for IFC register. | It can be enabled by setting IFCREADCLEAR in the MSC_CTRL register. |
| — | Authentication Access Port (AAP) lock bits for AAP lock. | Word 124 of lockbit (LB) page is the AAP lock word (ALW). |
| — | The SWITCHINGBANK command (MSC_CMD) initiates a bank switching to swap between two flash instances. | This command is only available on devices with dual-bank flash.[1]<br>This feature can be disabled by Config Lock Word 1 (CLW1) in word 123 of lockbit (LB) page. |
| — | Low voltage flash read when scaling down supply voltage to reduce current consumption. | The system clock frequency and flash wait states should be programmed accordingly since it takes a longer time to read from flash with a lower voltage supply.<br>Flash write/erase is not supported in low voltage mode. |
| — | Bootloader software reads and writes enable. | Reading and writing of bootloader area may be enabled with the MSC_BOOTLOADERCTRL register.<br>The BOOTLOADERCTRL register is write-once, so after writing the register, a reset of the system is required in order to change permissions again. |
| — | Advance cache control. | Through MSC_CACHECONFIG0 and MSC_RAMCTRL registers[2]. |
| **Limitations** | | |

| MCU Series 0 | MCU Series 1 | Notes |
| --- | --- | --- |
| Flash wait states:<br>• HFCLK > 16 MHz WS1<br>• HFCLK > 32 MHz WS2 | Flash wait states at voltage scaling level 2 for EFM32GG11 and EFM32GG12:<br>• HFCLK <= 18 MHz WS0<br>• HFCLK > 18 MHz WS1<br>• HFCLK > 36 MHz WS2<br>• HFCLK > 54 MHz WS3<br>Flash wait states at voltage scaling level 2 for EFM32TG11:<br>• HFCLK <= 25 MHz WS0<br>• 25 < HFCLK <= 48 MHz WS1 | Flash wait states (MODE field in MSC_READCTRL register) at voltage scaling level 0 for EFM32GG11 and EFM32GG112:<br>• HFCLK <= 7 MHz WS0<br>• 7 MHz < HFCLK <= 14 MHz WS1<br>• 14 MHz < HFCLK <= 20 MHz WS2<br>Flash wait states (MODE field in MSC_READCTRL register) at voltage scaling level 0 for EFM32TG11:<br>• HFCLK <= 10 MHz WS0<br>• 10 MHz < HFCLK <= 20 MHz WS1 |
| No Flash Startup time on transitions from EM2/3 to EM0 | Flash Startup time on transitions from EM2/3 to EM0 depends on the current operating conditions. | The related parameters are stored in MSC_STARTUP register. |
| Minimum 20000 erase cycles endurance. | Minimum 10000 erase cycles endurance. | — |

**Note:**

1. The bootloader size in EFM32TG11 is 18 kB.
2. Not supported by EFM32TG11.

## 4.3 Clock Management

### 4.3.1 Clock Management Unit (CMU)

The major changes are the new HFXO automatic start features and Digital Phased-Locked Loop (DPLL).

**Table 4.5. CMU**

| MCU Series 0 | MCU Series 1 | Notes |
|---|---|---|
| **Enhancements** | | |
| 1.2 MHz – 28 MHz HFRCO (1.2, 6.6, 11, 14, 21 and 28 MHz), HFRCO is 14 MHz after reset. | 1 MHz – 72 MHz HFRCO (1, 2, 4, 7, 13, 16, 19, 26, 32, 38, 48, 56, 64 and 72 MHz), HFRCO is 19 MHz after reset[1]. | Additional FINETUNING and FINETUNINGEN fields in MSC_HFRCOCTRL register for HFRCO frequency tuning. |
| 1.2 MHz – 28 MHz AUXHFRCO (1.2, 6.6, 11, 14, 21 and 28 MHz), AUXHFRCO is 14 MHz after reset. | 1 MHz – 50 MHz AUXHFRCO (1, 2, 4, 7, 13, 16, 19, 26, 32, 38, 48 and 50 MHz), AUXHFRCO is 19 MHz after reset[2]. | Additional FINETUNING and FINETUNINGEN fields in MSC_AUXHFRCOCTRL register for AUXHFRCO frequency tuning. |
| 24 or 48 MHz USHFRCO. | 1 MHz – 50 MHz USHFRCO, USHFRCO is 48 MHz after reset[3]. | — |
| LFRCO and LFXO ready interrupt are available in EM0 - EM1 energy modes. | LFRCO and LFXO ready interrupt are available in EM0 - EM2 energy modes. | The LFRCORDY and LFXORDY fields are in CMU_IEN register. |
| Startup time setup for LFXO and HFXO. | Startup time setup for LFRCO, LFXO, and HFXO. | The HFXO has a second time-out counter which can be used to achieve deterministic startup time based on timing from the LFXO, ULFRCO, or LFRCO. |
| Only TUNING field in CMU_LFRCOCTRL register. | More fields in CMU_LFRCOCTRL register to configure LFRCO. | — |
| — | More settings for HFXO startup with on-chip tunable capacitance[4]. | Add CMU_HFXOCTRL, CMU_HFXOSTARTUPCTRL, CMU_HFXOSTEADYSTATECTRL and CMU_HFXOTIMEOUTCTRL registers. |
| — | More settings for LFXO startup with on-chip tunable capacitance. | Add CMU_LFXOCTRL register. |
| One clock (HFPERCLK) drives all high-frequency peripherals. | Three clocks (HFPERCLK, HFPERBCLK, and HFPERCCLK) drive all high-frequency peripherals. | All the peripherals that are driven by these clocks can be clock gated individually when not in use. This is done by clearing the clock enable bit for the specific peripheral in CMU_HFPERCLKEN0 or CMU_HFPERCLKEN1.<br><br>The prescale factors for prescaling HFCLK into HFPERCLK, HFPERBCLK, and HFPERCCLK are set using the CMU_HFPERPRESC, CMU_HFPERPRESCB, and CMU_HFPERPRESCC registers respectively. |
| CMU to output clocks on two pins. | CMU to output clocks on three pins. | The clock selection is configured by CLKOUTSEL0, CLKOUTSEL1 and CLKOUTSEL2 fields respectively in CMU_CTRL register. |
| — | Clock output to PRS. | Selected by a PRS consumer as CMUCLKOUT0 or CMUCLKOUT1. |
| — | Calibration input from PRS. | Selected by PRSUPSEL and PRSDOWNSEL fields in CMU_CALCTRL register. |

| MCU Series 0 | MCU Series 1 | Notes |
|---|---|---|
| The Watchdog (WDOG) can be clocked by LFRCO, LFXO, and ULFRCO. | The Watchdog (WDOG) can be clocked by HFCORECLK, LFRCO, LFXO, and ULFRCO. | — |
| The SYSTICK can be clocked by HFCORECLK. | The SYSTICK can be clocked by HFCORECLK or LFBCLK. | The LFBCLK can be HFCLKLE, LFXO, LFRCO, or ULFRCO. |
| **New Features** | | |
| — | Internal clock doubler for HFXO, mainly for USB core clock[3]. | It generates a clock (HFXOX2) with double frequency compared to HFXO clock.<br>The use of HFXOX2 is only allowed for crystals up to 25 MHz. |
| — | Add LFECLK for RTCC. | LFECLK is available down to EM4H. |
| — | Add HFEXPCLK for HFCLK output. | Prescaled version of HFCLK to CMU_OUT0 or CMU_OUT1. |
| — | Add HFBUSCLK with prescaler, separate it from HFCLK and HFCORECLK. | The CRYPTO0, ETH (APB), GPIO, GPCRC and QSPI0 are clocked by HFBUSCLK, if enabled. |
| — | AUXHFRCO can clock ADC in EM2/3. | Selected by ADC0CLKSEL field in CMU_ADCCTRL register. |
| — | Automatic HFXO Start. | The enabling of the HFXO and its selection as HFCLK source can be performed automatically by hardware. |
| — | New clock sources, HFRCODIV2 and CLKIN0, for HFCLK. | The HFROCDIV2 is HFRCO divided by 2 and CLKIN0 is the external clock source from the dedicate CLKIN0 pin. |
| — | The Digital Phase-Locked Loop (DPLL) generates a digitally controlled oscillator (DCO), which is HFRCO, as a ratio of a reference clock source. | The reference clock source can be USHFRCO[3], HFXO, LFXO, or CLKIN0.<br>The DPLL is disabled automatically when entering EM2, EM3, EM4H or EM4S. |
| **Limitations** | | |
| HFXO range is 4 – 48 MHz. | HFXO range is 4 – 50 MHz[5]. | — |
| $HFCORECLK_{LE}$ > 24 or 32 MHz<br>• Set HFLE if available<br>• Set HFCORECLKLEDIV to DIV4 | $HFCLK_{LE}$ > 32 MHz<br>• Set WSHFLE<br>• Set HFCLKLEPRESC to DIV4 | The WSHFLE is in CMU_CTRL register and HFCLKLEPRESC is in CMU_HFPRESC register. |
| — | Peripheral access wait state:<br>• HFCLK > 50 MHz WS1 | The peripheral access wait state is enabled by WAITMODE field in the MSC_CTRL register[6]. |
| — | LFACLK cannot select $HFCLK_{LE}$ as clock source. | The LCD, LESENSE, LETIMER, and RTC are clocked by LFACLK. |

**Note:**
1. For EFM32TG11: 1 MHz – 48 MHz HFRCO (1, 2, 4, 7, 13, 16, 19, 26, 32, 38 and 48 MHz), HFRCO is 19 MHz after reset .
2. For EFM32TG11: 1 MHz – 48 MHz AUXHFRCO (1, 2, 4, 7, 13, 16, 19, 26, 32, 38 and 48 MHz), AUXHFRCO is 19 MHz after reset.
3. There is no USHFRCO in EFM32TG11.
4. In EFM32GG12, the Peak Detection Threshold can be configured using the CMU_HFXOCTRL1 register.
5. The range of HFXO for EFM32TG11 is 4 to 48 MHz.
6. Wait states cannot be configured in EFM32TG11.

### 4.4 Energy Management

#### 4.4.1 Energy Management Unit (EMU)

The major changes are new EM4 energy modes (EM4H and EM4S), 5V Sub-System, and the new DC-DC converter module.

**Table 4.6. EMU**

| MCU Series 0 | MCU Series 1 | Notes |
|---|---|---|
| **Enhancements** | | |
| No way to know the system wakes up from EM2 and EM3. | Interrupt to indicate system wakes up from EM2 and EM3. | EM23WAKEUP in EMU_IF register will be set when the system wakes from EM2 and EM3. |
| Single EM4 energy mode. | EM4 splits in EM4H (Hibernate) and EM4S (Shutoff) energy modes. | Set EM4STATE in EMU_EM4CTRL register to enter EM4H when entering EM4.<br>In EM4H, the regulator will be on in reduced mode allowing for RTCC. Otherwise, when entering in EM4, the regulator will be disabled allowing for lowest power mode, Shutoff state (EM4S). |
| When RAM block is powered down, it cannot be powered up again without reset. | Selected RAM block is powered down in EM2/3 with full access in EM0. | The RAM block 0 (upto 128 kB) will always be powered on for proper system functionality.<br>The stack must be located in retained memory. |
| The Brown-Out Detector (BOD) cannot be disabled. | The Brown-Out Detector (BOD) can be disabled in EM2 to minimize current. | Set EM2BODDIS in EMU_CTRL register to disable BOD when entering EM2. |
| Backup power domain with BURTC and 512 byte retention in EM4. | Backup power domain with RTCC, CRYOTIMER and 128 byte retention in EM4H. | Thresholds for backup entry and backup exit are monitored by Voltage Monitor (VMON). |
| **New Features** | | |
| No DC-DC converter module. | With a DC-DC converter module to power internal circuits and it requires an external inductor and capacitor. | The DC-DC converter allows up to two external hookup configurations with additional options giving flexible power architecture selection. |
| Use Voltage Supply Comparator (VCMP) to monitor the supply voltage. | Use Voltage Monitor (VMON) to monitor different voltage sources. | Trigger points for interrupts are preloaded but may be reconfigured. |
| ADC temperature sensor only. | EMU temperature sensor is always running (except in EM4 Shutoff) and is independent from ADC temperature sensor. | Temperature measurement is taken every 250ms with the 8-bit result stored in EMU_TEMP register.<br>The high and low temperature trigger points for interrupt are configurable. |
| — | 5V Sub-System[1] | The 5V sub-system manages the 5V power domains of a chip which includes a 5V regulator.<br>Add EMU_R5V* registers. |
| — | Separate voltage scaling controls are available for the different energy modes.<br>• EM01 Voltage Scaling<br>• EM23 Voltage Scaling<br>• EM4H Voltage Scaling | Voltage scaling allows for a tradeoff between power and performance, the user can scale voltages between Voltage Scale Level 2 (1.2 V) and Voltage Scale Level 0 (1.0 V).<br>The software should follow certain sequences for supply voltage scaling up and down. |

| MCU Series 0 | MCU Series 1 | Notes |
|---|---|---|
| — | Peripherals that are available in EM2 Deep Sleep or EM3 Stop can optionally be powered down during EM2 or EM3. | The EMU_EM23PERNORETAINCTRL register can be used to setup unused peripherals for powering down prior to EM23 entry. |
| **Limitations** | | |
| — | — | — |
| **Note:**<br>1. There is no 5V subsystem in EFM32TG11. | | |

### 4.4.2 Reset Management Unit (RMU)

The major change is configurable reset levels for Watchdog, Lockup, Pin, and System reset requests.

**Table 4.7. RMU**

| MCU Series 0 | MCU Series 1 | Notes |
|---|---|---|
| **Enhancements** | | |
| RESETn pin reset can only be hard reset. | RESETn pin reset can be configured to be either hard or soft reset. | The soft reset can be configured to be either DISABLED, LIMITED, EXTENDED or FULL.<br>Hard resets will reset the entire chip (= soft reset configured as FULL).<br>To configure RESETn pin reset as a hard reset, clear the PINRESETSOFT bit in CLW0 in the Lock bit page. |
| **New Features** | | |
| Watchdog, Lockup, Pin and System reset request are non configurable. | Watchdog, Lockup, Pin (RESETn) and System reset request are sources for soft reset. | The reset level (DISABLED, LIMITED, EXTENDED or FULL) for soft reset sources is configured in the xxxRMODE bitfields in RMU_CTRL register. |
| **Limitations** | | |
| Brown-out Detection (BOD) on Regulated domain, Unregulated domain, Analog Power Domain 0 (AVDD0) and Analog Power Domain 1 (AVDD1). | Brown-out Detection (BOD) on Analog Unregulated Power Domain (AVDD), Digital Unregulated Power Domain (DVDD) and Regulated Digital Domain (DECOUPLE). | — |

### 4.5 Serial Interfaces

#### 4.5.1 Inter-Integrated Circuit Interface (I2C)

There are no major changes for the I2C module.

**Table 4.8. I2C**

| MCU Series 0 | MCU Series 1 | Notes |
|---|---|---|
| **Enhancements** | | |
| Separate single buffer for transmit and receive. | Separate 2-level FIFO for transmit and receive. | Access through I2Cn_TXDOUBLE and I2Cn_RXDOUBLE registers. |
| **New Features** | | |
| — | — | — |
| **Limitations** | | |
| — | — | — |

#### 4.5.2 Low Energy Universal Asynchronous Receiver/Transmitter (LEUART)

There are no major changes for the LEUART module.

**Table 4.9. LEUART**

| MCU Series 0 | MCU Series 1 | Notes |
|---|---|---|
| **Enhancements** | | |
| — | — | — |
| **New Features** | | |
| — | — | — |
| **Limitations** | | |
| — | — | — |

#### 4.5.3 Universal Synchronous Asynchronous Receiver/Transmitter (USART)

The major changes are hardware flow control and the addition of a 8-bit timer.

**Table 4.10. USART**

| MCU Series 0 | MCU Series 1 | Notes |
|---|---|---|
| **Enhancements** | | |
| The baud rate divider is 13-bit integral part and a 2-bit fractional part. | Extend baud rate divider to 20-bit value, with a 15-bit integral part and a 5-bit fractional part. | Better accuracy on baud rate generation. |
| PRS RX input only. | PRS RX and PRS CLK inputs. | The USART can be configured to receive data and clock directly from PRS channels.<br>The PRS channels are selected by RXPRSSEL and CLKPRSSEL in USARTn_INPUT register. |
| **New Features** | | |
| — | Automatic Baud Rate Detection. | Setting AUTOBAUDEN in USARTn_CLKDIV register uses the first frame received to automatically set the baud rate provided that it contains 0x55 (IrDA uses 0x00). |
| — | Hardware Flow Control with debug halt. | This function is configured by USARTn_CTRLX register. |
| — | New 8-bit Timer to create timing for a variety of uses. | It can be used for different purposes such as RX timeout, break detection, response timeout, and RX enable delay. |
| — | Ultra high speed (36 MHz) operation. | It is only available on USART2. |
| **Limitations** | | |
| — | — | — |

#### 4.5.4 Universal Serial Bus Controller (USB)

The major changes are internal regulator specification and charger detection circuitry. There is no USB Controller on the EFM32TG11.

**Table 4.11. USB**

| MCU Series 0 | EFM32GG11 and EFM32GG12 | Notes |
|---|---|---|
| **Enhancements** | | |
| Internal 3.3V Regulator<br>• Output voltage: 3.3V<br>• Output current: 50 mA<br>• Input voltage range: 4.0 - 5.5V<br>• Low quiescent current: 100 µA | Internal 3.3V Regulator<br>• Output voltage: 2.4V to 3.8V<br>• Output current: 200 mA<br>• Input voltage range: 2.7 - 5.5V<br>• Low quiescent current: 9 µA | The USB operation requires 3.3V output voltage.<br>If USB is not available in the device, the regulator can use for 5V sub-system. |
| USB_VBUS cannot be the input supply to the internal regulator. | VBUS can also be the input supply to the internal regulator. | It is determined by the INPUTMODE field in the EMU_R5VCTRL register.<br>Eliminate the external connection between VBUS and VREGI in USB device mode. |
| USB_DMPU pin is required for USB low speed device. | USB_DMPU pin is not required and removed. | — |
| **New Features** | | |
| — | Charger detection circuitry. | The peripheral can distinguish between a standard downstream port (SDP), dedicated charging port (DCP), or a charging downstream port (CDP).<br>The device itself does not contain direct battery management or battery charging circuitry. |
| — | D+ and D- can be routed to ADC input to support ACM and proprietary charger architectures. | ADC multiplexer connections to the USB D+ and D- pins are provided internally for detecting the presence of non-standard charging hardware. |
| **Limitations** | | |
| Can be used in various OTG Dual Role Device, Host and Device configurations. | Can be used in Device and Host configurations. | — |
| The USB requires the device to run a 48 MHz crystal (2500 ppm or better). | In USB host mode, an external 48 MHz or 24 MHz crystal (2500ppm or better) is required. | For USB device mode, USB may use an external crystal or it may be clocked from its own internal oscillator (USHFRCO). |

## 4.6 I/O Ports

### 4.6.1 General Purpose Input/Output (GPIO)

The major change is the addition of configurable over voltage tolerance inputs.

**Table 4.12. GPIO**

| MCU Series 0 | MCU Series 1 | Notes |
|---|---|---|
| **Enhancements** | | |
| Use DOUTSET and DOUTCLR registers to set and clear GPIO pins. | Removed DOUTSET & DOUTCLR registers. | Use Peripheral Bit Set and Clear to set and clear GPIO pins. |
| All pins with the same pin number (n) are grouped together to trigger one interrupt flag or to form one PRS producer output. | All pins within a group of four (0-3, 4-7, 8-11, 12-15) from all ports are grouped together to trigger one interrupt flag index or to form one PRS producer output. | It is more flexible to configure GPIO interrupt source or PRS producer output while maintaining compatibility with MCU Series 0.<br>Add GPIO_EXTIPINSELL and GPIO_EXTIPINSELH registers. |
| **New Features** | | |
| — | Input disable. | The pin inputs can be disabled on a port-by-port basis. The input of pins configured using the normal or alternate MODEn settings can be disabled by setting DINDIS or DINDISALT in GPIO_Px_CTRL register. |
| — | Slewrate setting. | The slewrate can be applied to pins on a port-by-port basis. The actual slewrate applied to the selected pins is configured in the SLEWRATE and SLEWRATEALT fields in GPIO_Px_CTRL register. |
| — | Over voltage tolerance is available for most pins. | The over voltage tolerance applied to the selected pins is configured in the GPIO_Px_OVTDIS register (Default over voltage is enabled for each pin supporting that feature).<br>Disabling the over voltage tolerance for a pin will provide less distortion on that pin, which is useful when the pin is used as analog input. |
| — | Level interrupt for EM4 wakeup. | GPIO can generate a level interrupt using the input of any EM4 wake-up pin on the device. |
| **Limitations** | | |
| Drive strength – 0.1, 1, 6 and 20 mA. | Drive strength – 1 mA and 10 mA. | — |

#### 4.6.2 External Bus Interface (EBI)

The major change is TFT alpha blending color encoding. There is no EBI peripheral in the EFM32TG11 devices.

**Table 4.13. EBI**

| MCU Series 0 | EFM32GG11 and EFM32GG12 | Notes |
|---|---|---|
| **Enhancements** | | |
| Common location field for EBI/TFT address, data, and control signal pins. | Independent location field for EBI/TFT address, data, and control signal pins. | Add EBI_ROUTELOC0 and EBI_ROUTELOC1 registers. |
| The foreground and background color of TFT alpha blending are encoded in either 565 RGB or 555 RGB format. | The foreground and background color of TFT alpha blending can be encoded in different formats. | The foreground color can be encoded in any of these ARGB formats: 0555, 0565, 0666, 0888, 5555, 6565, 6666 or 8888.<br>The background color can be encoded in these RGB formats: 555, 565, 666, 888.<br>Add EBI_TFTCOLORFORMAT register. |
| The VBPORCH and VFPORCH fields in EBI_TFTVPORCH register are 8 bits. | The VBPORCH and VFPORCH fields in EBI_TFTVPORCH register are 12 bits. | — |
| The bit width of EBI_TFTDD, TFTPIXELn, TFTMASK registers is 16. | The bit width of EBI_TFTDD, TFTPIXELn, TFTMASK registers is 24. | — |
| **New Features** | | |
| — | — | — |
| **Limitations** | | |
| The 16 EBI_AD lines are all enabled by the EBIPEN in EBI_ROUTE register. | The 16 EBI_AD lines are all enabled by the EBIPEN in EBI_ROUTEPEN register. | It is not possible to only enable lower 8 bits of EBI_AD pins even the upper 8 bits are not used. |

### 4.7 Timers and Triggers

#### 4.7.1 Timer/Counter (TIMER)

The major change is dedicate TIMER now has 4 compare/capture channels, can be used for RGBW LED control.

**Table 4.14. TIMER**

| MCU Series 0 | MCU Series 1 | Notes |
|---|---|---|
| **Enhancements** | | |
| 3 Compare/Capture channels. | 3 or 4 Compare/Capture channels. | The 4 Compare/Capture channels are only available on TIMER1 and WTIMER1. |
| **New Features** | | |
| — | The new 32 bit general purpose timer WTIMER. | The TIMER and WTIMER peripherals are identical except for the timer width. |
| **Limitations** | | |
| — | — | — |

#### 4.7.2 Real Time Counter (RTC)

The major change is to increase the compare channels. There is no RTC peripheral in the EFM32TG11 device.

**Table 4.15. RTC**

| MCU Series 0 | EFM32GG11 and EFM32GG12 | Notes |
| --- | --- | --- |
| **Enhancements** | | |
| Two compare channels are available in the RTC. | Six compare channels are available in the RTC. | The compare value registers are RTC_COMPA_COMP to RTC_COMPF_COMP. |
| RTC_FREEZE and RTC_SYNCBUSY registers for data synchronization. | These registers are removed due to RTC can support immediate synchronization. | — |
| A compare event on either of the compare channels can start the LETIMER. | Use PRS from compare match event to start, stop, and clear LETIMER. | — |
| **New Features** | | |
| — | — | — |
| **Limitations** | | |
| — | — | — |

#### 4.7.3 Low Energy Timer (LETIMER)

The major change is to replace RTC trigger with PRS.

**Table 4.16. LETIMER**

| MCU Series 0 | MCU Series 1 | Notes |
| --- | --- | --- |
| **Enhancements** | | |
| — | — | — |
| **New Features** | | |
| LETIMER can be started by a RTC event or started, stopped, and cleared by software. | LETIMER can be started, stopped, and cleared by PRS or software. | The PRS mode and input are configured by LETIMERn_PRSSEL register. |
| **Limitations** | | |
| — | — | — |

#### 4.7.4 Peripheral Reflex System (PRS)

The major change is that PRS can trigger the core and DMA.

**Table 4.17. PRS**

| MCU Series 0 | MCU Series 1 | Notes |
|---|---|---|
| **Enhancements** | | |
| — | Optional channel output invert. | Set INV bit in PRS_CHx_CTRL register to invert channel output. |
| — | Pulse stretch for domains running at different frequency. | Set STRETCH bit in PRS_CHx_CTRL register to stretch channel output.<br>Stretches channel output to ensure that the target clock domain detects it. |
| — | Read back PRS channel value. | Access through PRS_PEEK register. |
| PRS channels 0-3 can output to GPIO with one common location field. | All PRS channels can output to GPIO with independent location field. | This function is configured by PRS_ROUTELOCn registers. |
| **New Features** | | |
| — | PRS channel can use as PRS source. | Use PRSL or PRSH[1] of SOURCESEL field in PRS_CHx_CTRL register to select PRS channel as PRS input source. |
| — | The PRS can be used to send events to the core. | This is very useful in combination with the Wait For Event (WFE) instruction.<br>This function is configured by PRS_CTRL register. |
| — | Up to two independent DMA requests can be generated by the PRS. | This function is configured by PRS_DMAREQ0 and PRS_DMAREQ1 registers<br>The PRS signals triggering the DMA requests are selected with the SOURCESEL (= 0x1 for PRS) and SIGNAL (= 0x0 for PRSREQ0 or = 0x1 for PRSREQ1) fields in the LDMA_CHx_REQSEL register. |
| — | Configurable PRS Logic (AND and OR). | Each PRS channel has three logic functions that can be used by themselves or in combination.<br>The selected PRS source can be AND'ed with the next PRS channel output, OR'ed with the previous PRS channel output and inverted. |
| **Limitations** | | |
| — | — | — |
| **Note:**<br>1. There are 16 Configurable Reflex Channels in EFM32GG12 and 8 Configurable Reflex Channels in EFM32TG11. Hence, there is no PRSH SOURCESEL field in the PRS_CHx_CTRL register in these devices. | | |

#### 4.7.5 Pulse Counter (PCNT)

The major change is new support for the quadrature decoder in oversampling mode.

**Table 4.18. PCNT**

| MCU Series 0 | MCU Series 1 | Notes |
|---|---|---|
| **Enhancements** | | |
| Fix filter length in oversample mode. | Programmable filter length in oversample mode. | This is configured by FILTEN field in PCNTn_OVSCFG register. |
| PCNTn_CNT reset through RSTEN. | PCNTn_CNT reset through RSTEN or CNTRSTEN. | CNTRSTEN in PCNTn_CTRL register works in a similar manner as RSTEN, but only resetting the counter, PCNTn_CNT. |
| **New Features** | | |
| Externally clocked quadrature decoder mode only. | Externally clocked quadrature decoder 1X mode and Over-sampling quadrature decoder 1X, 2X and 4X modes with flutter removal. | Flutter is removed when setting FLUTTERRM field in PCNTn_OVSCFG register. |
| — | Cascading PCNT through PRS. | Possible to form a 32-bit pulse counter. |
| **Limitations** | | |
| — | — | — |

#### 4.7.6 Watchdog Timer (WDOG)

The major change is the Watchdog timeout can either generate a reset or an interrupt.

**Table 4.19. WDOG**

| MCU Series 0 | MCU Series 1 | Notes |
|---|---|---|
| **Enhancements** | | |
| WDOG reset cannot be disabled. | Option to disable WDOG reset. | Set WDOGRSTDIS in WDOG_CTRL register to disable watchdog timeout reset, option to trigger timeout interrupt instead. |
| **New Features** | | |
| — | Configurable warning interrupt at a percentage of timeout period. | Use WARNSEL field in WDOG_CTRL register, the percentage are 25%, 50%, or 75%. |
| — | Configurable window interrupt at a percentage of timeout period. | Use WINSEL field in WDOG_CTRL register, the percentage are 12.5%, 25%, 37.5%, 50%, 62.5%, 75%, or 87.5%. |
| — | PRS as a watchdog clear, source is selected by WDOG_PCH0_PRSCTRL register. | Use CLRSRC bit in WDOG_CTRL register to select software or PRS to clear the watchdog counter. |
| — | Trigger an interrupt when a WDOG clear happens before a PRS event has been detected. | Use WDOGn_PCH0_PRSCTRL and WDOGn_PCH1_PRSCTRL registers to select PRS channels. |
| **Limitations** | | |
| — | — | — |

### 4.7.7 Low Energy Sensor Interface (LESENSE)

The major changes are ADC support and sensor evaluation.

**Table 4.20. LESENSE**

| MCU Series 0 | MCU Series 1 | Notes |
|---|---|---|
| **Enhancements** | | |
| Uses the analog comparators, ACMP, for measurement of sensor signals. | Uses the analog comparators, ACMP, or the ADC for measurement of sensor signals. | — |
| Sensor evaluation can be based on either ACMP outputs, or threshold comparison. | Sensor evaluation can be based on either ACMP outputs, threshold comparison, sliding window, or step detection. | In sliding window mode, the sensor data is compared against the upper and lower limits of a window range.<br><br>Step detection is used to detect steps in the sensor data compared to sensor data from the previous measurement. |
| LESENSE decoder, which is a configurable state machine with up to 16 states. | LESENSE decoder, which is a configurable state machine with up to 32 states. | — |
| **New Features** | | |
| — | The decoder has a PRS output named DECCMP. | This output can be used to indicate which state, or subset for states, the decoder is currently in. |
| **Limitations** | | |
| Maximum four inputs for LESENSE decoder. | Maximum four inputs for LESENSE decoder. | — |

## 4.8 Analog Interfaces

### 4.8.1 Analog Comparator (ACMP)

The major changes are the new APORT, and the external I/O can use as a reference voltage.

**Table 4.21. ACMP**

| MCU Series 0 | MCU Series 1 | Notes |
|---|---|---|
| **Enhancements** | | |
| FULLBIAS and HALFBIAS fields in ACMPn_CTRL register. | No HALFBIAS field in ACMPn_CTRL register. | The HALFBIAS field is merged with BIASPROG field (increase to 6-bit value) in ACMPn_CTRL register. |
| Symmetric hysteresis only. | Symmetric and asymmetric hysteresis. | Hysteresis is configured through the HYST field in ACMPn_HYSTERESIS0 and ACMPn_HYSTERESIS1 registers. The hysteresis value can be positive or negative. |
| Selectable internal voltage for negative input only. | Selectable internal voltage for both positive and negative inputs. | Internal voltages are VDD, 1.25 V, 2.5 V and VDAC channel output. |
| Scaler for VDD. | Voltage dividers for VDD, 1.25 V and 2.5 V. | Voltage dividers in the ACMPn_HYSTERESIS0/1 registers. |
| 4 resistance values for the internal capacitive sense resistor. | 8 resistance values for the internal capacitive sense resistor. | — |
| **New Features** | | |
| — | Up to 48 I/O (through APORT) can be used as a dividable reference voltage. | Not all selectable I/O are available on a given device, refer to the device data sheet for details. |
| — | Selected level of accuracy. | Configured by ACCURACY field in ACMPn_CTRL register, default is low accuracy mode to consume less current. |
| Voltage monitor function in VCMP. | The ACMP can be used to monitor supply voltages. | The voltage source (including AVDD, VREGVDD, IOVDD0 and IOVDD1) can be selected by PWRSEL field in ACMPn_CTRL register. |
| **Limitations** | | |
| The warm-up time is HFPERCLK dependent and should be >= 10us. | The warm-up time is self-timed and will complete within 5µs. | — |
| Dedicated capacitive sense mode with up to 16 inputs. | Dedicated capacitive sense mode with up to 8 inputs. | The capacitive sense mode inputs are configured through APORT. |
| Up to 8 external I/O for both positive and negative input terminals. | Up to 144 or 160 external I/O (through APORT) for both positive and negative input terminals. | Add ACMPn_APORTREQ and ACMPn_APORTCONFLICT registers.<br><br>Not all selectable I/O are available on a given device, refer to the device data sheet for details. |

### 4.8.2 Analog to Digital Converter (ADC)

The major changes are FIFO and APORT support, and the ADC can run in EM2/3 energy mode.

**Table 4.22. ADC**

| MCU Series 0 | MCU Series 1 | Notes |
|---|---|---|
| **Enhancements** | | |
| Separate single buffer for single and scan conversion. | Separate FIFO for single and scan conversion. | Four deep 32-bit FIFO to store conversion data along with channel ID and option to overwrite old data when full. |
| The ADC can run in EM0 - EM1 energy modes. | The ADC can run in EM0 - EM3 energy modes. | ADC must be in asynchronous mode with AUXHFRCO as clock source to run in EM2/3.<br><br>The ADC can wake the system from EM2/3 to EM0 on enabled interrupts and it can also work with the DMA so that the system does not have to be woken up to consume data. |
| Up to 8 external input channels. | Up to 144 external input channels (through APORT). | Add ADCn_APORTREQ and ADCn_APORTCONFLICT registers.<br><br>Not all selectable I/O are available on a given device, refer to the device data sheet for details. |
| Up to 8 configurable samples in scan sequence. | Up to 32 configurable samples in scan sequence. | — |
| **New Features** | | |
| — | Programmable watermark to generate interrupt. | The DVL field of the ADCn_SINGLECTRLX/SCANTRLX controls the FIFO watermark crossing which sets the SINGLEDV/SCANDV bit in ADCn_STATUS high and is cleared when the data is read and watermark falls below the DVL threshold. |
| — | Window Compare Function. | The ADC supports window compare function on both latest single and scan sample.<br><br>The compare thresholds, ADGT and ADLT, are defined in the ADCn_CMPTHR register. |
| — | Programmable full scale (peak-to-peak) voltage (VFS) with selectable reference sources. | Advanced VFS configuration is enabled by setting the REF field in ADCn_SINGLECTRL or ADCn_SCANCTRL register to CONF. |
| — | The ADC can be used as random number generator. | This is done simply by choosing the REF in the ADCn_SINGLECTRL as CONF and by setting the VREFSEL in the ADCn_SINGLECTRLX as VENTROPY. |
| — | Externally controllable conversion start time using PRS in TIMED mode. | In TIMED mode, a long PRS pulse is expected to trigger the ADC and its negative edge directly finishes input sampling and starts the approximation phase, giving precise sampling frequency management. |
| — | The Keep Warm setting for the Reference is programmable. | By default the scan mode reference is kept warm (CHCONREFWARMIDLE field in the ADCn_CTRL register). The user can also choose to keep the single channel mode reference warm or to keep the last used reference warm. |
| — | Programmable delay between repetitive conversions. | The REPDELAY field in the ADCn_SINGLECTRLX and ADCn_SCANCTRLX registers can be used to set the delay between two repeated conversions in single channel and scan mode respectively. |
| — | Scan conversions can be triggered using LESENSE. | — |

| MCU Series 0 | MCU Series 1 | Notes |
|---|---|---|
| — | Programmable ADC behavior in debug mode. | If DBGHALT field in the ADCn_CTRL register is set to 1, then in debug mode ADC completes the current conversions and then halts. This means that all conversion triggers that were received before the debug halt occurred will be serviced before the ADC halts. |
| **Limitations** | | |
| ADC_CLK from 32 kHz to 13 MHz. | ADC_CLK from 32 kHz to 16 MHz. | — |
| Gain calibration is available in all voltage references except unbuffered 2xVDD. | Gain calibration is not available in VDD and external references. | Use software to implement gain calibration on VDD and external references. |

#### 4.8.3 Current Digital to Analog Converter (IDAC)

The major change is support for the APORT. There is no IDAC peripheral in EFM32TG11.

**Table 4.23. IDAC**

| MCU Series 0 | EFM32GG11 and EFM32GG12 | Notes |
|---|---|---|
| **Enhancements** | | |
| TUNING field is in IDAC_CAL register. | TUNING field is in IDAC_CUR-PROG register. | The IDAC_CAL register is removed. |
| DUTYCYCLEEN bit in IDAC_DUTYCONFIG register. | No DUTYCYCLEEN bit in IDAC_DUTYCONFIG register. | Duty cycle enable or disable is not available in EM0/1. |
| **New Features** | | |
| — | Selects the power source for the IDAC. | Use PWRSEL bit in IDAC_CTRL register to select AVDD or IOVDD as IDAC power source. |
| — | Delays EM2 entry until IDAC-OUT is stable. | Set EM2DELAY bit in IDAC_CTRL register. |
| — | Status to indicate the IDAC is active and output is stable. | Add IDAC_STATUS register.<br>When the IDAC is enabled and warmed up, the CURSTABLE bit in IDAC_STATUS will indicate that the IDAC is active and output is stable. |
| — | Interrupt for IDAC status and APORT bus conflict. | Add IDAC_IEN, IDAC_IF, IDAC_IFS and IDAC_IFC registers.<br>The IDAC can generate two types of interrupts, CURSTABLE and APORTCONFLICT, located in IDAC_IF register. |
| **Limitations** | | |
| IDAC can output current either to IDAC0_OUT pin, or to the currently selected ADC channel. | IDAC can output current either to IDAC0_OUT pin, or APORT pins. | Add IDAC_APORTREQ and IDAC_APORTCONFLICT registers.<br>Not all selectable I/O are available on a given device, refer to the device data sheet for details. |

#### 4.8.4 Liquid Crystal Display Driver (LCD)

The major changes are charge redistribution feature and charge pump capabilities.

**Table 4.24. LCD**

| MCU Series 0 | MCU Series 1 | Notes |
| --- | --- | --- |
| **Enhancements** | | |
| Each bit (0 to 9) of LCD_SEGEN register represents a group of 4 segment lines. | Each bit (0 to 31) of LCD_SEGEN register represents one segment line. | LCD_SEGEN2 is added to enable segment lines 32 to 39. |
| Voltage boost capabilities. | Charge pump capabilities. | Three modes (MODE in LCD_DISPCTRL register) are available for setting LCD voltage bias levels: current source mode, step down mode, or charge pump mode.<br><br>The contrast of different modes is set by CONTRAST in LCD_DISPCTRL register. |
| External 22 nF capacitor on LCD_BCAP_P and LCD_BCAP_N pins for voltage booster. | External 1 µF capacitor between LCD_BEXT pin and VSS for step down and charge pump modes. | LCD_BCAP_P and LCD_BCAP_N pins are removed. |
| **New Features** | | |
| — | Charge redistribution feature reduces LCD module current consumption by up to 40%. | CHGRDST in LCD_DISPCTRL register is used to select the number of prescaled $LFCLK_{LCD}$ cycles used for charge redistribution.<br><br>FRDIV in LCD_FRAMERATE register is used to provide an additional increase of the phase period. |
| — | Tristate the LCD pads. | It is controlled by LCDGATE in LCD_FREEZE register. |
| **Limitations** | | |
| The LCD common and segment pads are on fixed pins. | The LCD common and segment pads are on fixed pins. | The common is LCD_COMx and segment is LCD_SEGx in data sheet. |

### 4.8.5 Operational Amplifier (OPAMP)

The major changes are APORT and OPAMP timing support.

**Table 4.25. OPAMP**

| MCU Series 0 | MCU Series 1 | Notes |
|---|---|---|
| **Enhancements** | | |
| The external pins for POSSEL and NEGSEL mux are fixed. | The external pins for POSSEL and NEGSEL mux can be fixed or from APORT. | The OPAn_P pin is for positive input and OPAn_N pin is for negative input.<br><br>Not all selectable I/O are available on a given device, refer to the device data sheet for details. |
| The opamp has two outputs, the main output and an alternative output network. | The opamp has three outputs, the main output, an alternative output network, and an APORT output. | The OPAn_OUT pin is for main output and OPAn_OUTALT pin is for alternative output.<br><br>Add VDACn_OPAx_APORTREQ and VDACn_OPAx_APORTCONFLICT registers.<br><br>Not all selectable I/O are available on a given device, refer to the device data sheet for details. |
| Alternative output with lower drive strength. | Programmable drive strength. | It is configured by DRIVESTRENGTH field in VDACn_OPAx_CTRL register. |
| The outputs of the opamps can be routed to the ADC. | The outputs of the opamps can be routed to the ADC and ACMP. | — |
| Opamp can be enabled with software. | Opamp can be enabled either with software or PRS. | The default source is software, setting PRSEN to 1 in VDACn_OPAx_CTRL register enables PRS mode. |
| **New Features** | | |
| — | Programmable startup delay. | It is configured by STARTDLY field in VDACn_OPAx_TIMER register. |
| — | Programmable warmup time. | It depends on the selected drive strength. It is configured by WARMUPTIME field in VDACn_OPAx_TIMER register. |
| — | Programmable settle time. | It depends on the load at OPAMP output and selected drive strength. It is configured by SETTLETIME field in VDACn_OPAx_TIMER register. |
| — | Preconfigured resistor ladder with 3X gain. | The 3x gain resistor ladder is enabled by setting GAIN3X in VDACn_OPAx_MUX register. By default all opamps are configured in 3x gain mode. |
| — | Unity gain bandwidth and opamp output scaling. | Unity gain bandwidth of an opamp can be scaled by setting the INCBW bit in VDACn_OPAx_CTRL register.<br><br>Opamp output drive strength is scaled by one half when the OUTSCALE bit in VDACn_OPAx_CTRL register is set. |
| — | Interrupt generation for opamp operation. | Interrupt flags for the opamp output is settled externally at the load, protocol error when the opamp is triggered, and APORT bus conflict occurs. |
| — | Asynchronous PRS output. | One of two asynchronous PRS outputs (opamp warmup and output valid status) can be enabled for each opamp by setting PRSOUTMODE in VDACn_OPAx_CTRL register. |
| **Limitations** | | |

| MCU Series 0 | MCU Series 1 | Notes |
|---|---|---|
| Two of the opamps are part of the DAC, while the others are standalone. | Two of the opamps are part of the VDAC, while the others are standalone. | Since OPA0 and OPA1 are part of the VDAC, special considerations need to be taken when both VDAC channel 0/channel 1 and OPA0/OPA1 are used. |

#### 4.8.6 Digital to Analog Converter (VDAC)

The major changes are APORT and VDAC timing support.

**Table 4.26. VDAC**

| MCU Series 0 | MCU Series 1 | Notes |
|---|---|---|
| **Enhancements** | | |
| Each DAC channel has two outputs, the main output (DACn_OUTx) and an alternative output (DACn_OUTxALT). | Each VDAC channel has three outputs, the main output (VDACn_OUTx), an alternative output (VDACn_OUTxALT), and an APORT output. | Add VDACn_OPAx_APORTREQ and VDACn_OPAx_APORTCONFLICT registers.<br><br>Not all selectable I/O are available on a given device, refer to the device data sheet for details. |
| The drive strength is fixed. | Programmable drive strength. | It is configured by DRIVESTRENGTH field in VDACn_OPAx_CTRL register. |
| Three internal voltage references are available. | Five internal and one external voltage (VDAC0_EXT) references are available. | The new internal references are 1.25 V and 2.5 V low noise bandgap references. |
| The DAC supports three conversion modes. | The DAC supports two conversion modes. | The Sample/Hold mode is removed. |
| Conversion is triggered either by software, PRS, or LESENSE. | Conversion is triggered either by software, PRS, refresh timer, or LESENSE. | The PRS pulse can be from a synchronous or asynchronous PRS producer. |
| **New Features** | | |
| — | Programmable warmup time. | It depends on the selected drive strength. It is configured by WARMUPTIME field in VDACn_OPAx_TIMER register. |
| — | Asynchronous clocking mode. | Uses internal 12 MHz VDAC oscillator to generate DAC_CLK to allow VDAC operation in EM2/3. |
| — | New Interrupt flags for VDAC operation. | These are channel buffer level (CHxBL), channel overflow (CHxOF), and EM23ERR interrupt flags. |
| **Limitations** | | |
| — | — | — |

# 5. Software Migration

The MCU Series 1 devices are software compatible with the existing MCU Series 0 devices, so only minor changes are required for peripherals that are common to MCU Series 0 and MCU Series 1 (especially when enhancements and new features are not used).

### 5.1 Peripheral Support Library (emlib) and energyAware Drivers (emdrv)

The Peripheral Support Library (emlib) abstracts the differences between MCU Series 0 and MCU Series 1 through the API, so software migration becomes transparent to the firmware author.

The emlib modules are found under the Simplicity Studio installation path. The default location on Windows is:

```
C:\SiliconLabs\SimplicityStudio\v4\developer\sdks\gecko_sdk_suite\vX.Y\platform\emlib
```

The energyAware Drivers Library (emdrv) is optimized for speed and power consumption while maintaining API compatibility between different product families.

It is highly recommended to develop peripherals' software with emdrv since it can maintain 100% software compatibility across MCU Series 0 and MCU Series 1.

The available peripherals' emdrv modules are found under the Simplicity Studio installation path. The default location on Windows is:

```
C:\SiliconLabs\SimplicityStudio\v4\developer\sdks\gecko_sdk_suite\vX.Y\platform\emdrv
```

**Table 5.1. Software Migration Checklist**

| Items | Support on emlib & emdrv | Notes |
|---|---|---|
| DCDC power configurations in MCU Series 1. | **emlib:**<br><br>New API for DCDC configuration in em_emu.c. | Call below function after power up to initialize the DCDC (if available) power configuration.<br><br>`bool EMU_DCDCInit(const EMU_DCDCInit_TypeDef *dcdcInit)` |
| HFXO startup initialization in MCU Series 0 and MCU Series 1. | **emlib:**<br><br>Common API for HFXO startup in em_cmu.c. | Call below function to configure HFXO to ensure safe startup for the given crystal.<br><br>`void CMU_HFXOInit(const CMU_HFXOInit_TypeDef *hfxoInit)` |
| LFXO startup initialization in MCU Series 0 and MCU Series 1. | **emlib:**<br><br>Common API for LFXO startup in em_cmu.c. | Call below function to optimize startup time and power consumption for a given low frequency crystal.<br><br>`void CMU_LFXOInit(const CMU_LFXOInit_TypeDef *lfxoInit)` |
| Use HFRCO as HFCLK. | **emlib:**<br><br>Common API in em_cmu.c to setup HFRCO frequency band. | Call below function to change HFRCO frequency band in Series 0.<br><br>`void CMU_HFRCOBandSet(CMU_HFRCOBand_TypeDef band)`<br><br>Call below function to change HFRCO frequency band in Series 1.<br><br>`void CMU_HFRCOBandSet(CMU_HFRCOFreq_TypeDef setFreq)`<br><br>Calibrate HFRCO of MCU Series 1 at startup to match with MCU Series 0 HFRCO frequency band.<br><br>Otherwise manual adjustment for HFRCO frequency dependent parameters are required. |
| Flash startup timing in MCU Series 1. | — | Use default timing or initialize the time to power up Flash on transitions from EM2/3 to EM0. |

| Items | Support on emlib & emdrv | Notes |
|---|---|---|
| Pin enable and location for digital peripherals. | **emlib:**<br><br>The new ROUTEPEN and ROUTELOC definitions of digital peripherals can be found in corresponding emlib header files. | At least two registers (ROUTEPEN and ROUTELOCn) are used for pin enable and location in MCU Series 1 peripherals (only one ROUTE register in MCU Series 0).<br><br>Independent location field for each pin allows pins in peripheral of MCU Series 1 configure to different locations.<br><br>Example for MCU Series 1:<br><br>`USART0->ROUTEPEN = USART_ROUTEPEN_RXPEN \| USART_ROUTEPEN_TXPEN;`<br><br>`USART0->ROUTELOC0 = ( USART0->ROUTELOC0 & ~( _USART_ROUTELOC0_TXLOC_MASK \| _USART_ROUTELOC0_RXLOC_MASK ) ) \| ( USART_ROUTELOC0_TXLOC_LOC0 << _USART_ROUTELOC0_TXLOC_SHIFT ) \| ( USART_ROUTELOC0_RXLOC_LOC0 << _USART_ROUTELOC0_RXLOC_SHIFT );`<br><br>Example for MCU Series 0:<br><br>`USART0->ROUTE = USART_ROUTE_RXPEN \| USART_ROUTE_TXPEN \| USART_ROUTE_LOCATION_LOC0;` |
| APORT for analog peripherals in MCU Series 1. | **emlib:**<br><br>The new APORT definitions of analog peripherals can be found in corresponding emlib header files. | To route signals between analog peripherals and GPIOs.<br><br>Example to use APORT BUS1X channel 6 for ADC input in MCU Series 1:<br><br>`ADC_InitSingle_TypeDef singleInit = ADC_INITSINGLE_DEFAULT;`<br><br>`singleInit.posSel = adcPosSelAPORT1XCH6;`<br><br>Example to use analog channel 1 for ADC input in MCU Series 0:<br><br>`ADC_InitSingle_TypeDef singleInit = ADC_INITSINGLE_DEFAULT;`<br><br>`singleInit.input = adcSingleInputCh1;`<br><br>APORT conflict between analog peripherals can be detected by XXX_APORTCONFLICT registers or APORTCONFLICT interrupt. |
| GPIO | **emlib:**<br><br>Common API for GPIO configuration in em_gpio.c.<br><br>**emdrv:**<br><br>gpiointerrupt | Different GPIO grouping (0-3, 4-7, 8-11, 12-15) in MCU Series 1 to trigger interrupt or to form PRS producer output.<br><br>Pins with the same pin number within a group can now be used as interrupt trigger sources or PRS producer outputs.<br><br>For example below, PC6 (interrupt source 6) and PF6 (interrupt source 4) in group 4-7 can both configure as interrupt sources without conflict.<br><br>`GPIO_PinModeSet(gpioPortC, 6, gpioModeInputPull, 1);`<br><br>`GPIO_ExtIntConfig(gpioPortC, 6, 6, false, true, true);`<br><br>`GPIO_PinModeSet(gpioPortF, 6, gpioModeInputPull, 1);`<br><br>`GPIO_ExtIntConfig(gpioPortF, 6, 4, false, true, true);` |
| Bootloader | — | The preprogrammed bootloader (if available) moves from Main page to Bootloader page in MCU Series 1. |

| Items | Support on emlib & emdrv | Notes |
| --- | --- | --- |
| DAC & VDAC | **emlib:**<br><br>New source file em_vdac.c for VDAC. | Except `VDAC_PrescaleCalc(…)`, the function prototypes of em_dac.c are compatible with em_vdac.c (`DAC_XXX(…)` to `VDAC_XXX(…)`). |
| VCMP & VMON | **emlib:**<br><br>New APIs for VMON in em_emu.c. | The VCMP function in MCU Series 0 is now handled by Voltage Monitor (VMON) in MCU Series 1.<br><br>`void EMU_VmonInit(const EMU_VmonInit_TypeDef *vmonInit)`<br><br>`void EMU_VmonHystInit(const EMU_VmonHystInit_TypeDef *vmonInit)`<br><br>`void EMU_VmonEnable(EMU_VmonChannel_TypeDef channel, bool enable)`<br><br>`bool EMU_VmonChannelStatusGet(EMU_VmonChannel_TypeDef channel)` |
| AES & CRYPTO | **emlib:**<br><br>New source file em_crypto.c for CRYPTO. | The AES function in MCU Series 0 is now handled by Crypto Accelerator (CRYPTO) in MCU Series 1.<br><br>The AES and new cryptographic operations in MCU Series 1 are handled by mbedTLS library.<br><br>The mbedTLS library is found under the Simplicity Studio installation path. The default location on Windows is:<br><br>`C:\SiliconLabs\SimplicityStudio\v4\developer\sdks\gecko_sdk_suite\vX.Y\util\third_party\mbedtls` |
| RTC & RTCC | **emlib:**<br><br>New source file em_rtcc.c for RTCC.<br><br>**emdrv:**<br><br>The rtcdrv now handles both RTC and RTCC. | APIs in em_rtcc.c are not compatible with em_rtc.c.<br><br>The rtcdrv provides common RTC APIs for MCU Series 0 and MCU Series 1. |
| µDMA & LDMA | **emlib:**<br><br>New source file em_ldma.c for LDMA.<br><br>**emdrv:**<br><br>The dmadrv now handles both µDMA and LDMA. | APIs in em_ldma.c are not compatible with em_dma.c (for µDMA in MCU Series 0).<br><br>The dmadrv provides common DMA APIs for MCU Series 0 and MCU Series 1. |

### 5.2 Migration Examples

The examples of this section describe the usage of emlib and emdrv to migrate software from MCU Series 0 to MCU Series 1.

Almost 100% of the code can be reused during migration and just minor modifications need to be made for code running on the new platform.

The hardware differences between platforms will be handled by the peripherals' initialization functions of emlib, it just needs to change the parameters in the function calls or peripherals' initialization structures to fit for the selected device.

Usually nothing needs to be changed when migrating the portion of emdrv, keeps all function calls the same and just modifies the IDE settings for the target device.

### 5.2.1 Migration Example for RTC and ADC

This example configures the RTC to trigger ADC conversion and toggle LED every 500ms.

Source code that runs on EFM32GG STK (EFM32GG_STK3700).

```
#include <stdio.h>
#include "em_device.h"
#include "em_adc.h"
#include "em_chip.h"
#include "em_emu.h"
#include "em_gpio.h"
#include "em_cmu.h"
#include "rtcdriver.h"
/* STK specific LED port and pin */
#define LED_PORT        gpioPortE
#define LED_PIN         2

uint32_t i;
uint32_t sample;
RTCDRV_TimerID_t id;

/***************************************************************************//**
 * @brief RTC Callback used to toggle LED and ADC measurement
 *****************************************************************************/
void rtcCallback( RTCDRV_TimerID_t id, void *user )
{
  /* Start ADC and get result */
  ADC_Start(ADC0, adcStartSingle);
  while (ADC0->STATUS & ADC_STATUS_SINGLEACT)
    ;
  sample = ADC_DataSingleGet(ADC0);

  /* Toggle LED */
  i++;
  if (i & 0x01)
  {
    GPIO_PinOutSet(LED_PORT, LED_PIN);
  }
  else
  {
    GPIO_PinOutClear(LED_PORT, LED_PIN);
  }
}

/***************************************************************************//**
 * @brief Initialize ADC for single conversion
 *****************************************************************************/
void setupAdc(void)
{
  ADC_Init_TypeDef       init       = ADC_INIT_DEFAULT;
  ADC_InitSingle_TypeDef singleInit = ADC_INITSINGLE_DEFAULT;

  /* Init common settings for both single and scan conversion */
  init.timebase = ADC_TimebaseCalc(0);
  init.prescale = ADC_PrescaleCalc(10000000, 0);
  ADC_Init(ADC0, &init);

  /* Init for single conversion */
  singleInit.reference  = adcRefVDD;
  singleInit.input = adcSingleInpCh1;
  ADC_InitSingle(ADC0, &singleInit);
}

/***************************************************************************//**
 * @brief  Main function
 *****************************************************************************/
int main(void)
{
  /* Chip errata */
  CHIP_Init();
```

```
  /* Setup GPIO and ADC. */
  CMU_ClockEnable(cmuClock_GPIO, true);
  GPIO_PinModeSet(LED_PORT, LED_PIN, gpioModePushPull, 0);

  CMU_ClockEnable(cmuClock_ADC0, true);
  setupAdc();

  /* Initialize RTC driver. */
  RTCDRV_Init();
  RTCDRV_AllocateTimer( &id );
  RTCDRV_StartTimer( id, rtcdrvTimerTypePeriodic, 500, rtcCallback, NULL );

  while (1)
  {
    /* Wait in EM1 */
    EMU_EnterEM1();
  }
}
```

To migrate this example from EFM32GG STK to EFM32GG11 STK, following changes are made.

**Table 5.2. Changes for RTC and ADC example migration**

| EFM32GG STK | EFM32GG11 STK | Notes |
|---|---|---|
| No need to setup power configuration. | Need to setup power configuration at startup if device has DC-DC converter. | Call `EMU_DCDCInit(&dcdcInit);` to initialize the power configuration. |
| Use RTC for ADC Trigger and LED toggle. | Use RTCC for ADC trigger and LED toggle. | The rtcdrv is used in this example so the migration from RTC to RTCC is transparent. |
| Use DOUTSET and DOUTCLR registers to set and clear LED pin. | No DOUTSET and DOUTCLR registers to set and clear GPIO pins, use Peripheral Bit Set and Clear to set and clear LED pins. | The `GPIO_PinOutSet()` and `GPIO_PinOutClear()` inline functions in em_gpio.h will handle it so the migration is transparent. |
| LED on pin PE2. | LED on pin PH10. | Change #define statements for LED port (LED_PORT) and pin (LED_PIN) in `GPIO_PinOutSet()`, `GPIO_PinOutClear()` and `GPIO_PinModeSet()`. |
| ADC input channel on PD1. | Use APORT to configure PD1 as ADC input. | Add below parameters to ADC single conversion initialization structure, the APORT definitions are included in em_adc.h.<br>`singleInit.posSel = adcPosSelAPORT0XCH1;` |

Modified source code that runs on EFM32GG11 STK (STK3701_EFM32GG11). In order to run the code on EFM32TG11 STK (SLSTK3301A_EFM32TG11) or EFM32GG12 SLTB (SLTB009A_EFM32GG12), change the LED port and pin definitions as shown in the comment below.

```
#include <stdio.h>
#include "em_device.h"
#include "em_adc.h"
#include "em_chip.h"
#include "em_emu.h"
#include "em_gpio.h"
#include "em_cmu.h"
#include "rtcdriver.h"

/* STK specific LED port and pin */
/*Note
 *
 * For EFM32TG11 STK
 * #define LED_PORT        gpioPortD
 * #define LED_PIN         2
 *
 * For EFM32GG12 SLTB
 * #define LED_PORT        gpioPortA
 * #define LED_PIN         14
```

```
 *
 */
#define LED_PORT        gpioPortH
#define LED_PIN         10

uint32_t i;
uint32_t sample;
RTCDRV_TimerID_t id;

/***************************************************************************//**
 * @brief RTC Callback used to toggle LED and ADC measurement
 ******************************************************************************/
void rtcCallback( RTCDRV_TimerID_t id, void *user )
{
  /* Start ADC and get result */
  ADC_Start(ADC0, adcStartSingle);
  while (ADC0->STATUS & ADC_STATUS_SINGLEACT)
    ;
  sample = ADC_DataSingleGet(ADC0);

  /* Toggle LED */
  i++;
  if (i & 0x01)
  {
    GPIO_PinOutSet(LED_PORT, LED_PIN);
  }
  else
  {
    GPIO_PinOutClear(LED_PORT, LED_PIN);
  }
}

/***************************************************************************//**
 * @brief Initialize ADC for single conversion
 ******************************************************************************/
void setupAdc(void)
{
  ADC_Init_TypeDef       init       = ADC_INIT_DEFAULT;
  ADC_InitSingle_TypeDef singleInit = ADC_INITSINGLE_DEFAULT;

  /* Init common settings for both single and scan conversion */
  init.timebase = ADC_TimebaseCalc(0);
  init.prescale = ADC_PrescaleCalc(10000000, 0);
  ADC_Init(ADC0, &init);

  /* Init for single conversion */
  singleInit.reference  = adcRefVDD;
  singleInit.posSel = adcPosSelAPORT0XCH1;
  ADC_InitSingle(ADC0, &singleInit);
}

/***************************************************************************//**
 * @brief  Main function
 ******************************************************************************/
int main(void)
{
  EMU_DCDCInit_TypeDef dcdcInit = EMU_DCDCINIT_DEFAULT;

  /* Chip errata */
  CHIP_Init();

  /* Init DCDC regulator with specific parameters */
  EMU_DCDCInit(&dcdcInit);

  /* Setup GPIO and ADC. */
  CMU_ClockEnable(cmuClock_GPIO, true);
  GPIO_PinModeSet(LED_PORT, LED_PIN, gpioModePushPull, 0);

  CMU_ClockEnable(cmuClock_ADC0, true);
  setupAdc();

  /* Initialize RTC driver. */
  RTCDRV_Init();
```

```
  RTCDRV_AllocateTimer( &id );
  RTCDRV_StartTimer( id, rtcdrvTimerTypePeriodic, 500, rtcCallback, NULL );

  while (1)
  {
    /* Wait in EM1 */
    EMU_EnterEM1();
  }
}
```

### 5.2.2 Migration Example for USART and DMA

This example configures the DMA to transmit data through USART.

Source code that runs on EFM32GG STK (EFM32GG_STK3700).

```
#include <stdio.h>
#include "em_chip.h"
#include "em_cmu.h"
#include "em_emu.h"
#include "dmadrv.h"
#include "retargetserial.h"

/* STK specific retarget serial output */
#define DMA_USART_SIGNAL dmadrvPeripheralSignal_UART0_TXBL

unsigned int channel;
uint8_t str[] = "Hello DMA !\n";

/***************************************************************************//**
 * @brief  Main function
 ******************************************************************************/
int main( void )
{
  CMU_HFXOInit_TypeDef hfxoInit = CMU_HFXOINIT_DEFAULT;

  /* Chip errata */
  CHIP_Init();

  /* Init HFXO with specific parameters */
  CMU_HFXOInit(&hfxoInit);

  /* Switch HFCLK to HFXO and disable HFRCO */
  CMU_ClockSelectSet(cmuClock_HF, cmuSelect_HFXO);
  CMU_OscillatorEnable(cmuOsc_HFRCO, false, false);

  /* Initialize USART. */
  RETARGET_SerialInit();

  /* Initialize DMA driver. */
  DMADRV_Init();

  /* Request a DMA channel. */
  DMADRV_AllocateChannel( &channel, NULL );

  /* Start the DMA transfer. */
  DMADRV_MemoryPeripheral( channel,
                           DMA_USART_SIGNAL,
                           (void*)&(RETARGET_UART->TXDATA),
                           str,
                           true,
                           sizeof( str ),
                           dmadrvDataSize1,
                           NULL,
                           NULL );

  while (1)
  {
    ;
  }
}
```

To migrate this example from EFM32GG STK to EFM32GG11 STK, following changes are made.

**Table 5.3. Changes for USART and DMA example migration**

| EFM32GG STK | EFM32GG11 STK | Notes |
| --- | --- | --- |
| No need to setup power configuration. | Need to setup power configuration at startup if device has DC-DC converter. | Call `EMU_DCDCInit(&dcdcInit);` to initialize the power configuration. |
| Use µDMA for USART transmit. | Use LDMA for USART transmit. | The dmadrv is used in this example so the migration from µDMA to LDMA is transparent. |
| Use UART0 as retarget serial output. | Use USART0 as retarget serial output. | Change #define statement for USART (DMA_USART_SIGNAL) with DMA in `DMADRV_MemoryPeripheral()`. |

Modified source code that runs on EFM32GG11 STK (STK3701_EFM32GG11). In order to run the code on EFM32TG11 STK (SLSTK3301A_EFM32TG11) or EFM32GG12 SLTB (SLTB009A_EFM32GG12), change the STK specific retarget retarget serial output definitions as shown in the comment below.

```
#include <stdio.h>
#include "em_chip.h"
#include "em_cmu.h"
#include "em_emu.h"
#include "dmadrv.h"
#include "retargetserial.h"

/* STK specific retarget serial output */
/*Note
 *
 * For EFM32TG11 STK
 * #define DMA_USART_SIGNAL dmadrvPeripheralSignal_USART1_TXBL
 *
 * For EFM32GG12 SLTB
 * #define DMA_USART_SIGNAL dmadrvPeripheralSignal_USART0_TXBL
 *
 */
#define DMA_USART_SIGNAL dmadrvPeripheralSignal_USART0_TXBL

unsigned int channel;
uint8_t str[] = "Hello DMA !\n";

/***************************************************************************//**
 * @brief  Main function
 ******************************************************************************/
int main( void )
{
  EMU_DCDCInit_TypeDef dcdcInit = EMU_DCDCINIT_DEFAULT;
  CMU_HFXOInit_TypeDef hfxoInit = CMU_HFXOINIT_DEFAULT;

  /* Chip errata */
  CHIP_Init();

  /* Init DCDC regulator and HFXO with specific parameters */
  EMU_DCDCInit(&dcdcInit);
  CMU_HFXOInit(&hfxoInit);

  /* Switch HFCLK to HFXO and disable HFRCO */
  CMU_ClockSelectSet(cmuClock_HF, cmuSelect_HFXO);
  CMU_OscillatorEnable(cmuOsc_HFRCO, false, false);

  /* Initialize USART. */
  RETARGET_SerialInit();

  /* Initialize DMA driver. */
  DMADRV_Init();

  /* Request a DMA channel. */
  DMADRV_AllocateChannel( &channel, NULL );

  /* Start the DMA transfer. */
```

```
  DMADRV_MemoryPeripheral( channel,
                           DMA_USART_SIGNAL,
                           (void*)&(RETARGET_UART->TXDATA),
                           str,
                           true,
                           sizeof( str ),
                           dmadrvDataSize1,
                           NULL,
                           NULL );

  while (1)
  {
    ;
  }
}
```

# 6. Hardware Migration

## 6.1 Pin Compatibility

The MCU Series 1 devices without DC-DC converter are pin compatible with MCU Series 0 devices and are described in the following table.

PCB layout modifications are required when migrating from MCU Series 0 to MCU Series 1 devices with DC-DC converter.

**Table 6.1. Pin Compatible Devices between MCU Series 0 and MCU Series 1**

| MCU Series 0 | USB/LCD | MCU Series 1 Devices without DC-DC Converter | USB/LCD | Package |
|---|---|---|---|---|
| EFM32xG108FX | —/— | No pin compatible part. | — | QFN24 |
| EFM32xG110FX | —/— | No pin compatible part. | — | QFN24 |
| EFM32xG200FX | —/— | No pin compatible part. | —/— | QFN32 |
| EFM32xG210FX | —/— | No pin compatible part. | —/— | QFN32 |
| EFM32xG222FX | —/— | • EFM32TG11B120F128GQ48 (32 kB RAM)<br>• EFM32TG11B140F64GQ48 (32 kB RAM) | —/— | QFP48 |
| EFM32xG225FX | —/— | No pin compatible part. | — | BGA48 |
| EFM32xG230FX | —/— | • EFM32GG11B110F2048GM64 (384 KB RAM)<br>• EFM32GG11B120F2048GM64 (512 KB RAM)<br>• EFM32TG11B120F128GM64 (32 kB RAM)<br>• EFM32TG11B140F64GM64 (32 kB RAM)<br>• EFM32GG12B110F1024GM64 (192 kB RAM) | —/— | QFN64 |
| EFM32xG232FX | —/— | • EFM32GG11B110F2048GQ64 (384 KB RAM)<br>• EFM32GG11B120F2048GQ64 (512 KB RAM)<br>• EFM32TG11B120F128GQ64 (32 kB RAM)<br>• EFM32TG11B140F64GQ64 (32 kB RAM)<br>• EFM32GG12B110F1024GQ64 (192 kB RAM) | —/— | QFP64 |
| EFM32xG280FX | —/— | • EFM32GG11B310F2048GQ100 (384 KB RAM)<br>• EFM32GG11B320F2048GQ100 (512 KB RAM)<br>• EFM32GG12B310F1024GQ100 (192 kB RAM)<br>• EFM32GG12B330F512GQ100 (192 kB RAM) | —/Y | QFP100 |
| EFM32xG290FX | —/— | • EFM32GG11B310F2048GL112 (384 KB RAM)<br>• EFM32GG11B320F2048GL112 (512 KB RAM)<br>• EFM32GG12B310F1024GL112 (192 kB RAM)<br>• EFM32GG12B330F512GL112 (192 kB RAM) | —/Y | BGA112 |
| EFM32xG295FX | —/— | • EFM32GG11B420F2048GL120 (512 KB RAM)<br>• EFM32GG12B410F1024GL120 (192 kB RAM)<br>• EFM32GG12B430F512GL120 (192 kB RAM) | Y/Y | BGA120 |
| EFM32xG308FX | Y/— | No pin compatible part. | — | QFN24 |
| EFM32xG309FX | Y/— | No pin compatible part. | — | QFN24 |
| EFM32xG310FX | Y/— | No pin compatible part. | — | QFN32 |
| EFM32xG321FX | Y/— | No pin compatible part. | | QFP48 |
| EFM32xG322FX | Y/— | No pin compatible part. | | QFP48 |
| EFM32xG330FX | Y/— | No pin compatible part. | — | QFN64 |

| MCU Series 0 | USB/LCD | MCU Series 1 Devices without DC-DC Converter | USB/LCD | Package |
|---|---|---|---|---|
| EFM32xG332FX | Y/— | No pin compatible part. | — | QFP64 |
| EFM32xG350FX | Y/— | No pin compatible part. | — | CSP36 |
| EFM32xG360FX | Y/— | No pin compatible part. | — | CSP81 |
| EFM32xG380FX | Y/— | • EFM32GG11B420F2048GQ100 (512 KB RAM)<br>• EFM32GG12B410F1024GQ100 (192 kB RAM)<br>• EFM32GG12B430F512GQ100 (192 kB RAM) | Y/Y | QFP100 |
| EFM32xG390FX | Y/— | • EFM32GG11B420F2048GL112 (512 KB RAM)<br>• EFM32GG12B410F1024GL112 (192 kB RAM)<br>• EFM32GG12B430F512GL112 (192 kB RAM) | Y/Y | BGA112 |
| EFM32xG395FX | Y/— | • EFM32GG11B420F2048GL120 (512 KB RAM)<br>• EFM32GG12B410F1024GL120 (192 kB RAM)<br>• EFM32GG12B430F512GL120 (192 kB RAM) | Y/Y | BGA120 |
| EFM32xG822FX | —/Y | • EFM32TG11B320F128GQ48 (32 kB RAM)<br>• EFM32TG11B340F64GQ48 (32 kB RAM) | —/Y | QFP48 |
| EFM32xG825FX | —/Y | No pin compatible part. | — | BGA48 |
| EFM32xG840FX | —/Y | • EFM32TG11B320F128GM64 (32 kB RAM)<br>• EFM32TG11B340F64GM64 (32 kB RAM) | —/Y | QFN64 |
| EFM32xG842FX | —/Y | • EFM32TG11B320F128GQ64 (32 kB RAM)<br>• EFM32TG11B340F64GQ64 (32 kB RAM) | —/Y | QFP64 |
| EFM32xG880FX | —/Y | • EFM32GG11B310F2048GQ100 (384 KB RAM)<br>• EFM32GG11B320F2048GQ100 (512 KB RAM)<br>• EFM32GG12B310F1024GQ100 (192 kB RAM)<br>• EFM32GG12B330F512GQ100 (192 kB RAM) | —/Y | QFP100 |
| EFM32xG890FX | —/Y | • EFM32GG11B310F2048GL112 (384 KB RAM)<br>• EFM32GG11B320F2048GL112 (512 KB RAM)<br>• EFM32GG12B310F1024GL112 (192 kB RAM)<br>• EFM32GG12B330F512GL112 (192 kB RAM) | —/Y | BGA112 |
| EFM32xG895FX | —/Y | • EFM32GG11B420F2048GL120 (512 KB RAM)<br>• EFM32GG12B410F1024GL120 (192 kB RAM)<br>• EFM32GG12B430F512GL120 (192 kB RAM) | Y/Y | BGA120 |
| EFM32xG940FX | Y/Y | • EFM32GG11B420F2048GM64 (512 KB RAM)<br>• EFM32GG12B410F1024GM64 (192 kB RAM)<br>• EFM32GG12B430F512GM64 (192 kB RAM) | Y/Y | QFN64 |
| EFM32xG942FX | Y/Y | • EFM32GG11B420F2048GQ64 (512 KB RAM)<br>• EFM32GG12B410F1024GQ64 (192 kB RAM)<br>• EFM32GG12B430F512GQ64 (192 kB RAM) | Y/Y | QFP64 |
| EFM32xG980FX | Y/Y | • EFM32GG11B420F2048GQ100 (512 KB RAM)<br>• EFM32GG12B410F1024GQ100 (192 kB RAM)<br>• EFM32GG12B430F512GQ100 (192 kB RAM) | Y/Y | QFP100 |
| EFM32xG990FX | Y/Y | • EFM32GG11B420F2048GL112 (512 KB RAM)<br>• EFM32GG12B410F1024GL112 (192 kB RAM)<br>• EFM32GG12B430F512GL112 (192 kB RAM) | Y/Y | BGA112 |

| MCU Series 0 | USB/LCD | MCU Series 1 Devices without DC-DC Converter | USB/LCD | Package |
|---|---|---|---|---|
| EFM32xG995FX | Y/Y | • EFM32GG11B420F2048GL120 (512 kB RAM)<br>• EFM32GG12B410F1024GL120 (192 kB RAM)<br>• EFM32GG12B430F512GL120 (192 kB RAM) | Y/Y | BGA120 |
| **Note:** Pin assignments, definitions, and PCB layout of pin-compatible devices should not need to change; however, the electrical specifications for the pins (drive strength, slew rate, power consumption) may be different. | | | | |

### 6.2 Peripherals Only on MCU Series 0

The following table describes how to migrate MCU Series 0 peripherals that are not available in MCU Series 1.

**Table 6.2. MCU Series 0-Only Peripherals Migration**

| MCU Series 0 | MCU Series 1 | Notes |
|---|---|---|
| **Energy Management** | | |
| Voltage Comparator (VCMP) | Voltage Monitor in EMU or Analog Comparator (ACMP) | Hardware change is not required.<br>Voltage Monitor can run down to EM4H, and ACMP can run down to EM3. |
| **Security** | | |
| Advanced Encryption Standard Accelerator (AES) | Crypto Accelerator (CRYPTO) | Hardware change is not required. |
| **Timers and Triggers** | | |
| Backup Real Time Counter (BURTC) and Retention Registers (512 bytes) | Real Time Counter and Calendar (RTCC) and Retention Registers | Hardware change is not required.<br>The size of the Retention Registers in the RTCC is 128 bytes. |
| **Note:** There are no LCD_BCAP_P, LCD_BCAP_N and USB_DMPU pins on MCU Series 1 devices. | | |

# 7. Revision History

**Revision 0.02**

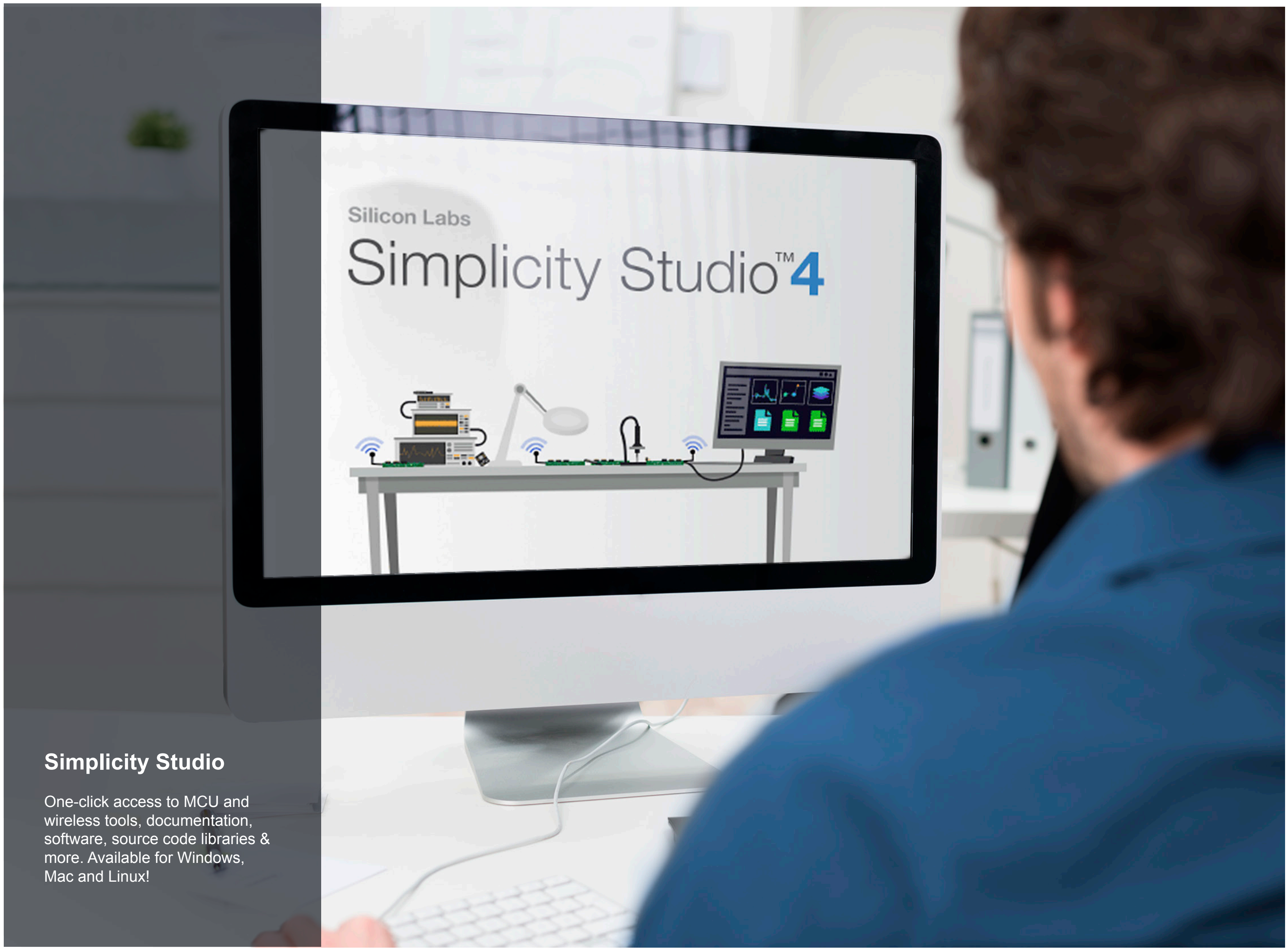
December, 2018

- Added EFM32TG11 and EFM32GG12 to MCU Series 1 in 1. Device Compatibility
- Replaced EFM32GG11 with MCU Series 1 to include EFM32TG11 and EFM32GG12 throughout the document as necessary.
- Added PDM to and changed footnote in Table 2.1 The emlib and emdrv Support for MCU Series 0 and MCU Series 1 on page 3 table.
- Added columns for EFM32TG11 and EFM32GG12 in the Table 3.1 Core and Memory on page 5 table.
- Updated the Table 3.3 New Peripherals in MCU Series 1 on page 7 table and added footnotes.
- Updated the Table 4.1 Common Peripherals between MCU Series 0 and MCU Series 1 on page 8 table by adding EFM32TG11 and EFM32GG11 columns.
- Updated the Table 4.2 DBG on page 9 table to apply to all MCU Series 1 devices.
- Updated the Table 4.3 Memory and Bus System on page 10 table to apply to all MCU Series 1 devices and added a footnote.
- Updated the Table 4.4 MSC on page 11 table to apply to all MCU Series 1 devices and added a footnote.
- Updated the Table 4.5 CMU on page 13 table to apply to all MCU Series 1 devices and added footnotes.
- Updated the Table 4.6 EMU on page 15 table to apply to all MCU Series 1 devices and added a footnote.
- Updated the Table 4.7 RMU on page 16 table to apply to all MCU Series 1 devices.
- Updated the Table 4.8 I2C on page 17 table to apply to all MCU Series 1 devices.
- Updated the Table 4.11 USB on page 19 table to apply to EFM32GG11 and EFM32GG12.
- Moved one of the Limitations in the Table 4.12 GPIO on page 20 table to Enhancements.
- Updated the Table 4.13 EBI on page 21 table to apply to EFM32GG11 and EFM32GG12.
- Updated the Table 4.15 RTC on page 22 table to apply to EFM32GG11 and EFM32GG12.
- Updated the Table 4.17 PRS on page 23 table to apply to all MCU Series 1 devices.
- Changed the number of capacitive sense mode inputs to 8 in the Table 4.21 ACMP on page 26 table.
- Moved two Limitations in the Table 4.22 ADC on page 27 table to Enhancements.
- Updated the Table 4.23 IDAC on page 28 table to apply to EFM32GG11 and EFM32GG12.
- Updated the external capacitor value in the Table 4.24 LCD on page 29 table.
- Added GPIO port pin definitions for EFM32TG11 and EFM32GG12 in 5.2.1 Migration Example for RTC and ADC.
- Added STK specific retarget serial output for EFM32TG11 and EFM32GG12 in 5.2.2 Migration Example for USART and DMA.
- Updated 6.1 Pin Compatibility to include all the MCU Series 0 OPNs and added pin compatible EFM32TG11 and EFM32GG12 devices.
- Corrected typos throughout the document.

**Revision 0.01**

June, 2017

- Initial revision




| IoT Portfolio | SW/HW | Quality | Support and Community |
|---|---|---|---|
| www.silabs.com/IoT | www.silabs.com/simplicity | www.silabs.com/quality | community.silabs.com |



SILICON LABS

**Silicon Laboratories Inc.**
**400 West Cesar Chavez**
**Austin, TX 78701**
**USA**

**http://www.silabs.com**