AN1377: EFR32xG21 Revision B to Revision C+ Compatibility and Migration Guide

This porting guide is for users migrating an existing EFR32BG21 or EFR32MG21 design from revision B to revision C+, hereafter respectively referred to as EFR32xG21-B and EFR32xG21-C+. Migration entails both hardware and software considerations.

KEY POINTS

- The EFR32xG21-C+ maintains a high degree of pin and feature compatibility with the EFR32xG21-B.
- Migrating designs from EFR32xG21-B to EFR32xG21-C+ requires no hardware changes and minimal software changes.
- Data sheets for each device detail specific performance metrics, which may differ between EFR32xG21-B and EFR32xG21-C+.

1. Introduction

This application note highlights the differences between EFR32xG21-B and EFR32xG21-C+.

EFR32xG21-B devices include:

- EFR32xG21-B

EFR32xG21-C+ devices include:

- EFR32xG21-C
- EFR32xG21-D
- EFR32MR21-C

The EFR32xG21-C+ device was designed to be both pin- and hardware-compatible, as well as largely code-compatible, with EFR32xG21-B devices, thus requiring only minor changes when porting designs. Refer to the device-specific data sheets for performace metrics that may differ between the EFR32xG21-B and EFR32xG21-C+.

2. Pin Compatibility

The EFR32xG21-B and EFR32xG21-C+ are pin-compatible. No PCB redesign is necessary when porting designs between the two devices. The mapping of all device supply and ground pins, as well as the radio interface, crystal, and reset pins, is unchaged. Similarly, all GPIO pins remain mapped to the same pin numbers, and the multiplexing of analog and digital peripheral functionality remains the same between the two versions. Both devices are only available in the same QFN32 package, so the dimensions and land pattern are unchanged. See the device-specific data sheets for pin definitions and package specifications.

3. Hardware Compatibility

EFR32xG21-B and EFR32xG21-C+ are designed to be hardware compatible. This means that both devices have the same power supply requirements and are generally compatible with the same crystals. None of the passive components used on a PCB originally designed for EFR32xG21-B have to be changed when populated with EFR32xG21-C+. Refer to AN0002.2 - EFM32 and EFR32 Wireless Gecko Series 2 Hardware Design Considerations for hardware considerations for Series 2 devices.

4. Peripheral Compatibility

Both the EFR32xG21-B and EFR32xG21-C+ contain the same peripherals across devices. However, there are a few changes in the following peripherals that are worth noting when migrating designs. Refer to the EFR32xG21 Reference Manual for the full list of peripherals available.

4.1 EMU

Wake-up and entry times are increased for EM2, EM3, and EM4 on EFR32xG21-C+; they are unchanged for EM1. Typical values can be found in the specific device data sheets.

For EFR32xG21-C+, power-on-resets and startup times are also increased.

4.2 Flash

The physical flash memory differs between EFR32xG21-B and EFR32xG21-C+ devices as follows:

- While the 8 kB page size remains the same, erase times are increased on EFR32xG21-C+. Details are covered in the specific data sheets for each device. The increased page erase time impacts SDK components, such as NVM3, and may require optimizations in timing critical applications. Firmware upgrades using Gecko Bootloader, including over-the-air image downloads to internal storage, may take longer to complete.
- Flash word programming times are decreased on EFR32xG21-C+. Details are covered in the specific data sheets for each device. The reduced programming time may impact the performance of applications that have a dependence on writing data to flash.

4.3 IADC

The voltage of the internal bandgap reference has changed between EFR32xG21-C+ and EFR32xG21-B. Table 4.1 IADC internal bandgap reference on page 6 lists the internal reference voltage between device revisions.

Table 4.1. IADC internal bandgap reference

Device	Internal reference voltage
EFR32xG21-B	1.21V
EFR32xG21-C	1.23V
EFR32xG21-D	1.18V
EFR32MR21-C	N/A, no IADC on this device

Analog-to-digital conversions made using the internal reference voltage must account for this change when porting software from EFR32xG21-B to EFR32xG21-C+.

For example, if the result of a conversion is expressed as a voltage as shown below...

```
// Raw IADC conversion result
static volatile IADC_Result_t sample;

// Result converted to volts
static volatile double singleResult;

// Read a result from the FIFO
sample = IADC_pullSingleFifoResult(IADC0);

// Convert the single-ended conversion result to a voltage
singleResult = sample.data * 1.21 / 0xFFF;
```

...then the last line would need to change for EFR32xG21-C+ as follows:

```
singleResult = sample.data * 1.23 / 0xFFF;  // For EFR32xG21-C
// OR
singleResult = sample.data * 1.18 / 0xFFF;  // For EFR32xG21-D
```

Gecko SDK releases that support EFR32xG21-C+ provide an API to assist with reference voltage determination that is discussed in 5.1.2 `em_iadc.c` and `em_iadc.h`.

Failure to account for the reference voltage change could cause firmware to behave incorrectly, even when the binary output from the IADC is used without converting it to a voltage. Raw conversion results on EFR32xG21-C+ will be slightly different compared with those returned by an EFR32xG21-B device in the same circuit because of the full scale voltage difference. The EFR32xG21-C devices will have a slightly lower raw conversion result than EFR32xG21-B because it has a slightly higher internal reference voltage. The EFR32xG21-D, on the other hand, will have a slightly higher raw conversion result than EFR32xG21-B because it has a slightly lower internal reference voltage. For example, 0.8 V would equate to a nominal conversion result of 2663 (0xA67) on EFR32xG21-C vs. 2707 (0xA93) on EFR32xG21-B vs. 2776 (0xAD8) on EFR32xG21-D .

The behavior of the IADC is not impacted if an external reference or one of the AVDD reference options is used.

4.4 LFRCO

An oscillator is said to be "requested" on EFR32xG21 when it is selected to provide the clock either for a specific peripheral or clock tree branch. A requested oscillator is started on-demand and remains active until it is no longer requested (e.g. the peripheral is disabled or, in the case of high-frequency clocks, the device enters a low-energy mode).

In this sense, it is important to understand that the LFRCO is always requested by the MSC in EM0 and EM1 on EFR32xG21-C+ to time flash erase operations. This differs from EFR32xG21-B, which used the FSRCO for this purpose. There are two ways in which this can impact firmware written originally for EFR32xG21-B that is intended to run on EFR32xG21-C+.

Because the LFRCO is always requested in EM0 and EM1 and, therefore, always running, the `LFRCO->STATUS` register `ENS` and `RDY` bits will always read as 1. If the LFRCO is used as the clock for a low-frequency peripheral (e.g., the LETIMER), then firmware that waits for the LFRCO to be ready (e.g., waits until the `RDY` status bit is set) before using said peripheral will experience no delay.

The absence of this delay might have unintended side effects. In particular, if the LFRCO ready interrupt (`LFRCO->IF.RDY`) is used, it will be requested immediately once it is enabled (`LFRCO->IEN.RDY` is set) and unmasked (`NVIC_EnableIRQ(LFRCO_IRQn)` is called). In rare cases, depending on how firmware is structured, use of this interrupt might prevent code that normally executes during the LFRCO start-up delay on EFR32xG21-B from executing at all on EFR32xG21-C+.

Care must also be taken with respect to calibration of the LFRCO on EFR32xG21-C+. The `LFRCO->CAL` register permits trimming of the LFRCO frequency and is typically used to maintain frequency stability over temperature through periodic recalibration.

However, the ability to trim the LFRCO can also be used in a limited way to tune it to an altogether different center frequency. While this presents no issue on EFR32xG21-B, it is not permissible on EFR32xG21-C+ when flash must be erased. The LFRCO must remain tuned to its nominal frequency of 32.768 kHz to guarantee proper flash erase functionality. Note that this does not preclude writing to `LFRCO->CAL` for calibration purposes so long as the 32.768 kHz frequency is maintained.

4.5 LFXO

The LFXO has programmable tuning capacitors on-chip that eliminate the need for the typical external C_L load capacitors required in most oscillator circuits. The minimum value for these tuning capacitors differs between EFR32xG21-B and EFR32xG21-C+ as shown below:

Table 4.2. Minimum LFXO On-Chip Tuning Capacitor Values

Device	C_{LFXO_MIN} (pF)
EFR32xG21-B	4
EFR32xG21-C	6
EFR32xG21-D	4
EFR32MR21-C	6

4.6 MSC

To accommodate the underlying physical flash memory, the Memory System Controller (MSC) on EFR32xG21-C+ can only support either a read from or a write to flash at any given time. For this reason, firmware must observe the following restrictions when writing to or erasing flash:

1. Any function that writes to or erases flash must be located in RAM.
2. Data written to flash must also be located in RAM.

Note: The requirement that data written to flash must reside in RAM applies regardless of whether the CPU or the LDMA is ultimately storing the data to the MSC Write Data register (`MSC->WDATA`).

While the MSC logically organizes flash memory as 64-bit double words on both versions of EFR32xG21, there are different limitations on how many times a memory location can be programmed before it must be erased.

On EFR32xG21-B, each 64-bit double word can be written twice between erase cycles. This means that...

- the lower and upper 32-bit words may each be written once in any order or
- one of the words may be written twice while the other is not written at all,

However, once two writes have occurred within this 64-bit double word, the page in which it is contained must be erased before another write is permitted.

This doubling of writes is not permitted on EFR32xG21-C+. Once a given 32-bit word has been written, it cannot be written again until the page in which it resides has been erased.

Because the low-level interface between the MSC and the physical flash has been changed on EFR32xG21-C+, a read of the `MSC->IPVERSION` register returns 0x6. The value returned on EFR32xG21-B is 0x0.

4.7 SYSCFG

Reflecting the fact that EFR32xG21-B and EFR32xG21-C+ are two different versions of the EFR32xG21 device, reads of the `SYSCFG->CHIPREVHW` (offset 0x10) and `SYSCFG->CHIPREV` (offset 0x14) registers return different values. Both registers hold the same `MINOR`, `FAMILY`, and `MAJOR` bit fields shown below.

For EFR32xG21-B:

	Bit Position																													
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x1								0x30						0x1					
Access													RW								RW						RW					
Name													MINOR								FAMILY						MAJOR					

For EFR32xG21-C+:

	Bit Position																													
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0								0x30						0x2					
Access													RW								RW						RW					
Name													MINOR								FAMILY						MAJOR					

5. Software Compatibility

EFR32xG21-C+ is specifically supported by patch releases of the following versions of Gecko SDK:

- 2.7.11
- 3.1.3
- 3.2.4
- 4.1

A very high degree of software compatibility is retained between EFR32xG21-B and EFR32xG21-C+. However, in all cases, firmware written originally for EFR32xG21-B must be recompiled against the patch release of the relevant SDK listed above in order for it to function properly on EFR32xG21-C+. Excluding possible performance differences and changes that may be required because of specific peripheral differences, the recompiled firmware can be executed on both EFR32xG21-B and EFR32xG21-C+ devices.

The sections that follow discuss specific components of Gecko SDK and how they are impacted by executing on the EFR32xG21-C+ device.

5.1 EMLIB

Because emlib is a peripheral abstraction API, it handles the small differences between individual devices, including specific revisions, in a particular series of EFM32/EFR32 family members. Certain emlib subcomponents are impacted by the changes introduced on EFR32xG21-C+.

5.1.1 `em_emu.c` and `em_emu.h`

None of the emlib EMU APIs have been functionally changed to accommodate EFR32xG21-C+. However, as noted in 4.1 EMU, entry and wake-up times for EM2, EM3, and EM4 are increased, and this will appear when measuring the execution times of `EMU_EnterEM2()`, `EMU_EnterEM3()`, and `EMU_EnterEM4()`.

5.1.2 `em_iadc.c` and `em_iadc.h`

To handle the bandgap reference voltage difference between EFR32xG21-B and EFR32xG21-C+, a new API has been added to emlib:

```
uint32_t IADC_getReferenceVoltage(IADC_CfgReference_t reference)
```

Firmware should make use of this API both when initializing the IADC and in calculations which convert IADC results to voltages instead of using a hardcoded value for the bandgap reference voltage. Consider the following initialization code segment:

```
// Declare initialization structures
IADC_Init_t init = IADC_INIT_DEFAULT;
IADC_AllConfigs_t initAllConfigs = IADC_ALLCONFIGS_DEFAULT;

// Shutdown between conversions to reduce current
init.warmup = iadcWarmupNormal;

/*
 * Configuration 0 is used for both scan and single conversions
 * by default.  Use the internal bandgap as the reference.
 */
initAllConfigs.configs[0].reference = iadcCfgReferenceInt1V2;

// Initialize IADC
IADC_init(IADC0, &init, &initAllConfigs);
```

Use of the `IADC_ALLCONFIGS_DEFAULT` initializer results in a hardcoded value of 1210 mV for the `.vRef` member of the initialization structure, which would generate slightly inaccurate conversion results when running on the EFR32xG21-C+ device because the value of the bandgap reference is 1232 mV on EFR32xG21-C and 1180 mV on EFR32xG21-D. The new `IADC_getReferenceVoltage()` function can be used here to set `.vRef` to the specifc value for either EFR32xG21-B or EFR32xG21-C+ as follows:

```
// Declare initialization structures
IADC_Init_t init = IADC_INIT_DEFAULT;
IADC_AllConfigs_t initAllConfigs = IADC_ALLCONFIGS_DEFAULT;

// Shutdown between conversions to reduce current
init.warmup = iadcWarmupNormal;

/*
 * Configuration 0 is used for both scan and single conversions
 * by default.  Use the internal bandgap as the reference and
 * specify its value in mV.
 */
initAllConfigs.configs[0].reference = iadcCfgReferenceInt1V2;
initAllConfigs.configs[0].vRef = IADC_getReferenceVoltage(iadcCfgReferenceInt1V2);

// Initialize IADC
IADC_init(IADC0, &init, &initAllConfigs);
```

Note, however, that even if firmware fails to explicitly set `.vRef` to the appropriate value, `IADC_init()` will account for this in the updated releases of Gecko SDK that support EFR32xG21-C+ because the specific value of the bandgap reference must be used when applying the IADC gain and offset calibration values from the DEVINFO page to the `IADC->SCALEx` registers.

As discussed in 4.3 IADC, results converted to voltages will be incorrect on EFR32xG21-C+ if the correct value of the bandgap reference is not used. Fortunately, this is readily solved by using the `IADC_getReferenceVoltage()` function in a way that allows the resulting code to run correctly on any revision of EFR32xG21 as shown below:

```
// Raw IADC conversion result
static volatile IADC_Result_t sample;

// Result converted to volts
static volatile double singleResult;

// Read a result from the FIFO
sample = IADC_pullSingleFifoResult(IADC0);

// Convert the single-ended conversion result to a voltage
singleResult = (sample.data * IADC_getReferenceVoltage() / 1000) / 0xFFF;
```

When converting a result to volts (`singleResult` is a floating point variable in this example), the value returned by `IADC_getReferenceVoltage()` is in millivolts and must be divided by 1000. If results are handled in millivolts only (e.g. to use only integer math), then division by 1000 is not necessary.

5.1.3 `em_msc.c` and `em_msc.h`

None of the emlib MSC APIs have been functionally changed to accommodate EFR32xG21-C+ because EFR32xG21-B already requires code that writes to or erases flash to execute from RAM. However, as noted in 4.2 Flash, write times are reduced and page erase times are increased. This will be observed if the execution times of `MSC_WriteWord()`, `MSC_WriteWordDMA()`, `MSC_ErasePage()`, and `MSC_MassErase()` are measured.

5.1.4 `em_system.c` and `em_system.h`

Reads of the chip revision information registers in the SYSCFG module return different values on EFR32xG21-B and EFR32xG21-C+ as explained in 4.7 SYSCFG. Consequently, it may be necessary to account for different values in the `.major` and `.minor` fields of the `SYSTEM_ChipRevision_TypeDef` structure returned by the `SYSTEM_ChipRevisionGet()` function depending on whether firmware is executing on EFR32xG21-B or EFR32xG21-C+.

5.2 Gecko Bootloader

The functionality of Gecko Bootloader is unchanged on EFR32xG21-C+. However, as noted in 4.1 EMU, start-up times are increased. Typical values can be found in the device-specific data sheet.

Similarly, because flash page erase times are increased on EFR32xG21-C+ (see 4.2 Flash), firmware upgrades will take more time depending on the number of pages that must be erased.

It may be necessary to consider firmware optimizations in specific applications where the increased start-up and/or upgrade times might prevent a device from being ready to process radio traffic as soon as is normally expected.

5.3 NVM3

The functionality of NVM3 is unchanged on EFR32xG21-C+. Consequently, there is no need to change the number of flash pages allocated to NVM3 in any given application. However, the changes in flash write and erase timing will be visible when objects are written to NVM3. In general, object writes can be expected to be slightly faster because writes times are reduced on EFR32xG21-C+.

Because page erase times are increased on EFR32xG21-C+, repack operations, whether initiated by firmware during periods of inactivity or by NVM3 when space must be freed to store an updated object, will also require more time. Applications in which radio activity is possible while an NVM3 object is being updated should be tested to ensure that the firmware behaves as expected when a repack occurs.

6. Revision History

Revision 0.2

July, 2023

- Updates EFR32xG21-C to EFR32xG21-C+.
- Includes EFR32xG21-D and EFR32MR21-C in EFR32xG21-C+.
- Updates IADC internal reference voltage for EFR32xG21-D in 4.3 IADC.
- Minimum LFXO on-chip tuning capacitor values added in 4.5 LFXO.
- Numerous style edits throughout.

Revision 0.1

March, 2022

- Initial revision.

Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

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

www.silabs.com