



Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!



Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832

Email & Skype: info@chipsmall.com Web: www.chipsmall.com

Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China



EFM32ZG Reference Manual

Zero Gecko Series

- 32-bit ARM Cortex-M0+ processor running at up to 24 MHz
- Up to 32 kB Flash and 4 kB RAM memory
- Energy efficient and autonomous peripherals
- Ultra low power Energy Modes with sub- μ A operation
- Fast wake-up time of only 2 μ s

The EFM32ZG microcontroller series revolutionizes the 8- to 32-bit market with a combination of unmatched performance and ultra low power consumption in both active- and sleep modes. EFM32ZG devices consume as little as 114 μ A/MHz in run mode.

EFM32ZG's low energy consumption outperforms any other available 8-, 16-, and 32-bit solution. The EFM32ZG includes autonomous and energy efficient peripherals, high overall chip- and analog integration, and the performance of the industry standard 32-bit ARM Cortex-M0+ processor.

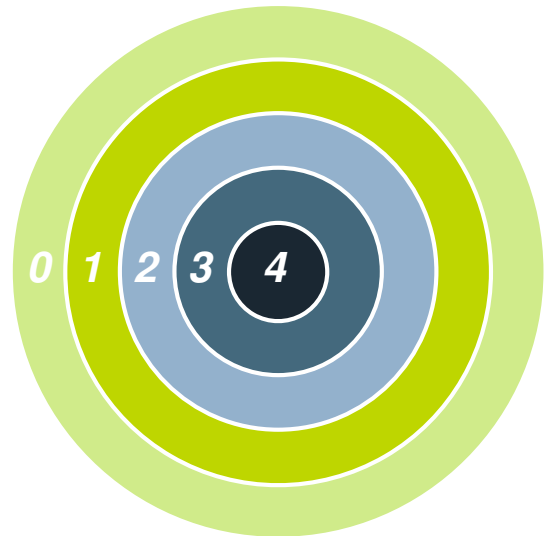
1 Energy Friendly Microcontrollers

1.1 Typical Applications

The EFM32ZG Zero Gecko is the ideal choice for demanding 8-, 16-, and 32-bit energy sensitive applications. These devices are developed to minimize the energy consumption by lowering both the power and the active time, over all phases of MCU operation. This unique combination of ultra low energy consumption and the performance of the 32-bit ARM Cortex-M0+ processor, help designers get more out of the available energy in a variety of applications.

Ultra low energy EFM32ZG microcontrollers are perfect for:

- Gas metering
- Energy metering
- Water metering
- Smart metering
- Alarm and security systems
- Health and fitness applications
- Industrial and home automation



1.2 EFM32ZG Development

Because EFM32ZG use the Cortex-M0+ CPU, embedded designers benefit from the largest development ecosystem in the industry, the ARM ecosystem. The development suite spans the whole design process and includes powerful debug tools, and some of the world's top brand compilers. Libraries with documentation and user examples shorten time from idea to market.

The range of EFM32ZG devices ensure easy migration and feature upgrade possibilities.

2 About This Document

This document contains reference material for the EFM32ZG series of microcontrollers. All modules and peripherals in the EFM32ZG series 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 pin-out, are covered in the device-specific datasheets.

2.1 Conventions

Register Names

Register names are given as a module name prefix followed by the short register name:

TIMERn_CTRL - Control Register

The "n" denotes the numeric instance for modules that might have more than one instance.

Some registers are grouped which leads to a group name following the module prefix:

GPIO_Px_DOUT - Port Data Out Register,

where x denotes the port instance (A,B,...).

Bit Fields

Registers contain one or more bit fields which can be 1 to 32 bits wide. Multi-bit fields are denoted with (x:y), where x is the start bit and y is the end bit.

Address

The address for each register can be found by adding the base address of the module (found in the Memory Map), and the offset address for the register (found in module Register Map).

Access Type

The register access types used in the register descriptions are explained in Table 2.1 (p. 3) .

Table 2.1. Register Access Types

Access Type	Description
R	Read only. Writes are ignored.
RW	Readable and writable.
RW1	Readable and writable. Only writes to 1 have effect.
RW1H	Readable, writable and updated by hardware. Only writes to 1 have effect.
W1	Read value undefined. Only writes to 1 have effect.
W	Write only. Read value undefined.
RWH	Readable, writable and updated by hardware.

Number format

0x prefix is used for hexadecimal numbers.

0b prefix is used for binary numbers.

Numbers without prefix are in decimal representation.

Reserved

Registers and bit fields marked with *reserved* are reserved for future use. These should be written to 0 unless otherwise stated in the Register Description. Reserved bits might be read as 1 in future devices.

Reset Value

The reset value denotes the value after reset.

Registers denoted with X have an unknown reset value and need to be initialized before use. Note that, before these registers are initialized, read-modify-write operations might result in undefined register values.

Pin Connections

Pin connections are given as a module prefix followed by a short pin name:

USn_TX (USARTn TX pin)

The pin locations referenced in this document are given in the device-specific datasheet.

2.2 Related Documentation

Further documentation on the EFM32ZG family and the ARM Cortex-M0+ can be found at the Silicon Laboratories and ARM web pages:

www.silabs.com

www.arm.com

3 System Overview

3.1 Introduction

The EFM32 MCUs are the world's most energy friendly microcontrollers. With a unique combination of the powerful 32-bit ARM Cortex-M0+, innovative low energy techniques, short wake-up time from energy saving modes, and a wide selection of peripherals, the EFM32ZG microcontroller is well suited for any battery operated application, as well as other systems requiring high performance and low-energy consumption, see Figure 3.1 (p. 6) .

3.2 Features

- **ARM Cortex-M0+ CPU platform**
 - High Performance 32-bit processor @ up to 24 MHz
 - Wake-up Interrupt Controller
- **Flexible Energy Management System**
 - 20 nA @ 3 V Shutoff Mode
 - 0.5 μ A @ 3 V Stop Mode, including Power-on Reset, Brown-out Detector, RAM and CPU retention
 - 0.9 μ A @ 3 V Deep Sleep Mode, including RTC with 32768 Hz oscillator, Power-on Reset, Brown-out Detector, RAM and CPU retention
 - 48 μ A/MHz @ 3 V Sleep Mode
 - 114 μ A/MHz @ 3 V Run Mode, with code executed from flash
- **32/16/8/4 KB Flash**
- **4/2 KB RAM**
- **Up to 37 General Purpose I/O pins**
 - Configurable push-pull, open-drain, pull-up/down, input filter, drive strength
 - Configurable peripheral I/O locations
 - 16 asynchronous external interrupts
 - Output state retention and wake-up from Shutoff Mode
- **4 Channel DMA Controller**
 - Alternate/primary descriptors with scatter-gather/ping-pong operation
- **4 Channel Peripheral Reflex System**
 - Autonomous inter-peripheral signaling enables smart operation in low energy modes
- **Hardware AES with 128-bit Keys in 54 cycles**
- **Communication interfaces**
 - 1 \times Universal Synchronous/Asynchronous Receiver/Transmitter
 - Triple buffered full/half-duplex operation
 - 4-16 data bits
 - 1 \times Low Energy UART
 - Autonomous operation with DMA in Deep Sleep Mode
 - 1 \times I²C Interface with SMBus support
 - Address recognition in Stop Mode
- **Timers/Counters**
 - 2 \times 16-bit Timer/Counter
 - 3 Compare/Capture/PWM channels
 - 24-bit Real-Time Counter
 - 1 \times 16-bit Pulse Counter
 - Asynchronous pulse counting/quadrature decoding
 - Watchdog Timer with dedicated RC oscillator @ 50 nA
- **Ultra low power precision analog peripherals**
 - 12-bit 1 Msamples/s Analog to Digital Converter

- 8 input channels and on-chip temperature sensor
- Single ended or differential operation
- Conversion tailgating for predictable latency
- Current Digital to Analog Converter
 - Source or sink a configurable constant current
- 1× Analog Comparator
 - Programmable speed/current
 - Capacitive sensing with up to 8 inputs
- Supply Voltage Comparator
- **Ultra efficient Power-on Reset and Brown-Out Detector**
- **2-pin Serial Wire Debug interface**
- **Temperature range -40 - 85°C**
- **Single power supply 1.98 - 3.8 V**
- **Packages**
 - QFN24
 - QFN32
 - TQFP48

3.3 Block Diagram

Figure 3.1 (p. 6) shows the block diagram of EFM32ZG. The color indicates peripheral availability in the different energy modes, described in Section 3.4 (p. 7) .

Figure 3.1. Block Diagram of EFM32ZG

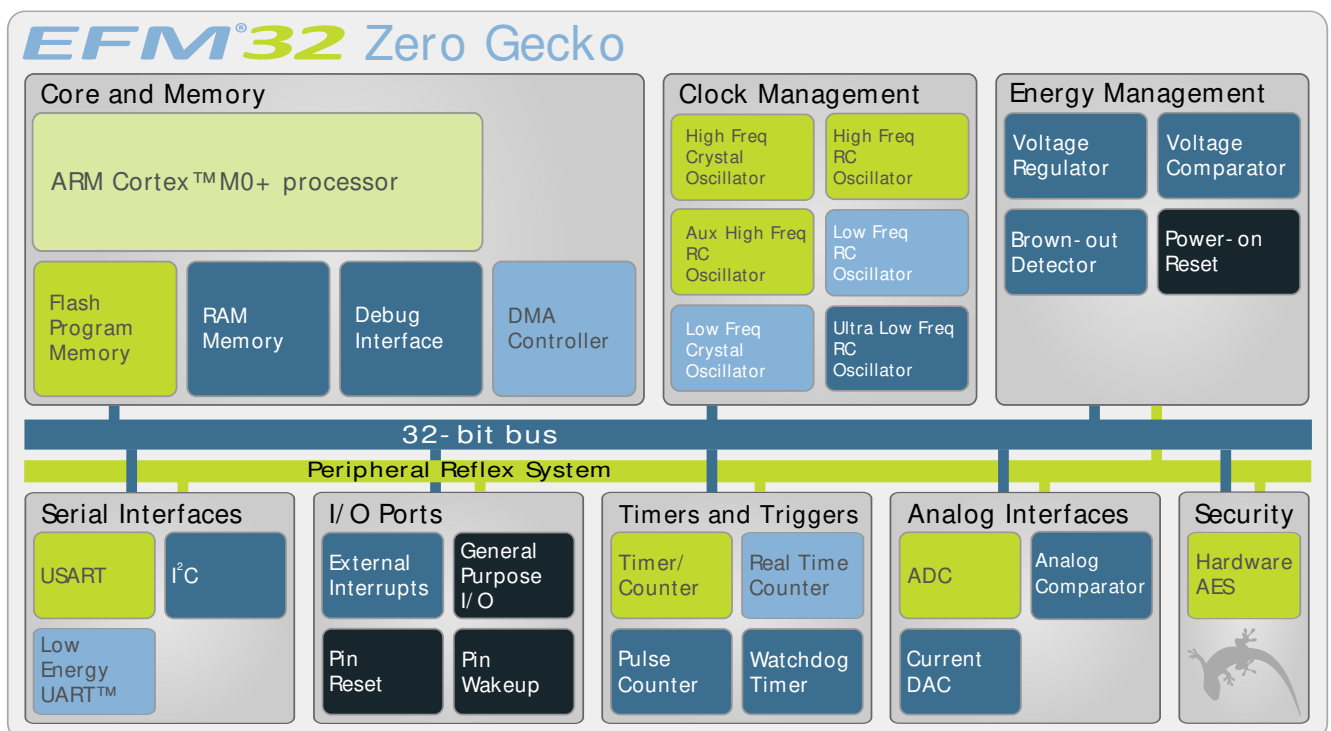
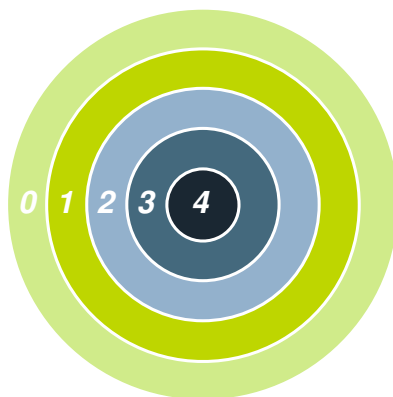


Figure 3.2. Energy Mode Indicator**Note**

In the energy mode indicator, the numbers indicates Energy Mode, i.e EM0-EM4.

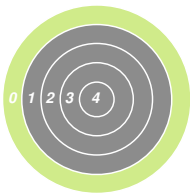
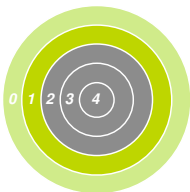
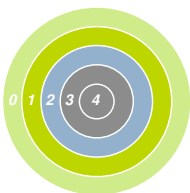
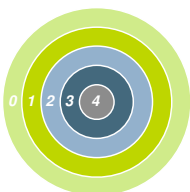
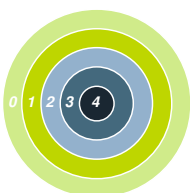
3.4 Energy Modes

There are five different Energy Modes (EM0-EM4) in the EFM32ZG, see Table 3.1 (p. 8). The EFM32ZG is designed to achieve a high degree of autonomous operation in low energy modes. The intelligent combination of peripherals, RAM with data retention, DMA, low-power oscillators, and short wake-up time, makes it attractive to remain in low energy modes for long periods and thus saving energy consumption.

Tip

Throughout this document, the first figure in every module description contains an Energy Mode Indicator showing which energy mode(s) the module can operate (see Table 3.1 (p. 8)).

Table 3.1. Energy Mode Description

Energy Mode	Name	Description
	EM0 – Energy Mode 0 (Run mode)	In EM0, the CPU is running and consuming as little as 114 μ A/MHz, when running code from flash. All peripherals can be active.
	EM1 – Energy Mode 1 (Sleep Mode)	In EM1, the CPU is sleeping and the power consumption is only 48 μ A/MHz. All peripherals, including DMA, PRS and memory system, are still available.
	EM2 – Energy Mode 2 (Deep Sleep Mode)	In EM2 the high frequency oscillator is turned off, but with the 32.768 kHz oscillator running, selected low energy peripherals (RTC, PCNT, LEUART, I ² C, WDOG and ACMP) are still available. This gives a high degree of autonomous operation with a current consumption as low as 0.9 μ A with RTC enabled. Power-on Reset, Brown-out Detection and full RAM and CPU retention is also included.
	EM3 - Energy Mode 3 (Stop Mode)	In EM3, the low-frequency oscillator is disabled, but there is still full CPU and RAM retention, as well as Power-on Reset, Pin reset, EM4 wake-up and Brown-out Detection, with a consumption of only 0.5 μ A. The low-power ACMP, asynchronous external interrupt, PCNT, and I ² C can wake-up the device. Even in this mode, the wake-up time is a few microseconds.
	EM4 – Energy Mode 4 (Shutoff Mode)	In EM4, the current is down to 20 nA and all chip functionality is turned off except the pin reset, GPIO pin wake-up, GPIO pin retention and the Power-On Reset. All pins are put into their reset state.

3.5 Product Overview

Table 3.2 (p. 8) shows a device overview of the EFM32ZG Microcontroller Series, including peripheral functionality. For more information, the reader is referred to the device specific datasheets.

Table 3.2. EFM32ZG Microcontroller Series

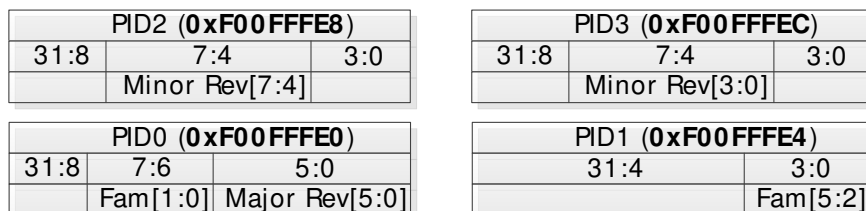
EFM32ZG Part #	Flash	RAM	GPIO(pins)	USB	LCD	USART+UART	LEUART	I ² C	Timer(PWM)	LETIMER	RTC	PCNT	Watchdog	ADC(pins)	DAC(pins)	ACMP(pins)	AES	EBI	LESENSE	Op-Amps	Package
108F4	4	2	17	-	-	1	1	1	2 (6)	-	1	1	1	-	-	1 (2)	Y	-	-	-	QFN24
108F8	8	2	17	-	-	1	1	1	2 (6)	-	1	1	1	-	-	1 (2)	Y	-	-	-	QFN24
108F16	16	4	17	-	-	1	1	1	2 (6)	-	1	1	1	-	-	1 (2)	Y	-	-	-	QFN24
108F32	32	4	17	-	-	1	1	1	2 (6)	-	1	1	1	-	-	1 (2)	Y	-	-	-	QFN24

EFM32ZG Part #	Flash	RAM	GPIO(pins)	USB	LCD	USART+UART	LEUART	I ² C	Timer(PWM)	LETIMER	RTC	PCNT	Watchdog	ADC(pins)	DAC(pins)	ACMP(pins)	AES	EBI	LESENSE	Op-Amps	Package
110F4	4	2	17	-	-	1	1	1	2 (6)	-	1	1	1	1 (2)	-	1 (2)	Y	-	-	-	QFN24
110F8	8	2	17	-	-	1	1	1	2 (6)	-	1	1	1	1 (2)	-	1 (2)	Y	-	-	-	QFN24
110F16	16	4	17	-	-	1	1	1	2 (6)	-	1	1	1	1 (2)	-	1 (2)	Y	-	-	-	QFN24
110F32	32	4	17	-	-	1	1	1	2 (6)	-	1	1	1	1 (2)	-	1 (2)	Y	-	-	-	QFN24
210F4	4	2	24	-	-	1	1	1	2 (6)	-	1	1	1	1 (4)	-	1 (2)	Y	-	-	-	QFN32
210F8	8	2	24	-	-	1	1	1	2 (6)	-	1	1	1	1 (4)	-	1 (2)	Y	-	-	-	QFN32
210F16	16	4	24	-	-	1	1	1	2 (6)	-	1	1	1	1 (4)	-	1 (2)	Y	-	-	-	QFN32
210F32	32	4	24	-	-	1	1	1	2 (6)	-	1	1	1	1 (4)	-	1 (2)	Y	-	-	-	QFN32
222F4	4	2	37	-	-	1	1	1	2 (6)	-	1	1	1	1 (5)	-	1 (2)	Y	-	-	-	QFP48
222F8	8	2	37	-	-	1	1	1	2 (6)	-	1	1	1	1 (5)	-	1 (2)	Y	-	-	-	QFP48
222F16	16	4	37	-	-	1	1	1	2 (6)	-	1	1	1	1 (5)	-	1 (2)	Y	-	-	-	QFP48
222F32	32	4	37	-	-	1	1	1	2 (6)	-	1	1	1	1 (5)	-	1 (2)	Y	-	-	-	QFP48

3.6 Device Revision

The device revision number is read from the ROM Table. The major revision number and the chip family number is read from PID0 and PID1 registers. The minor revision number is extracted from the PID2 and PID3 registers, as illustrated in Figure 3.3 (p. 9). The Fam[5:2] and Fam[1:0] must be combined to complete the chip family number, while the Minor Rev[7:4] and Minor Rev[3:0] must be combined to form the complete revision number.

Figure 3.3. Revision Number Extraction

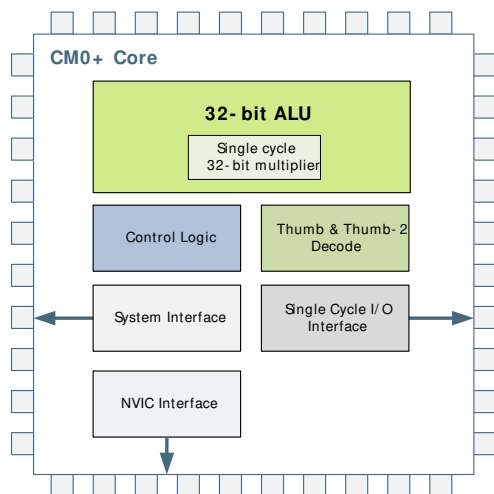
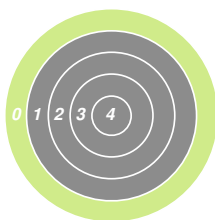


For the latest revision of the Zero Gecko family, the chip family number is 0x04 and the major revision number is 0x01. The minor revision number is to be interpreted according to Table 3.3 (p. 10) .

Table 3.3. Minor Revision Number Interpretation

Minor Rev[7:0]	Revision
0x00	A

4 System Processor



Quick Facts

What?

The industry leading Cortex-M0+ processor from ARM is the CPU in the EFM32ZG microcontrollers.

Why?

The ARM Cortex-M0+ is designed for exceptional short response time, high code density, and high 32-bit throughput while maintaining a strict cost and power consumption budget.

How?

Combined with the ultra low energy peripherals available, the Cortex-M0+ makes the EFM32ZG devices perfect for 8- to 32-bit applications. The processor is featuring a 2 stage pipeline, dedicated single cycle I/O interface, efficient single cycle instructions, Thumb/Thumb-2 instruction set support, and fast interrupt handling.

4.1 Introduction

The ARM Cortex-M0+ 32-bit RISC processor provides outstanding computational performance and exceptional system response to interrupts while meeting low cost requirements and low power consumption.

The ARM Cortex-M0+ implemented is revision r0p1.

4.2 Features

- 2-stage pipeline
- Thumb/Thumb-2 instruction subset
 - Enhanced levels of performance, energy efficiency, and code density
 - Enables direct portability to other ARM Cortex-M processors
- Hardware single-cycle multiplication
 - Enables 32-bit multiplication in a single cycle
- Dedicated Single-cycle I/O interface
 - Provides immediate access to all GPIO-registers
 - Enables the processor to simultaneously fetch the next instructions over the System bus
- Configurable IRQ-latency
 - Allows developers to select a trade-off between interrupt response time and predictability
- Up to 1.08 DMIPS/MHz
- 24-bit System Tick Timer for Real-Time Operating System (RTOS)
- Excellent 32-bit migration choice for 8/16 bit architecture based designs
 - Simplified stack-based programmer's model is compatible with traditional ARM architecture and retains the programming simplicity of legacy 8- and 16-bit architectures
- Integrated power modes

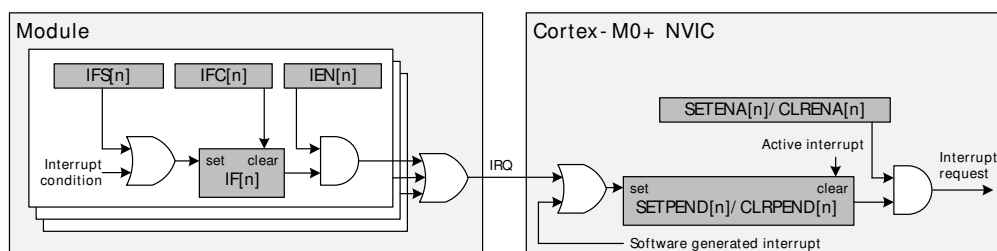
- Sleep Now mode for immediate transfer to low power state
- Sleep on Exit mode for entry into low power state after the servicing of an interrupt
- Ability to extend power savings to other system components
- Optimized for low latency, nested interrupts

4.3 Functional Description

For a full functional description of the ARM Cortex-M0+ (r0p1) implementation in the EFM32ZG family, the reader is referred to the *ARM Cortex-M0+ Devices Generic User Guide*.

4.3.1 Interrupt Operation

Figure 4.1. Interrupt Operation



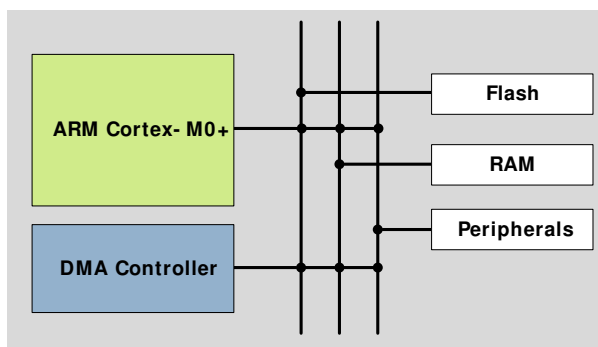
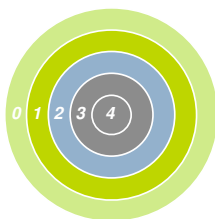
The EFM32ZG devices have up to 17 interrupt request lines (IRQ) which are connected to the Cortex-M0+. Each of these lines (shown in Table 4.1 (p. 12)) are connected to one or more interrupt flags in one or more modules. The interrupt flags are set by hardware on an interrupt condition. It is also possible to set/clear the interrupt flags through the IFS/IFC registers. Each interrupt flag is then qualified with its own interrupt enable bit (IEN register), before being OR'ed with the other interrupt flags to generate the IRQ. A high IRQ line will set the corresponding pending bit (can also be set/cleared with the SETPEND/CLRPEND bits in ISPR0/ICPR0) in the Cortex-M0+ NVIC. The pending bit is then qualified with an enable bit (set/cleared with SETENA/CLRENA bits in ISER0/ICER0) before generating an interrupt request to the core. Figure 4.1 (p. 12) illustrates the interrupt system. For more information on how the interrupts are handled inside the Cortex-M0+, the reader is referred to the *ARM Cortex-M0+ Devices Generic User Guide*.

Table 4.1. Interrupt Request Lines (IRQ)

IRQ #	Source
0	DMA
1	GPIO_EVEN
2	TIMER0
3	ACMP0
4	ADC0
5	I2C0
6	GPIO_ODD
7	TIMER1
8	USART1_RX
9	USART1_TX
10	LEUART0
11	PCNT0

IRQ #	Source
12	RTC
13	CMU
14	VCMP
15	MSC
16	AES

5 Memory and Bus System



Quick Facts

What?

A low latency memory system, including low energy flash and RAM with data retention, makes extended use of low-power energy-modes possible.

Why?

RAM retention reduces the need for storing data in flash and enables frequent use of the ultra low energy modes EM2 and EM3 with as little as 0.5 μ A current consumption.

How?

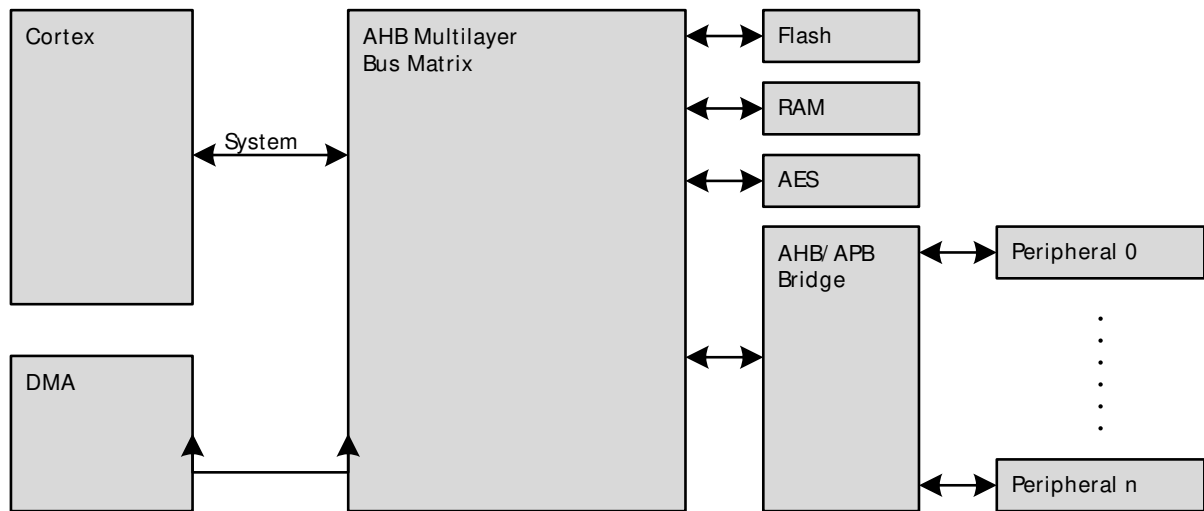
Low energy and non-volatile flash memory stores program and application data in all energy modes and can easily be reprogrammed in system. Low leakage RAM, with data retention in EM0 to EM3, removes the data restore time penalty, and the DMA ensures fast autonomous transfers with predictable response time.

5.1 Introduction

The EFM32ZG contains an AMBA AHB Bus system allowing bus masters to access the memory mapped address space. A multilayer AHB bus matrix, using a Round-robin arbitration scheme, connects the master bus interfaces to the AHB slaves (Figure 5.1 (p. 15)). The bus matrix allows several AHB slaves to be accessed simultaneously. An AMBA APB interface is used for the peripherals, which are accessed through an AHB-to-APB bridge connected to the AHB bus matrix. The AHB bus masters are:

- **Cortex-M0+ System:** Used for instruction fetches, data and debug access (0x00000000 - 0xDFFFFFFF).
- **DMA:** Can access SRAM, Flash and peripherals (0x00000000 - 0xDFFFFFFF), except GPIO (0x40006000 - 0x40007000).

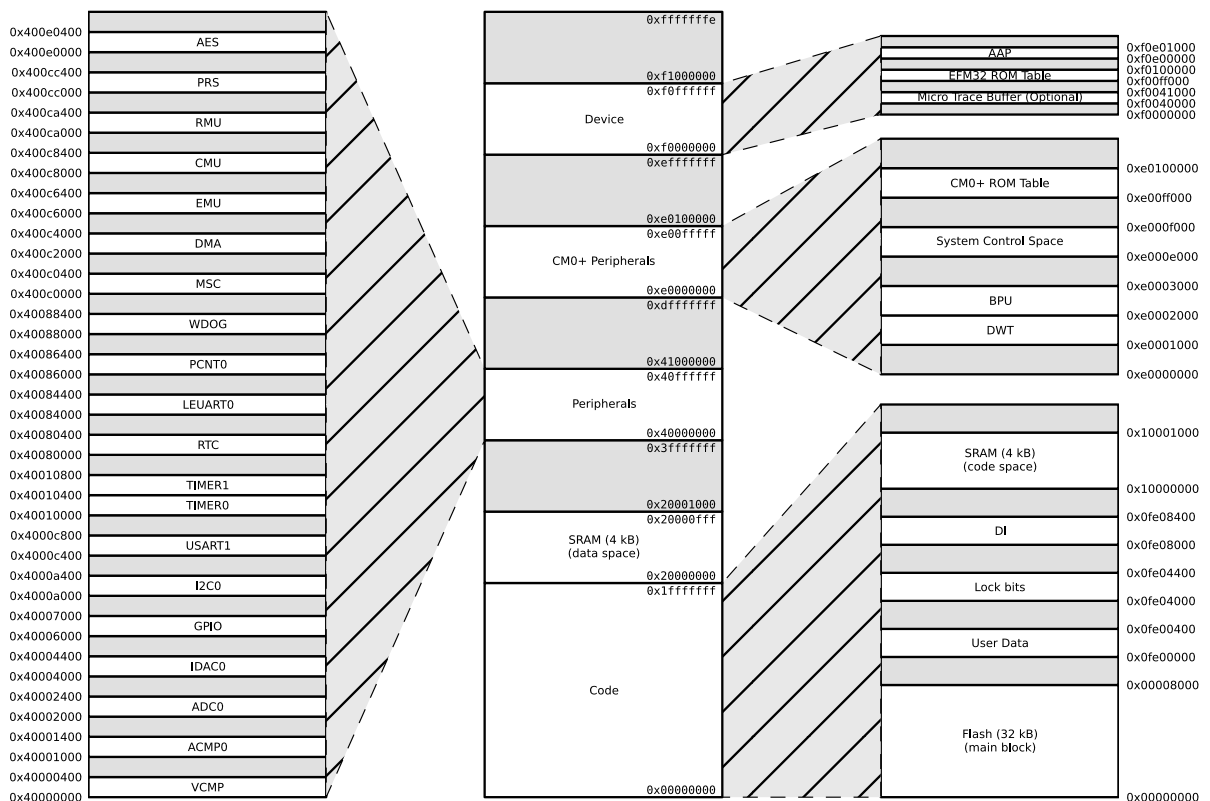
Figure 5.1. EFM32ZG Bus System



5.2 Functional Description

The memory segments are mapped together with the internal segments of the Cortex-M0+ into the system memory map shown by Figure 5.2 (p. 15)

Figure 5.2. System Address Space



The embedded SRAM is located at address 0x20000000 in the memory map of the EFM32ZG. It is also mapped in code space at address 0x10000000 to keep compatibility towards Cortex-M3 and Cortex-M4 EFM32-devices, that uses this code-space mapped SRAM for faster instruction fetching.

5.2.1 Peripherals

The peripherals are mapped into the peripheral memory segment, each with a fixed size address range according to Table 5.1 (p. 16) , Table 5.2 (p. 16) and Table 5.3 (p. 16) .

Table 5.1. Memory System Core Peripherals

Core peripherals	
Address Range	Module Name
0x400E0000 - 0x400E03FF	AES
0x400CA000 - 0x400CA3FF	RMU
0x400C8000 - 0x400C83FF	CMU
0x400C6000 - 0x400C63FF	EMU
0x400C2000 - 0x400C3FFF	DMA
0x400C0000 - 0x400C03FF	MSC

Table 5.2. Memory System Low Energy Peripherals

Low Energy peripherals	
Address Range	Module Name
0x40088000 - 0x400883FF	WDOG
0x40086000 - 0x400863FF	PCNT0
0x40084000 - 0x400843FF	LEUART0
0x40080000 - 0x400803FF	RTC

Table 5.3. Memory System Peripherals

Peripherals	
Address Range	Module Name
0x400CC000 - 0x400CC3FF	PRS
0x40010400 - 0x400107FF	TIMER1
0x40010000 - 0x400103FF	TIMER0
0x4000C400 - 0x4000C7FF	USART1
0x4000A000 - 0x4000A3FF	I2C0
0x40006000 - 0x40006FFF	GPIO
0x40004000 - 0x400043FF	IDAC0
0x40002000 - 0x400023FF	ADC0
0x40001000 - 0x400013FF	ACMP0
0x40000000 - 0x400003FF	VCMP

5.2.2 Bus Matrix

The Bus Matrix connects the memory segments to the bus masters:

- Code: CPU instruction or data fetches from the code space
- System: CPU read and write to the SRAM and peripherals
- DMA: Access to SRAM, Flash and peripherals

5.2.2.1 Arbitration

The Bus Matrix uses a round-robin arbitration algorithm which enables high throughput and low latency while starvation of simultaneous accesses to the same bus slave are eliminated. Round-robin does not assign a fixed priority to each bus master. The arbiter does not insert any bus wait-states.

5.2.2.2 Access Performance

The Bus Matrix is a multi-layer energy optimized AMBA AHB compliant bus with an internal bandwidth equal to 4 times a single AHB-bus.

The Bus Matrix accepts new transfers initiated by each master in every clock cycle without inserting any wait-states. The slaves, however, may insert wait-states depending on their internal throughput and the clock frequency.

The Cortex-M0+, the DMA Controller, and the peripherals run on clocks that can be prescaled separately. When accessing a peripheral which runs on a frequency equal to or faster than the HFCORECLK, the number of wait cycles per access, in addition to master arbitration, is given by:

Memory Wait Cycles with Clock Equal or Faster than HFCORECLK

$$N_{\text{cycles}} = 2 + N_{\text{slave cycles}}, \quad (5.1)$$

where $N_{\text{slave cycles}}$ is the wait cycles introduced by the slave.

When accessing a peripheral running on a clock slower than the HFCORECLK, wait-cycles are introduced to allow the transfer to complete on the peripheral clock. The number of wait cycles per access, in addition to master arbitration, is given by:

Memory Wait Cycles with Clock Slower than CPU

$$N_{\text{cycles}} = (2 + N_{\text{slave cycles}}) \times f_{\text{HFCORECLK}}/f_{\text{HFPERCLK}}, \quad (5.2)$$

where $N_{\text{slave cycles}}$ is the number of wait cycles introduced by the slave.

For general register access, $N_{\text{slave cycles}} = 1$.

More details on clocks and prescaling can be found in Chapter 11 (p. 92) .

5.3 Access to Low Energy Peripherals (Asynchronous Registers)

5.3.1 Introduction

The Low Energy Peripherals are capable of running when the high frequency oscillator and core system is powered off, i.e. in energy mode EM2 and in some cases also EM3. This enables the peripherals to perform tasks while the system energy consumption is minimal.

The Low Energy Peripherals are:

- Low Energy UART - LEUART
- Pulse Counter - PCNT
- Real Time Counter - RTC
- Watchdog - WDOG

All Low Energy Peripherals are memory mapped, with automatic data synchronization. Because the Low Energy Peripherals are running on clocks asynchronous to the core clock, there are some constraints on how register accesses can be done, as described in the following sections.

5.3.1.1 Writing

Every Low Energy Peripheral has one or more registers with data that needs to be synchronized into the Low Energy clock domain to maintain data consistency and predictable operation. There are two different synchronization mechanisms on the Zero Gecko; immediate synchronization, and delayed synchronization. Immediate synchronization is available for the RTC and results in an immediate update of the target registers. Delayed synchronization is used for the other Low Energy Peripherals, and for these peripherals, a write operation requires 3 positive edges on the clock of the Low Energy Peripheral being accessed. Registers requiring synchronization are marked "Asynchronous" in their description header.

5.3.1.1.1 Delayed synchronization

After writing data to a register which value is to be synchronized into the Low Energy Peripheral using delayed synchronization, a corresponding busy flag in the <module_name>_SYNCBUSY register (e.g. LEUART_SYNCBUSY) is set. This flag is set as long as synchronization is in progress and is cleared upon completion.

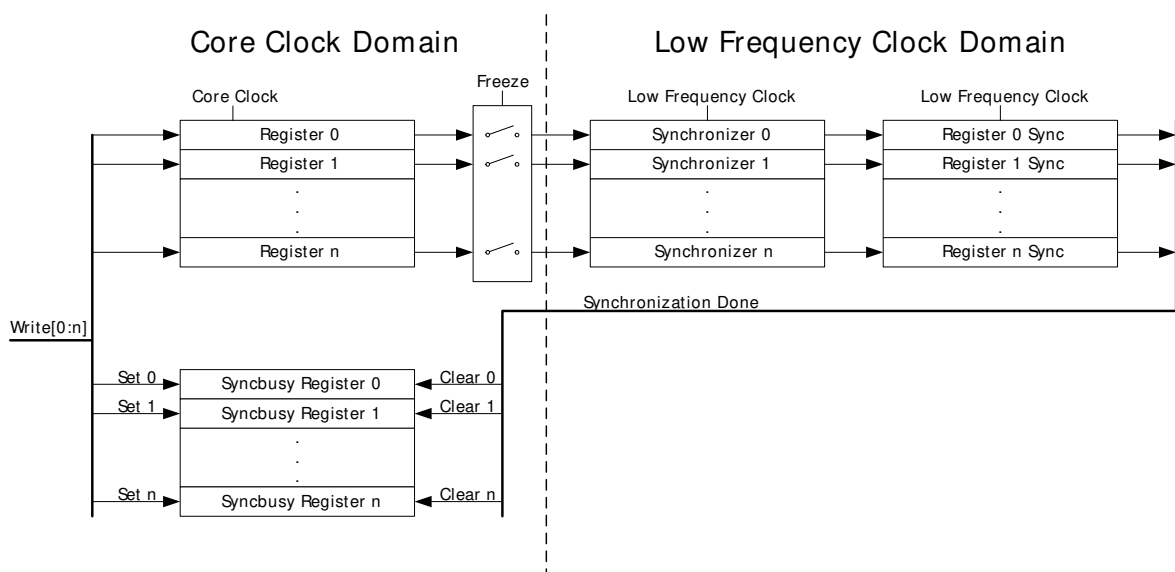
Note

Subsequent writes to the same register before the corresponding busy flag is cleared is not supported. Write before the busy flag is cleared may result in undefined behavior.

In general, the SYNCBUSY register only needs to be observed if there is a risk of multiple write access to a register (which must be prevented). It is not required to wait until the relevant flag in the SYNCBUSY register is cleared after writing a register. E.g EM2 can be entered immediately after writing a register.

See Figure 5.3 (p. 18) for a more detailed overview of the write operation.

Figure 5.3. Write operation to Low Energy Peripherals



5.3.1.1.2 Immediate synchronization

Contrary to the peripherals with delayed synchronization, data written to peripherals with immediate synchronization, takes effect in the peripheral immediately. They are updated immediately on the

peripheral write access. If a write is set up close to a peripheral clock edge, the write is delayed to after the clock edge. This will introduce wait-states on peripheral access. In the worst case, there can be three wait-state cycles of the HFCORECLK_LE and an additional wait-state equivalent of up to 315 ns.

For peripherals with immediate synchronization, the SYNCBUSY registers are still present and serve two purposes: (1) commands written to a peripheral with immediate synchronization are not executed before the first peripheral clock after the write. During this period, the SYNCBUSY flag in the command register is set, indicating that the command has not yet been executed; (2) to maintain backwards compatibility with the EFM32G series, SYNCBUSY registers are also present for other registers. These are however, always 0, indicating that register writes are always safe.

Note

If the application must be compatible with the EFM32G series, all Low Energy Peripherals should be accessed as if they only had delayed synchronization, i.e. using SYNCBUSY.

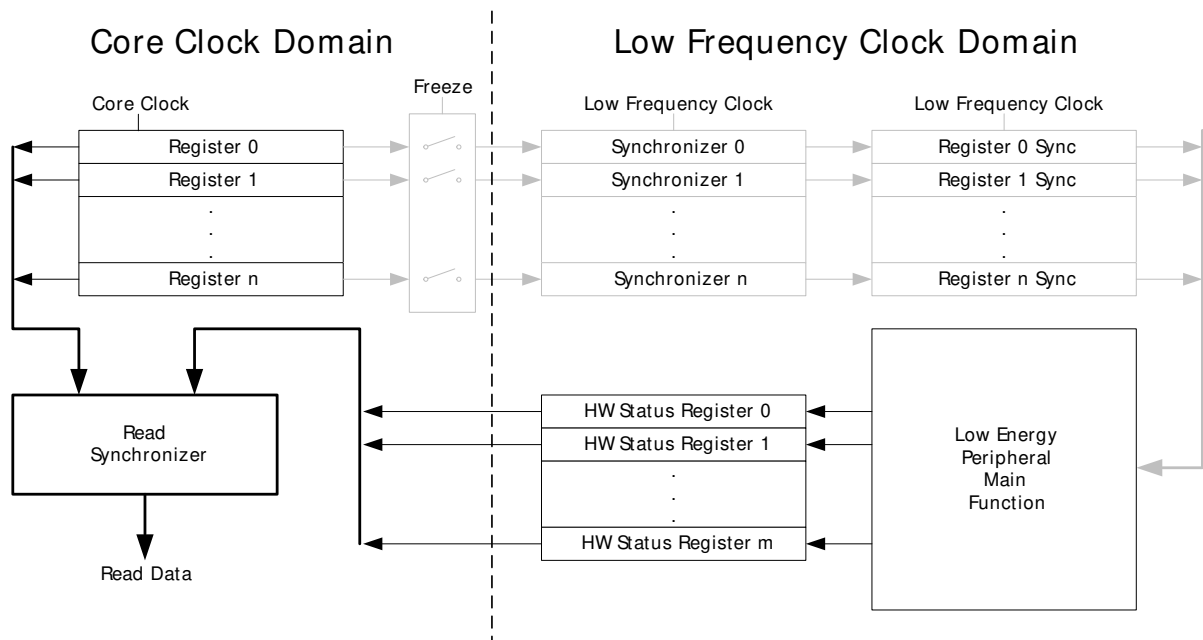
5.3.1.2 Reading

When reading from Low Energy Peripherals, the data is synchronized regardless of the originating clock domain. Registers updated/maintained by the Low Energy Peripheral are read directly from the Low Energy clock domain. Registers residing in the core clock domain, are read from the core clock domain. See Figure 5.4 (p. 19) for a more detailed overview of the read operation.

Note

Writing a register and then immediately reading back the value of the register may give the impression that the write operation is complete. This is not necessarily the case. Please refer to the SYNCBUSY register for correct status of the write operation to the Low Energy Peripheral.

Figure 5.4. Read operation from Low Energy Peripherals



5.3.2 FREEZE register

For Low Energy Peripherals with delayed synchronization there is a <module_name>_FREEZE register (e.g. RTC_FREEZE), containing a bit named REGFREEZE. If precise control of the synchronization process is required, this bit may be utilized. When REGFREEZE is set, the synchronization process is halted, allowing the software to write multiple Low Energy registers before starting the synchronization

process, thus providing precise control of the module update process. The synchronization process is started by clearing the REGFREEZE bit.

Note

The FREEZE register is also present on peripherals with immediate synchronization, but has no effect.

5.4 Flash

The Flash retains data in any state and typically stores the application code, special user data and security information. The Flash memory is typically programmed through the debug interface, but can also be erased and written to from software.

- Up to 32 kB of memory
- Page size of 1024 bytes (minimum erase unit)
- Minimum 20 000 erase cycles
- More than 10 years data retention at 85°C
- Lock-bits for memory protection
- Data retention in any state

5.5 SRAM

The primary task of the SRAM memory is to store application data. Additionally, it is possible to execute instructions from SRAM, and the DMA may be used to transfer data between the SRAM, Flash and peripherals.

- Up to 4 kB memory
- Data retention of the entire memory in EM0 to EM3

5.6 Device Information (DI) Page

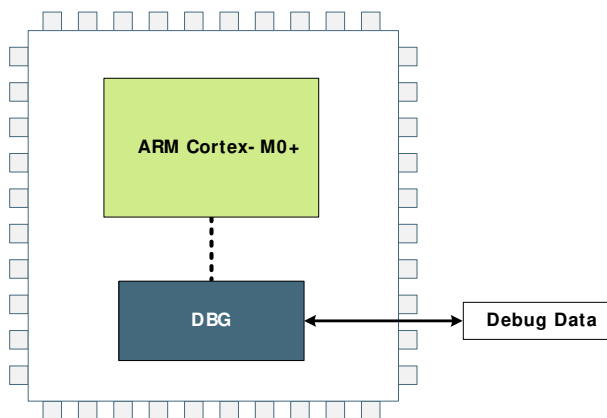
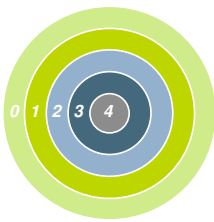
The DI page contains calibration values, a unique identification number and other useful data. See the table below for a complete overview.

Table 5.4. Device Information Page Contents

DI Address	Register	Description
0x0FE08020	CMU_LFRCTRL	Register reset value.
0x0FE08028	CMU_HFRCTRL	Register reset value.
0x0FE08030	CMU_AUXHFRCTRL	Register reset value.
0x0FE08040	ADC0_CAL	Register reset value.
0x0FE08048	ADC0_BIASPROG	Register reset value.
0x0FE08050	ACMP0_CTRL	Register reset value.
0x0FE08078	IDAC0_CAL	Register reset value.
0x0FE081B0	DI_CRC	[15:0]: DI data CRC-16.
0x0FE081B2	CAL_TEMP_0	[7:0] Calibration temperature (°C).
0x0FE081B4	ADC0_CAL_1V25	[14:8]: Gain for 1V25 reference, [6:0]: Offset for 1V25 reference.
0x0FE081B6	ADC0_CAL_2V5	[14:8]: Gain for 2V5 reference, [6:0]: Offset for 2V5 reference.

DI Address	Register	Description
0x0FE081B8	ADC0_CAL_VDD	[14:8]: Gain for VDD reference, [6:0]: Offset for VDD reference.
0x0FE081BA	ADC0_CAL_5VDIFF	[14:8]: Gain for 5VDIFF reference, [6:0]: Offset for 5VDIFF reference.
0x0FE081BC	ADC0_CAL_2XVDD	[14:8]: Reserved (gain for this reference cannot be calibrated), [6:0]: Offset for 2XVDD reference.
0x0FE081BE	ADC0_TEMP_0_READ_1V25	[15:4] Temperature reading at 1V25 reference, [3:0]: Reserved.
0x0FE081C8	IDAC0_CAL_RANGE0	[7:0]: Current range 0 tuning.
0x0FE081C9	IDAC0_CAL_RANGE1	[7:0]: Current range 1 tuning.
0x0FE081CA	IDAC0_CAL_RANGE2	[7:0]: Current range 2 tuning.
0x0FE081CB	IDAC0_CAL_RANGE3	[7:0]: Current range 3 tuning.
0x0FE081D4	AUXHFRCO_CALIB_BAND_1	[7:0]: Tuning for the 1.2 MHz AUXHFRCO band.
0x0FE081D5	AUXHFRCO_CALIB_BAND_7	[7:0]: Tuning for the 6.6 MHz AUXHFRCO band.
0x0FE081D6	AUXHFRCO_CALIB_BAND_11	[7:0]: Tuning for the 11 MHz AUXHFRCO band.
0x0FE081D7	AUXHFRCO_CALIB_BAND_14	[7:0]: Tuning for the 14 MHz AUXHFRCO band.
0x0FE081D8	AUXHFRCO_CALIB_BAND_21	[7:0]: Tuning for the 21 MHz AUXHFRCO band.
0x0FE081DC	HFRCO_CALIB_BAND_1	[7:0]: Tuning for the 1.2 MHz HFRCO band.
0x0FE081DD	HFRCO_CALIB_BAND_7	[7:0]: Tuning for the 6.6 MHz HFRCO band.
0x0FE081DE	HFRCO_CALIB_BAND_11	[7:0]: Tuning for the 11 MHz HFRCO band.
0x0FE081DF	HFRCO_CALIB_BAND_14	[7:0]: Tuning for the 14 MHz HFRCO band.
0x0FE081E0	HFRCO_CALIB_BAND_21	[7:0]: Tuning for the 21 MHz HFRCO band.
0x0FE081F0	UNIQUE_0	[31:0] Unique number.
0x0FE081F4	UNIQUE_1	[63:32] Unique number.
0x0FE081F8	MEM_INFO_FLASH	[15:0]: Flash size, kbyte count as unsigned integer (e.g. 128).
0x0FE081FA	MEM_INFO_RAM	[15:0]: Ram size, kbyte count as unsigned integer (e.g. 16).
0x0FE081FC	PART_NUMBER	[15:0]: EFM32 part number as unsigned integer (e.g. 230).
0x0FE081FE	PART_FAMILY	[7:0]: EFM32 part family number (Gecko = 71, Giant Gecko = 72, Tiny Gecko = 73, Leopard Gecko=74, Wonder Gecko=75, Zero Gecko=76).
0x0FE081FF	PROD_REV	[7:0]: EFM32 Production ID.

6 DBG - Debug Interface



Quick Facts

What?

The DBG (Debug Interface) is used to program and debug EFM32ZG devices.

Why?

The Debug Interface makes it easy to re-program and update the system in the field, and allows debugging with minimal I/O pin usage.

How?

The Cortex-M0+ supports advanced debugging features. EFM32ZG devices only use two port pins for debugging or programming. The internal and external state of the system can be examined with debug extensions supporting instruction or data access break- and watch points.

6.1 Introduction

The EFM32ZG devices include hardware debug support through a 2-pin serial-wire debug (SWD) interface.

For more technical information about the debug interface the reader is referred to:

- ARM Cortex-M0+ Technical Reference Manual
- ARM CoreSight Components Technical Reference Manual
- ARM Debug Interface v5 Architecture Specification

6.2 Features

- Flash Patch and Breakpoint (FPB) unit
 - Implement breakpoints and code patches
- Data Watch point and Trace (DWT) unit
 - Implement watch points, trigger resources and system profiling

6.3 Functional Description

There are two debug pins available on the device. Their operation is described in the following section.

6.3.1 Debug Pins

The following pins are the debug connections for the device:

- Serial Wire Clock input (SWCLK): This pin is enabled after reset and has a built-in pull down.
- Serial Wire Data Input/Output (SWDIO): This pin is enabled after reset and has a built-in pull-up.

The debug pins can be enabled and disabled through GPIO_ROUTE, see Section 25.3.4.1 (p. 361). Please remember that upon disabling, debug contact with the device is lost. Also note that, because the debug pins have pull-down and pull-up enabled by default, leaving them enabled might increase the current consumption with up to 200 µA if left connected to supply or ground.

6.3.2 Debug and EM2/EM3

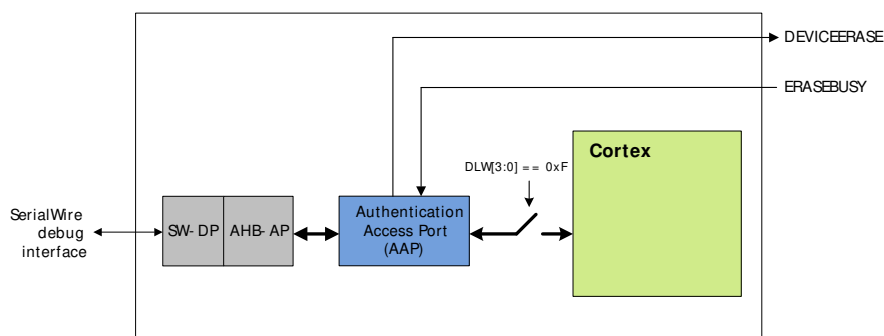
Leaving the debugger connected when issuing a WFI or WFE to enter EM2 or EM3 will make the system enter a special EM2. This mode differs from regular EM2 and EM3 in that the high frequency clocks are still enabled, and certain core functionality is still powered in order to maintain debug-functionality. Because of this, the current consumption in this mode is closer to EM1 and it is therefore important to disconnect the debugger before doing current consumption measurements.

6.4 Debug Lock and Device Erase

The debug access to the Cortex-M0+ is locked by clearing the Debug Lock Word (DLW) and resetting the device, see Section 7.3.2 (p. 29).

When debug access is locked, the debug interface remains accessible but the connection to the Cortex-M0+ core and the whole bus-system is blocked as shown in Figure 6.2 (p. 24). This mechanism is controlled by the Authentication Access Port (AAP) as illustrated by Figure 6.1 (p. 23). The AAP is only accessible from a debugger and not from the core.

Figure 6.1. AAP - Authentication Access Port



As seen from Figure 6.1 (p. 23), the AAP is situated after the AHB-AP, meaning it should be accessed like any other peripheral from the debug. The address of the AAP is 0xF0E00000 as can also be seen from Figure 5.2 (p. 15).

Note

This is different from some other EFM32 devices, where the AAP is integrated as a separate AP (Access Port), please see the reference manual of the respective devices.

The debugger can access the AAP-registers, and only these registers just after reset, for the time of the AAP-window outlined in Figure 6.2 (p. 24). If the device is locked, access to the core and bus-system is blocked even after code execution starts, and the debugger can only access the AAP-registers. If the device is not locked, the AAP is no longer accessible after code execution starts, and the debugger can access the core and bus-system normally. The AAP window can be extended by issuing the bit pattern on SWDIO/SWCLK as shown in Figure 6.3 (p. 24). This pattern should be applied just before reset is deasserted, and will give the debugger more time to access the AAP.

Figure 6.2. Device Unlock

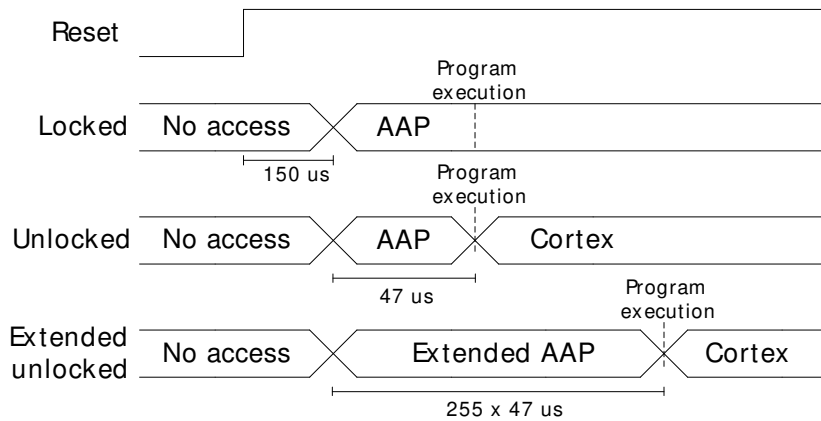
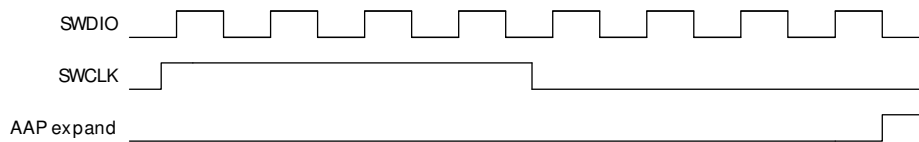


Figure 6.3. AAP Expansion



If the device is locked, it can be unlocked by writing a valid key to the AAP_CMDKEY register and then setting the DEVICEERASE bit of the AAP_CMD register via the debug interface. The commands are not executed before AAP_CMDKEY is invalidated, so this register should be cleared to to start the erase operation. This operation erases the main block of flash, all lock bits are reset and debug access through the AHB-AP is enabled. The operation takes 40 ms to complete. Note that the SRAM contents will also be deleted during a device erase, while the UD-page is not erased.

Even if the device is not locked, the can device can be erased through the AAP, using the above procedure during the AAP window. This can be useful if the device has been programmed with code that, e.g., disables the debug interface pins on start-up, or does something else that prevents communication with a debugger.

If the device is locked, the debugger may read the status from the AAP_STATUS register. When the ERASEBUSY bit is set low after DEVICEERASE of the AAP_CMD register is set, the debugger may set the SYSRESETREQ bit in the AAP_CMD register. After reset, the debugger may resume a normal debug session through the AHB-AP. If the device is not locked, the device erase starts when the AAP window closes, so it is not possible to poll the status.

6.5 Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x000	AAP_CMD	W1	Command Register
0x004	AAP_CMDKEY	W1	Command Key Register
0x008	AAP_STATUS	R	Status Register
0x0FC	AAP_IDR	R	AAP Identification Register

6.6 Register Description

6.6.1 AAP_CMD - Command Register

Offset	Bit Position																																	
0x000	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																																		
Access																																		
Name																																		
																															SYSRESETREQ	0	1	0
																															DEVICEERASE	0	1	0

Bit	Name	Reset	Access	Description
31:2	<i>Reserved</i>	<i>To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)</i>		
1	SYSRESETREQ	0	W1	System Reset Request A system reset request is generated when set to 1. This register is write enabled from the AAP_CMDKEY register.
0	DEVICEERASE	0	W1	Erase the Flash Main Block, SRAM and Lock Bits When set, all data and program code in the main block is erased, the SRAM is cleared and then the Lock Bit (LB) page is erased. This also includes the Debug Lock Word (DLW), causing debug access to be enabled after the next reset. The information block User Data page (UD) is left unchanged, but the User data page Lock Word (ULW) is erased. This register is write enabled from the AAP_CMDKEY register.

6.6.2 AAP_CMDKEY - Command Key Register

Offset	Bit Position																															
0x004	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	0x00000000																															
Access	W1																															
Name	WRITEKEY																															

Bit	Name	Reset	Access	Description
31:0	WRITEKEY	0x00000000	W1	CMD Key Register