AN1189: Incremental Analog to Digital Converter (IADC)

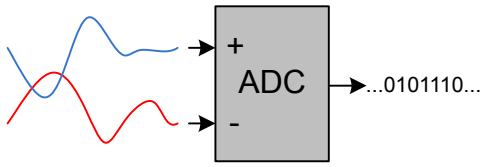
This application note describes the EFR32 Gecko Series 2 Incremental Analog to Digital Converter (IADC) operation and advanced features. In addition, this document explains how to use the IADC to convert an analog input voltage to a digital value and features a high-speed, low-power operation. Many aspects of the IADC, including inputs, references, and the different operating modes are described.

The provided software examples show how to use the different operating modes of the ADC. The example projects are configured for the EFR32 Gecko Series 2 devices. Calibration routines for offset and gain are also included.

For IADC example projects, see: https://github.com/SiliconLabs/peripheral_examples/tree/master/series2/iadc

KEY POINTS

- EFR32 Series 2 devices include new IADC features
- This document discusses IADC operation and advanced features
- The IADC supports offset and gain calibration



1. Device Compatibility

This application note supports Series 2 device families.

EFR32 Wireless Gecko Series 2 consists of the following:

- EFR32BG21
- EFR32MG21
- EFR32BG22
- EFR32FG22
- EFR32MG22
- EFR32FG23
- EFR32SG23
- EFR32ZG23
- EFR32BG24
- EFR32MG24
- EFR32FG25
- EFR32BG27
- EFR32MG27
- EFR32FG28
- EFR32SG28
- EFR32ZG28

EFM32 Series 2 consists of:

- EFM32PG22
- EFM32PG23
- EFM32PG28

2. Incremental Analog to Digital Converter

2.1 Introduction

The EFR32 Wireless Gecko Series 2 IADC is an intermediate architecture combining techniques from both Successive Approximation Register (SAR) and Delta-Sigma style converters. The maximum resolution for normal and high speed[1] modes is 12 bits without oversampling, which can achieve up to one million samples per second (1 Msps) in normal mode and up to two million samples per second (2 Msps) in high-speed[1] mode. The flexible incremental architecture uses oversampling to allow applications to trade speed for higher resolution. High-accuracy[1] mode enables greater than 15 bits of noise-free resolution at speeds up to 10.7 Ksps. An integrated input multiplexer can select from external I/Os and several internal signals. With PRS and DMA, the IADC can operate without CPU intervention, minimizing current consumption or allowing the core to do other work. The IADC can be clocked at different speeds, run using different warm-up modes, and shut down between conversions to reduce the energy consumption even further.

This application note discusses general operation and usage of the IADC. In addition, advanced features and power saving techniques are described. Software examples demonstrating IADC operation with DMA and PRS are included. Offset and gain calibration of the IADC is also described and included in the software examples.

For extremely low power periodic IADC sampling, a software example that enters Energy Mode 2 (EM2) between each IADC sample can be found on the GitHub repository previously linked.

2.2 Overview

The Figure 2.1 IADC Overview on page 4 shows the IADC module block diagram.

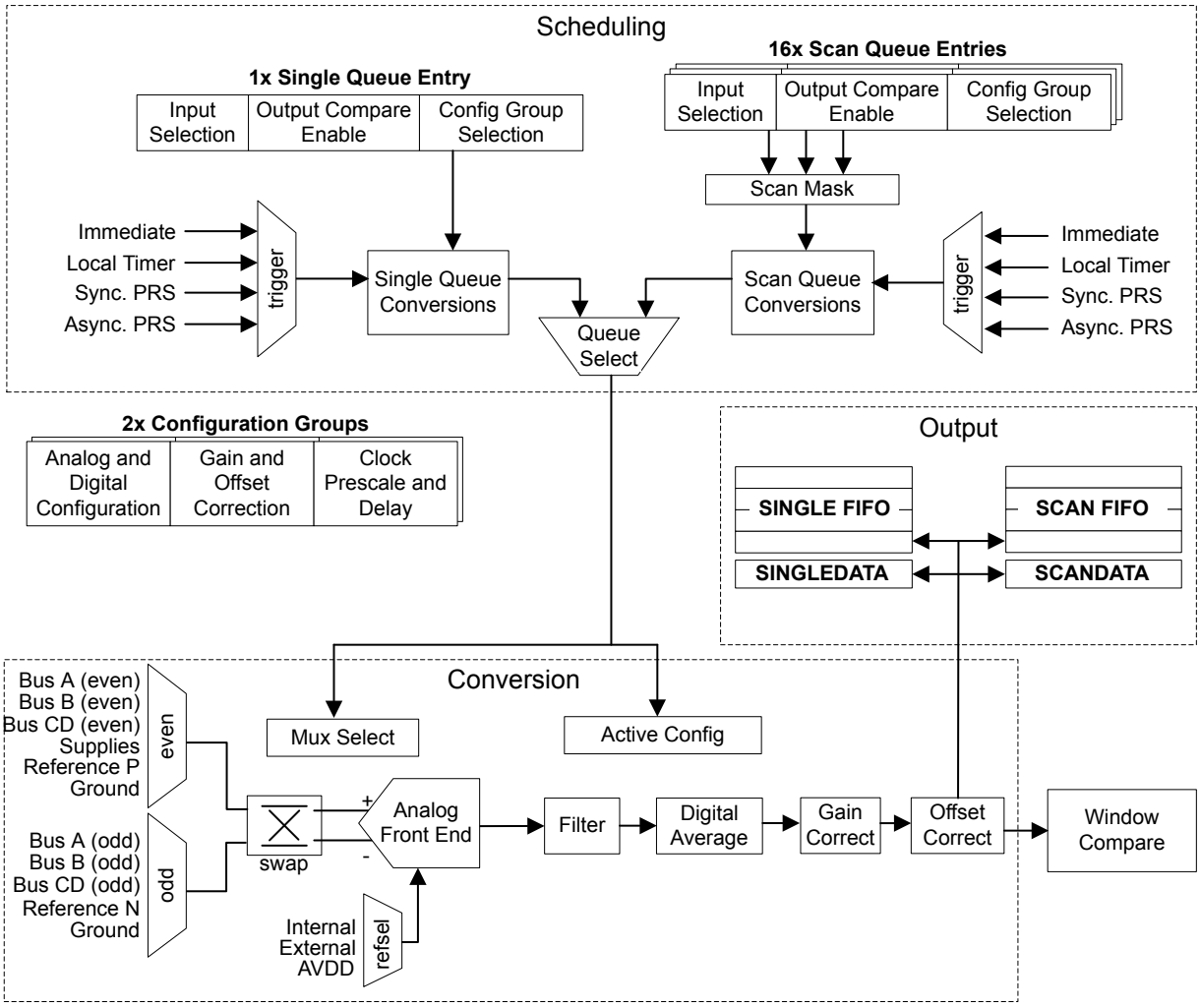


Figure 2.1. IADC Overview

Some new IADC features (see the list below) are added in the EFR32 Wireless Gecko Series 2 devices.

- Flexible architecture allows for tradeoffs between speed and resolution
  - Normal Mode (all devices): Flexible speed and performance, 12-16 bits output resolution
    - 11 bits ENOB performance at 1 Msps (OSR = 2)
    - 12 bits ENOB performace at 555 ksps (OSR = 4)
    - 14 bits ENOB performance at 76.9 ksps (OSR = 32)
  - High-speed[1] Mode (select devices): Doubles output speed of Normal mode with similar performance, 12-16 bits output resolution
    - 11 bits ENOB performance at 2 Msps (OSR = 2)
    - 14 bits ENOB performance at 153.8 ksps (OSR = 32)
  - High-accuracy[1] Mode (select devices): Optimized for low-rate, high-performance applications, with 16-20 bit output resolution
    - 16 bits ENOB performance at 3.8 ksps (OSR = 256)
    - 15 bits ENOB performance at 15.3 ksps (OSR = 64)
- Digital post-averaging for all devices except EFR32xG21

- Internal and external conversion trigger sources
  - Immediate (software triggered)
  - Local IADC timer
  - External TIMER module (synchronous with output / PWM generation)
  - General PRS hardware signal
  - LESENSE (SCAN only)[2]
- Integrated prescaler for conversion clock generation
- Can be run during EM2 and EM3, waking up the system on interrupts as needed
- Can be run during EM2 and EM3 with DMA enabled to pull data from the FIFOs without waking up the system
- Selectable reference sources
  - 1.21 V internal reference
  - External precision reference
  - Analog supply
- Support for offset and gain calibration
- Programmable input gain: 0.5x, 1x, 2x, 3x, or 4x
- Flexible output formatting
  - Unipolar or 2's complement bipolar data
  - Results can be saved in 12 bit, 16 bit, or 20 bit format, depending on device
  - Programmable left or right justification
  - Optional channel ID tag
- Digital window comparison function detects when results are inside/outside a programmable window
- Two independent groups of configuration registers for setting IADC mode, clock prescaler, reference selection, oversample rate, unipolar/bipolar output formatting, and analog gain
- Programmable single channel conversion
  - Can use either configuration group
  - Triggered by any conversion trigger source
  - Can be tailgated after a scan sequence
  - One shot or continuous mode
  - Local FIFO for immediate data storage
  - Programmable watermark level to generate interrupt or initiate DMA transfer
  - Supports overflow and underflow interrupt generation
  - Supports window compare function
- Autonomous multi-channel scan
  - Up to 16 configurable slots in scan sequence
  - Each slot allows independent selection of configuration group, channel selection, and window compare enable
  - Triggered by any conversion trigger source
  - One shot or continuous mode
  - Local FIFO for immediate data storage
  - Programmable watermark level to generate interrupt or initiate DMA transfer
  - Supports overflow and underflow interrupt generation
  - Conversion tailgating support for predictable periodic scans

- Available interrupt sources:
  - Single FIFO has data valid level (DVL) entries available (also generates DMA request)
  - Scan FIFO has DVL entries available (also generates DMA request)
  - Single FIFO result compared true for digital compare window
  - Scan FIFO result compared true for digital compare window
  - Single queue conversion has completed
  - Scan queue entry conversion has completed
  - Scan queue table conversion has completed
  - Single FIFO overflow or underflow
  - Scan FIFO overflow or underflow
  - Polarity Error interrupt
  - Port Allocation Error interrupt
  - EM23 clock configuration error

Note:

1. High-Speed and High-Accuracy modes are only available on select devices of EFM32PG23, EFR32xG24, and EFM32PG28. Refer to the device data sheet for details.
2. LESENSE peripheral is only available on select devices. Refer to the device data sheet for details.

3. General Operation

3.1 Register Access

Many of the IADC module's configuration registers can only be written while the module is disabled (IADC_EN_EN = 0). These are:

- IADC_CTRL
- IADC_TIMER
- IADC_CMPTHR
- IADC_TRIGGER
- IADC_CFGx
- IADC_SCALEx
- IADC_SCHEDx
- IADC_SCANx

A typical setup sequence for the IADC module is:

1. With the IADC disabled (IADC_EN_EN = 0), program all configuration registers listed above.
2. Enable the IADC by setting EN in IADC_EN to 1.
3. Program the remaining configuration registers.
4. Enable the single or scan queue.
5. The IADC is ready for use.

3.2 Modes

The IADC has two modes of operation: single channel mode and scan mode. Both modes have separate configuration and result registers. Both modes may be set up to run only once per trigger or to automatically repeat after each operation.

3.2.1 Single Channel Mode

In single channel mode, the ADC converts one input, either one time or continuously if the SINGLETRIGACTION bitfield in the IADC_TRIGGER register is set to CONTINUOUS.

3.2.2 Scan Mode

In scan mode, the IADC can be configured to convert a sequence of different inputs, either one time or continuously if the SCANTRIGACTION bitfield in the IADC_TRIGGER register is set to CONTINUOUS. The scan queue allows the IADC to automatically convert up to 16 channels in sequence without CPU intervention. Input and configuration selection for each channel in the scan table is specified by the IADC_SCANx register for that channel (channel 0 is configured with IADC_SCAN0, channel 1 is configured with IADC_SCAN1, etc). The IADC_MASKREQ register allows software to define which of the scan table entries (IADC_SCANx) to convert during a scan. For example, channels 0, 1, and 7 can be enabled by writing bits 0, 1, and 7 of IADC_MASKREQ to 1 (IADC_MASKREQ = 0x0083).

The IADC_SCANx registers must be configured when the IADC module is disabled (IADC_EN_EN = 0). IADC_MASKREQ can be written while IADC_EN_EN is set to 1. If a scan operation is in progress, MASKREQ will be synchronized and held until the current scan operation has completed. Then MASKREQ is copied into the STMASK register for the next scan operation. IADC_STMASK is the working copy of the MASKREQ used by the IADC during a scan. MASKREQ will only transfer to STMASK when the scan queue is not scanning and converting the scan table. IADC_STATUS_MASKWRITEPENDING can be used by software to see when the MASKREQ write has been transferred to STMASK. Writing a new MASKREQ in the middle of a scan will not corrupt the current scan. Software which writes to MASKREQ during a scan operation must ensure IADC_STATUS_MASKWRITEPENDING returns to 0 before updating IADC_MASKREQ again.

3.3 Reference Selection

To convert an analog voltage to a digital value, the IADC needs a reference voltage to which it compares the incoming analog voltage. Since the IADC cannot measure voltages larger than the reference voltage, the reference voltage should be above the maximum expected measured voltage.

The following table lists the available IADC references. The default IADC reference is the internal band gap circuit. The reference voltage is selected using the REFSEL field in the IADC_CFGx register.

Table 3.1. ADC Reference Selection

Reference Name	Normal and High-Speed[1] Mode Voltage Range	High-Accuracy[1] Mode VoltageRange
Internal bandgap reference (VBGR)	1.21 V	1.21 V
External reference (VREF)	1.0 V - AVDD (Calibrated for 1.25 V nominal.)	1.0 - 1.25 V
External reference (VREF2P5)	1.0 V - AVDD(Calibrated for 1.25 V nominal.)	1.0 - 2.5 V[2]
Analog Power Supply (VDDX)	AVDD (unbuffered)	AVDD (unbuffered)
Buffered Analog Power Supply (VDDX0P8BUF)	AVDD (buffered) x 0.8	AVDD (buffered) x 0.8

Note:

1. High-speed and High-accuracy modes are only available on select devices of EFM32PG23, EFR32xG24, and EFM32PG28. Refer to the device data sheet for details.
2. In High-accuracy mode with VREF2P5 selected, the AVDD supply must be at least 3 V.

The selected reference source determines the full-scale voltage (VFS) for the converter. VFS is the full input range of the converter, from the lowest possible input voltage to the highest. For single-ended conversions, the input range on the selected positive input is from 0 V to VFS.

For differential conversions, the input to the converter is the difference between the positive and negative input. The input range on the differential input is from -VFS to +VFS. A negative voltage with respect to ground should not be applied to the IADC positive input or the IADC negative input.

3.3.1 Analog Input Gain

The IADC has analog input gain selection, controlled via ANALOGGAIN field in IADC_CFGx. The gain can be set to 0.5x, 1x, 2x, 3x or 4x. The 2x, 3x and 4x gain modes require slower ADC_CLK, as shown in the following table. The gain impacts where the full-scale input reading occurs.

Table 3.2. Analog Input Gain Settings

Analog Input Gain	Maximum ADC_CLK	Input Voltage Corresponding to Full Scale Reading (1.25 V External Reference)
0.5x	10 MHz	1.25 V / 0.5 = 2.5 V
1x	10 MHz	1.25 V / 1 = 1.25 V
2x	5 MHz	1.25 V / 2 = 0.625 V
3x	3.3 MHz	1.25 V / 3 = 0.417 V
4x	2.5 MHz	1.25 V / 4 = 0.3125 V

The IADC is only capable of measuring inputs within the supply rails of the device. If the full scale is configured to be greater than the supply voltage, the maximum input will be limited to the supply.

3.4 Input Selection

The IADC supports measurement on a number of internal and external signals. External signals are routed to GPIO through shared ABUS resources on the device (see the *Analog Peripheral Connectivity* section of the device's data sheet).

The single queue and scan queue have separate registers available to select inputs and configurations. The IADC_SINGLE register is used to select the input and configuration for the single queue. The IADC_SCANx registers are used to select the inputs and configurations for each of the scan table entries. In both cases, the register contents and setup are similar.

The PORTPOS and PINPOS fields are used to select a signal for the positive IADC input, while PORTNEG and PINNEG are used to select a signal for the negative IADC input. The CFG field selects which of the two configuration sets will be used with the input (i.e., configuration options specified in IADC_CFGx, IADC_SCALEx, and IADC_SCHEDx).

Note: High-accuracy mode needs dedicated analog bus and a dedicated IADC GPIO for lowest noise.

3.4.1 External Inputs

GPIO input selections are routed through shared ABUS resources. For the IADC to use any GPIO as an input, the IADC must allocate appropriate analog bus resources in the GPIO_ABUSALLOC, GPIO_BBUSALLOC, or GPIO_CDBUSALLOC registers.

For example, if IADC0 will be using both odd and even numbered pins on GPIO port PA, then AEVEN0 and AODD0 in GPIO_ABUSALLOC could both be set to IADC0. This gives IADC0 access to these two buses. Generally, bus access is set to specific peripherals at configuration time and left alone - it is not normally required to change the bus allocation on the fly. If the IADC requests a pin from a bus that has not been allocated to the IADC, an error will be generated, the PORTALLOCERRIF in IADC_IF will be set, and any conversion result will be 0.

When the appropriate analog buses have been configured to route to the IADC, GPIO selection is a simple matter of programming the desired port and pin into the PORTPOS, PINPOS, PORTNEG, and PINNEG fields. For example, to configure a channel to convert the differential voltage between pins PA5, set PORTPOS = PORTA and PINPOS = 5; for PA4, set PORTNEG = PORTA and PINNEG = 4. If an invalid selection is made, a polarity error will be generated.

Note: While all GPIO pins retain their state in EM2, only pins on ports A and B remain fully functional in EM2. So careful pin selection is required for peripherals operating in EM2, as port C and D pins are static in EM2 and cannot be used as peripheral inputs or outputs. Thus, IADC applications must only use pins on ports A and B while in EM2.

3.4.1.1 Single-Ended

For a single-ended input, the input signal is measured with ground as the negative input. The voltage span between 0 V and the selected reference is divided in small steps according to the selected resolution.

The result is an unsigned number between 0 and $2^{resolution} - 1$, indicating where the input voltage is located in the span between ground and the reference voltage.

To perform single-ended conversions, the PORTNEG field should be set to GND. This indicates that the positive IADC input will be measured with reference to chip ground. PORTPOS and PINPOS fields should be used to select the desired input signal. The PINNEG field is not used for single-ended conversions.

3.4.1.2 Differential

For a differential input, the measured value is the difference between two inputs. Since one input is defined as the positive input and the other is defined as the negative input, the difference can be positive or negative depending on which input is higher. As a result, the conversion result is a signed number represented in two's complement form. If the negative input is higher than the positive input, the converted value is negative. A negative voltage with respect to ground should not be applied to the IADC positive input or the IADC negative input.

The table below shows the 12-bit IADC output for various differential input voltages, with the 1.25 V VREF selected as the reference voltage.

Table 3.3. IADC Output Values, REFSEL = VREF

Positive Input (V)	Negative Input (V)	Differential Input (V)	12-bit Output (hex)
+VREF	0 V	+VREF	0x7FF
+VREF/2	0 V	+VREF/2	0x3FF
+VREF/2048	0 V	+VREF/2048	0x001
0 V	0 V	0 V	0x000
0 V	+VREF/2048	-VREF/2048	0xFFF
0 V	+VREF/2	-VREF/2	0xC00
0 V	+VREF	-VREF	0x800

To perform differential conversions, PORTPOS and PINPOS are used to select the positive input to the IADC, while PORTNEG and PINNEG are used to select the negative input. There are two independent muxes in the IADC, and firmware cannot select two signals from the same mux for a differential measurement. The "even" mux consists of all EVEN ABUS selections, supply voltage options, GND, and VREFP. The "odd" mux consists of all ODD ABUS selections, GND, and VREFN. One selection from each mux is allowed on the positive and negative input. More details can be found in the differential input software example: 6.3 Single Conversion, Differential Input.

Note: Unlike the ADC in Series 0 and Series 1, the IADC does not have a DIFF bit that needs to be set to select differential mode. The PINPOS and PINNEG bitfields are available for single and scan modes. PINNEG can be set to GND for single-ended mode or set to any GPIO pin for differential mode.

3.4.2 Internal Inputs and Dedicated Inputs

Internal signals and dedicated inputs are not routed through the shared ABUS resources. In general, these resources are selected directly by the settings of PORTPOS and PORTNEG while the PINPOS and PINNEG fields are not used. When PORTPOS is set to SUPPLY, PINPOS is used to select which of the power supplies is connected.

To facilitate power supply measurements using internal reference options, higher voltage supplies are attenuated by a factor of 4. When selecting SUPPLY for PORTPOS and GND for PORTNEG, PINNEG should be configured for an odd number (1, 3, 5...) to avoid a polarity error. This is already handled when using emlib functions.

Table 3.4. ADC Internal Input Selection for EFR32xG21, EFx32xG22, EFx32xG23, EFR32xG24, and EFx32xG28 Devices

PORTPOS	PINPOS	Supply Connection	Voltage at Positive Input
SUPPLY	0	AVDD	AVDD / 4
	1	IOVDD	IOVDD / 4
	2	VSS	VSS
	3	VSS	VSS
	4	DVDD	DVDD / 4
	6	DECOUPLE	DECOUPLE

Table 3.5. ADC Internal Input Selection for EFR32xG25 Device

PORTPOS	PINPOS	Supply Connection	Voltage at Positive Input
SUPPLY	0	AVDD	AVDD / 4
	1	IOVDD0	IOVDD0 / 4
	2	IOVDD1	IOVDD1 / 4
	3	IOVDD2	IOVDD2 / 4
	4	DVDD	DVDD / 4
	6	DECOUPLE	DECOUPLE

Table 3.6. ADC Internal Input Selection for EFR32xG27 Device

PORTPOS	PINPOS	Supply Connection	Voltage at Positive Input
SUPPLY	0	AVDD	AVDD / 4
	1	IOVDD	IOVDD / 4
	2	VBAT	VBAT / 4
	3	VSS	VSS
	4	DVDD	DVDD / 4
	6	DECOUPLE	DECOUPLE

If an internal signal (PADREFPOS/PADREFNEG) is selected for PORTPOS or PORTNEG, selecting GND on the opposite input will instruct the converter to perform a single-ended conversion. In cases where PORTPOS = GND, the IADC logic will automatically swap the direct input selected by PORTNEG to the positive input of the ADC. Otherwise, a differential conversion is performed with PORTPOS selecting the positive and PORTNEG selecting the negative input.

Usage of dedicated AINx input pins for IADC is recommended when available, and provides improved performance for the following reasons:

- AINx pins are dedicated pads and not powered by IOVDD, therefore these inputs are not subject to the digital switching noise from other GPIO pins that are coupled into the IOVDD power bus.
- AINx pins have reduced impedence as they are 1) physically closer to IADC block on the die, and 2) as dedicated inputs, these connections only go through one level of analog bus switching and are not subjected to a second series RDS_{on} impedance associated with ABUS.

When configuring to measure a single-ended signal through the dedicated pins, the positive input selection should always point to the desired input, and PORTNEG should be programmed to GND. Otherwise, a differential conversion is performed with PORTPOS selecting the positive (PADANA0/PADANA2) and PORTNEG selecting the negative input (PADANA1/PADANA3).

Dedicated pins connect through dedicated analog busses instead of shared busses. The PINPOS/PINNEG field is still important for dedicated analog pads to allow the pad connections to be swapped inside the IADC. In order to use PADANA1(AIN1)/PADANA3(AIN3) dedicated inputs for single-ended conversion, the PORTNEG selection should be programmed to GND and use PINPOS bitfield to "swap" the even and odd muxes at the positive input.

Table 3.7. Some Examples of Port Connections with Dedicated Pins

Positive Input	Negative Input	ABUS Even	ABUS Odd	IADC+	IADC -
PADANA0	PADANA1	dedicated	dedicated	PADANA0	PADANA1
PADANA1	PADANA0	dedicated, swap	dedicated, swap	PADANA1	PADANA0
PADANA1	PADANA3	undefined	undefined	POLARITY_ERROR	—
PADANA0	PADANA2	undefined	undefined	POLARITY_ERROR	—
PADANA0	GND	dedicated	GND	PADANA0	GND
PADANA1	GND	GND	dedicated, swap	PADANA1	GND
GND	PADANA0	undefined	undefined	GND	POLARITY_ERROR
GND	PADANA1	undefined	undefined	GND	POLARITY_ERROR

For example, select devices of EFM32PG23 QFN48 have dedicated AIN0 and AIN1 inputs available for IADC on pin 24 and pin 23 respectively and they can be configured in the following way for the differential inputs:

```
#define IADC_INPUT_0_PORT_PIN   (_IADC_SCAN_PORTPOS_PADANA0
                                 << (_IADC_SCAN_PORTPOS_SHIFT - _IADC_SCAN_PINPOS_SHIFT));
#define IADC_INPUT_1_PORT_PIN   (_IADC_SCAN_PORTNEG_PADANA1
                                 << (_IADC_SCAN_PORTNEG_SHIFT - _IADC_SCAN_PINNEG_SHIFT));
```

Alternatively, the following emlib defines and enums can be used to connect posivite and negative inputs direclty to AIN0 and AIN1 input pin for the differential inputs:

```
IADC_ScanTable_t scanTable = IADC_SCANTABLE_DEFAULT;
scanTable.entries[x].posInput  = iadcPosInputPadAna0;
scanTable.entries[x].negInput  = iadcNegInputPadAna1;
```

Below is the configuration to set AIN1 dedicated input pin for single-ended conversions:

```
IADC_SingleInput_t singleInput = IADC_SINGLEINPUT_DEFAULT;
initSingleInput.posInput = iadcPosInputPadAna0 | 1; // connects to AIN1
initSingleInput.negInput = iadcNegInputGnd;
```

Note:

- Internal signals and dedicated inputs pins are device specific and can be found in the device data sheet.
- High-accuracy mode needs dedicated analog bus and a dedicated analog pin (PADANAx) for lowest noise.
- Highest speed mode needs dedicated analog bus and a dedicated analog pin (PADANAx) for 2 Msps.

3.5 Clock Selection

The IADC logic is partitioned into two clock domains: CLK_BUS (APBIF) and CLK_SRC_ADC (CORE). The APBIF domain contains the IADC registers and FIFO read logic. The rest of the IADC is clocked by the CLK_SRC_ADC and ADC_CLK, both of which are derived from CLK_CMU_ADC. Figure 3.1 IADC Clocking on page 13 displays how the CMU clock trees interact with the IADC module.

If the IADC is to be used synchronously with an external TIMER module, the clock should be configured to derive from Energy Mode 01 Group A Clock (EM01GRPACLK).

If configuring for operation in EM2 or EM3, clock sources available in EM2 and EM3 (such as FSRCO and HFRCOEM23) must be used directly, as the Energy Mode 01 Group A Clock mux will be shut down in EM2 and EM3.

CLK_SRC_ADC is derived from CLK_CMU_ADC, and must be no faster than 40 MHz. The HSCLKRATE field in IADC_CTRL sets the prescaler to divide CLK_CMU_ADC. If CLK_CMU_ADC is already 40 MHz or slower, HSCLKRATE can be set to 0x0 to pass the clock through to CLK_SRC_ADC without dividing it. CLK_SRC_ADC is the clock source used for the TIMEBASE prescaler as well as the local IADC timer.

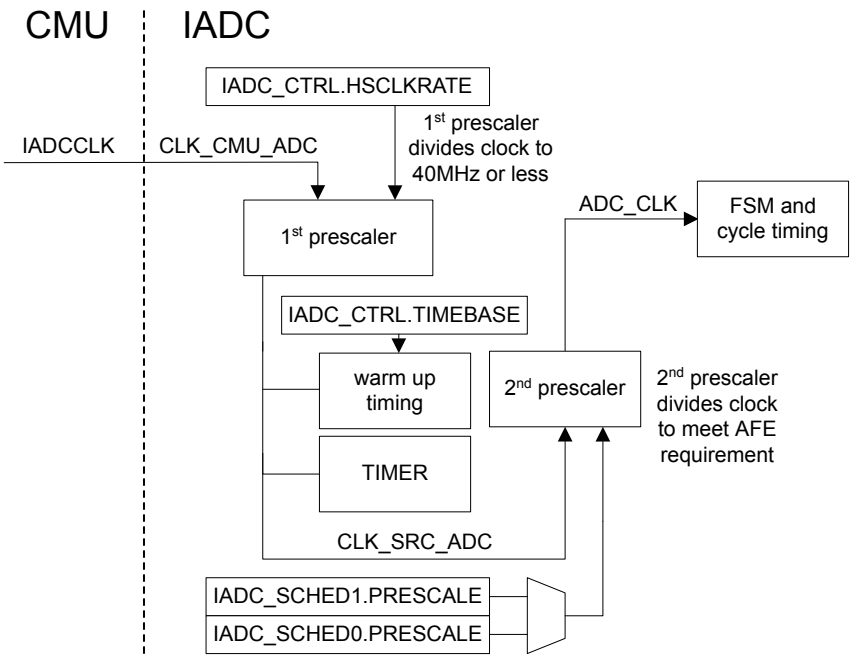


Figure 3.1. IADC Clocking

The suspend mode fields IADC_CTRL_ADCCLKSUSPEND0 (for scan conversions) or IADC_CTRL_ADCCLKSUSPEND1 (for single conversions) can be used to shut down the clock between conversions and save power. The IADC logic will wake up the clock before starting IADC warmup and performing a conversion. If the suspend mode is set, the clock will shut down again once the conversion is complete.

Table 3.8. ADC Clock Information

EFR32 Wireless Gecko Series 2 Clock	Information
CLK_BUS	Clocks the IADC registers and FIFO read logic
CLK_CMU_ADC	Incoming clock route to the IADC by the CMU. Can be up to 80 MHz.

EFR32 Wireless Gecko Series 2 Clock	Information
CLK_SRC_ADC	Derived from CLK_CMU_ADC. Must be no faster than 40 MHz. Used as the clock source for TIMEBASE prescaler and local IADC timer. The HSCLKRATE[1] field in IADC_CTRL sets the prescaler to divide CLK_CMU_ADC.
ADC_CLK	Used to drive the IADC front-end and state machine logic. PRESCALE bitfield in IADC_SCHEDx register must be set to limit ADC_CLK to no faster than the maximum ADC_CLK speeds. Maximum ADC_CLK limitations are based on mode and gain settings, and are listed in Table 3.9 Maximum ADC_CLK Speed vs Analog Gain and ADCMODE Settings on page 14 Also it is recommended to run ADC_CLK no slower than 100 kHz.

Note:

1. If HSCLKRATE does not equal 0, then PRESCALE must not be set to 0 (i.e., divide by 1). When this condition is detected, the programmed PRESCALE value will be ignored and PRESCALE = 1 will be used.

Table 3.9. Maximum ADC_CLK Speed vs Analog Gain and ADCMODE Settings

Analog Gain	CFGx.ADCMODE = NORMAL	CFGx.ADCMODE = HIGHSPEED[1]	CFGx.ADCMODE = HIGHACCURACY[1]
0.5 x	10 MHz	20 MHz	5 MHz
1 x	10 MHz	20 MHz	5 MHz
2 x	5 MHz	10 MHz	5 MHz
3 x	2.5 MHz	5 MHz	5 MHz
4 x	2.5 MHz	5 MHz	5 MHz

Note:

1. HIGHSPEED and HIGHACCURACY modes are only available on select devices of EFM32PG23, EFR32xG24, and EFM32PG28. Refer to the device data sheet for details.

4. Conversion Process

To produce one result, the IADC takes multiple samples of the analog signal. The number of input samples contributing to an output word is determined by the oversampling ratio (OSR). Higher OSR settings will improve the ADC's INL and DNL and reduce system-level noise, but require more time for each conversion. For Normal and High-Speed[1] modes, the OSR is configured with the OSRHS bitfield in the IADC_CFGx register. The OSR options for High-Accuracy[1] mode are different, and are configured with the OSRHA bit field in the IADC_CFGx register. Different OSRs may be specified for each configuration group. Oversampling is an analog process (pre-digital filter).

During a conversion, the effective front-end sampling frequency (F_{sample}) in Normal and High-Speed[1] modes is equal to ADC_CLK / 4. In High-Accuracy[1] mode, F_{sample} is ADC_CLK / 5.

On all devices except EFR32xG21, in addtion to analog oversampling, IADC also includes digital accumulation and averaging. The IADC may optionally accumulate and average several conversion results before posting an output word to the FIFO. Digital averaging is controlled by the DIGAVG field in the IADC_CFGx register. It can be configured to average 1, 2, 4, 8, or 16 samples. The IADC will collect the number of samples specified by DIGAVG on the selected channel slot back-to-back, and produce only one averaged output word.

4.1 Warmup Mode

To save energy, the IADC can be configured to power down completely or enter a standby state between conversions. The required IADC warm up time from a full powered-down state is 5 µs. Warmup from a standby state requires 1 µs.

Warmup is automatically timed by the IADC logic when it is required, but software must configure the TIMEBASE field in IADC_CTRL for a minimum 1 µs interval. The TIMEBASE counter receives CLK_SRC_ADC, and should be programmed based on that frequency.

The TIMEBASE can be calculated using the following formula:

TIMEBASE = [Warmup time (in µs) x CLK_SRC_ADC (in MHz)] - 1

For example, if CLK_SRC_ADC is 40 MHz, TIMEBASE should be set to at least 0x27 (39) to produce the minimum 1 µs interval. When transitioning from a powered-down state, the IADC will use five TIMEBASE intervals. When in standby, the IADC will use one TIMEBASE interval.

The WARMUPMODE field in the IADC_CTRL register defines whether the IADC is powered down between conversions (WARMUPMODE = NORMAL), in standby between conversions (WARMUPMODE = KEEPINSTANDBY), or remains powered up (WARMUPMODE = KEEPWARM). The resulting startup time is shown in Figure 4.2 Startup Timing on page 15. Even in WARMUPMODE = KEEPWARM or KEEPINSTANDBY, the ADC will implement five TIMEBASE intervals of warmup on initial power up, or any configuration change affecting PRESCALE, ADCMODE, or REFSEL. IADC_STATUS_ADCWARM reflects the current warmup status of the IADC.

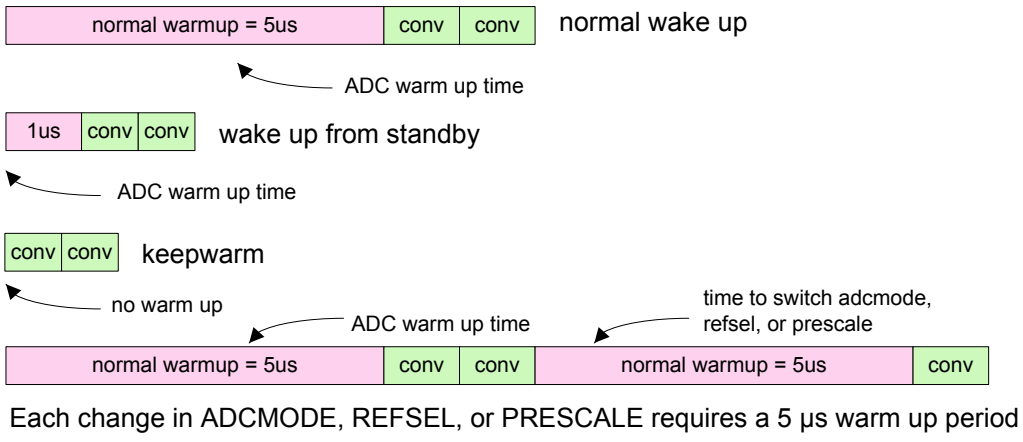


Figure 4.2. Startup Timing

4.2 Conversion Pipeline in Normal and High-Speed Modes

The IADC uses a pipelined architecture to perform different stages of the ADC conversion in parallel. The conversion time for a single sample can be determined from the OSR and the prescaled ADC_CLK frequency (f ADC_CLK) as:

$$\text{Conversion Time} = ((4 \times \text{OSR}) + 2) / f_{\text{ADC\_CLK}}$$

The minimum OSR is 2, meaning that the fastest possible conversion lasts 10 ADC_CLK clock cycles.

The IADC will automatically insert two additional cycles in the pipeline when changing channels to a new GPIO. This allows for hold timing on the previous conversion and allows for time to tristate the ABUS analog busses before connecting the next GPIO to the analog bus. Therefore, the maximum sampling rate while continuously sampling on one channel (with ADC_CLK = 10 MHz) in Normal mode is 1 Msps, and the maximum sampling rate while switching channels is 833 ksps.

In High-Speed[1] mode the allowed ADC_CLK speed is doubled to 20 MHz. The maximum sampling rate while continunously converting a single channel is 2 Msps, and the maximum sampling rate while switching channels is 1.67 Msps.

Figure 4.3 Normal Mode ADC Pipeline on page 16 and Figure 4.4 High-Speed Mode ADC Pipeline on page 17 shows both single-channel and channel-switching scenarios powering up from a shutdown state with WARMUPMODE = NORMAL. The 5 µs warmup is shown in pink, a first conversion pipeline in green, and a second conversion in orange. The blue area in the top diagram represents the extra time to tristate while changing channels.

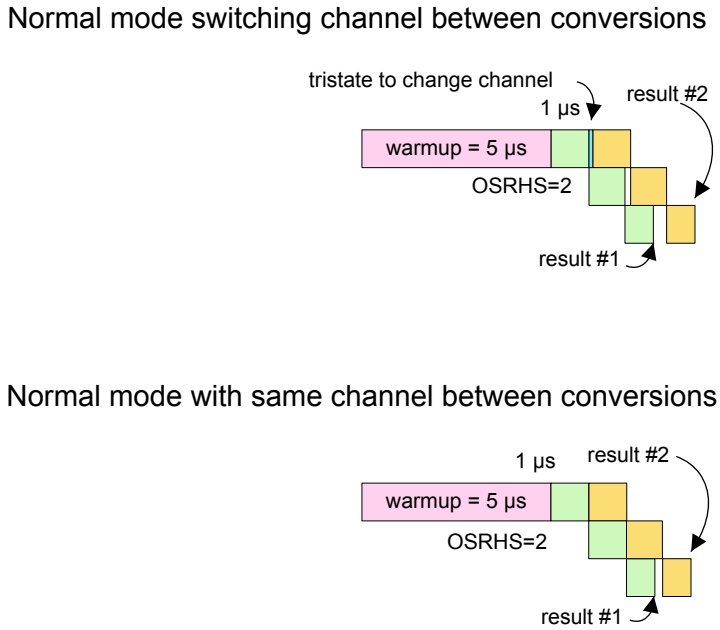


Figure 4.3. Normal Mode ADC Pipeline

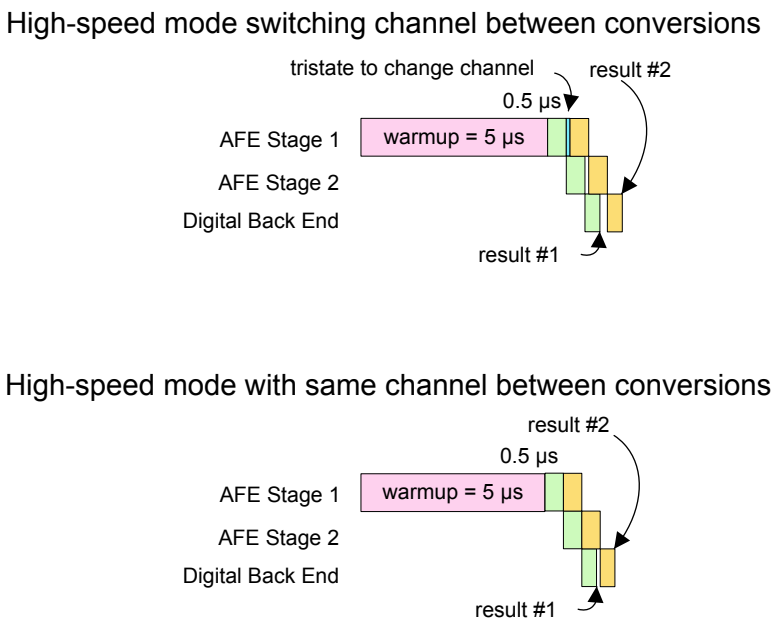


Figure 4.4. High-Speed Mode ADC Pipeline

4.3 Conversion Pipeline in High-Accuracy Mode

In High-accuracy[1] mode, the conversion time for a single sample can be determined from the OSR and the pre-scaled ADC_CLK frequency (f_{ADC_CLK}) as:

Conversion Time = ((5 * OSR) + 7) / f_{ADC_CLK}

The minimum OSR is 16, meaning that the fastest possible conversion in High-accuracy mode lasts 87 ADC_CLK clock cycles.

In High-accuracy mode, there are no additional cycles required to change channels to a new input. The figure below shows the pipeline timing when powering up from a shutdown state with WARMUPMODE = NORMAL. The 5 µs warmup is shown in pink, a first conversion pipeline in green, and a second conversion in orange.

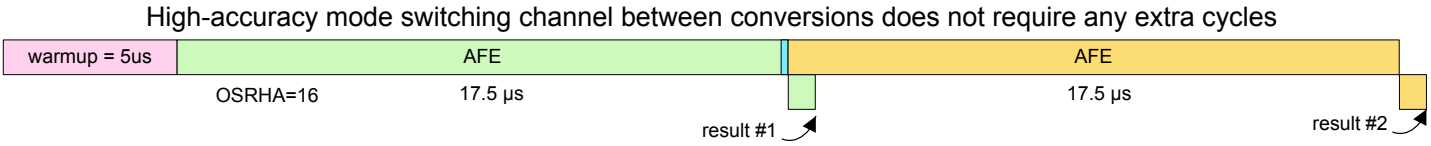


Figure 4.4. High-Accuracy Mode ADC Pipeline

4.4 Scheduling and Triggers

The IADC has several triggering options available for both the single queue and the scan queue. When a conversion trigger occurs and there are no other conversions active or pending, the request is serviced immediately. If both the single and scan queues are being used in an application, it is possible to serve the conversion requests as needed, and specify their priority.

Conversion triggering is configured using bitfields in the IADC_TRIGGER register. The SINGLETRIGSEL and SCANTRIGSEL fields specify the trigger source for Single and Scan conversion queues, respectively. The options for trigger source are:

- IMMEDIATE - Trigger from software. This is useful for triggering conversions on-demand from software with no specific sampling frequency requirements, or initiating continuous conversions at full speed.
- TIMER - Use the IADC local timer to trigger conversions. This is useful for triggering conversions at precise intervals.
- PRSCLKGRP, PRSPOS, and PRSNEG - peripheral reflex system (PRS) signals from other peripherals. See 5.1 Peripheral Reflex System for more details.
- LESENSE (SCAN Only) - Trigger from the LESENSE peripheral. When using this mode, only one entry in the SCAN table (specified by the LESENSE channel) is converted per conversion request, and the SCAN queue is unavailable for normal operation. See 5.2 LESENSE Interface for more details.

Both the single and scan trigger sources can be configured to generate one request per trigger or begin continuous conversions, if not specified. Setting SINGLETRIGACTION to ONCE in the IADC_TRIGGER register will make one conversion request each time the selected single trigger occurs, and a single ADC output will be converted. Setting SINGLETRIGACTION to CONTINUOUS in the IADC_TRIGGER register allows the single trigger to begin the first conversion, and when a conversion completes a new one will be requested immediately without requiring a new trigger.

Note: Channel selections and configuration should not be changed while SINGLETRIGACTION is set to CONTINUOUS. Doing so can produce conversion errors. The scan queue (See 3.2.2 Scan Mode for details) should be used if channel or configuration switching is required.

The SCANTRIGACTION field works to request conversion scans in a similar manner. Setting SCANTRIGACTION to ONCE in the IADC_TRIGGER register will make one request each time the selected scan trigger occurs, and the IADC will perform all conversions specified in the scan once before stopping. Setting SCANTRIGACTION to CONTINUOUS allows the scan trigger to initiate continuous scans. When a scan cycle completes, a new one will be requested immediately without requiring a new trigger.

Conversion priority can be adjusted using the SINGLETAILGATE bit in the IADC_TRIGGER register. By default, SINGLETAILGATE is set to TAILGATEOFF, meaning that conversion triggers are queued in the order they are received. Any conversion trigger for the Single queue or the Scan queue will initiate a conversion as soon as possible. If any conversion is already in progress or pending, the new conversion will be handled after the current operation.

Setting SINGLETAILGATE to TAILGATEON gives ultimate priority to the Scan queue. The IADC will only perform single conversions immediately after completion of a scan. This allows systems to use the scan queue for high-priority conversions with tight timing requirements, and the single queue for low-priority, on-demand conversion events.

The TAILGATEON setting should only be used when scan conversions are guaranteed to trigger. If no scan sequence is triggered, any single conversion trigger will remain pending indefinitely. Furthermore, if there is not enough time between scan conversions to perform a single conversion, the next scan conversion will be delayed.

5. Advanced Features

5.1 Peripheral Reflex System

The IADC can be configured as both a consumer and a producer of PRS signals.

As a PRS consumer, IADC single and scan conversions can be triggered by a PRS signal if the software trigger and local IADC timer trigger are not desired. The available PRS triggers are shown below, and they are selectable using the SINGLETRIGSEL and SCANTRIGSEL fields in the IADC_TRIGGER register:

- PRSCLKGRP - Use a synchronous PRS channel to trigger from an external peripheral in the same clock group domain (i.e., clock-group A). This is useful for synchronizing conversions precisely with external TIMER events or PWM outputs.

  Note: Configure the PRS consumer registers prior to enabling synchronous PRS triggers to avoid false triggers.

- PRSPOS - Use a positive edge of an asynchronous PRS channel to trigger conversions. The trigger source will require 1-2 ADC_SRC_CLK cycles to synchronize. This is useful for triggering conversions as needed from asynchronous peripheral sources-such as GPIO inputs, RTC events, etc.
- PRSNEG - Use a negative edge of an asynchronous PRS channel to trigger conversions. This is the same as PRSPOS, but operates on negative

As PRS producer, the IADC can generate the following PRS signals, allowing other peripherals to take action when the selected IADC conversion event has occured:

- SINGLEDONE
- SCANENTRYDONE
- SCANTABLEDONE

5.2 LESENSE Interface

The LESENSE peripheral can be set up to trigger IADC0 conversions and use data from IADC0 to evaluate sensor status. The channel scanner hardware is used by LESENSE in this mode and the IADC SCAN trigger must be set to LESENSE. The SCAN queue can only be used for LESENSE when operated in this mode, but the SINGLE queue may still be used independently.

When a LESENSE channel is active, the OFFSET field in LESENSE_CHx_INTERACT field is used to determine which of the ADC's 16 scanner channels is sampled. OFFSET = 0 corresponds to IADC_SCAN0, OFFSET = 1 corresponds to IADC_SCAN1, and so on.

The IADC sample is triggered when the sample delay configured in LESENSE_CHx_TIMING_SAMPLEDLY has expired. Results from the conversion are sent to the LESENSE result FIFO for further processing by LESENSE, and are not available in the SCAN FIFO.

Note: LESENSE peripheral is only available on select devices. Refer to the device data sheet for details.

5.3 Linked Direct Memory Access

The IADC has two LDMA request lines, IADC single request and IADC scan request, which are set when a single or scan FIFO reaches DVL# of samples. The requests are cleared when the corresponding single or scan result register is read and corresponding FIFO count is below DVL.

5.4 Interrupts

Interrupts are enabled in the IADC_IEN register, allowing interrupts to be generated on several different IADC conditions. Each of the flags in IADC_IF has a corresponding enable bit in the IADC_IEN register. A brief overview of the available interrupt sources is shown in the list below.

- SINGLEFIFODVL - The single FIFO watermark specified in SINGLEFIFOCFG_DVL has been reached or exceeded.
- SCANFIFODVL - The scan FIFO watermark specified in SCANFIFOCFG_DVL has been reached or exceeded.
- SINGLECMP - A conversion result from the single queue tripped the window comparator.
- SCANCMP - A conversion result from the scan queue tripped the window comparator.
- SCANENTRYDONE - One entry in the scan table has been converted.
- SCANTABLEDONE - All entries in the scan table have been converted once.
- SINGLEDONE - A single conversion has been completed.
- POLARITYERR - A channel polarity selection error has occurred, i.e. two channels from the EVEN mux or two channels from the ODD mux were selected for positive and negative inputs.
- PORTALLOCERR - A port allocation error has occurred, i.e., a pin not allocated to the IADC in the GPIO bus allocation registers was requested.
- SINGLEFIFOOF - A single FIFO overflow has occurred.
- SCANFIFOOF - A scan FIFO overflow has occurred.
- SINGLEFIFOUF - A single FIFO underflow has occurred.
- SCANFIFOUF - A scan FIFO underflow has occurred.
- EM23ABORTERROR - The system entered EM2 or EM3 while the IADC was converting and using a clock not supported in EM2 or EM3.

Hardware sets the interrupt flags in IADC_IF, and the flags remain set (sticky) until cleared by software. The interrupts flags should be cleared before enabling the IADC to remove any previous interrupt history. Clearing or setting interrupt bits can be done by writing to IADC_IF with a set or clear mask.

5.5 Gain and Offset Correction

The IADC has built-in gain and offset correction capabilites. Each of the two configuration groups contains its own correction values stored in the IADC_SCALEx register, allowing the IADC to automatically apply the appropriate correction for the IADC configuration that is being used.

Gain correction is performed through a fixed-point 16-bit value with a range from 0.75x to 1.2499x. The 3 MSBs of the gain value are not directly writeable. The GAIN3MSB bit in IADC_SCALEx is used to select between 011 and 100 for the 3 MSBs, and the lower 13 bits are programmed directly into IADC_SCALEx_GAIN13LSB. Clearing GAIN3MSB to 0 selects the most significant bits of the gain as 011, representing a range from 0.75x to 0.9999x. Setting GAIN3MSB to 1 selects the most significant bits of the gain as 100, representing a range from 1.00x to 1.2499x.

Offset correction is controlled by the OFFSET field in IADC_SCALEx. The offset correction does not have a direct 1-to-1 relationship with the LSB of the IADC output, and depends on both the OSR and gain correction settings. The offset correction range is +/- 12.5% of full scale. OFFSET is encoded as a 2's complement, 18-bit number with the LSB representing $1 / 2^{20}$ of full scale. Thus, bit 8 of OFFSET aligns with bit 0 of the 12-bit IADC output word.

IADC calibration is performed on every device during Silicon Labs production test, and production calibration parameters are stored in the flash DI page. The production calibration values are useful for a wide variety of possible IADC configurations, but do not map directly to the offset and gain correction fields in the IADC_SCALEx registers. Software must calculate the actual offset and gain correction values from the factory calibration values. The following sections explain this in detail.

5.5.1 Gain Correction

Gain error is measured during production test at various settings of ANALOGGAIN and stored in the DEVINFO_IADC0GAIN0 and DEVINFO_IADC0GAIN1 locations. The GAINCANA1 field is used for 0.5x and 1x ANALOGGAIN settings, while GAINCANA2, GAINCANA3, and GANCANA4 are used for ANALOGGAIN settings of 2x, 3x, and 4x, respectively.

The GAINCANAn values are expressed as the full 16-bit fixed-point gain and must be compressed before writing to the IADC_SCALEx register.

5.5.1.1 Gain Correction in Normal and High-speed Modes

When the IADC is operated in Normal / High-speed[1] mode, a 1st order filter is employed in the decimation. The nominal gain value in these modes for all OSRHS settings is 1.0, or 0x8000 as expressed in the fixed-point 16-bit format. The IADC gain error is designed to be minimal with the digital gain correction set to 1.0 (GAIN3MSB = 1 and GAIN13LSB = 0). Tighter gain error is achieved by adjusting these values in IADC_SCALEx. Using this gain correction mechanism will result in a slight increase to the DNL of the converter, which is reduced by higher OSR settings.

To apply a factory-calibrated gain:

1. Read the appropriate GAINCANAn field from the DEVINFO locations for the selected ANALOGGAIN.
2. Write the MSB (bit 15) of GAINCANAn to GAIN3MSB in IADC_SCALEx.
3. Write the 13 LSBs (bits 12-0) of GAINCANAn to GAIN13LSB in IADC_SCALEx.

5.5.1.2 Gain Correction in High-accuracy Mode

When the IADC is operated in High-accuracy[1] mode, a 2nd order filter is employed in the decimation. The nominal gain value of the filter is dependent on the OSRHA setting. The gain value stored in DEVINFO space must be adjusted before applying to the IADC_SCALEx register.

To apply a factory-calibrated gain:

1. Read the appropriate GAINCANAn field from the DEVINFO locations for the selected ANALOGGAIN.
2. Multiply the value by the OSR gain correction factor (ha_gain) found in Table 5.1 Ideal High-accuracy Gain Correction on page 21.
3. Write the MSB (bit 15) of the result to GAIN3MSB in IADC_SCALEx.
4. Write the 13 LSBs (bits 12-0) of the result to GAIN13LSB in IADC_SCALEx.

Table 5.1. Ideal High-accuracy Gain Correction

OSRHA Setting	OSR	16-bit OSR Gain Correction (ha_gain)
HIACC16	16 x	0x7879
HIACC32	32 x	0x7C1F
HIACC64	64 x	0x7E08
HIACC92	92 x	0x7A8E
HIACC128	128 x	0x7F02
HIACC256	256 x	0x7F80

5.5.2 Offset Correction

Offset is impacted by the selected ANALOGGAIN and OSR settings in IADC_CFGx, the GAIN3MSB and GAIN13LSB values in IADC_SCALEx, and the voltage reference. Offset is production calibrated for any combination of possibilities, but the OFFSET register value must be calculated for the given situation before it can be effectively used.

5.5.2.1 Offset Correction in Normal Mode

The production offset calibration consists of four 16-bit terms written to the DEVICEINFO space in the IADCOFFSETCAL0 and IADCOFFSETCAL1 locations: OFFSETANA1NORM, OFFSETANA2NORM, OFFSETANA3NORM, and OFFSETANABASE. The following procedures will determine the setting for the OFFSET register based on production calibration values.

Step 1: Determine the offset gain adjustment term (off_gain) based on ANALOGGAIN.

For ANALOGGAIN, set to 0.5x or 1x:

off_gain = 0

For ANALOGGAIN, set to 2x, 3x, or 4x, off_gain is calculated as:

off_gain = OFFSETANA2NORM x (gain - 1)

This is summarized in the following table.

Table 5.2. Offset Gain Adjustment

ANALOGGAIN Setting	Analog Front-end Gain	Offset Gain Adjustment Term (off_gain)
ANAGAIN0P5	0.5 x	0
ANAGAIN1	1 x	0
ANAGAIN2	2 x	OFFSETANA2NORM x 1
ANAGAIN3	3 x	OFFSETANA2NORM x 2
ANAGAIN4	4 x	OFFSETANA2NORM x 3

Step 2: Calculate the analog offset adjustment term (off_ana) based on OSR and off_gain.

For an OSR of 2x (OSRHS = 0):

off_ana = OFFSETANA1NORM + off_gain

For all other OSR settings, 4x - 64x:

off_ana = OFFSETANABASE + 2 x (OFFSETANA3NORM - off_gain)/OSR

The following table expresses these equations:

Table 5.3. Analog Offset Adjustment

OSRHS Setting	OSR	Analog Offset Adjustment Term (off_ana)
HISPD2	2 x	OFFSETANA1NORM + off_gain
HISPD4	4 x	OFFSETANABASE + (OFFSETANA3NORM - off_gain)/2
HISPD8	8 x	OFFSETANABASE + (OFFSETANA3NORM - off_gain)/4
HISPD16	16 x	OFFSETANABASE + (OFFSETANA3NORM - off_gain)/8
HISPD32	32 x	OFFSETANABASE + (OFFSETANA3NORM - off_gain)/16
HISPD64	64 x	OFFSETANABASE + (OFFSETANA3NORM - off_gain)/32

Step 3: Compensate for reference voltage differences.

The off_ana term represents the offset at the input of the ADC, meaning that the reference voltage will have an impact on the magnitude of the offset at the output. Production calibration values are determined with a 1.25 V reference source. If a voltage significantly different than 1.25 V is used for V_{REF}, adjust the off_ana term by a factor of 1.25 / V_{REF}.

off_ana = off_ana x (1.25 / V_{REF})

Step 4: Calculate total offset by adding the analog offset to the systematic offset.

Systematic offset is a fixed number dependent on OSR, and calculated according to the following equation:

off_sys = 640 x (256/OSR)

Total uncorrected offset (off_tot) is calculated by:

off_tot = (off_ana x 4 + off_sys)

Step 5: Apply gain error correction, if needed.

Before writing the OFFSET field, the total uncorrected offset must be multiplied by the gain calibration factor. If the gain calibration factor is equal to 1.0 (0x8000 in 16-bit hex, or GAIN3MSB = 1 and GAIN13LSB = 0), this step may be skipped. Otherwise, adjust off_tot according to the following equation:

off_tot = GAIN_FACTOR * (off_tot + 0x80000) - 0x80000

where GAIN_FACTOR = GAINCANAn / 32768.

Step 6: Write the offset correction value to the OFFSET field.

The OFFSET field holds an 18-bit 2's complement number, which should be the negation of the total offset, or -(off_tot). Before writing to the SCALE register, any leading sign bits should be masked off to avoid corrupting the programmed gain settings.

OFFSET = 0x3FFFF & (-off_tot)

5.5.2.2 Offset Correction in High Speed Mode

Offset correction for High-speed[1] mode is identical to the procedure for Normal mode, with the exception of the calibration terms used in the DEVINFO space. The same OFFSETANABASE term is used, but OFFSETANA1HISPD, OFFSETANA2HISPD, and OFFSETANA3HISPD are used instead of the OFFSETANAxNORM values. Refer to 5.5.2.1 Offset Correction in Normal Mode for the procedure, replacing OFFSETANAxNORM with OFFSETANAxHISPD.

5.5.2.3 Offset Correction in High-accuracy Mode

The production offset calibration for High-accuracy[1] mode uses two 16-bit terms written to the DEVINFO space: OFFSETANA1HIACC and OFFSETANABASE. The following procedure will determine the setting for the OFFSET register based on production calibration values.

Step 1: Calculate the analog offset adjustment term (off_ana) based on the OSR setting (OSRHA in IADC_CFGn).

off_ana = OFFSETANABASE + OFFSETANA1HIACC / (2^{OSRHA})

The following table expresses this relationship:

Table 5.4. Analog Offset Adjustment

OSRHA Setting	OSR	Analog Offset Adjustment Term (off_ana)
HIACC16	16 x	OFFSETANABASE + OFFSETANA1HIACC
HIACC32	32 x	OFFSETANABASE + (OFFSETANA1HIACC / 2)
HIACC64	64 x	OFFSETANABASE + (OFFSETANA1HIACC / 4)
HIACC92	92 x	OFFSETANABASE + (OFFSETANA1HIACC / 8)
HIACC128	128 x	OFFSETANABASE + (OFFSETANA1HIACC / 16)
HIACC256	256 x	OFFSETANABASE + (OFFSETANA1HIACC / 32)

Step 2: Compensate for reference voltage differences.

The off_ana term represents the offset at the input of the ADC, meaning that the reference voltage will have an impact on the magnitude of the offset at the output. Production calibration values are determined with a 1.25 V reference source. If a voltage significantly different than 1.25 V is used for V_{REF}, adjust the off_ana term by a factor of 1.25 / V_{REF}.

off_ana = off_ana * (1.25 / V_{REF})

Step 3: Calculate total offset by adding the analog offset to the systematic offset.

Systematic offset is a fixed number dependent on OSR, and calculated according to the following equation:

off_sys = 0x40000 / (OSR / (OSR+1))

Total uncorrected offset (off_tot) is calculated by:

off_tot = (off_ana * 4 + off_sys)

Step 4: Apply gain error correction.

Before writing the OFFSET field, the total uncorrected offset must be multiplied by the gain calibration factor according to the following equation:

off_tot = GAIN_FACTOR * (off_tot + 0x80000) - 0x80000

where GAIN_FACTOR = GAINCANAn / 32768.

Step 5: Write the offset correction value to the OFFSET field.

The OFFSET field holds an 18-bit 2's complement number, which should be the negation of the total offset, or -(off_tot). Before writing to the SCALE register, any leading sign bits should be masked off to avoid corrupting the programmed gain settings.

OFFSET = 0x3FFFF & (-off_tot)

5.5.3 Calibration

Calibration can be performed in-system to correct for external errors and provide more accurate measurements. The general calibration procedure is as follows:

1. Configure the ADC to the desired mode, OSR, analog gain settings, reference source, etc.
2. Force the IADC to use bipolar output for the conversion: IADC_CFGx_TWOSCOMPL = FORCEBIPOLAR.
3. Set the initial offset to the maximum negative value (IADC_SCALEx_OFFSET = 0x20000), and the initial gain to 1.0 (GAIN3MSB =1, GAIN13LSB = 0x0000). This will prevent output saturation when measuring full scale.
4. Apply a full-scale positive input to the IADC and perform a conversion (*result_fullscale*). Multiple conversions can be performed and averaged together to reduce any system-level noise.
5. Apply a zero input to the IADC and perform a conversion (*result_zero*). Multiple conversions can be performed and averaged together to reduce any system-level noise.
6. Calculate the gain correction factor: Divide the expected value by the difference in the measured values (*result_fullscale - result_zero*). The offset adjustment in Step 3 will be canceled out by this calculation.
7. Write the gain correction factor to the IADC using the GAIN3MSB and GAIN13LSB fields in IADC_SCALEx.
8. Set IADC_SCALEx_OFFSET to 0x00000 in preparation for the offset calibration.
9. Apply the desired zero voltage to the IADC input and perform a converison (*result_offset*). Multiple conversions can be performed and averaged together to reduce any system-level noise.
10. Multiply *result_offset* to convert to a 20-bit value (*result_offset_20*). For example, a 12-bit result should be multipled by 256.
11. Negate *result_offset_20* and write the value to IADC_SCALEx_OFFSET.

The IADC_SCALEx_OFFSET field is 18 bits. If the result is greater than (2^17 - 1) or less than (-2^17), the offset is too large to be corrected.

A software example demonstrating calibration is provided, see 6.4 Single Conversion, Calibration.

5.6 Output Data FIFOs

Conversion results are written to the output data FIFO associated with the queue. Single queue results are written to the single data FIFO and scan queue results are written to the scan data FIFO. The two queues are identical in operation, but independent. Each FIFO can store up to four entries.

Conversion results are read from the single FIFO using IADC_SINGLEFIFODATA. Reading SINGLEFIFODATA will pop the oldest result from the FIFO. It is also possible to read the most recent valid data word using IADC_SINGLEDATA. Reading SINGLEDATA does not pop a conversion from the FIFO. Similarly, the scan FIFO results are read with IADC_SCANFIFODATA, which reads the oldest result and pops the FIFO. The most recent scan result can be read using IADC_SCANDATA.

When the single FIFO has valid data, the SINGLEFIFODV flag in IADC_STATUS is set to 1. When the scan FIFO has valid data SCANFIFODV in IADC_STATUS is set to 1. These data valid status bits are cleared automatically whenever the associated FIFO is empty. For more granular FIFO status, the number of data words present in the FIFO is indicated in IADC_SINGLEFIFOSTAT (for single FIFO) or IADC_SCANFIFOSTAT (for scan FIFO).

For our EFR32xG22 devices, there are two additional status bits made available in the IADC_STATUS register. These include the SINGLEDATADV and the SCANDATADV bits, which get set when new data is available and cleared when the SINGLEDATA or SCANDATA registers are read, respectively.

A programmable data level watermark is also available for the FIFOs, allowing hardware to trigger interrupts or DMA operations when a specified number of conversion results are available. The DVL field in register SINGLEFIFOCFG or SCANFIFOCFG sets the watermark level, between 1 and 4 conversions. If the number of valid entries in the FIFO exceeds the level set in DVL, the SINGLEFIFODVLIF (for single FIFO) or SCANFIFODVLIF (for scan FIFO) flag in the IADC_IF register will be set to 1. If enabled, an interrupt or DMA request will be triggered when the flag is set.

By default, DMA requests are turned off for operation in EM2 or EM3. However, the DMAWUFIFOSINGLE or DMAWUFIFOSCAN bits in SINGLEFIFOCFG or SCANFIFOCFG may be used to enable DMA operations in these lower energy modes.

Overflow and underflow status flags are also available in IADC_IF. An overflow condition occurs when an IADC conversion completes, but the associated FIFO is already full. In an overflow case the SINGLEFIFOOFIF or SCANFIFOOFIF flag will be set. The most recent conversion will still be available in the SINGLEDATA or SCANDATA register, but the FIFO will not be updated with the new data. An underflow condition occurs when software or hardware attempts to read from an empty FIFO. In an underflow case the SINGLEFIFOUFIF or SCANFIFOUFIF flag will be set.

5.6.1 Data Alignment and Channel ID

The IADC has data alignment options and the ability to include a channel ID along with the conversion data. For the single queue, alignment and channel ID are configured in the IADC_SINGLEFIFOCFG register. For the scan queue, alignment and channel ID are configured in the IADC_SCANFIFOCFG register.

The ALIGNMENT bitfield specifies the data justification and the number of data bits as shown in Figure 5.1 Data Alignment on page 26. By default, the converter will produce 12-bit right-justified data, corresponding to ALIGNMENT = RIGHT12.

All devices except EFR32xG21 have additional data alignment options for 16-bit and 20-bit data. This allows for accumulation and additional resolution with the use of the front-end OSR.

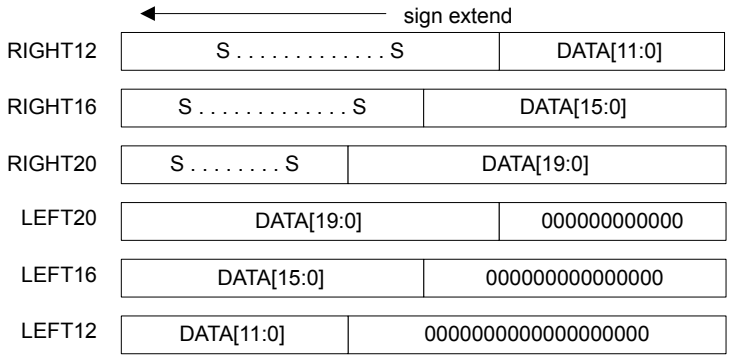


Figure 5.1. Data Alignment

The SHOWID bit controls whether the conversion channel ID is included in the output data word. This option is primarily used with the scan FIFO to help software determine which channel each conversion result came from. If SHOWID is enabled for single conversions, the ID will always be set to 0x20. Figure 5.2 Data Alignment With ID on page 26 shows output data formatting including the ID, when SHOWID = 1.

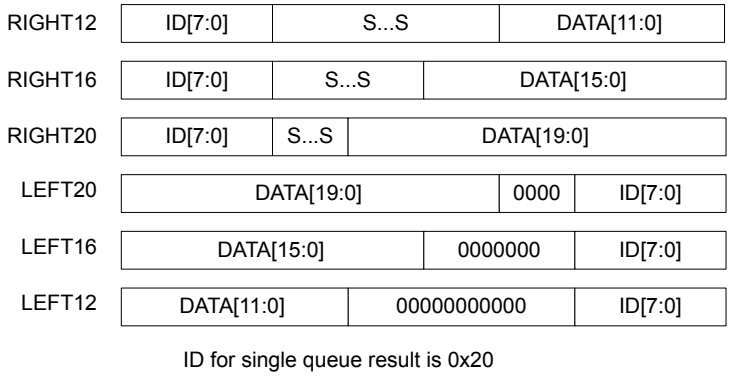


Figure 5.2. Data Alignment With ID

5.6.2 Output Resolution

The usable output resolution of the IADC is a minimum of 12 bits, when the oversampling ratio is set to 2 and no digital averaging is used (DIGAVG = AVG1). For all devices except EFR32xG21, an extra bit of output resolution is produced for every power of 2 increase in either of these settings. In other words, the output resolution of the ADC can be determined as:

Output Resolution = 11 + $\log_2$(OversamplingRatio × DigitalAveraging)

The MSB is always left-aligned within the DATA field, and the output word will be truncated to 12, 16, or 20 bits, as shown in Figure 5.1 Data Alignment on page 26 and Figure 5.2 Data Alignment With ID on page 26. When using 16 or 20 bit alignment with lower oversampling ratio and digital averaging settings, LSBs of the output can contain residual effects of the offset and gain computation. These residual effects do not represent additional information about the input signal. Any extra LSBs can be masked to 0 by software.

Table 5.5. Output Resolution Masking Examples

Alignment Setting	Oversampling Ratio	Digital Averaging	Number of averaged samples	Output Resolution	Recommended Mask for DATA field
16-bit	2x	1x	2	12 bits	0xFFF0
16-bit	8x	2x	16	15 bits	0xFFFE
20-bit	2x	1x	2	12 bits	0xFFF00
20-bit	16x	4x	64	17 bits	0xFFFF8

5.7 Window Compare

The IADC has a window comparison unit that can trigger interrupts conditional on the output data of the converter. The window comparison unit has two thresholds, a greater than or equal (ADGT) and a less than or equal (ADLT), which are programmable through the IADC_CMPTHR register. The ADGT and ADLT thresholds always use a 16-bit, left-justified format, regardless of the format specified by the FIFO. The 12-bit conversion result will be compared against the upper 12 bits of the threshold. The lower 4 bits of the threshold are ignored.

The window comparison unit is active on the ADC output on a conversion-by-conversion basis, and is shared between the two FIFOs. It is not possible to set different window comparison thresholds for different channels or for each FIFO. However, each channel specified in the IADC has a CMP bitfield to enable the window comparison on results from that channel. For example, it is possible to apply the window comparison and associated interrupt only to scan channel #3 by setting the CMP field in IADC_SCAN3 to 1. When the CMP field associated with a channel is 0, the window comparator will not be active for result from that channel.

The window comparator supports conditional triggering on output results which are inside or outside a specified window. When ADLT is greater than or equal to ADGT, the comparator will trigger on an "inside" condition, or when DATA <= ADLT **and** DATA >= GT. When ADLT is less than ADGT, the comparator will trigger on an "outside" condition, or when DATA <= ADLT **or** DATA >= GT.

Figure 5.3 Window Comparison Examples on page 28 shows different configurations of the ADLT and ADGT values and the resulting windows. When the window comparator detects that the appropriate conditions are met (shown by the shaded region in the figure), it will generate an interrupt via the SINGLECMPIF flag for conversions on the single queue, or via the SCANCMPIF flag for conversions on the scan queue.

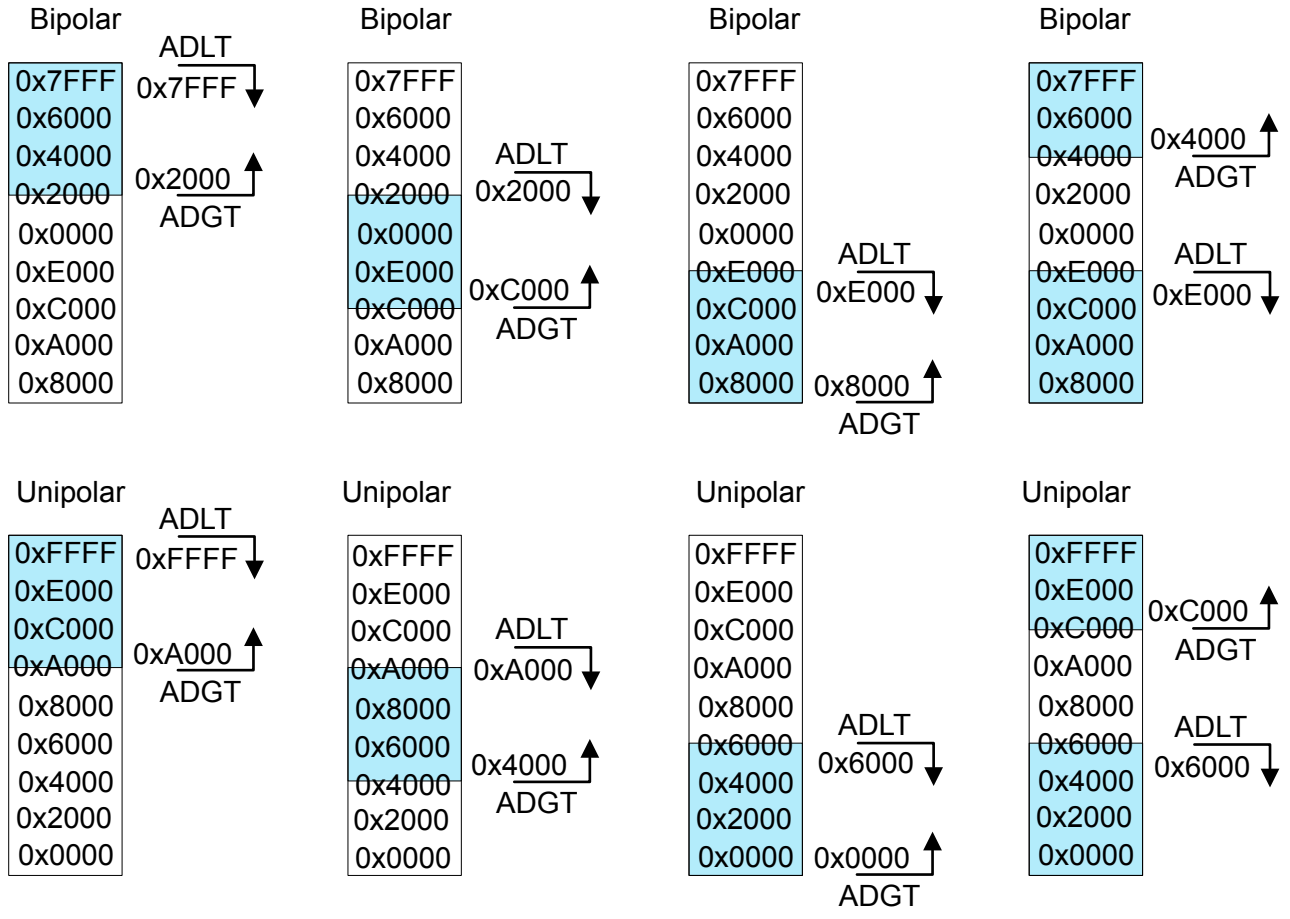


Figure 5.3. Window Comparison Examples

6. Software Examples for EFR32 Series 2

Software examples for the IADC have been relocated to Silicon Labs MCU Peripheral Example repository on GitHub - https://github.com/SiliconLabs/peripheral_examples

6.1 Single Conversion, Single-ended Input

In this example, `iadc_single_em2`, the IADC is configured as follows:

- 12-bit resolution
- Unbuffered 3.3 V AVDD reference
- KEEPWARM mode
- Input: PC04
- Output format: automatic 2's complement (unipolar for this example)
- Trigger: IMMEDIATE
- Trigger action: CONTINUOUS

This example configures the IADC to peform a single (one input) conversion on a single-ended input. The first conversion is triggered by software (SINGLETRIGSEL = IMMEDIATE), and subsequent conversions are automatically triggered immediately afterwards by hardware (SINGLETRIGGERACTION = CONTINUOUS). Once the IADC completes a conversion, an interrupt occurs, waking the device from EM2 to service the interrupt. After the interrupt is serviced, the device returns to EM2 and remains in this energy mode until the next conversion is complete.

To test, first build the project and download to a Starter Kit. Use a power supply to apply 0 - 3.3 V to PC04. Using the IDE debug window, observe the following variables:

- `sample` variable, the 12-bit IADC output, which ranges from 0x000 to 0xFFF (0 to 4095)
- `singleResult` variable, the IADC output converted to volts, which ranges from 0 to 3.3 V

6.2 Single Conversion, Single-ended Input, Window Compare

In this example, `iadc_single_window_compare`, the IADC is configured as follows:

- 12-bit resolution
- Unbuffered 3.3 V AVDD reference
- KEEPWARM mode
- Input: PC04
- Output format: automatic 2's complement (unipolar for this example)
- Trigger: IMMEDIATE
- Trigger action: CONTINUOUS
- Window compare: 0x400 - 0xC00

This example configures the IADC to peform a single (one input) conversion on a single-ended input. The first conversion is triggered by software (SINGLETRIGSEL = IMMEDIATE), and subsequent conversions are automatically triggered immediately afterwards by hardware (SINGLETRIGGERACTION = CONTINUOUS). If the IADC output is between 0x400 and 0xC00, i.e., if the input voltage is between 0.825 and 2.475 V, an interrupt will occur and a GPIO will be toggled.

Note: The two threshold bitfields (ADLT and ADGT) in the IADC_CMPTHR register are 16-bits each. The comparator uses the 12 MSBs and ignores the 4 LSBs.

To test, first build the project and download to a Starter Kit. Use a power supply to apply 0 - 3.3 V to PC04. Observe a toggle on LED0 of the WSTK when the IADC output is between 0x400 and 0xC00.

6.3 Single Conversion, Differential Input

In this example, `iadc_single_diff_polled`, the IADC is configured as follows:

- 12-bit resolution
- Unbuffered 3.3 V AVDD reference
- KEEPWARM mode
- Input: PC04 (positive), PC05 (negative)
- Output format: automatic 2's complement (bipolar for this example)
- Trigger: IMMEDIATE
- Trigger action: ONCE

This example configures the IADC to peform a single (one input) conversion on a differential input. The first conversion is triggered by software (SINGLETRIGSEL = IMMEDIATE), and subsequent conversions are also triggered by firmware (SINGLETRIGGERACTION = ONCE). Firmware will poll the IADC until the conversion is complete, then firmware will trigger the next conversion.

To test, first build the project and download to a Starter Kit. Use a power supply to apply 0 - 3.3 V to PC04 and 0 - 3.3 V on PC5. Using the IDE debug window, observe the following variables:

- `sample` variable, the 12-bit IADC output, which ranges from 0x800 to 0x7FF (-2048 to +2047)
- `singleResult` variable, the IADC output converted to volts, which ranges from -3.3 to 3.3 V

6.4 Single Conversion, Calibration

In this example, `iadc_single_calibration`, the IADC is configured as follows:

- 12-bit resolution
- Unbuffered 3.3 V AVDD reference
- KEEPWARM mode
- Input: PC04 (positive), PC05 (negative)
- Output format: 2's complement
- Trigger: IMMEDIATE
- Trigger action: ONCE

This example calibrates the IADC gain first by measuring a positive full scale, differential input voltage (average of 1024 samples), calculating the gain calibration value, then storing the value in the IADC calibration register. Then, the IADC offset is calibrated by measuring a 0 V differential input (average of 1024 samples), calculating the offset calibration value, and storing the value in the IADC calibration register.

To test, first build the project and download to a Starter Kit. Use a power supply to apply 3.3 V to PC04 and 0 V to PC05. Press and release Push Button 0 on the WTSK. Then, remove the external voltages from PC04 and PC05, and short PC04 and PC05 together. Press release button Push Button 0. The IADC is now calibrated. Apply 0 - 3.3 V to PC04 and 0 - 3.3 V to PC05. Using the IDE debug window, observe the following variables:

- `sample` variable, the calibrated 12-bit IADC output, which ranges from 0x800 to 0x7FF (-2048 to +2047)
- `singleResult` variable, the IADC output converted to volts, which ranges from -3.3 to 3.3 V

6.5 Single Conversion, Low Current

In this example, `iadc_single_low_current`, the IADC is configured as follows:

- 12-bit resolution
- Unbuffered 3.3 V AVDD reference
- KEEPWARM mode
- Input: PC04
- Output format: automatic 2's complement (unipolar for this example)
- Trigger: TIMER
- Trigger action: ONCE
- CLK_SRC_ADC = 1 MHz for EFR32xG21; 5 MHz for EFR32xG22
- ADC_CLK = 1 MHz

This example configures the IADC to peform single (one input) conversions on a single-ended input. All conversions are triggered by the local IADC timer (SINGLETRIGSEL = TIMER), at a sample rate of 100 samples per second. The LDMA transfers the IADC results to RAM, and stops the IADC single conversions once 1024 (`NUM_SAMPLES`) samples are have been transferred.

To test, first build the project and download to a Starter Kit. Use a power supply to apply 0 - 3.3 V to PC04. Using the Energy Profiler in Simplicity Studio, observe the device's current consumption in EM2 before the device returns to EM0 and turns on the LED.

When the IADC sample rate increases, the device current consumption will increase. Typical current consumption for an EFR32xG21 device is shown in the following figure for sample rates of 10 samples per second to 25,000 samples per second. For sample rates in this range and a 1 MHz HFRCOEM23 frequency, the optimal frequency for both `CLK_SRC_ADC` and `ADC_CLK` is 1 MHz, configured in firmware.

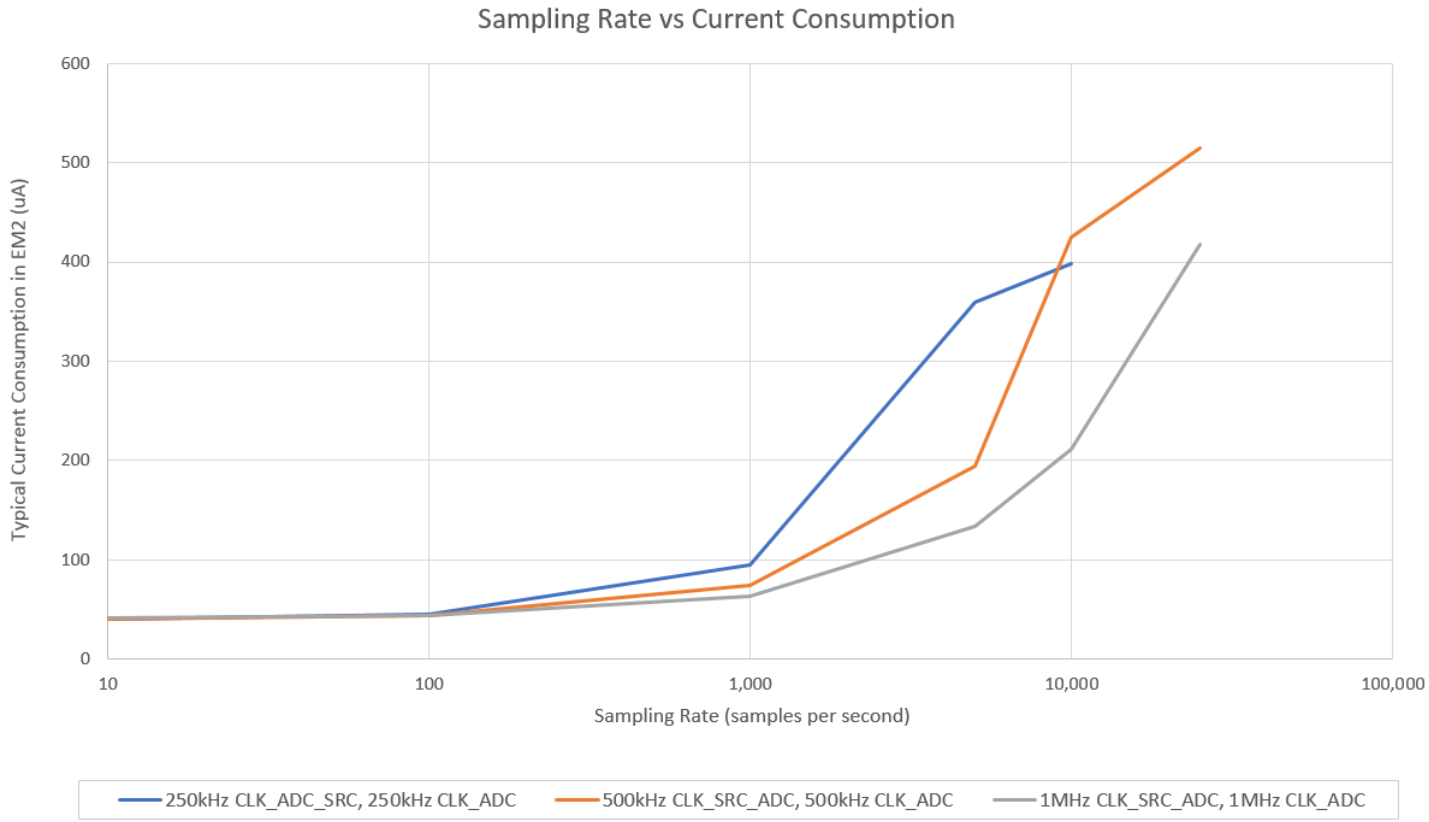


Figure 6.1. Sampling Rate vs Current Consumption, 1 MHz HFRCOEM23

6.6 Single Conversion, PRS

An example demonstrating an IADC single conversion triggered by a GPIO via PRS can be found in application note, AN0012: General Purpose Output software examples.

6.7 Scan Conversion, Interrupt

In this example, `iadc_scan_interrupt`, the IADC is configured as follows:

- 12-bit resolution
- Unbuffered 3.3 V AVDD reference
- KEEPWARM mode
- Input 0: PC04
- Input 1: PC05
- Input 2: AVDD / 4
- Input 3: VDDIO / 4
- Input 4: VSS / 4
- Input 5: DVDD / 4
- Input 6: VDDx / 4
- Input 7: VDDlv
- Output format: automatic 2's complement (unipolar for this example)
- Trigger: IMMEDIATE
- Trigger action: ONCE

This example configures the IADC to peform a scan (multiple input) conversion on two external inputs and six supply/internal voltages. All inputs are single-ended. The first conversion is triggered by software (SINGLETRIGSEL = IMMEDIATE), and subsequent conversions are also triggered by firmware (SINGLETRIGGERACTION = ONCE). Once a scan conversion is complete, an interrupt occurs and firmware triggers the next scan conversion.

To test, first build the project and download to a Starter Kit. Use a power supply to apply 0 - 3.3 to PC04, and 0 - 3.3 V on PC5. Using the IDE debug window, observe the following variables:

- `scanResult[]` variable, array of IADC outputs converted to volts, which range from 0 to 3.3 V

6.8 Scan Conversion, Timer

In this example, `iadc_scan_timer`, the IADC is configured as follows:

- 12-bit resolution
- Unbuffered 3.3 V AVDD reference
- KEEPWARM mode
- Input 0: PC04
- Input 1: PC05
- Output format: automatic 2's complement (unipolar for this example)
- Trigger: TIMER
- Trigger action: ONCE

This example configures the IADC to peform a scan (multiple input) conversion on two external inputs. All inputs are single-ended. All conversions are triggered by the local IADC timer (SCANTRIGSEL = TIMER) every 1 ms. Once a scan conversion is complete, an interrupt occurs.

To test, first build the project and download to a Starter Kit. Use a power supply to apply 0 - 3.3 V to PC04, and 0 - 3.3 V on PC5. Using the IDE debug window, observe the following variables:

- `scanResult[]` variable, array of IADC outputs, converted to volts, which range from 0 to 3.3 V

6.9 Scan Conversion, LDMA

In this example, `iadc_scan_ldma`, the IADC is configured as follows:

- 12-bit resolution
- Unbuffered 3.3 V AVDD reference
- KEEPWARM mode
- Input 0: PC04
- Input 1: PC05
- Output format: automatic 2's complement (unipolar for this example)
- Trigger: IMMEDIATE
- Trigger action: CONTINUOUS

This example configures the IADC to peform a scan (multiple input) conversion on two external inputs. All inputs are single-ended. The first conversion is triggered by software (SCANTRIGSEL = IMMEDIATE), and subsequent conversions are automatically triggered immediately afterwards by hardware (SCANTRIGGERACTION = CONTINUOUS). The LDMA transfers the IADC results to RAM and stops the IADC scan once 1024 (`NUM_SAMPLES`) samples have been transferred.

To test, first build the project and download to a Starter Kit. Use a power supply to apply 0 - 3.3 V to PC04, and 0 - 3.3 V on PC5. Using the IDE debug window, observe the following variables:

- `scanBuffer[]` variable, array of 12-bit IADC outputs, which range from 0x000 to 0xFFF

7. Revision History

Revision 1.2

November, 2023

- Minor verbiage updates and typographical corrections, including capitalization, mis-spellings, and punctuation marks, throughout.

Revision 1.1

October, 2023

- Removed invalid device names from 1. Device Compatibility

Revision 1.0

August, 2023

- Added EFx32xG23, EFR32xG24, EFR32xG25, EFR32xG27 and EFx32xG28 to the supported parts list
- Added information about High Speed and High Accuracy modes throughout the document
- Added a note on GPIO port selection limitations in 3.4.1 External Inputs
- Added information about internal inputs for EFx32xG23, EFR32xG24, EFR32xG25, EFR32xG27 and EFx32xG28 devices in 3.4.2 Internal Inputs and Dedicated Inputs
- Added information about dedicated inputs in 3.4.2 Internal Inputs and Dedicated Inputs
- Added information about clock limitations in 3.5 Clock Selection
- Added information about LESENSE trigger source in 4.4 Scheduling and Triggers and 5.2 LESENSE Interface

Revision 0.3

June, 2020

- Added new section 5.6.2 Output Resolution

Revision 0.2

February, 2020

- Added new OPNs to 1. Device Compatibility.
- Updated IADC format capabilities - 2.2 Overview, 5.6.1 Data Alignment and Channel ID.
- Corrected IADC register names - 3.1 Register Access.
- Corrected IADC reference selection table regarding external reference - 3.3 Reference Selection.
- Added updated IADC clock tree for EFR32xG22 in 3.5 Clock Selection.
- 4. Conversion Process - Noted additional digital accumulate and average feature available in EFR32xG22.
- Added information regarding new status bits for EFR32xG22 - 5.6 Output Data FIFOs.
- 6. Software Examples for EFR32 Series 2 - description updates - Software now available in GitHub repository.
- Minor verbiage updates and typographical corrections, including capitalization, mis-spellings and punctuation marks, throughout document.

Revision 0.1

February, 2019

- 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