EFM32®

... the world's most energy friendly microcontrollers

Digital to Analog Converter

AN0022 - Application Note

Introduction

This application note describes how to use the EFM32 Digital to Analog Converter. The features of the Digital to Analog Converter are described, and the software examples include both a signal generator and audio playback using Direct Memory Access.

This application note includes:

- This PDF document
- Source files (zip)
  - Example C-code
  - Multiple IDE projects

1 Digital to Analog Converter

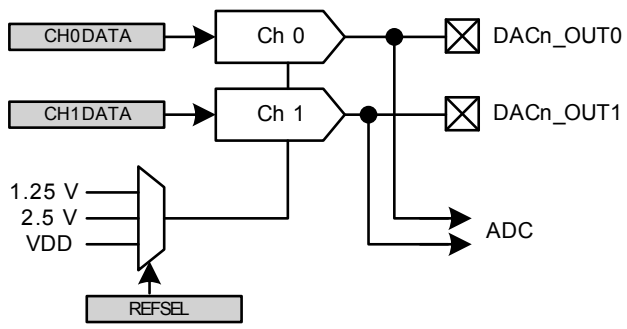
1.1 Introduction

The EFM32 DAC is a 12 bit rail to rail Digital to Analog converter with either two single ended outputs, or one differential output. The maximum conversion rate is 500 ksamples/s at 12 bits resolution. One of the DAC's internal bandgap references or the VDD voltage can be selected as reference. The DAC may be used for a number of different applications such as sensor interfaces, voltage generation or sound/signal output.

This application note shows general operation and usage of the DAC. In addition the sine generator mode is described and the different conversion modes are explained. The software examples include both an advanced signal generator and audio playback that both use DMA (Direct Memory Access) and the PRS (Peripheral Reflex System).

1.2 Overview

Figure 1.1. Digital to Analog Converter Overview



In Figure 1.1 (p. 2), the two data registers and the reference selection register are shown in grey. The DAC has a single reference selection for both channels. The output can be routed to the ADC and output on a pin at the same time.

2 General Operation

2.1 Clock Prescaling

The DAC clock is supplied by a DAC-clock prescaler which divides the peripheral clock (HFPERCLK) by a factor of 2^n between 1 and 128. The resulting DAC_CLK is used by the converter core and the frequency is given by Equation 2.1 (p. 3) :

Clock Prescaling

$$f_{DAC\_CLK} = f_{HFPERCLK} / 2 \text{ ^ } PRESC \quad (2.1)$$

where $f_{HFPERCLK}$ is the HFPERCLK frequency. One conversion takes 2 DAC_CLK cycles and the DAC_CLK should not be set higher than 1 MHz. This is taken care of when using the emlib function for calculating the prescaler.

Normally the PRESCALER runs continuously when either of the channels are enabled. When running with a prescaler setting higher than 0, there will be an unpredictable delay from the time the conversion was triggered to the time the actual conversion takes place. This is because the conversions is controlled by the prescaled clock and the conversion can arrive at any time during a prescaled clock (DAC_CLK) period. However, if the CH0PRESCRST bit in DACn_CTRL is set, the prescaler will be reset every time a conversion is triggered on channel 0. This leads to a predictable latency between channel 0 trigger and conversion.

2.2 Output Selection

The 2 single ended outputs of the DAC can either be used as two separate channels, or combined to form one differential output channel by setting the DIFF bit in DACn_CTRL.

2.2.1 Single Ended Mode

When operating in single ended mode, the channel 0 output is on DACn_OUT0 and the channel 1 output is on DACn_OUT1. The output voltage can be calculated using Equation 2.2 (p. 3)

Single Ended Output Voltage

$$V_{OUT} = V_{DACn\_OUTx} - V_{SS} = V_{ref} \text{ x } CHxDATA/4096 \quad (2.2)$$

where CHxDATA is a 12-bit unsigned integer.

2.2.2 Differential Mode

When operating in differential mode, both DAC outputs are used for the differential voltage. The differential conversion uses DACn_CH0DATA as source. The positive output is on DACn_OUT1 and the negative output is on DACn_OUT0. Since the output can be negative, it is expected that the data is written in 2's complement form with the MSB of the 12-bit value being the sign bit. The output voltage can be calculated using Equation 2.3 (p. 3) :

Differential Output Voltage

$$V_{OUT} = V_{DACn\_OUT1} - V_{DACn\_OUT0} = V_{ref} \text{ x } CH0DATA/2048 \quad (2.3)$$

where CH0DATA is a 12-bit signed integer. The common mode voltage is $V_{ref}/2$.

2.3 Reference Selection

Three internal voltage references are available and are selected by setting the REFSEL bits in DACn_CTRL:

- Internal 2.5V
- Internal 1.25V
- V_{DD}

The reference selection should only be changed while both channels are disabled. The references for the DAC need to be enabled for some time before they can be used. This is called the warm-up period, and starts when one of the channels is enabled. For a bandgap reference, this period is 5 DAC_CLK cycles while the V_{DD} reference needs 1 DAC_CLK cycle. The DAC will time this period automatically (given that the prescaler is set correctly) and delay any conversion triggers received during the warm-up until the references have stabilized.

The bias current of the bandgap reference and the DAC output buffer can be scaled by writing the BIASPROG and HALFBIAS bit fields of the DACn_BIASPROG register. To be sure the DAC has the characteristics stated in the datasheet within the specified frequency range, the default values for the bias currents should be used.

2.4 Conversion Mode

The DAC supports three different conversion modes, continuous, sample-hold and sample-off. By selecting sample-hold or sample-off mode the power consumption can be reduced compared to continuous mode since the DAC is shut off between new samples or output refresh cycles.

2.4.1 Continuous Mode

In continuous mode the DAC channels will drive their outputs continuously with the data in the DACn_CHxDATA registers. This mode will maintain the output voltage and refresh is therefore not needed. Since the continuous mode makes the DAC convert constantly, this mode consumes the most power.

2.4.2 Sample-Hold Mode

In sample-hold mode, the DAC converts data on a triggered conversion and then holds the output at the converted voltage. When not converting between samples, the DAC core is turned off, which reduces the energy consumption. Because of holding element leakage, the output voltage will drift if the output is not refreshed by the DAC. The drift rate is specified in the datasheet.

A refresh of the channel will happen when a new value is written to the DACn_CHxDATA register. Automatic refresh can also be enabled by setting the bit REFREN in DACn_CHxCTRL. The refresh period can be selected by changing the REFRSEL value in the DACn_CTRL register. 4 different refresh periods can be selected, a longer period will be more energy efficient, while a shorter period gives less voltage drift.

Since the refresh period is defined by a number of DAC clock cycles, the refresh interval will be dependent on the DAC clock speed.

2.4.3 Sample-Off Mode

In sample-off mode the DAC and the sample-hold element are turned completely off between samples, tristating the DAC output. This requires the DAC output voltage to be held externally. The references are also turned off between samples, which means that a new warm-up period is needed before each conversion. The DAC will time this period automatically and delay any conversion triggers received during the warm- up until the references have stabilized.

2.5 Conversion Start

The DAC channel must be enabled before it can be used. When the channel is enabled, a conversion can be started by writing to the DACn_CHxDATA register. These data registers are also mapped into

a combined data register, DACn_COMBDATA, where the data values for both channels can be written simultaneously. Writing to this register will start a conversion on all enabled channels.

If the PRSEN bit in DACn_CHxCTRL is set, a DAC conversion on channel x will not be started by data write, but when a positive one HFPERCLK cycle pulse is received on the PRS input selected by PRSSEL in DACn_CHxCTRL.

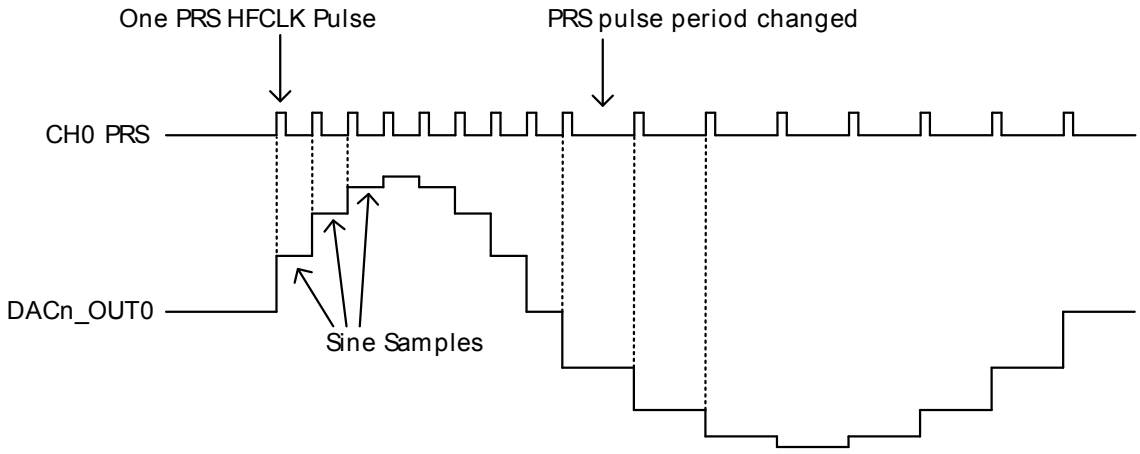
The CH0DV and CH1DV bits in DACn_STATUS are set high when the corresponding channel contains data that has not yet been converted.

3 Advanced Features

3.1 Sine Generator Mode

The DAC contains an automatic sine generator mode, which is enabled by setting the SINEMODE bit in the DACn_CTRL register. In this mode, the DAC data registers are overridden with conversion data taken from a hardware sine lookup table. The lookup table consist of 16 samples. When the OUTENPRS bit in DACn_CTRL is cleared, the sine generator will output the next sine-sample when a PRS conversion trigger pulse is received. This is illustrated in Figure 3.1 (p. 6) .

Figure 3.1. Sine Generator Mode with PRS-triggered Samples



By having a timer supply the PRS pulses on compare or overflow the frequency of the sine wave can be accurately controlled.

If the OUTENPRS bit is set, the PRS channels selected by PRSSEL in DACn_CH0CTRL and DACn_CH1CTRL switches on and off the sine wave (CH0 PRSSEL) and output driver (CH1 PRSSEL). The sine generator will now trigger new conversions by itself. If the channel 0 PRS line is low, a voltage of Vref/2 will be produced. When the line is high, the sine wave will be produced. The PRS line selected for channel 1 controls the output-driver which is tristated when the PRS line is low. This is illustrated in Figure 3.2 (p. 7) .

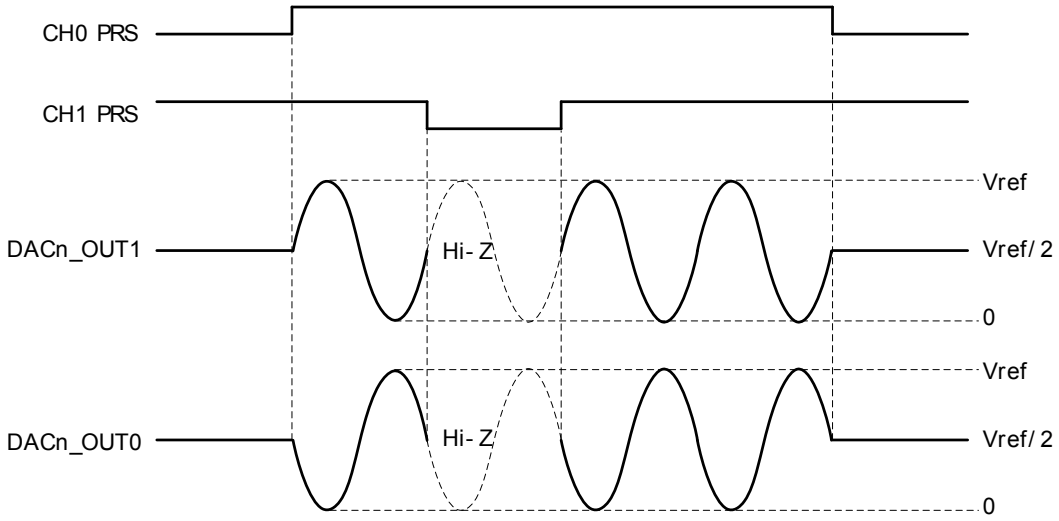
The frequency when the OUTENPRS bit is set is given by Equation 3.1 (p. 6) :

Sine Generation with OUTENPRS bit set

$$f_{sine} = f_{HFPERCLK} / 32 \times (PRESC + 1) \quad (3.1)$$

The sine wave will be output on channel 0. If the DIFF bit is set in DACn_CTRL, the sine wave will be output on both channels (if enabled), but inverted on the second channel (see Figure 3.2 (p. 7)). Note that when OUTENPRS in DACn_CTRL is set, the sine output will be reset to 0 degrees when the PRS line selected by CH0PRSSEL is low.

Figure 3.2. Sine Generation with OUTENPRS bit set



In Figure 3.2 (p. 7), when the OUTENPRS bit in DACn_CH0CTRL is set, the PRS-lines selected by CH0PRSSEL and CH1PRSSEL enables the output driver and starts or stops the sine generator.

3.2 Peripheral Reflex System and Interrupts

Both DAC channels have separate interrupt flags (in DACn_IF) indicating that a conversion has finished on the channel and that new data can be written to the data registers. Setting one of these flags will result in a DAC interrupt request if the corresponding interrupt enable bit is set in DACn_IEN. All generated interrupt requests from the DAC will activate the same interrupt vector when enabled.

The DAC has two PRS outputs which will carry a one cycle (HFPERCLK) high pulse when the corresponding channel has finished a conversion.

3.3 Direct Memory Access

In addition to the PRS output, the DAC sends out a DMA request when a conversion on a channel is complete. This request is cleared when the corresponding channel´s data register or the COMBDATA register is written.

3.4 Calibration

The DAC contains a calibration register, DACn_CAL, where calibration values for both offset and gain correction can be written. Offset calibration is done separately for each channel through the CHxOFFSET bitfields. Gain is calibrated in one common register field, GAIN.

The calibration values are linked to the reference and when the reference is changed, the correct calibration values should be written to the calibration register. Gain and offset for the 1V25, 2V5 and VDD references are calibrated during production and the calibration values for these can be found in the Device Information page. The correct calibration values are loaded automatically when using the emlib for configuring the DAC.

The DAC can also be calibrated manually. Either the internal ADC or an external voltage meter can be used to measure the DAC output voltage. If the ADC is used, it should be calibrated first as described in the AN0021 - ADC application note. The procedure for calibrating the DAC is similar to the ADC calibration procedure.

- Offset is calibrated first by converting the value 0, and changing the calibration register until the output equals 0 V. The algorithm should take into account that the offset might be far below 0, but the output will still be 0.
- Gain should be calibrated after offset by converting the top value: 2^{12}-1 and adjusting the gain calibration register until the output almost equals the reference value (1-2 LSB's below) in order to avoid overshoot.

4 Software Examples

4.1 Programmable Voltage Source

This example configures the DAC in continuous mode and converts a single value continuously. By entering EM1 the DAC operates independently. Since the DAC is operating in continuous mode, the output voltage will be stable even if the output is loaded. The fact that the DAC is continuously working will be reflected by the power consumption.

4.2 Sample-Hold Mode with Automatic Refresh

This example basically does exactly the same thing as the first example, but now with the sample-hold mode which causes the DAC to be disabled between conversions. The internal output stage will hold the output value while the DAC is shut off for a number of cycles. The refresh rate which controls how often the DAC is woken up to refresh the holding element is adjustable.

4.3 Sine Generator Mode with Timer

This example demonstrates how to use a timer to send PRS-pulses which in turn triggers new DAC conversions. The DAC is configured in Sine Generator Mode. A sine wave with variable frequency is available on the DAC output pins. The frequency is adjusted by changing the timer overflow value. After configuration, energymode 1 is entered to save power.

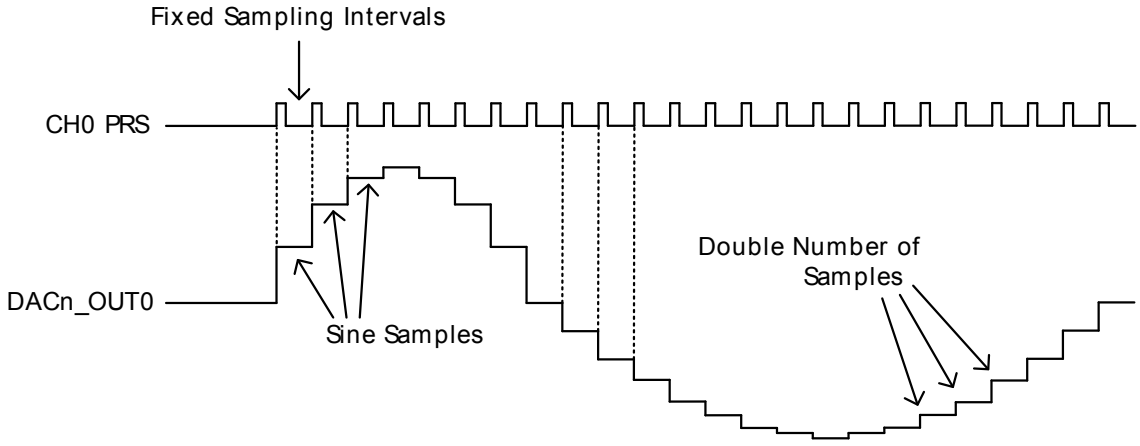
4.4 Signal Generator

The Signal Generator shows how TIMER0, PRS, DMA and the DAC can be used in cooperation to generate different glitch-free waveforms from tables in flash memory. The example code only works on the DK because it uses the joystick and the buttons to change the signal parameters. It can be easily modified to work on other platforms than the DK, but without the user interface.

The DMA is configured in ping pong mode to be able to supply new data to the DAC continuously without software intervention between memory-buffer changes. New DAC samples are triggered by a PRS-pulse from the Timer. The DAC sends DMA-requests immediately after a conversion to have a new data value ready for the next PRS-pulse.

The signal generator can operate in both fixed and variable sampling frequency mode. In the fixed sampling frequency mode, the amount of samples used for each period of the signal is adjusted to match the desired signal frequency. In variable sampling frequency mode the time between PRS-pulses is adjusted and each signal period is made up of a fixed number of samples. Because of the limited amount of look up table samples available for each cycle, the frequency in fixed sampling frequency mode can not be adjusted as accurately as in the variable sampling frequency case. The fixed sampling frequency case is illustrated in Figure 4.1 (p. 10), while an example of variable sampling frequency can be seen in Figure 3.1 (p. 6) .

Figure 4.1. Fixed Sampling Frequency



The buttons and joystick on the DK are used to configure offset, amplitude, frequency, signal type and to switch between fixed/variable sampling frequency. After pushing the AEM button on the DK to enable the switches and joystick for the application, the user can select different waveforms by pushing up or down on the joystick. Frequency is adjusted by pushing left or right, and amplitude/offset is adjusted with the buttons. By pressing the center button of the joystick, the signal generator switches between fixed and variable sampling frequency.

The code flow and operation is outlined below:

- The main function enables and configures the DAC, Timer, PRS and DMA. It then fills the two ping pong memory buffers with the first waveform data. Then it enters a while loop that just enters EM1 after each interrupt. The actual handling of the memory buffers are done in the DMA-finished interrupt routine.

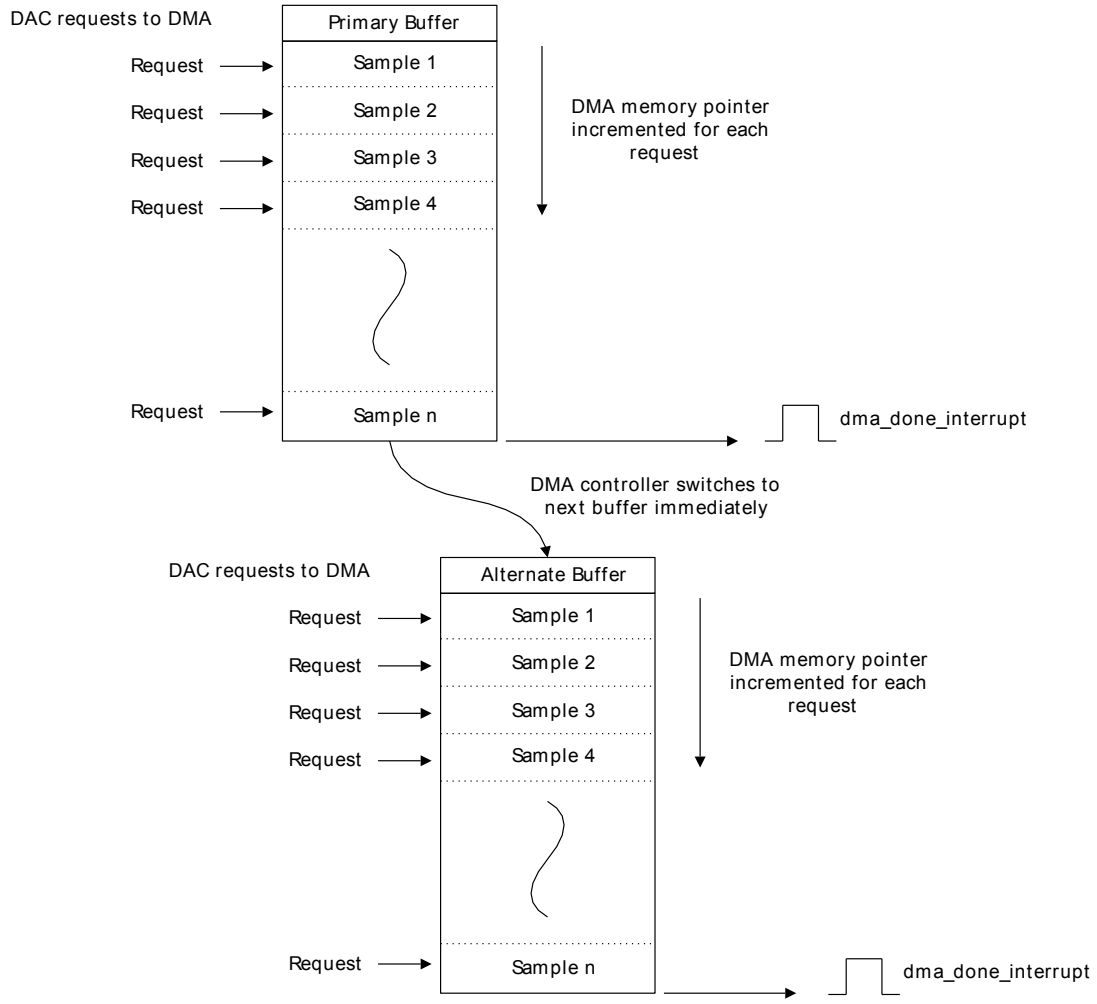
- A new sample is converted by the DAC on each new PRS-pulse from the Timer. The DAC then requests new data from the DMA, which immediately after a conversion fills the DAC data register with a new sample from the currently active buffer.

- When the DMA has iterated through the currently active buffer it immediately switches to the alternate buffer which is ready with new samples.

- The switch to the new buffer also triggers an interrupt. The interrupt routine reads user input from the buttons and joystick and changes the signal parameters if new input is received. It then iterates through and fills the buffer that was just finished with new sample data. The sample data are calculated based on the correct waveform table and the current frequency/amplitude/offset settings. The last task done by the interrupt is to reactivate the newly filled buffer for the DMA.

The ping pong operation of the DMA is illustrated in Figure 4.2 (p. 11) .

Figure 4.2. Ping Pong Operation of DMA



4.5 Audio Playback

This example requires the DK with a FAT formatted microSD card that supports SPI communication. By default it looks for a file named "soundfile.wav" and plays this file at the sampling frequency specified in the wav-file. Both stereo and mono files are supported, but only 16 bit sample size. The example uses the same ping pong DMA transfer scheme as the signal generator, but the sample data are fetched from the microSD-card instead of tables in flash. The code flow is very similar, only adding microSD-card access and excluding user input.

The audio out line on the DK consists of a line driver amplifier with a filter which works best at or above 44.1kHz sampling frequency because of a designed cut-off frequency around 20kHz. There is no volume control therefore you should not connect headphones directly to the output. Computer speakers with volume control are a better choice.

4.6 DAC Calibration

If the EFM32 is operated at high/low temperatures it could be necessary to recalibrate the DAC to obtain the specified performance. The calibration values that are stored in the Device Information Page are valid for room temperature, but a significant change in temperature could decrease the DAC accuracy. A calibration routine example that recalibrates the DAC using the internal ADC is supplied with this application note.

DAC calibration is performed in two steps, first the offset is calibrated by adjusting the DAC calibration register until a digital value of 0 gives 0 V on the output. The second step is gain calibration, now the calibration register is adjusted so that the maximum digital value gives an output voltage that is close or equal to the reference voltage selected. The calibration routine uses the ADC to calibrate the DAC. The internal connection between the DAC and the ADC is used, so no external connection between the ADC and DAC is necessary.

Since the ADC is used to calibrate the DAC, it is important that the ADC is calibrated before it is used to calibrate the DAC. This calibration is included in the example, but it requires an external reference voltage to be applied to ADC input channel 4 (all channels can be used).

5 Revision History

5.1 Revision 1.10

2013-09-03

New cover layout

5.2 Revision 1.09

2013-05-08

Added software projects for ARM-GCC and Atollic TrueStudio.

Adapted signal generator software project to different user input on gecko vs. giant gecko development kits.

5.3 Revision 1.08

2012-11-12

Adapted software projects to new kit-driver and bsp structure.

5.4 Revision 1.07

2012-08-14

Added projects for Tiny and Giant Gecko STKs.

Updated file paths for fatfs

5.5 Revision 1.06

2012-04-20

Adapted software projects to new peripheral library naming and CMSIS_V3.

5.6 Revision 1.05

2011-11-17

Updated IDE project paths with new kits directory.

5.7 Revision 1.04

2011-05-18

Updated projects to align with new bsp version.

5.8 Revision 1.03

2011-05-16

Code update, low pass filter removed from DAC initialization structure.

5.9 Revision 1.02

2011-04-07

Changed error in DMA_ActivatePingPong() function calling

5.10 Revision 1.01

2010-11-16

Updated chip init function to newest efm32lib version.

Updated register defines in code to match newest efm32lib release.

5.11 Revision 1.00

2010-11-08

Initial revision.

A Disclaimer and Trademarks

A.1 Disclaimer

Silicon Laboratories intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Laboratories products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Laboratories reserves the right to make changes without further notice and limitation to product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Silicon Laboratories shall have no liability for the consequences of use of the information supplied herein. This document does not imply or express copyright licenses granted hereunder to design or fabricate any integrated circuits. The products must not be used within any Life Support System without the specific written consent of Silicon Laboratories. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Silicon Laboratories products are generally not intended for military applications. Silicon Laboratories products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons.

A.2 Trademark Information

Silicon Laboratories Inc., Silicon Laboratories, the Silicon Labs logo, Energy Micro, EFM, EFM32, EFR, logo and combinations thereof, and others are the registered trademarks or trademarks of Silicon Laboratories Inc. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. All other products or brand names mentioned herein are trademarks of their respective holders.

B Contact Information

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

Please visit the Silicon Labs Technical Support web page:
http://www.silabs.com/support/pages/contacttechnicalsupport.aspx
and register to submit a technical support request.

Table of Contents

List of Figures

List of Equations

silabs.com


ZERO
ARM Cortex-M0+
TINY
ARM Cortex-M3
GECKO
ARM Cortex-M3
LEOPARD
ARM Cortex-M3
GIANT
ARM Cortex-M3
WONDER
ARM Cortex-M4