ANALOG DEVICES

AN-1549
APPLICATION NOTE

One Technology Way • P.O. Box 9106 • Norwood, MA 02062-9106, U.S.A. • Tel: 781.329.4700 • Fax: 781.461.3113 • www.analog.com

Performance Optimization of the Integrated ADC on the ADuCM4050

INTRODUCTION

Many applications, such as vital signal monitoring, temperature sensing, and imaging, require high resolution analog-to-digital converters (ADCs) with high dynamic range, which are costly and power consuming.

This application note discusses oversampling and averaging techniques and the effect of sampling clock and voltage reference selection in optimizing the performance of the 12-bit ADC in the ADuCM4050 system. This application note also discusses the programming flow and an example application that implements the oversampling and averaging technique on the samples obtained from the ADC in the ADuCM4050.

ABOUT THE ADUCM4050

The ADuCM4050 microcontroller unit (MCU) is an ultra low power, integrated mixed-signal microcontroller system for processing, control, and connectivity. The MCU system is based on the Arm® Cortex-M4F processor. The MCU also has a collection of digital peripherals, embedded static random access memory (SRAM) and flash memory, and an analog subsystem that provides clocking, reset, and power management capabilities in addition to an ADC subsystem.

The ADuCM4050 MCU provides a collection of power modes and features such as dynamic and software controlled clock gating and power gating to support extremely low dynamic and hibernate power.

Full specifications for the ADuCM4050 are available in the ADuCM4050 data sheet.

ADC OPTIMIZATION

The ADuCM4050 microcontrollers incorporate a fast, multi-channel 12-bit ADC capable of operating up to a maximum of 1.8 MSPS. The ADC controller can be set up to perform a series of conversions and transfer data to the system using a dedicated direct memory access (DMA) channel. This setting allows the processor to be in Flexi™ mode (minimizing the overall device power consumption) and to perform other tasks.

The performance of the ADC can be improved in terms of over-sampling and averaging, selection of the voltage reference, selection of the sampling clock, and power consumption.

TABLE OF CONTENTS

REVISION HISTORY

7/2018—Revision 0: Initial Version

OVERSAMPLING AND AVERAGING

The oversampling technique is one solution for effectively using the integrated 12-bit ADC in applications.

The ADC subsystem in the ADuCM4050 microcontrollers has an option to implement the oversampling and averaging technique in a hardware controller without the need to have dedicated software running in the MCU. The ADC subsystem therefore reduces the processor overhead and the energy spent in computing the algorithm. The Oversampling Concept section explores the performance improvement of the ADC using the oversampling and averaging techniques.

OVERSAMPLING CONCEPT

Oversampling is a process of sampling the input signal with a frequency higher than the Nyquist frequency, $f_S/2$, where f_S is the sampling frequency.

Sample the input signal at $M \times f_S$ and then average M samples to obtain one sample. The factor M is referred to as the oversampling factor.

Consider an ideal n-bit ADC that has the quantization noise distributed uniformly within the Nyquist band dc to $f_S/2$, as shown in Part A of Figure 1. Sampling an input signal with this ADC at a much higher sampling rate (Mf_S) distributes the noise over a wide bandwidth from dc to $Mf_S/2$, as shown in Part B of Figure 1. Averaging every M sample acts as a digital filter and removes much of the wideband noise without affecting the input signal bandwidth. This process results in an increase in the signal-to-noise ratio (SNR) in decibels, as shown in the following equation:

$$SNR = 6.02N + 1.76 + 10\log_{10}M \quad (1)$$

where N is the effective number of bits (ENOB) of an ideal ADC.

Thus, a high resolution analog-to-digital conversion can be accomplished with a low resolution ADC.

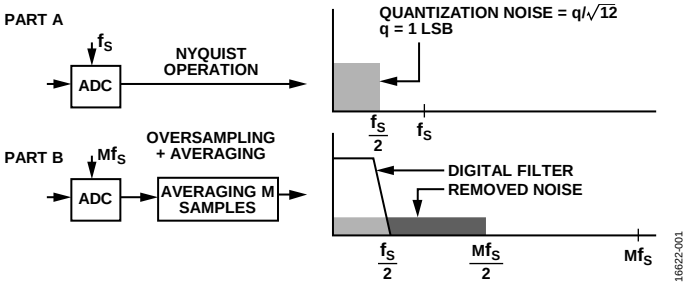

Figure 1. Effect of Oversampling and Averaging in Frequency Domain

This technique filters out the noise outside the signal bandwidth, which increases the dynamic range by $10\log_{10}M$, and increases the ENOB by $\sqrt{\sqrt{M}}$, for an oversampling factor of M.

Generally, for every doubling of the sampling frequency, the SNR increases by 3 dB and the ENOB improves by 0.5 bits.

For example, in an ideal case, averaging the four 12-bit ADC samples increases the ENOB by one bit and improves the dynamic range by 6 dB.

Generally, an ADC has quantization noise, thermal noise, distortions due to the nonlinearity of the ADC, and some external noise due to various factors such as clock and reference. Due to these noises, the actual performance measures are not as exact as the theoretical values. Consider a 12-bit ADC that has 10.5 noise free bits at 50 kSPS. Averaging two adjacent samples of an input signal reduces the effective sampling frequency to 25 kSPS and increases the noise free bits to 11 bits. Further averaging the four adjacent samples reduces the effective sampling rate to 12.5 kSPS and increases the noise free bits to 12.5 bits.

Averaging adjacent samples reduces only the uncorrelated noise present in the signal, which reduces the noise floor of the ADC and improves the SNR. However, the integral nonlinearity remains unchanged. This technique can therefore be used to effectively increase the dynamic range of the ADC at the expense of the overall output sampling rate and extra digital hardware.

If the noise seen by the ADC is only the quantization noise, averaging does not improve the performance. In this case, introducing a small amount of white noise, known as dither, on the input helps achieve a higher resolution by means of averaging. The introduced noise must be uncorrelated to the input signal and must be of sufficient amplitude to randomly toggle the converted output. The histogram of the converted samples for such dithered input follows a Gaussian distribution. Averaging on these samples acts as a low-pass filter, which filters the noise and increases the SNR.

Even if sufficient noise is introduced on the input and averaging is performed on the oversampled values, the noise free bits may not be as exact as the theoretical values. This limitation is due to the total harmonic distortion (THD) of the converter.

SELECTION OF THE REFERENCE VOLTAGE

Proper selection of the reference voltage is an important factor in the ADC performance.

The ADuCM4050 microcontroller integrates a reference buffer that can generate either 2.50 V or 1.25 V as the reference using the integrated band gap reference. The battery voltage (V_{BAT}) or external voltage can also be selected as the reference. These settings are mutually exclusive. Selecting both the internal buffer and V_{BAT} as the reference source disables the internal reference buffer and selects V_{BAT} as the reference voltage.

INTERNAL REFERENCE VOLTAGE

Depending on the battery range indicated by the power supply monitor status register (PMG_PSM_STAT, as shown in Figure 2), either 2.5 V or 1.25 V can be selected as the reference in the ADC configuration register (Register ADC_CFG, Bit VREFSEL, as shown in Figure 3).

If the battery voltage is above 2.75 V, select 2.5 V or 1.25 V as the internal ADC reference. If the battery voltage is less than 2.75 V, select 1.25 V as the reference.

FAST DISCHARGE OF THE INTERNAL REFERENCE BUFFER

The ADuCM4050 MCU allows fast switching from a higher to a lower reference voltage. Perform this switching by enabling the fast discharge bit in the ADC configuration register (set Register ADC_CFG, Bit FAST_DISCH to 1).

The fast discharge can be enabled when switching

- From 2.5 V to 1.25 V.
- From V_{BAT} to 1.25 V.
- From V_{BAT} to 2.5 V (if V_{BAT} > 2.5 V).

For example, when switching from 2.5 V to 1.25 V at room temperature, the switchover time reduces by 2% (from 45 ms to 0.9 ms).

EXTERNAL REFERENCE VOLTAGE

To select an external reference voltage, disable the internal reference buffer by setting the internal reference enable bit in the ADC configuration register (Register ADC_CFG, Bit REFBUFEN) to 0. The external reference must not be more than the battery voltage.

The external reference source must be connected to the VREFP_ADC pin. While using the internal buffer, the VREFP_ADC pin must be floating. The external reference source must be noise free to improve the performance of the ADC.

V_{BAT} AS REFERENCE VOLTAGE

The ADuCM4050 MCU has a special feature that switches the internal reference to V_{BAT}. This feature allows the ADC to use the V_{BAT} supply as the voltage reference and avoids the need to connect an additional external source to the VREFP_ADC pin or an on-board wire connection from the supply pin to the VREFP_ADC pin.

To enable VREFP_ADC as V_{BAT}, set the Bit VREFVBAT in Register ADC_CFG to 1. Wait for a minimum of 700 µs and then set the Bit VREFVBAT_DEL to 1. The delay of at least 700 µs is essential for proper switchover of internal reference to V_{BAT}.

PCB RECOMMENDATION

It is recommended to decouple the VREFP_ADC pin with a 4.7 µF capacitor in parallel with a 0.1 µF capacitor close to the GND_VREFADC pin, which in turn must be tied to GND_DIG, GND_ANA, and the exposed pad on the ADuCM4050. To achieve the best performance from these decoupling components, the capacitors must be placed as close as possible to the device, ideally right up against the device.

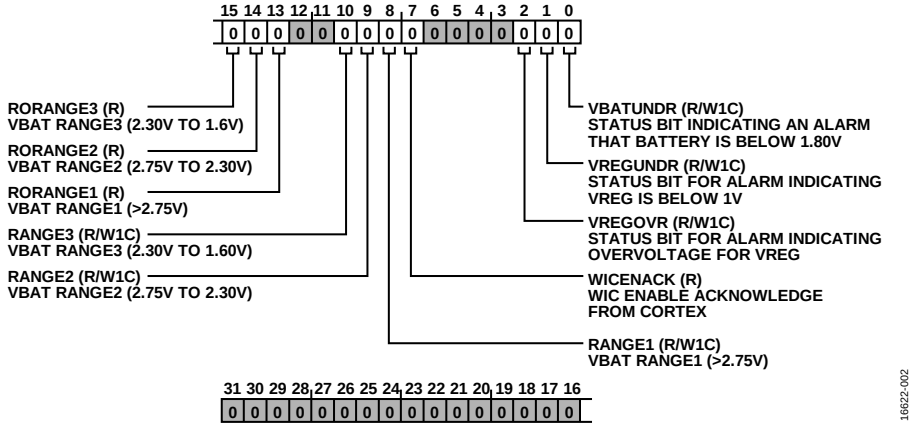

Figure 2. PMG_PSM_STAT Register

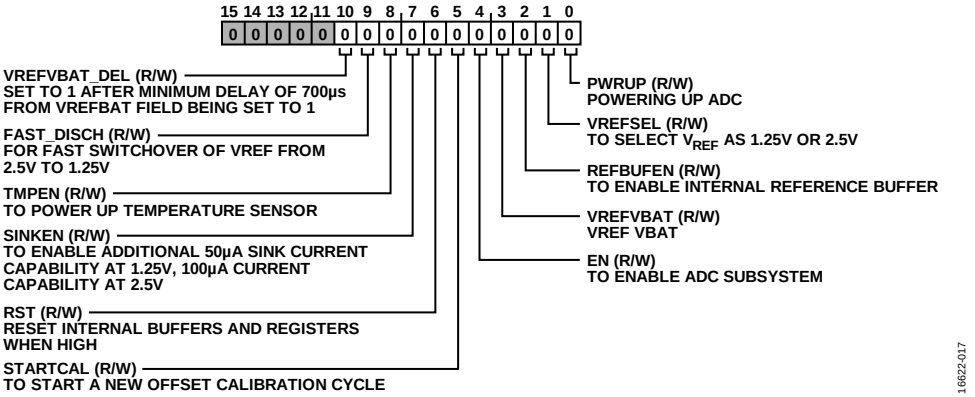

Figure 3. ADC_CFG Register

SELECTION OF THE SAMPLING CLOCK

The quality of the sampling clock in terms of jitter determines the ADC performance. Optimal selection of the sampling clock is critical based on the choice of application. Jitter is the variation of the placement of the clock edges, which results in a sampling time error. This uncertainty in the sampling edges leads to non-uniformly spaced samples and degrades the converter noise performance.

Estimate the clock jitter by observing the degradation in the noise performance of the converter. Determine the SNR limitations due to jitter by using the following equation:

$$SNR = -20\log(2\pi f_{IN} t_{JITTER(RMS)}) \text{ dB} \quad (2)$$

where:
f_{IN} is the input frequency.
$t_{JITTER(RMS)}$ is the rms jitter of the sampling clock.

For an ideal 12-bit ADC with a sampling rate of 154 kSPS, the clock jitter requirement is approximately 412 ps for an input frequency of 70 kHz, whereas if the input frequency is 300 Hz, the jitter requirement is approximately 105 ns.

A traditional way of observing clock jitter is by looking at clock jitter spectrally, as shown in Figure 4. In Figure 4, because of the jitter present in the sampling clock, the ideal impulse response is spread out, resulting in spectral leakage. Much of the energy is distributed close to the desired frequency (known as close in noise), and much of the energy is also contained in the wide bandwidth broadband (known as wideband noise). The close in phase noise smears the fundamental signal into a number of frequency bins, which reduces the overall spectral resolution.

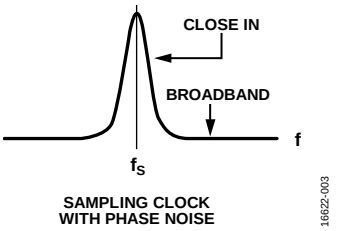

Figure 4. Sampling Clock with Jitter in Frequency Domain

The sampling process is a multiplication of the sampling clock and the analog input signal. This multiplication in the time domain is equivalent to convolution in the frequency domain. Therefore, when the spectrum of the sampling clock is convolved with a pure sinewave input signal spectrum, the resultant spectrum also contains the spectral leakage of the sampling clock spectrum. Due to this leakage, the noise performance of the converter is degraded.

Figure 5 and Figure 6 show the effects of clock jitter on the spectrum of ADC samples. The ADC is enabled in oversampling mode to obtain 16-bit samples, and an input signal of 157 Hz is sampled at 604 SPS. When a free running oscillator with a jitter of approximately 4 ns over a period of 1 sec is used to sample the input frequency, the SNR of the sampled signal is degraded, as shown in Figure 5. In the same setup, if a clock source with considerably less jitter (approximately 200 ps) is used to sample the input signal, the degradation seen in the spectrum for the previous case is not observed (as shown in Figure 6).

Refer to the AN-756 Application Note, *Sampled Systems and the Effects of Clock Phase Noise and Jitter*, for more information.

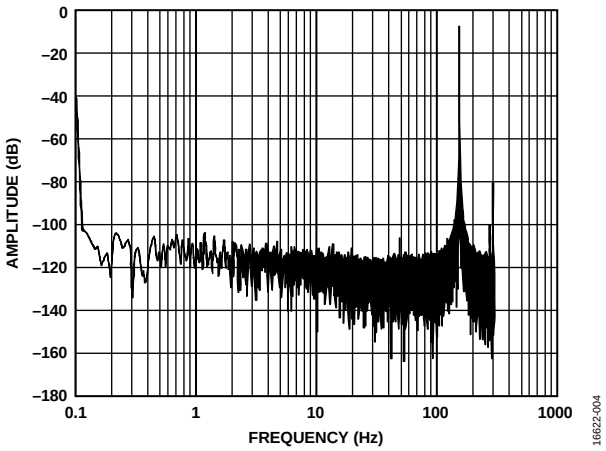

Figure 5. Effect of ADC Clock Jitter on ADC Samples in Frequency Domain, Jitter ≈ 4 ns

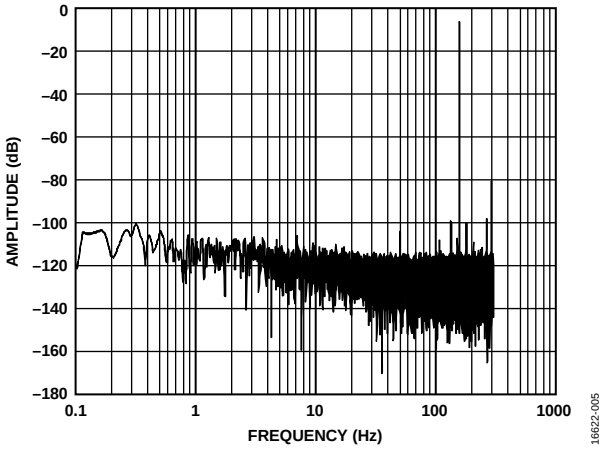

Figure 6. Effect of ADC Clock Jitter on ADC Samples in Frequency Domain, Jitter ≈ 200 ps

SELECTING SAMPLING CLOCK SOURCE

The MCU allows the sampling clock source to be selected as either the internal 26 MHz oscillator or the external 26 MHz crystal oscillator. See the ADuCM4050 Ultra Low Power Arm Cortex-M4F MCU with Integrated Power Management Hardware Reference for details.

POWER OPTIMIZATION

When considering a system that uses an MCU with an integrated ADC and that interfaces with many external sensors, using the low power capability of the ADuCM4050 MCU enhances the performance of the ADC in terms of reducing the overall noise seen by the ADC subsystem. Enabling low power options also optimizes the power consumption of the MCU. The Low Power Mode in the Reference Buffer section, the Flexi Mode section, and the High Power Buck section describe the various features present in the ADuCM4050 microcontroller.

LOW POWER MODE IN THE REFERENCE BUFFER

The internal reference buffer is capable of operating in low power mode when the ADC sampling rate is less than 100 kSPS. In low power mode, the buffer consumes approximately 100 µA less current than in the regular mode of operation. Enable low power mode by setting the RBUFLP bit field of the reference buffer, low power mode register (Register ADC_CFG1) to 1.

Flexi MODE

While ADC is converting multiple samples, enable the DMA to transfer the samples from the ADC output buffer to the memory. During this time, the Arm Cortex-M4F core can be put into Flexi mode or can be used for any other functions, thereby reducing the overall power consumption.

In Flexi mode, the peripherals and DMA are still clocked while the core is sleeping. Therefore, DMA transfers can continue between peripheral and memory as well as from one memory location to another memory location. This low power MCU mode also ensures a noise free environment during ADC conversions.

HIGH POWER BUCK

The high power buck (HP buck) is a capacitive buck converter that reduces the overall current consumption of the device based on the V_{BAT} level. For battery voltages greater than 2.8 V, the current consumption of the device decreases by approximately 50%.

IMPLEMENTATION OF THE OVERSAMPLING AND AVERAGING TECHNIQUE

PROGRAMMING FLOW

The following steps describe how to perform averaging over multiple conversions on a single channel:

1. Set the ADC clock divider in the clock control register (Register CLKG_CLK_CTL1, Bit ACLKDIVCNT) to define the ADC clock, ACLK, as determined by the following equation:

$$ACLK = \frac{Root\ Clock}{ACLKDIV} \quad (3)$$

where *Root Clock* is the root clock frequency (26 MHz) and can be sourced from high frequency oscillator, high frequency crystal, phase-locked loop (PLL), or an external general-purpose input/output (GPIO) clock source. By default, the root clock is HFOSC. Ensure that *ACLK* ≤ PCLK and *ACLK* ≤ HCLK. The maximum configured *ACLK* must be 26 MHz.

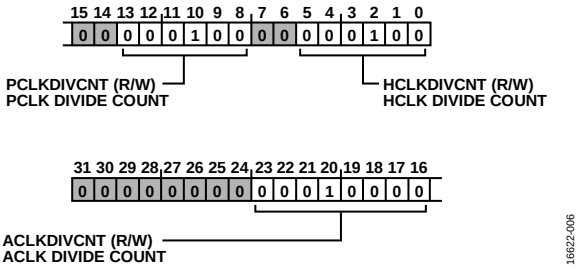

Figure 7. CLKG_CLK_CTL1 Register

The sampling rate of the ADC is internally calculated from ACLK, in which the acquisition phase is Register ADC_CNV_TIME, Bit SAMPTIME + 1 ACLK cycle, and the conversion phase is 13 ACLK cycles taken by successive approximation. The conversion phase is initiated at the end of the acquisition phase. Calculate the sampling rate by using the following equation:

$$ADC\ Sampling\ Rate = \frac{Root\ Clock \div ACLKDIV}{\left((14 + Sampling\ Time) \times Oversampling\ Factor\right)} \quad (4)$$

The oversampling factor determines the number of samples to be oversampled and averaged. The oversampling factor can accept values from 1 to 256.

If the delay time is given in between ADC samples, use Equation 5 to calculate the sampling rate.

Note that the *Delay Time* cannot be a value less than 1.

2. Set the power-up ADC bit in the ADC configuration register (Register ADC_CFG, Bit PWRUP) to 1 to power up the ADC (see Figure 3).
3. Set the wait bit fields of the ADC power-up time register (Register ADC_PWRUP, wait bit fields) as 526 divided by Register CLKG_CLK_CTL1, Bit PCLKDIVCNT.

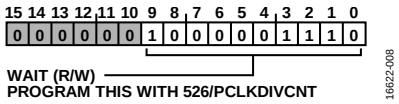

Figure 8. ADC_PWRUP Register

Power-up wait is necessary for proper functioning of the ADC. If Register CLKG_CLK_CTL1, Bit PCLKDIVCNT is 1, the maximum power-up wait time needed for the ADC is 526 clock cycles.

4. Select the internal 1.25 V or 2.5 V reference buffer as the reference voltage using the VREFSEL bit in Register ADC_CFG.
5. Enable the internal reference buffer by setting the REFBUFEN bit in Register ADC_CFG.
6. Enable low power mode for the reference buffer by setting the RBUFLP bit in Register ADC_CFG1.

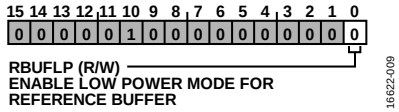

Figure 9. ADC_CFG1 Register Diagram

7. Enable the ADC subsystem by setting the EN bit in Register ADC_CFG.
8. Wait at least 3.5 ms. This wait time is required by the internal reference buffer to reach the set reference buffer voltage level (1.25 V or 2.5 V). One of the general-purpose (GP) timers can be used to wait for 3.5 ms. During this wait period, the device can be put into Flexi mode to save system power and can be woken up by the GP timer interrupt.
9. Check the state of the ADC ready to start converting the bit on the ADC status register (Bit RDY, Register ADC_STAT). If the bit is 1, the ADC is ready to start converting.
10. Write 1 to clear the RDY bit on Register ADC_STAT (see Figure 11).
11. Set the STARTCAL bit on Register ADC_CFG to start the calibration cycle (see Figure 3).
12. Check the CALDONE bit on Register ADC_STAT. If the bit is asserted, write 1 to clear the bit (see Figure 11).
13. Set the SEL bits in the ADC conversion configuration register, Register ADC_CNV_CFG (see Figure 12), to select a channel for conversion. For example, writing 1 to these bits selects Channel 0.

$$ADC\ Sampling\ Rate = \frac{Root\ Clock \div ACLKDIV}{\left((14 + Sampling\ Time) \times Oversampling\ Factor\right) + \left(Delay\ Time + 2\right)} \quad (5)$$

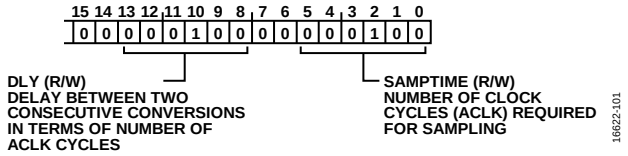

Figure 10. ADC_CNV_TIME Register Diagram

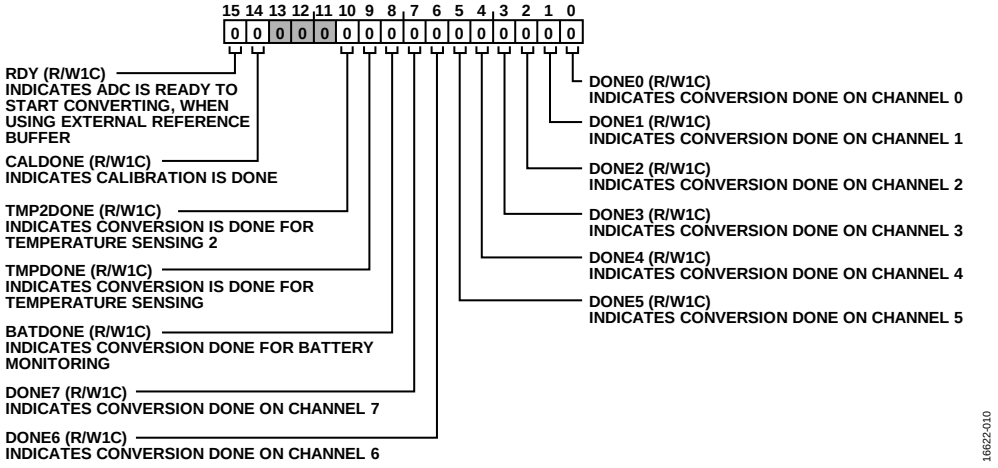

Figure 11. ADC_STAT Register Diagram

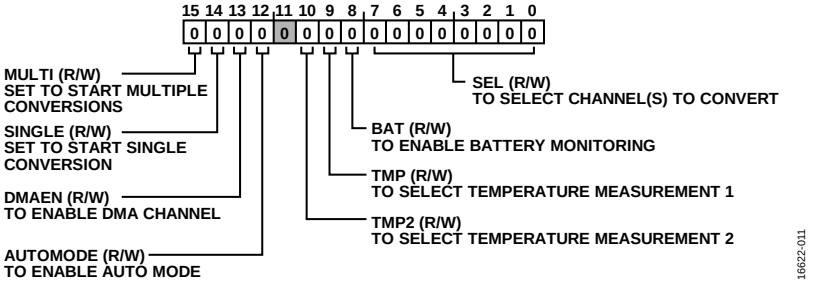

Figure 12. ADC_CVN_CFG Register Diagram

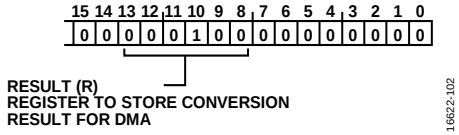

Figure 13. ADC_DMA_OUT Register Diagram

14. Set the CNVDONE bit to 1 in the ADC interrupt enable register (Register ADC_IRQ_EN) to enable the interrupt function when the conversion is complete (see Figure 14).
15. Set the OS bit and the EN bit to 1 in the ADC averaging configuration register (ADC_AVG_CFG) to enable oversampling and averaging (see Figure 15).
16. Set the oversampling and averaging factor (M) in the factor bit fields in the ADC averaging configuration register (ADC_AVG_CFG). Table 1 lists the factor that must be programmed for a particular required resolution.
17. Set the single bit in Register ADC_CNV_CFG to 1 to start a single conversion (see Figure 12).
18. The conversion done interrupt is generated if the CNVDONE bit in the ADC_IRQ_EN register is set. Check the conversion done bit in the ADC_STAT register for the corresponding channel (for example, Register ADC_STAT, Bit DONE0) when the conversion done interrupt is generated.
19. Read the conversion output from the conversion result register of the corresponding channel (for example, ADC_CH0_OUT).
20. Write 1 to clear the conversion done bit in the ADC_STAT register (for example, set Bit DONE0 to 1).
21. To obtain multiple oversampled and averaged samples, repeat Step 1 to Step 15.

22. Set the following DMA configurations:
    - DMA count = 9 for 10 conversions (DMA count = number of conversions − 1).
    - Source address is the address of the ADC_DMA_OUT register (see Figure 13).
    - Source size is the half word.
    - Set the destination address of DMA as the SRAM memory location address to store the conversion result.
    - Program the required increment in the destination address.
23. Set the DMAEN bit in Register ADC_CNV_CFG to 1 in the ADC conversion configuration register and enable the DMA.
24. Set the MULTI bit in Register ADC_CNV_CFG to 1 in the ADC conversion configuration register to start the conversion.
25. When the DMA_DONE interrupt is generated, clear the MULTI bit to disable further conversions in the interrupt service routine (see Figure 16).

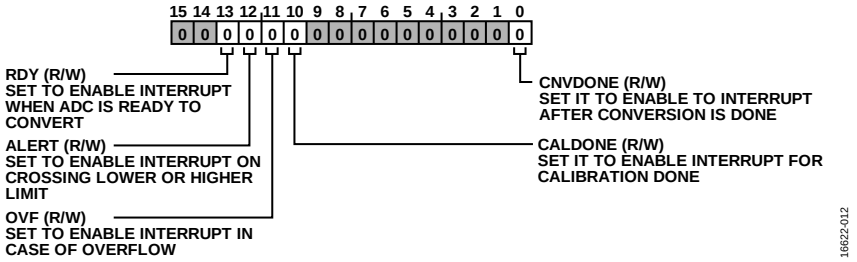

Figure 14. ADC_IRQ_EN Register Diagram

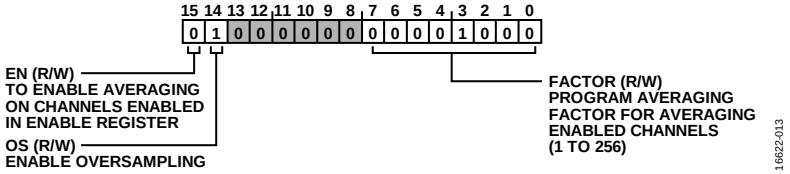

Figure 15. ADC_AVG_CFG Register Diagram

Table 1. Oversampling and Averaging Factor for Enhanced Resolution

Resolution Required	Oversampling and Averaging Factor (M)	Number of Samples Used for Averaging
13-Bit	0x02	4
14-Bit	0x08	16
15-Bit	0x20	64
16-Bit	0x80	256

APPLICATION EXAMPLE OVERVIEW

This section describes an application sequence to perform ADC oversampling and averaging in power optimized mode.

In this example, the ADC is enabled in oversampling and averaging mode. The oversampling factor (M) is set at 256 to obtain 16-bit samples. The DMA is configured to capture and store 1024 16-bit samples in SRAM locations. The samples are later obtained by means of the universal asynchronous receiver/transmitter (UART) interface. During ADC conversion and DMA transfers, the core is in Flexi mode.

Figure 16 shows the application flow for oversampling in multiconversion mode.

Figure 17 shows the application sequence.

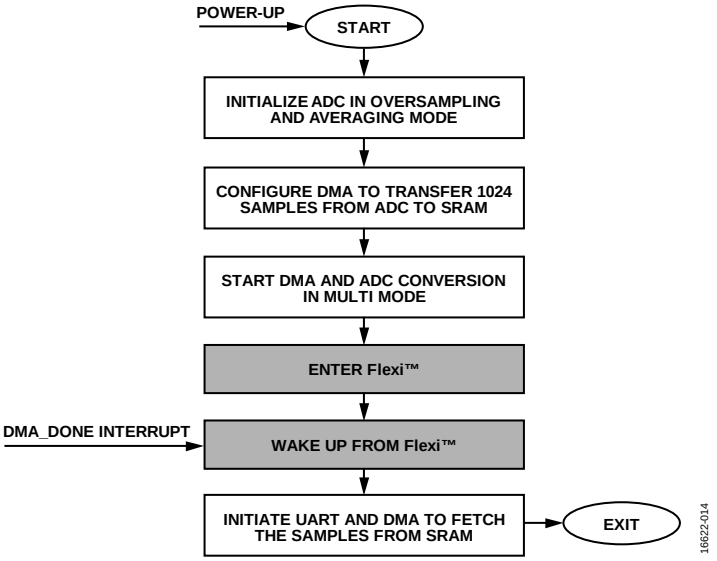

Figure 16. Application Flow for Oversampling in Multiconversion Mode

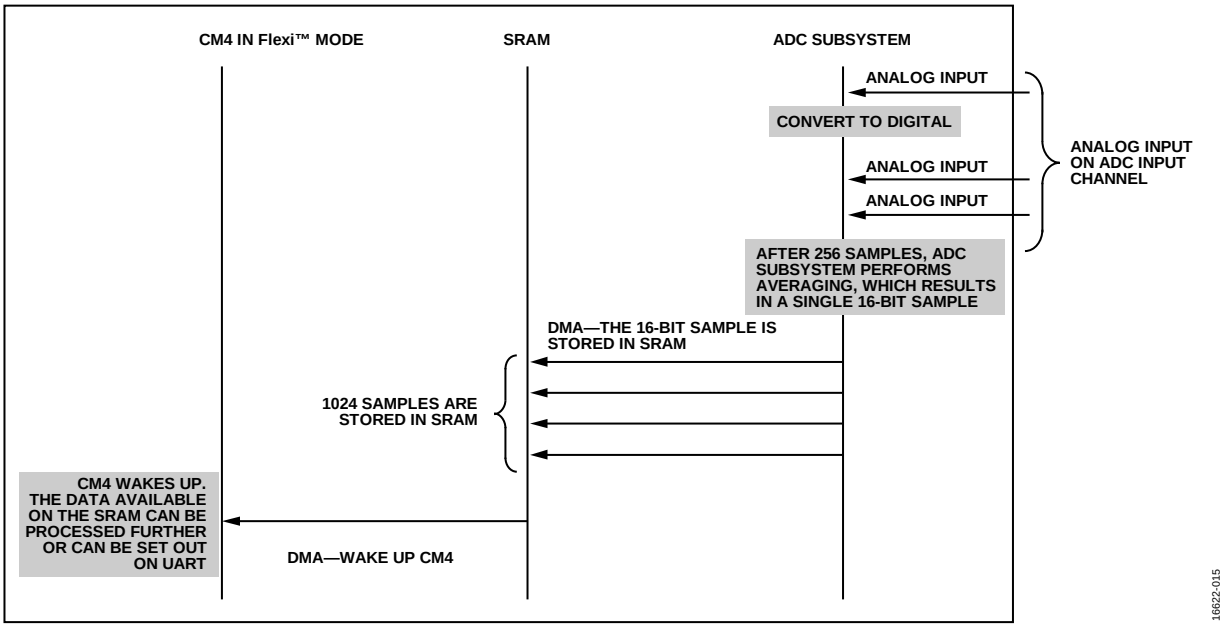

Figure 17. Application Sequence

RESULTS

Figure 18 shows the ADC performance measurement describing ENOB vs. the oversampling and averaging factor. Because the ADC is not ideal, the ENOB is calculated based on the SNR in Equation 1. The SNR is obtained by inputting a sinewave signal with a frequency of 157 Hz from a signal generator to ADC Channel 0.

For each oversampling factor (M), the ADC clock frequency is adjusted such that the throughput is always 604 SPS.

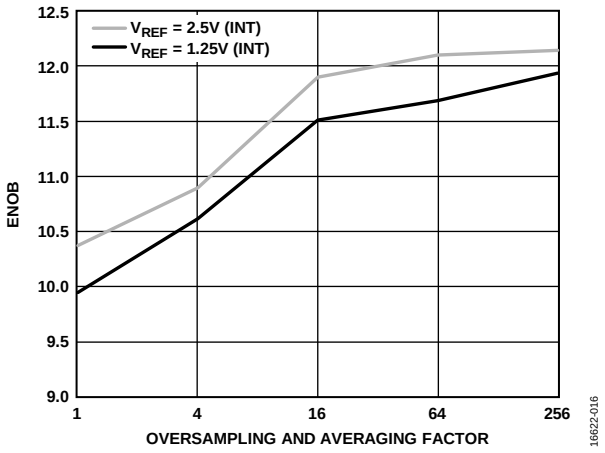

Figure 18. ENOB vs. Oversampling and Averaging Factor

REFERENCES

Brannon, Brad. AN-756 Application Note. Sampled Systems and the Effects of Clock Phase Noise and Jitter. Analog Devices, 2004.

Kester, Walt. "ADC Input Noise: The Good, the Bad, and the Ugly. Is No Noise Good Noise?" Analog Dialogue Vol. 40, February 2006.

Kester, Walt. 2005. The Data Conversion Handbook. Analog Devices.

Reeder, Rob, Wayne Green, and Robert Shillito. "Analog-to-Digital Converter Clock Optimization: A Test Engineering Perspective." Analog Dialogue Vol. 42, February 2008.