FRONTGRADE

APPLICATION NOTE

UT32M0R500

ADC Users Guide UT32M0R500 32-Bit Arm®
Cortex®-M0+ Microcontroller

4/23/2020
Version #: 1.0.1

1.0 Overview

This User Guide supplies users with additional information on the use of the UT32M0R500's ADC peripheral. The guide includes a general description of the ADC module and its features, the different conversion modes, how to set the sequence delay for multi-channel sweeps, how to set the over-sampling rate, information about the stability control register, what each channels' error flag means, how to get a more accurate signal out of the A DC, and ADC related errata.

2.0 Description of ADC-SPB Module and Features

The UTM0R500 Analog-to-Digital Converter Signal Processing Block is a complete and flexible digitizing solution. The system designer can choose to use any combination of single-ended or differential analog inputs, apply a unique gain setting to each enabled channel, and convert all enabled channels in a single sweep or a continuous stream.

16 analog inputs (A IN0-AIN15) are individually configurable to meet an application's signal conditioning requirements. They can be defined as single-ended inputs, or fully-differential signals, or a combination of single-ended and differential. Additionally, an internal temperature measurement channel is available. Because the temperature channel is internal, it can be enabled without restricting the availability of the 16 analog inputs. There are 25 channel-specific configuration registers: 16 for single-ended channels, 8 for differential channels, and 1 for the temperature channel.

A DC.SECHAN_CFG_0, ADC.SECHAN_CFG_1, … ADC.SECHAN_CFG_15.

A DC.DIFFCHAN_CFG_0, ADC.DIFFCHAN_CFG_1, … ADC.DIFFCHAN_CFG_7.

A DC.TEMPCHAN_CFG.

Each input signal is routed to an analog MUX, which selects and isolates the appropriate channel before sending that channel to the Programmable Gain Amplifier (PGA).

The PGA has a programmable gain range of 30dB available for each enabled channel. Each channel can have a unique gain from the following list: 0.5, 1, 2, 4, 8, or 16 V /V. After the PGA, the signal is routed through an antialias filter.

A n analog third-order low-pass filter delivers anti-alias protection against corruption in a noisy digital processing environment. This active filter also buffers the high-speed sampling modulator. The Delta-Sigma Modulator (DSM) over-samples the analog input at 12.5MSPS, nominally. Afterwards, an additional selectable Digital Decimation Filter removes the high-frequency shaped quantization noise and decimates the data into the down-converted 12bit output word.

The design is optimized for 12-bit accuracy at 100kHz output data rate. However, the configuration registers allow for flexibility depending on the requirements of the application.

3.0 Convert Modes and ADC_SEQDLY

Setting the A DC_TRIGGER bit will initialize a conversion of all enabled ADC channels. When converting more than one channel in this way (multiplexing at the analog input) each enabled channel's CFG_REG register should be setup to use the default DDF1 (COI) digital filter. With the DDF1 active, the A DC resets both the DDF and DSM at the beginning of each conversion.

When converting more than one channel, each input requires time to settle before being converted, and each input requires time to store the data read by a given conversion. These timing requirements are met by changing the ADC_SEQDLY[5:0] bits in the A DC_SEQ_CTRL register. When finding the minimum value for ADC_SEQDLY[5:0], both of the below equations must be true:

$$\text{ADC\_SEQDLYmin} \geq \text{ ADC\_SEQDLY min Input Settling Time}$$

$$\text{ADC\_SEQDLYmin} \geq \text{ADC\_SEQDLYmin Conversion Storage Time}$$

The first of these two values, *ADC_SEQDLYmin Iput Settling Time,* accounts for the Input Settling Time, which is the amount of time between the analog mux changing to the next enabled channel and that channel being converted. The following equation calculates the Input Settling Time:

$$\text{Input Settling Time} = \text{ADC\_SEQDLYmin Input Settling Time} * \text{ADC\_CLK Period} * 25$$

Re-written, this equation is:

$$\text{ADC\_SEQDLY}_{\text{min Input Settling Time}} = \frac{\text{Input Settling Time}}{\text{ADC\_CLK period}*25}$$

The maximum Input Settling Time listed in the datasheet is 28µs. To ensure that the A DC_SEQDLY accounts for this settling time, using an A DC_CLK setting of 12.5MHz (or 0.08µs):

$$\text{ADC\_SEQDLY}_{\text{min Input Settling Time}} = \frac{28\mu\text{s}}{0.08\mu\text{s} * 25}$$

$$\text{ADC\_SEQDLY}_{\text{min Input Settling Time}} = 14 \text{ (decimal)}$$

NOTE: The delay does not apply to the first channel in a sequence. Converting all 16 channels using continuous sweep mode alone is not possible. In this case, it may be necessary to use a dedicated single convert cycle to repeat a sweep of the first channel using a software timing delay , or to keep the first channel as a dummy (non-delayed) channel in the continuous sweep. However, using Single-sweep mode generally does produce accurate results on all channels including the first in a sequence under nominal conditions. This is due to application software execution time. During execution, as the program enables the channels, the MUX control is switched to the first enabled channel in a sequence. If the stimulus source is present and the A DC circuit blocks are enabled, then the first channel can be prepared for the TRIGGER when it arrives.

The second of the two values, ADC_SEQDLY*min Conversion Storage Time,* accounts for the amount of time required for the ADC to store the conversion data. This data is transferred from the ADC_CLK domain into the PC LK domain, and requires four PCLK periods minimum to complete in the subsequent conversion period. The equation is more complex, as both CLKs and the OSR value affect the calculation:

$$\text{ADC\_SEQDLYmin Conversion Storage Time} = \frac{\left(4*\left(\frac{\text{ADC\_CLK frequency}}{\text{PCLK frequency}}\right)\text{-OSR}\right)}{25}$$

To continue the above example, assuming a PCLK frequency of 1MHz, and an OSR of 101:

$$\text{ADC\_SEQDLY min Conversion Storage Time} = \frac{\left(4^*\left(\frac{12.5\text{MHz}}{1\text{MHz}}\right)\text{-}101\right)}{25}$$

ADC_SEQDLY min Conversion Storage Time = 0 (decimal) (Negative numbers round up to 0)

With these two calculations complete, the true minimum value of ADC_SEQDLY in this example is:

$$\text{ADC}_{\text{SEQDLY}_{\text{min}}} = 14 \text{ (decimal)}$$

4.0 The Programmable Over-Sampling Rate (OSR)

The Delta-Sigma Modulator oversampling rate, DSM_OSR, is set within the Timing Control register A DC.TIM_CTRL. The OSR is the number of samples the modulator will use to perform a conversion. The UT32M0R500 Functional Manual has a table of optimized values for the OSR depending on the Digital Decimation Filter. The **Stability Control Register** section investigates the ADC Noise Levels against PGA Gain and Modulator OSR values.

The A DC is designed to operate around a nominal conversion rate of 100KSPS. By decreasing the OSR value, the throughput rate can be somewhat faster, but the noise level will rise and resolution will decrease. As example of OSR adjustment, while holding the A DC oscillator divider at the recommended value of 4, then the DSM samples the input at 12.5MHz (50E6 / 4 = 12.5E6). Therefore, setting the OSR value to 127, the output data rate ODR is 98.4 KSPS. This is calculated using the following equation:

$$\text{Output Data Rate} = \frac{\text{OSR}}{\text{ADC\_CLK}}$$

$$\text{Output Data Rate} = \frac{127 \text{ samples}}{12.5\text{MHz}}$$

$$\text{Output Data Rate} = 98.4 \text{ KSPS}$$

The below table lists the Output Data Rate compared to recommended OSR values:

Table 2: Converter Throughput vs. OSR with OSC_DIV = 4

Recommended OSR values	Output data rate [SPS]
63	198.4K
80	156.3K
101	123.8K
127	98.4K
160	78.1K
202	61.9K
255	49.0K

5.0 Stability Control Register

The A DC Stability Control Register allows user to set the overload detection level for the ADC modulator. A higherorder modulator can become saturated and in some conditions unstable. The circuitry includes a hardware reset which can refresh the modulator so that it can recover. In fact, initiating a conversion does just exactly the same, it begins with a reset of the modulator and digital filter.

For design evaluation, and as a fail-safe backup, the stability register set is included. The overload flag OVL_FLAG is set when an overload is detected and the overload reset OVL_RST is applied. An overload is indicated by repeated codes from the modulator into the digital filter, implying the modulator is either stuck or out of range. The overload count OVL_CNT register value is a threshold of repeated codes (either 1's or 0's) which, when reached, sets the overload flag. The default value is thirty. During design evaluation, input dependent overload was forced by over-ranging the input and lowering the overload count to ten, which did assert the overload flag. The converter continued valid operation.

The UT32M0R500 ADC Block has not exhibited incorrect indication of overload, nor tendency toward instability, during characterization testing.

6.0 Conversion Error Flags and Meanings

Each channel has its own A DC->DATA register, which means each channel has its own set of error flags, and the configuration of each channel affects what flags are applicable. Knowing what each error flag means is important for ensuring the A DC is operating as intended. The below paragraphs describe what is actually happening when one or more of these bits are set.

- DATA_ERROR
- COI_OVER
- SINC4_OVER
- DSM_OVL_FLAG
- TRIG_UNDER

The **DATA_ERROR** flag (bit 14) is a combinational OR of all the other error flags, including the reserved error flag located in bit 15. Use the DATA_ERROR flag to check if any of the other error flags are set.

The **DATA_VALUE_STALE** flag (bit 15, now reserved, see errata section) sets when either a conversion has not completed for this channel since a reset state, or a conversion has not completed since the last read of this specific DATA register. Due to errata concerning the A DC_CONV_COMPL bit, a bit the DATA_VALUE_STALE bit depended upon, this bit is reserved.

The **COI_OVER** flag (bit 23) and the **SINC4_OVER** flag (bit 24) perform similarly. One of these two flags, depending on which of the two digital filters the channel in question uses, will set if the data coming out of the Delta-Sigma Modulator (DSM) has:

- Exceeded the maximum input range. The number of 1's sent from the DSM to the filter indicate the input voltage has exceeded the maximum input range.
- Fallen below the minimum input range. The number of 0's sent from the DSM to the filter indicate the input voltage has fallen below the minimum input range.

Note: Users can pick an OSR value outside of the recommended table, but this will change the full-scale data range. The full-scale data range is a value used by the COI_OVER or SINC4_OVER flags to determine if an input value is out of range based upon a channel's gain setting. Using a non-recommended OSR value will set the full scale data range to the maximum value of 0x7FF, regardless of the PGA gain.

The **DSM_OVL_FLAG** flag (bit 25) is a copy of the A DC->STAB_CTRL.DSM_OVL_FLAG. This bit sets when the DeltaSigma Modulator (DSM) output bitstream repeats the same value too many times (determined by STAB_CTRL.DSM_OVL_CNT), indicating the DSM is overloaded. The overloaded DSM will automatically reset without affecting either of the digital filters (C OI3 or SINC4), or the data read from those filters. STAB_CTRL.DSM_OVL_RST determines the number of A DC_CLK cycles taken to reset. This feature has been included as a fail-safe backup. A s the DATA registers' bit copies the STAB_CTRL bit, clearing the ADC->STAB_CTRL version is the only way to clear the DATA bit.

The **TRIG_UNDER** flag (bit 26) is set to indicate a trigger happened mid-conversion. Note that triggers that happen mid-conversion cause a new conversion to start in addition to setting this bit. The start of a new conversion (conversion caused by a new trigger, not a mid-conversion trigger) clears this bit.

7.0 ADC Noise Level and Effect to DC Conversion

Table 3 – Table 7 show the RMS noise, effective resolution, and peak-to-peak resolution of the A DC Signal Processing Block for various operating modes. Relevant details of the procedure to compile this information is included for reference. This information is helpful for systems that have unavoidable and irreducible noise. This data is a measurement example from a typical bench setup. Frontgrade used a very low-noise source.

This data relates the noise levels of DC conversions to the adjustable PGA Gain range and the modulator OSR. The programmable OSR directly controls the output data rate (ODR). Herein priority is given to single-ended input configuration; however, the first data set is differential input to provide a comparison baseline. Differential input is preferred for achieving the lowest noise levels. The dynamic range is increased and noise at the analog input is reduced, but driving the input signal differentially could be impractical or unnecessary in particular applications. The table below and the associated plots can be used for guidance during system design.

The DDF1 (COI filter) is used, as this is appropriate for accurate DC digitization. The SPB_CFG_0 register is set for single-sweep convert mode and the convert command is repeated 1000 times. After each trigger, the DATAOUT[11:0] register is read, and each consecutive 12-bit word is captured into a single record.

Noise in the system corrupts the accuracy of the DC signal measurement when its magnitude approaches or exceeds the distance between quantization levels (LSB). It is evident in these plots, when the noise increases, such as with increased gain, there is an increasing uncertainty in the resultant code. The Effective Resolution (B_{eff}) for the DC measurement is defined:

$$2^{B_{eff}} = \frac{\text{DC FSR}}{\sigma\text{noise}} = \frac{2^{B}}{\sigma\text{codes}}$$

Where B is bits of resolution, FSR is full-scale range, and σ is a normal standard deviation. This relation shows the resolution of the quantization is reduced according to the noise RMS value; and is valid for $\sigma > 1$ LSB. The calculation can be completed by producing a population of samples, determining the standard deviation about the mean, and using:

$$B_{eff} = B - \frac{\log_{10} \sigma\text{codes}}{0.301}$$

After the RMS noise in the system is obtained, perhaps more useful, the Peak-to-peak Resolution (BP-P) is available. This $B_{P\text{-}P}$ is the approximate flicker-free level of quantization to an expected value by taking 6σ of the noise variation.

Finally, in some cases, the resolution of a noisy conversion system can be improved by averaging. First, the noise must be large relative to an LSB size. And, if the noise is "white", with a uniform power spectral density in the measurement bandwidth and approximated by a Gaussian PDF, then the input can be oversampled by X, and averaging will reduce noise by √X (that is, the error in the variance of the sample mean improves by 1/X). The downside to this approach is lower throughput by performing many conversions and the computation time. Care must be taken not to truncate the data during the summation and averaging, and potential improvement is limited by the overall system nonlinearity.

Table 3: ADC Noise vs. Gain for Differential Input; DC Input, OSR = 127, DDF1 (COI3)

Full-scale Input, Differential (V)	PGA Gain (V/V)	ODR (SPS)	Noise, RMS (LSB)	Noise, Input (µVRMS)	Effective Resolution (bit)	Peak-peak Resolution (bit)
3 (±1.5)	1	98.4k	< 1	< 730	12	12
1.5	2	98.4k	< 1	< 370	12	11.4
0.75	4	98.4k	< 1	< 183	12	10.5
0.375	8	98.4k	< 1	< 92	12	10.2
0.1875	16	98.4k	~ 1	48	12	9.5

Table 4: ADC Noise vs. Gain for Single-ended Input; DC Input, OSR = 127, DDF1 (COI3)

Full-scale Input, Single-ended (V)	PGA Gain (V/V)	ODR (SPS)	Noise, RMS (LSB)	Noise, Input (µVRMS)	Effective Resolution (bit)	Peak-peak Resolution (bit)
1.5	1	98.4k	< 1	< 370	12	10.6
750m	2	98.4k	< 1	< 183	12	10.3
375m	4	98.4k	< 1	< 92	12	10.1
187.5m	8	98.4k	~ 1	48	12	9.7
93.75m	16	98.4k	1.4	32	11.5	8.9

Table 5: ADC Noise vs. OSR for PGA Gain = 1; Single-ended DC Input

Full-scale Input, Single-ended (V)	PGA Gain (V/V)	ODR (SPS)	Noise, RMS (LSB)	Noise, Input (µVRMS)	Effective Resolution (bit)	Peak-peak Resolution (bit)
1.5	1	198.4k	0.8 LSB	300	12	9.7
1.5	1	156.3k	< 1 LSB	< 370	12	10.3
1.5	1	123.8k	< 1 LSB	< 370	12	10.3
1.5	1	98.4k	< 1 LSB	< 370	12	10.6
1.5	1	78.1k	< 1 LSB	< 370	12	10.7
1.5	1	61.9k	< 1 LSB	< 370	12	10.7
1.5	1	49.0k	< 1 LSB	< 370	12	11.2

Table 6: ADC Noise vs. OSR for PGA Gain = 4; Single-ended DC Input

Full-scale Input, Single-ended (V)	PGA Gain (V/V)	ODR (SPS)	Noise, RMS (LSB)	Noise, Input (µVRMS)	Effective Resolution (bit)	Peak-peak Resolution (bit)
375m	4	198.4k	0.9 LSB	81	12	9.6
375m	4	156.3k	< 1 LSB	< 92	12	9.8
375m	4	123.8k	< 1 LSB	< 92	12	10.1
375m	4	98.4k	< 1 LSB	< 92	12	10.1
375m	4	78.1k	< 1 LSB	< 92	12	10.3
375m	4	61.9k	< 1 LSB	< 92	12	10.4
375m	4	49.0k	< 1 LSB	< 92	12	10.3

Table 7: ADC Noise vs. OSR for PGA Gain = 16; Single-ended DC Input

Full-scale Input, Single-ended (V)	PGA Gain (V/V)	ODR (SPS)	Noise, RMS (LSB)	Noise, Input (µVRMS)	Effective Resolution (bit)	Peak-peak Resolution (bit)
93.75m	16	198.4k	1.8 LSB	40.5	11.2	8.6
93.75m	16	156.3k	1.7 LSB	37.9	11.3	8.7
93.75m	16	123.8k	1.6 LSB	35.9	11.4	8.8
93.75m	16	98.4k	1.4 LSB	31.5	11.5	9.0
93.75m	16	78.1k	1.4 LSB	31.0	11.6	9.0
93.75m	16	61.9k	1.2 LSB	28.5	11.7	9.1
93.75m	16	49.0k	1.2 LSB	27.4	11.7	9.2

7.1 ADC Noise Performance Plots

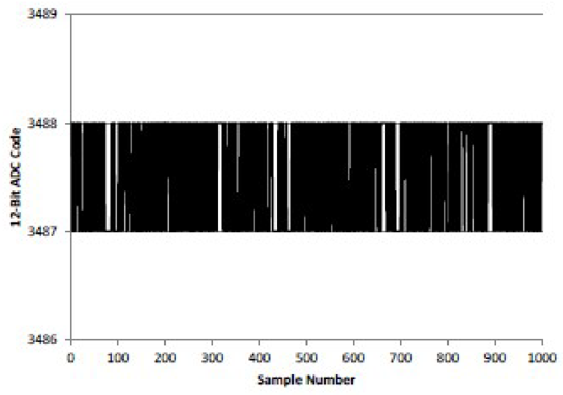

Figure 1. ADC Noise, Differential DC Input, Gain = 1

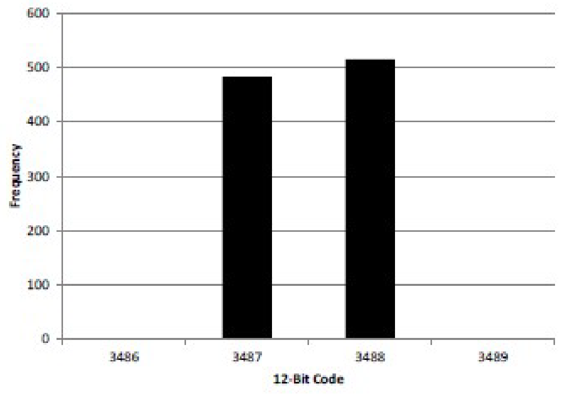

Figure 2. Histogram, Diff, Gain = 1, ODR = 98 KSPS

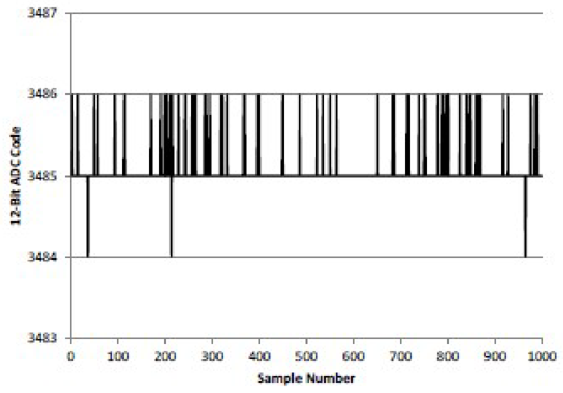

Figure 3. ADC Noise, Differential DC Input, Gain = 2

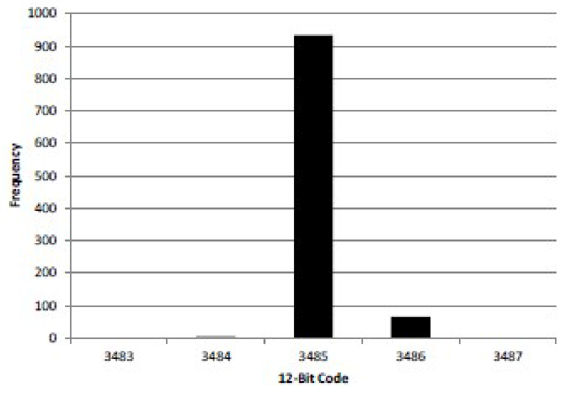

Figure 4. Histogram, Diff, Gain = 2, ODR = 98 KSPS

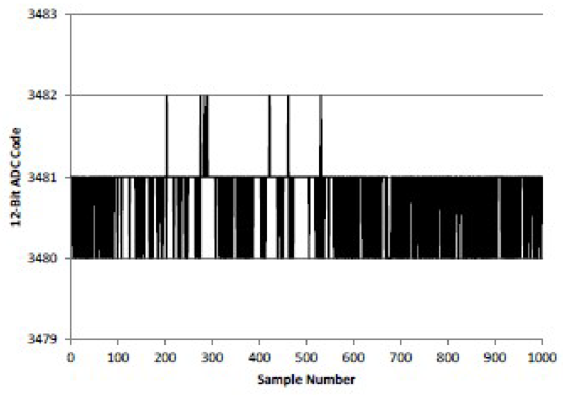

Figure 5. ADC Noise, Differential DC Input, Gain = 4

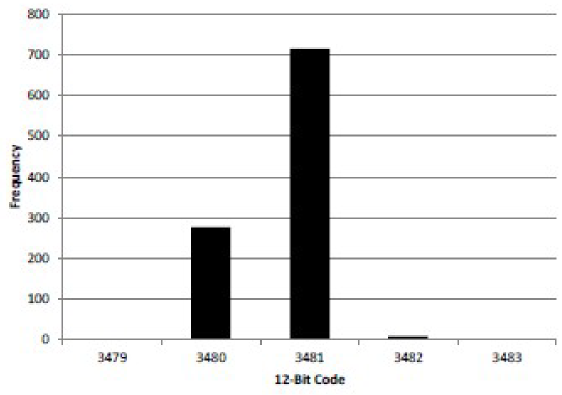

Figure 6. Histogram, Diff, Gain = 4, ODR = 98 KSPS

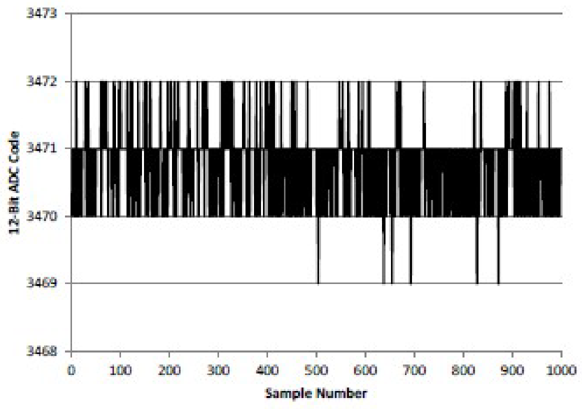

Figure 7. ADC Noise, Differential DC Input, Gain = 8

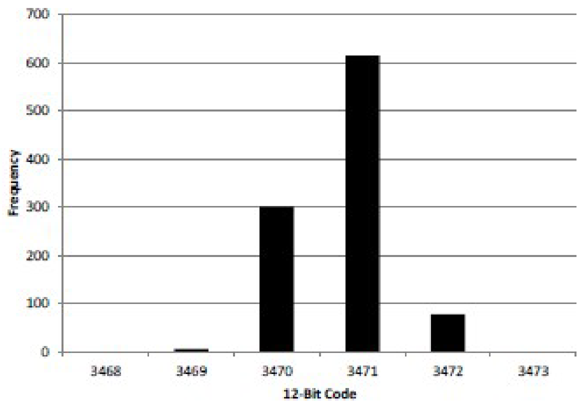

Figure 8. Histogram, Diff, Gain = 8, ODR = 98 KSPS

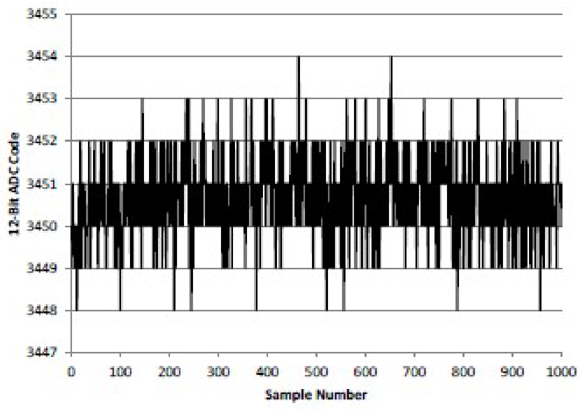

Figure 9. ADC Noise, Differential DC Input, Gain = 16

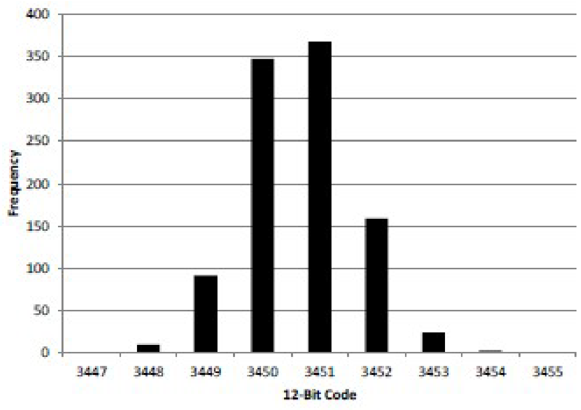

Figure 10. Histogram, Diff, Gain = 16, ODR = 98 KSPS

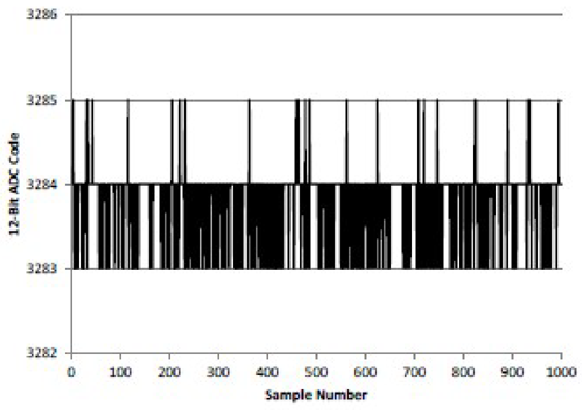

Figure 11. ADC Noise, Single-ended DC Input, Gain = 1

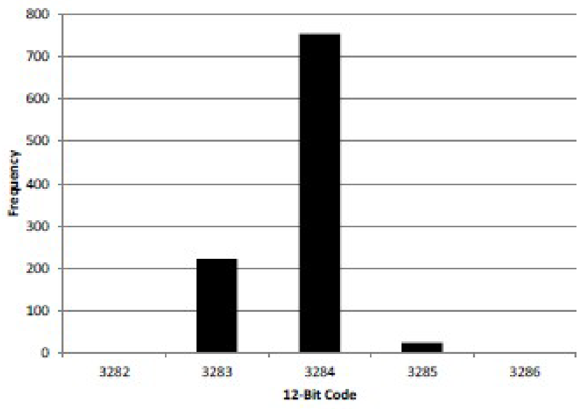

Figure 12. Histogram, SE, Gain = 1, ODR = 98 KSPS

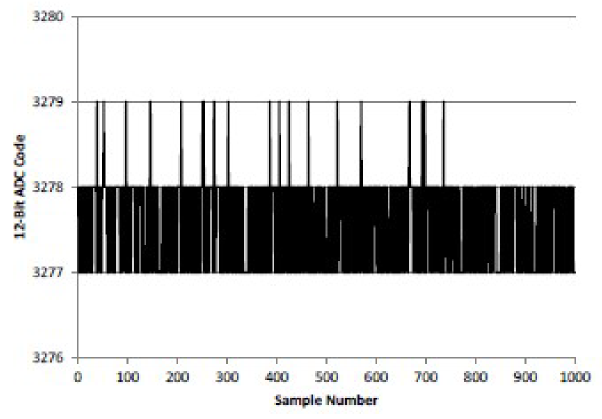

Figure 13. ADC Noise, Single-ended DC Input, Gain = 2

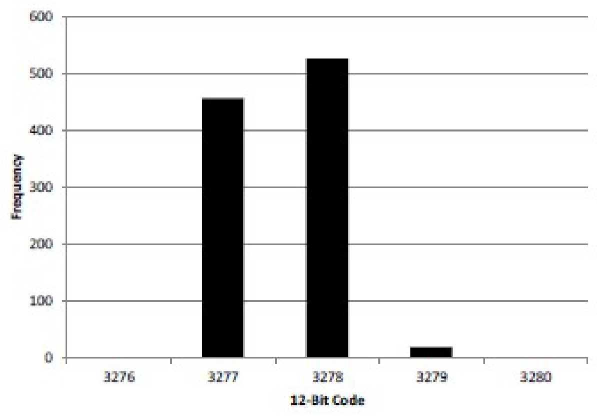

Figure 14. Histogram, SE, Gain = 2, ODR = 98 KSPS

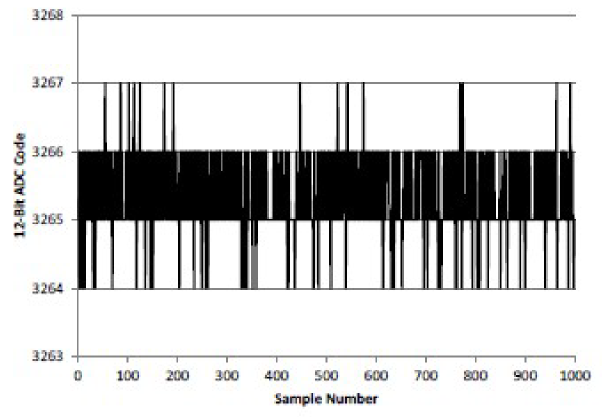

Figure 15. ADC Noise, Single-ended DC Input, Gain = 4

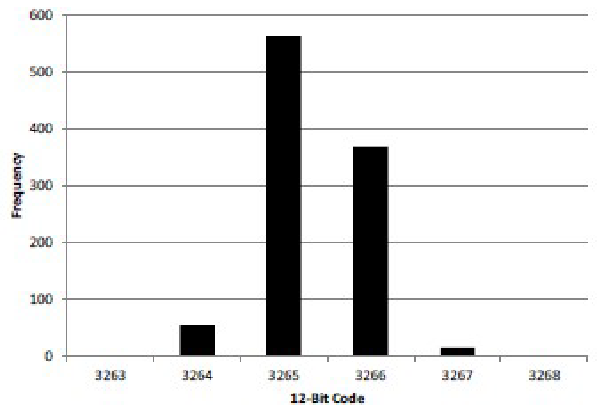

Figure 16. Histogram, SE, Gain = 4, ODR = 98 KSPS

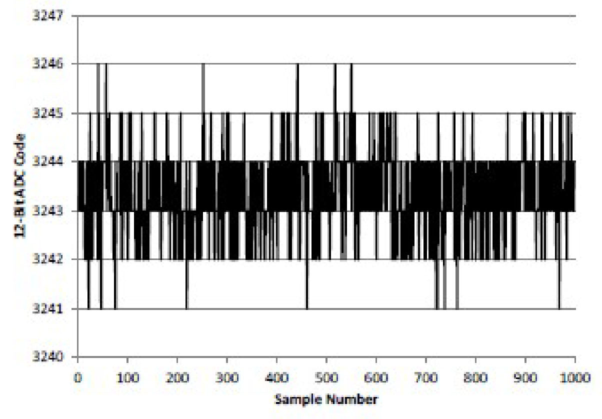

Figure 17. ADC Noise, Single-ended DC Input, Gain = 8

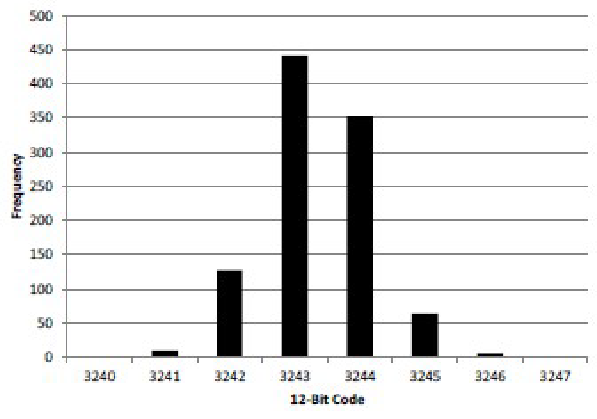

Figure 18. Histogram, SE, Gain = 8, ODR = 98 KSPS

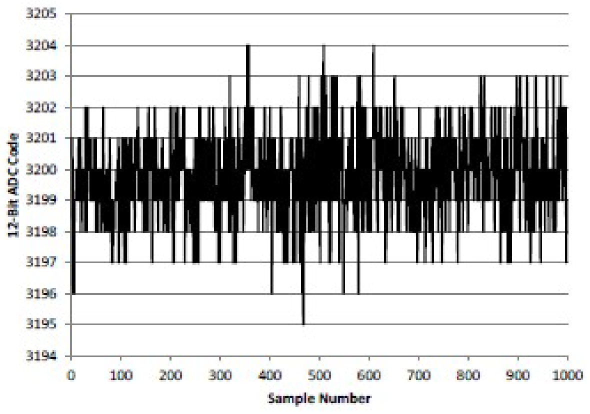

Figure 19. ADC Noise, Single-ended DC Input, Gain = 16

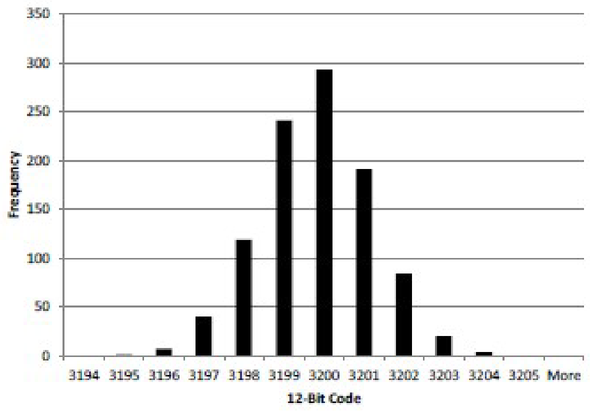

Figure 20. Histogram, SE, Gain = 16, ODR = 98 KSPS

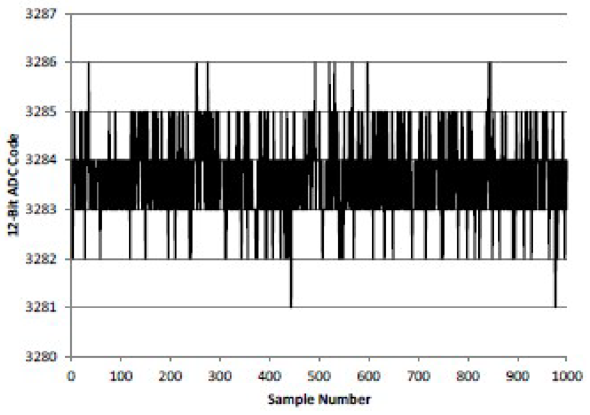

Figure 21. ADC Noise, Single-ended DC Input, Gain = 1, ODR = 198 KSPS

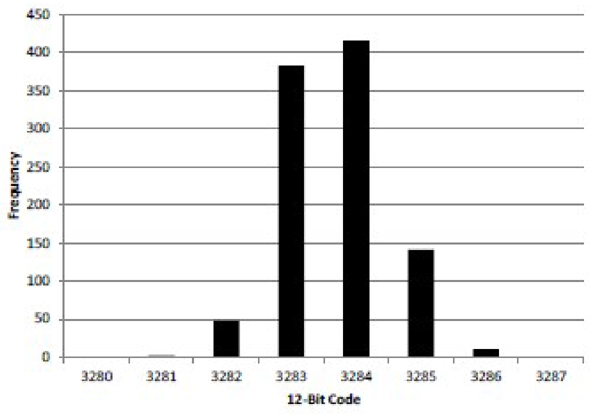

Figure 22. Histogram, SE, Gain = 1, ODR = 198 KSPS

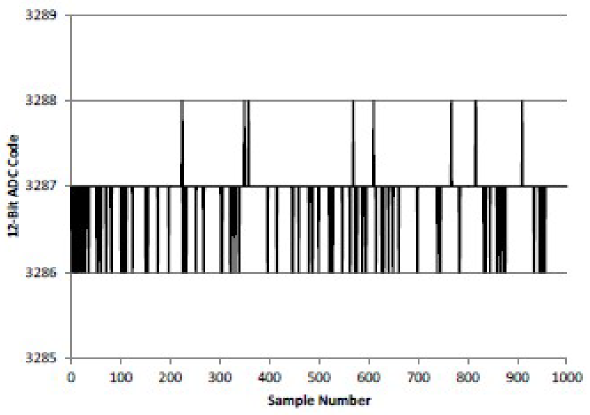

Figure 239. ADC Noise, Single-ended DC Input, Gain = 1, ODR = 49 KSPS

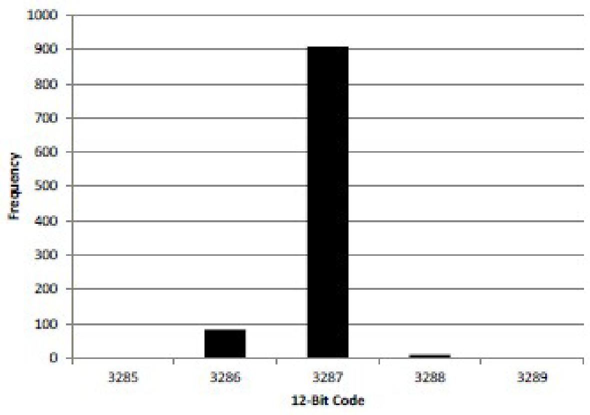

Figure 24. Histogram, SE, Gain = 1, ODR = 49 KSPS

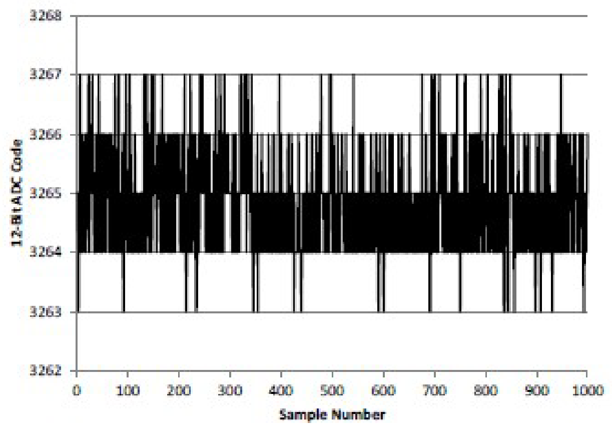

Figure 25. ADC Noise, Single-ended DC Input, Gain = 4, ODR = 198 KSPS

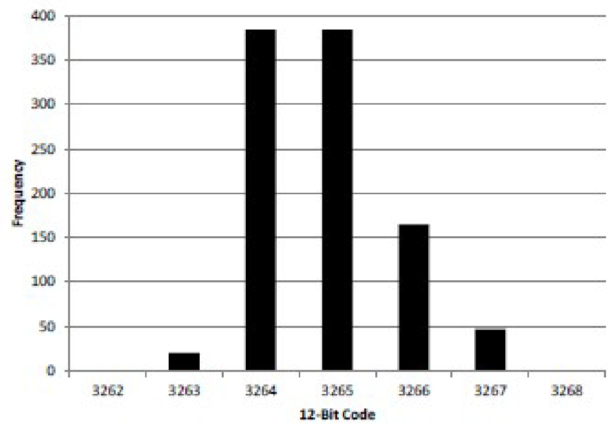

Figure 26. Histogram, SE, Gain = 4, ODR = 198 KSPS

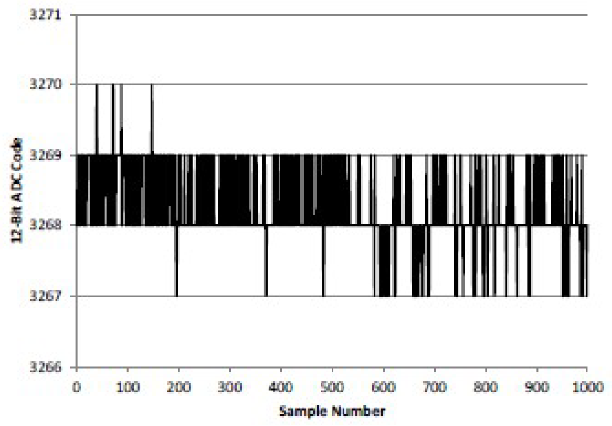

Figure 27. ADC Noise, Single-ended DC Input, Gain = 4, ODR = 49 KSPS

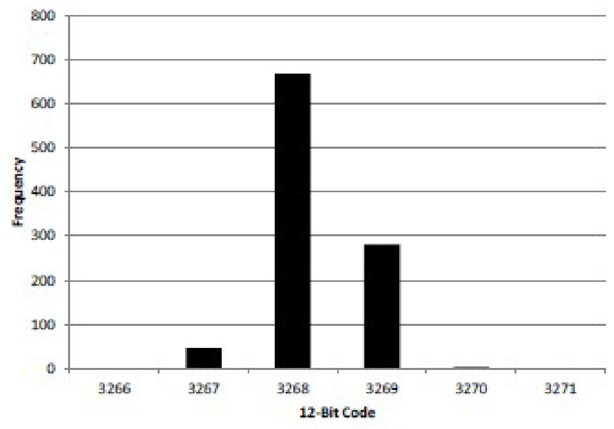

Figure 28. Histogram, SE, Gain = 4, ODR = 49 KSPS

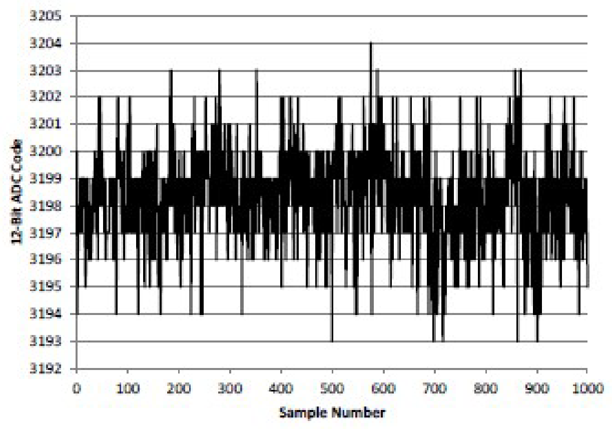

Figure 29. ADC Noise, Single-ended DC Input, Gain = 16, ODR = 198 KSPS

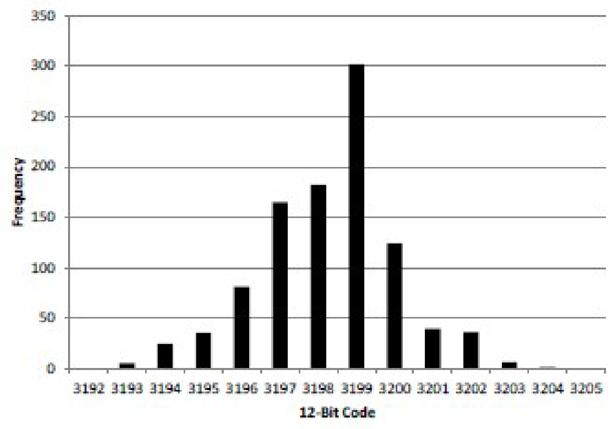

Figure 30. Histogram, SE, Gain = 16, ODR = 198 KSPS

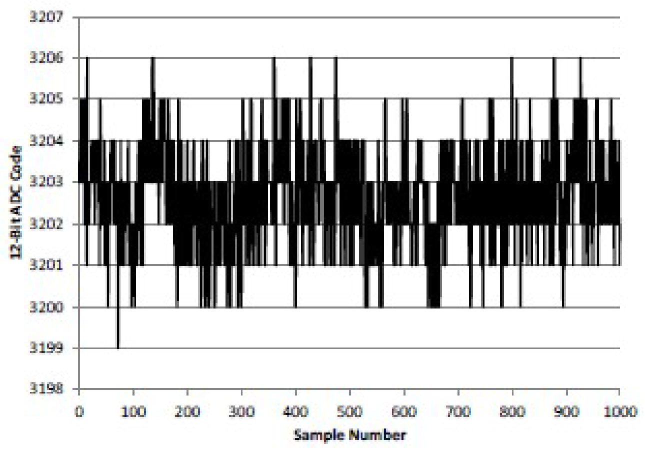

Figure 31. ADC Noise, Single-ended DC Input, Gain = 16, ODR = 49 KSPS

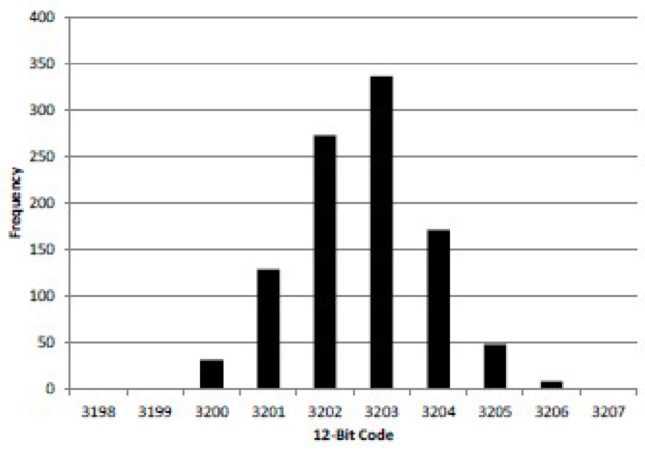

Figure 32. Histogram, SE, Gain = 16, ODR = 49 KSPS

7.2 Differential and Integral Linearity Plots

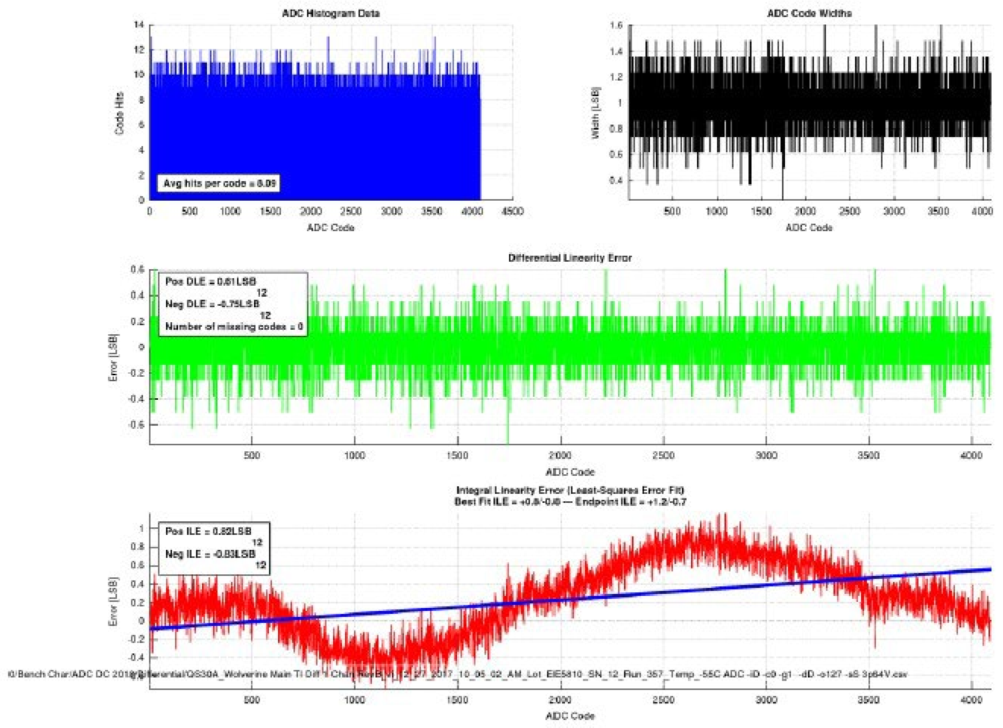

Figure 1: Differential and Integral Linearity Plot - Differential Analog Input

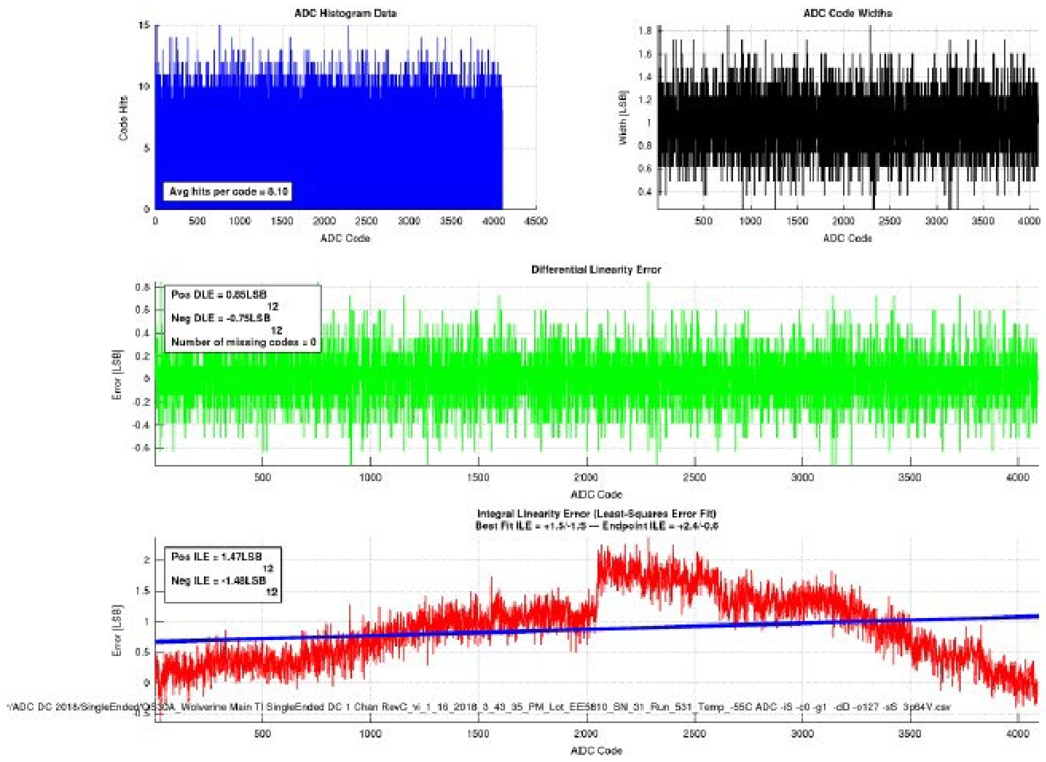

Figure 2: Differential and Integral Linearity Plot - Single-Ended Analog Input

7.3 FFT Spectral Analysis Plots

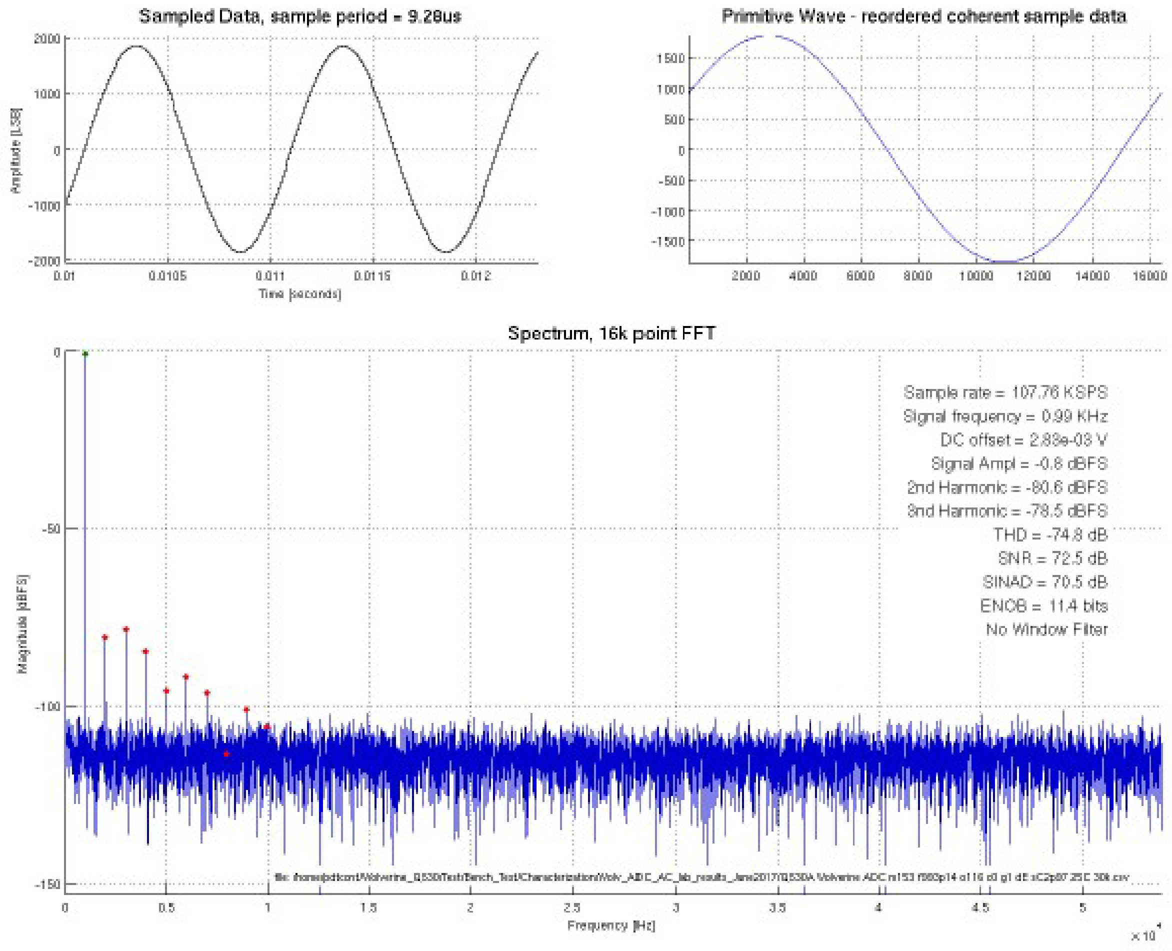

Figure 3: FFT Spectral Analysis - Differential Analog Input

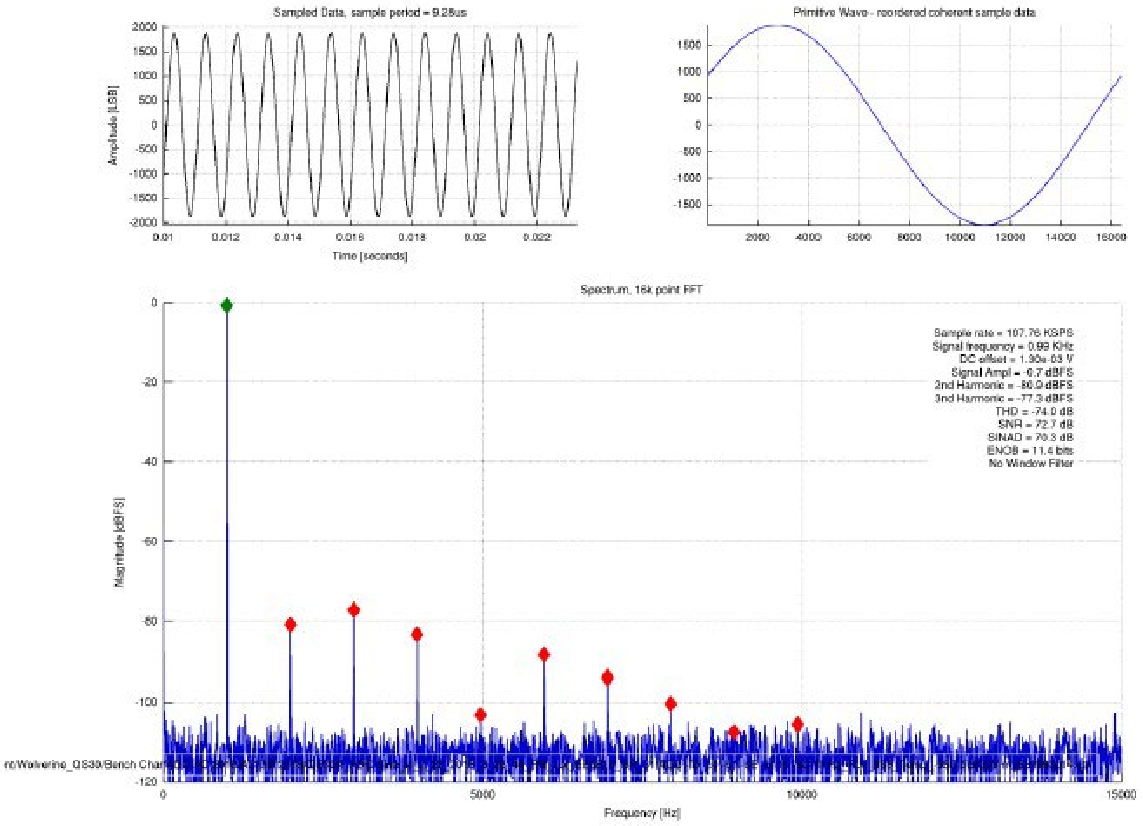

Figure 4: FFT Spectral Analysis - Differential Analog Input, Zoomed 15kHz

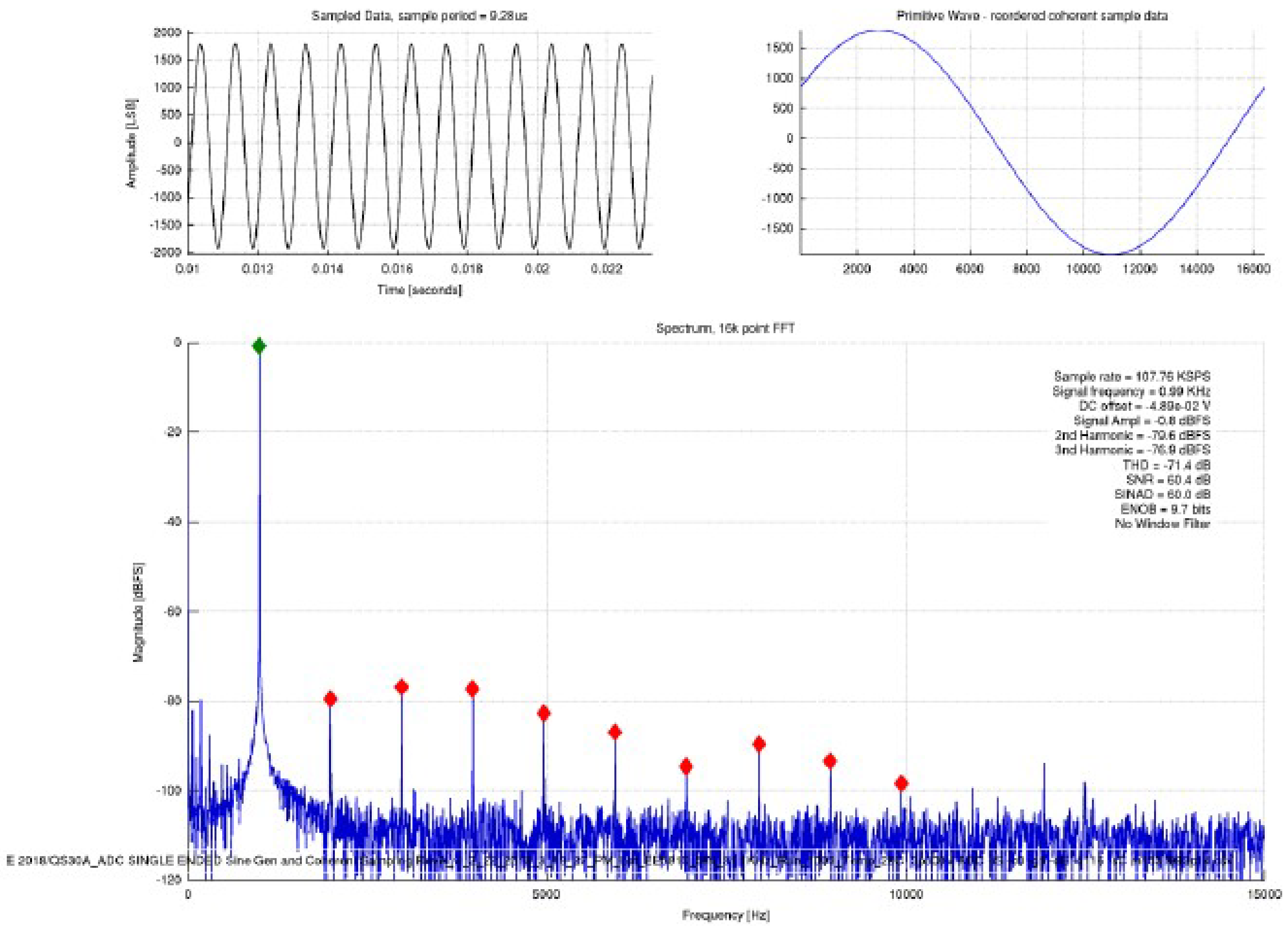

Figure 5: FFT Spectral Analysis – Single-ended Analog Input, Zoomed 15KHz

8 ADC Errata

8.1 ADC Continuous Mode Wrap Around Hardware Inaccurate Delay

In Continuous mode, the ADC performs continuous conversions sequentially on all enabled channels. From one channel to the next, the hardware has a predefined delay for setup time that allows for an accurate A DC conversion. Once the last channel finishes an A DC conversion, the hardware wraps around to the first enabled channel in the sequence to begin a new conversion. When the hardware wraps around from last to first channel in the sequence, the predefined delay for setup time is reduced and causes an inaccurate ADC conversion reading in the first enabled channel.

Note: the “first channel” is not physical channel #1, but the first enabled channel in the sequence, i.e., channel #2 in enabled channel 2-3, channel #5 in enabled channel 5-10, channel #1 in enabled channel 1-15 or any other sequence combination.

Workarounds

1. Run the A DC in Single Sweep mode, and add a software delay of 28µs minimum between each sweep completion (C ONV_COMPL_COMB and INTR_PEND bit of highest enabled channel) and the triggering of the next sweep
2. In continuous mode, configure the first enabled channel as a "dummy" channel, and only keep data from the second enabled channel and beyond.

8.2 ADC External vs Internal Oscillator CONV_COMPL Conflict

When the UT32M0R500 uses an external clock as its system clock (PCLK), the ADC peripheral still uses the internal 50MHz oscillator to perform conversions (at a clock divided rate determined by the ADC_OSCDIV[1:0] bits in the ADC.TIM_CTRL register). When using an external system clock, the ADC_CONV_COMPL bit, which is supposed to set at the end of a conversion, will sometimes not set. The DATA_VALUE_STALE bit located in the A DC->DATA register, bit 15, used the A DC_CONV_COMPL bit to determine if data was stale. Both bits have been marked as having this chance for error. This is due to a hardware error that occurs a small fraction of the time caused by using two different clocks. When using the internal 50MHz oscillator as the system clock (PCLK), this is not an issue.

Workarounds

Bits 31 and [24:0] found in A DC->INT_STATUS are still accurate and can be used to determine which channels have completed a conversion. Once an enabled channel has been sampled, its respective INTR_PEND bit will set along with the CONV_COMPL_COMB bit. Once the INTR_PEND bit for the highest enabled channel is set along with the CONV_COMPL_COMB, the full A DC sweep is complete. This method does not require users to enable A DC interrupts, regardless of the bit/register names.

8.3 ADC COI3_OVER Low Voltage False Flag

The COI3_OVER flag signals to the user when the input voltage from the most recent measurement is out of range. Although it is called COI3_OVER, the flag signals both over-voltage and under-voltage measurements. When the input into a Single-Ended channel is at or below 0.003V (3mV), the COI3_OVER flag has a chance to set. This chance increases as the input voltage approaches 0V, to the point where the COI3_OVER flag is always set. This false flag only occurs when all of the data bits in the respective channel’s A DC.DATA register are equal to zero (both the DATA_OUT[11:0] bits and the DATA_OUT_LSB[3:0] bits. If any of these data bits are set in conjunction with the COI3_OVER flag, the flag is operating as intended.

Note that the 3mV value was measured using a high-precision voltage supply and noise-reducing practices. Noise on the voltage input will cause the COI3_OVER false flags to occur at a higher voltage threshold.

Workarounds

If this flag is set and the DATA_OUT and DATA_OUT_LSB bits of the A DC.DATA register are zero, users can ignore the COI3_OVER flag.

9 Appendix A

9.1 Example 1: Quick Start to Conversion

Here is a brief tutorial to get started quickly using the ADC. Producing a quick and accurate conversion is a straightforward process. A simple routine for digitizing a voltage applied to one input of the A nalog MUX, beginning with the registers' default settings, requires only a few instructions.

Configuration: Analog DC Voltage on a single channel

ADC Register Configuration Details

- ADC.SPB_CFG_1, ADC_REGDEF = 1 (Reset all registers to default)
- ADC.SPB_CFG_0, ADC_SINGLESWEEP = 1 (single sweep conversion mode)
- ADC.SPB_CFG_0, ADC_EN = 1 (power up ADC blocks)
- ADC.SECHAN_CFG_REG_0, EN = 1 (single-ended channel 0 enable)
- ADC.SECHAN_CFG_REG_0, GAIN = 001 (PGA set 1V/V default from reset)
- ADC.SECHAN_CFG_REG_0, DDF2 = 0 (use digital filter type COI3 default from reset)
- ADC.TIM_CTRL, ADC_OSCDIV = 0 (modulator clock = 12.5MHz default from reset) ADC.TIM_CTRL, ADC_DSMOSR =64h (modulator OSR = 100d default from reset)

After the setup is complete, the conversion is performed setting the trigger.

ADC.SPB_CFG_1, ADC_TRIGGER = 1 (convert command)

Poll the status flag and wait for the indication that your conversion is completed.

ADC.INT_STATUS, ADC_INT_CONV_COMP_COMB = 1

ADC.INT_STATUS, ADC_INT_(highest enabled channel) = 1

Read the conversion result from the output data register ADC.DATA_DATA_OUT[11:0]

Here is a working sample block of C code:

```
// Simple ADC Conversion Example: single-channel (channel 0), polled for completion
//
// Definition of SYSCON_TypeDef and ADC_TypeDef structures can be found in UT32M0R500.h // Definition of
SYSCON_BASE and ADC_BASE address constants can be found in UT32M0R500.h
// Definition of ADC_InitTypeDef and ADC_ChanCfgTypeDef structures can be found in ut32m0_adc.h
// ADC constants -- ADC_SWEEP_SINGLE, for example -- can be found in ut32m0_adc.h
// ADC_WxYz() function prototypes can be found in ut32m0_adc.h
#include "UT32M0R500.h" #include "ut32m0_adc.h"
// pointer to SYSCON structure
SYSCON_TypeDef *SYSCON = (SYSCON_TypeDef *) SYSCON_BASE;
// pointer to ADC structure; ADC/channel initialization/configuration structures
ADC_TypeDef *ADC = (ADC_TypeDef *) ADC_BASE;
ADC_InitTypeDef ADC_InitStruct;
ADC_ChanCfgTypeDef ADC_ChanCfgStruct;
uint16_t Result;
// turn ON the precision references
SYSCON->ANALOG_SHUTDOWNS = 0;
// set the ADC initialization structure to its defaults, ...
ADC_StructInit (&ADC_InitStruct);
// ... change the settings according to the example, ...
ADC_InitStruct.SweepType = ADC_SWEEP_SINGLE;
ADC_InitStruct.OscillatorDivisor = ADC_OSCDIV_BY_4;
ADC_InitStruct.ModulatorSamples = 0x64;
// ... and initialize the ADC
ADC_Init (ADC, &ADC_InitStruct);
// set the channel initialization structure according to the example
ADC_ChanCfgStruct.Enable = ENABLE;
ADC_ChanCfgStruct.Gain = ADC_GAIN_1VperV;
ADC_ChanCfgStruct.UseDDF2 = FALSE;
// configure channel 0
ADC_SetChannelConfig (ADC, ADC_SE_CHAN_0, &ADC_ChanCfgStruct);
// start the conversion
ADC_Sweep (ADC, ENABLE);
// wait for the conversion to complete (see External vs Internal Oscillator Errata)
while ((ADC->INT_STATUS & 0x10000001)!=0x10000001) ;
// read the result data from channel 0, ignoring any error data
ADC_ReadChannel (ADC, ADC_SE_DATA_0, &Result, NULL);
// clear the completion flag
ADC_ClearConvCompleteFlag (ADC); // print the result to STDIO printf ("ADC channel 0 result = 0x04X\r\n", Result);
////////////////////////////
```

10 Revision History

Date	Revision #	Author	Change Description	Page #
02/01/2019	0.1.0	BY	Initial Draft	
07/15/2019	0.1.1	OW	Converted Template	
04/15/2020	1.0.0	OW	Initial Release	
04/23/2020	1.0.1	OW	Template Update	

Datasheet Definitions

	Definition
Advanced Datasheet	Frontgrade reserves the right to make changes to any products and services described herein at any time without notice. The product is still in the development stage and the **datasheet is subject to change**. Specifications can be **TBD** and the part package and pinout are **not final.**
Preliminary Datasheet	Frontgrade reserves the right to make changes to any products and services described herein at any time without notice. The product is in the characterization stage and prototypes are available.
Datasheet	Product is in production and any changes to the product and services described herein will follow a formal customer notification process for form, fit or function changes.