Technical notes on using Analog Devices DSPs, processors and development tools
Visit our Web resources http://www.analog.com/ee-notes and http://www.analog.com/processors or e-mail processor.support@analog.com or processor.tools.support@analog.com for technical support.

Maximizing ADC Sampling Rate on ADSP-CM40x Mixed-Signal Control Processors

Contributed by Prashant Gawade *Rev 1 – September 20, 2013*

Introduction

The ADSP-CM40x Mixed-Signal Control Processors is a 3-die SiP (System in Package) consisting of a digital die (ARM® Cortex-M4™ processor core, hardware accelerators and peripherals block), the Serial Peripheral Interface (SPI) flash memory die and the analog die (Analog-to-Digital Converter, ADC, and Digital-to-Analog Converter, DAC). The analog die contains two 16-bit high performance, low power, successive approximation register (SAR) ADCs, with sampling rates of up to 2.63 mega samples per second (MSPS), and also contains two 12-bit DACs[2].

The digital die also consists of an on-chip Analog-to-Digital Conversion Controller (ADCC) which is tightly coupled to control the real-time operation of the ADCs on the analog die. It automates the ADC sampling sequences precisely, thereby reducing the core overhead required as compared to traditional methods of ADC sampling and data acquisition.

This EE-Note focuses on the ADC part of the analog die, providing a detailed description of how the ADCC should be configured to sample the internal ADC to achieve the maximum supported rate of 2.63 MSPS. This document concludes explaining the ways to measure the key parameters of the ADC, such as Signal-to-Noise ratio (SNR), Signal to Noise-plus-Distortion Ratio (SNDR), Total Harmonic Distortion (THD) and Effective Number of Bits (ENOB).

This EE-Note is particularly helpful for applications requiring continuous ADC data sampling at high data rates. However, the approach descried hereafter may not be needed in applications where a single ADC data sample is captured based on a given system event or when sampling at low speeds.

Sample code provided in the associated `.ZIP` file calculates the ADC performance metrics by performing a 64K-point Fast Fourier Transform (FFT) on the samples acquired through the ADCC. The example also illustrates the effective usage of processor's memory for such complex applications.

This document assumes that the reader is familiar with the ADSP-CM40x ADCC module and key ADC performance parameters. For a complete description of the ADCC module, refer to the *ADSP-CM40x Hardware Reference Manual* [1].

ADC Sampling Sequence

The ADCC controller provides clock and chip select signals required for ADC sampling. Two control lines are provided to send the control word for the ADC conversion and two data lines for reading the ADC converted data. The ADC sampling sequence is initiated by the trigger signal and it's based on activation time configured into the `Event Time` register. ADCC provides 24 events for initiating these sampling sequences, each of which can be independently configured to initiate sampling of one of the ADC channels at the required time with respect to the incoming trigger signal.

A typical ADC sampling timing for a single ADCC event is as shown in Figure 1 below:

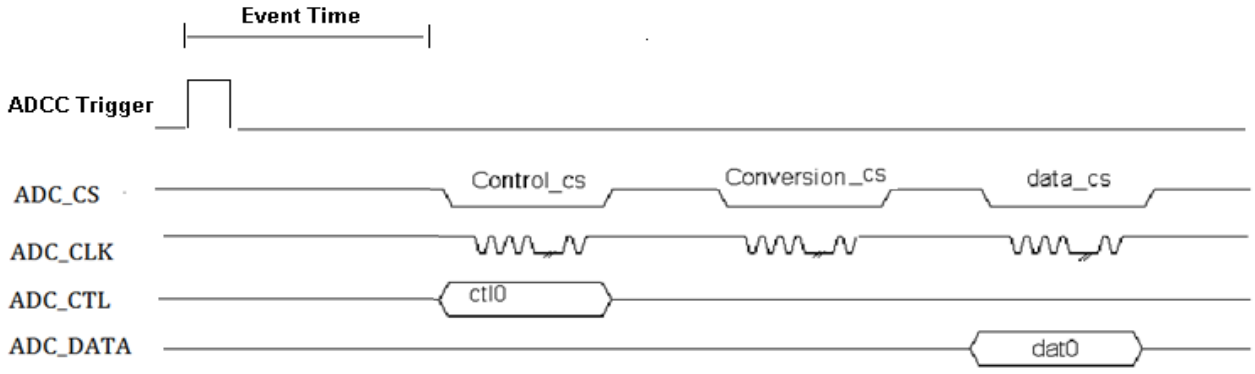

Figure 1. Internal ADC sampling sequence timing

The ADC sampling sequence is divided into three phases. The ADCC controller provides gated clock and chip select/start of conversion signal for each of these phases:

- In the first phase, the ADCC controller sends the control word to select the ADC channel for conversion. Each ADC provides multiple input channels and one of them can be selected for conversion by specifying the channel ID while sending the control word.
- In the second phase, the ADCC controller sends a conversion pulse to initiate the conversion for the selected ADC channel.
- In the third phase, the ADCC controller reads the converted data from the ADC. The converted data is directly available in the data register of that event, which can be read in either core mode or directly transferred via DMA (Direct Memory Access) into the processor's memory.

The ADCC controller provides two timers to handle these events. Each timer can accept independent trigger signals and activates the events assigned to it when their event time expires with respect to the incoming trigger signal. The activated events are forwarded to the ADC interface to carry out the ADC sampling sequence.

Once all the events associated with an ADCC timer are completed, an ADCC Timer frame is said to be completed for that trigger input and a status bit is flagged. This status bit must be cleared before arrival of next trigger input; otherwise upcoming trigger inputs will be ignored, leading to trigger overrun condition for that ADCC Timer and uneven ADC sampling.

When an ADCC Timer frame is completed, a trigger output signal is also generated by ADCC. The ADCC control signals are not available externally on processor pins. Therefore, for debug purpose, the ADCC Timer trigger input signals and frame completion ADCC Timers trigger output signals can be used to understand frame completion timing. If the time between input and related output trigger signal is more than expected value, then it can be interpreted as servicing events might be getting shifted and it may result in uneven sampling or event miss error condition.

ADC Timing at Maximum Throughput rate

As shown in Figure 1, when the ADCC controller starts to execute an event, the ADC conversion data for that event is only available during the 3rd frame sent across. So, if another event becomes active during this time, it is said to be collided and it's servicing would get delayed. In order to reduce the latency in servicing of such collided events, the ADCC controller may decide to pipeline the ADC sampling sequence of different events.

The below example shows execution time of events that are configured such that the ADC interface tightly pipelines the ADC sampling sequences associated with those events. In this case, when the ADC interface sends the control word for an event, it provides the conversion pulse to the previous event and also reads the converted data of the event before it. This case is referred to as the "maximum throughput" case, as the ADC would be continuously sampling its channels, providing samples at every cycle of the ADC sampling phase, as shown in Figure 2:

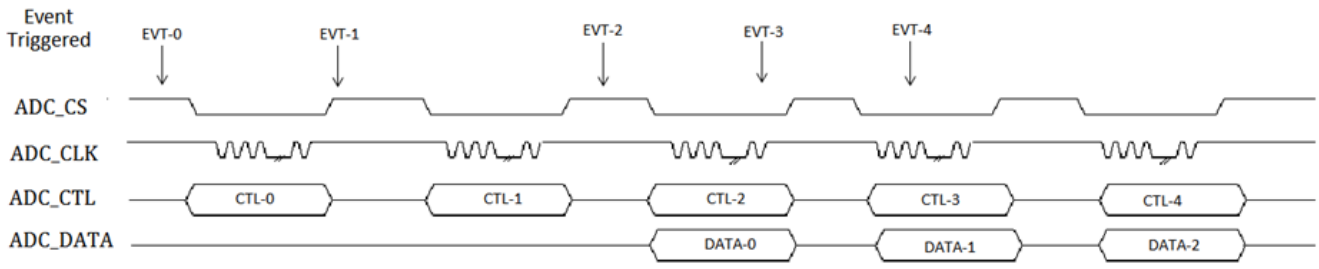

Figure 2. ADC timing in maximum throughput case

In this case, for example in the third `ADC_CS` cycle, the ADC sampling sequence phases of `EVT-0, EVT-1` and `EVT-2` are tightly pipelined; i.e., the control word for `EVT-2` has been sent, the conversion pulse for `EVT-1` has been provided and also the ADC conversion data for `EVT-0` has been read.

The `Event Time` register of events should be configured such that it becomes active during the control phase of the ADC sampling sequence of the previous event or earlier, but it should not lead to event misses. The event time, in this case, depends on the configuration of the ADCC timing control registers, which determine the period of the ADC sampling phases, which further depends on the ADC conversion time. The activated events waiting to be serviced are placed into a 4-deep pending event FIFO.

The maximum throughput case indicates the ADC maximum sampling rate. It doesn't necessarily have to be the best case for the ADC performance parameters.

Problem in Implementing the above Approach

The ADC sampling sequences are initiated by ADCC events. Consider a case where the channels from an ADC are to be sampled at maximum throughput rate and therefore sampling sequences need to be pipelined. Since the ADCC hardware provides 24 events, a maximum of 24 sampling sequences can be initiated per ADCC trigger, assuming all events are enabled and assigned to single ADCC timer. Also, as explained in ADC Sampling Sequence section, the ADCC Timer frame completion interrupt bit in `ADCC_FISTAT` register must be cleared in the software before arrival of next trigger pulse.

So, considering these points, the ADCC timing to achieve maximum throughput will look as shown in Figure 3:

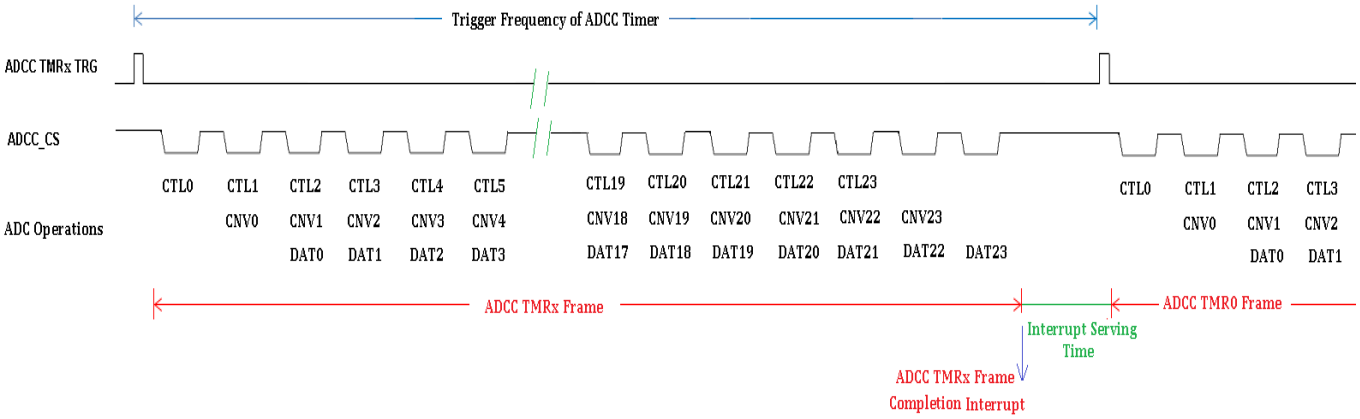

Figure 3. ADCC timing for maximum throughput case with simple approach

Note that, there are no ADC samples at the start of frame for the first two phase cycles and while serving frame completion interrupt, thereby producing uneven ADC sampling. Also the consideration for interrupt service time makes difficult to calculate the periodicity of trigger input signal when required to sample the ADC at maximum possible rate. Therefore, this simplest approach of using single ADCC Timer, may not sample the ADC at its maximum sampling rate.

Proposed ADCC Configuration

In order to continuously receive the ADC samples without causing uneven sampling, both ADCC timers should be used instead of a single timer, as shown earlier. The ADCC events can be equally divided between these two timers. The trigger inputs of these timers should be interleaved such that the last phase of an ADCC timer frame should overlap with the initial phase of the other ADCC timer frame to create tightly pipelined ADC sampling across successive triggers. Therefore, the frames of both ADCC timers will be interleaved, and spaced apart such that processor will get sufficient time to clear the frame completion bits in-between to avoid the trigger overrun condition.

Different triggers would be given to these ADCC timers to interleave their frames, as shown in Figure 4.

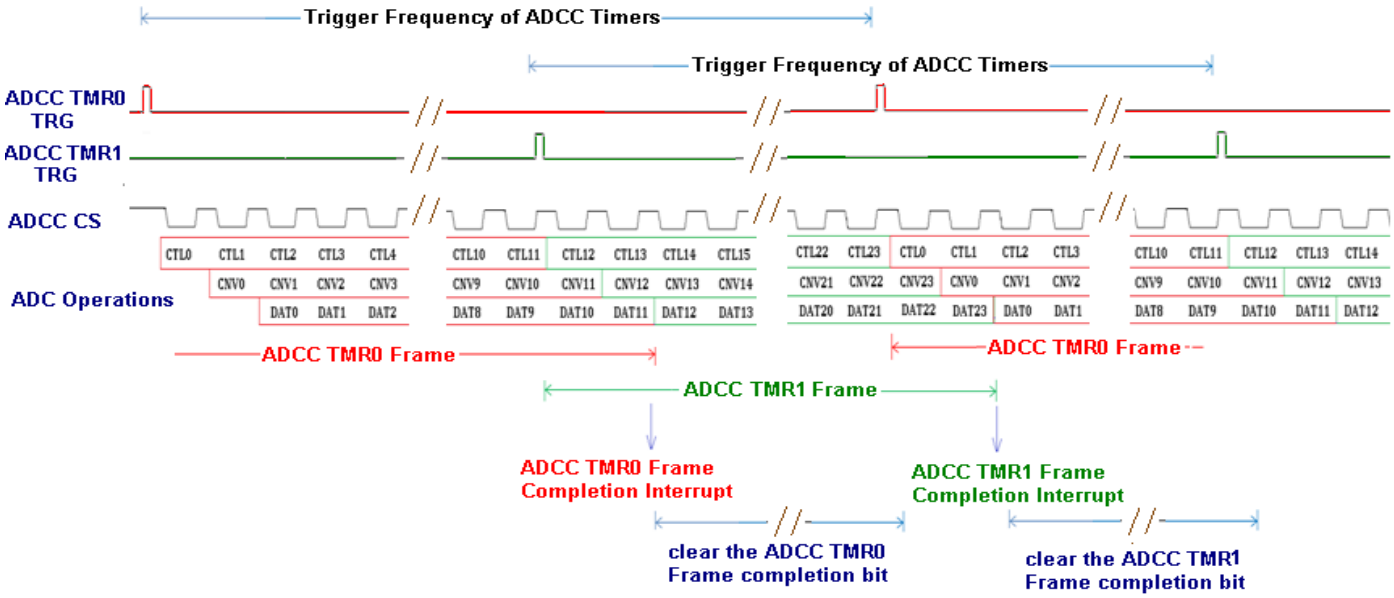

Figure 4. Proposed ADCC timing for maximum throughput case

In the above case, 12 events are assigned to each ADCC timer: `EVT0-11` to ADCC `Timer0` and `EVT12-23` to ADCC `Timer1`. The frequency of triggers to both ADCC timers is constant and is equal to 24 `ADCC_CS` phase cycles; but the two triggers are separated by 12 phase cycles from each other. This should be sufficient to clear the frame completion bit before the arrival of the next trigger corresponding to its ADCC timer. The ADCC timer frame completion interrupts should be given higher priority to be served in a timely manner.

Optionally, the ADCC timer frame completion interrupt latches can be cleared using Memory-to-Memory DMA channels (MDMA), if available for use. For this, MDMA can be configured in trigger wait mode to write to the `ADCC_FISTAT` register. The trigger to the MDMA channel can come from the ADCC controller. This approach requires two MDMA channels; each will be accepting trigger input from one of the ADCC Timer for each write. Whenever an ADCC timer frame is completed, an output trigger signal is generated, and this can be internally routed to the trigger input signal of the MDMA channel, which will in turn write to the `ADCC_FISTAT` register to clear the required interrupt latch bit, without requiring any processor time.

To confirm whether ADC is being sampled continuously at required sampling rate, the application should enable the ADCC interrupt for error conditions such as Trigger Overrun and Event Miss. Since all the events are being activated when ADCC is busy, the error interrupt for Event Collision should not be enabled when using this approach.

For this approach to work reliably, the accurate timing of ADCC trigger input signals plays an important role. If possible, try to generate it internally from GP Timer using its advanced features such as trigger output and delay in the period.

Software Details

Two example codes are provided in the associated `.ZIP` file to demonstrate the ADCC configuration methods. The first example, `"Single_channel_2631"` illustrates how an ADC channel can be sampled at the maximum ADC throughput rate of 2.631 MSPS, and then calculates some of the key ADC performance metrics on the samples acquired, to show the effective usage of processor's memory for such applications.

The second example code `"Single_channel_2631_MDMA"` is a slightly modified version of the above example in which the ADCC timer frame completion interrupts are cleared by the MDMA channel to save processor bandwidth.

The provided example projects have been tested using the ADSP-CM408F EZ-KIT Lite® evaluation board[3]. The following sections provide some of important information about these two examples. Refer to the *Programming Guidelines* section in the *Processor Hardware Reference Manual* [1] for more details.

Clock Generation Unit Configuration

The ADC maximum throughput is achieved when the ADC clock is set to 50 MHz. To achieve this, it is best to configure the processor Core Clock (`CCLK`) to 225 MHz and System Clock (`SCLK`) to 100 MHz. Note that default processor state is active mode, which should be changed to Full-On mode before configuring the processor's Clock Generation Unit (CGU).

ADCC Event Configuration

After completion of each ADCC timer frame, it is required to clear the associated frame completion interrupt latch bit in the status register. Clearing it within the Interrupt Service Routine (ISR) can be one approach. In order to service it in a timely manner and to avoid ADCC trigger overrun errors, the frame completion interrupts should be given high priority. In order to reduce this interrupt rate, it is best to enable all the available ADCC events, i.e., 24 events.

In the code example, one of the ADC channels is sampled at the maximum rate, so all events are configured to initiate sampling of that ADC channel. But these 24 events are divided into two groups of 12 events: `EVT00-11` assigned to ADCC `Timer0`, and `EVT12-23` assigned to ADCC `Timer1`.

The event time of these events is configured such that each event becomes active while sending the control word phase of the previous event. This makes sure that the control word of a newly active event is sent when the conversion pulse of the previous event is being sent. Therefore, the ADC sampling sequences for the events are tightly pipelined to achieve the maximum ADC sampling rate. See Proposed ADCC Configuration section above.

The event offset field of the event control registers are configured such that samples from the ADC are stored successively in a buffer irrespective of the ADCC timer to which the event is assigned.

ADC Interface Timing Configuration

The ADSP-CM40x on-chip ADC conversion time is 380 nanoseconds. With the ADC clock set to 50 MHz, there should be 19 ADC clocks in one `ADCC_CS` period, which are divided in terms of fields of the ADCC timing control registers as follows: `TCSCK = 1`, `NCK = 8`, `TCKCS = 0` and `TCSCS = 10` (Figure 5):

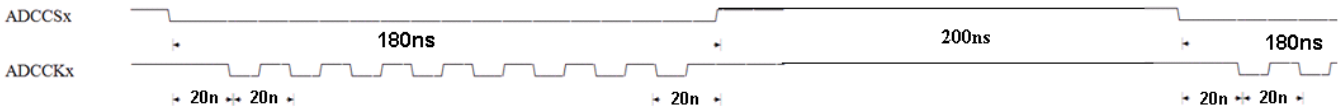

Figure 5. ADCC timing control register setting

With data being received in each `ADC_CS` period, the ADC throughput can be calculated as:
`1/380nsec = 2.631MSPS`

ADCC Trigger configuration

The events are distributed to assign `EVT0-11` to ADCC `Timer0` and `EVT12-23` to ADCC `Timer1`. The trigger signals to these ADCC timers should be generated suitably to realize the timing explained in the Proposed ADCC Configuration section. Two trigger inputs are used to interleave the frames from each ADCC timer. One trigger input is assigned to each ADCC timer, based on which the timer will activate the 12 events assigned to it as per the event time programmed.

Though each trigger handles 12 events, the interleaving of frames from both ADCC timers is such that period of trigger input is equal to 24 ADC sampling phase cycles. Therefore, the trigger input period in terms of ADC clocks, is calculated as:

```
Total No. of Events x (TCSCK + NCK + TCKCS + TCSCS)
24 x (1 + 8 + 0 + 10) = 24 x 19 = 456 ADC clocks
```

In the example code, these two ADCC timer trigger inputs are generated from trigger outputs of General Purpose (GP) Timers configured in `PWM_OUT` mode. Two GP timers are used to generate these two triggers, with period equal to `456 ADC clocks = 456 x (SCLK/ADC_CLK) = 456 x (100M/50M) = 912 SCLKs`

These two triggers should have the timing with respect to each other such that they should be interleaved at half of their period.

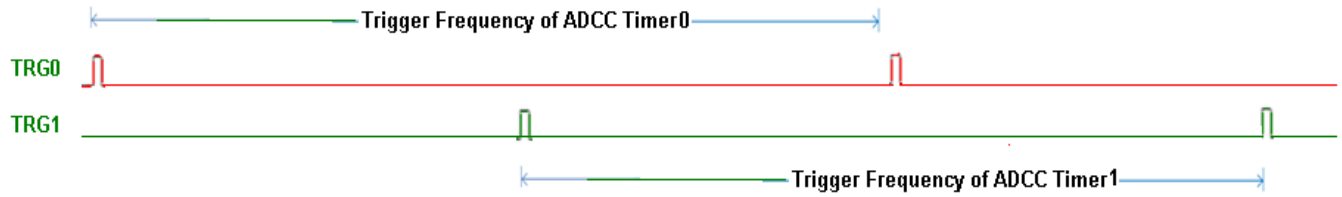

Figure 6. ADCC timer triggers

In fact, it is best if the trigger appears just before the actual timing, so that the associated ADC timer will have sufficient time to pipeline the events related to it.

The delay feature of the GP timer is used to achieve such timing. The delay register of one timer is programmed such that the trigger output appears just near `Timer_Period/2`, while the delay register of the other timer is programmed such that the trigger output appears just before the end of the period.

The GP timer triggers are configured to generate on expiry of delay. Enabling interrupt for these timer units is not required.

DMA configuration

The ADC sampling process is automated by the ADCC hardware, such that after sending control and conversion cycle, the converted data is directly available to read in the data register of the related event. The ADCC controller supports core mode and DMA mode to read these received ADC samples.

In the provided examples, the DMA-based approach is used, as it is useful when receiving ADC samples at the maximum sampling rate. In this mode, the ADCC controller directly stores the received ADC samples in the specified processor's internal memory space, thereby reducing the core overhead.

The event offset field of each event control register is configured such that received samples are stored sequentially in the data buffer submitted to the DMA channel; in the order the ADC has been sampled.

Interrupt handling

In the "Single_channel_2631" example code, the frame completion interrupts of each ADCC timer are enabled to clear the required frame completion status bit. The interrupts have been given high priority to service it in a timely manner.

When circular buffer mode is used for DMA operation, and needs any processing at the end of reception of whole buffer, then a check should be performed on number of ADCC frames completed and then disable the ADCC trigger inputs temporarily to avoid overwriting the same buffer. This check is performed in the ISR. Once the processing is done, the trigger inputs can be re-enabled.

In the "Single_channel_2631_MDMA" example code, these frame completion status bits are cleared using MDMA channels, which accepts the trigger from the ADCC controller to write to the status register. The MDMA count can be configured to the required number of ADCC frames, after which trigger overrun error will stop the ADCC operation. After this, processing on acquired samples is carried out and then the MDMA channel is re-enabled to acquire new samples.

Additional Details

The code calculates some of the ADC performance parameters by processing the samples acquired through the ADCC controller, performing a 64K-point FFT. The coefficients required for this processing task are stored in the internal SPI flash memory, which can be memory mapped for read operations.

These coefficients must be flashed before executing these ADCC codes, which calculate the ADC performance parameters. "SPI_Int_Flash_Write" example code is also provided in the associated .ZIP file. These coefficients are flashed in the upper sectors of flash memory.

The code example displays the calculated ADC performance parameters on the 20 x 2 character display through the Two-Wire Interface (TWI)[3].

The code first acquires the required number of samples, to then calculate the ADC performance parameters. During this time, the ADCC controller operation is halted, as to not overwrite these samples. Once the parameters are calculated and displayed on the character display, the process repeats by re-enabling the ADCC controller operation. LED4 on the evaluation board blinks after completion of each process.

Processing ADC Samples in Internal Memory

The processor core in the digital die can be used to calculate some of the key ADC performance parameters by processing the samples received. These parameters include SNR, SNDR, THD and ENOB.

Signal-to-Noise ratio (SNR) quantifies how much a signal has been corrupted by noise, by comparing the level of a desired signal to the level of background noise. While calculating SNR, generally the noise components at the harmonics of the signal are ignored from the noise power. These harmonic noise components are referred as Total Harmonic Distortion (THD). But these harmonic noise components are considered while calculating Signal-to-Noise-and-Distortion ratio (SNDR). Since SNDR is combination of SNR and THD parameters, it is a good choice for representing the overall dynamic performance of an ADC. The effective number of bits (ENOB) is another measure of quality of the ADC. It specifies the number of bits in the digitized signal above the noise floor and is calculated as (SNDR – 1.76)/6.02.

Since the ADC sampling rate is high, a large buffer is required to store the ADC samples. Typically with a 1 KHz sine wave as an analog input signal to the ADC and a sampling rate of 2.631 MSPS, the number of samples received per cycle will be 2631 samples. For better calculation of ADC performance parameters, at least 10-15 cycles should be acquired, so the buffer size can be nearly of the size of 40K samples.

For signal and noise power calculations, the received samples in the time domain should be converted to the frequency domain by performing an FFT. Typically, a 64K-point FFT is needed to process this buffer. For demonstration purposes, a storage buffer of 70K samples is used, which will allow to select the required portion of ADC samples. Processing such a large buffer to calculate the ADC parameters is a challenge given the available internal memory space. Note that the ADC is 16-bits, so each sample requires 2 bytes for storage. This data buffer can fit into the processor's internal L1 memory (up to 384 Kbytes[2]).

A windowed filter on the ADC samples is required before calculating the FFT. The example code uses Blackman Windowing, though other windowing functions can also be employed. Since the ADC provides the samples in unsigned short format, these are first converted into signed format. The 64K-point FFT needs a Blackman window of 64K points, which in turn needs 64K floating point filter coefficients; 32K floating point sine coefficients and 32K floating point cosine coefficients.

The ADSP-CM40x mixed-signal control processor contains up to 384 Kbytes of on-chip SRAM, which may not be sufficient to carry out these processing tasks. So, other memories such as the on-board SPI flash memory (up to 2 MBytes) and external SRAM memory connected to the Static Memory Controller (SMC) interface might be used [2].

Since the filter and FFT coefficients will remain constant and will only be used for read operations, these values can be stored in the internal SPI flash memory. The processor's on-board SPI flash space can be used as memory mapped for read operations. A separate sample code is provided to write these coefficients into SPI flash memory. Furthermore, the ADSP-CM40x EZ-KIT board also features a 512 KBytes external SRAM memory, which can be used for storing intermediate processing results.

Once the FFT has been completed, the power components in the frequency spectrum are processed to calculate the ADC performance parameters. For SNDR calculation, four bins around either side of the fundamental component are considered into the signal's power. And the noise power is calculated from the 10^{th} bin, as the ADC output will have a large DC component because of its 0-V_{REF} analog input range. For calculating the THD, the four harmonics of the input signal are considered. Once the SNDR and THD parameters are calculated, the SNR and ENOB parameters can be easily calculated.

ADC Performance Measurements

Hardware Platform

The ADC performance measurements were performed on the ADSP-CM408F EZ-KIT evaluation system platform. A precision audio test instrument, AP2722 low distortion sine wave generator[5], was used for providing the ADC analog input. The analog input from the AP2722 is a bipolar signal, which is converted to a unipolar signal using ADA4899-1 op-amp based level shifter circuit. The level shifter also boosts the driving capacity analog signal to drive the ADC input channels. This opamp circuit has, RC load of 220hm and 33nF at the output.

For testing purposes, a sine wave signal of amplitude 0 Volts to 2.50 Volts and frequency of 1064 Hz was generated. This analog input signal was connected to the `ADC1_VIN0` signal on the SMC analog connector (`J6`) available on the ADSP-CM408F EZ-KIT board[3]. The other ADC channel inputs are available on Analog Connector (`J9`).

The code can be configured to use internal ADC reference or an external ADC reference of 2.5V.

The board is connected to Host PC for running the application through IAR Embedded Workbench[4]. The code can be loaded into internal SPI flash to make the application standalone.

The overall system is as shown in Figure 7.

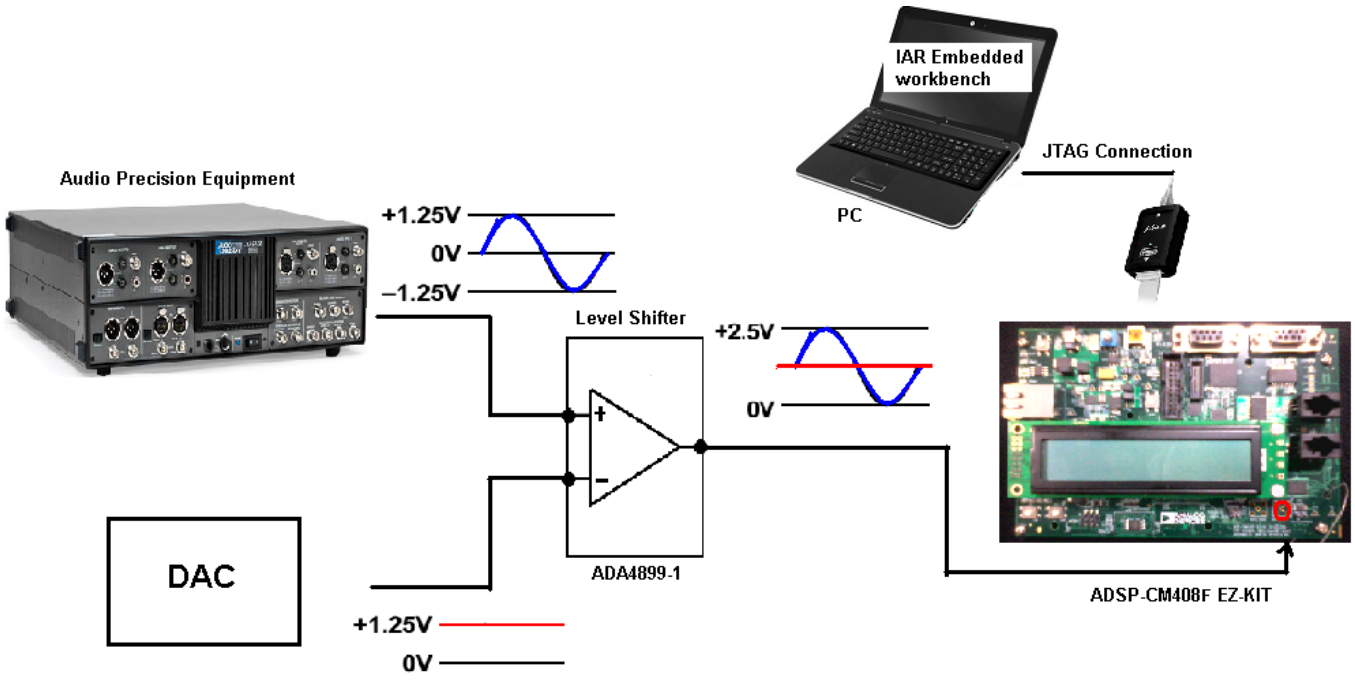

Figure 7. Test hardware setup

The ADC samples are collected through ADCC controller of processor and then are processed internally to calculate ADC performance parameters and are displayed on the character display as explained in Software Details section. An LED is blinked after each process.

Test Results

The test results obtained, using the example code provided in the associated `.ZIP` file, are displayed below (Figure 8):

Figure 8. ADC performance measurements results

References

[1] *ADSP-CM40x Mixed-Signal Control Processor with ARM Cortex-M4 Hardware Reference.* Preliminary Rev 0.2, September 2013. Analog Devices, Inc.

[2] *ADSP-CM40x ARM Cortex-M4 Mixed-Signal Control Processor Data Sheet.* Rev PrD, September 2013. Analog Devices, Inc.

[3] *ADSP-CM408F EZ-KIT Lite Evaluation System Manual.* Rev 1.1, September 2013. Analog Devices, Inc.

[4] *IAR Embedded Workbench for ARM (`http://www.iar.com/`). 6.50* IAR Systems AB.

[5] *High Performance 2700 Series Audio Analyzer.* Audio Precision Inc.

Document History

Revision	Description
Rev 1 – September 20, 2013 *by Prashant Gawade*	Initial release.