F2810, F2811, and F2812 ADC Calibration

Alex Tessarolo *C2000*

ABSTRACT

This application note describes a method for improving the absolute accuracy of the 12-bit analog-to-digital converter (ADC) found on the F2810/F2811/F2812 devices. Due to inherent gain and offset errors, the absolute accuracy of the ADC is impacted. The methods described in this application note can improve the absolute accuracy of the ADC to achieve levels better than 0.5%. This application note is accompanied by an example program (ADCcalibration.zip) that executes from RAM on the F2812 EzDSP. Project collateral discussed in this application report can be downloaded from the following URL: http://www.ti.com/lit/zip/SPRA989.

Contents

1 Gain and Offset Error Definition 2
2 Gain and Offset Error Impact 2
3 Calibration 4
4 Hardware Connectivity 6
5 ADC Sampling Techniques 8
5.1 Sequential Sampling mode 8
5.2 Simultaneous Sampling Mode 10
6 Example Software Calibration Driver 14
7 Useful Tips 16

List of Figures

1. Graph of Actual and Ideal Gain 2
2. Simplified Circuit for Ideal ADC Case 3
3. Circuit Modified to Allow for Different Characteristics 4
4. Graph of Equations to Measure Actual Gain and Offset 5
5. Channels for Calibration, 14 User Channels 7
6. Two Channels for Calibration, 16 User Channels 8
7. Timing of 16-Channel Conversion in Sequential Mode 9
8. Channel Selection Control Register Values to Program for Direct Mapping 10
9. Timing for Conversion of 8 Pairs of Channels 11
10. Values to Schedule Channel Pairs in Normal Order 13

List of Tables

1. Worst-Case Scenarios for Linear Input Range 3
2. Conversion Time for Conversion..... 9
3. Conversion Time to Convert All 16 Channels 12

1 Gain and Offset Error Definition

An ideal 12-bit ADC with no gain and offset error would be described by equation 1:

1. y = x * mi x = input count = input voltage * 4095/3.0 V
   y = output count
   mi = ideal gain = 1.0000

The F2810/12 ADC exhibits gain and offset error defined by equation 2:

2. y = x * ma + b ma = actual gain
   b = actual offset (relative to 0 input)

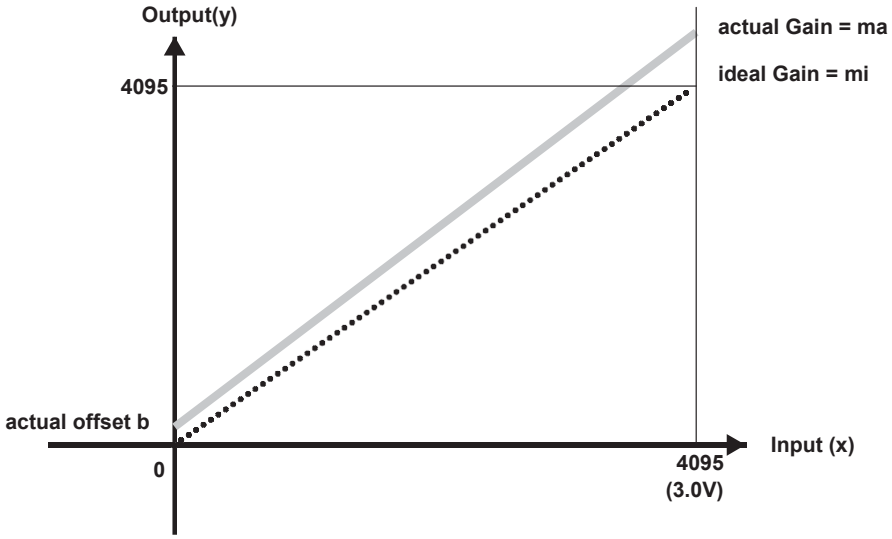


Figure 1. Graph of Actual and Ideal Gain

Gain and offset errors measured on the F2810/12 (Rev-E, TMS) ADC are:

Gain Error (ma):	< ± 5% max	0.95 < ma < 1.05
Offset Error (b):	< ± 2% max	− 80 < b < 80

NOTE: Most F2810/F2811/F2812 TMS devices exhibit gain errors of < ± 3.0% and offset errors of < ± 1%; however, some devices exhibit errors above these values, reaching the maximum values outlined above.

2 Gain and Offset Error Impact

Gain and offset errors contribute to errors in the control system. Understanding these errors will enable you to compensate for these errors in the design.

Linear Input Range: The available input voltage range is impacted by the gain and offset errors. The effective resolution is also reduced. See Table 1.

Table 1. Worst-Case Scenarios for Linear Input Range

	Linear Input Range (V)	Linear Output Range (Counts)	Input Swing (V)	Effective Number Of Bits	mV/ Count Resolution
y = x * 1.00	0.0000 to 3.0000	0 to 4095	1.5000 ± 1.5000	12.000	0.7326
y = x * 1.00 + 80	0.0000 to 2.9414	80 to 4095	1.4707 ± 1.4707	11.971	0.7326
y = x * 1.00 − 80	0.0586 to 3.0000	0 to 4015	1.5293 ± 1.4707	11.971	0.7326
y = x * 1.05 + 80	0.0000 to 2.8013	80 to 4095	1.4007 ± 1.4007	11.971	0.6977
y = x * 1.05 − 80	0.0558 to 2.9130	0 to 4095	1.4844 ± 1.4286	12.000	0.6977
y = x * 0.95 + 80	0.0000 to 3.0000	80 to 3970	1.5000 ± 1.5000	11.926	0.7710
y = x * 0.95−80	0.0617 to 3.0000	0 to 3810	1.5309 ± 1.4691	11.896	0.7710
Safe Range	0.0617 to 2.8013	80 to 3810	1.4315 ± 1.3698	11.865	0.7345

The last row in the table shows the safe parameters for operation of the devices. The effective number of bits of the ADC is only slightly reduced (11.865 bits). The mV/count is increased from 0.7326 to 0.7345, which is approximately 0.2% decrease in resolution.

Bipolar Offset Error: In many applications, the input sensor is a bipolar input and this must be converted to a unipolar signal before being fed to the ADC. Figure 2 shows a typical simplified circuit used for this purpose.

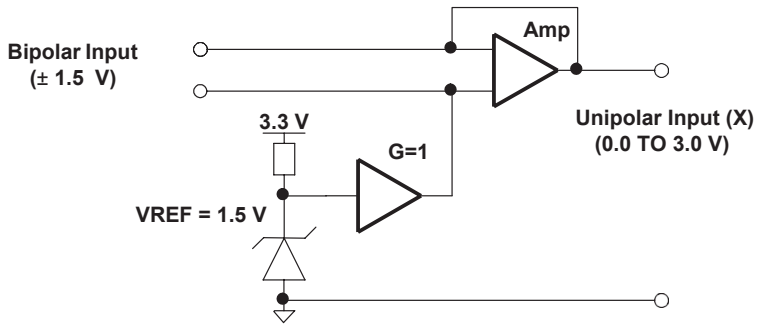


Figure 2. Simplified Circuit for Ideal ADC Case

Taking into account gain and offset errors and the impact on the input range, the circuit must be modified as shown in Figure 3 to allow for different device characteristics.

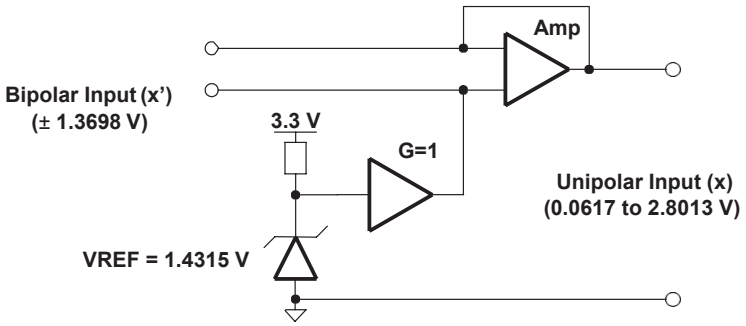


Figure 3. Circuit Modified to Allow for Different Characteristics

The frame of reference for the input offset error has changed. The offset error is measured relative to the bipolar input when the input value is zero (x' = 0). This corresponds to a unipolar input value of x = 1.4315 V. The inherent ADC gain and offset errors tend to magnify the error relative to the ideal value.

For example, if the ADC has a + 5% gain error and + 2% offset error, then the bipolar offset error count is:

Bipolar Input:	x' = 0.0000 V
Unipolar ADC Input:	x = 1.4315 V
Expected Count:	ye = 1.4315 * 4095/3 = 1954
Actual Count:	ya = 1954 * 1.05 + 80 = 2132
Bipolar Offset Error:	ya – ye = 2132 – 1954 = 178 counts (9.1% error)

This is a much higher error than a user would otherwise expect. This error can effectively be removed by calibration.

3 Calibration

Calibration is performed by feeding two known reference values into two ADC channels and calculating a calibration gain and offset to compensate for the input readings from the other channels. This is possible because the channel-to-channel errors are small. The achievable accuracy using calibration is largely dependent on the accuracy of the known references fed into the ADC. The best possible accuracy achievable is limited by the channel-to-channel gain and offset errors of the ADC.

NOTE: On the production devices, typical channel-to-channel gain and offset errors in the order of ± 0.2% have been observed.

Figure 4 shows how the equations to measure the ADC actual gain and offset and calculations of the calibration gain and offset are derived.

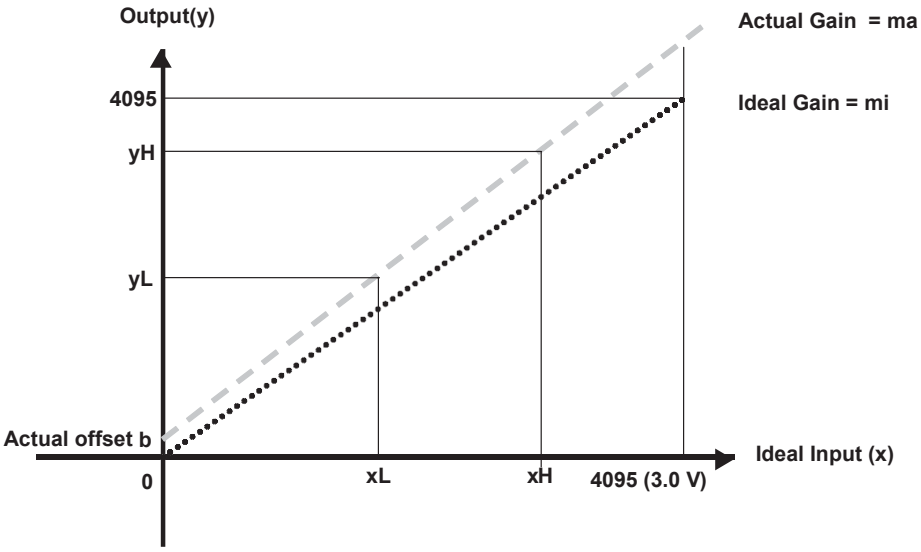


Figure 4. Graph of Equations to Measure Actual Gain and Offset

Using equation 2 again as follows:

$y = x * ma + b$

ma = actual gain
b = actual offset (relative to 0 input)

3. $ma = (yH – yL)/(xH – xL)$

xL = Known reference low input
xH = Known reference high input
yL = Reference low ADC output
yH = Reference high ADC output

4. $b = yL – xL*ma$

The calibration equation is derived by inverting the input and output of equation 2 describing the ADC actual gain and offset:

$y = x * ma + b$

$x = (y – b)/ma$

$x = y/ma – b/ma$

5. $x = y * CalGain - CalOffset$

CalGain = 1/ma
CalOffset = b/ma

6. $CalGain = (xH – xL)/(yH – yL)$
   $CalOffset = (yL – xL*ma)/ma$
   $CalOffset = yL/ma – xL$

7. $CalOffset = yL * CalGain - xL$

In summary, using two known references (xL, yL) and (xH, yH) you can calculate the actual offset and gain error and calculate the calibration gain and offset using the following formulas:

1. $y = x * ma + b$ ADC actual equation
2. $ma = (yH – yL)/(xH – xL)$ ADC actual gain

3. b = yL – xL*ma ADC actual offset
4. x = y * CalGain – CalOffset ADC calibration equation
5. CalGain = (xH – xL)/(yH – yL) ADC calibration gain
6. CalOffset = yL * CalGain – xL ADC calibration offset

The calibration process involves the following four basic steps:

Step 1: Read the known reference values input channels (yL and yH).

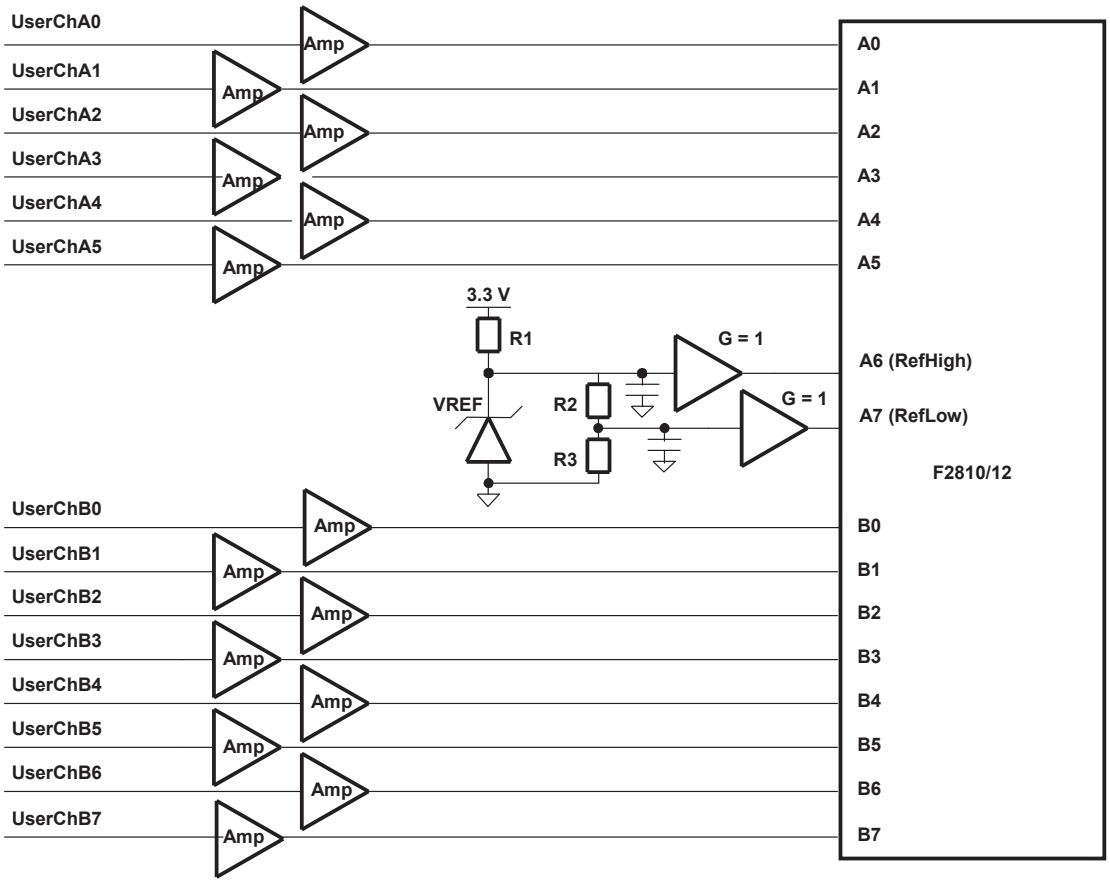
Step 2: Calculate the calibration gain (CalGain) using equation 6.

Step 3: Calculate the calibration offset (CalOffset) using equation 7.

Step 4: Cycle through all channels applying the calibration equation 5.

4 Hardware Connectivity

Calibration requires that two ADC channels be dedicated to supply two known reference inputs and this leaves 14 user channels. Figure 5 shows the recommended connection.

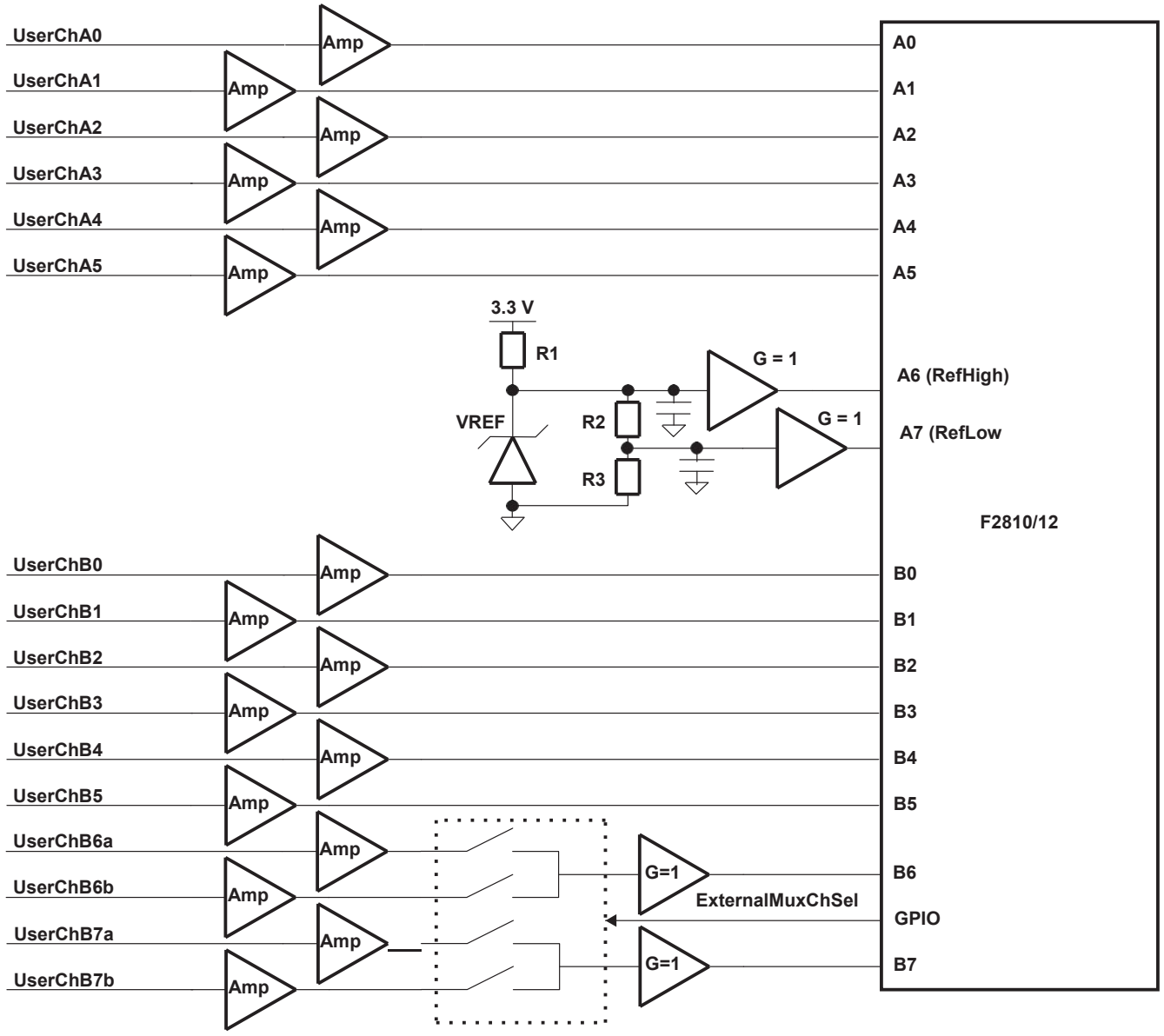


NOTE: The typical ch-to-ch error is approximately 0.2 % across all channels. Therefore the choice of channels for calibration is user selectable, but it is preferable to have them be in the same group. At lower sampling frequencies (< ~8 MHz) the ch-to-ch error can approach 0.1 % within a group. In such cases, improved accuracy can be achieved by connecting the calibration references to each group. Each group should be calibrated from its own reference channel readings.

If converting a bipolar input to unipolar, then a mid-point reference (~1.5 V) would be a good choice as one reference input. The other reference input could either be a higher value (~2.5 V) or lower value (~0.5 V) reference.

Figure 5. Channels for Calibration, 14 User Channels

To retain 16 user channels, use the system shown in Figure 6. In this scenario, an external analog switch is added that expands the user channels to 16 and is controlled by a GPIO pin using software. In a simple software implementation, the multiplexed channels are sampled on every alternate cycle, relative to the nonmultiplexed channels. This means that the multiplexed channels should be used for slower, supervisory-type functions. This system leaves you with 6 channel pairs (if using simultaneous sampling mode) that can be used for critical functions.



NOTE: A buffer is required at the output of the MUX (or any high resistance source) to prevent errors due to high source impedance when sampling of the ADC channels.

Figure 6. Two Channels for Calibration, 16 User Channels

5 ADC Sampling Techniques

5.1 Sequential Sampling mode

THe ADC converter on an F2810/F2811/F2812 device can operate in sequential sampling mode or simultaneous sampling mode. In sequential sampling mode, ADC samples one channel at a time and then the sampled signal is passed through four stages of ADC pipeline for conversion.

Figure 7 shows the timing details of 16-channel conversion (A0−A7 & B0−B7) in sequential mode based on an event trigger.

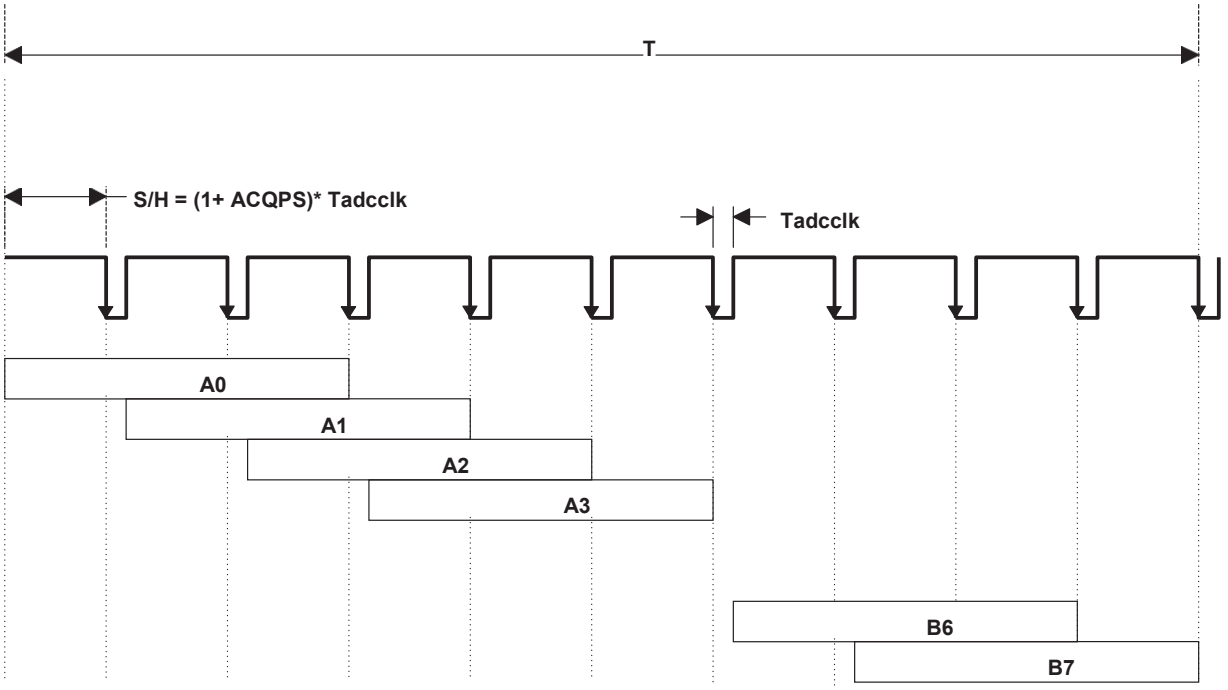


Figure 7. Timing of 16-Channel Conversion in Sequential Mode

Total Time for converting 16 channels in sequential mode:

$$T = 17*Tadcclk + 18*(1 + ACQPS)*Tadcclk$$

Table 2 shows the conversion time required to convert all 16 channels in sequential sampling mode under different ADC clock frequencies and sample/hold windows.

Table 2. Conversion Time for Conversion

ACQPS	Number Of TADCCLK Periods	T in µs (ADCCLK = 25 Mhz)	T in µs (ADCCLK = 12.5 Mhz)	T in µs (ADCCLK = 6.25 Mhz)
0	35	1.4	2.8	5.6
3	89	3.56	7.12	14.24
7	161	6.44	12.88	25.76
11	233	9.32	18.64	37.28
15	305	12.2	24.4	48.8

NOTE: ACQSPS is the acquisition window width. Value of 0 is equal to one ADCCLK period.

In sequential sampling mode, ADC can be configured in cascade mode or dual sequencer mode. In cascade mode, you can schedule 16 ADC conversions sequentially based on an even trigger. The order in which the channels are converted and stored in the result register is controlled by the ADC Channel Selection Control Register (CHSELSEQ1, CHSELSEQ2, CHSELSEQ3, CHSELSEQ4). For direct mapping of channels to the corresponding result register, you need to program the channel selection control register to the values (see Figure 8).

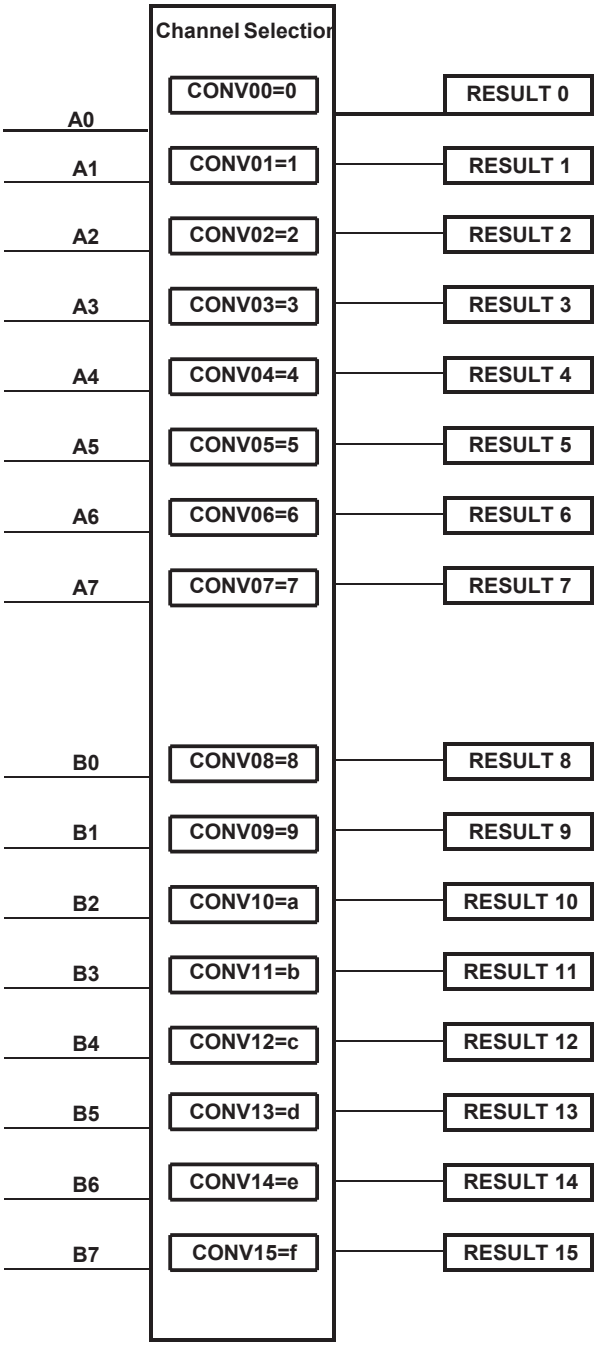


Figure 8. Channel Selection Control Register Values to Program for Direct Mapping

5.2 Simultaneous Sampling Mode

In simultaneous sampling mode, the ADC can convert input signals on any pair of channels (A0/B0 to A7/B7). Basically, two channels are sampled simultaneously and passed through four stages of the ADC pipeline for conversion. Figure 9 shows the timing details of 8 pairs of channels (A0/B0 to A7/B7) converted in simultaneous sampling mode based on an event trigger.

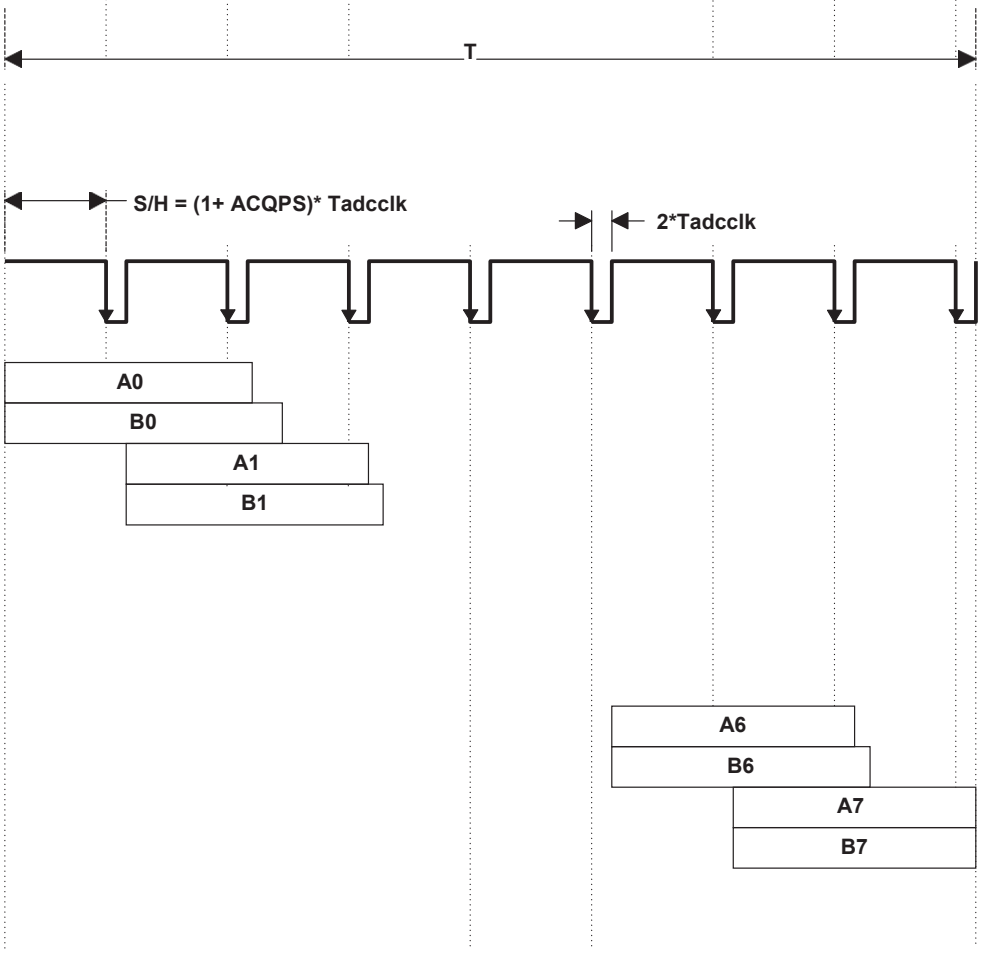


Figure 9. Timing for Conversion of 8 Pairs of Channels

Total Time for converting 16 channels in simultaneous mode:

$$T = 9*2*Tadcclk + 9*(1+ACQPS)*Tadcclk$$

Table 3 shows the conversion time required to convert all 16 channels in simultaneous sampling mode under different ADC clock frequency and S/H window.

Table 3. Conversion Time to Convert All 16 Channels

ACQPS	Number Of Tadcclk Periods	T in µs (ADCCLK = 25 Mhz)	T in µs (ADCCLK = 12.5 Mhz)	T in µs (ADCCLK = 6.25 Mhz)
0	27	1.08	2.16	4.32
3	54	2.16	4.32	8.64
7	90	3.6	7.2	14.4
11	126	5.04	10.08	20.16
15	162	6.48	12.96	25.92

In simultaneous sampling mode, you can schedule 8 pairs of simultaneous A/D conversions based on an even trigger (A0/B0, A1/B1...A7/B7). The order in which the channel pairs are converted and stored in the result register is controlled by the ADC Channel Selection Control Register (CHSELSEQ1, CHSELSEQ2). To schedule the channel pairs in normal order A0/B0, A1/B1,... A7/B7, the channel selection register must be programmed to the values in Figure 10.

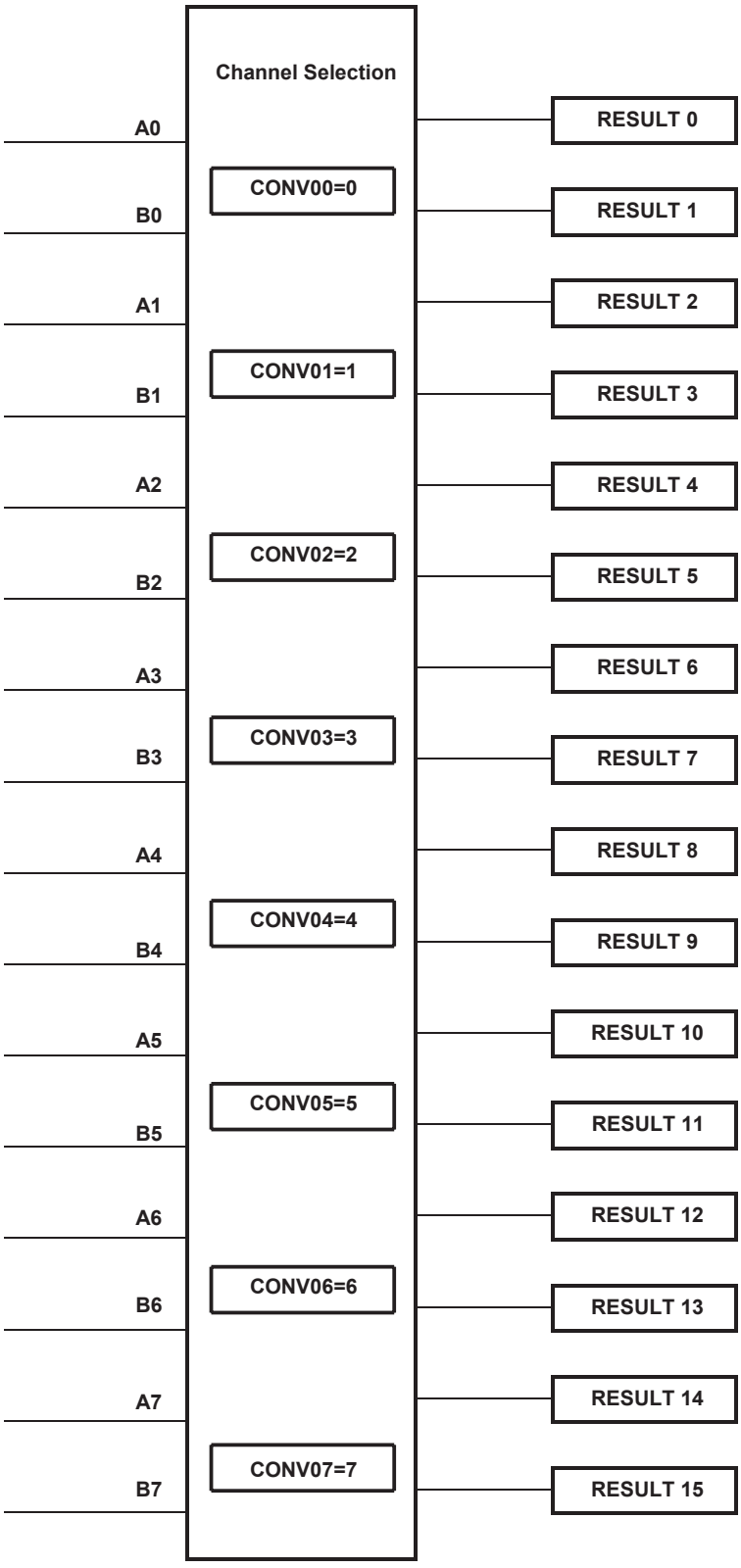


Figure 10. Values to Schedule Channel Pairs in Normal Order

6 Example Software Calibration Driver

This application note comes with an example C program for simultaneous sampling mode and sequential sampling mode that executes from RAM on the F2812 EzDSP. The example program configures the Event Manager to generate a periodic start of conversion pulse to the ADC. The ADC is configured to process all 16- input channels and then generate an interrupt. In the interrupt service routine, a call is made to an optimized assembly driver that will read the user-selected ADC channel references, calculate the calibration gain and offset and calibrate all other channels and store the information in a RAM structure.

The program supports configuring the ADC for simultaneous or sequential conversion modes:

SEQUENTIAL:	ADC channels are converted one at a time: A0 → A1 → A2 → ... B0 → B1 → B2 →
SIMULTANEOUS:	ADC channels are converted in pairs: A0,B0 → A1,B1 → A2,B2 →

The calibrated and converted channels are stored in a RAM structure that contains the following information:

```
typedef struct {
    Uint16  *RefHighChAddr;          // Channel Address of RefHigh
    Uint16  *RefLowChAddr;           // Channel Address of RefLow
    Uint16  *Ch0Addr;                // Channel 0 Address
    Uint16  Avg_RefHighActualCount;  // Ideal RefHigh Count (Q4)
    Uint16  Avg_RefLowActualCount;   // Ideal RefLow  Count (Q4)
    Uint16  RefHighIdealCount;       // Ideal RefHigh Count (Q0)
    Uint16  RefLowIdealCount;        // Ideal RefLow  Count (Q0)
    Uint16  CalGain;                 // Calibration Gain   (Q12)
    Uint16  CalOffset;               // Calibration Offset (Q0)
                                     // Store Calibrated ADC Data (Q0):
                                     // Simultaneous   Sequential
                                     // ============  ============
    Uint16  ch0;                     //      A0            A0
    Uint16  ch1;                     //      B0            A1
    Uint16  ch2;                     //      A1            A2
    Uint16  ch3;                     //      B1            A3
    Uint16  ch4;                     //      A2            A4
    Uint16  ch5;                     //      B2            A5
    Uint16  ch6;                     //      A3            A6
    Uint16  ch7;                     //      B3            A7
    Uint16  ch8;                     //      A4            B0
    Uint16  ch9;                     //      B4            B1
    Uint16  ch10;                    //      A5            B2
    Uint16  ch11;                    //      B5            B3
    Uint16  ch12;                    //      A6            B4
    Uint16  ch13;                    //      B6            B5
    Uint16  ch14;                    //      A7            B6
    Uint16  ch15;                    //      B7            B7
    Uint16   StatusExtMux;           // Indicates Status Of External MUX For
                                     // Current Conversion
}ADC CALIBRATION DRIVER VARS;
```

This program also supports your toggling a GPIO pin for toggling an external analog MUX to expand the usable channels.

To adapt to your system needs, you must configure assembly time switches and settings contained in the header file:

ADCcalibrationDriver.h

You must select simultaneous or sequential sampling mode of operation. For example:

```
#define     SEQUENTIAL            1
#define     SIMULTANEOUS          0
#define     ADC_SAMPLING_MODE     SIMULTANEOUS
```

You must also select which ADC channels are connected to reference high and reference low and the ideal count value. For example:

A6 = RefHigh = 2.5 V (2.5*4095/3.0 = 3413 ideal count)

A7 = RefLow = 1.25 V (1.25*4095/3.0 = 1707 ideal count)

```
#define     REF_HIGH_CH           A6
#define     REF_LOW_CH            A7
#define     REF_HIGH_IDEAL_COUNT  3413
#define     REF_LOW_IDEAL_COUNT   1707
```

The number of cycles required to execute the calibration driver is:

137cycles or 0.91 μS @150 MHz (9.7 cycles per user channel)

Without calibration, the driver would take approximately 4.7 cycles per user channel to read and store the ADC input. The calibration overhead is therefore approximately an additional 5 cycles per channel.

NOTE: In the example C program, the gain and error calculations are performed on every ADC interrupt. If you want to reduce the overhead, these calculations can be performed in the background at regular intervals. It was done this way in the example program for easier encapsulation of the example.

Only the running weighted average of the reference inputs should be retained in the interrupt routine. The calculation of CalGain and CalOffset can be pushed to the background. This can save up to 20 cycles per interrupt.

If further performance optimization is required, the calibration driver can be in-lined rather than making it a C function call.

7 Useful Tips

This is a collection of tips or options that you should consider to improve accuracy:

- Provide a low resistance path for ADCLO pin. Always make sure that the ADCLO pin on the F2810/F2811/F2812 device is connected directly to analog ground. Any resistance on this pin further degrades offset and gain errors.
- Use bipolar input conversion reference as calibration input. When converting bipolar to unipolar signals, make sure that the reference voltage used as the mid-point of the ADC range is fed as an input calibration channel. This removes any bipolar offset error as discussed earlier.
- Reduce ADCCLK frequency to minimum. At higher sampling frequencies (above 10 MHz), the channel-to-channel errors begin to increase. To improve accuracy, you should try to use the lowest frequency that your control system will tolerate. Increasing the sampling window will not have much of an effect on this error at high frequencies.
- Digital and analog grounds connected at one point. To avoid noise created by digital current loops, connect the digital and analog grounds at one point, making sure that any analog or digital current loops do not cross through this point. This is common practice when mixing digital and analog on a single board/device.

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265