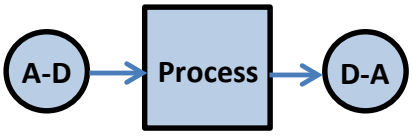
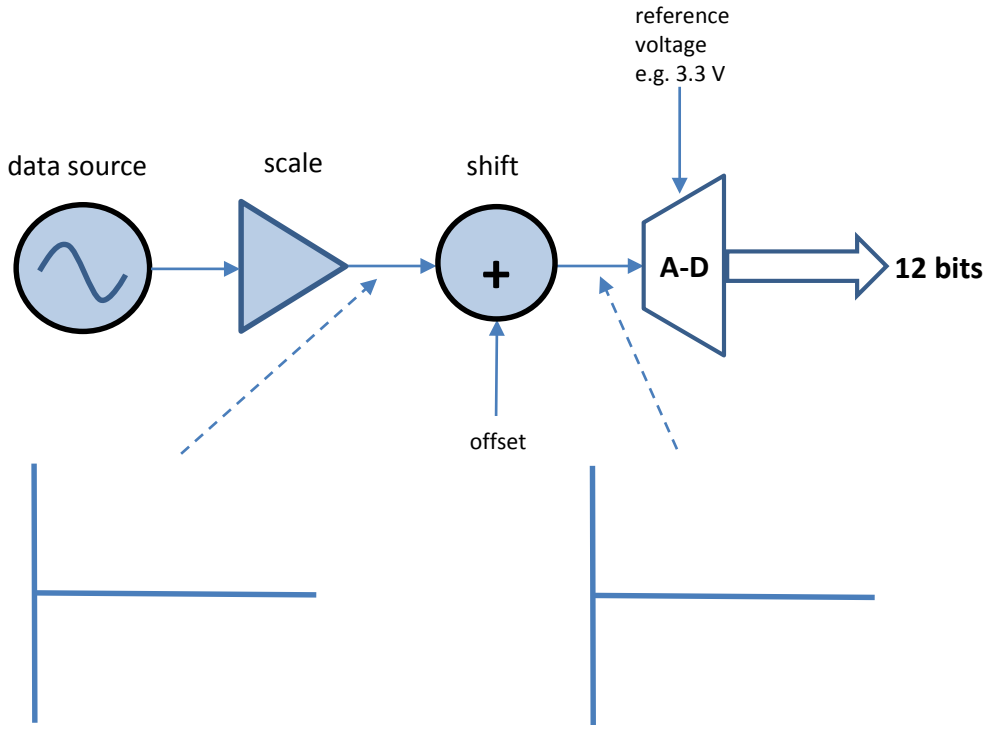
A-D Conversion

Recall ubiquitous part of embedded system:



Review of Basic A-D Signal Flow



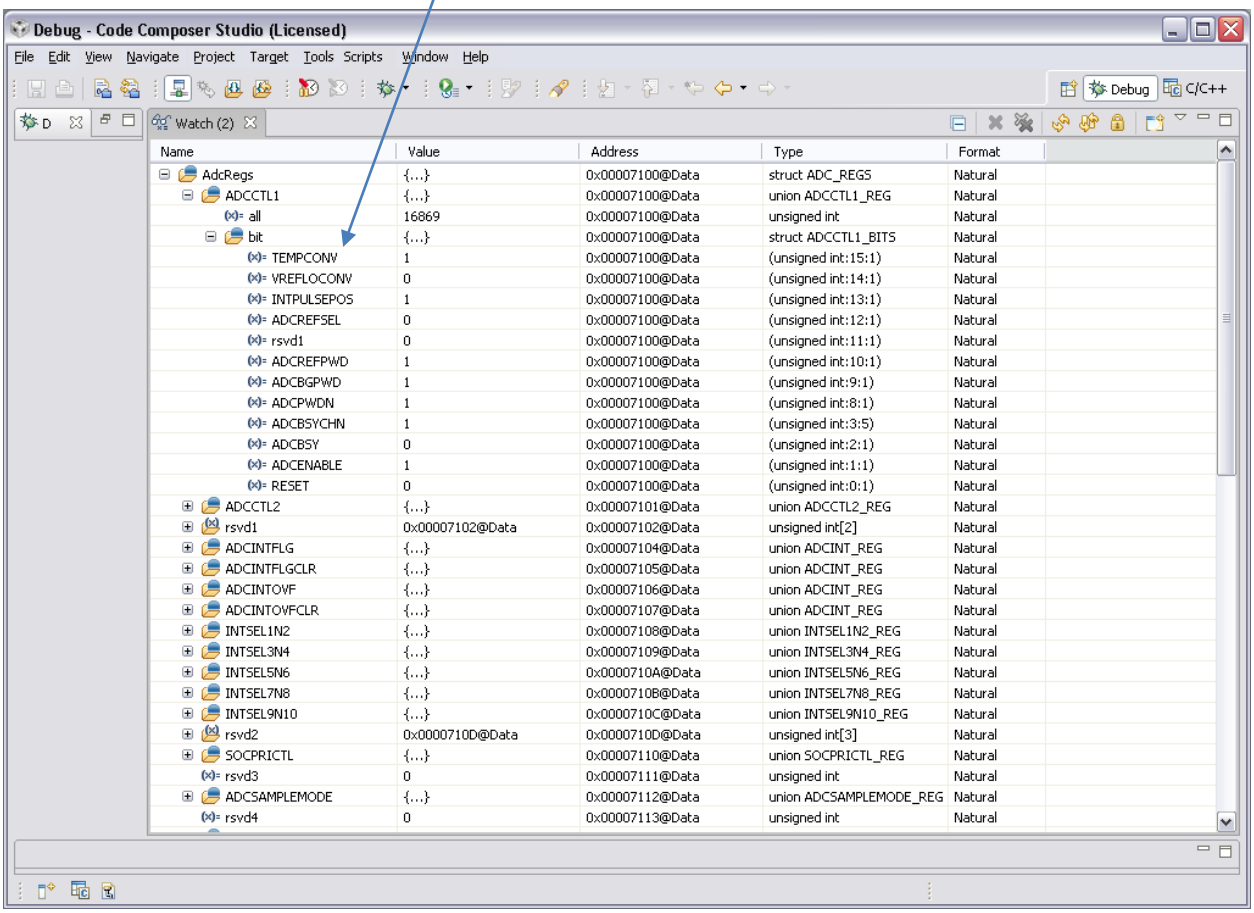
V_{in} (V)	12-Bits (Binary)	Decimal (Decimal)	MSB Bit Flipped

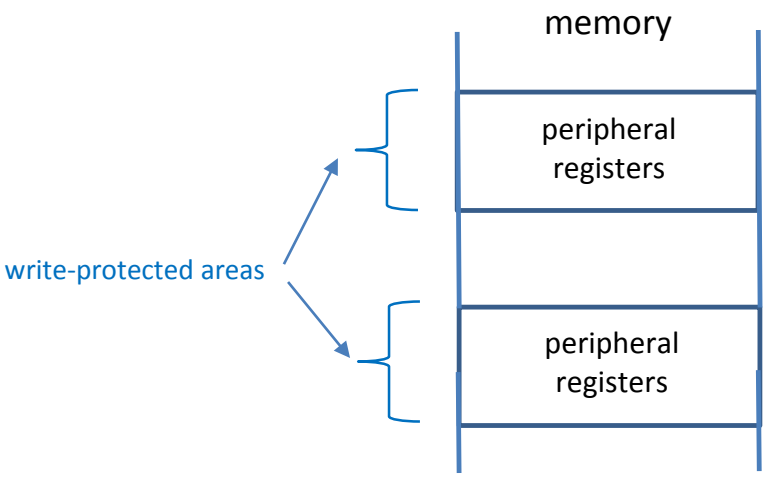
A-D Converter in c2000

Similar to other peripherals, the A-D converter on the TMS320F28027 is register-based and the registers are memory-mapped in that they appear in addressable memory locations.

The registers can be accessed from C code via the "structure" hierarchy as defined in the header file "DSP2802xAdc.h".

e.g. AdcRegs.ADCCTL1.bit.TEMPCONV





To allow access to write-protected areas:

C code:

```
. . .
EALLOW; //(this is a macro)
modify register values
EDIS; //(this is a macro)
. . .
```

Assembly instructions:

```
asm(" EALLOW");
. . .
asm(" EDIS");
```

The basic approach to using the A-D converter in your code is to:

1. Power up the A-D by setting various bits in the control registers:
    a. ADCENCLK enable A-D clock
    b. ADCPWDN turn on A-D power
    c. ADCBGPWD turn on A-D bandgap
    d. ADCREFPWD turn on voltage reference buffer

2. Initialize the A-D by setting various bits in the control registers *for each "SOC"*.
    a. CHSEL[3:0] which analog channel to convert
    b. ACQPS[5:0] what acquisition window size to use
    c. TRIGSEL[4:0] what source to use to trigger conversion
    d. configure any interrupt(s) to be used

3. Start conversion (either force by s/w or enable interrupt)

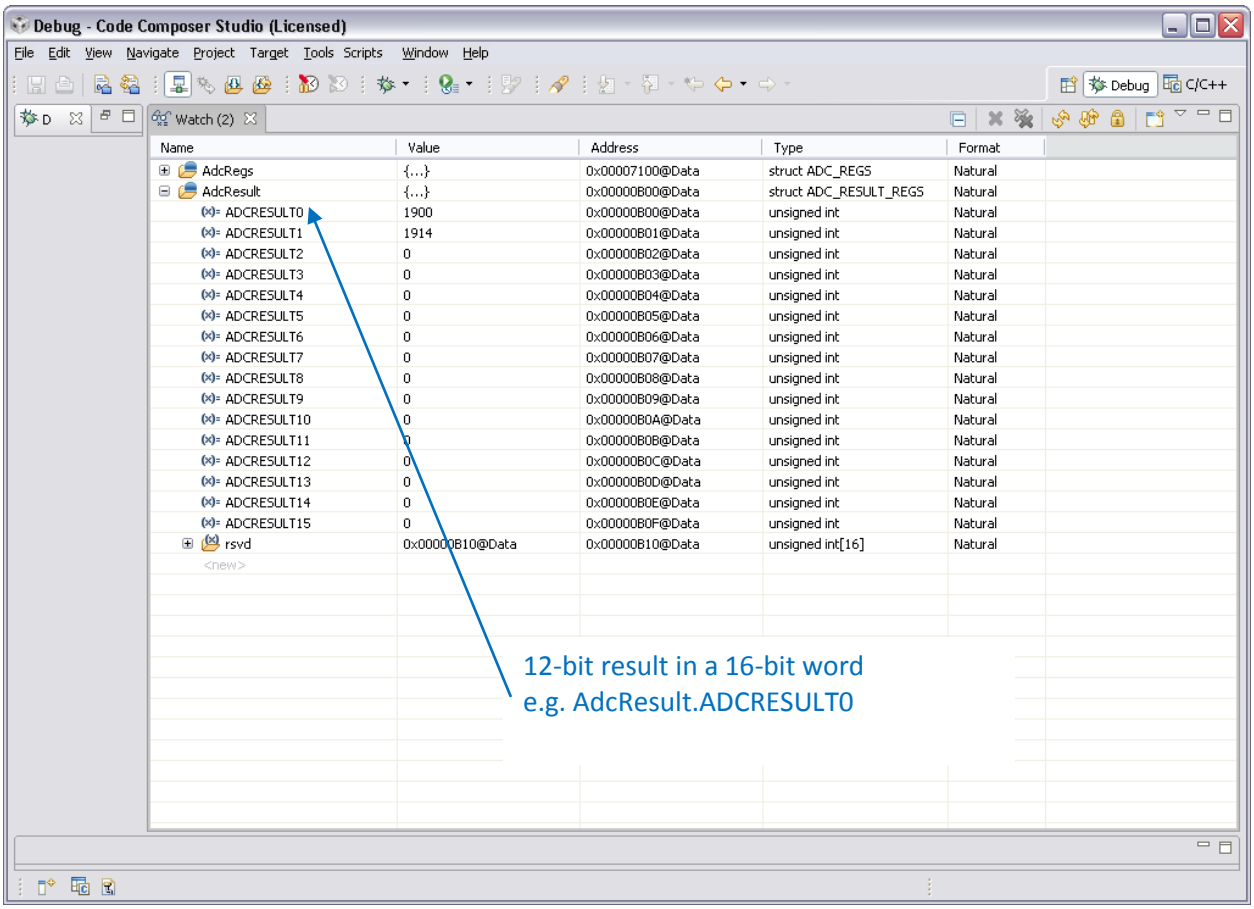
4. Read result from AdcResult.ADCRESULTx

A nice thing is that most of the default settings can be left as is and we only need to modify relatively few values.

Refer to:

"spruge5f (or newer) - TMS320x2802x, 2803x Piccolo Analog-to-Digital Converter (ADC) and Comparator Reference Guide.pdf"

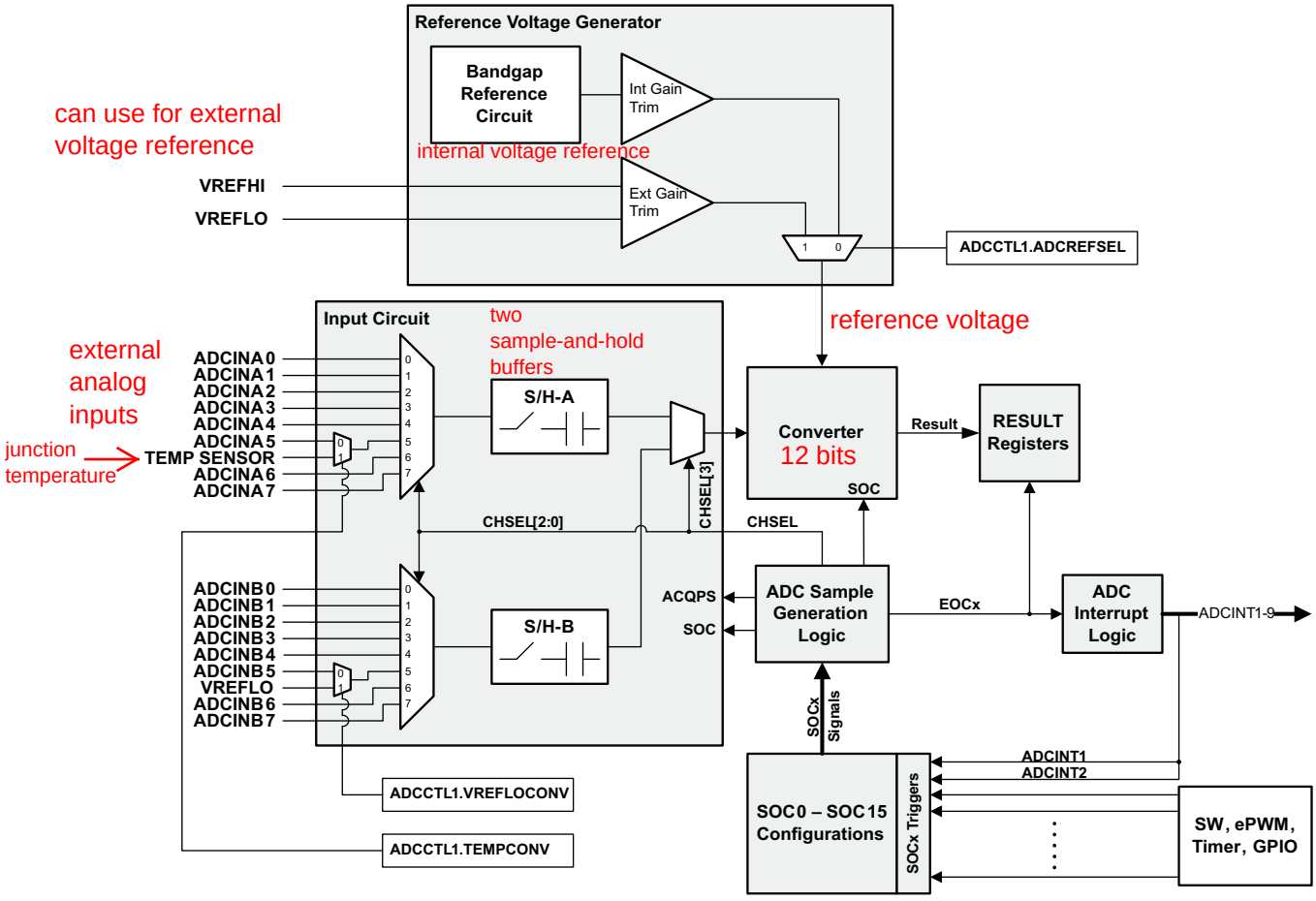
for more information.


Debug - Code Composer Studio (Licensed)
File Edit View Navigate Project Target Tools Scripts Window Help
Debug C/C++
Watch (2)
Name Value Address Type Format
AdcRegs {...} 0x00007100@Data struct ADC_REGS Natural
AdcResult {...} 0x00000B00@Data struct ADC_RESULT_REGS Natural
ADCRESULT0 1900 0x00000B00@Data unsigned int Natural
ADCRESULT1 1914 0x00000B01@Data unsigned int Natural
ADCRESULT2 0 0x00000B02@Data unsigned int Natural
ADCRESULT3 0 0x00000B03@Data unsigned int Natural
ADCRESULT4 0 0x00000B04@Data unsigned int Natural
ADCRESULT5 0 0x00000B05@Data unsigned int Natural
ADCRESULT6 0 0x00000B06@Data unsigned int Natural
ADCRESULT7 0 0x00000B07@Data unsigned int Natural
ADCRESULT8 0 0x00000B08@Data unsigned int Natural
ADCRESULT9 0 0x00000B09@Data unsigned int Natural
ADCRESULT10 0 0x00000B0A@Data unsigned int Natural
ADCRESULT11 0 0x00000B0B@Data unsigned int Natural
ADCRESULT12 0 0x00000B0C@Data unsigned int Natural
ADCRESULT13 0 0x00000B0D@Data unsigned int Natural
ADCRESULT14 0 0x00000B0E@Data unsigned int Natural
ADCRESULT15 0 0x00000B0F@Data unsigned int Natural
rsvd 0x00000B10@Data 0x00000B10@Data unsigned int[16] Natural
<new>
12-bit result in a 16-bit word
e.g. AdcResult.ADCRESULT0

1.2 Block Diagram

Figure 1 shows the block diagram of the ADC module.

Figure 1. ADC Block Diagram

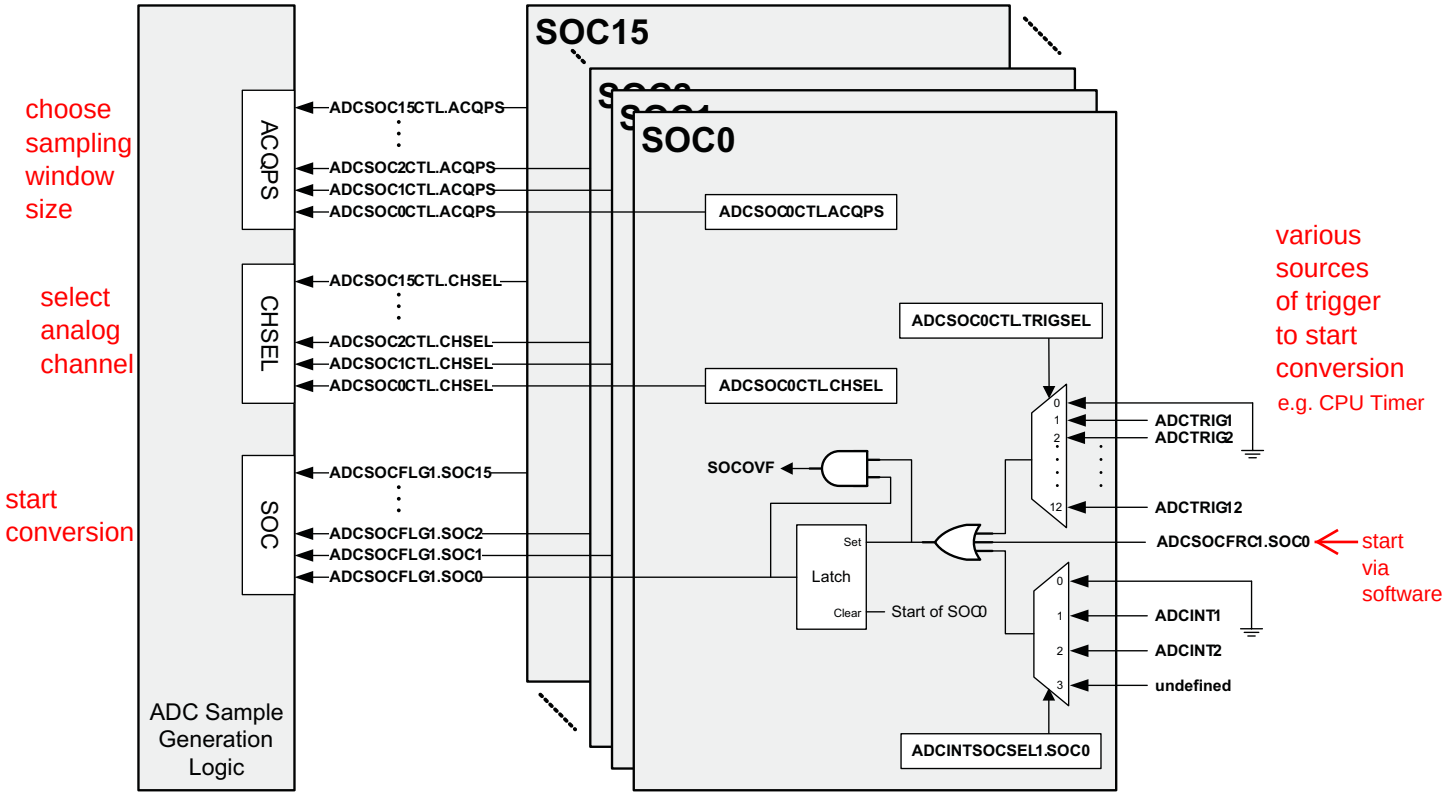


1.3 SOC Principle of Operation

Contrary to previous ADC types, this ADC is not sequencer based. Instead, it is SOC based. The term SOC is configuration set defining the single conversion of a single channel. In that set there are three configurations: the trigger source that starts the conversion, the channel to convert, and the acquisition (sample) window size. Each SOC is independently configured and can have any combination of the trigger, channel, and sample window size available. Multiple SOC's can be configured for the same trigger, channel, and/or acquisition window as desired. This provides a very flexible means of configurating conversions ranging from individual samples of different channels with different triggers, to oversampling the same channel using a single trigger, to creating your own series of conversions of different channels all from a single trigger.

The trigger source for SOCx is configured by a combination of the TRIGSEL field in the ADCSOCxCTL register and the appropriate bits in the ADCINTSOCSEL1 or ADCINTSOCSEL2 register. Software can also force an SOC event with the ADCSOCFRC1 register. The channel and sample window size for SOCx are configured with the CHSEL and ACQPS fields of the ADCSOCxCTL register.

Figure 2. SOC Block Diagram



For example, to configure a single conversion on channel ADCINA1 to occur when the ePWM3 timer reaches its period match you must first setup ePWM3 to output an SOCA or SOCB signal on a period match. See the TMS320x2802x Piccolo Enhanced Pulse Width Modulator Module User's Guide (SPRUGE9) on how to do this. In this case, we'll use SOCA. Then, setup one of the SOC's using its ADCSOCxCTL register. It makes no difference which SOC we choose, so we'll use SOC0. The fastest allowable sample window for the ADC is 7 cycles. Choosing the fastest time for the sample window, channel ADCINA1 for the channel to convert, and ePWM3 for the SOC0 trigger, we'll set the ACQPS field to 6, the CHSEL field to 1, and the TRIGSEL field to 9, respectively. The resulting value written into the register will be:

```
ADCSOC0CTL = 4846h;          // (ACQPS=6, CHSEL=1, TRIGSEL=9)
```

When configured as such, a single conversion of ADCINA1 will be started on an ePWM3 SOCA event with the resulting value stored in the ADCRESULT0 register.

If instead ADCINA1 needed to be oversampled by 3X, then SOC1, SOC2, and SOC3 could all be given the same configuration as SOC0.

```
ADCSOC1CTL = 4846h;          // (ACQPS=6, CHSEL=1, TRIGSEL=9)
ADCSOC2CTL = 4846h;          // (ACQPS=6, CHSEL=1, TRIGSEL=9)
ADCSOC3CTL = 4846h;          // (ACQPS=6, CHSEL=1, TRIGSEL=9)
```

When configured as such, four conversions of ADCINA1 will be started in series on an ePWM3 SOCA event with the resulting values stored in the ADCRESULT0 – ADCRESULT3 registers.

Another application may require 3 different signals to be sampled from the same trigger. This can be done by simply changing the CHSEL field for SOC0-SOC2 while leaving the TRIGSEL field unchanged.

```
ADCSOC0CTL = 4846h;   // (ACQPS=6, CHSEL=1, TRIGSEL=9)
ADCSOC1CTL = 4886h;   // (ACQPS=6, CHSEL=2, TRIGSEL=9)
ADCSOC2CTL = 48C6h;   // (ACQPS=6, CHSEL=3, TRIGSEL=9)
```

When configured this way, three conversions will be started in series on an ePWM3 SOCA event. The result of the conversion on channel ADCINA1 will show up in ADCRESULT0. The result of the conversion on channel ADCINA2 will show up in ADCRESULT1. The result of the conversion on channel ADCINA3 will show up in ADCRESULT2. The channel converted and the trigger have no bearing on where the result of the conversion shows up. The RESULT register is associated with the SOC.

NOTE: These examples are incomplete. Clocks must be enabled via the PCLKCR0 register and the ADC must be powered to work correctly. For a description of the PCLKCR0 register see the *TMS320F2802x Piccolo System Control and Interrupts Reference Guide* (SPRUFN3). For the power up sequence of the ADC, see Section 1.7. The CLKDIV2EN bit in the ADCCTL2 register must also be set to a proper value to obtain correct frequency of operation. For more information on the ADCCTL2 register please refer to Section 1.3.1.

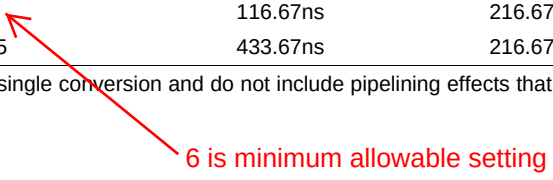
1.3.1 ADC Acquisition (Sample and Hold) Window

External drivers vary in their ability to drive an analog signal quickly and effectively. Some circuits require longer times to properly transfer the charge into the sampling capacitor of an ADC. To address this, the ADC supports control over the sample window length for each individual SOC configuration. Each ADCSOCxCTL register has a 6-bit field, ACQPS, that determines the sample and hold (S/H) window size. The value written to this field is one less than the number of cycles desired for the sampling window for that SOC. Thus, a value of 15 in this field will give 16 clock cycles of sample time. The minimum number of sample cycles allowed is 7 (ACQPS=6). The total sampling time is found by adding the sample window size to the conversion time of the ADC, 13 ADC clocks. Examples of various sample times are shown below in Table 1.

Table 1. Sample timings with different values of ACQPS

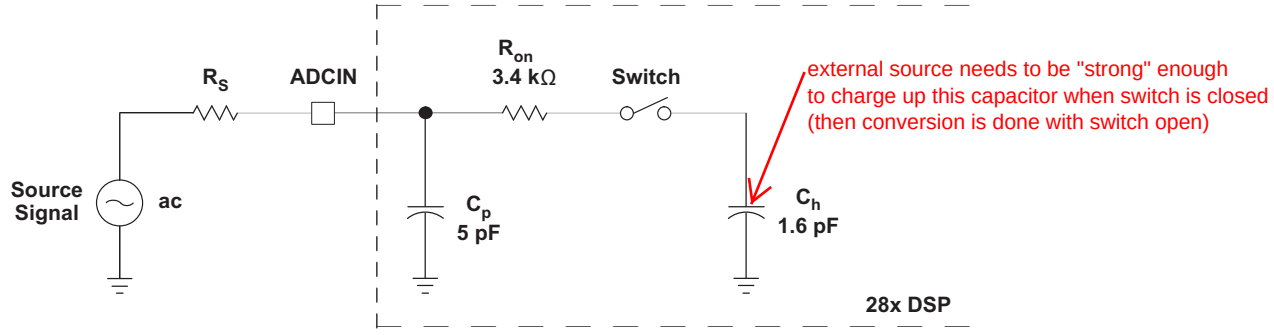
ADC Clock	ACQPS	Sample Window	Conversion Time (13 cycles)	Total Time to Process Analog Voltage (1)
40MHz	6	175 ns	325ns	500.00ns
40MHz	25	625 ns	325ns	950.00ns
60MHz	6	116.67ns	216.67ns	333.33ns
60MHz	25	433.67ns	216.67ns	650ns

(1) The total times are for a single conversion and do not include pipelining effects that increase the average speed over time.



As shown in Figure 3 , the ADCIN pins can be modeled as an RC circuit. With VREFLO connected to ground, a voltage swing from 0 to 3.3v on ADCIN yields a typical RC time constant of 2ns.

Figure 3. ADCINx Input Model



Typical Values of the Input Circuit Components:

Switch Resistance (R_{on}): 3.4 kΩ
Sampling Capacitor (C_h): 1.6 pF
Parasitic Capacitance (C_p): 5 pF
Source Resistance (R_S): 50 Ω

"strong" = means to provide enough current in a timely fashion, i.e. much quicker than the sampling period

1.3.2 Trigger Operation

Each SOC can be configured to start on one of many input triggers. Multiple SOC's can be configured for the same channel if desired. Following is a list of the available input triggers:

- Software i.e., via your code
- CPU Timers 0/1/2 interrupts via timeout
- XINT2 SOC external source
- ePWM1-8 SOCA and SOCB PWM generators

See the ADCSOCxCTL Register Bit Definitions for the configuration details of these triggers.

Additionally ADCINT1 and ADCINT2 can be fed back to trigger another conversion. This configuration is controlled in the ADCINTSOCSEL1/2 registers. This mode is useful if a continuous stream of conversions is desired. See section 1.6 for information on the ADC interrupt signals.

1.3.3 Channel Selection

Each SOC can be configured to convert any of the available ADCIN input channels. When an SOC is configured for sequential sampling mode, the four bit CHSEL field of the ADCSOCxCTL register defines which channel to convert. When an SOC is configured for simultaneous sampling mode, the most significant bit of the CHSEL field is dropped and the lower three bits determine which pair of channels are converted.

ADCINA0 is shared with VREFHI, and therefore cannot be used as a variable input source when using external reference voltage mode. See Section 1.9 for details on this mode.

1.3.4 ONESHOT Single Conversion Support

This mode will allow you to perform a single conversion on the next triggered SOC in the round robin scheme. The ONESHOT mode is only valid for channels present in the round robin wheel. Channels which are not configured for triggered SOC in the round robin scheme will get priority based on contents of the SOCPRIORITY field in the ADCSOCPRIORITYCTL register.