**MSP430’s 12-bit Analog-to-Digital Converter (ADC12)**

**by Alex Milenkovich, milenkovic@computer.org**

The MSP430’s ADC12 peripheral is a highly sophisticated analog-to-digital converter. It is based around a 12-bit successive-approximation core with configurable inputs, reference voltages, clocks, sampling time, and triggering sources. Note: this type of ADC converters is often referred to as SAR contracted from Successive-Approximation Register. The ADC core converts an analog input to its 12-bit digital representation and stores the result in a conversion memory.

Let us first give some basic definitions related to ADC conversion. We assume an input analog signal with the minimum of 0 V and the maximum of 3V. We say that the full scale input voltage is VFS=Vmax – Vmin. Let us assume a 12-bit AD converter. Thus, the converter can represent 212 distinct binary values, ranging from 0x0000 to 0x0FFF (0 to 4095 in decimal). A change in the input of 3V/4096 = 0.7 mV corresponds to a change of 1 bit in the output binary value. This defines resolution of the ADC and we could say that ADC12 converts its input to a precision of 0.7 mV. We could also say that 1 LSB (Least Significant Bit) = 0.7 mV. We would like that the digital output follows the changes on the input as closely as possible. If we have 0 V at the input, the output should be 0x000. When the input voltage reaches ½ LSB, we would like to see 0x001 at the output (the input value is closer to the 1 LSB than to 0). The output should remain at 0x001 as long as the input voltage is between 1/2 LSB and 3/2 LSB. The output should be 0x002 if the input voltage is between 3/2 LSB and 5/2 LSB, etc. Finally, the maximum digital output 0x0FFF is when the input voltage belongs to the range between (Vmax – 3/2 LSB) and Vmax. Please note that the range of voltage for the 0x0000 output is 1/2 LSB (from 0 to 1/2 LSB), and the range for the maximum output (0x0FFF is 3/2 LSB). Consequently, we can write a transfer function for the ADC as NADC = nint(2NVin/VFS), where *nint* gives the nearest integer to the argument, Vin is the input voltage, and VFS is the full scale voltage at the input, and N denotes the resolution of the converter (the number of bits).

Let us take a look at the ADC12 block diagram.



Figure . ADC12 block diagram.

## Core

At the heart of the ADC12 is a 12-bit switched capacitor SAR core. It is guaranteed monotonic with no missing codes. The ADC12 core is configured by two control registers, ADC12CTL0 and ADC12CTL1. The core is enabled with the ADC12ON bit. The ADC12 can be turned off when not in use to save power. With few exceptions the ADC12 control bits can only be modified when ENC = 0. ENC must be set to 1 before any conversion can take place. The BUSY flag is set while sampling and conversion is in progress. The result is written to ADC12MEMx memory buffers.

The core uses two programmable/selectable voltage levels (VR+ and VR−) to define the upper and lower limits of the conversion. The digital output (NADC) is full scale (0x0FFF) when the input signal is equal to or higher than VR+, and zero when the input signal is equal to or lower than VR−. The input channel and the reference voltage levels (VR+ and VR−) are defined in the conversion-control memory. The conversion formula for the ADC result NADC is:

NADC = 4095\* (Vin – VR-)/( VR+ - VR-)

## Clock

The SAR block uses ADC12CLK signal that feeds both the sample-and-hold and the SAR core blocks. The ADC12CLK can be sourced from ACLK, SMCLK, MCLK, or the module’s internal oscillator ADC12OSC; the source clock is selected by the ADC12SSELx bits. The ADC12OSC, generated internally, is in the 5-MHz range but varies with individual devices, supply voltage, and temperature.

The user must ensure that the clock chosen for ADC12CLK remains active until the end of a conversion. If the clock is removed during a conversion, the operation will not complete and any result will be invalid. If selected, the ADC12OSC is automatically enabled when needed and disabled when conversions have finished.

The frequency of the selected clock can be divided by 1, 2, 4, or 8, controlled by the ADC12DIVx bits.

## ADC12 Inputs and Multiplexer

An analog multiplexer selects one channel for conversion out of eight external (A0 – A7) and four internal (A12-A15) analog signals. The multiplexer is designed to reduce input-to-input noise injection and to minimize the coupling between channels (see figure below). The channels that are not selected are isolated from the ADC core and the intermediate node is connected to analog ground (AVss). The INCHx control bits control the analog multiplexer.



Figure . Input analog multiplexer.

## Reference Voltage

The ADC12 module contains a built-in voltage reference with two selectable voltage levels, 1.5 V and 2.5 V. Either of these reference voltages may be used internally and externally on pin VREF+. Setting REFON=1 enables the internal reference. When REF2\_5V = 1, the internal reference is 2.5 V, the reference is 1.5 V when REF2\_5V = 0. The reference can be turned off to save power when not in use.

For proper operation the internal voltage reference generator must be supplied with storage capacitance across VREF+ and AVSS. The recommended storage capacitance is a parallel combination of 10-μF and 0.1-μF capacitors. From turn-on, a maximum of 17 ms must be allowed for the voltage reference generator to bias the recommended storage capacitors. If the internal reference generator is not used for the conversion, the storage capacitors are not required.

External references may be supplied for VR+ and VR− through pins VeREF+ and VREF−/VeREF− respectively.

## Sample –and-Hold Unit

Before the conversion starts, the ADC12 core needs to sample the input signal. The input to the SAR block can be modeled as a capacitor. This capacitor must be “fully” charged before the conversion starts and the time needed for this step usually sets the maximum speed of the converter. With 12-bit resolution we want to resolve differences of VFS/212 V in the input analog signal, and we want the error to be less than ½ LSB (which is VFS/213). From this observation we can determine the time required to charge the input capacitor: e-t/τ < 2-13 = 0.00012 = e-9, where τ is the time constant of the equivalent RC input circuitry (where R is the resistance that comes from the input switches and multiplexer and C is the input capacitance). This means that a time of at least 9τ is needed for the charging input capacitor.

## Sample and Conversion Timing Control

An analog-to-digital conversion is initiated with a rising edge of the sample input signal SHI. The source for SHI is selected with the SHSx bits and includes the following:

* The ADC12SC bit
* The Timer\_A Output Unit 1
* The Timer\_B Output Unit 0
* The Timer\_B Output Unit 1

The polarity of the SHI signal source can be inverted with the ISSH bit. The SAMPCON signal controls the sample period and start of conversion. When SAMPCON is high, sampling is active. The high-to-low SAMPCON transition starts the analog-to-digital conversion, which requires 13 ADC12CLK cycles.

The sampling time can be controlled in two ways. The first, called *pulse mode*, allows for a selected number of cycles of ADC12CLK. Alternatively, the sampling time can be controlled directly by the SAMPCON input from the signal that triggers the conversion. This is called *extended sample mode*.

**Pulse mode.**

The pulse sample mode is selected when SHP = 1. The SHI signal is used to trigger the sampling timer. The SHT0x and SHT1x bits in ADC12CTL0 control the interval of the sampling timer that defines the SAMPCON sample period tsample. The sampling timer keeps SAMPCON high after synchronization with AD12CLK for a programmed interval tsample. The total sampling time is tsample plus tsync (see Figure 3). The SHTx bits select the sampling time in 4x multiples of ADC12CLK. SHT0x selects the sampling time for ADC12MCTL0 to 7 and SHT1x selects the sampling time for ADC12MCTL8 to 15.



Figure . Pulse mode sampling. (SHTx=0000b corresponds to 4 ADC12CLK, 0001b corresponds to 8 ADC12CLK, 0010b corresponds to 16 ADC12CLK, etc; see the reference manual for details).

**Extended sample mode.**

The extended sample mode is selected when SHP = 0. The SHI signal directly controls SAMPCON and defines the length of the sample period tsample. When SAMPCON is high, sampling is active. The high-to-low SAMPCON transition starts the conversion after synchronization with ADC12CLK (see Figure 4).



Figure . Extended sample mode.

## Conversion Memory

Conversions are specified and the results are stored as follows. There are 16 ADC12MEMx conversion memory registers to store conversion results. Each ADC12MEMx is configured with an associated ADC12MCTLx control register. The SREFx bits define the voltage reference and the INCHx bits select the input channel. The EOS bit defines the end of sequence when a sequential conversion mode is used. A sequence rolls over from ADC12MEM15 to ADC12MEM0 when the EOS bit in ADC12MCTL15 is not set.

The CSTARTADDx bits define the first ADC12MCTLx used for any conversion. If the conversion mode is single-channel or repeat-single-channel the CSTARTADDx points to the single ADC12MCTLx to be used.

If the conversion mode selected is either sequence-of-channels or repeat-sequence-of-channels, CSTARTADDx points to the first ADC12MCTLx location to be used in a sequence. A pointer, not visible to software, is incremented automatically to the next ADC12MCTLx in a sequence when each conversion completes. The sequence continues until an EOS bit in ADC12MCTLx is processed - this is the last control byte processed. When conversion results are written to a selected ADC12MEMx, the corresponding flag in the ADC12IFGx register is set.

## Conversion Modes

The ADC12 has 4 conversion modes specified by the CONSEQx control bits as shown below.

|  |  |  |  |
| --- | --- | --- | --- |
| CONSEQx | Mode | Operation | Description |
| 00 | Single channel single-conversion | A single channel is converted once | A single channel is sampled and converted once.   1. The ADC result is written to the ADC12MEMx defined by the CSTARTADDx bits. 2. When ADC12SC triggers a conversion, successive conversions can be triggered by the ADC12SC bit. 3. When any other trigger source is used, ENC must be toggled between each conversion. |
| 01 | Sequence-of-channels | A sequence of channels is converted once | A sequence of channels is sampled and converted once.  a.-c. (from above)  d. The sequence stops after the measurement of the channel with a set EOS bit. |
| 10 | Repeat single-channel | A single channels is converted repeatedly | A single channel is sampled and converted continuously. The ADC results are written to the ADC12MEMx defined by the CSTARTADDx bits. It is necessary to read the result after the completed conversion because only one ADC12MEMx memory is used and is overwritten by the next conversion. |
| 11 | Repeat sequence of channels | A sequence of channels is converted repeatedly | A sequence of channels is sampled and converted repeatedly. The ADC results are written to the conversion memories starting with the ADC12MEMx defined by the CSTARTADDx bits. The sequence ends after the measurement of the channel with a set EOS bit and the next trigger signal re-starts the sequence. |

## Example #1 (SW triggered conversion, AVcc reference voltage)

Problem statement: Sample an analog input A0 with reference to AVcc (see below). Start conversion from software (ADC12SC bit). If A0 > 0.5\*AVcc, set P5.1 output, otherwise reset it.

// MSP430xG461x

// -----------------

// /|\| XIN|-

// | | | 32kHz

// --|RST XOUT|-

// | |

// Vin -->|P6.0/A0 P5.1|--> LED

//

Design:

Step 1: Clocks: ACLK = 32kHz, MCLK = SMCLK = default DCO 1048576Hz, ADC12CLK = ADC12OSC

Step 2: Stop watchdog timer.

Step 3: Ports initialization: P6.0 as special function port (analog input channel A0), P5.1 is digital output.

Step 4: ADC12 initialization:

* Single-channel single-conversion (CONSEQx=00; Default, no need to set it).
* Input channel 0 (INCHx=0000; Default, no need to set it).
* Sample-and-hold select: use software trigger from ADC12SC (SHSx=00; Default, no need to change it).
* Sample-and-hold time: take 16 ADC12CLKs (SHT0x=0x0010; SHT0x=0x0000 by default, should set it in your program).
* SHP bit (type of sampling); Select pulse mode (SHP=1, internal timer generate sampling time). Default is extended sample mode (SHP=0), so this bit should be set in the program.
* Starting address: CSTARTADDx=0000 (register 0); Default, no need to set it.
* Reference voltages for ADC12MCTL0: VR+ = AVCC and VR− = AVSS, (SREF=000, Default).
* Enable interrupts from ADC12, enable ADC (ENC bit).

Step 5: Software organization. Have ADC12 ISR (read sample, compare, and control P5.1). As soon as the conversion is done (ADC12SC bit is cleared), start a new conversion. Wake up the processor in the main loop to request new conversion.

See the code on the next page.

#include "msp430xG46x.h"

void main(void)

{

WDTCTL = WDTPW + WDTHOLD; // Stop WDT

ADC12CTL0 = SHT0\_2 + ADC12ON; // Sampling time, ADC12 on

ADC12CTL1 = SHP; // Use sampling timer

ADC12IE = 0x01; // Enable interrupt

ADC12CTL0 |= ENC;

P6SEL |= 0x01; // P6.0 ADC option select

P5DIR |= 0x02; // P5.1 output

while (1)

{

ADC12CTL0 |= ADC12SC; // Start sampling/conversion

\_\_bis\_SR\_register(LPM0\_bits + GIE); // LPM0, ADC12\_ISR will force exit

}

}

#pragma vector = ADC12\_VECTOR

\_\_interrupt void ADC12\_ISR(void)

{

if (ADC12MEM0 >= 0x7ff) // ADC12MEM = A0 > 0.5AVcc?

P5OUT |= 0x02; // P5.1 = 1

else

P5OUT &= ~0x02; // P5.1 = 0

\_\_bic\_SR\_register\_on\_exit(LPM0\_bits); // Exit LPM0

}

## Example #2 (SW triggered conversion, 2.5 V internal reference voltage)

Problem statement: Sample an analog input A0 with reference to the internal reference voltage (2.5 V). Repeat the previous assignment.

Design: Similar to the previous example.

Step 1: See previous example.

Step 2: Enable internal reference voltage; it requires 16 ms delay to settle down, before it can be used.

Step 3: Select input reference voltages (VR+ = VREF+ and VR− = AVSS): SREF=001 (default is 000, should be set).

See the next page.

#include "msp430xG46x.h"

volatile unsigned int i;

void main(void)

{

WDTCTL = WDTPW + WDTHOLD; // Stop watchdog

P6SEL |= 0x01; // Enable A/D channel A0

P5DIR |= 0x02; // P5.1 output

**ADC12CTL0 = REFON + REF2\_5V + ADC12ON + SHT0\_2;**

// turn on 2.5V ref, set samp time (16x)

ADC12CTL1 = SHP; // Use sampling timer

**ADC12MCTL0 = SREF\_1; // Vr+=Vref+**

**for (i = 0x3600; i; i--); // Delay for needed ref start-up.**

// See datasheet for details.

ADC12CTL0 |= ENC; // Enable conversions

while (1)

{

ADC12CTL0 |= ADC12SC; // Start conversions

**while (!(ADC12IFG & 0x0001)); // Conversion done?**

if (ADC12MEM0 >= 0x7ff) // ADC12MEM = A0 > 0.5AVcc?

P5OUT |= 0x02; // P5.1 = 1

else

P5OUT &= ~0x02; // P5.1 = 0

}

}

## Example #3 (Multiple-channels)

Problem statement: Perform a repeated sequence of conversions using "repeat sequence-of-channels" mode on Channels A0, A1, A2, and A3. AVcc is used for the reference. Each conversion result is stored in ADC12MEM0, ADC12MEM1, ADC12MEM2, and ADC12MEM3 respectively. After each sequence, the 4 conversion results are moved to A0results[], A1results[], A2results[], and A3results[]. Test by applying voltages to channels A0 - A3. Open a watch window with the debugger and view the results.

Start new conversion as soon as the previous sequence is completed. This can be done by setting the MSC control bit (the first rising edge of the SHI signal triggers the sampling timer, but further sample-and-conversions are performed automatically as soon as the prior conversion is completed).

// MSP430xG461x

// -----------------

// | XIN|-

// Vin0 -->|P6.0/A0 | 32kHz

// Vin1 -->|P6.1/A1 XOUT|-

// Vin2 -->|P6.2/A2 |

// Vin3 -->|P6.3/A3 |

// | |

Design:

Step 1: Four input channels.

Step 2: Reference voltage (default).

Step 3: Repeat sequence of channels (MSC bit).

Step 4: Sample and hold (1000 – 256 ADC12CLK).

Step 5: Internal arrays to keep sampled values for each channel (A0Results[8]… A3Results[8]).

Step 6: Where do samples go? ADC12MEM0 – ADC12MEM3. Control register for each channel.

Step 7: ISR to read samples. When?

#include "msp430xG46x.h"

volatile unsigned int A0results[8];

volatile unsigned int A1results[8];

volatile unsigned int A2results[8];

volatile unsigned int A3results[8];

unsigned int Index;

void main(void)

{

WDTCTL = WDTPW + WDTHOLD; // Stop watchdog

P6SEL |= 0x0f; // Enable A/D inputs

ADC12CTL0 = SHT0\_8 + MSC + ADC12ON; // Turn on ADC12, use int. osc.

// extend sampling time so won't

// get overflow

// Set MSC so conversions triggered

// automatically

ADC12CTL1 = SHP + CONSEQ\_3; // Use sampling timer, set mode

ADC12IE = 0x08; // Enable ADC12IFG.3 for ADC12MEM3

ADC12MCTL0 = INCH\_0; // A0 goes to MEM0

ADC12MCTL1 = INCH\_1; // A1 goes to MEM1

ADC12MCTL2 = INCH\_2; // A2 goes to MEM2

ADC12MCTL3 = EOS + INCH\_3; // A3 goes to MEM3, end of sequence

while (1)

{

ADC12CTL0 |= ENC; // Enable conversions

ADC12CTL0 |= ADC12SC; // Start conversions

\_\_bis\_SR\_register(LPM0\_bits + GIE); // LPM0

}

}

#pragma vector = ADC12\_VECTOR

\_\_interrupt void ADC12\_ISR(void)

{

A0results[Index] = ADC12MEM0; // Move A0 results, IFG is cleared

A1results[Index] = ADC12MEM1; // Move A1 results, IFG is cleared

A2results[Index] = ADC12MEM2; // Move A2 results, IFG is cleared

A3results[Index] = ADC12MEM3; // Move A3 results, IFG is cleared

Index = (Index + 1) & 0x7; // Increment results index, modulo

\_\_no\_operation(); // SET BREAKPOINT HERE

\_\_bic\_SR\_register\_on\_exit(LPM0\_bits); // Exit LPM0

}