

Implementing an Analog-to-Digital Converter on the Texas Instruments MSP430 Microcontroller

Karl Anderson

April 5, 2013

ECE 480 Design Team 4

1 | Abstract

The MSP430 is a line of inexpensive and low-power microcontrollers (MCUs) produced by Texas Instruments. These MCUs feature several peripherals that can be used to accomplish a wide variety of tasks. One such peripheral is an analog-to-digital converter (ADC) that can be used to measure and capture analog signals so they can be further processed by the microcontroller. The ADCs that are featured in the various MSP430 devices have multiple channels to handle several analog input signals as well as multiple modes of operation.

1.1 | Keywords

Analog-to-Digital Converter (ADC), Code Composer Studio (CCS), Microcontroller Unit (MCU), Conversion Modes, Interrupt Service Routine (ISR), Interrupt, Timer, Control Register

Table of Contents

1 Abstract	1
1.1 Keywords	1
2 Introduction	3
3 Objective	3
4 Hardware & Software	4
5 ADC Modes	4
6 ADC Conversion Process	4
7 ADC Registers	6
8 Configuring the ADC	7
9 Conclusion	8
10 References	8

2 | Introduction

The MSP430 line of microcontrollers from Texas Instruments features an 8-channel 10-bit analog-to-digital converter (ADC). This means that the MSP430 can perform analog-to-digital conversions on up to eight input signals with 10 bit resolution. This ADC is a very powerful tool in many applications as it enables the MCU to measure analog input signals so that they can be processed. This allows the MSP430 to dynamically respond to analog inputs and perform various tasks based on these inputs. An example is a control application where the MSP430 measures an analog sensor output and makes decisions based on this changing analog value.



Figure 1. MSP430 LaunchPad

3 | Objective

The objective of this application note is to inform users about the ADC peripherals of the MSP430 line of microcontrollers. Topics that will be covered include the various operating modes and MCU registers associated with the ADC peripheral and software configuration of the ADC with examples of code relating to various aspects of the ADC.

4 | Hardware and Software

Along with the MSP430 microcontroller, Texas Instruments also produces a development board that is an essential tool for creating MSP430 applications called the LaunchPad. It features on-board emulation for debugging the MSP430. The integrated development environment (IDE) used to develop code for the MSP430 is TI's Code Composer Studio (CCS). The LaunchPad can connect to a personal computer via a mini-USB cable included with the LaunchPad, which will power the board and allow the user to program and debug the MSP430. Once it has been programmed, the microcontroller can be removed from the LaunchPad and installed in the system designed for it.

5 | ADC Modes

The ADC featured on the MSP430 has four modes of operation: single channel single-conversion, sequence-of-channels, repeat single channel, and repeat sequence-of-channels. Below is a table describing each mode of the ADC.

CONSEQx	Mode	Operation
00	Single channel single-conversion	A single channel is converted once.
01	Sequence-of-channels	A sequence of channels is converted once.
10	Repeat single channel	A single channel is converted repeatedly.
11	Repeat sequence-of-channels	A sequence of channels is converted repeatedly.

Figure 2. ADC Conversion Modes

6 | ADC Conversion Process

The basic sequence of events that occurs during each conversion that the ADC performs is demonstrated in the single channel single-conversion mode:

1. The conversion mode is selected
2. The ADC is turned on
3. The input channel is selected
4. The ADC waits to be enabled or instructed to start conversion
5. When a conversion is triggered, the voltage at the selected input is sampled

6. This voltage is converted to a digital value
7. This value is written to ADC memory register
8. An ADC interrupt is triggered
9. The ADC again waits to be instructed to start a conversion

This process is illustrated below for the single-channel single-conversion mode.

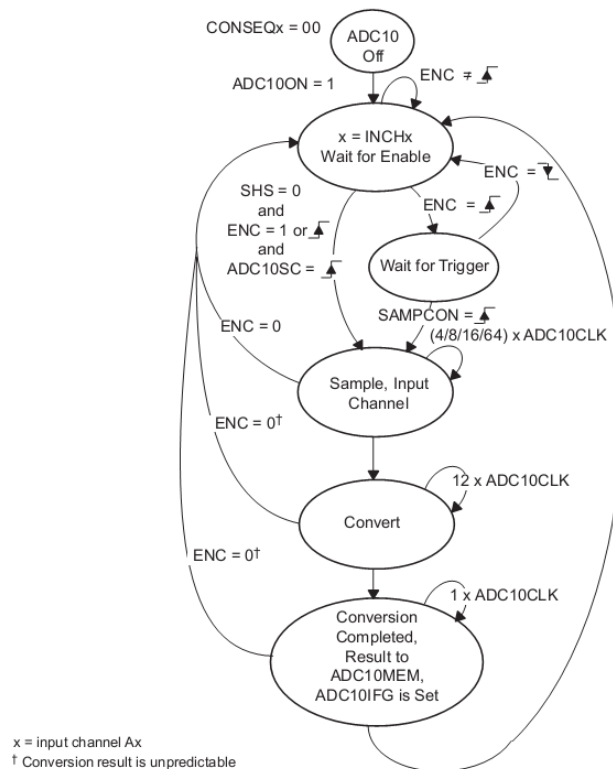


Figure 3. Conversion Process for Single-Channel Single-Conversion Mode

In the other conversion modes, this process only differs in that multiple channels are automatically samples successively or individual channels are automatically sampled repeatedly.

7 | ADC Registers

The MSP430 has several control registers that are used to configure various operating parameters of the peripheral that they correspond to. The registers that are used by the ADC are listed in the table below.

Register	Short Form	Register Type	Address	Initial State
ADC10 input enable register 0	ADC10AE0	Read/write	04Ah	Reset with POR
ADC10 input enable register 1	ADC10AE1	Read/write	04Bh	Reset with POR
ADC10 control register 0	ADC10CTL0	Read/write	01B0h	Reset with POR
ADC10 control register 1	ADC10CTL1	Read/write	01B2h	Reset with POR
ADC10 memory	ADC10MEM	Read	01B4h	Unchanged
ADC10 data transfer control register 0	ADC10DTC0	Read/write	048h	Reset with POR
ADC10 data transfer control register 1	ADC10DTC1	Read/write	049h	Reset with POR
ADC10 data transfer start address	ADC10SA	Read/write	01BCh	0200h with POR

Figure 4. ADC Registers

The registers that are used to configure the operation of the ADC are the control registers ADC10CTL0 and ADC10CTL1. The figures below show the contents of these registers.

15	14	13	12	11	10	9	8
SREFx			ADC10SHTx		ADC10SR	REFOUT	REFBURST
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
MSC	REF2_5V	REFON	ADC10ON	ADC10IE	ADC10IFG	ENC	ADC10SC
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Figure 5. ADC10CTL0 Register

15	14	13	12	11	10	9	8
INCHx				SHSx		ADC10DF	ISSH
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
ADC10DIVx			ADC10SSELx		CONSEQx		ADC10BUSY
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	r-0

Figure 6. ADC10CTL1 Register

Below is a table describing the portions of these control registers that are essential for basic use of the ADC.

Register Section	Description
SREFx	Selects high and low reference voltage for ADC
ADC10SHTx	Sets ADC sample and hold time
ADC10ON	Turns ADC on or off
ADC10IE	Enables or disables ADC interrupt
ADC10IFG	Interrupt flag set when ADC10MEM is loaded with conversion result
ENC	Enables conversion
ADC10SC	Starts conversion
INCHx	Selects input channel
ADC10DIVx	Selects divider for the selected ADC clock
ADC10SSELx	Selects source of the clock that will control the ADC
CONSEQx	Selects conversion sequence mode
ADC10BUSY	Indicates whether the ADC is active

Figure 7. Description of key control register sections

8 | Configuring the ADC

The ADC can be configured with only a few lines of code. Below is an example of a simple configuration function, written in the C programming language.

```

1 void ADC_capture(void)
2 {
3     /* Configure ADC Channel */
4     ADC10CTL0 = ADC10SHT_3 + ADC10ON + ADC10IE; //64 clk ticks, ADC on, enable interrupt
5     ADC10CTL1 = ADC10SSEL_0 + INCH_5; //SMCLK, channel 5
6     ADC10CTL0 |= ENC + ADC10SC; //Enable and start conversion
7     while ((ADC10CTL1 & ADC10BUSY) == 0x01); //Wait for conversion to end
8 }

```

Figure 8. ADC Configuration Code Example

In this example, the control registers are initialized to perform in the way the user intends to use the ADC. The symbols such as ADC10SHT_3 seen in the code above stand for individual binary values corresponding to the different configuration options for each section of the ADC control registers. These options and the corresponding symbols are defined in the MSP430 User's Guide, which can be found in a link in the references section. In this example, the ADC is turned on and channel 5 is selected as the input channel. The SMCLK is selected as the ADC clock and the ADC interrupt is enabled. The ADC is instructed to hold its samples for 64 clock ticks before making a

conversion. By default, the ADC is in the single channel single-conversion mode. When instructed, the ADC begins conversion and the CPU will wait until the ADC10BUSY bit is cleared before continuing program execution. The ADC triggers an interrupt every time it performs a conversion, so it is necessary to write an interrupt service routine to handle the interrupt for the ADC to function properly. Below is an example of an interrupt service routine to handle the ADC interrupt.

```
1 // ADC10 interrupt service routine
2 #pragma vector=ADC10_VECTOR
3 __interrupt void ADC10_ISR (void)
4 {
5     __bic_SR_register_on_exit(CPUOFF);          // Return to active mode
6 }
```

Figure 9. ADC Interrupt Service Routine Example

This code can be executed at time intervals defined by a timer peripheral on the MSP430 to achieve the desired sample rate.

9 | Conclusion

The analog-to-digital converter featured on the MSP430 microcontrollers is a very powerful tool for developers of applications using this device. It allows users to measure analog signals such as sensor outputs for instrumentation or control applications. This application note introduced users to the various operating modes and registers associated with the ADC and showed how it can be configured with software for the desired application.

10 | References

[1] MSP430 User's Guide

<http://www.ti.com/product/msp430g2553>

[2] MSP430 LaunchPad Wiki

[http://processors.wiki.ti.com/index.php/MSP430_LaunchPad_\(MSP-EXP430G2\)](http://processors.wiki.ti.com/index.php/MSP430_LaunchPad_(MSP-EXP430G2))