

Application Note

Analog to Digital Conversion

Eaton, David

11/16/2012

Keywords

ADC- Analog to Digital Conversion

MC- Microcontroller, is a small computer on a single [integrated circuit](#) containing a processor core, memory, and programmable [input/output](#) peripherals

Register- are small memory elements of size in few byte or word.

Bit- These are slots located in the Register correlating to different settings for the ADC

The purpose of this paper is to describe the aspects of the ADC10 ability of the MSP430. After reading this paper the user should be able utilized the ADC on future projects

Introduction

In order to effectively communicate between the MSP430G2452 Microcontroller and the Ti accelerometer, we need to utilize the ADC channels on the Microcontroller. The ADC will be able to successfully input a DC voltage from the Accelerometer translate that voltage into a digital value. With that digital value the microcontroller will be able to compare it with a setpoint to see if the movement of accelerator is under an acceptable value.

An important reason why we picked the MSP430G2452 is the fact that it has a 10 bit conversion rather than an 8 bit converter like most MC's. This is beneficial because it gives out a better range and more sensitivity to more accurately capture of the analog signal.

Objectives

The objective for this application note is to explain and show how to use the ADC application on a MSP430G2X. This ranges from setting up the ports on the MC, running the initialization of the ADC, different settings of the ADC, and finally how to access this new data. Below is a picture of the different ports located on our MC. The ports highlighted in red represent the channels we are using for our project. However, the ports from P1.0 to P1.7 can all be used for ADC

	DVCC	1	20	DVSS
P1.0/TA0CLK/ACLK/A0/CA0		2	19	XIN/P2.6/TA0.1
P1.1/TA0.0/A1/CA1		3	18	XOUT/P2.7
P1.2/TA0.1/A2/CA2		4	17	TEST/SBWTCK
P1.3/ADC10CLK/CAOUT/VREF-/VEREF-/A3/CA3		5	16	RST/NMI/SBWTIO
P1.4/TA0.2/SMCLK/A4/VREF+/VEREF+/CA4/TCK		6	15	P1.7/SDI/SDA/CAOUT/A7/CA7/TDO/TDI
P1.5/TA0.0/SCLK/A5/CA5/TMS		7	14	P1.6/TA0.1/SDO/SCL/A6/CA6/TDI/TCLK
P2.0		8	13	P2.5
P2.1		9	12	P2.4
P2.2		10	11	P2.3

NOTE: ADC10 pin functions are available only on MSP430G2x52.

Configuring the ADC

ENABLE ADC For ADC to work the user must first set the ADC10ON bit of the ADC10CTL0 register. This turns on the ADC, so that the ADC isn't constantly on and drawing power.

ADC Conversion MODES

REFERENCE VOLTAGE SELECTION The ADC10ON value in the ADC10CTL0 register represents the reference voltage for the conversion. For example in this case the SREF_0 selection

```
        return 0;
    }
    unsigned int analogRead(unsigned int pin) {
        ADC10CTL0 = ADC10ON + ADC10SHT_2 + SREF_0;
        ADC10CTL1 = ADC10SSEL_0 + pin;
        if (pin==INCH_3){
            ADC10AE0 = 0x08;
        }
        else if (pin==INCH_4){
            ADC10AE0 = 0x10;
        }
        else if (pin==INCH_2){
            ADC10AE0 = 0x04;
        }
        ADC10CTL0 |= ENC + ADC10SC;
        while (1) {
            if (((ADC10CTL0 & ADC10IFG)==ADC10IFG)) {
                ADC10CTL0 &= ~(ADC10IFG + ENC);
                break;
            }
        }
        return ADC10MEM;
    }
}
```

more accurate the conversion is, however as a drawback the MC will be drawing more power. We can select a certain number of Clock cycles to be used for ADC sampling.

ADC10SHT_0 => 4 Clock Cycles

ADC10SHT_1 => 8 Clock Cycles

ADC10SHT_2 => 16 Clock Cycles

ADC10SHT_3 => 64 Clock Cycles

For our design we decided to use the *ADC10SHT_2* at 16 clock cycles because it gave us enough accuracy. Also, we are trying to conserve as much power as possible, so the 64 clock cycle would be a wasteful usage of our power.

CLOCK SOURCE SELECTION :

Now that we have selected the sampling time, it is important to make sure we select a clock appropriate for our usage. These are also done in *ADC10CTL1* register with the **ADC10SSELx** bit. Below is a table of the setting you can choose for the clock source selection.

00 => *ADC10OSC*

01 => *ACLK*

10 => *MCLK*

11 => *SMCLK* **Configuring the ADC**

ENABLE ADC For ADC to work the user must first set the *ADC10ON* bit of the *ADC10CTL0* register. This turns on the ADC, so that the ADC isn't constantly on and drawing power.

ADC Conversion MODES

REFERENCE VOLTAGE SELECTION The *ADC10ON* value in the *ADC10CTL0* register represents the reference voltage for the conversion. For example in this case the *SREF_0* selection

Enable Analog inputs

Finally to enable the ADC conversion on a specific pin the corresponding *ADC10AE0* bit needs to be set on the *ADC10AE0* register. For our project we have to enable an ADC on three different channels at different times. The overall function of *analogRead* function reads an input from the user, corresponding to the channel that the user wants to sample.

```
#define INPUT_0 INCH_2 //x
```

```
adcvaluex0 = analogRead( INPUT_0 )
```

The above two code lines represent the passing of the input that the user wants to sample. When the *analogRead* function receives the *INPUT_0* value it translates that *INPUT_0* value to be *INCH_2* based on the define function. Referencing back to the code, if the program inputs an *INCH_2* value, the corresponding input port will be selected for ADC.

ADC Conversion MODES :

There are also four different modes available for conversion :

- Single channel, Single conversion
- Sequence of channels
- Repeat single channel
- Repeat sequence of channels

These modes can be selected by **CONSEQx** Bits in **ADC10CTL1** register, however for our project we will simply call each channel independently, so the default mode is appropriate for our usage.

Utilizing Digital value after Conversion

At the end of the code the program will return the value stored in *ADC10MEM* to the integer value that was predetermined. For example our ADC will return a digital value and assign it the integer *adcvaluex0*. This value can then be utilized like another value

Conclusion

Those the simple steps to configure and use the ADC10 on the MSP430G2452, and some of the code used on our project. Hopefully this gives a better understanding to how ADC10 works and is utilized.

3. The application note must be developed for web viewing, i.e., either as HTML or PDF format. The documents will be accessible from the design-team web site. A paper copy also needs to be submitted to your facilitator, unless he/she asks for only an electronic copy.
4. Each AN should begin with the following information: title, student name, date, executive summary (or abstract), and keywords. The body of the AN should be divided into subsections, beginning with an introduction to the topic and the objective of the note. Based on the objective, it should then describe the issues, steps, examples, hardware or software developed, etc. of the study undertaken. It should end with results, conclusions, and/or recommendations, as appropriate to the objective, and references.
5. Examples of commercial and trade application notes can be found on the web sites of many electronics and software companies.
6. The topic of the application note should be relevant to your role on your design-team.
7. The due date is Friday, April 1 by 5:00PM. A copy should be turned in on paper to your facilitator or delivered electronically to him/her (typically as a .docx (or .doc) file, so it is editable/commentable by him/her), depending on his/her preference. A copy (.html or .pdf file) must also be posted on your web site.