

Analog to Digital Conversion of Sound with the MSP430F2013

Christopher Johnson

4/2/2010

Abstract

Several modern-day applications require that analog signals be converted to digital signals in order to be processed by either a computer or microcontroller. Conversion of sound waves to a digital signal with the MSP430F2013 will be examined. The MSP430F2013 analog to digital converter is well documented. However, specific applications are not well explained. Digital sampling with the MSP430 will be outlined.

Keywords: MSP430F2013, Analog to Digital Conversion, Sound, Amplifier

Introduction

The MSP430F2013 includes a five channel multiplexed sigma delta analog to digital (AD) converter. The AD converter has up to 16 bit resolution as well as an on board reference voltage. This means there are 2^{16} levels of resolution, which is very high. Even though all of these features are included on chip, there is not readily available documentation explaining how to initialize and then make an AD conversion with the microcontroller.

Objective

The following will detail the necessary pre-amp required to boost the audio signal to a usable level. This application note will also provide the necessary supporting circuitry required for AD conversion. In terms of software, initialization of the AD converter on the MSP430F2013, including explanations of necessary registers will be discussed. Source code and schematics are included in order to explicitly show the correct implementation of a simple, continuous AD conversion.

Methods

Signal Pre-amplifier

Before programming the microcontroller begins, the analog input signal needs to be conditioned. A op-amp circuit is used for this. In addition to the op-amp, several filters can be added in order to further condition the input. The circuit below in Figure1 shows a possible implementation. The circuit below acts as a simple first order low pass filter, serving the purpose of capping the highest frequency willing to be passed. This is important because excessive input noise can be eliminated. This filter should be set with a cut-off frequency of $2 \times f_{\text{sampling}}$ where f_{sampling} is the sampling speed for the intended application. This is in agreement with the nyquist theorem which states that to correctly reconstruct a signal; it must be sampled at twice rate of the highest present frequency. The sampling rate can be determined through software by inserting delays between each sample command. The filter can then be designed to fit this specification.

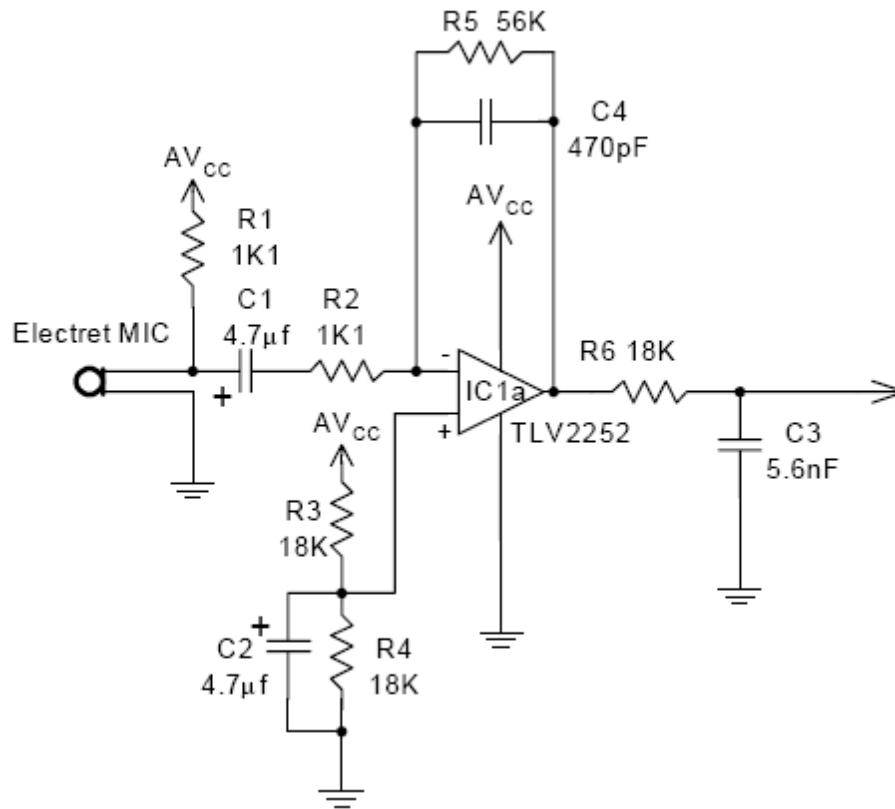


Figure 1: Pre-amp circuit design. Taken from, [Solid State Voice Recorder Using Flash MSP430 \(1\)](#) by Texas Instruments

Several other components must be added to the amplifier circuit in order to accurately acquire the analog signal. A pull-up resistor must be added to the positive terminal of the microphone, R1, a value of 10kΩ should suffice. Figure 1 above shows R1 as 1KΩ, however depending on the microphone being used this is not always a high enough value. C1 acts as a high pass filter to eliminate any DC signal from entering the amplifier. C4 in the feedback loop filters out high frequency noise. C3 can be adjusted to condition the output frequency by the following equation.

$$f = \left(\frac{1}{R6 * C3} \right) * 2\pi$$

The gain of the circuit can also be tuned by adjusting the value of the feedback resistor R5 through the following equation.

$$Gain = -\frac{R5}{R2}$$

The maximum output voltage of the circuit will be from 0 to Vcc. Any input can only be amplified to Vcc. Anything input after amplification resulting in a signal size higher than this will produce a clipped waveform and will not be able to be reproduced effectively.

Microcontroller Connections

Once the signal conditioning is completed the MSP430F2013 needs to be properly connected. The connections provided will work for the program in the Appendix. First power needs to be supplied to the microcontroller. To do this pin 1 is connected to Vdd and pin 14 is connected to GND. The program will sample through input channel A1. Therefore, pin 4 (A1+) will be the input voltage, and pin 5 (A1-) will be connected to ground.

Initialize the AD Converter

In order to initialize the ADC several parameters, registers, and bits need to be understood and set to appropriate values. First the reference voltage needs to be set. In the example below, the on chip 1.2V reference source is used. The total resolution is defined over the total voltage range as defined below.

$$\pm V_{FSR} = \frac{\frac{V_{ref}}{2}}{GAIN_{PGA}}$$

With a reference of 1.2V the total voltage range ends up being $\pm 0.6V$. The internal voltage reference is activated by setting the SD16REFON bit. An external voltage source can also be used. In order to activate the external voltage source reference, the reference voltage is attached to the Vref pin and SD16REFON and SD16VMIDON are both reset.

Next, the analog needs to be configured. The inputs are configured using the SD16CTL0 and SD16AE registers. The MSP430F2013 includes a five channel multiplexed AD converter. Therefore up to five separate inputs can be monitored by the microcontroller. The SD16INCHx bits, where x is the channel (0 – 4), select the analog input for conversion. The SD16AEx bits, where x is the input pin, enables or disables the input pin. If desired, the input signal can be boosted by setting the SD16GAINx bits, where x is the input channel. The microcontroller has 6 available gain settings. However, if the onboard gain adjustment is used, the settling time required to obtain a signal is altered. This leads to a lower maximum sampling frequency and therefore a lower maximally recorded frequency via the nyquist theorem.

Setting the `SD16SNGL` bit selects the type of conversion that will be performed. Setting the bit to 1 selects single conversion mode and setting the bit to 0 initiates a continuous conversion. Setting the `SD16SC` of a specific channel will initiate a conversion. When `SD16SNGL` is set to 1 and `SD16SC` is set to one, the AD converter will make a measurement and place the measurement into the `SD16MEMO` register. It is recommended to read the `SD16MEMO` register prior to clearing the `SD16SC` bit to avoid corrupting the measurement. In the example located in the appendix, the `SD16SC` bit is set to 0 in order to invoke a continuous conversion. Measurements are read when the interrupt is triggered by the `_BIS_SR(LPM0_bits + GIE);` line. The logic inside of the interrupt can then be modified to suite whatever application is to be completed.

Discussion

Making AD conversions is a very widely used application for any microcontroller. The MSP430F2013 has a very powerful AD converter that can be used for these applications. The specific application to making samples from an audio signal was discussed here. However, the basic framework provided here can be applied to a variety of other applications. Almost any sensor will return an analog source that needs to be amplified. The amplifier design above can be applied and modified to either enhance gain, or modify the conditioned signal to limit high or low frequency noise further.

The parameters that were set in this application note to initialize the AD converter can also be modified to suit the needs of the intended application. For example, the sampling speed could be modified, as well as the AD resolution. This is possible with the flexibility provided by the AD converter on the MSP430F2013.

References

1. Solid State Voice Recorder Using Flash MSP430. Texas Instruments. Application Note SLA123. January 2001. Raju, Murugavel.
2. MSP430X2XXX Family User Guide. Texas Instruments. SLAU144E. 2008.

Appendix

```
//This example will continuously sample the source provided at pin4 (A1+)
//The on chip reference voltage is used

#include <msp430x20x3.h>

Static unsigned int sample;           // Define location to store sample

void main(void)
{
    WDTCTL = WDTPW + WDTHOLD;         // Stop watchdog timer
    SD16CTL = SD16REFON + SD16SSEL_1; // 1.2V ref, SMCLK
    SD16INCTL0 = SD16INCH_1;          // A1+/-
    SD16CCTL0 = SD16UNI + SD16IE;     // 256OSR, unipolar, interrupt enable
    SD16AE = SD16AE2;                 // P1.1 A1+, A1- = VSS
    SD16CCTL0 |= SD16SC;              // Set bit to start conversion

    _BIS_SR(LPM0_bits + GIE);         // Enter LPM0 with interrupt
}

#pragma vector = SD16_VECTOR           // Interrupt service routine for
__interrupt void SD16ISR(void)         // conversion
{
    if (SD16MEM0 != sample)           // If the value changes, update sample
        sample = SD16MEM0;
}
```