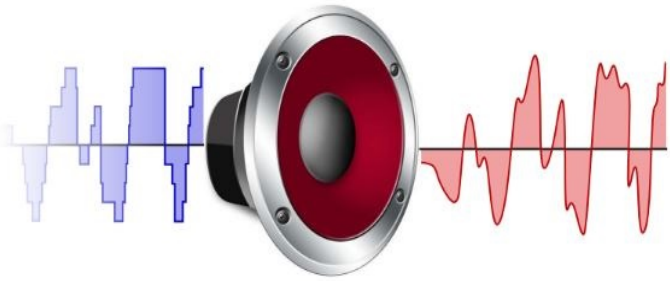
ELECTRICAL AND ELECTRONICS
ENGINEERING
DEPARTMENT



Microprocessor Interfacing
of
A/D Converters

Experiment 5 - Analog to Digital Converter

Objectives

The TM4C is equipped with an analog-to-digital (ATD) conversion system that samples an analog (continuous) signal at regular intervals and then converts each of these analog samples into its corresponding binary value using the successive approximation technique [1]. While doing this lab, you will learn,

- How to program the TM4C ATD converter system.
- How to display the ATD converted analog signal on the termite.

1 Background Information

The TM4C analog-to-digital (ATD) conversion system:

The TM4C ATD conversion system consists of a 12-channel, 12-bit, multiplexed input analog-to-digital converter block. Like many peripherals, in order to use the ATD feature, you must first "power up" one of the ATD modules. To do this, we again use the Run Clock Gate Control register like in Lab 2, only this time for the ATD (RCGCADC at address 0x400F.E638). Set bit 0 of the RCGCADC register to 1 in order to use ATD module 0 (ADC0). Since the ATD module will need access to the external signal being sampled, a GPIO port must be configured to "connect" an external pin to the ATD. Therefore, we must also "power up" the GPIO peripheral just like in Lab 2 using the RCGCGPIO register. Each of the ATD input channels are associated with a pin on the board. For this lab, we will use pin PE3. Figure 1 shows the assignment of each channel.

Every sample, or sequence of samples, is controlled by a sequencer. There are 4 sequencers total that can control every channel. This allows for greater flexibility and ability to work for many applications. Each sequencer also has its own set of attributes that may make it better for one application than another. As seen in Figure 2, each sequencer will take a different number of samples, and has an equivalent size of FIFO buffer. The FIFO buffer is where the results of each sample is stored.

Since this lab will be fairly straight forward in that there is only one signal to sample at a relatively slow sample rate, sequencer 3 (SS3) will work perfectly.

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
AIN0	6	PE3	I	Analog	Analog-to-digital converter input 0.
AIN1	7	PE2	I	Analog	Analog-to-digital converter input 1.
AIN2	8	PE1	I	Analog	Analog-to-digital converter input 2.
AIN3	9	PE0	I	Analog	Analog-to-digital converter input 3.
AIN4	64	PD3	I	Analog	Analog-to-digital converter input 4.
AIN5	63	PD2	I	Analog	Analog-to-digital converter input 5.
AIN6	62	PD1	I	Analog	Analog-to-digital converter input 6.
AIN7	61	PD0	I	Analog	Analog-to-digital converter input 7.
AIN8	60	PE5	I	Analog	Analog-to-digital converter input 8.
AIN9	59	PE4	I	Analog	Analog-to-digital converter input 9.
AIN10	58	PB4	I	Analog	Analog-to-digital converter input 10.
AIN11	57	PB5	I	Analog	Analog-to-digital converter input 11.

Figure 1: ATD channel – pin assignments (Table 13-1 of the datasheet)

Sequencer	Number of Samples	Depth of FIFO
SS3	1	1
SS2	4	4
SS1	4	4
SS0	8	8

Figure 2: ATD channel – pin assignments (ATD Sequencer list (Table 13-2 of the datasheet))

GPIO Setup

As mentioned before, the ATD needs access to the outside world via the GPIO port, so it is a good idea to set this up first. We need to configure the GPIO port so that it can receive a signal (input), and then send that signal to the ATD for sampling. (Refer to Lab 2 or datasheet for a reminder of GPIO registers). After the RCGCGPIO register has been configured to enable port E (by setting bit 4), the Alternate Function register (AFSEL) will need to be set for pin 3. The Alternate Function register tells the TM4C that we will not be using PE3 as a simple on/off switch like we did in Lab 2. Instead, enabling the AFSEL register tells the TM4C that we would like to "connect" the associated pin with some other peripheral in the TM4C. Since we are using PE3 (pin 3 of port E), set bit 3 in the AFSEL register associated with port E. When using an alternate function, the next step is normally to configure the PCTL register to tell the TM4C which of the many peripherals available we would like to use as the alternate function. However, in this case, AIN0 happens to be the default for PE3. Next, set the direction of PE3 to input by setting bit 3 to 1 in the DIR register. Since we will be measuring a continuous (analog) signal, we must enable analog on PE3 by setting bit 3 to 1 in the AMSEL register.

ATD Setup

The base address for the ADC0 configuration registers is 0x4003.8000, and the following configuration registers will be referenced by their offset instead of their full address. Changes to the ATD configuration should only be made while the ATD is disabled! Controlling the ATD is done by controlling its associated "sequencer". The sequencers control when a sample should be taken, and which sequencer is activated is controlled by the ADC Active Sample Sequencer register (ADCACTSS, offset 0x000). For this lab we will be using sequencer 3, therefore, in order to disable sequencer 3, clear bit 3 of the ADCACTSS.

ADC Active Sample Sequencer (ADCACTSS)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x000
Type RW, reset 0x0000.0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved															BUSY
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved												ASEN3	ASEN2	ASEN1	ASEN0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3: The ADC Active Sample Sequencer register

The TM4C has the capability to trigger a sample based on a variety of signals. For instance, a sample can start on the edge of a square wave, by software, or from the timer module. Again, like the interrupts, the trigger to start an ATD conversion is sequencer specific. What triggers the sequencer is configured using the ADC Event Multiplexer Select register (ADCEMUX, offset 0x014). We will be triggering each sample in software for this lab, therefore, bits 15:12 need to be cleared.

ADC Event Multiplexer Select (ADCEMUX)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x014
Type RW, reset 0x0000.0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EM3				EM2				EM1				EM0			
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4: ADC Event Multiplexer Select register

So far we have told the TM4C that we need to create an input through GPIO to be used to receive our signal to be sampled, we have told which ATD module and which sequencer to use. However, we have not told the ATD module which channel to use. To do this, write 0x0 (meaning AIN0) to section 0 (meaning the first sample) of the ADC Sample Sequence Input Multiplexer Select 3 register (meaning sample sequencer 3) to select channel AIN0. The offset for this register ADCSSMUX3 is 0x0A0. The default channel is already AIN0 and SS3 already takes a single sample, so you can as well skip this step.

ADC Sample Sequence Input Multiplexer Select 0 (ADCSSMUX0)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x040
Type RW, reset 0x0000.0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX7				MUX6				MUX5				MUX4			
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX3				MUX2				MUX1				MUX0			
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5: ADC Sample Sequence Input Multiplexer Select 0 register

Next, we will use a polling method in order to know when a sample is ready. To do this we monitor the interrupt flags in the RIS register. However, in order to tell the TM4C to set flags in the RIS register, we must set the IE0 bit in the ADC Sample Sequence Control register (ADCSSCTL3, offset 0x0A4). (Note that we are using SSCTL3 because we are using sequencer 3.) We also need to set the END0 bit to tell the sequencer to stop sampling after one sample. This seems redundant since sequencer 3 only takes one sample, but it is required. Note, that if you were using a different sequencer that had the capability to take many samples, the appropriate SSCTL register is where you would specify the number of samples desired.

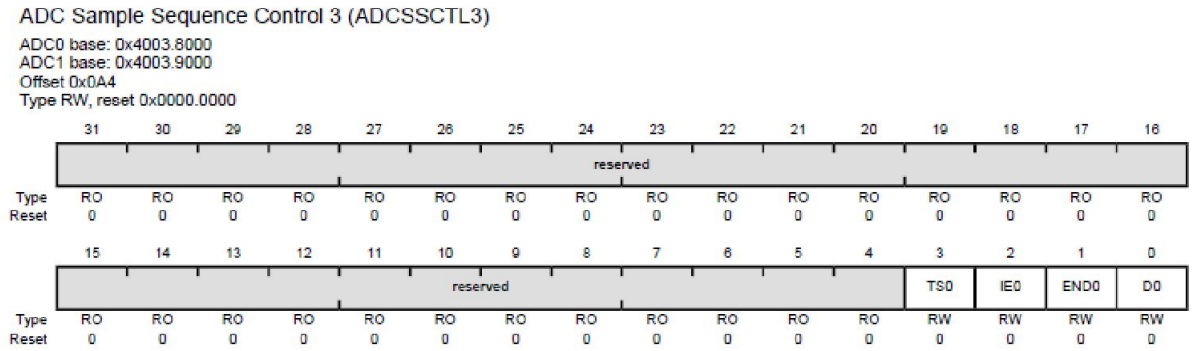


Figure 6: ADC Sample Sequence Control 3 register

ATD sampling rate is controlled by the ADC Peripheral Configuration register (ADCPC, offset 0xFC4). The ATD can sample at 125 kilo-samples per second (ksps), 250 ksps, 500 ksps, and 1 Mega-samples per second (Msps). Since we are triggering via software, and doing so on a relatively slow interval, we can choose a sample rate of 125 ksps. Set bits 3:0 to 0x01 in the ADCPC register to select 125 ksps.

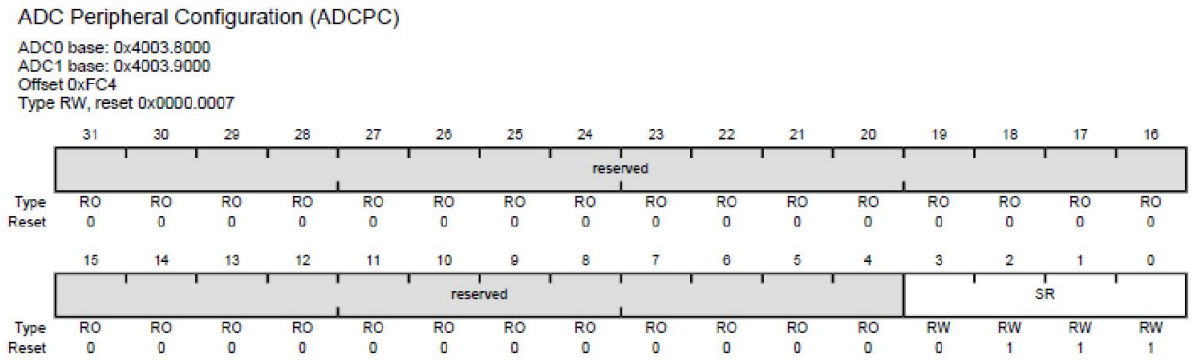


Figure 7: ADC Peripheral Configuration register

The ATD system is now configured and can be enabled. Once enabled, (by setting bit 3 of the ADCACTSS register) the ATD system is ready to start sampling on your trigger.

Sampling

Since we set the ATD module up to initiate a sample from a software trigger, the program needs to tell the ATD module (specifically, the sequencer we are using) to start sampling. This is done using the ADC Processor Sample Sequence Initiate register (ADCPSSI, offset 0x028). Bits 3:0 represent each sequencer. Since we are using sequencer 3, setting bit 3 to 1 tells the sequencer to start sampling.

ADC Processor Sample Sequence Initiate (ADCPSSI)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x028
Type RW, reset -

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GSYNC	reserved			SYNCWAIT	reserved										
Type	RW	RO	RO	RO	RW	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved												SS3	SS2	SS1	SS0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	WO	WO	WO	WO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-	-

Figure 8: ADC Processor Sample Sequence register

A sampling sequence will take a few clock cycles, so we have to monitor when the sequence is complete. The ADC Raw Interrupt Status register (ADCRIS, offset 0x004) contains flags corresponding to each sequencer (bits 3:0) that are set to 1 when a sequence has completed. In our case, checking bit 3 monitors if / when sequencer 3 is done sampling.

ADC Raw Interrupt Status (ADCRIS)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x004
Type RO, reset 0x0000.0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved															INRDC
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved												INR3	INR2	INR1	INR0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9: ADC Raw Interrupt Status register

Each sequencer stores its results in its own FIFO buffer register (ADCSSFIFOn). Address offset: ADCSSFIFO0 0x048 ADCSSFIFO1 0x068 ADCSSFIFO2 0x088 ADCSSFIFO3 0x0A8 Once sampling is complete on sequencer 3, the value of the oldest sample will be in ADC Sample Sequence Result FIFO 3 (ADCSSFIFO3, offset 0x0A8).

ADC Sample Sequence Result FIFO n (ADCSSFIFOn)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x048
Type RO, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved				DATA											
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	-	-	-	-	-	-	-	-	-	-	-	-

Figure 10: ADC Sample Sequence Result FIFO n register

The ADC lets us know a sample value is ready for us by setting the appropriate bit in the ISR, then it waits for us to load the sample from the FIFO register. Therefore, we need to tell the ADC we are ready for it to continue sampling. To do this, we set bit 3 (for sequencer 3) of the Interrupt Status and Clear Register (ADCISC, offset 0x00C).

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved												DCINSS3	DCINSS2	DCINSS1	DCINSS0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												IN3	IN2	IN1	IN0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW1C	RW1C	RW1C	RW1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11: ADC Interrupt Status and Clear register

2 Preliminary Work

1. (40%) Program the ATD conversion system on the board, in C language, to convert the analog signal to a 12-bit number between 0x000 and 0xFFF (4095). The input will be taken from PE3. The output value should be stored in an integer variable. Put a screenshot of Keil showing this variable on your report.
2. (5%) Assume that there is a 1.65 Volt DC offset in your input. Subtract the value corresponding to this offset from your reading. Note that new result is a signed number.
3. (20%) Convert the resulting value to a char array containing a decimal representation with two decimal places (X.YZ) between -1.65 and 1.65. Put a screenshot on your report.
4. (10%) Write a C program that, in an infinite loop, samples the ATD at 1 second intervals, removes the 1.65 Volt DC offset, converts the result to char array and outputs the result on Termite. You may write and use a 1-second delay subroutine. Put a screenshot of Termite on your report.
5. (25%) Write another C program that, in an infinite loop, samples the ATD at 0.1 second intervals. If the result is larger than 1.65 Volt, the onboard red LED (PF1) is turned on. If the result is smaller than 1.65 Volt, red LED is turned off. You may write and use a 100 ms delay subroutine. Put a screenshot of your code in your report.

3 Parts List

1 x Potentiometer
1 x TM4C123G Board

References

[1] TI, "Tiva™ tm4c123gh6pm microcontroller data sheet." http://www.ti.com/lit/ds/spms376e/spms376e.pdf.