6.1 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. While doing this lab, you will learn,

- How to program the TM4C ATD converter system.
- How to convert the binary results to BCD for display as text on the Termite terminal window.
- How to populate a message template in RAM with new values.

62 Related material to read:

- Chapter 20 of text; good for concepts but not our specific ATD converter.
- Valvano Volume 1, Section 10.4, Analog to Digital Conversion.
- TM4C Data Sheet, Chapter 13.

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 as before, 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 4 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 6.1 shows the assignment of each channel.

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Pin Number</th>
<th>Pin Mux / Pin Assignment</th>
<th>Pin Type</th>
<th>Buffer Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIN0</td>
<td>6</td>
<td>PE3</td>
<td>I</td>
<td>Analog</td>
<td>Analog-to-digital converter input 0.</td>
</tr>
<tr>
<td>AIN1</td>
<td>7</td>
<td>PE2</td>
<td>I</td>
<td>Analog</td>
<td>Analog-to-digital converter input 1.</td>
</tr>
<tr>
<td>AIN2</td>
<td>8</td>
<td>PE1</td>
<td>I</td>
<td>Analog</td>
<td>Analog-to-digital converter input 2.</td>
</tr>
<tr>
<td>AIN3</td>
<td>9</td>
<td>PE0</td>
<td>I</td>
<td>Analog</td>
<td>Analog-to-digital converter input 3.</td>
</tr>
<tr>
<td>AIN4</td>
<td>64</td>
<td>PD3</td>
<td>I</td>
<td>Analog</td>
<td>Analog-to-digital converter input 4.</td>
</tr>
<tr>
<td>AIN5</td>
<td>63</td>
<td>PD2</td>
<td>I</td>
<td>Analog</td>
<td>Analog-to-digital converter input 5.</td>
</tr>
<tr>
<td>AIN6</td>
<td>62</td>
<td>PD1</td>
<td>I</td>
<td>Analog</td>
<td>Analog-to-digital converter input 6.</td>
</tr>
<tr>
<td>AIN7</td>
<td>61</td>
<td>PO0</td>
<td>I</td>
<td>Analog</td>
<td>Analog-to-digital converter input 7.</td>
</tr>
<tr>
<td>AIN8</td>
<td>60</td>
<td>PE5</td>
<td>I</td>
<td>Analog</td>
<td>Analog-to-digital converter input 8.</td>
</tr>
<tr>
<td>ATN11</td>
<td>57</td>
<td>PB5</td>
<td>I</td>
<td>Analog</td>
<td>Analog-to-digital converter input 11.</td>
</tr>
</tbody>
</table>

Figure 6.1: ATD channel – pin assignments (Table 13-1 of the datasheet)

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 6.2, 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.

<table>
<thead>
<tr>
<th>Sequencer</th>
<th>Number of Samples</th>
<th>Depth of FIFO</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SS2</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>SS1</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>SS9</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

Figure 6.2: ATD Sequencer list (Table 13-2 of the datasheet)

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.

6.3 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 4 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 4. 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 to configure the PCTL register to tell the TM4C which, of the many peripherals available, we would like to use as the alternate function. It can be seen in Figure 6.4, that writing 1-15 to one of the PCTL 4 bit groups will enable a specific alternate digital function. Writing zero the appropriate 4 bits enables the analog alternate function. (A full list of alternate functions per pin is provided in Table 23-5 of the datasheet.)

Next, set the direction of PE3 to input by setting bit 3 to 0 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.

\[\text{Figure 6.3: The GPIO PCTL register}\]

\[\text{Figure 6.4: Partial table of GPIO PCTL alternate function assignments}\]
6.4 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.

![Figure 6.5: The ADC Active Sample Sequencer register](image)

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 (seen later). 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 6.6: ADC Sample Sequence Control 3 register](image)

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.

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, clear bits 3:0 of the ADC Sample Sequence Input Multiplexer Select 0 register (ADCSSMUX3, offset 0x0A0) to select channel AIN0.

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.
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.

6.5 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.

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.
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).

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).
6.6 Procedure:

Before the lab, draw a flowchart of a program you will write to convert an analog input applied to pin PE3 to its digital value and output the results to Termite every 1s. This flowchart should show the method to convert the digital value from the ATD converter to a BCD representation to be used in the display. The results should be displayed following the format “Signal: X.Y Volts”. Show the value to the number of digits that approximate the resolution of the ATD system.

During the lab, program the ATD conversion system on the board to convert the analog signal to a 12-bit number between 0x00 and 0xFFF. Then convert this number to a value between 0.0 – 3.3 Volts, represented as an ASCII string, including the decimal point. Use this ASCII value to populate the results string in RAM and print the string using OutStr.

Test your software by connecting the ATD inputs to GND, to VDD, and then to the analog signal from the power supply, adjusting the power and observing the changing output voltage in Termite. Demo this to your TA.

Questions:

1. If VRH = 3.3V, what digital value is returned when the ATD system converts the following voltage using 12-bit conversion?
   - 3.2 V
   - -0.1 V
   - 2.6 V
   - 1.5 V
   - 3.6 V

2. What will be the digital value when the ATD system converts the above voltages using VRH = 3.25V? What is the conversion resolution for this case? From this digital value, what is the voltage estimate that would minimize the maximum error of the measurement/calculation, i.e. \( V_{\text{min-max}} \)

6.7 Lab report:

For the lab write-up, include

1. Your flowcharts that you wrote before the lab and your program.
2. A copy of your working .s files.
3. A brief discussion of the objectives of the lab and the procedures performed in the lab.
4. Answers to any questions in the discussion, procedure, or question sections of the lab.