Lab 8: Using Analog Devices with an AVR

CompEng 3151: Digital Engineering Lab II

Last revised: July 16, 2019 (BJZ, RJS)

Description/Overview

The "real" world we are living in is analog. Whether it is sound, light, temperature or humidity, all are continuously varying things. However, when we want to communicate with the digital world, it is crucial to use digital values which can be easily recognized by the computing systems. The computing system understands this by converting them into binary numbers.

So for transferring external continuous information (analog information) into a digital/computing system, we must convert them into integer (digital) values. This type of conversion is carried out by Analog to Digital Converter (ADC). The process of converting an analog value into digital value is known as Analog to Digital Conversion. In short, analog signals are real world signals around us like sound and light.

During this lab, we will use the ADC on the AVR to read the analog sensors on a joystick and use the values to calibrate a display. The values you get and calibration will be displayed (you guessed it) on the LCD display.

Objectives

Students will:

- Read data from an analog sensor device using the ADC
- Display the data from the analog sensor on an LCD display

Materials Required

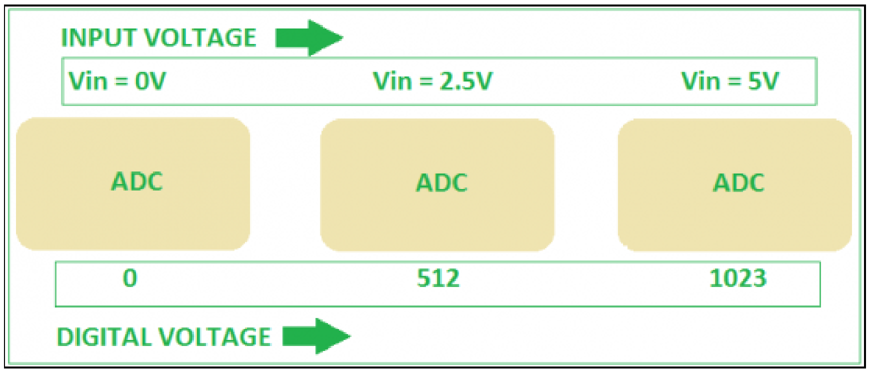
- Atmel Studio 7 Installed on PC (ECE CLC Computers will have this)
- AVR Simon Board with a USB cable
- 2x16 Character LCD Module
- Connection Wires
- Joystick

Preparation

In order to be successful with this lab, students should review Chapter 13 in the Mazidi textbook.

Background *(excerpted from electroschematics.com - used by permission of author T.K. Hareendran)*

Digital signals are analog equivalents in digital or numeric format which are well understood by digital systems like microcontrollers. ADC is one such hardware which measures analog signals and produces a digital equivalent of the same signal. AVR microcontrollers have an inbuilt ADC facility to convert analog voltage into an integer. The AVR converts an analog voltage into a 10-bit number in the range 0 to 1023.



Note that we have an Analog Reference (Aref) Voltage also, which will be considered equivalent to 1023 and any voltage value less than this Aref will have less number than 1023. The input range is 0 to Aref and the digital output is 0-1023. Here 0V will be equal to 0, and Aref/2 will be equal to 512 and so on.

ADC in ATmega324

Now you have the basics of ADC, let us move to the inbuilt ADC of AVR microcontrollers. First of all, note that the ADC is multiplexed with Port A and the ADC can be operated in single conversion mode and free-running mode. In single conversion mode, the ADC does a single conversion and stops. But in free-running mode, the ADC is continuously converting (i.e. it does a conversion and then starts the next conversion instantly after that). Here are a few concepts with regards to Atmega8 to know beforehand:

- ADC Prescaler: The ADC needs a clock pulse for the job and for this the system clock is divided by a number (2, 4, 16, 32, 64 and 128) to get the lesser frequency (ADC requires a frequency between 50KHz to 200KHz)
- ADC Channels: The ADC in Atmega324PB has 8 channels, allows you to take samples from 8 different pins
- ADC Registers: Register provides the communication link between CPU and the ADC. You can configure the ADC according to your need using these registers. The ADC has 3 registers only:
  1. ADC Multiplexer Selection Register – ADMUX: For selecting the reference voltage and the input channel
  2. ADC Control and Status Register A – ADCSRA: It has the status of ADC and is also used to control it
  3. The ADC Data Register – ADCL and ADCH: Final result of the conversion is stored here
- AVCC: This pin supplies power to ADC. AVCC must not differ more than ± 0.3V from Vcc
- AREF: Another pin which can optionally be used as an external voltage reference pin.

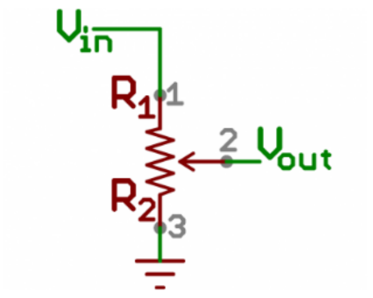
- Voltage Resolution: This is the smallest voltage increment which can be measured. For 10-bit ADC, there can be 1024 different voltages (for an 8-bit ADC, there can be 256 different voltages)

In principle, ADC in AVR microcontrollers uses a technique known as successive approximation by comparing input voltage with half of the reference voltage generated internally. The comparison continues by dividing the voltage further down and updating each bit in ADC register by 1 if input voltage is high, 0 otherwise. This process lasts 10 times (for 10-bit ADC) and generates resulting binary output.

Joystick

In this lab, you're going to interface a joystick to the AVR. A joystick contains two analog devices – a potentiometer for each axis of movement (x & y). These potentiometers can be measured using an analog read when they are connected as voltage dividers.

You may have learned about potentiometers in your circuits course. Internal to the potentiometer is a single resistor and a wiper, which cuts the resistor in two and moves to adjust the ratio between both halves. Externally there are usually three pins: two pins connect to each end of the resistor, while the third connects to the pot's wiper.

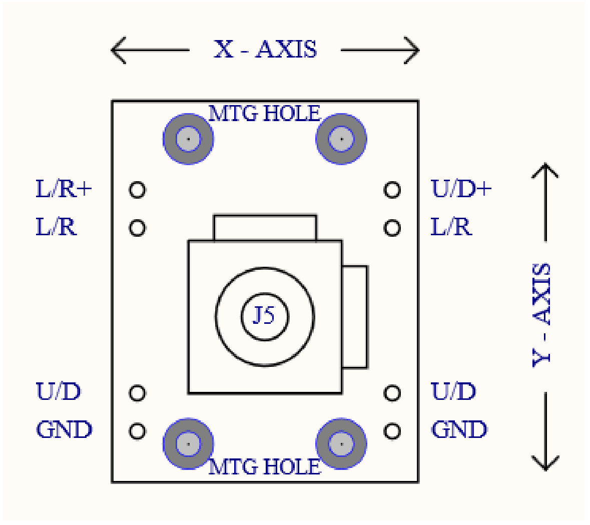


If the outside pins connect to a voltage source (one to ground, the other to Vin), the output (Vout at the middle pin will mimic a voltage divider. Turn the pot all the way in one direction, and the voltage may be zero; turned to the other side the output voltage approaches the input; a wiper in the middle position means the output voltage will be half of the input.

Potentiometers come in a variety of packages, and have many applications of their own. They may be used to create a reference voltage, adjust radio stations, measure position on a joystick, or in tons of other applications which require a variable input voltage.

Procedure

1) Obtain a joystick from your TA. You are going to connect it up so that L/R+ AND U/D+ pins are connected to 3.3v from the AVR Simon Board, one of the ground pins is connected to the ground on the board, and the x- and y-axis pins are connected to two of the ADC input pins on Port A (PA0 & PA1 should work). Connecting it this way, you have created two voltage



divider circuits that can be measured with the ADC on the AVR.

2) Create a program that allows you to read the x- and y-values from the ADC. Your program will output these values to the LCD display.

3) Once you have the range of values for your joystick, you will modify your program to output what direction the joystick is oriented based on the values (i.e. N, NE, E, SE, S, SW, W, NW). The display should also continue to show the coordinates of the joystick.

Deliverables

As you have in the past, your solution to the problem should be thoroughly documented and demonstrated to your TA. Make sure your code also includes a documented configuration of how the hardware is connected (i.e. what pins are connected to what devices and what they are used for). A picture of the hardware setup/wiring would also be helpful. In the lab report, the questions/observations below should be included in the results or conclusions section of your report.

Questions/Observations

1. What are the two factors that affect the step size of the ADC?
2. What is the resolution of the ADC in the configuration it is being used for this lab?

References

- Muhammad Ali Mazidi, Sarmad Naimi, and Sepehr Naimi. 2010. *AVR Microcontroller and Embedded Systems: Using Assembly and C (1st ed.).* Prentice Hall Press, Upper Saddle River, NJ, USA.
- LCD Module datasheet
- Hareendran, T.K. *ATmega8 Advanced Guide: VR Analog to Digital Conversion (ADC) – Tutorial #13.* Accessed from https://www.electroschematics.com/10053/avr-adc/ Accessed on January 18, 2019.