AVR Analog to Digital Conversion (ADC) – Tutorial #13

T.K. HAREENDRAN

AVR tutorial

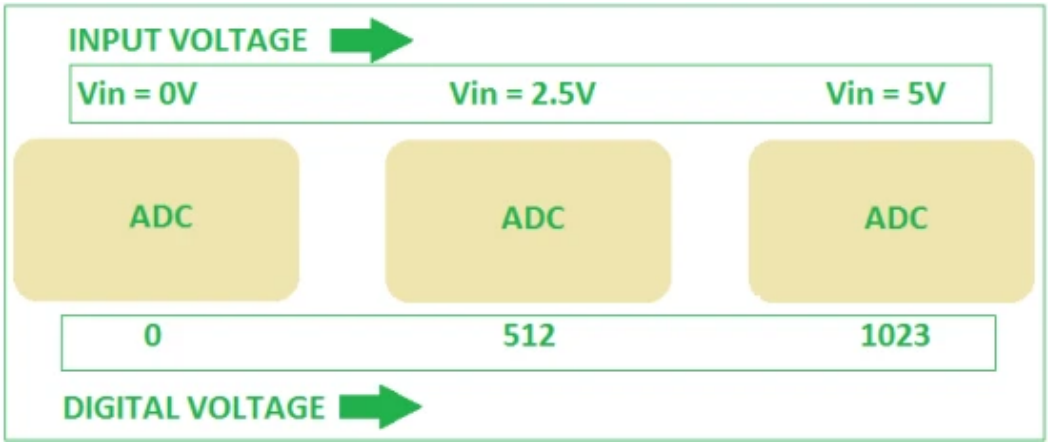
Share this:

Did You Know? 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.

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 has inbuilt ADC facility to convert analog voltage into an integer. AVR convert it into 10-bit number of 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-Aref and 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 Atmega8

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 C, 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, ie it does a conversion and then start the next conversion instantly after that. Here are a few concepts with regards to Atmega8 to know beforehand:

- ADC Prescaler: The ADC needsa 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 Atmega8 PDIP package has 6 channels, allows you to take samples from 6 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.

A short break to the boring theory! Let's start a simple test program to see our ADC in action. In this example, Atmega8 reads a 10K potmeter connected to PORT C and turns on (or off) the LEDs connected to PORT D. Here, the ADC is used in single conversion mode with 10 bit precision. Wire the hardware as shown, and feed the code to Atmega8. When the work is finished, slowly turn the 10K potmeter and check the result. As coded, if the value is less than 512 then LED1 (at PD6) should be turned on, else LED2 (at PD7).

(This is the right time to grab the Atmega8 datasheet again, for a jump to the ADC chapter. Just open the ADC chapter and carefully read description of every register, plus all about the ADC circuitry)