LAB 08
Voltmeter Design (Open Ended Lab)

EE-222 Microprocessor Systems

Contents

1 Administrivia **2**
- 1.1 Open-Ended-Lab . . . 2
- 1.2 Learning Outcomes . . . 2
- 1.3 Deliverable . . . 2

2 Problem Statement **3**

3 Brief Introduction to AVR Analog-to-Digital Converter (ADC) **3**
- 3.1 Analog signal quantization . . . 3
- 3.2 AVR ADC specifications . . . 4
- 3.3 AVR ADC Pins . . . 4
- 3.4 ATmega16 ADC Registers . . . 4
  - 3.4.1 ADC Multiplexer Selection Register (ADMUX) . . . 4
  - 3.4.2 ADC Control and Status Register A (ADCSRA) . . . 5
  - 3.4.3 ADC Data Register . . . 5
  - 3.4.4 Steps to perform A-to-D conversion . . . 5
- 3.5 Additional information . . . 6

4 Lab Task **7**
- 4.1 Task A . . . 7
- 4.2 Task B . . . 7

1 Administrivia

1.1 Open-Ended-Lab

An open-ended lab is where students are given the freedom to develop their own solution, instead of merely following the already set guidelines from a lab manual or elsewhere. The teacher gives the students an objective/purpose and not the procedure. The students would then have to come up with their own solution to fulfill the purpose.

1.2 Learning Outcomes

By the end of this lab you will be able to:

1. Solve daily life problems through microcontroller.
2. Develop a voltmeter using ADC.
3. Propose your engineering solution to the problem.

1.3 Deliverable

You are required to submit

- Appropriately Commented code.
- Answer to the questions asked in the lab tasks.
- Issues in Developing the Solution and your Response

in the beginning of next lab

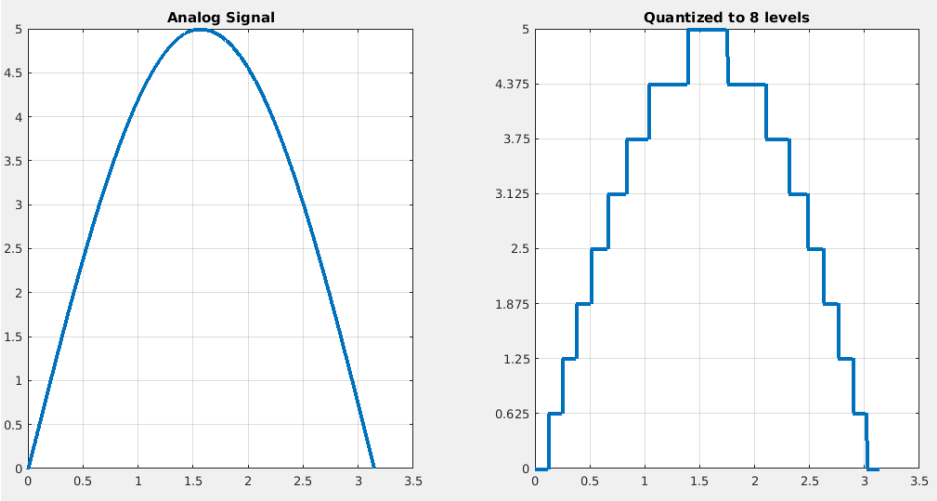
2 Problem Statement

In this lab you are required to build a digital voltmeter using ATmega16 of range `0-5 volts` and give answers to the questions asked in the sections below.

3 Brief Introduction to AVR Analog-to-Digital Converter (ADC)

3.1 Analog signal quantization

An analog signal of range `0-5 volts` can have infinite values of voltages between the two boundaries (for example `3.5555V, 2.18V, 1.7123V, etc.`). Microcontrollers/Microprocessors have limited memory so they cannot store these infinite values (infact no possible method on earth can). However digital systems perform sampling and quantization of analog signal in order to capture and store them. The range of voltage is divided into `n` levels and each level is assigned a decimal value from `0-n`. This process is called quantization.



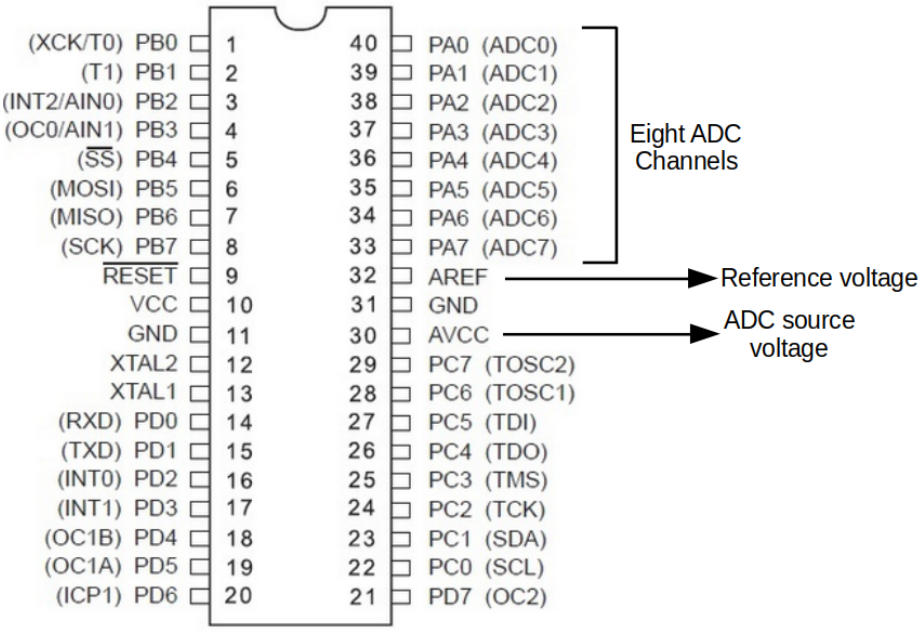
For example, in the figure above, the analog signal of range `0-5V` is divided into 8 levels each of `0.625V`. The microcontroller, stores the voltage lavel from `0-to-0.625V` as decimal `0` in its memory, voltage level `0.625-to-1.25V` as decimal `1` in its memory and so on. The number of bits required to save decimal values of eight voltage intervals (–that is from `0-7`) is 3-bits because $2^3 - 1 = 7$. This conversion is performed by a piece of circuitry called ADC (Analog to Digital Converter). The resolution of any ADC is defined through its sample data width. In previous example the sample data width was 3-bits, that's why the voltage range was divided into 8 intervals. If the sample data width of an ADC is 4-bits, it will divide the voltage range to $2^4 = 16$ intervals.
The sample data width of ATmega16 is 10-bits, so it can divide a given voltage range to $2^{10} = 1024$ **levels. Hence each level will represent** $5/1024 = 4.88mV$.

3.2 AVR ADC specifications

- 10-bit Resolution
- Maximum 260μs conversion time
- Eight single ended input channels (one active at a time)
- For maximum accuracy operates at `50-200kHz`.

3.3 AVR ADC Pins

- **ADC0-ADC7:** Eight analog input channels. Only one can be used at a time in single input mode.
- **AREF:** Reference voltage for the ADC. The ADC volatge division intervals are created from `0V` to voltage applied at `AREF` pin. Voltage greater than V_{cc} of microcontroller cannot be applied.
- **AVCC:** The power to ADC is given by applying `5V` (of the same source as connected to V_{cc}) at this pin.



3.4 ATmega16 ADC Registers

3.4.1 ADC Multiplexer Selection Register (ADMUX)

REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0

REFS1	REFS0	**Functionality**
0	0	Apply reference voltage at AREF pin from outside.
0	1	Connect AREF pin to AVCC internally without outside connection.
1	0	Reserved.
1	1	Use internal `2.56V` voltage generator as reference.

The value of `MUX4:MUX0` signifies which channel is currently being used for ADC. For example 00000_2 means ADC0 pin is under use, 00001_2 means ADC1 pin is under use and so on.

3.4.2 ADC Control and Status Register A (ADCSRA)

ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0

Use `ADPS2:ADPS0` to apply prescalar to your microcontroller's clock frequency and feed the resulting frequency to ADC circuitry. Keep in mind that the ADC works best when the frequency fed to it is in range `50-200kHz`.

ADPS2	ADPS1	ADPS0	**Prescalar factor**
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

`ADEN` bit when raised, the microcontroller power ups the ADC circuitry. `ADSC` bit when raised, the voltage at the selected ADC channel is captured by the ADC circuitry and the conversion of this sample to decimal value between (`0-1023`) starts. When the circuitry finishes conversion, this bit is automatically turned 0, which indicated the completion of Analog-to-digital conversion and we can fetch the converted value from ADC data register. Rest of the bits are related to ADC interrupts.

3.4.3 ADC Data Register

When ADC circuitry completes the analog-to-digital conversion, the decimal result is stored in this register for the user to fetch from. The name of register itself is `ADC` and it is 16-bits wide (–that is, `ADC = ADCH:ADCL`).

3.4.4 Steps to perform A-to-D conversion

1. Set the reference voltage in `ADMUX`.
2. Enable ADC through `ADCSRA`.
3. Set prescalar such that resultant frequency is within `50-200kHz` range through `ADCSRA`.

4. Select the analog channel through `ADMUX`. (Keep in mind that you should not disturb other bits in ADMUX while choosing channel).

5. Start conversion by setting `ADSC` bit in `ADCSRA`.

6. Poll on `ADSC` bit to check when it becomes zero.

7. Read the value from `ADC` register.

3.5 Additional information

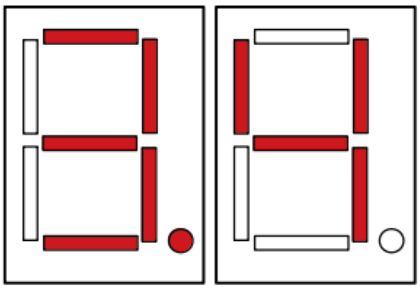
- AVR microcontroller and embedded systems using assembly and C by Mazidi et al. Chapter 13.
- ATmega16 Data Sheet[1] page `204-221`.
- "The ADC of the AVR"[2], article by MaxEmbedded.

[1] http://ww1.microchip.com/downloads/en/devicedoc/doc2466.pdf
[2] http://maxembedded.com/2011/06/the-adc-of-the-avr/

4 Lab Task

4.1 Task A

Design a (`0-5V` range) digital voltmeter using ATmega16 ADC channel zero and display your readings on a 2-digit (units and a decimal point digit) seven segment display.For example `3.4` should be displayed as,



Hint-1: Since, AVR is an 8-bit architecture without an FPU (Floating Point Unit) so probably your IDE will not support "float" data type. You must use "division (/)" and "modulo (%)" function to calculate the integer and decimal part of your resultant captured voltage.
Hint-2: The voltage captured by ADC circuitry will be transformed to some value between `0-1023`. It is your task to map this conversion back to voltage range of 0-5V in order to display it.

4.2 Task B

Build the given three circuits using all same resistances and measure V_x using the voltmeter you made. Also calculate V_x theoretically using circuit analysis techniques and compare results. Include the images of readings from your voltmeter and the calculations in your lab report.