EMBEDDED SYSTEMS LAB 1

Fall Semester 2018

Lab Experiment Lab 6 – Analog to Digital Converter (ADC)

Author of the report: Mian Muhammad Taimour.

Mailbox No: 773

Experiment conducted by: Mian Muhammad Taimour, Muhammad Mumshad Hassan, Pavli Mele

Place of execution: Research I

Date of execution: Tuesday 2nd October 2018

INTRODUCTION

In this lab our main task is to create analog input for the ATmega328 using a potentiometer. To do this we will use the Analog-to-Digital converter of the ATmega to read this analog values and use them to change the value of the pins in port D and light 3 LEDs connected to it. The Analog-to-Digital converter converts an input voltage into a 10-bit digital value by successive approximation. The lowest value will represent ground and the maximum value a reference voltage that we will choose to be the 5V from the VCC pin.

PRELAB

REGISTERS

We Will use the following registers to our advantage when writing the code for this experiment:

ADC

This is the data register for the Analog to Digital Conversion. This register will store the value after the conversion has been done. The register will be blocked for the converter until the value has been read, which means that it has to be read in order for the converter to keep converting values. It is a 16-bit register divided into ADCL and ADCH.

ADCSRA

The ADC Control and Status Register A will be in charge of enabling/disabling Analog-to-Digital conversions, starting them, enabling their associated interrupt and selecting the converter's prescaler. The 7th bit (ADEN) will enable/disable the Analog-to-Digital converter, the 6th bit is used to start the conversion (ADSC), the 5th bit (ADATE) is used to enable triggering the conversion automatically as specified in ADCSRB, the 4th bit (ADIF) is the interrupt flag and will be set when the converter has finished converting a value to trigger the interrupt, the 3rd bit (ADIE) enables an interrupt on interrupt vector ADC vect whenever a conversion is complete if the I-bit is also set in SREG. The remaining three bits (ADPS2, ADPS1, ADPS0) will determine the prescaler used by the converter; their decimal equivalent (d) will determine a prescaler of 2^d.

ADMUX

The ADC Multiplexer Selection Register will be used to select the input channel for the ADC conversion and to select the Voltage Reference used to do the conversion (which will give ADC = 1024· V in V ref). The 7th and 6th bit (REFS1 and REFS0) are the reference selection bits and will select the following (according to the data sheet):

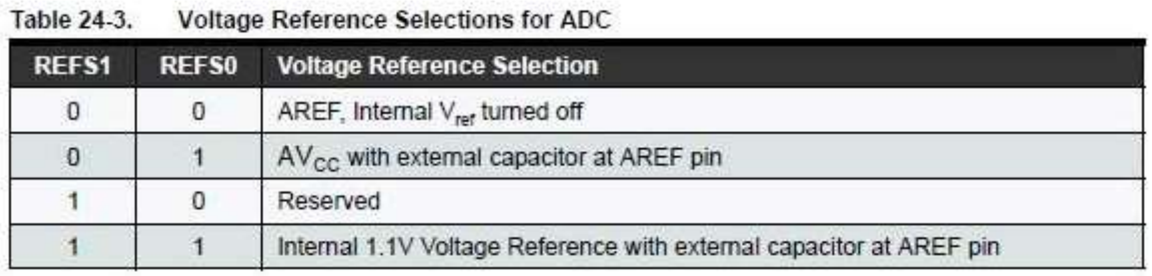


Table 24-3. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal V_{ref} turned off
0	1	AV_{CC} with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 1.1V Voltage Reference with external capacitor at AREF pin

Bits 3, 2, 1 and 0 (MUX3, MUX2, MUX1, MUX0) will determine the input to the ADC. Writing values from 0 to 8 (in binary) will select pins ADC0 to ADC8 respectively as inputs.

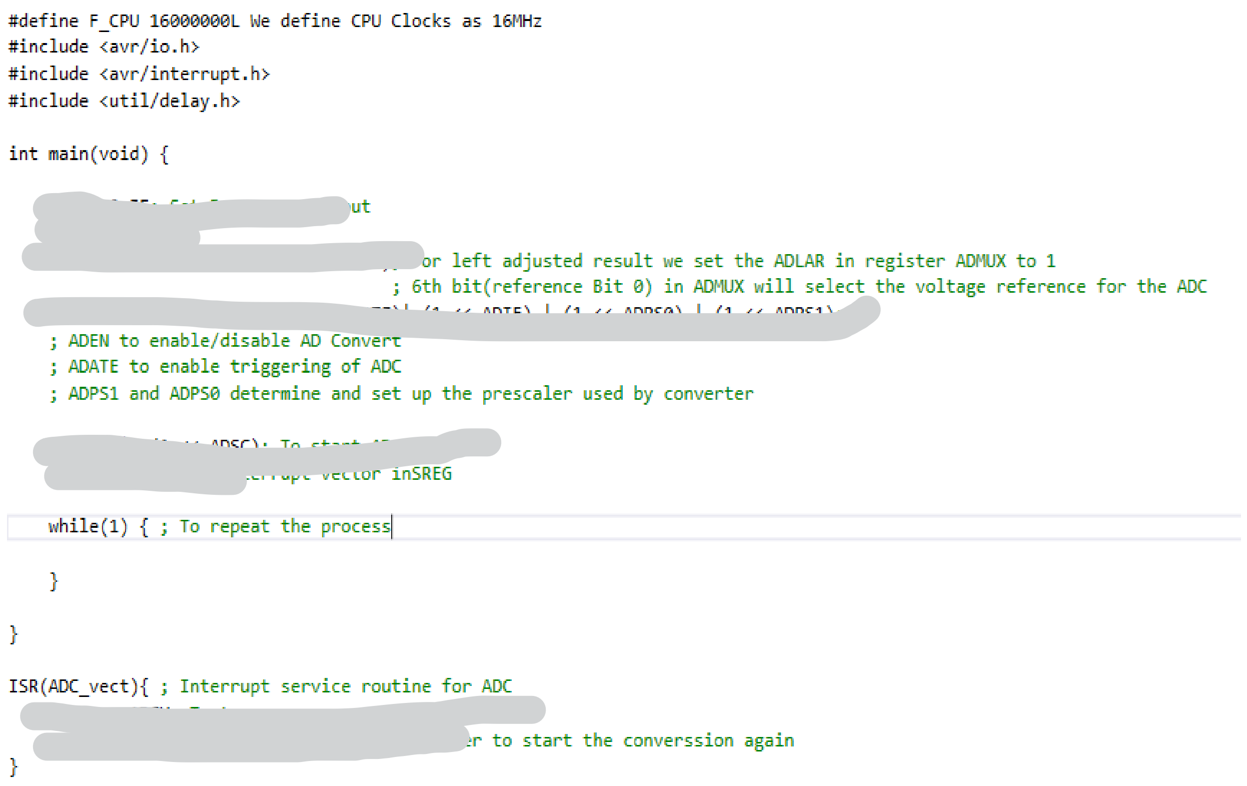
ADC PRE SCALER

We want a conversion which's frequency lies between 50kHz and 200kHz. In order to achieve this, we use the biggest possible prescaler which is 2^7 = 128. Because the clock frequency of the ATmega328 is 16, 000,000hz, we will get frequency of = 16,000,000/128. 128 = 156250Hz. This is within the desired range. To set this prescaler, we convert 7 to binary first: 710 = 1112. This is the value that we should write in ***ADCSRA***.

Design your program to convert an analog voltage input to a 10 bits value by ADC inturrupt.

First we set the ADSC to start all AD Converts. We set it to 1 to start the first conversion. Then, in the ISR (Interrupt Service Routine for ADC), the ADCL register was read first, followed by the ADCH register. ADCL must be read first because when ADCL is read, the ADC data register is not updated until ADCH is read. This way, the value from the ADCL register contains bits 0 to 7 and the value from ADCH contains bits 8 and 9.

CODE



```
#define F_CPU 16000000L We define CPU Clocks as 16MHz
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

int main(void) {

    [redacted]ut
    [redacted]or left adjusted result we set the ADLAR in register ADMUX to 1
                                    ; 6th bit(reference Bit 0) in ADMUX will select the voltage reference for the ADC
    [redacted]
    ; ADEN to enable/disable AD Convert
    ; ADATE to enable triggering of ADC
    ; ADPS1 and ADPS0 determine and set up the prescaler used by converter

    [redacted]terrupt vector inSREG

    while(1) { ; To repeat the process

    }

}

ISR(ADC_vect){ ; Interrupt service routine for ADC
    [redacted]r to start the converssion again
}
```

CODE EXPLANATION

Firstly, we checked the Arduino Schematic and then designed our circuit as shown in the diagram below. In our lab experiment, we will set up the circuit by connecting three LEDs to PORTD Pin 5,6 and 7 in our case, combined with three 220 Ohm Resistors in series. Our goal is to design a program in order to the light up three LEDs.
Our goal is to use a Potentiometer to control the input voltage. Then, we will write a code to convert an input voltage into 16 bits value and Output any 8 bits out of these 12 bits to PORTD. We will connect 3 pins of PORTD to 3 LEDs and observe the AD converting results.
As we turn the potentiometer, we will be able to see and analyse that the LEDs will light up one by one in a binary counting sequence.

First, global interrupts are enabled, the ADC interrupt and the converter are also enabled. A prescaler of 128 is selected by setting ADPS2, ADPS1 and ADPS0 in ADCSRA all to 1. This achieves a conversion frequency of 125000 Hz. Inside the loop, port D will receive the value of the previous conversion. When the interrupt is called, the value of the conversion is stored, and a new conversion is initiated. Then, we set the bit ADSC to start every single AD convert. In Free Running mode, we write this bit to one to start the first conversion. Lastly, We enable the Global Interrupt pins using sei(). After this we use the interrupt service routine to call the interrupt and a new conversion ins initiated.

CIRCUIT DESIGN

We designed the following circuit to carry out our lab experiment:

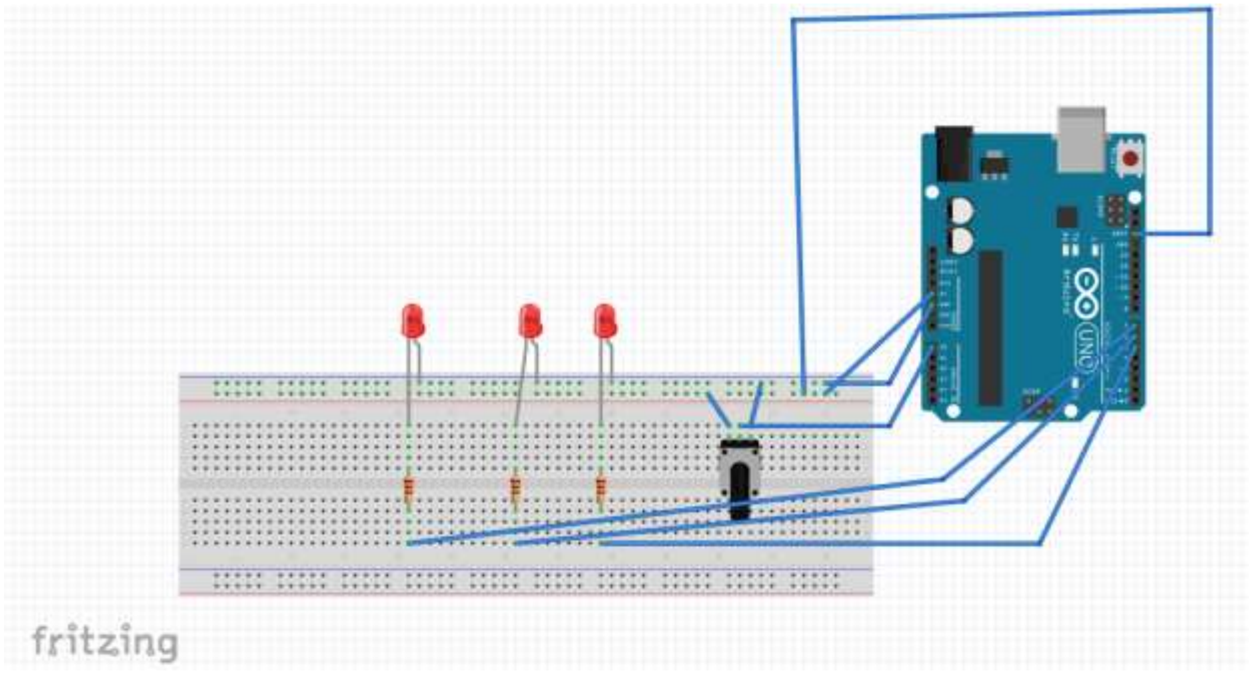
MATERIALS

The materials used for the circuit were the following:

- Arduino UNO (ATmega328)
- 3 220 Ohm resistor
- 3 LED’s
- Breadboard
- Jumper Wires
- Potentiometer

SETUP

Schematics using Fritzig



REFERECES

- http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A48PA-datasheet_Complete.pdf1.
- http://www.microchip.com/design-centers/8-bit/avr-mcus.
- http://embsys-fhu.user.jacobs-university.de/.

- http://www.avr-tutorials.com/digital/about-avr-8-bit-microcontrollers-digital-ioports.
- https://en.wikipedia.org/wiki/Atmel_AVR_instruction_set .
- https://sites.google.com/site/qeewiki/books/avr-guide/timers-on-the-atmega328 .