Experiment-8

Aim: Hands-on experimentation of ATMega32 ADC programming in C.

Objectives: After successful completion of this experiment, students will be able to,

- Use C language for ATMega32 microcontroller programming on AVRStudio.
- Experiment with ADC of ATMega32 on ATMega32 AVR Development Board. Students will be able to interface the LM35 sensor to the ATMega32.

Equipment required:

- Windows7 or later based host computer
- ATMega32 Development Board
- USBasp Programmer
- Jumper Wires
- LCD

Software required:

- AVR Studio7 installation setup
- USBasp driver installation setup

Theory:

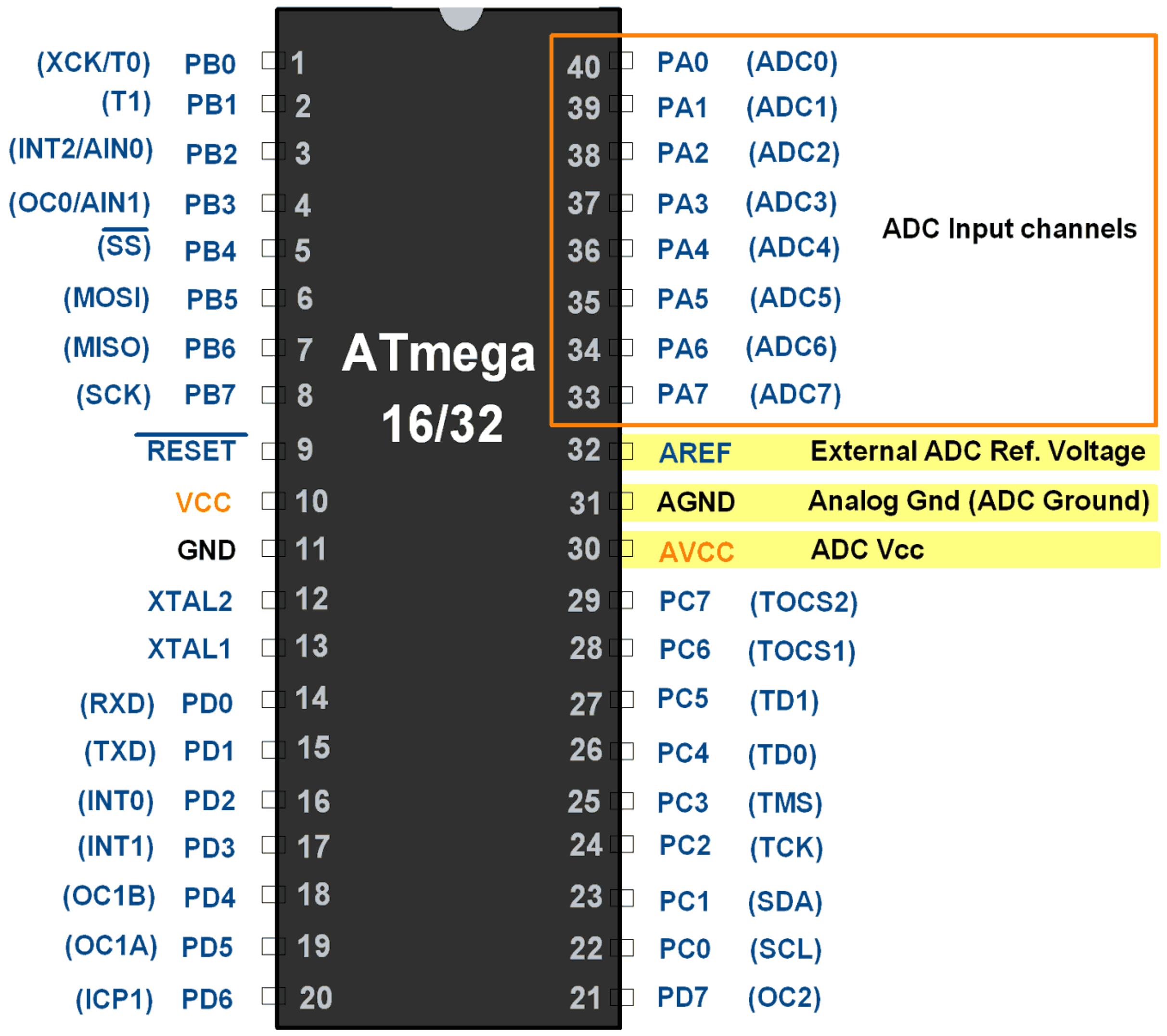
An ADC, or Analog to Digital Converter, allows one to convert an analog voltage to a digital value that can be used by a microcontroller. There are many sources of analog signals that one might like to measure. There are analog sensors available that measure temperature, light intensity, distance, position, and force, just to name a few.

The AVR ADC allows the AVR microcontroller to convert analog voltages to digital values with few to no external parts. The ATmega32 features a 10-bit successive approximation ADC.ATmega32 has 8 channel ADC at Port A i.e. from pin 33 to 40. The ADC has a separate analog supply voltage pin, AVCC. AVCC must not differ more than ± 0.3V from VCC.. The voltage reference may be externally decoupled at the AREF pin. AVCC is used as the voltage reference. The ADC can also be set to run continuously (the free-running mode) or to do only one conversion.

For the conversion, use the below formula

$$ADC = \frac{Vin * 1024}{Vref}$$

Where Vin is the voltage on the selected input pin and Vref the selected voltage reference.



The controller has 10 bit ADC, which means we will get digital output 0 to 1023.

i.e. When the input is 0V, the digital output will be 0V & when input is 5V (and Vref=5V), we will get the highest digital output corresponding to 1023 steps, which is 5V.

So controller ADC has 1023 steps and

- **Step size with Vref=5V : 5/1023 = 4.88 mV.**
- **Step size with Vref=2.56 : 2.56/1023 = 2.5 mV.**

So Digital data output will be D*out*= Vin / step size.

ADC Register

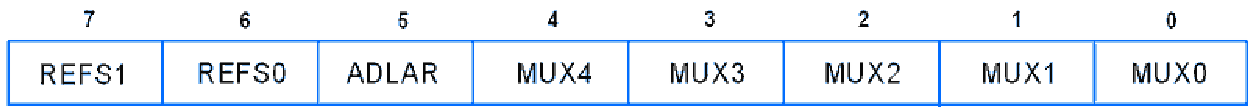
In AVR ADC, we need to understand four main register -

1. **ADCH:** Holds digital converted data higher byte
2. **ADCL:** Holds digital converted data lower byte
3. **ADMUX:** ADC Multiplexer selection register
4. **ADCSRA:** ADC Control and status register

ADCH: ADCL register

First, two-register holds the digital converted data, which is 10-bit.

ADMUX Register



7	6	5	4	3	2	1	0
REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0

Bit 7: 6 – REFS1 : 0: Reference Selection Bits

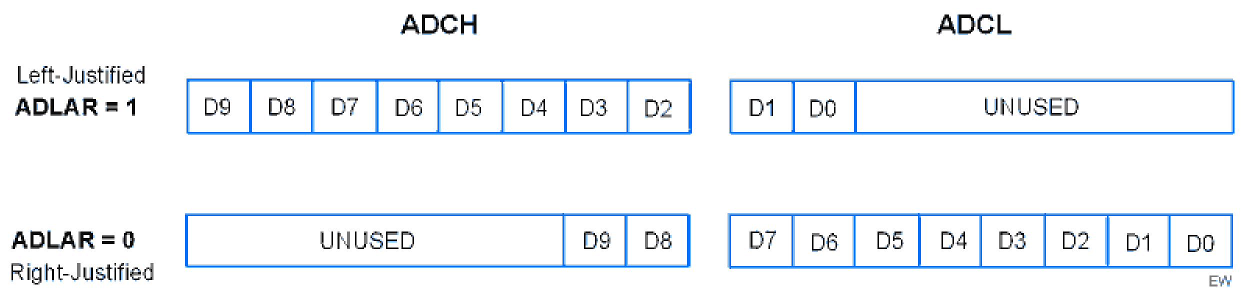
Reference voltage selection for ADC

REFS1	**REFS0**	**Vref to ADC**
0	0	AREF pin

0	1	AVCC pin i.e. Vcc 5 V
1	0	Reserved
1	1	Internal 2

Bit 5 – ADLAR: ADC Left Adjust Result

Use 10-bits output as upper bits or lower bits in ADCH & ADCL.



Bits 4 : 0 – MUX4 : 0: Analog Channel and Gain Selection Bits

We can select input channel ADC0 to ADC7 by using these bits. These bits are also used to select comparator (inbuilt in AVR) inputs with various gain. We will cover these comparator operations in another part.

Selecting a channel is very easy, just put the channel number in MUX4 : 0.

Suppose you are connecting the input to ADC channel 2 then put 00010 in MUX4 : 0.

Suppose you are connecting the input to ADC channel 5 then put 00101 in MUX4 : 0.

ADCSRA Register:

7	6	5	4	3	2	1	0
ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0

- **Bit 7 – ADEN: ADC Enable**

Writing one to this bit enables the ADC. By writing it to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress, will terminate this conversion.

- **Bit 6 – ADSC: ADC Start Conversion**

Writing one to this bit starts the conversion.

- **Bit 5 – ADATE: ADC Auto Trigger Enable**

Writing one to this bit, results in Auto Triggering of the ADC is enabled.

- **Bit 4 – ADIF: ADC Interrupt Flag**

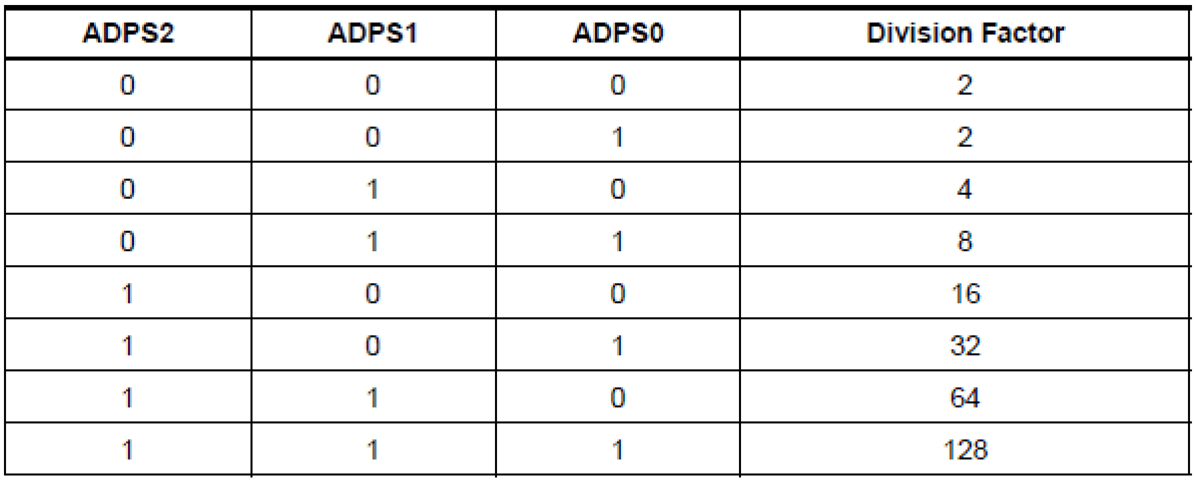
This bit is set when an ADC conversion completes and the Data Registers are updated.

- **Bit 3 – ADIE: ADC Interrupt Enable**

Writing one to this bit, the ADC Conversion Complete Interrupt is activated.

- **Bits 2 : 0 – ADPS2 : 0: ADC Prescaler Select Bits**

These bits determine the division factor between the XTAL frequency and the input clock to the ADC



ADPS2	ADPS1	ADPS0	Division 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

We can select any divisor and set frequency Fosc/2, Fosc/4, etc. for ADC, But in AVR, ADC requires an input clock frequency less than 200KHz for max. accuracy. So we have to always take care of not exceeding ADC frequency more than 200KHz.

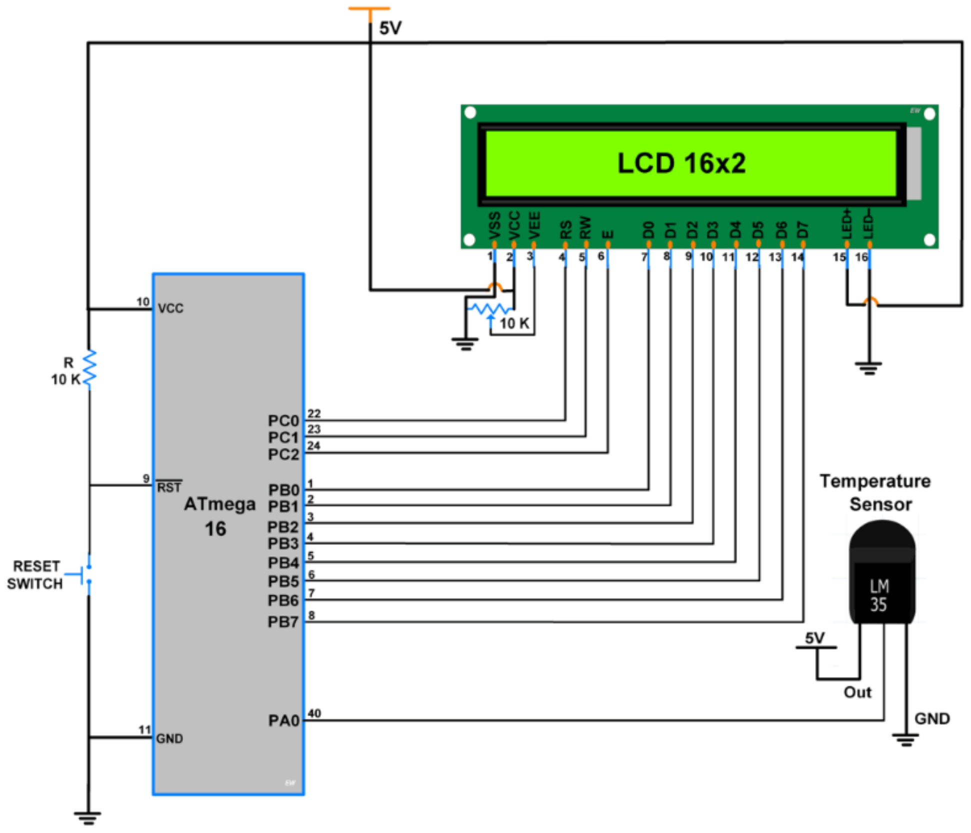
Suppose your clock frequency of AVR is 8MHz, then we must have to use divisor 64 or 128. Because it gives 8MHz/64 = 125KHz, which is lesser than 200KHz.

Steps to Program ADC

1. Make the ADC channel pin as an input.
2. Set ADC enable bit in ADCSRA, select the conversion speed using ADPS2 : 0. For example, we will select devisor 128.
3. Select ADC reference voltage using REFS1: REFS0 in ADMUX register, for example, we will use AVcc as a reference voltage.
4. Select the ADC input channel using MUX4 : 0 in ADMUX, for example, we will use channel 0.
5. So our value in register ADCSRA = 0x87 and ADMUX = 0x40.
6. Start conversion by setting bit ADSC in ADCSRA. E.g.ADCSRA |= (1<<ADSC);
7. Wait for conversion to complete by polling ADIF bit in ADCSRA register.
8. After the ADIF bit gone high, read ADCL and ADCH register to get digital output.
9. Notice that read ADCL before ADCH; otherwise result will not be valid.

Here, we are using the LM35 sensor which is an on-chip peripheral of our AVR development board. As it is an analog sensor and microcontroller is digital, this interfacing requires helps of an ADC.

Interfacing Diagram of LM35 analog sensor with AVR development board.



CODE:

```
#define F_CPU 8000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <string.h>
#include <stdio.h>
#include "LCD_16x2_H_file.h"

#define degree_sysmbol 0xdf

voidADC_Init(){
	DDRA = 0x00;		/* Make ADC port as input */
	ADCSRA = 0x87;		/* Enable ADC, with freq/128  */
	ADMUX = 0x40;		/* Vref: Avcc, ADC channel: 0 */
}


intADC_Read(char channel)
```

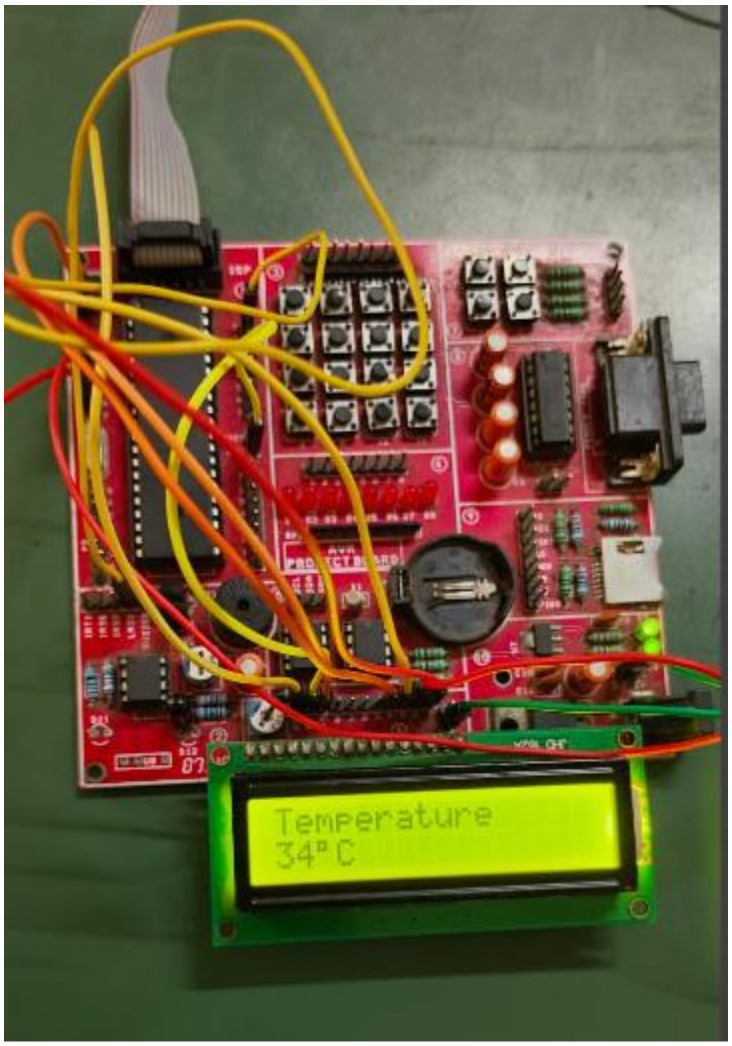
```
{
        ADMUX = 0x40 | (channel & 0x07);   /* set input channel to read */
        ADCSRA |= (1<<ADSC);               /* Start ADC conversion */
        while (!(ADCSRA & (1<<ADIF)));     /* Wait until end of conversion by polling ADC
interrupt flag */
        ADCSRA |= (1<<ADIF);               /* Clear interrupt flag */
        _delay_ms(1);                      /* Wait a little bit */
        return ADCW;                       /* Return ADC word */
}


int main()
{
        char Temperature[10];
        floatcelsius;

        LCD_Init();                 /* initialize 16x2 LCD*/
        ADC_Init();                 /* initialize ADC*/

        while(1)
        {
        LCD_String_xy(1,0,"Temperature");
        celsius = (ADC_Read(0)*4.88);
        celsius = (celsius/10.00);
        sprintf(Temperature,"%d%cC  ", (int)celsius, degree_sysmbol);/* convert integer value to
ASCII string */
        LCD_String_xy(2,0,Temperature);/* send string data for printing */
          _delay_ms(1000);
        memset(Temperature,0,10);
        }
}
```

OUTPUT:



CONCLUSION:

By performing this experiment I came to know about the On-board temperature sensor programming and its interfacing with LCD.

Experiment- 9 Post Lab Exercise

Student Name: Aryan Langhanaoja
Enrollment No: 92200133030

Answer the following questions:

1) How many ADC channels are there in ATMega32? Which channel we have used in the experiment?

- ➢ The ATMega32 microcontroller has 8 ADC channels. The specific channel used in an experiment would depend on the configuration and requirements of the experiment.

2) What is the resolution of ADC in ATMega32?

- ➢ The resolution of the ADC in ATMega32 is 10 bits.

3) In an ADC, input is **analog** (digital, analog) and output is **digital** (digital, analog).
4) In the A/D of ATMega32, what should be the Vref value be if we want a step size of 4mV?

- ➢ To achieve a step size of 4mV in the ADC of ATMega32, the Vref (reference voltage) should be set to 4.096V.

5) Which of the following ADC sizes provides the best resolution?
   (a) 8 bit
   (b) 10 bit
   (c) 12 bit
   (d) Bit size does not matter.

- ➢ (c) 12 bit ADC size provides the best resolution.