

Embedded System on AVR Microcontroller (ATMEGA32)

Exp8: ADC in ATMEGA32

Submitted by
Ronit Dutta, MS in IOT and Signal Processing
Department of Electrical Engineering, IIT Kharagpur

Under the guidance of
Aurobinda Routray, Professor, Electrical Engineering



Department of Electrical Engineering
Indian Institute of Technology Kharagpur
February, 2025

• Introduction to ADC

Analog-to-digital converters are among the most widely used devices for data acquisition. Digital computers use binary (discrete) values, but in the physical world everything is analog (continuous).



Temperature, pressure (wind or liquid), humidity and velocity are a few examples of physical quantities that we deal with every day. A physical quantity is converted to electrical (voltage, current) signals using a device called a transducer. Transducers are also referred to as sensors. Sensors for temperature, velocity, pressure, light, and many other natural quantities produce an output that is voltage (or current). Therefore, we need an analog-to-digital converter to translate the analog signals to digital numbers so that the microcontroller can read and process them.

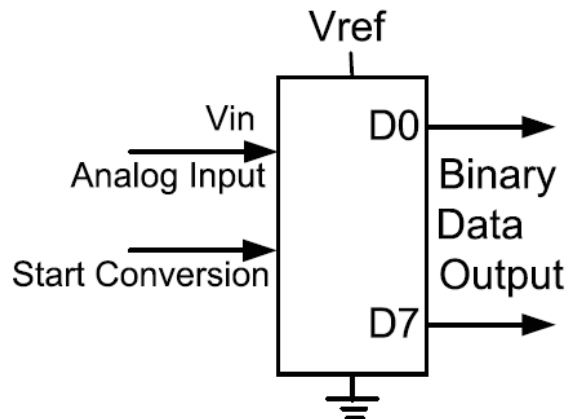
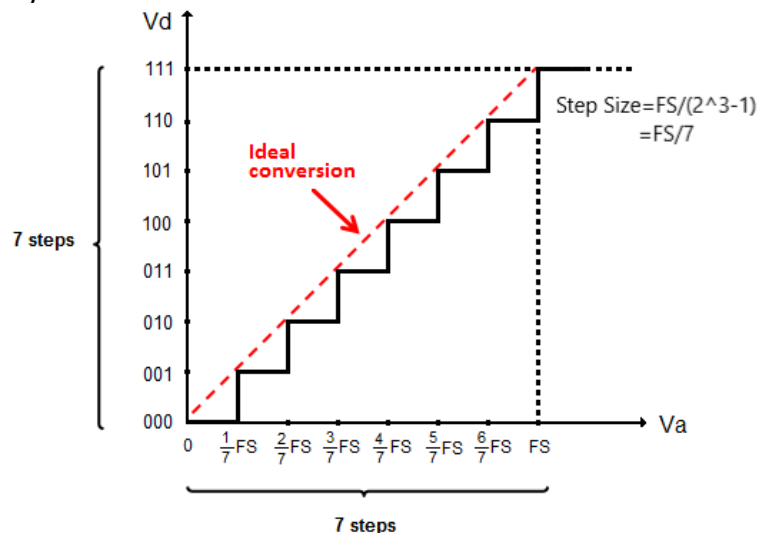


Fig. An 8-bit ADC Block Diagram

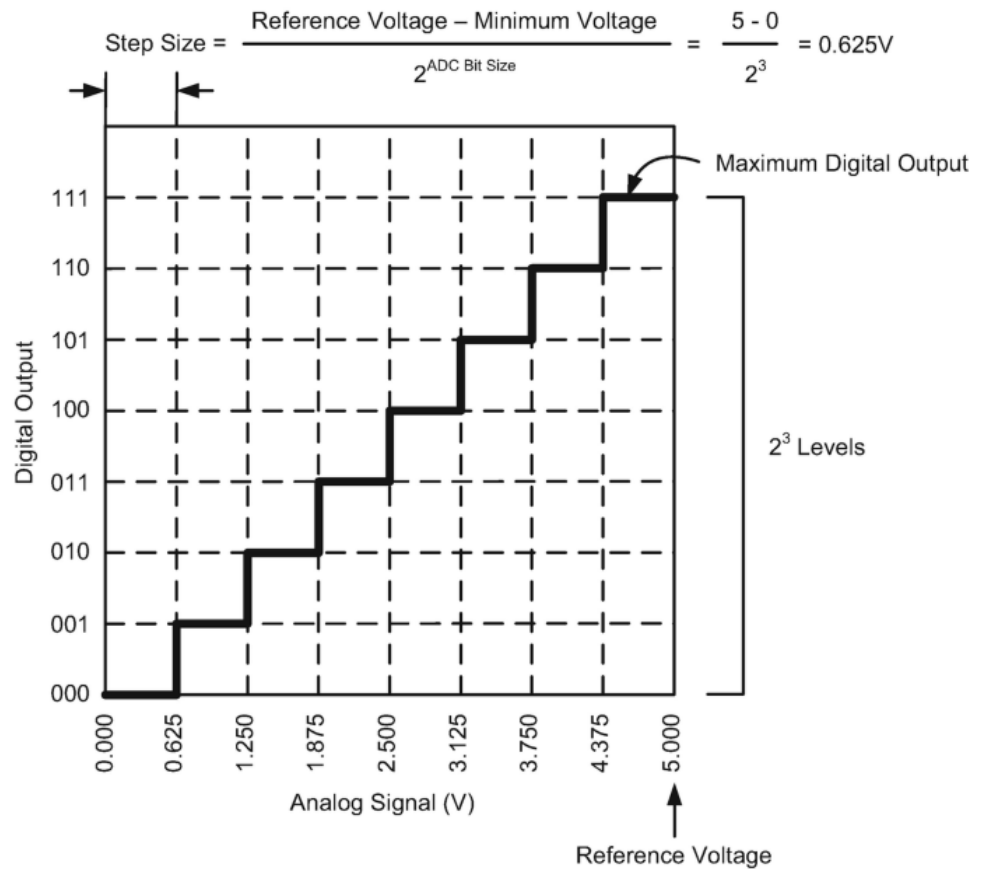
• Some of the major characteristics of the ADC

- I. **Resolution:** Step size (resolution) is the smallest change that can be discerned by an ADC.



The above figure is the ideal conversion with step size for ADC. Here, $V_{ref}=FS$ and resolution is $FS/(2^n-1)$. For 3-bit ADC, the ideal resolution is $FS/7$. We all know that we cannot apply analog voltage more than reference voltage of the ADC. So, here if we apply exactly FS as analog voltage then only we can achieve 111 as digital conversion bits.

To overcome this situation, the resolution of n -bit ADC is defined as $V_{ref}/2^n$. The below figure represents the practical 3-bit ADC resolution with corresponding analog voltage with converted digital bits.



The ADC has n -bit resolution, where n can be 8, 10, 12, 16, or even 24 bits. Higher-resolution ADCs provide a smaller step size, where step size is the smallest change that can be discerned by an ADC. Some widely used resolutions for ADCs are shown in below Table.

n-bit	Number of Steps	Step Size (mV)
8	256	19.53
10	1024	4.88
12	4096	1.22
16	65536	0.076

Table: Resolution versus Step Size for ADC ($V_{ref}=5V$)

- II. **Conversion time:** In addition to resolution, conversion time is another major factor in judging an ADC. Conversion time is defined as the time it takes the ADC to convert the analog input to a digital (binary) number. The conversion time is dictated by the clock source connected to the ADC in addition to the method used for data conversion and technology used in the fabrication of the ADC chip such as MOS or TTL technology.
- III. **V_{ref}:** V_{ref} is an input voltage used for the reference voltage. The voltage connected to this pin, along with the resolution of the ADC chip, dictate the step size. For an 8-bit ADC, the step size is V_{ref}/256 because it is an 8-bit ADC, and 2 to the power 8 gives us 256 hence 256 steps. For example, if the analog input range needs to be 0 to 4volts, V_{ref} is connected to 4 volts. That gives $4V/256 = 15.625mV$ for the step size of an 8-bit ADC. In another case, if we need a step size of 10mV for an 8-bit ADC, then V_{ref}=2.56V, because $2.56V/256=10mV$.

V _{ref} (V)	V _{in} (V)	Step Size (mV)
5.00	0 to 5	$5/1024= 4.88$
4.096	0 to 4.096	$4.096/1024= 4$
3.3	0 to 3.3	$3.3/1024= 3.22$
2.56	0 to 2.56	$2.56/1024= 2.5$

Table: V_{ref} Relation to V_{in} Range for an 10-bit ADC

- IV. **Digital Data Output:** In an 8-bit ADC we have an 8-bit digital data output of D0–D7, while in the 10-bit ADC the data output is D0–D9. To calculate the output voltage, we use the following formula:

$$D_{out} = \frac{V_{in}}{step\ size}$$

Where D_{out}= digital data output (in decimal), V_{in}= analog input voltage, and step size (resolution) is the smallest change, which is V_{ref}/1024 for an 10-bit ADC.

• ADC Programming in the ATMEGA32

Because the ADC is widely used in data acquisition, in recent years an increasing number of microcontrollers have had an on-chip ADC peripheral, just like timers and USART. An on-chip ADC eliminates the need for an external ADC connection, which leaves more pins for other I/O activities. The vast majority of the AVR chips come with ADC. In this section we discuss the ADC feature of the ATmega32 and show how it is programmed in Assembly.

ATmega32 ADC features:

The ADC peripheral of the ATmega32 has the following characteristics:

- It is a 10-bit ADC.

- 0 - V_{CC} ADC Input Voltage Range.
- It has 8 analog input channels, 7 differential input channels, and 2 differential input channels with optional gain of 1x, 10x and 200x.
- The converted output binary data is held by two special function registers called ADCL (A/D Result Low) and ADCH (A/D Result High).
- Because the ADCH:ADCL registers give us 16 bits and the ADC data out is only 10 bits wide, 6 bits of the 16 are unused. We have the option of making either the upper 6 bits or the lower 6 bits unused. When the lower 6 bits are unused then it is called 8bit ADC mode and the converted data will be stored in ADCH.
- We have three options for V_{ref} . V_{ref} can be connected to AVCC (Analog V_{CC}), internal 2.56 V reference, or external AREF pin.
- The conversion time is dictated by the crystal frequency connected to the XTAL pins (F_{osc}) and ADPS0:2 bits. 13 μs - 260 μs Conversion Time and up to 15 kSPS at Maximum Resolution.

• ADC Registers in the ATMEGA32

In the AVR microcontroller five major registers are associated with the ADC that we deal with in this experiment. They are **ADMUX (ADC multiplexer selection register)**, **ADCSRA (ADC Control and Status Register)**, **ADCH (high data)**, **ADCL (low data)**, and **SPIOR (Special Function I/O Register)**. We examine each of them in this section.

i. ADMUX (ADC Multiplexer Selection Register)

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

REFS1,REFS0 (Bit 7,Bit 6): Reference Selection Bits

These bits select the reference voltage for the ADC.

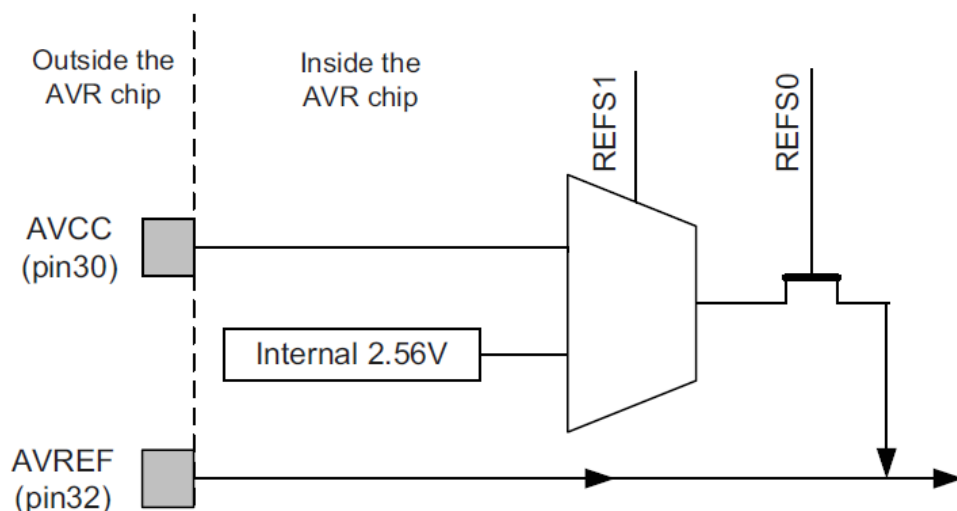


Fig. ADC Reference Source Selection

Above figure shows the block diagram of internal circuitry of V_{ref} selection. As we can see that we have three options: (a) AREF pin, (b) AVCC pin, or (c) internal 2.56V. The below table shows how REFS1 and REFS0 bits of the ADMUX register can be used to select the V_{ref} source.

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal Vref turned off
0	1	AVCC with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

Table: Voltage Reference Selections for ADC

ADLAR (Bit 5): ADC Left Adjust Results

The AVR has a 10-bit ADC, which means that the result is 10 bits long and cannot be stored in a single byte. In AVR two 8-bit registers are dedicated to the ADC result, but only 10 of the 16 bits are used and 6 bits are unused. You can select the position of used bits in the bytes. This ADLAR bit dictates either the left bits or the right bits of the result registers ADCH:ADCL that are used to store the result. If we write a one to ADLAR, the result will be left adjusted; otherwise, the result is right adjusted.

	ADCH	ADCL
Left-Justified		
ADLAR = 1	D9 D8 D7 D6 D5 D4 D3 D2	D1 D0 UNUSED
ADLAR = 0	UNUSED D9 D8	D7 D6 D5 D4 D3 D2 D1 D0
Right-Justified		

MUX4:0 (Bit 4:0): Analog Channel and Gain Selection Bits

The value of these bits selects the gain for the differential channels and also selects which combination of analog inputs are connected to the ADC.

MUX4...0	Single-ended Input
00000	ADC0
00001	ADC1
00010	ADC2
00011	ADC3
00100	ADC4
00101	ADC5
00110	ADC6
00111	ADC7

ii. ADCSRA (ADC Control and Status Register A)

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The ADCSRA register is the status and control register of ADC. Bits of this register control or monitor the operation of the ADC.

ADEN (Bit 7): ADC Enable

This bit enables or disables the ADC. Setting this bit to one will enable the ADC, and clearing this bit to zero will disable it even while a conversion is in progress.

ADSC (Bit 6): ADC Start Conversion

To start each conversion, you have to set this bit to one.

ADATE (Bit 5): ADC Auto Trigger Enable

When this bit is written to one, Auto Triggering of the ADC is enabled. The ADC will start a conversion on a positive edge of the selected trigger signal. The trigger source is selected by setting the ADC Trigger Select bits, ADTS in SFIOR.

ADIF (Bit 4): ADC Interrupt Flag

This bit is set when an ADC conversion completes and the Data Registers are updated. The ADC Conversion Complete Interrupt is executed if the ADIE bit and the I-bit in SREG are set. ADIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ADIF is cleared by writing a logical one to the flag.

ADIE (Bit 3): ADC Interrupt Enable

When this bit is written to one and the I-bit in SREG is set, the ADC conversion Complete Interrupt is activated.

ADPS2:0 (Bit 2:0): ADC Prescaler Select Bits

These bits determine the division factor between the oscillator frequency of the microcontroller 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

A/D conversion time

By using the ADPS2:0 bits of the ADCSRA register we can set the A/D conversion time. To select the conversion time, we can select any of $F_{osc}/2$, $F_{osc}/4$, $F_{osc}/8$, $F_{osc}/16$, $F_{osc}/32$, $F_{osc}/64$, or $F_{osc}/128$ for ADC clock, where F_{osc} is the speed of the crystal frequency connected to the AVR chip. For the AVR, the ADC requires an input clock frequency less than 200KHz for the maximum accuracy.

Example: An AVR is connected to the 8 MHz crystal oscillator. Calculate the ADC frequency for (a) ADPS2:0 = 001 (b) ADPS2:0 = 100 (c) ADPS2:0 = 111

Solution:

- (a) Because ADPS2:0 = 001 (1 decimal), the $ck/2$ input will be activated; we have $8 \text{ MHz} / 2 = 4 \text{ MHz}$ (greater than 200 kHz and not valid)
- (b) Because ADPS2:0 = 100 (4 decimal), the $ck/8$ input will be activated; we have $8 \text{ MHz} / 16 = 500 \text{ kHz}$ (greater than 200 kHz and not valid)
- (c) Because ADPS2:0 = 111 (7 decimal), the $ck/128$ input will be activated; we have $8 \text{ MHz} / 128 = 62 \text{ kHz}$ (a valid option since it is less than 200 kHz)

A timing factor that we should know about is the acquisition time. After an ADC channel is selected, the ADC allows some time for the sample-and-hold capacitor (C hold) to charge fully to the input voltage level present at the channel.

In the AVR, the first conversion takes 25 ADC clock cycles in order to initialize the analog circuitry and pass the sample-and-hold time. Then each consecutive conversion takes 13 ADC clock cycles.

Below table lists the conversion times for some different conditions. Notice that sample-and-hold time is the first part of each conversion.

Condition	Sample & Hold (Cycles from Start of Conversion)	Conversion Time (Cycles)
First conversion	13.5	25
Normal conversions, single ended	1.5	13
Auto Triggered conversions	2	13.5
Normal conversions, differential	1.5/2.5	13/14

Table: ADC Conversion Time

If the conversion time is not critical in your application and you do not want to deal with calculation of ADPS2:0 you can use ADPS2:0 = 111 to get the maximum accuracy of ADC.

iii. ADCH: ADCL registers

After the A/D conversion is complete, the result sits in registers ADCL(A/D Result Low Byte) and ACDH (A/D Result High Byte).

As we mentioned before, the ADLAR bit of the ADMUX is used for making it right-justified or left-justified because we need only 10 of the 16 bits.

iv. SFIOR (Special Function IO Register – SFIOR)

Bit	7	6	5	4	3	2	1	0	
	ADTS2	ADTS1	ADTS0	–	ACME	PUD	PSR2	PSR10	SFIOR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

ADTS2:0 (Bit 7:5): ADC Auto Trigger Source

If ADSC in ADCSRA is written to one, the value of these bits selects which source will trigger an ADC conversion. If ADSC is cleared, the ADTS2:0 settings will have no effect.

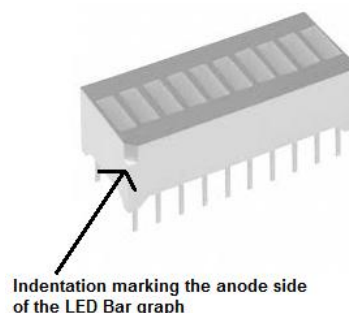
ADTS2	ADTS1	ADTS0	Trigger Source
0	0	0	Free Running mode
0	0	1	Analog Comparator
0	1	0	External Interrupt Request 0
0	1	1	Timer/Counter0 Compare Match
1	0	0	Timer/Counter0 Overflow
1	0	1	Timer/Counter1 Compare Match B
1	1	0	Timer/Counter1 Overflow
1	1	1	Timer/Counter1 Capture Event

Reserved Bit (Bit4): This bit is reserved for future use in the ATmega32. For ensuring compatibility with future devices, this bit must be written zero when SFIOR is written.

The remaining SFIOR bits are not used in ADC experiment.

*Since the General Purpose GPIO is covered, so to display the digital conversion the LED bar graph is used **or** 10 LEDs can be used instead of the LED bar graph.*

How to know which sides pins are anode and cathode respectively?



Experiment1: /* Analog to Digital conversion with Right Justified and display the digital conversion on LED bar graph */

```
.INCLUDE "M32DEF.INC"
.ORG 0x0000

//Data direction register of PORTB and PORTD
LDI R16,0xFF
OUT DDRB,R16
OUT DDRD,R16

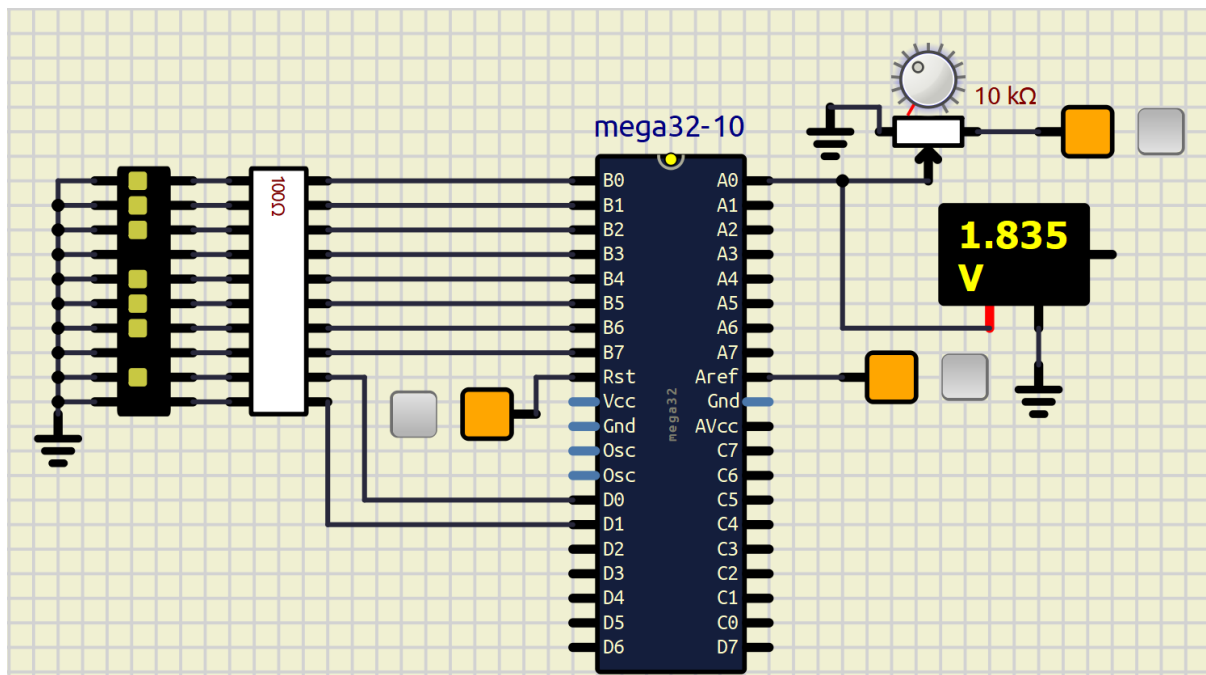
//stack declaration
LDI R16,HIGH(RAMEND)
OUT SPH,R16
LDI R16,LOW(RAMEND)
OUT SPL,R16

// ADC Initialization
LDI R16,0x00
OUT ADMUX,R16

LDI R16,0xC3
OUT ADCSRA,R16

MAIN:  SBI  ADCSRA,ADSC
        CONV:  SBIC  ADCSRA,ADSC
                RJMP CONV //Jump over next instruction if portbit clear
        IN  R16,ADCL
        //IN R17,ADCH
        OUT PORTB,R16
        OUT PORTD,R17
        JMP MAIN
```

Make the below circuit for simulation on SimulIDE.



Right Adjusted Video Demonstration: <https://drive.google.com/file/d/1TnfWoh-2xP3MBVLoudyCIAutnYd8aPBo/view?usp=sharing>

Verify the digital conversion with corresponding analog voltage by hand calculation.

Vref=5Volt, ADC Bits=10bit, Analog Applied Voltage=1.835Volt

ADC Value= $(1024/5) * 1.835 = 375.808$

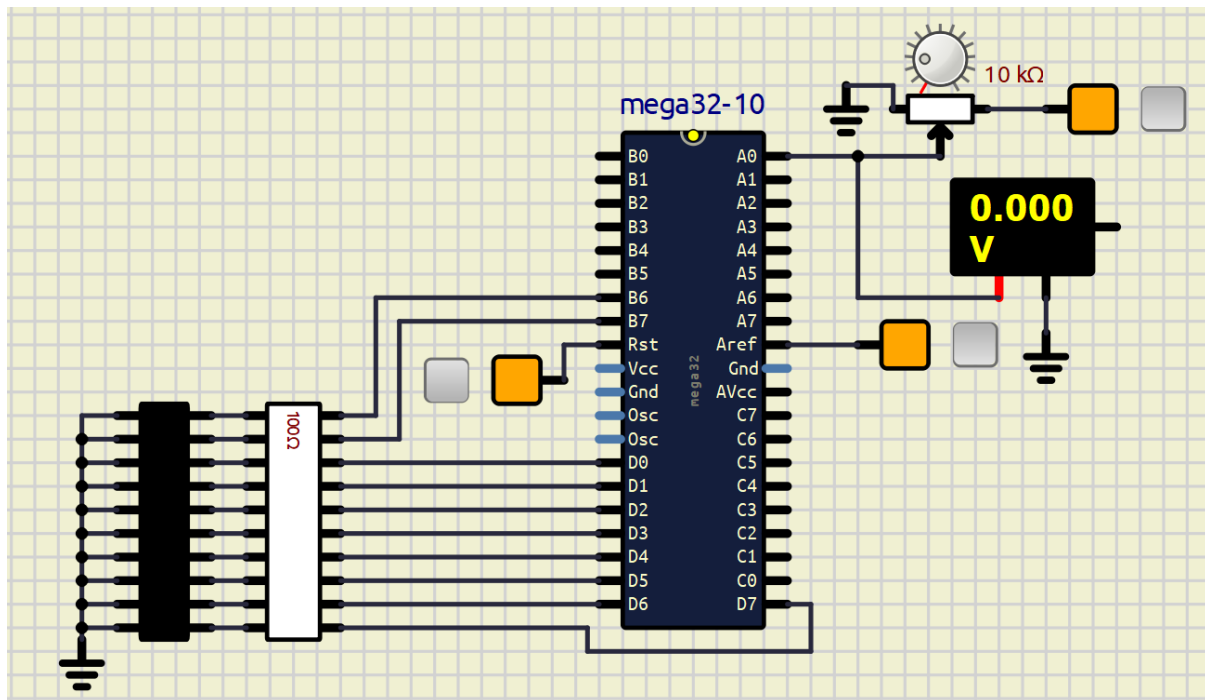
Since the ADC follows the floor converted digital value. Hence,

ADC Value= $\text{int}(375.808) = 375$

The corresponding 10-bit binary value is 0101110111

Note: The resolution of the ADC is 4.88mV. To show this minimum analog input change the 4-decimal Voltmeter will be required. Since here 3-decimal voltmeter is present so we cannot verify the LSB value of the analog input. After hand conversion there may be 1-bit error due to this.

Class Assignment 1: Write Assembly Code for Analog to Digital conversion with Left Justified and display the digital conversion on LED bar graph as shown in the below figure. Verify the digital conversion with corresponding analog voltage by hand calculation.



Left Adjusted Video Demonstration:

<https://drive.google.com/file/d/1CsWjE2xZg3AN0T4JI4BznG-l2zKvf4wc/view?usp=sharing>

After simulation, make the above two circuits on Hardware and verify these.



After learning LCD and UART, you can verify the ADC conversion through these.

Experiment2: /* Write a program for the AVR for ADC and transfer the ADC Value serially through UART at 9600 baud, continuously. Oscillator Frequency=8MHz, U2X=0, StopBit=1, No Parity Bit. The data will be printed on realterm horizontally. */

```
.INCLUDE "M32DEF.INC"
.ORG 0x0000

LDI R16,HIGH(RAMEND)
OUT SPH,R16
LDI R16,LOW(RAMEND)
OUT SPL,R16
// ADC Initialization

LDI R16,0x00
OUT ADMUX,R16

LDI R16,0xC3
OUT ADCSRA,R16

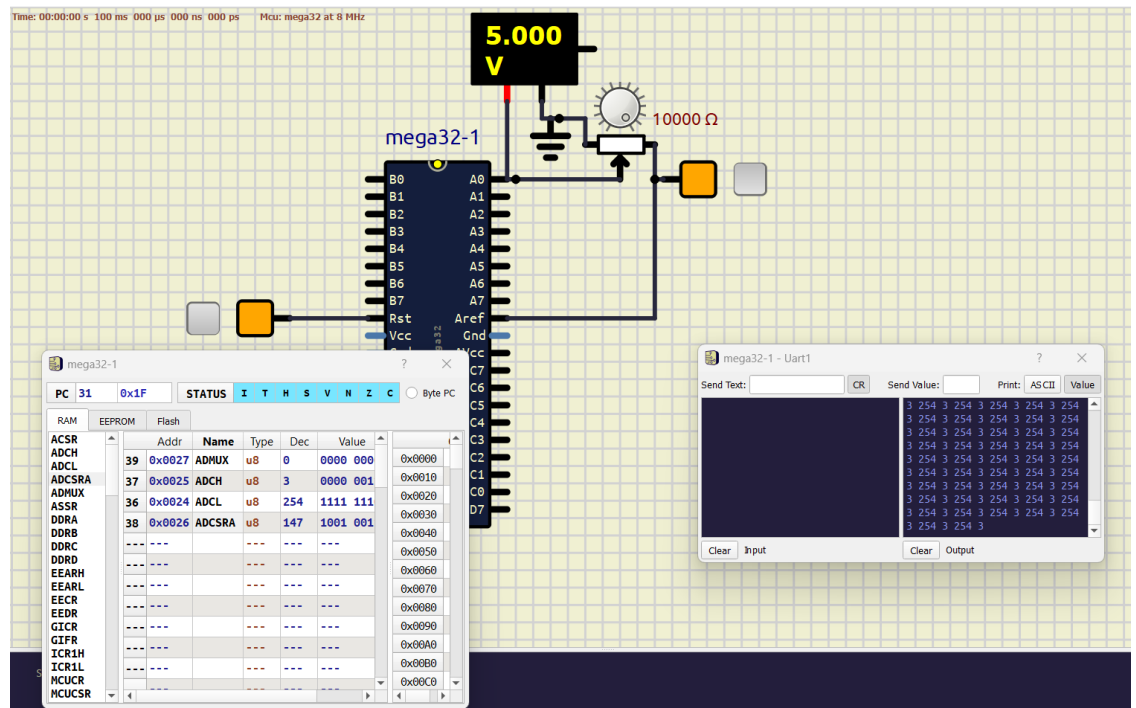
//UART Initialization

SBI DDRD,PIND1 // Set Tx as Output Pin
SBI PORTD,PIND1

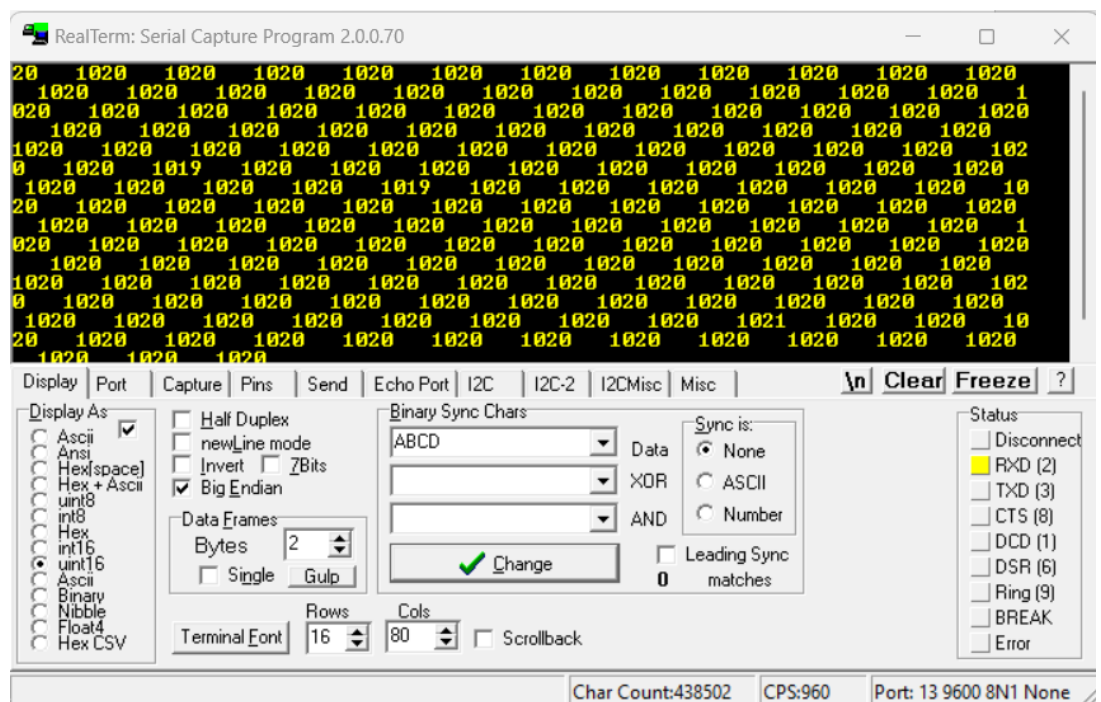
//UART Initialization
LDI R16,0x08// Enabling Tx Rx
OUT UCSRB,R16
LDI R17,0x86// 8 bit data mode
OUT UCSRC,R17
//9600bps Baud-Rate Settings for 8MHz Oscillator
LDI R16,51
OUT UBRRL,R16

MAIN:      SBI ADCSRA,ADSC
           CONV: SBIC ADCSRA,ADSC
           RJMP CONV //Jump over next instruction if portbit clear
           IN R16,ADCL
           IN R17,ADCH
           OUT UDR,R17
           AGAIN1: SBIS UCSRA, UDRE
                   RJMP AGAIN1
           OUT UDR,R16
           AGAIN2: SBIS UCSRA, UDRE
                   RJMP AGAIN2
           JMP MAIN
```

Make the below circuit for simulation on SimulIDE.



Make the above hardware and UART hardware connection to display the result on realterm as shown in below figure. Set the display as unsigned int16 and verify the 10-bit ADC value ranging from 0 to 1023.





Experiment3: /* Write a program for the AVR for ADC and transfer the ADC Value serially through UART at 9600 baud, continuously. Oscillator Frequency=8MHz, U2X=0, StopBit=1, No Parity Bit. The data will be printed on realterm vertically i.e. new line program. */

```
.INCLUDE "M32DEF.INC"
.ORG 0x0000

LDI R16,HIGH(RAMEND)
OUT SPH,R16
LDI R16,LOW(RAMEND)
OUT SPL,R16
// ADC Initialization

LDI R16,0x00
OUT ADMUX,R16

LDI R16,0xC3
OUT ADCSRA,R16

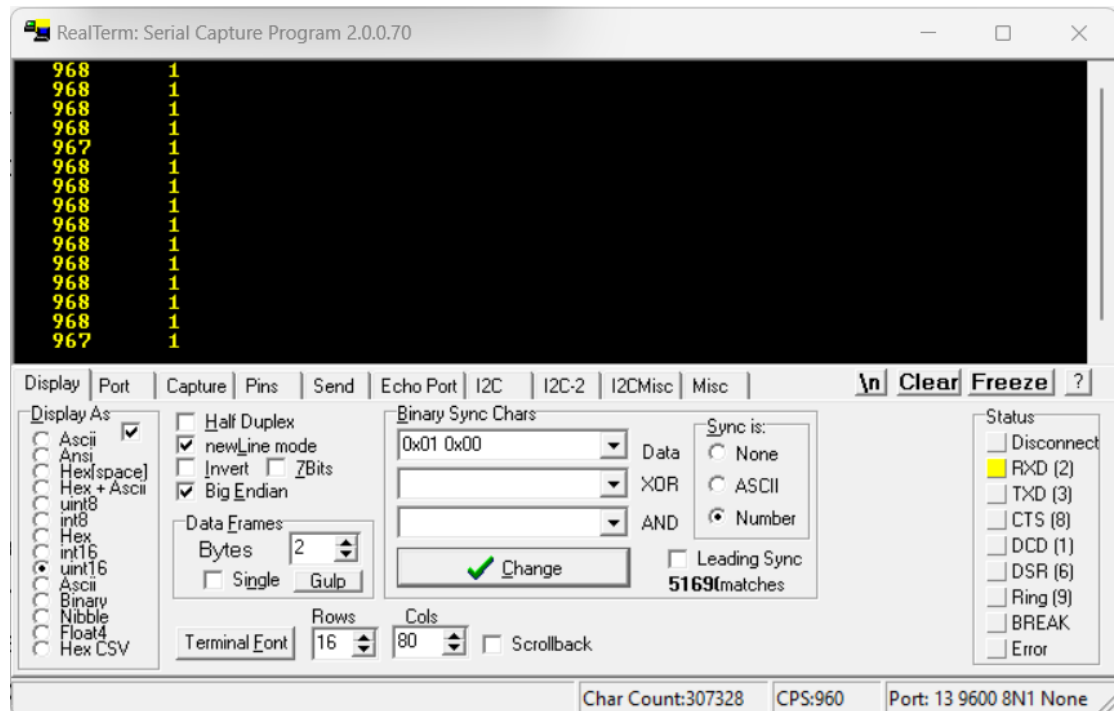
//UART Initialization

SBI DDRD,PIND1 // Set Tx as Output Pin
SBI PORTD,PIND1

//UART Initialization
LDI R16,0x08// Enabling Tx Rx
OUT UCSRB,R16
LDI R17,0x86// 8 bit data mode
OUT UCSRC,R17
//9600bps Baud-Rate Settings for 8MHz Oscillator
LDI R16,51
OUT UBRRL,R16

MAIN:      SBI ADCSRA,ADSC
           CONV: SBIC ADCSRA,ADSC
           RJMP CONV //Jump over next instruction if portbit clear
           IN R16,ADCL
           IN R17,ADCH
           OUT UDR,R16
           AGAIN1: SBIS UCSRA, UDRE
                   RJMP AGAIN1
           OUT UDR,R17
           AGAIN2: SBIS UCSRA, UDRE
                   RJMP AGAIN2
           LDI R18,0x01
           OUT UDR,R18
           AGAIN3: SBIS UCSRA, UDRE
                   RJMP AGAIN3
           LDI R18,0x00
           OUT UDR,R18
           AGAIN4: SBIS UCSRA, UDRE
                   RJMP AGAIN4
           JMP MAIN
```

Set the display as unsigned int16, then select new line mode and select binary sync characters to verify the 10-bit ADC value ranging from 0 to 1023 vertically.



Experiment4: /* Print ADC data on LCD. Oscillator frequency=8MHz*/

```
.INCLUDE "M32DEF.INC"
.ORG 0x0000

LDI R16,HIGH(RAMEND)
OUT SPH,R16
LDI R16,LOW(RAMEND)
OUT SPL,R16

MAIN: //Data Direction Register of LCD
      LDI R16,0xFF
      OUT DDRB,R16
      SBI DDRD,PIND4 //Falling Edged Enable
      SBI DDRD,PIND5 //Register Select

      // LCD Initialization
      CBI PORTD,PIND5 // Command Register Enable
      LDI R16,0x38 //2 lines and 5x7 matrix
      OUT PORTB,R16
      CALL ENABLE
      LDI R16,0x02 // Return Home
      OUT PORTB,R16
      CALL ENABLE
      LDI R16,0x01 // Clear display screen
      OUT PORTB,R16
      CALL ENABLE
      LDI R16,0x0C //Display on, cursor off
      OUT PORTB,R16
      CALL ENABLE
      LDI R16,0x06 // Shift Cursor to right after print on LCD
      OUT PORTB,R16
      CALL ENABLE
```



```
//Set Cursor Coordinate
LDI R16,0x80 //Set Cursor at beginning of 1st Line
OUT PORTB,R16
CALL ENABLE

//ADC Initialization
LDI R16,0x00
OUT ADMUX,R16
LDI R16,0xC3
OUT ADCSRA,R16

Infinite_Loop:      SBI ADCSRA,ADSC
                    ADC_CONV:      SBIC ADCSRA,ADSC
                                   RJMP ADC_CONV

                    IN R24,ADCL
                    IN R25,ADCH

//Digit separation to print on LCD
LDI R21,0
LOOP_Digit3: INC R21
              LDI R23,19
THOUSAND:    SBIW R25:R24,50
              DEC R23
              BRNE THOUSAND
              SBIW R25:R24,50
              BRPL LOOP_Digit3

DEC R21
LDI R23,20
THOUSAND_ADD: ADIW R25:R24,50
              DEC R23
              BRNE THOUSAND_ADD

LDI R19,0
LOOP_Digit2: INC R19
              SBIW R25:R24,50
              SBIW R25:R24,50
              BRPL LOOP_Digit2

DEC R19
ADIW R25:R24,50
ADIW R25:R24,50

LDI R20,0
LOOP_Digit1: INC R20
              SBIW R25:R24,10
              BRPL LOOP_Digit1

DEC R20
ADIW R25:R24,10

//Digit to Ascii Digit conversion
LDI R22,48
ADD R21,R22
ADD R19,R22
ADD R20,R22
ADD R24,R22

//Clear LCD Screen
CBI PORTD,PIND5
LDI R16,0x01 // Clear display screen
OUT PORTB,R16
CALL ENABLE
//Print on LCD
SBI PORTD,PIND5
OUT PORTB,R21
```



```
CALL ENABLE
OUT PORTB,R19
CALL ENABLE
OUT PORTB,R20
CALL ENABLE
OUT PORTB,R24
CALL ENABLE
//Wait for few times
CALL DELAY
JMP Infinite_Loop
```

```
ENABLE:
SBI PORTD,PIND4
LDI R18,0xFF
LOOP2_EN:    LDI R17,0x00
              LOOP1_EN:    NOP
              DEC R17
              BRNE LOOP1_EN

              DEC R18
              BRNE LOOP2_EN

CBI PORTD,PIND4
RET

DELAY:
LDI R16,0x0F
LDI R18,0xFF
LOOP3_DL:    LDI R17,0xFF
              LOOP1_DL:    NOP
              DEC R17
              BRNE LOOP1_DL

              DEC R18
              BRNE LOOP3_DL

DEC R16
BRNE LOOP3_DL
RET
```

Make the below circuit and simulate it then make the hardware to realize practically.

