ICT5307: Embedded System Design

Lecture 5
Keypad Interfacing and Analogue to Digital Converter

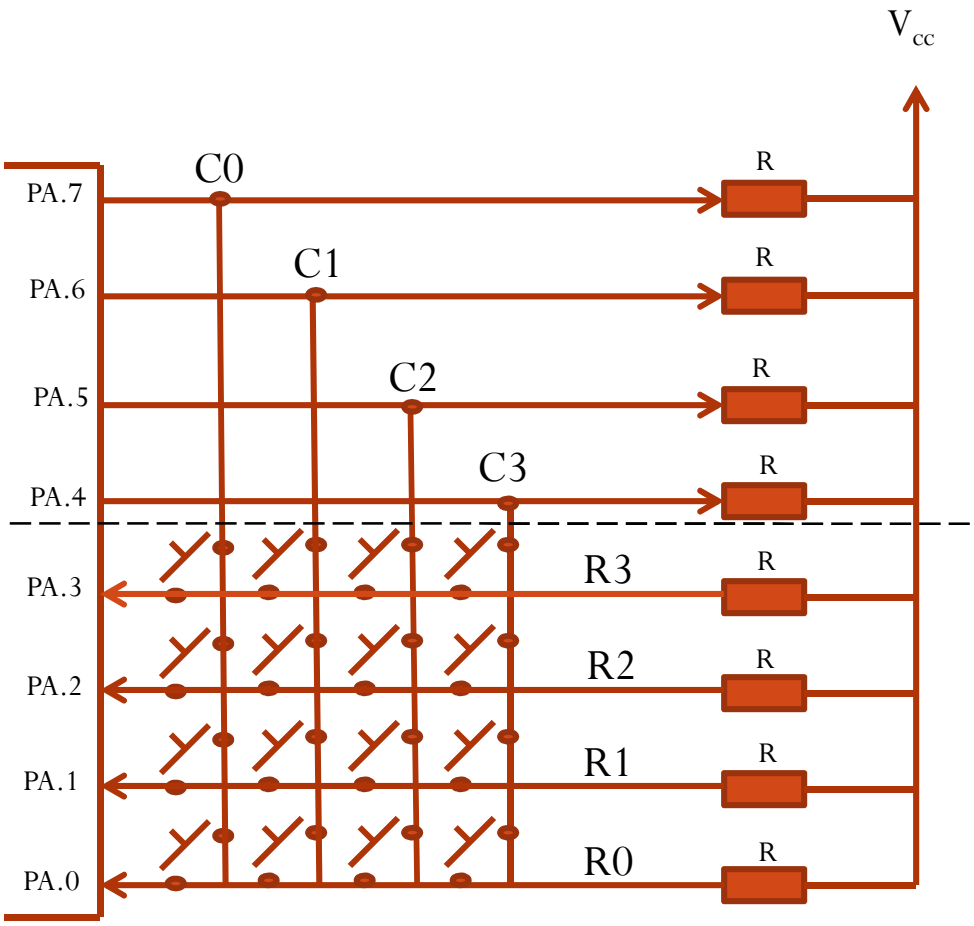
Professor S.M. Lutful Kabir
IICT, BUET

Keypad Interfacing

- Keypads and LCDs are the most widely used input/output devices in microcontrollers such as the AVR.
- We shall discuss keypad fundamentals, key pressed and key detection mechanism.
- At the lowest level, keypads are organized in a matrix of rows and columns.
- The CPU accesses both the rows and columns through ports.
- Therefore, with a 8-bit ports, an 4X4 keypad can be connected to a microcontroller.

Keypad in a Matrix

- Figure shows 16 key interfaced using one port.
- The rows are connected to lower 4 pins of port A and the columns are connected to upper 4 pins of the same port.
- If no key has been pressed and rows are read, all rows will be read as '1' since VCC is connected to the rows through resistors



Row Scanning Technique

- First of all, send '0' only in column no. 0 (C0, i.e. PA.7)
- Now if the row lines [PA.3 to PA.0] are read, and if any '0' is read (say) from row 'r' [r=0 to 3], it will identify that the switch [in between column no. '0' and the specific row 'r'] is pressed
- If all rows are read as '1', it will correspond to 'no' key pressed in that column no. '0'.
- Then send '0' to other columns one after another [column no. c=0 to 3] and similarly the row lines is read, if this process is continuously repeated and if any key is pressed, it will be detected in this process of scanning.

Code for 0-9, A, b, C, d, E and F

0=3F	8=7F
1=06	9=6F
2=5B	A=77
3=4F	b=7C
4=66	C=39
5=6D	d=5E
6=7D	E=79
7=07	F=71

Program on Row Scanning Technique (Part-I)

```
while (1)
    {
    PORTB=0X00;
    // column 1 is grounded
    PORTA=0b01110000;
        if (PINA.0==0)
          PORTB=0x06;     // code for "1"
        if (PINA.1==0)
          PORTB=0x66;     // code for "4"
        if (PINA.2==0)
          PORTB=0x07;     // code for "7"
        if (PINA.3==0)
          PORTB=0x79;     // code for "E"
```

Program on Row Scanning Technique (Part-II)

```
// column 2 is grounded
    PORTA=0b10110000;
        if (PINA.0==0)    // check for row 0
          PORTB=0x5B;     // code for "2"
        if (PINA.1==0)    // check for row 1
          PORTB=0x6D;     // code for "5"
        if (PINA.2==0)    // check for row 2
          PORTB=0x7F;     // code for "8"
        if (PINA.3==0)    // check for row 3
          PORTB=0x3F;     // code for "0"
```

Program on Row Scanning Technique (Part-III)

```
// column 3 is grounded
PORTA=0b11010000;
      if (PINA.0==0)    // check for row 0
        PORTB=0x4F;     // code for "3"
      if (PINA.1==0)    // check for row 1
        PORTB=0x7D;     // code for "6"
      if (PINA.2==0)    // check for row 2
        PORTB=0x6F;     // code for "9"
      if (PINA.3==0)    // check for row 3
        PORTB=0x71;     // code for "F"
```

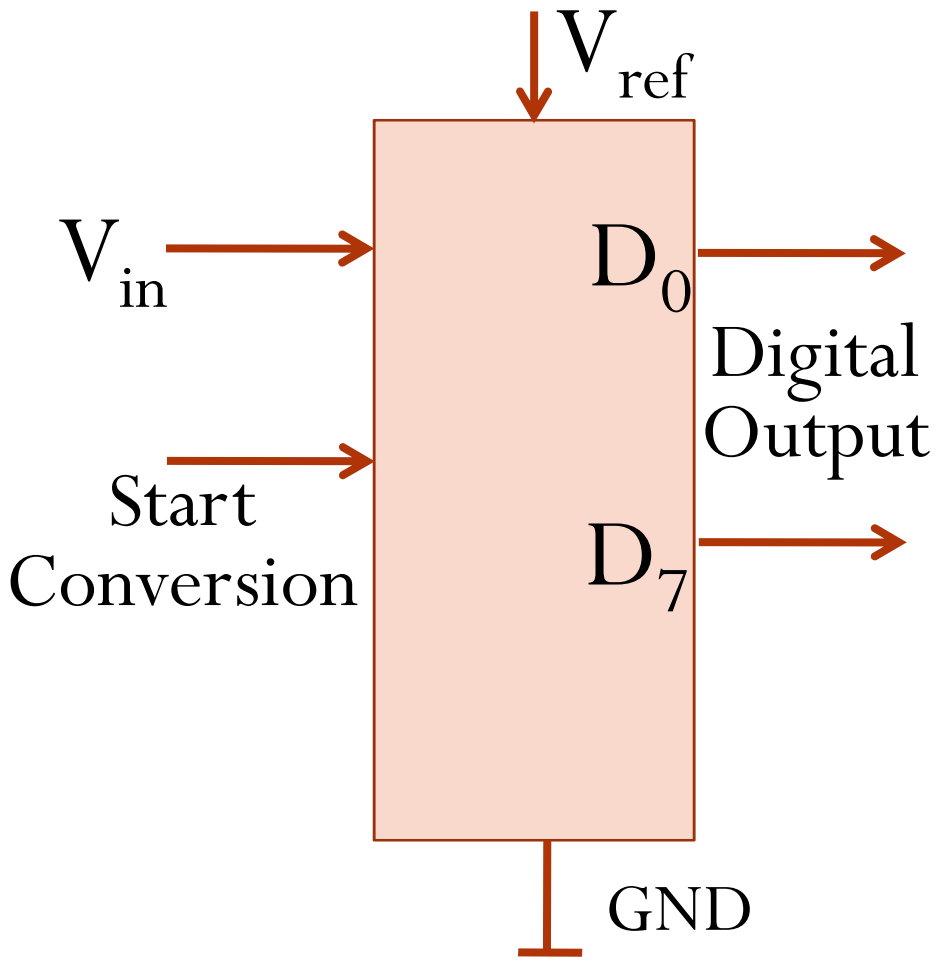
Program on Row Scanning Technique (Part-IV)

```
// column 4 is grounded
PORTA=0b11100000;
        if (PINA.0==0)    // check for row 0
          PORTB=0x77;     // code for "A"
        if (PINA.1==0)    // check for row 1
          PORTB=0x7C;     // code for "b"
        if (PINA.2==0)    // check for row 2
          PORTB=0x39;     // code for "C"
        if (PINA.3==0)    // check for row 3
          PORTB=0x5E;     // code for "D"
  }
```

ADC Devices

- Temperature, pressure, humidity, velocity are a few examples of physical quantities we deal with every day.
- A physical quantity is converted into electrical (voltage or current) signals by devices which are called transducer
- Transducers are also referred to as sensors.
- Most of the sensors produce an output that is voltage (or current)
- Therefore, we need an Analogue-to-Digital (ADC) converter to translate analogue signals to digital numbers so that microcontroller can read and process them.

Block Diagram of an ADC



Some of the major characteristics of ADC

- Resolution
- Conversion Time
- V_{ref}
- Digital Data Output
- Parallel versus Serial ADC
- Analogue Input Channels
- Start of Conversion and End of Conversion signals
- Successive approximation ADC

Resolution

- The ADC has n-bit resolution where n can be 8,10,12,16, or even 24.
- Higher resolution ADCs provide a smaller step size, where step size is the smallest change that can be distinguished by the ADC.
- We can not change the resolution (number of step) of an ADC but we can control the step size by modifying Vref.

Conversion Time

- Conversion time is defined as the time that an ADC takes to convert the analogue input into the digital (binary) output.
- The conversion depends on
  - Clock source
  - Method of Conversion
  - Technology used, TTL or CMOS

Reference Voltage, V_{ref}

- V_{ref} is the input voltage used for the reference voltage.
- The step size (the width of the smallest voltage to be recognized) is determined by this V_{ref} and resolution of the ADC (no. of bits).
- For an 8-bit ADC, the step size is $V_{ref}/2^8$ or $V_{ref}/256$.
- With a 8-bit ADC, if Vref is equal to 2.56V, then the step size will be 2.56V/256, or 10mV.
- In some ADC, differential voltage is required for V_{ref}. In that $V_{ref}=V_{ref}(+) - V_{ref}(-)$.

Digital Data Output

- In 8-bit ADC, the output is D0-D7, whereas for 10-bit ADC, it is D0-D9.
- The digital output is given by, $D_{out}=V_{in}$/step size
- For example, for Vref=2.56V and 8-bit ADC, if V_{in}=2.0V, the digital output in decimal is 2.0V/10mV=200D or 11001000b
- Find the binary output for the above case, if it was a 10-bit ADC and Vref=5V?
- Ans=0110011001b = 1.9971V ?

Parallel versus Serial ADC

- The ADC chips are either serial or parallel.
- For parallel output, number of output pin same of resolution of the ADC.
- If the resolution is more that 8-bit (say 12-bit), in order to save pin, the data is divided into multiple segments of 8-bits so that 8-pin is used for the output.
- For serial ADC, only one pin is required
- For serial out, the converted data is brought to microcontroller one bit at a time, so there must be a parallel in serial out shift register inside the ADC.
- For this reason serial ADCs are slow.

Analogue Input Channels

- Many data acquisition applications require more than one ADC.
- For that reason we see ADC chips with 2, 4, 8 or even 16 channels on a single chip.
- The channels are multiplexed.

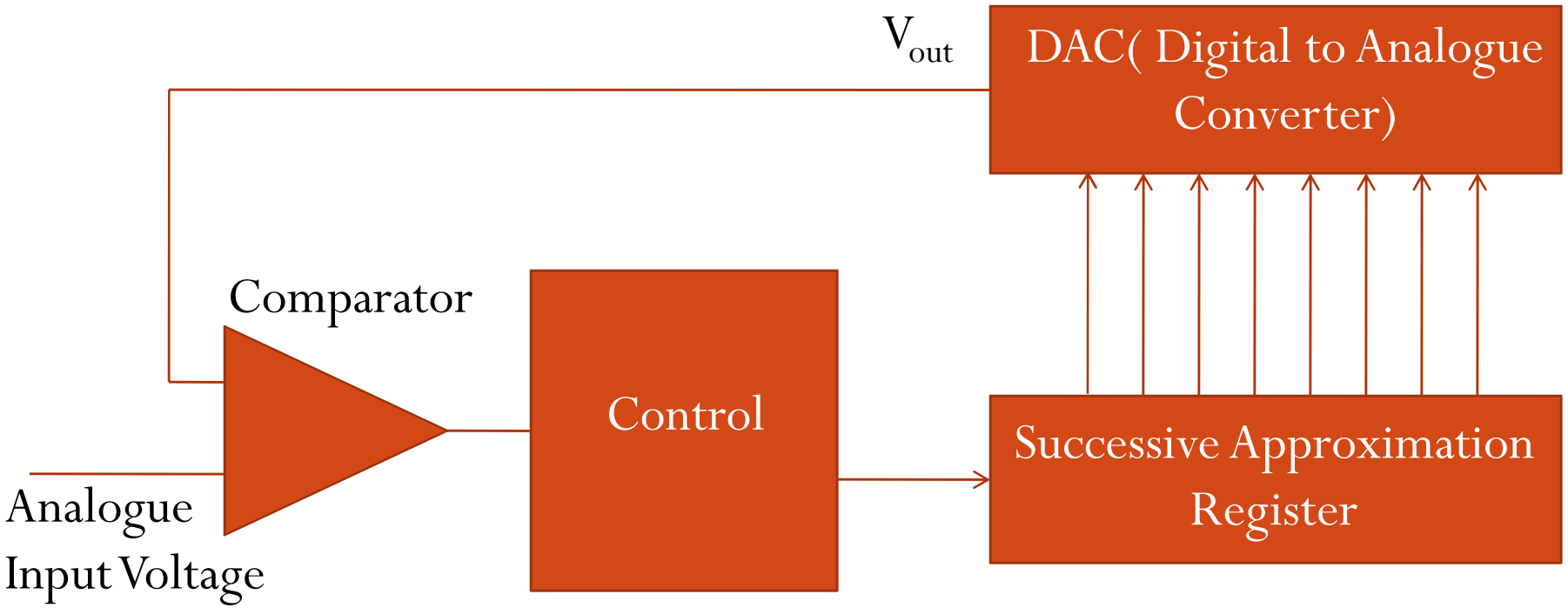
Start of Conversion and End of Conversion signals

- Since there are multiple input channel and one single output channel, there is a need for a Start of conversion (SC) and End of Conversion (EOC) signals.
- When SC is activated, the ADC starts converting the input value of Vin to an n-bit binary number.
- When the data conversion is complete, the end of conversion signal notifies the CPU that the converted data is ready for pick up.

Successive approximation ADC

- Successive approximation method is widely used method of converting an analog input to digital output.
- It has three major components:
  - Successive Approximation Register (SAR)
  - Comparator
  - Control Unit

Successive approximation ADC



Features of ATmega32 ADC

- The ADC peripheral of ATmega32 has the following characteristics
- 10-bit ADC
- 8 analogue input channels, 7 differential input channel and two input channel with optional gain of 10x and 200x.
- Converted binary data is stored in ADCL and ADCH
- ADCH:ADCL makes 16-bit register, but since 10-bit data is saved. 6-bit remains unused.
- We have the option of keeping upper 6-bit or lower 6-bit of ADCH:ADCL register unused.

Features of ATmega32 ADC

- We have three option for V_{ref}.
- V_{ref} can be connected to AVCC (Analog V_{CC}) or internal 2.56V reference or to external AREF.
- The conversion time is dictated by the crystal frequency connected to XTAL pins (Fosc) and ADPS00:02 bits.

NOTE: The AVCC and Vref should be constant. If they vary, the accuracy of ADC conversion is affected.

ADC programming in AVR

- In AVR five registers are used for AVR programming
  - ADCH (ADC data register high)
  - ADCL (ADC data register low)
  - ADCSRA (ADC Control and Status and Register A)
  - ADCMUX (ADC Mutliplexer)
  - SPIOR (Special Function I/O register)

ADCMUX Register

- REFS1:0 – Bit 7:6 – Reference Selection bits

  This bits select reference voltage for the ADC. 00- AREF pin, 01- AVCC pin and 10- Reserved and 11- Internal 2.56V

- ADLAR – Bit 5 – ADC Left adjust result

  1- left adjusted and 0- for right adjusted result

- MUX 4:0 – Bit 4:0 – Analogue Channel and Gain Selection bits

  Selects the gain of differential channels and selects which combination of inputs are connected to the ADC

Remembering the name of the bits in ADMUX Register

7-Ref bit 1	-	REFS1
6-Ref bit 0	-	REFS0
5-Left Adjust Register	-	ADLAR
4-Channel and Gain bit4	-	MUX4
3-Channel and Gain bit3	-	MUX3
2-Channel and Gain bit2	-	MUX2
1-Channel and Gain bit1	-	MUX1
0-Channel and Gain bit0	-	MUX0

ADMUX Register

- Either single ended or differential input can be selected for conversion.
- If you select single ended input, you can choose the input channel among ADC0 – ADC7. Table shows the MUX4:0 inputs for different single ended selection

MUX4:0	ADC Channel
00000	ADC0
00001	ADC1
00010	ADC2
00011	ADC3
00100	ADC4
00101	ADC5
00110	ADC6
00111	ADC7

ADMUX Register

- For differential inputs the MUX4:0 bits are shown in the table
- The Table shows that you can select the positive input to be one of the pins ADC0 to ADC7 and negative input to be ADC0, ADC1 or ADC2

MUX4:0	Dif(+)	Diff(-)	Gain
01001	ADC1	ADC0	10X
01011	ADC1	ADC0	200X
01101	ADC3	ADC2	10X
01111	ADC3	ADC2	200X
10000	ADC0	ADC1	1X
10010	ADC2	ADC1	1X
10011	ADC3	ADC1	1X
10100	ADC4	ADC1	1X
10101	ADC5	ADC1	1X
10110	ADC6	ADC1	1X
10111	ADC7	ADC1	1X
11000	ADC0	ADC2	1X
11001	ADC1	ADC2	1X
11011	ADC3	ADC2	1X
11100	ADC4	ADC2	1X
11101	ADC5	ADC2	1X

ADLAR bit operation

ADLAR=1

Left Justified

ADLAR=0

Right Justified

ADCSRA Register

- ADEN – Bit 7 – ADC Enable
- ADSC – Bit 6 – ADC start conversion
- ADATE – Bit 5 – ADC auto triggered enable
- ADIF – Bit 4 – ADC Interrupt flag

  This bit is set when ADC conversion completes and the data registers are updated
- ADIE – Bit 3 – ADC Interrupt Enable
- ADPS2:ADPS0 – Bit 2:1 – ADC Prescaler Select bits

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

Remembering the name of the bits in ADCSRA Register

- 7-**EN**able - EN - **AD**EN
- 6-**S**tart **C**onversion - SC - **AD**SC
- 5-**A**uto **T**rigger **E**nable - ATE - **AD**ATE
- 4-**I**nterrupt **F**lag - IF - **AD**IF
- 3-**I**nterrupt **E**nable - IE - **AD**IE
- 2-**P**re-**S**caler bit **2** - PS2 - **AD**PS2
- 1-**P**re-**S**caler bit **1** - PS1 - **AD**PS1
- 0-**P**re-**S**caler bit **0** - PS0 - **AD**PS0

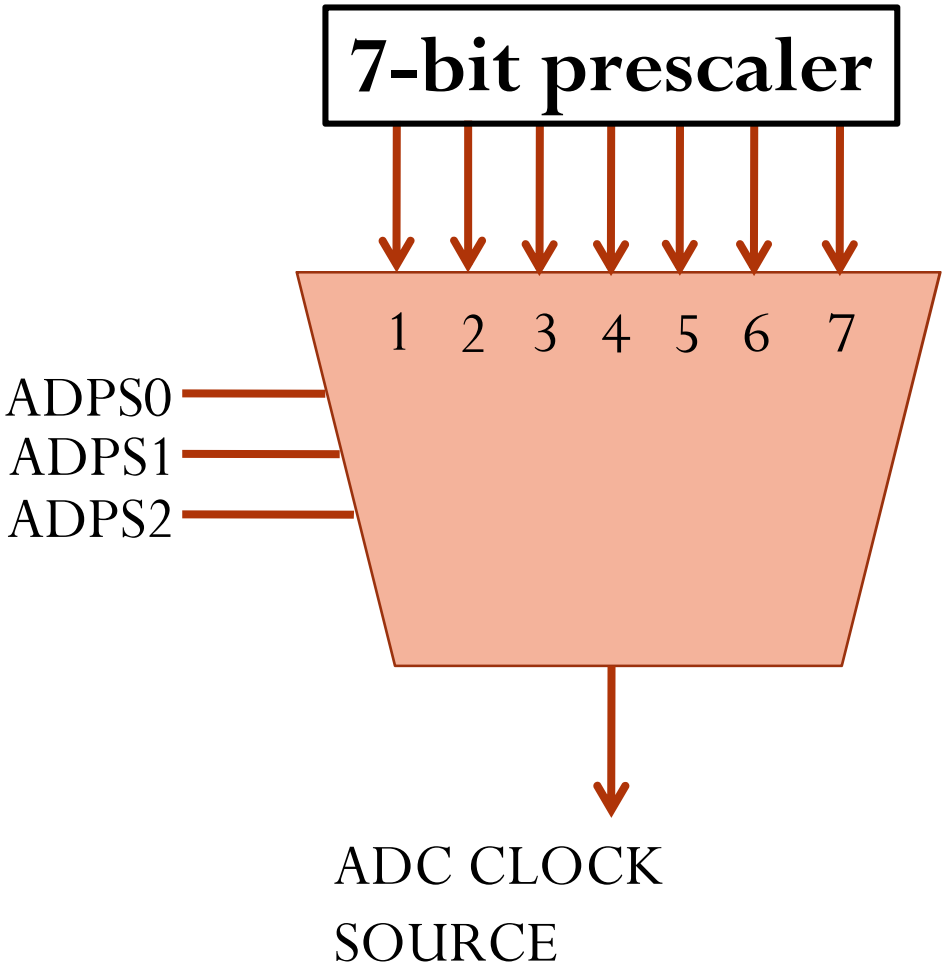
ADC Start Conversion bit

- ADSC – by setting this bit conversion of ADC starts. ADC has a special circuit to trigger start conversion
- Apart from this bit there are option to select Auto triggering mode by setting ADATE bit
- Auto triggering mode or manual triggering can be understood by the block diagram shown in the next slide.

ADC Conversion Time

- By using ADPS2:0 bits we can set the ADC conversion time.
- To select conversion time we can select any of Fosc/2, Fosc/4, Fosc/8, Fsoc/16, Fosc/32, Fosc/64 or Fosc/128 for ADC clock, where Fosc is the frequency of the crystal connected to AVR chip
- For AVR, the ADC requires an input clock frequency less than 200 kHz for max^{m} accuracy

AVR ADC Clock Selection



ADPS2	ADPS1	ADPS0	ADC CLOCK
0	0	1	CLK/2
0	1	0	CLK/4
0	1	1	CLK/8
1	0	0	CLK/16
1	0	1	CLK/32
1	1	0	CLK/64
1	1	1	CLK/128

Sample-and-Hold time in ADC

- After an ADC channel is selected, the ADC allows some time for the sample-and-hold capacitor to charge fully to the input voltage level.
- Table shows the conversion time for some different conditions

Conditions	Sample-and-Hold time (cycles)	Total conversion time (cycles)
First conversion	14.5	25
Normal conversion, single ended	1.5	13
Normal conversion, differential	2	13.5
Auto trigger conversion	1.5/2.5	13/14

Steps of Programming

- Step #1: Make the pin for the selected ADC channel as Input pin
- Step #2: Enable the ADC
- Step #3: Select the conversion speed
- Step #4: Select voltage reference and ADC input channels
- Step #5: Set the Start Conversion bit, ADSC
- Step #6: Wait for some time and continuously monitor the ADIF bit.
- Step #7: After ADIF bit is found to be high, ADCL and ADCH bytes are read
- Step #8: If want to read the selected channel again go to step 5

An Experiment on ADC

- A variable voltage source will be connected to one of the channel of ADC. It is channel ADC0 (1st channel).
- The analogue voltage will be converted into digital form by the ADC.
- The voltage read and after converting into decimal will be displayed on LCD.
- Reference voltage will be connected to AVCC.
- ADC frequency to be chosen to CLK/128. So, for 16 MHz crystal, ADC frequency will be 16 MHz/128=125 kHz

Program (Part-I)

```
#include <mega32.h>
#include <stdlib.h>
#include <delay.h>
#include <alcd.h>

unsigned char display[16];
float volt=0;

// Voltage Reference: AREF pin
#define ADC_VREF_TYPE ((0<<REFS1) | (1<<REFS0) | (0<<ADLAR))
```

Program (Part-II)

REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0

ADMUX

```
// Read the AD conversion result
unsigned int read_adc(unsigned char adc_input)
{
ADMUX=adc_input | ADC_VREF_TYPE;
// Delay needed for the stabilization of the ADC input voltage
delay_us(10);
// Start the AD conversion
ADCSRA|=(1<<ADSC);
// Wait for the AD conversion to complete
while ((ADCSRA & (1<<ADIF))==0);
//ADCSRA|=(1<<ADIF);
return ADCW;
}
```

MUX 4:0	ADC Ch.
00000	ADC0
00001	ADC1
00010	ADC2
00011	ADC3
00100	ADC4
00101	ADC5
00110	ADC6
00111	ADC7

ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0

ADCSRA

Program (Part-III)

```
while (1)
    {
    volt=read_adc(0)/204.8;
    delay_ms(100);
    lcd_gotoxy(0,0);
    ftoa(volt,2,display);
    lcd_puts(display);
    delay_ms(200);
    }
}
```

Thanks