Module 3 Illustrate the interfacing of Microcontroller with external peripherals.
M3.01 Illustrate the interfacing of Serial Port, LCD and Keyboard Interfacing.
M3.02 Illustrate the interfacing of ADC, DAC and Sensors.

Contents:
Interfacing: ATMega32 Connection to RS232, Serial Port Programming in C, LCD Interfacing, Keyboard Interfacing, ADC, DAC and Sensor interfacing and Programming in C.

BASICS OF SERIAL COMMUNICATION

- Serial communication is used for transferring data between two systems located at distances of hundreds of feet to millions of miles apart.
- For long-distance data transfers, serial data communication requires a modem to modulate (convert from Os and 1s to audio tones) and demodulate (convert from audio tones to Os and 1s).
- Serial data communication uses two methods:
  ✓ Asynchronous
  ✓ Synchronous.
- The synchronous method transfers a block of data (characters) at a time, whereas the asynchronous method transfers a single byte at a time.
- Special IC chips are made by many manufacturers for serial data communications.
- These chips are commonly referred to as UART(universal asynchronous receiver transmitter) and USART(universal synchronous asynchronous receiver-transmitter).
- The AVR chip has a built-in USART.
- Data transmission methods are:
  - Duplex – Data can be both transmitted and received.Duplex transmissions can be half or full duplex.Half duplex – Data is transmitted one way at a time.Full duplex – Data can go both ways at the same time.
  - Simplex – only sends data.

RS232 standards

- RS232 is one of the most widely used serial I/O interfacing standards.
- it is used in PCs and numerous types of equipment.
- Different versions- RS232A, RS232B, RS232C.
- Voltage representation-:1 is –3 to -25, 0 is +3 to +25
- -3 to +3 are undefined. For this reason, to connect RS232 to a microcontroller, we use a voltage converter. e.g.MAX232 IC.
- This IC is commonly known as line driver.
- RS232 cables nad labels are commonly referred as DB-25 connector.
- The simplest connection between a PC and a microcontroller requires minimum 3 pins- TX,RX and ground.

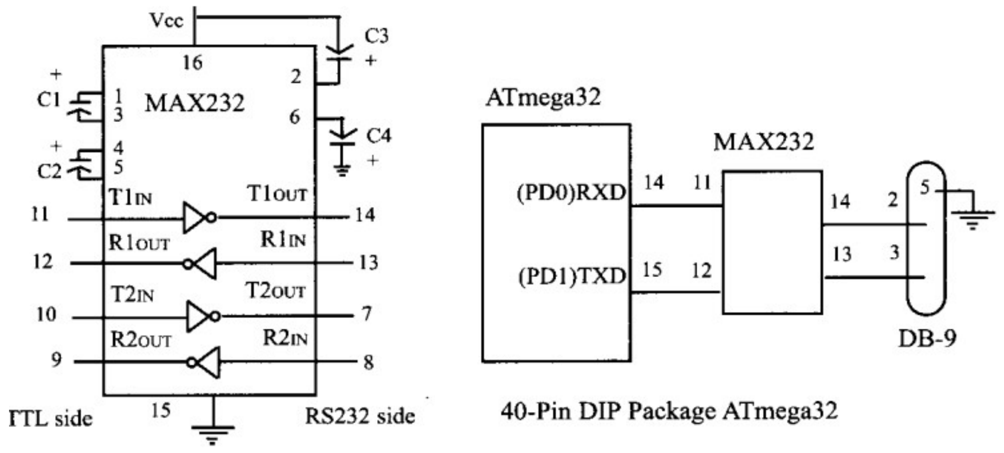
ATMEGA32 CONNECTION TO RS232

- The ATmega32 has two pins that are used specifically for transferring and receiving data serially.
- These two pins are called TX and RX and are part of the PortD group (PD0 and PD1) of the 40-pinpackage.
- Pin15 of the ATmega32 is assigned to TX and pin 14 is designated as RX.

MAX232

- The MAX232 converts from RS232 voltage levels to TTL voltage levels, and vice versa.
- One advantage of the MAX232 chip is that it uses a +5Vpower source, which is the same as the source voltage fo r the AVR.

- The MAX232 has two sets of line drivers for transferring and receiving data.
- The line drivers used for TX are called T1 and T2 while the line drivers for RX are designated as R1 and R2.
- It requires 4 capacitors ranges from 0.1 to 22microfarad.



Inside MAX232 and its connection to ATMega32

AVR serial port Programming in C

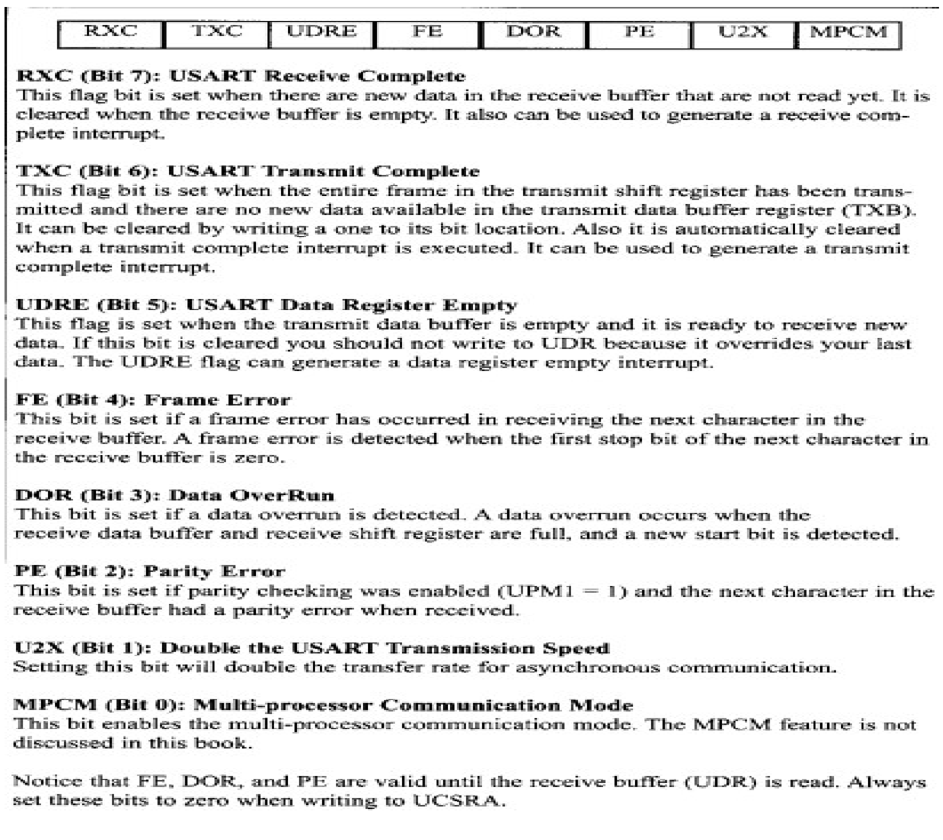
There are five registers related to serial port.
1.UDR
2.UCSRA
3.UCSRB
4.UCSRC
5.UBRR

UDR(USART Data Register)

- to provide full duplex serial data communication, there are 2 shift registers-transmit shift register, receive shift register. UDR share the transmit buffer register and receive buuffer register.

UCSR registers

- it is used for controlling serial communication.
- There are 3 USART control register.-UCSRA,UCSRB,UCSRC
- UCSRA register



RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM

RXC (Bit 7): USART Receive Complete
This flag bit is set when there are new data in the receive buffer that are not read yet. It is cleared when the receive buffer is empty. It also can be used to generate a receive complete interrupt.

TXC (Bit 6): USART Transmit Complete
This flag bit is set when the entire frame in the transmit shift register has been transmitted and there are no new data available in the transmit data buffer register (TXB). It can be cleared by writing a one to its bit location. Also it is automatically cleared when a transmit complete interrupt is executed. It can be used to generate a transmit complete interrupt.

UDRE (Bit 5): USART Data Register Empty
This flag is set when the transmit data buffer is empty and it is ready to receive new data. If this bit is cleared you should not write to UDR because it overrides your last data. The UDRE flag can generate a data register empty interrupt.

FE (Bit 4): Frame Error
This bit is set if a frame error has occurred in receiving the next character in the receive buffer. A frame error is detected when the first stop bit of the next character in the receive buffer is zero.

DOR (Bit 3): Data OverRun
This bit is set if a data overrun is detected. A data overrun occurs when the receive data buffer and receive shift register are full, and a new start bit is detected.

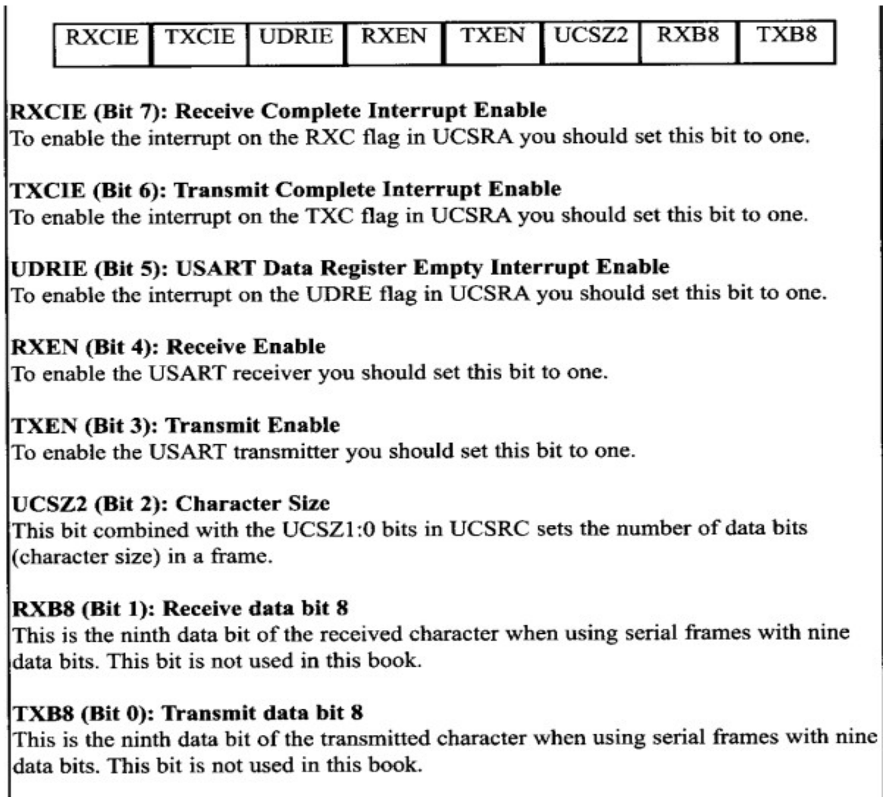
PE (Bit 2): Parity Error
This bit is set if parity checking was enabled (UPM1 = 1) and the next character in the receive buffer had a parity error when received.

U2X (Bit 1): Double the USART Transmission Speed
Setting this bit will double the transfer rate for asynchronous communication.

MPCM (Bit 0): Multi-processor Communication Mode
This bit enables the multi-processor communication mode. The MPCM feature is not discussed in this book.

Notice that FE, DOR, and PE are valid until the receive buffer (UDR) is read. Always set these bits to zero when writing to UCSRA.

UCSRB register



RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8

RXCIE (Bit 7): Receive Complete Interrupt Enable
To enable the interrupt on the RXC flag in UCSRA you should set this bit to one.

TXCIE (Bit 6): Transmit Complete Interrupt Enable
To enable the interrupt on the TXC flag in UCSRA you should set this bit to one.

UDRIE (Bit 5): USART Data Register Empty Interrupt Enable
To enable the interrupt on the UDRE flag in UCSRA you should set this bit to one.

RXEN (Bit 4): Receive Enable
To enable the USART receiver you should set this bit to one.

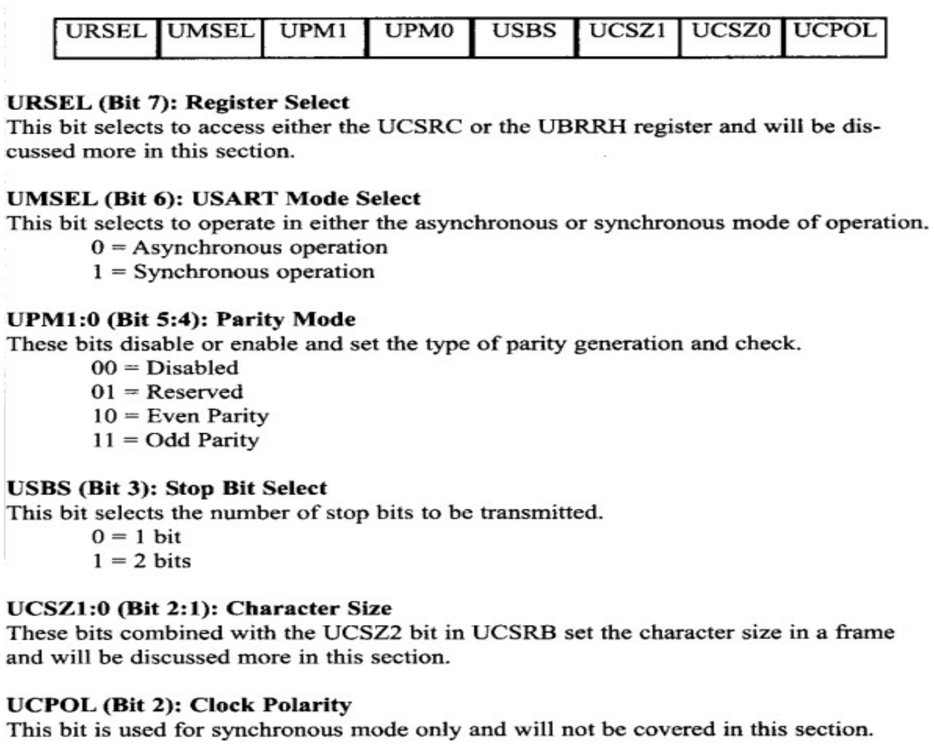
TXEN (Bit 3): Transmit Enable
To enable the USART transmitter you should set this bit to one.

UCSZ2 (Bit 2): Character Size
This bit combined with the UCSZ1:0 bits in UCSRC sets the number of data bits (character size) in a frame.

RXB8 (Bit 1): Receive data bit 8
This is the ninth data bit of the received character when using serial frames with nine data bits. This bit is not used in this book.

TXB8 (Bit 0): Transmit data bit 8
This is the ninth data bit of the transmitted character when using serial frames with nine data bits. This bit is not used in this book.

UCSRC register



URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL

URSEL (Bit 7): Register Select
This bit selects to access either the UCSRC or the UBRRH register and will be discussed more in this section.

UMSEL (Bit 6): USART Mode Select
This bit selects to operate in either the asynchronous or synchronous mode of operation.
0 = Asynchronous operation
1 = Synchronous operation

UPM1:0 (Bit 5:4): Parity Mode
These bits disable or enable and set the type of parity generation and check.
00 = Disabled
01 = Reserved
10 = Even Parity
11 = Odd Parity

USBS (Bit 3): Stop Bit Select
This bit selects the number of stop bits to be transmitted.
0 = 1 bit
1 = 2 bits

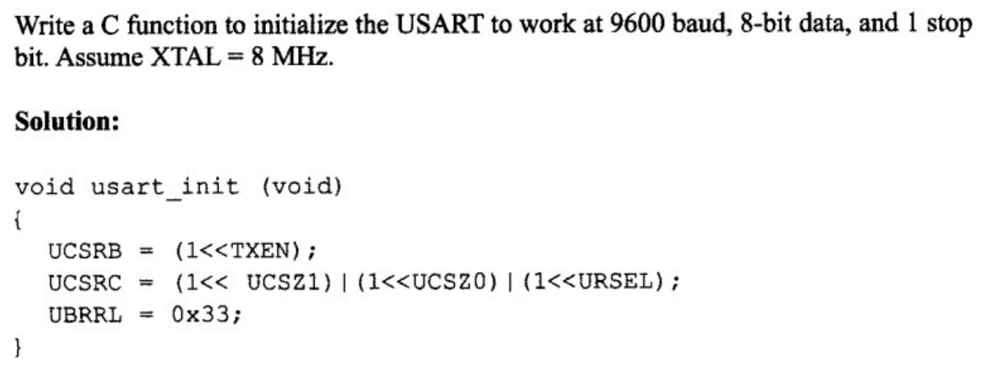
UCSZ1:0 (Bit 2:1): Character Size
These bits combined with the UCSZ2 bit in UCSRB set the character size in a frame and will be discussed more in this section.

UCPOL (Bit 2): Clock Polarity
This bit is used for synchronous mode only and will not be covered in this section.

UBRR register

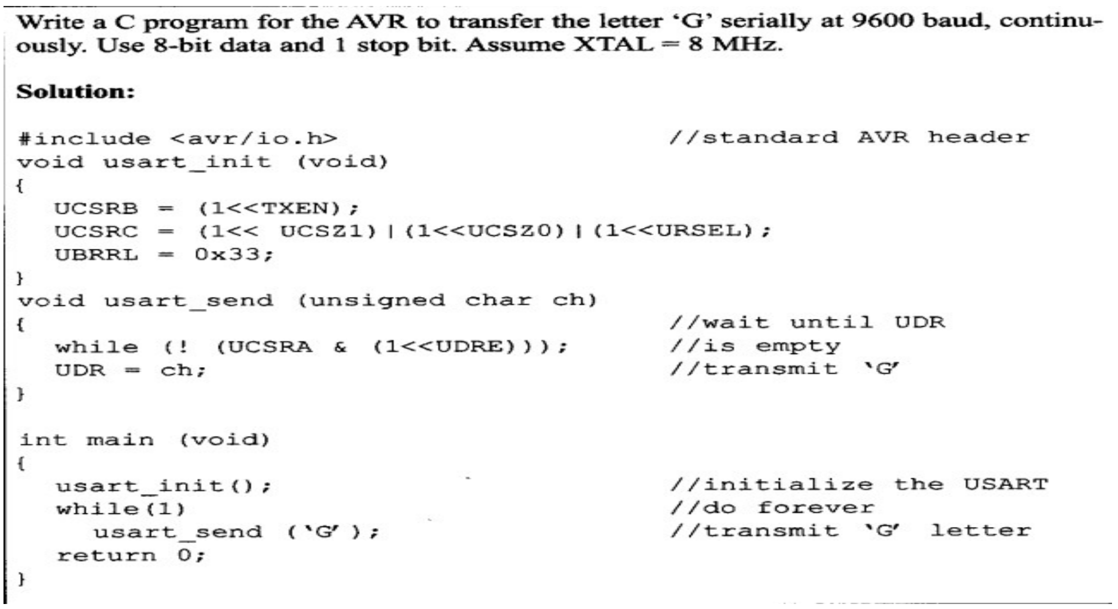
- the value loaded into the UBRR decides the baud rate.
- Desired baud rate= fosc/16(X+1), where fosc- frequency of oscillator connected to the XTAL1 and XTAL2, X-value loaded into the UBRR register.

Program


Write a C function to initialize the USART to work at 9600 baud, 8-bit data, and 1 stop bit. Assume XTAL = 8 MHz.

Solution:

```
void usart_init (void)
{
   UCSRB = (1<<TXEN);
   UCSRC = (1<< UCSZ1)|(1<<UCSZ0)|(1<<URSEL);
   UBRRL = 0x33;
}
```


Write a C program for the AVR to transfer the letter 'G' serially at 9600 baud, continuously. Use 8-bit data and 1 stop bit. Assume XTAL = 8 MHz.

Solution:

```
#include <avr/io.h>                    //standard AVR header
void usart_init (void)
{
   UCSRB = (1<<TXEN);
   UCSRC = (1<< UCSZ1)|(1<<UCSZ0)|(1<<URSEL);
   UBRRL = 0x33;
}
void usart_send (unsigned char ch)
{                                      //wait until UDR
   while (! (UCSRA & (1<<UDRE)));      //is empty
   UDR = ch;                           //transmit 'G'
}

int main (void)
{
   usart_init();                       //initialize the USART
   while(1)                            //do forever
     usart_send ('G');                 //transmit 'G' letter
   return 0;
}
```

LCD INTERFACING

- In recent years the LCD is finding wide spread use replacing LEDs (seven segment LEDs or other multisegment LEDs). This is due to the following reasons:

1. The declining prices of LCDs.
2. The ability to display numbers, characters, and graphics. This is in contrast to LEDs, w h i c h are limited to numbers and a few characters.
3. Incorporation of a refreshing controller into the LCD, there by relieving the CPU of the task of refreshing the LCD. In contrast, the LED must be refreshed by the CPU (or in some other way) to keep displaying the data.
4. Ease of programming for characters and graphics.

LCD pin descriptions

- LCD has 14pins..
- The function of each pins are:

Vcc, VEE,and Vss

- Vcc : +5 V
- Vss : ground
- VEE : used for controlling LCD contrast

RS (Register Select)

- There are two very important registers inside the LCD.-command code register, data register
- The RS pin is used for their selection as follows.
- If RS=0, the instruction command code register is selected, allowing the user to send commands such as clear display, cursor at home, and so on.
- If RS= 1 the data register is selected, allowing the user to send data to be displayed on the LCD.

R/W (Read/Write)

- R/W input allows the user to write information to the LCD or read information from it.
- R/W=1 when reading; R/W=0 when writing.

E (Enable)

- The enable pin is used by the LCD to latch information presented to its data pins.

D0-D7

- The 8-bit data pins, DO-D7,are used to send information to the LCD or read the contents of the LCD's internal registers.

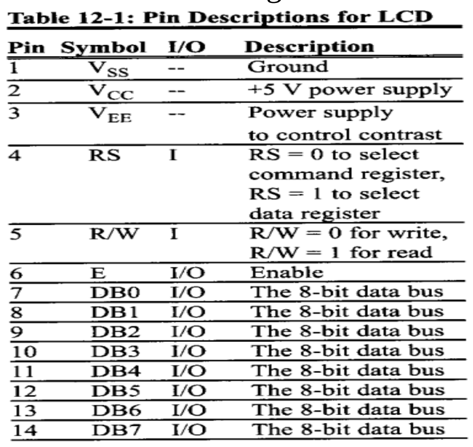


Table 12-1: Pin Descriptions for LCD

Pin	Symbol	I/O	Description
1	V_{SS}	--	Ground
2	V_{CC}	--	+5 V power supply
3	V_{EE}	--	Power supply to control contrast
4	RS	I	RS = 0 to select command register, RS = 1 to select data register
5	R/W	I	R/W = 0 for write, R/W = 1 for read
6	E	I/O	Enable
7	DB0	I/O	The 8-bit data bus
8	DB1	I/O	The 8-bit data bus
9	DB2	I/O	The 8-bit data bus
10	DB3	I/O	The 8-bit data bus
11	DB4	I/O	The 8-bit data bus
12	DB5	I/O	The 8-bit data bus
13	DB6	I/O	The 8-bit data bus
14	DB7	I/O	The 8-bit data bus

Sending commands and data to LCDs

- To send data and commands to LCDs you should do the following steps. Notice that steps 2 and 3 can be repeated many times:

1.Initialize the LCD .
✓ To initialize the LCD for 5x 7matrix and 8-bit operation, the
following sequence of commands should be sent to the LCD: Ox38, Ox0E , and 0x01.
✓ Next we will show how to send a command to the LCD.
✓ After power-up you should wait about 15ms before sending initializing commands to
the LCD.
2.Send any of the commands to the LCD

- To send any of the commands to the LCD ,make pins RS and RIW=0 and put the command number on the data pins (D0-D7).

✓ Then send a high-to-low pulse to the E pin to enable the internal latch of the LCD.
✓ Notice that after each command you should wait about 100µs.
✓ After the 0x01 a n d Ox02 c o m m a n d s you should wait for about 2ms.
3.Send the character to be shown on the LCD.
✓ To send data to the LCD, make pins RS= 1 and RIW=0.
✓ Then put the data on the data pins (DO-D7)and send a high-to-low pulse to the E
pin to enable the internal latch of the LCD.
✓ After sending data you should wait about 100 µs to let the LCD module write the
data on the screen.

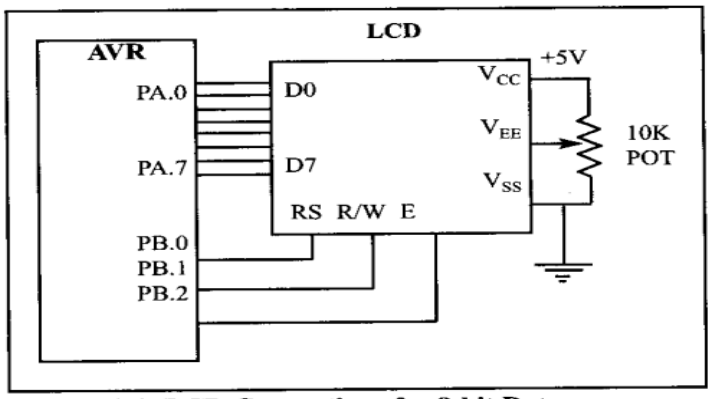


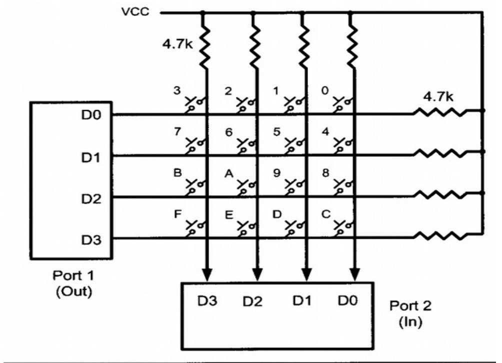
Figure 12-2 LCD Connections for 8-bit Data

Program

```
#include<avr/io.h>
#include<util/delay.h>
void lcdcommand(unsigned char cmnd)
{
PORTA=cmnd;   // portA = cmnd
PORTB=0x00;    //r=0,r/w=0,(selection for command register)
PORTB=0x04;   //e=1
PORTB=0x00;   //e=0  ( a high pulse to low pulse on E, lcd starts its operation
}
void lcddata(unsigned char data)
{
PORTA=data;     // portA = data
PORTB=0x01;     //r=1,r/w=0,(selection for data register)
PORTB=0x04;    //e=1
PORTB=0x01;   //e=0  ( a hih pulse to low pulse on E, lcd starts its operation
}
void lcd_print(char *str)
{
unsigned char i=0;
while(str[i]!=0)
{
lcddata(str[i]);  // pass the character for printing
i++;
}
}
int main(void)
{
DDRA=0xFF;  //set portA as output
DDRB=0xFF;    //set portB as output
lcdcommand(0x38);    //select 2 line LCD monitor
lcdcommand(0x0E);   //display on , cursor blinking
lcdcommand(0x01);   // clear display
_delay_us(2000);       // call built in delay function
lcd_print(“hai”);       // print message by calling function lcd_print
lcdcommand(0xC0);  // set the cursor position to second line
lcd_print(“welcome:”);  // print the message
while(1);         // infinte loop. Display the message forever
return 0;
}
```

KEYBOARD INTERFACING

- Keyboards are organised in a matrix of rows and columns.
- When a key is pressed , a row and column make a contact, otherwise there is no connection between rows and columns.



igure 12-7. Matrix Keyboard Connection to Ports

- The rows are connected to an outport. The columns are connected to input port.
- To detect a pressed key, the microcontroller grounds all rows,i.e 0 and reads the columns.
- If the data reads from the columns are 1s, then no key has been pressed.
- If one of the column bit has 0, this means that one of the keu has been pressed.
- After a key press is detected,the microcontroller will go through the process of identifying the key.
- Starting with the top row, the microcontroller grounds it and read the columns.
- If no key press is detected, then it grounds the next ro nd read columns.
- Thi process continues until the row is identified.
- After identifying the row, next task is to find out the column.
- For this, microcontroller check the position of 0.

program

write a c program to read the keypad and send the result to PORTD.PC0-PC2- connected to columns and PC4-PC7 -connected to rows.

```
#include<avr/io.h>
#include<util/delay.h>
unsigned char key[4][3]={'1','2','3','4','5','6','7','8','9','0','*','#'};
int main(void)
{
unsigned char col,row;
DDRD=0xFF;  //portd as output
DDRC=0xF8;  //PC4-PC7 as output, PC@-PC0 as input
while(1)
{
do
{
PORTC=0x07;   //ground all rows
col=PINC&0x07;  //read columns
}while(col!=0x07);
while(1)
{
PORTC=0xEF;    // ground PC4, row=0
col=PINC&0x07;  // read the column
if(col!=0x07)   // if  key is pressed this row
{
row=0; break;  // assign row value and break  from the loop
}

PORTC=0xDF;    // ground PC5, row=1
col=PINC&0x07;  // read the column
if(col!=0x07)   // if  key is pressed this row
{
row=1; break;  //assign row value and break  from the loop
}

PORTC=0xBF;    // ground PC6, row=2
col=PINC&0x07;  // read the column
if(col!=0x07)   // if key is pressed this row
{
row=2; break;       //assign row value and break  from the loop
}

PORTC=0x7F;    // ground PC7, row=3
col=PINC&0x07;  // read the column
if(col!=0x07)   // if key is pressed this row
{
row=3; break;
}

if(col==0x06)   // col==0(PC2=0)
{
PORTD=key[row][0];
}
else if(col==0x05)   // col==0(PC1=0)
{
PORTD=key[row][1];
}
        else      // col==2(PC0=0)
{
PORTD=key[row][2];
}
}
}
return 0;
}
```

ADC(Analog to Digital Converter)

- ADC converts the analog signal to digital number.
- These are most widely used device for data acquisition.
- Transducer is a device that converts the physical quantity to electrical signal.
- Characteristics of ADC

1) resolution – high resolution ADC provide a smaller step size. Step size is the smallest change that can be discerned by an ADC.
2) Conversion time- time it takes the ADC to convert the analog input to digital
3) vref- reference voltange. Range is 0 to 5V
4) digital data output- 8 bit ADC has 8 bit digital data output.
5) Analog input channels- 16 ADC channels

ATMega32 ADC features

1. 10 bit ADC
2. 8 input channels,7 differential input channels and 2 differential input channels with optiona gain of 10x & 200X
3. 2 special function register- ADCL and ADCH. These are 8 bit each.
4. we have the option of making eithe upper 6 bits or lower 6 bits of ADCL:ADCH register unused.
5. Vref can be- Analog VCC, internal 2.56V,external AREF pin
6. conversion time is dictated by the crystal frequency(Fosc pin),and ADP0:2 bits

Hardware considerations

- a small variation in voltage level has no effect on the output.
- To get a better accuracy of AVR ADC we must provide a stable voltage source to the AVCC pin.
- By connecting a capacitor between the AVREF pin and GND you can make the Vref voltage more stable and increase the precision.

ADC programming in C

5 major registers associated with ADC.
1. ADCH(high data)
2. ADCL(low data)

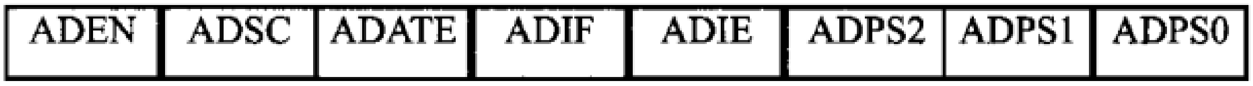
3.ADCSRA(ADC control and status register)
4. ADMUX(ADC multiplex selection register)
5.SPIOR(special function I/O register)

ADCH:ADCL

- result is stored in ADCL and ADCH register.

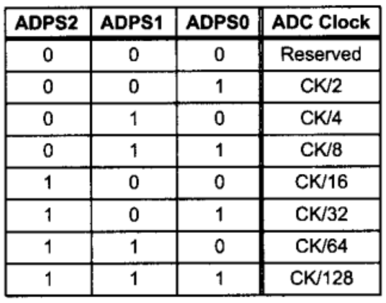
ADCSRA

- this bits control or monitor the operation of the ADC.



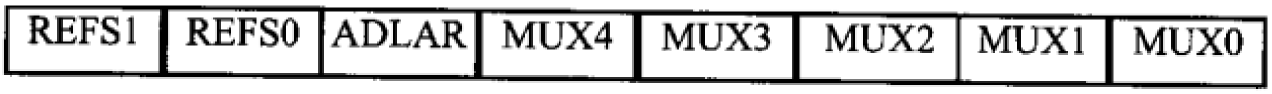
ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0

ADEN- ADC enable, 1-ADC enable,0-ADC disable
ADSC- ADC start conversion, 1-start ADC conversion
ADATE-ADC Auto Trigger Enable, 1-Auto trigger enable
ADIF- ADC Interrupt Flag, this bit=1, when ADC conversion completes
ADIE-ADC interrupt Enable,setting this bit to 1 enables when conversion completes
ADPS2:0



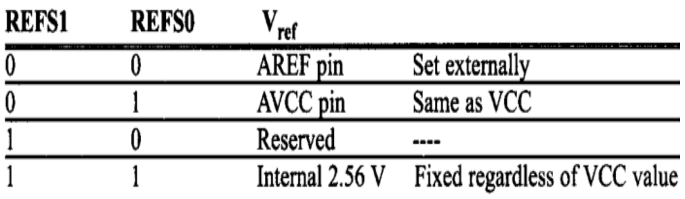
ADPS2	ADPS1	ADPS0	ADC Clock
0	0	0	Reserved
0	0	1	CK/2
0	1	0	CK/4
0	1	1	CK/8
1	0	0	CK/16
1	0	1	CK/32
1	1	0	CK/64
1	1	1	CK/128

ADMUX



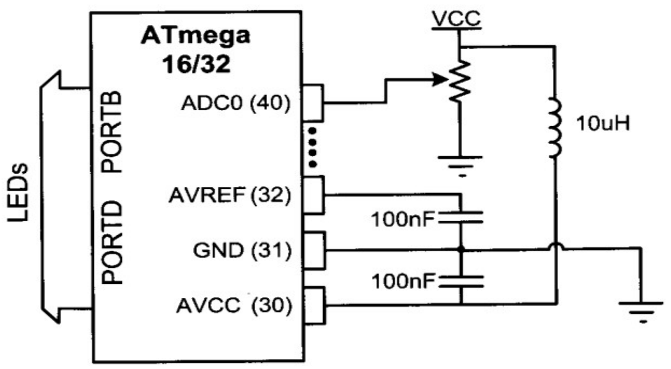
REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0

REFS1- Reference Selection Bits- select reference voltage



REFS1	REFS0	V_{ref}	
0	0	AREF pin	Set externally
0	1	AVCC pin	Same as VCC
1	0	Reserved	----
1	1	Internal 2.56 V	Fixed regardless of VCC value

ADLAR- ADC Left Adjust Result
=1, result is left adjusted
=0, result is right adjusted
MUX4:0
analog channel and gain selection bits



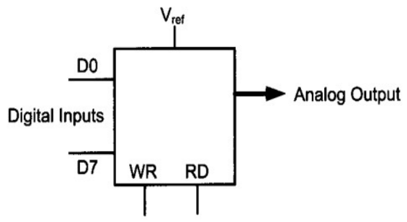
ADC connection

program

```
#include <avr/io.h>          //standard AVR header
int main (void)
{
   DDRB = 0xFF;              //make Port B an output
   DDRD = 0xFF;              //make Port D an output
   DDRA = 0;                 //make Port A an input for ADC input
   ADCSRA= 0x87;             //make ADC enable and select ck/128
   ADMUX= 0xC0;              //2.56V Vref, ADC0 single ended input
                             //data will be right-justified
   while (1){
     ADCSRA|=(1<<ADSC);      //start conversion
     while((ADCSRA&(1<<ADIF))==0);//wait for conversion to finis
     PORTD = ADCL;           //give the low byte to PORTD
     PORTB = ADCH;           //give the high byte to PORTB
   }
   return 0;
}
```

DAC Digital to Analog Converter

- it is widely used to convert digital signal to analog.
- Creating DAC using two methods-binary weighted and R/2R ladder.
- The number of data bit input decides the resolution of DAC.
- The analog output voltage level=2^n
- block diagram



- 12 bit DAC provides 4096 voltage levels.

program

```
#include <avr/io.h>          //standard AVR header

int main (void)
{
   unsigned char i = 0;      //define a counter
   DDRB = 0xFF;              //make Port B an output
   while (1){                //do forever
     PORTB = i;              //copy i into PORTB to be converted
     i++;                    //increment the counter
   }
   return 0;
}
```

sensors

- a transducer that converts temperature to electrical signal called thermistor.
- A thermistor responds to temperature change by changing resistance.
- Its response is not linear.
- Simple and widely used temperature sensors are- LM34 &LM35

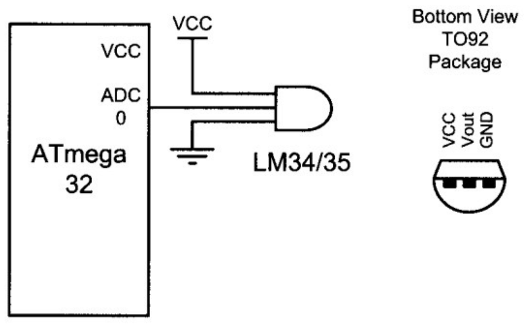
LM34

- precision integrated- circuit temperature sensor
- output is in fahrenheit temperature
- requires no external calibration
- for each fahrenheit, its output is 10mv

LM35

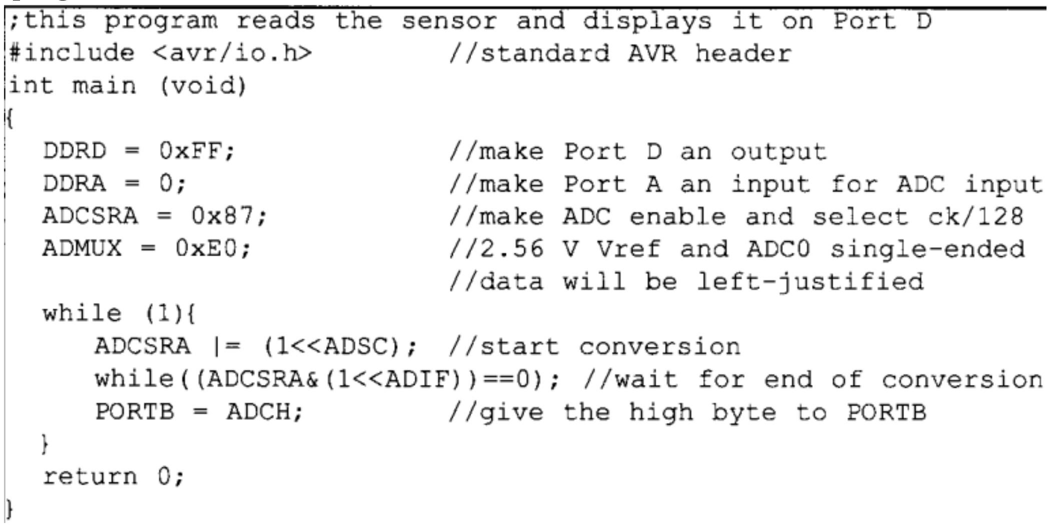
- precision integrated- circuit temperature sensor
- output is in celsius temperature
- requires no external calibration
- for each celsius temperature, its output is 10mv

Interfacing LM34 to atmega32



- max. Temperature sensed by LM34 is 3000f
- highest output for ADC is 3.00v
- LM34 is connected to channel0(ADC0 pin)
- 10 bit output of ADC is divided by 4 to get real temperature.

program



```
;this program reads the sensor and displays it on Port D
#include <avr/io.h>         //standard AVR header
int main (void)
{
  DDRD = 0xFF;              //make Port D an output
  DDRA = 0;                 //make Port A an input for ADC input
  ADCSRA = 0x87;            //make ADC enable and select ck/128
  ADMUX = 0xE0;             //2.56 V Vref and ADC0 single-ended
                            //data will be left-justified
  while (1){
     ADCSRA |= (1<<ADSC);  //start conversion
     while((ADCSRA&(1<<ADIF))==0); //wait for end of conversion
     PORTB = ADCH;          //give the high byte to PORTB
  }
  return 0;
}
```

important questions

1. explain the interfacing of LCD.
2. illustrate the interfacing of Keyboard.
3. explain the registers related to USART interface.
4. differentiate LM34 and LM35.
5. Explain DAC .
6. Explain the interfacing of ADC.
7. what do you mean by sensors.
8. explain the pin description of LCD.
9. explain the ADC features of ATMega32.
10.Explain RS232.

Video links

1. serial communication-https://www.youtube.com/watch?v=PE8n5oSfRBc, https://www.electronicwings.com/avr-atmega/atmega1632-usart
2.LDC interfacing-https://www.youtube.com/watch?v=rp9EIwFuv4g
3. keyboard interfacing-https://www.youtube.com/watch?v=3jMoF9HPChw
4. ADC interfacing-https://www.youtube.com/watch?v=QFD6D4fhTc8
5.DAC interfacing-

6. sensors-https://www.youtube.com/watch?v=q5hn0O6czJU

onword questions
1. what voltage levels are used for binary 0 in RS232.
2.true or false. The AVR has a built in UART.
3. what is the advantage of MAX233 over the MAX232 chip?
4. which registers of the AVR are used to set te baud rate?
5. UCSRA stands for........................
6. the E pin requires an ....pulse to latch in information at the data pins of LCD.
7. what is the difference between Vcc and Vee pin on the LCD.
8.what is the internal Vref of the Atmega32.
9. how many single ended inputs are available in the Atmega32 ADC?
10. the LM34 sensor produces ....mv for each degree of temperature.
11. in the Atmega32, what should the Vref value if we want a step size of 2mv.