MODULE – IV Interfacing sub systems with AVR

Basics of Serial Communication – ATMEGA32 connection to RS232 - AVR serial port programming in C- LCD interfacing - Keyboard interfacing - ADC interfacing – DAC interfacing - Sensor interfacing

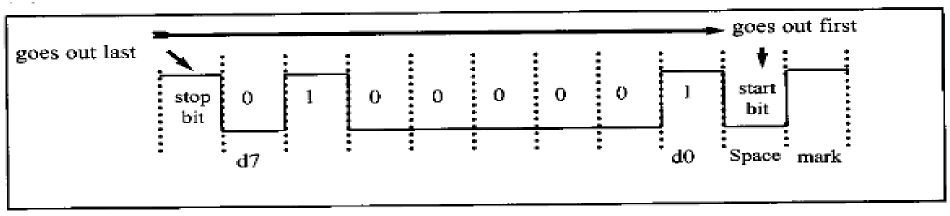
SERIAL COMMUNICATION

- Serial communication is used for transferring data between two systems located at distances of hundreds of feet to millions of miles apart.
- For serial data communication to work, the byte of data must be converted to serial bits
- Using a parallel-in-serial-out shift register; then it can be transmitted over a single data line.
- For long-distance data transfers, serial data communication requires a modem to *modulate* (convert from 0s and 1s to audio tones) and *demodulate* (convert from audio tones to 0s 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.
- Simplex – only sends data.
- 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.

Asynchronous serial communication

- Is widely used for character-oriented transmissions.
- Each character is placed between start and stop bits. This is called *framing.*
- In data framing for asynchronous communications , the data ,such as ASCII characters, are packed between a start bit and a stop bit.
- The start bit is always one bit, but the stop bit can be one or two bits.
- The start bit is always a 0(low), and the stop bit (s) is 1(high).

Framing ASCII 'A'(41H)



- From the above figure, when there is no transfer, the signal is 1(high), which is referred to as *mark*.
- The 0(low) is referred to as *space*.
- Notice that the transmission begins with a start bit(space) followed by D0, the LSB, then the rest of the bits until the MSB(07) ,and finally, the one stop bit indicating the end of the character "A".
- The rate of data transfer in serial data communication is stated in *bps* (bits per second). Another widely used terminology for bps is *baud rate*.

RS232 standards

- An interfacing standard is RS232
- To allow compatibility among data communication equipment made by various manufacturers.
- RS232 is one of the most widely used serial I/O interfacing standards.
- This standard is used in PCs and numerous types of equipment.

ATMEGA32 CONNECTION TO RS232

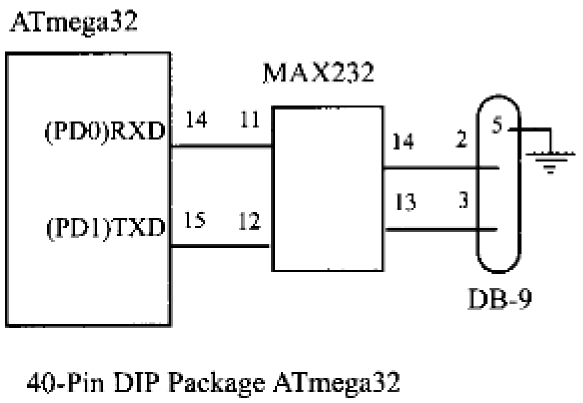
RX and TX pins in the ATmega32

- 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.
- These pins are TTL compatible; therefore , theyrequire a line driverto make them RS232 compatible.
- One such line driver is the MAX232chip.

MAX232

- Because the RS232 is not compatible with today's microprocessors and microcontrollers, we need a line driver (voltage converter) to convert the RS232's signals to TTL voltage levels that will be accept able to the AVR's TX and RX pins.
- One example of such a converter is MAX232 from Maxim Corp.

MAX232 connection to the ATmega32(NullModem)



40-Pin DIP Package ATmega32

- 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 forthe AVR.
- The MAX232 has two sets of line drivers for transferring and receiving data.
- The line drivers used for TX are called T1andT2 while the line drivers for RX are designated as R1and R2.

MAX233

- To save board space, some designers use the MAX233 chip from Maxim.
- The MAX233 performs the same job as the MAX232 but eliminates the need for capacitors.
- However, the MAX233 chip is much more expensive than the MAX232.

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, which 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 descriptons

- 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)

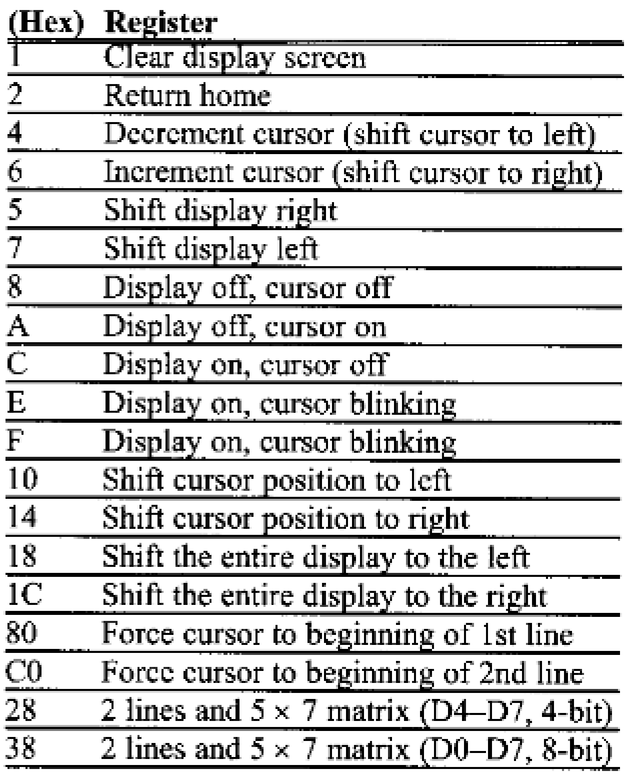
- The enable pin is used by the LCD to latch information presented to its data pins.

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.

Command code are:

Code Command to LCD instruction



(Hex)	Register
1	Clear display screen
2	Return home
4	Decrement cursor (shift cursor to left)
6	Increment cursor (shift cursor to right)
5	Shift display right
7	Shift display left
8	Display off, cursor off
A	Display off, cursor on
C	Display on, cursor off
E	Display on, cursor blinking
F	Display on, cursor blinking
10	Shift cursor position to left
14	Shift cursor position to right
18	Shift the entire display to the left
1C	Shift the entire display to the right
80	Force cursor to beginning of 1st line
C0	Force cursor to beginning of 2nd line
28	2 lines and 5 × 7 matrix (D4–D7, 4-bit)
38	2 lines and 5 × 7 matrix (D0–D7, 8-bit)

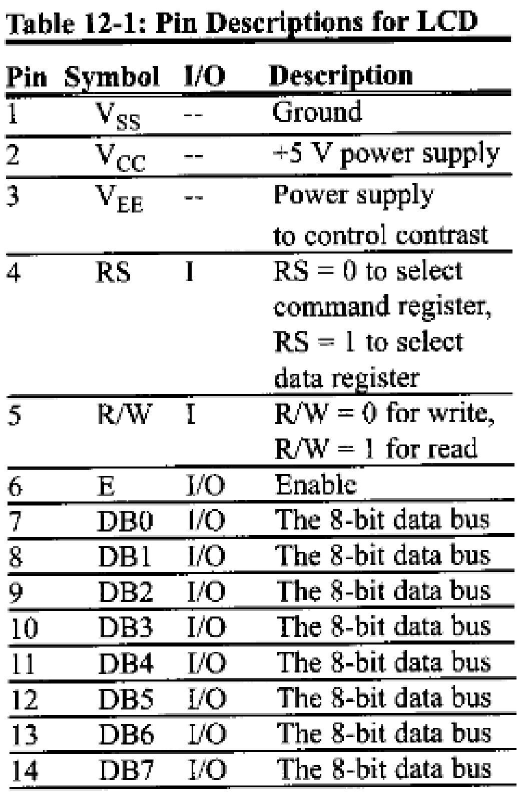


Table 12-1: Pin Descriptions for LCD

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

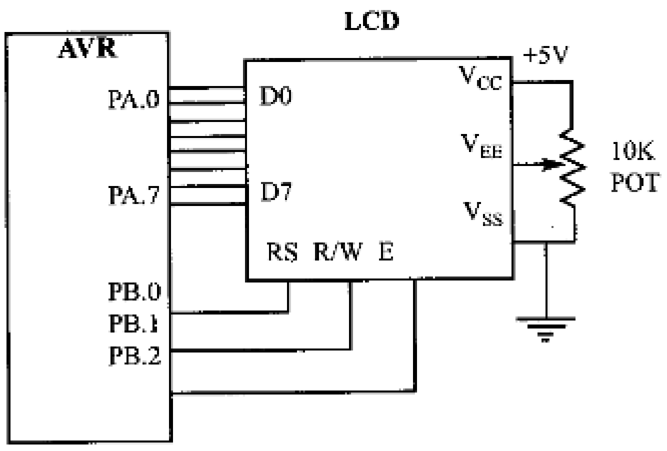
- To send data and commands to LCDs you should do the following steps. Notice that steps 2 and 3 can be repeated many times:

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:0x38, 0x0E , 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 and 0x02 commands you should wait for about 2ms.

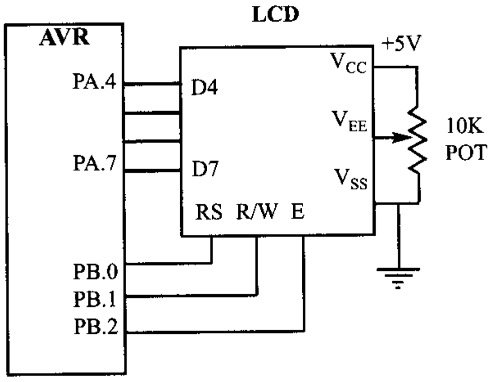
3. Send the character to be shown on the LCD.
   - ✓ To send data to the LCD, make pins RS= 1 and *RIW*=0.
   - ✓ Then put the data on the data pins (D0–D7)and send a high-to-low pulse to the E pin to enable the internal latch of the LCD.
   - ✓ After sending data you should wait about 100 µs to let the LCD module write the data on the screen.

The AVR connection to the LCD for 8-bit data is shown below:



Sending code or data to the LCD 4 bits at a time

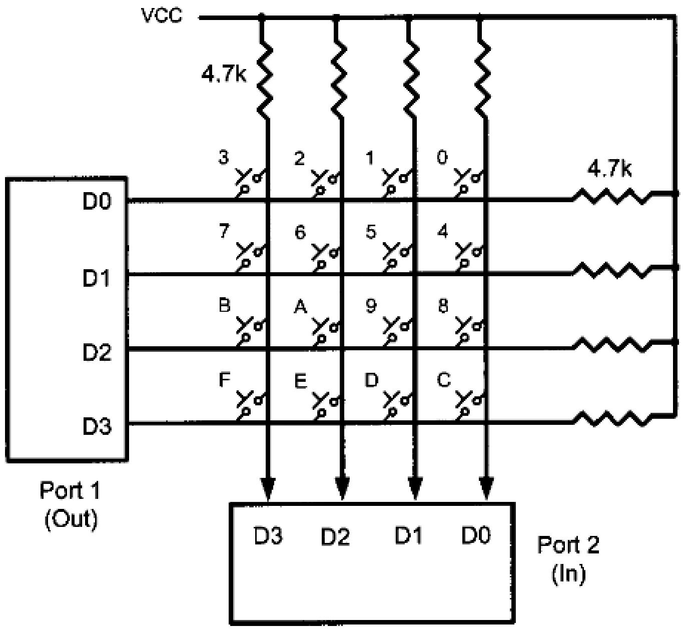
- In 4-bit mode, we initialize the LCD with the command code as 33, 32 and 28 in hex.
- The value $28 initializes the display for5x7 matrix
- 33,32 tells the LCD to go into 4-bit mode.
- LCD connection using 4-bit data.



KEYBOARD INTERFACING

- Keyboards are organized in a matrix of rows and columns.
- The CPU accesses both rows and columns through ports.
- Therefore, with two 8-bit ports, an8x 8matrix of keys can be connected to a microcontroller.
- When a key is pressed, a row and a column make a contact; otherwise, there is no connection between rows and columns.

Figure shows the matrix keyboard connection to ports:



- The rows are connected to an output port and the columns are connected to an input port.
- Columns and rows are all connected to high(VCC).
- It is the function of the microcontroller to scan the keyboard continuously to detect and identify the key pressed.

How this is done is explained next.

1. To detect a pressed key, the microcontroller grounds all rows by providing 0 to the output latch, and then it reads the columns.
2. If the data read from the columns is D3-DO= 1111,no key has been pressed and the process continues until a key press is detected. If one of the column bits has a zero, this means that a key press has occurred.
3. After a key press is detected, the microcontroller will go through the process of identifying the key.
4. Starting with the top row, the microcontroller grounds it by providing a low to row D0 only; then it reads the columns.
5. If the data read is all 1s,nokey in that row is activated and the process is moved to the next row.

6. It grounds the next row, reads the columns, and checks for any zero. This process continues until the row is identified.
7. After identification of the row in which the key has been pressed ,the next task is to find out which column the pressed key belongs to.
8. After identifying the row and column, the microcontroller will identify which key has been pressed.

ADC INTERFACING

- A non-chip ADC eliminates the need for an external ADC connection, which leaves more pins for other I/0 activities.
- The vast majority o f the AVR chips come with ADC.

ATmega32 ADC features

- The ADC peripheral of the ATmega32 has the following characteristics:

1) It is a10-bit ADC.
2) It has 8 analog input channels, 7 differential input channels, and 2 differential Input channels with optional gain of 10x and 200x.
3) The converted output binary data is held by two special function registers called ADCL *(AID* Result Low) and ADCH *(AID* Result High).
4) Because the ADCH:ADCL registers give us 16bits and the ADC data out is only l0bits wide, 6 bits of the 16 are unused. We have the option of making either the upper 6 bits or the lower 6bits unused.
5) We have three options for Vref- V ref can be connected t o AVCC (AnalogVcc), internal 2.56V reference, or external AREF pin.
6) The conversion time is dictated by the crystal frequency connected to the XTAL pins (Fosc) and ADPS0:2 bits.

AVR programming in Assembly and C

- In the AVR microcontroller five major registers are associated with the ADC.
- They are ADCH (high data),ADCL (low data),ADCSRA (ADC Control and Status Register), ADMUX (ADC multiplexer selection register), and SPIOR (Special Function I/O Register).

1. ADCH	4. ADMUX
2. ADCL	5. SPIOR
3. ADCSRA	

1. ADMUX register

- It is ADC multiplexer selection register.
- This register is used to select the reference voltage, select whether the result is left adjusted or right adjusted and select the analog inputs.
- The bits of ADMUX register is shown below:

REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0

REFS1:0 Bit 7:6 Reference Selection Bits
These bits select the reference voltage for the ADC.

ADLAR Bit 5 ADC Left Adjust Results
This 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.

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.

Vref source

- The REFS1 and REFS0 bits of the ADMUX register can be used to select the Vref source.

V_{ref} Source Selection Table for AVR

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

ADC input channel source

- Either single-ended or the differential input can be selected to be converted to digital data.
- If you select single-ended input, you can choose the input channel among ADC0 to ACD7.
- MUX4-MUX0 are used to select the internal circuitry of input channel either Single ended or differential input can be selected.
- The values of MUX4-MUX0 bits for different single ended inputs are listed below:

Single-ended Channels

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

ADLAR bit operation

- The AVRs have 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-bitregisters are dedicated to the ADC result, but only 10 of the 16 hits are used and 6bits are unused.
- You can select the position of used bits in the bytes.
- If you set the ADLAR bit in ADMUX register, the result bits will be left-justified; otherwise, the result bits will be right justified.

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

2. ADCH:ADCL registers

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

3. ADCSRA register

- The ADCSRA register is the status and control register of ADC.
- Bits of this register control or monitor the operation of the ADC.
- Description of each bits of the ADCSRA register is shown below:

ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0

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
Auto triggering of the ADC is enabled when you set this bit to one.

ADIF Bit 4 ADC Interrupt Flag
This bit is set when an ADC conversion completes and the data registers are updated.

ADIE Bit 3 ADC Interrupt Enable
Setting this bit to one enables the ADC conversion complete interrupt.

ADPS2:0 Bit 2:0 ADC Prescaler Select Bits
These bits determine the division factor between the XTAL frequency and the input clock to the ADC.

Steps in programming the A/D converter using polling

- To program the *A*/D converter of the AVR, the following steps must be taken:

1. Make the pin for the selected ADC channel an input pin.
2. Turn on the ADC module of the AVR because it is disabled upon power-on reset to save power.
3. Select the conversion speed. We use registers ADPS2:0 to select the conversion speed.
4. Select voltage reference and ADC input channels. We use the REFS0 and REFS1 bits in the ADMUX register to select voltage reference and the MUX4:0 bits in ADMUX to select the ADC input channel.
5. Activate the start conversion bit by writing a one to the ADSC bit of ADCSRA.
6. Wait for the conversion to be completed by polling the ADIF bit in the ADCSRA register.
7. After the ADIF bit has gone HIGH, read the ADCL and ADCH registers to get the digital data output. Notice that you have to read ADCL before ADCH; otherwise, the result will not be valid.
8. If you want to read the selected channel again, go back to step5.

9. If you want to select another Vref source or input channel, go back to step 4.

Steps in programming the A/D converter using interrupts

- We can use interrupts instead of polling to avoid tying down the microcontroller:

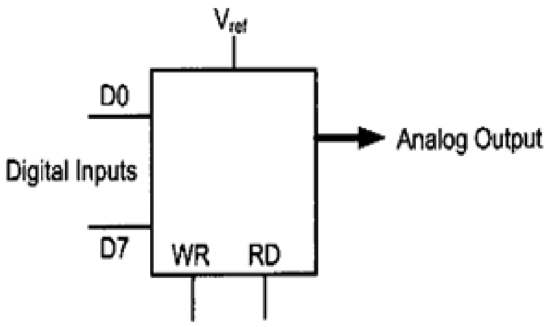
➢ Set HIGH the ADIE (A/D interrupt enable) flag.

➢ Upon completion of the conversion, the ADIF (A/D interrupt flag) changes to HIGH, if ADIE=1 it will force CPU to jump to the ADC interrupt handler.

DAC INTERFACING

- The digital-to- analog converter (DAC) is a device widely used to convert digital pulses to analog signals.
- There are two methods of creating a DAC:
  1. binary weighted 2. R/2R ladder.
- Most of the integrated circuit DACs, including the MC1408(DAC0808),use R/2R method because it can achieve a much higher degree of precision.
- The first criterion for judging a DAC is its resolution, which is a function of the number of binary inputs.
- The common ones are 8, 10, and 12 bits.

The block diagram of a DAC is shown below:

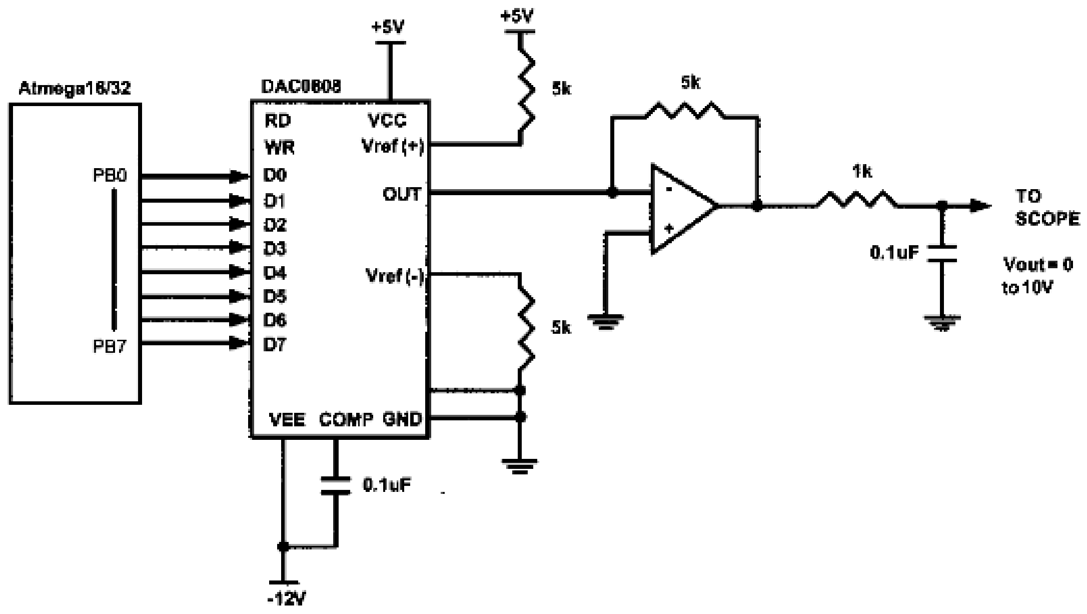


- The number of data bit inputs decides there solution of the DAC because the number of analog output levels is equal to 2^n, where *n* is the number of data bit inputs.
- Therefore, an 8-input DAC such as the DAC0808 provides 256 discrete voltage (or current) levels of output.
- The 12-bit DAC provides 4096 discrete voltage levels.

MC1408 DAC (or DAC0808)

AVR connection To DAC0808

- In the MC1408 (DAC0808), the digital inputs are converted to current (I_{out}), and by connecting a resistor to the Iout pin, we convert the result to voltage.



- The total current provided by the Iout pin is a function of the binary numbers at the DO-D7 inputs of the DAC0808 and the reference current (I_{ref}).
- Connect the output pin I_{out} to a resistor, convert this current to voltage, and monitor the output on the scope.

$$I_{out} = I_{ref}\left(\frac{D7}{2}+\frac{D6}{4}+\frac{D5}{8}+\frac{D4}{16}+\frac{D3}{32}+\frac{D2}{64}+\frac{D1}{128}+\frac{D0}{256}\right)$$

- Where D0 is the LSB, D7 is the MSB for the inputs and Iref is the input current.
- $\mathbf{I_{ref}}$ Current is generally set to 2.0 mA.
- If all the inputs to the DAC are high, the maximum output current is 1.99mA.
- $\mathbf{I_{ref}}$ Current output is isolated by connecting it to an op-amp, with Rf=5k for feedback resistor, to avoid inaccuracy.

TEMPERATURE SENSOR INTERFACING

- *Transducers* convert physical data such as temperature, light intensity, flow, and speed to electrical signals.
- Depending on the transducer, the output produced is in the form of voltage, current , resistance, or capacitance.
- For example, temperature is converted to electrical signals using a transducer called a *thermistor*.
- A thermistor responds to temperature change by changing resistance its response is not linear.
- Widely used linear temperature sensors include the LM34 and LM35 series from National Semiconductor Corp.

LM34 and LM35 temperature sensors

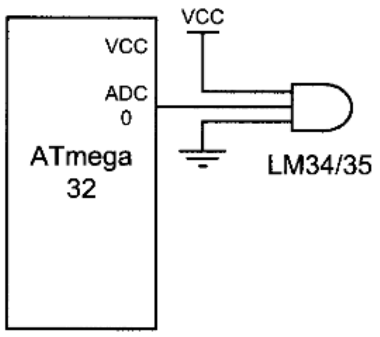
- The sensors of the LM34 series are precision integrated-circuit temperature sensors whose output voltage is linearly proportional to the Fahrenheit temperature.
- The LM34 requires no external calibration because it is internally calibrated.
- It outputs 10mV for each degree of Fahrenheit temperature.
- The LM35 series sensors are precision integrated-circuit temperature sensors whose output voltage is linearly proportional to the Celsius (centigrade) temperature.
- It outputs 10mV for each degree of centigrade temperature.

Interfacing the LM34 to the AVR

Signal Conditioning

- The most common transducers produce an output in the form of voltage, current, charge, capacitance and resistance.
- We need to convert these signals to voltage, in order to send input to an A-to-D converter. This conversion is called signal conditioning.
- Signal conditioning can be current – to – voltage conversion or signal amplification.
- Thermistor changes resistance with temperature.
- The change of resistance must be translated into voltages by ADC.

The following diagram shows the pin configuration of the LM34/LM35 temperature sensor and the connection of the temperature sensor to the ATmega32.



- The *A*/D has 10-bit resolution with a maximum of 1024 steps, and the LM34 (orLM35) produces 10mV for every degree of temperature change.
- Now, if we use the step size of 10mV, the Vout will be 10, 240mV (10.24 V) for full scale output.
- This is not acceptable even though the maximum temperature sensed by the LM34 is 300 degrees F, and the highest output we will get for the *A*/D is 3000mV(3.00V).
- If we use the internal 2.56V reference voltage the step size would be 2.56V/1024=2.5mV. This makes the binary output number for the ADC four times the real temperature.
- We can scale it by dividing it by 4 to get the real number for temperature.