MICRO CONTROLLERS AND INTERFACES

1. Difference between Micro controller and Micro processor:

Microprocessor:

- Microprocessor is a general purpose processor such as intelX86 family such as 8086,80286,80386,80486 and pendium and motrolo family are 680X0 such as 68000,68010,68020 etc.
- Mp is the heart of computer system and mainly used in computer system.
- Microprocessor doesn't contain no RAM, noROM,and no I/P-O/P PORTS on chip itself.
- A system designer using this general purpose processor by adding RAM, ROM , I/P&O/P PORTS and timers to make them functional and size become bulky and cost becomes expensive.
- Since memory and I/O connected externally the circuit becomes large.
- It has less number of registers hence more operations on memory based.
- Mp doesn't having power saving features.
- It's clock speed is low i.e(30-50MHZ)
- It used in Real time applications.

Microcontroller:

- Micro Controller has CPU (Micro Processor) in addition to a fixed amount of RAM, ROM, I/P-O/P PORTS and timers on a single chip.
  Eg:TV Remote control, Telephone, Music instruments, printer etc.
- Microcontroller is the heart of Embedded system and mainly used in MP_3 player.
- These applications require I/P-O/P operations to read signals and turn on &off certain bits for this reason. They use processor IBP are known as ITTY-BITTY processor.
- It is a 40 pin dip and quard flat package (qft) and is require +5V power.
- In a embedded system the microprocessor and micro controller are widely used in embedded system products to do one task.
  Eg:Printer.
- It has unique instruction set and register and they are not compactable to each other and their also has 16 bit and 32 bit processor.
- In 1981 IntelCorporation introduce 8 bit micro controller. It's having
  128 bytes RAM

4K bytes ROM
1-serial port
4- I/P and O/P ports
6- Interrupt sources, 2- timers.

- It has 2 versions 8052 and 8031.
- In 8052 its having 256 bytes RAM
  8K bytes ROM
  3 timers
  32 I/P-O/P ports
  One serial port
  6 interrupt sources
- The program written in 8051 can run in 8052 but the vice-versa is not possible.
- In 8031 microcontroller its having 128 bytes RAM
  0K byte ROM
  32 I/P-O/P ports
  One serial port
  6 interrupt sources
  2 timers
- In 8751 micro controller the flash memory can be erased in **20 min** are more whereas 8051 requires **few seconds**.
  Eg:→Atmel 89C51-12PC
  →At89C51-16PC
  →At89C51-32PC
- There are 4 major companies for 8 bit micro controller are free scale 6811, Intel 8051,Zilog 8051,PIC 16X.
- The 8051 Phillips company feature are A/D,D/A extended for I/P-O/P port on time programmable(OTP) and flash.
- Since memory and I/O connected internally the circuit becomes small.
- It has more number of registers hence programme is easier to write.
- Microcontroller having power saving modes like idle & power saving mode.
- It's clock speed is high i.e(1GHZ)
- It used in Real time applications.

2.8051 Architecture:

- It was developed by using NMOS technology but these require more power to operate then Intel re-design micro controller using CMOS technology.
- It is an 8-bit B register to perform arithmetic and logic operations.
- DPTR (Data pointer) is used for control the program instruction and can specified by its 16-bits.The 8051 have a built in RAM for internal processing. This memory is primary memory and is used for storage of temporary data (volatile memory) i.e., it contains gets vanish when power is turned off.

CPU:

CPU is the brain of processing device. It monitors and controls all operations that all perform in micro controller.

INTERRUPTS:

- It is a sub routine device call that interrupts micro controller main operation or work causes it to execute some other program which is more important at that time. The feature of interrupt is very useful as it help in cases of emergency.
- There are 5 interrupt sources, 2 of them are external and 3 of them are internal interrupts.

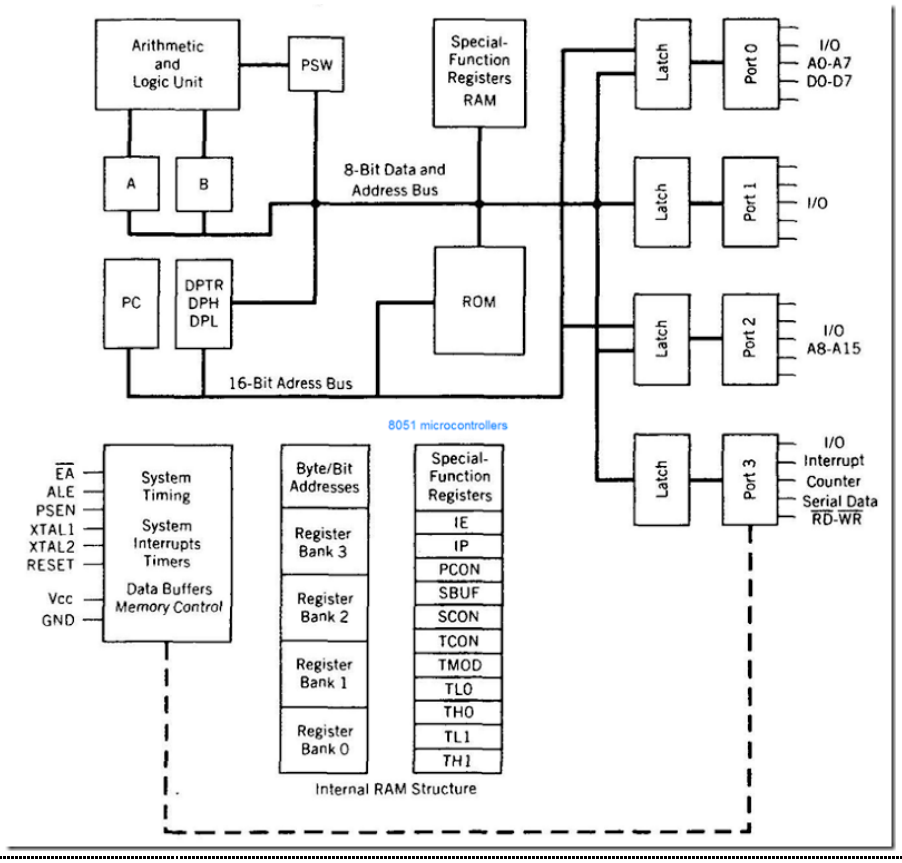
MEMORY:

Memory is a storage device to store data.

- It is used to store the program of micro controller is known as code memory or program memory (ROM).
- It is used to store data or operands temporarily then it is known as data memory(RAM).

BUS:

- It is a collection of wires which work as a communicate channel or



medium to transfer the data.

- The bus consists of 8 or 16 wires.

→Address Bus:

It is a uni-directional (16 bit).It transfer data from CPU to memory(A0-A7).

→Data Bus:

It is a bi-directional (8 bit).It carry data from cpu to memory vice-verssa(D0-D7).

Control bus: It controls the clock frequency between AB&DB.

OSCILLATOR:

- It is used to generate clock pulse to synchronize all internal operations.
- It generates frequency range from minimum to maximum (1MHz-12MHz).
- Generally quartz crystals are used for stability(11.0592MHZ) whereas ceramic oscillators are unstable.

PROGRAM STATUS WORD (PS):

- It is a 8-bit register referred as flag register.
  psw.7 psw.6 psw.5 psw.4 psw.3 psw.2 psw.1 psw.0

Cy	AC	FO	RS1	RS0	OV	-	P

- In this 6 bits used in 8051 and 2 bits are user definable flags and 4 flags are conditional flags.

→Carry Flag:

This flag is set whenever there is a carry out from D_7 bit.It can directly set to 1 or 0 by using SET C-1, CLR c-0(addition, subtraction).

→Auxiliary Carry Flag:

If there is a carry from D_3 to D_4 during addition or subtraction operation, This bit is set or clear (BCD operation).

→Parity Carry Flag:

1. It reflects no. of 1's in accumulator register only.

2.If A register contains odd no. of 1's then p=1.

3.If A register contains even no. of 1's then p=0.

→Overflow Flag:

1.This flag is set whenever the result of signed number operations is too large.

2.The overflow flag detect errors in signed numbers whereas carry flag detect errors in unsigned numbers.

→FO Flag:

It is used for general purpose.

Rs1	Rs0	Register Bit	Address
0	0	0	00H-07H
0	1	1	08H-0FH
1	0	2	10H-17H
1	1	3	18H-1FH

Eg: Adding 38 and 2FH

0011 1000

0010 1111

0110 0111

Carry=0(since there is no carry beyond D7 bit.

Auxiliary carry=1(since there is carry from D3 to D4 bit.

Parity flag=1(A has five 1's).

→Timer-Counters:

It has two 16 timers and counters .The timers are used for measurement of intervals to determine pulse width etc.

→I/P-O/P ports:

It has 4 ports port 0,1,2,3.Each port is 8 bit wide. It is used in embedded systems to control the operation of systems.

3.Types of Micro-Controller:

Micro controller is classified into 4 types depending on memory, architecture and instruction set. They are

- 4 bit:
  It is small in size, minimum pin count,low cost and it is used in low end applications like LED, LCD, display drivers and portable battery charges etc. It is a 20 pin dip with 4K ROM,256 bytes RAM, 14 I/O pins,2 counters. Its Intel family is RENASA 34501.

- 8 bit:
  It is most popular micro controller. Above 55% CPU's sold in world are 8 bit micro controllers. It has 8 bit internal bus 256 bytes RAM,4K ROM, 32 bit I/O ports, 2 timers/counters and its Intel family is 8051. Intel family designs a micro controller in the year 1981.
- 16 bit:
  It has 16 bit ALU at an instruction. It is suitable for high languages like C++ and its Intel family is 8096.
- 32 bit:
  It has 32 bit internal bus and is used in high end applications like robotics, communication networks, cell phones etc.
  Eg: PIC 32,ARM 7,and ARM 9.

4.Micro controller development tools:

The various development tools are required for microcontroller programming.

- Editor:
  An editor is a program which allows to create a file containing the assemble language statements for the program.
  Eg: As he writes the program the editor stores the ASCII codes for the letters and numbers in successive RAM location. After typing the entire program we have to save the program this we call it as source file.
  The next step is to process the source file with an assembler. Eg: sample.asm.
- Assembler:
  An assembler is used to translate to assemble language mnemonics into machine language(binary code) when we run the assembler it reads the source file of your program where we have saved it. The assembler generates a file with an extension ".hex"(hex decimal codes).
- Compiler:

It converts a high level program like C into binary or machine codes.
Eg: Kiel, Ride, IAR work bench or popular.

- Debugger:
  1)A debugger is a program which allows stopping the program for each instruction then it is known as single step debug(static).
  2)A debugger allows to set a break point in the program the debugger will run the program up to the instruction where the break point is given then it stop the execution. Hence it known as break point debug(dynamic).
  Eg: 10 instructions, 20 instructions…….

5.Special Function Register:

There are 21 SFR used in micro controller 8051. This includes register A, register B, PSW, PCON etc. There are 21unique locations each of these register is one byte size.

Sum of this SFR are bit addressable(which means wecan access individual bits inside a single bit) while some other are byte addressable.

- Accumulator:
  The accumulator holds the result of both for arithmetic and logic operations. It is an 8 bit byte. Accumulator is usually accessed by direct addressing and it physical address is $E0_H$. Accumulator is both bit and byte addressable.

Acc.7	----------	Acc.1	Acc.0

- B-Register:
  The major product of register executing its MUL and DIV. In 8051 the multiplication is done by repeat subtraction. It is used for general purpose operation for both bit and byte address and physical address is $F0_H$.

B7	----------	B1	B0

- Port Register:
  It having 4 I/O ports named as P_0,P_1,P_2,P_3. Data must written in port registers must be first to send it out any other external devices through ports. All 4 ports are bit and byte address. Po-80,P1-90,P2-A0,P3-B0

- Stack Pointer:
  It is an 8 bit register the direct address of stack pointer is 81_H. It is only byte addressable. Stack pointer having two instructions PUSH&POP(LIFO).
  In PUSH the stack pointer is used to increment by 2(SP+2).
  In Pop the stack pointer is used to decrement by 2(Sp-2).
- PCON(Power Management Register):
  We can see every day it is used in mobile phones. It is commonly referred as Power management alone. It has 2 modes-ideal mode, power down mode.

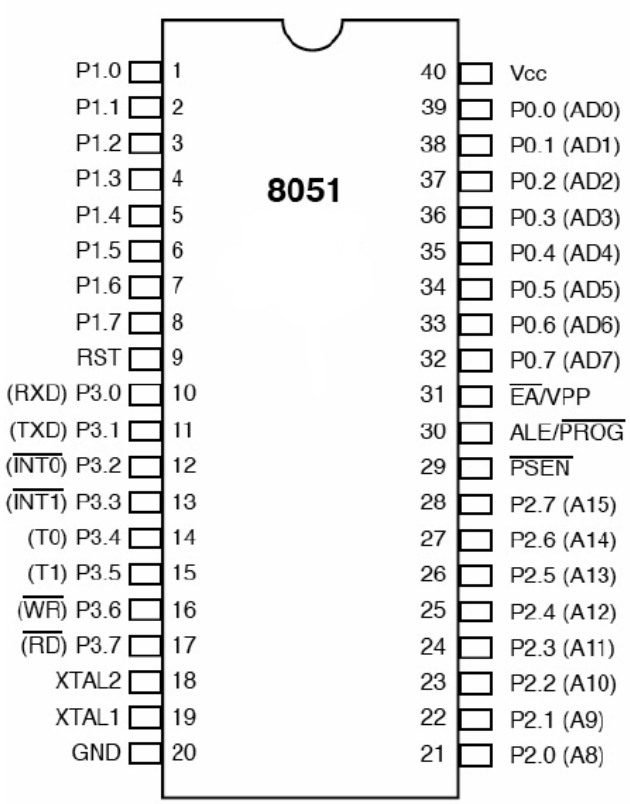
SMOD	----------	GP1	GP2	PDL	IDL

Setting bit '0' we will move the micro controller into ideal mode.

Setting bit '1' we will move the micro controller into power down mode.

6. Pin Diagram of 8051:

- 8051 family members are 8751,89C51,89C52,DS89CX40(all belongs to Intel). Its available packages are DIP, QFP, LLC(lead less carries). They are all having 40 pins.
- It must be noted that some companies made a 20 pin version of 8051 by reducing no. of I/O pins for less applications. Majority of developers use 40 pin dip. The pins are designated as VCC, GND, XTAL2, XTAL1, EA, PSEN, ALE are used by all members of 8051 and 8031 families.
- It is a 40 pin dip provides a supply voltage to the chip is +5V.
- 8051 is a chip oscillator but requires an external clock to run it. Most often we use Quartz crystal rather than TTL Oscillator.
- It also needs 2 capacitors "30pf" values. One side of each capacitor value is connected ground.



RST(Reset pin):

It is a input pin , active high (normally low). It is often referred as power on reset by activating this pin it will set the program counter to all 0's.

EA(External Access):

In 8051 family members all come with one chip ROM to store program. In such case

EA connected to ground or VCC. It cannot be left unconnected.

PSEN(Program stored Enable):

In 8031 or 8051 micro controller systems in which external ROM holds the program code.

ALE(Address latch enable pin):

It is an output pin and active high. It can MUX both address and data through port 0 to save pins. It is used for de-muxing the address and data by connecting to the chip 4LS373 chip.

Ports:

It having 32 bit I/O ports and each port is 8 bit wide.

Port 0:

If ALE=0 it provides data D_0 to D_7 when ALE=1 it has address A_0 to A_7. Port 0 provides both address and data. It also designated as AD_0 to AD_7.

In 8051 there is no external memory connection the pins of port 0 are connected to pull-up resistor(10KΩ).

Port 1&2:

Port 1 & 2 acts as simple I/O. In 8051 or 8031 basic system port 2 must be used along with port 0 to provide 16 bit address. P2 designated as A_8 to A_{15}&p0 designated as-A_0-A_7.For p1,p2and p3 does not required external pull-up resistors because they connected internally.

Port 3:

Its pins are 10 to 17. It can be used as input (or) output. It has addition function providing some external important systems such as interrupts.

Ports	Pins	Description	
$P_{3.0}$	10	RXD	serial communication signals
$P_{3.1}$	11	TXD	serial communication signals
$P_{3.2}$	12	INT0	external interrupts
$P_{3.3}$	13	INT1	external interrupts
$P_{3.4}$	14	T_0	Timer 0
$P_{3.5}$	15	T_1	Timer 1
$P_{3.6}$	16	WR	write signal
$P_{3.7}$	17	RD	read signal

Machine cycle and crystal frequencies:

A no. of machine cycles is to take execution an instruction is not same for AT89651 and DS89C4X0. In this X1 and X2 dedicates the speed of clock in machine cycle.

$$\text{Machine Cycle} = \frac{\text{No. of clocks}}{\text{Frequency}} = \frac{1}{11.0592\ \text{MHz}} = 90.42\ \text{ns}$$

Chip	Clock per machine cycle
AT89C51	12
DS89C4X0	1
DS5000	4

7. 8051 Memory Organization:

- It is divided into 2 types and there are 1.Program Memory 2.Data Memory.
- Program memory is used for permanent saving the program being executed.
- Data Memory is used for temporarily storing data(register).
- In program memory is read only memory (ROM). Depending on setting made in compiler and also store a constant variable.
- 8051 allows external program memory to be added.
- Microcontroller handle external memory depends on EA logic state. If EA=0,the external ROM memory is 64K.
- If EA=1 the additional ROM memory 64K+4K embedded memory.

Internal Memory:

Up to 256 bytes of internal data memory are available in 8051.

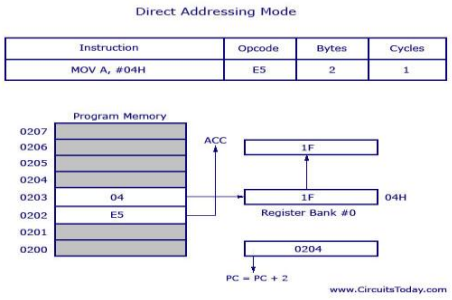
The first 128 bytes of internal memory are both direct and indirect addressable.

The upper 128 bytes of data memory is 0X80 to 0XFF can be addressed only indirectly.

The memory block is in the range of 20H to 2FH is bit addressable which means each bit mean where has its own address from 0 to 7FH.

Since there are 16 registers the block contains total of 128 bits with separate address. In this 3 memory type specifies is used to refer the internal data memory data, idata, bdata.

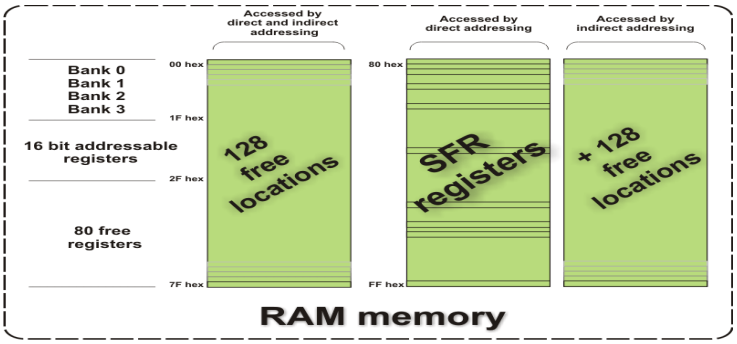
External data memory:



It is slower than internal data memory and made up of 64K bytes of external data memory setting of the registers must be manually done in code, before any access to external memory (or) X-RAM space is made. It has two specifiers i.e X-data, P-data.

SFR memory:

8051 provide 128 bytes of memory of SFR memory are bit, byte or word sized registers that are used to control timers, counters, serial I/O, Port I/O and peripherals.



8. Addressing Modes:

Addressing mode is a way we address and operand means a data we are operating a source (Register,address of memory,any numeric data). It is divided into five types.

1.Immediate addressing mode.

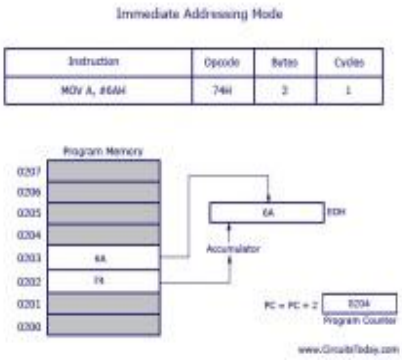
2.Direct addressing mode.

3.Register addressing mode.

4.Register index addressing mode.

5.Index addressing mode.

Immediate Addressing mode:



It can transfer data immediately to accumulator. The opcode for MOVA,#data is 74. The opcode saved in program memory at 0021. The instruction required 2 bytes to execute in 1 cycle. So, after the execution of instruction the program counter will add two to and move to 0024.

Instruction	Opcode	Bytes	Cycles
MOVA,#6A	74	2	1

Note: The # symbol before 6A indicates the operand is an 8 bit data, when # is not present it taken as hexadecimal.

Direct Addressing Mode:

It is another way of addressing the data. Here an address of data(resource) is given as operand.

Eg: MOV A,04.

Here 04 is address of register 4 (bank 0) when it is executed.

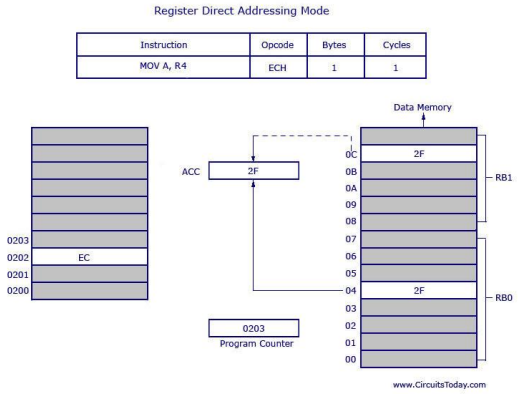
Instruction	Opcode	Bytes	Cycles
MOV A,04	E5	2	1

Note: If # is gives the data value of 04H transfer to accumulator instead of 1FH.

Register Direct Addressing Mode:

It is used to move the data from the register to accumulator.

Eg: MOV A,R4.

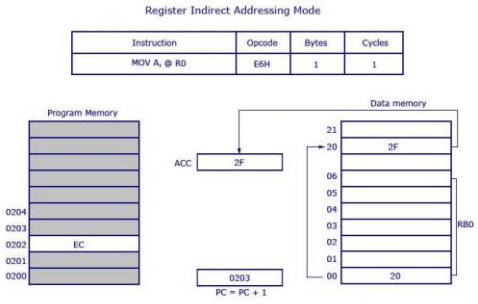


At a time register can take value from R0 to R7 In 8051 we are having 32 such registers & 4 register banks(0,1,2,3) . Each bank has 8 registers name R0 to R7 selection of register bank is made to PSW (SFR)(PSW.4,PSW.3)

Instruction	Opcode	Bytes	Cycles
MOV A,R4	EC	1	1

Register Indirect Addressing Mode:

In this address of data is to transfer to an accumulator. Here the value inside R0 is considered as an address which holds the data to transfer in to accumulator.

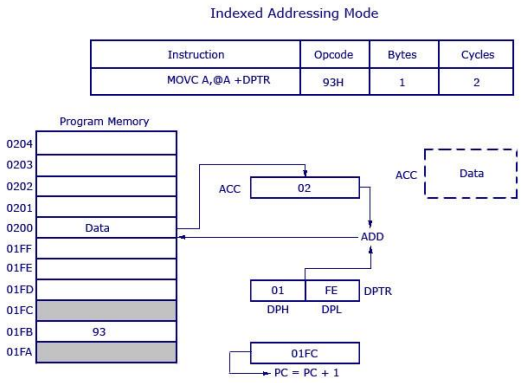


Eg: MOV A,@R0.

In this R0 holds the value 20H we have data 2FH store at address 20H then the value of 2FH is get transferred to accumulator after executing the instruction.

Instruction	Opcode	Bytes	Cycles
MOVA,@R0	F6	1	1

Index Addressing Mode:

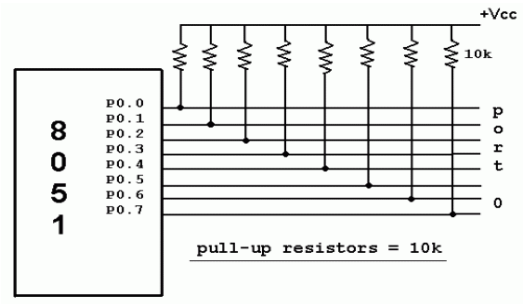


The source operand @A+DPTR and we get the data from this location it is nothing but adding contents of DPTR with current content of accumulator . DPTR holds the value 01FE where 01 is located in DPH(high order 8 bits) and FE is located in DFC(low order 8 bits), Now accumulator has the value 02H the addition is 16 bit and now 01FE+02 results in 0200H will get transfer to accumulator. It is a one-byte instruction and 2 cycles need for execution.

Instruction	Opcode	Bytes	Cycles
MOVA,@A+DPTR	93H	1	2

9. Port Organization:

8051 has four parallel input /output ports. It provides the user 32 input output lines for connecting the microcontrollers to peripherals.In order to make them input all ports must be SET i.e., high bit must be send to all port pins. It is normally done by using "SETB".



Port 0:

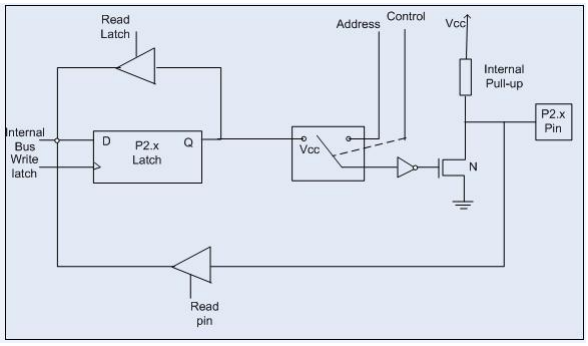
It is an 8 bit input /output port with dual purpose if external memory is used these pins are used for low address byte (AD_0 to AD_7). Unlike other ports, port '0' is not provided by internal pull up resistors . So, that 10k are connected externally. **Port-0 can be used as a both bidirectional I/O port (or) address/data interfacing for accessing external memory. (i)When control is '1', the port is used for address/data interfacing.**

(ii) When the control is '0', the port can be used as a bidirectional I/O port.

Eg: MOV A,#0FFh

MOV P_0, A

Port 1:



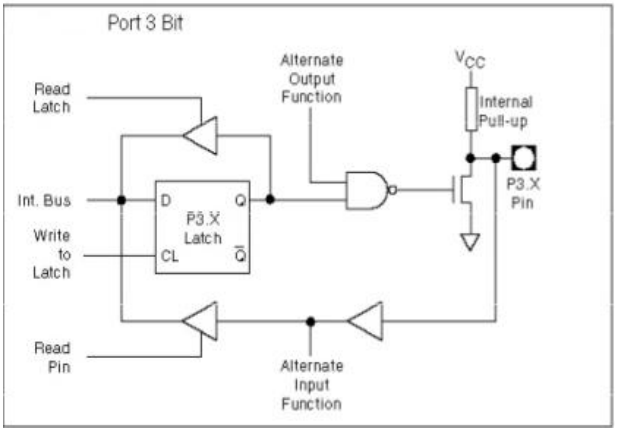
It acts as input or output. If we reset, port 1 configured as output port. If it is set, port 1 act as input port. **The pin is pulled up or down through internal pull-up when we want to initialize as an output port. To use port-1 as input port, '1' has to be written to the latch. In this input mode when '1' is written to the pin by the external device then it read fine. But when '0' is written to the pin by the external device then the external source must sink current due to internal pull-up. If the external device is not able to sink the current the pin voltage may rise, leading to a possible wrong reading**

Ex: Mov A,#0FFh

Mov P_1, A

Port 2:

It is an 8 bit parallel port. It does not require external resistors. It is used as input or output port. If we reset, port 2 acts as output port. If we set port 2 acts as input port. **Port-2 we use for higher external address byte (or) a normal input/output port. The I/O operation is similar to Port-1. Port-2 latch remains stable when Port-2 pin are used for external memory access. Here, again due to internal pull-up there is limited current driving capability.**



Ex: MOV A, #0FFh

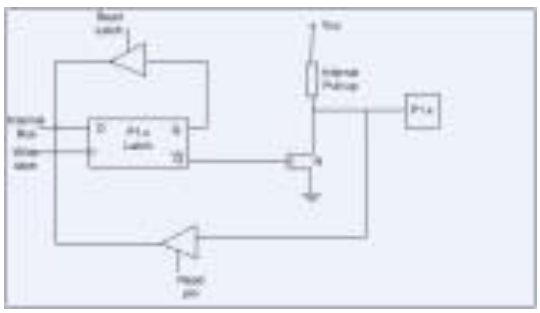
MOV P_2, A

Dual Port:

It is associated with high order address lines. In systems like 89C51, DS5000, port is used as simple input output port. Where as in 8031 basic system, ports 2 is used along with port 0 and provides 16 bit address. The external memory is P_0- (A_0 A_7), P_2=(A_8 A_{15}).In 8031 , it is connected to external memory and it is use upper 8 bits of 16 bit address and cannot be used for input output ports.

Port 3:

It is an 8 bit parallel port with dual function. It is used for input output operation as well as for control operations.If we Reset, port 3 acts as output port.If we Set, port 3 acts as input port.



Eg: MOV A,#0FFh

MOV P_3, A

Alternative function:

P3.0	RxD
P3.1	TxD
P3.2	INT0 bar
P3.3	INT1 bar
P3.4	T0
P3.5	T1
P3.6	WR bar
P3.7	RD bar

10. Interrupt structure:

An interrupt is an external or internal event that disturbs the micro controller to inform it that a device needs its service. The program which is associated with the interrupt called ISR (Interrupt service routine).Upon receiving the interrupt signal, the micro controller finish the current instruction and save pc on stack.Fixed location to memory depending on interrupt until RET, the micro controller returns to place where it was interrupt get pop from the stack.

It has five interrupt sources timer0, timer1. External interrupts are INT_0, INT_1, Serial port events.Each interrupt has a specific place in code memory where program execution ISR begins.

INT_0-0003H, INT_1- 0013H, T_0-000BH, T_1-001BH

Upon Reset all interrupts are disable and do not respond to micro controller. This interrupts must be enable by software in order to respond and this is done by IER (Interrupt Enable Register).

EA	–	–	ES	ET1	EX1	ET0	EX0

EA	IE.7	Disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, interrupt source is individually enable or disabled by setting or clearing its enable bit.
-	IE.6	Not implemented, reserved for future use*.
-	IE.5	Not implemented, reserved for future use*.
ES	IE.4	Enable or disable the Serial port interrupt.
ET1	IE.3	Enable or disable the Timer 1 overflow interrupt.
EX1	IE.2	Enable or disable External interrupt 1.
ET0	IE.1	Enable or disable the Timer 0 overflow interrupt.
EX0	IE.0	Enable or disable External Interrupt 0.

Upon Rest the interrupts have been following priority from top to bottom. The interrupt which are having high priority can be serviced first INT_0, IT_0, INT_1, IT_1, Serial communication R_1- T_1.

IP :

If th
prior
high

-
-
-
PS
PT1
PX1
PT0
PX0

Timer 0

11. Timers in 8051:

It has 2 timers (T_0 and T_1) which are 16 bit timers. These timers are accessed as 2 8-bit registers. T_{LO} , T_{HO} and T_{L1} , T_{H1}.

It is used to generate accurate time delays or event counters.

Timer '0': It is a 16 bit register and can be treated as two 8-bit register (T_{LO} , $T_{HO)}$ and this registers' can accessed similarly registers like A, B (or)R_0, R_1 etc.

Ex:

MOV T_{LO},#07 move the value 07 to low order byte timer T_0 .

MOV R_5, T_{HO} saves the contents of T_{HO} to R_5 register.

Timer 1:

It is a 16 bit register and can be treated as two 8-bit registers(T_{L1} , T_{H1}) and this register can accessed usually to any other register like A, B(or) R_0, R_1 etc.

Ex:

MOV T_{L1},#07 move the value 07 to low order byte timer T_1.

MOV R_5, T_{H1} saves the contents of T_{H1} to R_5 register.

TMOD Register:

The various operating modes of both the timers T_1 and T_0 are set by 8 bit register called TMOD Register.

In this lower 4 bits are meant for T_0 and higher 4 bits are meant for T_1.

TMOD : Timer/Counter Mode Control Register (Not Bit Addressable)

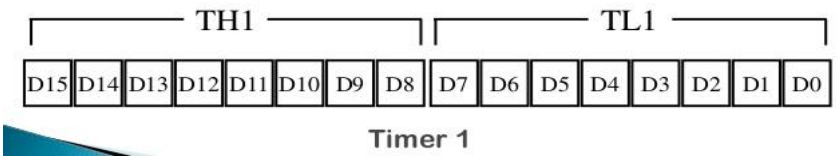
GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0
TIMER 1				TIMER 0			

Gate:

- This bit is used to start or stop the timers by hardware.
- When gate=0, the timer can start or stop by software instructions like SETB TR0 , SETB TR1
- When gate=1, the timer can start or stop by external sources.

C/T:(CLOCK/TIMER)

- This bit decides whether the timer is used as delay generator or event counter.
- C/T=0 then the timer is used as delay generator.
- C/T=1 then the timer is used as event counter.



M_1 and M_0: The two bits are timer mode bits .The timers of 8051 can be configured in 3 modes.mode0,1,2.

M_0	M_1	MODE	DESCRIPTION
0	0	0	13-BIT timer mode/8-bit timer/counter
0	1	1	16-bit timer mode, 16-bit timer/counter
1	0	2	8-bit Auto re-load.
1	1	3	Split timer mode

TCON Register:

- 8051 has two 16-bit timers/counters
- In this the timer registers is incremented once every machine cycle.
- In this count is about (1/12) oscillator frequency.
- In case of counter register is incremented in response to a 1-0 transition to its corresponding external input pin. In this function, the external input is sampled during s5-p2 of every machine cycle.
- Address is 88H(bit addressable)

TCON : Timer/Counter Control Register (Bit Addressable)

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1	TCON.7	Timer 1 overflow flag. Set by hardware when the Timer/Counter 1 overflows. Cleared by hardware as processor vectors to the interrupt service routine.
TR1	TCON.6	Timer 1 run control bit. Set/cleared by software to turn Timer/Counter ON/OFF.
TF0	TCON.5	Timer 0 overflow flag. Set by hardware when the Timer/Counter 0 overflows. Cleared by hardware as processor vectors to the service routine.
TR0	TCON.4	Timer 0 run control bit. Set/cleared by software to turn Timer/Counter 0 ON/OFF.
IE1	TCON.3	External Interrupt 1 edge flag. Set by hardware when External interrupt edge is detected. Cleared by hardware when interrupt is processed.
IT1	TCON.2	Interrupt 1 type control bit. Set/cleared by software to specify falling edge/flow level triggered External Interrupt.
IE0	TCON.1	External Interrupt 0 edge flag. Set by hardware when External Interrupt edge detected. Cleared by hardware when interrupt is processed.
IT0	TCON.0	Interrupt 0 type control bit. Set/cleared by software to specify falling edge/low level triggered External Interrupt.

SCON Register:

Address is 98H (Bit addressable)

SM0	SM1	SM2	REN	TN8	RB8	TI	RI

SM0	SCON.7	Serial Port mode specifier (NOTE 1).
SM1	SCON.6	Serial Port mode specifier (NOTE 1).
SM2	SCON.5	Enables the multiprocessor communication feature in mode 2 & 3. In mode 2 or 3, if SM2 is set to 1 then RI will not be activated if the received 9th data bit (RB8) is 0. In mode 1, if SM2 = 1 then RI will not be activated if a valid stop bit was not received. In mode 0, SM2 should be 0 (See table 9).
REN	SCON.4	Set/Cleared by software to Enable/Disable reception.
TB8	SCON.3	The 9th bit that will be transmitted in modes 2 & 3. Set/Cleared by software.
RB8	SCON.2	In modes 2 & 3, is the 9th data bit that was received. In mode 1, if SM2 = 0, RB8 is the stop bit that was received. In mode 0, RB8 is not used.
TI	SCON.1	Transmit interrupt flag. Set by hardware at the end of the 8th bit time in mode 0, or at the beginning of the stop bit in the other modes. Must be cleared by software.
RI	SCON.0	Receive interrupt flag. Set by hardware at the end of the 8th bit time in mode 0, or half way through the stop bit time in the other modes (except see SM2). Must be cleared by software.

Note 1 :

SM0	SM1	MODE	DESCRIPTION	BAUD RATE
0	0	0	SHIFT REGISTER	Fosc./12
0	1	1	8 bit UART	Variable
1	0	2	8 bit UART	Fosc./64 OR Fosc./32
1	1	3	8 bit UART	Variable

SMOD Register:

Serial bit mode is used to determine the Baud rate of Timer1.

Baud rate= Oscillator frequency in Hz / N [256-(TH1)]

If SMOD=0 then N=384

If SMOD=1 then N=192,

TH1 is high byte of timer 1 when it is 8in 8-bit auto re-load mode.

T_2CON:

Timer 2 control register.

TF_2	EXF_2	RCLK	TCLK	$EXEN_2$	TR_2	C/T_2	CP/RL_2

TF_2---Timer 2 over flow flag

EXF_2---Timer 2 external flag

RCLK----Receive clock. When set causes the serial port to use timer2 for reception.

TCLK—Transmit clock, when set causes the serial port to use timer 2 for transmission.

C/T_2----counter/timer , if 0 use internal timer, if 1 use external pin.

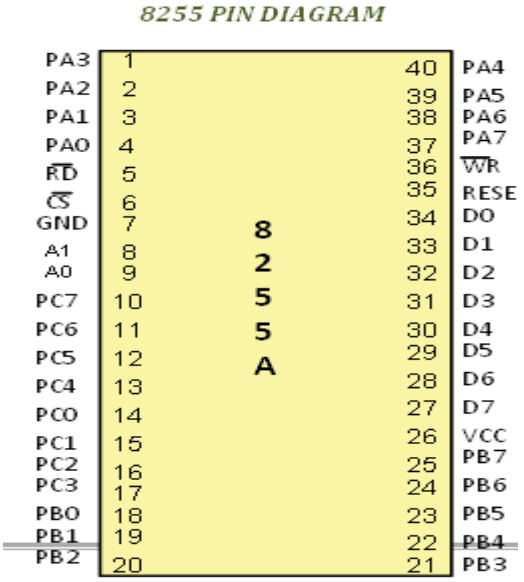
CP/RL_2-----capture/reload flag.

INTERFACING WITH 8051

Q1.8255A:

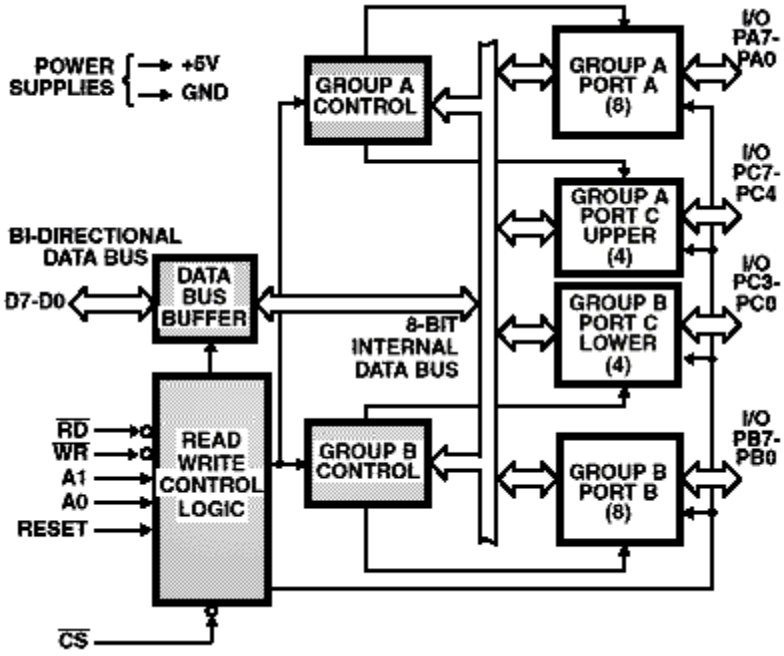
- It is a programmable peripheral interface (Parallel communication)
- It is a 40 pin dip terminal having a supply voltage of +5V.
- It has 24 input and output pins i.e., port A, B, C.
- Each port can be enabling by writing a control word in control register.
- It has two I/O modes and one BSR mode.
- I/O mode divided into three modes
    - i. simple input output
    - ii. Handshake (Control signals)
    - iii. Data transfer.

It transfers data from simple I/O to interrupt I/O.



D7	D6	D5	D4	D3	D2	D1	D0
	x	x	x				S/R

8255 is a SPI. Interface 8255 with AT89S51, AT89S52, AT89C51, AT89C52, AT89C2051, AT89C4051, and AT89C1051. Basically 8255 is a port expansion used to expand the port and its data with external circuitry. Every 8255 chip has 3 off 8 bit TTL logics input output port which can control 24 output signals or logic levels or reading 24 individual inputs or outputs.



Functions of PPI 8255 Features:

8255 is a 40 pin DIP chip that is used to expand the ports of microcontrollers. It has three 8 bit ports that can be accessed separately having the name Port A, Port B and Port C in short PA, PB and PC of 8255. Ports can be programmed separately and for sending or receiving data these 8255 ports can be changed dynamically using programming in assembly or KEIL C language. This chip also includes handshaking features and functions to detect the conditions and signals. It can be interfaced with another devices having handshaking capability 8255.

Interfacing PA0 - PA7 Ports 8255A with 89C51 Microcontroller:

The 8 bit port of 8255 A can be used for dual purpose for input and output for 8051 microcontrollers. PA 8255 is bidirectional port can be programmed easily to send or receive data from external devices like memories, chips and electronic devices.

Interfacing PB0 - PB7 Port 8255 with 89C52 Microcontroller:

Port B of 8255 is also a dual role in getting and receiving data from external devices and is programmed as a bidirectional port using assembly language or C language. Port PB of 8255 is a 8 bit port and 8 bits can be handled individually with 8255 and 89S51 or 89S52 Microcontrollers.

Interfacing PC0 - PC7 Port 8255A with 89S51 Microcontroller:

Port C of 8255 chip is also a bidirectional 8 bit port. Another function of 8255 Port C, that it can be split into two parts to be programmed. One of them is CU 8255, upper bits PC4 - PC7 and CL 8255 lower bits PC0 - PC3. Each can be used as a input or output port to connect with 89C51, 89C52 Microcontrollers.

Interfacing D0 -D7 Data Pins 8255 Chip with 89S52 Microcontroller:

This is the data pins 8255 can be connected with 89S51, 89S52, 89C51, 89C52, 89C2051, 89C4051, 89C1051, any of the microcontroller to send or receive data over this line. Data port of 8255 can be used to send data or receive between AT89S51, AT89S52, AT89C51, AT89C52, AT89C2051, AT89C4051, AT89C1051 microcontroller.

RD, WR And RESET 8255 Chip with 89C1051, 89c2051 and 89c4051 Microcontroller:

RD and WR signals refers to reading and writing to 8255A IC with active low signal functionality. RD and WR lines of 8255 are connected with RD and WR signals of 8051 microcontroller or its other variants. RESET is an active high signal used to reset and clear the internal registers of 8255 chip. When we set RESET set to active high all ports of 8255 are set as input ports. This pin can be set grounded with the circuitry; It can be be left unconnected.

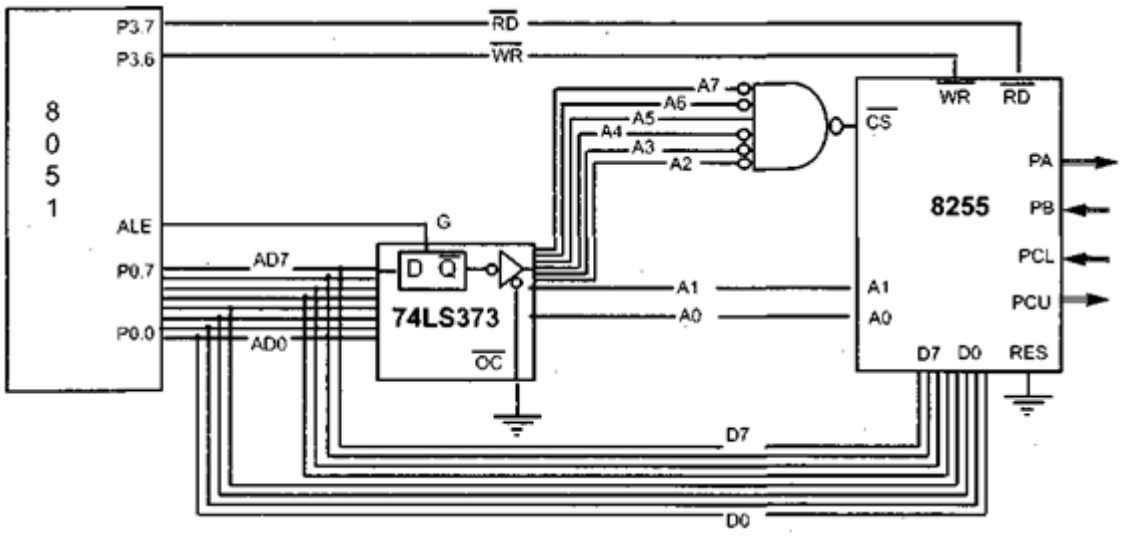
CS	A1	A0	SELECTION
0	0	0	Port A
0	0	1	Port B
0	1	0	Port C
0	1	1	Control register
1	x	x	8255A not selected

A0, A1 and CS or Chip Select 8255:

CS is used to select the chip and makes it works, A0 and A1 select a particular port of 8255 to send or receive data over lines. These three lines are used to access a particular port from 3 ports A, B and C of the chip.

8255 Hardware Connection with 8051 Microcontrollers:

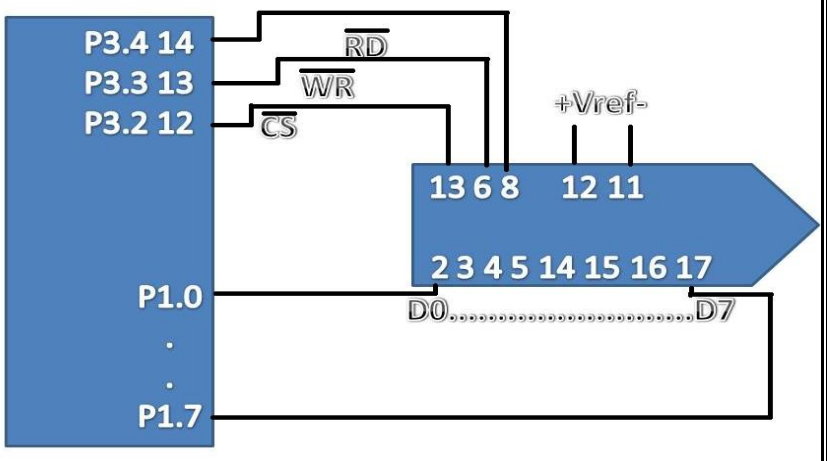
8255 chip can be interfaced very easily with 8051 microcontroller using its data lines and all other important lines with the external circuit. Following is the hardware diagram can be used to create a circuit used to interconnect with 8255 using Atmel microcontrollers.



Q2:ADC0804:

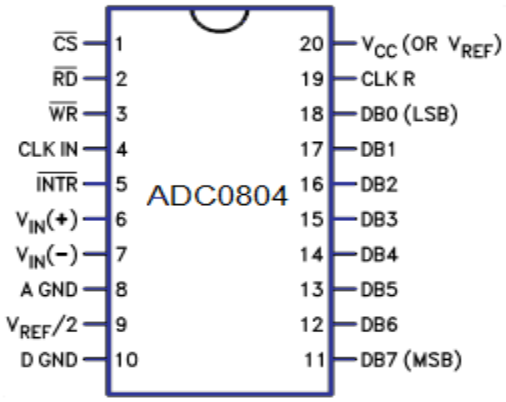
A/D conversion: low power, high performance, C MOS. The easiest A/D converters to use one flash types which make conversions based on array of internal comparators. The conversion is very fast typically in less than 1 micro second. Thus the converter can be told to start and digital equivalent of input analogue value will be read one (or) more instructions later. Modern successive approximation register (SAR) converters do not lag far behind however with conversion times in the 2-4 micro second range for 8-bits. Flash converters are more expensive(by a factor of two)than the traditional SAR types.

ADC 0804 is an 8-bit successive approximation analogue to digital converter from national semiconductor.
Its voltage inputs 0-5v input voltage range, no zero adjustment, built in clock generator, reference voltage can be externally adjusted to smaller analogue voltage span to 8-bit resolution etc.,



ADC (Analog to digital converter) forms a very essential part in many embedded projects and this article is about interfacing an ADC to 8051 embedded controller. ADC 0804 is the ADC used here and before going through the interfacing procedure, we must neatly understand how the ADC 0804 works. ADC0804 is an 8 bit successive approximation analogue to digital converter from National semiconductors. The features of ADC0804 are differential analogue voltage inputs, 0-5V input voltage range, no zero adjustment, built in clock generator, reference voltage can be externally adjusted to convert smaller analogue voltage span to 8 bit resolution etc. The pin out diagram of ADC0804 is shown in the figure below.

Output pin:-



The voltage at Vref/2 (pin9) of ADC0804 can be externally adjusted to convert smaller input voltage spans to full 8 bit resolution. Vref/2 (pin9) left open means input voltage span is 0-5V and step size is 5/255=19.6V. Have a look at the table below for different Vref/2 voltages and corresponding analogue input voltage spans.

Steps for converting the analogue input and reading the output from ADC0804:

Make CS=0 and send a low to high pulse to WR pin to start the conversion.

Now keep checking the INTR pin. INTR will be 1 if conversion is not finished and INTR will be 0 if conversion is finished.

If conversion is not finished (INTR=1) , poll until it is finished.

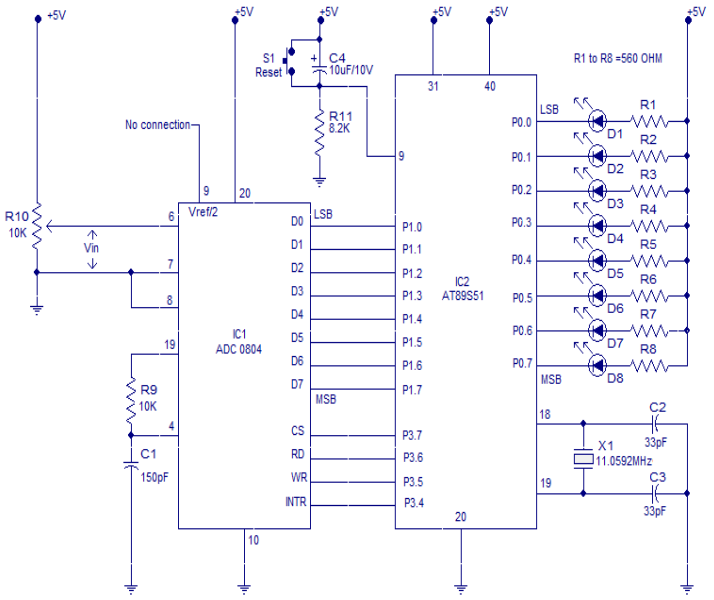
If conversion is finished (INTR=0), go to the next step.

Make CS=0 and send a high to low pulse to RD pin to read the data from the ADC.

CS : Chip Select: It is an active low input used to activate the ADC804 IC. To activate ADC804, this pin must be low

RD : Read: It is an active low input used to get the converted data out of the ADC chip. The ADC converts the analog input to its binary equivalent and holds in an internal register. When a CS = 0 and high-to-low pulse is applied to the RD pin, then 8-bit digital output is available at the D0-D7 data pins.

WR : Write: This is an active low input used to inform the ADC to start the conversion process. ADC starts converting analog input to digital, when CS = 0 and a low-to-high pulse is applied to WR pin. The amount it takes to convert varies depending on the CLK IN and CLK R values. When the data conversion is complete, the INTR pin is forced low by the ADC804.



Interfacing Circuit ADC 804 Analog To Digital Converter with 8051 Microcontroller: Here ADC 0804 is connected to port1 of 8051. WR and INTR of ADC is connected to P3.4 and P3.5 respectively. Analog input is applied to pin 6 of

ADC. Here WR is the start of conversion and INTR is the end of conversion.

Interfacing ADC to 8051:

The figure above shows the schematic for interfacing ADC0804 to 8051. The circuit initiates the ADC to convert a given analogue input , then accepts the corresponding digital data and displays it on the LED array connected at P0. For example, if the analogue input voltage Vin is 5V then all LEDs will glow indicating 11111111 in binary which is the equivalent of 255 in decimal. AT89s51 is the microcontroller used here. Data out pins (D0 to D7) of the ADC0804 are connected to the port pins P1.0 to P1.7 respectively. LEDs D1 to D8 are connected to the port pins P0.0 to P0.7 respectively. Resistors R1 to R8 are current limiting resistors. In simple words P1 of the microcontroller is the input port and P0 is the output port. Control signals for the ADC (INTR, WR, RD and CS) are available at port pins P3.4 to P3.7 respectively. Resistor R9 and capacitor C1 are associated with the internal clock circuitry of the ADC. Preset resistor R10 forms a voltage divider which can be used to apply a particular input analogue voltage to the ADC. Push button S1, resistor R11 and capacitor C4 forms a debouncing reset mechanism. Crystal X1 and capacitors C2,C3 are associated with the clock circuitry of the microcontroller.

Program:

```
ORG 00H

MOV P1,#11111111B // initiates P1 as the input port

MAIN: CLR P3.7 // makes CS=0

    SETB P3.6 // makes RD high

    CLR P3.5 // makes WR low

    SETB P3.5 // low to high pulse to WR for starting conversion

WAIT: JB P3.4, WAIT // polls until INTR=0

    CLR P3.7 // ensures CS=0

    CLR P3.6 // high to low pulse to RD for reading the data from ADC

    MOV A,P1 // moves the digital data to accumulator

    CPL A // complements the digital data (*see the notes)
```

```
MOV P0,A // outputs the data to P0 for the LEDs

SJMP MAIN // jumps back to the MAIN program

END
```

Note:

- The entire circuit can be powered from 5V DC.
- ADC 0804 has active low outputs and the instruction CPL A complements it t0 have a straight forward display. For example, if input is 5V then the output will be 11111111 and if CPL A was not used it would have been 00000000 which is rather awkward to see.

Q3: Interfacing LM35 Temperature Sensor 8051 Microcontrollers:

Interfacing Temperature Sensors with 8051 Microcontroller Family Models:--

Atmel, NXP, Philips, 8051, 8052, 89C51, 89C52, 89S51, 89s52, 89C1051, 89C1051, 89C2051, AT89C4051, AT89S8252,l AT89C1051, AT89C2051, AT89C4051,P89C51RB+, P89C51RC+, P89C51RD+, P89C51RB2Hxx, P89C51RC2Hxx, P89C51RD2Hxx, P89C660, P89C662, P89C664, P89C668, P89C669, P89C51RA2xx, P89C51RB2xx, P89C51RC2xx, P89C51RD2xx, P89C60X2, P89C61X2,P89LV51RB2, P89LV51RC2, P89LV51RD2, P89V51RB2, P89V51RC2, P89V51RD2, P89V660, P89V662, P89V664.

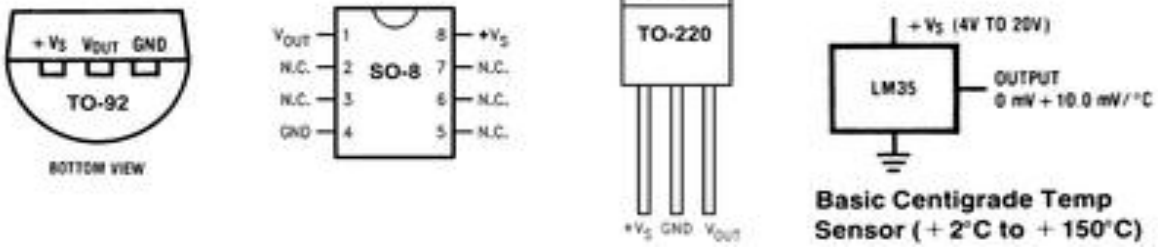
LM35 Temperature Sensor Introduction and Fundamentals:-

- The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature.
- The LM35 thus has an advantage over linear temperature sensors calibrated in ° Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling.
- The LM35 does not require any external calibration or trimming to provide typical accuracies of ±¼°C at room temperature and ±¾°C over a full -55 to +150°C temperature range.
- Low cost is assured by trimming and calibration at the wafer level. The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy.
- It can be used with single power supplies, or with plus and minus supplies.
- As it draws only 60 µA from its supply, it has very low self-heating, less than 0.1°C in still air.
- The LM35 is rated to operate over a -55° to +150°C temperature range.
- A digital thermometer can be easily created by using LM35 temperature sensor and can be interfaced any microcontrollers.

The LM 35 IC generates a 10mV variation to its output voltage for every degree Celsius change in temperature. The Output of the temperature sensor is analog in nature so we need an analog to digital converter for converting the analog input to its equivalent binary output. The ADC 0804 is the analog to digital converter IC used in the project. 0804 is a single channel converter which converts the analog input up to a range of 5V to an equivalent 8-bit binary output.

Features of LM35 Temperature Sensors - 8051 Microcontrollers:

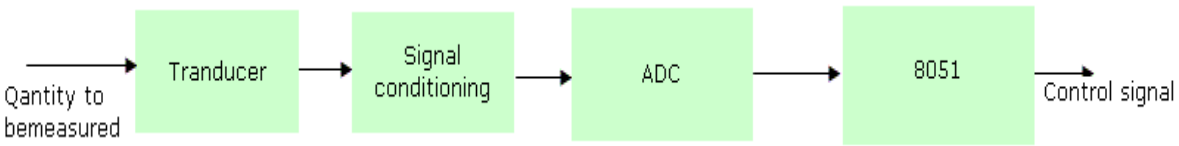
- Calibrated directly in ° Celsius (Centigrade)



- Linear + 10.0 mV/°C scale factor
- 0.5°C accuracy guarantee able (at +25°C)
- Rated for full -55° to +150°C range
- Suitable for remote applications
- Low cost due to wafer-level trimming
- Operates from 4 to 30 volts
- Less than 60 µA current drain
- Low self-heating, 0.08°C in still air
- Nonlinearity only ±¼°C typical
- Low impedance output, 0.1 Ohm for 1 mA load

Interfacing LM25 Temperature Sensor with 8051 Microcontroller:

This Digital Thermometer Design can be integrated by using the 8051 microcontroller and analog to digital converter IC ADC 804 and interfaced with any microcontroller AT8051, AT8052, AT89C51, AT89C52, AT89S51, AT89s52 and the output can be displayed on any output device may be computer monitor or any LCD display. The sensor used for this design is the LM35 which output an analogue voltage per centigrade Celsius. A circuit amplification is done between the LM35 and the microcontroller.

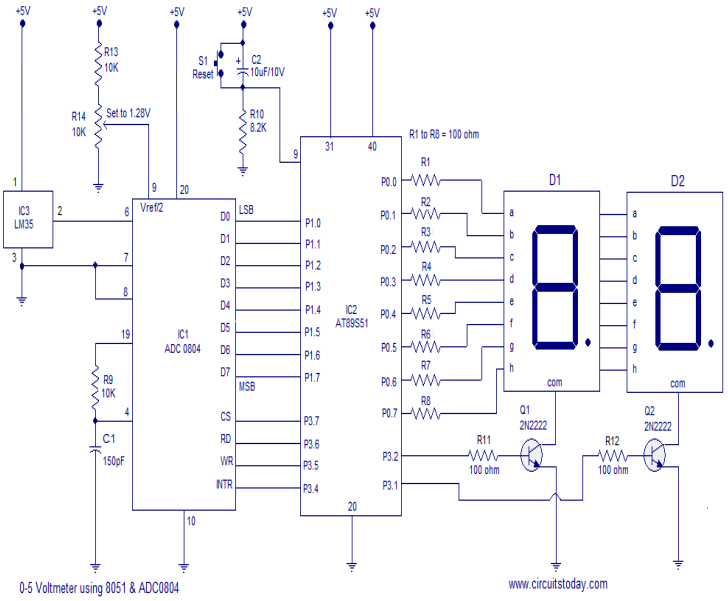


Q4: LCD:-

LCD display is an inevitable part in almost all embedded projects and this article is about interfacing 16×2 LCD with 8051 microcontroller. Many guys find it hard to interface LCD module with the 8051 but the fact is that if you learn it properly, it's a very easy job and by knowing it you can easily design embedded projects like digital voltmeter / ammeter, digital clock, home automation displays, status indicator display, digital code locks, digital speedometer/ odometer, display for music players etc. Thoroughly going through this article will make you able to display any text (including the extended characters) on any part of the 16×2 display screen. In order to understand the interfacing first you have to know about the 16×2 LCD module.

16×2 LCD module:

16×2 LCD module is a very common type of LCD module that is used in 8051 based embedded projects. It consists of 16 rows and 2 columns of 5×7 or 5×8 LCD dot matrices. The module were are talking about here is type number JHD162A which is a very popular one. It is available in a 16 pin package with back light, contrast adjustment function and each dot matrix has 5×8 dot resolution. The pin numbers, their name and corresponding functions are shown in the table below.



Pin No:	Name	Function
1	VSS	This pin must be connected to the ground
2	VCC	Positive supply voltage pin (5V DC)
3	VEE	Contrast adjustment
4	RS	Register selection
6	E	Enable

7-14	**DB0-DB7**	Data
15	**LED+**	Back light LED+
16	**LED-**	Back light LED-

VEE pin is meant for adjusting the contrast of the LCD display and the contrast can be adjusted by varying the voltage at this pin. This is done by connecting one end of a POT to the Vcc (5V), other end to the Ground and connecting the centre terminal (wiper) of the POT to the VEE pin. See the circuit diagram for better understanding.

The JHD162A has two built in registers namely data register and command register. Data register is for placing the data to be displayed, and the command register is to place the commands. The 16×2 LCD module has a set of commands each meant for doing a particular job with the display. We will discuss in detail about the commands later.

Register select: High logic at the RS pin will select the data register and Low logic at the RS pin will select the command register. If we make the RS pin high and the put a data in the 8 bit data line (DB0 to DB7) , the LCD module will recognize it as a data to be displayed . If we make RS pin low and put a data on the data line, the module will recognize it as a command.

R/W logic: R/W pin is meant for selecting between read and write modes. High level at this pin enables read mode and low level at this pin enables write mode.

Enable pin: E pin is for enabling the module. A high to low transition at this pin will enable the module.

Data bus: DB0 to DB7 are the data pins. The data to be displayed and the command instructions are placed on these pins.

Led +&Led:- LED+ is the anode of the back light LED and this pin must be connected to Vcc through a suitable series current limiting resistor. LED- is the cathode of the back light LED and this pin must be connected to ground.

16×2 LCD module commands:

16×2 LCD module has a set of preset command instructions. Each command will make the module to do a particular task. The commonly used commands and their function are given in the table below.

Command	Function
0F	LCD ON, Cursor ON, Cursor blinking ON
01	Clear screen
02	Return home
04	Decrement cursor
06	Increment cursor
0E	Display ON ,Cursor blinking OFF
80	Force cursor to the beginning of 1st line
C0	Force cursor to the beginning of 2nd line
38	Use 2 lines and 5×7 matrix
83	Cursor line 1 position 3
3C	Activate second line
08	Display OFF, Cursor OFF
C1	Jump to second line, position1
OC	Display ON, Cursor OFF
C1	Jump to second line, position1

C2	Jump to second line, position2

LCD initialization: The steps that have to be done for initializing the LCD display is given below and these steps are common for almost all applications.

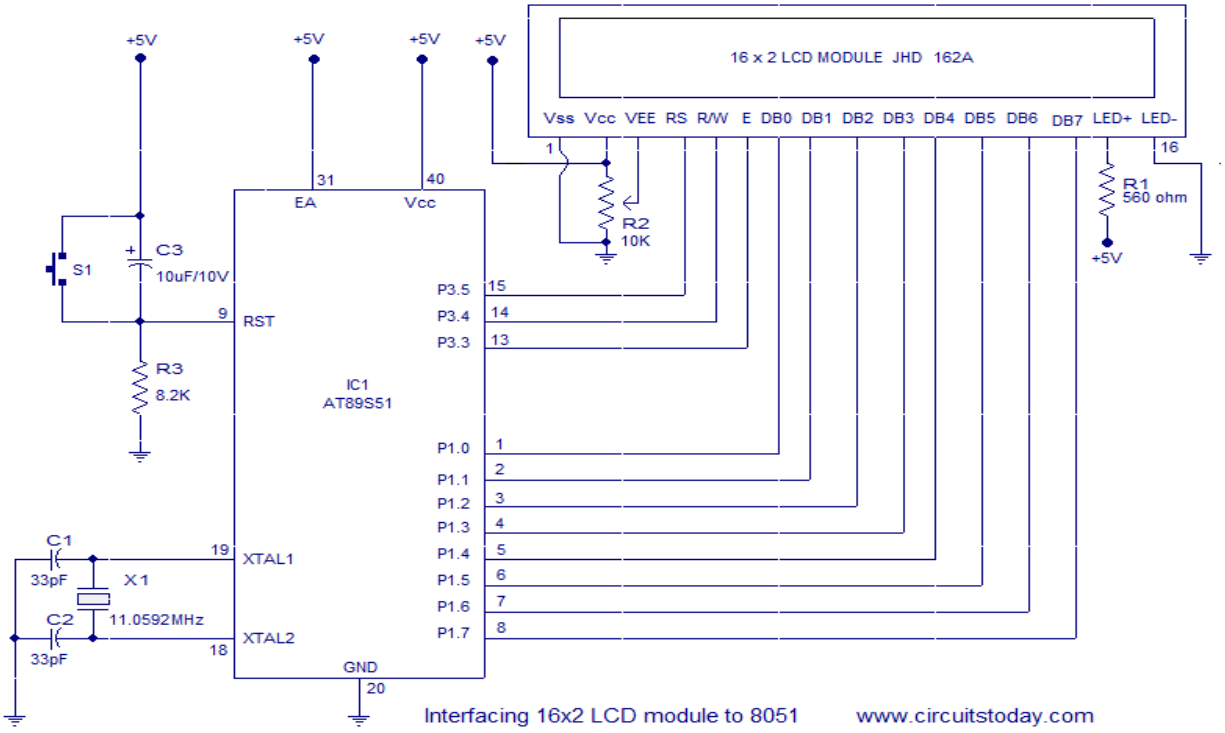
- Send 38H to the 8 bit data line for initialization
- Send 0FH for making LCD ON, cursor ON and cursor blinking ON.
- Send 06H for incrementing cursor position.
- Send 01H for clearing the display and return the cursor.

Sending data to the LCD:

The steps for sending data to the LCD module are given below. I have already said that the LCD module has pins namely RS, R/W and E. It is the logic state of these pins that make the module to determine whether a given data input is a command or data to be displayed.

- Make R/W low.
- Make RS=0 if data byte is a command and make RS=1 if the data byte is a data to be displayed.
- Place data byte on the data register.
- Pulse E from high to low.
- Repeat above steps for sending another data.

Circuit diagram.



Interfacing 16×2 LCD module to 8051

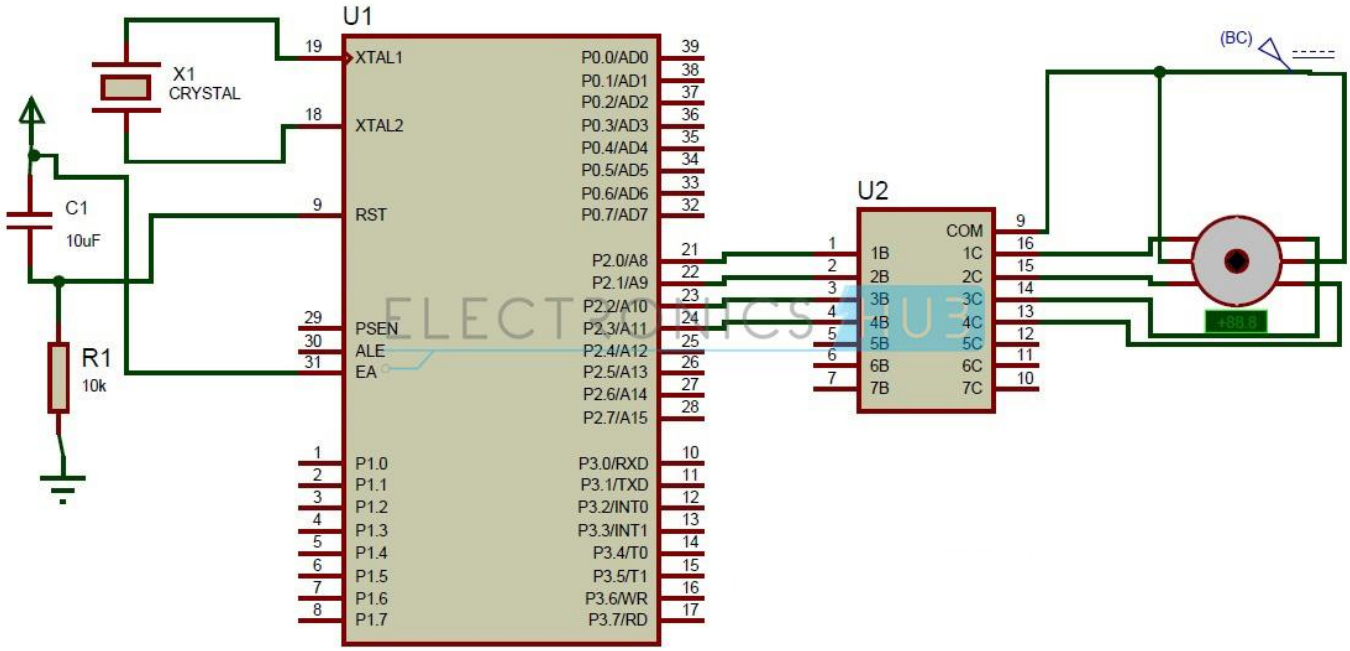
The circuit diagram given above shows how to interface a 16×2 LCD module with AT89S1 microcontroller. Capacitor C3, resistor R3 and push button switch S1 forms the reset circuitry. Ceramic capacitors C1,C2 and crystal X1 is related to the clock circuitry which produces the system clock frequency. P1.0 to P1.7 pins of the microcontroller is connected to the DB0 to DB7 pins of the module respectively and through this route the data goes to the LCD module. P3.3, P3.4 and P3.5 are connected to the E, R/W, RS pins of the microcontroller and through this route the control signals are transferred to the LCD module. Resistor R1 limits the current through the back light LED and so do the back light intensity. POT R2 is used for adjusting the contrast of the display. Program for interfacing LCD to 8051 microcontroller is shown below.

Q5:Stepper motor:

A stepper motor is a brushless and synchronous motor which divides the complete rotation into number of steps. Each stepper motor will have some fixed step angle and motor rotates at this angle. Here in this article, interfacing of stepper to 8051 and ULN 2003 is explained

PRINCIPLE: The main principle of this circuit is to rotate the stepper motor step wise at a particular step angle. The ULN2003 IC is used to drive the stepper motor as the controller cannot provide current required by the motor.

Stepper Motor Control using 8051 Microcontroller Circuit Diagram:



Circuit Diagram of Stepper Motor Control using AT89C51 Microcontroller

Stepper Motor Control using 8051 Microcontroller Circuit Design: The circuit consists of AT89C51 microcontroller, ULN2003A, Motor. AT89c51 is low power, high-performance, CMOS 8bit, 8051 family microcontroller. It has 32 programmable I/O lines. It has 4K bytes of Flash programmable and erasable memory. An external crystal oscillator is connected at the 18 and 19 pins of the microcontroller. Motor is

connected to the port2 of the microcontroller through a driver IC.The ULN2003A is a current driver IC. It is used to drive the current of the stepper motor as it requires more than 60mA of current. It is an array of Darlington pairs. It consists of seven pairs of Darlington arrays with common emitter. The IC consists of 16 pins in which 7 are input pins, 7 are output pins and remaining are VCC and Ground. The first four input pins are connected to the microcontroller. In the same way, four output pins are connected to the stepper motor.

Stepper motor:- Stepper motor has 6 pins. In these six pins, 2 pins are connected to the supply of 12V and the remaining is connected to the output of the stepper motor. Stepper rotates at a given step angle. Each step in rotation is a fraction of full cycle. This depends on the mechanical parts and the driving method. Similar to all the motors, stepper motors will have stator and rotor. Rotor has permanent magnet and stator has coil. The basic stepper motor has 4 coils with 90 degrees rotation step. These four coils are activated in the cyclic order. The below figure shows you the direction of rotation of the shaft. There are different methods to drive a stepper motor. Some of these are explained below.

Full Step Drive: In this method two coils are energized at a time. Thus, here two opposite coils are excited at a time.

Half Step Drive: In this method coils are energized alternatively. Thus it rotates with half step angle. In this method, two coils can be energized at a time or single coil can be energized. Thus it increases the number of rotations per cycle. It is shown in the below figure.

How to Operate this Stepper Motor Driver Circuit?

- Initially, switch on the circuit.
- Microcontroller starts driving the stepper motor.
- One can observe the rotation of the stepper motor
- The stepper motor has four wires. They are yellow, blue, red and white. These are energized alternatively as given below.
- In full step driving, use the following sequence

Yellow	Blue	Red	White
1	0	1	0
0	1	1	0
0	1	0	1
1	0	0	1

To drive the motor in half step angle, use the following sequence

Yellow	Blue	Red	White
1	0	0	0
1	0	1	0
0	0	1	0
0	1	1	0
0	1	0	0
0	1	0	1
0	0	0	1
1	0	0	1

Stepper Motor Controller Circuit Advantages:

- It consumes less power.
- It requires low operating voltage

Stepper Motor Control Applications:

- This circuit can be used in the robotic applications.
- This can also be used in mechantronics applications.
- The stepper motors can be used in disk drives, matrix printers, etc.

Q6:Seven segment display:

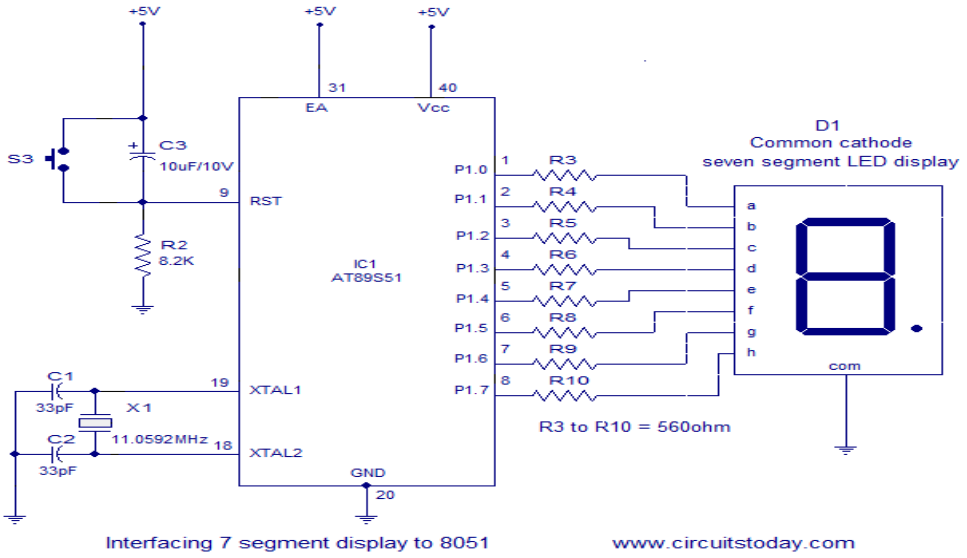
This article is about how to interface a seven segment LED display to an 8051 microcontroller. 7 segment LED display is very popular and it can display digits from 0 to 9 and quite a few characters like A, b, C, ., H, E, e, F, n, o,t,u,y, etc. Knowledge about how to interface a seven segment display to a micro controller is very essential in designing embedded systems. A seven segment display consists of seven LEDs arranged in the form of a squarish **'8'** slightly inclined to the right and a single LED as the dot character. Different characters can be displayed by selectively glowing the required LED segments. Seven segment displays are of two types, **common cathode and common anode.** In common cathode type , the cathode of all LEDs are tied together to a single terminal which is usually labelled as '**com**' and the anode of all LEDs are left alone as individual pins labelled as a, b, c, d, e, f, g & h (or dot) . In common anode type, the anode of all LEDs is tied together as a single terminal and cathodes are left alone as individual pins. The pin out scheme and picture of a typical 7 segment LED display is shown in the image below.

Digit drive pattern: Digit drive pattern of a seven segment LED display is simply the different logic combinations of its terminals **'a' to 'h'** in order to display different digits and characters. The common digit drive patterns (0 to 9) of a seven segment display are shown in the table below.

Digit	A	b	c	d	E	f	g
0	1	1	1	1	1	1	0
1	0	1	1	0	0	0	0
2	1	1	0	1	1	0	1

3	1	1	1	1	0	0	1
4	0	1	1	0	0	1	1
5	1	0	1	1	0	1	1
6	1	0	1	1	1	1	1
7	1	1	1	0	0	0	0
8	1	1	1	1	1	1	1
9	1	1	1	1	0	1	1

Interfacing seven segment displays to 8051.



<u>Interfacing 7 segment display to 8051:</u>

The circuit diagram shown above is of an AT89S51 microcontroller based 0 to 9 counter which has a 7 segment LED display interfaced to it in order to display the count. This simple circuit illustrates two things. How to setup simple 0 to 9 up counter using 8051 and more importantly how to interface a seven segment LED display to 8051 in order to display a particular result. The common cathode seven segment display D1 is connected to the Port 1 of the microcontroller (AT89S51) as shown in the circuit diagram. R3 to R10 are current limiting resistors. S3 is the reset switch and R2,C3

forms a debouncing circuitry. C1, C2 and X1 are related to the clock circuit. The software part of the project has to do the following tasks.

- Form a 0 to 9 counter with a predetermined delay (around 1/2 second here).
- Convert the current count into digit drive pattern.
- Put the current digit drive pattern into a port for displaying.

All the above said tasks are accomplished by the program given below.

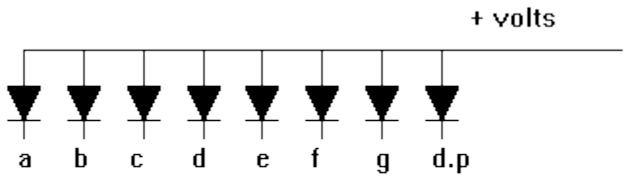
Program:

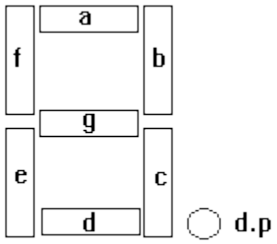
```
ORG 000H //initial starting address
START: MOV A,#00001001B // initial value of accumulator
MOV B,A
MOV R0,#0AH //Register R0 initialized as counter which counts from 10 to 0
LABEL: MOV A,B
INC A
MOV B,A
MOVC A,@A+PC // adds the byte in A to the program counters address
MOV P1,A
ACALL DELAY // calls the delay of the timer
DEC R0//Counter R0 decremented by 1
MOV A,R0 // R0 moved to accumulator to check if it is zero in next instruction.
JZ START // Checks accumulator for zero and jumps to START.
Done to check if counting has been finished.
SJMP LABEL
DB 3FH // digit drive pattern for 0
DB 06H // digit drive pattern for 1
DB 5BH // digit drive pattern for 2
DB 4FH // digit drive pattern for 3
DB 66H // digit drive pattern for 4
DB 6DH // digit drive pattern for 5
DB 7DH // digit drive pattern for 6
DB 07H // digit drive pattern for 7
DB 7FH // digit drive pattern for 8
DB 6FH // digit drive pattern for 9
DELAY: MOV R4,#05H // subroutine for delay
WAIT1: MOV R3,#00H
WAIT2: MOV R2,#00H
WAIT3: DJNZ R2,WAIT3
DJNZ R3,WAIT2
DJNZ R4,WAIT1
```

RET
END

7 SEGMENT DISPLAY:

The 7 segment display is used as a numerical indicator on many types of test equipment. It is an assembly of light emitting diodes which can be powered individually. They most commonly emit red light. They are arranged and labelled as shown in the below diagram.





Powering all the segments will display the number 8. Powering a,b,c d and g will display the number 3. Numbers 0 to 9 can be displayed. The d.p represents a decimal point.

The one shown is a common anode display since all anodes are joined together and go to the positive supply. The cathodes are connected individually to zero volts. Resistors must be placed in series with each diode to limit the current through each diode to a safe value.

Early wrist watches used this type of display but they used so much current that the display was normally switched off. To see the time you had to push a button.

Common cathode displays where all the cathodes are joined are also available.

Liquid crystal displays do a similar job and consume much less power.

Alphanumeric displays are available which can show letters as well as numbers.

KEYBOARD:

Basically, the 4x3 keypad contains push buttons that are arranged in four rows and three columns produce twelve characters as shown in the figure1.Sometimes this called as "4x3 switch matrix" due to the arrangement of switches in a matrix form. The internal construction of these keypads includes metal dome contacts and conductive rubber. Ok! Construction of keypad is out of scope the tutorial. In this tutorial I am only concentrating on interfacing of keypad with 8051 microcontroller.

1. CONNECTIONS:

The three column lines of the keypad as shown in the figure 1 are connected to the PORT 1 upper pins (P1.0 – COL1, P1.1 – COL2, P1.2 – COL3) and the four row lines are connected to PORT1 lower pins (P1.4 – ROW1, P1.5 – ROW2, P1.6 – ROW3, P1.7 – ROW4). Three resistors of 10k are connected between the column lines and power supply, to make the column lines are always high. Here one thing should be clear, that the column lines connected to the microcontroller should act as input lines and the row lines acts as output lines.

2. WORKING:

2.1 INITIALIZATION:

Remember that, the column lines of keypad are connected to port1 pins and these pins should be configured as input by placing logic high ('LOGIC 1') during port initialization. Similarly, the row lines of keypad are connected to port1 pins and configured them as output by placing logic zero ('LOGIC 0') during port initialization.

```
ROW1 = 0; //MAKE ALL ROW LINES OF KEY TO ZERO
ROW2 = 0; // TO MAKE THEM AS OUTPUT LINES
ROW3 = 0;
ROW4 = 0;
COL1 = 1; // MAKE ALL COL'S AS HIGH
COL2 = 1; // TO MAKE THEM AS INPUT LINES
COL3 = 1;
```

2.2 SCANNING MECHANISM:

The scanning mechanism starts by making the first row (ROW1) of keypad to LOW (LOGIC '0') and all column lines should be high (LOGIC '1'). Now check any one of the column lines goes low. If any column line goes low means that particular button is pressed, otherwise nothing is pressed.
Now the question in our mind is "which button is pressed". This can be identified by checking which column is goes low.

If ROW1 = 0, COL1 = 0, remaining all lines are high means BUTTON1 is pressed.
If ROW1 = 0, COL2 = 0, remaining all lines are high means BUTTON2 is pressed.
If ROW1 = 0, COL3 = 0, remaining all lines are high means BUTTON3 is pressed.
If ROW2 = 0, COL1 = 0, remaining all lines are high means BUTTON4 is pressed.

Likewise we can identify the remaining buttons also and the logic is shown in the figure1. This can be done by repeatedly, that why use a infinite loop in the routine "**key()**" – check out the c program.

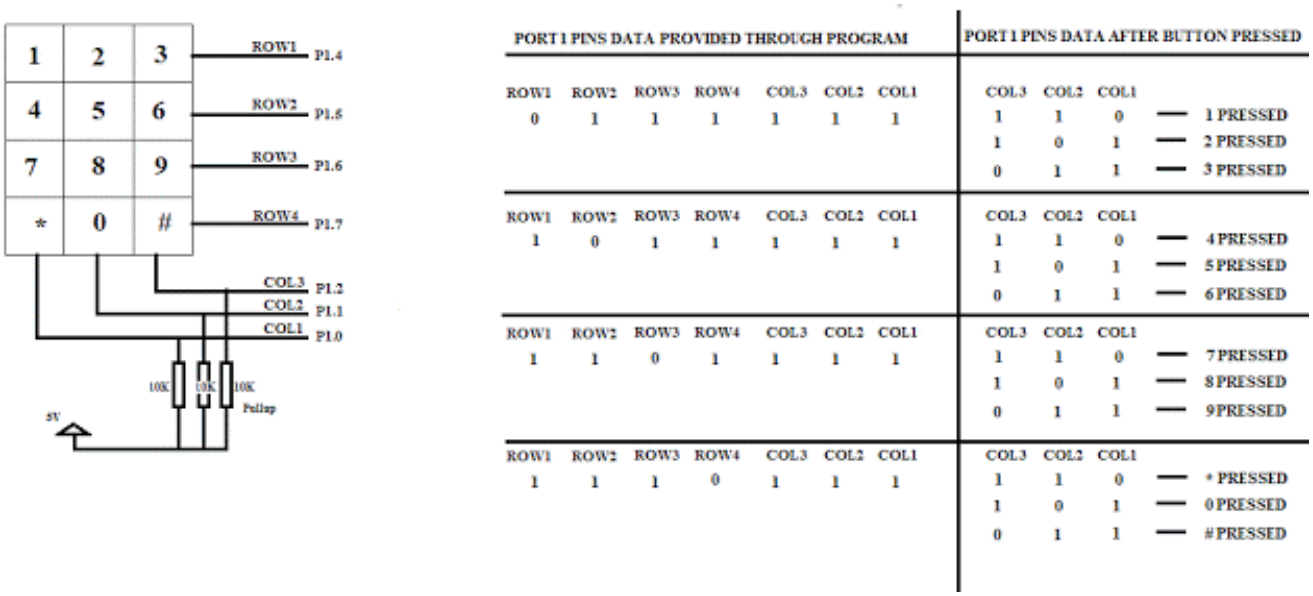


Figure1 shows the keypad connection and its logic to interface with MCU.

3. PROGRAM DESCRIPTION:

Firstly create a 2-D array (look up table) of numbers like below. This array holds the numbers represented by the keypad. Off course you can change the data as you need, means you define characters, special symbols, numbers, etc..,

```
char keypad[4][3]=    {
        '1','2','3',
                                '4','5','6',
                                '7','8','9',
                                '*','0','#'
        };        //look up table
```

Now in the subroutine "**key()**", first make first row lines to zero one and scan for the column status. And next make the next row and scan for the column and so on.

```
ROW1 = 0; //MAKE FIRST ROW IN KEYPAD TO ZERO
ROW2 = 1;
ROW3 = 1;
ROW4 = 1;
COL1 = 1; // MAKE ALL COL'S AS HIGH
COL2 = 1;
COL3 = 1;
```

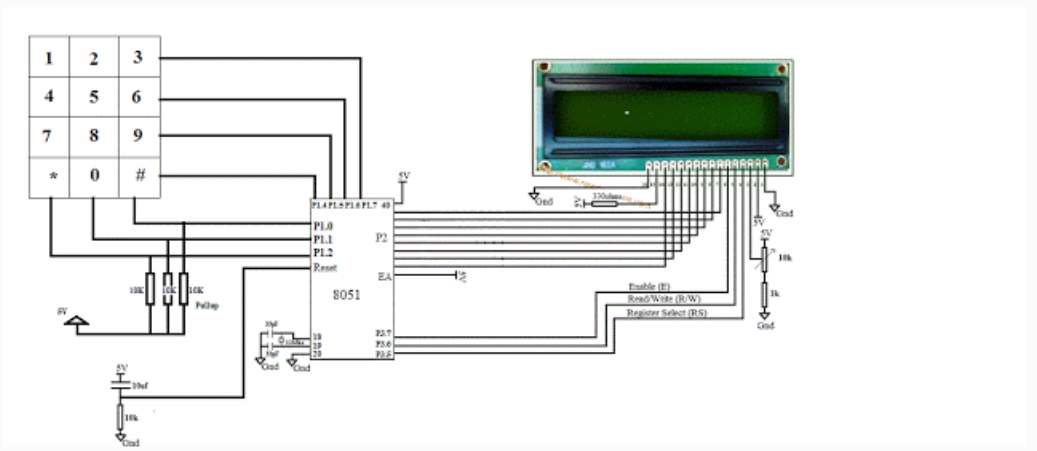
If any one of the column line is low then wait until it back to logic high by continuously checking the col line (while(COL1 == 0)). If col line goes high then place column and row values to get the correct

value from the keypad array. Finally place a break statement to get out of the loop, otherwise it stay in infinite loop.

if (COL1 == 0){while(COL1 == 0);row=1;col=1;break;}.

For LCD interface I encourage you to read the tutorial to interface 16x2 LCD with 8051.

Circuit Diagram of 4x3 keypad interface with 8051 along with 16x2 LCD:



circuit diagram for 4x3 keypad interface with 8051 along with LCD

DAC (DIGITAL TO ANALOG CONVERTER) :----

In digital the signals either of two levels representing the binary value 1(or) Zero. In ADC obtains a digital value representing an analog input voltage, while DAC change a digital value back to analog.

Digital-to-analog (DAC) converter

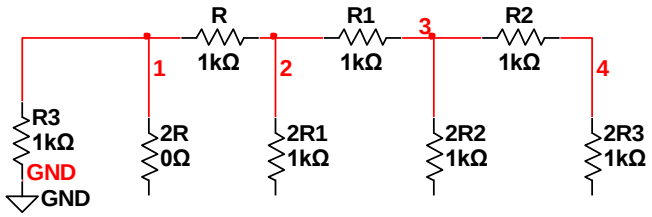
The digital-to-analog converter (DAC) is a device widely used to convert digital pulses to analog signals. In this section we discuss the basics of interfacing a DAC to the 8051.

A binary weighted and R/2R ladder. The vast majority of integrated circuit DACs, including the MC1408 (DAC0808) used in this section use the R/2R method since it can achieve a much higher degree of precision. It accepts inputs of binary values typically 0v (or) Vref and provides an output voltage proportional to binary input value .The first criterion for judging a DAC is its resolution, which is a function of the number of binary inputs. In four input values representing 4-bits of digital data and DC voltage output .The output voltage is proportional to digital input value.

$$V = \frac{Do\ x\ 2 + D1\ x\ 2 + D2\ x\ 2 + D3\ x\ 2 + D4\ x\ 2}{2}$$

$$v = \frac{0\ x\ 1 + 1\ x\ 2 + 1\ x\ 4 + 0\ x\ 8}{2} X(16v)$$

$$= 6V$$



(a) 4-stage ladder network

In above example the output voltage resulting a binary value $(0110)_2$ digital converts in to 6v analog. The function of ladder network is to convert 16 possible values (0000-1111) into 1 of 16 voltage level in steps Vref/16 .

DAC INTERFACING:

This section will show how to interface a DAC (digital-to-analog converter) to the 8051. Then we demonstrate how to generate a sine wave on the scope using the DAC.

The common ones are 8, 10, and 12 bits. The number of data bit inputs decides the resolution of the DAC since the number of analog output levels is equal to 2″, 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.
Similarly, the 12-bit DAC provides 4096 discrete voltage levels. There are also
16-bit DACs, but they are more expensive.

MC1408 DAC (or DAC0808)

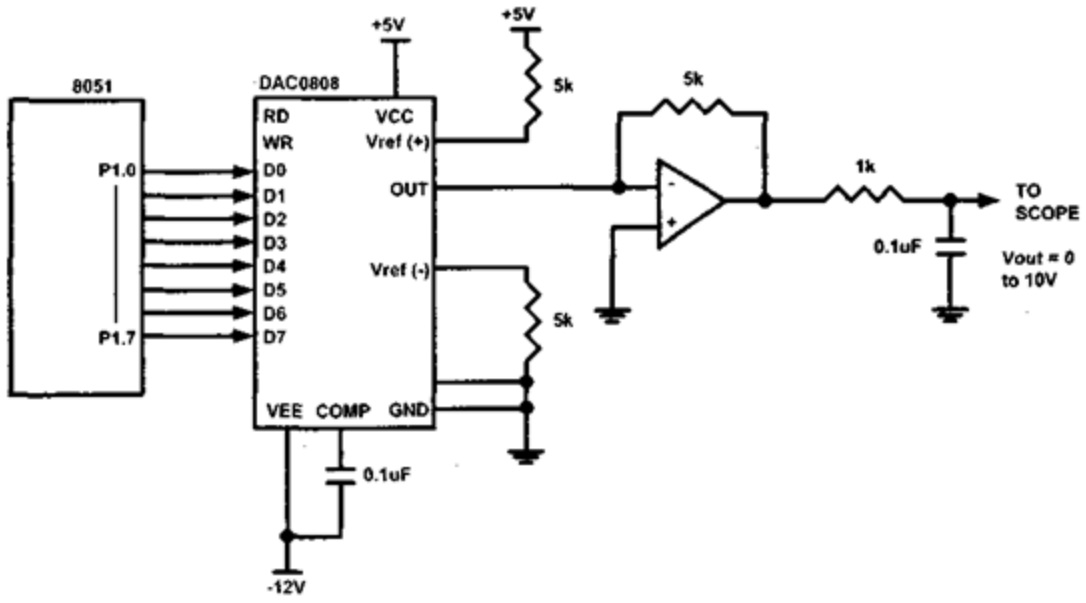
In the MC1408 (DAC0808), the digital inputs are converted to current (I_{out}), and by connecting a resistor to the I_{out} pin, we convert the result to voltage.
The total current provided by the I_{out} pin is a function of the binary numbers at the DO – D7 inputs of the DAC0808 and the reference current (I_{ref}), and is as follows:

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

where DO is the LSB, D7 is the MSB for the inputs, and I_{ref} is the input current that must be applied to pin 14. The I_{ref} current is generally set to 2.0 mA. Figure 13-18 shows the generation of current reference (setting I_{ref} = 2 mA) by using the standard 5-V power supply and IK and 1.5K-ohm standard resistors. Some DACs also use the zener diode (LM336), which overcomes any fluctuation associated

DAC INTERFACING:

This section will show how to interface a DAC (digital-to-analog converter) to the 8051. Then we demonstrate how to generate a sine wave on the scope using the DAC.



8051 Connection to DAC808

with the power supply voltage. Now assuming that I_{ref} = 2 mA, if all the inputs to the DAC are high, the maximum output current is 1.99 mA (verify this for yourself).

Converting I_{out} to voltage in DAC0808

Ideally we connect the output pin I_{out} to a resistor, convert this current to voltage, and monitor the output on the scope. In real life, however, this can cause inaccuracy since the input resistance of the load where it is connected will also affect the output voltage. For this reason, the I_{ref} current output is isolated by connecting it to an op-amp such as the 741 with R_f = 5K ohms for the feedback resistor. Assuming that R = 5K ohms, by changing the binary input, the output voltage changes as shown in Example 13-4.

Example

In order to generate a stair-step ramp, set up the circuit in Figure 13-18 and connect the output to an oscilloscope. Then write a program to send data to the DAC to generate a stair-step ramp.

Solution:

```
        CLR   A
AGAIN:  MOV   P1,A          ;send data to DAC
        INC   A             ;count from 0 to FFH
        ACALL DELAY         ;let DAC recover
        SJMP  AGAIN
```

Generating a sine wave

To generate a sine wave, we first need a table whose values represent the magnitude of the sine of angles between 0 and 360 degrees. The values for the sine function vary from -1.0 to +1.0 for 0- to 360-degree angles. Therefore, the table values are integer numbers representing the voltage magnitude for the sine of theta. This method ensures that only integer numbers are output to the DAC by the 8051 microcontroller. Table 13-7 shows the angles, the sine values, the voltage magnitudes, and the integer values representing the voltage magnitude for each angle (with 30-degree increments). To generate Table 13-7, we assumed the full-scale voltage of 10 V for DAC output (as designed in Figure 13-18). Full-scale output of the DAC is achieved when all the data inputs of the DAC are high. Therefore, to achieve the full-scale 10 V output, we use the following equation.

$$V_{out} = 5\ \text{V} + (5 \times \sin\theta)$$

V_{out} of DAC for various angles is calculated and shown in Table 13-7. See Example 13-5 for verification of the calculations.

Program:

```
ORG 0000h
mov P2,#00H
repeat:          call squarwave      ; generate square wave
calltriwave            ; generate triangular wave
callstairwave       ; generate staircase wave
jmp repeat


squarwave:  mov P2,#FFH
call delay2sec
mov P2,#00H
call delay2sec
ret


triwave:        mov R7,#00H
triwave1:      mov P2,R7
inc R7
cjne R7,#FFH,triwave1


mov R7,#FFH
triwave2:      mov P2,R7
djnz R7,triwave2
ret


stairwave:     mov P2,#00H
```

```
call delay2sec
mov P2,#20H
call delay2sec
mov P2,#40H
call delay2sec
mov P2,#80H
call delay2sec
ret

delay1sec:  mov r0,#10
del2:           mov r1,#250
del1:            mov r2,#250
djnz r2,$
djnz r1,del1
djnz r0,del2
ret

delay2sec:   mov r0,#20
del22:          mov r1,#250
del21:          mov r2,#250
djnz r2,$
djnz r1,del21
djnz r0,del22
ret
    END
```

```
        org 0000H
        call delay2sec
        inc r0
        mov P1,#00H
        jb P0.7,back
        clr P2.0
        jmp back
```

8051 data type and directives

In 8051 microcontroller has only one data type. It is 8 bits, and the size of each register is also 8 bits. It is the job of the programmer to break down data larger than 8 bits (00 to FFH, or 0 to 255 in decimal) to be processed by the CPU. The data types used by the 8051 can be positive or negative.

DB (define byte)

The DB directive is the most widely used data directive in the assembler. It is used to define the 8-bit data. When DB is used to define data, the numbers can be in decimal, binary, hex, or ASCII formats. For decimal, the "D" after the decimal number is optional, but using "B" (binary) and "H" (hexadecimal) for the others is required. Regardless of which is used, the assembler will convert the numbers into hex. To indicate ASCII, simply place the characters in quotation marks ('like this'). The assembler will assign the ASCII code for the numbers or characters automatically. The DB directive is the only directive that can be used to define ASCII strings larger than two characters; therefore, it should be used for all ASCII data definitions. Following are some DB examples:

```
        ORG  500H
DATA1:  DB   28                ;DECIMAL(1C in hex)
DATA2:  DB   00110101B         ;BINARY (35 in hex)
DATA3:  DB   39H               ;HEX
        ORG  510H
DATA4:  DB   "2591"            ;ASCII NUMBERS
        ORG  518H
DATA6:  DB   "My name is Joe"  ;ASCII CHARACTERS
```

Either single or double quotes can be used around ASCII strings. This can be useful for strings, which contain a single quote such as "O'Leary". DB is also used to allocate memory in byte-sized chunks.

Assembler directives

The following are some more widely used directives of the 8051.

ORG (origin)

The ORG directive is used to indicate the beginning of the address. The number that comes after ORG can be either in hex or in decimal. If the number is not followed by H, it is decimal and the assembler will convert it to hex. Some assemblers use ". ORG" (notice the dot) instead of "ORG" for the origin directive. Check your assembler.

EQU (equate)

This is used to define a constant without occupying a memory location. The EQU directive does not set aside storage for a data item but associates a constant value with a data label so that when the label appears in the program, it constant value will be substituted for the label. The following uses EQU for the counter constant and then the constant is used to load the R3 register.

```
COUNT     EQU  25
...       ....
MOV       R3,#COUNT
```

When executing the instruction “MOV R3, #COUNT”, the register R3 will be loaded with the value 25 (notice the # sign). What is the advantage of using EQU? Assume that there is a constant (a fixed value) used in many different places in the program, and the programmer wants to change its value throughout. By the use of EQU, the programmer can change it once and the assembler will change* all of its occurrences, rather than search the entire program trying to find every occurrence.

END directive

Another important pseudocode is the END directive. This indicates to the assembler the end of the source (asm) file. The END directive is the last line of an 8051 program, meaning that in the source code anything after the END directive is ignored by the assembler. Some assemblers use “. END” (notice the dot) instead
of “END”.

Rules for labels in Assembly language

By choosing label names that are meaningful, a programmer can make a program much easier to read and maintain. There are several rules that names must follow. First, each label name must be unique. The names used for labels in Assembly language programming consist of alphabetic letters in both uppercase and lowercase, the digits 0 through 9, and the special characters question mark (?), period (.), at (@), underline (_), and dollar sign ($). The first character of the label must be an alphabetic character. In other words it cannot be a number. Every assembler has some reserved words that must not be used as labels in the program. Foremost among the reserved words are the mnemonics for the instructions. For example, “MOV” and “ADD” are reserved since they are instruction mnemonics. In addition to the mnemonics there are some other reserved words. Check your assembler for the list of reserved words.