# INTEL 8051 MICRCONTROLLER

**Introduction :**

A decade back the process and control operations were totally implemented by the Microprocessors only. But now a days the situation is totally changed and it is occupied by the new devices called Microcontroller. The development is so drastic that we can't find any electronic gadget without the use of a microcontroller. This microcontroller changed the embedded system design so simple and advanced that the embedded market has become one of the most sought after for not only entrepreneurs but for design engineers also.

**What is a Microcontroller?**

A single chip computer or A CPU with all the peripherals like RAM, ROM, I/O Ports, Timers , ADCs etc... on the same chip. For ex: Motorola's 6811, Intel's 8051, Zilog's Z8 and PIC 16X etc…

**MICROPROCESSORS & MICROCONTROLLERS:**

**Microprocessor:**

A CPU built into a single VLSI chip is called a microprocessor. It is a general-purpose device and additional external circuitry are added to make it a microcomputer. The microprocessor contains arithmetic and logic unit (ALU), Instruction decoder and control unit, Instruction register, Program counter (PC), clock circuit (internal or external), reset circuit (internal or external) and registers. But the microprocessor has no on chip I/O Ports, Timers , Memory etc.

For example, Intel 8085 is an 8-bit microprocessor and Intel 8086/8088 a 16-bit microprocessor. The block diagram of the Microprocessor is shown in Fig.1
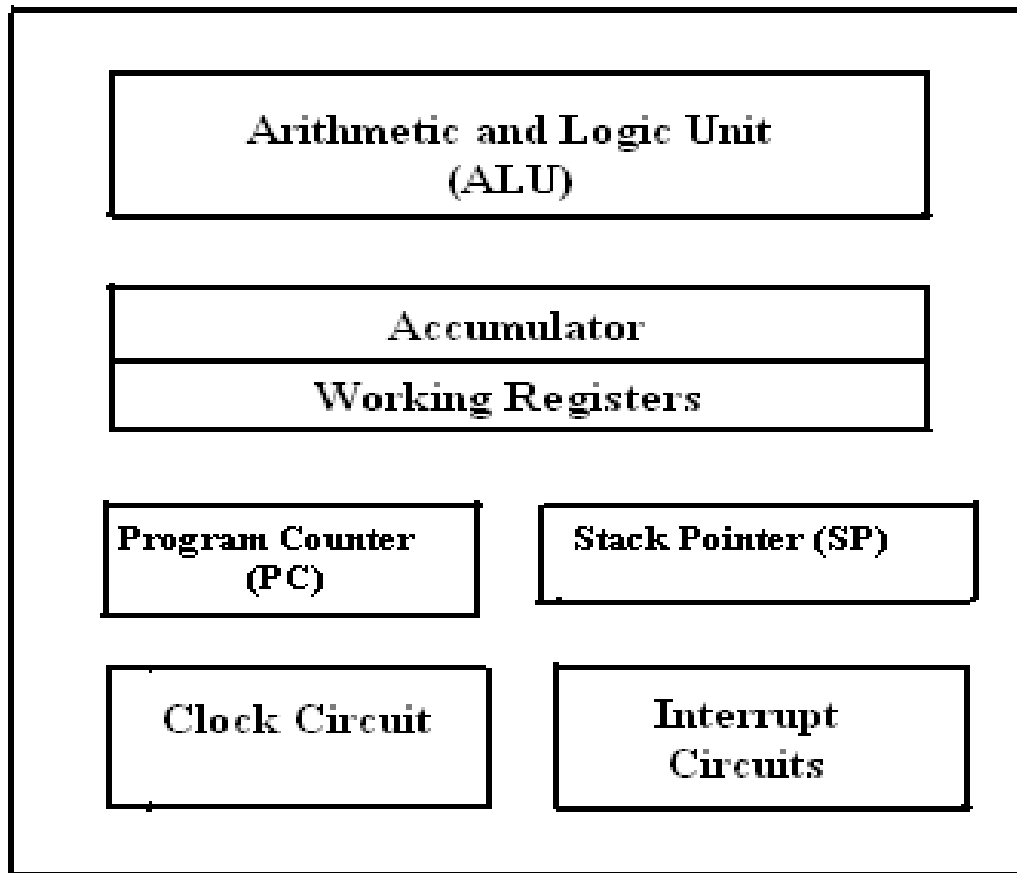
**Fig.1 Block diagram of a Microprocessor.**

**MICROCONTROLLER :**

A microcontroller is   a highly integrated single chip, which consists of  on chip  CPU (Central Processing Unit), RAM (Random Access Memory), EPROM/PROM/ROM (Erasable Programmable Read Only Memory), I/O (input/output) – serial and parallel, timers, interrupt controller.   For   example,   Intel   8051   is   8-bit   microcontroller   and   Intel   8096   is   16-bit microcontroller. The block diagram of Microcontroller is shown in Fig.2.
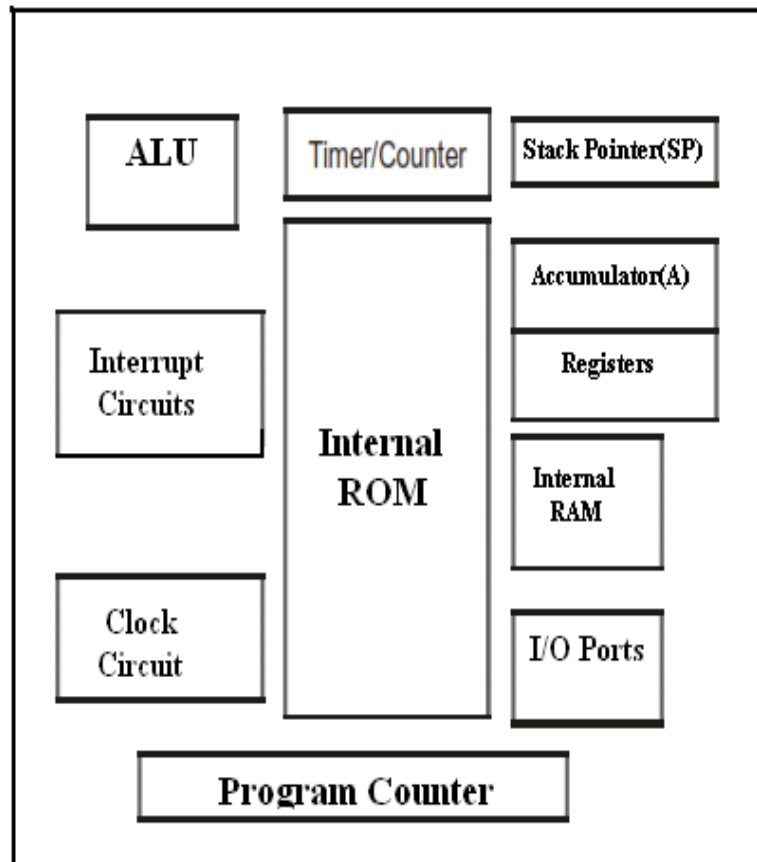
**Fig.2.Block Diagram of a Microcontroller**

**Distinguish between Microprocessor and Microcontroller**

| S.No | Microprocessor | Microcontroller |
|---|---|---|
| 1 | A microprocessor is a general purpose device which is called a CPU | A microcontroller is a dedicated chip which is also called single chip computer. |
| 2 | A microprocessor do not contain onchip I/OPorts, Timers, Memories etc.. | A microcontroller includes RAM, ROM, serial and parallel interface, timers, interrupt circuitry (in addition to CPU) in a single chip. |
| 3 | Microprocessors are most commonly used as the CPU in microcomputer systems | Microcontrollers are used in small, minimum component designs performing control-oriented applications. |
| 4 | Microprocessor instructions are mainly nibble or byte addressable | Microcontroller instructions are both bit addressable as well as byte addressable. |
| 5 | Microprocessor instruction sets are mainly intended for catering to large volumes of data. | Microcontrollers have instruction sets catering to the control of inputs and outputs. |
| 6 | Microprocessor based system design is complex and expensive | Microcontroller based system design is rather simple and cost effective |
| 7 | The Instruction set of microprocessor is complex with large number of instructions. | The instruction set of a Microcontroller is very simple with less number of instructions. For, ex: PIC microcontrollers have only 35 instructions. |
| 8 | A microprocessor has zero status flag | A microcontroller has no zero flag. |

**EVOLUTION OF MICROCONTROLLERS :**

The first microcontroller TMS1000 was introduced by Texas Instruments in the year 1974. In the year 1976, Motorola designed a Microprocessor chip called 6801 which replaced its earlier chip 6800 with certain add-on chips to make a computer. This paved the way for the new revolution in the history of chip design and gave birth to a new entity called "**Microcontroller".** Later **t**he Intel company produced its first Microcontroller 8048 with a CPU and 1K bytes of EPROM, 64 Bytes of RAM an 8-Bit Timer and 27 I/O pins in 1976. Then followed the most popular controller 8051 in the year 1980 with 4K bytes of ROM,128 Bytes of RAM , a serial port, two 16-bit Timers , and 32 I/O pins. The 8051 family has many additions and improvements over the years and remains a most acclaimed tool for today's circuit designers. INTEL introduced a 16 bit microcontroller 8096 in the year 1982 . Later INTEL introduced 80c196 series of 16-bit Microcontrollers for mainly industrial applications. Microchip, another company has introduced an 8-bit Microcontroller PIC 16C64 in the year 1985.The 32-bit microcontrollers have been developed by IBM and Motorola. MPC 505 is a 32-bit RISC controller of Motorola. The 403 GA is a 32 -bit RISC embedded controller of IBM.

In recent times ARM company (Advanced RISC machines) has developed and introduced 32 bit controllers for high-end application devices like mobiles , Ipods etc...

**TYPES OF MICROCONTROLLERS :**

Microcontrollers can be classified on the basis of internal bus width, architecture, memory and instruction set as 4-bit,8-bit,16-bit and 32-bit micrcontrollers.

**4-bit Microcontrollers**: These 4-bit microcontrollers are small size, minimum pin count and low cost controllers which are widely used for low end applications like LED & LCD display drivers ,portable battery chargers etc.. Their power consumption is also low. The popular 4-bit controllers are Renasa M34501 which is a 20 pin DIP chip with 4kB of ROM,256 Bytes of RAM,2-Counters and 14 I/O Pins. Similarly ATAM862 series from ATMEL.

**8-bit Microcontrollers :** These are the most popular and widely used microcontrollers .About 55% of all CPUs sold in the world are 8-bit microcontrollers only.The 8-bit microcontroller has 8-bitinternal bus and the ALU performs all the arithmetic and logical operations on a byte instruction. The well known 8-bit microcontroller is 8051 which was designed by Intel in the

year 1980 for the use in embedded systems. Other 8-bit microcontrollers are Intel 8031/8052 and Motorola MC68HC11 and AVR Microcontrollers, Microchip's PIC Microcontrollers 12C5XX ,16C5X and 16C505 etc...

**16-bit Microcontrollers :** When the microcontroller performs 16-bit arithmetic and logical operations at an instruction, the microcontroller is said to be a 16-bit microcontroller. The internal bus width of 16-bit microcontroller is of 16-bit. These microcontrollers are having increased memory size and speed of operation when compared to 8-bit microcontrollers.These are most suitable for programming in Highlevel languages like C or $C^{++}$ .They find applications in disk drivers,modems,printers,scanners and servomotor control. Examples of 16-bit microcontrollers are Intel 8096 family and Motorola MC68HC12 and MC68332 families, The performance and computing capability of 16 bit microcontrollers are enhanced with greater precision as compared to the 8-bit microcontrollers.

**32-Bit Microcontrollers :**These microcontrollers used in highend applications like Automative control, Communication networks,Robotics,Cell phones ,GPRS & PDAs etc..For EX:PIC32,ARM 7,ARM9 ,SHARP LH79520 ,ATMEL 32 (AVR) ,Texas Instrument's –. TMS320F2802x/2803x etc..are some of the popular 32-bit microcontrollers.

**COMMERCIAL MICROCONTROLLERS**

There are various manufacturers who are supplying various types of microcontrollers suitable for different applications depending on the power consumption and the available features..They are given below in tables . First the various members of INTEL 51 family are given in below table.

**INTEL MCS 51 Family**

| Microcontroller | On chip RAM (Bytes) | On chip program memory | Timers/Counters | Interrupts | Serial ports |
|---|---|---|---|---|---|
| 8031 | 128 | None | 2 | 5 | 1 |
| 8032 | 256 | None | 3 | 6 | 1 |

| 8051 | 128 | 4K ROM | 2 | 5 | 1 |
|------|-----|--------|---|---|---|
| 8052 | 256 | 8K ROM | 3 | 6 | 1 |
| 8751 | 128 | 4K EPROM | 2 | 5 | 1 |
| 8752 | 256 | 8K EPROM | 3 | 6 | 1 |

The following table gives the 4-bit microcontrollers from different manufacturers.

| Manufacturer | I/O | Pins | RAM (bytes) | ROM (bytes) | Counters | Features |
|--------------|-----|------|-------------|-------------|----------|----------|
| COP400 Family (National) | 23 | 28 | 64 | 1K | 1 | Serial bit I/O |
| HMCS40 (Hitachi) | 10 | 28 | 32 | 512 | — | 10-bit ROM |
| TMS 1000 (Texas Instruments) | 23 | 28 | 64 | 1K | — | LED display |

**4-Bit Microcontrollers**.

| Manufacturer | I/O | Pins | RAM (bytes) | ROM (bytes) | Counters | Features |
|--------------|-----|------|-------------|-------------|----------|----------|
| 8048 (Intel) | 27 | 40 | 64 | 1K | 1 | 8k External memory |
| 8051 (Intel) | 32 | 40 | 128 | 4K | 2 | 128k External memory, Boolean processing, serial port |
| COP800 Family (National) | 24 | 28 | 64 | 1K | 1 | Serial bit I/O, 8- channel A/D converter |
| 6805 (Motorola) | 20 | 28 | 64 | 1K | 1 | PLL frequency synthe-sizer, |
| 68hc11 (Motorola) | 40 | 52 | 256 | 8K | 2 | A/D, PWM generator, pulse accumulator |
| TMS370 (Texas) | 55 | 68 | 256 | 4K | 2 | watchdog timer, Instru-ments) Serial ports, A/D (8 bit, 8 channel) |
| PIC (Micro Chip) | 12 | 18 | 25 | 1K | 0 | small pin count, very low power consumption |

## 8-Bit Microcontrollers.

| Manufacturer | I/O | Pins | RAM (bytes) | | ROM (bytes) | Counters | Features |
|---|---|---|---|---|---|---|---|
| 80c196(INTEL) | 40 | 68 | 232 | 8K | 2 | —— | PWM generator, watchdog timer |
| HPC Family (National) | 52 | 68 | 512 | 16K | 4 | —— | PWM generator, watchdog timer, 8-channel A/D, serial port |

## 16-Bit Microcontrollers

The following table gives the list of PIC microcontrollers from Micro chip Inc

| Microcontroller | Pins | I/O Lines | On chip ADCs | EPROM X 12 words | On chip RAM (Bytes) |
|---|---|---|---|---|---|
| 16C54 | 18 | 12 | None | 512 | 25 |
| 16C55 | 28 | 20 | None | 512 | 24 |
| 16C56 | 18 | 12 | None | 1k | 25 |
| 16C57 | 28 | 20 | None | 2k | 72 |
| 17C42A | 40 | 33 | None | 2k | 232 |
| 17C43 | 40 | 33 | None | 4k | 454 |
| 17C44 | 40 | 33 | None | 8k | 454 |
| 17C71 | 18 | 13 | 8bit ADCs | 1kx14 | 36 |
| 17C752 | 40 | 33 | 10Bit ADC | 8kx16 | 678 |

**MICROCONTROLLER   DEVELOPMENT TOOLS:**

To develop an assembly language program we need certain  program development tools. An assembly language program consists of Mnemonics  which are nothing but short abbreviated English  instructions  given  to  the  controller.The  various  development  tools  required  for Microcontroller   programming are explained below.

**1. Editor :** An Editor is a program which allows us to create a file containing the assembly language statements for the  program. Examples of some editors are PC write Wordstar. As we type the program the   editor stores the ACSII codes for the letters and numbers in successive RAM locations. If any typing mistake is done editor will alert us   to correct it. If we leave out a program statement an editor will let you move everything down and insert a line. After typing all the 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.

 Ex: Sample. asm

**2.Assembler :**  An Assembler is used to translate the assembly language mnemonics into machine language( i.e binary codes).  When you run the assembler it reads the source file of your program from where you have saved it. The assembler generates a filee with the extension  **.hex**. This file consists of hexadecimal values encoding a sequence of data and their starting offset or absolute address.

**3.Compiler :** A compiler is a program which converts the high level  language program like "C" into binary or machine code. Using   high level languages it is easy to manage complex data structures which are often required for data manipulation. Because of its ease , flexibility and debug options now a days the  compilers have become very popular in the market. Compilers like Keil ,Ride and IAR workbench are very popular.

**3. Debugger/Simulator :**  A debugger is a program which allows   execute  the program, and troubleshoot or debug it. The debugger allows  to look into the contents of registers and memory locations after the program runs. We can also change the contents of registers and memory locations and rerun the program. Some debuggers allows   to stop the program after each

instruction so that you can check or alter memory and register contents. This is called single step debug. A debugger also allows to set a breakpoint at any point in the program. If we insert a break point , the debugger will run the program up to the instruction where the breakpoint is put and then stop the execution.

A simulator is a software program which virtually executes the instructions similar to a microcontroller and shows the results. This will help in evaluating the results without committing any errors. By doing so we can detect the possible logic errors

**INTEL 8051 MICRCONTROLLER :**

The 8051 microcontroller is a very popular 8-bit microcontroller introduced by Intel in the year 1981 and it has become almost the academic standard now a days. The 8051 is based on an 8-bit CISC core with Harvard architecture. Its 8-bit architecture is optimized for control applications with extensive Boolean processing. It is available as a 40-pin DIP chip and works at +5 Volts DC. The salient features of 8051 controller are given below.

**SALIENT FEATURES :** The salient features of 8051 Microcontroller are

i. 4 KB on chip program memory (ROM or EPROM)).

ii. 128 bytes on chip data memory(RAM).

iii. 8-bit data bus

iv. 16-bit address bus

v. 32 general purpose registers each of 8 bits

vi. Two -16 bit timers $T_0$ and $T_1$

vii. Five Interrupts (3 internal and 2 external).

ix. Four Parallel ports each of 8-bits (PORT0, PORT1,PORT2,PORT3) with a total of 32 I/O lines.

x. One 16-bit program counter and One 16-bit DPTR ( data pointer)

xi. One 8-bit stack pointer

xii. One Microsecond instruction cycle with 12 MHz Crystal.

xiii. One full duplex  serial communication port.

## ARCHITECTURE & BLOCK DIAGRAM OF 8051 MICROCONTROLLER:

The architecture of the 8051 microcontroller can be understood from the block diagram. It has Harward architecture with RISC (Reduced Instruction Set Computer)  concept. The block diagram of 8051 microcontroller is shown  in Fig 3. below1.It  consists of an 8-bit ALU, one 8-bit  PSW(Program Status Register), A and B registers , one 16-bit Program counter , one 16-bit Data pointer register(DPTR),128 bytes of RAM and 4kB of ROM and four parallel I/O ports each of 8-bit width.
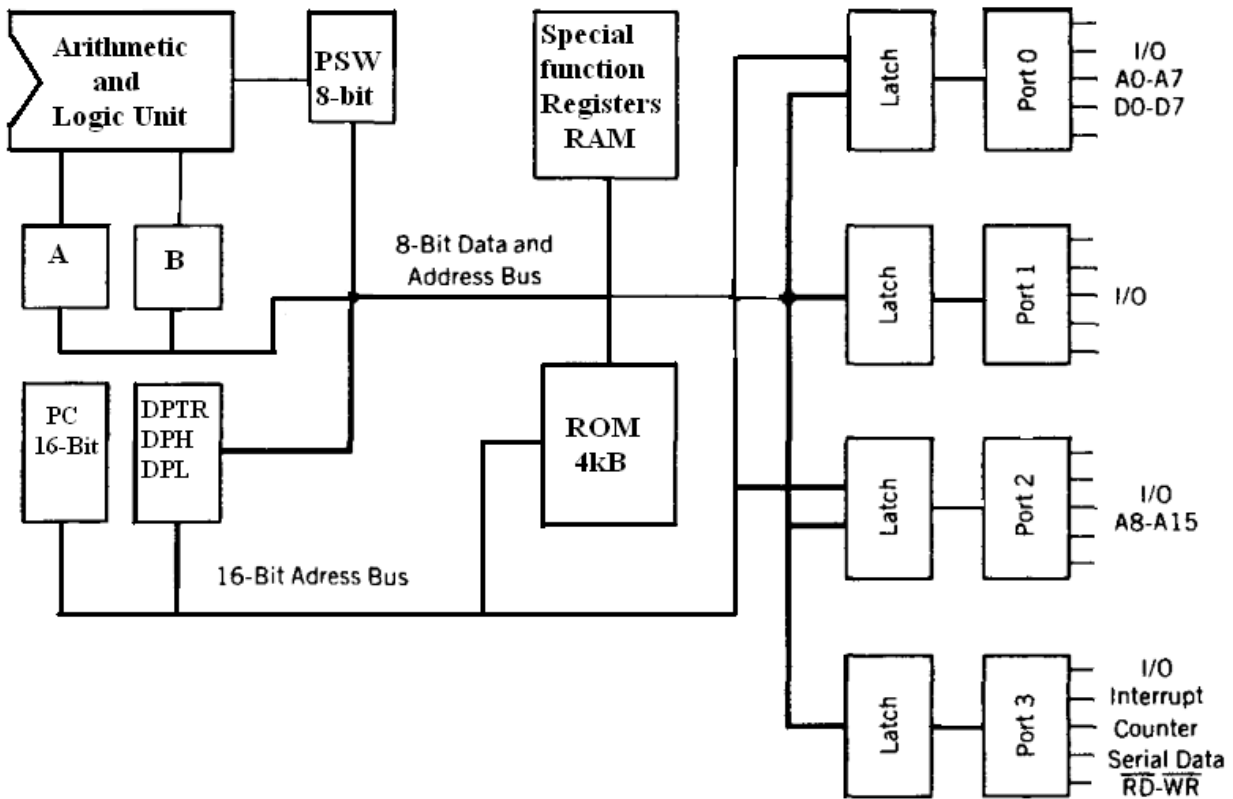


**Fig.3. Block Diagram of 8051 Microcontroller**

8051 has 8-bit ALU which can perform all the 8-bit arithmetic and logical operations in one machine cycle. The ALU is associated with two registers A & B

**A and B Registers** : The A  and B registers are special function registers which hold the results of many arithmetic and logical operations of 8051.The A register is also called the **Accumulator**

11

and    as it's name suggests, is used as a general register to accumulate the results of a large number of instructions. By default it is used for all mathematical operations and also data transfer operations between CPU and any external memory.

The B register is mainly used for multiplication and division operations along with A register.

MUL AB   :            DIV AB.

It has no other function other than as a location where data may be stored.

**The R registers**: The "R" registers are a set of eight registers that are named R0, R1, etc. up to and including R7. These registers are used as auxiliary registers in many operations. The "R" registers are also used to temporarily store values.

**Program Counter(PC) :** 8051 has a 16-bit program counter .The program counter always points to the address of the next instruction to be executed. After execution of one instruction the program counter is incremented to point to the address of the next instruction to be executed. It is the contents of the PC that are placed on the address bus to find and fetch the desired instruction. Since the PC is 16-bit width, 8051 can access program addresses from 0000H to FFFFH, a total of 64kB of code.

**Stack Pointer Register (SP):** It is an 8-bit register which stores the address of the stack top. i.e the Stack Pointer is used to indicate where the next value  to be removed  from the stack should be taken from. When a value is pushed onto the stack, the 8051 first increments the value of SP and then stores the value at the resulting memory location. Similarly when a value is popped off the stack, the 8051 returns the value from the memory location indicated by SP, and then decrements the value of SP. Since the SP is only 8-bit wide it is incremented or decremented by two. SP is modified directly by the 8051 by six instructions: PUSH, POP, ACALL, LCALL, RET, and RETI. It is also used intrinsically whenever an interrupt is triggered.

Fig. 1.2 Block diagram of 8051
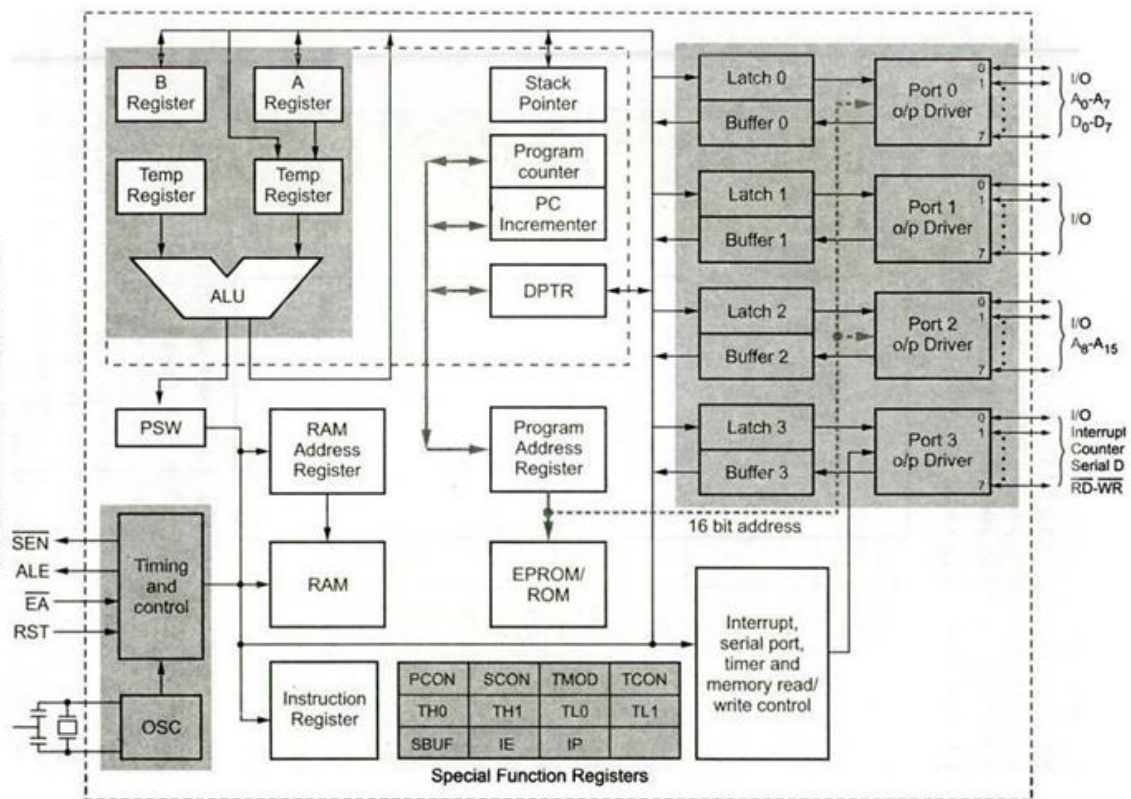
**STACK in 8051 Microcontroller:** The stack is a part of RAM used by the CPU to store information temporarily. This information may be either data or an address. The CPU needs this storage area as there are only limited number of registers. The register used to access the stack is called the Stack pointer which is an 8-bit register..So, it can take values of 00 to FFH. When the 8051 is powered up, the SP register contains the value 07.i.e the RAM location value 08 is the first location being used for the stack by the 8051 controller

There are two important instructions to handle this stack. One is the PUSH and the other is the POP. The loading of data from CPU registers to the stack is done by PUSH and the loading of the contents of the stack back into a CPU register is done by POP.

13

EX : MOV R6 ,#35 H

MOV R1 ,#21 H

PUSH 6

PUSH 1

In the above instructions the contents of the Registers R6 and R1 are moved to stack and they occupy the 08 and 09 locations of the stack. Now the contents of the SP are incremented by two and it is 0A

Similarly POP 3 instruction pops the contents of stack into R3 register. Now the contents of the SP is decremented by 1

In 8051 the RAM locations 08 to 1F (24 bytes) can be used for the Stack. In any program if we need more than 24 bytes of stack ,we can change the SP point to RAM locations 30-7F H. this can be done with the instruction MOV SP,# XX.

**Data Pointer Register(DPTR) :** It is a 16-bit register which is the only user-accessible. DPTR, as the name suggests, is used to point to data. It is used by a number of commands which allow the 8051 to access external memory. When the 8051 accesses external memory it will access external memory at the address indicated by DPTR. This DPTR can also be used as two 8-registers DPH and DPL.

**Program Status Register (PSW):** The 8051 has a 8-bit PSW register which is also known as Flag register. In the 8-bit register only 6-bits are used by 8051.The two unused bits are user definable bits. In the 6-bits four of them are conditional flags .They are Carry –CY, Auxiliary Carry-AC, Parity-P, and Overflow-OV. These flag bits indicate some conditions that resulted after an instruction was executed.

| D7 | | | | | | | D0 |
|------|------|------|------|------|------|------|------|
| CY | AC | FO | RS1 | RS0 | OV | - | P |
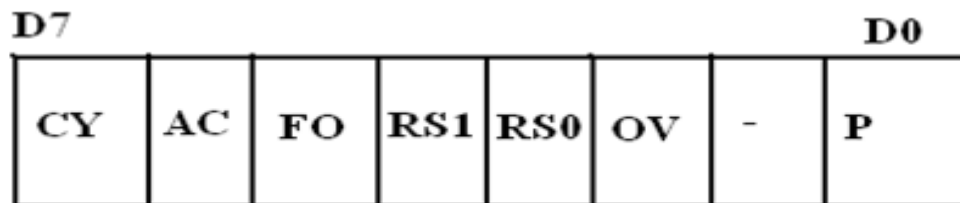
The bits PSW3 and PSW4 are denoted as RS0 and RS1 and these bits are used to select the bank registers of the RAM location. The meaning of various bits of PSW register is shown below.

| | | |
|---|---|---|
| CY | PSW.7 | Carry Flag |
| AC | PSW.6 | Auxiliary Carry Flag |
| FO | PSW.5 | Flag 0 available for general purpose |
| RS1 | PSW.4 | Register Bank select bit 1 |
| RS0 | PSW.3 | Register bank select bit 0 |
| OV | PSW.2 | Overflow flag |
| --- | PSW.1 | User definable flag |
| P | PSW.0 | Parity flag .set/cleared by hardware. |

The selection of the register Banks and their addresses are given below.

| RS1 | RS0 | Register Bank | Address |
|---|---|---|---|
| 0 | 0 | 0 | 00H-07H |
| 0 | 1 | 1 | 08H-0FH |
| 1 | 0 | 2 | 10H-17H |
| 1 | 1 | 3 | 18H-1FH |

**Memory organization :** The 8051 microcontroller has 128 bytes of Internal RAM and 4kB of on chip ROM .The RAM is also known as Data memory and the ROM is known as program memory. The program memory is also known as Code memory. This Code memory holds the actual 8051 program that is to be executed. In 8051 this memory is limited to 64K. Code memory may be found on-chip, as ROM or EPROM. It may also be stored completely off-chip in an external ROM or, more commonly, an external EPROM. The 8051 has only 128 bytes of Internal RAM but it supports 64kB of external RAM. As the name suggests, external RAM is any random access memory which is off-chip. Since the memory is off-chip it is not as flexible in terms of accessing, and is also slower. For example, to increment an Internal RAM location by 1, it requires only 1 instruction and 1 instruction cycle but to increment a 1-byte value stored in

External RAM requires 4 instructions and 7 instruction cycles. So, here the external memory is 7 times slower.

**Internal RAM OF 8051 :**

This Internal RAM is found on-chip on the 8051 .So it is the fastest RAM available, and it is also the most flexible in terms of reading, writing, and modifying it's contents. Internal RAM is volatile, so when the 8051 is reset this memory is cleared. The 128 bytes of internal RAM is organized as below.

(i) Four register banks (Bank0, Bank1, Bank2 and Bank3) each of 8-bits (total 32 bytes). The default bank register is Bank0. The remaining Banks are selected with the help of RS0 and RS1 bits of PSW Register.

(ii) 16 bytes of bit addressable area and

(iii) 80 bytes of general purpose area (Scratch pad memory) as shown in the diagram below. This area is also utilized by the microcontroller as a storage area for the operating stack.
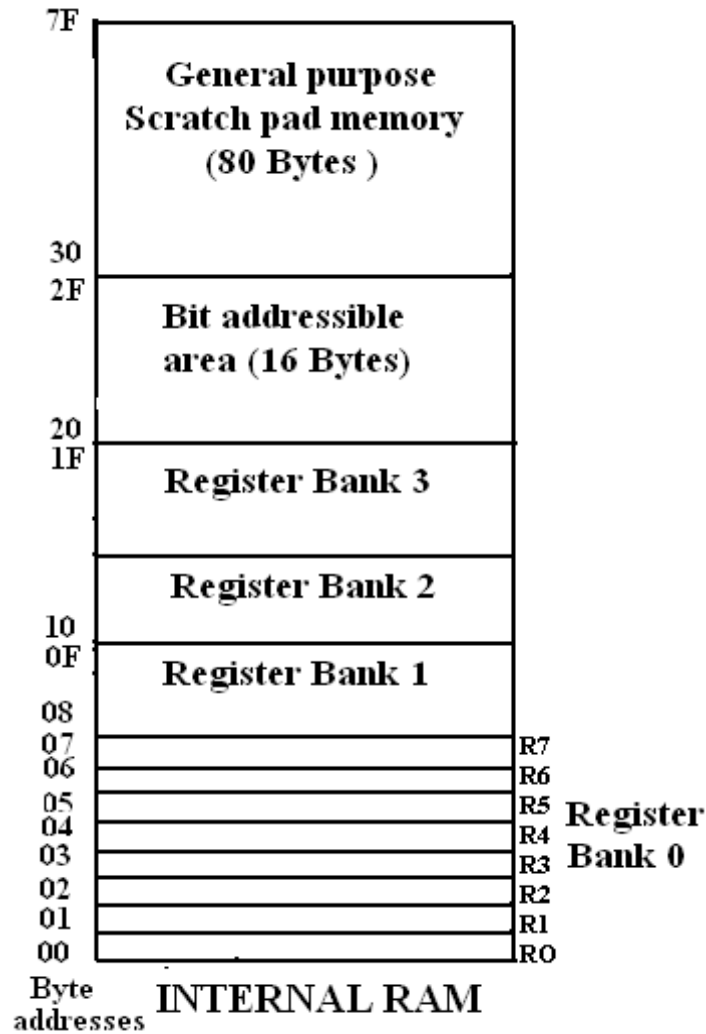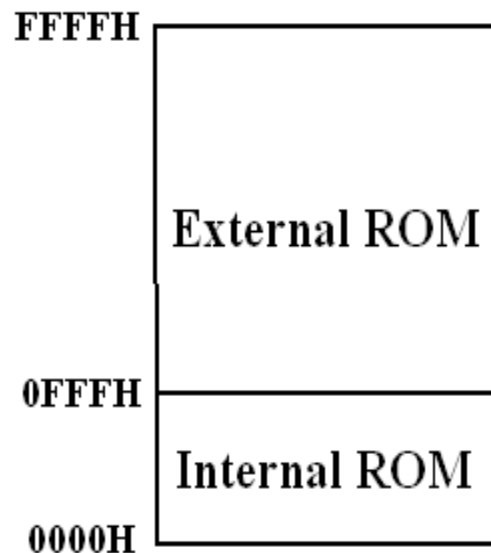
The 32 bytes of RAM from address 00 H to 1FH are used as working registers organized as four banks of eight registers each. The registers are named as R0-R7 .Each register can be addressed by its name or by its RAM address.

For     EX :  MOV A, R7     or    MOV R7,#05H

**Internal ROM (On –chip ROM):**  The 8051 microcontroller has 4kB of on chip ROM  but it can be extended up to  64kB. This ROM is also called program memory or code memory.  The CODE segment is accessed  using the program counter (PC) for opcode  fetches and by DPTR for data. The external ROM is accessed when the EA(active low)  pin is connected to ground or

the contents of program counter exceeds 0FFFH.When the Internal ROM address is exceeded the 8051 automatically fetches the code bytes from the external program memory.

```
FFFFH ┌─────────────────┐
      │                 │
      │                 │
      │                 │
      │  External ROM   │
      │                 │
      │                 │
0FFFH ├─────────────────┤
      │                 │
      │  Internal ROM   │
      │                 │
0000H └─────────────────┘
```

**SPECIAL FUNCTION REGISTERS (SFRs) :** In 8051 microcontroller there certain registers which uses the RAM addresses from 80h to FFh and they are meant for certain specific operations .These registers are called Special function registers (SFRs). Some of these registers are bit addressable also.

The list of SFRs and their functional names are given below. In these SFRs some of them are related to I/O ports (P0,P1,P2 and P3) and some of them are meant for control operations (TCON,SCON, PCON..) and remaining are the auxiliary SFRs, in the sense that they don't directly configure the 8051.

| S.No | Symbol | | Name of SFR | Address (Hex) |
|---|---|---|---|---|
| 1 | ACC* | | Accumulator | 0E0 |
| 2 | B* | | B-Register | 0F0 |
| 3 | PSW* | | Program Status word register | 0DO |
| 4 | SP | | Stack Pointer Register | 81 |
| 5 | DPTR | DPL | Data pointer low byte | 82 |
| | | DPH | Data pointer high byte | 83 |
| 6 | P0* | | Port 0 | 80 |
| | P1* | | Port 1 | 90 |
| 8 | P2* | | Port 2 | 0A |
| 9 | P3* | | Port 3 | 0B |
| 10 | IP* | | Interrupt Priority control | 0B8 |
| 11 | IE* | | Interrupt Enable control | 0A8 |
| 12 | TMOD | | Tmier mode register | 89 |
| 13 | TCON* | | Timer control register | 88 |
| 14 | TH0 | | Timer 0 Higher byte | 8C |
| 15 | TL0 | | Timer 0 Lower byte | 8A |
| 16 | TH1 | | Timer 1Higher byte | 8D |
| 17 | TL1 | | Timer 1 lower byte | 8B |
| 18 | SCON* | | Serial control register | 98 |
| 19 | SBUF | | Serial buffer register | 99 |
| 20 | PCON | | Power control register | 87 |

**The * indicates the bit addressable SFRs**

**Table:SFRs of 8051 Microcontroller**

## PARALLEL I /O PORTS :

The 8051 microcontroller has four parallel I/O ports , each of 8-bits. So, it provides the user 32 I/O lines for connecting the microcontroller to the peripherals. The four ports are P0 (Port 0), P1(Port1) ,P2(Port 2) and P3 (Port3). Upon reset all the ports are output ports. In order to make them input, all the ports must be set i.e a high bit must be sent to all the port pins. This is normally done by the instruction "SETB".

Ex: MOV A,#0FFH      ; A = FF

    MOV P0,A            ; make P0 an input port

## PORT 0:

Port 0 is an 8-bit I/O port with dual purpose. If external memory is used, these port pins are used for the lower address byte address/data ($AD_0$-$AD_7$), otherwise all bits of the port are either input or output.. Unlike other ports, Port 0 is not provided with pull-up resistors internally ,so for PORT0 pull-up resistors of nearly 10k are to be connected externally as shown in the fig.2.

**Dual role of port 0**: Port 0 can also be used as address/data bus(AD0-AD7), allowing it to be used for both address and data. When connecting the 8051 to an external memory, port 0 provides both address and data. The 8051 multiplexes address and data through port 0 to save the pins. ALE indicates whether P0 has address or data. When ALE = 0, it provides data D0-D7, and when ALE =1 it provides address and data with the help of a 74LS373 latch.
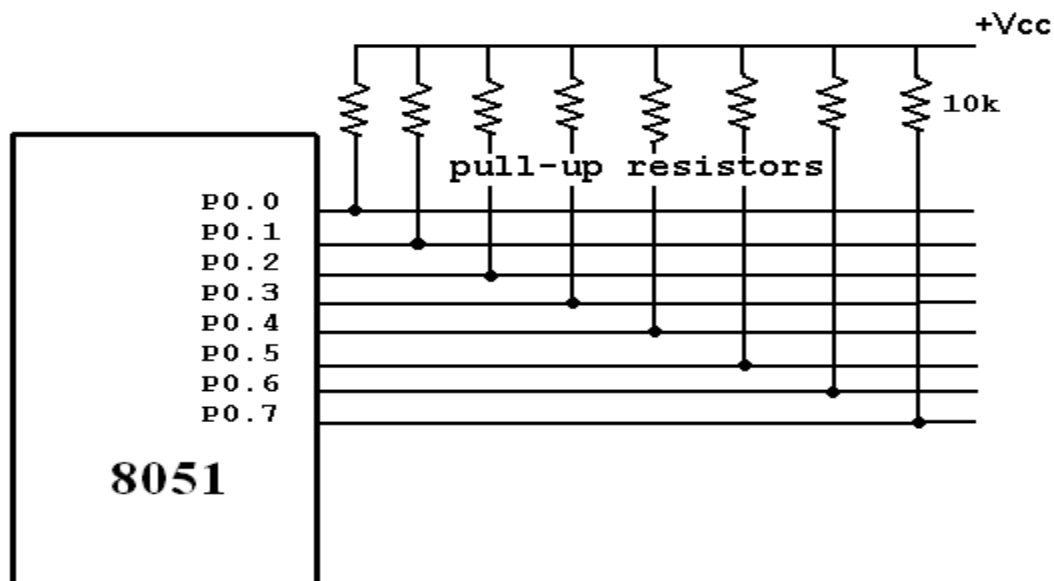
**Port 1:** Port 1 occupies a total of 8 pins (pins 1 through 8). It has no dual application and acts only as input or output port. In contrast to port 0, this port does not need any pull-up resistors since pull-up resistors connected internally. Upon reset, Port 1 is configured as an output port. To configure it as an input port , port bits must be set i.e a high bit must be sent to all the port pins. This is normally done by the instruction "SETB". For Ex :

MOV A, #0FFH ; A=FF HEX

MOV P1,A ; make P1 an input port by writing 1's to all of its pins

**Port 2 :** Port 2 is also an eight bit parallel port. (pins 21- 28). It can be used as input or output port. As this port is provided with internal pull-up resistors it does not need any external pull-up resistors. Upon reset, Port 2 is configured as an output port. If the port is to be used as input port, all the port bits must be made high by sending FF to the port. For ex,

MOV A, #0FFH ; A=FF hex

MOV $P_2$, A ; make $P_2$ an input port by writing all 1's to it

**Dual role of port 2** : Port2 lines are also associated with the higher order address lines A8-A15. In systems based on the 8751, 8951, and DS5000, Port2 is used as simple I/O port.. But, in 8031-based systems, port 2 is used along with P0 to provide the 16-bit address for the external memory. Since an 8031 is capable of accessing 64K bytes of external memory, it needs a path for the 16 bits of the address. While P0 provides the lower 8 bits via A0-A7, it is the job of P2 to provide bits A8-A15 of the address. In other words, when 8031 is connected to external memory, Port 2 is used for the upper 8 bits of the 16 bit address, and it cannot be used for I/O operations.

**PORT 3** : Port3 is also an 8-bit parallel port with dual function.( pins 10 to 17). The port pins can be used for I/O operations as well as for control operations. The details of these additional operations are given below in the table. Port 3 also do not need any external pull-up resistors as they are provided internally similar to the case of Port2 & Port 1. Upon reset port 3 is configured as an output port . If the port is to be used as input port, all the port bits must be made high by sending FF to the port. For ex,

MOV A, #0FFH ; A= FF hex

MOV $P_3$, A ; make $P_3$ an input port by writing all 1's to it

**Alternate Functions of Port 3 :** P3.0 and P3.1 are used for the RxD (Receive Data) and TxD (Transmit Data) serial communications signals. Bits P3.2 and P3.3 are meant for external interrupts. Bits P3.4 and P3.5 are used for Timers 0 and 1 and P3.6 and P3.7 are used to provide the write and read signals of external memories connected in 8031 based systems

| S.No | Port 3 bit | Pin No | Function |
|------|-----------|--------|----------|
| 1 | P3.0 | 10 | RxD |
| 2 | P3.1 | 11 | TxD |
| 3 | P3.2 | 12 | $\overline{\text{INT0}}$ |
| 4 | P3.3 | 13 | $\overline{\text{INT1}}$ |
| 5 | P3.4 | 14 | T0 |
| 6 | P3.5 | 15 | T1 |
| 7 | P3.6 | 16 | $\overline{\text{WR}}$ |
| 8 | P3.7 | 17 | $\overline{\text{RD}}$ |

**Table: PORT 3 alternate functions**

**Interrupt Structure**: An interrupt is an external or internal event that disturbs the microcontroller to inform it that a device needs its service. The program which is associated with the interrupt is called the **interrupt service routine** (ISR) or **interrupt handler**. Upon receiving the interrupt signal the Microcontroller , finish current instruction and saves the PC on stack. Jumps to a fixed location in memory depending on type of interrupt Starts to execute the interrupt service routine until RETI (return from interrupt)Upon executing the RETI the microcontroller returns to the place where it was interrupted. Get pop PC from stack

The 8051 microcontroller has **FIVE** interrupts in addition to Reset. They are

- Timer 0 overflow  Interrupt

- Timer 1 overflow Interrupt

- External Interrupt 0(INT0)

- External Interrupt 1(INT1)

- Serial Port events (buffer full, buffer empty, etc) Interrupt

Each interrupt has a specific place in code memory where program execution (interrupt service routine) begins.

- External Interrupt 0:   0003 H

- Timer 0 overflow:      000B H

- External Interrupt 1:   0013 H

- Timer 1 overflow:      001B H

- Serial  Interrupt :       0023 H

Upon reset all Interrupts are disabled & do not respond to the Microcontroller. These interrupts must be enabled by software in order for the Microcontroller to respond to them. This is done by an 8-bit register called Interrupt Enable Register (IE).

**Interrupt Enable Register :**

| EA | — | ET2 | ES | ET1 | EX1 | ET0 | EX0 |
|----|----|-----|----|-----|-----|-----|-----|

- EA   : Global enable/disable. To enable the interrupts this bit must be set High.

- ---      : Undefined-reserved for future use.

- ET2 : Enable /disable  Timer 2  overflow interrupt.

- ES   : Enable/disable  Serial port interrupt.

- ET1 : Enable /disable Timer 1  overflow interrupt.

- EX1 : Enable/disable  External  interrupt1.

- ET0 :  Enable /disable  Timer 0 overflow  interrupt.

- EX0 : Enable/disable  External  interrupt0

Upon reset the interrupts have the following priority.(Top to down).  The interrupt with the highest PRIORITY gets serviced first.

1. External interrupt 0 (INT0)

2. Timer interrupt0 (TF0)

3. External interrupt 1 (INT1)

4. Timer interrupt1 (TF1)

5. Serial communication (RI+TI)

Priority can also be set to "high" or "low" by 8-bit IP register.- Interrupt priority register

| — | — | PT2 | PS | PT1 | PX1 | PT0 | PX0 |
|---|---|---|---|---|---|---|---|

IP.7: reserved

IP.6: reserved

IP.5: Timer 2 interrupt priority bit (8052 only)

IP.4: Serial port interrupt priority bit

IP.3: Timer 1 interrupt priority bit

IP.2: External interrupt 1 priority bit

IP.1: Timer 0 interrupt priority bit

IP.0: External interrupt 0 priority bit

**TIMERS in 8051 Microcontrollers :** The 8051 microcontroller has two 16-bit timers Timer 0 (T0) and Timer 1(T1) which can be used either to generate accurate time delays or as event counters. These timers are accessed as two 8-bit registers TLO, THO & TL1, TH1 because the 8051 microcontroller has 8-bit architecture.

**TIMER 0 :** The Timer 0 is a 16-bit register and can be treated as two 8-bit registers (TL0 & TH0) and these registers can be accessed similar to any other registers like A,B or R1,R2,R3 etc…

Ex : The instruction Mov TL0,#07 moves the value 07 into lower byte of Timer0.

Similarly Mov R5,TH0 saves the contents of TH0 in the R5 register.
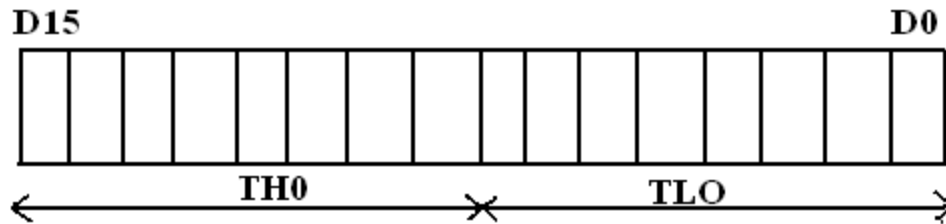
**TIMER 1 :** The Timer 1 is also a 16-bit register and can be treated as two 8-bit registers (TL1 & TH1) and these registers can be accessed similar to any other registers like A,B or R1,R2,R3 etc…

Ex : The instruction MOV TL1,#05 moves the value 05 into lower byte of Timer1.

Similarly MOV R0,TH1 saves the contents of TH1 in the R0 register
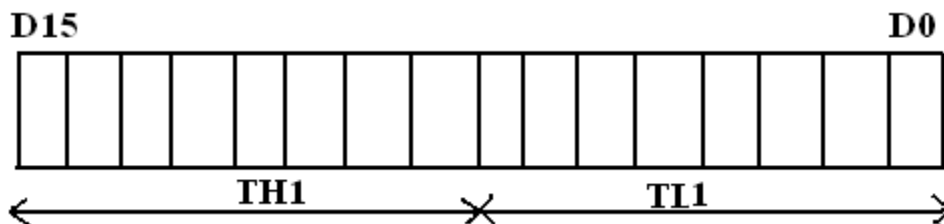


**TMOD Register :** The various operating modes of both the timers T0 and T1 are set by an 8-bit register called TMOD register**.** In this TMOD register the lower 4-bits are meant for Timer 0 and the higher 4-bits are meant for Timer1.
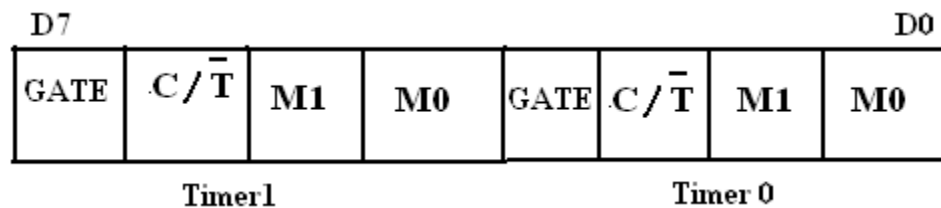


**GATE**: This bit is used to start or stop the timers by hardware .When GATE= 1, the timers can be started / stopped by the external sources. When GATE= 0, the timers can be started or stopped by software instructions like SETB TR0 or SETB TR1
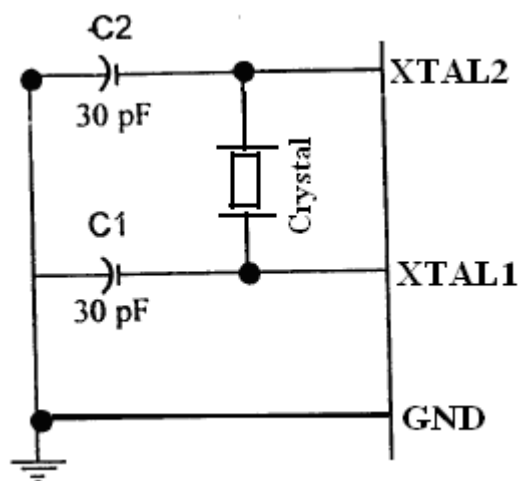
**C/T (clock/Timer):** This bit decides whether the timer is used as delay generator or event counter. When **C/T = 0,** the Timer is used as delay generator and if C/T=1 the timer is used as an event counter. The clock source for the time delay is the crystal frequency of 8051.
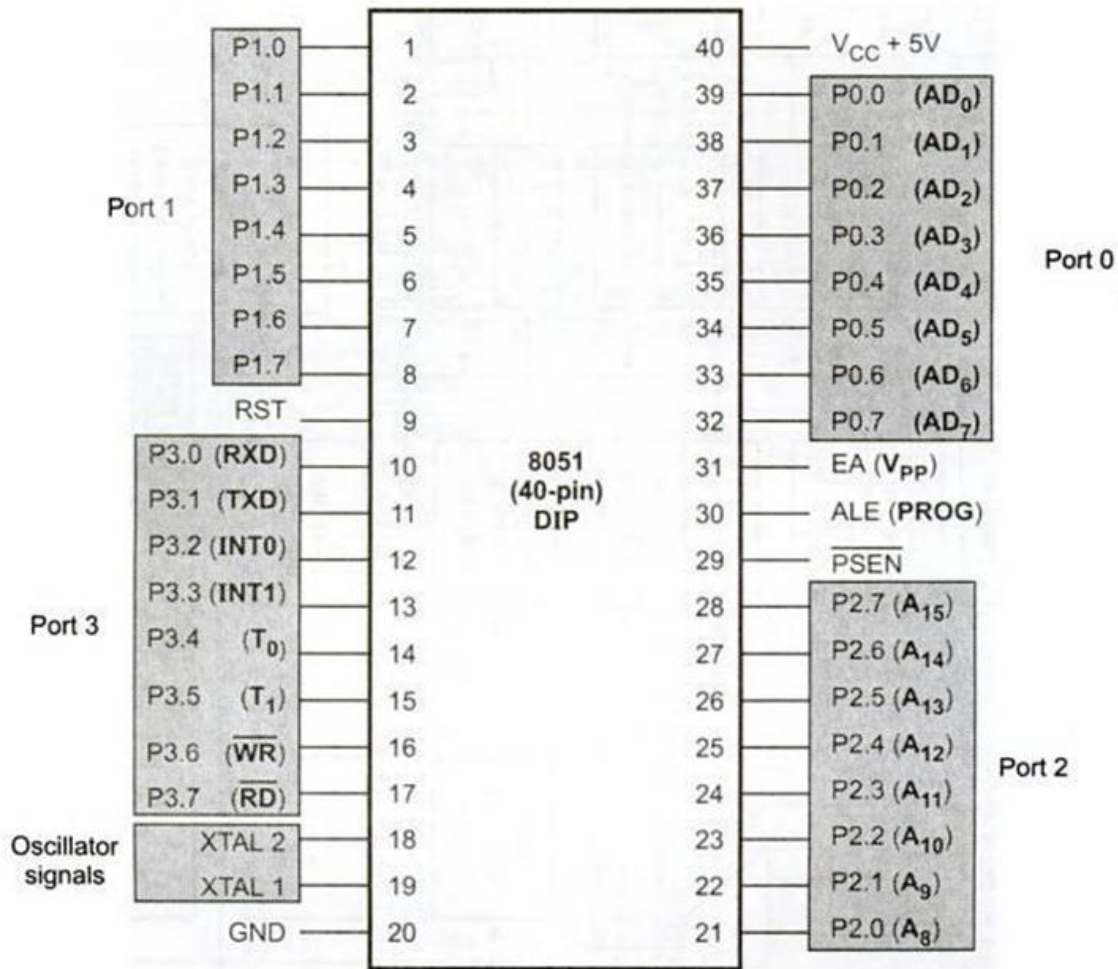
**M1, M0 (Mode) :** These two bits are the timer mode bits. The timers of the 8051 can be configured in three modes.Mode0, Mode1 and Mode2. The selection and operation of the modes is shown below.

| S.No | M0 | M1 | Mode | Operation |
|------|----|----|------|-----------|
| 1 | 0 | 0 | 0 | 13-bit Timer mode<br>8-bit Timer/counter. THx with TLx as 5-bit prescalar |
| 2 | 0 | 1 | 1 | 16-bit Timer mode.16-bit timer /counter without pre-scalar |
| 3 | 1 | 0 | 2 | 8-bit auto reload. THx contains a value that is to be loaded into TLx each time it overflows |
| 4 | 1 | 1 | 3 | Split timer mode |

**PIN Diagram of 8051 Microcontroller :** The 8051 microcontroller is available as a 40 pin DIP chip and it works at +5 volts DC. Among the 40 pins , a total of 32 pins are allotted for the four parallel ports P0,P1,P2 and P3 i.e each port occupies 8-pins .The remaining pins are VCC, GND, XTAL1, XTAL2, RST, EA ,PSEN.

**XTAL1,XTAL2**: These two pins are connected to Quartz crystal oscillator which runs the on-chip oscillator. The quartz crystal oscillator is connected to the two pins along with a capacitor of 30pF as shown in the circuit. If we use a source other than the crystal oscillator, it will be connected to XTAL1 and XTAL2 is left unconnected.

**8051 (40-pin) DIP**

Port 1:
- P1.0 — 1
- P1.1 — 2
- P1.2 — 3
- P1.3 — 4
- P1.4 — 5
- P1.5 — 6
- P1.6 — 7
- P1.7 — 8

- RST — 9

Port 3:
- P3.0 (RXD) — 10
- P3.1 (TXD) — 11
- P3.2 (INT0) — 12
- P3.3 (INT1) — 13
- P3.4 (T$_0$) — 14
- P3.5 (T$_1$) — 15
- P3.6 ($\overline{\text{WR}}$) — 16
- P3.7 ($\overline{\text{RD}}$) — 17

Oscillator signals:
- XTAL 2 — 18
- XTAL 1 — 19

- GND — 20

Right side:
- 40 — V$_{CC}$ + 5V
- 39 — P0.0 (AD$_0$)
- 38 — P0.1 (AD$_1$)
- 37 — P0.2 (AD$_2$)
- 36 — P0.3 (AD$_3$)
- 35 — P0.4 (AD$_4$)
- 34 — P0.5 (AD$_5$)
- 33 — P0.6 (AD$_6$)
- 32 — P0.7 (AD$_7$)  — Port 0
- 31 — EA (V$_{PP}$)
- 30 — ALE (PROG)
- 29 — $\overline{\text{PSEN}}$
- 28 — P2.7 (A$_{15}$)
- 27 — P2.6 (A$_{14}$)
- 26 — P2.5 (A$_{13}$)
- 25 — P2.4 (A$_{12}$)
- 24 — P2.3 (A$_{11}$)
- 23 — P2.2 (A$_{10}$)
- 22 — P2.1 (A$_9$)
- 21 — P2.0 (A$_8$)  — Port 2

**RST**: The RESET pin is an input pin and it is an active high pin. When a high pulse is applied to this pin the microcontroller will reset and terminate all activities. Upon reset all the registers except PC will reset to 0000 Value and PC register will reset to 0007 value.

$\overline{\text{EA}}$ **(External Access):** This pin is an active low pin. This pin is connected to ground when microcontroller is accessing the program code stored in the external memory and connected to Vcc when it is accessing the program code in the on chip memory. This pin should not be left unconnected.

$\overline{\text{PSEN}}$ **(Program Store Enable) :** This is an output pin which is active low. When the microcontroller is accessing the program code stored in the external ROM ,this pin is connected to the OE (Output Enable) pin of the ROM.

**ALE (Address latch enable):** This is an output pin, which is active high**.** When connected to external memory , port 0 provides both address and data i.e address and data are multiplexed through port 0 .This ALE pin will demultiplex the address and data bus .When the pin is High , the AD bus will act as address bus  otherwise the AD bus will act as Data bus.

**P0.0- P0.7(AD0-AD7) :** The port 0 pins multiplexed with Address/data pins .If the microcontroller is accessing external memory these pins will act as address/data pins otherwise they are used for Port 0 pins.

**P2.0- P2.7(A8-A15) :** The port2 pins are multiplexed with the higher order address pins **.**When the microcontroller is accessing external memory these pins provide the higher order address byte otherwise they act as Port 2 pins.

**P1.0- P1.7 :**These 8-pins are dedicated for Port1 to perform input or output port operations.

**P3.0- P3.7  :**These 8-pins are meant for Port3 operations and also for some control operations like  Read,Write,Timer0,Timer1 ,INT0,INT1 ,RxD and TxD

**ADDRESSING MODES OF 8051 :**

The way in which the data operands are accessed by different instructions is known as  the addressing modes. There are various methods of denoting the data operands in the instruction. The 8051 microcontroller supports  mainly 5 addressing modes. They are

1.Immediate addressing mode

2.Direct Addressing mode

3.Register addressing mode

4. Register Indirect addressing mode

5.Indexed addressing mode

**Immediate addressing mode :** The addressing mode in which the data operand is a constant and it is a  part of the instruction itself is known as Immediate addressing mode. Normally the data must be preceded by a # sign. This addressing mode can be used to transfer the data into any of the registers including DPTR.

Ex: MOV A , # 27 H   : The data (constant) 27 is moved to the accumulator register

ADD R1 ,#45 H : Add the constant 45 to the contents of the accumulator

MOV DPTR ,# 8245H :Move the data  8245 into the data pointer register.

MOV P1,#21 H

**Direct addressing mode**: The addressing mode in which the data operand is in the RAM location (00 -7FH) and the address of the data operand  is given in the instruction is known as Direct addressing mode. The direct addressing mode uses the lower 128 bytes of Internal RAM and the SFRs

MOV R1, 42H   : Move the contents of RAM location 42 into R1 register

MOV 49H,A : Move the contents of the accumulator into the RAM location 49.

ADD A, 56H : Add the contents of the RAM location 56 to the accumulator

**Register addressing mode** :The addressing mode in which the data operand to be manipulated lies in one of the registers  is known as register addressing mode.

MOV A,R0 : Move the contents of the register R0 to the accumulator

ADD A,R6 :Add the contents of R6 register to the accumulator

MOV P1, R2 :   Move the contents of the R2 register into port 1

MOV R5, R2  : This is invalid .The data transfer between the registers is not allowed.

**Register Indirect addressing mode :**The addressing mode in which a register is used as a pointer to the data memory block is known as Register indirect addressing mode.

MOV  A,@  R0 :Move  the  contents  of   RAM  location  whose  address  is  in  R0  into **A** (accumulator)

MOV @ R1 , B : Move the contents of B into RAM location whose address is held by R1

When R0 and R1 are used as pointers, they must be preceded by @ sign

**One of the advantages of register indirect addressing mode is that it makes accessing the data more dynamic than static as in the case of direct addressing mode.**

**Indexed addressing mode :** This addressing mode is used in accessing the data elements of lookup table entries located in program ROM space of 8051.

Ex : MOVC A,@ A+DPTR

The 16-bit register DPTR and register A are used to form the address of the data element stored in on-chip ROM. Here C denotes code .In this instruction the contents of A are added to the 16-bit DPTR register to form the 16-bit address of the data operand.

**Interfacing of ADC 0804 to 8051 Microcontroller :**

ADC 0804 is a single channel analog to digital converter i.e., it can take only one analog signal. ADC 0804 has 8 bit resolution. The higher resolution ADC gives smaller step size. Step size is smallest change that can be measured by an ADC. For an ADC with resolution of 8 bits, the step size is 19.53mV (5V/255). The time taken by the ADC to convert analog data into digital form depends on the frequency of clock source. The conversion time of ADC 0804 is around 110us. To use the internal clock a capacitor and resistor are used as shown in the circuit. The input to the ADC is given from a regulated power supply and a 10K potentiometer

The 8051 Microcontroller is used to provide the control signals to the ADC. CS(chip select) pin of ADC is directly connected to ground. The pin P1.1, P1.0 and P1.2 are connected to the pin WR, RD and INTR of the ADC respectively. When the input voltage from the preset is varied the output of ADC varies also varies.
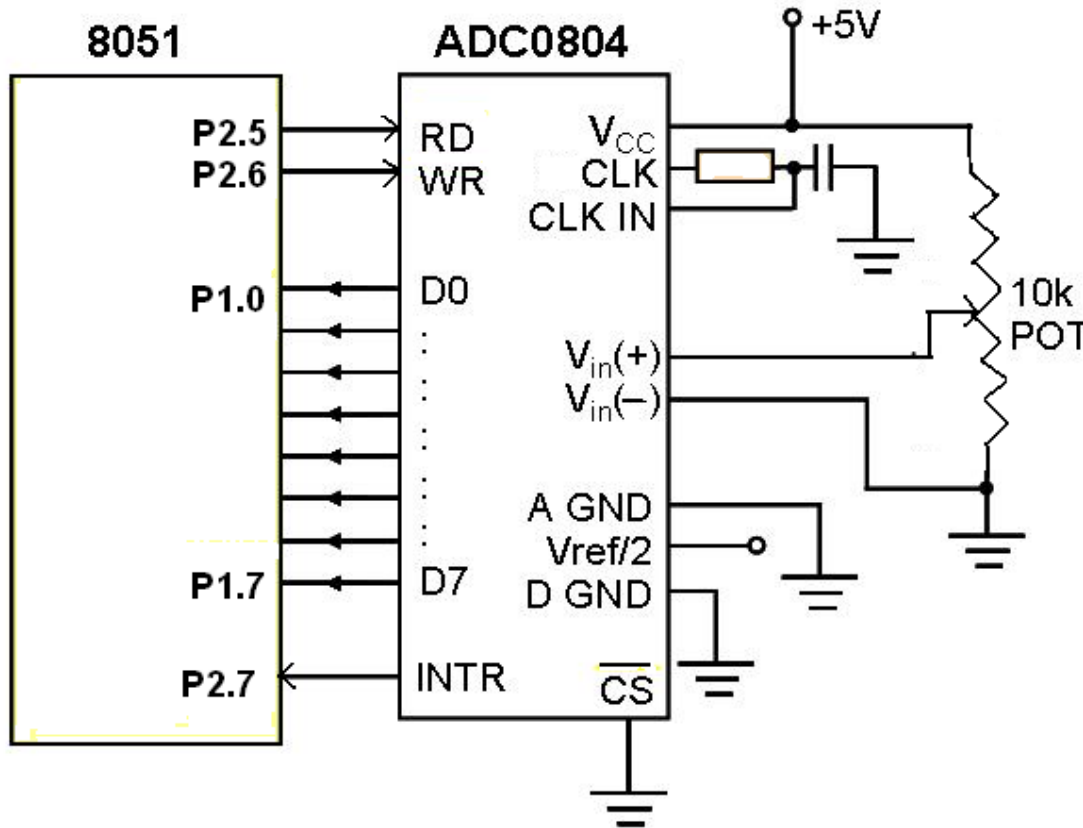
From the circuit it is clear that the ADC interfaced directly to the microcontroller. The Port1 is used as an input port which receives the digital data from the ADC.Port pins P2.5 and P2.6 are used for SOC and EOC operation.When the conversion is over the ADC will send an interrupt signal to the microcontroller through the pin P2.7 .Now the Microcontroller receives digital data through the Port1.This data after conversion to decimal data is displayed on the LCD module .

The assembly language program for ADC is given below .

        MOV P1 , 0FF H  ;  Make the port1 high and configure port1 as Input port

**BACK**: CLR P2.6            ; Generation of SOC pulse

        SETB P2.5        ;

**LOOP**   JB P2.7, LOOP   ; Wait for conversion, Is conversion over?

        CLR  P2.5         ;   Enable Read the digital data

```
MOV A ,P1           ;  Read digital data through Port1

SETB P2.5           ;   Disable read after read operation

CALL DISPLAY  ;  Display the data on LCD module

SJMP BACK         ; Continue the conversion process
```

## Stepper motor Interfacing

A stepper motor is a device that translates electrical pulses into mechanical movement. The stepper motor rotates in steps in response to the applied signals. It is used in  applications such as disk drives, dot matrix printers, plotters and robotics. It is mainly used for position control. Stepper motors  have a permanent magnet  called rotor (also called the shaft) surrounded by a stator . There are also steppers called variable reluctance stepper motors that do not have a PM rotor. The most common stepper motors have four stator windings that are paired with a center-tapped. This type of stepper motor is commonly referred to as a. four-phase or unipolar stepper motor. The center tap allows a change of current direction in each of two coils when a winding is grounded, thereby resulting in a polarity change of the stator.
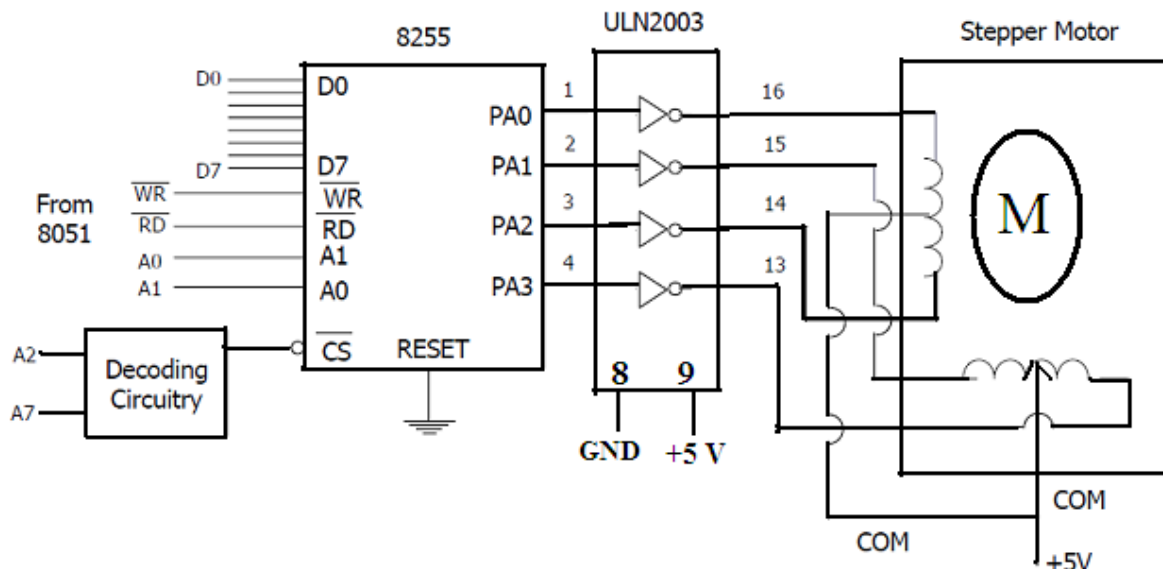
**ASSEMBLY LANGUAGE PROGRAM**

```
Main    mov stepper,  #0CH ;  move the  code to phase1 into the port
        acall delay
        mov stepper, #06H  ; phase II code
        acall delay
        mov stepper, #03H      ;Phase III code
        acall delay                   ;Call delay subroutine program
        mov stepper, #09H      ;Phase IV code
        acall delay
        sjmp   Main
    CALL DELAY PROGRAM :
        mov r7,#4
        wait2:
        mov r6,#0FFH
        wait1:
        mov r5,#0FFH
        wait:
        djnz r5,wait
        djnz r6,wait1
        djnz r7,wait2
        ret
        end
```

# 8051-SERIAL COMMUNICATION :

## Basics of Serial communication

Data transfer between two electronic devices (Ex Between a computer and microcontroller or a peripheral device) is generally done in two ways
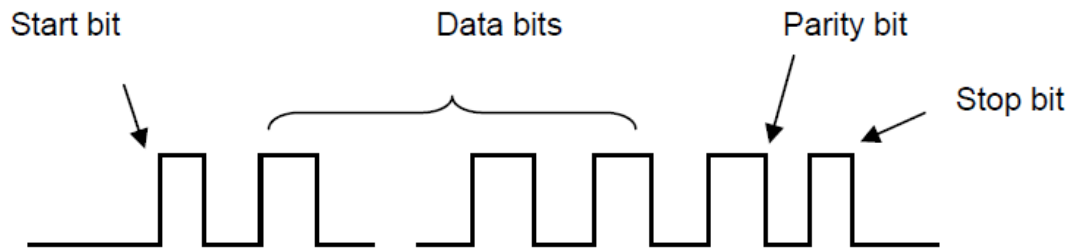
(i).Serial data Transfer     and

(ii).Parallel data Transfer

Serial communication uses only one or two data lines to transfer data and is generally used for long distance communication. In serial communication the data is sent as one bit at a time in a timed sequence on a single wire. Serial Communication takes place in two methods, Asynchronous data Transfer and Synchronous data Transfer.
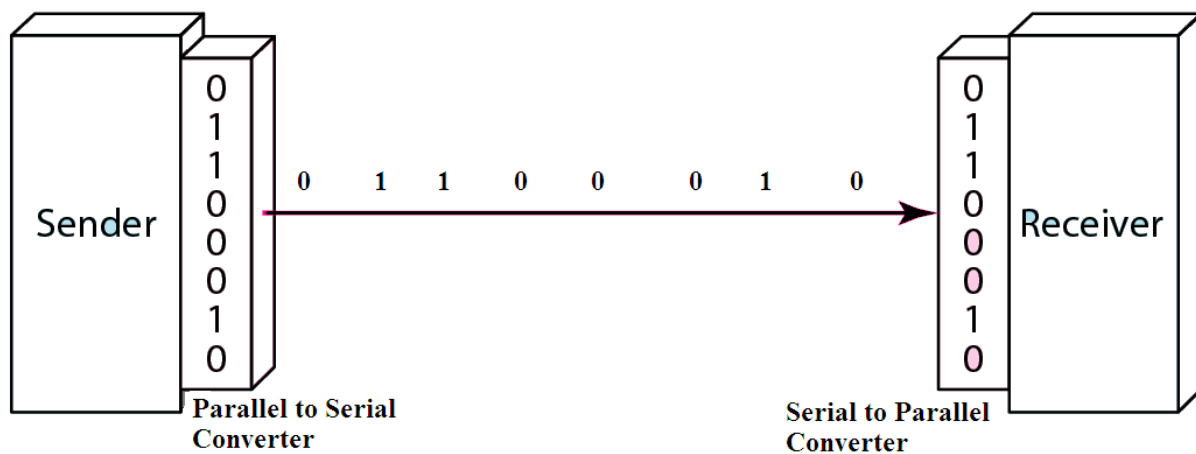


**Serial Data Transfer**

 **Asynchronous data transfer** allows data to be transmitted without the sender having to send a clock signal to the receiver. Instead, special bits will be added to each word in order to synchronize the sending and receiving of the data. When a word is given to the UART for Asynchronous transmissions, a bit called the "Start Bit" is added to the beginning of each word that is to be transmitted. The Start Bit is used to alert the receiver that a word of data is about to be sent, and to force the clock in the receiver into synchronization with the clock in the transmitter.

**Serial data transmission**

After the Start Bit, the individual bits of the word of data are sent .Here each bit in the word is transmitted for exactly the same amount of time as all of the other bits. When the entire data word has been sent, the transmitter may add a Parity Bit that the transmitter generates. The Parity bit may be used by the receiver to perform simple error checking. Then at least one Stop Bit is sent by the transmitter. If the Stop Bit does not appear when it is supposed to, the UART considers the entire word to be corrupted and will report a Framing Error.

Baud rate is a measurement of transmission speed in asynchronous communication , it represents the number of bits/sec that are actually being sent over the serial link. The Baud count includes the overhead bits Start, Stop and Parity that are generated by the sending UART and removed by the receiving UART.

In the **Synchronous data transfer** method the receiver knows when to "read" the next bit coming from the sender. This is achieved by sharing a clock between sender and receiver. In most forms of serial Synchronous communication, if there is no data available at a given time to transmit, a fill character will be sent instead so that data is always being transmitted. Synchronous communication is usually more efficient because only data bits are transmitted between sender and receiver, however it will be more costly because extra wiring and control circuits are required to share a clock signal between the sender and receiver.

Devices that use serial cables for their communication are split into two categories.

1. DTE (Data Terminal Equipment). Examples of DTE are computers, printers & terminals.
2. DCE (Data Communication Equipment). Example of DCE is modems.

**Parallel Data Transfer :**

Parallel communication uses multiple wires (bus) running parallel to each other, and can transmit data on all the wires simultaneously. i.e all the bits of the byte are transmitted at a time.

So, speed of the parallel data transfer is extremely high compared to serial data transfer. An 8-bit parallel data transfer is 8-times faster than serial data transfer. Hence with in the computer all data transfer is mainly based on Parallel data transfer. But only limitation is due to the high cost ,this method is limited to only short distance communications.

**Differences between Serial data transfer and Parallel data transfer**

| S.No | Serial Communication | Parallel Communication |
|------|----------------------|------------------------|
| 1 | Data is transmitted bit after the bit in a single line | Data is transmitted simultaneously through group of lines(Bus) |
| 2 | Data congestion takes place | No, Data congestion |
| 3 | Low speed transmission | High speed transmission |
| 4 | Implementation of serial links is not an easy task. | Parallel data links are easily implemented in hardware |
| 5. | In terms of transmission channel costs such as data bus cable length, data bus buffers, interface connectors, it is less expensive | It is more expensive |
| 6 | No , crosstalk problem | Crosstalk creates interference between the parallel lines. |
| 7 | No effect of inter symbol interference and noise | Parallel ports suffer extremely from inter-symbol interference (ISI) and noise, and therefore the data can be corrupted over long distances. |
| 8 | The bandwidth of serial wires is much higher. | The bandwidth of parallel wires is much lower. |
| 9 | Serial interface is more flexible to upgrade , without changing the hardware | Parallel data transfer mechanism rely on hardware resources and hence not flexible to upgrade. |
| 10 | Serial communication work effectively even at high frequencies. | Parallel buses are hard to run at high frequencies. |

**SERIAL COMMUNICATION IN 8051 MICROCONTROLLER**

The 8051 has two pins for transferring and receiving data by serial communication. These two pins are part of the Port3(P3.0 &P3.1) .These pins are TTL compatible and hence they require a line driver to make them RS232 compatible .Max232 chip is one such line driver in use. Serial communication is controlled by an 8-bit register called SCON register,it is a bit addressable register.

**SCON (Serial control) register :**

| SM0 | SM1 | SM2 | REN | TB8 | RB8 | TI | RI |
|-----|-----|-----|-----|-----|-----|-----|-----|

| SM0 | SCON.7 | Serial port mode selector |
|-----|--------|---------------------------|
| SM1 | SCON.6 | Serial port mode selector |
| SM2 | SCON.5 | Used for multiprocessor mode communication (not applicable for 8051) |
| REN | SCON.4 | Receive enable. Set or cleared by making this bit either 1 or 0 foe enable /disable reception. |
| TB8 | SCON.3 | $9^{th}$ data bit transmitted in modes 2 and 3 |
| RB8 | SCON.2 | $9^{th}$ data bit received in modes 2 and 3.it is not used in mode 0 & mode 1.If SM2 = 0 RB8 is the stop bit . |
| TI | SCON.1 | Transmit interrupt flag |
| RI | SCON.0 | Receive interrupt flag. |

- M0 , SM1 : These two bits of SCON register determine the framing of data by specifying the number of bits per character and start bit and stop bits. There are 4 serial modes.

  SM0   SM1

   0      0    :  Serial Mode 0

   0      1    :  Serial Mode 1,  8 bit data,   1 stop bit, 1 start bit

   1      0    :  Serial Mode 2

   1      1    :  Serial Mode 3


- REN (Receive Enable) also referred as SCON.4. When it is high,it allows the 8051 to receive data on the RxD pin. So to receive and transfer data REN must be set to 1.When REN=0,the receiver is disabled. This is achieved as below

  SETB  SCON.4

  &   CLR   SCON.4

- TI (Transmit interrupt) is the D1 bit of SCON register. When 8051 finishes the transfer of 8-bit character, it raises the TI flag to indicate that it is ready to transfer another  byte. The TI bit is raised at the beginning of the stop bit.

- RI (Receive interrupt) is the D0 bit of the SCON register. When the 8051 receives data serially ,via  RxD, it gets rid of the start and stop bits and places the byte in the SBUF register. Then it raises the RI flag bit to indicate that a byte has been received and should be picked up before it is lost. RI is raised halfway through the stop bit.

**Communication through RS232**

 A personal computer has a serial port known as communication port or COM Port used to connect a modem for example or any other device, there could be more then one COM Port in a PC. Serial ports are controlled by a special chip called UART (Universal Asynchronous Receiver Transmitter).

RS 232  standard describes a communication method where information is sent bit by bit on a physical channel. The **RS stands for Recommended Standard**.The information must be broken up in data words. The length of a data word is variable.

It is one of the popularly known interface standard for serial communication between DTE & DCE. This RS-232-C is the commonly used standard when data are transmitted as voltage .This standard was first developed by Electronic industries association(EIA). For the RS-232C, a 25 pin D type connector is used . DB-25P male and DB-25S female. RS-232 standard was first introduced in 1960's by Telecommunications Industry Association(TIA).

**Interfacing the 8051 Microcontroller to PC:**

As the RS-232 standard is developed earlier to TTL devices ,a USART such as 8251 is not directly compatible with these signal levels .Because of this ,voltage transistors called line drivers and line receivers are used to interface TTL logic with RS-232 signals . The line driver MC 1488 is used to convert RS-232 to TTL.The microcontroller is connected to the PC using the DB9 connector.
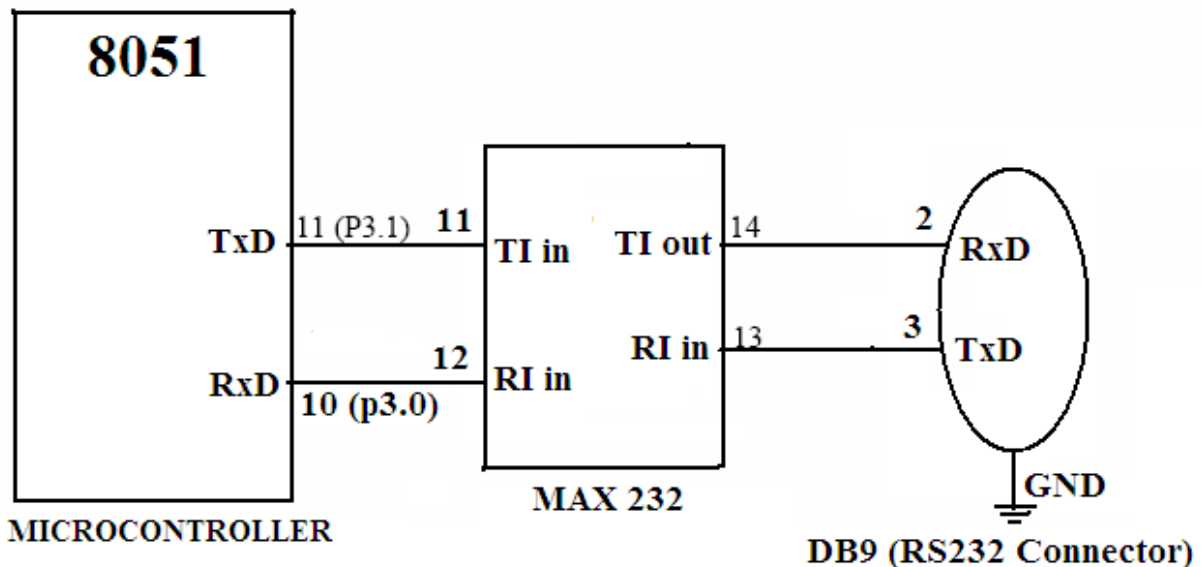


The TxD and Rx D pins are connected to the TI in and RI in pins of the MAX 232 IC and the TI out and RI in pins of the MAX IC are connected to the RxD and TxD pins of the DB9 connector as shown in the interface diagram.

## INTERFACING DC MOTOR- 8051

A DC motor runs with the help of Direct Current. It produces torque by using both electricity and magnetic fields. The DC motor has rotor, stator, field magnet, brushes, shaft, commutator. The DC motor requires more current to produce initial torque than in running state.Interfacing the DC motor directly to 8051 microcontroller is not possible. Because the DC motor uses large current (200-300mA in small DC motors) to run. When this current flow into the 8051 microcontroller, the IC will get damaged. Therefore we use a driving circuit with an opto isolator and a L298 Dual H-Bridge driver. The opto-isolator provides additional protection to the microcontroller.
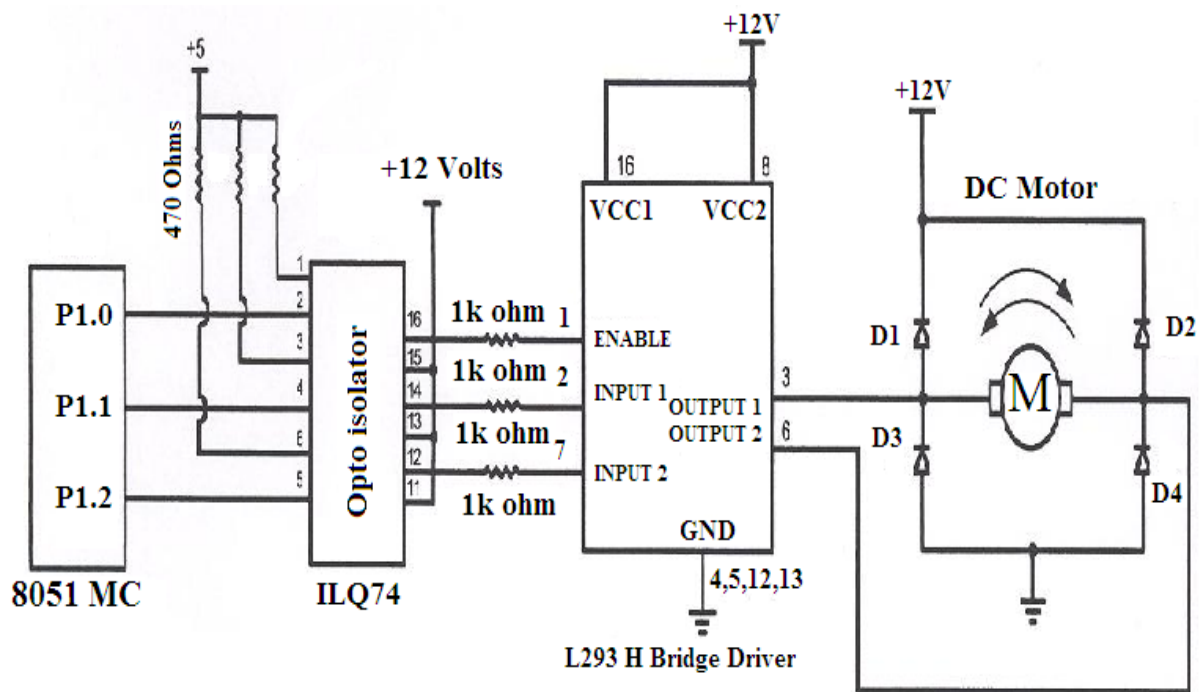


Continuous, sustained operation of the motor will cause the L293 Dual H-Bridge driver to overheat. So,a suitable heat sink must be used.

**Assembly Language program**

| ORG | 0000H | | Remarks |
|---|---|---|---|
| MAIN | CLR | P1.0 | |
| | CLR | P1.1 | |
| | CLR | P1.2 | |
| | SETB | P2.7 | |
| MONITOR | | | |
| | SETB | P1.0 | **Enable the H-bridge driver** |
| | JNB | P2.7 CLOCKWISE | |
| | CLR | P1.1 | **01 is for Counter clockwise** |
| | SETB | P1.2 | |
| | SJMP | MONITOR | |
| CLOCKWISE | SETB | P1.1 | **10 is for clockwise** |
| | CLR P1.2 | | |
| | SJMP | MONITOR | |

## INTERFACING DAC -8051 MICROCONTROLLER

The DAC 0800 is a simple monolithic 8-bit D/A converter.  It has fast settling time of 100ns. It can be directly interfaced to TTL, CMOS, PMOS and others. It operates at 4.5V to +18V supply. The number of data bit inputs decides the resolution of the DAC since 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 interfacing circuit is shown  below.  port 1(8 bits of the microcontroller is connected to the input  data  lines  of  DAC-08.The  reference  current  is  determined  by  the  resistor  $R_1$  and  the

reference voltage V ref. The resistor $R_2$ is generally equal to R1 to match the input impedance of reference source. The output (taken from pin number 4 is observed either on a digital multimeter or on a cathode ray oscilloscope.

The output current Io is calculated as follows:

$$I_o = V_{ref}/R_1[A_o/2 + A_1/4 + A_2/8 + \ldots + A7/256]$$

The output voltage Vo is obtained as follows:       $V_o = I_o * R_1$
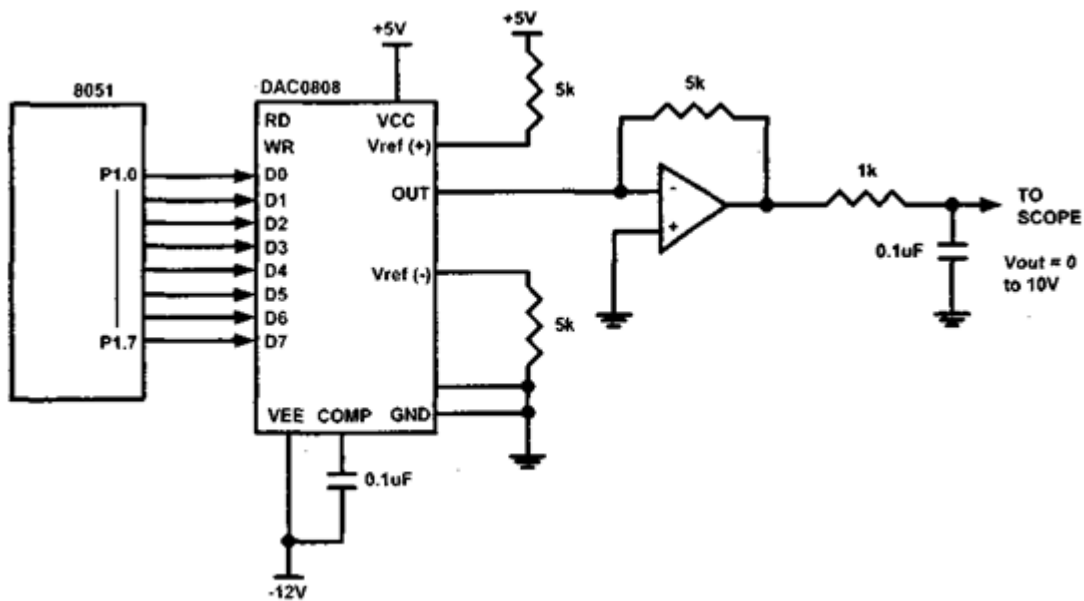


## Assembly Language Program

```
                MOV     A, #DATA*    ; (A) = #Data

    START :     MOV     90H, A       ; (port -1) = (A)

                INC A

                LJMP    START        ;  Repeat
```

--------------------xxx--------------------