

## \* Introduction To Micro Controllers \*

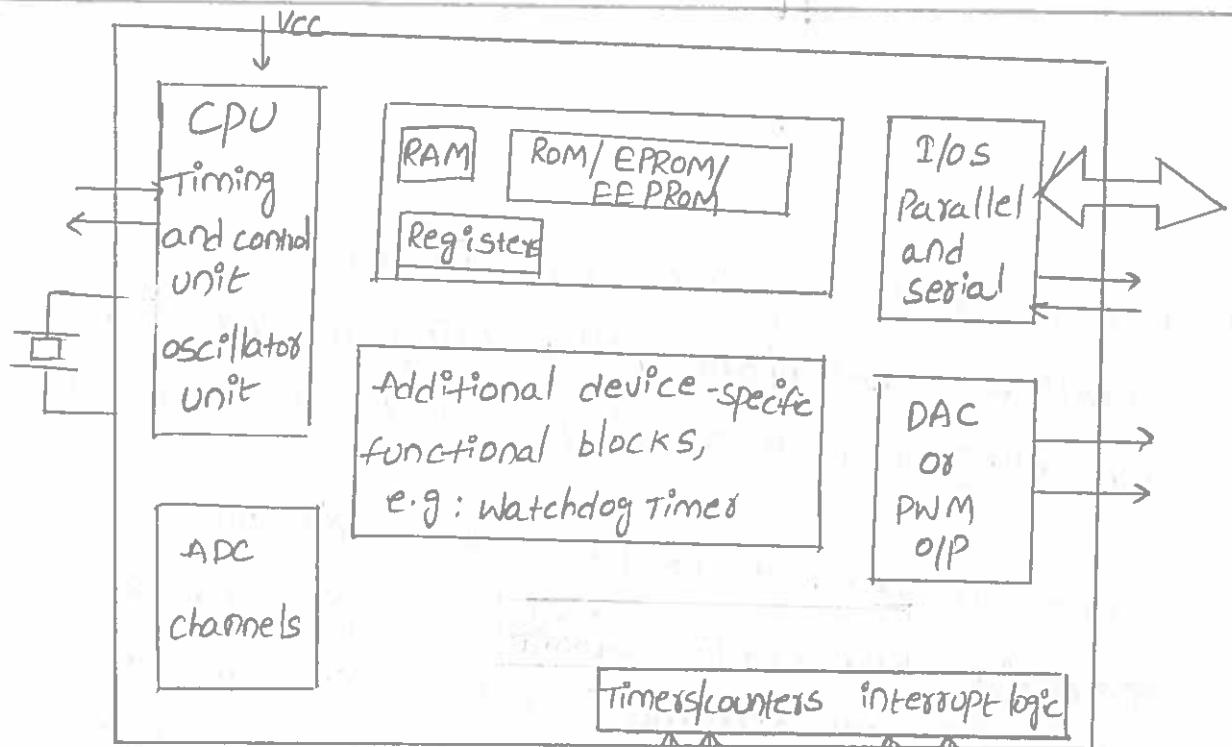
### Introduction:

Today, we see many industrial and domestic products like remote controllers, telephone bill printing machine, automobiles, engines, indicating and measuring instruments and similar products.

Automation is needed to facilitate the processor mechanism for its operation and control. Data storage and processing is an integral part of any automatic control system. The need is to have a device, so called "microcontroller" which allows controlling the timing and sequencing of these machines and processes.

Microcontrollers are single-chip microcomputers, more suited for control and automation of machines and processes. Microcontrollers have CPU, memory, input/output ports (I/O), timers and counters, ADC, DAC and many more functional blocks on chip. Fig(1) shows a general functional block diagram of a micro controller. All the functional blocks from a single integrated circuit (IC), results into a reduced size of control board, low power consumption, more reliability and ease of integration within an application design. The usage of microcontrollers not only reduces the cost of automation but also provides more flexibility. Some of the commonly used microcontrollers are Intel MCS-51, MCS-96, Motorola 68HC12 family, microchip's peripheral interface controller (PIC) family of microcontrollers, 16CXX, 17CXX etc. The designer is little bit relieved from the complex interfacing of external peripherals like ADC/DACs etc. and can concentrate on applications and development aspects. The device can be programmed to make the system intelligent. This is possible because of the data processing and memory capability of intelligent.

Fig(1): Generalized functional block diagram of microcontroller



### History of microcontrollers and microprocessors :

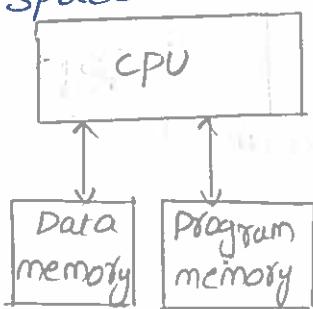
Intel 4004 was the first 4-bit processor which appeared in 1971. The instructions were 8-bits long but the data processed was 4-bit data. It had separate external memories for programs (4K) and data (1K). There were 46 instructions and the clock frequency was 740 kHz. Then, during 1972, Intel 4040 had 14 more instructions with 8K program memory and interrupt capability.

In 1974, Texas Instruments introduced the first microcontroller TMS 1000. TMS 1000 had on-chip RAM, ROM, and I/Os. Intel 8080 was developed in 1976. This could operate on +5V supply and 3MHz frequency. In June 1997, ATMEL 8-bit AVR microcontrollers were introduced which also have reduced instruction set.

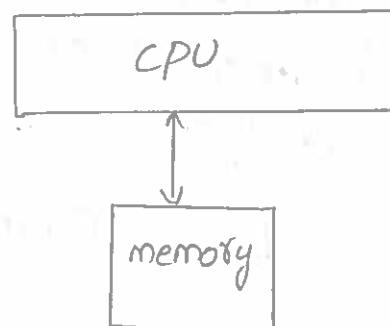
### Harvard And von Neumann Architectures :

There are two major classes of computer architectures, namely, 'Harvard architectures', and 'von Neumann architectures'.

Harvard architecture uses separate memories for program and data with their independent address and data bus. In von neumann architecture, programs and data share the same memory space.



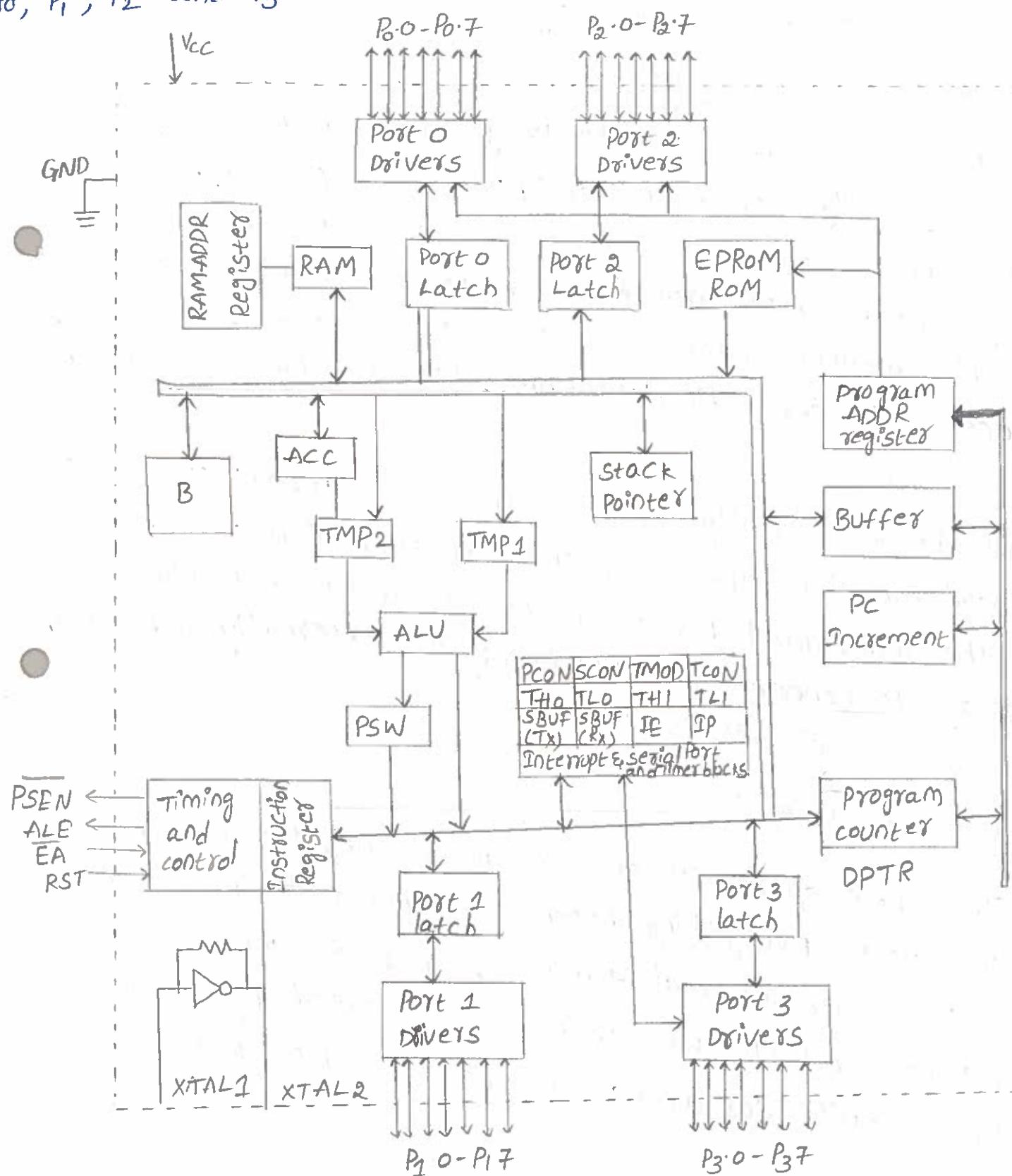
Fig(a): Harvard Architecture



fig(b): von Neumann Architecture  
unit - 2, 2/36

## MCS-51 Architecture:

The block diagram of 8051 microcontroller is shown in fig. The term '8051' refers here to all the MCS-51 family members, unless specifically mentioned; similarly, '8052' is used to refer 8032 and 8052 micro controllers. The various functional blocks of 8051 micro controller are ALU, control and timing unit, RAM/EPRom/ Rom, registers, latches and drivers for ports P0, P1, P2 and P3. Each of blocks is discussed as follows.



## ALU (Arithmetic and logic unit):

ALU of 8051 performs arithmetic and logical operation on 8-bit operands. Accumulator is the register, which gets the o/p of the ALU in most of the arithmetic and logical operations with few exceptions. Each of logical operations involves digital gates AND, OR, NOT Ex-OR operations are possible.

## Boolean processor:

There is a separate Boolean processor integrated within 8051 microcontroller. It has its own instruction set, accumulator and bit addressable RAM. carry flag serves as the accumulator.

## Program and Data memories:

There are two separate program and data memories. The code is typically stored in ROM/EPROM. The program is one of factors that differentiate among the various members of the 8051 family. The program members of the 8051 family. The external off-chip memory space is accessible in most of 8051 members. To access the off-chip data RAM, 16-bit address is used.

## The oscillator:

All the 8051 family members use an external crystal for oscillator function. The frequency of operation can be depending upon the individual device. It must be noted that only it is necessary to connect the quartz crystal externally and all the other oscillator circuit is on-chip.

## Timing and control:

The whole operation of 8051 microcontroller is synchronous with the clock. Everything happens in step with the clock. A part from the internal timings, there are control signals ALE, PSEN and RD, WR that are generated by timing and control unit, for accessing the off-chip devices.

## 8051 parallel I/O port

8051 has four 8-bit I/O ports that are used either as four 8-bit ports or each of the port pins could be addressed individually.

- Each port contains of a latch, an output driver and an input buffer.
- The bit latch is shown as a D-flip flop, which blocks in a value from the internal data bus in response to the write to latch signal from the CPU.
- The Q output from the flip-flop can be read onto the internal data bus in response to a Read latch signal from the CPU.
- Read pin is a different operation from reading a latch.
- The port pin status can be read onto the internal data bus when CPU gives a 'read pin' command.

Port 1, 2 and 3:

- port 1, 2 & 3 are quasi-bi-directional ports and have fixed internal pull-up resistors.
- Fig (i) shows a quasi-bi-directional port. Ports 1, 2 & 3 when configured as input, they are pulled high and source current if externally pulled low.
- For configuring a port pin as input, a '1' must be written to a port latch.
- this turns off the FET and the pin is simply pulled high by the pull-up resistor, the external device may pull it low.
- the pin status can now be read onto the internal data bus.
- On reset, S0S1 port latches have '1's written to them and are configured as inputs.
- one can drive the pin as output any time; however, for input function, the FET must be OFF.
- If the pin is to be used as output, writing a '0' onto a pin requires that the FET should be ON.
- similarly, to write a '1' onto the pin, the FET should be OFF, outputting a '1' because of the pull-up resistor.
- One important thing that must be observed here is that the S0S1 ports can sink more current than it can source.
- port pins can sink around  $0.5\text{mA}$  but can source only tens of  $\mu\text{A}$ .

Port 0:

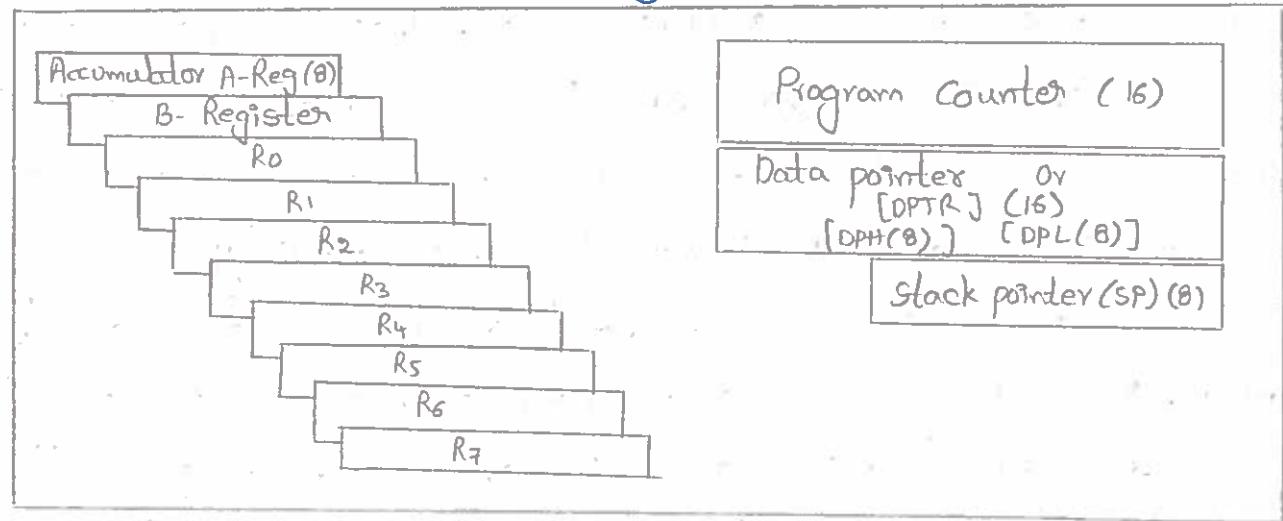
- port 0 does not have internal pull-up resistors. When configured as an input, it floats and is therefore a true bi-directional port.
- port 0 structure can be seen from fig (a). It has two output FETs.

- For normal operation, the upper pull-up FET is OFF, providing 'open drain' output pin & external pull-up resistor is required.
- If a '1' is written to a port 0 latch, either FETs go OFF & the pin floats and can be used as high-impedance input.
- The pull-up FET only operates when there is an access to external memory.
- Port 0 output buffers can drive 8 LS TTL inputs.
- The output drivers of port 0 can be switched to an internal ADDR bus by using an internal control signal while accessing the external memory.
- When used as ADDR & ADDR/DATA bus, port 0 + port 2 pins, respectively, can't be used as general-purpose I/Os.

the 19th century helped to make more and more  
of the land available for agriculture. This  
is the period of the Industrial Revolution.  
The first industrial revolution began in the  
middle of the 18th century and ended in the  
middle of the 19th century. It was caused by  
the invention of new tools and machines.  
The second industrial revolution began in the  
middle of the 20th century and ended in the  
middle of the 21st century. It was caused by  
the invention of new tools and machines.

## Register Organization

There are General-purpose or Working Registers, Stack Pointer, Program Counter and in addition to these CPU registers, there are Special Function Registers (SFRs). These registers are discussed in the following p



CPU Registers

### Accumulator

8-bit accumulator is used by all the arithmetic and logical instructions. Accumulator has a special importance in the sense that, one of the operands is stored in it before the execution of an instruction and it also stores the result after the execution of an instruction. It is referred to as register 'A'. Access to accumulator is faster than access to main memory. Accumulator has direct path to ALU and can immediately store the intermediate result of operation.

### B- Register

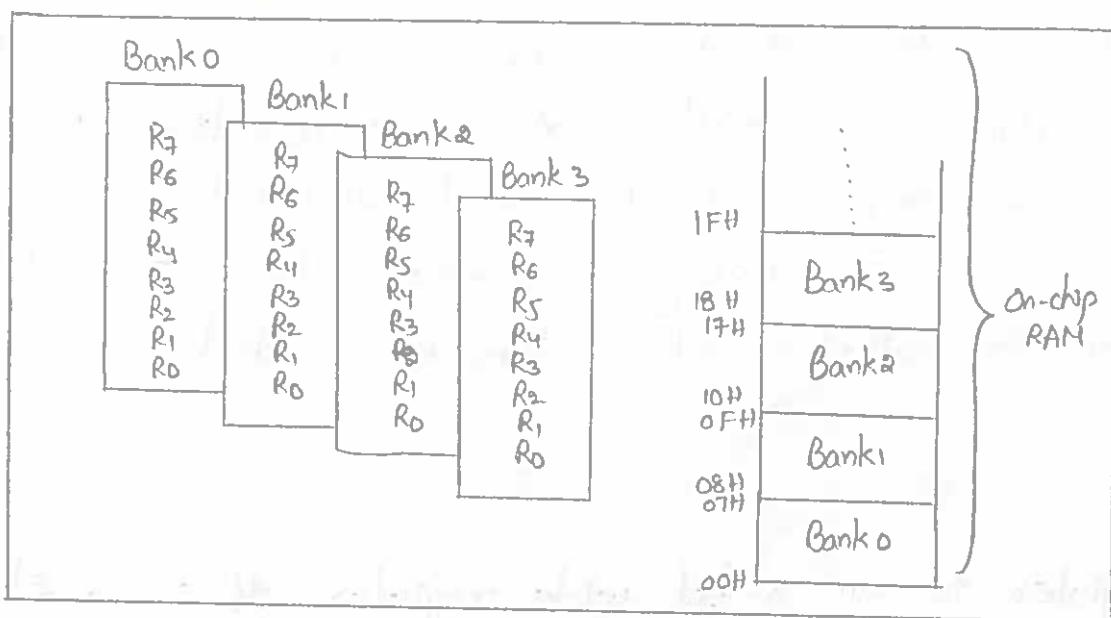
B-register is an 8-bit wide register. It is available when it is not being used by multiplication and division operations. While multiplying, it holds one of the 8-bit operand and after the execution of the multiplication instruction it stores the higher byte of the result. While dividing, it holds an 8-bit divisor and after the execution of division instruction, the remainder is stored in B-register.

## Registers R<sub>0</sub> through R<sub>7</sub>

These eight registers are used as scratch pad registers.

There are 4 register banks each containing R<sub>0</sub> through R<sub>7</sub> registers. Each of these registers is 8-bit wide. At a time only one bank can be selected by appropriate setting of bits in the program status word. These register banks are located in the on-chip RAM. Each reg bank can allow 32 registers to be used while writing programs. Certain instructions can access these registers in RAM directly.

Power-up-reset causes bank 0 to be selected by default. Now, if one were writing a byte in R<sub>4</sub>, it would be stored at RAM location 04H. If in an other case, the programmer is selecting bank 1 and writing a byte in R<sub>4</sub> it would store the byte at RAM location 0CH. The advantage of this type of access to this registers is that for programmer it becomes simpler to refer those by register names R<sub>0</sub>, R<sub>1</sub> ... etc.



Four Register Banks and their locations in the on-chip RAM

## Stack pointer

Stack pointer is a 8-bit wide. It is incremented during push or call operations and is decremented during pop or return operation. It may be initialized anywhere in the available on-chip data RAM. After the RESET operation, the stack pointer is initialized to 07H, causing stack begin at 08H.

## Program Counter (pc) :

Instruction opcode bytes are fetched from the program memory locations addressed by the program counter. The program counter is 16-bit wide, it can address 64k code bytes. PC always points to the instruction to be fetched and is automatically incremented after fetching the instruction. PC is affected by call and jump instructions. PC register has no (internal) on-chip RAM address.

## Data Pointer (DPTR) :

DPTR is a 16-bit register consisting of two bytes. The higher byte is referred to as DPH, lower byte as DPL. The data pointer is used for addressing the off-chip data and code with MOVX and MOVC commands, with 16-bit DPTR, a maximum of 64k of off-chip data memory and a maximum of 64k of off-chip program memory can be addressed. There is an instruction "INC DPTR" for incrementing 16-bit contents of DPTR. It is also possible to load the DPTR with a 16-bit immediate data using the MOV instruction.

## MEMORY ORGANIZATION

Memory Organization of 8051 microcontroller is

1. Program Memory
2. Data Memory.

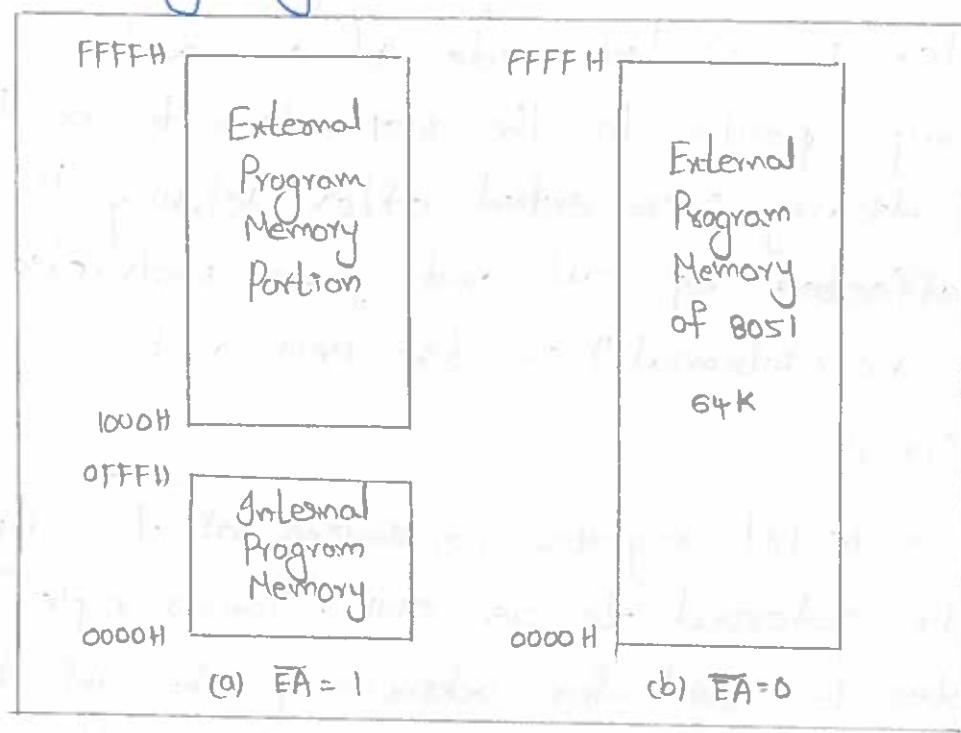
### Program Memory

The 64k program memory space of 8051 is divided into internal and external memory. Program memory is accessed through EA pin.

→ If EA pin is high, then internal program memory is accessed till  $0FFFH$  memory location and external memory is accessed from  $1000H$  to  $FFFFH$  memory location.

→ If EA pin is low, only external program memory is accessed from 0000H to FFFFH memory locations (64k).

Program memory organization is shown as.



### Program Memory of 8051

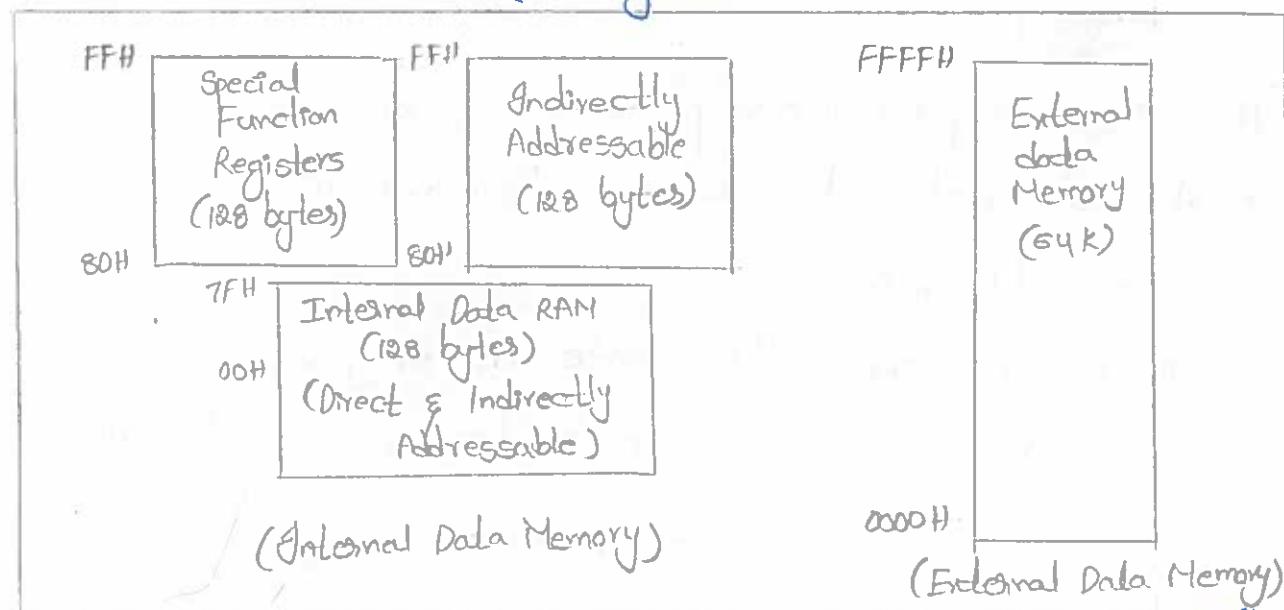
Program Memory of 8051 and program Execution

Status of EA pin	Program execution from 0000H through 0FFFH	Program execution from 1000H through 0FFFH.
High(1)	Internal program memory	External program memory
Low(0)	External program memory	External program memory.

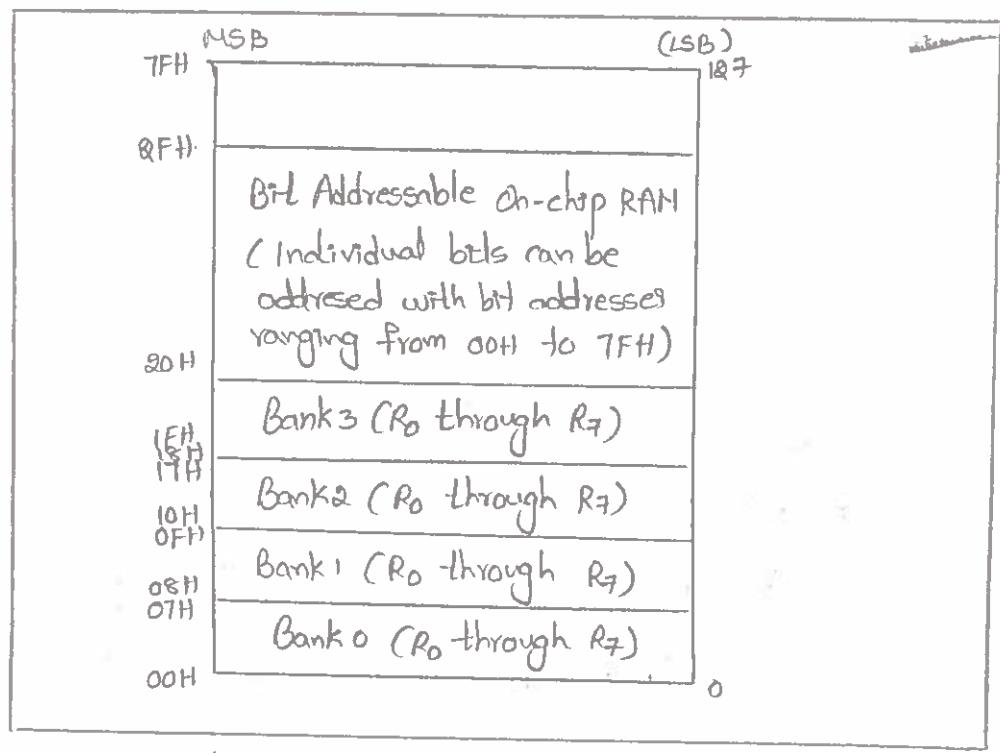
### Data Memory

Data Memory Organization is as shown below. The instruction MOVX is used to access external data memory of size 64k. Here internal data memory of 256 bytes has 2 partitions.

- (i) 00H - FFH for internal data RAM (128 bytes)
- (ii) 80H - FFH for SFRs (128 bytes)



The internal data memory of 8051 is 256 bytes, which is divided into two parts again. The lower 128 bytes (00H through 7FH) called as the internal data RAM and the upper 128 bytes (80H through FFH) consists of SFRs. In case of 8032/52, the upper 128 bytes of internal data memory are also addressable having same address space, they are different and accessed through different addressing modes. The lower 128 bytes of on-chip RAM



On-chip RAM (Lower 128 bytes)

P1.0	1	40	Vcc
P1.1	2	39	P0.0/AD0
P1.2	3	38	P0.1/AD1
P1.3	4	37	P0.2/AD2
P1.4	5	36	P0.3/AD3
P1.5	6	35	P0.4/AD4
P1.6	7	34	P0.5/AD5
P1.7	8	33	P0.6/AD6
RST	9	8051	
RXD/P3.0	10	32	P0.7/AD7
TXD/P3.1	11	31	EA
INT0/P3.2	12	30	ALE
INT1/P3.3	13	29	PSEN
T0/P3.4	14	28	P2.7/A15
T1/P3.5	15	27	P2.6/A14
WR/P3.6	16	26	P2.5/A13
RD/P3.7	17	25	P2.4/A12
XTAL2	18	24	P2.3/A11
XTAL1	19	23	P2.2/A10
VSS	20	22	P2.1/A9
		21	P2.0/A8

Pin Configuration of 8051 Microcontroller

### 8051 PIN DESCRIPTION

Pin diagram of 8051 is shown above. One can easily see the pins corresponding to four ports P0, P1, P2 and P3 pins are having alternate functions. There are pins for power supply Vcc (pin 40) and ground (pin 20). Pins 18 and 19 for crystal connections. Finally, pins PSEN (pin 29), ALE (pin 30), EA (pin 31) are Program Store Enable, Address Latch Enable and External Access, respectively. A brief discussion of these pins is given on next page.

### XTAL2 (Pin 18)

Output of inverting amplifier that forms a part of the oscillator and input to the internal clock generator. In case of external clock, it must be connected to XTAL2.

### Port 3 (Pins 10-17)

Same function as port 2 but there are other functions multiplexed with port 3 pins. Related to external interrupts, serial port, timer/counter and read/write control signals.

P3.0	RXD serial input
P3.1	TXD serial output
P3.2	INT0 external interrupt
P3.3	INT1 external interrupt
P3.4	To timer/counter0 external input
P3.5	T1 timer/counter1 external input
P3.6	WR external data memory write strobe
P3.7	RD external data memory read strobe

### Pin 9 (RST)

For resetting the device, the RST pin of 8051 is made high for two machine cycles, while the oscillator is running. A power-on-reset ckt, a pull-down resistor of 8.2k from the RST pin to Vss and a capacitor of 10nF from the RST pin to Vcc from the reset ckt. These component values are sufficient to provide a delay, so as to make the RST line high for 24 oscillator periods.

### ALE (Pin 30) [Address Latch Enable]

ALE o/p is used for latching the low address byte during external memory access. ALE is activated periodically with a constant rate of 1/6 the oscillator freq. However, during the external data memory access, one ALE pulse is skipped.

### PSEN (Pin 29) [Program store enable]

It is o/p control sig, activated every six oscillator periods, while fetching the external program memory. It is the read strobe to external program memory. During the internal program execution, it becomes high.

### EA (Pin 31) [External access]

If it is high, executes instruction from the internal program memory till address OFFFH ; beyond this addressed fetched from external program memory. If EA=0 instructions fetched from the external memory. During normal operation, this pin should not be floated.

## XTAL1 (Pin 19)

XTAL1 is the input to the inverting amplifier that forms part of the oscillator circuit. In case of external clock, this pin must be connected to ground.

## V<sub>CC</sub> (Pin 40)

V<sub>CC</sub> pin is connected to +5V power supply. Rated power supply current for 8031 / 8051 is 125 mA. In case of 8751, the maximum supply current is 250 mA. For 8031 / 8051, the maximum power dissipation rating is 1W.

## V<sub>SS</sub> (Pin 20)

V<sub>SS</sub> is the circuit ground. All the voltages are specified with respect to this pin. For example, voltage on any pin with respect to V<sub>SS</sub> should be within -0.5 to +7V range.

## Port 0 (Pins 32-39)

Port 0 is an 8-bit true bi-directional open drain I/O. Low order address and data bus is also multiplexed with Port 0. Port 0 is open drain and must be pulled high externally through a pull-up resistor.

## Port 1 (Pins 1-8)

Port 1 is an 8-bit quasi-bi-directional I/O. The term quasi-bi-directional port is due to the fact that port 1 pins are internally pulled high with fixed pull-up resistors. One has to configure it either as I/P or O/P. Writing a '1' to the port latch causes it to act as I/P. When configured as input, the port pin is pulled high and will source current if it is made low externally.

## Port 2 (Pins 21-28)

Port 2 is also an 8-bit quasi-bi-directional I/O port. Port pins are pulled high internally. It is multiplexed with the higher order address bus.

## 8051 Addressing Modes:-

Addressing modes define the way in which the Operands are accessed by the instruction.

There are five addressing modes supported by MCS-51.

- Register addressing mode.
- Direct addressing mode
- Register Indirect Addressing mode.
- Immediate Addressing mode.
- Base Register plus Index Register Addressing mode.

### Register Addressing mode:-

In register addressing mode, registers R<sub>0</sub> through R<sub>7</sub> from the selected register bank, accumulator, B-register, carry bit and DPTR are used. An MCS-51 instruction using this addressing mode refers the registers R<sub>0</sub> through R<sub>7</sub> in the opcode itself. The least significant bits of the opcode indicate which register is to be used. This is shown in below.

Opcode part (5 bits)	Register (3 bits)
MSB   1   1   0   1   1   1   LSb	

Ex:- Opcode of MOV A, R<sub>7</sub> instruction.

(lower significant three bits refer to the Register used).

### Direct Addressing mode:-

In direct addressing mode, the direct address of the operand is specified in the instruction itself. Direct addressing mode uses the lower 128 bytes of internal RAM and the special function registers.

Ex 1:- MOV A, Direct

[The direct address of the source operand is used]

Ex:-2 MOV A #54H

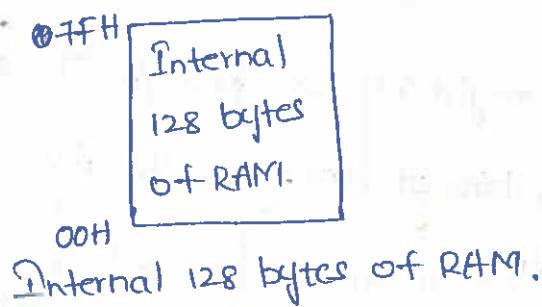
MOV A, 54H will transfer the contents of the on-chip memory location, whose address is 54H, to accumulator.

Ex:-3: MOV A, SBUF .

Use this instruction, to read the content of the SFR SBUF into the accumulator. SFR SBUF has the direct address 99H, which lies in the upper 128 bytes of the on-chip RAM.

Register Indirect Addressing mode:-

Register Indirect addressing mode uses any one of the registers R0 or R1, from the selected register bank, as a pointer to the locations in the 256 bytes of data memory block. It may point in the lower 128 bytes of the internal RAM.



Immediate Addressing mode:-

Immediate Addressing mode allows using immediate data as a part of the instruction.

Ex:- MOV A, #45H .

will store the constant or immediate data 45H in the accumulator. The symbol # preceding the constant, this indicates the immediate data type.

Base Register plus Index Register - Indirect Addressing mode :-

This mode allows a byte to be accessed from the Program memory, whose address is calculated as the sum of a base register (DPTR or PC) and index register, accumulator.

This is shown in below.

Ex:- MOVC A, @A + DPTR.

will fetch a byte from the Program memory, whose address is calculated by adding the original 8-bit Unsigned contents of the accumulators and the 16-bit contents of the DPTR. If DPTR contains OFFFOH and the accumulator contains 05H, then the byte stored at OFFF5H will be copied into the accumulator. This method facilitates the look-up table access.

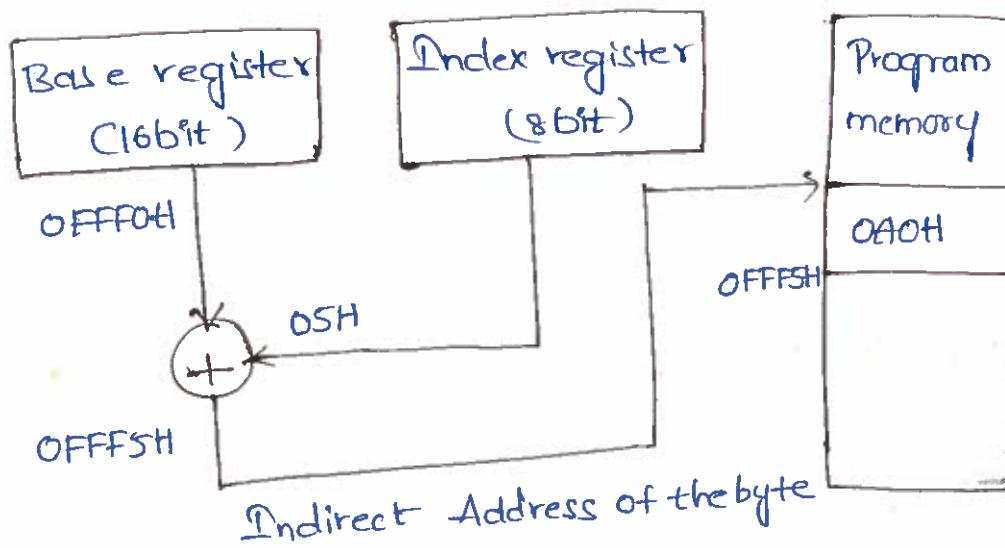


Fig: Base Register plus Index Register Indirect Addressing

1. What is the difference between a primary and secondary market?  
 Primary market - new shares issued by company  
 Secondary market - existing shares traded on exchanges  
  
 2. What is the difference between a public and private company?  
 Public company - open to the general public  
 Private company - restricted to specific individuals or institutions  
  
 3. What is the difference between a listed and unlisted company?  
 Listed company - traded on stock exchange  
 Unlisted company - not traded on stock exchange  
  
 4. What is the difference between a quoted and unquoted company?  
 Quoted company - traded on stock exchange  
 Unquoted company - not traded on stock exchange  
  
 5. What is the difference between a listed and unlisted company?  
 Listed company - traded on stock exchange  
 Unlisted company - not traded on stock exchange  
  
 6. What is the difference between a quoted and unquoted company?  
 Quoted company - traded on stock exchange  
 Unquoted company - not traded on stock exchange  
  
 7. What is the difference between a listed and unlisted company?  
 Listed company - traded on stock exchange  
 Unlisted company - not traded on stock exchange  
  
 8. What is the difference between a quoted and unquoted company?  
 Quoted company - traded on stock exchange  
 Unquoted company - not traded on stock exchange  
  
 9. What is the difference between a listed and unlisted company?  
 Listed company - traded on stock exchange  
 Unquoted company - not traded on stock exchange  
  
 10. What is the difference between a quoted and unquoted company?  
 Quoted company - traded on stock exchange  
 Unquoted company - not traded on stock exchange

## MCS-51 Instruction Set :-

The instruction set of MCS-51 consists of data transfer instructions, arithmetic instructions, logical instructions, Boolean variable manipulation instructions and control transfer instructions. There are 111 instructions supported by MCS-51.

The instructions can further be classified as single-byte, two byte and three byte instructions.

The instruction format consists of a mnemonic followed by destination and source fields,

mnemonic	destination operand	source operand
----------	---------------------	----------------

### i) Data Transfer Instructions :-

Data Transfer instructions of 8051 are mov, movx, movc, push, pop, and exchange XCHG, XCH instructions. Data transfer instructions do not affect any of the PSW flags. However, if there is a mov or pop directly to PSW, it can affect the PSW.

### ii) Arithmetic Instructions :-

8-bit arithmetic unsigned operations are supported by MCS-51. However, it is possible to carry out both signed and unsigned addition and subtraction by using the OV flag.

BCD arithmetic is also packed BCD. There are unsigned multiplication and division operations directly supported by the instructions.

### Logical Operations:

Bitwise logical AND, OR, Exclusive-OR operations are possible in MCS-51. These instructions accept two 8-bit operands and the result is stored at the destination, no flags affected by ANL, OR and XOR instructions. There

are single-operand instructions like CLR, SETB and CPL; rotate instructions RR, RRC, RL, RLC; and swap instructions SWAP. It is important to note that the CPL instruction complements the accumulator without affecting any of the flags. Rotate instructions RL & RR do not affect any flag. However, RLC and RRC modify CY flag.

RLC instructions moves bit-7 of the accumulator into CY position. Similarly, RRC instruction moves bit-0 of the accumulator into CY flag. SWAP A instruction simply interchanges the lower and higher nibbles of the accumulator and no flags are affected.

#### 4 Boolean Variable Manipulation Instructions :-

Operands for Boolean variable manipulation are defined in these instructions. In case of two-operand instruction, there is a destination bit and source bit. For example, MOV instruction has two single-bit operands, OR, AND operations with two single-bit operands are possible. Single-operand instructions CLR, SETB, CPL modify the bit location specified in the instruction.

# \* \* 8051 REAL TIME CONTROL \*

## Interrupt structure of 8051:-

An interrupt is an external or internal signal to processor architecture that temporarily changes the flow of execution of the main program and may execute another program before continuing with the main program.

Many big microprocessor architectures and systems support several interrupts. For example, a pentium based personal computer system supports more than hundred interrupts. These interrupts are either directly supported by the processor architectures or an external peripheral like programmable interrupt controller. Microprocessors are mainly used in small dedicated systems, so in general less no. of interrupts are required.

If a specific application demands large no. of external hardware interrupts, the programmable interrupt controllers can be interfaced and even be cascaded to implement large no. of external hardware interrupts. In microprocessor systems, the interrupts may be used for implementing the following applications :-

- 1, Data communication and I/O
- 2, Task switching, Multitasking, Multiprogramming
- 3, Timing & Real-time clock, counting.
- 4, Interfacing peripherals and I/O devices
- 5, Handling of system faults and errors.
- 6, Interlinking of external events with system operation and program execution.

In the interrupt structure of 8051, when an interrupt is occurred internally (b) externally the processor will be activated and reacts to the interrupt caused and finds the location of interrupt with the help of processor. That location is called as vector location and interrupt is called as vector interrupt.

### 8051 Interrupts:-

8051 architecture supports total 5 interrupts. out of these five; two interrupts namely INT0 and INT1 are external hardware active low interrupt. they can be enabled and programmed using the least significant four bits of TCON register and the interrupt enable and priority registers. The remaining three interrupts are Timer 0 overflow, Timer 1 overflow and serial transmission or reception interrupt.

The serial transmission and reception cause only internal interrupt as signals generated by the serial transmission and reception units are ORed internally. The interrupts available in 8051 are presented below in the order of their decreasing default priority and the respective addresses of vector.

### 8051 Interrupts and vector address

S.NO	Interrupt	Default priority	Vector address
1	INT0	Highest	0003H
2	Timer 0	:	000BH
3	INT1	:	0013H
4	Timer 1	:	001BH
5	Serial [trans/Rec]	: [Lowest]	0023H [in code memory]

The structure of this bit addressable register is presented below

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
EA	R	R	RES	ET <sub>0</sub>	EX <sub>1</sub>	ET <sub>1</sub>	EX <sub>0</sub>

Bit definitions of IE register

EA - Enable all ; R = Reserved ; ET<sub>0</sub> & ET<sub>1</sub> = Enable Timer 0 & Timer 1 ; EX<sub>1</sub> & EX<sub>0</sub> = External interrupts INT<sub>0</sub>, INT<sub>1</sub>.

Thus using the EA bit all the interrupts can be enable or disabled. using the individual respective bit, the respective interrupt can be enabled or disabled.

### Interrupt priority :-

When more than one interrupt are enabled user can program the interrupt based on priority levels by setting or clearing the bits. this process is called Interrupt priority [IP].

Each interrupt of 8051 can have two levels of priority: Level 0 and Level 1. Level 1 is considered as a higher priority level compared to Level 0. the internal architecture of 8051 scans or polls the interrupt flags in the order given below table.

### interrupt priority register:-

IP <sub>5</sub>	P <sub>T2</sub>	Timer 2 priority in case of 8039/8052 only
IP <sub>4</sub>	P <sub>S</sub>	serial interrupt priorities
IP <sub>3</sub>	P <sub>T1</sub>	Timer 1 interrupt
IP <sub>2</sub>	P <sub>X1</sub>	External interrupt
IP <sub>1</sub>	P <sub>T0</sub>	Timer 0 interrupt
IP <sub>0</sub>	P <sub>X0</sub>	External interrupt 0

However this default order of priority can be changed by appropriately programming the IP register. For example the INT0 has the highest priority while the serial interrupt (S2), has the lowest priority when they are working at equal priority level [level 0 & level 1]. But if the S2 interrupt is programmed for level 1 while the INT0 is programmed to work at level 0, the S2 interrupt will have higher priority than the INT0 interrupt. At equal level of priority, the interrupts follows the default priority order that is the same as order or sequence of polling. The IP register is presented in below fig.

D7	D6	D5	D4	D3	D2	D1	D0
R	R	R	PS	PT1	PX1	PT0	PX0

Bit definitions of IP register.

R → Reserved ; PS → priority level of serial interrupt  
 PT1 → Priority level of Timer 1 interrupt ; PT0 → Priority level of Timer 0 ; PX0 → priority level of INT0

PX1 → Priority level of INT1.

### TIMER/ COUNTER CONTROL REGISTER [TCON]

Timer and counters: On the chip timer and counter facility has improved the capabilities of microcontrollers for implementing the real time applications. The applications are:-

- ⇒ 1, pulse counting
- ⇒ 2, frequency measurement.
- ⇒ 3, pulse width measurement
- ⇒ 4, Baud rate generation etc

## TCON Register :-

This bit addressable register houses 8-bits. Most significant four bits either control the operation of the respective timers/counters or reflect the overflow condition. The least significant four bits are used to program the operating modes of the External Interrupts or to store the received interrupt requests. The bit definitions of this register are presented in below fig.

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
TF <sub>1</sub>	TR <sub>1</sub>	TF <sub>0</sub>	TR <sub>0</sub>	IE <sub>1</sub>	IE <sub>1</sub>	IE <sub>0</sub>	IE <sub>0</sub>
<u>Bit definition of TCON Register.</u>							
Timer/counter		Timer/counter		INT <sub>1</sub>		INT <sub>0</sub>	
1	0						

TF<sub>0</sub>/TF<sub>1</sub> bits are called as overflow bits and set by the external hardware when the respective timer/counters overflows.

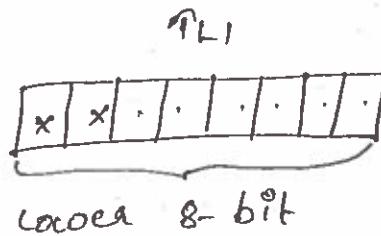
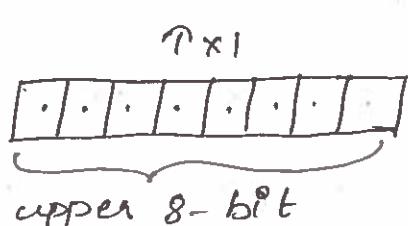
## Modes of Operation in 8051 :-

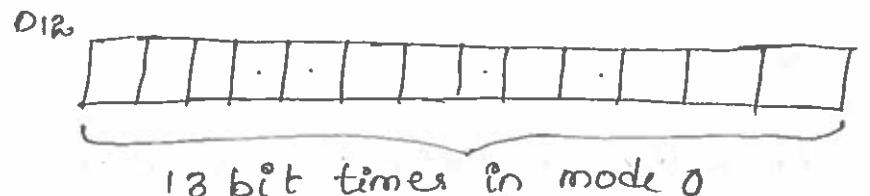
8051 timers or counters can operate in four modes of operations.

- \* Mode 0
- \* Mode 1
- \* Mode 2
- \* Mode 3.

Mode 0 :- Both the timers/counters of 8051 can act in mode 0, a timer/counter acts as a 13 bit wide counter.

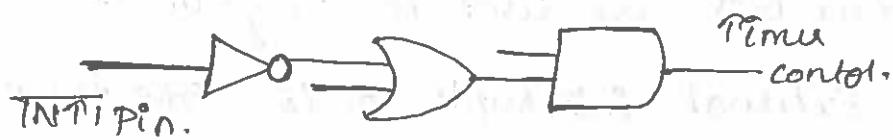
This is same for timer 0 and timer 1 that is TH<sub>0</sub>, TH<sub>1</sub> and TL<sub>0</sub>, TL<sub>1</sub>.



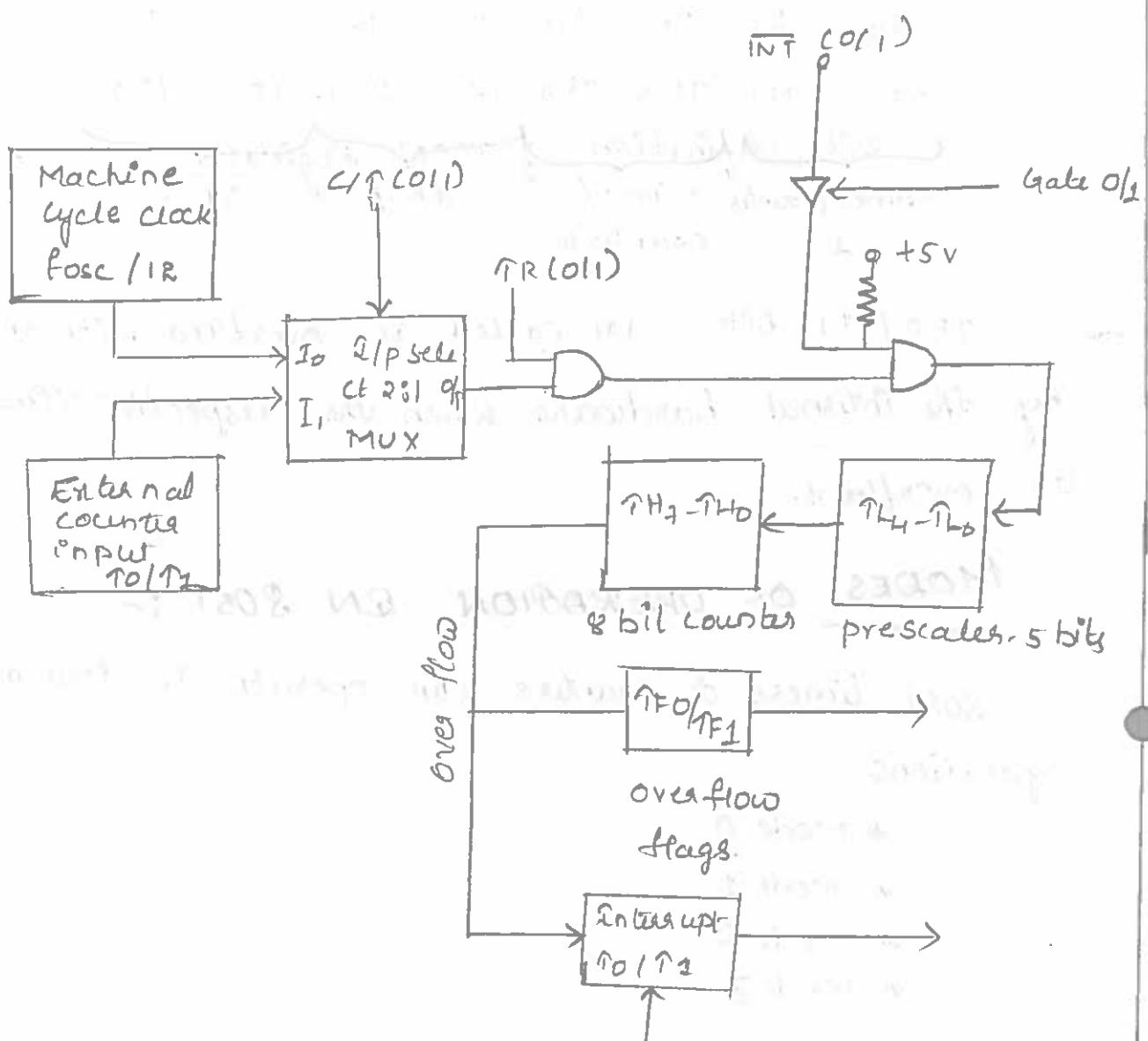


13 bit timer in mode 0

Pimer control logic in mode 0 and this is same for mode 1 also.

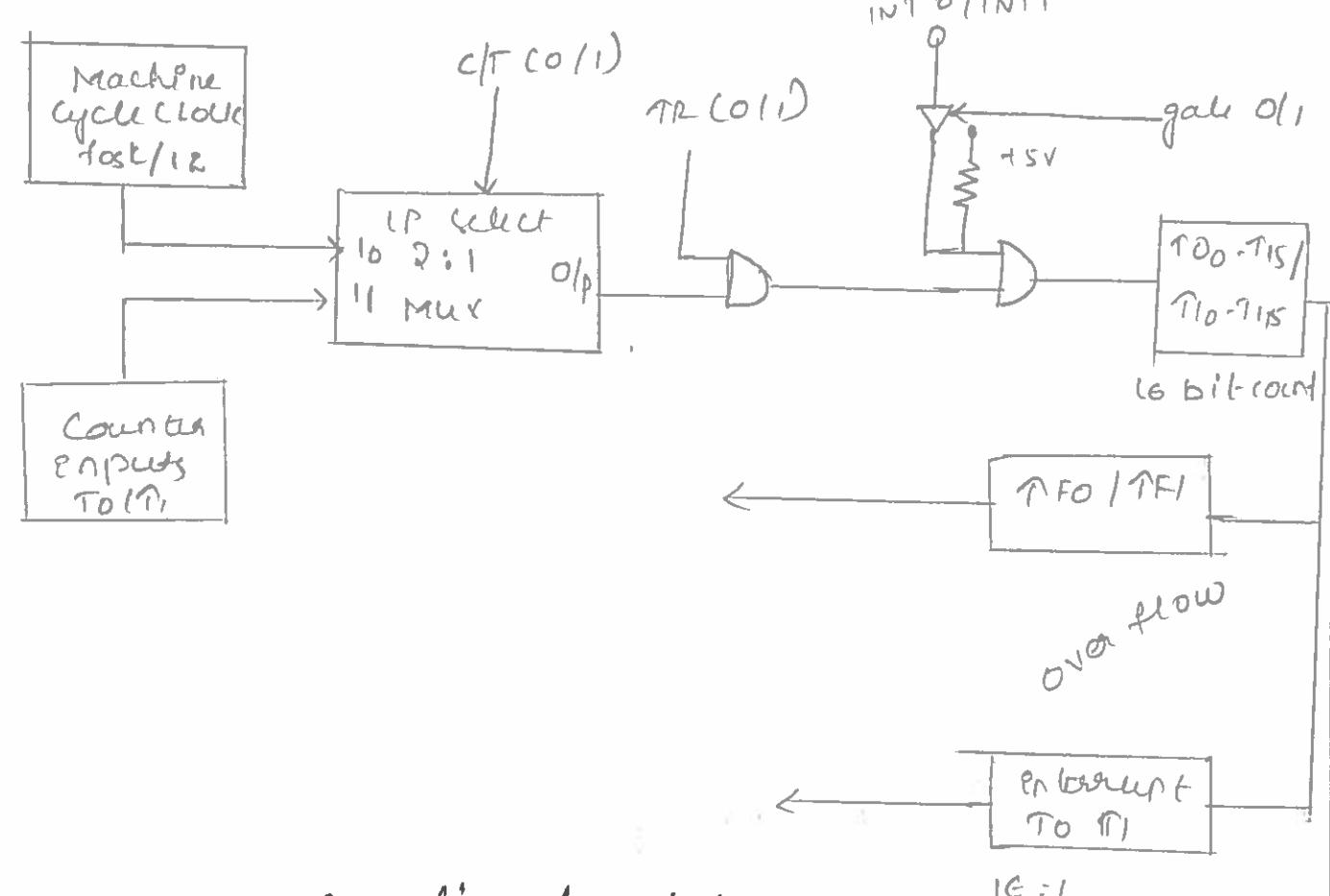


"The Band rate is reciprocal of the time send in one bit."



### Operation of Mode 0.

**Mode 1:** In this mode of operation, both the timers/ counters work as 16 bit timer/ counter. The remaining mode operation is exactly similar to that of mode 0. The timer/ counter mode, Gate bit significance  $\text{INT}0/\text{INT}1$  functions as gates of  $T_0$  and  $T_1$  are exactly similar to mode 0.

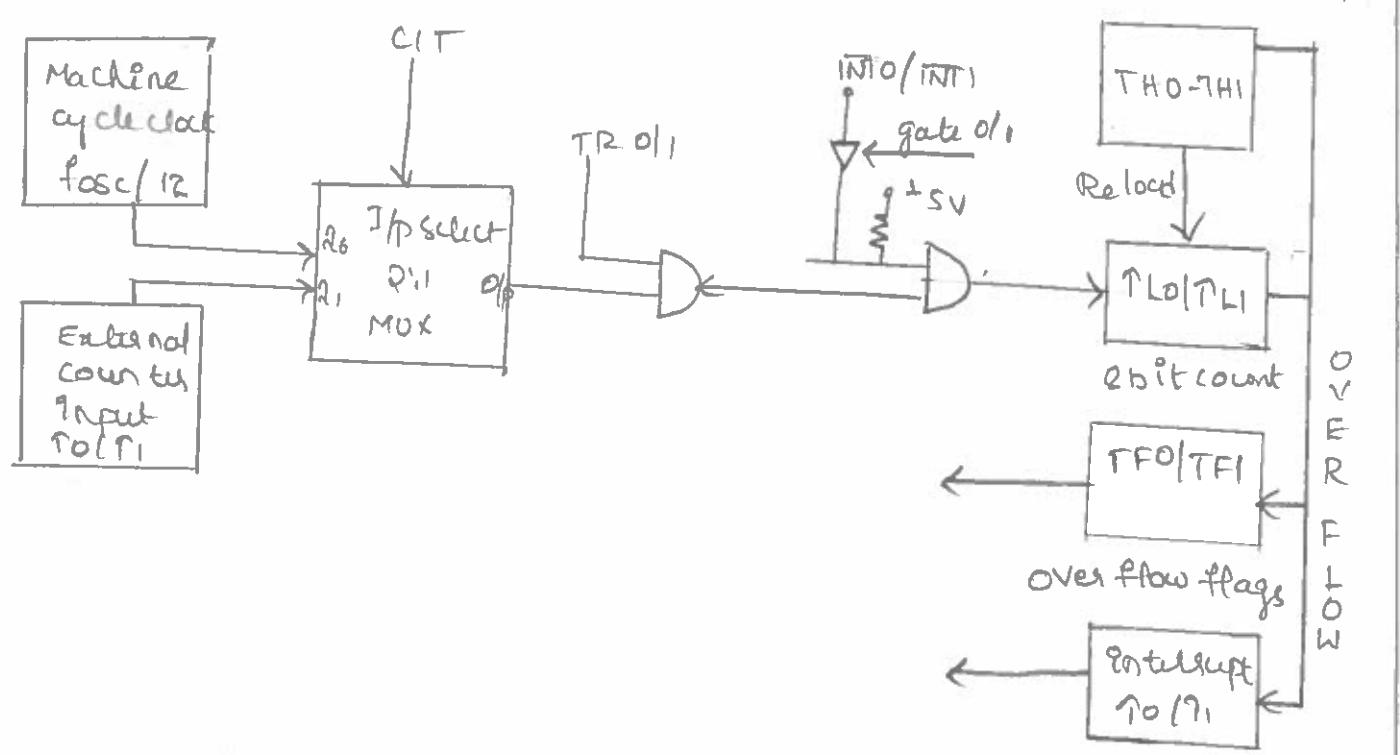


### Operation of Mode 2.

**Mode 2:-** This mode is also called 8-bit autoreload mode, the 16 bit timer registers  $T_0$  and  $T_1$  are considered in terms of their higher bytes  $T_{H0}/T_{H1}$  and lower bytes  $T_{L0}/T_{L1}$ . The higher bytes are used to store 8-bit reload values.

The timers/ counters can be in different modes independent of each other. When the generated interrupts after each overflow are vectored the TFO/TFI are automatically cleared. If the interrupts are not enabled, the TFO/TFI need to be cleared using program instructions. The timer 1 is mode 2 with interrupt disabled is used by default for baud rate generation for serial communication applications.

The lower bytes are used to store the actual 8-bit count that progresses and that starts from the reload values.



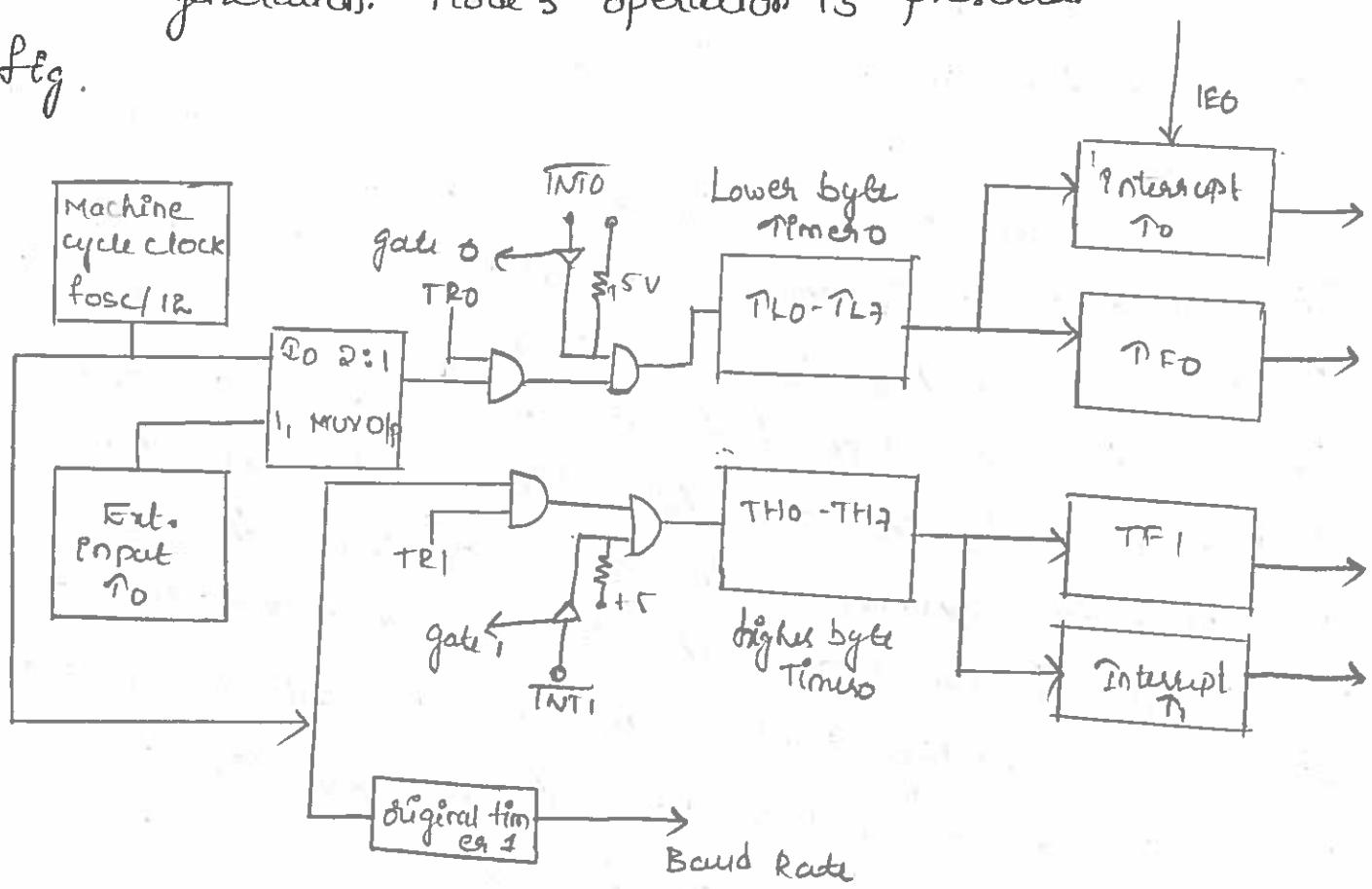
operation of mode 2.

Mode 3:-

In mode 3, the timer/counter 0 is used as two independent 8 bit counters while the timer/counter 1 just holds its count and is non functional. The lower and higher bytes of T0 i.e., T<sub>LO</sub> and T<sub>HO</sub> are parallelly driven by the internal machine cycle clock or external clock input pin T0 for T<sub>HO</sub> respectively depending upon whether T<sub>LO</sub> is used as a timer or a counter. T<sub>HO</sub> is only used as a timer.

The upper 8 bit timer T<sub>HO</sub> can work as only timer and it can't work as a counter i.e., it cannot count pulses on T1. While the timer/counter [T<sub>LO</sub>] is programmed in mode 3, the original timer/counter, that is non functional in mode 3, can be programmed in mode 0, mode 1 & mode 2.

But in this case the original timer 1 will not be able to indicate its overflow as all its mechanisms are used by TFO. However the original timer/counter 1 can be used for generating serial communication baud rate by programming it in mode 2 as this application does not require any overflow flag or interrupt generation. Thus in mode 3, The timer/counter 0 offers one 8 bit timer / counter TFO, one additional 8 bit only timer TFO and original Timer/ counter for application like baud rate generation. Mode 3 operation is presented in below fig.



operation of Mode 3.

## \* Serial Communication :-

Serial data transmission is very commonly used for digital data communications. Its main advantage is that the number of wires needed is reduced as compared to that in parallel communication. 8051 supports a full duplex serial port. Full duplex means, it can transmit and receive a byte simultaneously. 8051 supports two TXD and RXD pins for transmission and reception of serial data respectively. The 8051 serial communication is supported by RS232 standard. The term "RS" stands for Recommended Standard. Communication b/w two Microcontrollers and multiprocessors communication is also possible. In serial transmission, baud rate is one important factor. The baud rate is the reciprocal of the time to send 1 bit. Baud rate need not be equal to number of bits per second. This is because, each byte is preceded by a start bit and followed by one stop bit. The start and stop bits are used to synchronize the serial receiver, the data byte is always transmitted with least-significant-bit first. For error checking purpose, it is possible to include a parity bit as well, just prior to the stop bit. Thus, the parity bit along with its status be the same at the transmitter and receiver ends.

The basic mechanism of serial transmission is that a data byte in parallel form is converted into serial data stream. Along with some more bits like start,

Stop and parity bits, a serial data frame is sent over a line. There are four modes of serial data transmission in 8051. In each of these modes, it is important to decide the baud rate, the way in which serial data frame is sent and any other information, etc. what is common in all these modes is the use of the SFR called "SBUF", in all these modes is the use of the SFR called "SBUF", for transmission as well as reception. The data to be transmitted must be transferred to SBUF. The same SBUF address is defined; however, physically there are two entities, but the user need not bother about this in the real sense, because 8051 takes care for it. Thus, the programmer feels the same SBUF for transmission and reception as well. one more SFR that

SCON.7	SCON.6	SCON.5	SCON.4	SCON.3	SCON.2	SCON.1	SCON.0
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Bit address	Scan bit	Description
9FH	SM0	Serial communication mode.
9EH	SM1	
9DH	SM2	In modes 2 and 3, if set, this will enable multi-processor communication.
9CH	REN	(Receive enable) Enables serial reception.
9BH	TB8	This is the 9th data bit that is transmitted in modes 2 & 3.
9AH	RB8	9th data bit that is received in modes 2 & 3. It is not used in mode 0. In mode 1, if SM2 = 0, then RB8 is the stop bit that is received.
99H	TI	Transmit interrupt flag, set by hardware, must be cleared by software.
98H	RI	Receive interrupt flag, set by hardware, must be cleared by software.

SM0	SM1	Mode	Description	Baud rate
0	0	0	8bit shift register mode	fosc/12
0	1	1	8bit UART	variable (set by timer 1)
1	0	2	8bit UART	fosc/16000 fosc/32
1	1	3	8bit UART	variable (set by timer 1)

fig : serial control register (SCON)



LSB = most significant bit

LSB : least significant bit

Fig: A frame of 8 bits transmitted serially in mode 0, with LSB first.

Controls the serial communication operation is the Serial Control register SCON. Details of SCON are shown in Fig 6.10. Bits SM0 and SM1 in SCON define serial port mode. Bit SM2 enables the multiprocessor communication in modes 2 and 3. Transmission is initiated by the execution of any instruction that uses SBUF as the destination.

\* Serial communication modes :-

These are four modes in which 8051 serial port

can be configured.

\* Mode 0 :-

This is also called as shift register mode. Only RXD is the pin through which data enters our exits. TXD pin outputs the shift clock only. Eight data bits are transmitted or received. The baud rate is fixed and is totally determined by the system clock frequency. If fosc is the clock frequency, then  $fosc/12$  will be the baud rate.

To see exactly how the operation of serial data transfer takes place or, see programming example # 6.11.

; Programming Example # 6.11

; serial transmission mode 0

ORG 000H

; Program starts at 000H

MOV SCON, #0000 0000B ; mode 0

; Now write the data byte to be transmitted in SBUF

MOV SBUF, #44H ; ; Transmit 0100 0100

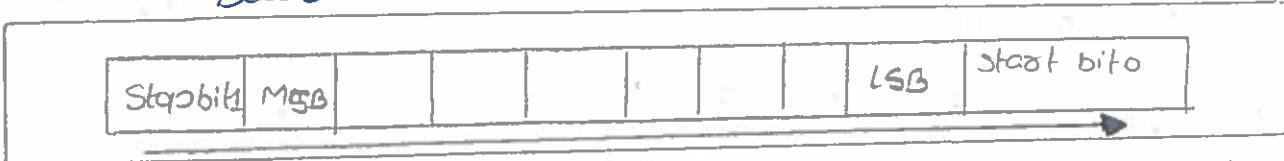
- ; After transmission, TI flag in SCON will be set by hardware, this can be
- ; tested for assessing the transmission operation  
Hence: JNB TI, HLoc ; wait till all 8 bits are transmitted
- CLR TI ; TI flag is reset ; remember TI flag must be cleared.

\* Mode 1 :-

In mode 1, 10 bits are transmitted through TXD pin and received through RXD pin. There is a start bit(0); then 8 data bits (LSB first) and a stop bit(1). This is shown in Fig 6.12 on receiving, the stop bit goes into RB8 in SCON. The baud rate is variable and is determined by timer 1 overflow rate. Therefore, before using this mode, one has to initialize timer 1. A simple program to initialize serial port in mode 1 is given in programming example # 6.12.

given in programming example # 6.12.  
The baud rate is calculated using the formula.

$$\text{Baud rate} = 2^{SMOD}/32 \times (\text{timer 1 overflow rate})$$



Ten bits transmitted in mode 1 of serial transmission.

```

; Programming example # 6.12
; Initializing the serial port in mode 1
MOV SP, #51H
; Note that SMOD is '0' after RESET
MOV SCON, #0100 0000B ; serial port in mode 1
MOV TMOD, #0010 0000B ; timer 1 in auto-reload mode
MOV TH1, #230D ; Baud rate = 1200 at 12MHz
SETB TR1
MOV SBUF, #56H ; wait till the transmission is over
CLR TI ; reset bit TI after transmission

```

If timer 1 is configured in auto-reload mode (or mode 2), with reload value in TH1 will be loaded into T1. This is convenient for generating baud rate. In this mode, TMOD high nibble will be 0010B. At 12MHz oscillator frequency, the timer clocking time is 1us. Now, the baud rate formula is simplified to

$$\text{Baud rate} = \left[ 2^{SMOD/32} \right] \times (\text{oscillator frequency}) / [12 \times (256 - (TH1))]$$

For example, if TH1 contents are 230D, and SMOD bit in PCON is 0, then the baud rate at 12MHz is 1201 baud or 1.2k approximately. To get exactly 1200 baud, the oscillator frequency must be 11.059 MHz. This shows the degree of dependency of the baud rate on the operating frequency. Thus, to be precise, the actual oscillator frequency must be measured on the oscilloscope.

To receive a byte in mode 1, the RI bit in SCON is tested for 1. Similarly, the REN bit in SCON must be 1. The following programming example # 6.13 will receive a byte through pin RXD.

```
; Programming example # 6.13
; Receiving a serial byte through RXD
MOV SCON, # 0101 0000B ; serial port mode 1 and REN bit is set
; SMOD w/o after RESET
MOV TMOD, #0010 0000B ; Timer1 in mode 2
MOV TH1, #230D ; baud rate 1.2k at 12MHz
SETB TR1 ; start timer 1
CLR RI ; ready to receive
JNB RI, $ ; wait till a byte is received in SBUF
MOV A, SBUF ; get the received byte in accumulator.
```

\* Mode 2 :-

In mode 2, 11 bits are transmitted with a low start bit, then 8 data bits, a 9th bit and a stop bit '1'. This is shown in Fig 6.13. The 9th bit is programmable. user program can define 9th bit as T8 in SCON. It may be the parity of data byte. On reception, SCON. If it may be the parity of data byte. On reception, SCON. If this 9th data bit goes into P08 in SCON. In mode 2, the bit SMOD in PCON and the oscillator frequency defines the baud rate and is given by.

$$\text{Baud rate} = \left[ 2^{SMOD/64} \right] \times [\text{oscillator frequency}]$$

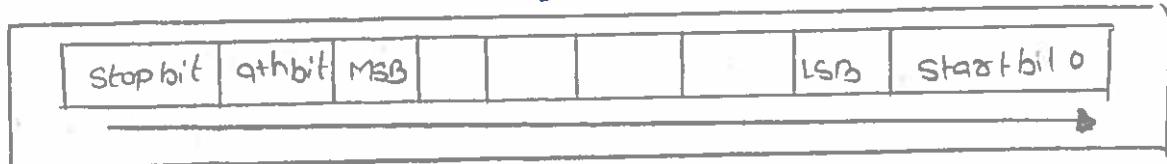


Fig: Eleven bits Transmitted in 8051 serial communication mode2

Now consider programming example # 6.14 to initialize the serial port in mode 2. At 12 MHz oscillator frequency if SMOD bit is 1, then the baud rate will be 375/1000 or 375k.

```
; Programming example # 6.14
; initializing the serial port in mode 2
CLR TI
MOV SCON, #1000 0000B ; serial port mode 2
SETB SMOD; SMOD=1 and baud rate = 375k at 12MHz
MOV SBUF, # 42H
JNB TI, $ ; wait till transmission is over
CLR TI
```

### \* Mode 3 :-

Again 11 bits are transmitted as shown in Fig 6.12. This is almost same as mode 2, except that the baud rate is defined by the timer 1 overflow rate. The baud rate calculations are exactly same as that of mode 1.

### \* Multiprocessor Communication :-

So far our discussion was about how to communicate between two 8051 devices. However, a general requirement may be to communicate among many 8051 devices. Setting SM2 bit in SCON register does this. The multiprocessor communication is supported in modes 2 and 3 only. Again 9 bits configuration of multiprocessor system assume the transmitter as a master and all others are slaves. When the master wants to transmit a block of data, it sends first the address byte of the slave. Now the question is, how this address byte is distinguished from the data byte? The 9th bit, while transmitting the address byte, does this. The 9th bit is '1' in case of address byte and '0' in case of data byte. SM2 bit of all the slaves is 1 after initialization in the multiprocessor mode. With  $SM2 = 1$ , the slave will be interrupted with the address byte only and no data byte. The addressed slave will clear its SM2 bit, and can do so. The addressed slave will clear its SM2 bit, and start receiving the data bytes. Other slaves who are not addressed will continue their own operations.