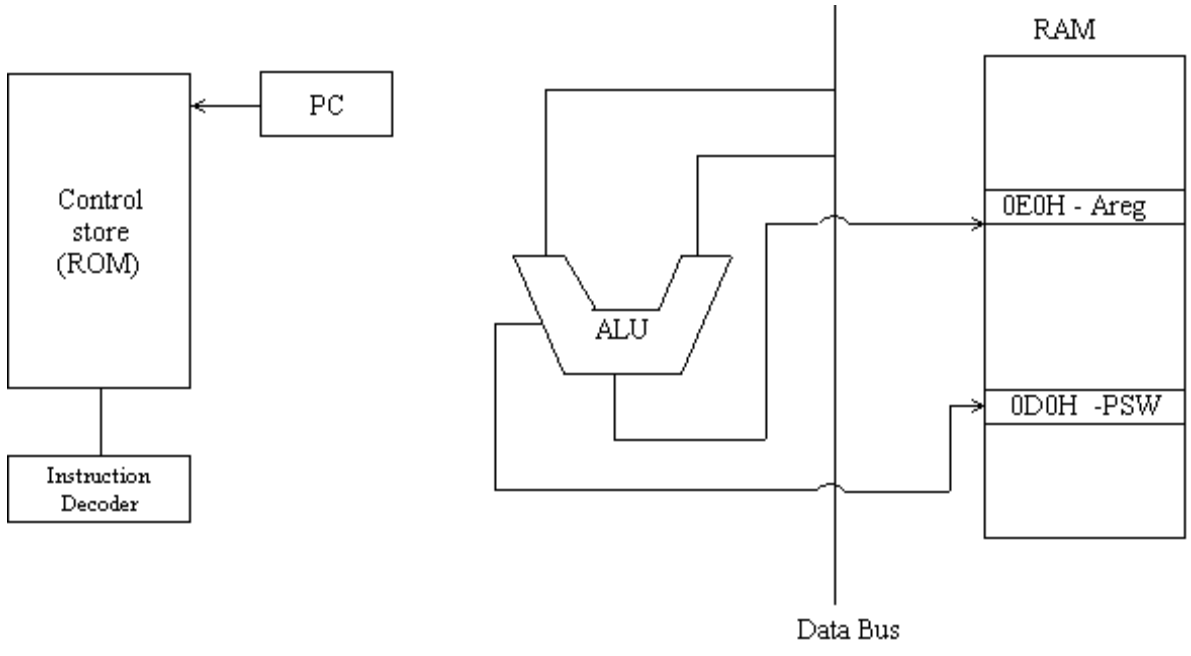


# 8051 MICROCONTROLLER:

8-bit microcontroller

- 8-bit CPU, Reg A, Reg B
- 15-bit PC and Data Pointer(DPTR)
- 8-bit PSW
- 8-bit SP
- Internal ROM and EPROM (8751)
- Internal RAM of 256 bytes

## Architecture of 8051:



## Registers in 8051

### Math Register

Bit	Address	Bit	Address
8	EO*	8	FO*
A		B	

\* indicates that each bit a of the register can be programmed.

### Interrupt Register

8	B8*	8	A8*
IP		IE	

7FH

General Purpose Register		
7F	Bit Addressable Register	00
MSB		LSB
} 20-2FH		
18-1FH	Register Base 3	8 byte
8 byte	Register Bank 2	} 10H-17H
0FH	Register Bank 1	} 8 byte
08H	R <sub>7</sub>	} Bank 0
07H	R <sub>6</sub>	
	R <sub>5</sub>	
	R <sub>4</sub>	
	R <sub>3</sub>	
	R <sub>2</sub>	
	R <sub>1</sub>	
00H	R <sub>0</sub>	

### Timer Control Register

8	89
TMOD	

8	88*
TCON	

Assembler- Assembly level language to Machine code

Compiler- High level language to Machine code

Simulator- software to simulate the function of a microcontroller

Emulator- Combination of software and hardware to simulate function of a microcontroller

### Timer Counter Register

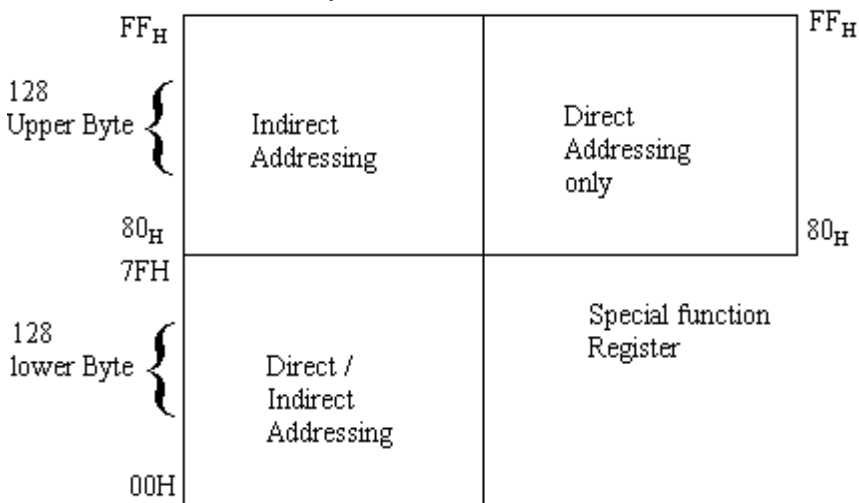
8	8C
TH0	

8	8A
TLO	

8	8D
THI	

8	8B
TLI	


### 8051 Internal Data Memory



It has 256 byte of internal RAM

SFR MAP

F8								
F0	b							
E8								
E0	ACC							
D8								
D0	PSW							
C8	T2CON	T2MOD	RCAP2L	RCAP2H	TL2	TH2		
C0								
B8	IP							
B0	IP3							
A8	IE							
A0	P2							
98	SCON	SBUF						
90	P1							
88	TCON	TMOD	TLO	TLO	THO	TH1	T	
80	PO	SP	DPL	DPH				PCON

 ← These n in the 8052 microcontroller.

**Addressing Modes:**

Ex: ADD A, #77 (immediate addressing)  
 A=A + 77(decimal)

**1) Immediate Addressing Mode:** Where data is available in the instruction itself.

**2) Bank Register Addressing:** Add A, RO  
 since it has 4 different banks each bank each having 8 bytes and the register RO of which bank is selected by the SFR PSW.

PSW: RSI and RSO selects which bank is to be selected.

CY	AC	FO	RSO	RS1	OV	--	P
----	----	----	-----	-----	----	----	---

**3)Direct Addressing Mode:**

Add A,80 A ← A+(Data of 80 in SFR)  
 ie, A ← A + PO

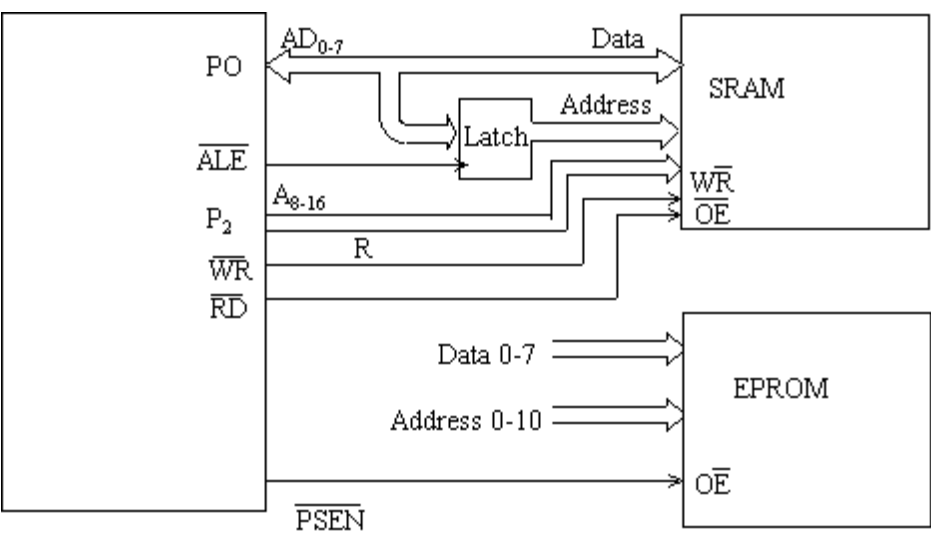
**4)Indirect Addressing Mode:** Registers R<sub>0</sub>, R<sub>1</sub>, DPTR r used to store address of the 8-bit and 16-bit.

Add A, @ RO ← Indirect memory addressing  
 A ← A + R<sub>0</sub>

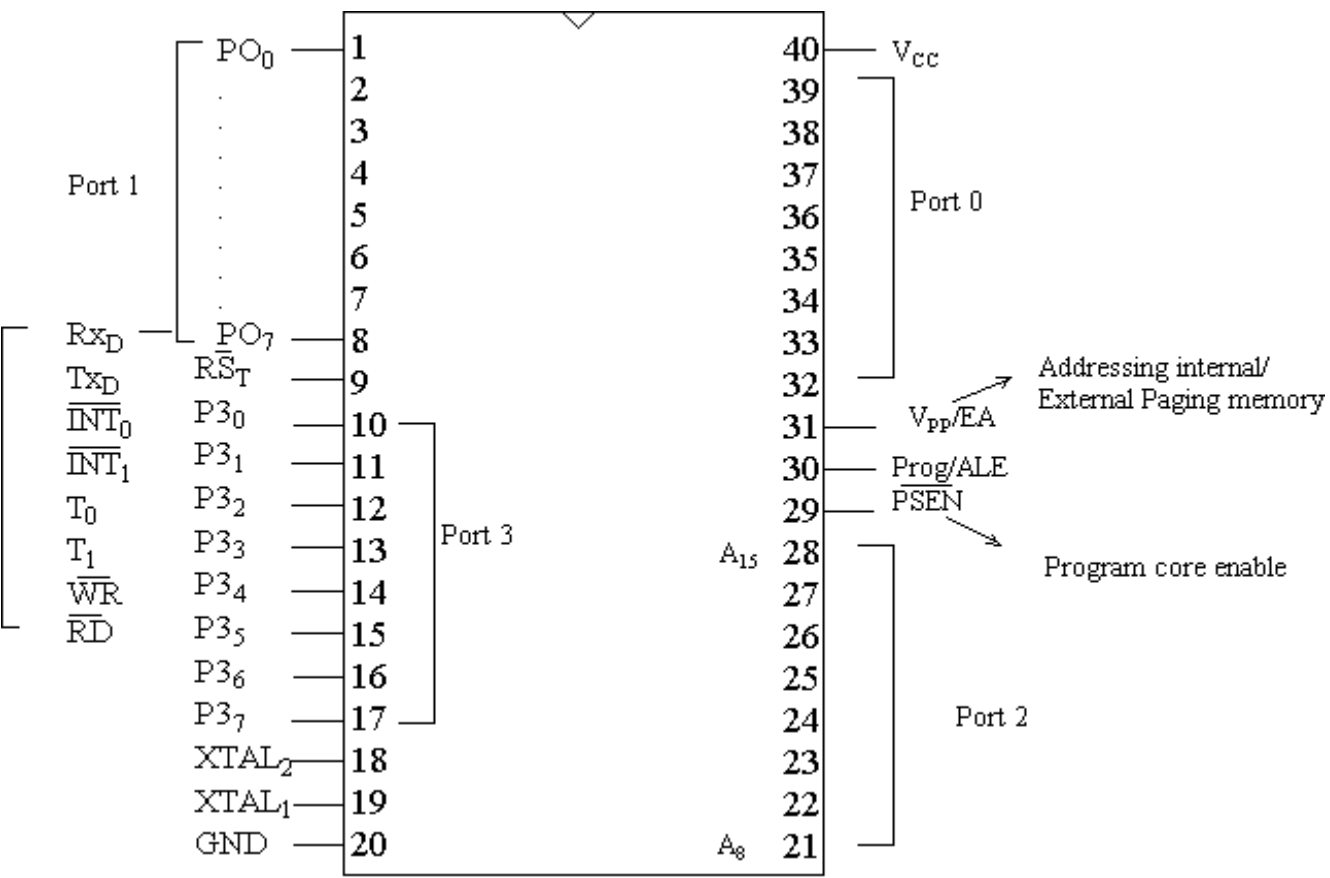
**5) Register Indirect Addressing**

Add A, @ DPTR  
 In this address in the DPTR may be internal or external memory address.  
 Ex. MOV A,30H  
 This takes 2 instruction cycles ie, 24 clock cycles.  
 MOV A,@ RO  
 This instruction takes only 1 instruction cycle ie, 12 clock cycles.

**External Memory Access:**

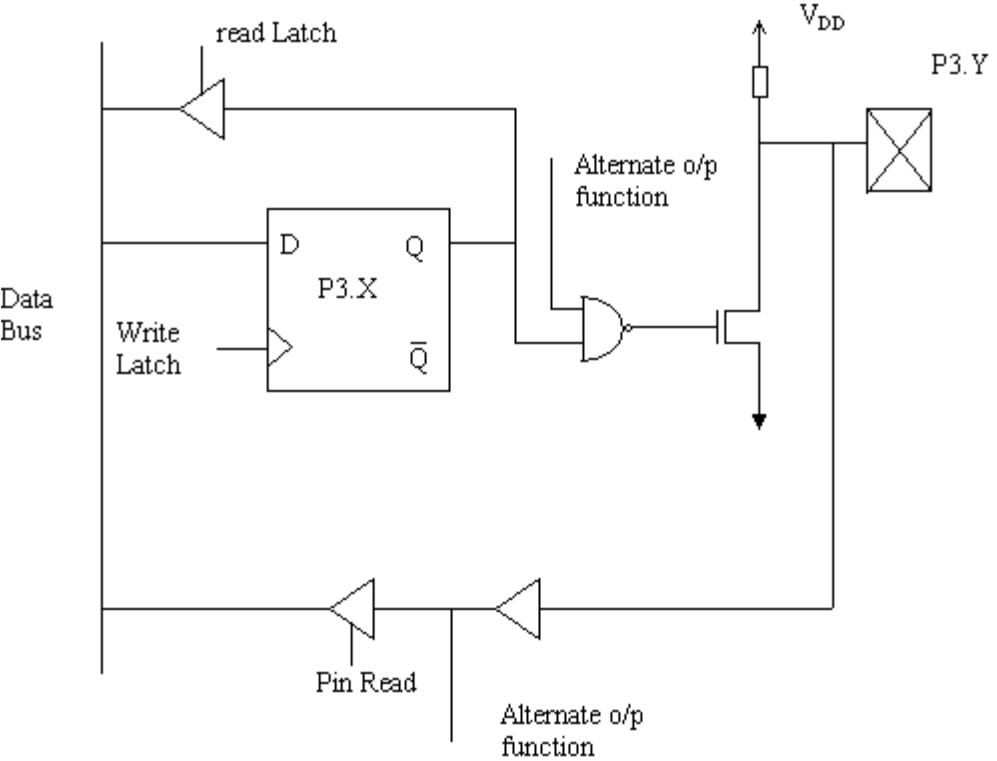


**40 Pin DIP Package of 8051**



**How to latch 1 to P3 X ?**

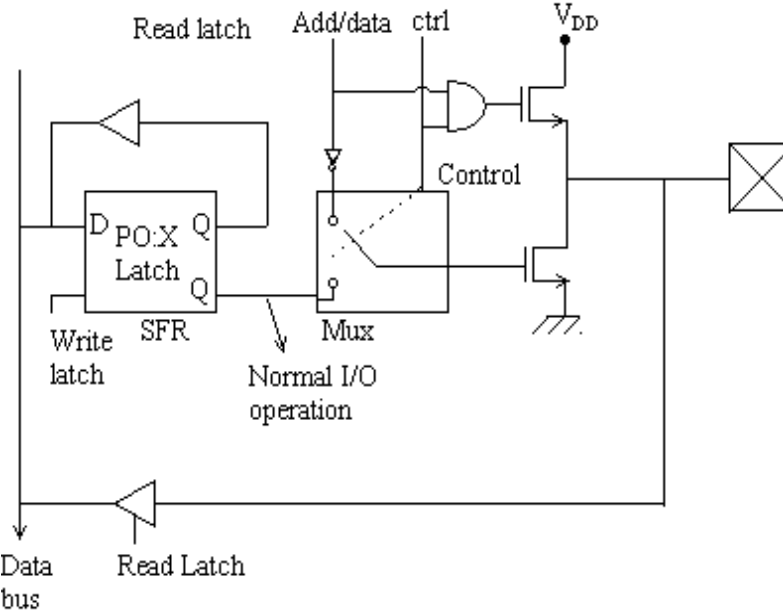
This latching is done in the following process.



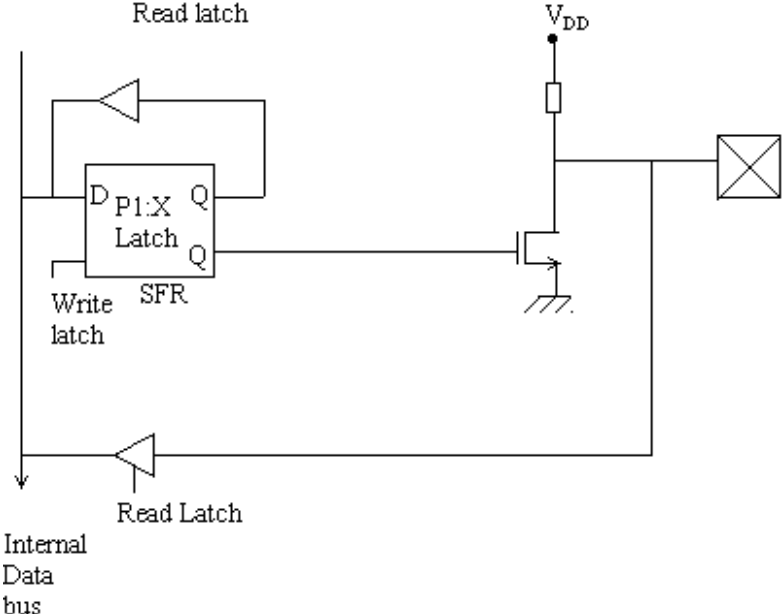
That to latch P3 to 1, first write to latch and load 1 into it. Hence the output of the gate is the alternate output function which latches the other functions.

**8051 I/O port Structure**

Port 0 AD<sub>0</sub>- AD<sub>7</sub>



Port 1

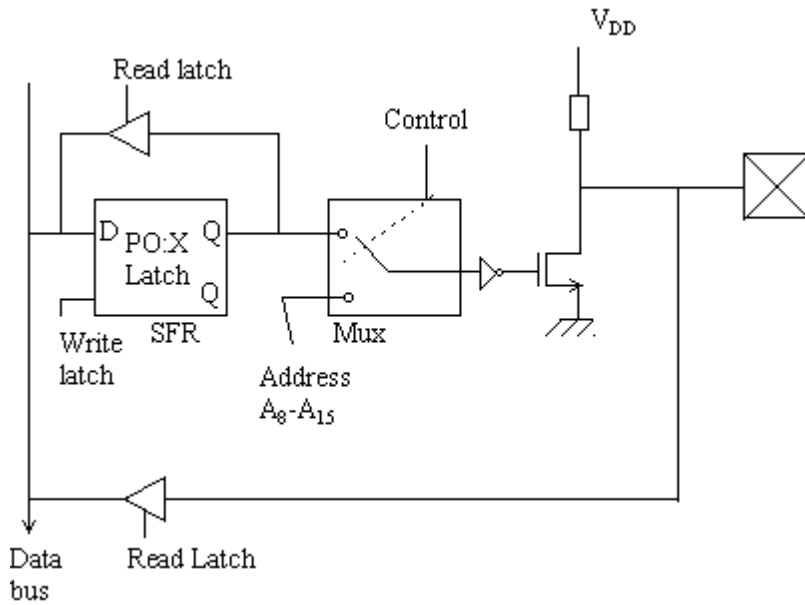


- This can be used as normal I/O port or address bus  
Mux switches between normal I/O port and address/data bus
- Pure bidirectional port

- This port is used for normal I/O port only
- Quasi bidirectional port

- For external memory read operation, SFR PO is over written by FFH
- While accessing 16-bit external memory

### Port 2 (A<sub>8</sub>-A<sub>15</sub>)



### Port 3 I/O Port + Alternative function

Discussed in previous class

Port can be used as normal I/O port and address bus

While addressing 16-bit external memory, P<sub>2</sub> SFR value remains unchanged.

ALL THE PORTS HAVE LIMITED CURRENT SOURCING CAPABILITY.

### External Memory Access

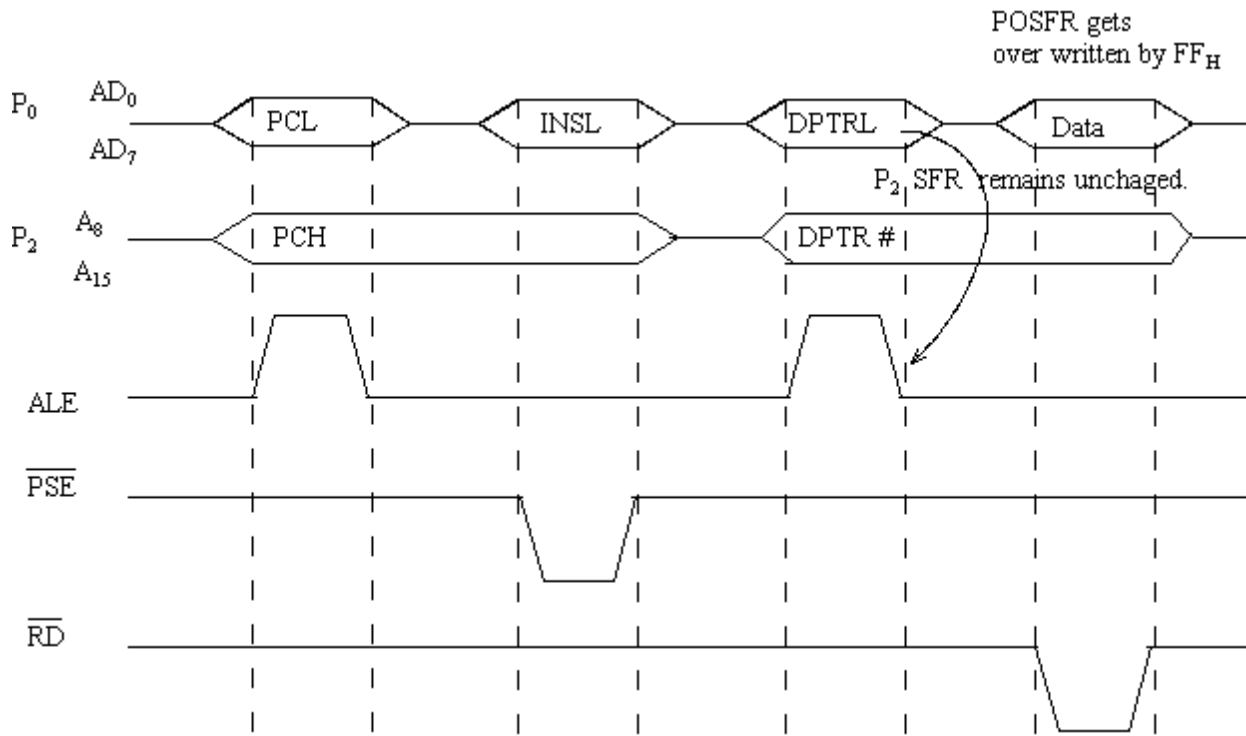
#### ROM (Programming Memory)

- Always 16-bit address
- $\overline{EA}$  Always external Memory is accessed  
If PC > FFF<sub>H</sub> then ROM is accessed
- 000-FFF<sub>H</sub> is internal Memory  
1000-FFFF<sub>H</sub> is external Memory  
 $\overline{EA}$  access internal memory

#### RAM CData

- 8-bit or 16-bit address
- If the address is stored in a 8-bit register then it points to 8-bit address  
MOV A, @RO
- For 16-bit address  
MOV A, @DPTR

### How MOV A,@DPTR works



For 8-bit Memory address access, P<sub>2</sub> Pins o/p the SFR register contents and helps in memory pages.

The higher order 8-bit address is taken the address available in the P<sub>2</sub> SFSR and the lower order 8-bit address is the data available in register RO.

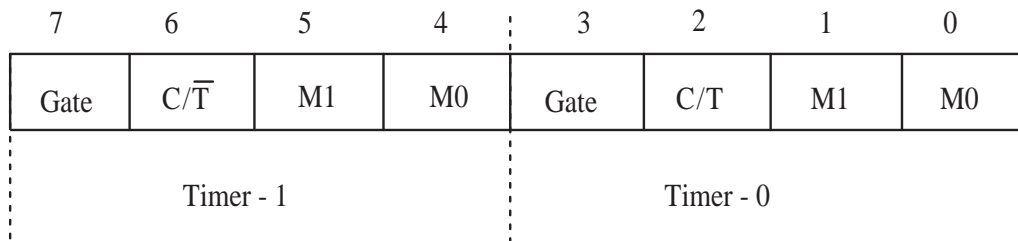
# Timer/Counter in 8051

8051 has two 16-bit Timer/counter registers. 8052 has these two plus one more: Timer 2. All these can be configured to operate either as *times* or *event counters*.

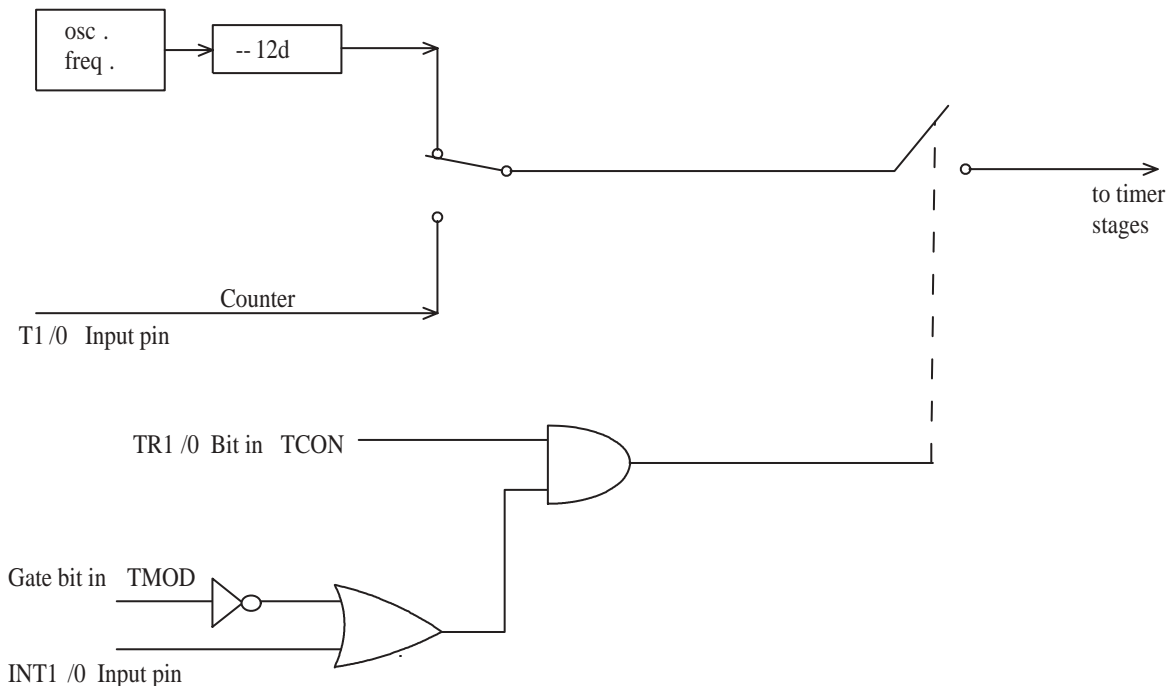
In the "Timer" function, the register is incremented every machine cycle. Now one can think of it as counting machine cycles (instruction cycles) and the clock rate is  $\frac{1}{12}$  oscillator frequency ( $\because$  one instruction cycle = 12 clock cycles). In "Counter" function, the register is incremented in response to a 1 to 0 transition at its corresponding external input pins, ie. T0, T1 or (in 8052) T2. In this function, the external inputs is sampled during *S5P2* of every machine cycle. When the samples show a high in one cycle and a low in the next cycle, the count is incremented. The new count value appears in the register during *S3P1* of the cycle following the one in which the transition was detected. Since it takes 2 machine cycles (24 oscillator periods) to recognize a 1-to-0 transition, the maximum count rate is  $\frac{1}{24}$  of the oscillator frequency.

Timer 0 and Timer 1 have 4 operating modes. Timer 2 in 8052 has 3 modes of operation: "Capture", "auto-reload" and "bandrate generator".

## Timer mode control (TMOD) Special function Register



### Timer/Counter Control logic





## Timer Mode-0

Setting timer x mode bits to 00 in the TMOD register results in using the THX register as a 8-bit counter and TLX as a 5-bit register (lower bits). The upper 3-bits of TLX are indeterminate and should be ignored. The timer overflow flag in TCON is set whenever THX goes from FFh to 00h.



## Timer Mode-1

Mode-1 is similar to mode-0 except TLX is configured as a full 8-bit counter. When the mode bits are set to 01 in TMOD.



## The Timer Control (TCON) Special Function Register

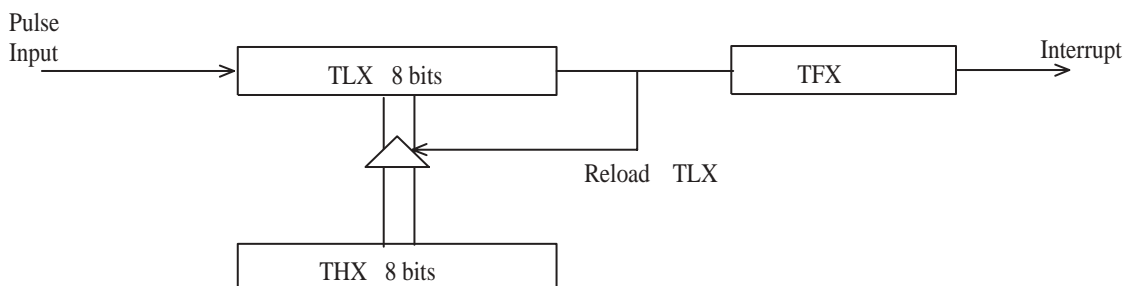
7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

**TF1** → Timer 1 overflow flag. Set when timer rolls from all 1's to all 0. Cleared when processor vectors to execute int. Since routine at 00/Bh

**TR1** → Timer 1 run control bit. Set to 1 by program to enable time to count. Clear to 0 by program (TRO - for Timer 0)

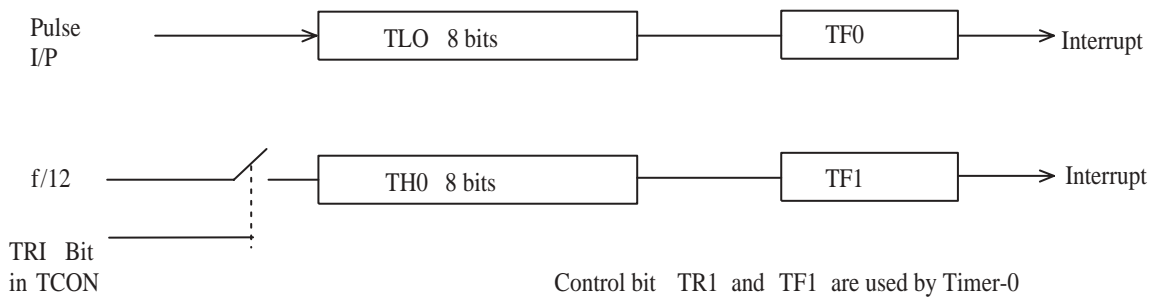
## Timer Mode-2

(Auto-reload feature) TLX is used as a 8-bit counter only. THX is used to hold a value that is loaded into TLX everytime TLX overflows from FFh to 00h. The time flag is also set when TLX overflows. The mode shows auto reload feature where TLX will be initialized to the content of THX after TLX overflows.



## Timer Mode-3

Timer 1 in Mode-3 simply holds its count. The effect is the same as setting TR1=0. Timer 0 in Mode 3 establishes TL0 and TH0 as two sperate counters.



Timer-1 can still be used in Mode-0,1 and 2 but no interrupt will be generated by Timer-1 while Timer-0 is in Mode-3.

## Timer 2

Like Timer-0 and 1, it can operate either as a timer or as an event counter.

T2CON

7	6	5	4	3	2	1	0
TF2	EXF2	RCLK	TCLK	EXEN2	TR2	$C/\overline{T2}$	$CP/\overline{RL2}$

This is selected by bits  $C/\overline{T2}$  in special function register T2CON. It has three operating modes: "Capture", "auto-load" and "band rate generator".

RCLK + TCLK	CP/RL2	TR 2	Mode
0	0	1	16-bit Auto-reload
0	1	1	16-bit capture
1	x	1	Bandrate gen
x	x	0	off

## Serial Interface:

The serial port is *full duplex*.

*SBUF* → *Special function Register*.

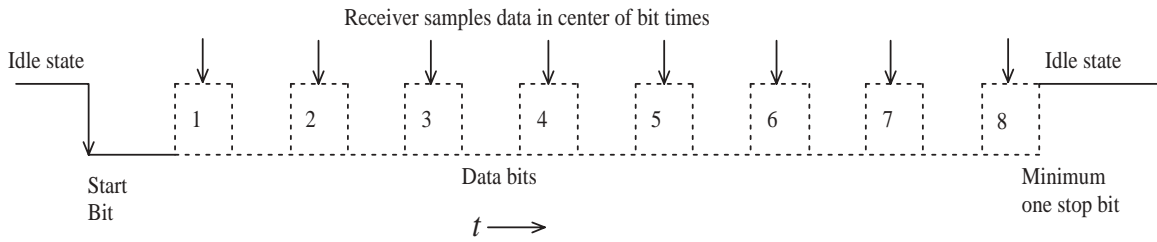
*Mode - 0* Shift register mode.

Serial data enters and exists through RXD. 8-bits are transmitted/received. Pin TXD is connected to the internal *shift frequency* pulse source to supply shift pulses to external circuits. The shift frequency or bandrate is fixed at 1/2 of the oscillator frequency.



**Mode - 1** Standard UART

10 bits are transmitted (through TXD) or received through (RXD), a start bit(0), 8 data bits (LSB first), and a stop bit(1). Once received, the stop bits go into RB8 in special function register SCON. The band rate is variable.



$$\text{Bit time} = \frac{1}{f}$$

**Mode - 2** Multiprocessor Mode.

11 bits are transmitted through TXD or recieved through RXD, a start bit (0), 8 data bits (LSB first), a programmable 9th bit and a stop bit(1).

On transmission, the 9th data bit (TB8 in SCON) can be assigned the value 0 or 1. Or, for example, the parity bit (P in the PSN) could be moved into TB8. On receive, the 9th bit goes into RB8 in SFR SCON, which the stop bit is ignored. The bandwidth is programmable to either 1/32 or 1/64 of oscillator frequency.

$$f_{band} = \frac{2^{SMOD}}{64} f_{osc}$$

**Mode - 3**

11 bits are transmitted through TXD or received through RXD: a start bit, 8 data bits (LSB first), a programmable 9th bit, and a stop bit (1). In fact, Mode 3 is same as Mode 2 in all respects except the band rate. The band rate in *Mode 3 is variable*.

**Serial Port Control Register**

SCON

(MSB)

(LSB)

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

SM0	SM1	MODE	Description	Band rate
0	0	0	shift register	$f_{osc}/12$
0	1	1	8-bit UART	Variable
1	0	2	9-bit UART	$f/32, f/64$
1	1	3	9-bit UART	Variable

SM2 Enables multiprocessor communication in Mode 2 and 3.

REN Enables serial reception.

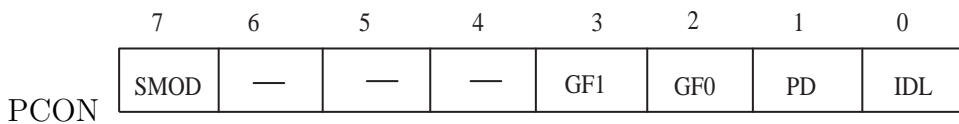
TB8 9th data bit that will be transmitted in Mode 2 and 3.

RB8 9th data bit that was received. In mode-1, a SM2=0, RB8 is the stop bit that was received. In mode-0, RB8 is not used.

T1 Transmit interrupt flag

R1 Receive interrupt flag

## The Power Mode Control (PCON) Special function reg.



$$f_{band} = \frac{2^{SMOD}}{32} \times \frac{f_{osc}}{12 \times [256 - TH1]}$$

Timer-1 is used to generate band rate for mode-1 using overflow flag of the timer to determine the band frequency. Typically Timer-1 is used in mode-2 as an auto load 8-bit that generates the band frequency.

If Timer-1 is not run in timer mode-2 then the band rate is

$$F_{band} = \frac{2^{SMOD}}{32} \times \text{timer 1 overflow frequency}$$

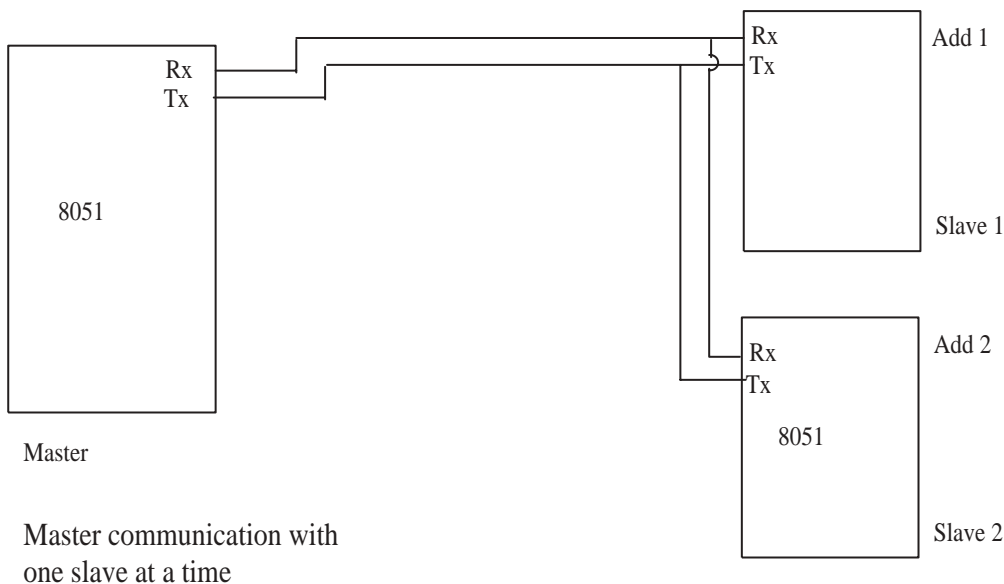
$$f_{clock} = 11.0592 \text{ MHz}$$

$$f_{band} = 9600$$

$$TH1 = 256 - \left( \frac{2^0}{32} \times \frac{11.0592 \times 10^6}{12 \times 9600} \right) = 253 = 0FDH$$

## Multiprocessor Communication

Mode 2 and 3 have a special provision for multiprocessor communication. In this mode, 9 data bits are received/transmitted. The port can be programmed such that when the stop bit is received, the serial interrupt will be activated only if RB8=1. This feature is enabled by setting bit SM2 in SCON. A way to use this feature is given here.

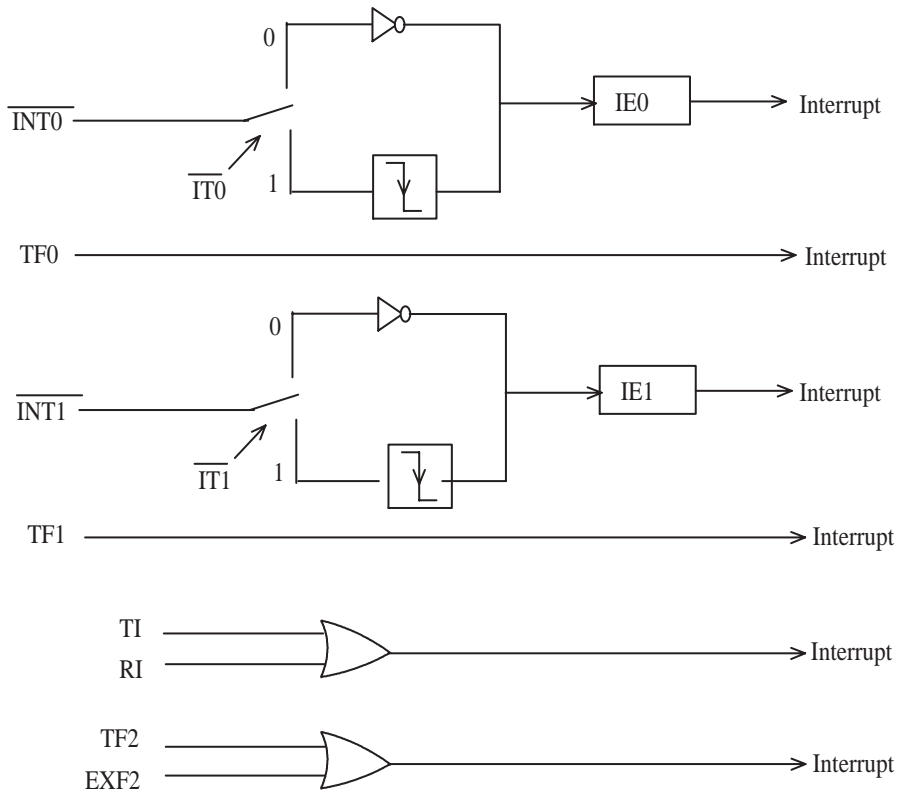


When a master processor wants to send a data byte to a slave, the master sends the address of the slave first. There may be many slave processors. An address byte differs from a data byte in that the 9th bit is 1 in an address byte and 0 in a data byte. An address byte interrupts all slaves when SM2=1. But a data byte does not interrupt the slaves if they have SM2=1. The address byte is checked by each slave and the target/addressed slave clears its SM2 so that it can receive the data

byte. The slaves that are not addressed leave their SM2's set and go on with their business, ignoring the incoming data bytes. SM2 has no effect in mode 0, and in mode 1, it can be used to check the reliability of the stop bit. In mode 1, if SM2=1, then receive interrupt will not be activated unless a valid stop bit is received.

## Interrupts

8051 provides 5 interrupt sources. The 8052 provides 6.



$\overline{INT0}$  and  $\overline{INT1}$  are external interrupts and can be level triggered or edge triggered (negative) depending upon  $\overline{IT0}/\overline{IT1}$  in TCON SFR.

$\overline{IT0}$  Set → falling edge triggered for  $\overline{INT0}$

$\overline{IT0}$  Cleared → low level triggered for  $\overline{INT0}$

IE0/IE1 Interrupt 0/1 edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.

Timer 0 and Timer 1 interrupts are generated by TF0 and TF1, which are set by a rollover in their respective Timer/Counter register (except Timer 0 in Mode 3). When a timer interrupt is generated, the flag that generated it is cleared by on chip hardware when the interrupt service routine is vectored to. The serial port interrupt is generated by the logical OR of RI and TI. Neither of these flags is cleared by hardware when the interrupt service routine is vectored to. These have to be cleared by software.

All of the bits that generate interrupts can be set or cleared by software, with the same result as though it had been cleared by hardware. That is, interrupts can be generated or pending interrupts can be canceled in software.

Each of these interrupts can be individually enabled or disabled by setting or clearing bit in the SFR IE. IE contains a global disable bit, EA which disable all interrupts at once.

## Interrupt Enable Register(IE)

(MSB)

(LSB)

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

**EA = 0** no interrupt is acknowledged.

**= 1** each int source is individually enabled or disabled

**ET2 = 0** disables Timer 2 overflow int

**= 1** enables Timer 2 overflow int

**ES = 0** Serial port int is disabled

**= 1** Serial port int is enabled

**ET1 = 0** Timer 1 overflow int is disabled

**= 1** Timer 1 overflow int is enabled

**EX1 = 0** External int 1 ( $\overline{INT1}$ ) is disabled.

**= 1** External int 1 ( $\overline{INT1}$ ) is enabled.

**ET0 = 0/1** Disables Timer 0 OF int

**EX0 = 0/1** Disables/enables external int ( $\overline{INT0}$ )

### Priority structure

Each interrupt source can be individually programmed to one of two priority levels by setting or cleaning a bit in special function register IP. A low priority interrupt can itself be interrupted by a high priority interrupt, but not by another low priority interrupt. If two requests of different priority levels are received simultaneously, the request of higher priority level is serviced. If requests of same priority level are received simultaneously, an internal polling sequence determines which request has to be serviced. Thus within each priority level, there is a second priority structure determined by the polling sequence, as follows:

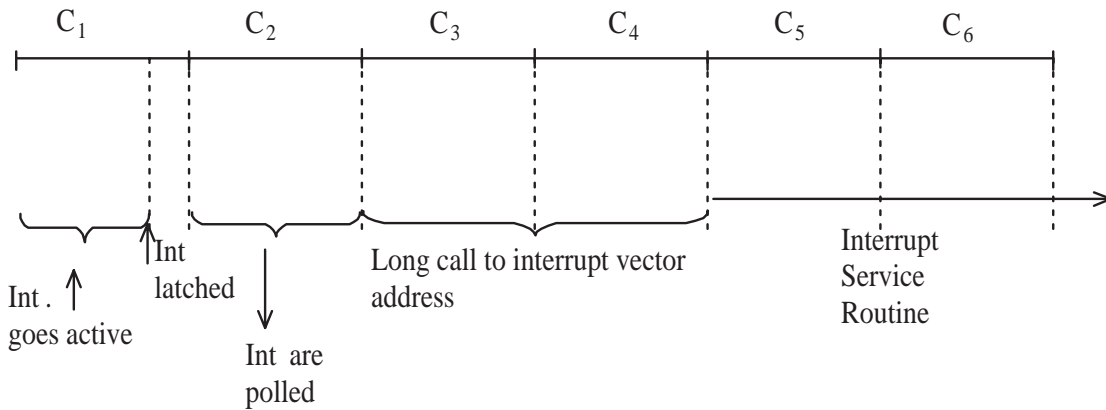
Source	Priority within level
1. IE0	(Highest)
2. TF0	↓
3. IE1	↓
4. TE1	↓
5. RI+TI	↓
6. TF2+EXF2	(Lowest)

**How the interrupts are handled** Interrupt flags are sampled in *S5P2* of each instruction cycle. The samples are polled during the following instruction cycle (machine cycle). If one of the flags was

in a set condition at S5P2 of the preceding cycle, the polling will find it and the interrupt system will generate an LCALL to the appropriate service routine, provided this hardware generated LCALL is not blocked by any of the following conditions:

1. An interrupt of equal or higher priority level is already in progress.
2. The current polling is not the final machine cycle of the instruction in progress
3. The instruction in progress is RET1 or any write to IE or IP registers.

If an interrupt flag is active but not being responded to for one of the above conditions, if the flag is not still active when the blocking condition is removed, the denied interrupt will not be serviced. This is because the interrupt flag once active but not serviced is not remembered. Every polling cycle is new



After an interrupt is vectored to, some interrupt flags are cleared and some are not by hardware. For example: Serial port and Timer 2 interrupt flags are not cleared automatically. This has to be done by user's software. IEU and IE1 are cleared if the interrupts are *transition activated*. TF0 and TF1 are cleared by hardware generated LCALL pushes PC into the stack but not PSW.

Source	Vector address
IE0	→ 0003H
TF0	→ 000BH
IE1	→ 0013H
TF1	→ 001BH
RI+TI	→ 0023H
TF2+EXF2	→ 002BH

### External Interrupt

$\overline{INT0}$  and  $\overline{INT1}$

Level triggered (Low) or Transition triggered (1-to-0 transition)

IT0/IT1=0 IT0/IT1=1

Low or high should be maintained at the pin for at least *12 clock cycles (1 machine cycle)*.

Special function Register (IP) Interrupt Priority Register (IP)

(MSB)				(LSB)			
—	—	PT2	PS	PT1	PX1	PT0	PX0

PT2	→	Timer 2	1-high priority	0-Low priority
PS	→	Serial Port Interrupt	1-high priority	0-Low priority
PT1	→	Timer 1 Interrupt	1-high priority	0-Low priority
PX1	→	External Int. 0	1-high priority	0-Low priority
PT0	→	Timer 0 Int.	1-high priority	0-Low priority
PX0	→	External Int 1	1-high priority	0-Low priority

### Software generated interrupts

When any interrupt flag is set to 1 by any means, an interrupt is generated unless blocked. This means that the program itself can cause interrupts of any kind to be generated by simply setting the desired interrupt flag to 1 using program instruction.

### Example of interrupt Use

#### Single-Step operation

The following program enables simple step operation

```
JNB P3.2, FDH
JB P3.2, FDH
RETI
```

#### Reset

Non-maskable Interrupt

Holding RST pin high for at least 2 machine cycles while the osc is running.

After RST is made low

PC ← 0000H

SP ← 07H

Postlatches ← FFH

SBUF ← XX

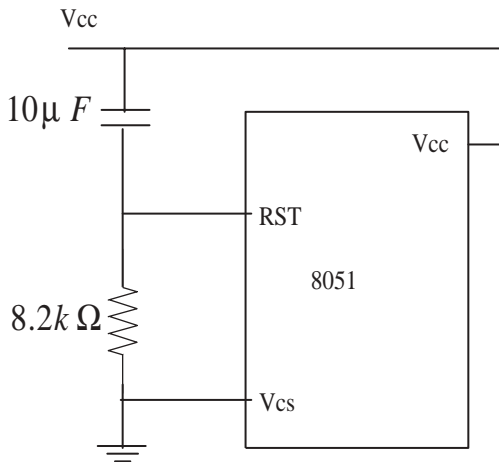
All other SFR ← 00

RAM content is not changed.



### Power-on Reset

When power is switched on ( $V_{cc}$ ), the capacitor behaves as a short circuit and RST pin remains high for considerable amount of time to enable the micro controller to go into RESET mode.

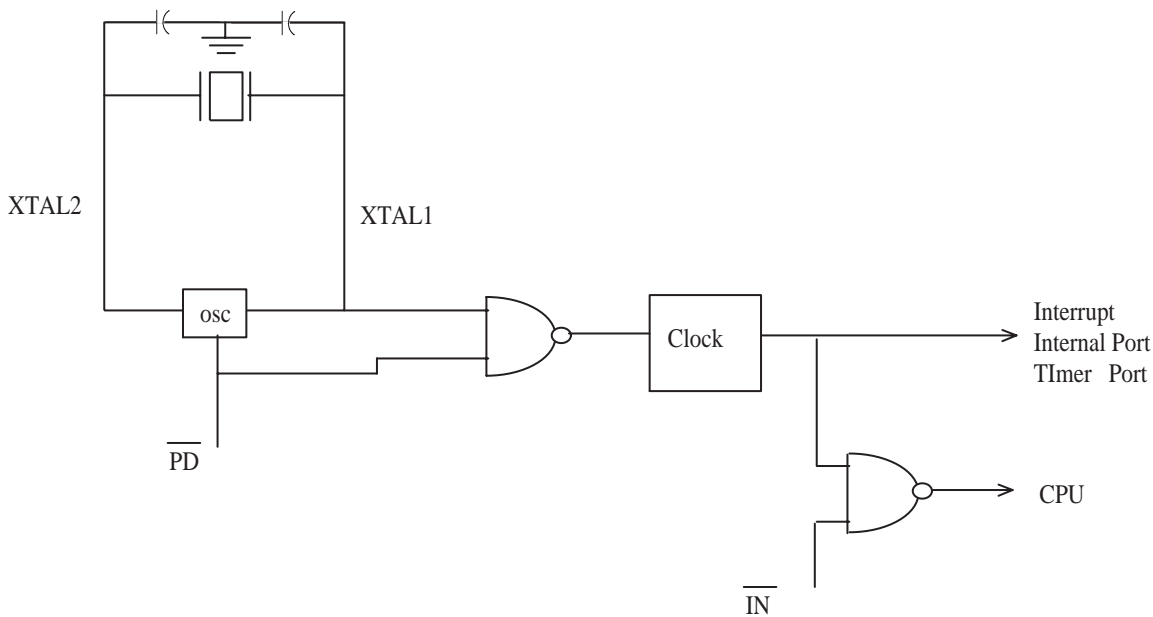


### Power-saving modes of operation

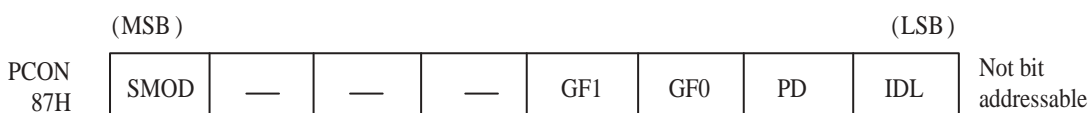
Often, power saving becomes important for microcontroller-based applications. The CHMOS version of 8051 provides reduced power modes of operation as a standard feature.

CHMOS → (Complementary High density MOS)- A chip with high density of CMOS transistors.

**CHMOS Power reduction modes** CHMOS version has *two power-reducing* modes, idle and power down. The internal circuitry which implements these features is given below.



Idle and power down modes are activated by setting the corresponding bits (IDL and PD respectively) in PCON special function register.



PD - Power down bits. Setting this bit activates power down operation

IDL - Idle mode bit. Setting this bit activates idle mode operation.

## Idle Mode

An instruction that sets  $PCON_0$  causes that to be the last instruction executed before going into the idle mode. In the idle mode, the internal clock signal is gated off to the CPU, but not to the interrupt, Timer and Serial port functions. The CPU status is preserved entirely. SP, PC, PSW, Acc, and all other registers maintain their data during the idle mode. The port pins hold their logical status they had at the time idle was activated. ALE and  $\overline{PSEN}$  are held at logic high levels.

There are two ways to terminate idle. Activation of any enabled interrupt will cause PCON0 to be cleared by hardware, terminating the idle mode. After RET1 is executed, the next instruction starts from the one following the instruction which enabled the idle mode.

The flag bits GF0 and GF1 can be used to give an indication if an interrupt occurred during normal operation or during the idle mode. For example, an instruction that activated idle can also set and or both flags. When idle is terminated by an interrupt, the ISR can examine the flag bits.

The other way of terminating the idle mode is with a *hardware reset*. (Reset should be high for two machine cycle or 24 clock cycles). The signal at the RST pin clears IDL bit directly and asynchronously. At this time the CPU resumes the program execution from where it left off; that is, at the instruction following the one that involved idle mode. Two to three machine cycles should be executed before the internal reset algorithm takes control.

## Power down mode

An instruction that sets PCON.1 causes that to be the last instruction executed before going into the power down mode. In the power down mode, the on-chip oscillator is stopped. With the clock frozen, all functions are stopped, but the on-chip RAM and special function register are held. The port pins output the values held by their respective SFRs, ALE and  $\overline{PSEN}$  output.

The only exit from Power down mode of operation is by a hardware reset. Reset redefines all SFRs, but does not change the on-chip RAM.

$V_{cc}$  may be reduced to as low as 2V during Power down mode. However  $V_{cc}$  should be restored to the rated value and allow clock to stabilize ( $\leq 10ms$ ) before the Power down mode is exited, ie, hardware Reset is pressed.

# 8051 Instruction Set

## Some notes

**Rn** Register R0-R7 of the currently selected register bank.

**direct** 8-bit internal data location address  
(Internal data with address 0-127 or SFR)

**@Ri** 8-bit internal Data RAM location addressed

**#data** 8-bit constant included in instruction  
(immediate 8-bit data)

**#data 16** 16-bit immediate data included in the instruction.

**add 11** 11-bit destination address. Used by ACALL and AJMP. The branch will be within the same 2k byte page of Program Memory as the first byte of the following instruction.

**addr 16** 16-bit destination address. Used by LCALL and LJUMP. A branch anywhere within 64 kbyte of Program Memory space.

**rel** Signed (2's complement) 8-bit offset byte used by SJUMP and all conditional jumps. Range is -128 to +127 bytes relative to first byte of the following instruction

**bit** Direct addressed bit in internal RAM or SFR.

## Arithmetic Instructions

MNEMONICS	DESCRIPTION	BYTES	INSTRUCTION CYCLES
ADD A,Rn	$A \leftarrow A + Rn$	1	1
ADD A,direct	$A \leftarrow A + Direct$	2	1
ADD A,@Ri	$A \leftarrow A + @Ri$	1	1
ADD A,#data	$A \leftarrow A + data$	1	1
ADDC A,Rn	$A \leftarrow A + Rn + C \text{ bit}$	1	1
ADDC A, direct	$A \leftarrow A + (direct) + C \text{ bit}$	2	1
ADDC A,@Ri	$A \leftarrow A + @Ri + C \text{ bit}$	1	1
ADDC A,#data	$A \leftarrow A + data + C\text{-bit}$	2	1
SUBB A,Rn	$A \leftarrow A - Rn - C\text{-bit}$	1	1
SUBB A,direct	$A \leftarrow A - (direct) - C\text{-bit}$	2	1
SUBB A,@Ri	$A \leftarrow A - @Ri - C\text{-bit}$	1	1
SUBB A,#data	$A \leftarrow A - data - C\text{-bit}$	2	1
INC A	$A \leftarrow A + 1$	1	1
INC Rn	$Rn \leftarrow Rn + 1$	1	1
INC direct	$direct \leftarrow (direct) + 1$	2	1
INC @Ri	$@Ri \leftarrow @Ri + 1$	1	1
DEC A	$A \leftarrow A - 1$	1	1
DEC Rn	$Rn \leftarrow Rn - 1$	1	1
DEC direct	$direct \leftarrow (direct) - 1$	2	1
DEC @Ri	$@Ri \leftarrow @Ri - 1$	1	1
INC DPTR	$DPTR \leftarrow DPTR + 1$	1	2
MUL AB	$A \leftarrow \text{lowbyte}(A * B)$ $B \leftarrow \text{highbyte}(A * B)$	1	4
DIV AB	$A \leftarrow \text{quotient}(A / B)$ $B \leftarrow \text{remainder}(A / B)$	1	4
DA A	decimal adjust acc	1	1

## Logical Instructions

MNEMONICS	DESCRIPTION	BYTES	INSTRUCTION CYCLES
ANL A,Rn	$A \leftarrow A.Rn$	1	1
ANL A,direct	$A \leftarrow A.Direct$	2	1
ANL A,@Ri	$A \leftarrow A.@Ri$	1	1
ANL A,#data	$A \leftarrow A.data$	2	1
ANL direct,A	$(direct) \leftarrow (direct)A$	2	1
ANL direct,#data	$(direct) \leftarrow (direct).data$	3	2
ORL A,Rn	$A \leftarrow A+Rn$	1	1
ORL A,direct	$A \leftarrow +(direct)$	2	1
ORL A,@Ri	$A \leftarrow A+@Ri$	1	1
ORL A,#data	$A \leftarrow A+data$	2	1
ORL direct,A	$(direct) \leftarrow (direct)+A$	2	1
ORL direct,#data	$(direct) \leftarrow (direct)+data$	3	2
XRL A,Rn	$A \leftarrow A \oplus Rn$	1	1
XRL A,direct	$A \leftarrow A \oplus (direct)$	2	2
XRL A,@Ri	$A \leftarrow A \oplus @ Ri$	1	1
XRL A,#data	$A \leftarrow A \oplus data$	2	1
XRL direct,A	$A \leftarrow direct \oplus A$	2	1
XRL direct,#data	$direct \leftarrow direct \oplus data$	3	2
CLR A	$A \leftarrow 00H$	1	1
CPL A	$A \leftarrow \overline{A}$	1	1
RL A	Rotate Left	1	1
RLC A	Rotate left through carry	1	1
RR A	Rotate right	1	1
RRC A	Rotate right through carry	1	1
SWAP A	Swap nibbles within the ACC	1	1

## Data transfer Instructions

MNEMONICS	DESCRIPTION	BYTES	INSTRUCTION CYCLES
MOV A, Rn			
MOV A, direct			
MOV A, @Ri			
MOV A, #data			
MOV Rn, A			
MOV Rn, direct			
MOV Rn, #data			
MOV direct, A			
MOV direct, Rn			
MOV direct, direct			
MOV direct, @Ri			
MOV direct, #data			
MOV direct, A			
MOV @ Ri, A			
MOV @ Ri, direct			
MOV @ Ri, #data			
MOV DPTR,#data16 A			
MOV A, @A+DPTR	Code byte move to A relative to DPTR		
MOV A, @A+PC	Code byte move to A relative to PC		
MOVBX A, @Ri	External data memory move to A (8-bit addr)		
MOVBX A, @DPTR	External data memory move to A (16-bit addr)		
MOVBX @Ri, A	Move A to ext data memory (8-bit addr)		
MOVBX @DPTR, A	Move A to ext data memory (16-bit addr)		
PUSH direct			
POP direct			
XCH A, Rn			
XCH A, direct			
XCH A, @Ri			
XCHD A, @Ri			

## Bit Operations / Jump instructions

MNEMONICS	DESCRIPTION	BYTES	INSTRUCTION CYCLES
CLR C	Carry $\leftarrow$ 0		
CLR bit	bit $\leftarrow$ 0		
SETB C	Carry $\leftarrow$ 1		
SETB bit	bit $\leftarrow$ 1		
CPL C	$C \leftarrow \overline{C}$		
CLR bit	bit $\leftarrow \overline{\text{bit}}$		
ANL C,bit	$C \leftarrow C.\text{bit}$		
ANL C, /bit	$C \leftarrow C.\overline{\text{bit}}$		
ORL C,bit	$C \leftarrow C+\text{bit}$		
ORL C,/bit	$C \leftarrow C + \overline{\text{bit}}$		
MOV C,bit	$C \leftarrow \text{bit}$		
MOV bit, C	bit $\leftarrow C$		
JC rel	Jump if C=1		
JNC rel	Jump if C=0		
JB bit, rel	Jump if bit = 1		
JNB bit, rel	Jump if bit = 0		
JBC bit, rel	Jump if b=1 and b $\leftarrow$ 0		
ACALL addr11	Absolute jump		
LCALL addr16	Long jump		
RET	Return from subroutine		
RET1	Return from interrupt		
AJUMP addr11	Absolute jump	2	2
LJUMP addr16	Long jump	3	2
SJMP rel	Short jump		
JMP @ A+ DPTR	Jump relative to DPTR		
JZ rel			
JNZ rel			
CJNE A, direct, rel	Compare with Acc and jump if not equal		
CJNE A, #data, rel			
CJNE Rn,#data, rel			
CJNE @Ri,#data, rel			
DJNZ Rn, rel	Decrementreg and jump if not 0		
DJNZ direct, rel			

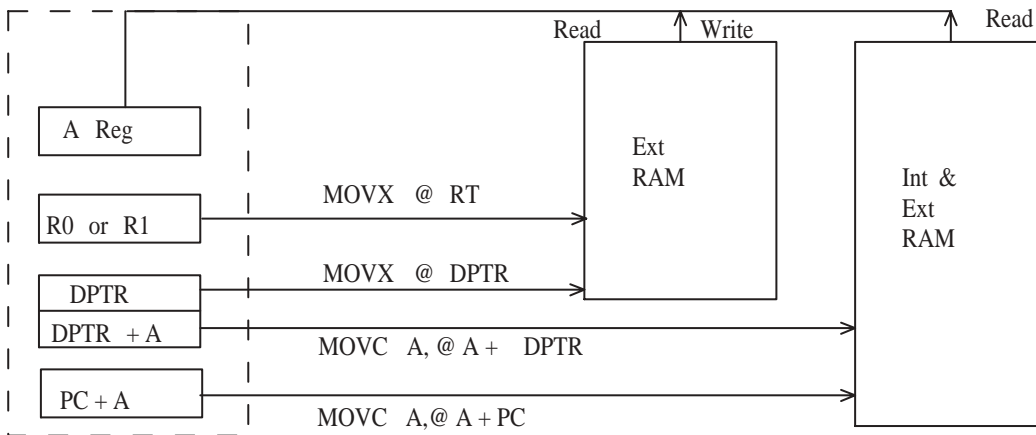
### Direct bit addressing

Values between 0 and 127 (00H and 7FH) define bits in a block of 16 bytes of on-chip RAM between addresses 20H-2FH. They are numbered consecutively from the lowest-order bytes lowest order bit through the highest order bit.

Bit addresses between 128 and 255 (80H and 0FFH) correspond to bits in a number of special function registers mostly used for I/O or peripheral device control. These positions are numbered with a different scheme than RAM. The five high-order address bits match those of the registers own address

while the three low-order bits identifies the bit position within that register.

## External Addressing using MOVX and MOVC



## Jump and Call Program Range

### Relative Range:

Jump that replaces the program counter content with a new address that is greater than the address of the instruction following the jump by 127 or less than the address of the instruction following jump by 128 are called relative jumps. The address following the jump is used to calculate the relative jump because the PC is incremented to the next instruction before the current instruction is extended.

Relative jump has two advantages. First, only 1 byte of data (2's complement) need to be specified for jumping ahead(positive range 0-127) or jumping back (negative range -128). Specifying only 1 byte saves program bytes and speeds up program execution. Second, the program that is written using relative jumps can be relocated anywhere in the program namely without reassembling the code to generate absolute addresses.

The disadvantage of relative jump is the short jump range (-128 to 127). This can be problematic in large programs where multiple relative jump may be require if higher jump range is required. Instructions using relative range jump are SJMP rel, and all conditional jumps.

### Short Absolute Range:

Short Absolute range makes use of the concept of dividing memory into logical divisions called pages. Program memory may be regarded as one continuous stretch of addresses from 0000H to 0FFFFH or it can be divided into a series of pages of any convenient binary size.

The 8051 program memory is arranged on 2k byte pages giving a total of 32 (20H) pages. The



hexadecimal address of each page is shown in the following table.

<i>Page</i>	<i>Address Range</i>	<i>Page</i>	<i>Address Range</i>
00	0000 - 07FF	10	8000 - 87FF
01	0800 - 0FFF	11	8800 - 8FFF
02	1000 - 17FF	12	9000 - 97FF
03	1800 - 1FFF	13	9800 - 9FFF
04	2000 - 27FF	14	A000 - A7FF
05	2800 - 2FFF	15	A800 - AFFF
06	3000 - 37FF	16	B000 - B7FF
07	3800 - 3FFF	17	B800 - BFFF
08	4000 - 47FF	18	C000 - C7FF
09	4800 - 4FFF	19	C800 - CFFF
0A	5000 - 57FF	1A	D000 - D7FF
0B	5800 - 5FFF	1B	D800 - DFFF
0C	6000 - 67FF	1C	E000 - E7FF
0D	6800 - 6FFF	1D	E800 - EFFF
0E	7000 - 77FF	1E	F000 - F7FF
0F	7800 - 7FFF	1F	F800 - FFFF

It can be seen that the upper 5 bits of the program counter hold the page number and the lower 11 bits of the program counter hold the address within each page. Thus an absolute address is formed by taking page number of the instruction following the branch and attaching the absolute page range address of 11 bits to it to form the 16-bit address.

Difficulty is encountered when the next instruction (the instruction following the jump instruction) starts at X800H or X000H. This places the jump or call address on the same page as the next instruction. This does not give rise to any problem on forward jump, but results in error if the branch is backward in the program. This should be checked by assembler and the user should be instructed to relocate the program suitably.

Short absolute range jump is also relocatable as the relative jump. Instructions using short absolute range are

```
ACALL  addr  11
AJMP   addr  11
```

### **Long Absolute Jump:**

Address that can access the entire program from 0000H to FFFFH use long-range addressing. Long range addresses require more bytes of code to specify and relocatable only at the beginning of 64 K byte pages. Since the normal code memory is only 64k bytes, the program must be reassembled every time a long-range address changes and then branches are not generally relocatable.

Instructions using long absolute range are

```
LCALL  addr 16
LJMP   addr 16
JMP    @ A+DPTR
```

## Example of Conditional Jump

```
                Org 0100H

Loop:   MOV A, #10H
        MOV RO, A
ADDA:   ADD A, RO
        JNC ADDA

                MOV A, #10H
ADDR:   ADD A,RO
        JNB 0D7H, ADDR
        JBC 0D7H, LOOP
```

## Example of External Data

```
ORG      0
MOV      RO,#043H
MOV      A, #12
MOVX     @RO, A
MOVX     A, #34
MOVX     A, @RO
```

## Character transmission using a time delay

A program called Senddata takes the character in A register, transmit it, delays for transmission time, and then returns to the calling program. Time 1 must be used to set the bandrate, which is 2400 band in this program. The delay for one character is  $\frac{10}{2400}sec = 0.00416s$  or  $4.16msec$ . The software delay of 5msec is used. Time 1 generates a bandrate close to 2400. Using a 12 MHz crystal, the reload value is  $256 - 12 \times 10^6 / (32 \times 12 \times 2400)$ , which is 242.98 or 243. This gives rise to an actual bandrate of 2404. SMOD is programmed to be 0.

```
; Send data, using a 12 Mhz Crystal for VART timing.
; 2400 nominal band rate fir an actual band rate of 2404.
; Delay between characters for 5 msec.
```

```
EQU     BAUDNUM, 0F3H
EQU     DELAY, 0A6H
EQU     DLYLSB, 05H
EQU     DLYMSB, 00H

ORG     0000H
ANL     PCON, #7FH
ANL     TMOD, #3FH
ORL     TMOD, #20H
MOV     THI, #BAUDNUM
SETB    TRI
MOV     SCON, #40H
```

XMIT:       MOV SBUF, #'A'  
             ACALL XMITTIME  
             SJMP XMIT

XMITTIME:   MOV A, #DLYLSB  
             MOV B, #DLYMSB  
             ACALL SOFTIME  
             RET

SOFTIME:     PUSH 07H  
             PUSH ACC  
             ORL A,B  
             CJNE A, #00H, OK  
             POP ACC  
             SJMP DONE

OK:          POP ACC

8031: No on chip program memory  
 128 KB of on chip RAM  
 8032: No ROM, 256KB on chip RAM  
 8051: 4K byte of Masked ROM (factory programmable)  
 128 KB of on chip RAM  
 Upto 12 or 16 MHz  
 8052: 8K byte of ROM, 256Kb RAM  
 8751: 4K byte of EPROM (UV Erasable)  
 Upto 12 or 16 MHz

**ATMEL**

AT 89C51: 4K byte of Reprogrammable flash memory  
 128K byte of RAM(on-chip)  
 From 0-24MHz

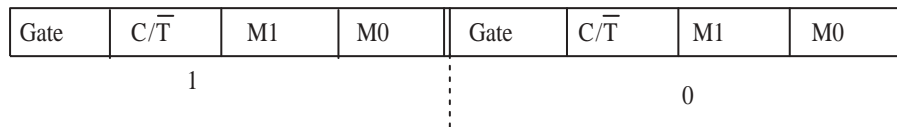
**Interrupt driven data reception**

```
BAUDNUM EQU 0F3H
ORG 0000H
SJUMP OVER
```

```
ORG 0023H
```

```
RECEIVE: CLR RI
          MOV PI, SBUF
          RETI
```

TMOD Register



```
OVER: ANL PCON, #7FH ; Set SMOD=0
      ANL TMOD, #0FH ; Alter Timer=1 conf only
      ORL TMOD, #20H ; Timer 1 in mode -2
      MOV TH1, #BAUDNUM ; Reload value
      SETB TR1, ; Run Timer - 1
      MOV SCON, #40H ; Serial Mode - 1
      SETB REN, ; Enable serial reception
      ORL IE, #90H ; Enable Interrupt
WAIT: SJMP WAIT, ; Wait for recieving data
      END
```

**Interrupt driven character transmission**

```
BAUDNUM EQU 0F3H
ORG 0000H
SJUMP OVER
```

```

SERIAL CLR TI
        MOV SBUF,#'A'
        RET1
OVER:   ANL PCON,#7FH ; SMOD=0
        ANL PCON,#7FH ; SMOD=0
        ANL TMOD,#0FH ;
        ORL TMOD,#20H ; Timer-1 in mode - 2
        MOV TH1,#BAUDNUM ; Reload value
        SETB TR1, ; Run Timer-1
        MOV SCON,#40H ; Serial mode-1
        ORL IE,#90H ; Enable serial int
        MOV SBUF,#'A' ; Send data
WAIT:   SJMP WAIT,
        END

```