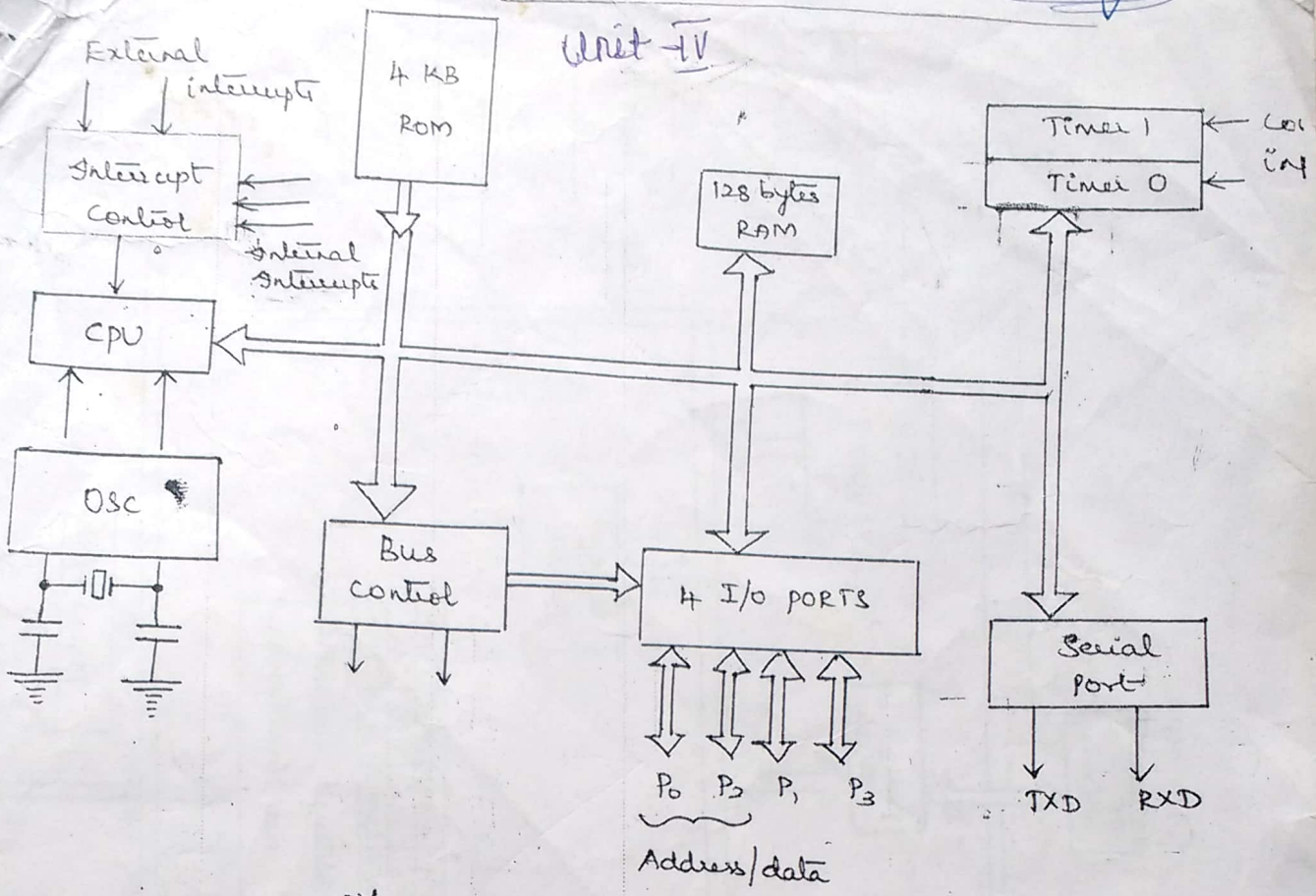
UNIT - IV

8051 MICROCONTROLLER

Unit - IV

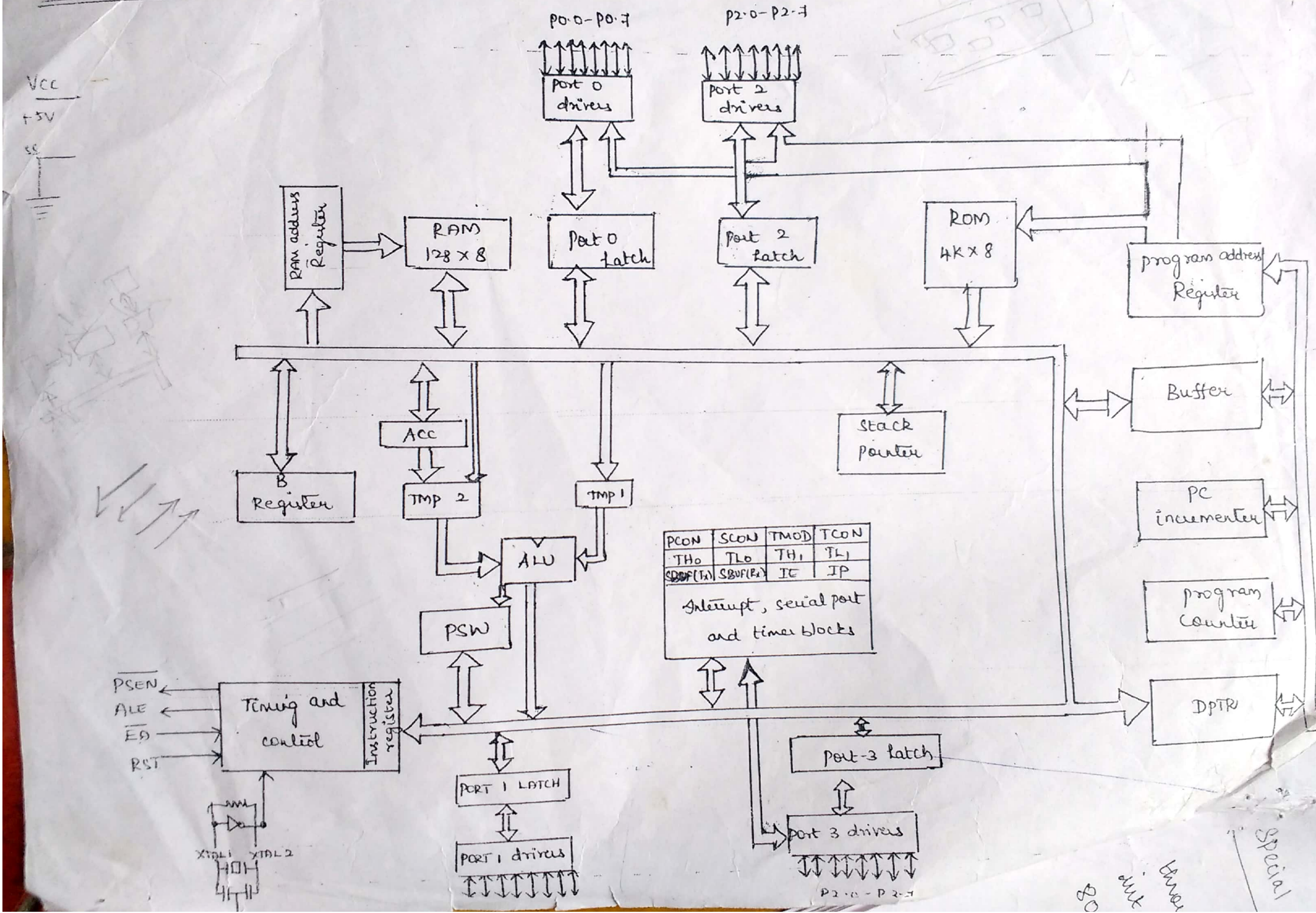
BLOCK DIAGRAM :-



The following are the resources offered by 8051 μc

* 8 bit CPU
* On-chip oscillator
* 4 KB of ROM (program memory)
* 128 bytes of RAM (Data memory)
* 21 special function register
* 32 I/O lines
* 64 KB address space for external data memory
* 64 KB address space for program memory
* Two 16 bit timer/counter
* Five source interrupt structure
* Full duplex serial port
* Bit addressability

7 Architecture :-



Special

Special Function Registers

All the resources in 8051 can be accessed through special function registers maintained in the internal data memory. There are 21 SFRs in the 8051. Some special function registers are bit-addressab

ACC – accumilator
B – B register
PSW – Program Status Word
SP – Stack Pointer
DPTR (low) – Data Pointer Low
DPTR (high) – " High
P0 – Port 0
P1 – Port 1
P2 – Port 2
P3 – Port 3
IP – Interrupt Priority
IE – Interrupt Enable
TMOD – Timer / counter Mode
TCON – Timer / counter Control
T2CON
TH0 – Timer 0 reg (High)
TL0 – " " (Low)
TH1 – Timer 1 reg (High)
TL1 – " (Low)
SCON – Serial Control
SBUF – Serial Data Buffer
PCON – Power Control.

I/O Ports

Port 0 (P0.0 – P0.7):

These are 8 bit bidirectional I/O port which are bit addressable. These pins also act as the lower part of the address bus as well as data bus while accessing external data memory.

Port 1 (P1.0 – P1.7)

This is an 8 bit bidirectional I/O port with internal pull up.

Port 2 (P2.0 – P2.7)

This is also an 8 bit bidirectional I/o with internal pull ups. This also has the alternate function of higher order address byte, while accessing the external memory.

Port 3 (P3.0 – P3.7)

This is an 8 bit bidirectional I/o port with internal pullups. This also has other alternate functions which are listed below.

- P3.0 – RXD (Serial input)
- P3.1 – TXD (serial output)
- P3.2 – $\overline{INT0}$ (External interrupt 0)
- P3.3 – $\overline{INT1}$ (External interrupt 1)
- P3.4 – T0 (Timer 0 external i/p)
- P3.5 – T1 (Timer 1 ")
- P3.6 – $\overline{WR}$ (write for ext data memory)

Timer / Counters

The 8051 has two 16 bit timers / counters. These can be programmed independently. There is a bit in TMOD SFR that specifies whether it is a timer or counter. If this bit is set, it works as a counter and if reset, works as a timer.

Each timer can be operated in any of the four modes.

Mode 0 : 13 bit timer

Mode 1 : 16 bit timer

Mode 2 : 8 bit timer with auto reload

Mode 3 : Two split timer.

Timers can be used for

1) Generating delay
2) Measuring time length between events
3) Counter
4) As a baud rate Generator

Serial Port

The 8051 μc has got all the circuitry in it for serial transmission. The RXD pin is used to receive input data serially and TXD pin is used to transmit data serially. The serial communication is full duplex, meaning that 8051 can receive & transmit at the same time. The receiving unit is buffered a

transmitting & receiving the

This can be configured into four mode depending on the application.

Mode 0 : Shift reg with constant baud rate
Mode 1 : 8 bit UART with variable "
Mode 2 : 9 bit UART with constant "
Mode 3 : 9 bit UART with variable "

INTERRUPT

There are five interrupt sources in 8051. They are

(1) External interrupt 0 (Highest priority)
(2) External " 1
(3) Timer 0 interrupt
(4) Timer 1 "
(5) Serial interrupt (Lowest priority)

All the five interrupts are maskeable and are vectored interrupts. Each has its own ISR address fixed by the manufacturer.

Each of these interrupts can be individually enabled or disabled by setting or clearing a bit in the special function register IE. All these sources can be programmed either to high priority level or lower priority level, by setting or clearing bits in the IP register.

Memory Organisation

The 8051 can access upto 64 KB program memory and 64 KB of external data memory and internal RAM locations

This belongs to Harvard Architecture since program & data memory are seperate.

Program Memory

If $\overline{EA}$ = 0V

External prog. memory ⇒ 0000 - FFFF (64K)

If $\overline{EA}$ = 5V

Internal prog. memory ⇒ 0000 – 0FFF (4K)

External " ⇒ 1000 – FFFF (60K)

Data Memory

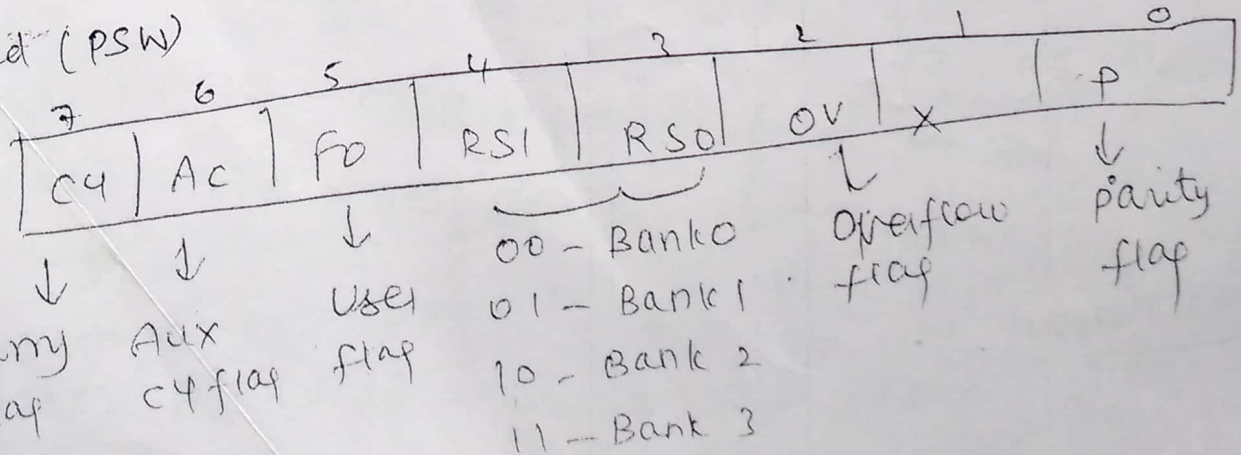
The 8051 can access 64 KB of external ... and 128 bytes of internal data RAM

and also 21 special function registers.

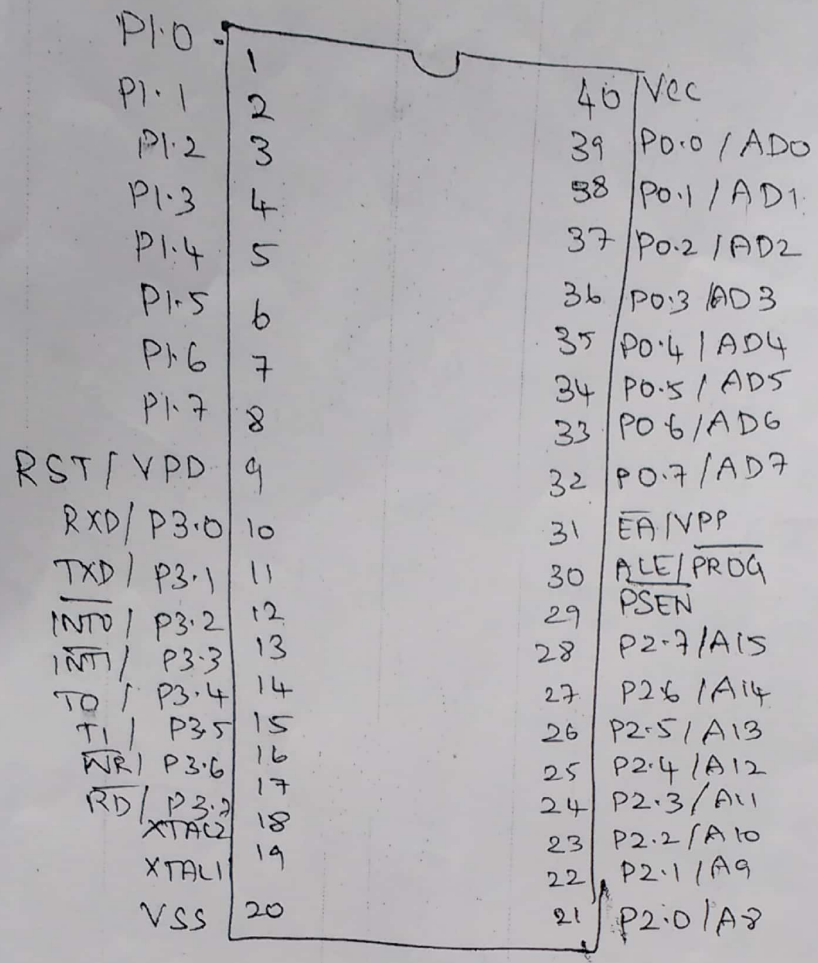
For accessing external data memory, the processor can issue either 8 bit or 16 bit address. To access internal data memory, 8 bit address is used. The lower 128 addresses are used as 128 byte on chip RAM and upper (128-255) are used as address for various SFR's.

Internal RAM Partitioning

The lowest 32 bytes (0-31) are reserved for 4 banks of 8 registers, each R0-R7, out of which one bank may be used at any time. The working bank is specified in two bits of program status word (PSW)



Pin diagram of 8051



Vcc → power supply Vss → Gnd

P0.0 - P0.7 → Port 0 (I/O)

or

AD0 / AD7 (Address / data for ext memory

P1.0 – P1.7 → Port 1 (I/O)

P2.0 – P2.7 → Port 2 (I/O) or A8 - A15 / Address for ext mem

P3.0 – P3.7 → Port 3 (I/O)

Other fns: RXD – (serial i/p) TXD – (Serial o/p) INT0 → Ext int i/p

INT1 – Ext interrupt T0 – Timer 0 ext i/p T1 → Tmr 1 ext i/p

WR – Ext memory write RD – External memory read

ALE / PROG → Address latch enable

XTAL2 & XTAL1 → Internal clk crystal

PSEN → for fetching data from external program memo

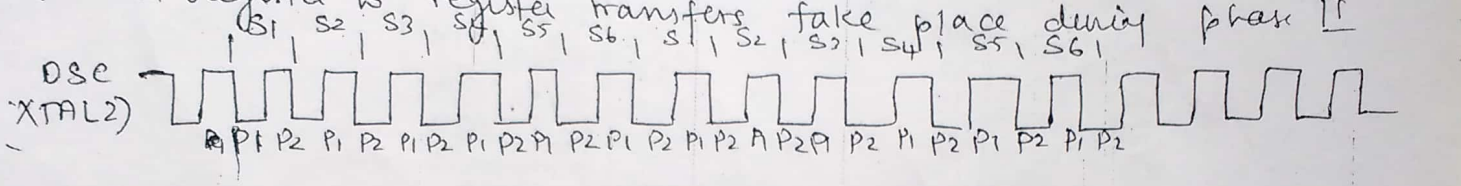
EA / VPP → 1 → Internal program memory 0 → Ext prog memo

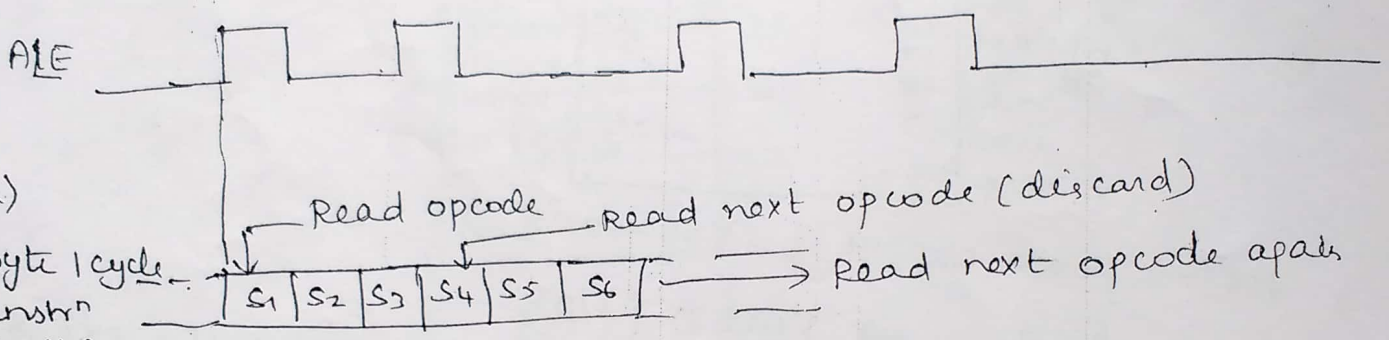
1 → reset the chip.

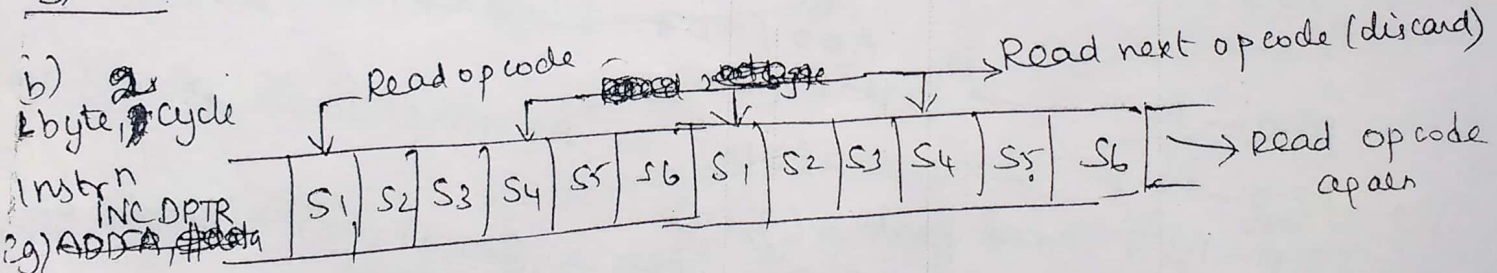
Timing Diagram

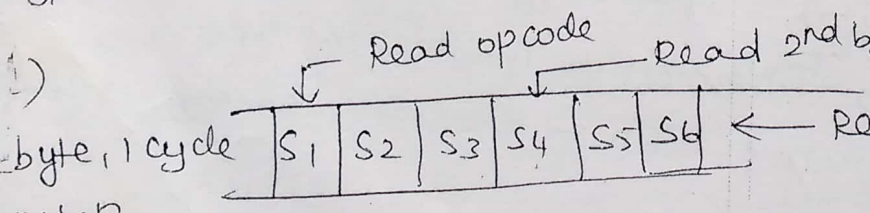
The clock generating circuit in 8051 divides the oscillator clock by 2 and provides a two phase clock to the CPU. Phase 1 (P_1) will be active during first half of each of clock periods and phase-2 (P_2) will be active during second half.

A machine cycle consists of six internal clock periods S_1 to S_6 (also called states) or 12 oscillator clock cycles. All arithmetic and logic operations takes place during phase-1. Internal register to register transfers take place during phase-2.









eg: ADD A

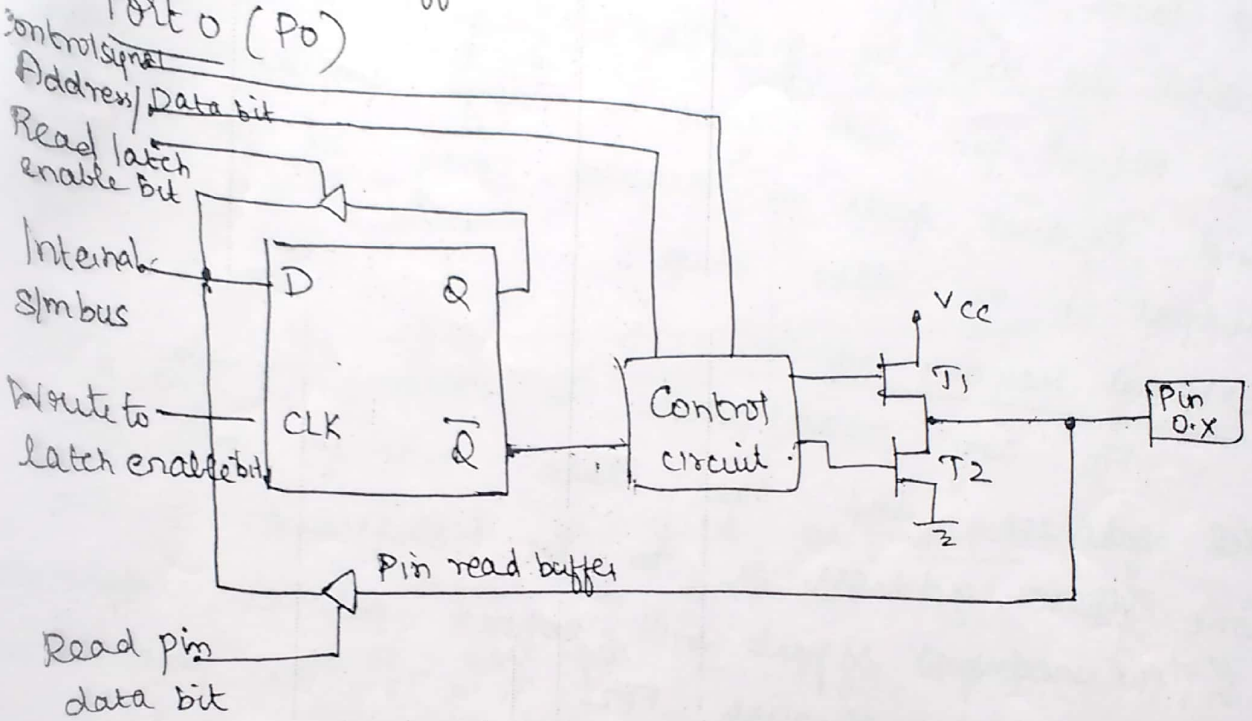
ALE is generated twice in each machine cycle, once during S1P2 and S2P1 and second time during S4P2 and S5P1. This signal is used for latching lower order address from data. However, during [illegible] memory access, ALE is not generated

I/O PORTS

The 8051 has four bidirectional ports P0 - P3. P0 has open collector o/p and P1, P2, P3 have internal pullup. Each I/O line can be independently used as either input or output line.

Each port line contains a latch, a driver and an input buffer.

Port 0 (P0)



This is an 8 bit open collector bidirectional I/O port. It can perform two functions

* Simple I/O operation
* External memory interface for lower order address bus and data bus.

(i) Simple I/O operation

When a 1 is loaded to the latch as part of o/p operation, both the FETs T1 and T2 are turned off. This causes port 0 pin to float to high impedance state and it gets connected to the pin read buffer. An external pull up register is

required to supply a high output. Thus the port is configured for input operations. Information on Po pi is transferred to internal bus when read pin sign is generated.

(When used as an output port, data will be directly loaded to latch. If a 0 is the output on the latch, it will turn on FET T_2 and thus the output pin will be grounded. If a 1 is being output, the output pin will float to high impedance state and external pull up register may be used to output 1 as data state).

ii) External Memory Interface

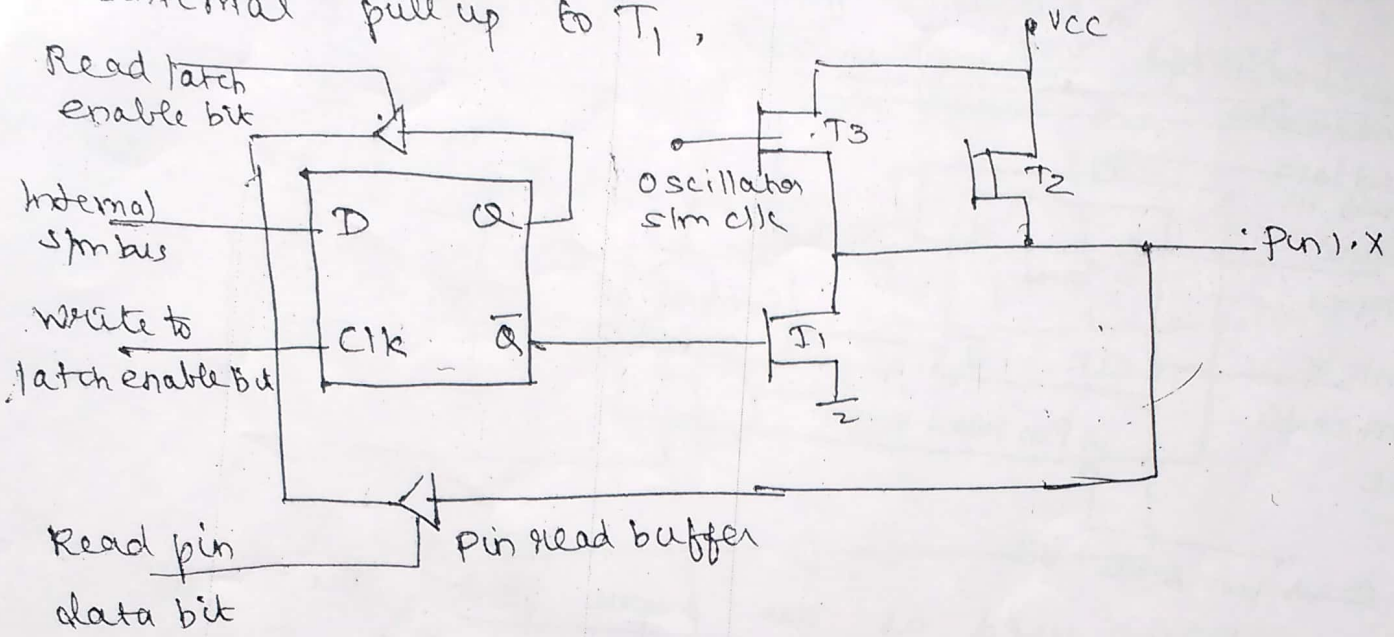
For any read/write to external memory, first the address and then data transfer will be done. When address is to be transferred to the pin, the control signals will connect address information directly to pin through FETs T_1 & T_2. The port latch is disconnected in this case.

If address bit = 1, then T_1 will be turned ON and T_2 will be turned off. This will cause logic 1 signal on the port pin. When address bit = 0, then T_1 is turned off and T_2 is turned ON. This will ground the output pin and provide logic 0 signal

& After the transfer of address information, if memory read is required, then 8051 places a '1' on latch and reconfigures pin to receive data

PORT 1 (P_1)

This is a bidirectional I/O port with internal pull ups. The circuit has three FETs T_1, T_2 and T_3. FETs T_2 and T_3 work as internal pull up to T_1,



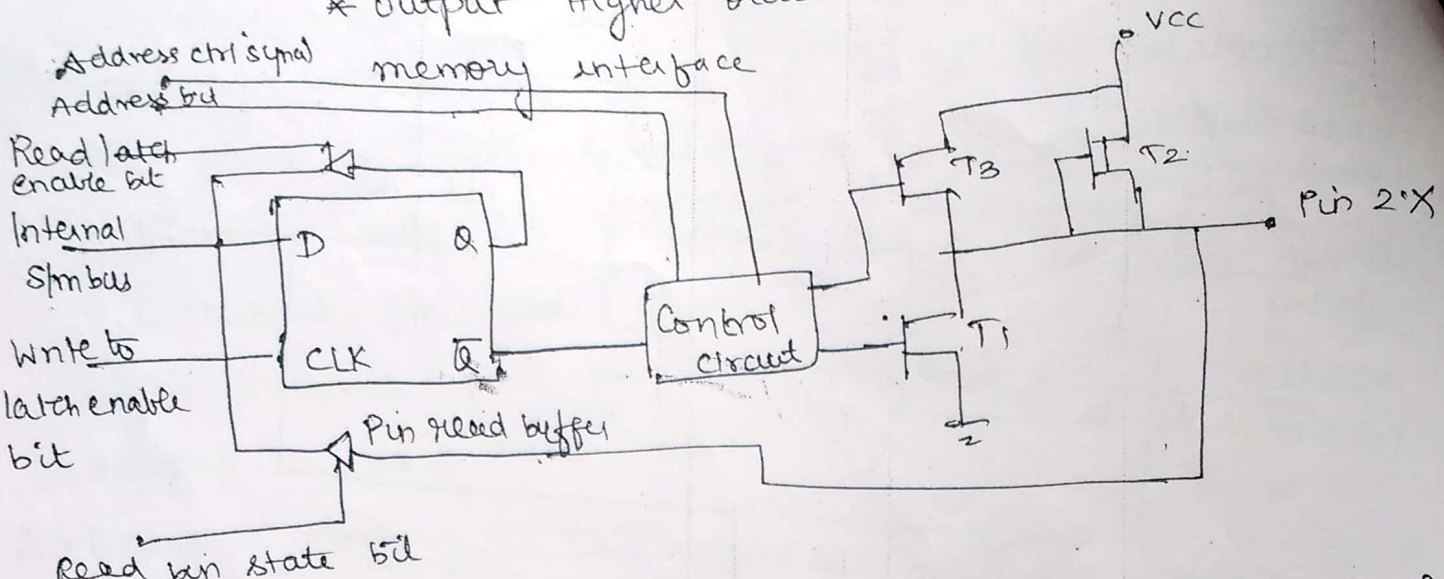
When Port 1 is used as, input, logic level '1' is loaded to latch. This will turn off FET T_1. This will effectively float port pin to high impedance state and will be connected to pin read buffer. The pin is at logic level 1 at this instant. An external device may place a 0 by driving the pin to ground, or it may place a 1 on pin by leaving it at logic state 1.

When used as an output port, the value will be loaded to latch. The latches will contain level '0' will turn off FET T_1 and this will drive the pin high through the pull up register formed by T_2 & T_3. The latches containing level 0 will turn on FET T_1. This will ground port pin and will show logic level '0'.

PORT 2

This can used as a

* Bidirectional I/o port
* Output Higher order address bus for external memory interface



When used as an input port, the logic level '1' is stored in the latch. The FET T_1 is turned off and floats in the high impedance state. The pin is directly connected to Pin read buffer. The external device can place a '0' or '1' on pin.

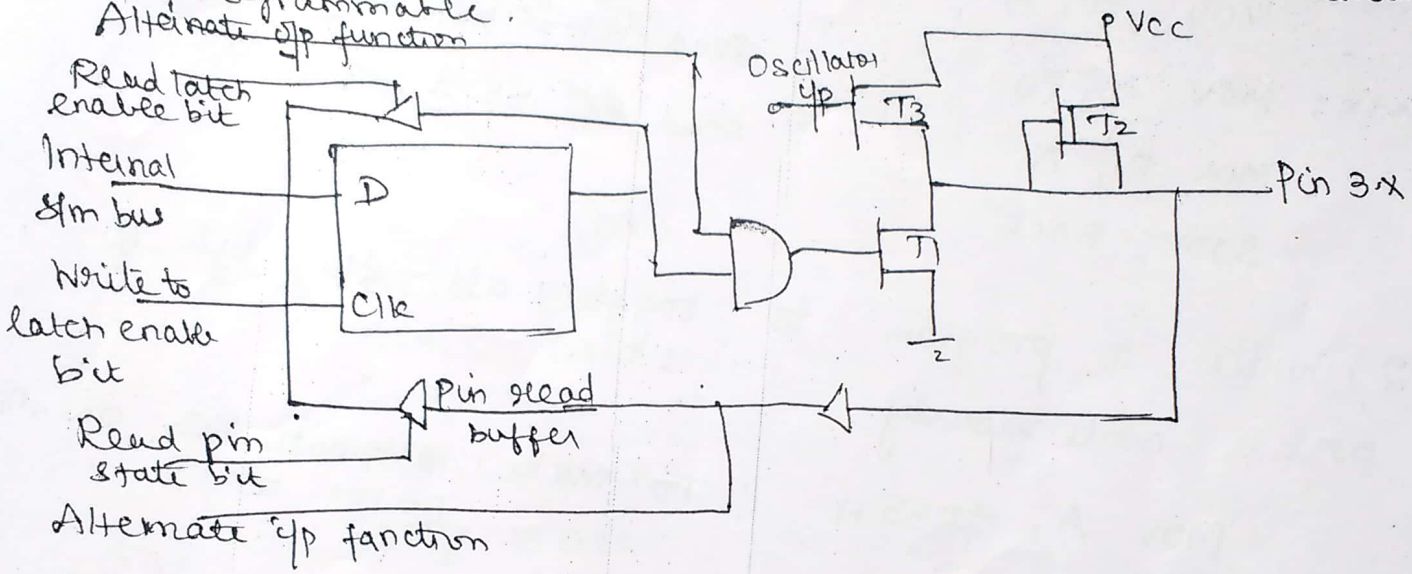
When used as an output port, the working is as follows. The latch containing 0 will turn on FET T_1, thus grounding pin. The latch containing 1 will turn off FET, thus the pin will be driven to logic level '1'.

When Port 2 is used to output higher order address bits for external memory access, the control signal will bypass the latch and the address bits are directly connected to pin through FETs

PORT 3

(8)

Port 3 is a bidirectional I/O port with internal pull up. Different pins of port 3 also facilitate alternate functions. The function of port 3 is either under the control of port latches or Special function registers. These pins are individually programmable.



Alternate functions of different pins of Port 3

- P3.0 → Serial data i/p (RXD)
- P3.1 → Serial data o/p (TXD)
- P3.2 → External interrupt 0 ($\overline{INT0}$)
- P3.3 → " " 1 ($\overline{INT1}$)
- P3.4 → Timer 0 external i/p (T_0)
- P3.5 → Timer 1 " (T_1)
- P3.6 → External data memory write strobe ($\overline{WR}$)
- P3.7 → " " read " ($\overline{RD}$)

Simple programs for I/O Port Programming

1) Write a program to receive data from port and send the same to port 1

```
        MOV  A, #0FFH   ; Make P0 as
        MOV  P0, A      ; input port
BACK:   MOV  A, P0      ; Read data from P0
        MOV  P1, A      ; send to P1
        SJMP BACK
```

2) Write a program to toggle all the bits of port 1 continuously.

```
        MOV  A, #55H    ; To make alternate bits ON & OFF
BACK:   MOV  P1, A              (0101 0101)
        ACALL DELAY
        CPL  A          ; To make alternate bits OFF & ON
        SJMP BACK               (1010 1010)
        END
DELAY:  MOV  R4, #255
H2:     MOV  R3, #55
H1:     DJNZ R3, H1
        DJNZ R4, H2
        RET
```

(or)

```
BACK:   MOV  A, #55H
        MOV  P1, A
        ACALL DELAY
        MOV  A, #AAH
        MOV  P1, A
        ACALL DELAY
```

Write a program to toggle (blink) bit P1.2 continuously. (9)

```
BACK: CPL  P1.2
      ACALL DELAY
      SJMP  BACK
DELAY: MOV  R1, #255
H1:   DJNZ R1, H1
      RET
```

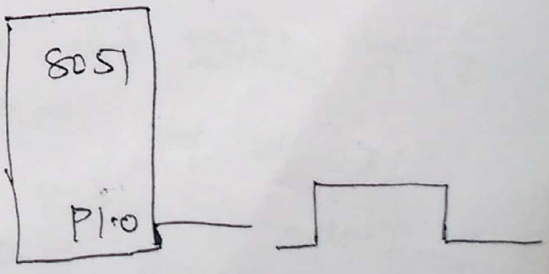
(OR)

```
BACK: SETB P1.2
      ACALL DELAY
      CLR  P1.2
      ACALL DELAY
      SJMP BACK
```

Write a program to create a square wave of 50% duty cycle on bit 0 of port 1

50% duty cycle means on and off states have same length.

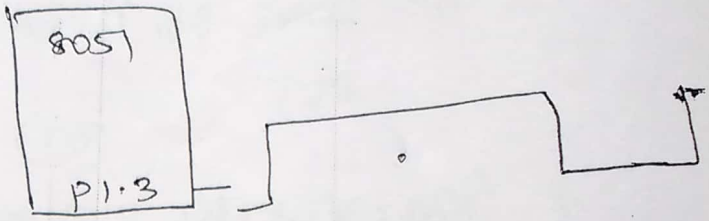
```
BACK: SETB P1.0   ; make P1.0 high
      ACALL DELAY
      CLR  P1.0   ; make P1.0 low
      ACALL DELAY
      SJMP BACK
DELAY: MOV R2, #255
H2:   MOV R3, #20
H1:   DJNZ R3, H1
      DJNZ R2, H2
```



5) Write a program to create a square wave 66% duty cycle on bit 3 of port 1

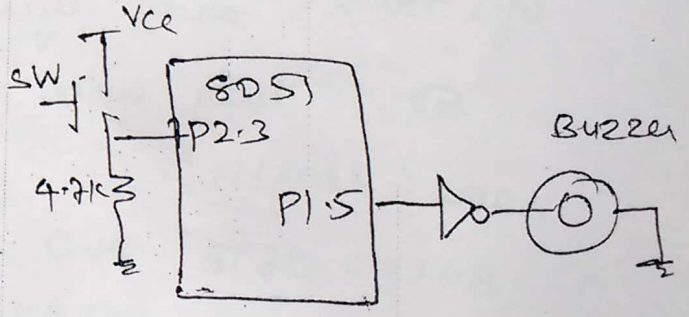
66% duty cycle means 'on' state is twice the off state.

```
BACK: SETB  P1.3
      ACALL DELAY
      ACALL DELAY
      CLR   P1.3
      ACALL DELAY
      SJMP  BACK
```



6) Assume that bit P2.3 is an input and represents the condition of oven. If it goes high, it means that oven is hot. Monitor bit continuously. Whenever it goes high, send a high to low pulse to port P1.5 to turn on buzzer

```
      SETB P2.3      ; make as i/p
here: JNB  P2.3,here ; monitor
      SETB P1.5      ; set P1.5
      CLR  P1.5      ; low pulse
                       to P1.5
      SJMP here
```



7) A switch is connected to pin P1.0 and an LED to pin P2.7. Write a program to get status of switch and send to LED

```
      SETB P1.0    ; make as i/p
AGAIN: MOV C,P1.0  ; read sw status to CF
      MOV P2.7,C   ; send to LED
```

BIT directive.

This is a widely used directive to assign bit addressable I/O and RAM locations. This allows a program to assign I/O or RAM bit at the beginning of program, making it easier to modify them.

EQU directive

This is used to assign addresses for the byte addressable locations.

8) An LED is connected to pin P1.7. Write a program to toggle LED forever

```
LED  BIT  P1.7  ; using bit directive
HERE: CPL  LED
      LCALL  DELAY
      SJMP  HERE
DELAY: MOV  R3, #255
here:  DJNZ  R3, here
       RET
```

9) A switch is connected to pin P1.7, Write a program to check the status of switch and make the following decision.

a) If SW=0, send "No" to P2

b) If SW=1, send "Yes" to P2

```
SW   BIT  P1.7
MYDATA EQU  P2
here: MOV  C,SW          ; read SW
      JC   OVER          ; go to over if SW=1
      MOV  MYDATA,#'N'   ; send 'N' to port 2
      MOV  MYDATA,#'O'   ; send 'O' to  "
      SJMP here
OVER: MOV  MYDATA,#'4'
      MOV  MYDATA,#'E'
      MOV  MYDAMA,#'S'
      SJMP here
      END
```

TIMERS

The 8051 has two 16 bit timer/counters. These can be programmed independently. There is a bit in TMOD SFR that specifies whether it is a timer or counter. If this bit is set, the timer/counter will work as a counter ~~x~~ if this bit is reset, the timer/counter will work as timer

Timer mode

When functioning as timer, the timer register (TH1 | TL1) for Timer 1 or (TH0 | TL0) for Timer 0 is incremented after every m/c cycle.

Counter mode

If it is required to count the number of occurrences of particular event, the timer/counter needs to be used in counter mode.

When used as an event counter, the registers are incremented whenever a 1 to 0 transition is sensed at T0 or T1 pins of 8051. The T0 & T1 pins are scanned during every m/c cycle.

Different Modes of Timer

The timers can function in four different modes

(i) Mode 0 – 13 bit timer

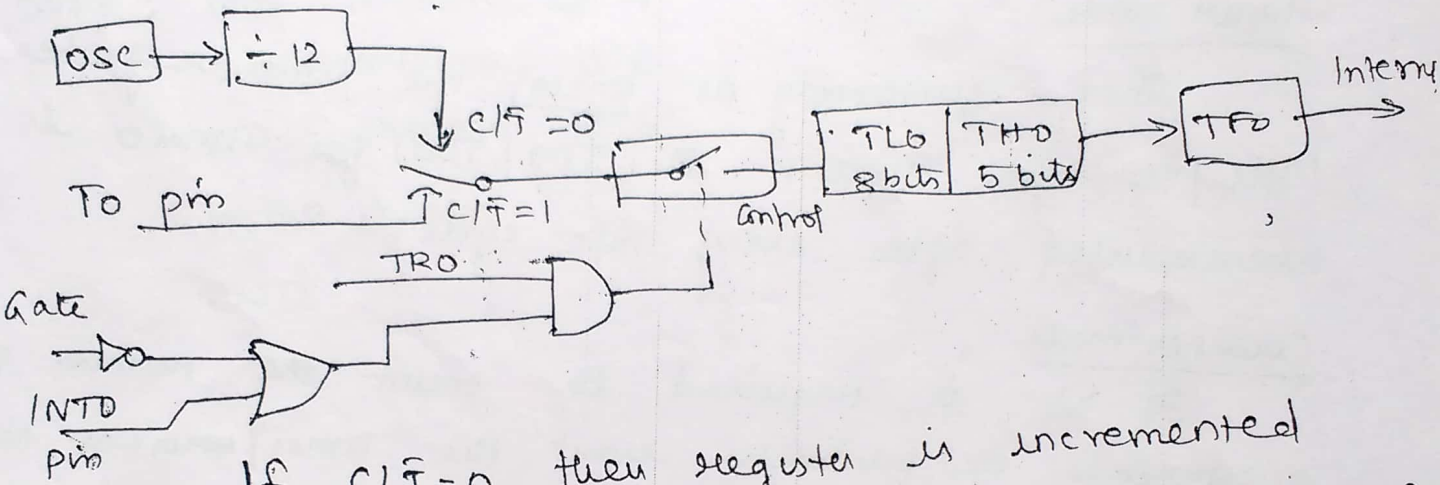
(ii) Mode 1 – 16 bit timer

(iii) Mode 2 → 8 bit timer with auto-reload

(iv) Mode 3 – Split Timer 0 as two 8 bit timers

MODE 0 :

In this mode, TL0 and TH0 for Timer0 (TL1, TH1 Timer1) are used as 13 bit register. (ie) 8 bits of TL0/TL1 and 5 lower most bits of TH0/TH1 are used for counting purpose. As the count rolls over from all 1's in the register to all 0's (ie max count = 1FFF), the interrupt (TF0 / TF1) in TCON SFR is set.



If C/T̄ = 0, then register is incremented after every m/c cycle (ie) at rate of 1/12th of osc freq) For register to be incremented, the control switch should be closed. There are many ways by which this logic can be implemented.

* Case 1 (Independent of external ctrl)

TR0 = 1

Gate = 0 & INT0 = 0/1

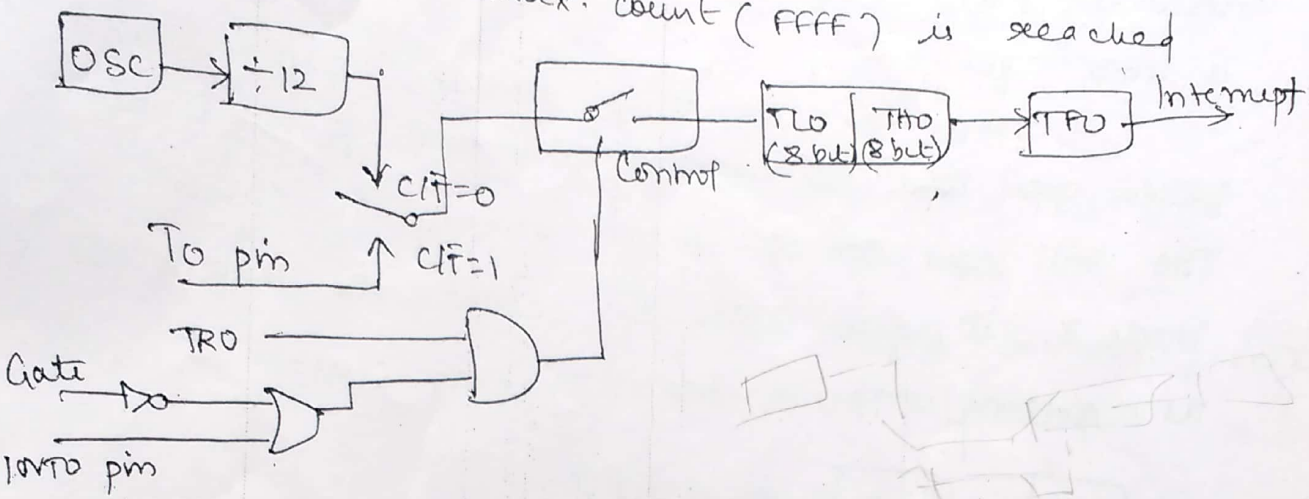
* Case 2 (dependent of external ctrl)

TR0 = 1

Gate = 1 & INT0 = 1

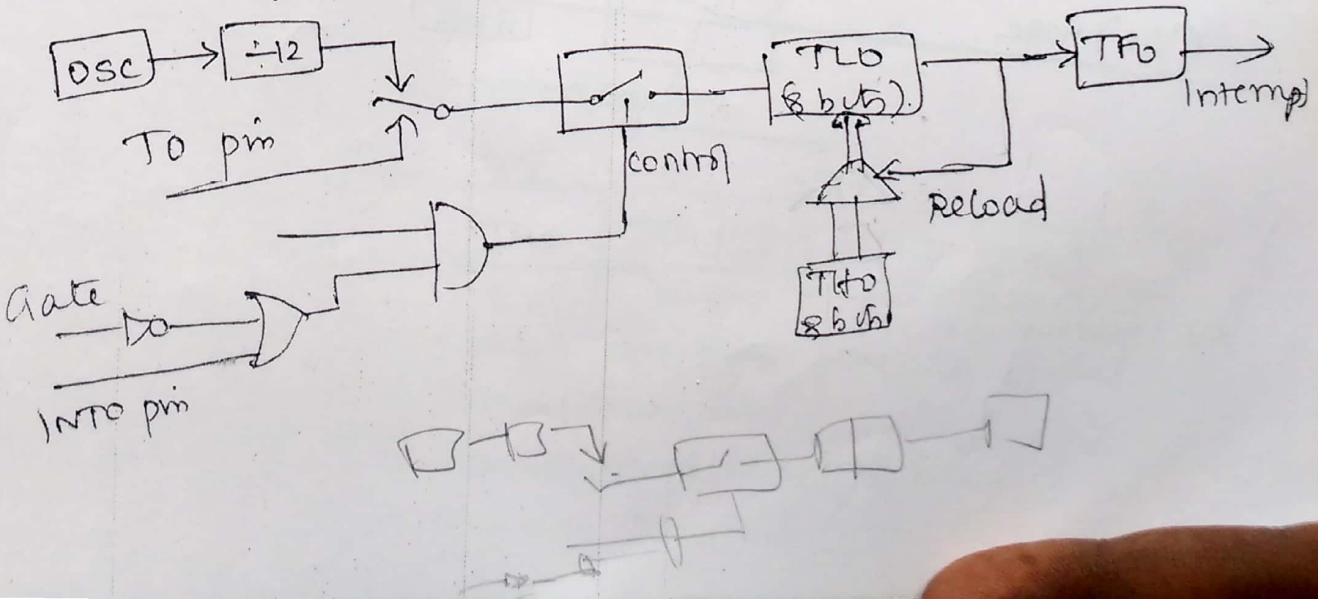
MODE 1:

This is similar to mode 0, except that in this, 16 bits (all) full TL0, TH0 for Timer 0 or (TL1, TH1 for Timer 1) are used for counting. The interrupt flag is set only when max. count (FFFF) is reached



MODE 2:

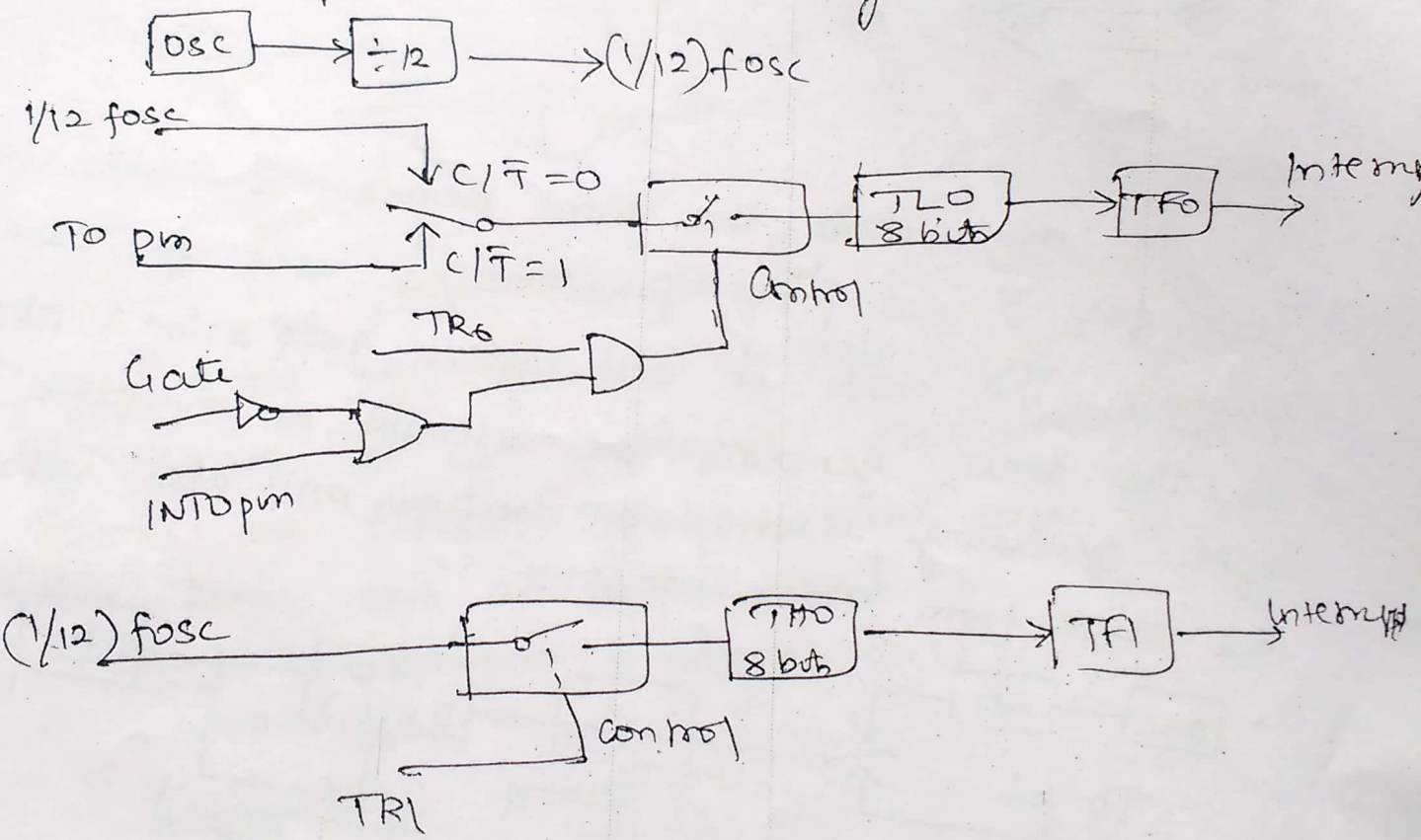
In this mode, the timer register is 8 bits wide. TL0 for Timer 0 (TL1 for Timer 1) is used for this purpose. This mode is also called auto reload mode as the timer generates an interrupt on overflow and after generating interrupt on overflow, will also reload the value from TH0 into TL0.



MODE 3:

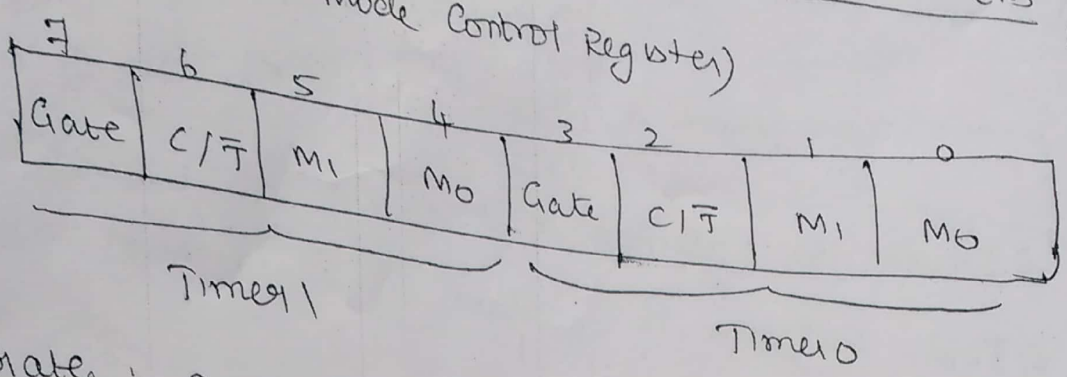
If Timer 0 is put into mode 3, then it acts as two split 8 bit counters. In this case, all Timer 0 control bits ($C/\bar{T}$, Gate, TR0, TF0, $\overline{INT0}$) are used by TL0 itself and TH0 register is locked in a timer function.

TH0 is counting machine cycles and has taken over the use of TR1 and TF1 from Timer 1. ∴, TH0 will now control Timer 1 interrupt. If Timer 1 is put in mode 3, it just holds the count. The effect is same as setting TR1=0, hence opening switch



Special function Registers (SFR's) used with TIMERS

(i) TMOD (Timer mode Control Register)



Gate : 0 → Timer controlled by TR1 / TR0 bits irrespective of INT0 / INT1 pin

1 → Timer control depend on INT0 / INT1 pin and also on TR1 / TR0 bits

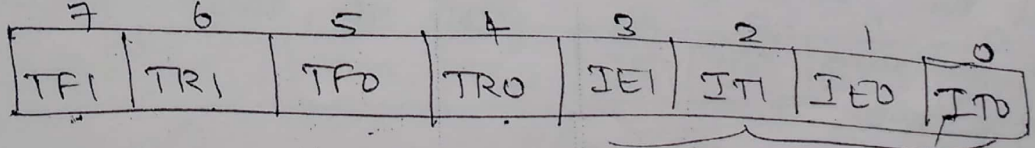
C/T : 0 → Timers (count no of m/c cycles (internal clk)

1 → Counter (count negative transitions at T0/T1 pins (ie) external clk).

M1, M0 : Specify the mode of timer / counter

M1	M0	Mode	Function
0	0	0	13 bit timer / counter
0	1	1	16 bit timer / "
1	0	2	8 bit with auto reload
1	1	3	Split Timer 0 into two 8 bit counters

ii) TCON : Timer Control Register



TF1 → Timer 1 overflow flag bit

TR1 → Timer 1 run bit

TF0 → Timer 0 overflow flag bit

TR0 → Timer 0 run bit

Finding values to be loaded into timer (Mode 1)

Assume that crystal freq, $f = 11.0592$ MHz

Time period $= \frac{1}{f/12} = 1.085\mu s$.

1) Divide the desired time delay by $1.085\mu s$

2) Find $65536 - n$, where n is decimal value from step 1

3) Convert the result from step 2 into hex, where yyxx is the hex value to be loaded into timer's register

4) Set TL = xx & TH = yy.

For Mode 2

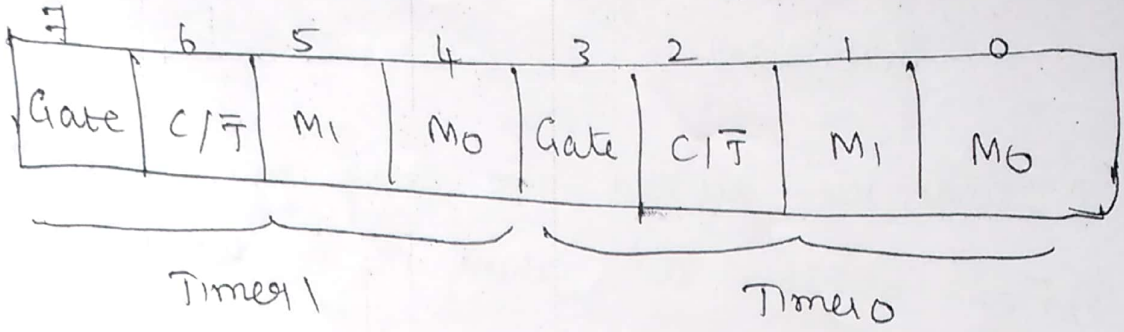
Find $256 - n$ instead of $65536 - n$ in the above and load the equiv hex value to TH register

Steps to program in mode 1

* Load TMOD value reg indicating which Timer is to be used and the mode
* Load registers TL and TH with initial count values
* Start the timer (TR0/TR1 = 1)
* Keep monitoring timer flag (JNB TFx, target). Get out of loop when $TF_x = 1$
* Stop the timer (TRx = 0)
* Clear the TF flag for next round
* Go back to step 2 to load TH and TL again

Special function Registers (SFR's) used with TIMERS

3. (i) **TMOD** (Timer mode Control Register)



Gate : 0 → Timer controlled by TR1 / TR0 bits irrespective of INT0 / INT1 pin

1 → Timer control depend on INT0 / INT1 pin and also on TR1 / TR0 bits

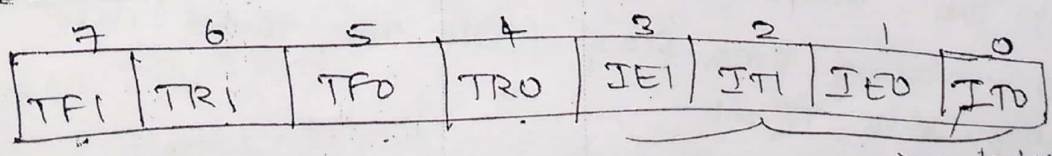
C/T : 0 → Timers (count no of m/c cycles internal clk)

1 → Counter (count negative transitions at T0/T1 pins of external clk).

M1, M0 : Specify the mode of timer/counter

M1	M0	Mode	Function
0	0	0	13 bit timer / counter
0	1	1	16 bit timer / "
1	0	2	8 bit with auto reload
1	1	3	Split Timer 0 into two 8 bit Counters

ii) **TCON** : Timer Control Register



TF1 → Timer 1 overflow flag bit

TR1 → Timer 1 run bit

TF0 → Timer 0 overflow flag bit

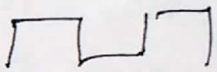
TR0 → Timer 0 run bit

Steps in mode 2

* Load TMOD reg
* Load reg TH alone (8 bit)
* Start the timer
* Keep monitoring timer flag
* When set, stop timer & clear flag
* Go back to step 4, since mode 2 is auto reload

Examples for Timer Programming

1) Write a program to generate square wave of 50% duty cycle on P1.5 bit. Use Timer 0, mode 1 to generate delay of 5ms.



TMOD

0 0 0 0 0 0 0 1 $\Rightarrow (01)_H$ Timer 0, Mode 1

To calculate values for Timer reg

Desired delay = 5 ms

$$\frac{5\,ms}{1.085\,\mu s} = 4608$$

$65536 - 4608 = 60928 \Rightarrow (EE00)_H$.

$TH_0 = (EE)_H$ $TL_0 = (00)_H$

Program

```
HERE: CPL  P1.5
      ACALL DELAY
      SJMP  HERE
```

```
DELAY:
        MOV  TMOD, #01H
        MOV  TL0, #00H
        MOV  TH0, #0EEH
        SETB TR0
AGAIN:  JNB  TF0, AGAIN
        CLR  TR0
        CLR  TF0
        RET
```

(OR)

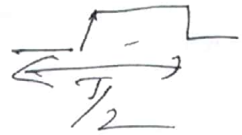
```
        MOV  TMOD, #01H
HERE:   MOV  TL0, #00H
        MOV  TH0, #0EEH
        CPL  P1.5
        ACALL DELAY
        SJMP HERE

DELAY:
        SETB TR0
AGAIN:  JNB  TF0, AGAIN
        CLR  TR0
        CLR  TF0
        RET
```

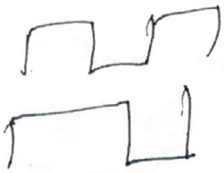
Write a program to generate a square wave (1:1) of 2 KHz frequency on pin P2.3 (Use Timer 1, mode 2; 66% duty cycle)

$$T = \frac{1}{2 \times 10^3} = 0.5 \text{ ms}$$

$$\frac{1}{2} \times 0.5 \text{ ms} = 0.25 \text{ ms}$$



$$65536 - \frac{0.25 \text{ ms}}{1.085 \mu s} \Rightarrow (6A)_H$$



TMOD

0 0 1 0 0000 $\Rightarrow$ (20) H

Program

```
        MOV  TMOD, #20H
        MOV  TL1, #00H
        MOV  TH1, #6AH
AGAIN:  SETB P2.3
        ACALL DELAY
        ACALL DELAY
        CLR  P2.3
        ACALL DELAY
        SJMP AGAIN

DELAY:
        SETB TR1
HERE:   JNB  TF1, HERE
        CLR  TR0
        CLR  TF0
        RET
```

3) Assume that clock pulses are fed into pin T1 write a program for counter 1 in mode 2 to count the pulses and display state of TL1 count on P2.

TMOD

0 1 1 0 0 0 0 0 $\Rightarrow (60)_H$

Program

```
        MOV  TMOD, #60H   ; Counter 1 mode 2
        MOV  TH1, #0      ; clear reg
        SETB P3.5         ; make T1 input
AGAIN:  SETB TR1          ; start
BACK:   MOV  A, TL1       ; get count TL1
        MOV  P2, A        ; display on port 2
        JNB  TF1, BACK
        CLR  TR1
        CLR  TF1
        SJMP AGAIN
```

SERIAL PORT

The 8051 µc has got all the circuitry in it for serial transmission. The RXD pin is used to receive input serially and TXD pin is used to transmit data serially.

The serial communication is full duplex, in that the 8051 can receive and transmit at same time. The receiving unit is buffered as well. Thus the reception of second byte of the frame data can start even before the first byte is received by CPU. Since the buffer is only one byte wide, the CPU must read previous byte before second byte is received, otherwise one of the bytes will be lost. The special function register SBUF is used for both receiving & transmitting the byte. A write into SBUF will initiate the process of transmission.

The serial port in 8051 can be configured into four different modes.

* Mode 0
* Mode 1
* Mode 2
* Mode 3.

MODE 0:

* Receives & transmits through RXD pin.
* TXD outputs the shift clocks
* 8 bits of data are transmitted / received
* While transmitting / receiving, LSB of the byte is sent / received first

* Baud rate is constant & is 1/12th of the SF oscillator frequency
* TI / RI interrupt flag is raised after Tx / Rx.

MODE 1:

* Transmits 10 bits of information through TXD and receives 10 bits of information through RXD
* The first bit is the start bit followed by 8 bits of data (LSB first) and then a stop bit (high).
* Once the stop bit is received, it means that the reception of one frame is complete & a byte of data has come.
* TI / RI interrupt flag is raised after Tx or Rx
* Baud rate is variable.

MODE 2:

* 11 bits are transmitted or received
  - 1 bit for start
  - 8 bits for data
  - 1 bit can be programmed (Parity bit).
  - 1 bit for stop
* Baud rate is programmable (1/32nd or 1/64th of osc freq)

Most commonly used mode

MODE 3:

This is the same as mode 2 except that it has variable baud rate

SFR's associated with Serial Port

(1) SCON (Serial control register).

Used to define operating modes.

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

SM1	SM0	Mode	Description	Baud rate
0	0	0	Shift reg	1/12th osc freq
0	1	1	8 bit UART	variable
1	0	2	9 bit UART	1/64 or 1/32 (SMOD=0)
1	1	3	9 bit UART	Variable.

SM2 → 1 → Multiprocessor Communication
0 → Single " "

REN → 1 → Enables serial reception
0 → Disables " "

TB8 → 9th bit of data to be transmitted

RB8 → 9th bit of " " received

TI → Transmit interrupt flag (set when Tx is comp
must be cleared by s/w

RI → Receive interrupt flag (set when Rx is complet
must be cleared by s/w

(2) PCON

7th bit alone is used here

Bit 7 → SMOD → 0 → Normal baud rate
1 → double "

Examples for Serial Port Programming

Finding Baud Rate values to be loaded into Timer reg

eg) To get baudrate of 9600 Hz

$$\frac{28800}{9600} = 3$$

(28800 → UART freq)

$\therefore$ TH1 = -3 (or) $(FD)_H$

For 4800 Hz baudrate

$$\frac{28800}{4800} = 6$$

TH1 = -6.

eg) 1) Write a program for 8051 to transfer letter 'A' serially at 4800 baud rate continuously.

Soln:

```
        MOV   TMOD, #20H   ; Timer 1, mode 2
        MOV   TH1, #-6     ; 4800 baud rate
        MOV   SCON, #50H   ; Serial port Mode 2 enable
                           ; start timer 1
        SETB  TR1
AGAIN:  MOV   SBUF, #'A'
here:   JNB   TI, here
        CLR   TI
        SJMP  AGAIN
```

Write a program to transfer the message "YES" serially at 9600 baud rate continuously. (18)

Soln:

```
        MOV  TMOD, #20H  ; Timer 1, mode 2
        MOV  TH1, #-3    ; 9600 baud
        MOV  SCON, #50H  ; Serial port mode 2
        SETB TR1
AGAIN:  MOV  A, #'Y'
        ACALL TRANS
        MOV  A, #'E'
        ACALL TRANS
        MOV  A, #'S'
        ACALL TRANS
        SJMP AGAIN
TRANS:  MOV  SBUF, A
here:   JNB  TI, here
        CLR  TI
        RET
```

3) Write a program to receive bytes of data serially and put them in P1. Set baudrate at 4800 Hz

Soln:

```
        MOV  TMOD, #20H
        MOV  TH1, #-6
        MOV  SCON, #50H
        SETB TR1
here:   JNB  RI, here
        MOV  A, SBUF
        MOV  P1, A
```

INTERRUPTS

The 8051 has five interrupt sources. other than RESET, Each of these interrupts can be programmed independently. They are

Priority level	Source	Description
(Highest)	RESET	To reset processor
	$\overline{INT0}$	External request from P3.2 pin
	Timer 0	Overflow from Timer 0 activates interrupt request flag TF0
	$\overline{INT1}$	External request from P3.3
	Timer 1	Overflow from Timer 1 activates interrupt request flag TF1
(Lowest)	Serial port	Completion of transmission or reception of serial frame activates the flags TI or RI

Each of these interrupt sources can be individually enabled or disabled by clearing a bit in the special function register IE except Reset (non-maskeable).

All these sources can be programmed to a high priority level or low priority level by setting or clearing bits in the IP (Interrupt priority) register.

All these interrupts are seperately scanned during each machine cycle. The processor will go to the interrupt source routine (ISR) in the next machine cycle, provided it is not blocked by any of the following conditions.

*) An interrupt of equal or higher priority level is already in progress.

*) No interrupt request will be responded to until the instruction in progress is completed

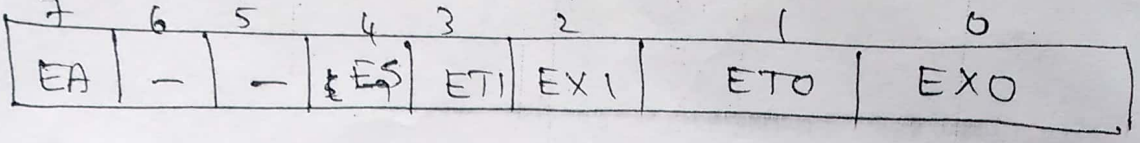
ISR locations

Source	location
RESET	0000H
$\overline{INT0}$	0003H
T0	000BH
$\overline{INT1}$	0013H
T1	001BH
Seual port	0023H

Before going to these ISR locations, it stores the program counter value in stack, so that when RETI instruction of ISR is executed, the processor can retuen to the location where the interrupt took place.

Special function registers associated with interrupts

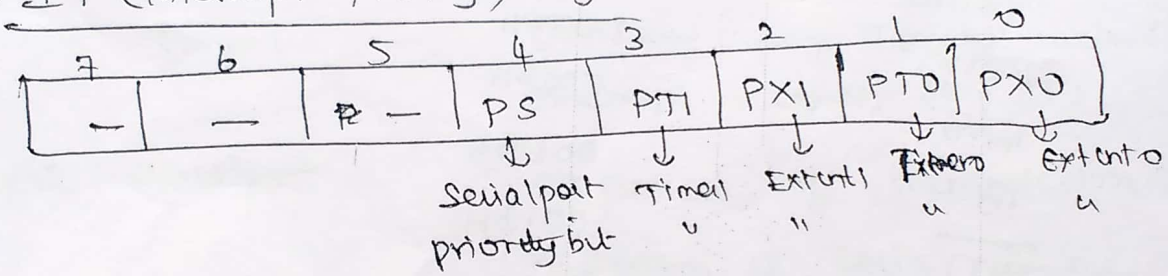
1) IE (Interrupt Enable) Register



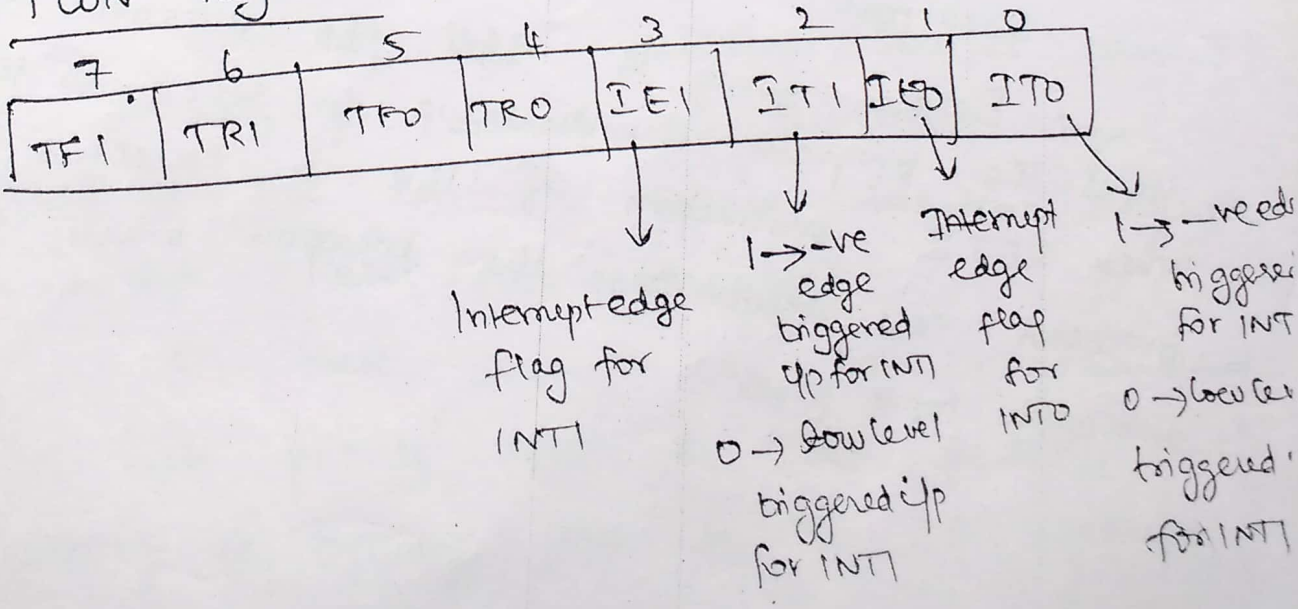
EA → 1 – Enables all interrupts
0 → Disables "

ES → Enables / disables serial port interrupt
ET1 → " Timer 1 "
EX1 → " external interrupt 1
ET0 → " Timer 0 "
EX0 → " External interrupt 0.

2) IP (Interrupt Priority) Register



3) TCON Register



Examples for Interrupt Programming

1) External Interrupt Programs

(i) Assume that INT1 pin is connected to a switch that is normally high. Whenever it goes low, it should turn on an LED. The LED is connected to P1.3 and is normally off. When it is turned on, it should stay on for a fraction of second. As long as the switch is pressed low, the LED should stay ON.

Soln:

```
        ORG   4100H
MAIN:   MOV   IE, #84H  ; enable INT1
here:   Sjmp  here
        END

        ORG   0013H     ; INT1 ISR
ISR:    SETB  P1.3      ; turn on LED
        MOV   R3, #255
here:   DJN2  R3, here  ; stay on for a second
        CLR   P1.3
        RETI
```

ii) Write a program to count the no of pulse (falling edge) connected to INT0 pin (P3.2) and display the count on P2.

```
        ORG   4100H
MAIN:   MOV   IE, #81H  ; enable INT0
```

```
        MOV   R4, #00H    ; Initialise count reg
back:   MOV   A, R4
        MOV   P2, A
        Sjmp  back
        END

        ORG   0003H       ; ISR for INT0
ISR:    INC   R4
        RET I
```

TIMER INTERRUPT PROGRAMS

3) Write a program that continuously get 8 bit data from P0 and sends it to P1 while simultaneously creating square wave of 200 μs period on P2.1. Use Timer 0 (Mode 2) to create square wave

$$TH0 = 100\,\mu s / 1.085\,\mu s = 92 \Rightarrow (A4)_H$$

```
        ORG   4100H
MAIN:   MOV   TMOD, #02H  ; Timer 0, mode 2
        MOV   P0, #0FFH   ; Make P0 input port
        MOV   TH0, #A4H   ; set time ON value
        MOV   IE, #82H    ; Enable Timer 0 interrupt
        SETB  TR0         ; Start timer 0
BACK:   MOV   A, P0       ; data from P0
        MOV   P1, A       ; sent to P1
        SJMP  BACK        ; continuously
```

```
ORG 000BH   ; ISR for Timer1
CPL P2.1    ; create square wave
RETI
```

NOTE: for Timer mode 1, in ISR, reload values of TL0, TH0 again

SERIAL INTERRUPT PROGRAMS

4) Write a program in which the 8051 gets data from P_1 and sends to P_2 continuously while incoming data from serial port is sent to P0. Set the baudrate at 9600 (Assume XTAL = 11.0592 MHz)

Soln:

```
        ORG 4100H
MAIN:   MOV P1, #0FFH   ; make P1 an input port
        MOV TMOD, #20H  ; Timer1 Mode 2
        MOV TH1, #0FDH  ; 9600 baud rate   (FD = -3)
        MOV SCON, #50H  ; enable serial port
        MOV IE, #90H    ; enable serial port interrupt
        SETB TR1
BACK:   MOV A, P1
        MOV P2, A
        SJMP BACK

        ORG 0023H       ; serial ISR
        JB TI, here
        MOV A, SBUF     ; receive data
```

```
        MOV  P0, A      ; send incomming data to P0
        CLR  RI
        RETI
here:   CLR  TI
        RETI
        END
```