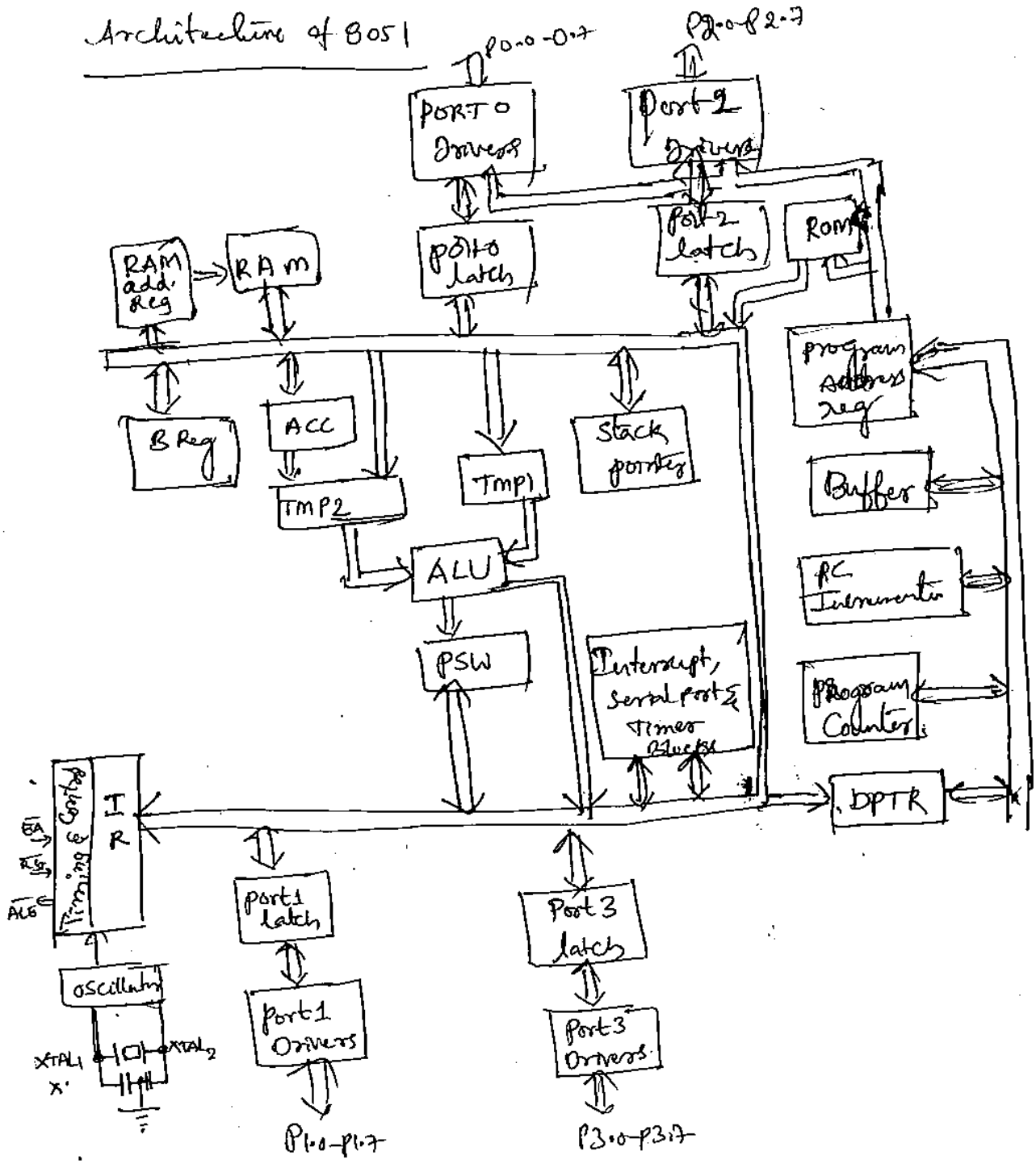
Architecture of 8051



Accumulator (ACC): The ACC (or) 'A' acts as an operand register. This is either be implicit or specified in the instruction.

The ACC Register address is alloted in the on-chip Special function register bank.

B Register: It is used to store one of the operands for multiply & Divide ~~operat~~ Instructions.

Program status word: This set of flags contains the status information & is considered as one of the SFR.

Stacks pointer: This register contains 8 bit stack top address. The stack may be defined anywhere in the on-chip 128 byte RAM. After Reset, the SP register is initialized to 07.

DPTR: This 16-bit Register contains a higher byte (DPH) & the lower byte (DPL) of a 16-bit external data RAM address. It is accessed as a 16-bit register or two 8-bit registers as specified above.

Port 0 to 3 latches & Drivers: These four latches & driver pairs are alloted to each of the four on-chip I/O ports. These latches have been allotted address in the SFR bank. Using allotted address User can communicate with these ports. These are P0, P1, P2, P3.

Timer Register: There are two timer registers T0, T1. These two are 16-bit registers. These can accessed as two 8 bit registers.

Control registers: The Special function registers IP, IE, TMOD, TCON, SCON, & PCON contain control & status information for the interrupts, timers/counters & serial port.

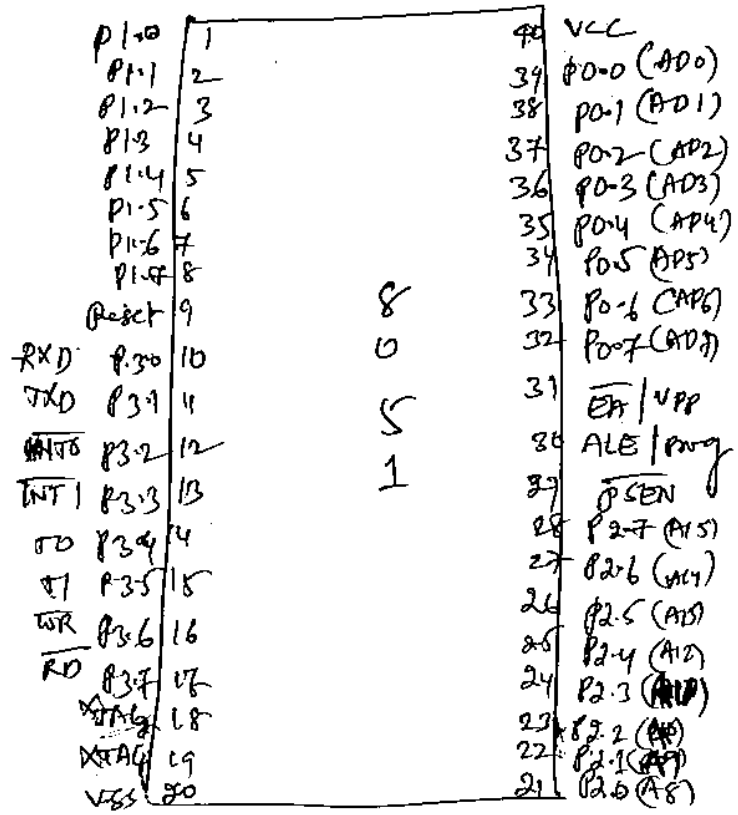
Timing & control unit: This unit derives all the necessary timing & control signals required for the internal operation of the circuit. It also derives control signals required for controlling the external system bus.

Oscillator : This circuit generates the basic timing clock signal for the operation of the circuit using Crystal oscillator

Instruction register : This register decodes the opcode of an instruction to be executed & gives the information to the timing & control unit to generate the necessary signals for the execution of the instruction.

ALU : The arithmatic and logic unit performs 8 bit arithmatic and logical operations over the operands held by the temp. registers TMP1 & TMP2. Users can not access these temporary registers.

pin diagram of 8051



Register set of 8051

Registers of 8051.

A, B, PSW, P0, P1, P2, P3, IP, IE, TCON, SCON,

SP, DPH, DPL, TMOD, TH0, TL0, TH1, TL1, SBUF, PCON.

- And also 4 banks of general purpose registers

Bank 0	→ R0 – R7
Bank 1	R0 – R7
Bank 2	R0 – R7
Bank 3	R0 – R7

- General purpose reg. are stored in the on chip RAM. Starting 32 bytes are reserved for this (0000 to 001FH).

- Addresses of the remaining registers are available in the Special Function Bank.

PSW : (Program status word)

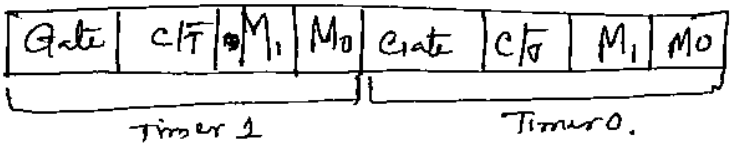
CY	AC	F0	RS1	RS0	OV	–	P

RS1	RS0	Reg. Bank	Address
0	0	Bank 0	00 – 07H
0	1	Bank 1	08 – 0FH
1	0	Bank 2	10H – 17H
1	1	Bank 3	18H – 1FH

OV – overflow flag

P – parity flag

TMOD Format

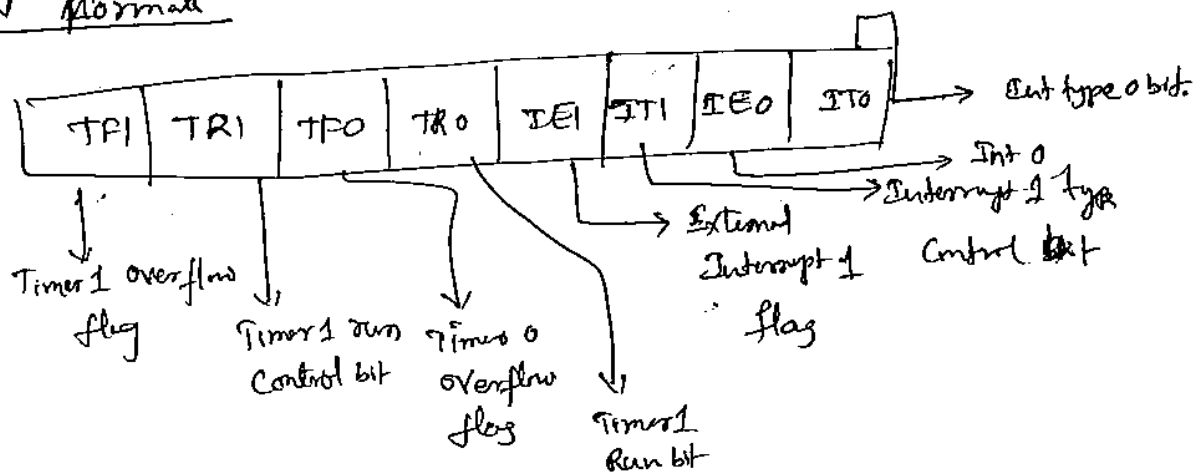


Gate : When TRx (in TCON) is set & Gate = 1, Timer/Counter x will run only while INTx pin is high, when Gate=0, Timer/Counter x will run only while TRX = 1.

C/T → Timer/Counter selector.
It is zero select the timer operation otherwise Counter.

M1 M0	operation
0 0	Mode 0, 13 bit Timer
0 1	Mode 1, 16 bit Timer/Counter
1 0	Mode 2 8 bit auto Reload Timer
1 1	Mode 3. (Timer 0) TL0 is an 8 bit Timer/Counter Controlled by the Timer 0 control bits. TH0 is an 8 bit timer & controlled by Timer 1 control bits.
1 1	Mode 3 - (Timer 1) - Timer/Counter 1 stopped.

TCON Format

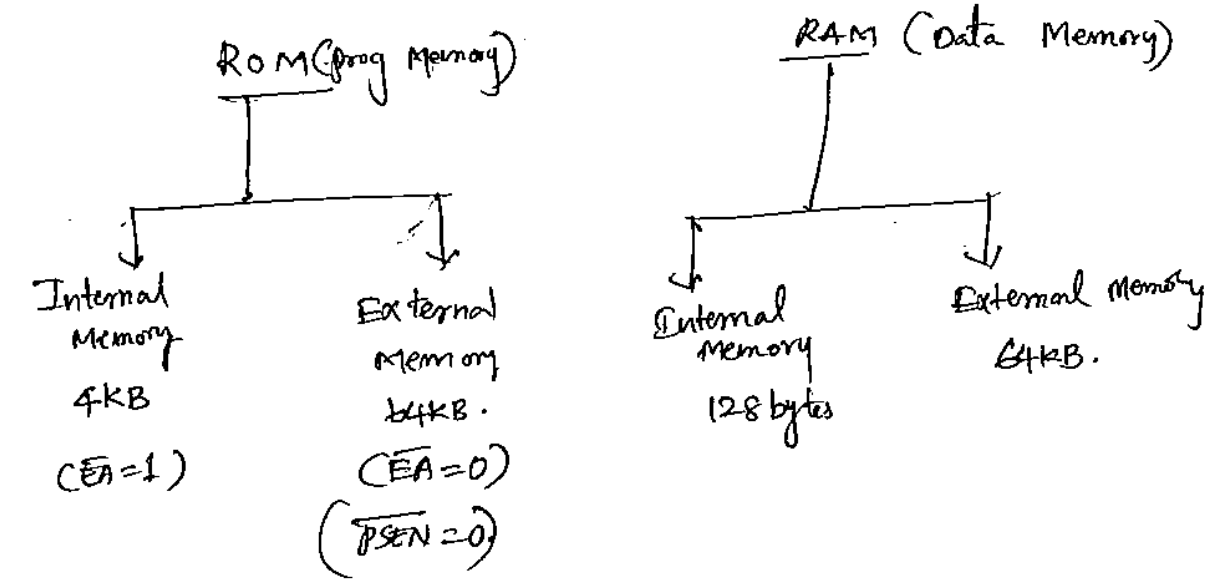


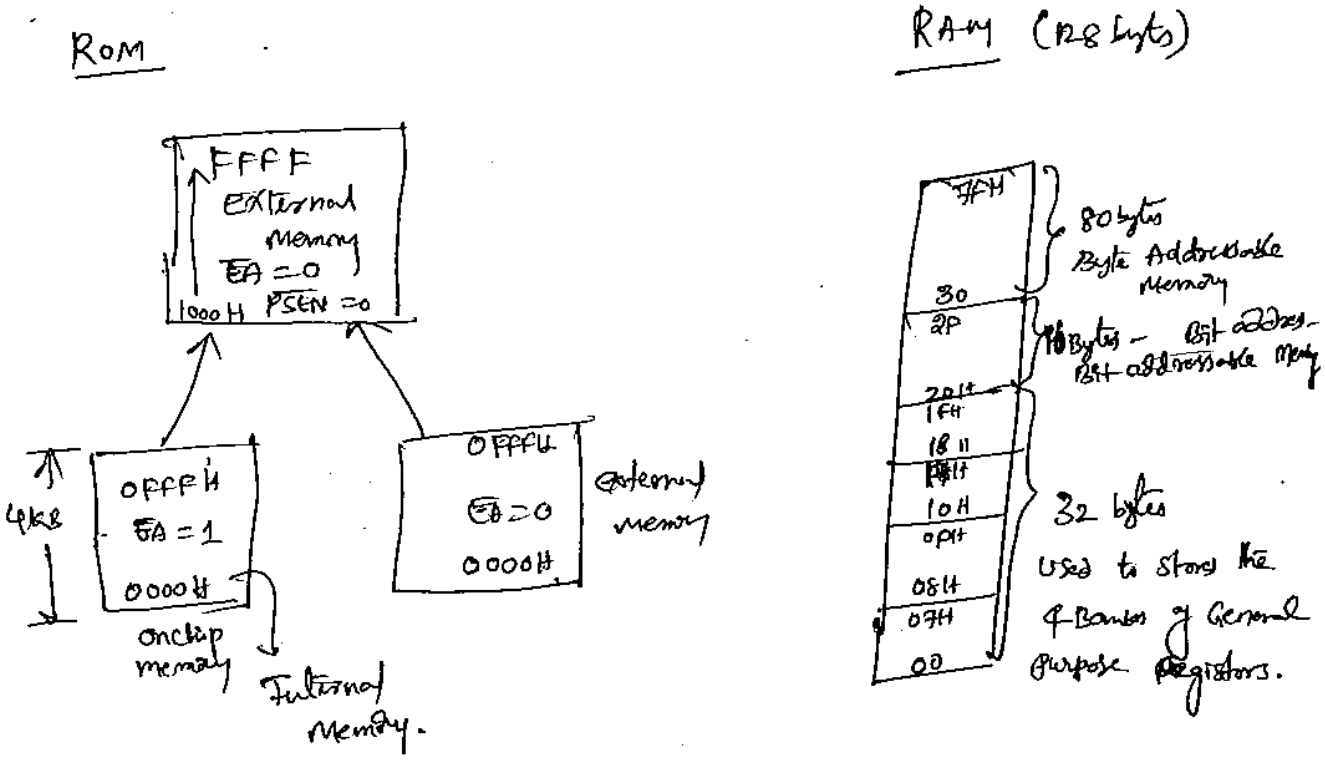
TF1 → this is set by H/w when Timer/Counter 1 overflows & is cleared by H/w as processor vectors to the interrupt service routine.

TR1 → This is set/cleared by S/w to run Timer/Counter 1 ON/OFF.

IE1 → This is set by H/w when external interrupt edge is detected & is cleared by H/w when the interrupt is processed.

Memory organization





Internal RAM

1. First 32 bytes from address 00H to 1FH are reserved for 4 banks of 32 general purpose registers.
2. Next 16 bytes that is from 20H to 2F are bit addressable memory. An addressable bit may be specified by its bit address of 00H to 7FH. For Ex. the bit address 4FH is also a bit of 7 of the byte address 29H. Addressable bits are useful when the prog. need only remember a binary event. (Switch ON (or) light OFF. etc).

Interrupts 8051

8051 five sources of interrupts.

Interrupt source	priority
IE0 — External (INT0)	Highest
TF0 (Timer 0)	
IE1 (External INT 1)	
TF1 (Timer 1)	
RI = TI (Serial port)	Lowest

Interrupt Enable register

EA	-	ET2	ES	ET1	EX1	ET0	EX0

→ If EA=0, no interrupt will be acknowledged.
EA=1, each interrupt source is enabled or disabled by setting (or) clearing its enable bit.

→ ET2 — This enables Timer 2 overflow (or) [8052]

- ES — This enables (or) disables the serial port interrupt.
- ET1 — This enables (or) disables Timer 1 overflow interrupt.

Interrupt priority Register:

-	-	PT2	PS	PT1	PX1	PT0	PX0

PT2 — This defines the Timer 2 interrupt priority level.
PS — This defines the serial port interrupt priority level.
PT1/PT0 → This defines the Timer 1/Timer 0 interrupt priority level
PX1/PX0 — This defines the $\overline{INT1}$ / $\overline{INT0}$ priority level.

Addressing Modes:

(1) Direct addressing mode
(2) Indirect addressing mode
(3) Register addressing mode
(4) Register specific (Register implicit addressing mode)
(5) Immediate addressing mode.
(6) Indexed addressing mode.

(1) Direct addressing Mode:

In this addressing mode, the 8 bit address of an operand are specified is specified directly in the instruction.

Ex MOV R0, 89 H.

(2) Indirect addressing mode:

In this mode, the 8 bit address of an operand is stored in register & the register, instead of 8 bit address, is specified in the instruction.

ADD A, @R0.

(3) Register Addressing mode: Specify the operand my means of any register.

Ex: MOV A, R0.
MOV A, R1.

(4) Immediate Addressing mode:

Specify the data directly in the instructions.

Ex MOV A, #50H.

(3) Instruction set

External data MOVe Instruction :

MovX A, @Rp ; copy of the contents of the external address in Rp to A

MovX @DPTR, A Copy data A to the 16 bit external address in DPTR.

MovX @R0, A ; copy data from A to the 8 bit address in R0.

Code memory Read only data moves

MOVC A, @A+DPTR ; Copy the code byte from address found by adding of A & DPTR to A.

MovC A, @A+PC ; Copy the code byte from address found by adding of A and the PC to A.

Push & pop instructions :-

push add ; Increment SP ; Copy the data in address to the internal RAM address contained in SP.

POP add ; copy the data from the internal RAM address contained in SP to add ; decrement the SP.

Data exchanges :

XCH A, Rn ; Exchange the data bytes b/w reg Rn and A.

XCH A, add ; Exchange the data bytes b/w add And A.

XCHD A, @Rp ; Exchange the lower nibble in A & the add. in Rp.

Byte level Logical Operations :

ANL A, #n ; AND each bit of A with the same bit of immediate number n ; & put the result in A.

ORL A, #n ; OR each bit of A with the same bit of immediate number n ; & put the result in A.

XRL A, #n ; XOR each bit of A with the same bit of immediate number n ; & put the results in A.

CLR A ; clear each bit of the A register to 0.
CPL A ; complement each bit of A ; every 1 becomes a 0 ; & each 0 becomes a 1.

Bit level logical operations.

ANL C,b ; AND C and the addressed bit ; put the result in C.
ANL C, /b AND C and the compliment of the addressed bit. & put the result in C & the addressed bit is not altered.
ORL C,b OR C and the addressed bit ; put the result in C.
CLR b Clear the addressed bit is 0.
MOV C,b : copy the addressed bit to the C flag.
SETB C Set the c flag to 1.
SETB b Set the addressed bit to 1.

Rotate and SWAP operation Instructions.

RL A ; Rotate A register one bit position to the left.
RLC A Rotate the A register & the carry flag as ninth bit, on bit position to the left.
RR A ; Rotate A register one bit position to the right.
RRC A ; Rotate A register & Carry flag as ninth bit, one bit position to right ie bit A0 to C, C to A7, A7 to A6, A6 to A5 etc.
SWAP A ; Interchange the nibble of register A.
ie put the higher nibble in the low nibble position & the lower nibble in the high nibble position.

Arithmatic Instructions.

ADD A, #n ; Add A & the immediate number n; put the sum in A.

SUBB A, #n ; Subtract immediate value from A and the result is stored into A.

MUL AB ; Multiply A by B. put the low order byte of the result in A & put the high order byte in B.

DIV AB ; Divide by A by B ; put the integer part Quotient into A & integer part of remainder into B.

INC A ; Add 1 to the A reg.

DEC A Subtract '1' to the A reg.

DA A Adjust the sum of two packed BCD numbers found in A register, leave the adjusted number in A.

Jump Instructions

JC radd ; Jump to relative address if the carry is set to 1.

JB b,radd ; Jump to relative address if the addressable bit is set to 1.

JNB b,radd ; " " if the addressable bit is reseted.

JBC b,radd ; " " if the addressable bit is set & clear the addressable bit to 0.

CJNE A,add, radd ; compare the contents of A reg with the contents of the direct address; if they are not equal, then jump to the relative address. Set the carry flag 1 if A is less than the contents of the direct address.

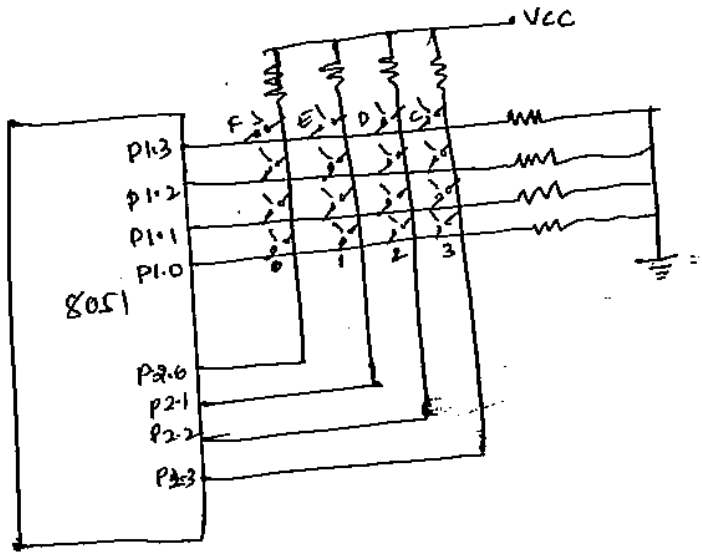
CJNE A, #n,radd.

DJNE Rn, add ; Decrement the reg. Rn by 1 & Jump to the relative address if the result is not zero. No. flags are effected.

Differences between Microprocessors & Microcontrollers

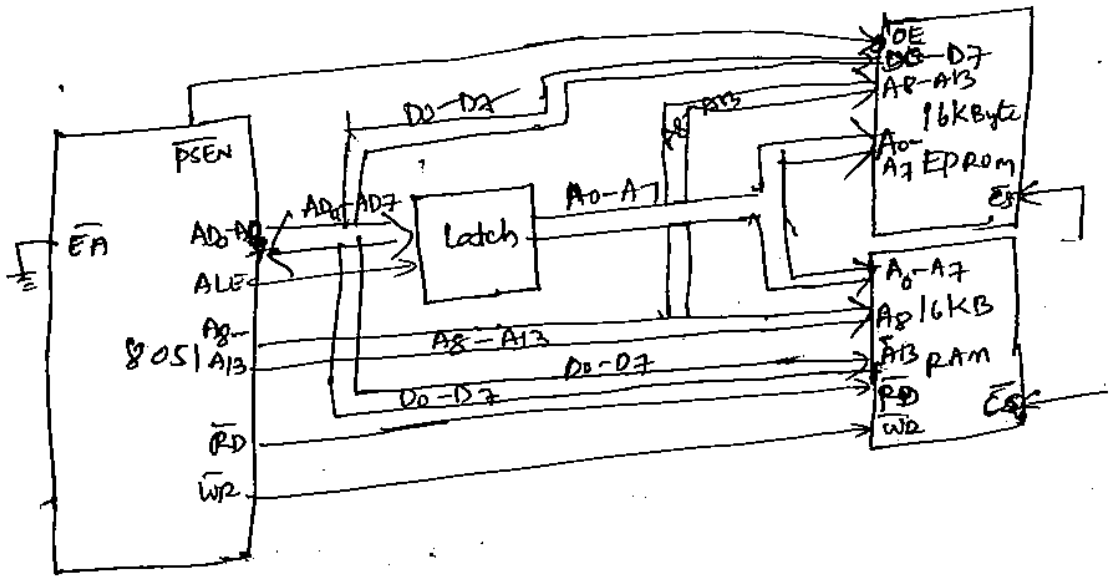
Micro processors	Micro Controllers
(1) Micro processors does not have On chip memory, timers, I/O ports.	(1) Micro Controllers has on chip memory, timers & IO ports.
(2) It has one (or) two bit handling Instructions	(2) It has more number of bit handling Instructions.
(3) Access time for memory & IO is more.	(3) Access time is less.
(4) It requires more hardware	(4) It requires less H/w.
(5) More flexible.	(5) Less Flexible.
(6) Less Number of pins are multiplexed.	(6) More No. of Pins are Multiplexed.

Key Board Interfacing with 8051




```
        Mov P2, # FFH
Go:     Mov P1, #00H
        MOV A, P2
        ANL A, #0FH
        CJNE A, #0FH, Go
        ACALL Delay
Go2:    MOV P1 ; #00H
        MOV A, P2
        ANL A, #0FH
        CJNE A, #0FH Go1
        SJMP Go2
Go1:    ACALL Delay
        MOV P1, # 1111 1110B (FEH)
        MOV A, P2
        ANL A #0FH
        CJNE A, #0FH, ROW-0
        MOV P1, # 1111 1101 B
        MOV A, P2
        ANL A, #0FH
        CJNE A, #0FH, ROW-1
        MOV P1, # 1111 1011 B
        MOV A, P2
        ANL A, #0FH
        CJNE A, #0FH, ROW-2
        MOV P1, # 1111 0111 B
        MOV A, P2
        ANL A, #0FH
        CJNE A, #0FH, ROW-3
        LJMP Go2.
ROW-0:  MOV DPTR, #KCode0
        SJMP FIND
ROW-1:  MOV DPTR, #KCode1
        SJMP Find
ROW-2:  MOV DPTR, #KCode2
        SJMP Find
ROW-3:  MOV DPTR, #KCode3
        SJMP Find.
Find:   RRC A
        JNC Bit
        INC DPTR
        SJMP Find
Bit:    CLR A
        MOVC A, @A+DPTR
        MOV P0, A
        SJMP Go
ORG 0000H
KCode0: DB '0', '1', '2', '3'
KCode1: DB '4', '5', '6', '7'
KCode2: DB '8', '9', 'A', 'B'
KCode3: DB 'C', 'D', 'E', 'F'
END.
Delay:  MOV R1, # FFH
```

Memory Interfacing of 8051



LCD interfacing

If RS = 0; Command (Code) register is selected, allowing the User to send a Command such as clear display, cursor at home.

RS = 1, data reg. is selected, allowing the User to send data to be displayed on the LCD.

R/W = 0 → write the information to the LCD.
R/W = 1 → Read the information from it.

E = Enable the LCD.

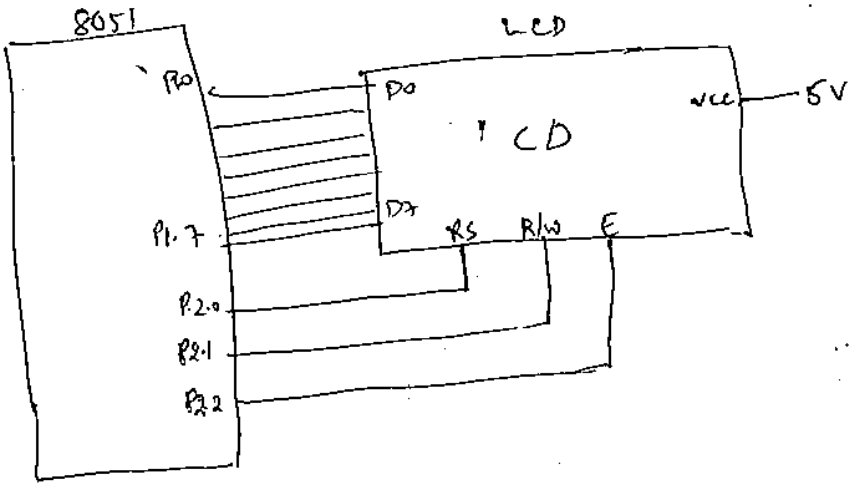
D_0–D_7 → Used to send the information to the LCD (or) read the contents of the LCD's internal reg.

LCD Command Codes

01 → clear display screen.
02 → Return home.
04 — Shift Cursor to left
06 — Shift Cursor to Right.
05 — Shift display right
07 — Shift display left.
08 — Display off, Cursor off
0A — Display off, Cursor on
0C — Display on, Cursor off
0E — Display on, Cursor blinking
0F — Display ON, Cursor blinking

10 — Shift Cursor position to left
14 — Shift Cursor position to Rt
18 — Shift entire display to left
1C — Shift entire display to Rt
80 — Force cursor to beginning of 1st line
C0 — Force " " 2nd line
38 — 2 lines & 5×7 matrix.

Interfacing diagram



```
ORG 0000H
MOV A, #38H   ; Initialize LCD 2 lines, 5x7 matrix.
ACALL COMMWRT ; Call Command Subroutine.
ACall delay.
MOV A, #0EH ; Display ON, Cursor ON.
ACALL Command
ACALL Delay.
MOV A, #01  ; clear LCD
ACALL Command
ACALL Delay
MOV A, #06H ; Shift Cursor right.
ACALL Command
ACALL Delay.
MOV A, #80H ; Cursor at line 1, pos 0
ACALL Command
ACALL Delay.
MOV A, #'H', ; Display letter H.
ACALL Display
ACALL delay
MOV A, #'E' ;
ACALL Display
ACALL Delay.

again: SJMP again

Command: MOV P1, A ; Copy reg A to Port A
         CLR P2.0   ; RS=0, for command.
         CLR P2.1   R/W=0, for write
         SETB P2.2  ; E=1 for high pulse
         ACALL delay
         CLR P2.2   ; E=0, for H to L pulse.
         RET.

Display: MOV P1, A  ; Copy reg A to port A
         SETB P2.0  ; RS=1, for data
         CLR P2.1.  R/W=0 for write
         SETB P2.2  E=1
         ACALL Delay
         CLR P2.2   E=0 for H to L pulse.
         RET.

Delay: MOV R3, #50
here1: MOV R4, #200
here: DJNZ R4, here
      DJNZ R3, here1
```

Serial Data Communication

SCON is used to controls the data Communication & SBUF is used to hold the data, PCON Controls the data rates.

The serial data flags in SCON, TI & RI are set when ever a data byte is transmitted (TI) or received (RI).

Data transmission:

Transmission of serial data bits begins any time data is written to SBUF. TI is set to 1 when the data is transmitted & signifies that SBUF is empty. & that another data byte can be sent.

Data Reception:

Reception of serial data will begin if the receive enable bit (REN) bit in SCON is set to 1 for all modes. In addition, for mode 0 only RI must be cleared to 0. Receiver Interrupt flag (RI) is set after data has been received in all modes.

SCON Register format

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0	SM1	mode	Description	Baud Rate
0	0	0	Shift register	fosciliator/12
0	1	1	8-bit UART	variable
1	0	2	9 bit UART	f/32 (or) f/64
1	1	3	9 bit UART	Variable.

SM2 → This enables the multiprocessor communication feature in mode 2 & 3.

In mode 2 (or) 3, if SM2 = 1 & then RI will not be activated, if the received 9th bit (RB8) is 0.

In mode 1, if SM2 = 1, then RI will not be activated, if a valid stop bit was not received.

In mode 0, SM2 is should be 0.

REN - 1: Receiving Enabled
=0 ; Receiving disabled.

TB8 - This selects 9th bit that will be transmitted in modes 2 & 3.

RB8 - This is 9th data bit that was received in mode 2 & 3.

TI → Transmit interrupt flag - this is set by H/w at the end of the 8th bit time in mode 0 (or) at the begining of the stop bit in other modes. This is must be cleared by S/w.

RI - Receive interrupt flag - This is set by H/w at the end of the 8th bit time in mode 0 (or) Half way through the stop bit time in other modes excepting the case where SM2 is set. This must be cleared by S/w.

PCON

SMOD	—	—	—	GF1	GF0	PD	IDL

SMOD = 1 = Double baud rate is selected for timer 1 in mode 1,2,3
SMOD = 0 = same Baud rate of timer 1.

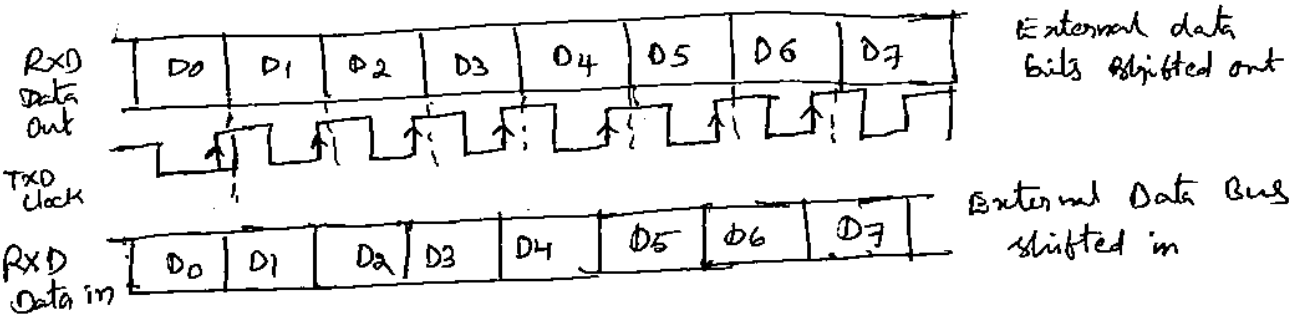
GF1 & GF0 → General purpose user defined flags.
PD = 1 → power down mode is selected
IDL = 1 → Idle mode is selected.

Serial Data transmission modes

Mode 0 : Shift Register mode :

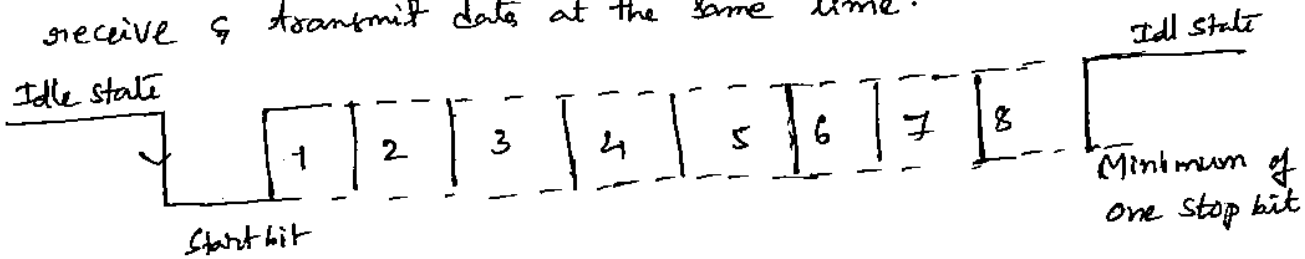
Setting bits SM0, SM1 in SCON is 00 Configures SBUF to receive or transmit 8 bits using Pin RXD for both functions. Pin TXD is connected to the internal shift frequency pulse source to supply the pulses to external circuits.

When transmitting, data is shifted out of RXD, the data changes on the falling edge (or) one clock pulse after the raising edge of the o/p TXD shift clock.



Mode 1 (Standard UART)

SBUF becomes 10 bit full duplex receiver / Transmitter that may receive & transmit data at the same time.



Mode 1 Baud rates :

Timer 1 is used in timer mode 2, then Baud rate

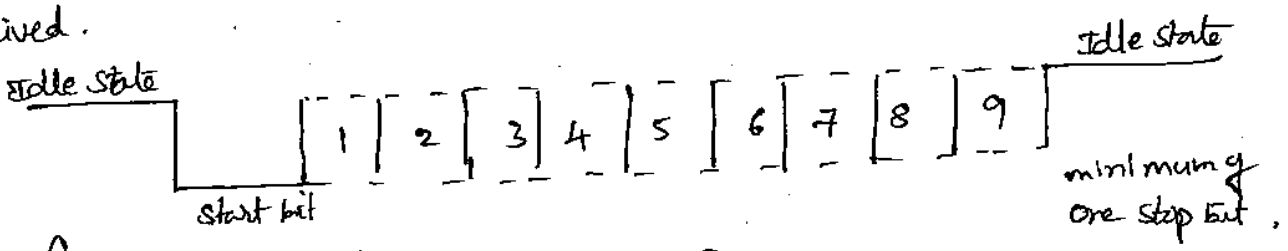
$$f_{Baud} = \frac{2^{SMOD}}{32d} \times \frac{\text{Oscillator frequency}}{12d\,[256d - (TH1)]}$$

If Timer 1 is not run in timer mode 2, then Baud rate is

$$f_{Baud} = \frac{2^{SMOD}}{32d} \times (\text{Timer 1 Overflow frequency}).$$

Serial Data Mode 2 : Multiprocessor mode.

Similar to mode 1, except 11 bits are transmitted ; a Start bit, 9 data bits, one Stop bit. The 9th data bit is copied from bit TB8 in SCON during transmit & stored in bit RB8 of SCON when data is received.



$$f_{Baud} = \frac{2^{SMOD}}{\quad} \times \text{oscillator frequency}.$$

Mode 3 :

Mode 3 is identical to mode 2, except that the baud rate is determined exactly as in mode 1 using timer 1.

Timers & Counters :

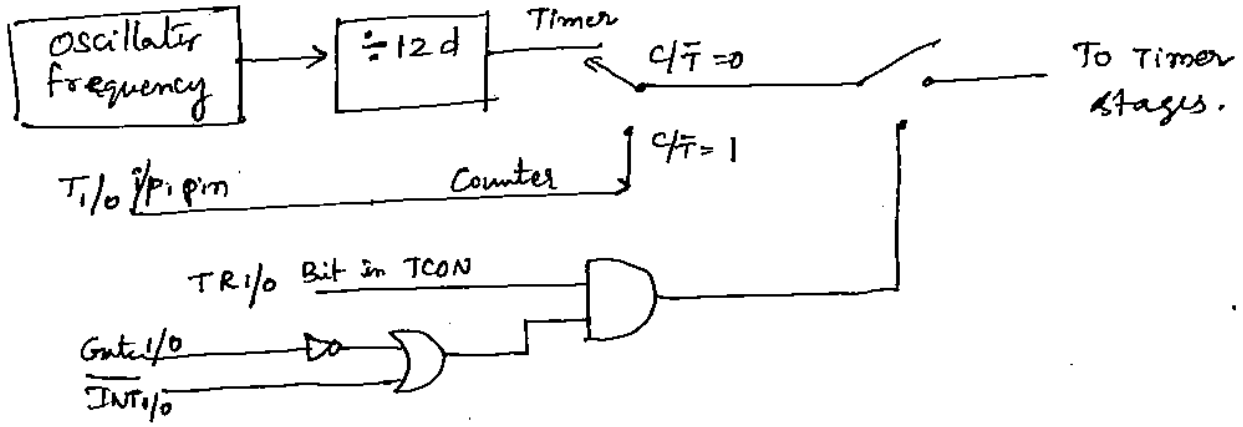
Explain TCON @

Counter may be programmed to Count the internal clock pulses as a timer (or) programmed to Count external pulses as a Counter.

When used as a timer, clock pulses are sourced from the oscillator through the divide by 12 circuit, when used as a Counter Pin T_0 supplies pulses to counter 0 & pin T_1 supplies pulses to Counter 1.

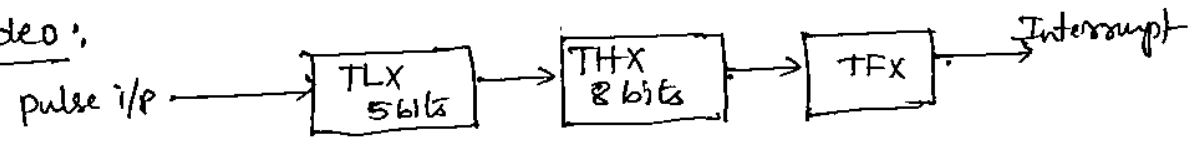
Then explain TCON & TMOD registers. These are explained in the previous topic.

Logic diagram of Timer/ Counter



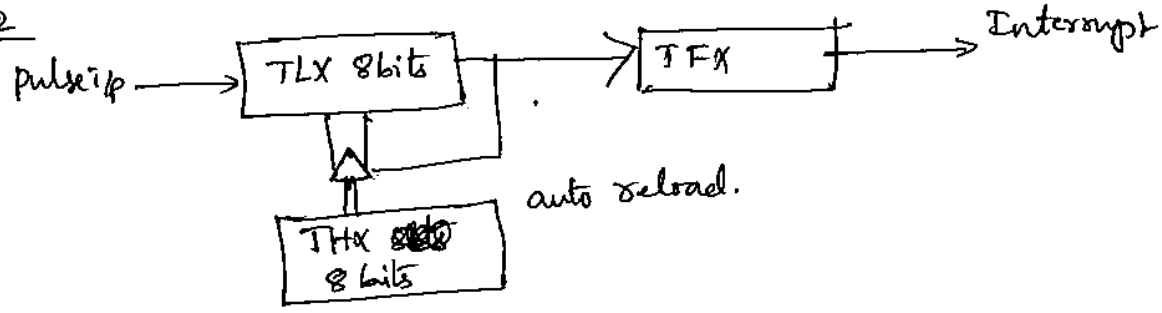
Timer Modes

Mode 0 :

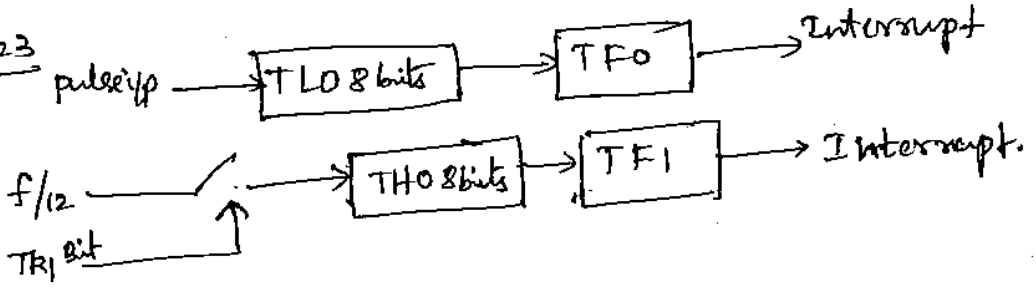


Mode 1

Mode 2



Mode 3



IO ports :

IO ports

port 0.0 to port 0.7 (P0.0 to P0.7)

Port 0 is an 8 bit bidirectional bit addressable IO port. This has been alloted an address in the SFR address range. port 0 act as multiplexed address / data lines during external memory access.

Port 1 (P1.0 to P1.7) :

Port 1 act as a 8-bit bidirectional bit addressable IO port. This has been allotted an address in SFR address range.

Port 2 (P2.0 to P2.7) :

Port 2 acts a 8 bit bidirectional bit addressable IO port. During the external memory access, port 2 emits higher eight bits of address lines which are valid if ALE=1, $\overline{EA}$=0.

Ports (P3.0 to P3.7) :

Port 3 is an 8 bit bidirectional bit addressable IO port. The port 3 pins also serve the alternate functions.

P3.0 → Acts as Serial i/p data pin (RXD)
P3.1 → Acts as Serial O/P data Pin (TXD)
P3.2 → Acts as External interrupt Pin 0 ($\overline{INT_0}$)
P3.3 → Acts as " " " 1 ($\overline{INT_1}$)
P3.4 → Acts as External i/p to times 0 (T_0)

P3.6 - Acts as write control signal for external data memory ($\overline{WR}$)

P3.7 → Acts as ~~Read~~ control signal for external data ~~mem~~ memory ($\overline{RD}$).

(1) Assigning interrupt priorities.

```
MOV IE, #8CH ; Enable EX1 & ET1
SETB PT1 ; Timer 1 interrupt has high priority.
```

(2) Initializing Timer1 in mode1

```
MOV TMOD, #10H ; Timer 1 mode1.
SETB TR1 ; start Timer 1.
CLR TR1 ; Stop timer 1
$: SJMP $.
```

(3) Program to Initialize timer 1 in mode 1

```
MOV SP, #54
MOV TMOD, # 00001000B ; Timer 1 in mode1
SETB ET1 ; Enable the timer 1 interrupt
SETB TR1 ; Start timer 1
SETB EA ; Enable all interrupt Access.
$: SJMP $
```

Note: The above prog. will strat timer1 and when it overflows, timer1 interrupt is generated, which will cause the PC to Jump to vector location 001BH.

(4) Initializing timer0 in mode 2.

```
MOV TMOD, # 0000 0010 B.
MOV TH0, # 33H
MOV TL0, # 33H.
SETB TR0
$. SJMP $
```

(5) prog. to generate 21kHz square waves on P1.0 of port 1 using Timer 0 ——— auto reload mo

```
MOV SP, #54H
MOV TMOD, # 0000 0010 B ; Timer0 mode 2
MOV TH0, #06H
MOV TL0 #06H
SETB TR0; Start timer0
Loop: JB TF0, Compli
      SJMP Loop
Compli: CPL P1.0 ; Toggle bit P1.0
      SJMP Loop.
```

```
        ORG 0000H
        AJMP start
        ORG 000BH
        AJMP INT_TF0
start:  MOV SP, #54H
        SETB ET0
        SETB EA
        MOV TMOD, 00000010B
        MOV TH0, #06H
        MOV TL0, #06H
        SETB TR0
here:   SJMP here
INT_TF0: CPL P1.0
        RETI
        END
```

(7) write 8051 program to receive a serial byte through RxD

```
        ORG 0000H
        MOV SCON, #01010000
        MOV TMOD, #0010 0000
        MOV TH1, #230d   (1200 Baud rate)
        SETB TR1
        CLR RI
here:   JNB RI, here
        MOV A, SBUF
        END
```

Transmission

```
        ORG 0000H
        MOV SCON, #01000000B
        MOV TMOD, #0010000
        MOV TH1, #230H
        SETB TR1
        MOV SBUF, #56H
here:   JNB TI, here
        CLR TI.
```

(8) write 8051 prog. as example interrupt call to routine, timer 0 is used in mode 0 to overflow & set the timer 0 interrupt flag. when interrupt is generated, the program vectors to the interrupt routine, resets the timer 0 interrupt flag, stops the timer & returns.

```
        MOV TMOD, #00H.
        CLR TF0.
        MOV IE, #82H
        SETB TR0.
Wait:   SJMP wait
        ORG 000BH
        MOV EA, #00H
        CLR TR0
        RETI
```

(9) what is the use of mode 0 of serial Communication in 8051. write a program to transmit a data 45H in mode 0.

(P)

```
        ORG 0000H
        MOV SCON, #00H
        MOV SBUF, #45H.
Again:  JNB TI, Again
        CLR TI.
```