Pheonix

14/06/24

EO-2

Microprocessor :- It is used for general purpose

→ If a device consists of an and off it is a controllers.
→ Microcontroller is used for small application.
→ Microcontroller is lower powered device, and it is applied for low competition application.
→ 8051 microcontroller are used for specific purpose
→ All the peripheral devices are integrated on a single chip is nothing but a Microcontroller.
→ All the peripheral devices are integrated on a mother board is nothing but a Microprocessor.

Applications :-
1. Home purposes
2. Embedded purposes
3. General purposes.

→ There are 2 types of Architectures :-
i) VONNUAN Architecdure.
ii) Harward Architecture

→ Microprocessor are built based on the
→ Most of the Microcontrollers built on Hardwaard Architecutre

→ **Difference b/w Microprocessor and Microcontroller.**

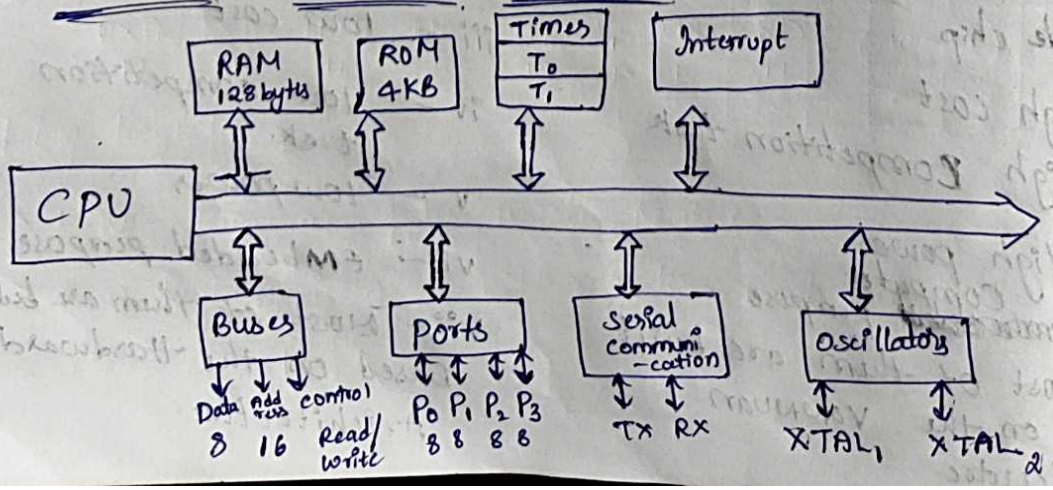
Microprocessor	Microcontroller.
i- General purpose	i- specific purpose.
ii- All the peripheral devices are not integrated in sigle chip.	ii- All the peripheral devices are integrated in single chip
iii- High cost	iii - low cost
iv- High Competition task	iv - low competition task.
v - High powes.	v - low powes
vi - Embedded purpose	vii - Embedded purpose.
vii - Most of them are built based on the VONNUAN Architecutre.	Vii - Most of them are built based on the Hardwaard Arichitecture.

Features of 8051 :-

→ It is 8-bit Microcontroller.
→ It is a 08-bit data bus
→ It consists of 16-bit address bus
→ It consists of two control signals
  i- Read
  ii- write
→ It consists of 128 bytes RAM
→ It consists of 4 Kilobytes ROM
→ It consists of 4 8-bits port.
  i- port 0 (P0)
  ii- port 1 (P1)
  iii- port 2 (P2)
  iv - port 3 (P3)
→ It consists of two timers.
  i- t0 - 16-bits
  ii- t1 - 16-bits
→ It consists of one Serial Communication Module.
  i- TX
  ii- RX
→ It consists of 5 Interrupts - 2 Hardware
  3 software
→ It consists of 1 crystal oscillator, generates a freque of 11.0592 Megahz
  Mega = 10^6
  Micro = 10^{-6}

* It consists of 4 Registers banks

Architecture of 8051 Microcontroller :-



Register Organisation of 8051 Micro Controller :-

→ It consists of two categories of Registers.
  i- Data Registers
  ii- Address Registers.

→ Data Registers are A, B, R_0, R_1, R_2, R_3, R_4, R_5, R_6, R_7
→ Data Registers are 8-bits.
→ Address Registers are DPTR (Data pointer).
→ Address Registers are size is 16-bits.
→ Used for storing the address
→ PC means program counter.
→ program counter is used to hold the address of the next instruction to be executed

Ex:-

2100H	MOV A, #05H
MC:2101H	MOV B, #06H
PC:2102H	ADD A, B

Data Bit - 8bit

→ Memory 16-Bit (or) Address.

→ DPTR is sometimes used as 8-bit register.

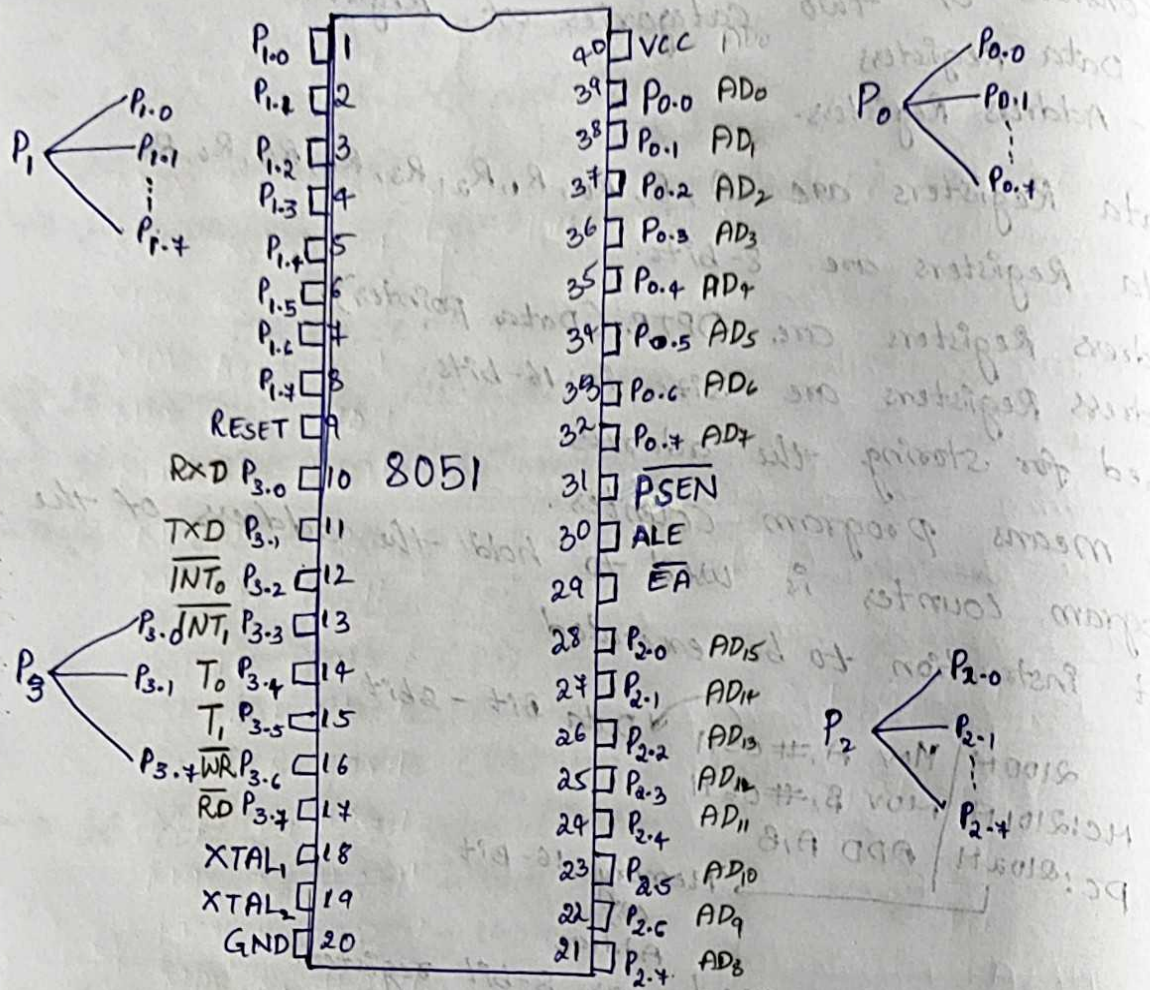
DPTR - 16-bits

↓

DP_H DP_L
(8-bits) (8bits)

Pin Diagram of 8051 Microcontroller :-

→ It is a 40 pin IC, available in DIP (Dual In Package)
→ Port are used for interface the external device to the Microcontrollers.
  Ex:- LED's
→ 8 → Data AD_0 — AD_7
  16 → Address AD_8 — AD_{15}
→ $XTAL_1$, $XTAL_2$ → crystal oscillators
→ PSEN - program store Enable
  EA - External Access
→ PSEN & EA are used to interface the External memory



→ If $\overline{PSEN}$ and $\overline{EA}$ is 1 it is access the Internal memory.

→ If 0 it is external memory.

$\overline{O}$ is 1 $\overline{I}$ is 0

Q) Write the Alternate pins for port(P_0). AD_0-AD_7 (32-39)

Q) write the Alternate pins for port2 (P_2). AD_8-AD_{15} (21-28)

Q) write the Alternate pins for port1 (P_1). No pins.

Q) write the Alternate pins for port3 (P_3).

RXD, TXD, $\overline{INT_0}$, $\overline{INT_1}$, T_0, T_1, $\overline{WR}$, $\overline{RD}$ (10-17)
10 11 12 13 14 15 16 17

Flag Register :-

→ PSW - Program status word Register.

→ PSW Register is a 8-bit register. Each bit can represent either 0 (or) 1.

Carry Flag (CY)
Auxillary Flag (AF)
Parity Flag (P)
Overflow Flag (OV)
Register selection (RS_0)
Register selection (RS_1)

PSW.7	PSW.6	PSW.5	PSW.4	PSW.3	PSW.2	PSW.1	PSW.0
CY	AF	F	RS_1	RS_0	OV	-	P

F - Future use

Parity Flag (F) :-

→ In the result if there are odd no. of "ones" then Parity Flag = 1.

PF = 1

→ In the result if there are even no. of "ones" then parity flag = 0.

PF = 0

→ 1 - ODD no. of one's
0 - Even no. of one's

PSW_4 RS_1	PSW_3 RS_0	Register Bank selection	PSW
0	0	RB_0	CLR PSW_3 CLR PSW_4
0	1	RB_1	SETB PSW_3 CLR PSW_4
1	0	RB_2	CLR PSW_3 SETB PSW_4
1	1	RB_3	SETB PSW_3 SETB PSW_4

→ RB_0 → PSW_3, PSW_4 → 00
↳ $R_0, R_1, R_2, R_3, R_4, R_5, R_6, R_7$.

→ RB_1 → PSW_3, PSW_4 → 01
↳ $R_0, R_1, R_2, R_3, R_4, R_5, R_6, R_7$

→ RB_2 → PSW_3, PSW_4 → 10
↳ $R_0, R_1, R_2, R_3, R_4, R_5, R_6, R_7$

→ RB_3 → PSW_3, PSW_4 → 11
↳ $R_0, R_1, R_2, R_3, R_4, R_5, R_6, R_7$.

Q) Write the procedure to select Registers Bank 3.

→ '1' - SETB
'0' - CLR
} → To give '1' or '0' in micro controller

Usd:- SETB PSW4
SETB PSW3

29) Write the procedure to select Register Bank 1.

sol:-
```
SETB  PSW3
CLR   PSW4
```

30)
```
MOV R0,#02H
MOV R1,#03H
MOV R2,#04H
MOV R3,#05H
MOV R4,#10H
MOV R5,#22H
MOV R6,#30H
MOV R7,#33H
```
} These inputs should be seen in RB_3.

```
SETB PSW3
SETB PSW4
MOV  R0,#02H
MOV  R1,#03H
MOV  R2,#04H
MOV  R3,#05H
MOV  R4,#06H
MOV  R5,#10H
MOV  R6,#20H
MOV  R7,#33H
```

31) Show the status of the bank using the given program.

```
CLR PSW3
CLR PSW4
MOV  R0,#02H
MOV  R1,#03H
SETB PSW3
CLR  PSW4
MOV  R2,#04H
MOV  R3,#05H
CLR  PSW3
SETB PSW4
MOV  R4,#10H
MOV  R5,#20H
SETB PSW3
SETB PSW4
MOV  R6,#33H
MOV  R7,40H
```

RB_0	R_7	R_6	R_5	R_4	R_3	R_2	R_1	02
RB_1	R_7	R_6	R_5	R_4	05	04	R_1	R_0
RB_2	R_7	R_6	20	10	R_3	R_2	R_1	R_0
RB_3	40	33	R_5	R_4	R_3	R_2	R_1	R_0

→ In 8051 programming only 2 symbols we used :-

i- #

ii- @

→ If we want to send the data immediately to register we use "#" before the value

Ex:- MOV A, #02H ✓

MOV B, 03H ✗

→ If we want to access the data from memory/address we are executed use @ before the registers.

5Q) Show the status of PSW Register after the following instructions are executed.

```
MOV A, #02H
MOV B, #05H
ADD A,B
```

sols-

CY	AF	F	RS_1	RS_0	OV	-	P

```
  0000  0010
  0000  0101
 ------------
  0000  0111
```

CY=0 RS_0=0

AF=0 RS_1=0

F=0

P=1

OV=0

0	0	0	0	0	0	-	1

0000 0001

0 1

01H

Status of PSW = 01H

6Q) Show the status of PSW Register after the following instructions are executed.

```
MOV A, #08H
MOV B, #08H
ADD A,B
```

```
  0000  1000
  0000  1000
 ------------
  0001  0000
```

CY	AF	F	RS_1	RS_0	OV	-	P

CY=0

AF=1 RS_1=0

F=0 RS_2=0

P=1

OV=0

0	1	0	0	0	0	0	1

0100 0001

4 1H

Q) Show the status of PSW Register after the following instructions are Executed.

```
SETB  PSW3
SETB  PSW4
MOV A,#08H
MOV B,#08H
ADD A,B
```

Sol:-

```
  0000 1000
  0000 1000
  ---------
  0001 0000
```

CY	AF	F	RS_1	RS_0	OV	-	P

CY=0
AF=1
OV=0
F=0
P=1

RS_1 = 1
RS_0 = 1

0	1	0	1	1	0	0	1

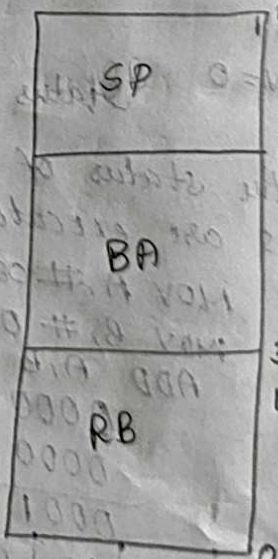
0101 1001
5 9
59H

RAM Architecture of 8051 :-

→ RAM is 128 bytes.
→ Further the RAM is classified into 3 parts.
  i- Register Bank (32 bytes)
  ii- Bit Addressble (16 bytes)
  iii- Scratch Pad (80bytes)

→ Decimal → 00 Byte
  ↓
  127 byte

  Hexa Decimal → 00 H
  ↓
  7F H



→ RB → 00 H → 1F H
  BA → 20H → 2FH
  SP → 30H → 7FH

18/6
→ Bit Address means Either 0 (or) 1
  SetB(1) or CLR(0)
  5V GND

→ In Bit Addressible one register / one digit is stored in BA

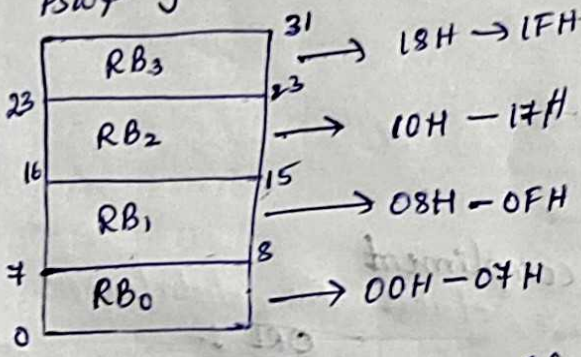
Ex:- 16 → PSW 2A 3-B 4-P_0 5-P_1 6-P_2 7-P_3

→ If we use SETB or CLR for any Program those are stored in bit addressible of RAM.

→ If we want to retreive any memory from RAM we use scratch Pad.

Q) Select the Register Bank 3 and specify the address of Register Bank 3.

sol:- SETB PSW3 } selection
SETB PSW4



00H 01H 02H 03H 04H
R_0 R_1 R_2 R_3 R_4
05H 06H 07H
R_5 R_6 R_7

Q) Select the Register Bank 2 and specify the addresses of Register Bank 2.

sol:-
SETB PSW4 } selection bits
CLR PSW3

Address range from – 10H – 17H

Q) Select the Register Bank 1 and specify the addresses of ~~Register~~ R_1 of Register Bank 1

SETB PSW3
CLR PSW4

Address range :-

Q) write an ALP program to perform the arithmetic operations of 8-bit using 8051 microcontrolles.

sol:- ADDITION 8bit

```
MOV A, #05H
MOV B, #02H
ADD A,B
RET
```

SUBB :- 8-bit

```
MOV A, #05H
MOV B, #02H
SUBB A,B
RET
```

MUL of 8-bit

```
MOV A, #05H
MOV B, #02H
MUL AB
RET
```

Higher 8bits stored in A.
lower 8bits stored in B.

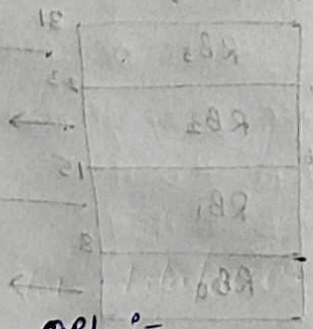
DIV of 8-bit

```
MOV A, #05H
MOV B, #02H
DIV AB
RET
```

Quotient stored in A.
Remainder stored in B.

Q) write an ALP program to perform the logical operations using 8051 microcontroller.

sol:- AND - ANL
OR - ORL
XOR - XRL
NOT - CPL - compliment



ANL :-

```
MOV A, #05H
MOV B, #02H
ANL A, B
RET
```

ORL :-

```
MOV A, #05H
MOV B, #02H
ORL A, B
RET
```

XRL :-

```
MOV A, #05H
MOV B, #02H
XRL A, B
RET
```

CPL :-

```
MOV A, #05H
CPL A
RET
```

Q) write an ALP program to perform the rotate instructions using 8051 Microcontroller.

sol:- Rotate Right - RR.
Rotate Left - RL.
Rotate Right with carry - RRC
Rotate left with carry - RLC

RRL :-

RR :-

```
MOV A, #05H
MOV B, #04H
RR A, 01H
RET
```

RRC :-

```
MOV A, #05H
MOV B, #04H
RRC
RET
```

RL :-

```
MOV A, #05H
MOV B, #04H
RL A, 01H
RET
```

RLC :-

```
MOV A, #05H
MOV B, #04H
RLC A, 01H
RET
```

8) addressing mode of 8051 microcontroller :-

- Immediate addressing mode
- Register addressing mode
- direct addressing mode
- Register indirect addressing mode
- Index addressing mode

Immediate :-

→ MOV A, #02H
MOV DPTR, #0200H

Register :- Transfer the data from Register to Register.

→ MOV A, B
→ MOV A, R0

→ Direct
Register Indirect } → RAM
Index → ROM

Direct :- Here we take data from RAM memory directly.

30H	05H
31H	06H

30H → 7FH

Ex:- MOV A, 30H
MOV B, 31H

Register Indirect Address :-

→ We have to use only 2 registers to point the address of the RAM i.e., either R_0 or R_1.

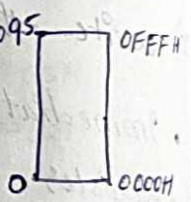
→ We also have to use only A to move the address.

30H — 7FH

Ex:- MOV R_0, #30H
MOV A, @R_0
Inc R_0
MOV A, @R_0
End

Index Addressing Mode :-

→ Index Addressing Mode takes the data/access the data from ROM.



→ Ex:- MOVC A, @A+DPTR
(code)

→ we use DPTR to hold the ROM address.

→ we use "C" after MOV because indicates code.

→ Ex:- MOV DPTR, #0200H
CLR A
MOVC A, @A+DPTR

0200H	05H
0201H	06H

Destination is filled i.e., A

Q) write a program to copy the value 55H into the RAM memory locations 40H & 41H.

a) by using direct addressing Mode
b) Register indirect addressing Mode.
c) Register indirect addressing Mode with loop.

a)

Sol:- MOV A, 40H
MOV A, 41H.

MOV A, #55H
MOV 40H, A
MOV 41H, A
END

A → 55H
40H → 55H
41H → 55H

b) By using Register Indirect Addressing Mode :-

```
MOV A, #55H
MOV R0, #40H
MOV @R0, A
INC R0
MOV @R0, A
END
```

40H ⟶ 55H

c) Register Indirect Addressing Mode loop :-

```
MOV R3, #02H
MOV A, #55H
MOV R0, #40H
GO: MOV @R0, A
INC R0
DJNZ R3, GO
END
```

Q) write a program to clear 16RAM locations the starting address of RAM is 60H

```
MOV R3, #10H
MOV A, #00H
MOV R2, #10H
MOV R0, #60H
GO: MOV @R0, A
INC R0
DJNZ R2,
```

Q) write a program to send 55H into 5 RAM locations. the starting address of RAM is 40H

Sol:-

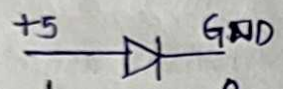
```
MOV A, #55H
MOV R4, #5d
MOV R0, #40H
GO: MOV @R0, A
INC R0
DJNZ R4, GO
End
```

8) Write a program to copy a block of 10 bytes of data from 35H to 60H

```
MOV R2,#04H
MOV R0,#35H
MOV R1,#60H
GO: MOV A,@R0
    MOV @R1,A
    INC R0 36H
    INC R1 61H
```

35H	to	60H
36		61
37		62
38		63
39		64
40		65
41		66
42		67
43		68
44		69

9) Write an ALP program to blink an led by using SETB, CLR, complement.



SETB CLR

Random Delay

Ex:-

```
    ON
    MOV R0,#10d
GO: DJNZ R0, GO
    OFF
```

→

```
Back: CLR P0.0
      Acall Delay
      SETB P0.0
      Acall Delay
      SJMP Back
Delay:-
      MOV R0,#10d
GO:   DJNZ R0, GO
      RET
      End
```

} using SetB and CLR

→

```
    MOV A,#00H
Back: MOV P0,A
      Acall Delay
      CPL A
```

} using complument (CPL)

```
    SJMP Back
Delay:-
    MOV R0, #10d
GO: DJNZ P0, GO
    RET
    End.
```

6) write an ALP program to create a patterns using 8051 micro-controller.

Sol:-

```
    MOV A, #80H
Back: MOV P0, A
    Acall Delay
    RR A
    SJmp Back
Delay:- MOV R0, #0d
GO:- DJNZ P0, GO
    RET
    End.
```

→ During Reset all register value consider as "0". except stack pointer (SP) is 07H

→ stack pointer value during Reset is 07H

Stack operations:

→ Last In First Out (LIFO)

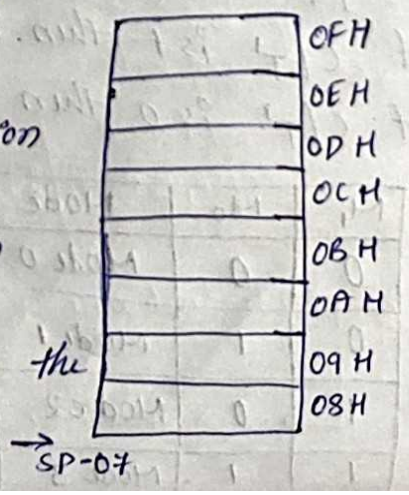
→ In RAM we can use RB_1 i.e., Register Bank$_1$ as stack Memory/ stack pointer.

→ stack Pointer (SP) Address → 08H - 0FH

→ Push - i. Increments stack Pointer by 1.
ii. Data moved to Increment location

→ Pop - i. Decrement stack pointer by 1.
ii. Data should be retreived into register.

→ Push and pop is used to read/write the data from the stack



→ MOV R_0, #22H
MOV R_1, #33H
Push 0
push 1
pop 4
pop 6

Timers :-

→ Timers are used to generate a delay.

→ There are two types (↑8bits ↑8bits)
i- T_0 – 16 bits → TH_0 TL_0
ii – T_1 – 16 bits → TH_1 TL_1
(↓8bit ↓8bit)

→ If we want to control the Timer there are two types of special Function Registers (SFR's)
i- TMOD (Timer mode of Operation) – 8 bits
ii - TCON (Timer controller) – 8 bits

→ Structure of TMOD Register

→

T_1				T_0			
G	$\frac{C}{T}$	M_1	M_0	G	$C/\overline{T}$	M_1	M_0

G – Gate
C/T – counter/Timer

→ If G=0, software operate the Timer

→ If G=1, Hardware operate then Timer whenever interrupt is occurs

→ If $\frac{C}{T}$ = 0 is T
$C/\overline{T}$ = 1 is C

→ If C/T' is 1 then it works as counters.

→ If C/T' is 0 then it works as Timers

→

M_1	M_0	Mode	operation
0	0	Mode 0	13-bit Timer
0	1	Mode 1	16-bit Timer
1	0	Mode 2	8-bit Auto Reload
1	1	Mode 3	split Mode

16-Bit Timer :-

→ Timers – FFFFH (High)
|
0000H (LOW)

→ If the timer reaches to higher limit then TF=1.
→ To know the status of Timer i.e., whether it reached high/max level we use TF (Timer Flag)
→ Timer Run (TR) - Give/show the time it has to run.
→ Higher limit is given by user, not FFFFH.

13-Bit Timer :-

→ Timer – 1FFF H (High)
|
0000 H (low)

→ Most of the applications we prefer mode 1 i.e., 16-bit Timer because we can give max mode

8-bit Auto Reload :-

→ 00H – FFH.
→ Timer doesnot stop in Auto Reload.
→ when TF=1, then value in TH automatically gets reloaded in TL.
→ TL – 00H ← TF=1
|
TH – FFH
→ when timer reaches to max limit TL automatically gets reloaded with TH value.

Split Mode :-

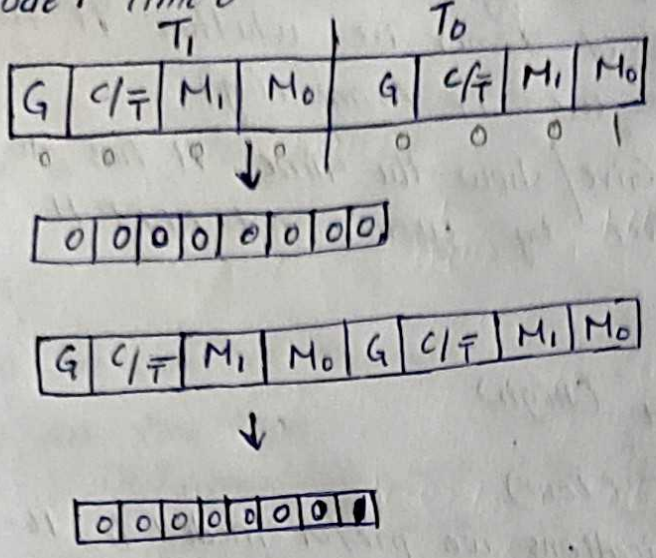
→ Either TH are TL

→ Find the value of TMOD Register from the following information.

a) Mode 0 Timer 1

b) Mode 1 Time 0

a)



b)

19/6/24

→ TMOD used the mode of register

→ Timers also tell whether the software is used / Hardware is used

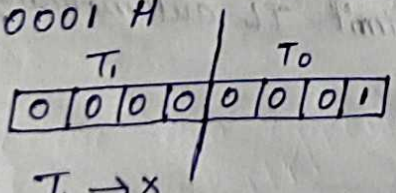
→ T_0 / T_1

→ Gate bit is enabled when software (or) Hardware.

→ To control timer we use TR_x

→ To drag timer we use TF_x

→ a) 0000 0001 H



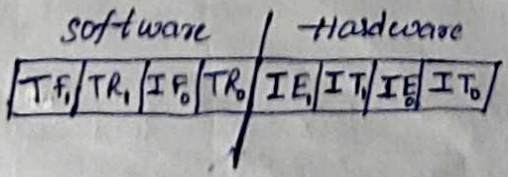
$T_1 \rightarrow X$

01 → Mode 1 → 0001 H → FFFF H

TCON Register :-

→ TCON – Timer Controller

→ size is 8 bits.



→ G → 0 → Software Enabled – TR_x or TF_x

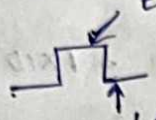
TR_0 / TR_1 TF_0 / TF_1

→ 1 → Hardware Enabled

INT_0 / INT_1

→ Higher bits software Enabled
→ lower bits Hardware Enabled

→ Interrupts are 2types.
i) low level Triggered
ii) high level Triggered



→ $IT_x = 1$ → Edge Triggered.
$IT_x = 0$ → low level Triggered

→ If TF_1 = overflow of Timer 1.
TF_0 = overflow doesnot occure of Timer 1.
No overflow of Timer 1.

→ If IE_0 is Interrupt not occured.
If IE_1 is Interrupt occured.

Steps for executing the program by using Timers.

st-1:- load the TMOD value
st-2:- load the Time Delay value into the registers (TH_x & TL_x)
st-3:- start the Timer TR_x
st-4:- Track the timer by using timer flag (TF_x)
st-5:- Clear TR_x and TF_x
st-6:- Repeat the steps from 2 to 5.

8) write an ALP program to blink an led by using Timers with 8051 microcontroller.

```
    MOV TMOD, #01H
Back: MOV TH0, #FFH
    MOV TL0, #0FH
    CLR P0.0
    Acall Delay
    SJMP Back
Delay: SETB TR0
    GO: JNB TF0, GO
    CLR TR0
    CLR TF0 RET
```

Q) Calculate the timer frequency and it's time period. Assume that crystal frequency is equal to 12MHz.

Note:- Timer assumes $\frac{1}{12}$th of crystal frequency irrespective of machine cycles.

Time period $= \frac{1}{TF \times F}$

Sol:-

$$TF = \frac{12MHz}{12} = 1MHz \qquad M = 10^6$$

$$\text{Time period} = \frac{1}{F} = \frac{1}{1MHz} = 1\mu sec = 1 \times 10^{-6} sec$$

Q) Calculate the timer frequency and it's time period. Assume the crystal frequency is equal to 11.0592

Sol:- Crystal Frequency = 11.0592 (XTAL)

$$TF = \frac{11.0592}{12} = 0.9216 \times 10^{-6} = 921\ KHz$$

$$\text{Time period} = \frac{1}{0.9216} = 1.085069444$$

$$= 1.085\ \mu sec$$

Q) Write an ALP program to blink an led with switching time (delay) of 2 millisecond.

Serial Communication :-

→ Transfer of matter from source to destination means Communication.

→ There are two types of communication

i- parallel communication

ii- serial communication

→ In serial communication, transfer of information is done bit by bit.

→ In parallel communication, transfer of information at a time.

0001 0010 TX ⟶ RX — serial

→ Transmitter ——— Receiver

1 communication

Communication devices/ ways are air and wires.

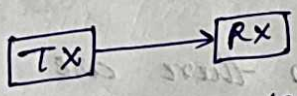
Difference b/w Serial Communication and parallel Communication

Serial	Parallel
i- transfer of information is bit by bit	i- transfer of information is at a time.
ii- Low cost	ii- High cost
iii- More efficiency	iii- Less efficiency compared to serial
iv- serial communication used for longes distance	iv- parallel communication used for shortes distance
v- low complexicity	v- High complexicity
vi- low speed	vi- High speed

Modes of operations :-

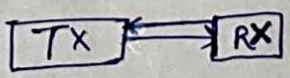
→ Simplex
→ Half duplex
→ Full duplex

→ Simplex communication is nothing but one way
→ It transfers information from Transmitter to Receiver
Ex:- Computer - Transmitter
printer - Receiver



→ Half-duplex communication is a two way communication but not at the same interval of time
→ It transfer information from Transmitter to Receiver and from Receiver to Transmitter Ex:- walky-talky

→ Full duplex is also a two way communication.
It transfer the same interval of time
→ Both Transmission and Receiving are done at the same time (or) at same interval of time.



Ex:- Mobile communication

Data transfer Communication :-

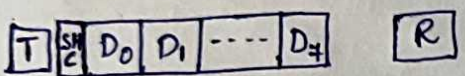
→ They are two types of Data transfers
  i- Synchronous
  ii- Asynchronous

Synchronous

i- If we want to transmit the data both transmitter and Receiver are in synchronous

ii- Singleclock

iii- Frame structure :-



Syn bit should be with data in Synchronous

Ex:- USART
Universal Synchronous Asynchronous Receiver Transmitter

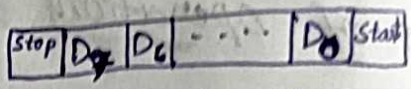
v- These are serial communication

Asynchronous

i- Both transmitter and Receiver need not to be in synchronous.

ii- two clocks

iii- Frame structure :-



No need of sync bit but instead start and stop bits are used.

Ex:- UART
Universal Asynchronous Receiver Transmitter.

V- These are parallel communication

Serial Communication :-

→ To control serial Communication there are
  • Serial communication Buffer (SBUF)
  • Serial Communication controller (SCON)
  • Power controller (PCON)

→ Data is measured by Baud Rate.

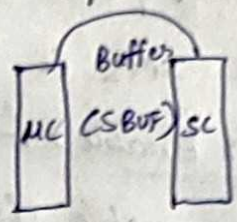
→ No. of bits per second. Baud Rate is measured by no. of bits per seconds.

→ Data transfer is measured by Baud Rate.

→ Default baud rate in systems is 9600. (max system)

SBUF :-

→ To store temporary information of 8-bits we use buffers
→ SBUF - Serial Communication Buffer
→ Serial communication data bit by bit is stored in Buffer.
→ Serial communication to Microcontollers / Microcontollers to Serial communication



D_7	...	D_1	D_0

SCON :-

→

$SCON_7$	$SCON_6$	$SCON_5$	$SCON_4$	$SCON_3$	$SCON_2$	$SCON_1$	$SCON_0$
SM_0	SM_1	SM_2	REN	TB_8	RB_8	TI	RI

SM_0	SM_1	Mode	Description	Baud Rate
0	0	Mode 0	Shift Register	XTAL/12
0	1	Mode 1	8-bit	Variable
1	0	Mode 2	9-bit	XTAL/32 (or) XTAL/64
1	1	Mode 3	9-bit	Variable

Mode 0 :-

TX	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	RX

→ Data bit by bit is transferred by shift operation 8 times. Fixed Baud Rate.

Mode 1 :-

TX	Stop	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	start	RX

→ Data bit-by-bit is transferred by start and stop. Variable Baud Rate

Mode 2 :-

→ Parity bit - It checks whether the data is transmitted Successfully or not. Error indication is identified by Parity Bit. fixed Baud Rate.

stop	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	P	start

Mode 3 :-

→ Parity bit - It checks whether the data is transmitted Successfully or not. Error indication is identified by parity Bit. Variable Baud Rate.

stop	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	P	start

SM_2 :-

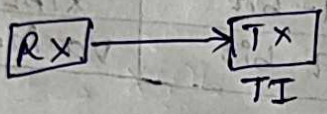
→ SM_2 mode is enabled for Multiprocessor Mode (More than 2)
→ Multiprocessor communication - $SM_2 - 1$
Single processor / one-to-one communication - $SM_2 - 0$
→ Maximum 'SM_2' is in zero state.

REN (Receiver Enable) :-

→ If Receiver enable is "zero" then no data is transferred
→ If Receives enable is "one" then only data is transfared

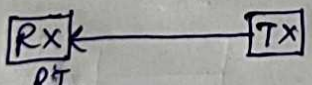
TI :-

→ Transmitter Interrupt.
→ During the communication TI is "0".
→ communication - transfer of data.
→ Successfull data transfer / communication TI is "1".



RI :-

→ Receiver Interrupt
→ During the communication RI is "0".
→ Succesfully data received from transmitter then RI is "1"



TB_8 :-

→ Transmission Bit 8
→ In 9 bits, 9th Bit is Transmission Bit.
→ It is used only during Mode 2 and Mode 3
→ To check the 9th Bit we use TB_8

RB_8 :-

→ Receiver Bit 8
→ It is used only during Mode 2 and Mode 3
→ TO check the 9th Bit we use RB_8.
→ In 9bits, 9th Bit is Receiver Bit.

a) show the status of SCON register assume that the data is transfered in mode 1 and status is successfully transmitted.

SM_0	SM_1	SM_2	REN	TB_8	RB_8	TI	RI

$SM_0, SM_1 - 0,1$
8 bit variable

$SM_2 - 0$
$REN - 1$
$TB_8 - 0$
$RB_8 - 0$

$TI - 1$
$RI - 0$

0	1	0	1	0	0	1	0

0101 0010
5 2
52H

b) show the status of SCON register assume that the data is Received successfully in mode 0

SM_0	SM_1	SM_2	REN	TB_8	RB_8	TI	RI

$SM_0, SM_1 - 0,0$
$SM_2 - 0$
$REN - 1$
$TB_8 - 0$
$RB_8 - 0$

$TI - 0$
$RI - 1$

0	0	0	1	0	0	0	1

0001 0001
1 1
→ 11H

c) show the status of SCON Register. Assume that data is received and transmitted

SM_0	SM_1	SM_2	REN	TB_8	RB_8	TI	RI

$SM_0, SM_1 - 0,1$
$SM_2 - 0$
$REN - 1$
$TB_8 - 0$
$RB_8 - 0$

$TI - 1$
$RI - 1$

0	1	0	1	0	0	1	1

0101 0011
5 3
→ 53H

→ PCON Register :-

→ It is a 8 bit

→ | SMOD | - | - | - | GF1 | GF0 | PD | IDL |

• SMOD :-

→ If SMOD=0, normal Baurd Rate

If SMOD=1, double the Baurd Rate.

→ GF1 and GF0 are General purpose

→ PD is powesDown

→ PD → 1 → Power Down is Enabled
→ 0 → Power Down not Enabled.

→ IDL means Ideal.

→ IDL → 1 → Ideal Mode Enabled
→ 0 → Ideal Mode not Enabled.

• Baurd Rate :-

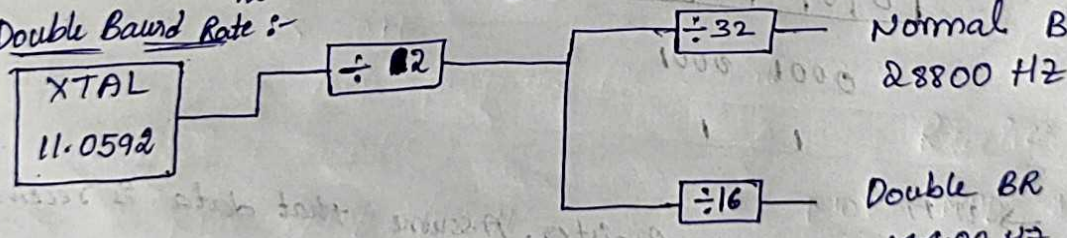
→ In Serial Communication, Baurd Rate is depended on XTAL (crystal Fraquency)

→ XTAL → $\frac{XTAL}{12}$ (A) → $\frac{XTAL}{32}$ (Ans)

a) Find the Baurd Rate for the Frequency 11.0592 MHz

Sol: $\frac{11.0592\,MHz}{12} = \frac{0.9216\,M}{32} = 28800\ Hz$

• Double Baurd Rate :-



PCON - SMOD=1

↓

Directly Baurd Rate gets doubled

20/6/24

a) Steps to write a program in Serial Communication.

st-1:- load the TMOD value.

st-2:- load the SCON value

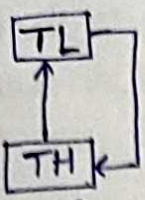
st-3:- load the Baurd rate in to TH, registes.

st-4:- Run the timer TR1

st-5:- Track the TI/RI

st-6:- clear TI/RI

Note :- In Serial Communication we can use mode 2 timer 1



Automatically T_L value should get loaded into TH so we use mode 2 Timer 1 in Serial Communication.

→ we give Baud Rate as input in Timer (TH_1).

Baud Rate Calculation :-

→ Ex:- 1200 Baud Rate

Frequency - ?

$$\frac{11.0592\ MHz}{12} = 0.9216\ MHz = 921.6K$$

$$\frac{921.6}{32} = 28800\ Hz$$

$$\frac{\text{Frequency}}{\text{Baud Rate}} = \frac{28800}{1200} = 24$$

FF - 24 ⇒ 256 - 24 = 232d

= E8H

Find the Baud rate for 2400

5) Find the value load in TH registes assume that baud rate is 2400 and crystal frequency = 11.0592 MHz.

$$\frac{11.0592}{12} = 0.9216\ MHz = 921.6K$$

$$\frac{921.6K}{32} = 28800\ H$$

$$\frac{\text{Frequency}}{\text{Baud Rate}} = \frac{28800}{2400} = 12$$

FF - 12 ⇒ 256 - 12 = 244d

= F4H

6) Baud Rate = 4800

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

FF - 6 ⇒ 256 - 6 = 250d

= FAH

→ 9600

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

FF - 3 ⇒ 256 - 3 = 253d

= FDH

→ TH maximum value is "FF". So in calculating Baud Rate we use "FF".

→ $\frac{28800}{1200} = -24 \Rightarrow E8$

$\frac{28800}{2400} = -12 \Rightarrow F4$

$\frac{28800}{4800} = -6 \Rightarrow FA$

$\frac{28800}{9600} = -3 \Rightarrow Fd$

6) what is the TMOD value for Mode 2 Timer 1

G	C/T	M_1	M_0	G	C/T	M_1	M_0
0	0	1	0	0	0	0	0

0010 0000

2 0

20H

We have to send 20H in TMOD

7) SCON value for Mode 1

SM_0	SM_1	SM_2	REN	TB_8	RB_8	TI	RI
0	1	0	1	0	0	0	0

5 0

⇒ 50H

8) Write an ALP program for Serial communication of Baud Rate 9600 Transfes Receives

solu:

```
MOV TMOD, #20H
MOV SCON, #50H
MOV TH, # FDH (-3)
SETB TR1
MOV A, #'K'
Acall Transfer
Transfer:
    MOV SBUF, A
GO: JNB TI GO
    CLR TI
    RET
```

```
MOV TMOD, #20H
MOV SCON, #50H
MOV TH, # FDH (-3)
SETB TR1
MOV A, #' K'
Acall Receives
Receives:
    MOV A, SBUF
GO: JNB RI GO
    CLR RI
    RET
```

```
MOV A, #'L'
Acall Transfer
MOV A, #'U'
A call Transfer
End
```

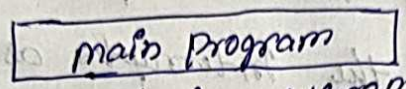
```
MOV A, #'L'
Acall Receives
MOV A, #'U'
A call Receiver
End
```

Interrupts :-

→ IE - Interrupt Enable.

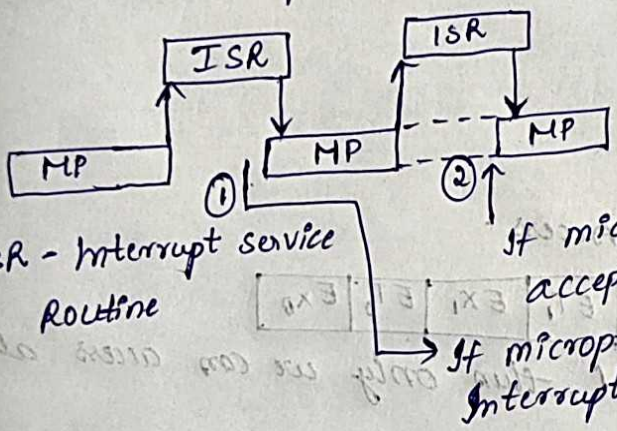
→ IP - Interrupt priority.

→ without Interrupt



No break in Microprocessor / Main program

→ with 2 Interrupts



→ INT_0, INT_1 → Hardware Interrupts.

→ Interrupts → 5
- 2 Hardware → INT_0 (X_0) External 0; INT_1 (X_1) External 1
- 3 Software → T_0 — TF_0; T_1 — TF_1; SC (serial communication) — TI/RI

Interrupt Vector Address :-

→

Interrupt	Vector Address
INT_0 (X_0)	0003 H
T_0	000B H
INT_1 (X_1)	0013 H
T_1	001B H
SC (S)	0023 H

Priority orders

Priority of Interrupts :-

→ 1st priority - INT_0 (X_0)
2nd priority - T_0
3rd priority - INT_1 (X_0)
4th priority - T_1
5th priority - SC (S)

→ If many no. of interrupts occurs at the same time in Microcontroller it follows the priority order of the Interrupts to respond to the interrupts.

→ Microcontroller respond to the Interrupts as per the priority order of Interrupts.

IE Register :-

→ IE - Interrupt Enable
→ Size is 8 bits

EA = Enable Access

EA	-	ET_2	ES	ET_1	EX_1	ET_0	EX_0

→ EA → should always be 1, then only we can access all the interrupts.
→ If EA is 0, then we can't access the interrupt.

IP Register :-

→ Interrupt Priority.
→ Size is 8-bits.

-	-	PT_2	PS	PT_1	PX_1	PT_0	PX_0

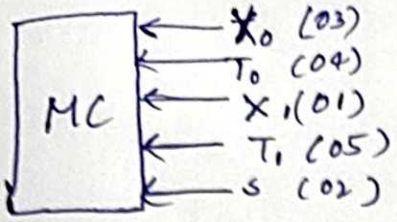
→ To change the priority order as per the programes we use IP Register
→ we change the priority of the interrupts through the IP Register

Ex:- 1st priority - SC

-	-	PT_2	PS	PT_1	PX_1	PT_0	PX_0
0	0	0	1	0	0	0	0

→ we have to keep "1" for the interrupt we want to give priority for remaining we have to keep '0'.

c)



change the priority orders.

-	-	PT_2	PS	PT_1	PX_1	PT_0	PX_0
0	0	0	1	0	1	0	0

d) Find the value of IE Register. Assume that Timer 0 and Serial Communication are enabled.

EA	-	ET_2	ES	ET_1	EX_1	ET_0	EX_0
1	0	0	1	0	0	1	0

9 2

e) Find the value of priority Register. Assume that priority for serial Communication. 92 H

-	-	PT_2	PS	PT_1	PX_1	PT_0	PX_0
0	0	0	1	0	0	0	0

1 0

=> 10 H