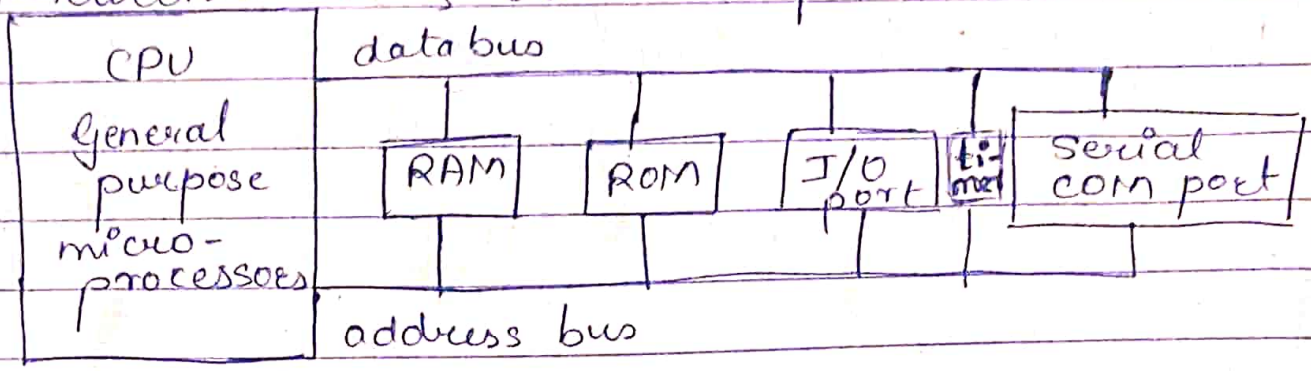
Unit 2 : The 8051 microcontroller

⇒ Microcontroller & embedded processors.



(a) Microprocessor

SoC (system on cheap)

CPU	RAM	ROM
I/O	Timer	Serial COM port

(b) Microcontroller.

Ques difference betn microprocessor & microcontroller

* Criteria for choosing microcontroller (uc)

major uC available.

1) FreeScale's 6811 (motorola)
2) Intel's 8051
3) Zilog's Z8
4) PIC 16X.

20MHz - max frequency is used to find this crystal oscillator (speed of uc)

Criteria

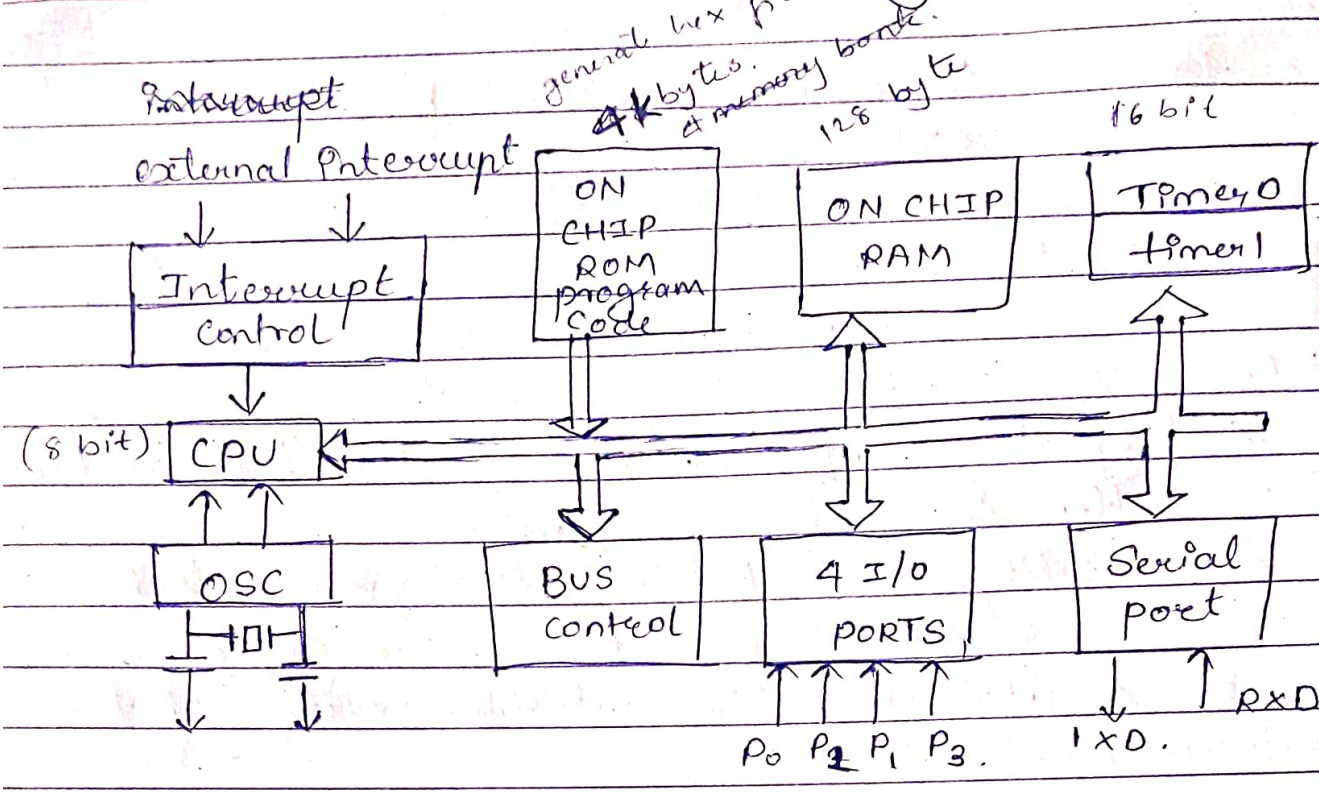
(1) Meeting the computing need of the task at hand efficiently & cost effectively
(2) availability of software development tool such as compilers, assemblers, debuggers.
(3) wide availability & reliable sources of the microcontroller

(a) speed. —
(b) packaging — size of uC
(c) power consumption - ↓
(d) amount of RAM or ROM require on chip
Intral ∴ Memory access is fast.
(e) IO port.
(f) upgradability - 8051 easily upgradable.
(g) cost.

(2) 8051 - available compilers, assemblers

(3) ↓

8051 microcontroller Block diagram.



* which type of architecture

8 - bit data bus → carry data from CPU to other dev
16 - bit address bus
control bus :-
control signal like read, write

P_0 P_2 → have dual mode
carring data & address bit at a time.

P_1 → purely

P_3 → alternate function

timer

T_0 → | TL0 | TH0 | - 16 bit

T_1 → | TL1 | TH1 | - 16 bit.

lower byte & highter bit timer

Serial port

two pins.

default board rate 9600

Table 1.2 1.4 1.6 1.7.

Programming 8051 in 'C'

* C is preferable that embedded assembly lang

Data types

1) unsigned char (0-255) 8 bit.
2) Signed char (-127 to +128) one bit for sign + → -
3) unsigned int (0 to 65,535) (0 to 2^{16}) 16 bit
4) signed int (-32768 to +32167)
5) sbit (1-bit). (single bit)
6) sfr
7) bit

Note :- Sbit data type is used to access single bit of special function registors of 8051 micro-controller.

eg Sbit P0B0 = P0.0 (0th bit of P0.)
Sbit regALSB = ACC.0.

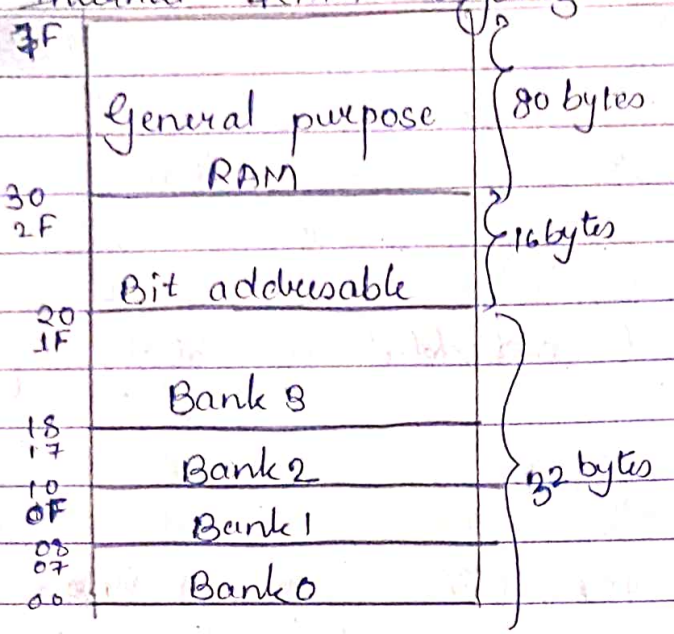
1) bit data type is used to access single bit of bit addressable memory locations 20 - 2FH.
16 memory locations

Sfr

to access the byte size special function registors we use sfr datatype.

Internal RAM organization

Internal RAM organization



Bit addressable RAM.

21	17	16	15	14	13	12	11	10	09	08
20	07	06	05	04	03	02	01	00

PSW (program status word). (flag register)

Cy	AC	F0	RS1	RS0	OV	-	P

Carry auxiliary

result produce carry flag. * devided into 4000 equal parts

Carry (during addition) Barrow is produced the also set.

ex 0110 1001
0001 1000
0001
set to 1.

F0 → user defined flag, reserved, not used.

RS1 → register select

RS_1	RS0	
0	0	→ Bank 0
0	1	→ Bank 1
1	0	→ Bank 2
1	1	→ Bank 3.

OV → overflow.

whenever arithmatic or logical result in greater memory than the capacity of machine.

P → parity
no. of 1's in bits. ① even parity ② odd.

new project → create new folder → new file (program)
→ select atmel → AT89C52E2 → target file created.
→ goto file menu → new file → save in some folder
(.c file)

for simulation use < >.
otherwise " "

```
#Include <reg51.h>
void main (void)
{
  unsigned char x;
  while(1)                    to repeat task or for(;;)
  {    for (x=0 ; x<255 ; x++)
       {
            P1 = x;
       }
  }
}
```

add .c file to source group. → compile → Build
→ Debug → go to step by step execution → Peripheral
→ IO port

Write an 8051 C program to send the ASCII values of hexadecimal values on port1

```
#Include <reg51.h>
void main (void) {
  unsigned char mynum[] = "0123456789ABCDEF";
  unsigned char z;
  for (z=0 ; z<16 ; z++)
    P1 = mynum[z];
}
```

3) write an 8051 C program to toggle the bits of port 1 forever.

```
#Include<reg51.h>
void main(void)
{
    for(;;)
    {
      P1 = 0x55;
      P1 = 0xAA;
    }
}
```

0x55 0101 0101
0xAA 1010 1010
OR
0x00 0000 0000
0xFF 1111 1111

4) write a 8051 C program to send the signed numbers 1 to 4 on port 1.

```
#Include<reg51.h>
void main(void)
{
    char mynum[] = {+1,-1,+2,-2,+3,-3,+4,-4};
    unsigned char x;
    for(x=0; x<8; x++)
      P1 = mynum[x];
}
```

+1 → 0x01 = 0000 0001
1's compliment → 1111 1110
add 1 → 1
2's compliment → -1 ⇒ 1111 1111 0xFF = -1

-2 = 0xFE -3 = 0xFD -4 = 0xFC.

Time delay generated

Time delay in 8051.

(1) for loop is used to include time delay in program (software delay)
(2) timer (T_0 & T_1) also used to generate time delay. (Hardware delay)

*parameters using for loop.

1) 8051 design
2) crystal frequency.
3) compiler used

machine cycle is the no. of clock pulses required to execute 1 fast complete instruction.

* Find the time period of the machine cycle for the 8051 microcontrollers operating at
1) 11.0592 MHz 2) 16 MHz 3) 22 MHz

⇒ 1 m/c cycle = 12 clk pulses.

① machine cycle frequency = $\frac{1}{12}$ × crystal frequency.

$= \frac{1}{12} \times 11.0592 \times 10^6$

$= 921600$

$= 921.6$ kHz.

t = 1/machine cycle freq.

$= 1/921.6\text{kHz} = \frac{1}{921.6 \times 10^3} = 1.085 \times 10^{-6}$ s.

$= 1.085$ μs.

② machine cycle freq. $\frac{1}{12} \times 16$

$= 1.33$ MHz.

$t = \frac{1}{1.33 \times 10^6} = 0.751$ μs.

③ machine cycle freq $= \frac{1}{12} \times 22 \times 10^6$.

$= 1.834$ MHz.

$t = \frac{1}{1.834 \times 10^6} = 5.45 \times 10^{-7}$

$= 0.545 \times 10^{-6}$ s

$= 0.545$ μs.

1) Write an 8051 C program to toggle bits of port 1 continuously with some delay

=>
```
// toggle with certain delay
#include <reg51.h>
void main (void)
{
    unsigned char x;
    while (1)
    {   P1 = 0x00;
        for (x = 0; x <400 ; x++); // delay use for
          P1 = 0xff;
        for (x=0; x<400; x++);
    }
}
```
(1275 - standard value to generate 1ms delay)

2) Write an 8051 C program to toggle the bits of P1 continuously with 250 ms delay.

=>
```
// toggle P1 with 250ms delay.
#include <reg51.h>
void main(void)
void MSDelay(unsigned int); //fn to create delay.
void main (void)
{
    while(1) {
    P1 = 0x00;
    MSDelay(250);
     P1 = 0xff;
    MSDelay(250);
    }
}
void MSDelay(unsigned int time)
{
    unsigned int i,j;
for (i=0; i<time; i++) {
  for (j=0; j<1275; j++)
}
```

3) Write an 8051 C program to toggle all the bits of P0, P2 continuously with 250

=>
```
#include<reg51.h>
void MSDelay(unsigned int);
void main (void)
{   while(1) {
    P0 = 0x00;
    P2 = 0x00;
    MSDelay(250);
    P0 = 0xff;
    P2 = 0xff;
    MSDelay(250);
    }
}

void MSDelay(unsigned int itime)
{   unsigned int i,j;
  for(i=0; i<itime; i++)
    for(j=0; j<1275; j++).
}
```

4) LED's are connected to bits P1 & P2 with an 8051 C program that shows the count from 0 to FFH on the LEDs

=>
```
#include<reg51.h>
#define LED P2
void main(void){
    P1= 0x00;   //clear P1
    LED= 0;     // clear P2.
  for(;;)       // repeat forever.
  {
    P1++;       // Increament P1
    LED++;
  }
}
```

5) Write an 8051 program to get a byte of data from P1 wait for half second and then send it to port 2.

⇒ P1 = i/p port

½ sec = 500ms.

P2 = o/p port

configure P1 as i/p.

```
⇒ #include<reg51.h>
  void delay(unsigned int);
  void main(void)
  {
    unsigned char mybyte;
    P1 = 0xff ;      // configure P1 as i/p port
    while(1) {
      myByte = P1 ;
      delay(500);
      P2 = mybyte
    }
  }
  void delay (unsigned int itime)
  {
    unsigned int i, j ;
    for (i=0 ; i<itime ; i++)
      for(j=0; j<1275 ; j++) ;
  }
```

6) Write an 8051 c program to get a byte of data from P0. If it is less than 100 send it to P1 otherwise send it P2.

```
⇒ #include <reg51.h>
  void main()
  {
    unsigned char mybyte;
    P0 = 0xff ;   // configure P0 as i/p port
    while(1)
    {
      mybyte = P0 ;
      if (mybyte<100)
        P1 = mybyte ;
      else
        P2 = mybyte ;
    }
  }
```

7) Write an 8051 c program to toggle only bit to continously toggle only port $P_{2.4}$ without disturbing the other bits of port 2.

format

P_x^y → ^ to declare particular port bit.

x → port number 0-3.

y → port num 0-7.

```
#include<reg51.h>
  sbit mybit = P2^4 ;
  void delay (unsigned int);
  void main()
  {
    while (1)
    {
      mybit = 0 ;
      delay (200);
      mybit = 1 ;
      delay (200);
    }
  }
  void delay (itime)
  {
    unsigned int i, j ;
    for (i=0 ; i<itime ; i++)
      for (j=0; j<1275 ; j++);
  }
```

8) write an 8051 C program to monitor bit P1.5 if it is high send 55H ($P_0 = 0x55$) to P_0, otherwise send AA ($P_2 = 0xAA$) to port P_2.

```
#include <reg51.h>
sbit mybit = P1^5;
void main()
{
    mybit = 1 ; // configure P1.5 as i/P.
    while(1)
    {
        if(mybit==1)
            P0 = 0x55;
        else
            P2 = 0xAA;
    }
}
```

9) Write an 8051 C program to turn bit P1.5 on & off 50,000 times.

⇒

10) write an 8051 C program to read the status of bit P1.0 save it & send it to P2.7 continously.

P1.0 → i/p port

P2.7 → o/p port

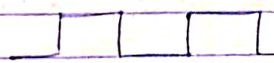
```
#include <reg51.h>
sbit inbit = P1^0;
sbit outbit = P2^7;
void main()
{
    bit membit; // use bit to declare bit-addressable
                // memory
    void main(void)
    {
        while(1) {
        membit = inbit; //get a bit from P1.0
        outbit = membit; //send it to P2.7.
        }
    }
```

11) A door sensor connected to P1.1 pin and the buzzer is connected to P1.7. Write an 8051 C program to monitor the door sensor and when it opens sound the buzzer by sending a square wave of few Hz.

(i/p)

⇒ P1.1 → Door sensor.

P1.7 → (o/p) → Buzzer.



high (open) → sound buzzer

low. (closed)

T_{ON} = Buzzer ON

T_{OFF} = " OFF

```
#Include<reg51.h>
sbit DSensor = P1^1;
sbit Buzzer = P1^7;
void delay (unsigned int);
void main()
{
    DSensor = 1;
    while(DSensor == 1)
    {
        Buzzer = 0;
        delay (200);
        Buzzer = 1;
        delay(200);
    }
}
void delay (unsigned int time)
{   unsigned int i,j;
    for (i=0; i<time; i++)
        for (j=0; j<1275; j++);
}
```

Logic operations in 'C'

1) Bitwise operators.

AND	&
OR	|
NOT	~
EX-OR	^
RIGHT SHIFT	>>
LEFT SHIFT	<<

Run the following program on a simulator & check the results.

```
P0 = 0x25 & 0x07;
P1 = 0x90 | 0x3F;
P2 = ~P1;

P3 = 0x80 ^ 0xFF;
P1 = 0x20 >> 3;
P2 = 0x77 >> 4;
```

P1 = 0x20 >>4

0010 0000 — divide by 2

1st bit 0001 0000 0x10 — 16 in decimal

2nd bit 0000 1000 0x08

3rd bit 0000 0100 0x04

P1 = 0x80 <<2

1000 0000

1st bit 0000 0000 ans 0x00.

P1 = 0x24. <<2. — decimal.

0010 0100. — 36 multiply by 2

1st shift 0100 1000 0x48. — 72

2nd " 1001 0000 0x90 — 144.

3rd " 0010 0000 0x20. exceed.. so

P1 = 0x29 ^ 0x00. — any no. EXOR with 0. then ans is same no. (byte).

0010 1001

0000 0000

0010 1001

P1 = 0x29 ^ 0xFF. — any no. EXOR with FF (1) then result is 1's complement of 1's of that no.

0010 1001

1111 1111

1101 0110

Write an 8051 c program to toggle all the bits of port 0 & P2 continuously with 256 ms delay. use the inverting operator to toggle the bits.

```
#Include<reg51.h>
void Delay(unsigned int);
void main (void)
{
    P0 = 0x55;      // can Take any value.
    P2 = 0x55;
```

```
while(1) {
  P0 = ~P0        // Inverting operator to toggle bits
  P2 = P2 ^ 0xFF  // toggling bits by using EXOR.
  delay(250);
}

void delay(unsigned int time)
{  unsigned int i,j;
   for(i=0; i<time; j++);
      for(j=0; j<1275; j++);
}
```

Write an 8051 C program to read a bit from P1.0 Invert it & send to P2.7

⇒
```
sbit P1.0 → I/P bit = 1
sbit inbit.
sbit membit.
sbit P2.7 = ~membit.
outbit.

#Include<stdi reg51.h>
sbit inbit = P1^0.;      (by default Po port is configured as I/P port)
sbit outbit = P2^7.;     No need to configure
bit membit;
void main()
{  while(1) {
     membit = inbit;     // get a bit from P1.0
     outbit = ~membit;   // Invert it & send to P2.7.
   }
}
```

Write an 8051 C program to read P1.0 & P1.1 bits & Issue an ASCI character to P0 according to the following table.

P1.1	P1.0	
0	0	send '0' to P0
0	1	send '1' to P0
1	0	send '2' to P0
1	1	send '3' to P0

```
0x25 H   0 0 1 0   0 1 0 1
&
0x03;    0 0 0 0   0 0 1 1   and with
         0 0 0 0   0 0 0 1
```

```
#Include<reg51.h>
void main()
{  unsigned char z;
   z = P1;
   z = z & 0x03;
   switch(z) {
     case(0):
        P0 = '0';      // ascI value of 0 is 30.
        break;
     case 1:
        P0 = '1';
        break;
     case 2:
        P0 = '2';
        break.
     case 3:
        P0 = '3';
     }
}
```

Data conversion Po 8051 'C'

BCD — 0-9. → Binary coded decimal.

ASCI — 0x30. – 0
0x31 – 1

Write an 8051 C program to convert packed BCD 0x29 to ASCII & Display the bytes on P1 & P2.

⇒ 0x29 = packed BCD.

		0x29 & 0x0F
2 | 0x30 = ASCII		0010 1001
9 | 0x30 = ASCII.		0000 1111
	0x09 ←	0000 1001

0x29. 0010 1001

0xF0 1111 0000

0010 0000

>>4 0000 0010 0x02.

then OR it with 30.

0000 0010.

0011 0000

0011 0010

```
#include <reg51.h>
void main()
{
unsigned char bcdbyte =0x29;
unsigned char x,y,z;
x = bcdbyte & 0x0F ; // mask higher nibble  x = 0x09
y = bcdbyte & 0xF0 // mask lower nibble ; y=0x20.
y = y>>4 ; // y = 0x02
P1 = x|0x30 ; // P1 = 0x39.
P2 = y|0x30; // P2 = 0x32.
}
```

Write an 8051 C program to convert ASCII digits of 4 & 7 to packed bcd. & display them on P1

⇒ '4' → 0x34 '7' → 0x37.

& 0x0F & 0x0f

= 0x04 | 0x07.

<< 4.

0x40 | 0x07.

```
#include <reg51.h>
void main ()
{
unsigned char bcdbyte;
unsigned char w = '4';
unsigned char z = '7';
w = w & 0x0F // mask higher nibble
                = 0x04
z = z & 0x0F // mask higher nible = 0x07
w = w<<4 ; // w = 0x40.
bcdbyte = w|z ; // bcdbyte = 0x47.
P1 = bcdbyte;
}
```

Checksum byte

1) For the given 4 bytes of hexadecimal data 25H, 62H, 3FH & 52H

i) Find the checksum byte

ii) Perform the checksum operation to ensure data integrity of the

iii) If the 2nd byte 62H has been changed to 22H show how checksum detects the error

⇒

i) Checksum byte 25H (16) mode → base → hexa

+ 62 H

+ 3F H.

+ 52H

carry ← (1)18 H (exclude carry)

1 8 H

0001 1000

1110 0111 1's compliment.

1110 1000 2's compliment

E 8H → checksumbyte

ii) Checksum operation.

25H + 62H + 3FH + 52H + E8H

= 2(00) → ∴ ensure the integrity

carry Engore it

No data currupted

iii) 25H + 22H + 3FH + 52H + E8H. (62H is changed)

= (1)C0 H.

C0 PS non zero ∴ ROM data byte is currupted.

2) Write an 8051 C program to calculate the checksum byte for the data given in previous example.

⇒
```
#include <reg51.h>
main()
{
  unsigned char mybyte[] = {0x25, 0x62, 0x3F,
                            0x52};
  unsigned char sum=0;
  unsigned char x, checksumbyte;
  for(x=0 ; x<4; x++)
  {
      sum += sum + mybyte[x];
  }
    checksumbyte = ~sum +1;
}
```

3) Write an 8051 C program to perform checksum operation to ensure data integrity if data is good send ASCII character 'G' to port 0 otherwise send ASCII character 'B'

⇒
```
#include <reg51.h>
  void main()
{
  unsigned char mybyte[]={0x25,0x62,0x3F,0x52,
                          0xE8};
  unsigned char chksum=0;
    unsigned char x;
  for (x=0; x<5; x++)
  {
    checksum += mybyte[x];
  }
  if (checksum==0)
    P0= 'G';
  else
  {    P0 = 'B';
```

4) Write an 8051 C program to convert 0XFD to decimal & display the digits on port0, P1, P2

⇒

UNIT-3

1) TMOD 20H

TMOD 0010 0000. → Timer 0 in mode 0.
Timer 1 in mode 2

2) 12H

0001 0010.
mod 1 mod 2.

Configure timer 0 in mode 2.

0000 0010

Configure timer 1 in mode 1

0010 0000.

Configure timer 0 in mode 1 & timer 1 mode 1

0 0 0 1 0 0 1 1.

TMOD

GATE	C/T	M1	M0	GATE	C/T	M1	M0
7							0
T1				T2			

Timer 1

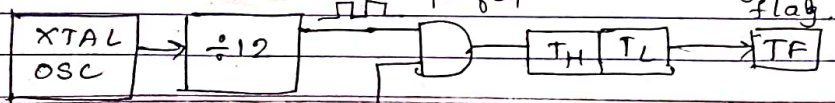
Gate = 1 h/w INT2 - bit control timer op.

0 s/w TR2.

To select mode for timers

M1	M0	
0	0	mode0 (13 bit)
0	1	mod1 (16-bit)
1	0	mod2 (8 bit)
1	1	mod3

Timer MODE1 - (16-bit) TH+TL



f = 11.0592 MHz

C/T = 0 machine cycle = 12 clk pulses

TR when high then only timer starts

TF goes high when FFFF → 0.

Configure

① Timer 0 in mode 1

Timer 0 ← → Timer 0.

G	C/T	M1	M0	G	C/T	M1	M2
0	0	0	0	0	0	0	1

TMOD = 0x01 (0x01) load TMOD with this value in order to configure timer 0 in mode 1

② Timer 1 in mode 2

TMOD :-

GATE	C/T	M1	M0	G	C/T	M1	M2
0	0	1	0	0	0	0	0

Timer 1 ← → Timer 0

TMOD = 0x20.

TMOD is loaded with 0x20 i.e 00100000 bits in order to configure timer 0 in mode 2.

③ Timer 0 in mod 1 & timer 1 in mode 2

timer 1 ← → Timer 0

G	C/T	M1	M0	G	C/T	M1	M0
0	0	1	0	0	0	0	1

TMOD = 0x21

Note :- Whenever a single timer is configure in a particular mode by default the other unused timer will be configure in mode 0.

Find the timers clock frequency & its period for various 8051 based system with the following xTAL freq.

1) 11.0592 MHz = xTAL freq.

M/c cycle freq = xTAL freq / 12 = 11.0592 MHz / 12

= 0.9216 × 10^6 Hz. = 921.6 KHz.

FFFF → max value loaded on model (16bit)

YYXX → TH
YYXX → TL

time period $t = \frac{1}{\text{m/c cycle freq}} = \frac{1}{921.6\,KHz}$

$= 1.085 \times 10^{-3}$ Ksec

$= 1.085 \times 10^{-6}$ sec

② xTAL freq = 22MHz

m/c cycle freq = $\frac{22\,MHz}{12} = 1.83 \times 10^{6}$ Hz

time period $t = \frac{1}{\text{m/c cycle freq}} = \frac{1}{1.83 \times 10^{6}\,Hz}$

$= 0.5464 \times 10^{-6}$ s

$= 0.546$ µs

Generate 25ms time delay using timer 1 in mode1
assume xtal freq = 11.0592 MHz

⇒ required delay = [(FFFF) − YYXX) + 1] * time period (rollback to initial value)

m/c cycle freq = $\frac{\text{xtal freq}}{12} = \frac{11.0592\,MHz}{12}$

$= 0.9216$ MHz

time period = $\frac{1}{0.9216\,MHz} = 1.085 \times 10^{-6}$ s.

$= 1.085$ µs.

25 ms = [(FFFF − YYXX) + 1] × 1.085 µs.

$\frac{25\,ms}{1.085\,\mu s} = 23041 - 1 = \frac{25 \times 10^{3}}{1.085 \times 10^{-6}}$

23041 − 1 = FFFF − YYXX

23040 = FFFF − YYXX

5A00 = FFFF − YYXX

YYXX = FFFF − 5A00

YYXX = A5FF

TH = A5

TL = FF

② obtain possible maximum delay using timer 0 in mode 1. Assume xtal frequency = 11.0592 MHz

⇒ time period = 1.085 µs.

minimum value in timer register to obtain maximum delay. minimum value in register is 0000.

max value that can be loaded in timer register is FF & gives minimum delay

required delay = (((FFFF − YYXX) + 1) * time period.

= (((FFFF − 0000) + 1) * time period.

= [FFFF + 1] × 1.085 µs.

= (65536) × 1.085 µs.

= 71106.56 µs.

= 71.10 ms.

③ Obtain the values to be loaded into TH & TL register to generate a delay of 10ms. mode 1

⇒ delay required = 10ms.

TH = ? TL = ?

mode = 1 xtal = 11.0592 MHz

required delay = (((FFFF − YYXX) + 1) × 1.085 µs.

$\frac{10\,ms}{1.085\,\mu s}$ = (FFFF − YYXX) + 1

9216.58

5405 9216 = FFFF − YYXX + 1

interval 9216 = FFFF − YYXX.

2400 = FFFF − YYXX

YYXX = FFFF − 2400

= DBFF

TH is loaded with DB

TL " " " FF

FFFF → max value loaded on model (16 bit)

YYXX → TH → TL

time period $t = \frac{1}{\text{m/c cycle freq}} = \frac{1}{921.6\,KHz}$

$= 1.085 \times 10^{-3}$ Ksec

$= 1.085 \times 10^{-6}$ sec

② xTAL freq = 22MHz.

M/C cycle freq = $\frac{22\,MHz}{12} = 1.83 \times 10^{6}$ Hz

time period $t = \frac{1}{\text{m/c cycle freq}} = \frac{1}{1.83 \times 10^{6}\,Hz}$

$= 0.5464 \times 10^{-6}$ s

$= 0.546$ µs

Generate 25ms time delay using timer 1 in mode1 assume xtal freq = 11.0592 MHz → rollback to initial value

⇒ required delay = [(FFFF − YYXX) + 1] * time period

time period = $\frac{\text{xtal freq}}{12} = \frac{11.0592\,MHz}{12}$

m/c cycle freq = 0.9216 MHz.

time period = $\frac{1}{0.9216\,MHz} = 1.085 \times 10^{-6}$ s.

$= 1.085$ µs.

25 ms = [(FFFF − YYXX) + 1] * 1.085 µs.

$\frac{25 \times 10^{3}}{1.085 \times 10^{3}} = 23041 - 1$

$\frac{25ms}{1.085\mu s} - 1 =$ FFFF − YYXX

23041 − 1 = FFFF − YYXX

23040 = FFFF − YYXX

5A00 = FFFF − YYXX

YYXX = FFFF − 5A00

YYXX = A5FF

TH = A5

TL = FF

② obtain possible maximum delay using timer 0 in mode 1. Assume xtal frequency = 11.0592 MHz

⇒ time period = 1.085 µs.

minimum value in timer register to obtain maximum delay. minimum value register is 0000.

max value that can be loaded in timer regist. is FF & gives minimum delay.

required delay = (((FFFF − YYXX) + 1) * time period.

= (((FFFF − 0000) + 1) * time period.

= [FFFF + 1] × 1.085 µs.

= (65536) × 1.085 µs.

= 71106.56 µs.

= 71.10 ms.

③ Obtain the values to be loaded into TH & TL register to generate a delay of 10ms. mode1

⇒ delay required = 10ms.

TH = ? TL = ?

mode = 1 xtal = 11.0592 MHz

required delay = ((FFFF − YYXX) + 1) × 1.085 µs.

$\frac{10ms}{1.085\mu s} = (FFFF - YYXX) + 1$

9216.58

5405 9216 = FFFF − YYXX + 1

9216 = FFFF − YYXX.

2400 = FFFF − YYXX

YYXX = FFFF − 2400

= DBFF

TH is loaded with DB

TL " " " FF

④ Calculate the delay required for the given count 5C00 using timers in mode 1

⇒ time period = 1.085 μs.

required delay period = ((FFFF − YYYY) + 1) * 1.085.

= ((FFFF − 5C00) + 1) * 1.085

= (A3FF + 1) * 1.085.

= A400 * 1.085.

= 41984 * 1.085.

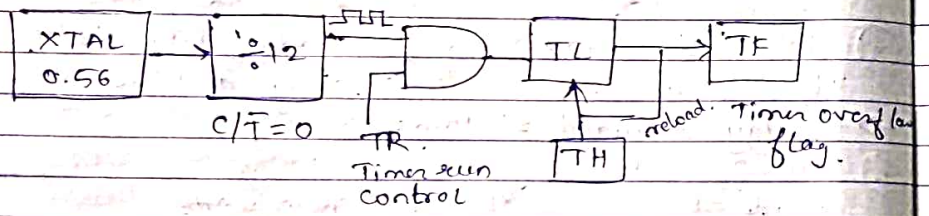
= 45552.64 × 10^{-6} s.

= 45.5 ms.

* Formula to count delay in decimal

timer required = [65536 − NNNNN] * time period.

MODE 2 : (8 bit mode) auto reload mode.



1) Calculate the max delay that can be generated using timers in mode 2. XTAL = 11.0592 MHz

⇒ desired delay = ((FF − YY) + 1) * time period.

= [(FF − 00) + 1] * 1.085 μs

= (255 + 1) * 1.085.

= 277.76 μs.

= 0.27 ms.

② Calculate the delay required using timers in mode 2 when TH = 2C

⇒ required delay = ((FF − YY) + 1) * time period.

= [(FF − 2C) + 1] * 1.085 μs.

= D4 * 1.085

= 212 * 1.085 = 230.02 μs = 0.23 ms

③ Calculate the count to be loaded into TH register to obtain delay of 50 μs. use timers in mode 2.

⇒ XTAL frequency = 11.0592 MHz.

required time = [(FF − YY) + 1] * time period.

50 μs = (FF − YY + 1) * 1.085 μs.

$\frac{50\ \mu s}{1.085\ \mu s} - 1 = FF - YY$

45.08 = FF − YY.

2D = FF − YY

YY = FF − 2D. = D2.

TH = D2.

④ Calculate the count to be loaded into TH register to obtain a delay of 500 μs using timers in mode 2, XTAL freq = 11.0592 MHz

⇒ required time = [(FF − YY) + 1] * time period

500 μs = ((FF − YY) + 1) * 1.085 μs.

$\frac{500}{1.085} - 1 = FF - YY.$

459.82 = FF − YY.

460 = FF − YY

1CC = FF − YY

YY = FF − 1CC.

=

We can not generate

⑤ delay of 3 μs. , XTAL = 11.0592 MHz

3 μs = (FF − YY + 1) * 1.085 μs.

$\frac{3}{1.085} - 1 = FF - YY$

(1.76) 2 = FF − YY

YY = FF − 2.

= FD.

TH = FD.

programming in mode 1

write an 8051 c program to toggle all the bits of port 1 continuously with some delay in between use timer 0 in mode 1 to generate the delay.

⇒ Steps to generate delay using timer 0 MODE 1

① TMOD = loaded with value to select timer 0 in mode 1.

= 0x01

Gate	C/T	M_1	M_0	gate	C/T	M_1	M_0
0	0	0	0	0	0	0	1

② TH = 0xBD } can take any value to generate
TL = 0xFF } some delay (not mentioned delay)
count.

③ Start the timer TR0 = 1.

④ Monitor TF0 (while TF0 == 0).

⑤ Stop timer by making TR0 = 0.

⑥ clear timer ~~TR0 = 0~~ TF0 = 0.

```
#include<reg51.h>
void TODelay(void);
void main()
while(1){
  P1 = 0x00;
  TODelay();
  P1 = 0xff;
  TODelay(); }
}
void TODelay(void)
{
  TMOD = 0x01;
  TL0 = 0xBD;
  TH1 = 0xFF;
  TR0 = 1;
  while(TF0 == 0);
  TR0 = 0;
  TF0 = 0;
}
```

② To toggle only bit P1.5 continously every 50 ms use timer 0 in mode 1

50 ms delay

time period = 1.085 µs.

required delay = [FFFF − YYXX + 1] * time period.

$50\,ms = (FFFF - YYXX + 1) * 1.085\,\mu s.$

$\frac{50\,ms}{1.085\,\mu s} = FFFF - YYXX + 1$

46083 = FFFF − YYXX + 1

4608 = FFFF − YYXX.

B40 = FFFF − YYXX.

YYXX = FFFF − B40.

= 4BFE

```
#include<reg51.h>
void TODelay(void)
~~void main()~~ sbit mybit = P1^5;
void main()
while(1){
  mybit = ~mybit;
  TODelay(); }
}

void TODelay(void)
{
  TMOD = 0x01;
  TL0 = 0xFE;          //to generate 50ms
  TH0 = 0x4B;          // delay.
  TR0 = 1;
  while(TF0 == 0);
  TR0 = 0;
  TF0 = 0;
}
```

③ Write an 8051 C program to toggle all bits of PORT 2 continously every 25ms. use timer 1 in mode 1.

⇒
```
#include <reg1.h>
void T1Delay (void);
void main()
{  while (1){
     P2 = 0x00;
     T1Delay ();
     P2 = 0xff;
     T1Delay(); }
}
void T1Delay()
{
    TMOD = 0x10;
    TL1 = 0xFF;
    TH1 = 0xA5;
    TR1 = 1;
    while (TF1 == 0);
    TR1 = 0;
    TF = 0;
}
```

④ To toggle bits of P2 continuously with 500ms delay use timer 1 in mode 1.

⇒
```
#include <reg51.h>
void T1Delay(void);
void main() {
  unsigned char i;
  while(1) {
    P2 = 0x00;
    for (i=0; i<10; i++)
        T1Delay();
    P2 = 0xFF;
    for (i=0; i<10; i++)
        T1Delay();
  }
}
void T1Delay(void) {
    TMOD = 0x10;     // Timer 1 in mode 1
    TR1 = 0xFE;      // delay of 50ms.
    TH1 = 0x4B;
    TR1 = 1;
    while (TF1==0);
    TR1 = 0;         // stop timer.
    TF1 = 0;         // clear timer overflow flag.
}
```

⑤ Write an 8051 C program to toggle only bit P2.7 continously with 200ms delay use timer 1 in mode 1.

0x02

Programming in mode 2.

⇒ write an 8051 c program to toggle Port 1.5 continuously every 250ms use timer 0 in mode 2

⇒ delay required = 250 ms.

choose = 25 µs. multiply by 10000 to get 250 ms

```
#Include <reg51.h>
void TOM2delay();
void main()
{ unsigned char x, y;
  while(1) {
  mybit = ~mybit
for(x=0; x<250; x++)
  { for(y=0; y<40; y++)
      TOM2delay(); }.
}
void TOM2delay()
{  TMOD = 0x02;
   TH0 = E9.
   TR0 = 1
   while(TF0 == 0);
   TR = 0;
   TF0 = 0;
}.
```

sbit mybit = P1^5;

value loaded to TH.

delay = [(FF-YY)+1] × 1.085µs

$\frac{25\mu s}{1.085\mu s} = FF - YY + 1$

25 − 1 = FF − YY

22 = FF − YY

FF − 16 = FF − YY

YY = FF − 16

= E9.

② write an 8051 c program to generate a frequency of 2500 Hz on P2.7. use timer 1 mode 2 to create a delay. xtal freq =

⇒ Square wave of frequency 2500 Hz.

$T = \frac{1}{2500}\ Hz = 400\ \mu s.$

$T = T_{ON} + T_{OFF}$

$T_{ON} = T_{OFF} = \frac{400}{2} = 200\ \mu s.$

delay = [(FF − YY) + 1] × 1.085 µs.

$\frac{200\mu s}{1.085\mu s} = (FF - YY) + 1$

184 − 1 = FF − YY

183 = FF − YY

B7 = FF − YY

YY = FF − B7 = 48.

TH = 48. to have delay of 200µs

```
#Include <reg51.h>
void TOM2delay (void);
 sbit mybit = P2^7;
void main () {
  while (1) {
    mybit = ~mybit;
    TOM2delay();
    }
}
void TOM2delay()
{    TMOD = 0x20;
     TH0 = 0x48;
     TR0 = 1;
     while (TF0 == 0);
     TR0 = 0;
     TF0 = 0;
}.
```

8) Switch connected to pin P1.2 write an 8051 C program to monitor Switch & create the following frequencies on P1.7 when Switch = 0 500Hz frequency.
Switch = 1, frequency is 750Hz
use timer 0 mode1 for both of them.

⇒ SW → P1.2
P1.7 → square wave
SW = 0 500Hz SW = 1 750 Hz

```
sbit switch = P2^2;
mybit = P1^7;
if (sw==0)
    500Hz  mybit = ~mybit;
```

SW = 0 time = $\frac{1}{500}$ = 2ms.
$t = T_{ON} + T_{OFF}$
$\therefore T_{ON} = T_{OFF} = T/2 = 1$ ms.

1 ms = [(FFFF − YYXX) + 1] * 1.085 µs.
921 − 1 = FFFF − YYXX
920 = FFFF − YYXX
YYXX = FFFF − 398 = FC67.

SW = 1, freq = 750 ∴ $t = \frac{1}{750} = 1.33$ms
$\therefore T_{ON} = T_{OFF} = 0.66$ms.
$\therefore \frac{0.665\text{ms}}{1.085\,\mu s} = \text{FFFF} - \text{YYXX} + 1$
613 − 1 = FFFF − YYXX
YYXX = FFFF − 264 = FD98

Programming timers as counters.

P3 Bit	Functn	Pin	
P3.0	RXD	10	Receive pin during serial data transmission
P3.1	TXD	11	transmit data bit serially
P3.2	INT0	12	
P3.3	INT1	13	
P3.4	T0	14	i/p to timer 0 when configured as counter
P3.5	T1	15	i/p to timer 1 " "
P3.6	WR	16	
P3.7	RD	17	

How to configure timer 0 in mode1 as counter
C/T = 1
10000101 TMOD = 0x05.

P3.4 → counter 0.
P3.5 → counter 1

Timer → count clock pulses generated by crystal oscilator use to generate time delay
Counter → count external event that are occuring

1) Assume that 1Hz external clock is being fed into pin T1 (P3.5) write C program for counter 1 in mode 2 to count up & display the state of TL count on P1 Halt start the counter at 00.

```
#Include<reg51.h>
// sbit T1 = P3.5   by default
void main(void)
{
   T1 = 1;          // make T1 as i/p.
   TMOD = 0X60 //01100000. configure timer2
               // as counter2 in mode 2.
   TH1 = 0.    // start counting from 0.
while(1) {
   do {
      TR1 = 1;      // start counter
      P1 = TL1;     // send count on P1
   } while(TF1==0);
   TR0 = 0;
   TF0 = 0;
   }
}
```

2) Assume that 1hz external clock is being fed into T0 (P3.4) write a C program for counter 0 in mode1 to count the pulses & display the state of TH0 & TL0 registers on P2 & P1 resp.

```
#Include<reg51.h>
void main(void)
{   T0 = 1;      // make T0 as i/p
    TMOD = 0X05; // configure timer0 as
                 //counter0 in mode1
    TL0 = 0;
    TH0 = 0     ; //clear TH & TL
    while(1) {
       do {
          TR0 = 1;     // start the counter.
          P1 = TL0;    // send TH & TL values
          P2 = TH0;    // on P2 & P1 resp
       } while(TF0==0);
       TR0 = 0;    // stop the counter
       TF0 = 0;    // clear TF0.
```

Serial data transmission.

Note :- Timer 1 is used in mode 2 to generate the baud rate for serial data transmission

with XTAL = 11.0592 MHz find the TH1 value needed to have the following baud rates
a) 9600 b) 2400 c) 1200

⇒ XTAL = 11.0592 MHz
machine cycle freq = xtal freq / 12
= 11.0592 / 12 = 921.6 KHz
921.6 KHz / 32 by UART = 28800 Hz to timer1 to set the baud rate.

(a) 28,800 / 9600 = 3
Convert 3 into 2's complement.
that will give the count to be loaded into TH1 to set the baud rate.

```
0000 0011
1111 1100      1's complement
           +1
1111 1101      2's complement of 3
```

TH1 = 0XFD
TH1 register loaded with 0XFD to generate 9600 baud rate.

* for 2400
28800 / 2400 = 12.

```
0000 1100
1111 0011
         +1
1111 0100   2's complement of 12.
```

TH1 = 0XF4

* for 1200
28800 / 1200 = 24.

```
0001 1000
1110 0111
         +1
1110 1000      E8.
```

start bit is low always.
Stop bit or bits high always.

SBUF
- there are 2 lines meant for serial data transmission / reception
TXD, RXD,
P3.1 pin TXD, P3.0 RXD
sbuffer.

SCON → 8 bit register used to program start bit, stop bit & data bit of data framing & among other things

SM0 SM1 SM2 REN TB8 RB8 TI RI

SM0 SM0 → modes always enable 1

```
0  1  0  0  0  0  0  0
```

SCON = 0X50.

Steps

① Write a C program for 8051 to transfer the character 'A' serially at 4800 baud rate continuously use 8 bit data & one stop bit

⇒ It is serial mode 1 ∵ 8 bit & 1 Stop bit (Start bit is always 1, stop bit one & more)

```
⇒ #include<reg51.h>
void main(void)
{ TMOD = 0x20 ; // timer 1 in mode 2 to set the baud rate
  TH1 = 0xFA
// 11.0592 MHz / 12 = 921.6 kHz
// 921.6 kHz / 32 (UART) = 28,800 Hz
// 28800 / 4800 = 6.
// 0000 0100
// 1111 1001
//        +1
// 1111 1010      0x FA
  SCON = 0x50 ; //Serial mode 1, REN enabled
  TR1 = 1 ; // start timer 1
  while(1) {
  SBUF = 'A' ; // place the byte in SBUF : ASCII 65D
  while(TI==0); // monitor transmit flag
  TI = 0; // clear TI after the completion of
          // transmission
  }
}
```

② To transfer letter 'GIT' at 9600 baud rate

TH1 = 0xFD → from previous ④

```
#include<reg51.h>
void SerTx (unsigned char);
void main {
  TMOD = 0x20 ; // timer 1 in mode 2
  TH1 = 0xFD ; // to generate 9600 baud rate
  SCON = 0x50
  TR1 = 1 ;      // Start timer 1
  while (1) {
    SerTx ('G');
    SerTx ('I');
    SerTx ('T');
  }
}
void SerTx (unsigned char x)
{ SBUF = x;
  while (TI==0);
  TI = 0;
}
```