GoodLuck | Page No. | Date

Microcontroller & Embedded Systems (22CS53)

Embedded system: Combination of hardware, software & peripherals.
devices.

↓

microprocessor / microcontroller
(computer)

Hardware: Computer

↳ CPU + Memory + I/O

↓ RAM / ROM

CU + ALU variables program

SoC → System on Chip → Microcontroller (complete computer)

Microprocessor → CPU

* Differentiate between microprocessor & microcontroller

meant for

- General applications.
- 8085
- 8086

- role is fixed / specific.
- widely used in developing embedded systems. (particular task repeatedly)
- 8051 - 8 bit
- ARM7
- Arduino UNO 32 bit
- ESP32. "

Kind of info:

address - of memory location

Bus data bus - collection of wires & connectors to

control → read/write carry signal from one device to another

↳ issuing the signal.

8-bit → data bus capacity to read how many bits &

ALU perform arithmetic & logical operations

0 - 9

A B C D E F Hexadecimal No. system. 0000 → 0

10 11 12 13 14 15

1111 → 15

Project →

P0 } Address / Data
P1

GoodLuck | Page No.
Date

Software to make user interact with machine to get some work application.

- set of instructions. (program)

higher → Assembly → machine level.

Embedded C ADD, SUB, MOV 10010000

Microcontroller Architecture

1) Von Neuman
→ Single memory for both code & data.
→ difficult to fetch.

2) Harvard ✓
→ separate memory for code & data.
→ efficient.

Features of 8051 Microcontroller:

8-bit CPU
4 I/O ports.
2 16-bit timers / counters. include delay b/w events
Crystal oscillator.
Serial Port UART
128 byte internal RAM
4K on chip ROM
Interrupts

one

Send characters on Port

Develop an 8051 C program to send the values A to F on Port P1. Do this continuously.

```
#include <reg51.h>
void main()
{ unsigned char arr[] = {"ABCDEF"};
  unsigned char i;
  while(1)
  { for(i=0; i<6; i++)
      P1 = arr[i];
  }
}
```

41 42 43 44 45 46
↓
01000001
65 01000010
66

GoodLuck | Page No. | Date

two

Develop 8051 C program to send the values -3, -2, -1, 0, 1, 2, 3 on Port 2 continuously.

8 4 2 1

0 0 0 0 0 0 1 1 → 3

MSB 1 1 1 1 1 1 0 0 LSB → 1's compliment

1 1 1 1 1 1 0 1 LSB+1 -3 FD

-2 → 0 0 0 0 0 0 1 0

FE 1 1 1 1 1 1 0 1 1's compliment

1 1 1 1 1 1 1 0

carry sum

FF-1 0 0 0 0 0 0 0 1

1 1 1 1 1 1 1 0

1 1 1 1 1 1 1 1

S C

0 + 0 = 0 0

0 + 1 = 1 0

1 + 0 = 1 0

1 + 1 = 0 1

```
#include<reg51.h>
void main()
{ signed char arr[] = {-3,-2,-1,0,1,2,3};
   unsigned char i;
   while(1) {
      for(i=0; i<7; i++)
        P2 = arr[i];
    }
}
```

three

Develop 8051 C program to toggle all the bits of Port 3

```
#include<reg51.h>
void main()
{ unsigned int i;
   while(1)
    { P3 = 0x00;
      for(i=0; i<500; i++);
      P3 = 0xFF;
      for(i=0; i<500; i++);
    }
}
```

hexadecimal

0x 00

8 bit hexadecimal no 255

msb → 7th bit
LSB → 0th bit

lsb 1 odd no
0 even no

GoodLuck | Page No.
Date

Serial → Synchronous, Asynchronous (conserve energy) → Simplex - 1 way
no control signals. → Half-duplex - send/receive at a time
→ Full- ,, ,,

universal asynchronous
Full duplex - 8051
send & receive data simultaneously
out TxD & RxD at certain speed bits/sec

Data types supported for 8051 'c' / embedded c.

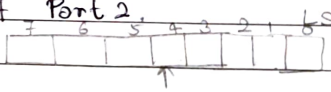
char	(signed & unsigned)	-128 to +127	0-255
int	(signed)	-32768 to +32767	0-65,535 (unsigned)

msb - value 0 +ve
1 -ve

sbit - single bit datatype is used to access single bit of bit addressable registers.
bit
sfr

Port 2 4th pin

Write 8051 C program to toggle only P2.4 continuously without disturbing rest of the bits of Port 2.



sbit mybit = P2^4; 4th pin of Port 2

→
```
#include<reg51.h>
sbit mybit = P2^4;
void main()
{ unsigned char i; // as 200 less than 255
  while(1){
    mybit=0;
    for(i=0; i<200; i++);
    mybit=1;
    for(i=0; i<200; i++); }
}
```

- non-maskable - no control over it → interrupt
- maskable - control over some extent.

09/09/24
Lab

GoodLuck | Page No. | Date

1. Software delay
2. Hardware delay - using timers

value 1275 delay of 1 ms (tested value) for 8051

```
for (i=0; i<itime; i++)
    for (j=0; j<1275; j++);
```

Develop 8051 C program to toggle all the bits of port 1 continuously with delay of 500 ms. Use simple for loop to generate the delay.

```
#include <reg51.h>
void delay (unsigned int);
void main()
{ while(1){
        P1 = 0x00;
        delay (500);
        P1 = 0xFF;
        delay (500);
        }
}
void delay (unsigned int itime)
  { for(unsigned int i,j;
        for (i=0; i<itime; i++)
         for (j=0; j<1275; j++);
  }
```

→ both P1 & P2 toggle

```
P1 = 0x00;
P2 = 0x00;
delay(500);
P1 = 0xFF;
P2 = 0xFF;
delay(500);
```

↳ square wave

Toggle msb bit of P1 & P2

```
#include <reg51.h>
sbit p1msb = P1^7;
sbit p2msb = P2^7;
void delay (unsigned int);
void main()
  { while(1)
    { p1msb=0;
      p2msb=0;
      delay (500);
      p1msb=1;
      p2msb=1;
      delay (500); }
  }
void delay (unsigned int itime)
  {
  =
  =
  }
```

GoodLuck | Page No.
Date

Two LED's are connected to port 2 pins (Pin no 4 & 5) of both
Write a C program to implement mod 4 counter on these 2 LED's

2 pins → 4 states
0 0 → both off
0 1
1 0
1 1 → both on

```
#Include <reg51.h>
void delay (unsigned int);
void main()
{ while(1) {
    P2 = 0x00;
    delay(500);
    P2 = 0x10;
    delay(500);
    P2 = 0x20;
    delay(500);
    P2 = 0x30;
    delay(500);
  }
}
void delay (unsigned int itime)
  { = }
```

Two

```
P2 = 0x30;
delay(500);
P2 = 0x20;
delay (500);
P2 = 0x10;
delay(500);
P2 = 0x00;
delay(500);
```

negative cycle

Give Read the byte of data from P1, wait for half sec. & then send it to P2

entire upon reset
FF → input port Simple* by default all ports configured as i/p port
00 → ouput port 0

```
#Include <reg51.h>
void delay (unsigned int);
void main()
{ unsigned char mybyte,
  P1 = 0xFF;   // configure P1 as i/p port
  while (1) {
    mybyte = P1;  // read a byte from P1 & assign it to mybyte
    delay(500);
    P2 = mybyte; }
}
```

```
void delay (unsigned int itime) {
    = }
```

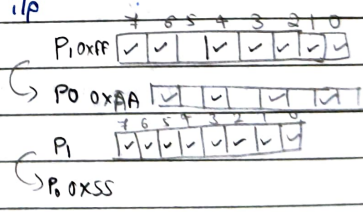
debug → peripherals P0 & P1
F11 (step) then uncheck F5 (Run)

six Write a C program to read a byte of data from port 0. If it is less than 100, Send it to P1. Otherwise send it to P2

```
#include<reg51.h>                    12 hex
void main()                          18 dec < 100
{ unsigned char mybyte;
  P0 = 0xFF ; // P0 is i/p port
  while(1) {
    mybyte=P0;
    if (mybyte < 100)
      P1 = mybyte;
    else
      P2 =mybyte; }
}
```

seven Write a program to monitor the bit P1.5. If it is high send 0x55 on port0 if it is low, send 0xAA on port 0. single bit sbit

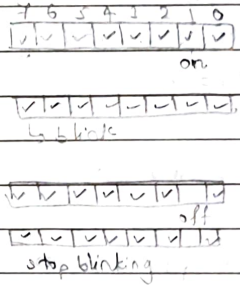
```
#include<reg51.h>
sbit mybit=P1^5;
void main()
{  mybit=1 ;  // configure my bit as i/p
   while(1) {
      if(mybit==1)
        P0 = 0x55;
      else
        P0 = 0xAA; }
}
```



GoodLuck | Page No.
Date

128 byte RAM
32 byte
16 bit addressable
80 general purpose

A door sensor is connected P1.1 & Buzzer connected to P1.7 Write 8051 C program to monitor door sensor & when it opens sound the buzzer. You can sound buzzer by sending square wave of 200 Hz.




```
#include <reg51.h>
void delay (unsigned int);
sbit Dsensor = P1^1;
sbit buzzer = P1^7;
void main()
{ Dsensor = 1;  // configure P1^1 as i/p
  while (Dsensor == 1)
    { buzzer = 0;
      delay(500);
      buzzer = 1;
      delay(500); }
}
void delay(unsigned int itime)
{ =
}
```

Logical Operators:

A	B	A & B	A | B	A ^ B
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0
		Logical multiplication	L. Add°	XOR unlike-1 like-0

& high when both high
| any 1 i/p is high
^ like comb° of i/ps → high

A = 0 } inversion/
~A = 1 } 1's complement

2 way switch

GoodLuck | Page No. | Date

Run the following program on your simulator & observe the output

P0 = 0xFA & 0x05 | P1 = 0x23 \| 0x45 | P2 = 0x75 ^ 0xC5

F	A		0 0 1 0	0 0 1 1		0 1 1 1	0 1 0 1
1 1 1 1	1 0 1 0		0 1 0 0	0 1 0 1		1 1 0 0	0 1 0 1
& 0 0 0 0	0 1 0 1		0 1 1 0	0 1 1 1		0 0 1 1	0 0 0 0
0 0 0 0	0 0 0 0		P1 = 0x67			P2 = 0xB0	

AND · any value adding with all 0 → 0.
· all 1 → value itself

XOR · XOR will all zero → same value.
· all 1's → 1's complement of input

prog → #include <reg51.h>
void main()
{ P0 = 0xFA & 0x05; P2 = 0x75 ^ 0xC5;
P1 = 0x23 | 0x45; P3 = ~0x55; }

ten To toggle all bits of P0 & P2 continuously with 250 ms delay. Use the inverting operator.

#include <reg51.h>
void delay (unsigned int);
void main ()
{ P0 = 0x55;
P2 = 0xAA;
while(1)
{ P0 = ~P0;
P1 = ~P2;
delay (250);
}
}
void delay (unsigned int itime)
{ =
}

P0 - not reflecting on pins only P0 port

P0	0x55	✓		✓		✓		✓	
~P0	0xAA		✓		✓		✓		✓

if we don't initialise, all pins blink.

→ by using shift operator 8 times → shows 0 as o/p bcz it is 8 bit operator

GoodLuck | Page No. | Date

Using XOR:

```
#include <reg51.h>
void delay (unsigned int);
void main(){
    while(1){
    P0 = P0^0xff;
    P1 = P1^0xFF;
    delay(250); }
}
void delay (unsigned int itime)
{ ==}
```

Shift Operators.

>> shift right

<< shift left

left shift

	MSB						LSB	
P0	1	0	0	1	0	1	0	1	P0 = 0x95
95*2 → (1)	0	0	1	0	1	0	1	0	P0 = 0x2A
2A*2 → (0)	0	1	0	1	0	1	0	0	P0 = 0x54

right shift

	1	0	0	0	0	0	1	0	P0 = 0x82
82*2 →	0	1	0	0	0	0	0	1	0x41
41*2 →	0	0	1	0	0	0	0	0	0x20

Write 8051 C program to illustrate the use of shift operators.

```
#include <reg51.h>
void main()
{  P0 = 0x95 << 1;
   P1 = 0x82 >> 1; }
```

Write 8051 C program to perform the following.
1). read a byte from P1 2). send it to port 2 by shifting P1 towards left by 1 time

→

```
#include <reg51.h>
void main()
{ unsigned char mybyte;
   P1 = 0x55 ;  // configure P1 as i/p.
   mybyte = P1 ; // read a byte from P1 & place in mybyte.
   P2 = mybyte << 1 ;
}
```

Unit 2 : Data Conversion
Accessing code ROM space
Data Serialization

GoodLuck | Page No.
Date

BCD to ASCII . BCD to ASCII

BCD - Binary Coded Decimal.

2 bits (0-3) (0-9)
3 bit (0-7)
4 bit (0-15)

grp of 4 bits - nibble
" 8 bits - byte.

0000 0000 → bytes
0000 1001
higher nibble lower nibble

BCD

Packed BCD
2 digit
Eg: 24
98

Unpacked BCD (single digit)
0
1
2
:
9

unpack & convert to ASCII

pre-packed to ASCII

Develop a C program to convert the given packed BCD to ASCII & display on port 1 & port 2. 0x75

Step1 : Unpack the packed BCD.
Mask higher nibble & extract lower nibble.

0x75 = 0111 0101
& 0000 1111 by anding with 0x0f
0000 0101 0x05

0x05 | 0x30 = 0x35 ASCII '5'
(hexa) ↳ $3 \times 16^1 + 5 \times 16^0 = 48 + 5 = 53$ (decimal)

Mask lower nibble
0111 0101 = 0x75
& 1111 0000
0111 0000 = 0x70

right shift 4 times
>>4
0000 0111 = 0x07
0x07 | 0x30 = 0x37 = 55 (decimal)

GoodLuck | Page No. | Date

```
#include <reg51.h>
void main()
{ unsigned char x,y;
  unsigned char mybyte=0x75;
  x = mybyte & 0x0F ; // mask higher nibble.
  P1 = x | 0x30 ; // send ascii value of 5 on P1.
  y = mybyte & 0xF0 ; // mask lower nibble.
  y = y >> 4;
  P2 = y | 0x30 ; // send ascii value of 7 on P2
}
```

Port P1

P1 0x35 | | | | | | | |

Port P2

P2 0x37 | | | | | | | |

Develop a C program to convert ASCII values '4' & '7' into packed BCD. & display on port 1

4 7

0x34 0x37

0x34 = 0011 0100

& 0x0f 0000 1111 = 0x04

0000 0100 = 0x04

<< 4

0100 0000 = 0x40

0x37 = 0011 0111

& 0000 1111

0000 0111 = 0x07

0x47

```
→ #include <reg51.h>
  void main()
  { unsigned char bcdbyte;
    unsigned char a='4';
    unsigned char b='7';
    a = a & 0x0f ;   // mask 3 from 0x34
    a = a << 4;      // bring 4 to higher nibble position
    b = b & 0x0f;    // mask 3 from 0x37
    bcdbyte = a|b;
    P1 = mybyte;
  }
```

$25H \rightarrow 2\times16^1 + 5\times16^0 =$ mod base n

$3FH \rightarrow 3\times16^1 + 15\times16^0$

GoodLuck | Page No. | Date

Checksum:

Technique to ensure data corruption ×
↳ data integrity.

Assume that we have 4 bytes of hexadecimal data.
25H, 62H, 3FH & 52H

a. Find the checksum byte
b. Perform the checksum operation to ensure data integrity
c. If the 2nd byte 62H changed to 22H, show how checksum detects the error.

a. Checksum byte calculation:
1. Add all bytes of data.
2. Drop carry, take 2's complement of sum.

$25H + 62H + 3FH + 52H = \overset{drop}{(1)}18H$

0001 1000

1's comp 1110 0111

add 1 to LSB + 1

1110 1000 = 0xE8 checksum byte

b. appending checksum byte & sending to receiver.

$25H + 62H + 3FH + 52H + E8H = \overset{drop}{200}$ so sum is 0.

So, data is not corrupted.

c. $25H + 22H + 3FH + 52H + E8H = 1C0$

sum is not 0.

So, data is corrupted.

0X47 → G 71
0X42 → B 66

GoodLuck | Page No.
Date

three

Develop a C program to calculate the checksum byte for the data given in previous example.

```
#include<reg51.h>
void main()
{ unsigned char mydata[] ={0x25, 0x62, 0x3F, 0x52};
  unsigned char sum=0;
  unsigned char checksumbyte, x;
  for(x=0; x<4; x++)
   { P2 = mydata[x];
     sum = sum + mydata[x];
     P1 = sum; }
  checksumbyte = ~sum+1;
   P1 = checksumbyte; }
```

four

Develop a C program to perform checksum op° on prev. ex. If data is good, send ASCII character G to P0. Otherwise send ASCII character B to P0.

```
#include <reg51.h>
void main(){
  unsigned char mydata[] = {0x25, 0x62, 0x3F, 0x52, 0x38};
  unsigned char checksum=0;
  unsigned char x;
  for (x=0; x<5; x++)
   { checksum = checksum + mydata[x]; }
  if (checksum ==0)
     P0 = 'G';
  else
     P0 = 'B';
}
```

code keyword : to rom space utilization

when we declare variable we use RAM

direct P1 'H' not RAM space used.

GoodLuck | Page No.
Date

Compile at single step the following program on your 8051 simulator. Check the contents of 128 byte RAM space to locate the ASCII values.

A 0x41
B
F 0x46

memory window

```
#include <reg51.h>
void main()
{ unsigned char mydata[] = "ABCDEF";
  unsigned char i;
  for (i=0; i<6; i++)  P1 = mydata[i]; }
```

Compare & contrast following. Discuss adv. & disadv. of each.

```
#include <reg51.h>
void main()
{ P1 = 'H';
  P1 = 'E';
  P1 = 'L';
  P1 = 'L';
  P1 = 'O'; }
```

In this program the characters are sent one at a time. It is short & simple but individual characters are embedded in the system. If we change the characters entire program changes.

```
#include <reg51.h>
void main()
{ unsigned char mydata[] = "HELLO";
  unsigned char i;
  for (i=0; i<5; i++)
  P1 = mydata[i]; }
```

This program uses RAM data space to store array elements, therefore the size of the array is limited.

```
#include <reg51.h>
void main()
{ code unsigned char mydata[] = "HELLO";
  unsigned char;
  for(i=0; i<5; i++)
  P1 = mydata[i]; }
```

This program uses separate area of codespace for data. This allows the size of the array to be as long as you want if you have the on-chip ROM

00 - FF (255)
0000 - FFFF
(65,535)

TMOD } bit & byte accessible
TCON

GoodLuck | Page No.
Date

However, the more code space you use for data, the less space is left for program code.

NOTE:

Programs 2 & 3 are easily upgradable if we want to change string itself or make it longer. That is not the case for Program 1.

Timer Programming:

2 16-bit timers / counters - 8051 microcontroller.

T0: 15 TH0 8 7 TL0 0 → accessing as 2 8 bit
T1: 15 TH1 8 7 TL1 0

Special function registers
1. TMOD Timer mod control register
2. TCON

Timer 1 Timer 0

TMOD	7	6	5	4	3	2	1	0
	Gate	$C/\overline{T}$	M_1	M_0	Gate	$C/\overline{T}$	M_1	M_0

$C/\overline{T}$ = 0; timer
1; Counter

Gate = 0, software means of timer control.
= 1, hardware means.

- configure timer is different modes.

M_1 M_0 : Mode control units.

0	0	Mode 0 (13-bit)
0	1	Mode 1 (16-bit)
1	0	Mode 2 (8-bit) - Auto control mode
1	1	Mode 3 (Split timer)

longer delays
Gate controlled

TCON Timer Control Register.

- to start / stop timers & control external interrupts
- TMOD (mode setting), TCON actually controls operation
- runs the timers (TR0 / TR1) & handles overflow flags (TF0/TF1)

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

Indicate start / stop timers
timer overflow.

GoodLuck | Page No.
Date

What value has to be loaded on TMOD register
i). to configure timer 0 on mode 1.
ii). " Timer 1 & Timer 0 in mode 2.

7	6	5	4	3	2	1	0	
Gate	C/T̄	M1	M0	Gate	C/T̄	M1	M0	
i) 0	0	0	0	0	0	0	1	TMOD = 0x01.
ii) 0	0	1	0	0	0	1	0	TMOD = 0x22.

Mode 1 program
Find the timer's clock frequency & its time period for 8051 microcontroller with the following crystal frequencies.
i) 12 MHz ii) 16 MHz iii) 11.0592 MHz.

one machine cycle = 12 clock pulses.
Timer's frequency = XTAL (Crystal freq.) / 12.

Time period = 1 / timer's frequency

i) T.F = 12/12 = 1 MHz.

Time Period = 1/1MHz = 1×10^{-6} s = 1 µs

ii) T.F = 16/12 = 1.33 MHz

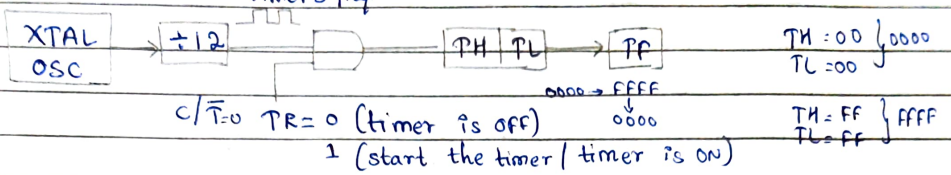
T.P = 1/1.33MHz = 0.75 µs.

iii) T.F = 11.0592/12 = 0.9216 MHz

T.P = 1/0.9216MHz = 1.085 µs

GoodLuck | Page No. | Date

Mode 1 Programming:



generate clock pulses - XTAL osc
TR=0
count increment

Lab 2019

Steps to generate delay using timer in mode 1.

1. Load TMOD with appropriate value to configure timer in mode 1. 0x10
2. Load TH & TL register with suitable value to generate the required amount of delay.
3. Start the timer using TR = 1.
4. Monitor timer overflow flag using while (TF==0);
5. Stop the timer using TR=0.
6. Clear timer overflow flag using TF=0
7. Repeat from step 2

Develop a C program to implement mod 4 counter on port 2 pins (P2.4 & P2.5) continuously with a delay of 50 ms. Use timer 1 in mode 1 to generate the delay. Assume XTAL (crystal freq.) = 11.0592 MHz

Delay = {[FFFF - YYXX] + 1} x 1.085 µs
50ms
TH TL
↳ T.P of 1 clock pulse

46083 - 1 = [FFFF - YYXX] YYXX = 4BFD
46082 = FFFF - YYXX
hexa decimal ↳ B402 = FFFF - YYXX TH = 4B
YYXX = FFFF - B402 TL = FD

Timer 1 in mod 1 0x10
0 0x01

FFFF - 19453 in decimal

GoodLuck | Page No.
Date

Calculate the max. delay generated using timer 1 in mod 1.
Assume XTAL = 11.0592 MHz.

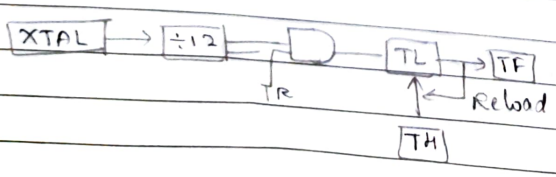
delay = {[FFFF - YYXX] + 1} × 1.085 μs
= {[FFFF - 0000] + 1} × 1.085 μs
= (FFFF + 1) × 1.085 μs
= 65536 × 1.085 μs
= 71.106 ms

NOTE: 1. Load TH & TL with 00 to generate the max. delay using mod 1. → 71.106 ms.
2. Load TH & TL with FF to generate the min. delay 1.085μ

```
#include <reg51.h>
void TIM1Delay();
void main()
{ while(1)
  { P2 = 0x00;
    TIM1Delay();
    P2 = 0x10;
    TIM1Delay();
    P2 = 0x20;
    TIM1Delay();
    P2 = 0x30;
    TIM1Delay();
  }
}
```

```
void TIM1Delay(){
TMOD = 0x10; //Timer1 in Mode1
TH1 = 0x4B;
TL1 = 0xFD; //Count for 50ms
TR1 = 1;                delay
while(TF1==0);
TR1 = 0; // Stop the timer
TF1 = 0; // Clear overflow flag
}
```

Mode 2 Programming:



mod 1 - auto reload 16 bit

mod 2 max delay 8 bit

GoodLuck | Page No. | Date

Implement mod 4 counter on Port 2 pins (P2.5)
continuously with delay of 25 ms. Use timer 1 in mod 2.

Delay = {[FF - YY] + 1} × 1.085 μs

only 8 bit TH only

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

↳ max to initial

23 - 1 = FF - YY

22 = FF - YY ⇒ 16 = FF - YY

YY = FF - 16 = E9

YY = E9

```
#include <reg51.h>
void TIM2Delay();
void main()
{ while(1)
  { P2=0x00;
    TIM2Delay();
    P2=0x10;
    TIM2Delay();
    P2=0x20;
    TIM2Delay();
    P2=0x30;
    TIM2Delay(); }
}
```

```
void TIM2Delay()
{ TMOD = 0x20;
  TH1 = 0xE9;
  TR1 = 1;
  while (TF1==0);
  TR1 = 0;
  TF1 = 0;
}
```

Generate the max. & min. delay for timer 1 using mode 2. Assume XTAL = 11.0592 MHz.

Delay = {[FF - YY] + 1} × 1.085 μs

max delay = {[FF - 00] + 1} × 1.085 μs

= 255 + 1 × 1.085 μs

= 256 × 1.085 μs

= 277.76 μs

GoodLuck | Page No.
Date

Develop a C program to toggle all the bits of port 1 continuously with a delay of 250 µs. use timer 0 in mode 2 to generate delay. Assume XTAL = 11.0592MHz

1. TMOD = 0x02.
2. TH0 = ?=0x1A(250 µs)
3. TR0 = 1
4. while (TF0 == 0);
5. TR0 = 0;
6. TF0 = 0;

delay = {[FF - YY] + 1} × 1.085 µs

$\frac{250\,\mu s}{1.085\,\mu s}$ = [FF - YY] + 1

230 - 1 = FF - YY

229 = FF - YY

YY = FF - E5 = 1A

```
#include<reg51.h>
void T0M2Delay();
void main()
{ while(1)
  { P1 = 0x00;
    T0M2 Delay();
    P1 = 0xFF;
    T0M2 Delay();
  }
}
```

```
void T0M2 Delay()
{ TMOD = 0x02; // Timer0 in mode2
  TH0 = 0x1A; // count for
  TR0 = 1; // start reg. delay timer0
  while (TF0 == 0);
    // monitor timer overflow flag
  TR0 = 0; // stop timer0
  TF0 = 0; // clear overflow
}          flag.
```

→

```
#include<reg51.h>
void main()
{ code unsigned char arr[] = "HELLO";
  unsigned char x;
    for (x=0; x<5; x++)
      { P1 = arr[x]; }
}
```