Unit – 4
Embedded ‘C’ programming

WHY PROGRAM 8051 IN C

- Embedded C is the most popular languages among Embedded Programmers for programming Embedded Systems. There are many popular programming languages like Assembly, BASIC, C++, Python etc. that are often used for developing Embedded Systems but Embedded C remains popular due to its efficiency, less development time and portability.
- Compilers produce hex files that is downloaded to ROM of microcontroller
- The size of hex file is the main concern
- Microcontrollers have limited on-chip ROM
- Code space for 8051 is limited to 64K bytes
- C programming is less time consuming, but has larger hex file size
- It is easier and less time consuming to write in C than Assembly
- C is easier to modify and update
- You can use code available in function libraries
- C code is portable to other microcontroller with little of no modification

Keywords in Embedded C

- A Keyword is a special word with a special meaning to the compiler (a C Compiler for example, is a software that is used to convert program written in C to Machine Code).
- For example, if we take the Keil's Cx51 Compiler (a popular C Compiler for 8051 based Microcontrollers) the following table lists out all the keywords associated with the Cx51 C Compiler.

_at_	alien	bdata
bit	code	compact
data	far	idata
interrupt	large	pdata
priority	reentrant	sbit
sfr	sfr16	small
task	using	xdata

Data Types in C

- A good understanding of C data types for 8051 can help programmers to create smaller hex files
- Unsigned char
- Signed char
- Unsigned int
- Signed int
- Sbit (single bit)
- Bit and sfr

Unsigned char

- The character data type is the most natural choice
- 8051 is an 8-bit microcontroller
- Unsigned char is an 8-bit data type in the range of

  0 – 255 (00 – FFH)
- One of the most widely used data types for the 8051 is
  - Counter value
  - ASCII characters
- C compilers use the signed char as the default if we do not put the keyword unsigned

Write an 8051 C program to send values 00 – FF to port P1.

Solution:

```
#include <reg51.h>
void main(void)
   {
     unsigned char z;
     for (z=0;z<=255;z++)
        P1=z;
   }
```

1. Pay careful attention to the size of the data
2. Try to use unsigned *char* instead of *int* if possible

Write an 8051 C program to send hex values for ASCII characters of 0, 1, 2, 3, 4, 5, A, B, C, and D to port P1.

Solution:

```
#include <reg51.h>   // Preprocessor Directive
void main(void)
   {
     unsigned char mynum[]="012345ABCD";
     unsigned char z;
     for (z=0;z<=10;z++)
        P1=mynum[z];
   }
```

Unsigned char (contd…)

- Write an 8051 C program to toggle all the bits of P1 continuously.

Solution:

```
//Toggle P1 forever
#include <reg51.h>
void main(void)
{                                   // Preprocessor Directive
for (;;)
{
P1=0x55;
P1=0xAA;
```

8	4	2	1	8	4	2	1	
P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	Hex Value
0	1	0	1	0	1	0	1	55 H
1	0	1	0	1	0	1	0	AA H

Signed char

- The signed char is an 8-bit data type use the MSB D7 to represent – or +
- Give values from –128 to +127
- We should stick with the unsigned char unless the data needs to be represented as signed numbers

- Write an 8051 C program to send values of –4 to +4 to port P1.

Solution:

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

Unsigned and Signed int

- The unsigned int is a 16-bit data type Takes a value in the range of 0 to 65535 (0000 – FFFFH)
- Define 16-bit variables such as memory addresses
- Set counter values of more than 256 Since registers and memory accesses are in 8-bit chunks, the misuse of int variables will result in a larger hex file
- Signed int is a 16-bit data type use the MSB D15 to represent – or +
- We have 15 bits for the magnitude of the number from –32768 to +32767

Unsigned and Signed int

Write an 8051 C program to toggle bit D0 of the port P1 (P1.0) 50,000 times.

Solution:

sbit keyword allows access to the single bits of the SFR registers

```
#include <reg51.h>
sbit MYBIT=P1^0;

void main(void)
  {
    unsigned int z;
    for (z=0;z<=50000;z++)
       {
         MYBIT=0;
         MYBIT=1;
       }
  }
```

Bit and sfr

- The bit data type allows access to single bits of bit-addressable memory spaces 20 – 2FH
- To access the byte-size SFR registers, we use the sfr data type

Data Type	Size in Bits	Data Range/Usage
unsigned char	8-bit	0 to 255
(signed) char	8-bit	-128 to +127
unsigned int	16-bit	0 to 65535
(signed) int	16-bit	-32768 to +32767
sbit	1-bit	SFR bit-addressable only
bit	1-bit	RAM bit-addressable only
sfr	8-bit	RAM addresses 80 – FFH only

TIME DELAY

- There are two way s to create a time delay in 8051 C
  - Using the 8051 timer
  - Using a simple for loop
- Three factors that can affect the accuracy of the delay
  - The 8051 design
  - The number of machine cycle
  - The number of clock periods per machine cycle

  The crystal frequency connected to the X1 – X2 input pins
- Compiler choice

  – C compiler converts the C statements and functions to Assembly language instructions

  – Different compilers produce different code

TIME DELAY

Write an 8051 C program to toggle bits of P1 continuously forever with some delay.

Solution:

```
//Toggle P1 forever with some delay in between
//"on" and "off"
#include <reg51.h>
void main(void)
   {
     unsigned int x;
     for (;;)                    //repeat forever
       {
          p1=0x55;
          for (x=0;x<40000;x++); //delay size
                                 //unknown
          p1=0xAA;
          for (x=0;x<40000;x++);
        }
   }
```

We must use the oscilloscope to measure the exact duration

8	4	2	1	8	4	2	1	
P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	Hex Value
0	1	0	1	0	1	0	1	55 H
1	0	1	0	1	0	1	0	AA H

TIME DELAY

Write an 8051 C program to toggle bits of P1 ports continuously with a 250 ms.

Solution:

```
#include <reg51.h>
void MSDelay(unsigned int);
void main(void)
   {
     while (1)                    //repeat forever
        {
          p1=0x55;
          MSDelay(250);
          p1=0xAA;
          MSDelay(250);
        }
   }

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

TIME DELAY

- Write an 8051 C program to toggle all the bits of P0, P1, and P2 continuously with a 250 ms delay. Use the sfr keyword to declare the port addresses

Solution:

```
//Accessing Ports as SFRs using sfr data type
sfr P0=0x80;
sfr P1=0x90;
sfr P2=0xA0;
void MSDelay(unsigned int);
void main(void)
{
while (1)
{
P0=0x55;
P1=0x55;
P2=0x55;
MSDelay(250);
P0=0xAA;
P1=0xAA;
P2=0xAA;
MSDelay(250);
}
}
```

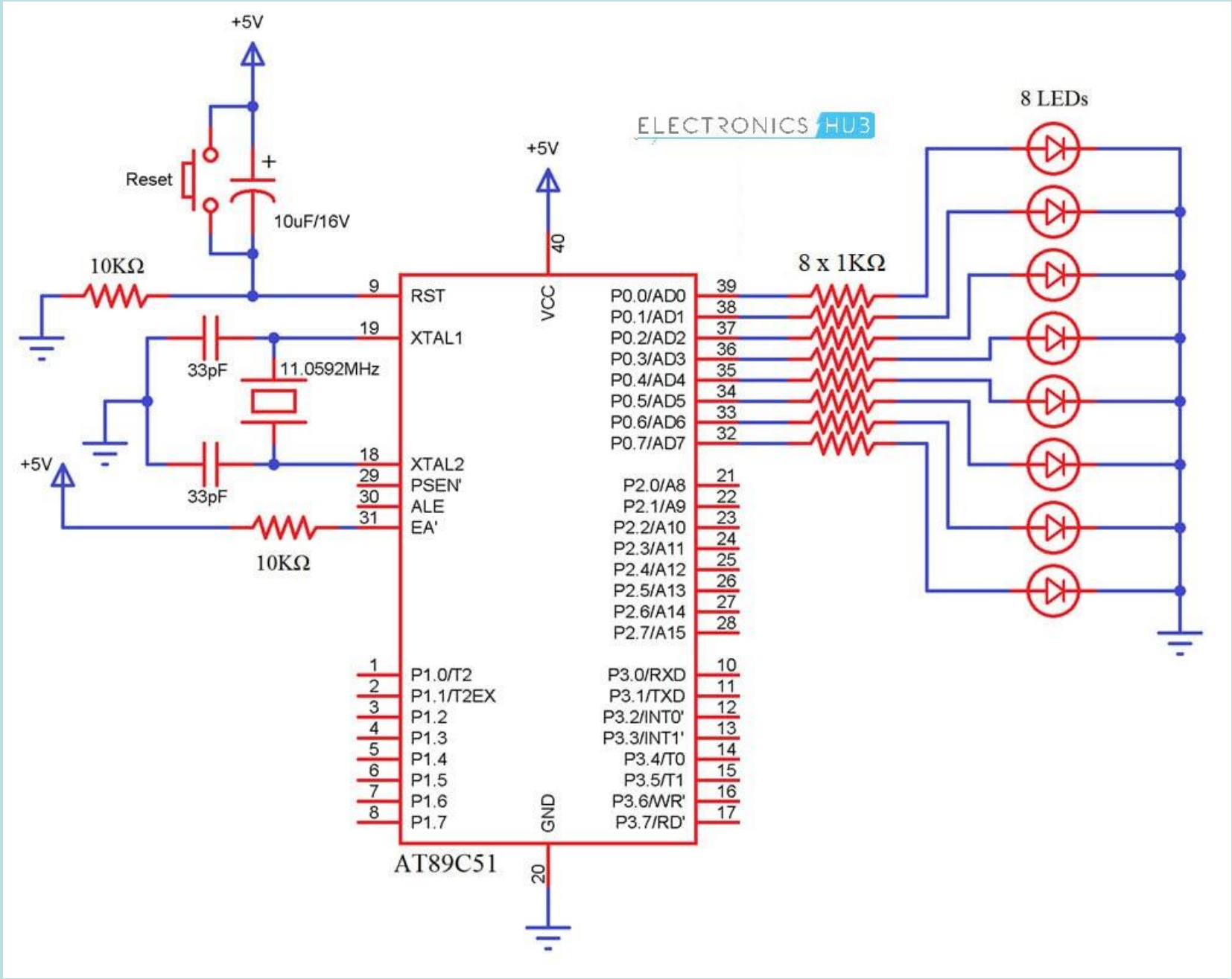
I/O PROGRAMMING

LEDs are connected to bits P1 and P2. Write an 8051 C program that shows the count from 0 to FFH (0000 0000 to 1111 1111 in binary) on the LEDs.

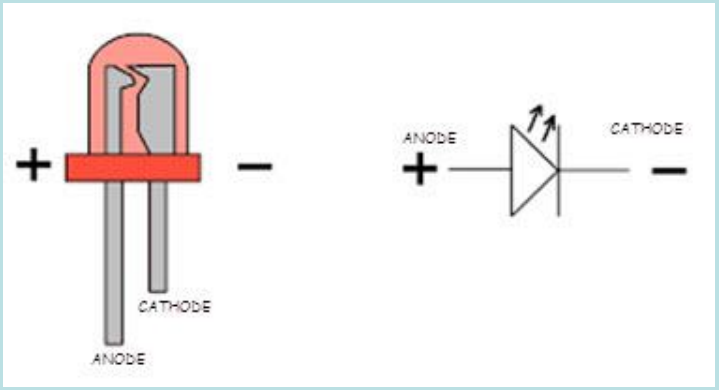
Solution:

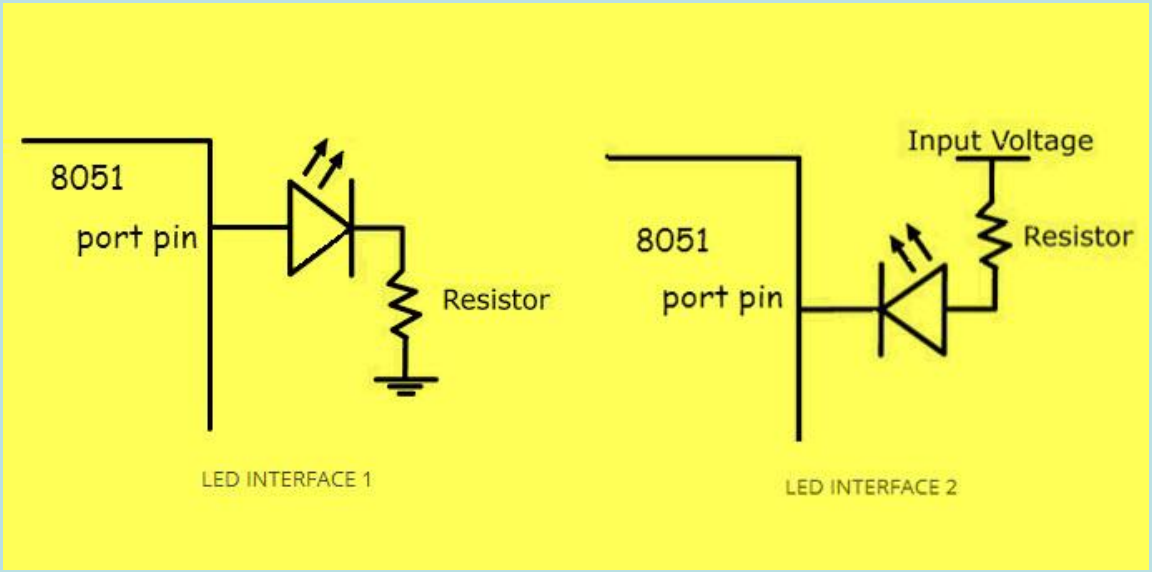
```
#include <reg51.h>
#defind LED P2;

void main(void)
  {
    P1=00;              //clear P1
    LED=0;              //clear P2
    for (;;)            //repeat forever
       {
         P1++;          //increment P1
         LED++;         //increment P2
       }
  }
```

Ports P0 – P3 are byte-accessable and we use the P0 – P3 labels as defined in the 8051/52 header file.

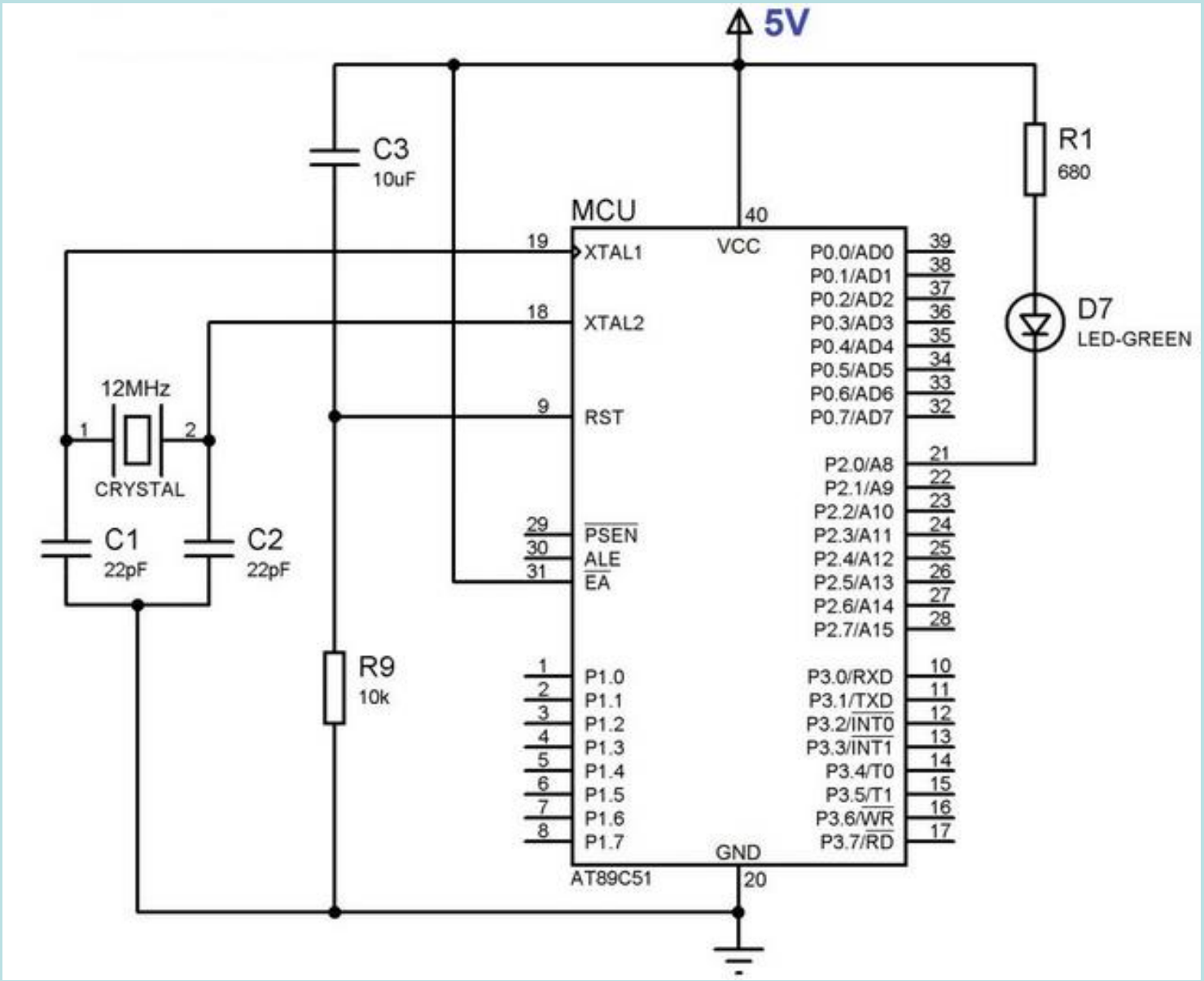

+5V
Reset
10uF/16V
10KΩ
33pF
11.0592MHz
33pF
+5V
10KΩ
ELECTRONICS HUB
+5V
40
9
RST
19
XTAL1
18
XTAL2
29
PSEN'
30
ALE
31
EA'
VCC
GND
20
AT89C51
1
P1.0/T2
2
P1.1/T2EX
3
P1.2
4
P1.3
5
P1.4
6
P1.5
7
P1.6
8
P1.7
39
P0.0/AD0
38
P0.1/AD1
37
P0.2/AD2
36
P0.3/AD3
35
P0.4/AD4
34
P0.5/AD5
33
P0.6/AD6
32
P0.7/AD7
21
P2.0/A8
22
P2.1/A9
23
P2.2/A10
24
P2.3/A11
25
P2.4/A12
26
P2.5/A13
27
P2.6/A14
28
P2.7/A15
10
P3.0/RXD
11
P3.1/TXD
12
P3.2/INT0'
13
P3.3/INT1'
14
P3.4/T0
15
P3.5/T1
16
P3.6/WR'
17
P3.7/RD'
8 x 1KΩ
8 LEDs

Interfacing single LED with 8051





Interfacing single LED with 8051




```
#include <reg51.h>
void main( )
{
    P2 =0xFF;

   P2_0=0;
}
```

Write an 8051 C program to get a byte of data form P1, wait 1/2 second, and then send it to P2.

Solution:

```
#include <reg51.h>
void MSDelay(unsigned int);

void main(void)
  {
    unsigned char mybyte;
    P1=0xFF;                    //make P1 input port
    while (1)
       {
         mybyte=P1;             //get a byte from P1
         MSDelay(500);
         P2=mybyte;             //send it to P2
       }
  }
```

- **To program any port to be set as output on 89cxx series microcontroller.**

```
#include <reg51.h>          // for 89c2051,89c4051,89c51, 89s51 controller
#include <reg52.h>          //for 89c52,89s52 controller

void main( )
{
   P0=0x00;    //Set port 0 to 0v logic
   P1=0x00;    //Set port 1 to 0v logic
   P2=0x00;    //Set port 2 to 0v logic
   P3=0x00;    //Set port 3 to 0v logic

   P1=0x01;       //set port 1 = 1. That is 00000001 on 8bit Port 1
   P2_1=0x01;  // set port 2 bit 1 =1. That is 00000010 on 8bit Port 2
   P3=P1;         //Set port 3 same as port 1. That is 00000001 on 8bit port 3.
   while(1)
   {
        //Go to infinity to stop the process here. If there is no infinity loop the code gets
restarted again.
   }

}
```

➢**To blink a pin or port (the below code will turn on and off port 1 pin 0)**

```
#include <reg51.h>  // for 89c2051,89c4051,89c51, 89s51 controller

void delay(int n);          //delay routine
{
    int i,j;
    for(i=0;i<=100;i++)
    {
       for(j=0;j<=n;j++);   //the loop will be occurring at n * 100 times
     }
}
void main( )
{
    P0=0x00;    //Set port 0 to 0v logic
    P1=0x00;    //Set port 1 to 0v logic
    P2=0x00;    //Set port 2 to 0v logic
    P3=0x00;    //Set port 3 to 0v logic

    while(1)
    {
            P1=0x01;        //set port 1 = 1. That is 00000001 on 8bit Port 1
         delay(1000);       //Call delay routine to pause port 1 state
         P1 = 0x00;         //set port 1 = 0. That is 00000000 on 8bit port 1
         delay(1000);       // Call delay routine to pause port 1 state
    }
}
```

Write an 8051 C program to get a byte of data form P0. If it is less than 100, send it to P1; otherwise, send it to P2.

Solution:

```
#include <reg51.h>

void main(void)
  {
    unsigned char mybyte;
    P0=0xFF;                    //make P0 input port
    while (1)
       {
         mybyte=P0;             //get a byte from P0
         if (mybyte<100)
            P1=mybyte;          //send it to P1
         else
            P2=mybyte;          //send it to P2
       }
  }
```

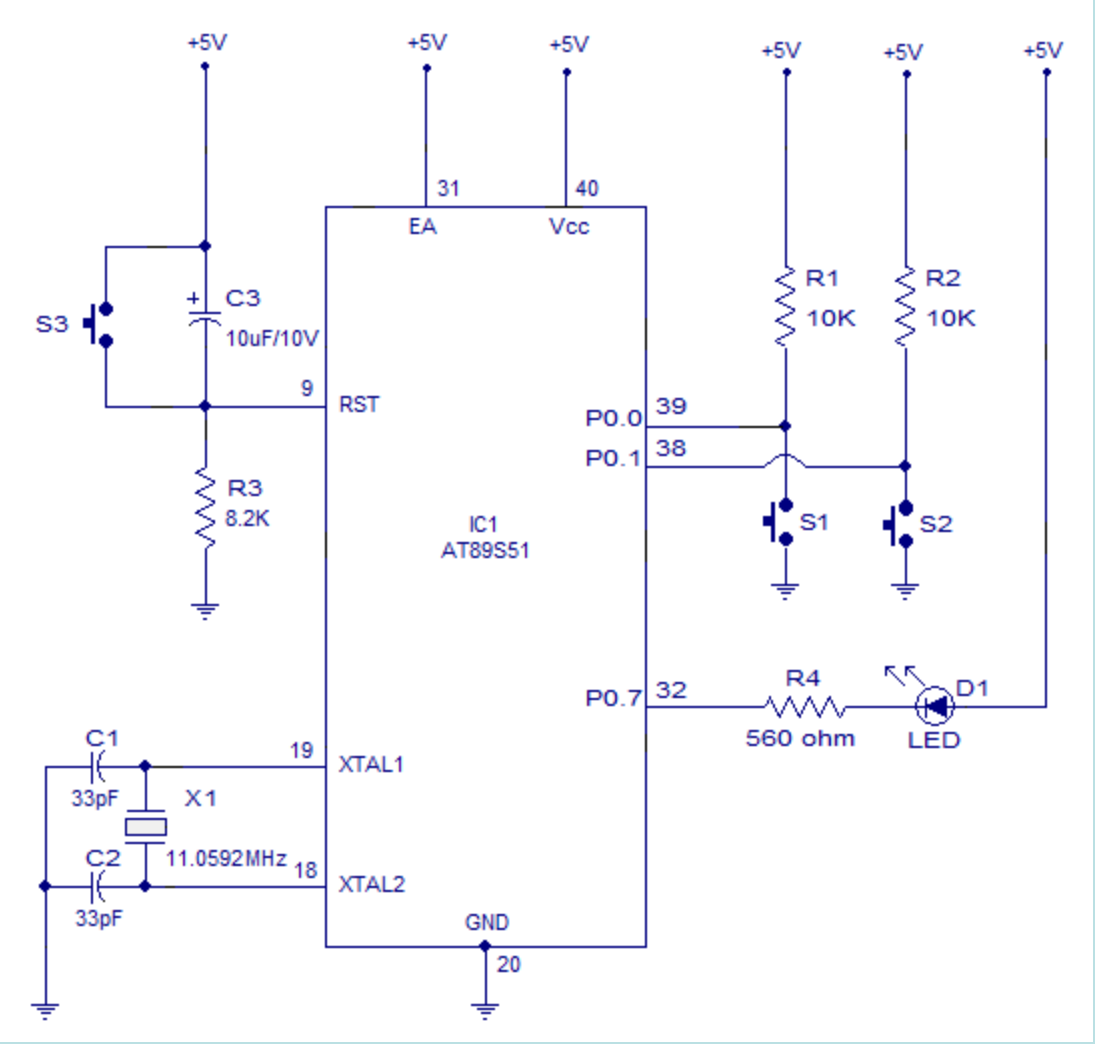
Write an 8051 C program to monitor bit P1.5. If it is high, send 55H to P0; otherwise, send AAH to P2.

Solution:

```
#include <reg51.h>
sbit mybit=P1^5;

void main(void)
  {
    mybit=1;                    //make mybit an input
    while (1)
       {
         if (mybit==1)
            P0=0x55;
         else
            P2=0xAA;
       }
  }
```

Switch Interface with microcontroller

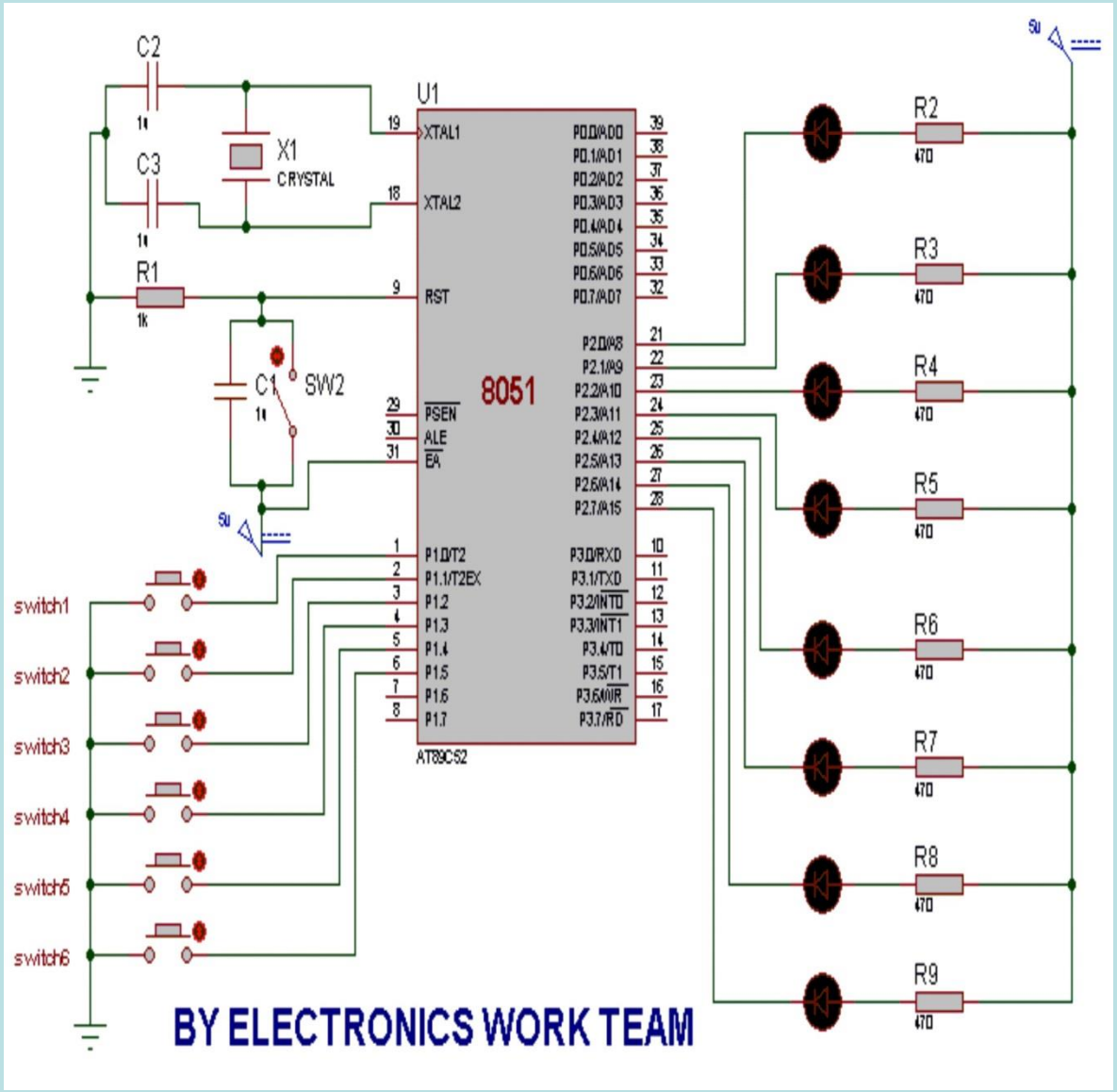
```
#include<stdio.H>
#include<at89x51.H>

void main()
{
while(1)
{
P0=0xFF;      // Port 0 as input
if(P0_0==0)  // if S1 Pressed
{
P0_7=0;          // LED Starts glow
}
if(P0_1==0)     // if S2 Pressed
{
P0_7=1;          //LED OFF
}
}
}
```



Write a program for the following requirements:

SWITCH-1 IS ON->ALL THE LED'S HAVE TO GLOW
SWITCH-2 IS ON->ALTERNATE LED'S HAVE TO GLOW

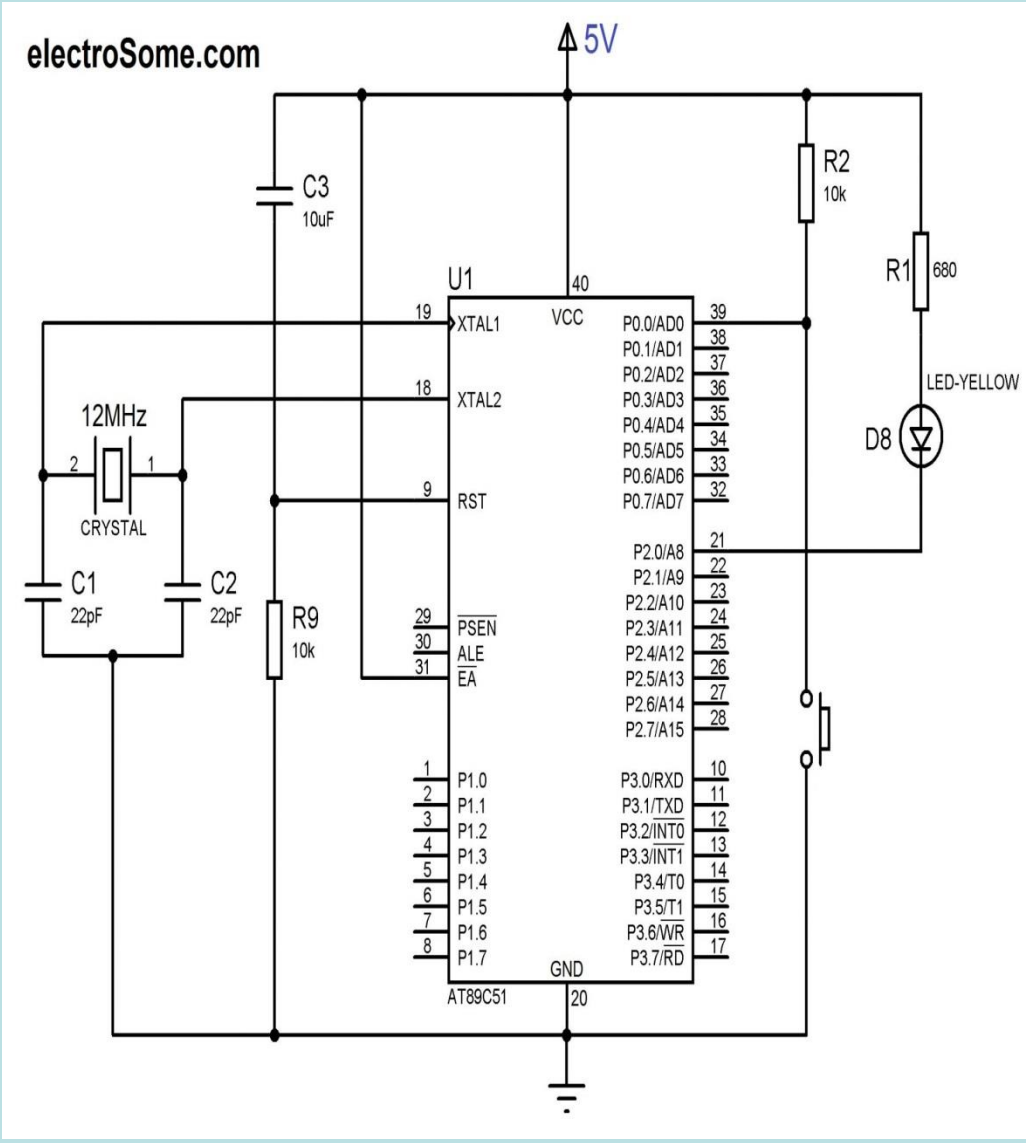
```
#include<stdio.H>
#include<at89x51.H>

void main()
{
while(1)
{
if(P1_0==0)    // if S1 Pressed
 {
P2=0x00;
}
else if(P1_1==0) // if S2 Pressed

{
P2=0xAA;
}
}
}
```



A door sensor or Switch is connected to the P0.0 pin, and a buzzer or LED is connected to P2.0. Write an 8051 C program to monitor the door sensor, and when it opens, sound the buzzer.

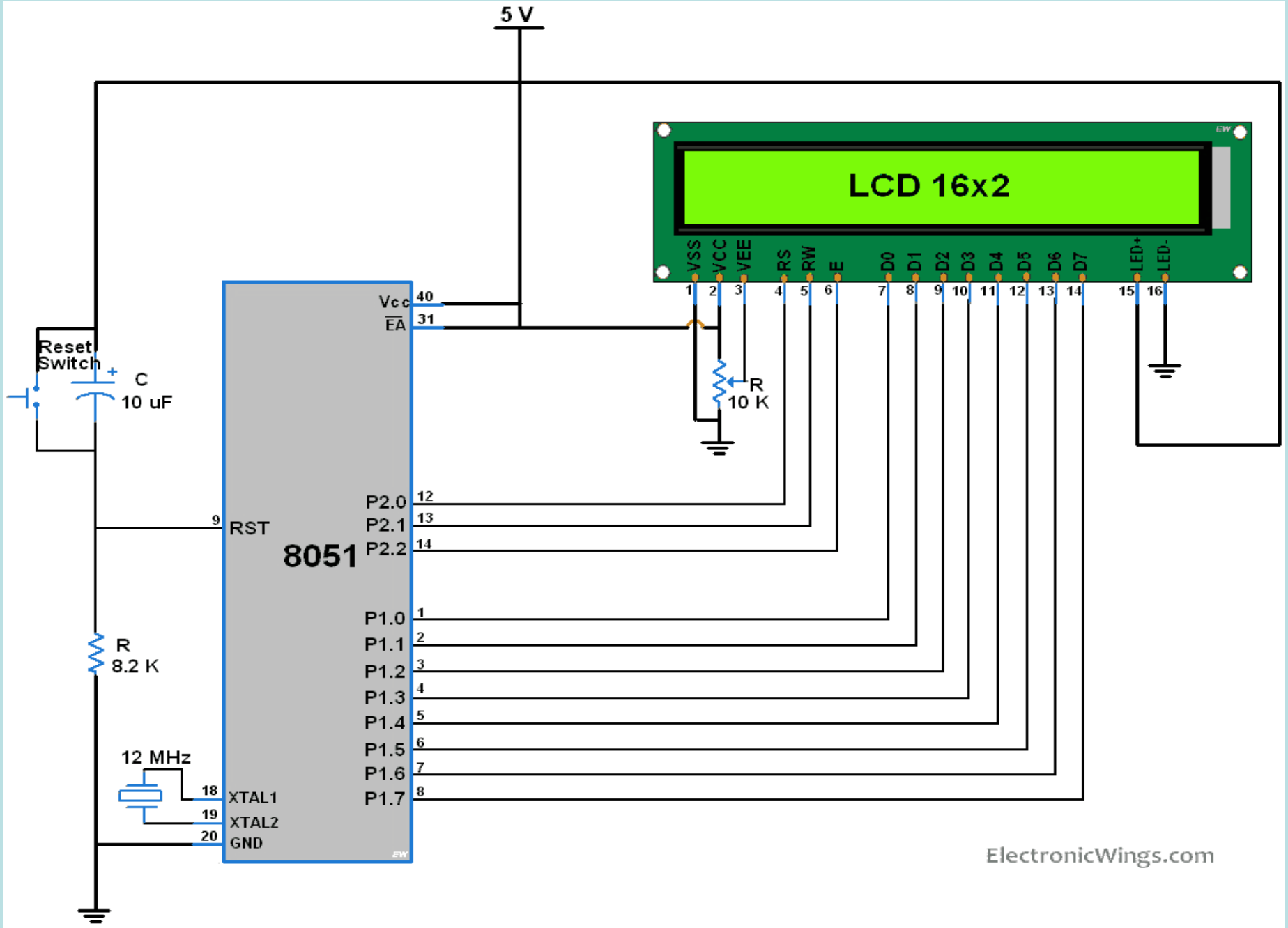
Solution:

```
#include <reg51.h>

void MSDelay(unsigned int);
sbit Dsensor=P0^0;
sbit Buzzer=P2^0;

void main(void)
{
Dsensor=1;              //make P0.0 as input
while (1)
{
while (Dsensor==1) //while it opens
{
Buzzer=0;
MSDelay(200);
Buzzer=1;
MSDelay(200);
}
}
}
```

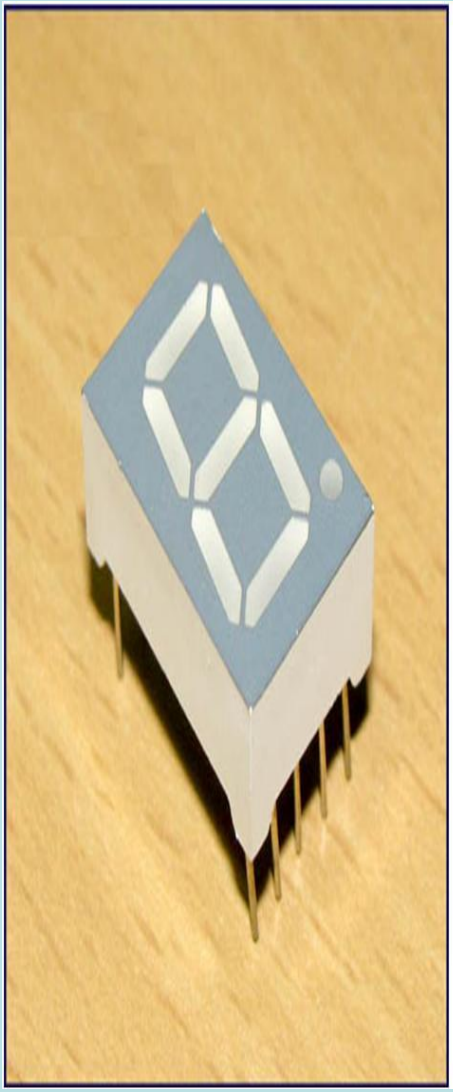


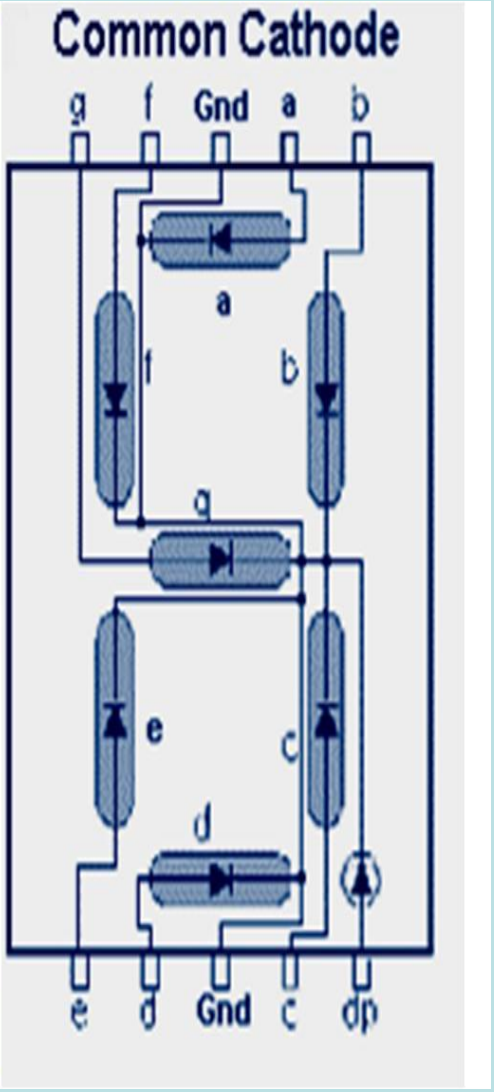
- The data pins of an LCD are connected to P1. The information is latched into the LCD whenever its Enable pin goes from high to low. Write an 8051 C program to display “Embedded Systems Design” to This LCD.
- **Solution:**

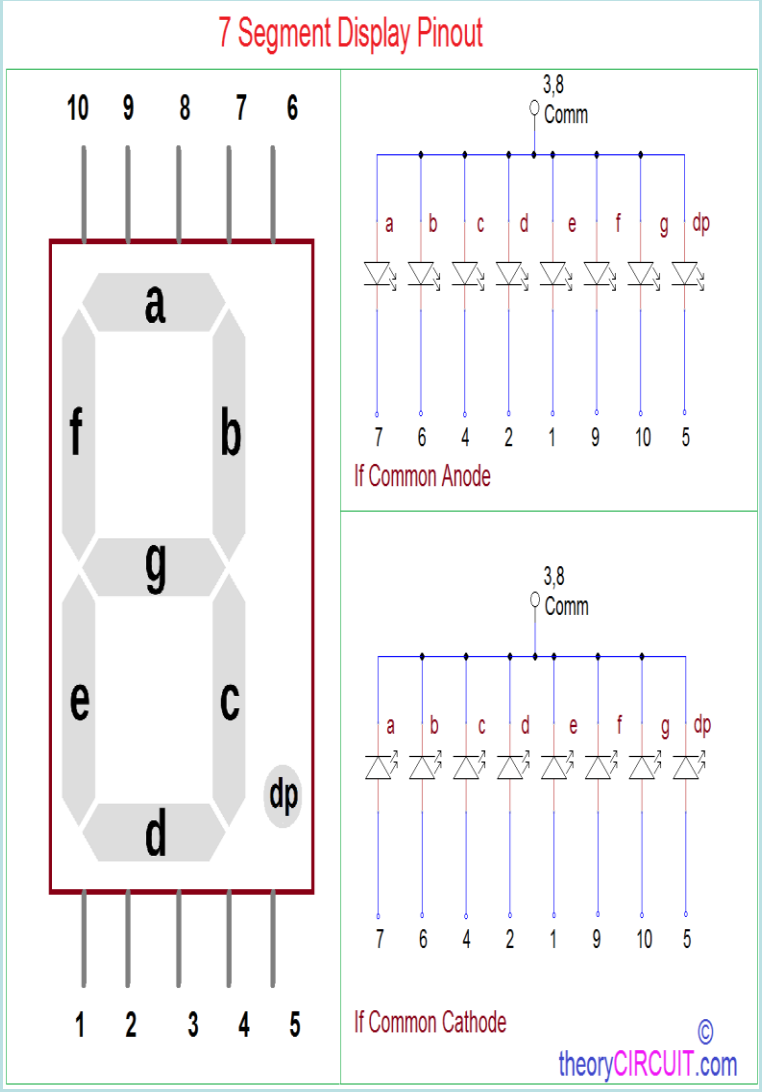
```
#include <reg51.h>
#define LCDData P1      //LCD Data declaration
sbit En=P2^2;           //the enable pin
void main(void)
{
unsigned char message[]=“Embedded Systems Design”;
unsigned char z;
for (z=0;z<23;z++)      //send 22 characters (including Space)
{
LCDData=message[z];
En=1;           //a high-
En=0;           //-to-low pulse to latch data
}
}
```


5 V
LCD 16x2
VSS
VCC
VEE
RS
RW
E
D0
D1
D2
D3
D4
D5
D6
D7
LED+
LED-
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
R
10 K
Reset
Switch
C
10 uF
8051
Vcc
40
EA
31
RST
9
P2.0
12
P2.1
13
P2.2
14
P1.0
1
P1.1
2
P1.2
3
P1.3
4
P1.4
5
P1.5
6
P1.6
7
P1.7
8
R
8.2 K
12 MHz
18
XTAL1
19
XTAL2
20
GND
ElectronicWings.com

7 Segment Display Interface







Common Anode Hex values

Digit	h	g	f	e	d	c	b	a	Hex Value
0	1	1	0	0	0	0	0	0	0xC0
1	1	1	1	1	1	0	0	1	0xF9
2	1	0	1	0	0	1	0	0	0xA4
3	1	0	1	1	0	0	0	0	0xB0
4	1	0	0	1	1	0	0	1	0x99
5	1	0	0	1	0	0	1	0	0x92
6	1	0	0	0	0	0	1	0	0x82
7	1	1	1	1	1	0	0	0	0xF8
8	1	0	0	0	0	0	0	0	0x80
9	1	0	0	1	0	0	0	0	0x90

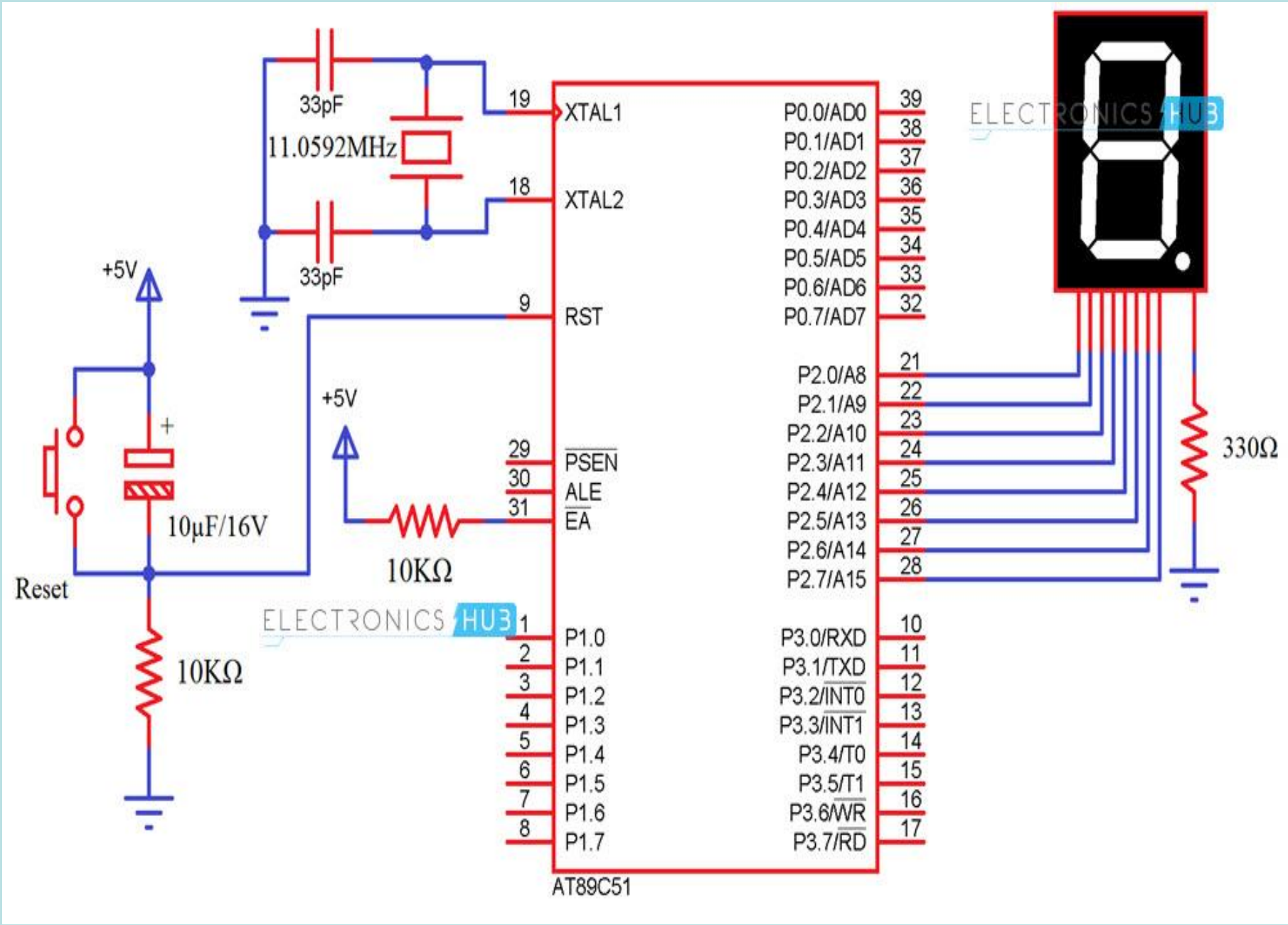
(P2.7) dp	(P2.6) g	(P2.5) f	(P2.4) e	(P2.3) d	(P2.2) C	(P2.1) b	(P2.0)a	Hex Value	Digit
1	1	0	0	0	0	0	0	C0 H	**0**
1	0	0	1	1	1	1	1	F9 H	**1**

Common Cathode Hex values

DIGIT	DP	G	F	E	D	C	B	A	HEX VALUE
0	0	0	1	1	1	1	1	1	0x3f
1	0	0	0	0	0	1	1	0	0x06
2	0	1	0	1	1	0	1	1	0x5b
3	0	1	0	0	1	1	1	1	0x4f
4	0	1	1	0	0	1	1	0	0x66
5	0	1	1	0	1	1	0	1	0x6d
6	0	1	1	1	1	1	0	1	0x7d
7	0	0	0	0	0	1	1	1	0x07
8	0	1	1	1	1	1	1	1	0x7f
9	0	1	1	0	0	1	1	1	0x67

(P2.7) dp	(P2.6) g	(P2.5) f	(P2.4) e	(P2.3) d	(P2.2) C	(P2.1) b	(P2.0)a	Hex Value	Digit
0	0	1	1	1	1	1	1	3F	**0**
0	0	0	0	0	1	1	0	06	**1**

```
#include <reg51.h>
void DELAY_ms(unsigned int ms_Count)
{
unsigned int i,j;
for(i=0;i<ms_Count;i++)
{
for(j=0;j<100;j++);
}
}
int main()
{
char seg_code[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
int i;
while (1)
{
for (i = 0; i <= 9; i++)  // loop to display 0-9
{
P2 = seg_code[i];
DELAY_ms(1000);
}
}
}
```


33pF
11.0592MHz
33pF
19
XTAL1
18
XTAL2
9
RST
29
PSEN
30
ALE
31
EA
1
P1.0
2
P1.1
3
P1.2
4
P1.3
5
P1.4
6
P1.5
7
P1.6
8
P1.7
AT89C51
39
P0.0/AD0
38
P0.1/AD1
37
P0.2/AD2
36
P0.3/AD3
35
P0.4/AD4
34
P0.5/AD5
33
P0.6/AD6
32
P0.7/AD7
21
P2.0/A8
22
P2.1/A9
23
P2.2/A10
24
P2.3/A11
25
P2.4/A12
26
P2.5/A13
27
P2.6/A14
28
P2.7/A15
10
P3.0/RXD
11
P3.1/TXD
12
P3.2/INT0
13
P3.3/INT1
14
P3.4/T0
15
P3.5/T1
16
P3.6/WR
17
P3.7/RD
+5V
+5V
10µF/16V
Reset
10KΩ
10KΩ
330Ω
ELECTRONICS HUB
ELECTRONICS HUB

LOGIC OPERATIONS

- Logical operators
  - AND (&&), OR (||), and NOT (!)
  - Bit-wise operators
- AND (&), OR (|), EX-OR (^), Inverter (~),
- Shift Right (>>), and Shift Left (<<)
  - These operators are widely used in software engineering for embedded systems and control

Bit-wise Logic Operators for C

		AND	OR	EX-OR	Inverter
A	B	A&B	A|B	A^B	~B
0	0	0	0	0	1
0	1	0	1	1	0
1	0	0	1	1	
1	1	1	1	0	

Run the following program on your simulator and examine the results.

Solution:

```
#include <reg51.h>

void main(void)
{
P0=0x35 & 0x0F; //ANDing
P1=0x04 | 0x68;  //ORing
P2=0x54 ^ 0x78;  //XORing
P0=~0x55;        //inversing
P1=0x9A >> 3;    //shifting right 3
P2=0x77 >> 4;    //shifting right 4
P0=0x6 << 4;     //shifting left 4
}
```

Write an 8051 C program to toggle all the bits of P0 and P2 continuously with a 250 ms delay. Using the inverting and Ex-OR operators, respectively.

Solution:

```
#include <reg51.h>

void MSDelay(unsigned int);
void main(void)
{
P0=0x55;
P2=0x55;
while (1)
{
P0=~P0;
P2=P2^0xFF;
MSDelay(250);
}
}
```

Write an 8051 C program to get bit P1.0 and send it to P2.7 after inverting it.

Solution:

```
#include <reg51.h>
sbit inbit=P1^0;     // P1.0 = 1
sbit outbit=P2^7;  // P2.7 =1
bit sendbit;

void main(void)
{
while (1)
{
sendbit =inbit;       //get a bit from P1.0 which is going to be send
outbit=~ sendbit;  //invert it and send it to P2.7, so P2.7=0
}
}
```

DATA CONVERSION

Write an 8051 C program to convert 11111101 (FD hex) to decimal and display the digits on P0, P1 and P2.

Solution:

```
#include <reg51.h>
void main(void)
{
unsigned char x,binbyte,d1,d2,d3;
binbyte=0xFD;      // hex digit
x=binbyte/10;      // divided by 10
d1=binbyte%10;   //   find reminder for LSD
d2=x%10;           // for middle digit
d3=x/10;           // for MSD
P0=d1;
P1=d2;
P2=d3;
}
```

Write an 8051 C program to convert ASCII digits of ‘4’ and ‘7’ to packed BCD and display them on P1.

Solution:

```
#include <reg51.h>
void main(void)
{
unsigned char bcdbyte;
unsigned char w=‘4’;
unsigned char z=‘7’;
w=w&0x0F;
w=w<<4;
z=z&0x0F;
bcdbyte=w|z;
P1=bcdbyte;
}
```

DATA SERIALIZATION

- Serializing data is a way of sending a byte of data one bit at a time through a single pin of microcontroller
- Using the serial port
- Transfer data one bit at a time and control the sequence of data and spaces in between them
- In many new generations of devices such as LCD, ADC, and ROM the serial versions are becoming popular since they take less space on a PCB

Write a C program to send out the value 44H serially one bit at a time via P1.0. The LSB should go out first.

Solution:

```
#include <reg51.h>
sbit P1b0=P1^0;
sbit regALSB=ACC^0;

void main(void)
{
unsigned char conbyte=0x44;
unsigned char x;
ACC=conbyte;
for (x=0;x<8;x++)
{
P1b0=regALSB;
ACC=ACC>>1;
}
}
```

Write a C program to send out the value 44H serially one bit at a time via P1.0. The MSB should go out first.

Solution:

```
#include <reg51.h>
sbit P1b0=P1^0;
sbit regAMSB=ACC^7;

void main(void)
{
unsigned char conbyte=0x44;
unsigned char x;
ACC=conbyte;
for (x=0;x<8;x++)
{
P1b0=regAMSB;
ACC=ACC<<1;
}
}
```

- Write a C program to bring in a byte of data serially one bit at a time
- via P1.0. The LSB should come in first.
- **Solution:**

```
#include <reg51.h>
sbit P1b0=P1^0;
sbit ACCMSB=ACC^7;
bit membit;
void main(void)
{
unsigned char x;
for (x=0;x<8;x++)
{
membit=P1b0;
ACC=ACC>>1;
ACCMSB=membit;
}
P2=ACC;
}
```

Write a C program to bring in a byte of data serially one bit at a time via P1.0. The MSB should come in first.

Solution:

```
#include <reg51.h>
sbit P1b0=P1^0;
sbit regALSB=ACC^0;
bit membit;
void main(void)
{
unsigned char x;
for (x=0;x<8;x++)
{
membit=P1b0;
ACC=ACC<<1;
regALSB=membit;
}
P2=ACC;
}
```

PROGRAMMING TIMERS IN C

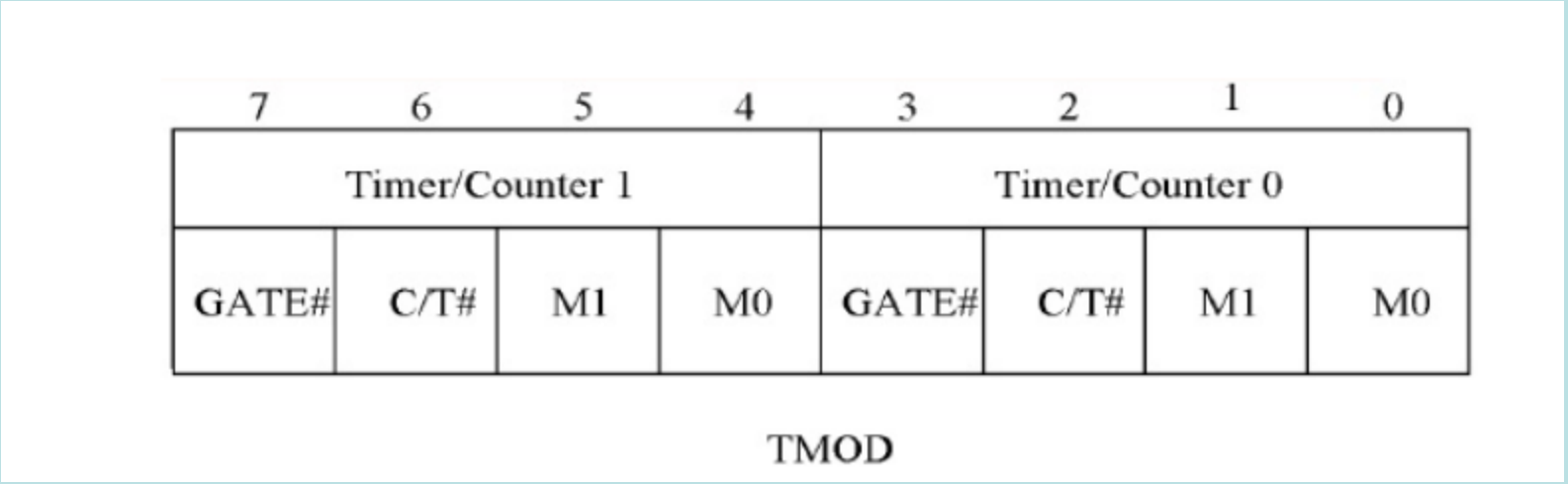
Write an 8051 C program to toggle all the bits of port P1 continuously with some delay in between. Use Timer 0, 16-bit mode to generate the delay.

Solution:

```
#include <reg51.h>
void T0Delay(void);
void main(void){
while (1) {
P1=0x55;
T0Delay();
P1=0xAA;
T0Delay();
}
}
void T0Delay(){
TMOD=0x01;
TL0=0x00;
TH0=0x35;
TR0=1;
while (TF0==0);
TR0=0;
TF0=0;
}
```

FFFFH – 3500H = CAFFH = 51967 + 1 = 51968
51968 × 1.085 µs = 56.384 ms is the approximate delay

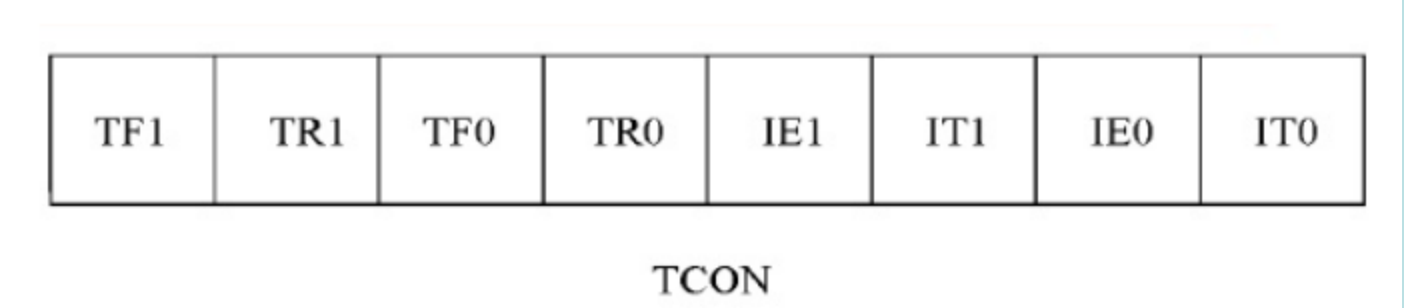
TMOD (Timer Mode Register) is a non-bit-addressable, 8-bit register



- Lower 4 bits are for **Timer0**
- Upper 4 bits are for **Timer1**
- **GATE bit** is used for choice of **internal or external control**
- **GATE=0 is for internal control,** start and stop are controlled by software
- **GATE=1 is for external control,** start and stop are controlled by software and external source
- C/T bit decides about timer type**: interval timer or counter**

TCON (Timer Control Register)

It is a bit-addressable, 8-bit register where 4 upper bits are responsible for timers/counters



TCON

TF1 : Timer 1 Overflow Flag.

TF1=1 : It is set by **hardware when timer/counter 1 overflows**.

TF1=0 : It is cleared by hardware processor vectors to the **Interrupt service routine.`**

TF0 : Timer 0 Overflow Flag

TF0=1 : Program software to enable timer 0 to count.

TF0=0 : It is cleared 0 by software to halt timer.

TR1 : Timer 1 Run control bit.

TR1=1 software program to enable timer 1 to count.

TR1=0 it is cleared to 0 by program to halt timer.

Write an 8051 C program to toggle only bit P1.5 continuously every 50 ms. Use Timer 0, mode 1 (16-bit) to create the delay. Test the program on the (a) AT89C51 and (b) DS89C420.

Solution:

```
#include <reg51.h>
void T0M1Delay(void);
sbit mybit=P1^5;
void main(void){
while (1) {
mybit=~mybit;
T0M1Delay();
}
}
void T0M1Delay(void){
TMOD=0x01;
TL0=0xFD;
TH0=0x4B;
TR0=1;
while (TF0==0);
TR0=0;
TF0=0;
}
```

FFFFH – 4BFDH = B402H = 46082 + 1 = 46083

46083 × 1.085 µs = 50 ms

Write an 8051 C program to toggle all bits of P2 continuously every 500 ms. Use Timer 1, mode 1 to create the delay.

Solution:

```
//tested for DS89C420, XTAL = 11.0592 MHz
#include <reg51.h>
void T1M1Delay(void);
void main(void){
unsigned char x;
P2=0x55;
while (1) {
P2=~P2;
for (x=0;x<20;x++)
T1M1Delay();
}
}
void T1M1Delay(void){
TMOD=0x10;
TL1=0xFE;
TH1=0xA5;
TR1=1;
while (TF1==0);
TR1=0;
TF1=0;
}
```

A5FEH = 42494 in decimal 65536 – 42494 = 23042
23042 × 1.085 μs = 25 ms and
20 × 25 ms = 500 ms

THANK YOU