Embedded Systems

ECE 4010

- Ankur Beohar
SEEE, VIT Bhopal

8051 PROGRAMMING IN C

8051 PROGRAMMING IN C – Time Delay (Software)

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

```
#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

8051 PROGRAMMING IN C – Time Delay (Software)

Write an 8051 C program to toggle bits of P1 ports continuously with a 100 ms.
Crystal Frequency = 12Mhz, 8051 – 1 instruction cycle = 12 clock cycles

```
#include <reg51.h>
void MSDelay(unsigned int);
void main(void)
{
while (1)
{
p1=0x55;
MSDelay(100);
p1=0xAA;
MSDelay(100);
}}
```

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

8051 PROGRAMMING IN C – Time Delay (Software)

Write an 8051 C program to toggle bits of P1 ports continuously with a 100 ms. Crystal Frequency = 12Mhz, 8051 – 1 instruction cycle = 12 clock cycles

```
Time for 1 Instruction
Cycle = 12 / 12 Mhz = 1µs

Y * 1µs = 100 ms

Y = 100 ms/ 1µs

Y = 10000
```

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

8051 PROGRAMMING IN C – Time Delay (Software)

Write an 8051 C program to toggle bits of P1 ports continuously with a 100 ms. Crystal Frequency = 12Mhz, 8051 – 1 instruction cycle = 12 clock cycles

```
#include <reg51.h>
void MSDelay(unsigned int);
void main(void)
{
while (1)
{
p1=0x55;
MSDelay(100);
p1=0xAA;
MSDelay(100);
}}
```

```
void MSDelay(unsigned int
itime)
{
unsigned int i,j;
for (i=0;i<itime;i++)
for (j=0;j<1000;j++);
}
```

8051 PROGRAMMING IN C – Time Delay (Software)

Write an 8051 C program to toggle bits of P1 ports continuously with a 20 ms.
Crystal Frequency = **11.0592 Mhz**, 8051 – 1 instruction cycle = 12 clock cycles

```
#include <reg51.h>
void MSDelay(unsigned int);
void main(void)
{
while (1)
{
p1=0x55;
MSDelay(100);
p1=0xAA;
MSDelay(100);
}}
```

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

8051 PROGRAMMING IN C – Input Output Ports

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

```
#include <reg51.h>
void MSDelay(unsigned int);
void main(void)
{
------ Logic-------------
}
```

```
------Logic----------
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
}
```

8051 PROGRAMMING IN C – Bit Addressing

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

```
#include <reg51.h>
sbit mybit=P2^4;

void main(void){
while (1){
mybit=1;
mybit=0;
}}
```

Ports P0 – P3 are bit- addressable and we use *sbit* data type to access a single bit of P0 - P3

Use the **Px^y** format, where x is the port 0, 1, 2, or 3 and y is the bit 0 – 7 of that port

8051 PROGRAMMING IN C – Bit Addressing + Delay (Software)

Write an 8051 C program to toggle only bit P2.4 every 100ms without disturbing the rest of the bits of P2.
Crystal Frequency = 12Mhz, 8051 – 1 instruction cycle = 12 clock cycles

```
#include <reg51.h>
sbit mybit=P2^4;
void MSDelay(unsigned int);

void main(){
while (1){
mybit= ~mybit;
MSDelay(100);
}}
```

```
void MSDelay(unsigned int
itime)
{
unsigned int i,j;
for (i=0;i<itime;i++)
for (j=0;j<1000;j++);
}
```

8051 PROGRAMMING IN C –Delay (Hardware)

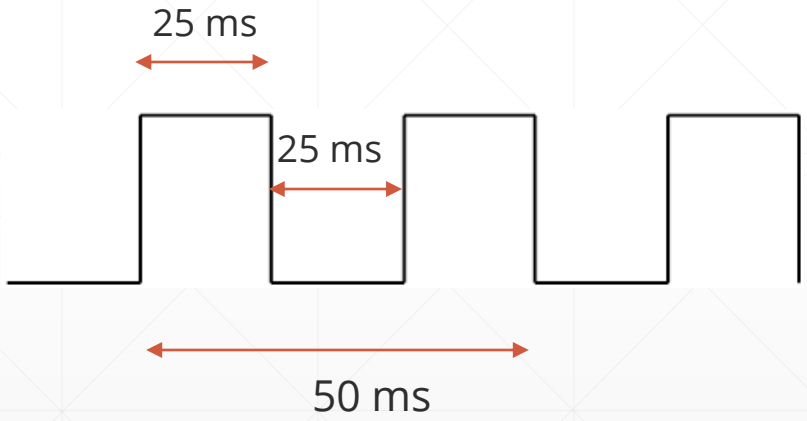
Write an 8051 C program to generate frequency of 20Hz on Pin P2.4
Crystal Frequency = 11.0592Mhz, 8051 – 1 instruction cycle = 12 clock cycles

```
Frequency = 20Hz, Time = 1/20 = 50ms
Time = 50ms = 25ms (ON) + 25ms (OFF)

Delay of 25 ms

Timer 0, Mode 1

(Use 8051 hardware ref manual for registers)
```



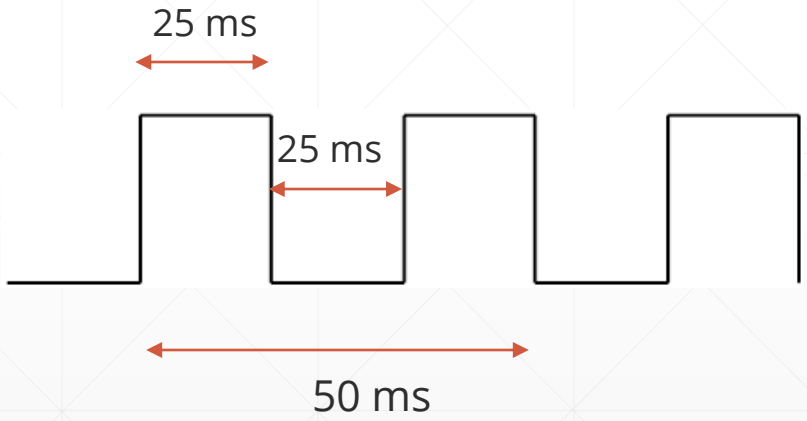
8051 PROGRAMMING IN C –Delay (Hardware)

Write an 8051 C program to generate frequency of 20Hz on Pin P2.4
Crystal Frequency = 11.0592Mhz, 8051 – 1 instruction cycle = 12 clock cycles

```
Delay of 25 ms

Timer 0, Mode 1

TMOD – Set timer/counter + mode
TCON – TR0, TF0
TR0 – Start timer
TF0 – Timer overflow
```



8051 PROGRAMMING IN C –Delay (Hardware)

Write an 8051 C program to generate frequency of 20Hz on Pin P2.4
Crystal Frequency = 11.0592Mhz, 8051 – 1 instruction cycle = 12 clock cycles

```
#include <reg51.h>
sbit mybit = P2^4;
void MSDelay(void);

void main(){
      TMOD = 0x01;
      while(1){
            mybit = ~mybit;
            MSDelay();
      }}
```

```
void MSDelay()
{
      TL0 = Lower Byte;
      TH0 = Upper Byte;
      TR0 = 1;
      while (TF0 ==0);
      TR0 = 0;
      TF0 = 0;
}
```

8051 PROGRAMMING IN C –Delay (Hardware)

Write an 8051 C program to generate frequency of 20Hz on Pin P2.4
Crystal Frequency = 11.0592MHz, 8051 – 1 instruction cycle = 12 clock cycles

Delay Calculation

XTAL = 11.0592 MHz
Time for 1 instru = 12/11.0592MHz
Time for 1 instru = 1.085 µs
Delay of 25ms = 25/1.085 µs = **23037**
Count = 65536 – 23037 = 42499
42499 = A5D1 (Hex)
TH0 = A5, TL0 = D1

```
void MSDelay()
{
        TL0 = D1;
        TH0 = A5;
        TR0 = 1;
        while (TF0 ==0);
        TR0 = 0;
        TF0 = 0;
}
```

8051 PROGRAMMING IN C –Interrupts

- Interrupts are useful in many cases wherein the process simply wants to continue doing its main job and other units(timers or external events) seek its attention when required
- In other words, the microcontroller, need not monitor the timers, the serial communication or the external pins P3.2 and P3.3
- Whenever an event related to these units occur, it is informed to the microcontroller with the help of interrupts

8051 PROGRAMMING IN C –Interrupts

- A single microcontroller can serve several devices by two ways:
- Polling
  - The microcontroller continuously monitors the status of all the devices.
  - Whenever any device needs the service, it provides the service and moves on to the next device until everyone is serviced.
  - This will be done in an infinite loop.
- Interrupts
  - Whenever any device needs its service, the device notifies the microcontroller by sending it an interrupt signal
  - Upon receiving an interrupt signal, the microcontroller interrupts whatever it is doing and serves the device.
  - The program which is associated with the interrupt is called the interrupt service routine (ISR) or interrupt handler