Timers

Lecture 4

Bilal Habib

DCSE, UET

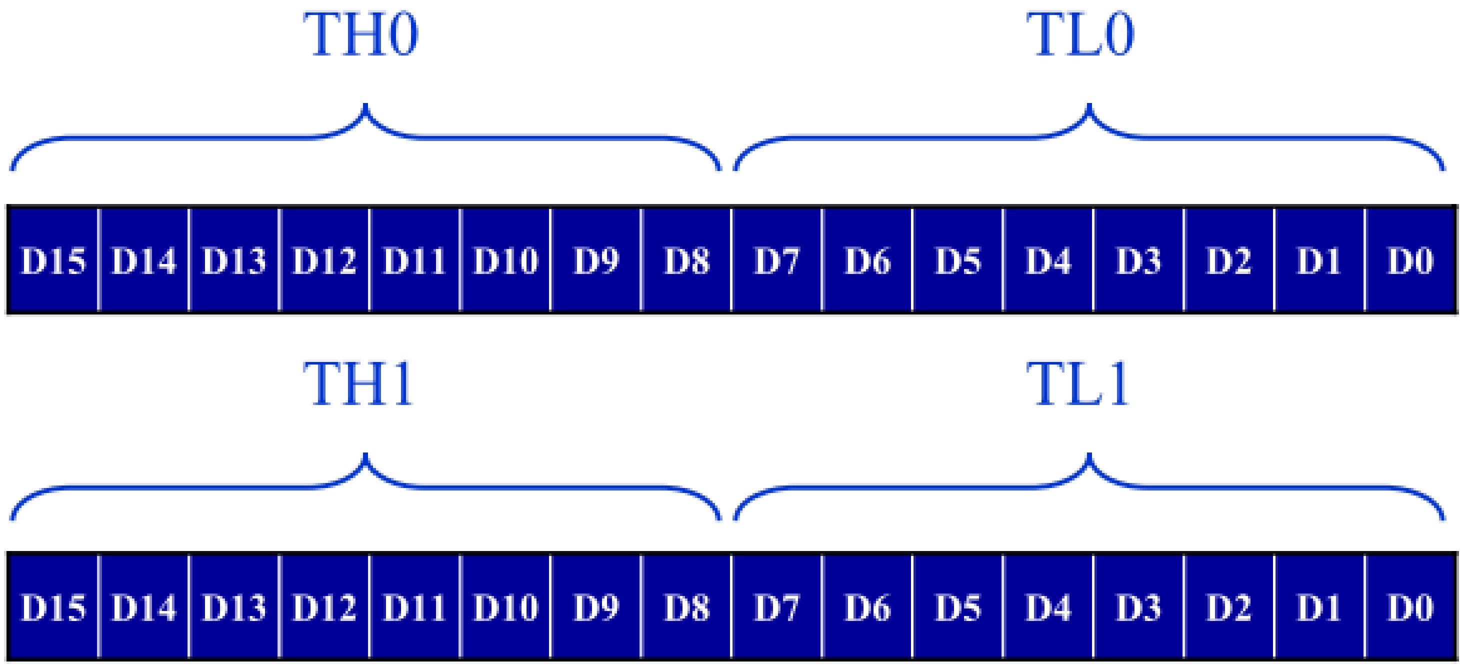
PROGRAMMING TIMERS

- The 8051 has two timers/counters, they can be used either as
  - Timers to generate a time delay or as
  - Event counters to count events happening outside the microcontroller
- Both Timer 0 and Timer 1 are 16 bits wide
  - Since 8051 has an 8-bit architecture, each 16-bits timer is accessed as two separate registers of low byte and high byte

PROGRAMMING TIMERS

Timer 0 & 1 Registers

- Accessed as low byte and high byte
  - The low byte register is called TL0/TL1 and
  - The high byte register is called TH0/TH1
  - Accessed like any other register
    - `MOV TL0,#4FH`
    - `MOV R5,TH0`



TH0
TL0
D15 D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0
TH1
TL1
D15 D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0

PIN DESCRIPTION

Port 3

- Port 3 can be used as input or output
  - Port 3 does not need any pull-up resistors
- Port 3 has the additional function of providing some extremely important signals

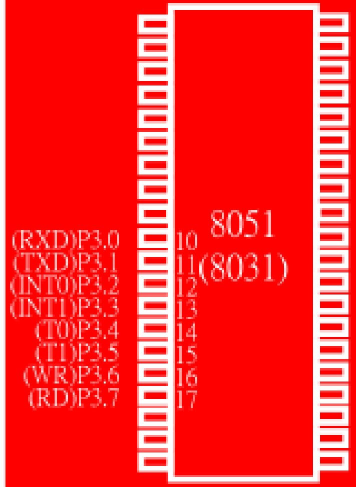
P3 Bit	Function	Pin
P3.0	RxD	10
P3.1	TxD	11
P3.2	$\overline{\text{INT0}}$	12
P3.3	$\overline{\text{INT1}}$	13
P3.4	T0	14
P3.5	T1	15
P3.6	$\overline{\text{WR}}$	16
P3.7	$\overline{\text{RD}}$	17

Serial communications

External interrupts

Timers

Read/Write signals of external memories



PROGRAMMING TIMERS

TMOD Register

- Both timers 0 and 1 use the same register, called TMOD (timer mode), to set the various timer operation modes
- TMOD is a 8-bit register
  - The lower 4 bits are for Timer 0
  - The upper 4 bits are for Timer 1
  - In each case,
    - The lower 2 bits are used to set the timer mode
    - The upper 2 bits to specify the operation

(MSB)							(LSB)
GATE	C/T	M1	M0	GATE	C/T	M1	M0
Timer1				Timer0			

PROGRAMMING TIMERS

TMOD Register (cont’)

(MSB) (LSB)

GATE	C/T	M1	M0	GATE	C/T	M1	M0
Timer1				Timer0			

M1	M0	Mode	Operating Mode
0	0	0	**13-bit timer mode** 8-bit timer/counter THx with TLx as 5-bit prescaler
0	1	1	**16-bit timer mode** 16-bit timer/counter THx and TLx are cascaded; there is no prescaler
1	0	2	**8-bit auto reload** 8-bit auto reload timer/counter; THx holds a value which is to be reloaded TLx each time it overfolws
1	1	3	**Split timer mode**

Gating control when set. Timer/counter is enable only while the INTx pin is high and the TRx control pin is set
When cleared, the timer is enabled whenever the TRx control bit is set

Timer or counter selected
Cleared for timer operation (input from internal system clock)
Set for counter operation (input from Tx input pin)

COUNTER PROGRAMMING

C/T Bit in TMOD Register

- The C/T bit in the TMOD registers decides the source of the clock for the timer
  - When C/T = 1, the timer is used as a counter and gets its pulses from outside the 8051
    - The counter counts up as pulses are fed from pins 14 and 15, these pins are called T0 (timer 0 input) and T1 (timer 1 input)

Port 3 pins used for Timers 0 and 1

Pin	Port Pin	Function	Description
14	P3.4	T0	Timer/counter 0 external input
15	P3.5	T1	Timer/counter 1 external input

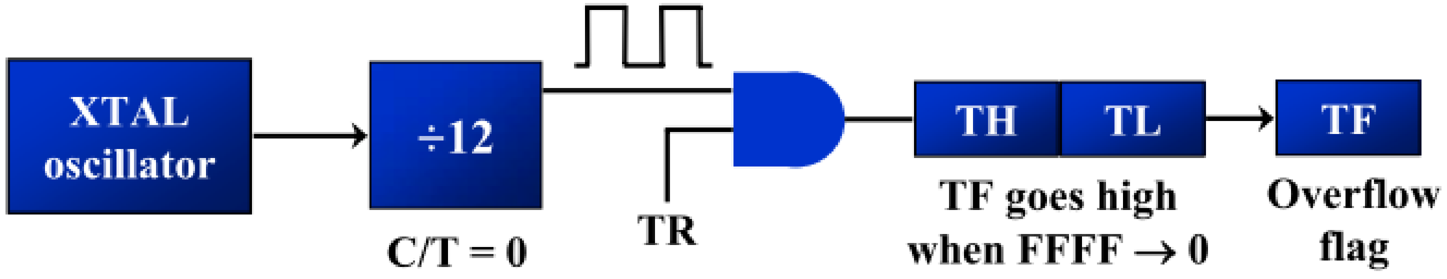
COUNTER PROGRAMMING

- Timers can also be used as counters counting events happening outside the 8051
  - When it is used as a counter, it is a pulse outside of the 8051 that increments the TH, TL registers
  - TMOD and TH, TL registers are the same as for the timer discussed previously
- Programming the timer in the last section also applies to programming it as a counter
  - Except the source of the frequency

PROGRAMMING TIMERS

Mode 1 Programming

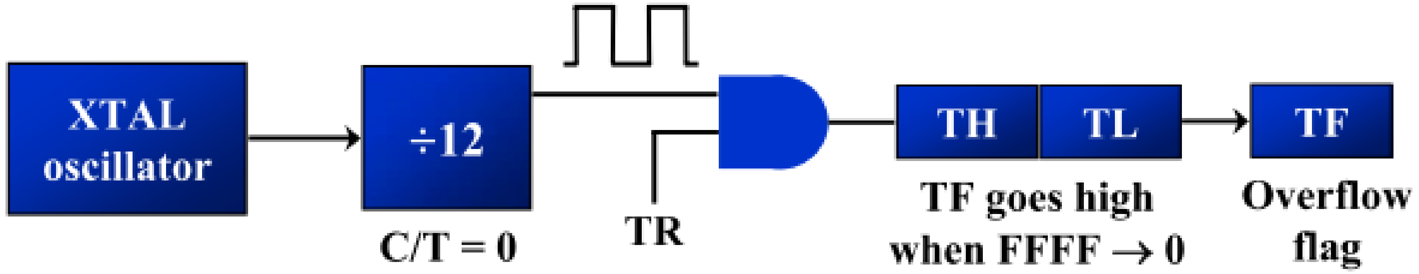
- The following are the characteristics and operations of mode1:
  1. It is a 16-bit timer; therefore, it allows value of 0000 to FFFFH to be loaded into the timer's register TL and TH
  2. After TH and TL are loaded with a 16-bit initial value, the timer must be started
     - This is done by `SETB TR0` for timer 0 and `SETB TR1` for timer 1
  3. After the timer is started, it starts to count up
     - It counts up until it reaches its limit of FFFFH



PROGRAMMING TIMERS

Mode 1 Programming (cont')

3. (cont')
   - When it rolls over from FFFFH to 0000, it sets high a flag bit called TF (timer flag)
     - Each timer has its own timer flag: TF0 for timer 0, and TF1 for timer 1
     - This timer flag can be monitored
   - When this timer flag is raised, one option would be to stop the timer with the instructions `CLR TR0` or `CLR TR1`, for timer 0 and timer 1, respectively
4. After the timer reaches its limit and rolls over, in order to repeat the process
   - TH and TL must be reloaded with the original value, and
   - TF must be reloaded to 0



PROGRAMMING TIMERS

Mode 1 Programming

Steps to Mode 1 Program

- To generate a time delay
  1. Load the TMOD value register indicating which timer (timer 0 or timer 1) is to be used and which timer mode (0 or 1) is selected
  2. Load registers TL and TH with initial count value
  3. Start the timer
  4. Keep monitoring the timer flag (TF) with the `JNB TFx,target` instruction to see if it is raised
     - Get out of the loop when TF becomes high
  5. Stop the timer
  6. Clear the TF flag for the next round
  7. Go back to Step 2 to load TH and TL again

Example 9-4

In the following program, we create a square wave of 50% duty cycle (with equal portions high and low) on the P1.5 bit. Timer 0 is used to generate the time delay. Analyze the program

```
        MOV   TMOD,#01    ;Timer 0, mode 1(16-bit mode)
HERE:   MOV   TL0,#0F2H   ;TL0=F2H, the low byte
        MOV   TH0,#0FFH   ;TH0=FFH, the high byte
        CPL   P1.5        ;toggle P1.5
        ACALL DELAY
        SJMP  HERE
```

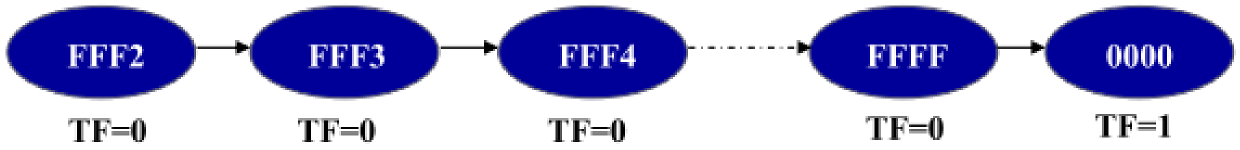
In the above program notice the following step.

1. TMOD is loaded.
2. FFF2H is loaded into TH0-TL0.
3. P1.5 is toggled for the high and low portions of the pulse.

Example 9-4 (cont')

```
DELAY:
        SETB TR0          ;start the timer 0
AGAIN:  JNB  TF0,AGAIN    ;monitor timer flag 0
                          ;until it rolls over
        CLR  TR0          ;stop timer 0
        CLR  TF0          ;clear timer 0 flag
        RET
```

4. The DELAY subroutine using the timer is called.
5. In the DELAY subroutine, timer 0 is started by the `SETB TR0` instruction.
6. Timer 0 counts up with the passing of each clock, which is provided by the crystal oscillator. As the timer counts up, it goes through the states of FFF3, FFF4, FFF5, FFF6, FFF7, FFF8, FFF9, FFFA, FFFB, and so on until it reaches FFFFH. One more clock rolls it to 0, raising the timer flag (TF0=1). At that point, the JNB instruction falls through.

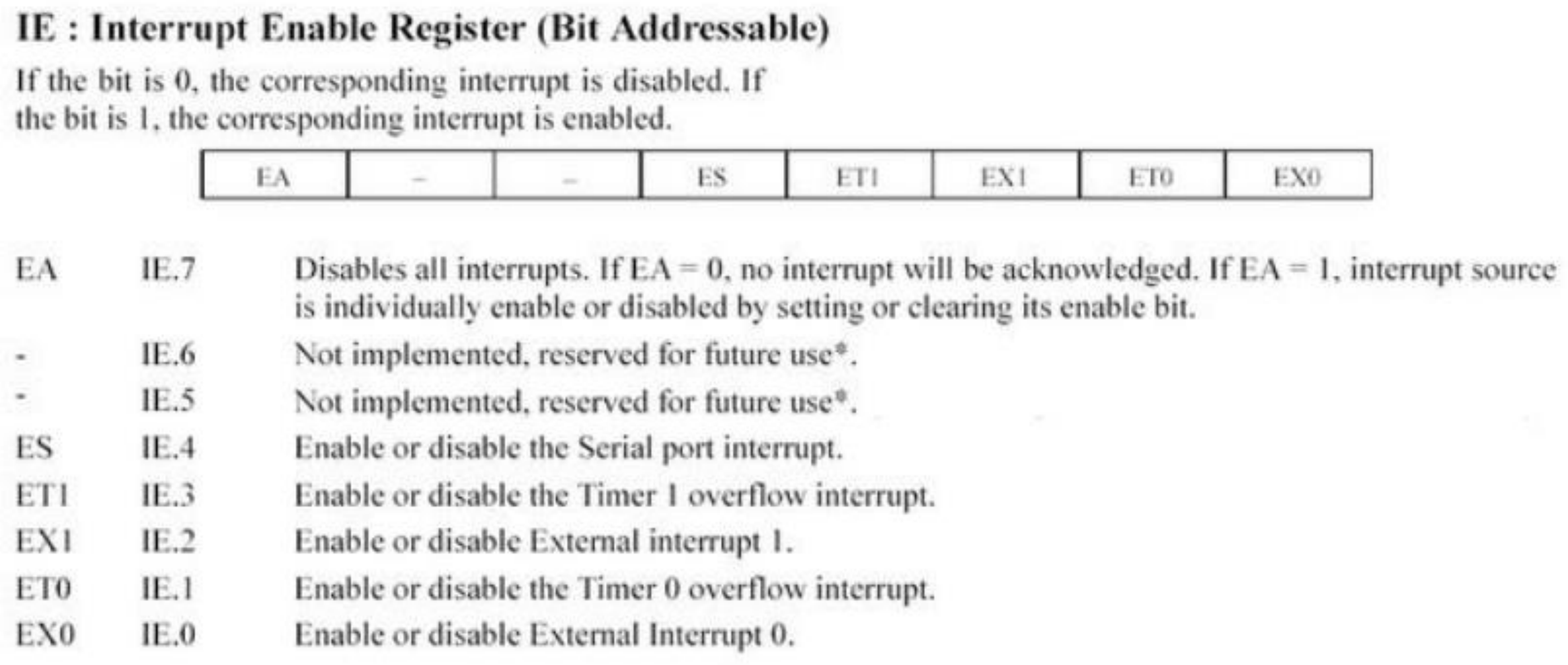


7. Timer 0 is stopped by the instruction `CLR TR0`. The DELAY subroutine ends, and the process is repeated.

Notice that to repeat the process, we must reload the TL and TH registers, and start the process is repeated

…

IE: Interrupt Enable



IE : Interrupt Enable Register (Bit Addressable)

If the bit is 0, the corresponding interrupt is disabled. If the bit is 1, the corresponding interrupt is enabled.

EA	–	–	ES	ET1	EX1	ET0	EX0

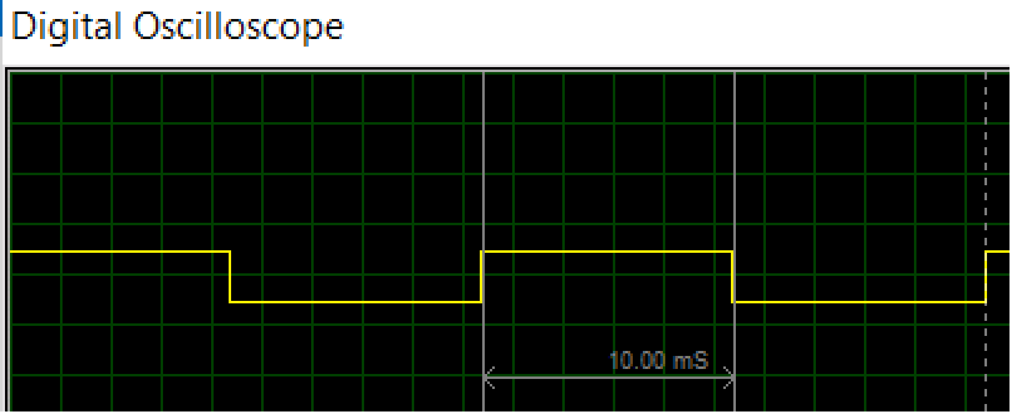
EA	IE.7	Disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, interrupt source is individually enable or disabled by setting or clearing its enable bit.
-	IE.6	Not implemented, reserved for future use*.
-	IE.5	Not implemented, reserved for future use*.
ES	IE.4	Enable or disable the Serial port interrupt.
ET1	IE.3	Enable or disable the Timer 1 overflow interrupt.
EX1	IE.2	Enable or disable External interrupt 1.
ET0	IE.1	Enable or disable the Timer 0 overflow interrupt.
EX0	IE.0	Enable or disable External Interrupt 0.

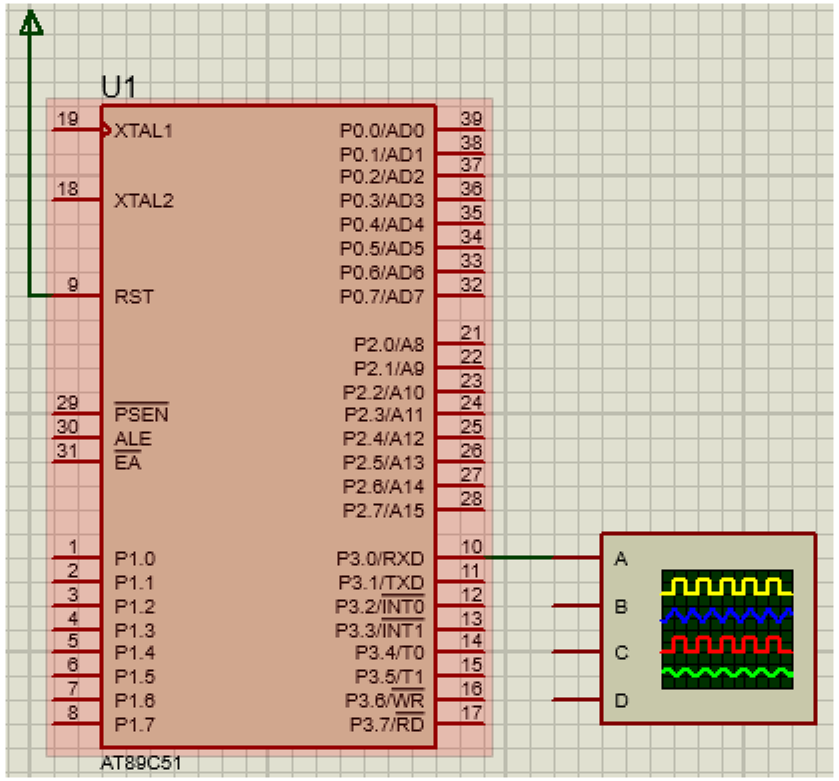
IE = 1000 1000 = 0x88

IE = 1000 0010 = 0x82 timer 0

IE = 1000 1000 = 0x88 timer 1

Timer Example 1: 10msec delay

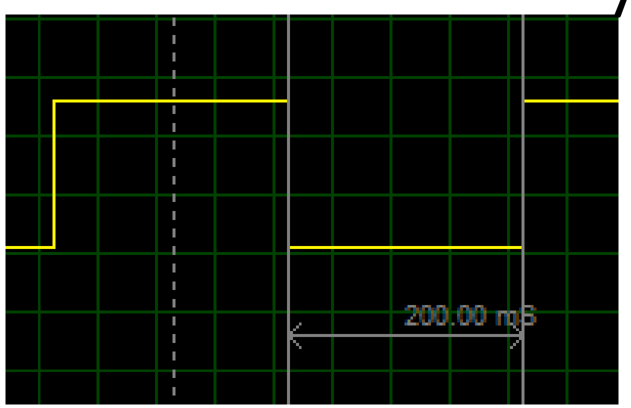


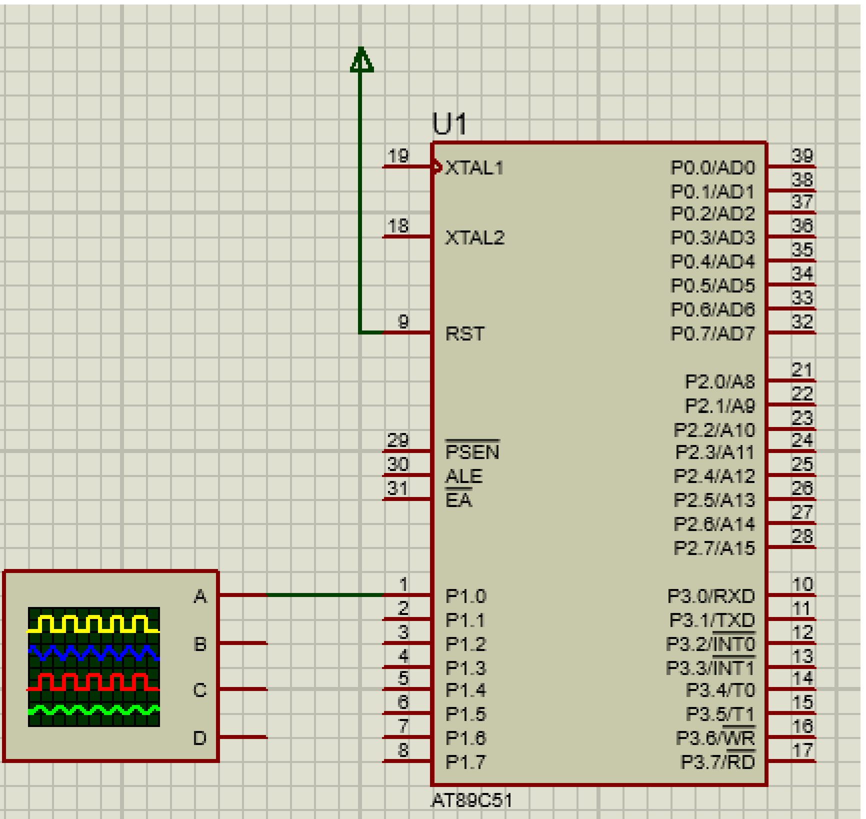


```
#include <reg51.h>
#include <stdio.h>
sbit pin = P3^0;
void Start_timer0(void)
{
    TR0 = 1;
}

void timer0(void) interrupt 1
{                                        10,000us
    TH0 = 0xD8;     // 10 msec delay
    TL0 = 0xEF;
}
void Init_timer0(void)
{
    TMOD = 0x01;    //16 bit mode of Timer 0.
    TH0 = 0xD8;     // 10 msec delay
    TL0 = 0xEF;
    IE = 0x82;  //Enable Timer0 and Global Interrupt
}
void main(void)
{
    Init_timer0();  //10msec delay
    Start_timer0();
    while(1)
    {
    while(TF0==0);
    pin = ~pin;
    }
}
```

Timer example 2: 200 msec delay





Using software variable

```c
#include <reg51.h>
#include <stdio.h>
sbit pin = P1^0;
int x = 0;
void Start_timer(void)
{
    TR1 = 1;
}

void timer1(void) interrupt 3   //ISR: Interrupt Service Routin
{
    x++;
    if(x==4){         //50msec x 4 = 200ms
        pin = ~pin;
        x = 0;}
    TH1 = 0x3C;       // 50 msec delay
    TL1 = 0xAF;
}
void Init_timer(void)
{
    TMOD = 0x10;      //16 bit mode of Timer 1.
    TH1 = 0x3C;       // 50 msec delay
    TL1 = 0xAF;
    IE = 0x88;    //Enable Timer1 and Global Interrupt
}

void main(void)
{
    Init_timer();   //50msec delay
    Start_timer();
    while(1)
    {}
}
```

50,000us

15,535