

Microprocessors and Microcontrollers (EE-231)

Lab-7

Main Objectives

- Understanding the hex file
- Timers Programming in 8051
 - Generating a square wave using timer mode 1 and mode 2

Intel Hex File

- Intel hex file is a widely used file format
- Designed to standardize the loading of executable machine codes into a ROM chip
- Loaders software that come with every ROM burner (programmer) support the Intel hex file format
- In Windows-based assemblers the Intel hex file is produced automatically
- In DOS-based PC a utility called OH (object-to-hex) was needed to produce it

Intel Hex File

- The hex file provides the following:
 1. The number of bytes of information to be loaded
 2. The starting address where the information must be placed
 3. The information itself

```
:1000000075805575905575A0557DFA111C7580AA9F
:100010007590AA75A0AA7DFA111C80E47C237B4F01
:07002000DBFEDCFADDF62235
:00000001FF

:CC AAAA TT DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD SS
:10 0000 00 75805575905575A0557DFA111C7580AA 9F
:10 0010 00 7590AA75A0AA7DFA111C80E47C237B4F 01
:07 0020 00 DBFEDCFADDF622 35
:00 0000 01 FF
```

Intel Hex File

Each line starts with a colon

Count byte – how many bytes,
00 to 16, are in the line

16-bit address – The loader
places the first byte of data
into this memory address

Type –
00, there are more
lines to come after
this line
01, this is the last
line and the
loading should
stop after this line

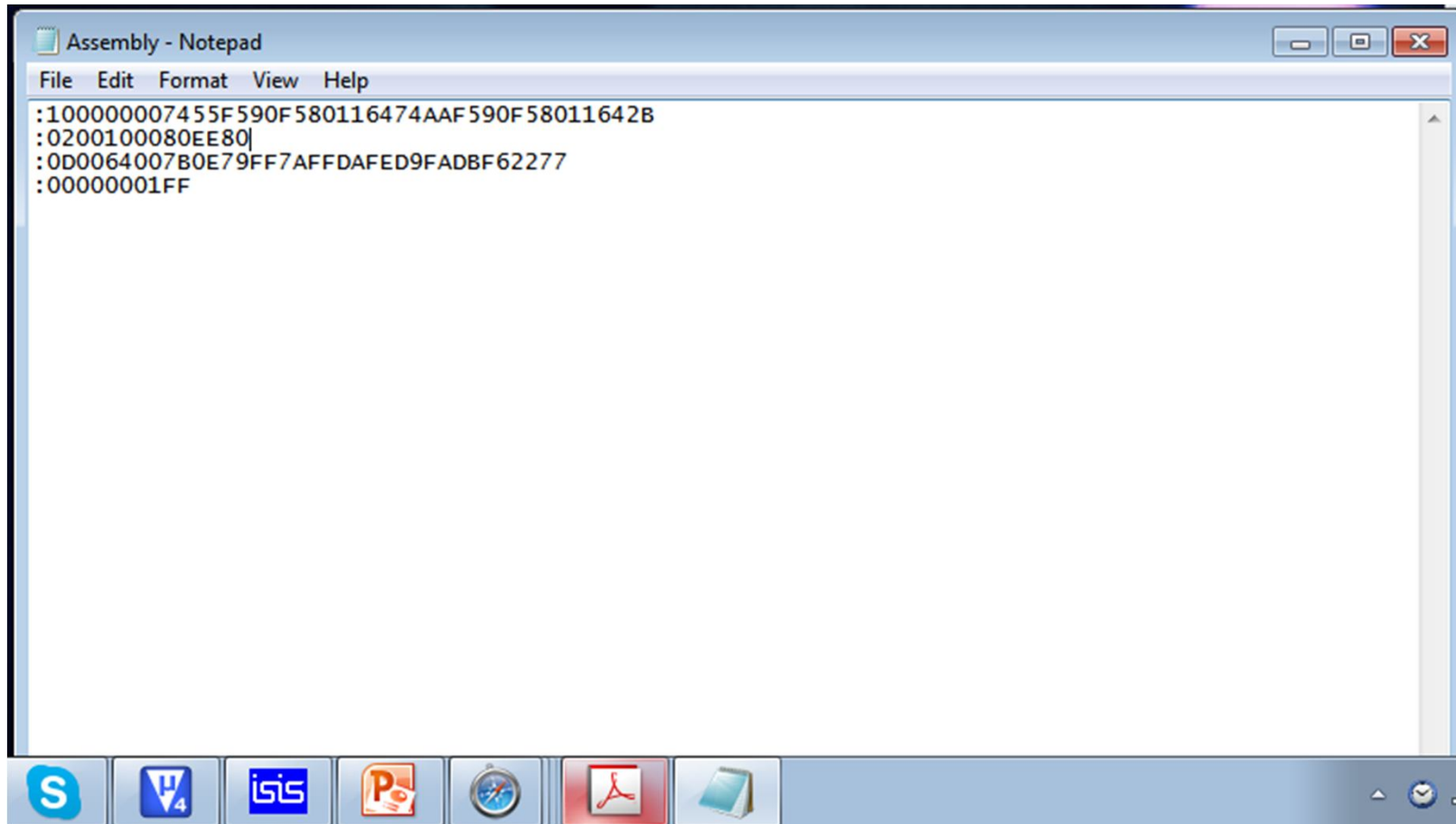
```
:CC AAAA TT DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD SS
:10 0000 00 75805575905575A0557DFA111C7580AA 9F
:10 0010 00 7590AA75A0AA7DFA111C80E47C237B4F 01
:07 0020 00 DBFEDCFADDE622 35
:00 0000 01 FF
```

Real information (data or code) – There is a maximum
of 16 bytes in this part. The loader places this
information into successive memory locations of ROM

Single byte – this last byte is the checksum
byte of everything in that line

Intel Hex File

- Verify the checksum of 2nd Line

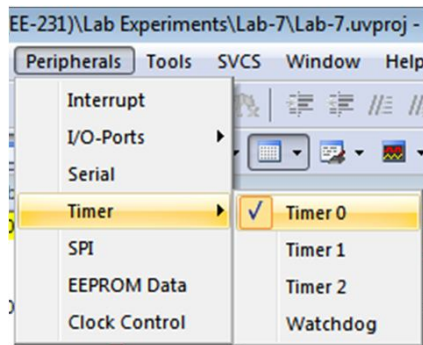


```
Assembly - Notepad
File Edit Format View Help
:100000007455F590F580116474AAF590F58011642B
:0200100080EE80
:0D0064007B0E79FF7AFFDAFED9FADBF62277
:00000001FF
```

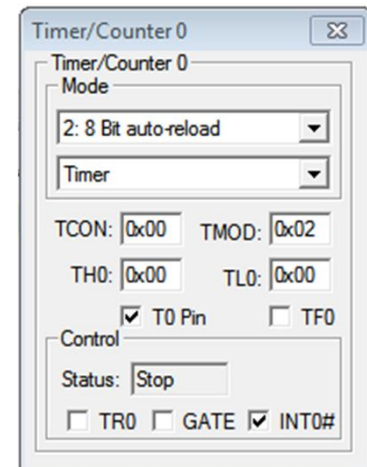
Viewing Timer Simulation

To view the timer simulation in the debug window

- Choose Timer from the peripherals and select whichever timer you are using.

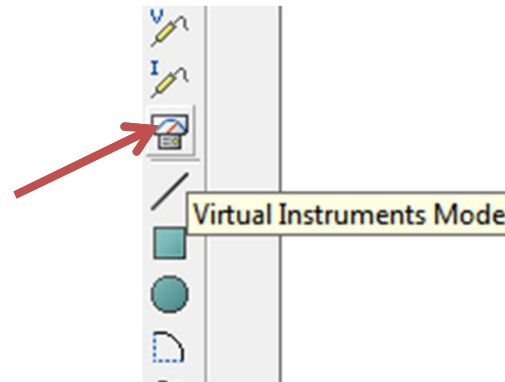


- You will see following window.
- Select the mode and chose the option of Timer or Counter
- You can either observe the values of the pins TRX and TRF and Status of the register or you can manually change their values and status

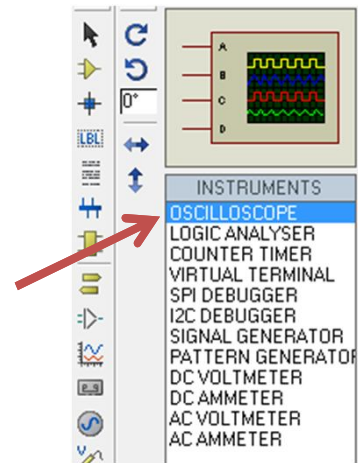


Viewing Oscilloscope in ISIS

- ❑ To find the oscilloscope, select **virtual instruments mode** from the toolbar.



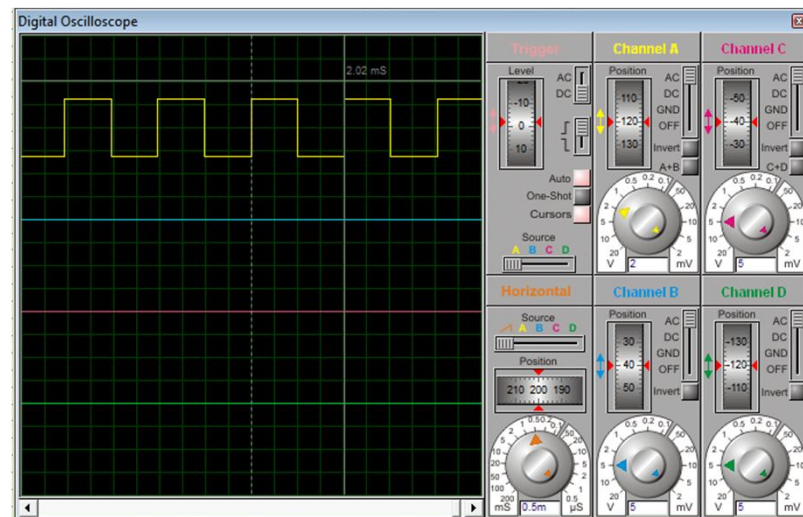
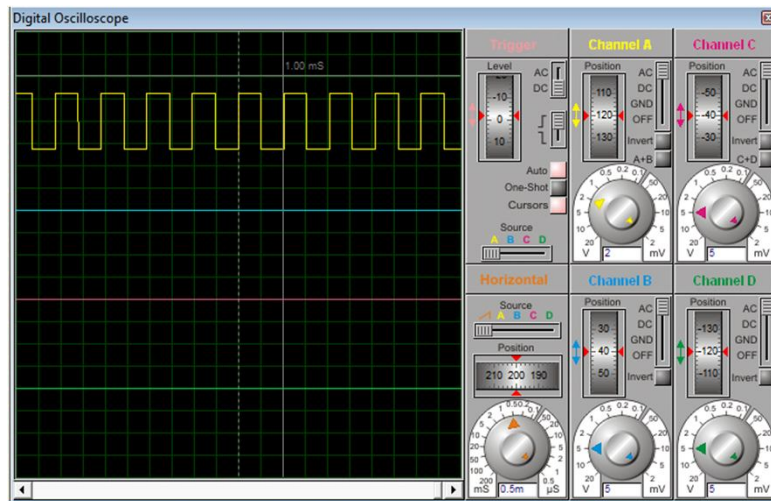
- ❑ Then select **oscilloscope** from the list.



- ❑ Oscilloscope has four channels(i.e. A,B,C,D) and does not need the signal ground.

Today's Task 1

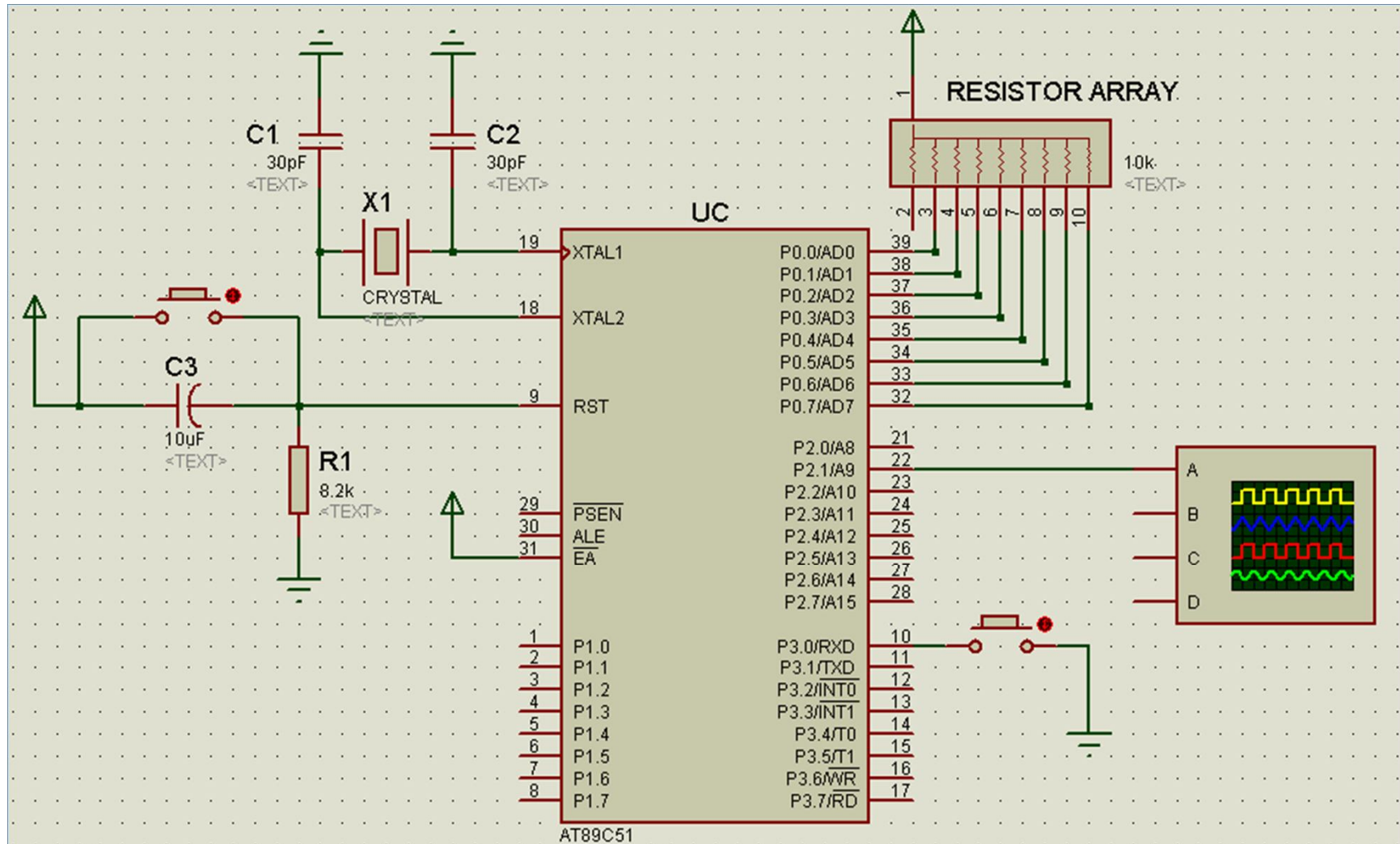
- Generate a **square wave** of 1 KHz and 500Hz on pin P2.1 depending upon the status of pin 3.0 i.e.,
- If Pin 3.0 is '1' then P2.1 has 1KHz
- If Pin 3.0 is '0' then P2.1 has 500Hz
- Use Timer in mode 1
- Verify the design in Proteus ISIS by measuring the frequency using oscilloscope



Task Code

```
1  ORG 0
2  SETB P3.0 ; Make it input
3  MOV TMOD,#01H ; Mode 1 for timer 0
4
5  AGAIN:
6  JNB P3.0,Option2
7  ;-----1 KHz-----
8  MOV TL0,#034H
9  MOV TH0,#0FEH
10 SJMP GO
11
12 Option2:
13 ;-----500Hz-----
14 MOV TL0,#067H
15 MOV TH0,#0FCH
16
17 GO:
18 SETB TR0
19 HERE: JNB TF0,HERE
20 CLR TR0
21 CPL P2.1
22 CLR TF0
23 SJMP AGAIN
24 END
```

Proteus Diagram



Today's Task 2

- Design a security system that monitors the intrusion in such a way that when an intruder is detected, an alarm is raised for **5 secs** after that a signal is issued to seal the doors.
- To implement it on easy 8051 board, monitor the status of **push button P3.0** until it is pressed ($P3.0 = 0$), after that start the timer to generate a **1KHz square wave** for a buzzer connected to pin **P2.3**. After 5 secs have passed, turn on LED connected to Pin **P3.7** by sending 0 to it.



Task Code

```
1  ORG 0
2  SETB P3.0 ; Make it input
3  MOV TMOD,#01H ; Mode 1 for timer 0
4
5  WAIT_HERE:JB P3.0,WAIT_HERE
6
7  MOV R1,#100
8  OUTER_LOOP:
9  MOV R0,#100
10 INNER_LOOP:
11 ;-----1 KHz-----
12 MOV TLO,#034H
13 MOV TH0,#0FEH
14 SETB TR0
15 HERE:JNB TFO,HERE
16 CLR TR0
17 CPL P2.3
18 CLR TFO
19 DJNZ R0,INNER_LOOP
20 DJNZ R1,OUTER_LOOP
21
22 CLR P3.7
23
24 END
```

Next Week Assignment

- Use Timer 0 as **counter** to count the external events from 0-FF and show the status of count (i.e. value of TL0) on **two seven segments**. When the count is completed, i.e. value is overflowed from FF to 00, then turn on an **LED** to indicate the completion of the count.
- Implement the design on **breadboard**.
- Use Push Button as input of external events (to be connected to T0 pin i.e. P3.4 pin)
- See the schematic on next page.

Next Week Assignment

