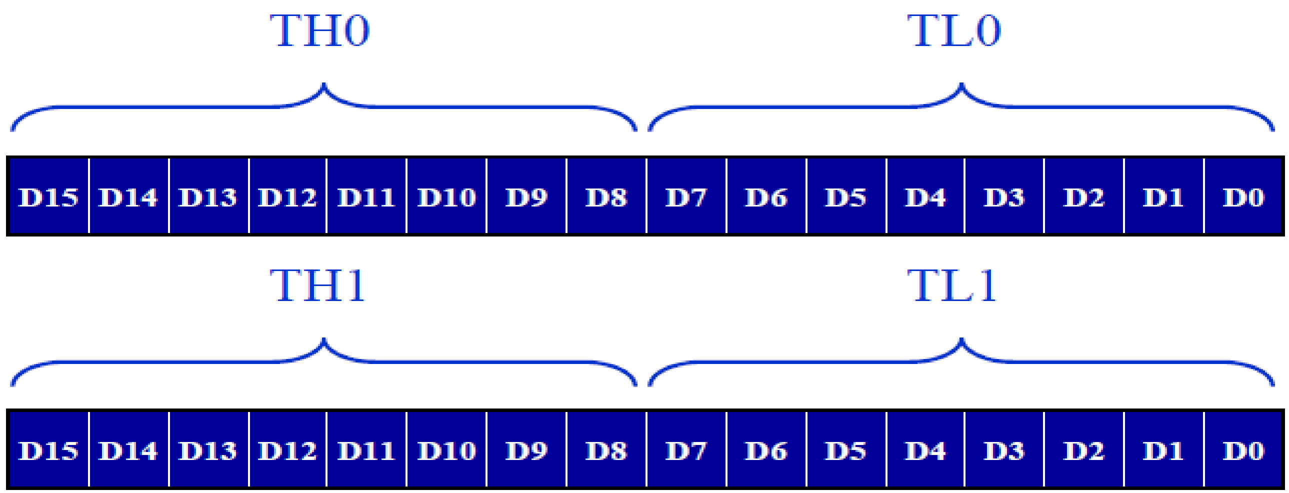
Module 4

- Timers
- Serial Port

- 8051 microcontroller have two timers timer 0 and timer 1.
- This are 16 bit timers, but 8051 is a 8 bit controller so they are referred as lower byte and higher byte shown below
  TL0, TH0, Tl1,TH1



- We can provide the required delay by loading the number machine cycles into those registers.

- Normally delay count value will be decremented in software delay program, but count value will be incremented in timers . i.e means delay count need to subtracted from max count value then timer need to activate.

- Time take for a machine cycle with different clock frequency are given below

Find the timer's clock frequency and its ~~period for various~~ 8051-based system, with the crystal frequency 11.0592 MHz when C/T bit of TMOD is 0.

Solution:

1/12 × 11.0529 MHz = 921.6 MHz;

T = 1/921.6 kHz = 1.085 us

- Do the same thing for clock frequencies for 12MHZ and 16MHZ.

1/12X12M=1MHz= and time =1 us

1/12X16M= 1.333MHz and time=.75 us

- Delay calculation and timer register feeding is shown below.

Assume that XTAL = 11.0592 MHz. What value do we need to load the timer's register if we want to have a time delay of 5 ms (milliseconds)? Show the program for timer 0 to create a pulse width of 5 ms on P2.3.

Solution:

Since XTAL = 11.0592 MHz, the counter counts up every 1.085 us. This means that out of many 1.085 us intervals we must make a 5 ms pulse. To get that, we divide one by the other. We need 5 ms / 1.085 us = 4608 clocks. To Achieve that we need to load into TL and TH the value 65536 – 4608 = EE00H. Therefore, we have TH = EE and TL = 00.

- Both the timers can work as a timers to provide particular delay and as counter to count the external activity.
- That mode can be set with the help of TMOD register which is shown below

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

- TMOD register also 8 bit register only. Lower nibble in TMOD is used to set timer 0's mode and higher nibble is used to set the mode of timer 1.
- Lower and higher nibbles bits functions are same only.

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**

C/T (Counter/ Timer) Bit:

This bit in the TMOD register is used to decide whether the timer is used As a delay generator or an event counter.

- If C/T =0, Then timer is working as delay generator
- If C/T=1, Then that timer is working as event counter

Gate Bit:

- Timer control will be decided Through this bit value
- Gate-=0 : indicates that timer can be controlled through program by TR and TF bit's in TCON register.
- Bit TR is used to start the timer by making TR=1 and stop the timer by making TR=0
- TF bit indicates the timer overflow from FFFFH to 0000h by making TF=1. then timer can stop.
- Gate=1: indicates that timer can control by external signals

- Timers setting with TMOD is explained here with an example.

Indicate which mode and which timer are selected for each of the following.
(a) MOV TMOD, #01H (b) MOV TMOD, #20H (c) MOV TMOD, #12H

Solution:

We convert the value from hex to binary. From Figure 9-3 we have:
(a) TMOD = 00000001, mode 1 of timer 0 is selected.
(b) TMOD = 00100000, mode 2 of timer 1 is selected.
(c) TMOD = 00010010, mode 2 of timer 0, and mode 1 of timer 1 are selected.

- Iam going to discuss about mode1 and mode2 of timer,bcz of the acadamaic purpose.

Generating time delay by using mode1 procedure is explained and example program Given below.

- 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.
- Load registers TL and TH with initial count value.
- Start the timer.
- 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
- Stop the timer.
- Clear the TF flag for the next round.
- Go back to Step 2 to load TH and TL again.

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
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
```

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.
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

Delay calculation:

The timer works with a clock frequency of 1/12 of the XTAL frequency; therefore, we have 11.0592 MHz / 12 = 921.6 kHz as the timer frequency. As a result, each clock has a period of T = 1/921.6kHz = 1.085us. In other words, Timer 0 counts up each 1.085 us resulting in delay = number of counts × 1.085us.

The number of counts for the roll over is FFFFH – FFF2H = 0DH (13 decimal). However, we add one to 13 because of the extra clock needed when it rolls over from FFFF to 0 and raise the TF flag. This gives $14 \times 1.085us = 15.19us$ for half the pulse. For the entire period it is $T = 2 \times 15.19us = 30.38us$ as the time delay generated by the timer.

calculate the frequency of the square wave generated on pin P1.5.

Solution:

In the timer delay calculation of Example , we did not include the overhead due to instruction in the loop. To get a more accurate timing, we need to add clock cycles due to this instructions in the loop. To do that, we use the machine cycle from Table A-1 in Appendix A, as shown below.

	Cycles
HERE: MOV TL0,#0F2H	2
MOV TH0,#0FFH	2
CPL P1.5	1
ACALL DELAY	2
SJMP HERE	2
DELAY:	
SETB TR0	1
AGAIN: JNB TF0,AGAIN	2*14
CLR TR0	1
CLR TF0	1
RET	2
Total 42	

T = 2 × 42 × 1.085 us = 91.94 us and F = 10.9KHz

Find the delay generated by timer 0 in the following code, using both of the Methods . Do not include the overhead due to instruction.

```
CLR P2.3 ;Clear P2.3
MOV TMOD,#01 ;Timer 0, 16-bitmode
HERE: MOV TL0,#3EH ;TL0=3Eh, the low byte
MOV TH0,#0B8H ;TH0=B8H, the high byte
SETB P2.3 ;
SETB TR0 ;Start the timer 0
AGAIN: JNB TF0,AGAIN ;Monitor timer flag 0
CLR TR0 ;Stop the timer 0
CLR TF0 ;Clear TF0 for next round
CLR P2.3
```

Solution:

(a) (FFFFH – B83E + 1) = 47C2H = 18370 in decimal and 18370 × 1.085 us = 19.93145 ms

(b) Since TH – TL = B83EH = 47166 (in decimal) we have 65536 – 47166 = 18370. This means that the timer counts from B38EH to FFFF. This plus Rolling over to 0 goes through a total of 18370 clock cycles, where each clock is 1.085 us in duration. Therefore, we have 18370 × 1.085 us = 19.93145 ms as the width of the pulse.

Assume that XTAL = 11.0592 MHz. What value do we need to load the timer's register if we want to have a time delay of 5 ms (milliseconds)? Show the program for timer 0 MODE 1 to create a pulse width of 5 ms on P2.3.

Solution:

Since XTAL = 11.0592 MHz, the counter counts up every 1.085 us. This means that out of many 1.085 us intervals we must make a 5 ms pulse. To get that, we divide one by the other. We need 5 ms / 1.085 us = 4608 clocks. To Achieve that we need to load into TL and TH the value 65536 – 4608 = EE00H. Therefore, we have TH = EE and TL = 00.

```
CLR P2.3 ;Clear P2.3
MOV TMOD,#01 ;Timer 0, 16-bitmode
HERE: MOV TL0,#0 ;TL0=0, the low byte
MOV TH0,#0EEH ;TH0=EE, the high byte
SETB P2.3 ;SET high P2.3
SETB TR0 ;Start timer 0
AGAIN: JNB TF0,AGAIN ;Monitor timer flag 0
CLR TR0 ;Stop the timer 0
CLR TF0 ;Clear timer 0 flag
```

- Assume that XTAL = 11.0592 MHz, write a program to generate a square wave of 2 kHz frequency on pin P1.5 using timer 1 mode 1.

Assume that XTAL = 11.0592 MHz, write a program to generate a square wave of 2 kHz frequency on pin P1.5.

Solution:

This is similar to Example 9-10, except that we must toggle the bit to generate the square wave. Look at the following steps.

(a) T = 1 / f = 1 / 2 kHz = 500 us the period of square wave.

(b) 1 / 2 of it for the high and low portion of the pulse is 250 us.

(c) 250 us / 1.085 us = 230 and 65536 – 230 = 65306 which in hex is FF1AH.

(d) TL = 1A and TH = FF, all in hex. The program is as follow.

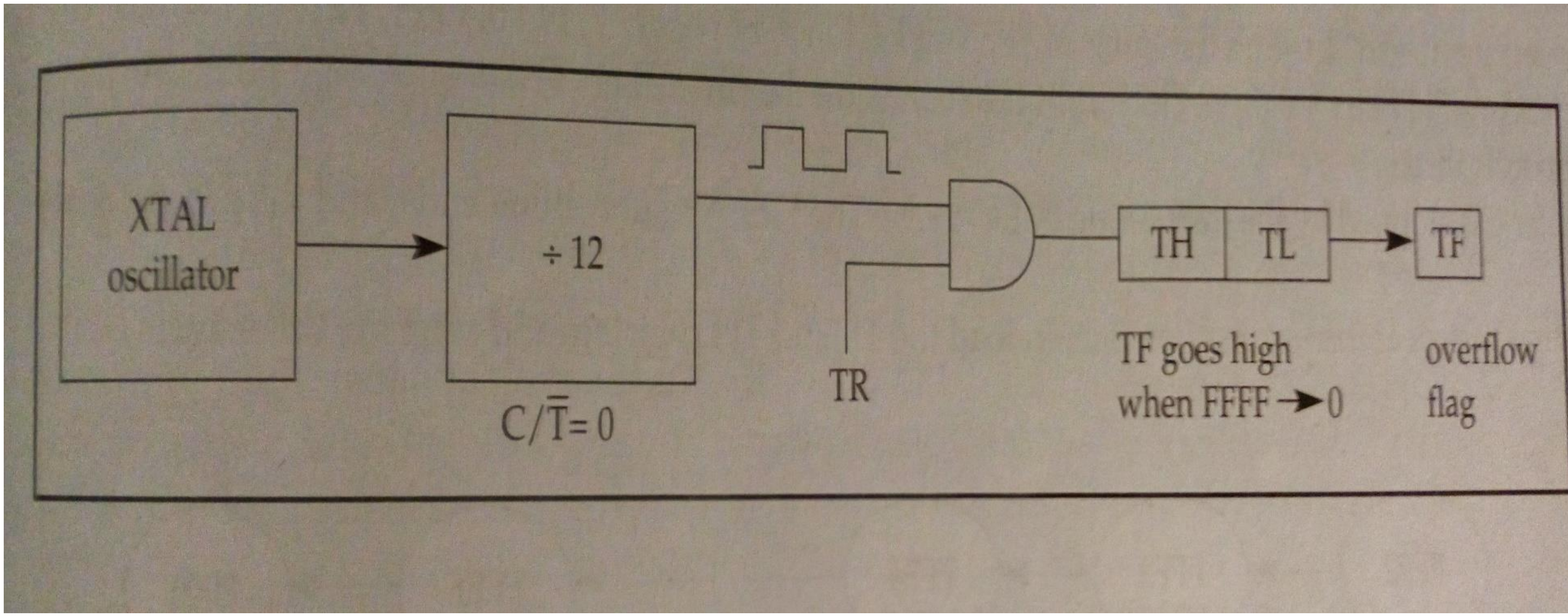
```
MOV TMOD,#10h ;Timer 1, 16-bitmode
AGAIN: MOV TL1,#1AH ;TL1=1A, low byte of timer
MOV TH1,#0FFH ;TH1=FF, the high byte
SETB TR1 ;Start timer 1
BACK: JNB TF1,BACK ;until timer rolls over
CLR TR1 ;Stop the timer 1
CPL P1.5 ;To create a sq wave
CLR TF1 ;Clear timer 1 flag
SJMP AGAIN ;Reload timer
```

What is count to be loaded to create maximum possible delay in mode 1.how much is the delay?

Count to be loaded is 0000h into timer registers.

Max delay possible is 71ms.

Block diagram for mode1



- Timers are programmed not only in assembly language, they can also programmed in C.

EX: 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;
}
```

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.

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;
}
```

Write an 8051 C program to toggle all bits of P2 continuously every 500ms.user timer1,mode1 to create delay.

A switch is connected to pin P1.2.Write an 8051 C program to monitor SW and create the following frequencies on pin P1.7:

SW=0:500Hz

SW=1:750Hz.Use timer 0,mode1.

For 500Hz-count value to be loaded-FC67h

For 750hz-count value to be loaded-FD9Ah

```
#include<reg51.h>
sbit mybit=P1^7;
sbit SW=P1^2;
void delay(unsigned char);
void main()
{
SW=1;
while(1)
    {
      mybit=~mybit;
      if(SW==0)
          delay(0);
      if(SW==1)
          delay(1);
      }
}
```

```
void delay(unsigned char c)
    {
        TMOD=0x01;
         if(c==0)
           {
             TL0 =0x67;
              TH0=0xFC;
           }
        else
            {
              TL0 =0x9A;
              TH0=0xFD;
             }
        TR0=1;
        while(TF==0);
        TR0=0;
         TF0=0;
}
```

Programming timer interrupts

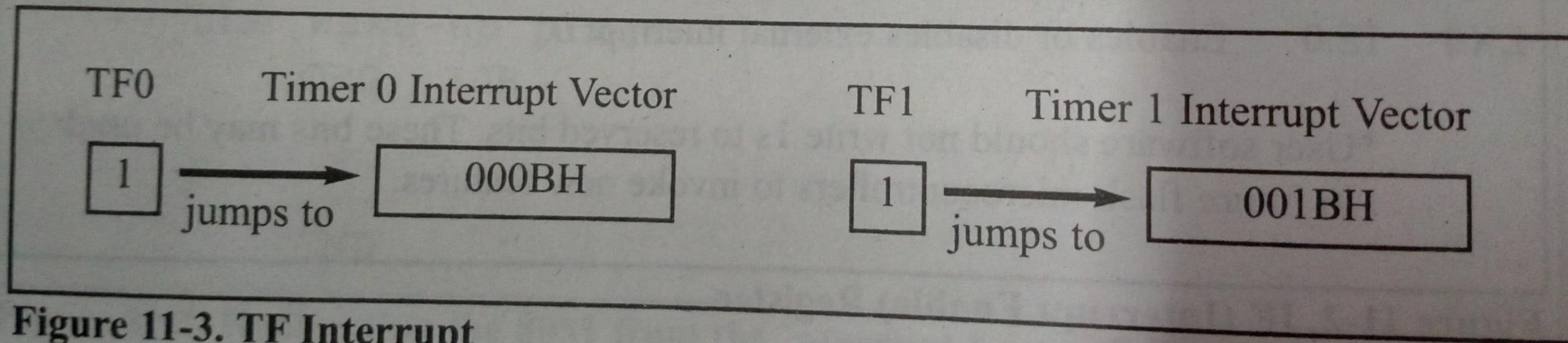


Figure 11-3. TF Interrupt

Write a program to generate a square wave of 50hz frequency on pin P1.2 in interrupt mode using timer 0 mode 1.Assume XTAL=11.0592MHz.

```
        ORG 00h
        LJMP Main
        ORG 000Bh
         CLR TR0
        CPL P1.2
         MOV  TL0,#00h
         MOV TH0,#0DCh
         SETB TR0
          RETI
          ORG 30H
MAIN:MOV TMOD,#01H
          MOV TL0,#00H
          MOV TH0,#0DCH
          MOV IE,#82H                 1 0 0 0 0 0 1 0B
          SETB TR0
          SJMP $
          END
```