8051 Timer Programming in Assembly and C

Objectives:

At the end of this chapter, we will be able to:

- List the timers of 8051 and their associated registers
- Describe the various modes of the 8051 timers.
- Program the 8051 timers in assembly and C and
- Program the 8051 counters in assembly and C.

Programming 8051 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.

Basic Timers of 8051:

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. The low byte register is called TL0/TL1 and the high byte register is called TH0/TH1. These registers can be accessed like any other register. For example:

- MOV TL0,#4FH
- MOV R5,TH0

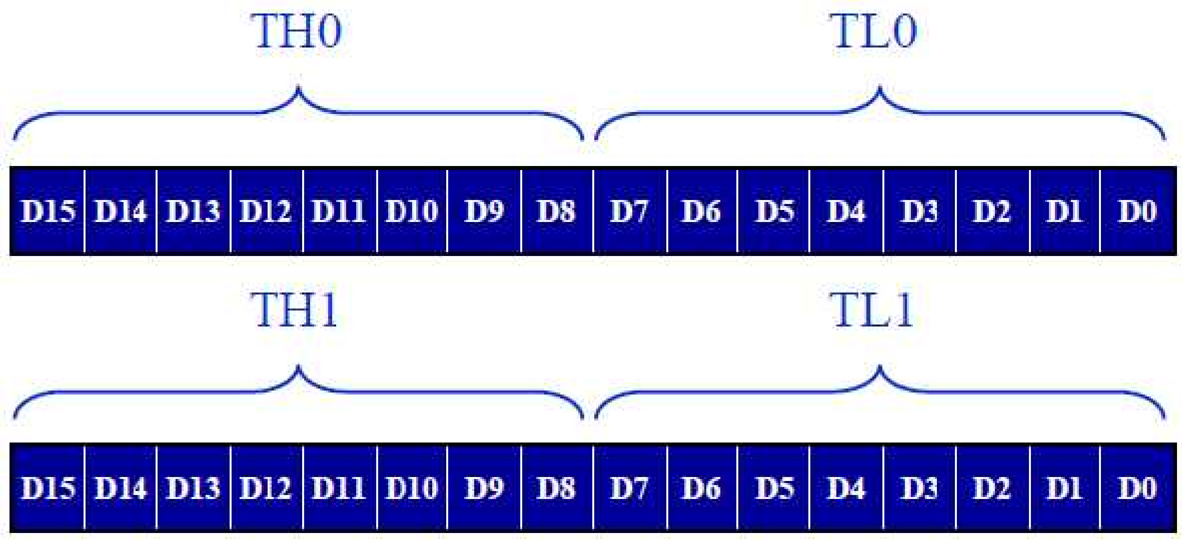
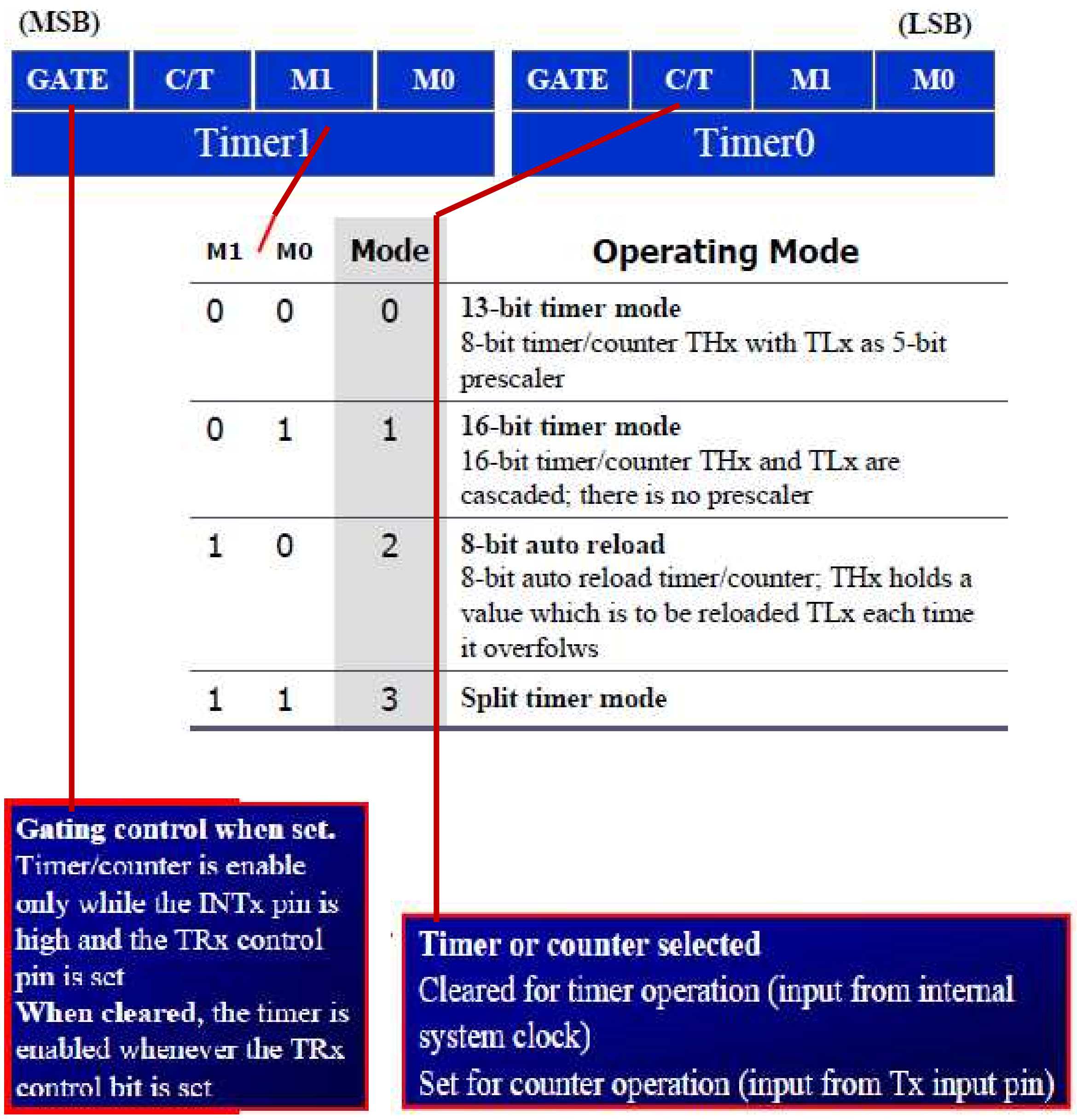


Figure 1: Timer 0 and Timer1 register

TMOD (timer mode) Register:

Both timers 0 and 1 use the same register, called TMOD (timer mode), to set the various timer operation modes. TMOD is an 8-bit register. The lower 4 bits are for Timer 0 and 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. The TMOD register is as shown in figure 2 below:



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

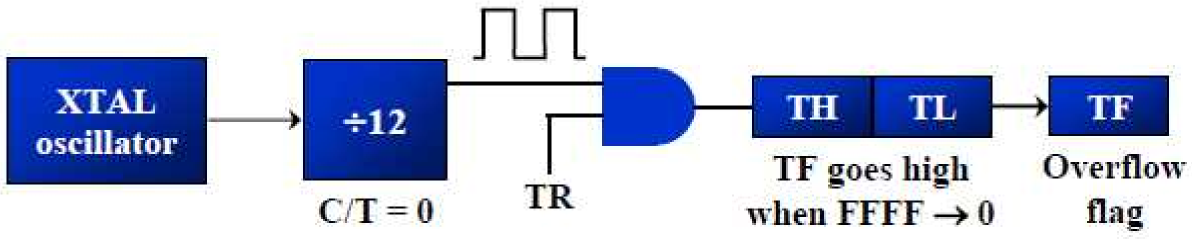
Timer or counter selected
Cleared for timer operation (input from internal system clock)
Set for counter operation (input from Tx input pin)

Timers of 8051 do starting and stopping by either software or hardware control. In using software to start and stop the timer where GATE=0. The start and stop of the timer are controlled by way of software by the TR (timer start) bits TR0 and TR1. The SETB instruction starts it, and it is stopped by the CLR instruction. These instructions start and stop the timers as long as GATE=0 in the TMOD register. The hardware way of starting and stopping the timer by an external source is achieved by making GATE=1 in the TMOD register.

Mode 1 Programming:

The following are the characteristics and operations of mode 1:
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. 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.



Steps to program in mode 1:

To generate a time delay, using timer in mode 1, following are the steps:
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 1
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. Also calculate the delay generated. Assume XTAL=11.0592MHz.

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
```

(a)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

Example 2: Find the delay generated by timer 0 in the following code, using hex as well as decimal method. Do not include the overhead due to instruction.

Program:

```
        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             ;SET high timer 0
        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 1.085μs in duration. Therefore, we have 18370 × 1.085 us = 19.93145 ms as the width of the pulse.

Finding values to be loaded into the timer:

To calculate the values to be loaded into the TL and TH registers, look at the following steps. Assume XTAL = 11.0592 MHz, we can use the following steps for finding the TH, TL registers' values:

1. Divide the desired time delay by 1.085µs.
2. Perform 65536 – n, where n is the decimal value we got in Step1.
3. Convert the result of Step2 to hex, where yyxx is the initial hex value to be loaded into the timer's register and
4. Set TL = xx and TH = yy.

Example 3: 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 ? 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 / 1.085µs = 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.

Program:

```
        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
```

Example 4: 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,#01            ;Timer 0, 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
        CLR P1.5                ;Clear timer flag 1
        CLR TF1                 ;Clear timer 1 flag
        SJMP AGAIN              ;Reload timer
```

Example 5: Assume XTAL = 11.0592 MHz, write a program to generate a square wave of 50 kHz frequency on pin P2.3.

Solution:

Look at the following steps.

(a) T = 1 / 50 = 20 ms, the period of square wave.

(b) 1 / 2 of it for the high and low portion of the pulse is 10 ms.

(c) 10 ms / 1.085 us = 9216 and 65536 – 9216 = 56320 in decimal, and in hex it is DC00H.

(d) TL = 00 and TH = DC (hex).

Program:

```
        MOV TMOD,#10H           ;Timer 1, mod 1
AGAIN: MOV TL1,#00              ;TL1=00,low byte of timer
        MOV TH1,#0DCH           ;TH1=DC, the high byte
        SETB TR1                ;Start timer 1
BACK: JNB TF1,BACK              ;until timer rolls over
       CLR TR1                  ;Stop the timer 1
       CLR P2.3                 ;Comp. p2.3 to get high and low
       SJMP AGAIN               ;Reload timer
                                ;mode 1 isn't auto-reload
```

Mode 2 Programming:

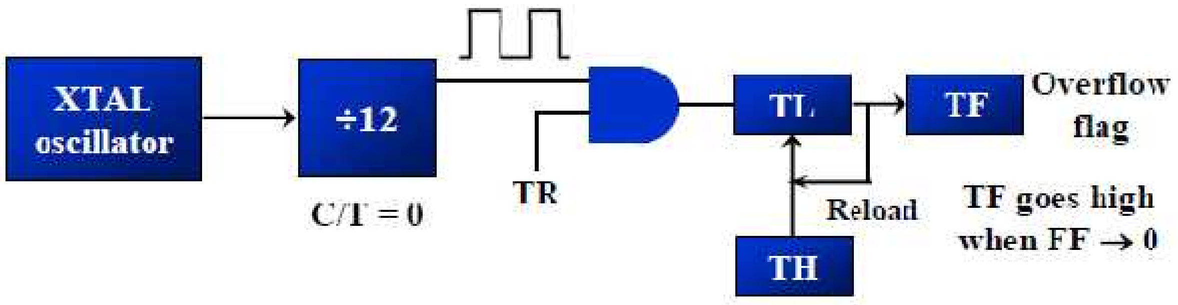
The following are the characteristics and operations of mode 2:

1. It is an 8-bit timer; therefore, it allows only values of 00 to FFH to be loaded into the timer's register TH

2. After TH is loaded with the 8-bit value, the 8051 gives a copy of it to TL
   - Then the timer must be started
   - This is done by the instruction SETB TR0 for timer 0 and SETB TR1 for timer 1

3. After the timer is started, it starts to count up by incrementing the TL register
   - It counts up until it reaches its limit of FFH
   - When it rolls over from FFH to 00, it sets high the TF (timer flag)

4. When the TL register rolls from FFH to 0 and TF is set to 1, TL is reloaded automatically with the original value kept by the TH register
   - To repeat the process, we must simply clear TF and let it go without any need by the programmer to reload the original value
   - This makes mode 2 an auto-reload, in contrast with mode 1 in which the programmer has to reload TH and TL



Steps to program in mode 2:

To generate a time delay
1. Load the TMOD value register indicating which timer (timer 0 or timer 1) is to be used, and the timer mode (mode 2) is selected.
2. Load the TH registers with the initial count value.
3. Start timer.
4. Keep monitoring the timer flag (TF) with the JNB TFx, target instruction to see whether it is raised. Get out of the loop when TF goes high.
5. Clear the TF flag and
6. Go back to Step 4, since mode 2 is auto reload.

Example 6: Assume XTAL = 11.0592 MHz, find the frequency of the square wave generated on pin P1.0 in the following program
Program:

```
        MOV TMOD, #20H          ; T1/8-bit/auto reload
        MOV TH1, #5             ; TH1 = 5
        SETB TR1                ; start the timer 1
BACK: JNB TF1, BACK             ; till timer rolls over
        CPL P1.0                ; P1.0 to high, low
        CLR TF1                 ; clear Timer 1 flag
        SJMP BACK               ; mode 2 is auto-reload
```

Solution:
First notice the target address of SJMP. In mode 2 we do not need to reload TH since it is auto-reload. Now (256 - 05) × 1.085 us = 251 × 1.085 us = 272.33 us is the high portion of the pulse. Since it is a 50% duty cycle square wave, the period T is twice that; as a result T = 2 × 272.33 us = 544.67 us and the frequency = 1.83597 kHz

Example 7: Write an ALP to generate a square wave of frequency 72Hz on pin P1.0.
Solution: Assume XTAL=11.0592MHz. With TH=00, the delay generated is 256 x 1.085 µs = 277.76 µs. therefore to generate a delay of (1 / 72) = 138.88ms, the count to be loaded is 250 x 2=500. That is

T = 2 (250 × 256 × 1.085 µs) = 138.88ms, and frequency = 72 Hz

Program:

```
          MOV TMOD, #2H         ; Timer 0, mod 2;(8-bit, auto reload)
          MOV TH0, #0
AGAIN: MOV R5, #250             ; multiple delay count
         ACALL DELAY
         CPL P1.0
         SJMP AGAIN
DELAY: SETB TR0                 ; start the timer 0
BACK: JNB TF0,BACK              ; stay timer rolls over
         CLR TR0                ; stop timer
         CLR TF0                ; clear TF for next round
         DJNZ R5,DELAY
         RET
```

Assemblers and Negative values:

Example 8: Assuming that we are programming the timers for mode 2, find the value (in hex) loaded into TH for each of the following cases.
(a) MOV TH1,#-200 (b) MOV TH0,#-60
(c) MOV TH1,#-3 (d) MOV TH1,#-12
(e) MOV TH0,#-48
Solution:
You can use the Windows scientific calculator to verify the result provided by the assembler. In Windows calculator, select decimal and enter 200. Then select hex, then +/- to get the TH value. Remember that we only use the right two digits and ignore the rest since our data is an 8-bit data.

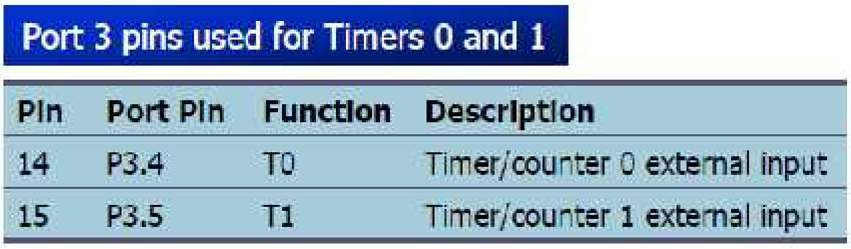
Decimal	**2's complement (TH value)**
-3	FDH
-12	F4H
-48	D0H
-60	C4H
-200	38H

The advantage of using negative values is that you don't need to calculate the value loaded to THx

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 register. 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.

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

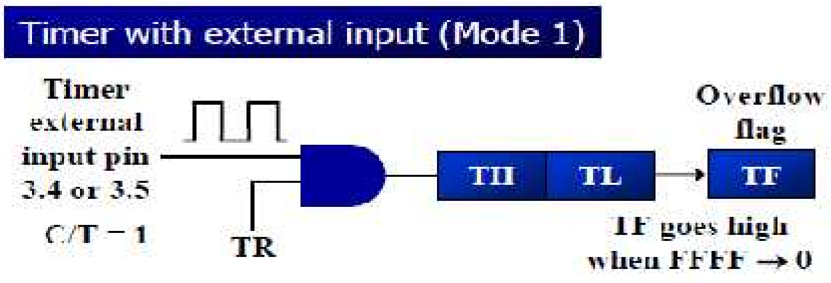
Example 9: Assuming that clock pulses are fed into pin T1, write a program for counter 1 in mode 2 to count the pulses and display the state of the TL1 count on P2, which connects to 8 LEDs.

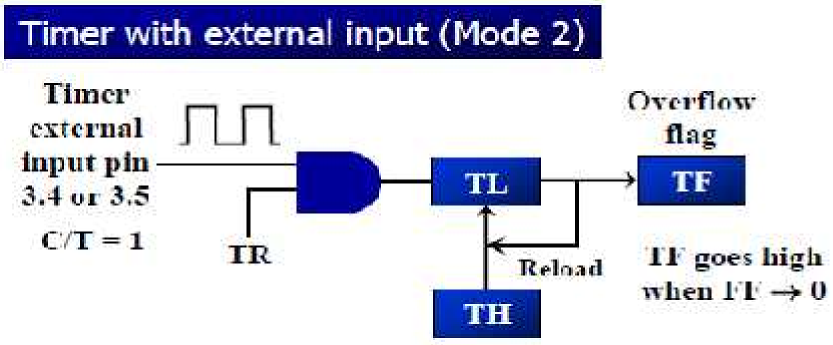
Program:

```
        MOV TM0D,#01100000B       ;counter 1, mode 2,C/T=1 external pulses
        MOV TH1,#0                ;clear TH1
        SETB P3.5                 ;make T1 input
AGAIN: SETB TR1                   ;start the counter
BACK: MOV A,TL1                   ;get copy of TL
        MOV P2,A                  ;display it on port 2
        JNB TF1,Back              ;keep doing, if TF = 0
        CLR TR1                   ;stop the counter 1
        CLR TF1                   ;make TF=0
        SJMP AGAIN                ;keep doing it
```

Solution:

Notice in the above program the role of the instruction SETB P3.5.
Since ports are set up for output when the 8051 is powered up, we make P3.5 an input port by making it high. In other words, we must configure (set high) the T1 pin (pin P3.5) to allow pulses to be fed into it.


Timer with external input (Mode 1)
Timer external input pin 3.4 or 3.5
C/T = 1
TR
TH
TL
Overflow flag
TF
TF goes high when FFFF → 0


Timer with external input (Mode 2)
Timer external input pin 3.4 or 3.5
C/T = 1
TR
TL
Overflow flag
TF
Reload
TH
TF goes high when FF → 0

TCON (timer control) register:
TCON is an 8- bit register. It is a bit addressable register.

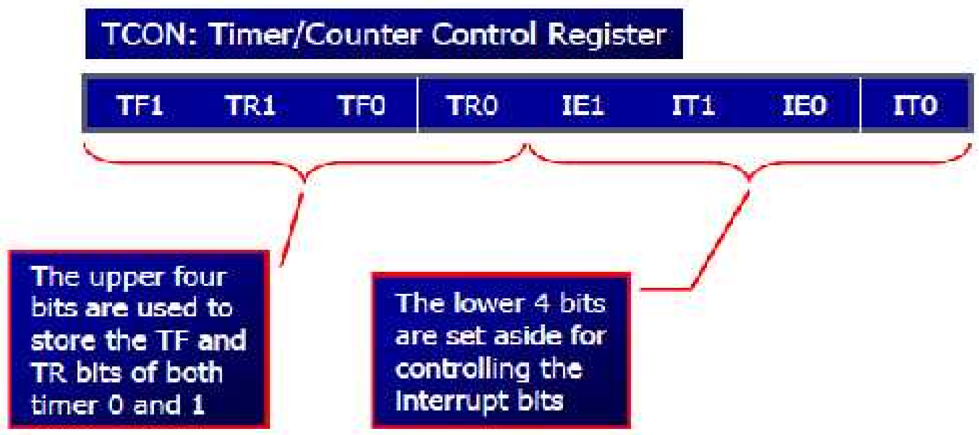
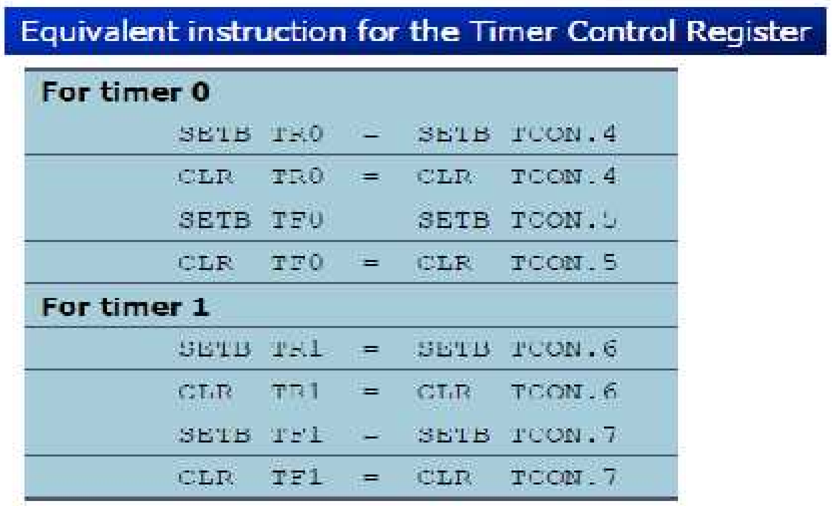

TCON: Timer/Counter Control Register
TF1
TR1
TF0
TR0
IE1
IT1
IE0
IT0
The upper four bits are used to store the TF and TR bits of both timer 0 and 1
The lower 4 bits are set aside for controlling the interrupt bits

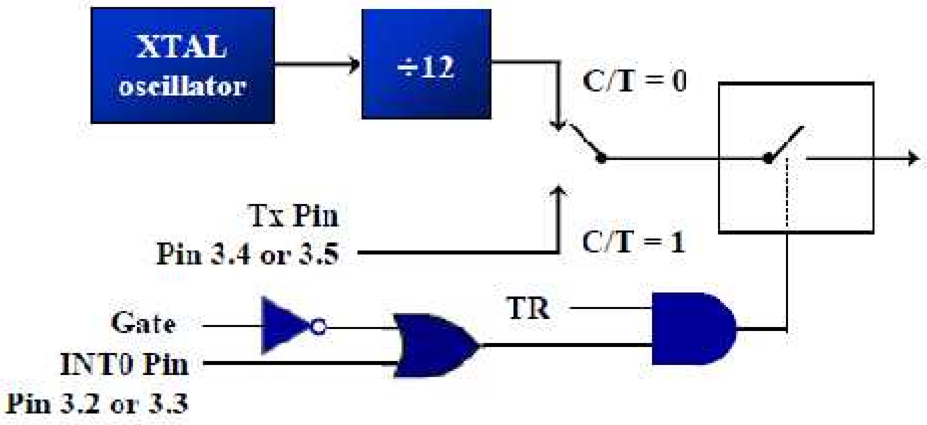
Figure 3: TCON register



Equivalent instruction for the Timer Control Register

For timer 0				
SETB	TR0	=	SETB	TCON.4
CLR	TR0	=	CLR	TCON.4
SETB	TF0	=	SETB	TCON.5
CLR	TF0	=	CLR	TCON.5
For timer 1				
SETB	TR1	=	SETB	TCON.6
CLR	TR1	=	CLR	TCON.6
SETB	TF1	=	SETB	TCON.7
CLR	TF1	=	CLR	TCON.7

If GATE = 1, the start and stop of the timer are done externally through pins P3.2 and P3.3 for timers 0 and 1, respectively. This hardware way allows starting or stopping the timer externally at any time via a simple switch.



Programming Timer0 and 1 in 8051 C:

In this section we study C programming for the 8051 timers. The general purpose registers such as R0-R7, A and B are not directly accessible by the C compiler, while the SFRs and RAM space 80-FFH is directly accessible using 8051 C statements.

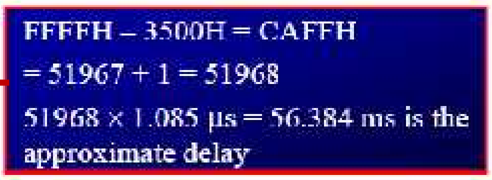
Accessing timer registers in C

In 8051 C we can access the timer registers TH, TL and TMOD directly using the reg51.h header file. This is shown through example 10. This example also shows how to access the TR and TF bits.

Example 10: 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) {                     //repeat forever
                P1=0x55;
                T0Delay();              //toggle all the bits of P1
                P1=0xAA;                // delay unknown
                T0Delay();
        }
}
void T0Delay()
{
        TMOD=0x01;              // timer 0 mode 1
```



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

```
        TL0=0x00;
        TH0=0x35;               // load TH and TL
        TR0=1;                  // turn on T0
        while (TF0==0);         // wait for TF0 to roll over
        TR0=0;                  // turn off T0
        TF0=0;                  // clear TF0
}
```

Timers 0/1 Delay Using Mode 1 (16-bit Non Auto reload)

Example 11, 12 and 13 show 8051 C programming of the timers 0 and 1 in mode 1.

Example 11: Write an 8051 C program to toggle only bit P1.5 continuously every 50ms. Use Timer 0, mode 1 (16-bit) to create the delay.

Solution:

Assume XTAL=11.0592 MHz=> T=1.085μs Count=50ms/1.085μs =46083
Initial count = 65536-46083 =19453
Count in Hex = 4BFDH

Program

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

Example 12: 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:

Assume XTAL=11.0592 MHz=> T=1.085μs Count=500ms/1.085μs =460829 >65536
Let us divide this delay as = 20x25ms, hence count for 25ms can be calculated.
Count = 25ms/1.085μs = 23042
Initial count = 65536 – 23042= 42494
Count in hex = A5FEH

Program

```
#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;
}
```

Example 13: 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, mode 1 for both of them.

Solution: **Values to be loaded into TH and TL for 500Hz and 750Hz can be calculated as done for example 11 and 12.**

```
#include <reg51.h>
sbit mybit=P1^5;
sbit SW=P1^7;
void T0M1Delay(unsigned char);
void main(void){
        SW=1;                       // make P1.7 as input
        while (1) {
                mybit=~mybit;       // toggle P1.5
                if (SW==0)          // check switch
                T0M1Delay(0);
                else
                T0M1Delay(1);
        }
}

void T0M1Delay(unsigned char c){
TMOD=0x01;
if (c==0) {
        TL0=0x67;
        TH0=0xFC;
}
else {
        TL0=0x9A;
        TH0=0xFD;
}
TR0=1;
while (TF0==0);
TR0=0;
TF0=0;
}
```

Timers 0 and 1 Delay Using Mode 2 (8-bit Auto-reload)

Examples 14 and 15 show 8051 C programming of timers 0 and 1 in mode 2.

Example 14: Write an 8051 C program to toggle only pin P1.5 continuously every 250 ms. Use Timer 0, mode 2 (8-bit auto-reload) to create the delay.

Solution:

Solution: Assume XTAL=11.0592MHz

For the delay of 250ms the count exceeds 256. hence, count for 25μs is calculated and count is 23

Therefore for 250ms =>25μs × 250 × 40 = 250 ms

Program

```
#include <reg51.h>
void T0M2Delay(void);
sbit mybit=P1^5;
void main(void){
        unsigned char x,y;
        while (1) {
                mybit=~mybit;
                for (x=0;x<250;x++)
                for (y=0;y<36;y++) //we put 36, not 40
                T0M2Delay();
        }
}
void T0M2Delay(void){
        TMOD=0x02;
        TH0=-23;
        TR0=1;
        while (TF0==0);
        TR0=0;
        TF0=0;
}
```

Example 15: Write an 8051 C program to create a frequency of 2500 Hz on pin P2.7. Use Timer 1, mode 2 to create delay.

Solution:

XTAL = 11.0592MHz, given freq of wave = 2500Hz
Time Period=1/2500Hz = 400µs
Ton=Toff= 400µs /2 = 200µs
Count = 200µs / 1.085µs = 184
Initial count =256-184 = 72
In hex = 48H

Program

```
#include <reg51.h>
void T1M2Delay(void);
sbit mybit=P2^7;
void main(void){
        unsigned char x;
        while (1) {
                mybit=~mybit;
                T1M2Delay();
        }
}
void T1M2Delay(void){
        TMOD=0x20;
        TH1=-184;
        TR1=1;
        while (TF1==0);
        TR1=0;
        TF1=0;
}
```

C Programming of Timers 0 and 1 as Counters

Example 16: Assume that a 1-Hz external clock is being fed into pin T1 (P3.5). Write a C program for counter 1 in mode 2 (8-bit auto reload) to count up and display the state of the TL1 count on P1. Start the count at 0H.

Program:

```
#include <reg51.h>
sbit T1=P3^5;
void main(void){
            T1=1;
            TMOD=0x60;
            TH1=0;
            while (1) {
                do {
                     TR1=1;
                      P1=TL1;
                     } while (TF1==0);
            TR1=0;
            TF1=0;
            }
}
```

Example 17: Assume that a 1-Hz external clock is being fed into pin T0 (P3.4). Write a C program for counter 0 in mode 1 (16-bit) to count the pulses and display the state of the TH0 and TL0 registers on P2 and P1, respectively.

Program:

```
#include <reg51.h>
void main(void){
            T0=1;
            TMOD=0x05;
            TL0=0
            TH0=0;
            while (1) {
                 do {
                       TR0=1;
                        P1=TL0;
                        P2=TH0;
                    } while (TF0==0);
            TR0=0;
            TF0=0;
            }
}
```

Summary

The 8051 has 2 timers/counters. When used as timers they cam generate time delays. When used as counters they can serve as event counters. This chapter showed how to program timer/counter for various modes.

The two timers are accessed as two 8-bit registers TH0/1 and TL0/1 for timer0/1. Both use TMOD register to set timer operation modes. The lower 4 bits of TMOD are used for timer 0 while the upper 4-bits are used for timer 1. The different modes with which timers/counters operate are: mode 0, mode 1 and mode 2.

When the timer/counter is used as timer, the 8051's crystal is used as the source of the frequency; when it is used as a counter, however, it is a pulse outside the 8051 that increments the TH and TL register.