

## PROGRAMMING 8051 TIMERS

### 8051 TIMERS

8051 has two timers/Counters(TIMER 0 and TIMER 1). They can be used as timers to generate time delays or as event counters.

### BASIC REGISTERS OF THE TIMER

#### 1.TIMER REGISTERS

It is a 16 bit register and can be accessed as 8 bit registers say Timer High(THx) and Timer Low (TLx), These registers can be accessed as any other registers like A,B,R0 etc.,

THx								TLx							
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

**Figure 5.1.1 Timer Registers**

[Source: “The 8051 Microcontroller and Embedded Systems: Using Assembly and C” by Mohamed Ali Mazidi, Janice Gillispie Mazidi, Rolin McKinlay]

#### 2. TMOD (Timer Mode) Register

TMOD Register is an 8-bit register used to control the mode of operation of both timers. The high four bits (bits 4 through 7) relate to Timer 1 whereas the low four bits (bits 0 through 3) perform the exact same functions, but for timer 0. In each case, the lower two bits are used to set the timer mode and upper two bits to specify the operation as shown in Figure 5.1.2.

(MSB)				(LSB)			
GATE	C/T	M1	M0	GATE	C/T	M1	M0
Timer 0				Timer 1			
GATE	Gating control when set, Timer/Counter "x" is enabled only while "INTx" pin is high and "TRx" control bit is set. When cleared Timer "x" is enabled whenever "TRx" control bit is set.	M1 M0		Operating Mode			
		0 0		8-bit Timer/Counter "THx" with "TLx"s 5-bit prescaler.			
		0 1		16-bit Timer/Counter "THx" with "TLx" are cascaded; there is no prescaler.			
		1 0		8-bit auto-reload Timer/Counter "THx" holds a value which is to be reloaded into "TLx" each time it overflows.			
		1 1		(Timer 0) TL0 is an 8-bit Timer/Counter controlled by the standard Timer 0 control bits. TH0 is an 8-bit timer only controlled by Timer 1 control bits.			
C/T	Timer or Counter selector cleared for Timer operation (input from internal system clock). Set for Counter operation (input from "Tx" input pin).	1 1		(Timer 1) Timer/Counter 1 stopped.			

**Figure 5.1.2 TMOD Register**

[Source: "The 8051 Microcontroller and Embedded Systems: Using Assembly and C" by Mohamed Ali Mazidi, Janice Gillispie Mazidi, Rolin McKinlay, pg.no.241]

In upper or lower 4 bits, first bit is a GATE bit. Every timer has a means of starting and stopping. Some timers do this by software, some by hardware, and some have both software and hardware controls. The hardware way of starting and stopping the timer by an external source is achieved by making GATE=1 in the TMOD register and if GATE=0 then we do start and stop the timers by programming.

The second bit is C/T bit and is used to decide whether a timer is used as a time delay generator or an event counter. If this bit is 0 then it is used as a timer and if it is 1 then it is used as a counter.

In upper or lower 4 bits, the last bits third and fourth are known as M1 and M0 respectively. These are used to select the timer mode.

### TIMER'S CLOCK FREQUENCY AND ITS PERIOD

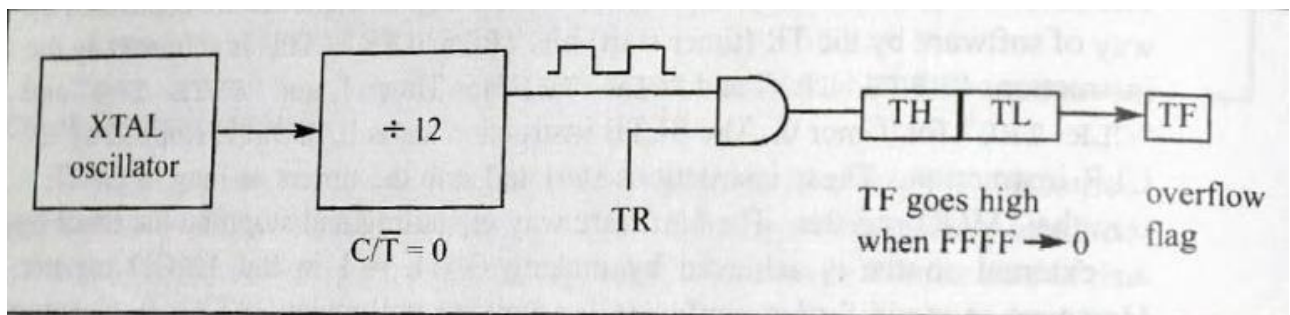
In 8051-based system, the crystal oscillator has a frequency of 11.0592 MHz when C/T bit of TMOD is 0. Each machine cycle is made up of 12 clock cycles.

Hence for a single machine cycle, the frequency becomes  $1/12 \times 11.0529 \text{ MHz} = 921.6 \text{ KHz}$ . For a single machine cycle, the time taken is  $T = 1/921.6 \text{ KHz} = 1.085 \text{ us}$ , so the oscillator takes  $1.085 \text{ us}$  for completing a single machine cycle.

## MODES OF OPERATION:

### MODE 1:

It is a 16-bit timer; therefore it allows values from 0000 to FFFFH to be loaded into the timer's registers TH and TL as shown in Figure 5.1.3. After TH and TL are loaded with a 16-bit initial value, the timer must be started. We can do it by "SETB TR0" for timer 0 and "SETB TR1" for timer 1. After the timer is started, it starts count up until it reaches its limit of FFFFH. When it rolls over from FFFF to 0000H, it sets high a flag bit called TFx (timer flag). This timer flag can be monitored. When this timer flag is raised, one option would be stop the timer with the instructions "CLR TR0" or CLR TR1 for timer 0 and timer 1 respectively. Again, it must be noted that each timer flag TF0 for timer 0 and TF1 for timer1. After the timer reaches its limit and rolls over, in order to repeat the process the registers TH and TL must be reloaded with the original value and TF must be reset to 0.



**Figure 5.1.3 Timer in Mode 1**

[Source: "The 8051Microcontroller and Embedded Systems: Using Assembly and C" by Mohamed Ali Mazidi, Janice Gillispie Mazidi, Rolin McKinlay, pg.no.244]

### MODE 0 :

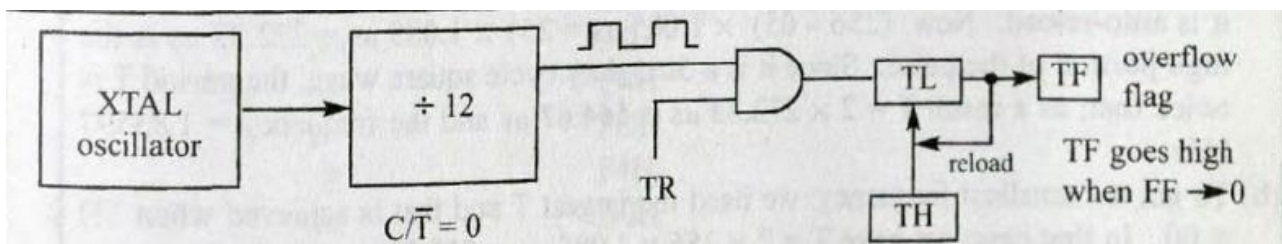
Mode 0 is exactly same like mode 1 except that it is a 13-bit timer instead of 16-bit. The 13- bit counter can hold values between 0000 to 1FFFH in TH-TL.

Therefore, when the timer reaches its maximum of 1FFH, it rolls over to 0000, and TF is raised.

### MODE 2:

It is an 8 bit timer that allows only values of 00 to FFH to be loaded into the timer's register TH as shown in Figure 5.1.4. After THx is loaded with 8 bit value, the 8051 gives a copy of it to TLx. Then the timer must be started. It is done by the instruction "SETB TR0" for timer 0 and "SETB TR1" for timer1. This is like mode 1.

After timer is started, it starts to count up by incrementing the TLx register. It counts up until it reaches its limit of FFH. When it rolls over from FFH to 00, it sets high the TFx (timer flag). If we are using timer 0, TF0 goes high; if using TF1 then TF1 is raised. When TLx register rolls from FFH to 00 and TF is set to 1, TLx is reloaded automatically with the original value kept by the THx register. To repeat the process, we must simply clear TFx and let it go without any need by the programmer to reload the original value. This makes mode 2 auto reload, in contrast in mode 1 in which programmer has to reload THx and TLx.



**Figure 5.1.4 Timer in Mode 2**

[Source: "The 8051 Microcontroller and Embedded Systems: Using Assembly and C" by Mohamed Ali Mazidi, Janice Gillispie Mazidi, Rolin McKinlay, pg.no.251]

### MODE 3:

Mode 3 is also known as a split timer mode. Timer 0 and 1 may be programmed to be in mode 0, 1 and 2 independently of similar mode for other timer. This is not true for mode 3; timers do not operate independently if mode 3 is chosen for timer

0. Placing timer 1 in mode 3 causes it to stop counting; the control bit TR1 and the timer 1 flag TF1 are then used by timer 0.

## **TIMER PROGRAMMING**

### **MODE 1 PROGRAMMING:**

It is a 16 bit Timer mode.

#### **STEPS TO PROGRAM IN MODE 1:**

1. Load the TMOD value register with mode and timer0 or 1.
2. Load registers TLx and THx with initial count corresponding to delay.
3. Start the timer.
4. Continuously monitor the timer flag (TFx) with the “JNB TFx,target” instruction to see if it is raised, if it is raised(TFx=1) then get out of the loop.
5. Stop the timer.
6. Clear the TFx flag for the next round.
7. Go back to Step 2 to load THx and TLx again.

#### **STEPS TO CALCULATE COUNT TO LOAD INTO THX-TLX TO GENERATE DESIRED DELAY**

(Assume XTAL = 11.0592 MHz)

Steps for finding the TH, TL registers values

1. Divide the desired time delay by 1.085 us
2. Perform  $65536 - n$ , where n is the decimal value we got in Step 1
3. Convert the result of Step 2 to hex, say we get yyxx. yyxx is the initial hex value to be loaded into the timer's register.
4. Set TL = xx and TH = yy **Example:**

Generate Delay =10ms with Clock frequency= 11.0592 MHz, using Timer0 in mode1.

#### **Solution:**

Given:

Time delay=10ms

Clock frequency=11.0592 MHz

**Step 1:** Divide the desired time delay by 1.085 us

$$\text{Count} = 10 \text{ ms} / 1.085 \text{ us} = 9216$$

**Step 2:** Perform  $65536 - n$

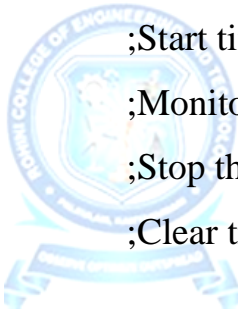
$$65536 - 9216 = 56320 = \text{DC00H}$$

**Step 3:** Set TL = xx and TH = yy

Here xx=DC and yy=00, Hence, TH0=DC and TL0=00.

*Following is the assembly code program to generate a delay of 10ms.*

```
MOV TMOD, #01          ;Timer 0, mode 1, 16-bitmode
HERE: MOV TL0, #00      ;TL0=0, the low byte
MOV TH0, #0DCH          ;TH0=DC, the high byte
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
```



*Program to generate a square wave of 5 kHz frequency on pin P1.0, clock frequency =11.0592 MHz Given:*

Square wave frequency=5 kHz

Clock frequency=11.0592 MHz

**Step 1:** Calculate the Time delay

$$T = 1/f = 1/5 \text{ kHz} = 0.2 \text{ ms}$$

T=0.2 ms which is the period of square wave

$$T/2 = 0.2/2 = 0.1 \text{ ms delay for high and low}$$

**Step 2:** Divide the desired time delay by 1.085 us

$$\text{Count} = 0.1 \text{ ms} / 1.085 \text{ us} = 921$$

**Step 3:** Perform  $65536 - n$

$$\text{TH0-TL0} = 65536 - 921 = 64615 = \text{FC67H}$$

Step 4: Set TL = xx and TH = yy

Here xx=FC and yy=67. Hence, TH0=FCh, TL0=67h



```
MOV TMOD,#01          ;Timer 0, mode 1, 16-bitmode
AGAIN: MOV TL1,#67H    ;TL1=67, low byte of timer
MOV TH1,#0FCH          ;TH1=FC, the high byte
SETB TR1               ;Start timer 1  BACK:  JNB  TF1,BACK  ;until
timer rolls over
CPL P1.0               ; compliment P1.0
CLR TR1                ;Stop the timer 1
CLR TF1                ;Clear timer 1 flag
SJMP AGAIN             ;Reload timer
```

### STEPS FOR GENERATING DELAY IN MODE 2

1. Select timer in TMOD register indicating which timer (timer 0 or timer 1) is to be used, and the timer mode (mode 2) to be selected 2. THx register loaded with initial count value
3. Start timer
4. Continuously monitoring the timer flag (TFx) with the JNB TFx,target instruction to see whether TFx is '1' (high) . Get out of the loop when TF goes high
5. Clear the TFx flag
6. Go back to Step4, since mode 2 is auto-reload

### ***Toggle LED connected at P1.0 with 5 microsec delay using timer1 and mode 2***

```
MOV TMOD,#20          ;Timer 1 mode 2, 8-bit auto reload
AGAIN: MOV TH1,#-5    ;TL1=256-5, low byte of timer
SETB TR1              ;Start timer 1
BACK: JNB TF1, BACK  ;until timer rolls over
CPL P1.0              ; compliment P1.0 toggle LED
CLR TF1               ;Stop the timer 1
SJMP BACK              ; loop
```