Programming 8051 Timers

- 8051 has two timers: Timer 0 and Timer 1.
- They can be used either as timers or as event counters.
- Both timers are 16 bits wide.
- Since 8051 has an 8-bit architecture, each 16-bit timer is accessed as two separate registers of low byte and high byte.
- Timer 0 registers:
  - The low byte register is called TL0 and high byte register is referred to as TH0.
  - These register can be accessed like any other register, such as A, B, R0 etc.
    Example: Instruction “MOV TL0, #4FH” moves the value 4FH into TL0.
  - These register can also be read like any other register.
    Example: “MOV R5, TH0”
- Timer 1 registers:
  - The low byte register is called TL1 and high byte register is referred to as TH1.

Programming 8051 Timers

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

Figure 9-1. Timer 0 Registers

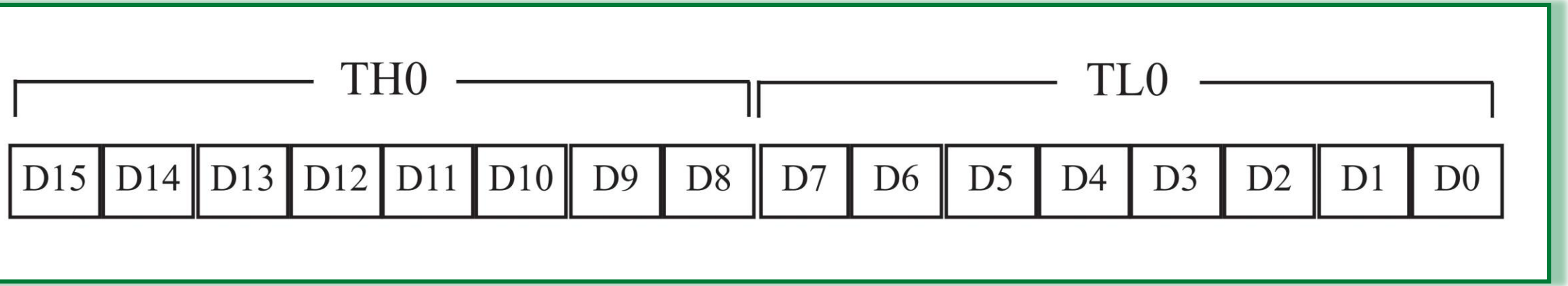


Figure 9-2. Timer 1 Registers

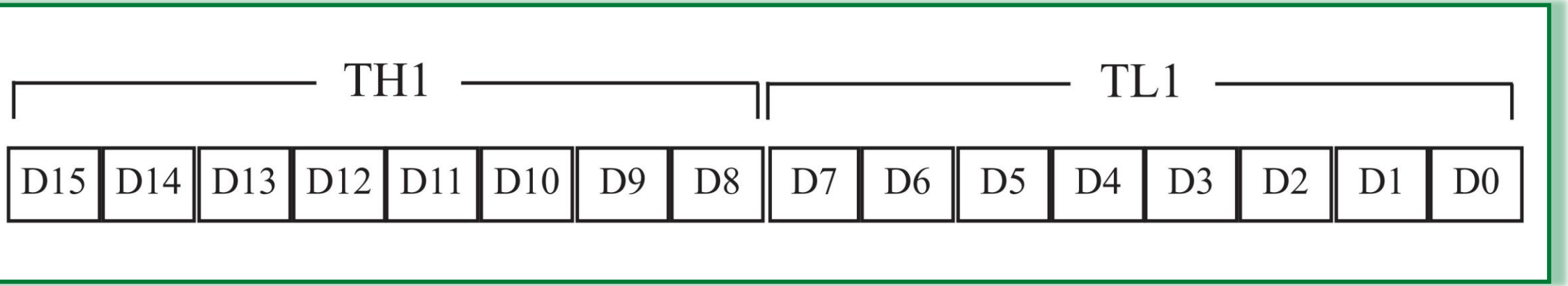


Figure 9-3. TMOD Register

(MSB) (LSB)

GATE	C/T	M1	M0	GATE	C/T	M1	M0
Timer 1				Timer 0			

GATE Gating control when set. The timer/counter is enabled 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.

C/T Timer or counter selected cleared for timer operation (input from internal system clock). Set for counter operation (input from Tx input pin).

M1 Mode bit 1

M0 Mode bit 0

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/counters 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 that is to be reloaded into TLx each time it overflows.
1	1	3	Split timer mode

C/T(clock/timer): Decide whether timer is used as delay generator or event counter. C/T=0, it is used as timer for delay generation.

Clock source for timer: Every timer needs a clock pulse to tick. Clock source for time delay is the crystal frequency of 8051.

GATE: Start and stop of timers are controlled by way of software by the TR (timer start) bits TR0 and TR1.
SETB start it and it is stopped by CLR as long as GATE=0 in TMOD register.
Eg. “SETB TR1” and “CLR TR1” for Timer 1

M1, M0: M0 and M1 select the timer mode

Example 9-1

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:

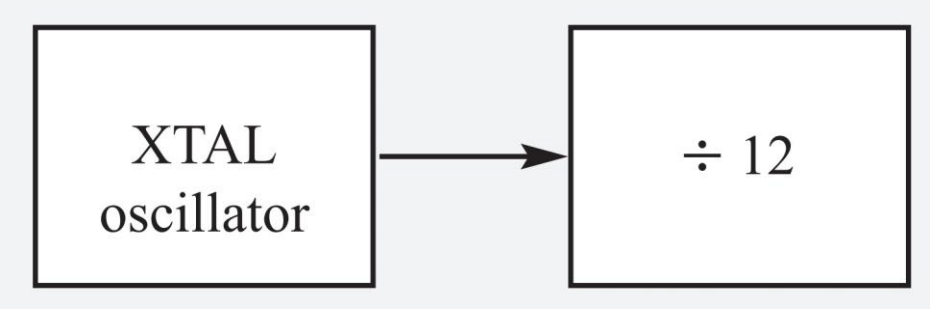
We convert the values from hex to binary. From Figure 9-3 we have:

(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.

Example 9-2

Find the timer's clock frequency and its period for various 8051-based systems, with the following crystal frequencies.
(a) 12 MHz (b) 16 MHz (c) 11.0592 MHz

Solution:



NOTE THAT 8051 TIMERS USE 1/12 OF XTAL FREQUENCY, REGARDLESS OF MACHINE CYCLE TIME.

(a) 1/12 x 12 MHz = 1 MHz and T = 1/1 MHz = 1 µs
(b) 1/12 x 16 MHz = 1.333 MHz and T = 1/1.333 MHz = .75 µs
(c) 1/12 x 11.0592 MHz = 921.6 kHz;
T = 1/921.6 kHz = 1.085 µs

Example 9-3

Find the value for TMOD if we want to program Timer 0 in mode 2, use 8051 XTAL for the clock source, and use instructions to start and stop the timer.

Solution:

TMOD = 0000 0010 Timer 0, mode 2,
C/T = 0 to use XTAL clock source, and
gate = 0 to use internal (software)
start and stop method.

Mode 1 programming

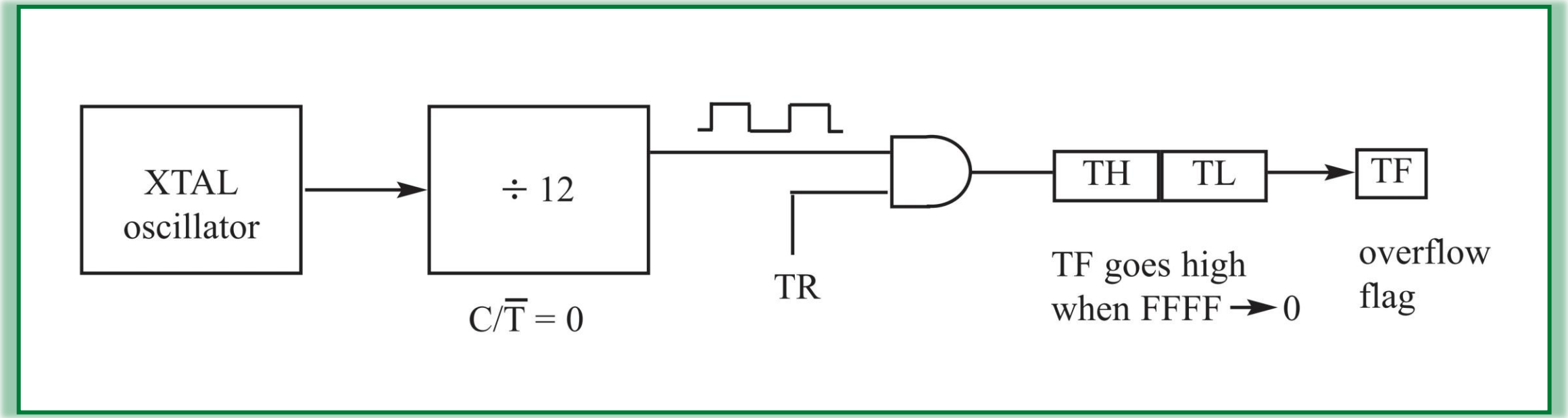


Figure 9-4. Timer Delay Calculation for XTAL = 11.0592 MHz

(a) in hex

(FFFF - YYXX + 1) x 1.085 µs
where YYXX are TH, TL initial values respectively. Notice that values YYXX are in hex.

(b) in decimal

Convert YYXX values of the TH,TL register to decimal to get a NNNNN decimal number, then(65536 - NNNNN) x 1.085 µs

Example 9-4

In the following program, we are creating 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              ;load TH, TL again
;————————delay using Timer 0
DELAY:
            SETB    TR0               ;start Timer 0
AGAIN:      JNB     TF0,AGAIN         ;monitor Timer 0 flag until
                                      ;it rolls over
            CLR     TR0               ;stop Timer 0
            CLR     TF0               ;clear Timer 0 flag
```

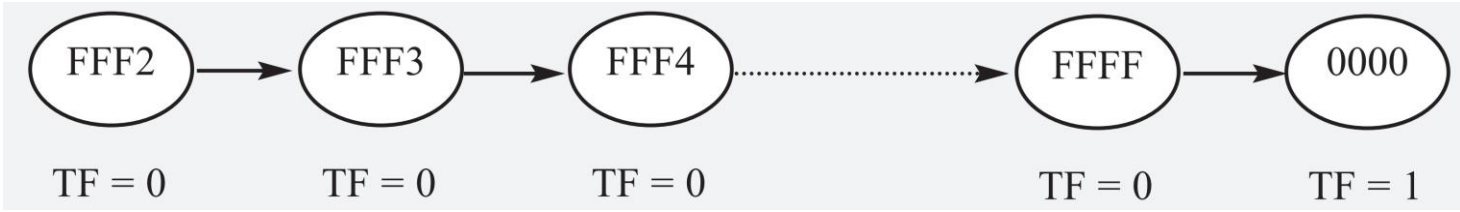
Example 9-4

Solution:

In the above program notice the following steps.

Notice that to repeat the process, we must reload the TL and TH registers and start the timer again.

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.



Example 9-5

In Example 9-4, calculate the amount of time delay in the DELAY subroutine generated by the timer. Assume that XTAL = 11.0592 MHz.

Solution:

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 / 921.6 kHz = 1.085 75 µs. In other words, Timer 0 counts up each 1.085 75 µs resulting in delay = number of counts x 1.085 75 µs. The number of counts for the rollover 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 raises the TF flag. This gives 14 x 1.085 75 µs = 15.19 75 µs for half the pulse. For the entire period T = 2 x 15.19 75 µs = 30.38 75 µs gives us the time delay generated by the timer.

Example 9-6

In Example 9-5, calculate the frequency of the square wave generated on pin P1.5.

Solution:

In the time delay calculation of Example 9-5, we did not include the overhead due to instructions in the loop. To get a more accurate timing, we need to add clock cycles due to the instructions in the loop. To do that, we use the machine cycles 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 using Timer 0
DELAY:
                SETB    TR0                         1
AGAIN:  JNB      TF0,AGAIN                          14
                CLR     TR0                         1
                CLR     TF0                         1
                RET                                 2
                                        Total       28
```

T = 2 x 28 x 1.085 µs = 60.76 µs and F = 16458.2 Hz.

NOTE THAT 8051 TIMERS USE 1/12 OF XTAL FREQUENCY, REGARDLESS OF MACHINE CYCLE TIME.

Example 9-7

Find the delay generated by Timer 0 in the following code, using both of the methods of Figure 9-4. Do not include the overhead due to instructions.

```
        CLR     P2.3            ;clear P2.3
        MOV     TMOD,#01        ;Timer 0, mode 1(16-bit mode)
HERE:   MOV     TL0,#3EH        ;TL0 = 3EH, Low byte
        MOV     TH0,#0B8H       ;TH0 = B8H, High byte
        SETB    P2.3            ;SET high P2.3
        SETB    TR0             ;start Timer 0
AGAIN:  JNB     TF0,AGAIN       ;monitor Timer 0 flag
        CLR     TR0             ;stop Timer 0
        CLR     TF0             ;clear Timer 0 flag for
                                ;next round
        CLR     P2.3
```

Solution:

(a) (FFFF - B83E + 1) = 47C2H = 18370 in decimal and 18370 x 1.085 µs = 19.93145 µs.

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

Example 9-8

Modify TL and TH in Example 9-7 to get the largest time delay possible. Find the delay in ms. In your calculation, exclude the overhead due to the instructions in the loop.

Solution:

To get the largest delay we make TL and TH both 0. This will count up from 0000 to FFFFH and then roll over to zero.

```
        CLR    P2.3          ;clear P2.3
        MOV    TMOD,#01      ;Timer 0, mode 1(16-bit mode)
HERE:   MOV    TL0,#0        ;TL0 = 0, Low byte
        MOV    TH0,#0        ;TH0 = 0, High byte
        SETB   P2.3          ;SET P2.3 high
        SETB   TR0           ;start Timer 0
AGAIN:  JNB    TF0,AGAIN     ;monitor Timer 0 flag
        CLR    TR0           ;stop Timer 0
        CLR    TF0           ;clear Timer 0 flag
        CLR    P2.3
```

Making TH and TL both zero means that the timer will count from 0000 to FFFFH, and then roll over to raise the TF flag. As a result, it goes through a total of 65536 states. Therefore, we have delay = (65536 - 0) x 1.085µs = 71.1065 ms.

Example 9-9

The following program generates a square wave on pin P1.5 continuously using Timer 1 for a time delay. Find the frequency of the square wave if XTAL = 11.0592 MHz. In your calculation do not include the overhead due to instructions in the loop.

```
        MOV   TMOD,#10H      ;Timer 1, mode 1(16-bit)
AGAIN:  MOV   TL1,#34H       ;TL1 = 34H, Low byte
        MOV   TH1,#76H       ;TH1 = 76H, High byte
                             ;(7634H = timer value)
        SETB  TR1            ;start Timer 1
BACK:   JNB   TF1,BACK       ;stay until timer rolls over
        CLR   TR1            ;stop Timer 1
        CPL   P1.5           ;comp. P1.5 to get hi, lo
        CLR   TF1            ;clear Timer 1 flag
        SJMP  AGAIN          ;reload timer since Mode 1
```

Example 9-9

Solution:

In the above program notice the target of SJMP. In mode 1, the program must reload the TH, TL register every time if we want to have a continuous wave. Now the calculation.
Since FFFFH - 7634H = 89CBH + 1 = 89CCH and 89CCH = 35276 clock count.
35276 x 1.085 µs = 38.274 ms for half of the square wave. The entire square wave length is 38.274 x 2 = 76.548 ms and has a frequency = 13.064 Hz.

Also notice that the high and low portions of the square wave pulse are equal. In the above calculation, the overhead due to all the instructions in the loop is not included.

Example 9-10

Assume that XTAL = 11.0592 MHz. What value do we need to load into the timer's registers 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 µs. This means that out of many 1.085 µs intervals we must make a 5 ms pulse. To get that, we divide one by the other. We need 5 ms / 1.085 µs = 4608 clocks. To achieve that we need to load into TL and TH the value 65536 - 4608 = 60928 = EE00H. Therefore, we have TH = EE and TL = 00.

```
        CLR     P2.3            ;clear P2.3
        MOV     TMOD,#01        ;Timer 0, mode 1 (16-bit mode)
HERE:   MOV     TL0,#0          ;TL0 = 0, Low byte
        MOV     TH0,#0EEH       ;TH0 = EE( hex), High byte
        SETB    P2.3            ;SET P2.3 high
        SETB    TR0             ;start Timer 0
AGAIN:  JNB     TF0,AGAIN       ;monitor Timer 0 flag
                                ;until it rolls over
        CLR     P2.3            ;clear P2.3
        CLR     TR0             ;stop Timer 0
        CLR     TF0             ;clear Timer 0 flag
```

Example 9-11

Assuming 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 µs the period of the square wave.

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

(c) 250 µs / 1.085 µs = 230 and 65536 - 230 = 65306, which in hex is FF1AH.

(d) TL = 1AH and TH = FFH, all in hex. The program is as follows.

```
                MOV     TMOD,#10H       ;Timer 1, mode 1(16-bit)
AGAIN:          MOV     TL1,#1AH        ;TL1=1AH, Low byte
                MOV     TH1,#0FFH       ;TH1=FFH, High byte
                SETB    TR1             ;start Timer 1
BACK:           JNB     TF1,BACK        ;stay until timer rolls over
                CLR     TR1             ;stop Timer 1
                CPL     P1.5            ;complement P1.5 to get hi, lo
                CLR     TF1             ;clear Timer 1 flag
                SJMP    AGAIN           ;reload timer since mode 1
                                        ;is not auto-reload
```

Example 9-12

Assuming XTAL = 11.0592 MHz, write a program to generate a square wave of 50 Hz frequency on pin P2.3.

Solution:

Look at the following steps.

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

(b) 1/2 of it for the high and low portions of the pulse = 10 ms

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

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

The program follows.

```
                MOV     TMOD,#10H           ;Timer 1, mode 1 (16-bit)
AGAIN:          MOV     TL1,#00             ;TL1 = 00, Low byte
                MOV     TH1,#0DCH           ;TH1 = DCH, High byte
                SETB    TR1                 ;start Timer 1
BACK:           JNB     TF1,BACK            ;stay until timer rolls over
                CLR     TR1                 ;stop Timer 1
                CPL     P2.3                ;comp. P2.3 to get hi, lo
                CLR     TF1                 ;clear Timer 1 flag
                SJMP    AGAIN               ;reload timer since mode 1
                                            ;is not auto-reload
```

Example 9-13

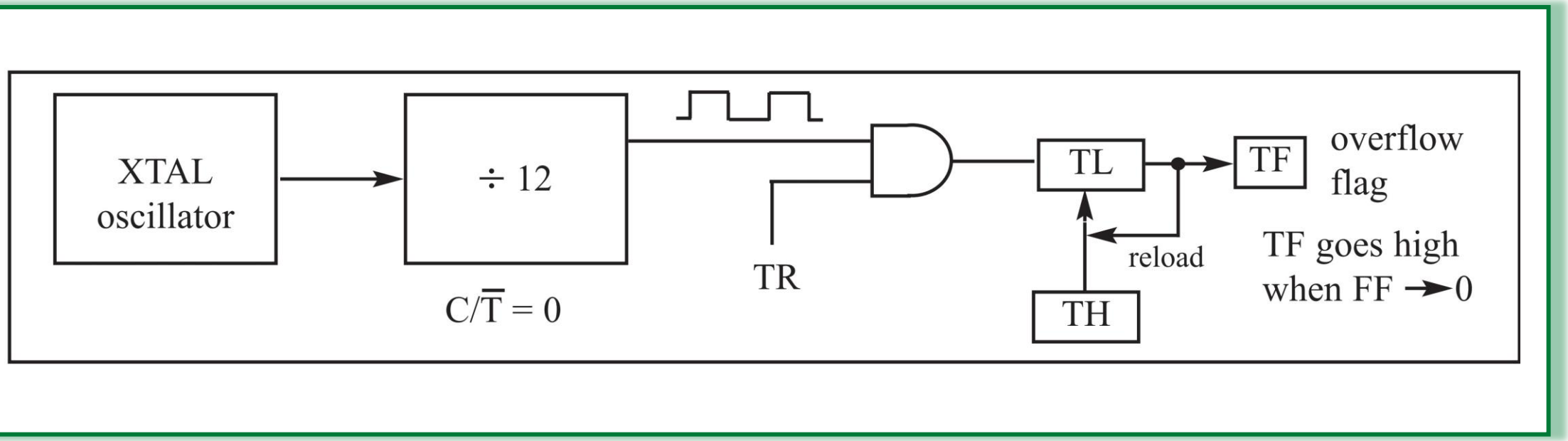
Examine the following program and find the time delay in seconds. Exclude the overhead due to the instructions in the loop.

```
                MOV     TMOD,#10H           ;Timer 1, mode 1(16-bit)
                MOV     R3,#200             ;counter for multiple delay
AGAIN:          MOV     TL1,#08H            ;TL1 = 08, Low byte
                MOV     TH1,#01H            ;TH1 = 01, High byte
                SETB    TR1                 ;start Timer 1
BACK:           JNB     TF1,BACK            ;stay until timer rolls over
                CLR     TR1                 ;stop Timer 1
                CLR     TF1                 ;clear Timer 1 flag
                DJNZ    R3,AGAIN            ;if R3 not zero then
                                            ;reload timer
```

Solution:

TH - TL = 0108H = 264 in decimal and 65536 - 264 = 65272. Now 65272 x 1.085 µs = 70.820 ms, and for 200 of them we have 200 x 70.820 ms = 14.164024 seconds.

Mode 2 programming



Example 9-14

Assuming that XTAL = 11.0592 MHz, find (a) the frequency of the square wave generated on pin P1.0 in the following program, and (b) the smallest frequency achievable in this program, and the TH value to do that.

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

Solution:

Solution:

(a) 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) x 1.085 µs = 251 x 1.085 µs = 272.33 µs 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 x 272.33 µs = 544.67 µs and the frequency = 1.83597 kHz.

(b) To get the smallest frequency, we need the largest T and that is achieved when TH = 00. In that case, we have T = 2 x 256 x 1.085 µs = 555.52 µs and the frequency = 1.8 kHz.

Example 9-15

Find the frequency of a square wave generated on pin P1.0.

Solution:

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

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

Example 9-16

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 results 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. The following is what we get.

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

Example 9-17

Find (a) the frequency of the square wave generated in the following code, and (b) the duty cycle of this wave.

```
                MOV     TMOD,#2H                ;Timer 0, mode 2
                                                ;(8-bit, auto-reload)
                MOV     TH0,#-150               ;TH0 = 6AH = 2's comp of -150
AGAIN:          SETB    P1.3                    ;P1.3 = 1
                ACALL   DELAY
                ACALL   DELAY
                CLR     P1.3                    ;P1.3 = 0
                ACALL   DELAY
                SJMP    AGAIN
DELAY:
                SETB    TR0                     ;start Timer 0
BACK:           JNB     TF0,BACK                ;stay until timer rolls over
                CLR     TR0                     ;stop Timer 0
                CLR     TF0                     ;clear TF for next round
                RET
```

Solution:

For the TH value in mode 2, the conversion is done by the assembler as long as we enter a negative number. This also makes the calculation easy. Since we are using 150 clocks, we have time for the DELAY subroutine = 150 x 1.085 µs = 162 µs. The high portion of the pulse is twice that of the low portion (66% duty cycle). Therefore, we have: T = high portion + low portion = 325.5 µs + 162.25 µs = 488.25 µs and frequency = 2.048 kHz.

Table 9-1: 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

(MSB) (LSB)

GATE	C/T	M1	M0	GATE	C/T	M1	M0
Timer 1				Timer 0			

Example 9-18

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.

Solution:

```
        MOV   TMOD,#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 count TL1
        MOV   P2,A            ;display it on port 2
        JNB   TF1,BACK        ;keep doing it if TF=0
        CLR   TR1             ;stop the counter 1
        CLR   TF1             ;make TF=0
        SJMP  AGAIN           ;keep doing it
```

P2 is connected to 8 LEDs and input T1 to pulse.

Notice in the above program the role of the instruction "SETB P3.5". Although ports are set up for input when the 8051 is powered up, we still make P3.5 an input port (by making it high) to make sure it is an input since some other programs could have used it as an output. In other words, we must configure (set high) the T1 pin (pin P3.5) to allow pulses to be fed into it.

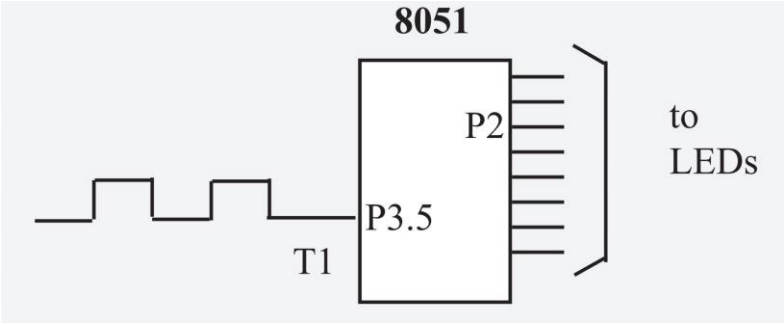
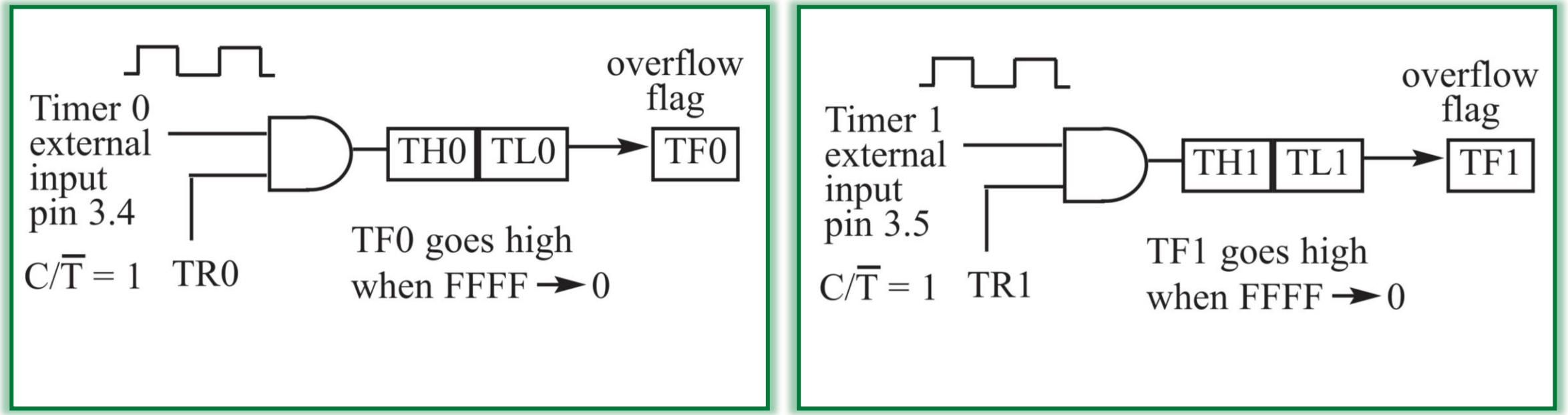


Figure 9-5



(a) Timer 0 with External Input (Mode 1)

(b) Timer 1 with External Input (Mode 1)

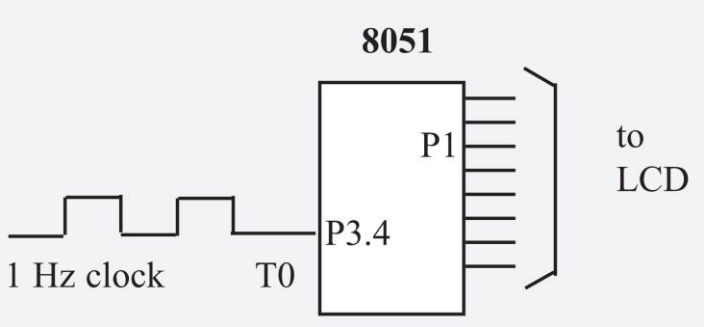
Example 9-19

Assume that a 1-Hz frequency pulse is connected to input pin 3.4. Write a program to display counter 0 on an LCD. Set the initial value of TH0 to -60.

Solution:

```
        ACALL   LCD_SET_UP      ;initialize the LCD
        MOV     TMOD,#00000110B ;counter 0,mode 2,C/T=1
        MOV     TH0,#-60        ;counting 60 pulses
        SETB    P3.4            ;make T0 as input
AGAIN:  SETB    TR0             ;starts the counter
BACK:   MOV     A,TL0           ;get copy of count TL0
        ACALL   CONV            ;convert in R2, R3, R4
        ACALL   DISPLAY         ;display on LCD
        JNB     TF0,BACK        ;loop if TF0=0
        CLR     TR0             ;stop the counter 0
        CLR     TF0             ;make TF0=0
        SJMP    AGAIN           ;keep doing it
;converting 8-bit binary to ASCII
;upon return, R4, R3, R2 have ASCII data (R2 has LSD)
CONV:   MOV     B,#10           ;divide by 10
        DIV     AB
        MOV     R2,B            ;save low digit
        MOV     B,#10           ;divide by 10 once more
        DIV     AB
        ORL     A,#30H          ;make it ASCII
        MOV     R4,A            ;save MSD
        MOV     A,B
        ORL     A,#30H          ;make 2nd digit an ASCII
        MOV     R3,A            ;save it
        MOV     A,R2
        ORL     A,#30H          ;make 3rd digit an ASCII
        MOV     R2,A            ;save the ASCII
        RET
```

To display the TL count on an LCD, we must convert 8-bit binary data to ASCII. See Chapter 6 for data conversion.



By using 60 Hz we can generate seconds, minutes, hours.

Note that on the first round, it starts from 0, since on RESET, TL0 = 0.
To solve this problem, load TL0 with -60 at the beginning of the program.

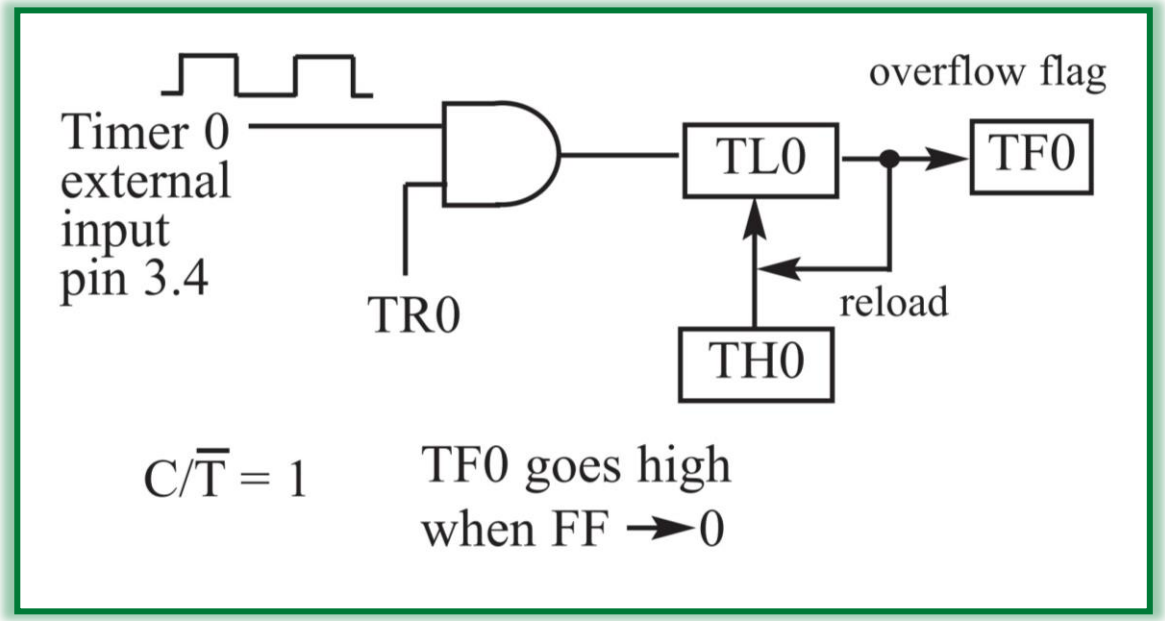


Figure 9-6. Timer 0 with External Input (Mode 2)

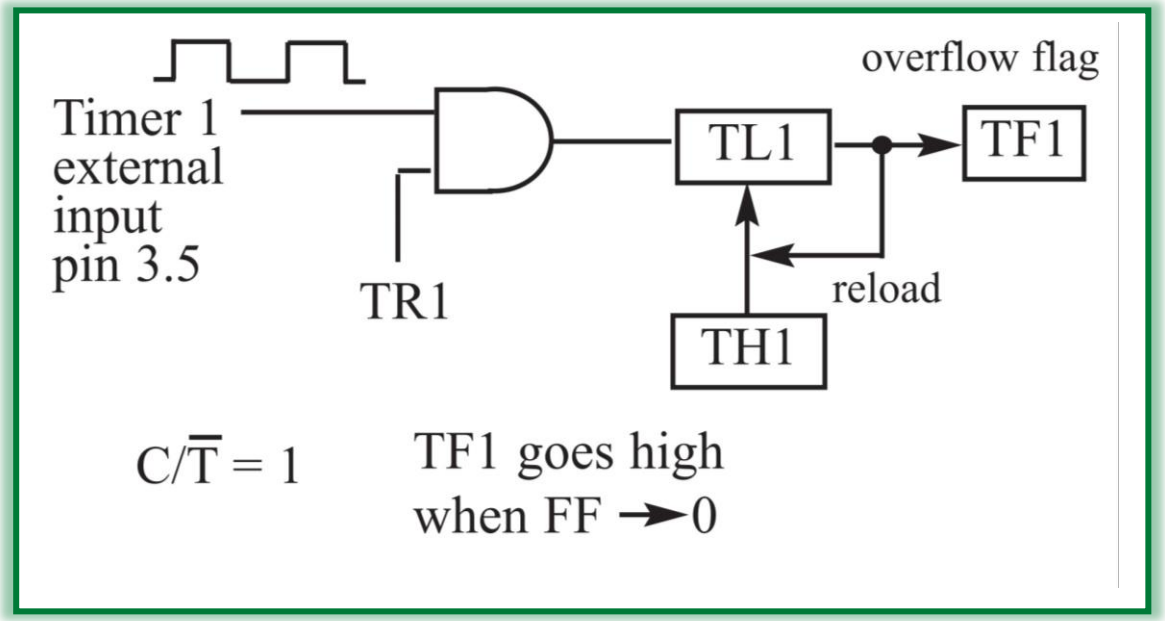


Figure 9-7. Timer 1 with External Input (Mode 2)

Table 9-2: Equivalent Instructions for the Timer Control Register (TCON)

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

TCON: Timer/Counter Control Register

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Figure 9-8. Timer/Counter 0

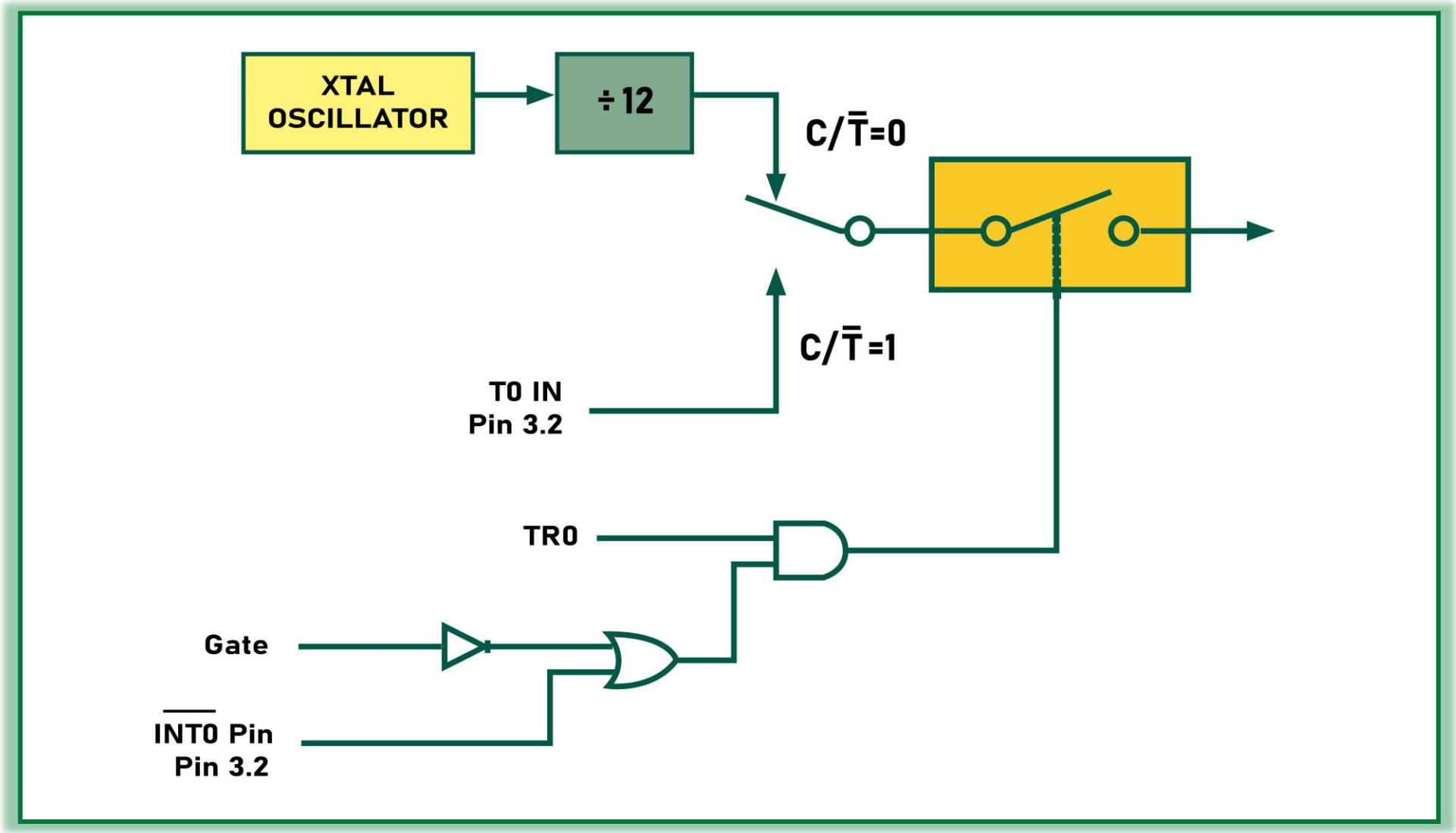


Figure 9-9. Timer/Counter 1