

# TIMER PROGRAMMING

---

*The 8051 Microcontroller and Embedded Systems: Using Assembly and C*  
Mazidi, Mazidi and McKinlay

Chung-Ping Young  
楊中平

*Home Automation, Networking, and Entertainment Lab*

Dept. of Computer Science and Information Engineering  
National Cheng Kung University, TAIWAN



## PROGRAMMING TIMERS

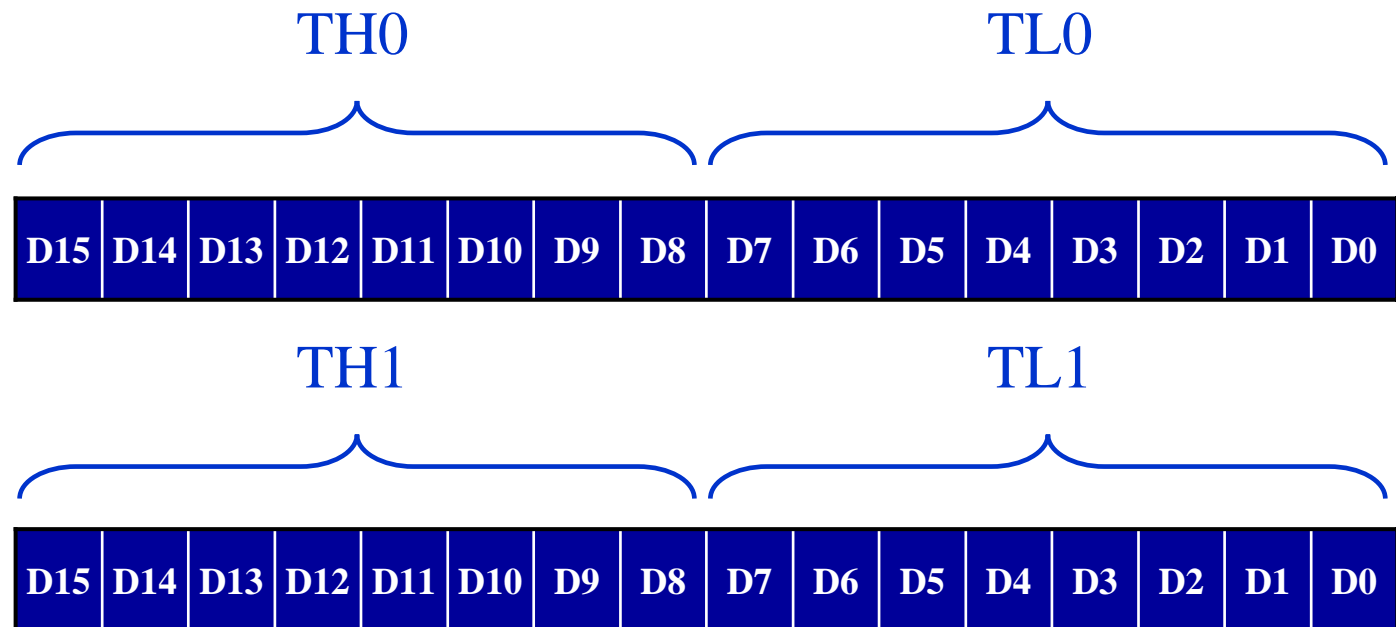
- ❑ The 8051 has two timers/counters, they can be used either as
  - Timers to generate a time delay or as
  - Event counters to count events happening outside the microcontroller
- ❑ Both Timer 0 and Timer 1 are 16 bits wide
  - Since 8051 has an 8-bit architecture, each 16-bits timer is accessed as two separate registers of low byte and high byte



## PROGRAMMING TIMERS

### Timer 0 & 1 Registers

- ❑ Accessed as low byte and high byte
  - The low byte register is called TL0/TL1 and
  - The high byte register is called TH0/TH1
  - Accessed like any other register
    - `MOV TL0, #4FH`
    - `MOV R5, TH0`



## PROGRAMMING TIMERS

### TMOD Register

- ❑ Both timers 0 and 1 use the same register, called TMOD (timer mode), to set the various timer operation modes
- ❑ TMOD is a 8-bit register
  - The lower 4 bits are for Timer 0
  - 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

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

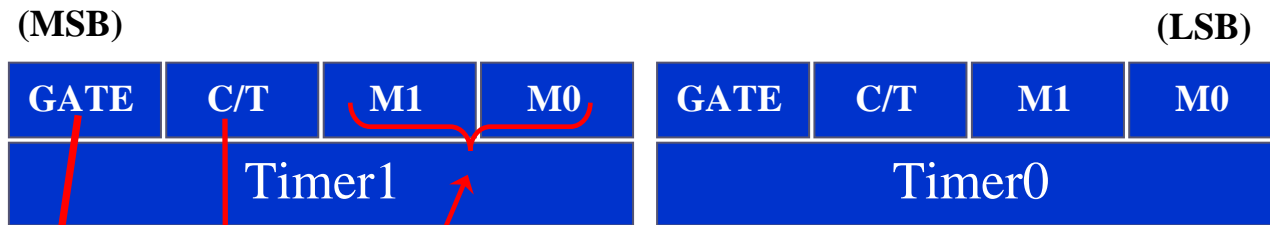


# PROGRAMMING TIMERS

## TMOD Register (cont')

**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



M1	M0	Mode	Operating Mode
0	0	0	<b>13-bit timer mode</b> 8-bit timer/counter THx with TLx as 5-bit prescaler
0	1	1	<b>16-bit timer mode</b> 16-bit timer/counter THx and TLx are cascaded; there is no prescaler
1	0	2	<b>8-bit auto reload</b> 8-bit auto reload timer/counter; THx holds a value which is to be reloaded TLx each time it overflows
1	1	3	<b>Split timer mode</b>

### Timer or counter selected

Cleared for timer operation (input from internal system clock)

Set for counter operation (input from Tx input pin)



HANEL

# PROGRAMMING TIMERS

## TMOD Register (cont')

If C/T = 0, it is used as a timer for time delay generation. The clock source for the time delay is the crystal frequency of the 8051

### 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 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.

### Example 9-2

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:



$$1/12 \times 11.0592 \text{ MHz} = 921.6 \text{ kHz};$$

$$T = 1/921.6 \text{ kHz} = 1.085 \text{ us}$$



## PROGRAMMING TIMERS

TMOD  
Register

GATE

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

- ❑ 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

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.

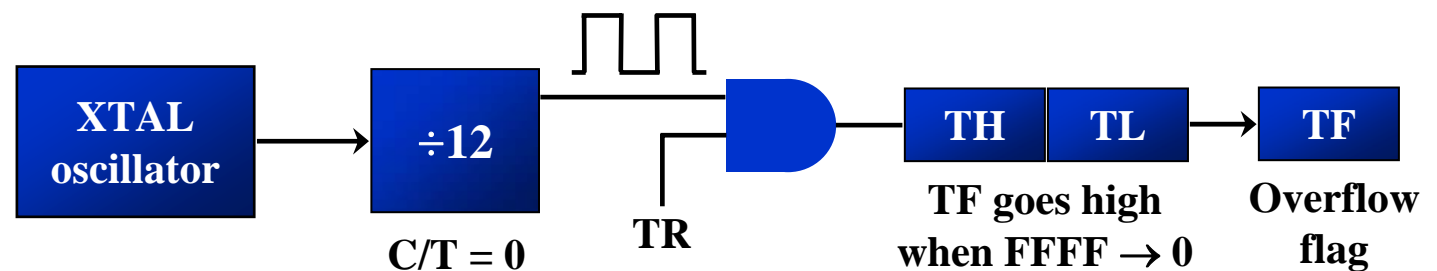
TMOD = 0000 0010



# PROGRAMMING TIMERS

## Mode 1 Programming

- ❑ The following are the characteristics and operations of mode1:
  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





# PROGRAMMING TIMERS

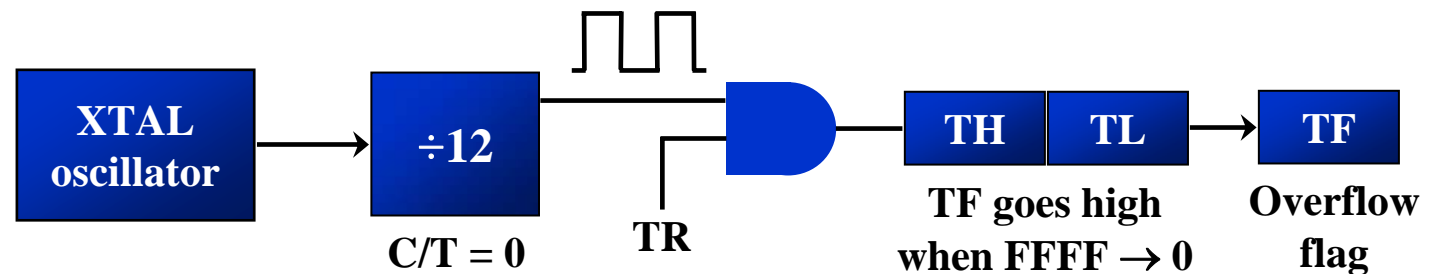
## Mode 1 Programming (cont')

### 3. (cont')

- 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



## PROGRAMMING TIMERS

### Mode 1 Programming

#### Steps to Mode 1 Program

- ❑ To generate a time delay
  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



# PROGRAMMING TIMERS

## Mode 1 Programming

### Steps to Mode 1 Program (cont')

#### Example 9-4

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
```

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.

...



# PROGRAMMING TIMERS

## Mode 1 Programming

### Steps to Mode 1 Program (cont')

#### Example 9-4 (cont')

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
```

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 ...



# PROGRAMMING TIMERS

## Mode 1 Programming

### Steps to Mode 1 Program (cont')

#### Example 9-5

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

#### Solution:

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

The number of counts for the roll over is  $\text{FFFFH} - \text{FFF2H} = 0\text{D}\text{H}$  (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  $14 \times 1.085 \mu\text{s} = 15.19 \mu\text{s}$  for half the pulse. For the entire period it is  $T = 2 \times 15.19 \mu\text{s} = 30.38 \mu\text{s}$  as the time delay generated by the timer.

#### (a) in hex

$(\text{FFFF} - \text{YYXX} + 1) \times 1.085 \mu\text{s}$ , where YYXX are TH, TL initial values respectively. Notice that value YYXX are in hex.

#### (b) in decimal

Convert YYXX values of the TH, TL register to decimal to get a NNNNN decimal, then  $(65536 - \text{NNNN}) \times 1.085 \mu\text{s}$



## PROGRAMMING TIMERS

### Mode 1 Programming

#### Steps to Mode 1 Program (cont')

#### Example 9-6

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

#### Solution:

In the timer delay calculation of Example 9-5, 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	14
CLR TR0	1
CLR TF0	1
RET	2
<b>Total</b>	<b>28</b>

$$T = 2 \times 28 \times 1.085 \text{ us} = 60.76 \text{ us and } F = 16458.2 \text{ Hz}$$



# PROGRAMMING TIMERS

## Mode 1 Programming

### Steps to Mode 1 Program (cont')

#### 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 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          ;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  $18370 \times 1.085 \text{ us} = 19.93145 \text{ ms}$

(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  $18370 \times 1.085 \text{ us} = 19.93145 \text{ ms}$  as the width of the pulse.



# PROGRAMMING TIMERS

## Mode 1 Programming

### Steps to Mode 1 Program (cont')

#### 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, 16-bitmode
HERE:  MOV    TL0,#0    ;TL0=0, the low byte
        MOV    TH0,#0    ;TH0=0, 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
        CLR    P2.3
```

Making TH and TL both zero means that the timer will count from 0000 to FFFF, and then roll over to raise the TF flag. As a result, it goes through a total Of 65536 states. Therefore, we have delay =  $(65536 - 0) \times 1.085 \text{ us} = 71.1065\text{ms}$ .





# PROGRAMMING TIMERS

## Mode 1 Programming

### Steps to Mode 1 Program (cont')

#### Example 9-9

The following program generates a square wave on 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,#10;Timer 1, mod 1 (16-bitmode)
AGAIN: MOV      TL1,#34H;TL1=34H, low byte of timer
                MOV      TH1,#76H;TH1=76H, high byte timer
                SETB    TR1        ;start the timer 1
BACK:  JNB      TF1,BACK ;till timer rolls over
                CLR     TR1        ;stop the timer 1
                CPL     P1.5       ;comp. p1. to get hi, lo
                CLR     TF1        ;clear timer flag 1
                SJMP    AGAIN      ;is not auto-reload
```

#### Solution:

Since  $FFFFH - 7634H = 89CBH + 1 = 89CCH$  and  $89CCH = 35276$  clock count and  $35276 \times 1.085 \text{ us} = 38.274 \text{ ms}$  for half of the square wave. The frequency = 13.064Hz.

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



## PROGRAMMING TIMERS

### Mode 1 Programming

#### Finding the Loaded Timer Values

- ❑ To calculate the values to be loaded into the TL and TH registers, look at the following example
  - 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 1.085 us
    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
    4. Set TL = xx and TH = yy



# PROGRAMMING TIMERS

## Mode 1 Programming

### Finding the Loaded Timer Values (cont')

#### Example 9-10

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  $5 \text{ ms} / 1.085 \text{ us} = 4608$  clocks. To Achieve that we need to load into TL and TH the value  $65536 - 4608 = \text{EE00H}$ . 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
```



# PROGRAMMING TIMERS

## Mode 1 Programming

### Finding the Loaded Timer Values (cont')

#### Example 9-11

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)  $T = 1 / f = 1 / 2 \text{ kHz} = 500 \text{ us}$  the period of square wave.
- (b)  $1 / 2$  of it for the high and low portion of the pulse is 250 us.
- (c)  $250 \text{ us} / 1.085 \text{ us} = 230$  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
```



# PROGRAMMING TIMERS

## Mode 1 Programming

### Finding the Loaded Timer Values (cont')

#### Example 9-12

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)  $10 \text{ ms} / 1.085 \text{ us} = 9216$  and  $65536 - 9216 = 56320$  in decimal, and in hex it is DC00H.
- (d) TL = 00 and TH = DC (hex).

```
                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 hi, lo
                SJMP   AGAIN         ;Reload timer
                                   ;mode 1 isn't auto-reload
```



# PROGRAMMING TIMERS

## Mode 1 Programming

### Generating Large Time Delay

#### 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, mod 1
        MOV    R3,#200      ;cnter for multiple delay
AGAIN:  MOV    TL1,#08H      ;TL1=08,low byte of timer
        MOV    TH1,#01H      ;TH1=01,high byte
        SETB   TR1           ;Start timer 1
BACK:   JNB    TF1,BACK      ;until timer rolls over
        CLR    TR1           ;Stop the 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  
 $65272 \times 1.085 \mu\text{s} = 70.820 \text{ ms}$ , and for 200 of them we have  
 $200 \times 70.820 \text{ ms} = 14.164024 \text{ seconds}$ .



## PROGRAMMING TIMERS

### Mode 2 Programming

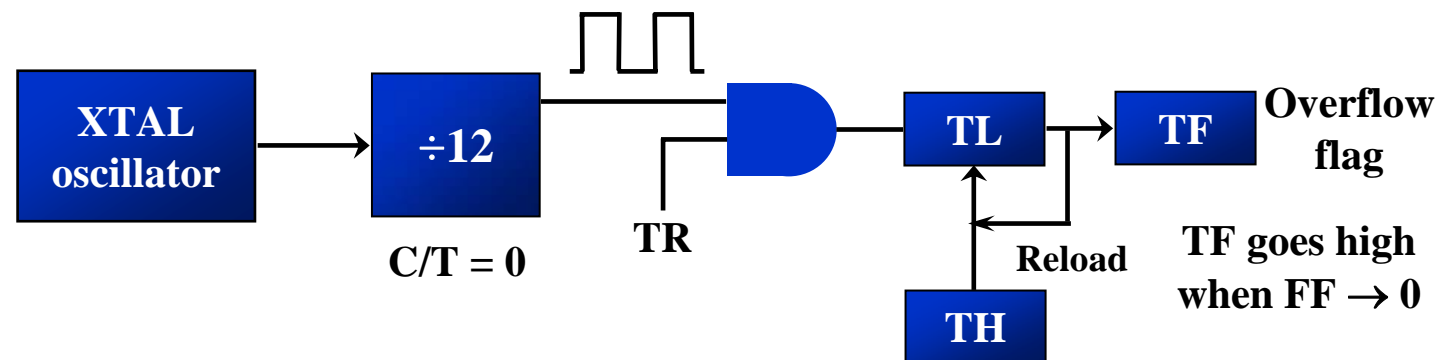
- ❑ The following are the characteristics and operations of mode 2:
  1. It is an 8-bit timer; therefore, it allows only values of 00 to FFH to be loaded into the timer's register TH
  2. After TH is loaded with the 8-bit value, the 8051 gives a copy of it to TL
    - Then the timer must be started
    - 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)



## PROGRAMMING TIMERS

### Mode 2 Programming (cont')

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





## PROGRAMMING TIMERS

### Mode 2 Programming

#### Steps to Mode 2 Program

- ❑ 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
  6. Go back to Step4, since mode 2 is auto-reload



# PROGRAMMING TIMERS

## Mode 2 Programming

### Steps to Mode 2 Program (cont')

#### Example 9-14

Assume XTAL = 11.0592 MHz, find the frequency of the square wave generated on pin P1.0 in the following 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 hi, lo
        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  $(256 - 05) \times 1.085 \text{ us} = 251 \times 1.085 \text{ us} = 272.33 \text{ us}$  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  $T = 2 \times 272.33 \text{ us} = 544.67 \text{ us}$  and the frequency = 1.83597 kHz



# PROGRAMMING TIMERS

## Mode 2 Programming

### Steps to Mode 2 Program (cont')

#### Example 9-15

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

#### Solution:

```
                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
```

$T = 2 ( 250 \times 256 \times 1.085 \text{ us} ) = 138.88\text{ms}$ , and frequency = 72 Hz



# PROGRAMMING TIMERS

## Mode 2 Programming

### Steps to Mode 2 Program (cont')

#### 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 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	E4H
-48	D0H
-60	C4H
-200	38H

The number 200 is the timer count till the TF is set to 1

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

