TIMERS in Embedded System

Dr. Qamaruddin

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`

TH0								TL0							
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

TH1								TL1							
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

- 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')

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

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)

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:

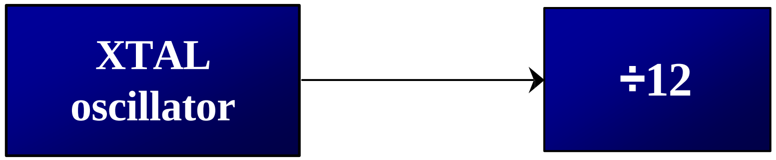
We convert the value from hex to binary. From Figure 9-3 we have:

(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:



$1/12 \times 11.0529$ MHz = 921.6 MHz;
T = 1/921.6 kHz = 1.085 us

PROGRAMMING TIMERS

TMOD Register

GATE

- 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

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

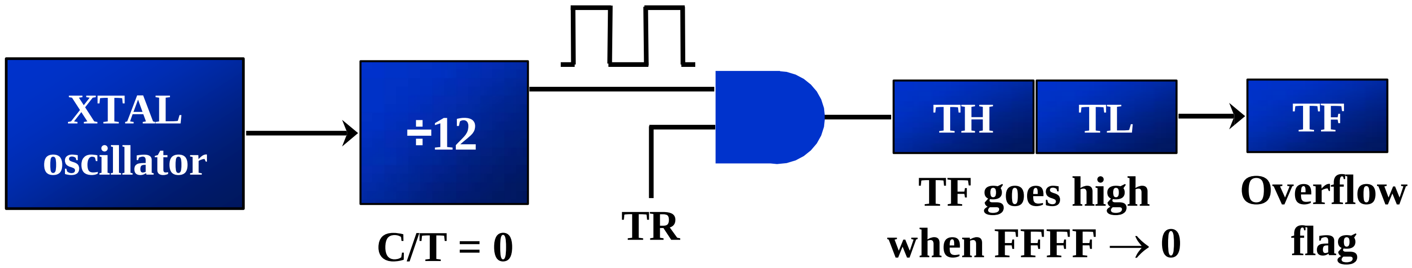
Find the value for TMOD if we want to program timer 0 in mode 2, use 8051 XTAL for the clock source, and use instructions to start and stop the timer.

TMOD = 0000 0010

PROGRAMMING TIMERS

Mode 1 Programming

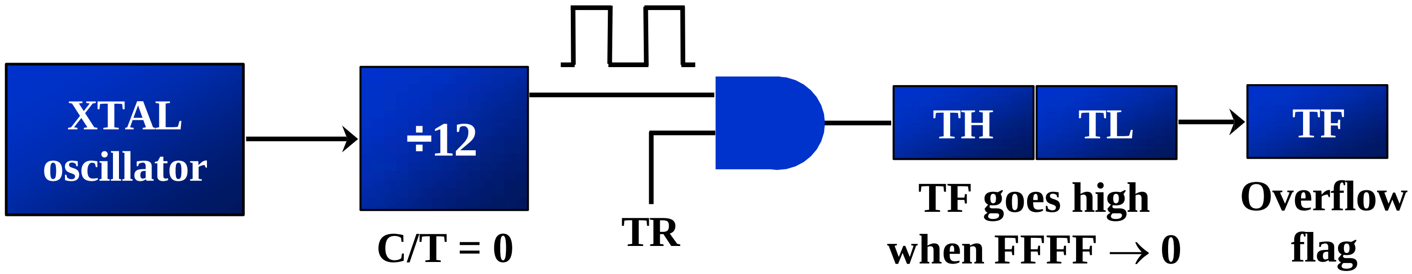
- The following are the characteristics and operations of mode1:
  1. It is a 16-bit timer; therefore, it allows value of 0000 to FFFFH to be loaded into the timer's register TL and TH
  2. After TH and TL are loaded with a 16-bit initial value, the timer must be started
     - This is done by `SETB 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
3. 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.

...

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 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 × 1.085us = 15.19us for half the pulse. For the entire period it is T = 2 × 15.19us = 30.38us as the time delay generated by the timer.

(a) in hex
(FFFF – YYXX + 1) × 1.085 us, 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 - NNNN) × 1.085 us

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
                                   Total   28
```

$T = 2 \times 28 \times 1.085$ us = 60.76 us and F = 16458.2 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 × 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.

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 = $(65536 - 0) \times 1.085$ us = 71.1065ms.

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 × 1.085 us = 38.274 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 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
```

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 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
```

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 ms / 1.085 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 $65272 \times 1.085\ \mu s = 70.820$ ms, and for 200 of them we have 200×70.820 ms = 14.164024 seconds.

PROGRAMMING TIMERS

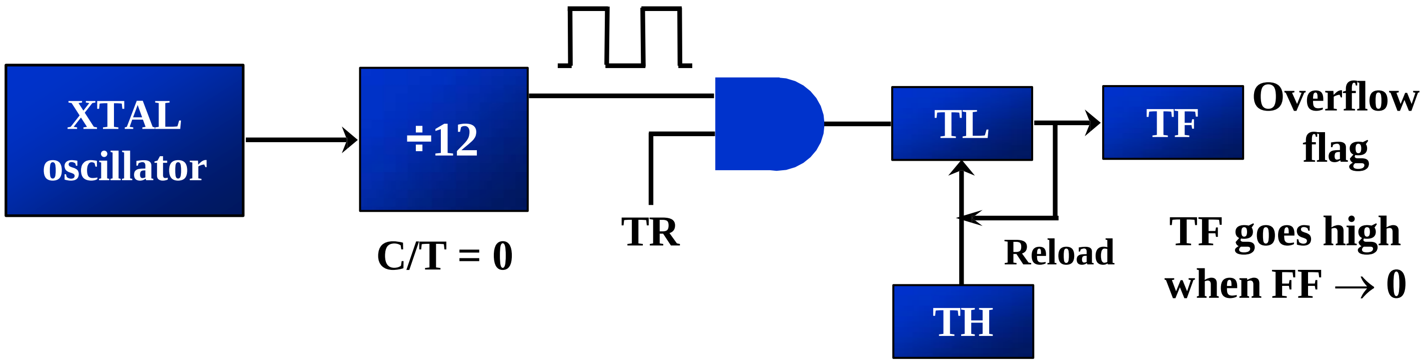
Mode 2 Programming

- 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 us = 251 $\times$ 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 $\times$ 272.33 us = 544.67 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 × 256 × 1.085 us) = 138.88ms, 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 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 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

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 registers
  - 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

COUNTER PROGRAMMING

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

COUNTER PROGRAMMING

C/T Bit in TMOD Register (cont')

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, which connects to 8 LEDs.

Solution:

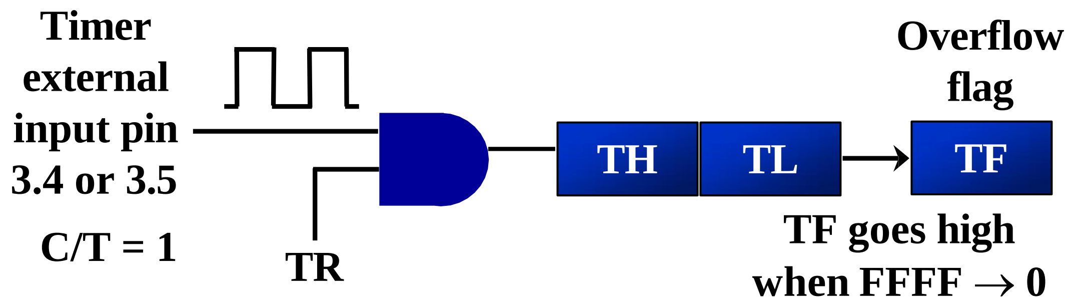
```
        MOV     TMOD,#01100000B ;counter 1, mode 2,
                        ;C/T=1 external pulses
        MOV     TH1,#0  ;clear TH1
        SETB    P3.5    ;make T1 input
AGAIN:  SETB    TR1     ;start the counter
BACK:   MOV     A,TL1   ;get copy of TL
        MOV     P2,A    ;display it on port 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
```

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.

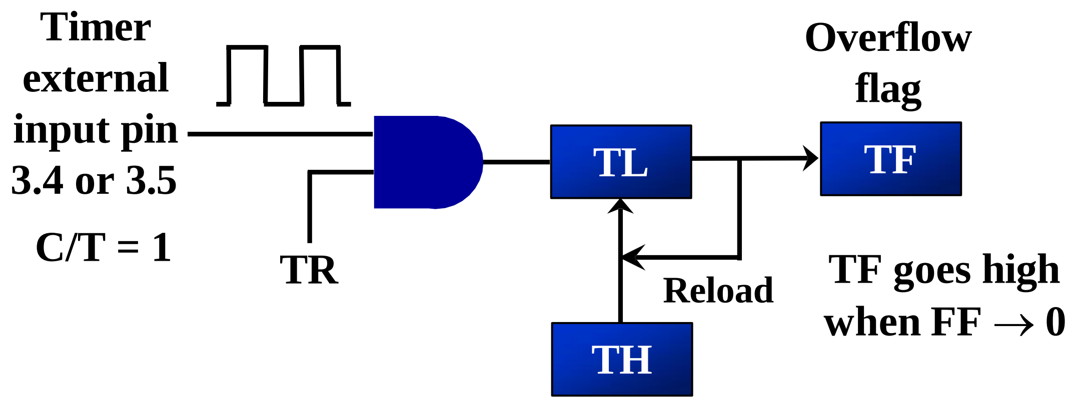
COUNTER PROGRAMMING

C/T Bit in TMOD Register (cont')

Timer with external input (Mode 1)



Timer with external input (Mode 2)



COUNTER PROGRAMMING

TCON Register

- TCON (timer control) register is an 8-bit 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

COUNTER PROGRAMMING

TCON Register (cont')

- TCON register is a bit-addressable register

Equivalent instruction for the Timer Control Register

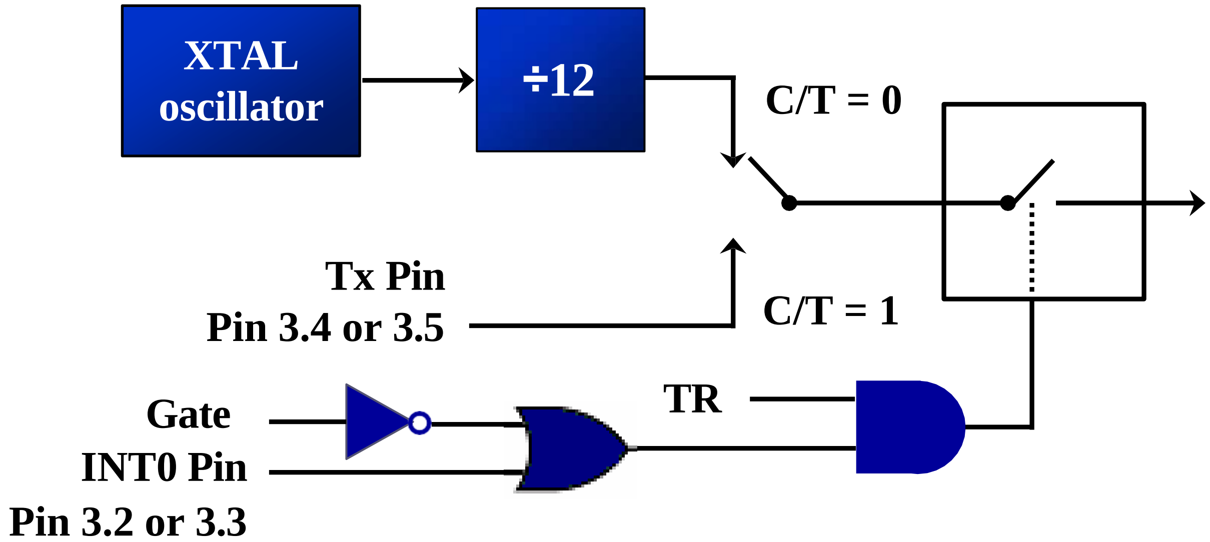
For timer 0			
SETB TR0	=	SETB TCON.4	
CLR TR0	=	CLR TCON.4	
SETB TF0	=	SETB TCON.5	
CLR TF0	=	CLR TCON.5	
For timer 1			
SETB TR1	=	SETB TCON.6	
CLR TR1	=	CLR TCON.6	
SETB TF1	=	SETB TCON.7	
CLR TF1	=	CLR TCON.7	

COUNTER PROGRAMMING

TCON Register

Case of GATE = 1

- 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 to start or stop the timer externally at any time via a simple switch



PROGRAMMING TIMERS IN C

Accessing Timer Registers

Example 9-20

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;
}
```

FFFFH – 3500H = CAFFH

= 51967 + 1 = 51968

51968 × 1.085 μs = 56.384 ms is the approximate delay

PROGRAMMING TIMERS IN C

Calculating Delay Length Using Timers

- To speed up the 8051, many recent versions of the 8051 have reduced the number of clocks per machine cycle from 12 to four, or even one
- The frequency for the timer is always 1/12th the frequency of the crystal attached to the 8051, regardless of the 8051 version

PROGRAMMING TIMERS IN C

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

Example 9-21

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. Test the program on the (a) AT89C51 and (b) DS89C420.

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;
}
```

FFFFH – 4BFDH = B402H

= 46082 + 1 = 46083

$46083 \times 1.085\ \mu s = 50\ ms$

PROGRAMMING TIMERS IN C

Times 0/1 Delay Using Mode 1 (16-bit Non Auto-reload) (cont’)

Example 9-22

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:

```
//tested for DS89C420, XTAL = 11.0592 MHz
#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;
}
```

A5FEH = 42494 in decimal

65536 – 42494 = 23042

$23042 \times 1.085\ \mu s = 25$ ms and

20×25 ms = 500 ms

PROGRAMMING TIMERS IN C

Times 0/1 Delay Using Mode 1 (16-bit Non Auto-reload)
(cont')

Example 9-25

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:

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

.....
```

PROGRAMMING TIMERS IN C

Times 0/1 Delay Using Mode 1 (16-bit Non Auto-reload) (cont')

Example 9-25

```
.....

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;
}
```

FC67H = 64615

65536 – 64615 = 921

$921 \times 1.085\ \mu s = 999.285\ \mu s$

$1 / (999.285\ \mu s \times 2) = 500\ Hz$

PROGRAMMING TIMERS IN C

Times 0/1 Delay Using Mode 2 (8-bit Auto-reload)

Example 9-23

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:

```
#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;
}
```

Due to overhead of the for loop in C, we put 36 instead of 40

256 – 23 = 233

23 × 1.085 µs = 25 µs and

25 µs × 250 × 40 = 250 ms

PROGRAMMING TIMERS IN C

Times 0/1 Delay Using Mode 2 (8-bit Auto-reload) (cont')

Example 9-24

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:

```
#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;
}
```

1/2500 Hz = 400 μs

400 μs /2 = 200 μs

200 μs / 1.085 μs = 184