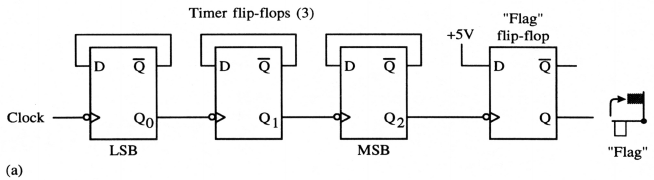
8051
timer/counter

Timers /Counters Programming

- The 8051 has 2 timers/counters: timer/counter 0 and timer/counter 1. They can be used as

1. The **timer** is used as a time delay generator.
   - The clock source is the **internal** crystal frequency of the 8051.
2. An event **counter**.
   - **External input** from input pin to count the number of events on registers.
   - These clock pulses cold represent the number of people passing through an entrance, or the number of wheel rotations, or any other event that can be converted to pulses.



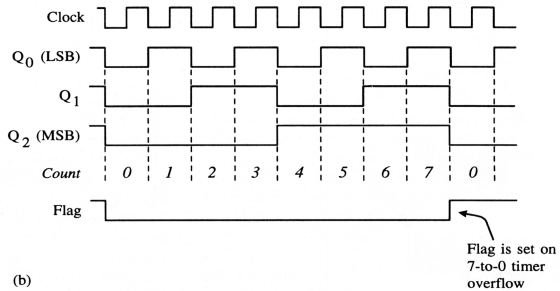
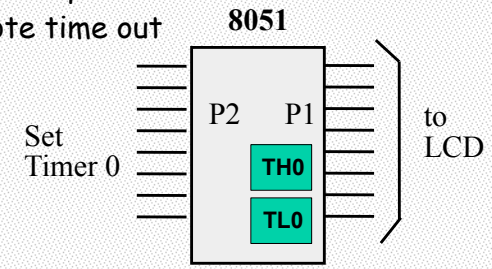


FIGURE 4–1
A 3-bit timer (a) Schematic (b) Timing diagram

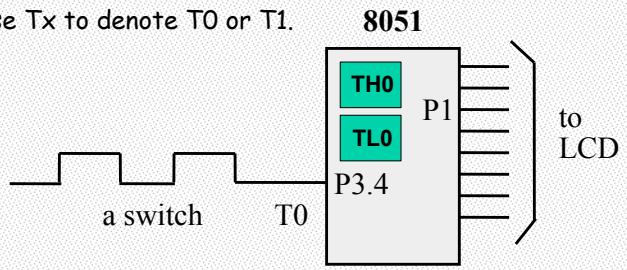
Timer

- Set the initial value of registers
- Start the timer and then the 8051 counts up.
- Input from internal system clock (machine cycle)
- When the registers equal to 0 and the 8051 sets a bit to denote time out



Counter

- Count the number of events
  - Show the number of events on registers
  - External input from T0 input pin (P3.4) for Counter 0
  - External input from T1 input pin (P3.5) for Counter 1
  - **External input** from Tx input pin.
  - We use Tx to denote T0 or T1.



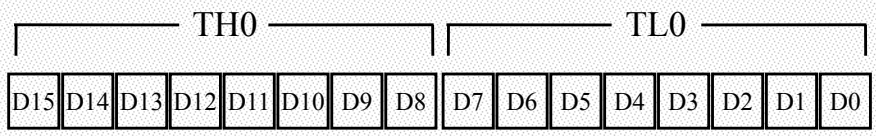
Registers Used in Timer/Counter

- TH0, TL0, TH1, TL1
- TMOD (Timer mode register)
- TCON (Timer control register)
- You can see Appendix H (pages 413-415) for details.
- Since 8052 has 3 timers/counters, the formats of these control registers are different.
  - T2CON (Timer 2 control register), TH2 and TL2

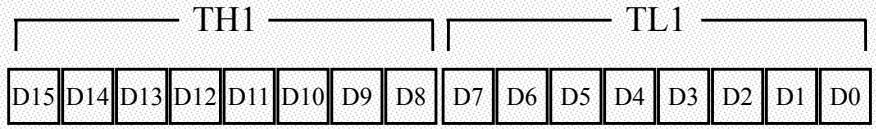
Basic Registers of the Timer

- Both timer 0 and timer 1 are 16 bits wide.
  - These registers stores
    - the time delay as a timer
    - the number of events as a counter
  - Timer 0: **TH0** & **TL0**
    - Timer 0 high byte, timer 0 low byte
  - Timer 1: **TH1** & **TL1**
    - Timer 1 high byte, timer 1 low byte
  - Each 16-bit timer can be accessed as two separate registers of low byte and high byte.

Timer Registers



Timer 0



Timer 1

TMOD Register

- Timer mode register: **TMOD**

  `MOV TMOD,#21H`

  - An 8-bit register
  - Set the usage mode for two timers
    - Set lower 4 bits for Timer 0 (Set to 0000 if not used)
    - Set upper 4 bits for Timer 1 (Set to 0000 if not used)
    - Not bit-addressable

(MSB) (LSB)

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

Figure 9-3. TMOD Register

GATE Gating control when set. 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

(MSB) (LSB)

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

C/T (Clock/Timer)

- This bit is used to decide whether the timer is used as a delay generator or an event counter.
- C/T = 0 : timer
- C/T = 1 : counter

Gate

- Every timer has a mean of starting and stopping.
  - GATE=0
    - **Internal** control
    - The start and stop of the timer are controlled by way of **software**.
    - Set/clear the TR for start/stop timer.
  - GATE=1
    - **External** control
    - The hardware way of starting and stopping the timer by **software** and **an external source**.
    - Timer/counter is enabled only while the INT pin is high and the TR control pin is set (TR).

M1, M0

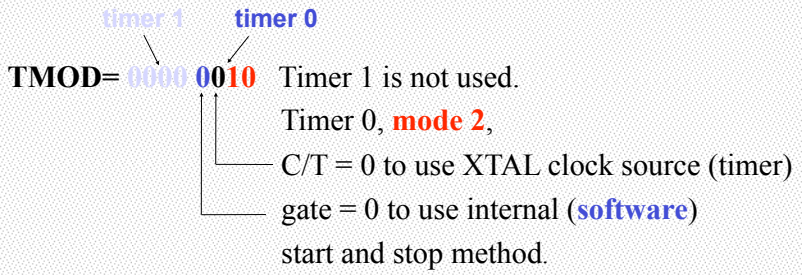
- M0 and M1 select the timer mode for timers 0 & 1.

M1	M0	Mode	Operating Mode
0	0	0	**13-bit timer** mode 8-bit THx + 5-bit TLx (x= 0 or 1)
0	1	1	**16-bit timer** mode 8-bit THx + 8-bit TLx
1	0	2	**8-bit auto reload** 8-bit auto reload timer/counter; THx holds a value which is to be reloaded into TLx each time it overflows.

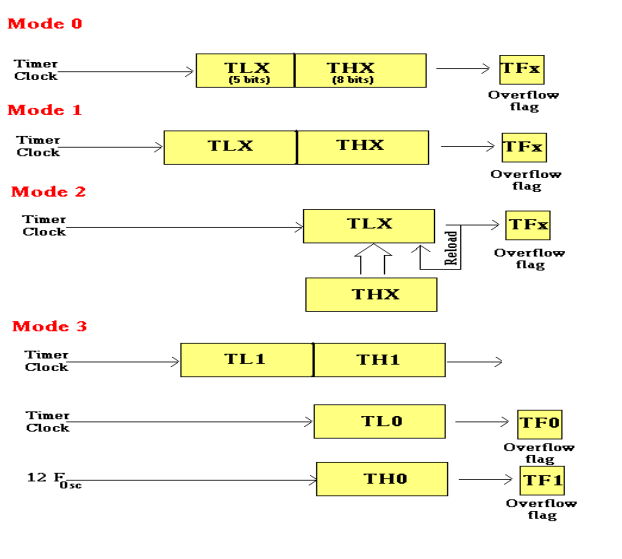
Example 9-3

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:



Timer modes


Mode 0
Timer Clock
TLX (5 bits)
THX (8 bits)
TFx
Overflow flag
Mode 1
Timer Clock
TLX
THX
TFx
Overflow flag
Mode 2
Timer Clock
TLX
TFx
Overflow flag
Reload
THX
Mode 3
Timer Clock
TL1
TH1
Timer Clock
TL0
TF0
Overflow flag
12 F Osc
TH0
TF1
Overflow flag

TCON Register (1/2)

- Timer control register: **TMOD**
  - Upper nibble for timer/counter, lower nibble for interrupts
- **TR** (run control bit)
  - TR0 for Timer/counter 0; TR1 for Timer/counter 1.
  - TR is set by programmer to turn timer/counter on/off.
    - TR=0: off (stop)
    - TR=1: on (start)

(MSB) (LSB)

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
Timer 1		Timer0		for Interrupt			

TCON Register (2/2)

- **TF** (timer flag, control flag)
  - TF0 for timer/counter 0; TF1 for timer/counter 1.
  - TF is like a carry. Originally, TF=0. When TH-TL roll over to 0000 from FFFFH, the TF is set to 1.
    - TF=0 : not reach
    - TF=1: reach
    - If we enable interrupt, TF=1 will trigger ISR.

(MSB) (LSB)

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
Timer 1		Timer0		for Interrupt			

Equivalent Instructions 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**

TCON: Timer/Counter Control Register

TF1	**TR1**	**TF0**	**TR0**	**IE1**	**IT1**	**IE0**	**IT0**

Timer Mode 1

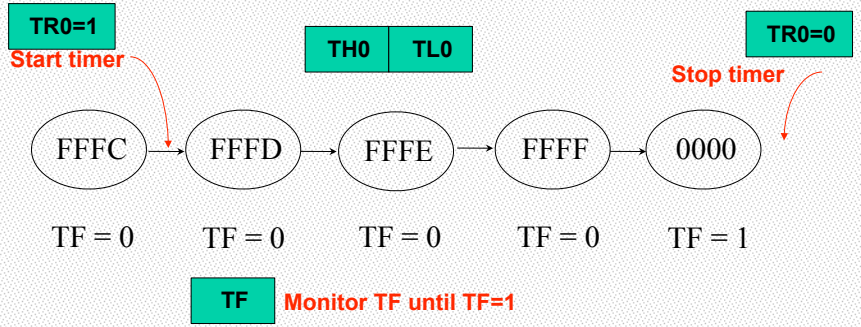
- In following, we all use timer 0 as an example.
- **16-bit** timer (TH0 and TL0)
- TH0-TL0 is incremented continuously when TR0 is set to 1. And the 8051 stops to increment TH0-TL0 when TR0 is cleared.
- The timer works with the internal system clock. In other words, the timer counts up each machine cycle.
- When the timer (TH0-TL0) reaches its maximum of FFFFH, it rolls over to 0000, and TF0 is raised.
- Programmer should check TF0 and stop the timer 0.

Steps of Mode 1 (1/3)

1. Choose mode 1 timer 0
   - `MOV TMOD,#01H`
2. Set the original value to TH0 and TL0.
   - `MOV TH0,#FFH`
   - `MOV TL0,#FCH`
3. You had better to clear the flag to monitor: TF0=0.
   - `CLR TF0`
4. Start the timer.
   - `SETB TR0`

Steps of Mode 1 (2/3)

5. The 8051 starts to count up by incrementing the TH0-TL0.
   - **TH0-TL0= FFFCH,FFFDH,FFFEH,FFFFH,0000H**



Steps of Mode 1 (3/3)

6. When TH0-TL0 rolls over from FFFFH to 0000, the 8051 set TF0=1.

```
TH0-TL0= FFFEH, FFFFH, 0000H (Now TF0=1)
```

7. Keep monitoring the timer flag (TF) to see if it is raised.

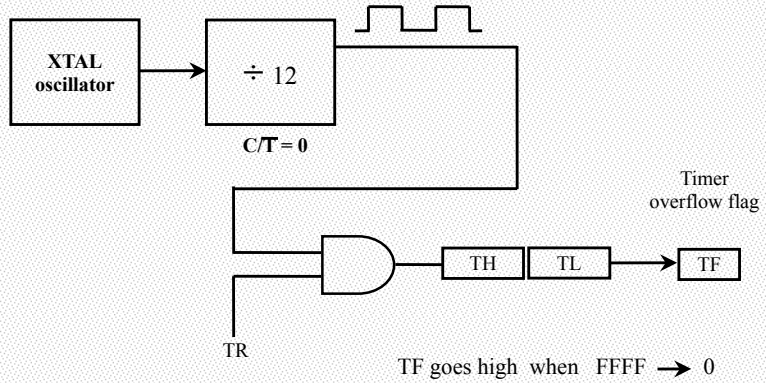
```
AGAIN:  JNB TF0, AGAIN
```

8. Clear TR0 to stop the process.

```
CLR TR0
```

9. Clear the TF flag for the next round.

Mode 1 Programming



Timer Delay Calculation for XTAL = 11.0592 MHz

(a) in hex

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

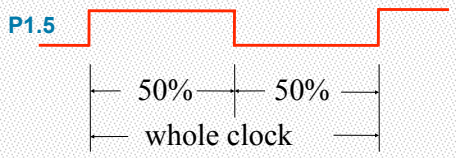
(b) in decimal

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

Example 9-4 (1/3)

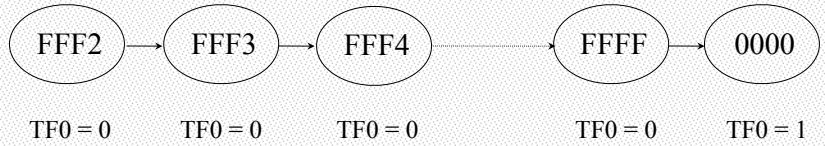
- square wave of 50% duty on P1.5
- Timer 0 is used

```
;each loop is a half clock
       MOV TMOD,#01    ;Timer 0,mode 1(16-bit)
HERE:  MOV TL0,#0F2H   ;Timer value = FFF2H
       MOV TH0,#0FFH
       CPL P1.5
       ACALL DELAY
       SJMP HERE
```



Example 9-4 (2/3)

```
;generate delay using timer 0
DELAY:
       SETB TR0          ;start the timer 0
AGAIN:JNB TF0,AGAIN
       CLR TR0           ;stop timer 0
       CLR TF0           ;clear timer 0 flag
       RET
```



TF0 = 0 TF0 = 0 TF0 = 0 TF0 = 0 TF0 = 1

Example 9-4 (3/3)

Solution:

In the above program notice the following steps.

1. TMOD = **0000 0001** 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, FFFC, FFFFD, FFFE, 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 timer again (in the main program).

Example 9-9 (1/2)

- This program generates a square wave on pin P1.5 Using timer 1
- Find the frequency.(dont include the overhead of instruction delay)
- XTAL = 11.0592 MHz

```
        MOV   TMOD,#10H    ;timer 1, mode 1
AGAIN:  MOV   TL1,#34H     ;timer value=3476H
        MOV   TH1,#76H
        SETB  TR1          ;start
BACK:   JNB   TF1,BACK
        CLR   TR1          ;stop
        CPL   P1.5         ;next half clock
        CLR   TF1          ;clear timer flag 1
        SJMP  AGAIN        ;reload timer1
```

Example 9-9 (2/2)

Solution:

FFFFH – 7634H + 1 = 89CCH = 35276 clock count

Half period = 35276 × 1.085 μs = 38.274 ms

Whole period = 2 × 38.274 ms = 76.548 ms

Frequency = 1/ 76.548 ms = 13.064 Hz.

Note

Mode 1 is not auto reload then the program must reload the TH1, TL1 register every timer overflow if we want to have a continuous wave.

Find Timer Values

- Assume that XTAL = 11.0592 MHz .
- And we know desired **delay**
- how to find the values for the TH,TL ?
  1. Divide the delay by 1.085 μs and get n.
  2. Perform 65536 –n
  3. Convert the result of Step 2 to hex (yyxx)
  4. Set TH = yy and TL = xx.

Example 9-12 (1/2)

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

Solution:

1. The period of the square wave = 1 / 50 Hz = 20 ms.
2. The high or low portion of the square wave = 10 ms.
3. 10 ms / 1.085 μs = 9216

Example 9-12 (2/2)

```
        MOV   TMOD,#10H     ;timer 1, mode 1
AGAIN:  MOV   TL1,#00       ;Timer value = DC00H
        MOV   TH1,#0DCH
        SETB  TR1           ;start
BACK:   JNB   TF1,BACK
        CLR   TR1           ;stop
        CPL   P2.3
        CLR   TF1           ;clear timer flag 1
        SJMP  AGAIN         ;reload timer since
                            ;mode 1 is not
                            ;auto-reload
```

Generate a Large Time Delay

- The size of the time delay depends on two factors:
  - They crystal frequency
  - The timer's 16-bit register, TH & TL
- The largest time delay is achieved by making TH=TL=0.
- What if that is not enough?
- Next Example show how to achieve 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
        MOV   R3,#200
AGAIN:  MOV   TL1,#08
        MOV   TH1,#01
        SETB  TR1
BACK:   JNB   TF1,BACK
        CLR   TR1
        CLR   TF1
        DJNZ  R3,AGAIN
```

Solution:

TH – TL = 0108H = 264 in decimal

65536 – 264 = 65272.

One of the timer delay = 65272 × 1.085 μs = 70.820 ms

Total delay = 200 × 70.820 ms = 14.164024 seconds

Timer Mode 0

- Mode 0 is exactly like mode 1 except that it is a 13-bit timer instead of 16-bit.
  - 8-bit TH0
  - 5-bit TL0
- The counter can hold values between 0000 to 1FFF in TH0-TL0.
  - 2^{13}-1= 2000H-1=1FFFH
- We set the initial values TH0-TL0 to count up.
- When the timer reaches its maximum of 1FFFH, it rolls over to 0000, and TF0 is raised.

Timer Mode 2

- 8-bit timer.
  - It allows only values of 00 to FFH to be loaded into TH0.
- Auto-reloading
- TL0 is incremented continuously when TR0=1.
- next example: 200 MCs delay on timer 0.
- See Examples 9-14 to 9-16

Steps of Mode 2 (1/2)

1. Chose mode 2 timer 0

```
MOV TMOD,#02H
```

2. Set the original value to TH0.

```
MOV TH0,#38H
```

3. Clear the flag to TF0=0.

```
CLR TF0
```

4. After TH0 is loaded with the 8-bit value, the 8051 gives a copy of it to TL0.

```
TL0=TH0=38H
```

5. Start the timer.

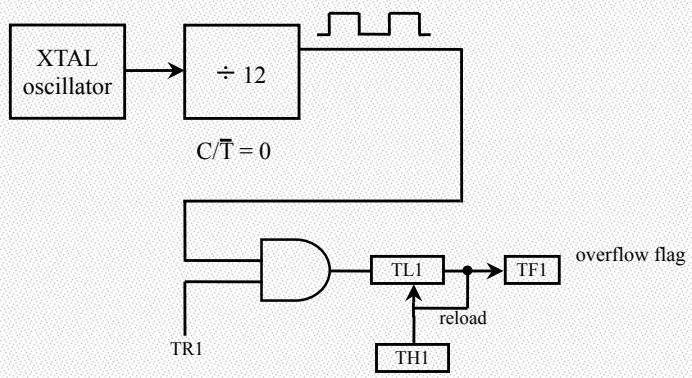
```
SETB TR0
```

Steps of Mode 2 (2/2)

6. The 8051 starts to count up by incrementing the TL0.
   - `TL0= 38H, 39H, 3AH,....`
7. When TL0 rolls over from FFH to 00, the 8051 set TF0=1. Also, TL0 is reloaded automatically with the value kept by the TH0.
   - `TL0= FEH, FFH, 00H (Now TF0=1)`
   - The 8051 auto reload `TL0=TH0=38H.`
   - `Clr TF0`
   - Go to Step 6 (i.e., TL0 is incrementing continuously).

- Note that **we must clear TF0** when TL0 rolls over. Thus, we can monitor TF0 in next process.
- Clear TR0 to stop the process.

Timer 1 Mode 2 with internal Input



TF goes high when FF → 0

Example 9-15

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

Solution:

```
        MOV  TMOD,#2H   ;Timer 0,mode 2
        MOV  TH0,#0
AGAIN:  MOV  R5,#250    ;count 250 times
        ACALL DELAY
        CPL  P1.0
        SJMP AGAIN
```

```
DELAY:  SETB TR0        ;start
BACK:   JNB  TF0,BACK   ;wait until TL0 ovrflw auto-reload
        CLR  TR0        ;stop
        CLR  TF0        ;clear TF
        DJNZ R5,DELAY
        RET
```

T = 2 (250 × 256 × 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:

Some 8051 assemblers provide this way.

-200 = -C8H ⇒ 2's complement of -200 = 100H - C8H = 38 H

Decimal	2's complement (TH value)
-200 = - C8H	38H
- 60 = - 3CH	C4H
- 3	FDH
- 12	F4H
- 48	D0H

Example 9-17 (1/2)

Find

(a) the frequency of the square wave generated in the following code

(b) the duty cycle of this wave.

Solution:

"`MOV TH0,#-150`" uses 150 clocks.

The DELAY subroutine = 150 × 1.085 μs = 162 μs.

The high portion is twice tat of the low portion (66% duty cycle).

The total period = high portion + low portion

T= 325.5 μs + 162.25 μs = 488.25 μs

Frequency = 2.048 kHz.

Example 9-17 (2/2)

```
      MOV  TMOD,#2H   ;Timer 0,mode 2
      MOV  TH0,#-150   ;Count=150
AGAIN:SETB P1.3        } high
      ACALL DELAY      } period
      ACALL DELAY      }
      CLR  P1.3        } low
      ACALL DEALY      } period
      SJMP AGAIN

DELAY:SETB TR0         ;start
BACK: JNB  TF0,BACK
      CLR  TR0         ;stop
      CLR  TF0         ;clear TF
      RET
```