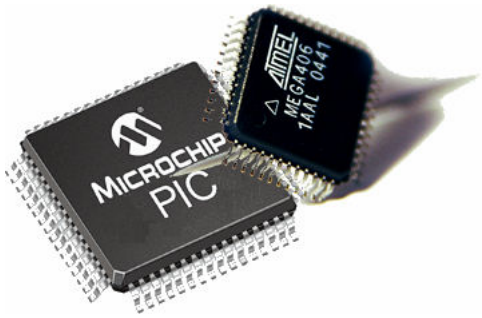
Timer/Counters



The 8051 has two internal sixteen bit hardware Timer/Counters. Each Timer/Counter can be configured in various modes, typically based on 8-bit or 16-bit operation. The 8052 product has an additional (third) Timer/Counter.

Figure 5.1 provides us with a brief refresher on what a hardware counter looks like. This is a circuit for a simple 3-bit counter which counts from 0 to 7 and then overflows, setting the overflow flag. A 3-bit counter would not be very useful in a microcomputer so it is more typical to find 8-bit and 16-bit counter circuits.

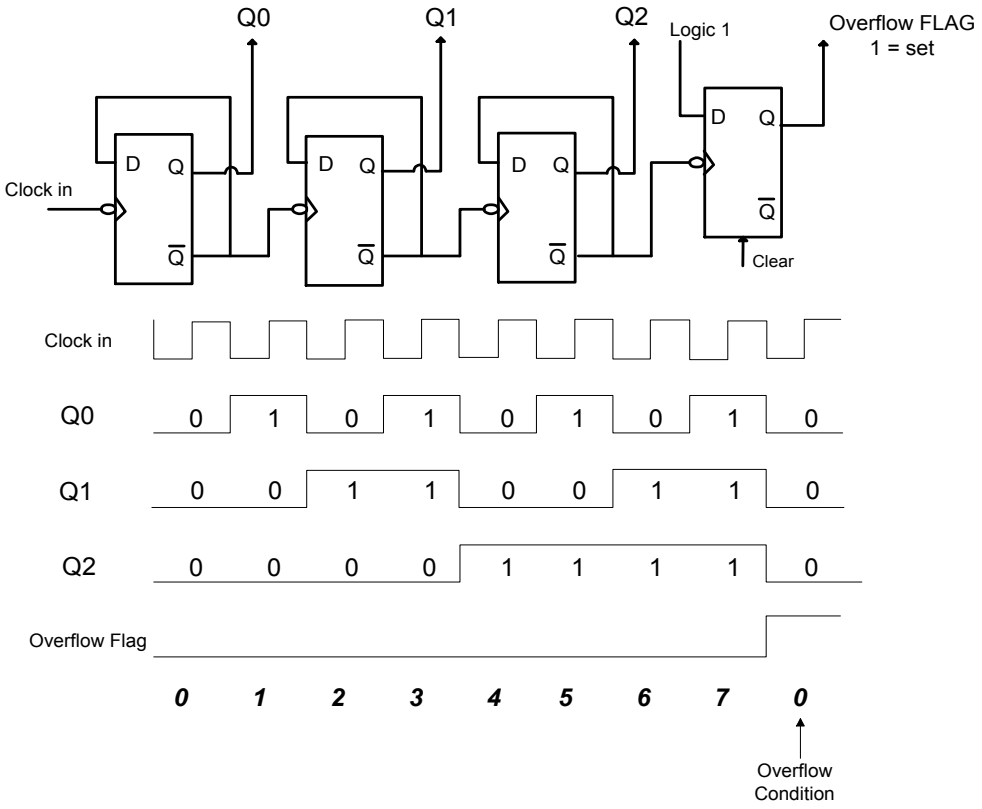


Figure 5.1 3-bit counter circuit

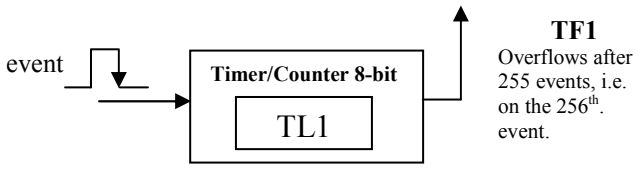
5.1 8-bit COUNTER OPERATION

First let us consider a simple 8-bit counter. Since this is a modulo-8 set up we are concerned with 256 numbers in the range 0 to 255 (2^8 =256). The counter will count in a continuous sequence as follows:

Hex	**Binary**	**Decimal**
00h	00000000	0
01h	00000001	1
02h	00000010	2
.	.	.
.	.	.
FEh	11111110	254

FFh	11111111	255	
00h	00000000	0	⟶ here the counter **overflows** to zero1
01h	00000001	1	
etc.			
etc.			

We will use **Timer/Counter 1** in our examples below.



Supposing we were to initialise this Timer/Counter with a number, say 252, then the counter would overflow after just four event pulses, i.e.:

FCh	11111100	252	counter is initialised at 252
FDh	11111101	253	
FEh	11111110	254	
FFh	11111111	255	
00h	00000000	0	⟶ here the counter **overflows**

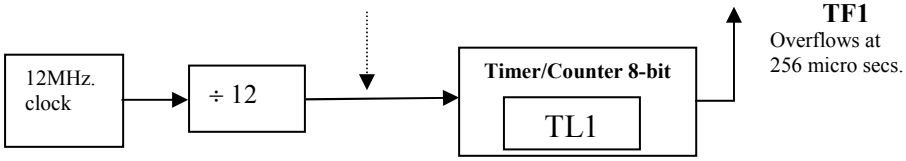
An 8-bit counter can count 255 events before overflow, and overflows on the 256th. event. When initialised with a predefined value of say 252 it overflows after counting just four events. Thus the number of events to be counted can be programmed by pre-loading the counter with a given number value.

5.2 8-bit TIMER OPERATION

The 8051 internally divides the processor clock by 12. If a 12 MHz. processor clock is used then a 1 MHz. instruction rate clock, or a pulse once every microsecond, is realised internally within the chip. If this 1 microsecond pulse is connected to a Timer/Counter input, in place of an *event* input, then the Timer/Counter becomes a timer which can delay by up to 255 microseconds. There is a clear difference between a timer and a counter. The counter will count events, up to 255 events before overflow, and the timer will count time pulses, thus creating delays up to 255 microseconds in our example.

To be precise we would refer to the *counter* as an *event counter* and we would refer to the *timer* as an *interval timer*.

1 MHz. i.e. pulse
every 1 micro. Sec.



If the timer is initialised to zero it will count 256 microseconds before overflow. If the timer is initialised to a value of 252, for example, it will count just 4 microseconds before overflow. Thus this timer is programmable between 1 microsecond and 256 microseconds**.**

5.2.1 HOW DO WE PROGRAM THE 8-BIT TIMER/COUNTER?

Let's look at how to do the following:

o Configure the Timer/Counter as a TIMER or as a COUNTER
o Program the Timer/Counter with a value between 0 and 255
o Enable and disable the Timer/Counter
o How to know when the timer has overflowed – interrupt vs. polling.

The TMOD register (**T**imer **Mod**e Control) is an SFR register at location 89h in internal RAM and is used to define the Timer/Counter mode of operation.

TMOD register

Gate *msb*	**C/T**	**M1**	**M0**	**Gate**	**C/T**	**M1**	**M0** *Lsb*

--------- timer 1 ----------|----------timer 0 ----------

Consider Timer/Counter 1 only. The Gate bit will be ignored for now and will be set to 0 in the examples. The C/T bit is set to 1 for COUNTER operation and it is set to 0 for TIMER operation. MI and M2 bits define different modes, where mode 2 is the 8 bit mode, i.e.:

M1	**M0**	
0	0	mode 0: 13 bit mode (seldom used).
0	1	mode 1: 16-bit mode
1	0	mode 2: 8-bit mode (with auto reload feature)
1	1	mode 3: ignore for now

To run in TIMER mode using 8-bit operation, the TMOD register is initialised as follows:

```
MOV   TMOD, #00100000b    ; assume timer 0 is not considered
```

Program the Timer/Counter value
The 8-bit Timer/Counter is pre-programmed with a value in the range 0..255. This is achieved by writing this value into the TH1 register for the Timer/Counter. TH1 is an SFR register (located at 8Dh in Internal RAM). An example is as follows:

```
MOV TH1, #129d      ; Timer/Counter 1 is programmed for 129 counts
```

How to know when the timer has overflowed?
The TCON register (**T**imer **Con**trol) has some bits which represent Timer/Counter status flags as well as some bits which can be set or cleared to control the Timer/Counter operation. The relevant bits for Timer/Counter 1 are bolded in the diagram. TR1 is set to 1 to enable Timer/Counter 1. Clearing TR1 turns the Timer/Counter off. TF1 is the Timer/Counter overflow flag. When the Timer/Counter overflows TF1 goes to a logic 1. Under interrupt operation TF1 is automatically cleared by hardware when the processor vectors to the associated ISR routine.

TCON register

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

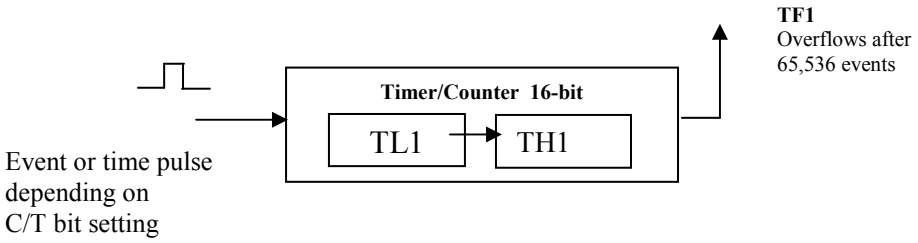
Auto reloading of the 8-bit Timer/Counter
The TL1 SFR register (located at 8Bh in Internal RAM) represents the current value in the 8-bit Timer/Counter. The Timer/Counter can be programmed by initialising this register with a number between 0 and 255. However, there is an interesting automatic reload feature in *mode 2*, where, when TL1 overflows (its value reaches 0), the Timer/Counter is automatically reloaded with the 8-bit value stored in SFR register TH1 (The pre-programmed value in TH1 does not change during this operation).

5.3 THE 16 BIT TIMER/CONTER

When the Timer/Counter is configured for *mode 1* operation it operates in 16 bit mode. Since this is a modulo-16 set up we are concerned with 65,536 numbers in the range 0 to 65,535 (2^{16} = 65,536). Consider a 16 bit Timer/Counter as shown below, which will count in the sequence as follows:

Hex	**Binary**	**Decimal**
0000h	0000000000000000	0
0001h	0000000000000001	1

0010h	0000000000000010	2		
.	.	.		
.	.	.		
FFFEh	1111111111111110	65,534		
FFFFh	1111111111111111	65,535		
00000h	0000000000000000	0	→	here it **overflows** to zero.



Now we have a 16-bit Timer/Counter and we can preload it with a sixteen bit number so as to cause a delay from bewteen 1 to 65,535 microseconds (65.535 millisecs.), or in counter mode it can count between 1 and 65,535 events. To preload the Timer/Counter value simply write the most significant byte into the TH1 register and the least significant byte into the TL1 register. The 16-bit counter is not automatically reloaded following an overflow and such reloading must be explicitly programmed. We will see this in some examples below.

Interrupt vs. polling operation
When a Timer/Counter overflow occurs it can be arranged to automatically cause an interrupt. Alternatively the Timer/Counter interrupt can be disabled and the software can test the TF1 (Timer 1 flag) bit in the TCON register to check that the overflow has occurred.

It is also possible to read the Timer/Conter value (TH1, TL1) so as to assertain the current value. (Beware, care must be exercised in reading the 16 bit values in this manner as the low byte might overflow into the high byte between the successive read operations).

5.4 EXAMPLE PROGRAMS

Here we will look at some short example programs to illustrate the following:

o TIMER1.A51 8-bit TIMER, polled overflow flag. See listing 5.1.
o TIMER2.A51 16-bit TIMER, polled overflow flag. See listing 5.2.
o TIMER3.A51 16-bit TIMER, interrupt driven. See listing 5.3.
o TIMER4.A51 16 bit COUNTER, interrupt driven. See listing 5.4.

TIMER1.A51 program example
The TIMER1.A51 program preloads the timer with value of *minus 250 decimal*. This means that the TIMER will have to count 250 microseconds to reach overflow. Port 1 bit 7, P1.7, is complemented every time the TIMER overflows hence a square wave with a period of 500 microseconds (2 x 250) is produced at P1.7, realising a frequency of 2kHz. The software simply polls the overflow bit, TF1, and acts on TF1 being set. Note, the TIMER is automatically reloaded (with –250d) each time it overflows so there is no need to explicitly reload the TIMER initialisation value. Note the line where the TIMER value is initialised, as follows:

MOV TH1, #-250d

Since (256-250) = 6 , this could have been written as:

MOV TH1, #6d

Figure 5.3 (a) shows a programmer's view of the 8-bit Timer/Counter showing how the Timer/Counter is accessed via the SFR registers.

TIMER2.A51 program example
The TIMER2.A51 program uses Timer/Counter 1 in 16 bit mode (mode 1). This example program delays for 10 milliseconds. By complementing P1.7 at every TIMER overflow a square wave with a period of 20 milliseconds, or a 50Hz. square wave, is achieved. Note how the 16-bit timer is not automatically reloaded and this must be explicitly done within the program. Since the timer is stopped for short periods during the program operation this will introduce some inaccuracies into the timing loop. Figure 5.3 (b) shows a programmer's view of the 16-bit Timer/Counter showing how the Timer/Counter is accessed via the SFR registers.

Note the Timer/Counter in the standard 8052 product (Timer/Counter 2) does have additional features such as 16-bit mode auto reload facility.

TIMER3.A51 program example
The TIMER3.A51 program shows a 16-bit TIMER operation where the overflow flag TF1 causes an interrupt. Like the TIMER2.A51 program, this program generates a 50Hz. Square wave; but because it is an interrupt driven program it does not need to use valuable processing time for polling purposes. The loop where it sits 'doing nothing' could be used for more productive processing, doing other tasks. Every time the TIMER overflows the interrupt can be serviced and then the program can return to the more productive work.

TIMER4.A51 program example
The TIMER4.A51 program shows an example where the 16-bit Timer/Counter is used as an *event counter*. The counter counts 20,000 events and then sets P1.7 to a logic high. The program is interrupt driven so when TF1 is set the program vectors to location

001Bh (Timer/Counter 1). A real application example for this sort of program might be to count **n** devices pasing down a manufacturing assembly line and then to take some action once **n** devices have passed through. Figure 5.2 illustrates the example.

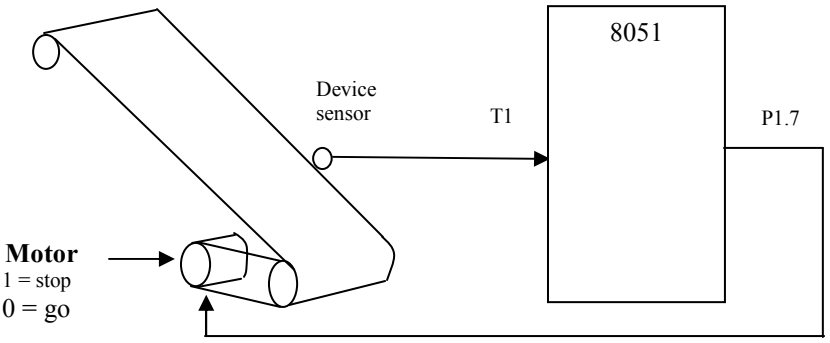
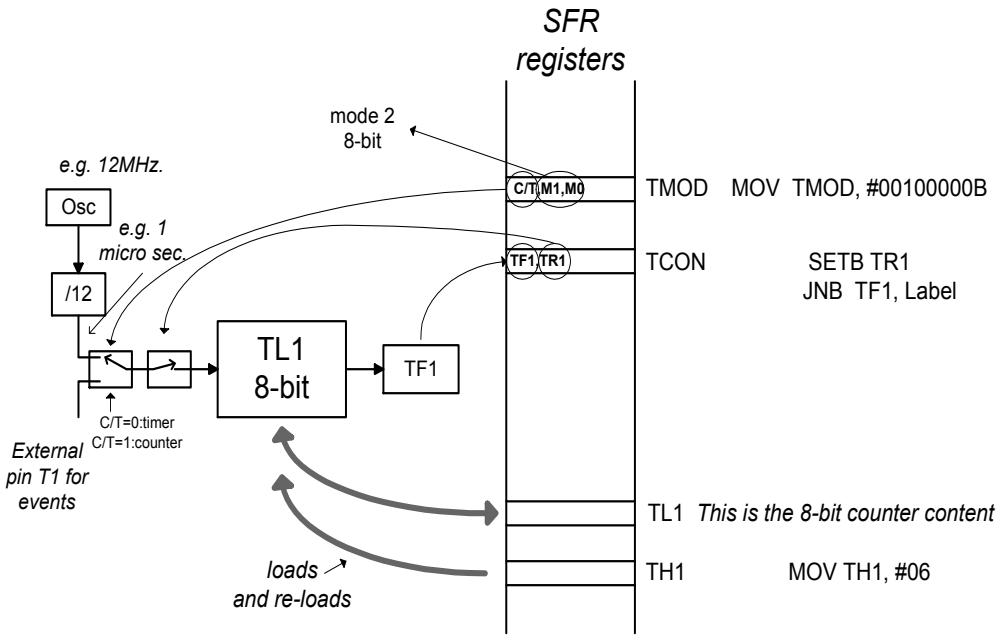
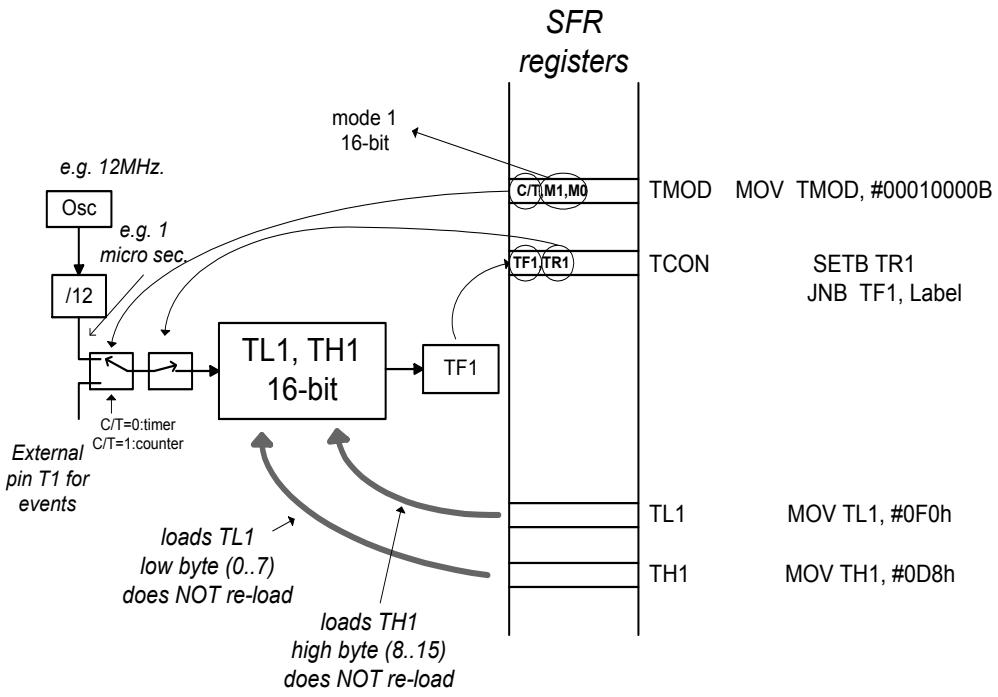


Figure 5.2 Counting items on a conveyor line.



a) Programmer's view of Timer/Counter 1, Mode 2, 8-bit



b) Programmers view of Timer/Counter 1, Mode 1, 16-bit

Figure 5.3 Programmer's view of Timer/Counter

```
;=============================================================
; TIMER1.A51
; Example program to generate a 2KHz. square wave at P1.7 using
; Timer/counter 1 in 8-bit TIMER mode. Polled , NOT interrupt driven.
;
;;=============================================================


        ORG 0000h       ; entry address for 8051 RESET
        LJMP MAIN       ; MAIN starts beyond interrupt vector space


;=============================================================
; MAIN initialises Timer/counter 1 and loops polling Timer overflow flag TF1
; and toggles Port 1 bit 7 each time Timer overflows (every 250 micro secs.)
;=============================================================
        ORG 0100h                       ; entry address for main
MAIN:

        MOV TH1, #-250d                 ; timer is initialised with -250 to count 250 usecs.
        MOV TMOD, #00100000b            ; timer 1 is set for mode 2, TIMER operation
        SETB TR1                        ; start Timer 1


LOOP:
        JNB TF1, LOOP                   ; loop around until Timer 1 overflows
        CLR TF1                         ; clear overflow flag
        CPL P1.7                        ; complement P1 bit 7
        LJMP LOOP                       ; jump back for polling

END
```

Listing 5.1 TIMER1.A51

```
;=============================================================
; TIMER2.A51
; Example program to generate a 50Hz. square wave at P1.7 using
; Timer/counter 1 in 16-bit mode. Polled, NOT interrupt driven.
;
;;=============================================================

        ORG 0000h       ; entry address for 8051 RESET
        LJMP MAIN       ; MAIN starts beyond interrupt vector space


;=============================================================
; MAIN initialises Timer 1 and loops polling Timer overflow flag TF1
; and toggles port 1 bit 7 each time Timer overflows (every 10 milli.secs.)
; 65536 - 10000 = 55536,or D8F0h
;=============================================================
        ORG 0100h                   ; entry address for main
MAIN:
        MOV TMOD, #00010000b        ; Timer 1 is set for mode 1, TIMER operation

LOOP:   MOV TH1, #0D8h              ; Timer 1 high byte is loaded
        MOV TL1, #0F0h              ; Timer 1 low byte is loaded
        SETB TR1                    ; start Timer 1


POLL:
        JNB TF1, POLL               ; loop around until Timer 1 overflows
        CLR TR1                     ; stop Timer 1
        CLR TF1                     ; clear overflow flag
        CPL P1.7                    ; complement P1 bit 7
        LJMP LOOP                   ; jump back for polling

END
```

Listing 5.2 TIMER2.A51

```
;===============================================================
; TIMER3.A51
; Example program to generate a 50Hz. square wave at P1.7 using
; Timer/counter 1 in 16-bit mode. INTERRUPT driven.
;
;;===============================================================

        ORG 0000h               ; entry address for 8051 RESET
        LJMP MAIN               ; MAIN starts beyond interrupt vector space

        ORG 001Bh               ; vector address for interrupt
        LJMP ISR_TIMER1         ; jump to start of ISR_TIMER1


;===================================================================
; MAIN initialises Timer 1 and enables Timer 1 interrupt, then
; it just waits around letting the interrupt routine ISR_TIMER1 do the work.
; The Timer 1 is loaded with a value (65536 - 10000 = 55536,
; or D8F0h) so that it interrupts every 10 milliseconds.
;===================================================================
        ORG 0100h                       ; entry address for main
MAIN:
        MOV TMOD, #00010000b            ; Timer 1 is set for mode 1, TIMER operation

        MOV TH1, #0D8h                  ; Timer 1 high byte is loaded
        MOV TL1, #0F0h                  ; Timer 1 low byte is loaded
        MOV IE, #10001000b              ; enable Timer 1 interrupt
        SETB TR1                        ; start Timer 1

 LOOP:  LJMP LOOP                       ; just loop around doing nothing



;===================================================================
; ISR_TIMER1
; In Timer 16 bit operation the Timer 1 must be reloaded each
; time it overflows. The overflow flag is cleared automatically.
;===================================================================
ISR_TIMER1:


        CLR TR1                         ; stop Timer 1
        MOV TH1, #0D8h                  ; reloads Timer 1 values in TH1
        MOV TL1, #0F0h                  ; and in TL1

        CPL P1.7                        ; complement P1 bit 7
        SETB TR1                        ; start Timer 1

        RETI                            ; return from interrupt


END
```

Listing 5.3 TIMER3.A51

```
;=============================================================
; TIMER4.A51
; Example program to count 20,000 events and to generate an interrupt
; following the 20,000 events, and then set bit P1.7 high. This
; example shows Timer/counter 1 being used as a COUNTER. The program is
; INTERRUPT driven.
;
;;=============================================================


        ORG 0000h           ; entry address for 8051 RESET
        LJMP MAIN           ; MAIN starts beyond interrupt vector space

        ORG 001Bh           ; vector address for interrupt
        LJMP ISR_TIMER1     ; jump to start of ISR_TIMER1

;=============================================================
; MAIN initialises Timer 1 as a COUNTER and enables Timer 1
; interrupt, then
; it just waits around letting the interrupt routine do the work.
; The Timer 1 is loaded with a value (65,536 - 20,000 = 45,536,
; or B1E0h) so that it interrupts after 20,000 events.
;=============================================================
        ORG 0100h                   ; entry address for main
MAIN:
        MOV TMOD, #01010000b        ; Timer 1 is set for mode 1, COUNTER operation

        MOV TH1, #0B1h              ; Timer 1 high byte is loaded
        MOV TL1, #0E0h              ; Timer 1 low byte is loaded
        MOV IE, #10001000b          ; enable Timer/counter 1 interrupt
        SETB TR1                    ; start Timer/counter 1

        LOOP:   LJMP LOOP           ; just loop around doing nothing



;=============================================================
; ISR_TIMER1
; P1.7 is set to logic 1 to flag that 20,000 counts have occurred
;=============================================================
ISR_TIMER1:


        CLR TR1             ; stop Timer 1 to be safe


        SETB P1.7           ; set high  P1 bit 7


        RETI                ; return from interrupt

END
```

Listing 5.4 TIMER4.A51