

Lecture SIX

Programmable Timer Interval (PTI)

REAL-TIME SYSTEMS

8254 PROGRAMMABLE INTERVAL TIMER

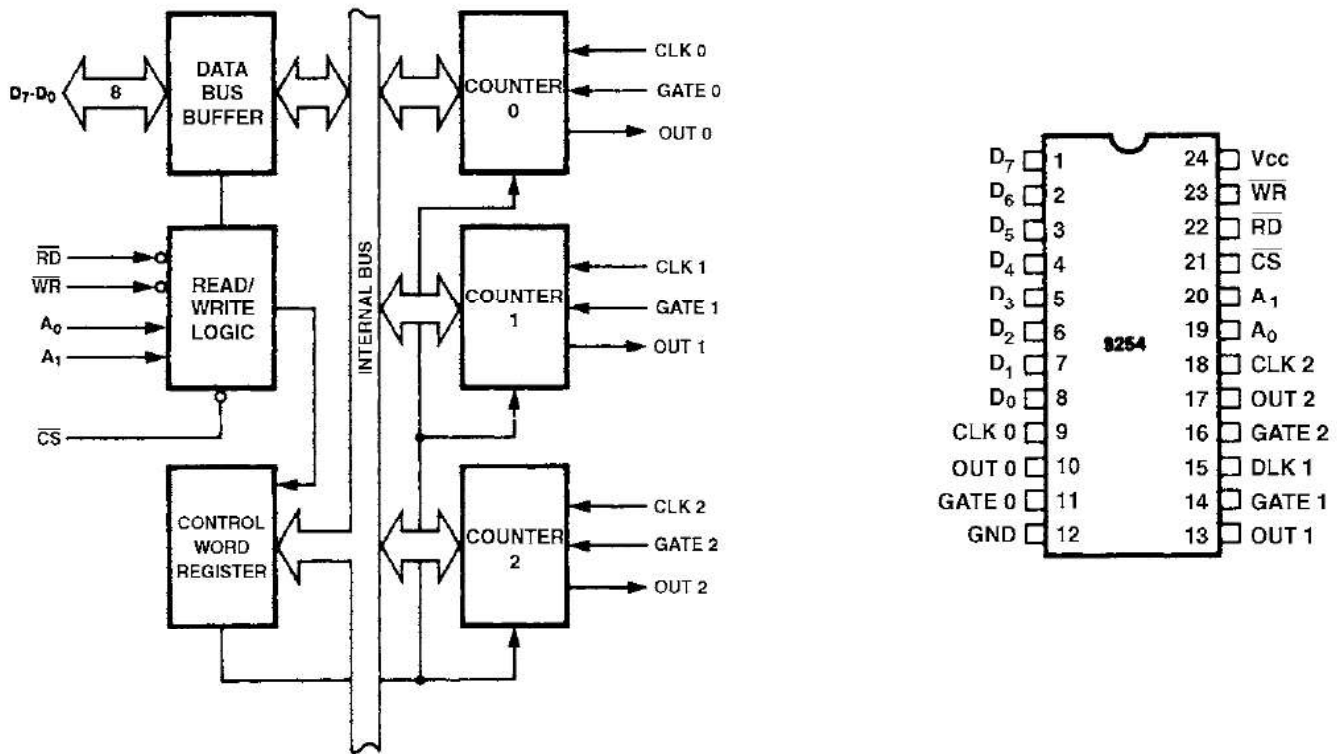
The 8254 programmable interval timer consists of three independent 16-bit programmable counters (**timers**). Each counter is capable of counting in binary or binary-coded decimal (BCD). The maximum allowable input frequency to any counter is 10 MHz. This device is useful wherever the microprocessor must control real-time events. Some examples of usage include real-time clock and an events counter, and for motor speed and direction control. This timer also appears in the personal computer decoded at ports 40H–43H to do the following:

1. Generate a basic timer interrupt that occurs at approximately 18.2 Hz.
2. Cause the DRAM memory system to be refreshed.
3. Provide a timing source to the internal speaker and other devices. The timer in the personal computer is an 8253 instead of an 8254.

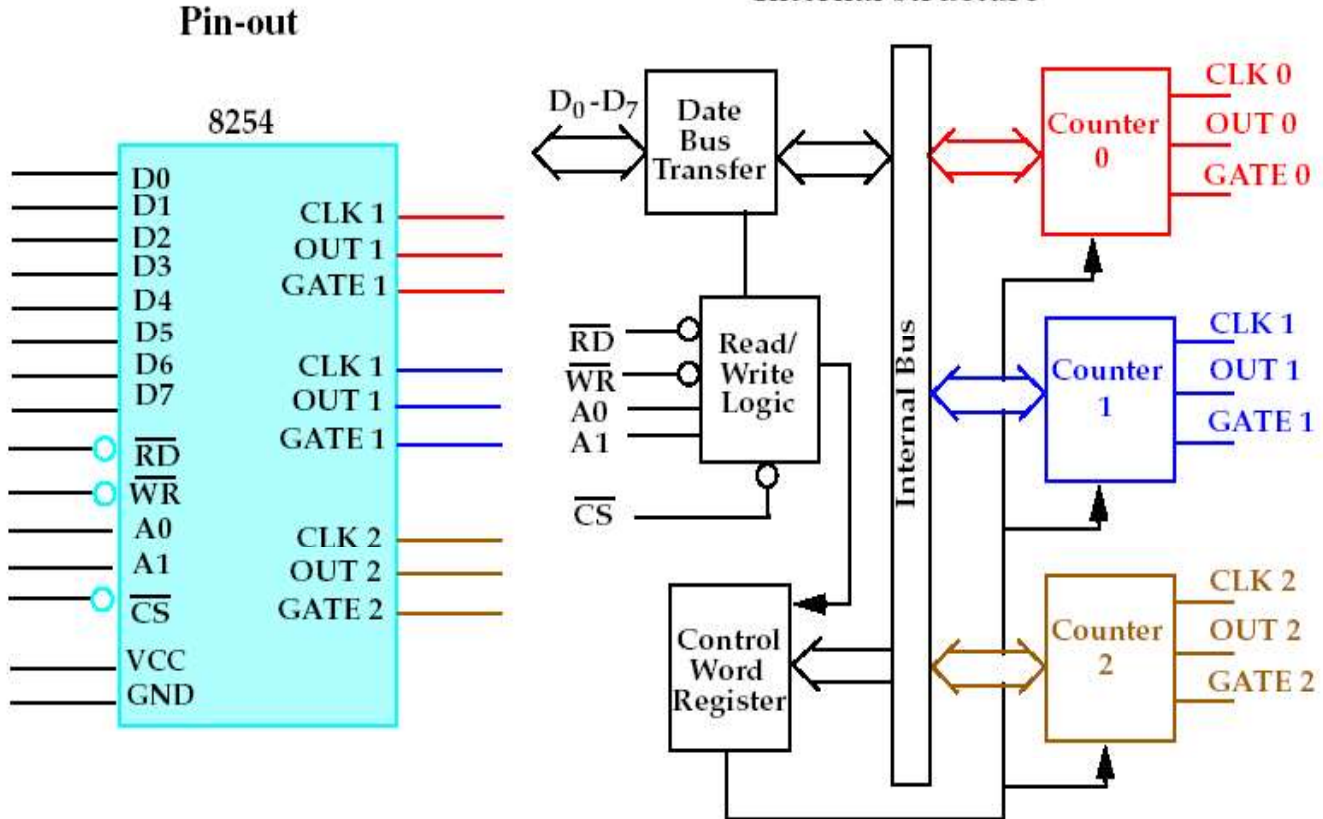
8254 Functional Description

Figure 11–33 shows the pin-out of the 8254, which is a higher-speed version of the 8253, and a diagram of one of the three counters. Each timer contains a CLK input, a gate input, and an output (OUT) connection. The CLK input provides the basic operating frequency to the timer, the gate pin controls the timer in some modes, and the OUT pin is where we obtain the output of the timer.

The signals that connect to the microprocessor are the data bus pins (D_7 – D_0), $\overline{RD}$, $\overline{WR}$, $\overline{CS}$, and address inputs A_1 and A_0 . The address inputs are present to select any of the four internal registers used for programming, reading, or writing to a counter. The personal computer contains an 8253 timer or its equivalent, decoded at I/O ports 40H–43H. Timer zero is programmed to generate an 18.2 Hz signal that interrupts the microprocessor at interrupt vector 8 for a clock tick. The tick is often used to time programs and events in DOS. Timer 1 is programmed for 15 μ s, which is used on the personal computer to request a DMA action used to refresh the dynamic RAM. Timer 2 is programmed to generate a tone on the personal computer speaker.



Internal structure



: 8253/54 TIMER DESCRIPTION AND INITIALIZATION

The 8253 chip was used in the IBM PC/XT, but starting with the IBM PC AT, the 8254 replaced the 8253. The 8254 and 8253 have exactly the same pinout. The 8254 is a superset of the 8253, meaning that all programs written for the 8253 will run on the 8254. The following are pin descriptions of the 8253/54.

A0, A1, $\overline{CS}$

Inside the 8253/54 timer, there are three counters. Each works independently and is programmed separately to divide the input frequency by a number from 1 to 65,536. Each counter is assigned an individual port address. The control register common to all three counters has its own port address. This means that a total of 4 ports are needed for a single 8253/54 timer. The ports are addressed by A0, A1, and $\overline{CS}$, as shown in Table 13-1. Each of the 3 counters has 3 pins associated with it, CLK (clock), GATE, and OUT, as shown in Figure 13-1.

Table 13-1: Addressing 8253/54

$\overline{CS}$	A1	A0	Port
0	0	0	Counter 0
0	0	1	Counter 1
0	1	0	Counter 2
0	1	1	Control register
1	x	x	8253/54 is not selected

(Reprinted by permission of Intel Corporation, Copyright Intel Corp. 1983)

CLK

CLK is the input clock frequency, which can range between 0 and 2 MHz for the 8253. For input frequencies higher than 2 MHz, the 8254 must be used; the 8254 can go as high as 8 MHz, and the 8254-2 can go as high as 10 MHz.

OUT

Although the input frequency is a square wave of 33% duty cycle, the shape of the output frequency coming from the OUT pin after being divided can be programmed. Among the options are square-wave, one-shot, and other square-shape waves of various duty cycles but no sine-wave or saw-tooth shapes.

GATE

This pin is used to enable or disable the counter. Putting HIGH (5 V) on GATE enables the counter, whereas LOW (0 V) disables it. In some modes a 0-to-1 pulse must be applied to GATE to enable the counter.

D0 - D7

The D0 - D7 data bus of the 8253/54 is a bidirectional bus connected to D0 - D7 of the system data bus. The data bus allows the CPU to access various registers inside the 8253/54 for both read and write operations. $\overline{RD}$ and $\overline{WR}$ (both active low) are connected to $\overline{IOR}$ and $\overline{IOW}$ control signals of the system bus.

Initialization of the 8253/54

Each of the three counters of the 8253/54 must be programmed separately. In order to program any of the three counters, the control byte must first be written into the control register, which among other things tells the 8253/54 what shape of output pulse is needed. In addition, the number that the input clock should be divided by must be written into that counter of the 8253/54. Since this number can be as high as FFFF (16-bit data) and the data bus for the 8253/54 timer is only 8 bits wide, the divisor must be sent in one byte at a time. The 8253/54 must be initialized before it is used.

Control word

Figure 13-2 shows the one-byte control word of the 8253/54. This byte, which is sent to the control register, has the following bits.

D0 chooses between a binary number divisor of 0000 to FFFFH or a BCD divisor of 0000 to 9999H. The lowest number that the input frequency can be divided by for both options is 0001. The highest number is 2^{16} for binary and 10^4 for BCD. To get the highest count (65,536 decimal and 10000 BCD), the counter is loaded with zeros.

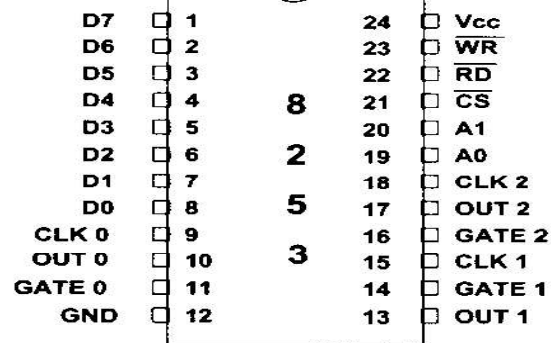
D1, D2, and D3 are for mode selection. There are six possible modes that determine the shape of the output signal.

Mode 0	Interrupt on terminal count
Mode 1	Programmable one-shot
Mode 2	Rate generator
Mode 3	Square wave rate generator
Mode 4	Software triggered strobe
Mode 5	Hardware triggered strobe

D4 and D5 are for RL0 and RL1. The data bus of the 8253/54 is 8 bits (1 byte), but the number that the input frequency can be divided by (divisor) can be as high as FFFFH. Therefore, RL0 and RL1 are used to indicate the size of the divisor. RL0 and RL1 have three options: (1) read/write the most significant byte (MSB) only, (2) read/write the least significant byte (LSB) only, (3) read/write the LSB first followed immediately by the MSB.

The options for RL0 and RL1 show that programmers cannot only write the value of the divisor into the 8253/54 timer but read the contents of the counter at any given time, as well. Since all counters are down counters, and the count register is decremented, the count register's contents can be read at any time, thus using the 8253/53 as an event counter.

D6 and D7 are used to select which of the three counters, counter 0, counter 1, or counter 2, is to be initialized by the control byte.



Block diagram

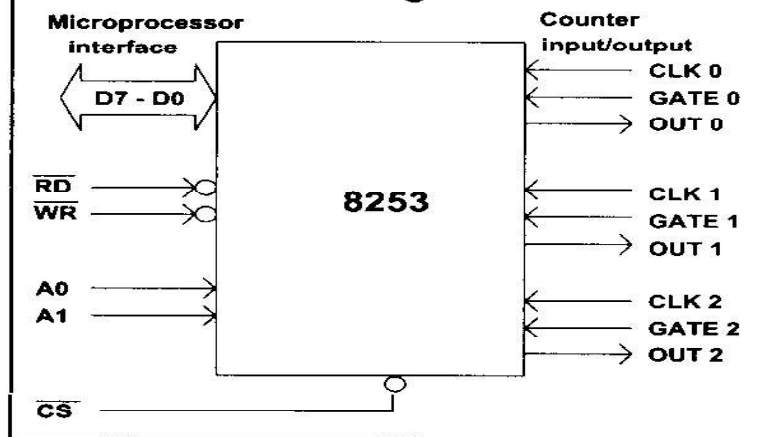


Figure 13-1. 8253 Pin and Function Diagram
(Reprinted by permission of Intel Corporation, Copyright Intel Corp. 1983)

Each counter is individually programmed by writing a control word, followed by the initial count.

The control word allows the programmer to select the counter, model of operation, binary or BCD count and type of operation (read/write).

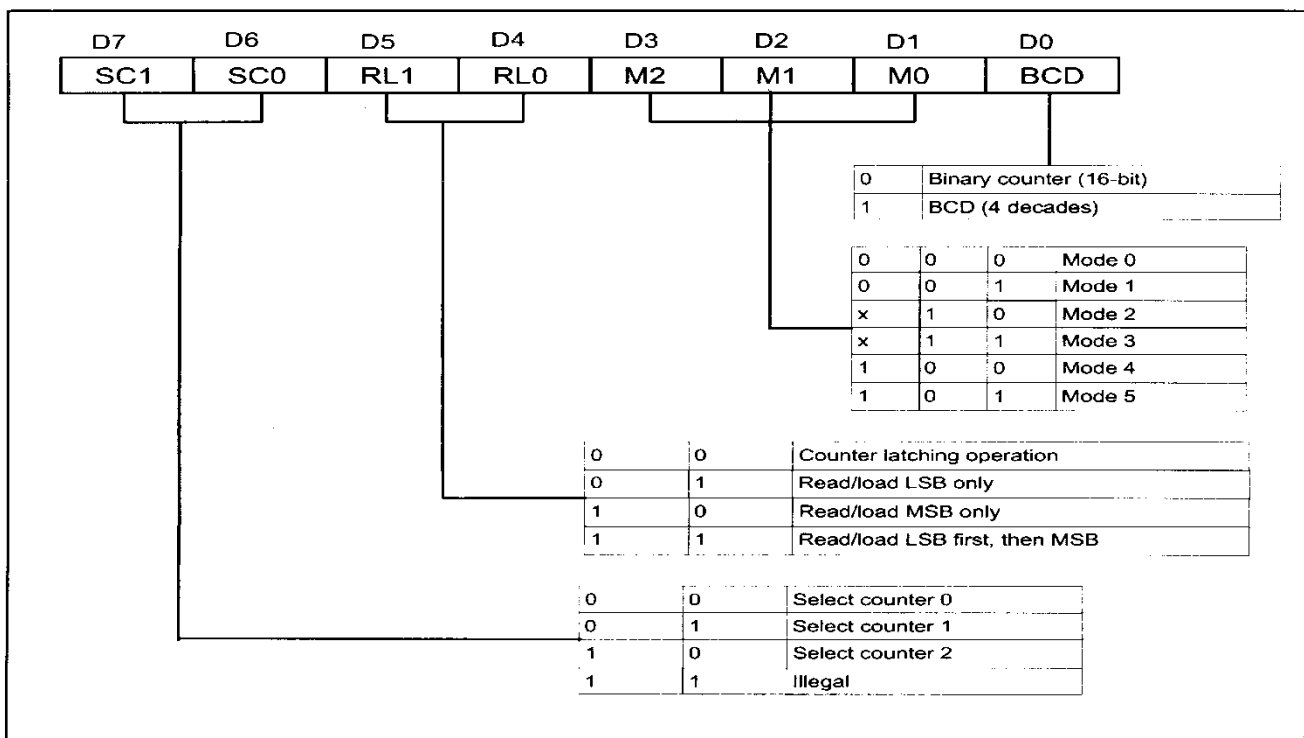
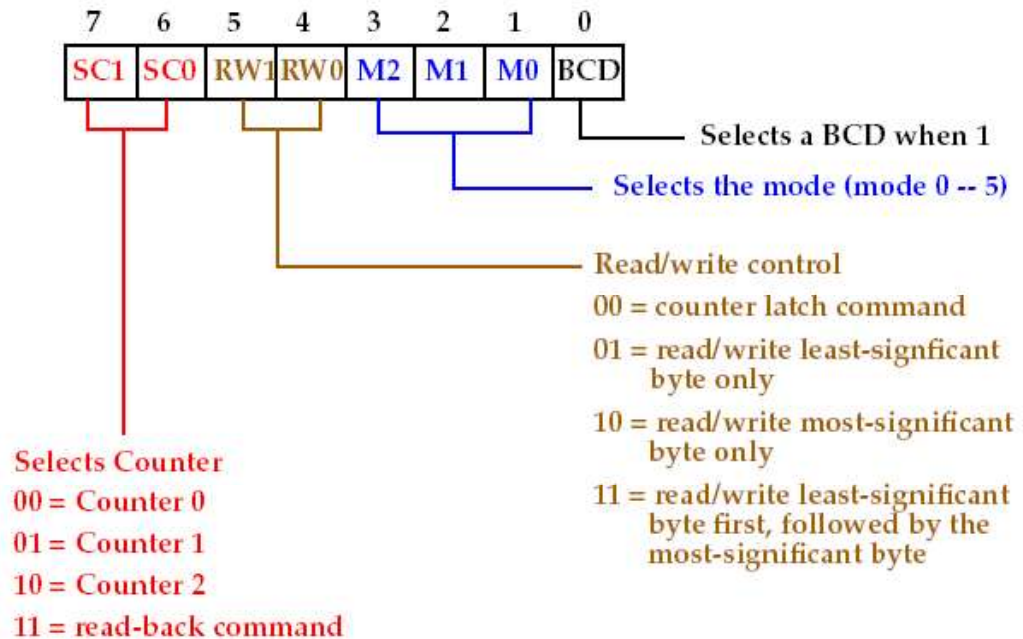


Figure 5-2. 8253/54 Control Word Format

READ Operations: Reading a counter while it is counting Three Ways, Contd.

3. Read Back Command

Written into the control word (A1A0=11)
To latch Status and/or Count of any
of the three counters
(Up to all 3 counters can be
specified simultaneously)

The command byte written during a
"Read back command"

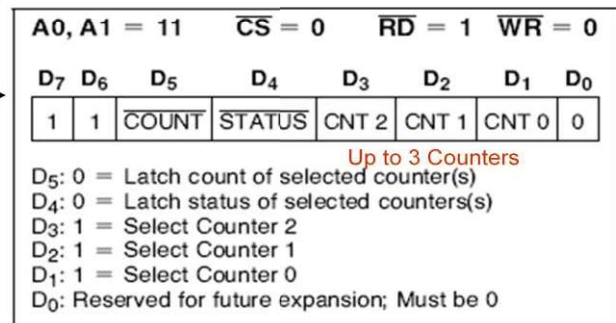


Figure 10. Read-Back Command Format

Shows how the counter was programmed

Format for the Counter Status Register
(latched into the Status latch
if so specified in the Read-Back Command)

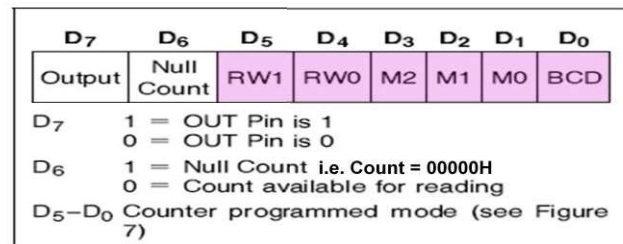


Figure 11. Status Byte

The 6 Counter Modes

- Mode 0: Event Counter
- Mode 1: Hardware triggered One-Shot
- Mode 2: Divide-by N Counter.
- Mode 3: Square-Wave Generator.
- Mode 4: Software-Triggered Strobe
- Mode 5: Hardware-Triggered Strobe

One off

Free
Running

One off

Mode 0: interrupt on terminal count

The output in this mode is initially low, and will remain low for the duration of the count if GATE = 1.

$$\text{Width of low pulse} = N \times T$$

Where N is the the clock count loaded into counter, and T is the clock period of the CLK input.

Mode 0: interrupt on terminal count

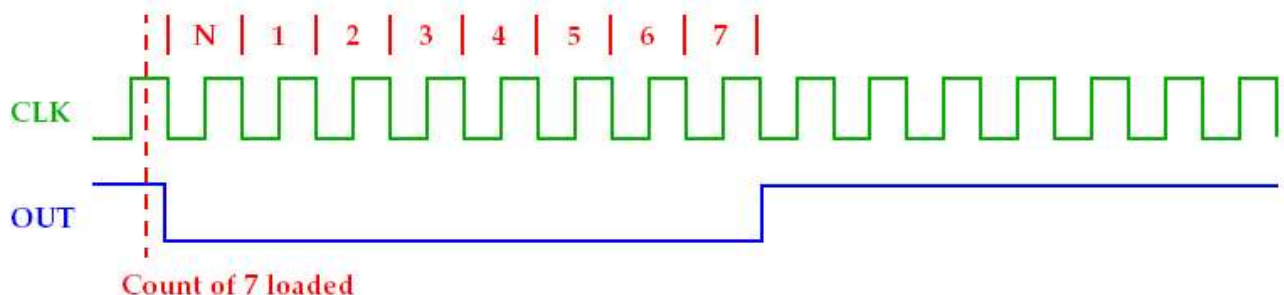
When the terminal count is reached, the output will go high and remain high until a new control word or new count number is loaded.

- In this mode, if GATE input becomes low at the middle of the count, the count will stop and the output will be low.
- The count resumes when the gate becomes high again.
- This in effect adds to the total time the output is low.

Mode 0 allows the 8254 counter to be used as an events counter.

The output becomes a logic 0 when the control word is written and remains there until N plus the number of programmed counts.

Gate (G) input must be a logic 1 for counting. If G becomes 0 in the middle of the count, the counter will stop until G again becomes a 1.



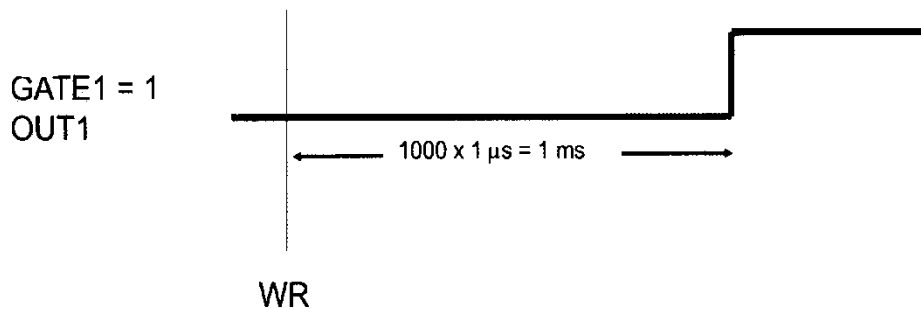
GATE The gate (G) input stops the count when 0.

Example 5-7

Assume that $GATE1 = 1$ and $CLK1 = 1\text{ MHz}$, and the clock count $N = 1000$. Show the output of $OUT1$ if it is programmed in mode 0.

Solution:

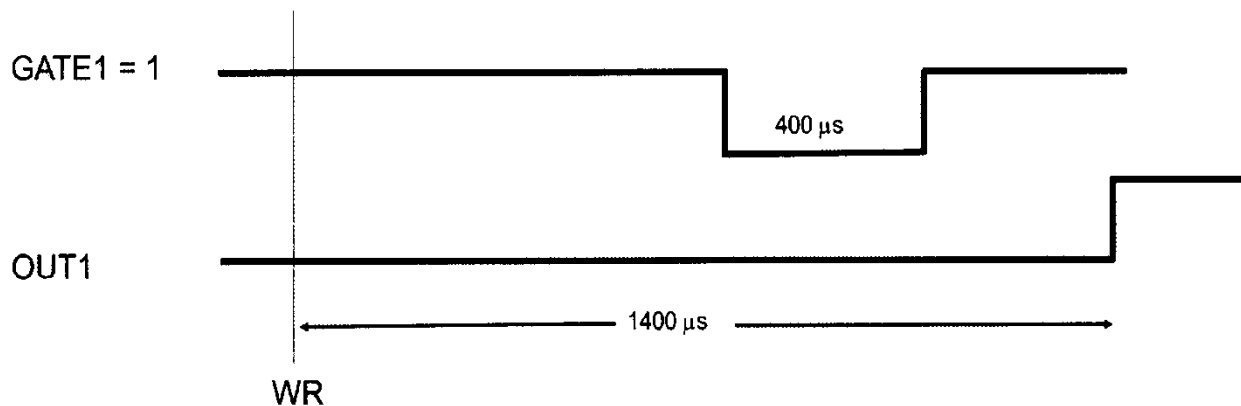
The clock period of $CLK1$ is $1\text{ }\mu\text{s}$; therefore, $OUT1$ is low for $1000 \times 1\text{ }\mu\text{s} = 1\text{ ms}$, before it goes high, as shown in the following diagram.

**Example 5-8**

In Example 5-7, assume that $GATE1$ becomes zero for $400\text{ }\mu\text{s}$. What is the width of the low pulse for $OUT1$?

Solution:

It is $1000\text{ }\mu\text{s} + 400\text{ }\mu\text{s} = 1400\text{ }\mu\text{s}$, as shown next.



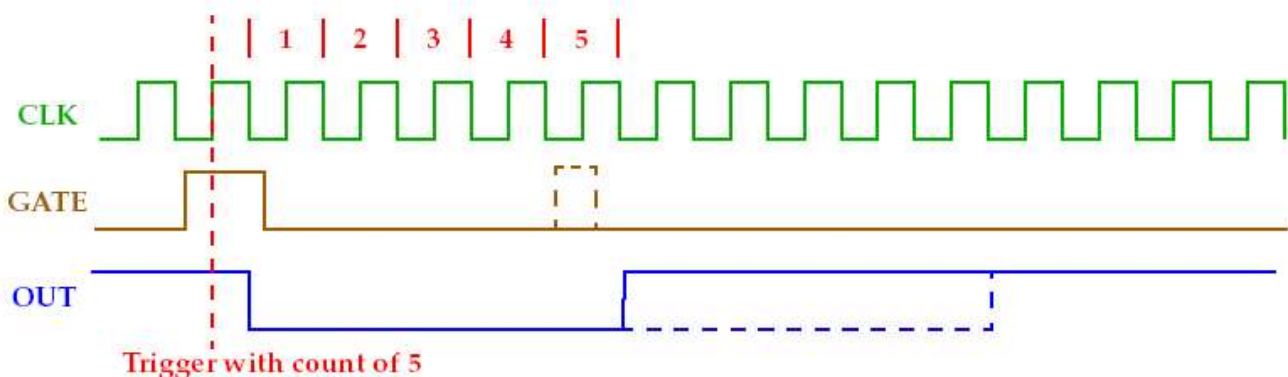
Mode 1: programmable one-shot

- This mode is also called hardware triggerable one-shot.
- The triggering must be done through the GATE input by sending a 0-to-1 pulse to it.
- The following two steps must be performed:
 1. Load the count registers.
 2. A 0-to-1 pulse must be sent to the GATE input to trigger the counter.
- Contrast this with mode 0, in which the counter produces the output immediately after the counter is loaded as long as $GATE = 1$.
- In mode 1 after sending the 0-to-1 pulse to GATE, OUT becomes low and stays low for a duration of $N \times T$, then becomes high and stays high until the gate is triggered again.

Mode 1 causes the counter to function as a retriggerable monostable multivibrator (one-shot)

The G input triggers the counter so that it develops a pulse at the OUT connection that becomes a logic 0 for the duration of the count.

if (G) occurs within the duration of the output pulse, the counter is reloaded with the count and the OUT continues for the total length of the count.

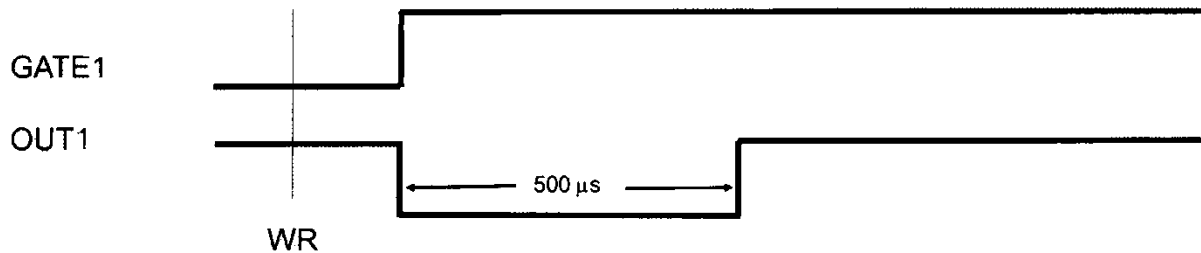


Example 5-9

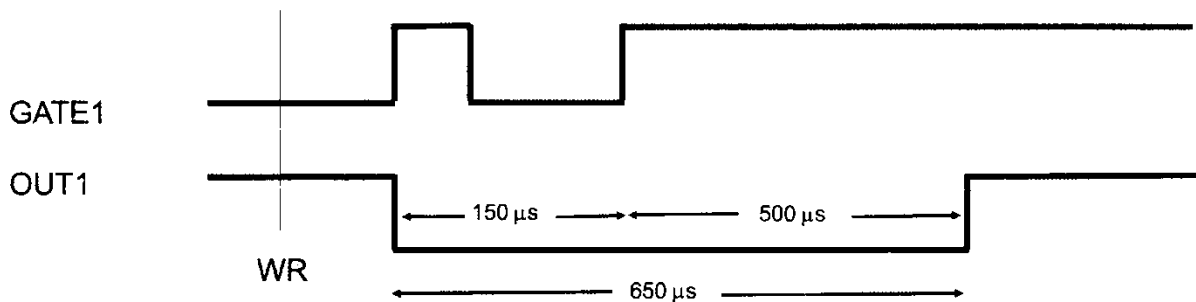
- (a) If $\text{CLK1} = 1 \text{ MHz}$ and $N = 500$, show the output for OUT1 if it is programmed for mode 1.
 (b) Assume that after 150 clock pulses, GATE1 is retriggered. Show the output for OUT1.

Solution:

- (a) Notice that OUT1 becomes low only when GATE1 goes from 0 to 1.



- (b) If GATE1 is retriggered after 150 clock pulses, COUNT1 is reloaded with $N = 500$ and the count starts all over again, making the OUT1 pulse duration $650 \mu\text{s}$ as shown next.



Mode 2: rate generator

Mode 2 is also called divide-by-N counter.

- In this mode, if $\text{GATE} = 1$, OUT will be high for the $N \times T$ clock period, goes low for only one clock pulse, then the count is reloaded automatically, and the process continues indefinitely.

Example 5-10

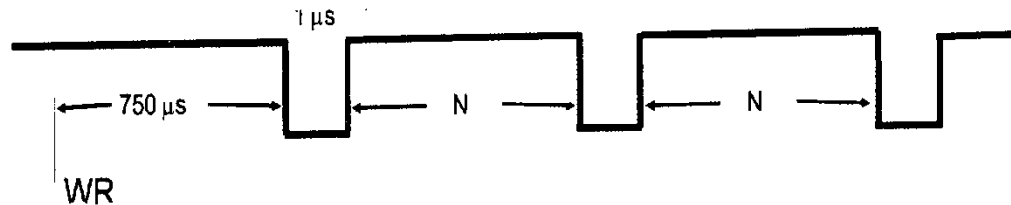
If CLK2 = 1 MHz, GATE2 = 1, and N = 750, show OUT2 if COUNT2 is programmed for mode 2.

Solution:

Notice that the count is reloaded automatically and the counter continues to produce OUT2.

GATE2 = 1

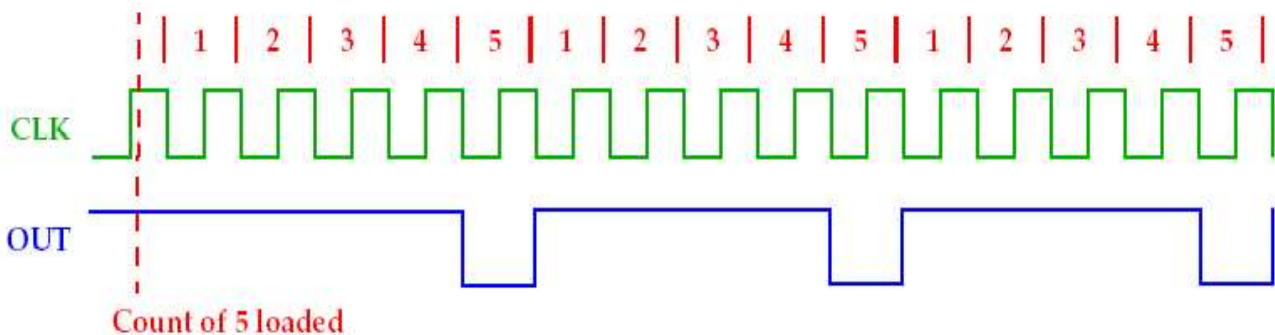
OUT1



Mode 2 allows the counter to generate a series of continuous pulses that are one clock pulse wide.

The separation between pulses is determined by the count. The cycle is repeated until the counter is programmed with a new count or the G pin is placed at logic 0.

To generate a continuous series of pulses the G input must be a logic 1.



GATE The gate (G) input stops the count when 0.

Mode 3: square wave rate generator

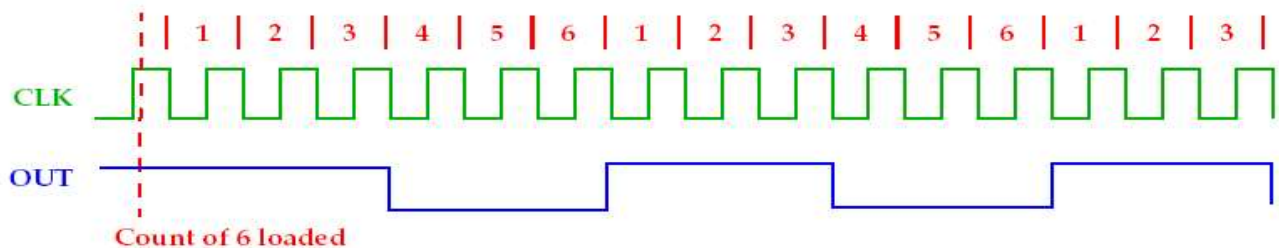
In this mode if $GATE = 1$, OUT is a square wave where the high pulse is equal to the low pulse if N is an even number.

- In this case the high part and low part of the pulse have the same duration and are equal to $(N/2) \times T$ (50% duty cycle)
- If N is an odd number, the high pulse is one clock pulse longer.
- This mode is widely used as a frequency divider and audio-tone generator.

In Mode 3 the counter generates a continuous square-wave at the OUT connection, provided that the G pins is a logic 1.

If the count is even, the output is high for one-half of the count and low for one-half of the count.

If the count is odd, the outputs is high for one clocking period longer that it is low.



GATE The gate (G) input stops the count when 0.

Example 5-11

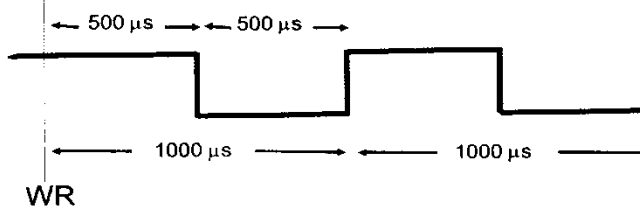
If $CLK2 = 1\text{ MHz}$, $GATE1 = 1$, $N = 1000$, show $OUT1$ if $COUNT1$ is programmed for mode 3.

Solution:

Since the clock period is $1\text{ }\mu\text{s}$, $OUT1$ is high for $500\text{ }\mu\text{s}$ and low for $500\text{ }\mu\text{s}$, producing the square wave of 1 ms period continuously, as shown next.

$GATE1 = 1$

$OUT1$



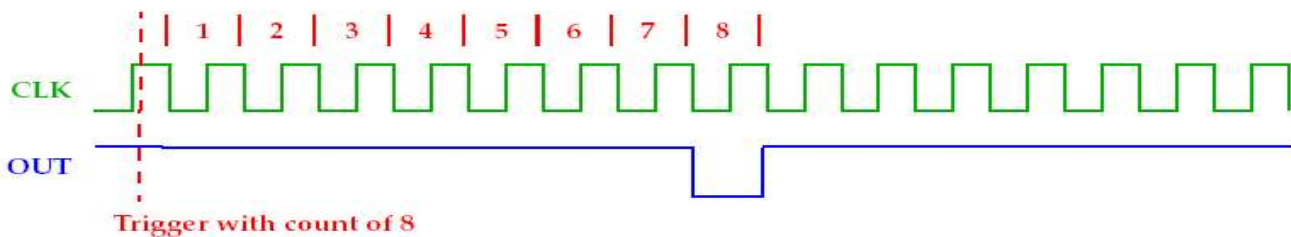
Mode 4: software trigger strobe

- In this mode if $GATE = 1$, the output will go high upon loading the count.
- It will stay high for the duration of $N \times T$.
- After the count reaches zero (terminal count), it becomes low for one clock pulse, then goes high again and stays high until a new command word or new count is loaded.
- To repeat the strobe, the count must be reloaded again.
- Mode 4 is similar to mode 2, except that the counter is not reloaded automatically.
- In this mode, the count starts the moment the count is written into the counter.

Mode 4 allows the counter to produce a single pulse at the output. This mode operates as a software triggered one-shot.

The cycle does not begin until the counter is loaded with its complete count.

Use the G input to enable the counter. The G input must be a logic 1 for the counter to operate for these three modes.



GATE The gate (G) input stops the count when 0.

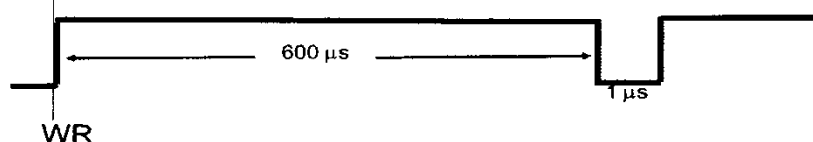
Example 5-12

If $CLK0 = 1\text{ MHz}$, $GATE0 = 1$, and $N = 600$, show the shape of $OUT0$ where counter 0 is programmed for mode 4.

Solution:

Since the $CLK0$ period is $1\text{ }\mu\text{s}$, after the count is loaded $OUT0$ will be high for $600\text{ }\mu\text{s}$ and will go low for $1\text{ }\mu\text{s}$. Then it will go high again and stay high until the counter is reprogrammed, as shown below.

$GATE0 = 1$
 $OUT0$



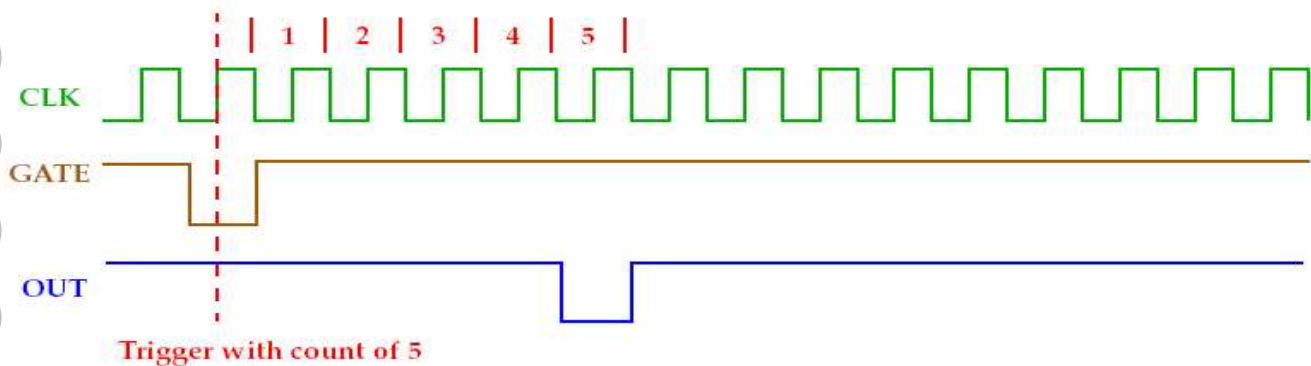
Mode 5: hardware trigger strobe

- This mode is similar to mode 4 except that the trigger must be done with the GATE input.
- In this mode after the count is loaded, we must send a low-to-high pulse to the gate to start the counter.

Mode 5 is similar to mode 4 and allows the counter to produce a single pulse at the output.

The difference is that this mode is a hardware triggered one-shot and is started by a trigger pulse on the G pin instead of by the software.

This mode is also similar to mode 1 because it is retriggerable.

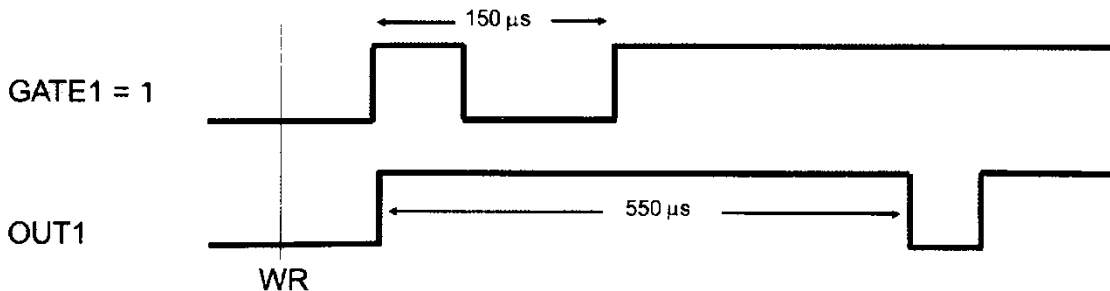


Example 5-14

In Example 5-13, assume that GATE1 is retriggered after 150 pulses. Show the output for OUT1.

Solution:

If GATE1 is retriggered after 150 clock pulses into the countdown, COUNT1 is reloaded with $N = 400$ and the counts begins again, making the OUT1 pulse duration $550 \mu s$, as shown next.

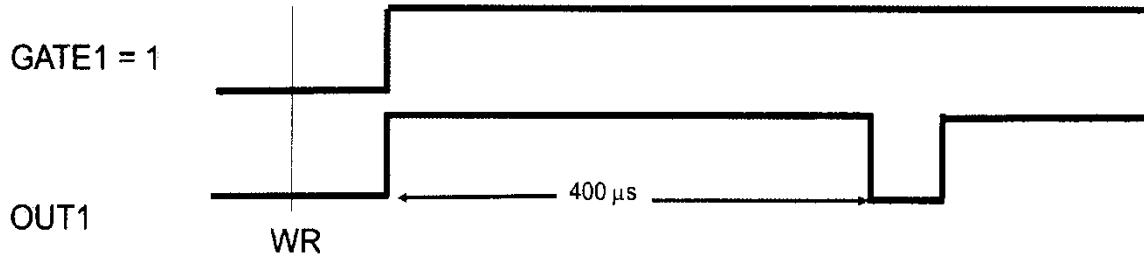


Example 5-13

If CLK1 = 1 MHz, and N = 400, show the output for OUT1 if it is programmed for mode 5.

Solution:

Notice that the count starts only when the 0-to-1 pulse is applied to GATE1.

**Example 5-1**

Pin $\overline{CS}$ of a given 8253/54 is activated by binary address A7 - A2 = 100101.

- Find the port addresses assigned to this 8253/54.
- Find the configuration for this 8253/54 if the control register is programmed as follows.

```
MOV AL,00110110
OUT 97H,AL
```

Solution:

- From Table 5-1, we have the following:

CS	A1A0	Port	Port address (hex)
1001 01	00	Counter 0	94
1001 01	01	Counter 1	95
1001 01	10	Counter 2	96
1001 01	11	Control register	97

- Breaking down the control word 00110110 and comparing it with Table 5-1 indicates counter 0 since the SC bits are 00. The RL bits of 11 indicates that the low-byte read/write is followed by the high byte. The mode selection is mode 3 (square wave), and finally binary counting is selected since the D0 bit is 0.

Example 5-2

Use the port addresses in Example 5-1 to program:

- (a) counter 0 for binary count of mode 3 (square wave) to divide CLK0 by number 4282 (BCD)
- (b) counter 2 for binary count of mode 3 (square wave) to divide CLK2 by number C26A hex
- (c) Find the frequency of OUT0 and OUT2 in (a) and (b) if CLK0 = 1.2 MHz, CLK2 = 1.8 MHz.

Solution:

- (a) To program counter 0 for mode 3, we have 00110111 for the control word. Therefore,

```
MOV  AL,37H      ;counter 0, mode 3, BCD
OUT  97H,AL      ;send it to control register
MOV  AX,4282H    ;load the divisor (BCD needs H for hex)
OUT  94H,AL      ;send the low byte
MOV  AL,AH       ;to counter 0
OUT  94H,AL      ;and then the high byte to counter 0
```

- (b) By the same token:

```
MOV  AL,B6H      ;counter2, mode 3, binary(hex)
OUT  97H,AL      ;send it to control register
MOV  AX,C26AH    ;load the divisor
OUT  96H,AL      ;send the low byte
MOV  AL,AH       ;to count 2
OUT  96H,AL      ;send the high byte to counter 2
```

- (c) The output frequency for OUT0 is 1.2MHz divided by 4282, which is 280 Hz. Notice that the program in part (a) used instruction "MOV AX,4282H" since BCD and hex numbers are represented in the same way, up to 9999. For OUT2, CLK2 of 1.8 MHz is divided by 49770 since C26AH = 49770 in decimal. Therefore, OUT2 frequency is a square wave of 36 Hz.

Example 5-3

Using the port addresses in Example 5-1, show the programming of counter 1 to divide CLK1 by 10,000, producing the mode 3 square wave. Use the BCD option in the control byte.

Solution:

```
MOV  AL,77H      ;counter1, mode 3, BCD
OUT  97H,AL      ;send it to control register
SUB  AL,AL       ;AL =0 load the divisor for 10,000
OUT  95H,AL      ;send the low byte
OUT  95H,AL      ;and then the high byte to counter 1
```

Section 5.2 IBM PC 8253/54 Timer Connections and Programming

Table 5-2: 8253/4 Port Address Calculation in the PC

Binary Address												
AEN	A9	A8	A7 A6 A5 A4				A3 A2 A1 A0		Hex Address	Function		
1	0	0	0	1	0	x	x	x	0	0	40	Counter 0
1	0	0	0	1	0	x	x	x	0	1	41	Counter 1
1	0	0	0	1	0	x	x	x	1	0	42	Counter 2
1	0	0	0	1	0	x	x	x	1	1	43	Control register

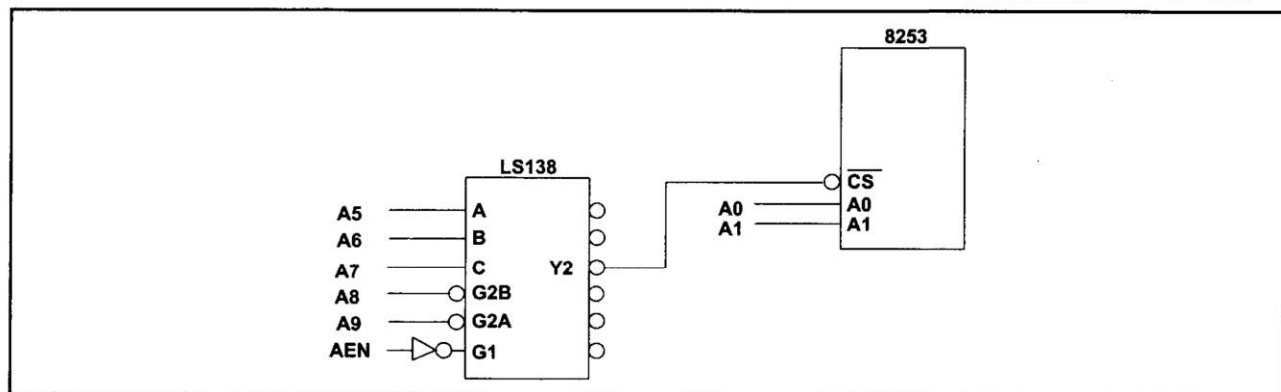


Figure 5-3. 8253 Port Selection in the PC/XT

Time delay for 80x86 PCs

Creating time delays in 8088/86-based PC/XT, PS2, and compatibles

```
MOV CX, N
```

```
AGAIN: LOOP AGAIN
```

The above codes can be used to generate delays ($N \times T \times 17$ seconds)

- Approximate delay time
- The delay is not only frequency dependent but also CPU dependent.
- IBM provides a scheme to create a time delay using hardware that is not only frequency but also CPU independent.

Time delays in 80x86 IBM PC

The following method of creating hardware time delays was first implemented in the IBM PC and compatible computers.

- To create a processor independent delay, IBM made PB4 of port 61H toggle every 15.085 μ s.
- CS holds 15.085 μ s in the following codes.

The following method of creating hardware time delays was first implemented in the IBM PC and compatible computers.

- To create a processor independent delay, IBM made PB4 of port 61H toggle every 15.085 μ s.
- CS holds 15.085 μ s in the following codes.

;(CX) = COUNT OF 15.085 MICROSECOND

```
WAITF      PROC  NEAR
            PUSH  AX
```

```
WAITF1:
```

```
            IN      AL,61H
            AND      AL,10H      ;CHECK PB4
            CMP      AL,AH      ;DID IT JUST CHANGE
            JE        WAITF1     ;WAIT FOR CHANGE
            MOV      AH,AL      ;SAVE THE NEW PB4 STATUS
            LOOP     WAITF1     ;CONTINUE UNTIL CX BECOMES 0
            POP      AX
```

```
            RET
WAITF      ENDP
```

• Now a time delay of any duration can be created regardless of the CPU frequency as long as it is a 286 and higher PC.

• For example set CX=33144
($33144 \times 15.085 \mu\text{s} = 0.5 \text{ second}$)

```
MOV     CX, 33144
```

```
CALL    WAITF
```

A delay of 250 ms

;this 250 ms delay can work only on 286,386,486, and Pentium PCs

```

;
DELAY  PROC    NEAR
        MOV     CX,16578      ;16578 x 15.08 microsec = 250 ms
        PUSH    AX
WAIT:   IN      AL,61H
        AND     AL,10H        ;check PB4
        CMP     AL,AH         ;did it just change?
        JE      WAIT         ;wait for change
        MOV     AH,AL         ;save the new PB4 status
        LOOP    WAIT         ;decrement CX and continue
                                ;until CX becomes 0
        POP     AX
DELAY  RET
        ENDP

```

A delay of 5 ms between notes can be achieved in the same way.

```

DELAY_OFF  PROC    NEAR
            MOV     CX,331      ;331 x 15.08 micro sec = 5 ms
            PUSH    AX
WAIT:      IN      AL,61H
            AND     AL,10H      ;check PB4
            CMP     AL,AH       ;did it just change?
            JE      WAIT        ;wait for change
            MOV     AH,AL       ;save the new PB4 status
            LOOP    WAIT        ;continue until CX becomes 0
            POP     AX
DELAY_OFF  RET
            ENDP

```

The following creates a delay for the 8088-based PC/XT of 4.7 MHz.

```

DELAY      PROC    NEAR
            SUB     CX,CX
G7:        LOOP    G7
            RET
ELAY       ENDP

```

Another way to get an approximate DELAY_OFF of 5 ms for the 8088 PC/XT of 4.7-MHz computers is

```

DELAY_OFF  PROC    NEAR
            MOV     CX,1400
G1:        LOOP    G1      ;1400 x 210 ns x 17 =5 ms
            RET
DELAY_OFF  ENDP

```

Example 5-4

Using the BIOS WAITF routine, show how to create a 1.5-second time delay.

Solution:

Since the 1.5-second delay requires the counter to be set to 99,436 ($1.5/15.085 \mu s = 99,436$) and the maximum value of CX is 65,536, the following method is used to generate the 1.5-second delay.

```

                MOV    BL,03
BACK:          MOV    CX,33144    ;1/2-second delay
                CALL   WAITF
                DEC     BL
                JNZ     BACK
  
```

Example 5-5

Show the values to be loaded into counter 2 in order to have the output frequency for the notes (a) D3, (b) A3, and (c) A4.

Solution:

From Figure 5-5, notice that the frequency for note D3 is 147. The value that must be loaded into counter 2 is 1.1931 MHz divided by 147, which is 8116. Going through this procedure for each note gives the following:

Note	Frequency	Value Loaded into Counter 2	
		Decimal	Hex
D3	147 Hz	8116	1FB4
A3	220 Hz	5423	152F
A4	440 Hz	2711	0A97

Now that the values to be loaded into counter 2 are known, the program for getting the speaker to sound the notes for a certain duration is shown in Example 5-6.

Example 5-6

Program counter 2 to play the following notes: D3, A3, A4, for durations of 250, 500, and 500 ms, respectively. Place a 5-ms silence between each note.

Solution: This program uses the values calculated in Example 5-5.

```

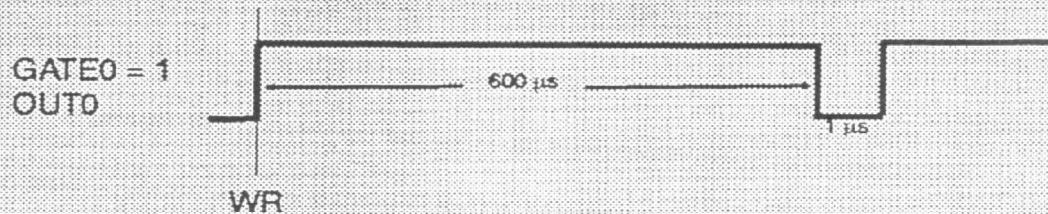
MOV AL,0B6H           ;control byte:counter2,lsb,msb,binary
OUT 43H,AL            ;send the control byte to control reg
;load the counter2 value for D3 and play it for 250 ms
MOV AX,1FB4H          ;for D3 note
OUT 42H,AL            ;the low byte
MOV AL,AH              ;the high byte
OUT 42H,AL
;turn the speaker on
IN AL,61H             ;get the current setting of port b
MOV AH,AL             ;save it
OR AL,00000011B       ;make pb0 =1 and pb1 =1
OUT 61H,AL            ;turn the speaker on
CALL DELAY             ;play this note for 250 ms
MOV AL,AH             ;get the original setting of port b
OUT 61H,AL            ;turn off the speaker
CALL DELAY_OFF         ;speaker off for this duration
;load the counter2 value for A3 and play it for 500 ms
MOV AX,152FH          ;for A3 note
OUT 42H,AL            ;the low byte
MOV AL,AH              ;the high byte
OUT 42H,AL
;turn the speaker on
IN AL,61H             ;get the current setting of port b
MOV AH,AL             ;save it
OR AL,00000011B       ;make PB0 =1 and PB1 =1
OUT 61H,AL            ;turn the speaker on
CALL DELAY             ;play for 250 ms
CALL DELAY             ;play for another 250 ms
MOV AL,AH             ;get the original setting of port b
OUT 61H,AL            ;turn off the speaker
CALL DELAY_OFF         ;speaker off for this duration
;load the counter2 value for A4 and play it for 500 ms
MOV AX,0A97H          ;for A4 note
OUT 42H,AL            ;the low byte
MOV AL,AH              ;the high byte
OUT 42H,AL
;turn the speaker on
IN AL,61H             ;get the current setting of port b
MOV AH,AL             ;save it
OR AL,00000011B       ;make PB0 =1 and PB1 =1
OUT 61H,AL            ;turn the speaker on
CALL DELAY             ;play for 250 ms
MOV AL,AH             ;get the original setting of port b
OUT 61H,AL            ;turn off the speaker
CALL DELAY_OFF         ;speaker off for this duration

```

If $\text{CLK0} = 1 \text{ MHz}$, $\text{GATE0} = 1$, and $N = 600$, show the shape of OUT0 where counter 0 is programmed for mode 4.

Solution:

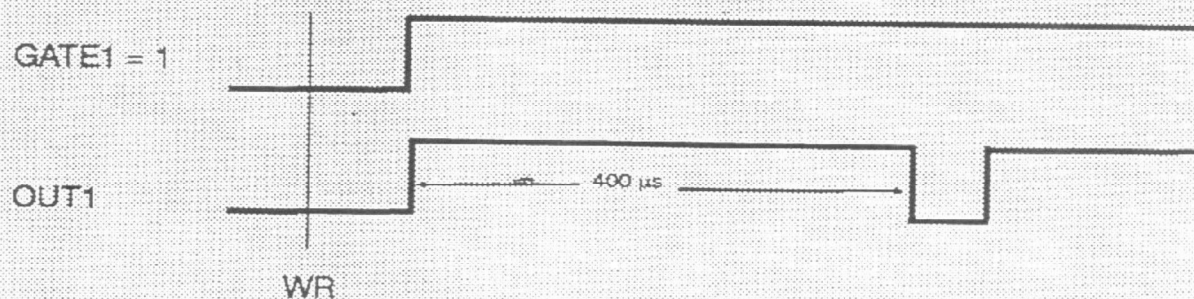
Since the CLK0 period is $1 \mu\text{s}$, after the count is loaded OUT0 will be high for $600 \mu\text{s}$ and will go low for $1 \mu\text{s}$. Then it will go high again and stay high until the counter is reprogrammed. This is shown as follows.



If $\text{CLK1} = 1 \text{ MHz}$, and $N = 400$, show the output for OUT1 if it is programmed for mode 5.

Solution:

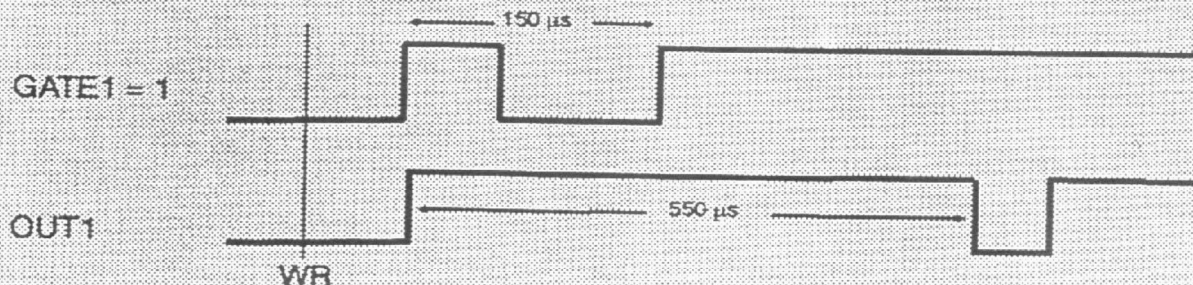
Notice that the count starts only when the 0-to-1 pulse is applied to GATE1 .



In Example 5-13, assume that GATE1 is retriggered after 150 pulses. Show the output for OUT1 .

Solution:

If GATE1 is retriggered after 150 clock pulses into the countdown, COUNT1 is reloaded with $N = 400$ and the count begins again, making the OUT1 pulse duration $550 \mu\text{s}$, as shown next.



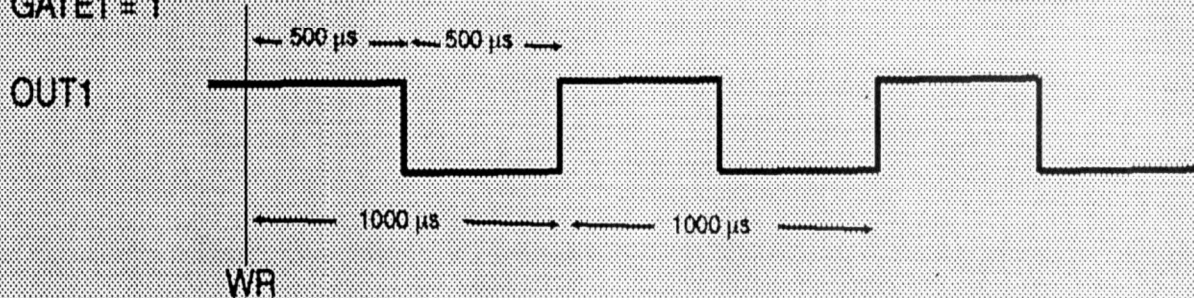
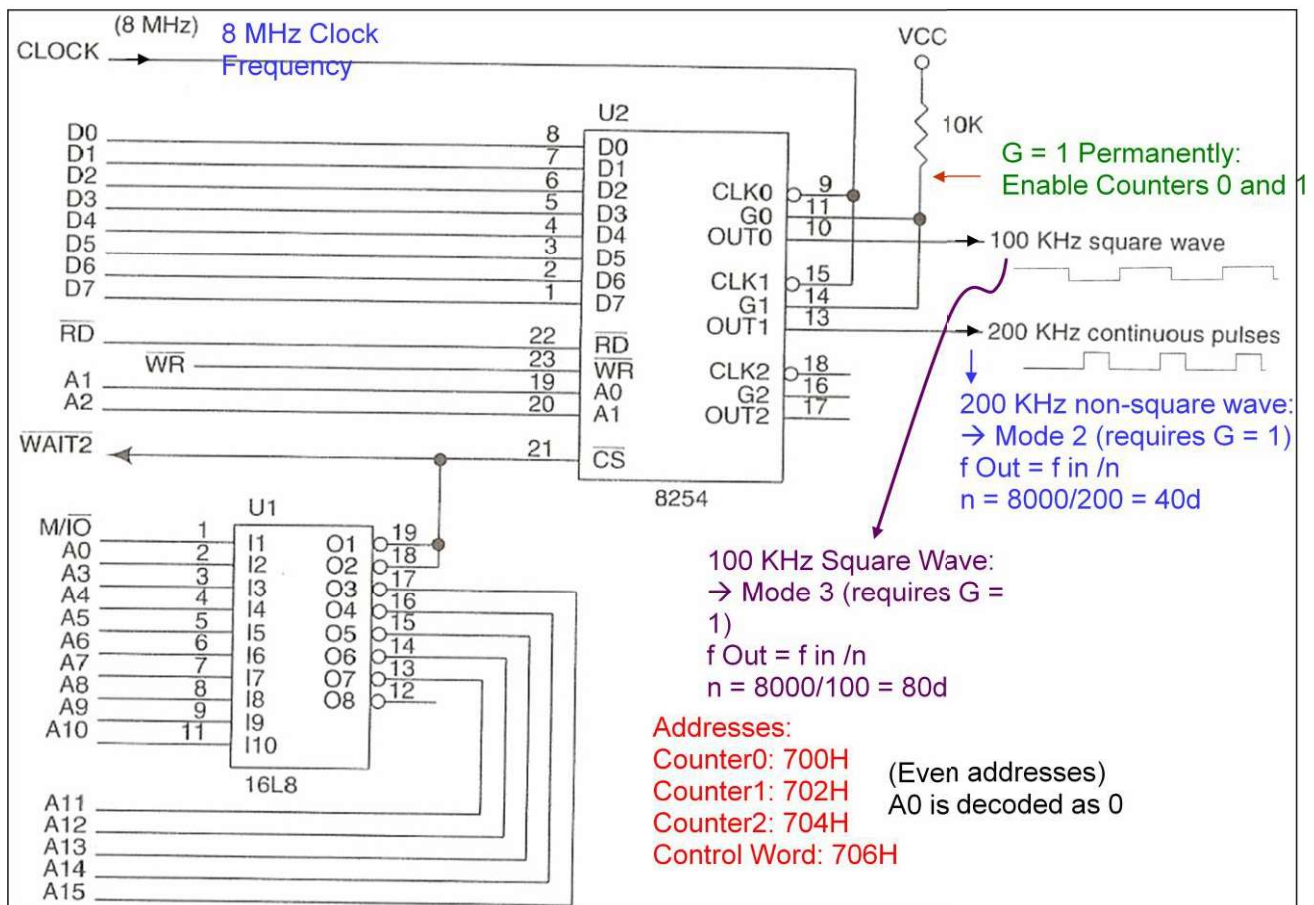
Example 5-11

If CLK2 = 1 MHz, GATE1 = 1, N = 1000, show OUT1 if COUNT1 is programmed for mode 3.

Solution:

Since the clock period is 1 μ s, OUT1 is high for 500 μ s and low for 500 μ s, producing the square wave of 1 ms period continuously, as shown next.

GATE1 = 1

**8254 PIT Programming Example**

;A procedure that controls the speed and direction of the motor
;in Figure 11-40.

;AH determines the speed and direction of the motor where
;AH is between 00H and FFH.

For Control Reg

CNTR EQU 706H ; PIT Control Word
CNT0 EQU 700H
CNT1 EQU 702H
COUNT EQU 30720
SPEED PROC NEAR USES BX DX AX

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
SC1	SC0	RW1	RW0	M2	M1	M0	BCD

MOV BL,AH ;calculate count corresponding to AH: AH has speed control byte (0→128→255)

MOV AX,120

; Multiply AH (speed input) by 120

MUL BL

; result in AX, **save in BX (BX has AH x 120)**

MOV BX,AX

MOV AX,COUNT

SUB AX,BX

MOV BX,AX

MOV DX,**CNTR**

MOV AL,**00110100B**

; program control word

OUT DX,AL

; for counter 0: Binary, Mode 2, 2 bytes R/W

MOV AL,01110100B

; same for counter 1

OUT DX,AL

; but do not start it yet by loading COUNT- do this after waiting time

MOV DX,CNT1

MOV AX,**COUNT**

OUT DX,AL

; LS byte of 30720 first

MOV AL,AH

OUT DX,AL

; then MS byte

.REPEAT

; **wait for counter 1 count to reach Waiting Count in BX**

IN AL,DX

XCHG AL,AH

IN AL,DX

XCHG AL,AH

; Read LS byte of counter 1 (goes as AL)

; Put it in AH

; Read MS byte of counter 1 (goes as AL)

; swap AL and AH to put things back to order

.UNTIL BX == AX

MOV DX,CNT0

MOV AX,COUNT

;program counter 0

; to generate a set (#PS) for Q (**free-running**) after that waiting

; delay by Counter 1. Note you also load COUNT as with Counter 1

; Actual outputting LS byte then MS byte

OUT DX,AL

MOV AL,AH

OUT DX,AL

RET

ENDP

Always LS then MS

Counter 0 starts, free running at 260

Hz, Repeatedly read counter 1
Until it reaches "Waiting Count"

Counter 0 starts free running at 260 Hz,
delayed by **waiting count**
from the start of Counter 1

SPEED