Microprocessor System Design
Timers

Omid Fatemi

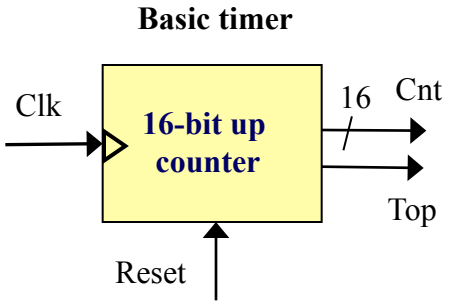
(omid@fatemi.net)

Outline

- Timers / counters / Watchdog timers
- 8253/4 description
- Programming the counters
- 8253 in a PC
- Generating sound
- Various modes of operation

Timers, counters, watchdog timers

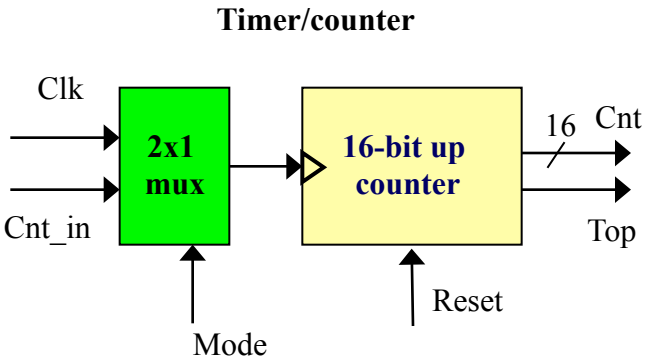
- **Timer: measures time intervals**
  - **To generate timed output events**
    - » **e.g., hold traffic light green for 10 s**
  - **To measure time between events**
    - » **e.g., measure a car's speed**
- **Based on counting clock pulses**
  - » **E.g., for a 100 MHz clock, Clk period would be 10 ns**
  - » **And we count 20,000 Clk pulses**
  - » **Then 200 microseconds have passed**
  - » **16-bit counter would count up to 2^{16} = 65,535*10 ns ➔ 655.35 μs (range), with a resolution of 10 ns**
  - » **Top: indicates top count reached, wrap-around**
  - » **How can we measure a time interval larger than the range?**



cnt: # of clock pulses since the counter was last reset (set to zero).

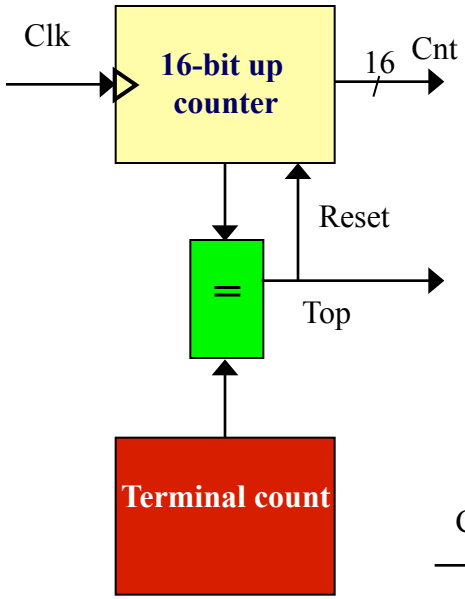
Counters

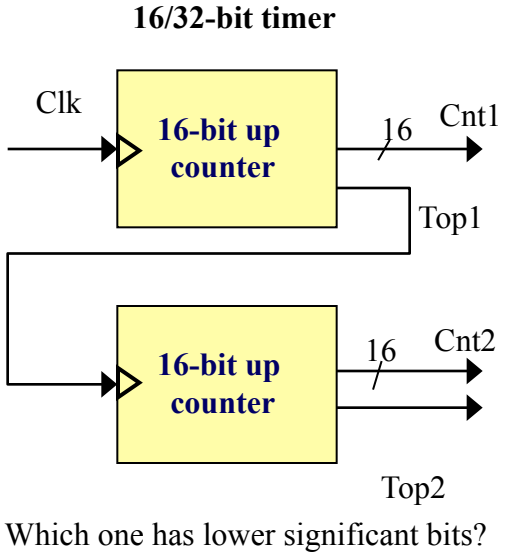
- **Counter: like a timer, but counts pulses on a *general input signal* rather than clock**
  - **e.g., count cars passing over a sensor**
  - **We can often configure device as either a timer or counter**
  - **Counters and timers can be combined to measure rates, such as the speed of a car (# of times wheel rotates in one second).**

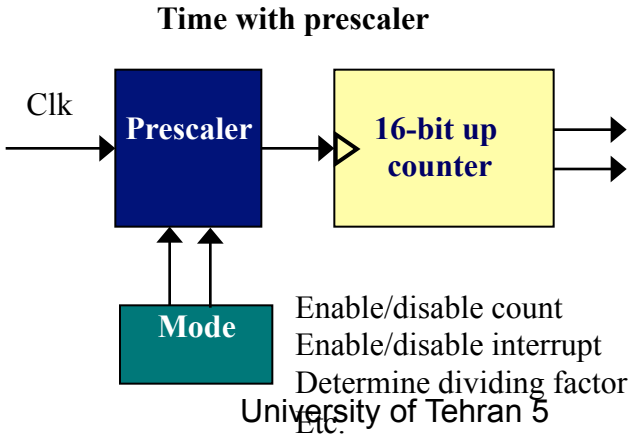


Other timer structures

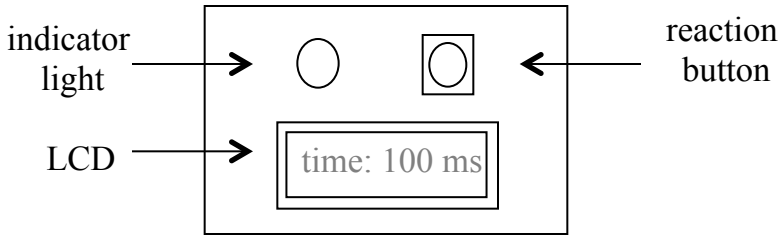
- **Interval timer**
  - Indicates when desired time interval has passed
  - We set terminal count to desired interval
    - » *Number of clock cycles = Desired time interval / Clock period*
    - » If f_{clock}=*100 MHZ*, how many times do we need to count to reach 3μs?
    - » *Top* both resets the counter, and informs us when the count is up. It is typically connected to an interrupt.
    - » We can also set the count back from terminal count to zero.
- **Cascaded counters**
- **Prescaler**
  - Divides clock
  - Increases range, decreases resolution.







Example: Reaction Timer

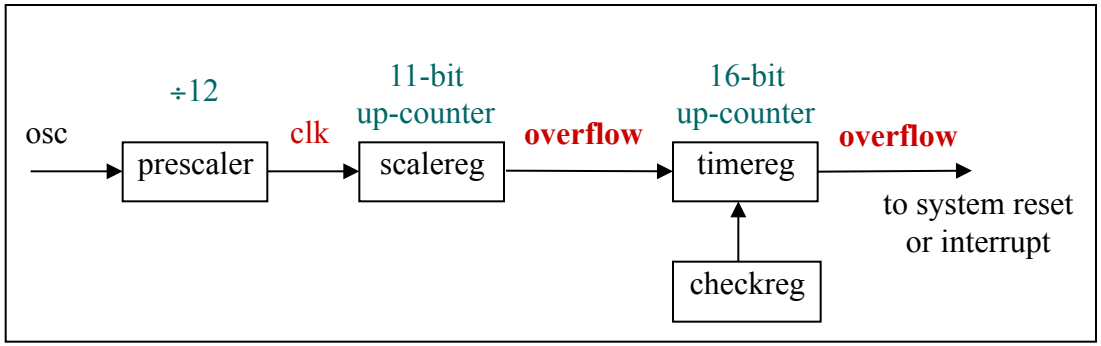


- **Measure time between turning light on and user pushing button**
  - **16-bit timer, clk period is 83.33 ns (12 MHz), counter increments every 6 instruction cycles**
  - **Resolution = 6*83.33=0.5 microsec (too high).**
  - **Range = 65535*0.5 microseconds = 32.77 ms (too low for this application)**
  - **Want program to count 100s of ms, w/o a prescalar, how the extend the range…?**
  - **Initialize timer such that it will overflow in 1 ms, then count the number of overflows!**
    - » **1 ms➔1 ms/(0.5 μs/inst.cycle) ➔2000 inst. cycles**
    - » **so initialize counter to 65535 – 2000 = 63535**
    - » **Counts from 63535 → 65535 in 1 ms ➔ overflow**
    - » **What inaccuracy does this solution have?**

```
/* pseudocode */

#define MS_INIT      63535
void main(void){
  int count_milliseconds = 0;

  configure timer mode
  set Cnt to MS_INIT

  wait a random amount of time
  turn on indicator light
  start timer

while (user has not pushed reaction button){
  if(Top) {
    stop timer
    set Cnt to MS_INIT
    start timer
    reset Top
    count_milliseconds++;
  }
}
turn light off
printf("time:  %i ms", count_milliseconds);
}
```

Watchdog Timer

- **Instead of timer generating a signal every X time units, we must reset timer every X time unit, else timer generates a signal** (time-out!)
- **Under normal operation, we deliberately reset the watch-dog timer every so often.**
- **Timer out is typically connected to the μP interrupt in. Common use: detect failure, self-reset**




```
/* main.c */

main(){
  wait until card inserted
  call watchdog_reset_routine

  while(transaction in progress){
    if(button pressed){
      perform corresponding action
      call watchdog_reset_routine
    }

/* if watchdog_reset_routine not called every
< 2 minutes, interrupt_service_routine is
called */
}
```

```
watchdog_reset_routine(){
/* checkreg is set so we can load value
into timereg.  Zero is loaded into scalereg
and 11070 is loaded into timereg */

  checkreg = 1
  scalereg = 0
  timereg  = 11070
}

void interrupt_service_routine(){
  eject card
  reset screen
}
```

➔ *timereg* range: $2*(2^{16}-1)=131070$ ms

➔ *timereg* = 11,070

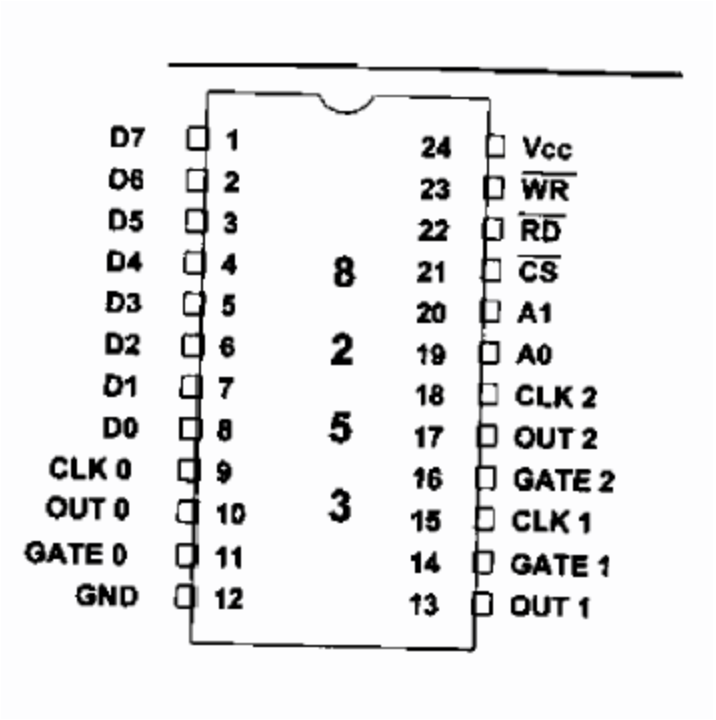
8253/54 Chip

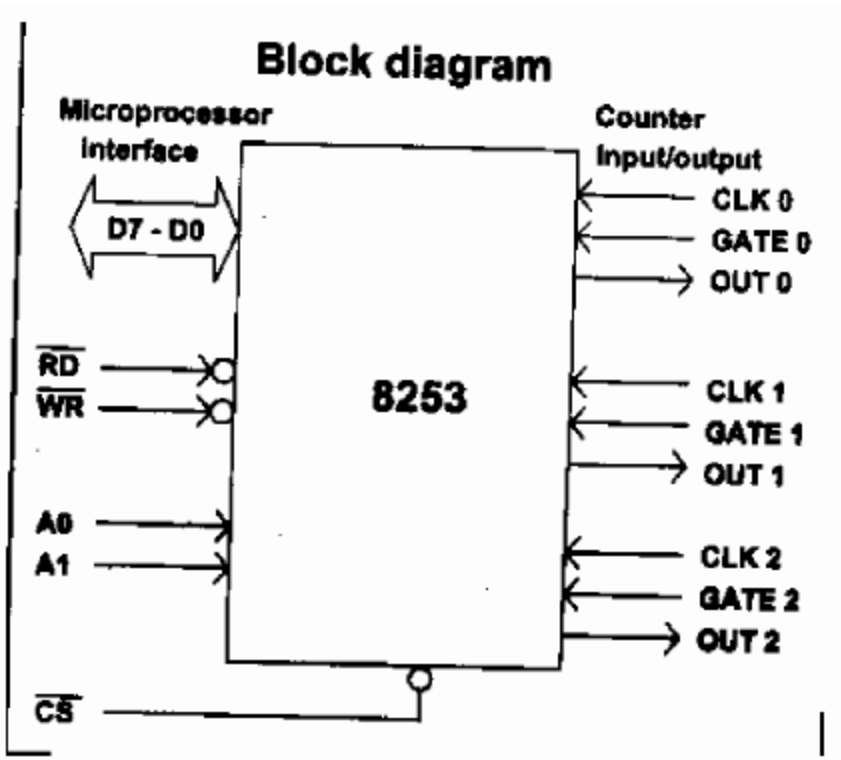
- **Main function:**
  - **Dividing clock frequency**
- **Three counters**
- **Models**
  - **8253: 2 MHz**
  - **8254: 8 MHz**
  - **8254-2: 10 MHz**

Addressing 8253

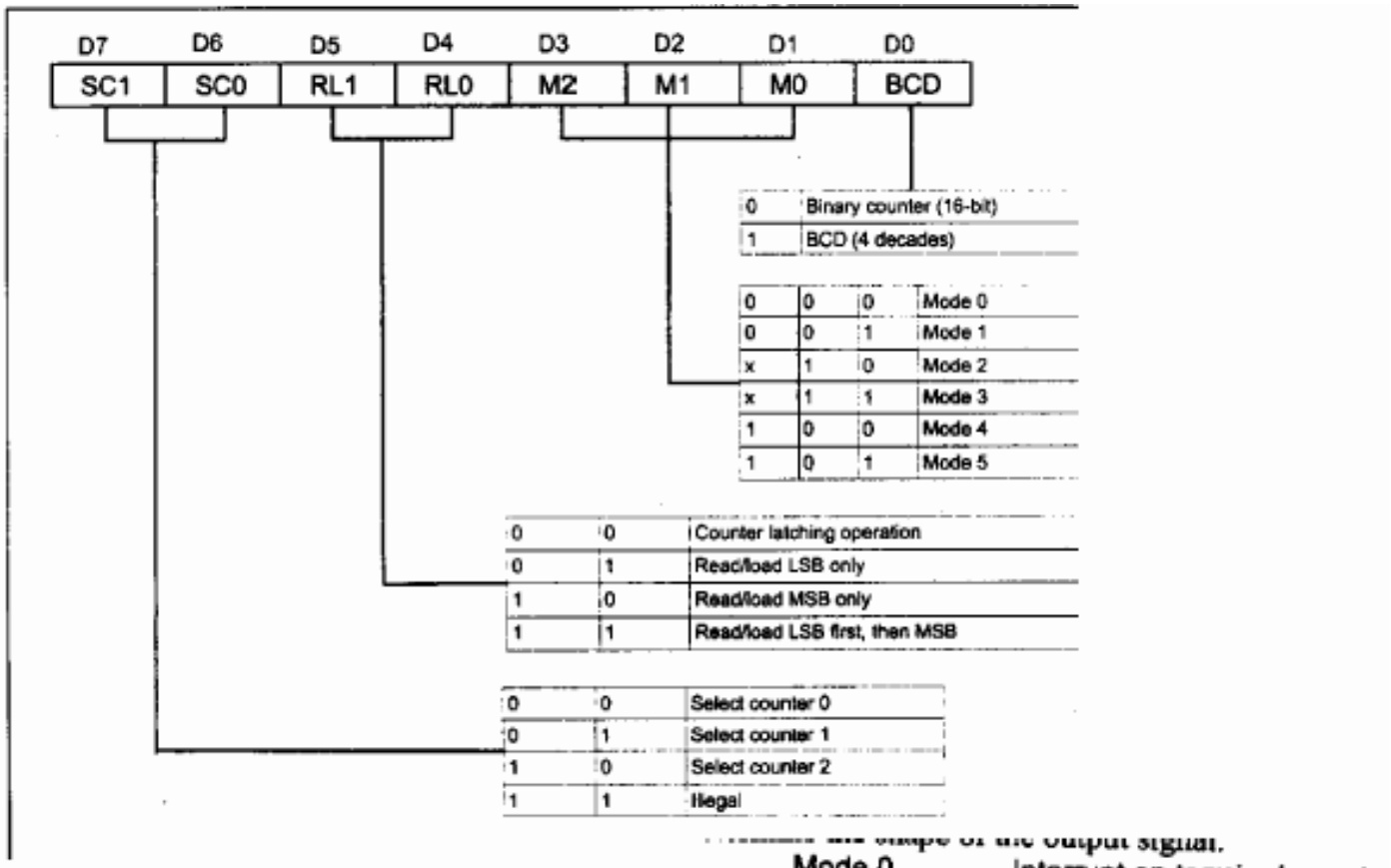
#CS	A1	A0	Port
0	0	0	Counter0
0	0	1	Counter1
0	1	0	Counter2
0	1	1	Control register
1	x	x	Not selected

Pin Description





Control Word



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

Example

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

(a) Find the port addresses assigned to this 8253/54.

(b) Find the configuration for this 8253/54 if the control register is programmed as follows.

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

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:

8253 Decoding in PC

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

Binary Address			Hex Address	Function
$\overline{AEN}$ A9 A8	A7 A6 A5 A4	A3 A2 A1 A0		
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

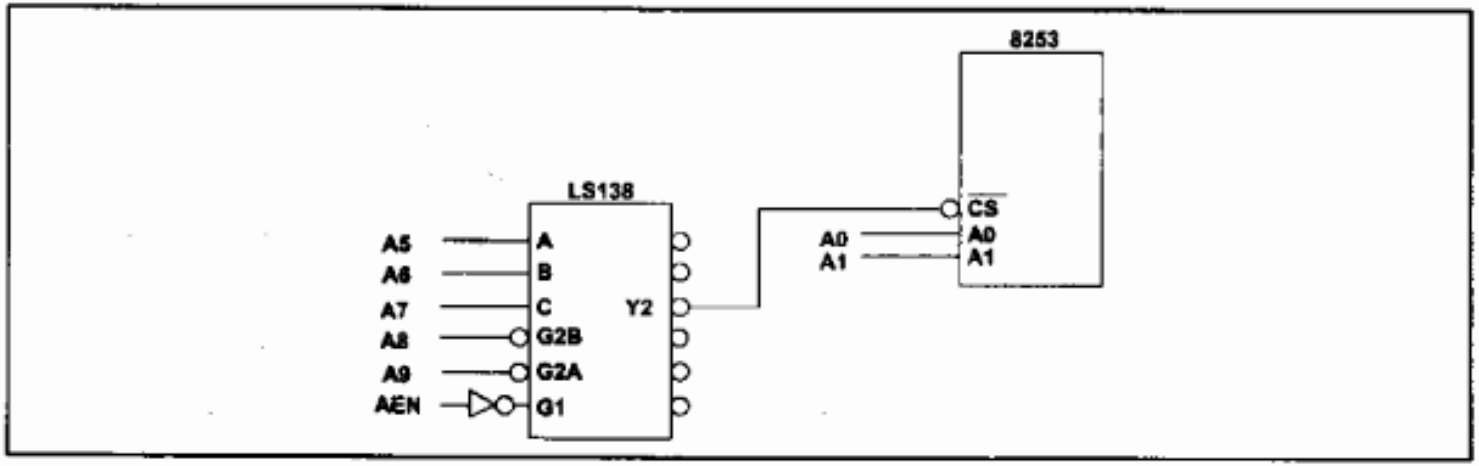


Figure 5-3. 8253 Port Selection in the PC/XT

PC Board

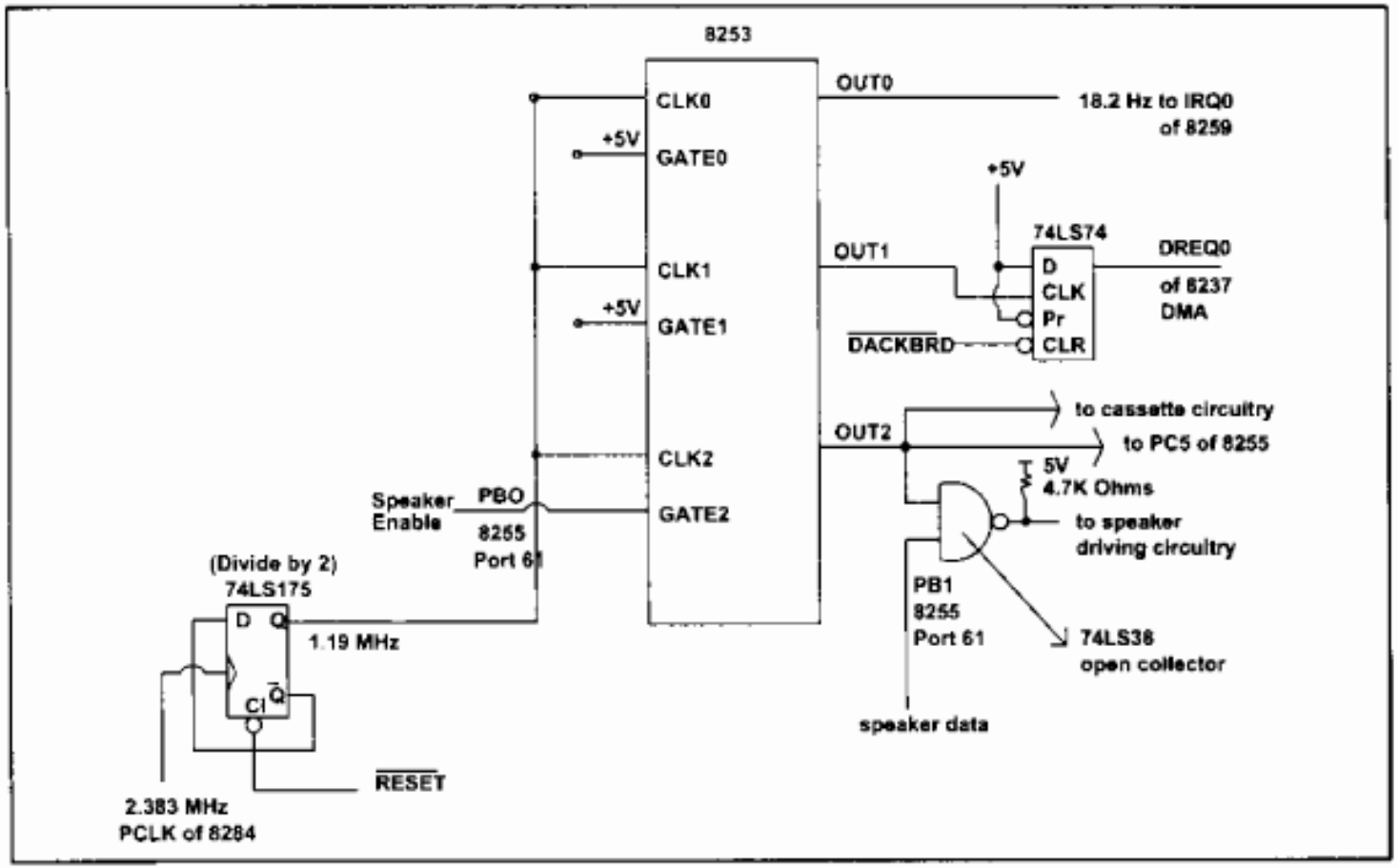


Figure 5-4. 8253 Chip Connections in the PC

Timers in PC

- Counter 0
  - IRQ0 – TOD (time of day)
  - 18.2 Hz (1.193 MHz / 65536)
  - Mode 3, control word: 36H
- Counter 1
  - DRAM refresh – using DMA (at least every .015ms)
  - 2 ms / 128 rows = .015 ms (.015ms = 66278Hz→1.193/18)
  - Mode 2, control word: 54H
- Counter 2
  - Speaker and PC5
  - 896 Hz (1.193MHz / 1331)
  - Mode 3, control word: B6H
  - GATE2 is connected to PB0 (port 61H)

Summarizing the above gives the following control word:

```
D7 D6 D5 D4    D3 D2 D1 D0
 0  0  1  1     0  1  1  0 = 36H
```

The programming of counter 0 is as follows:

```
        MOV   AL,36H        ;control word
        OUT   43H,AL        ;to control register of 8253
        MOV   AL,00         ;00 LSB and MSB of the divisor
        OUT   40H,AL        ;LSB to timer 0
        OUT   40H,AL        ;MSB  to timer 0
```

Time Delay in PC

- **Using software**
  - » **MOV CX, N**
  - » **AGAIN: Loop AGAIN (17 clock cycles)**
  - – **More than N * T(210ns) * 17**
    - » **SUB CX,CX**
    - » **G7: Loop G7 (234ms or better 250ms)**
    - » **DEC BL**
    - » **JNZ G7**
- **Hardware**
  - – **PB4 of port 61H toggle every 15.085 micro**
  - – **Delay.com program**

Music Using Beep

```
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
        RET
DELAY   ENDP
```

```
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
            RET
DELAY_OFF   ENDP
```

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

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

Music Program

D3 note

A3 note

A4 note

```
        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
        OUT   42H,AL            ;the high byte
;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
        OUT   42H,AL            ;the high byte
;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
        OUT   42H,AL            ;the high byte
;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
```

Output Shapes in PC

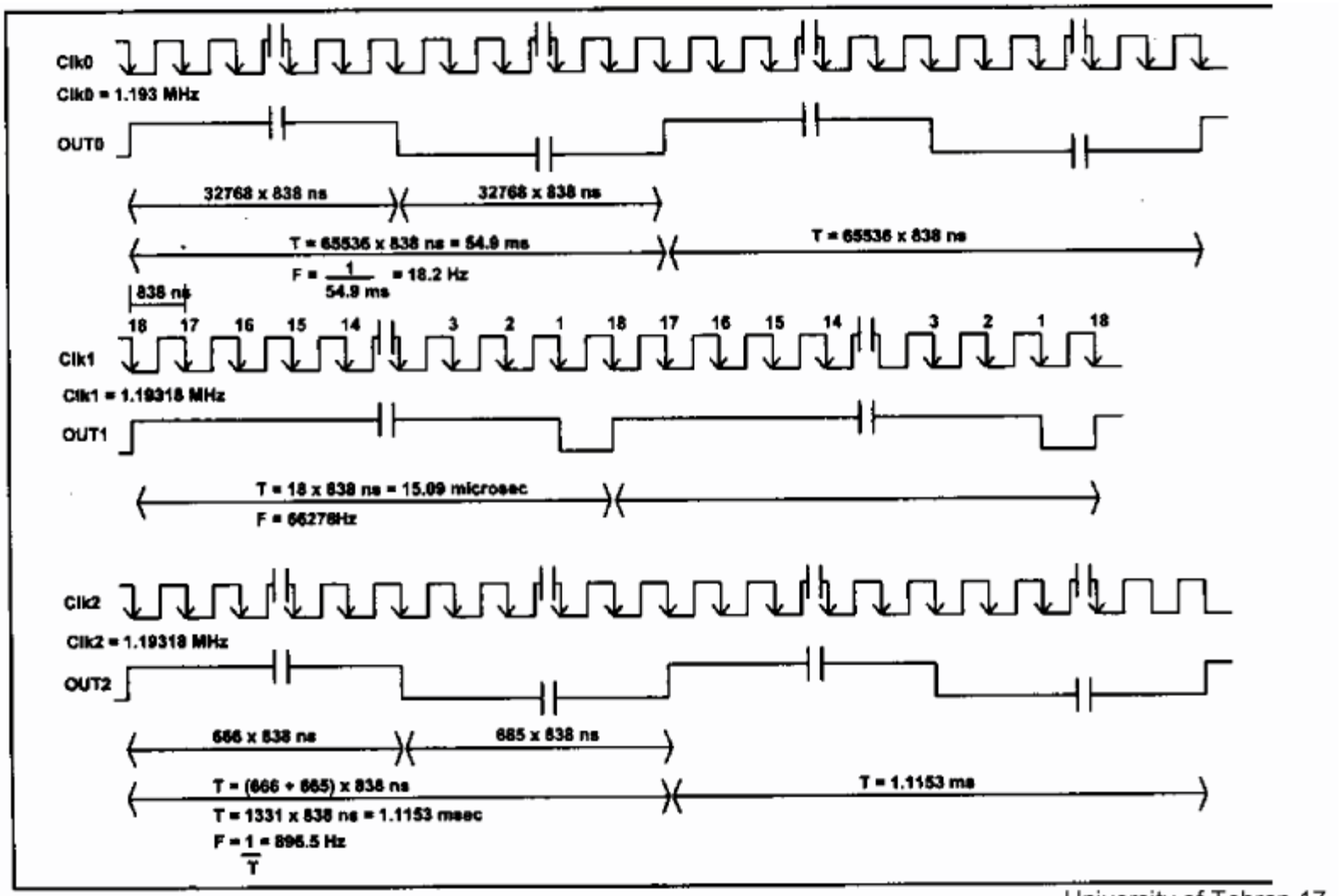


Figure 5-6. 8253/54 Out Timing Diagrams in the PC

hran 20

Mode 0

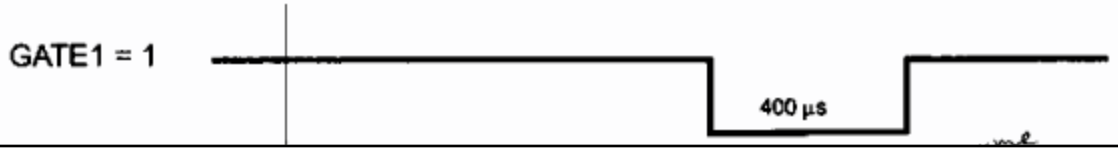
- **Interrupt on terminal count**
- **Low for N*T then high (Remain high until new control word or count number)**

Example 5-8

In Example 5-7, assume that GATE1 becomes zero for 400 µs. What is the width of the low pulse for OUT1?

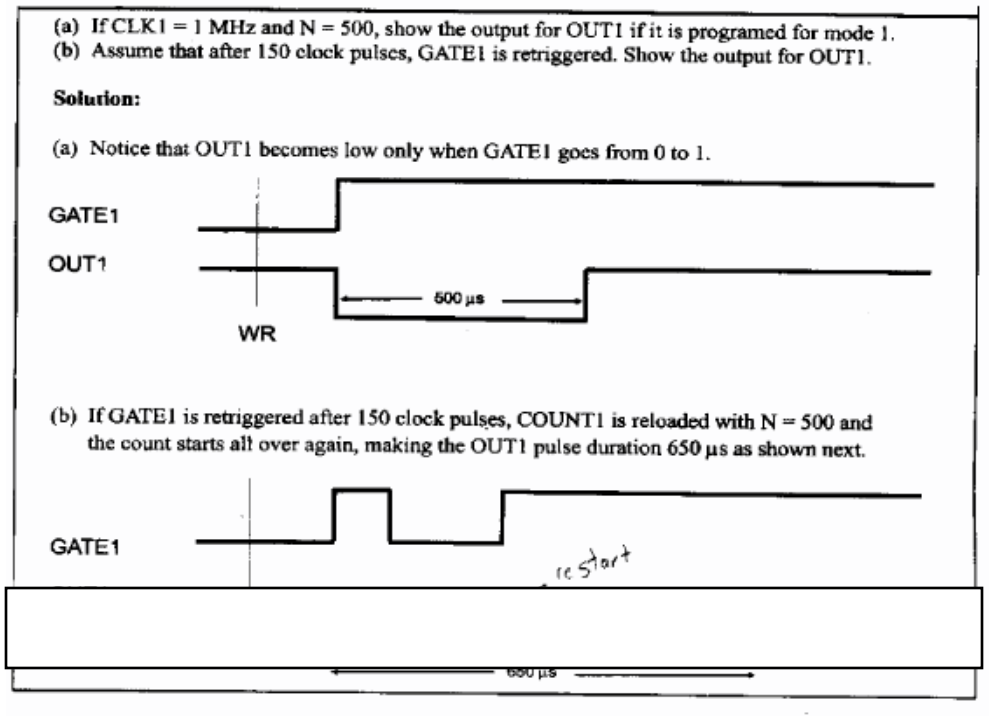
Solution:

It is 1000 µs + 400 µs = 1400 µs, as shown next.



Programmable One Shot (mode 1)

- **Programmable one-shot (hardware triggerable one shot)**
- **0 to 1 on GATE (low for N*T)**



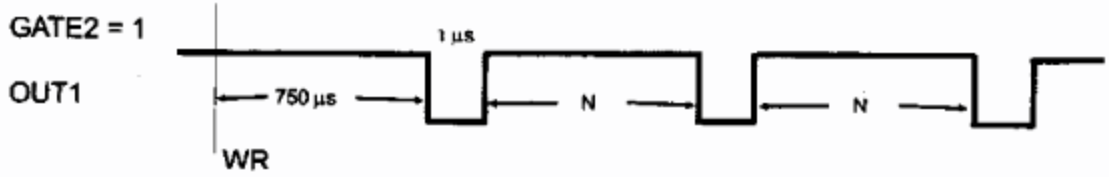
Rate Generator (mode 2)

- **Rate generator (divide by N counter)**
- **High for N*T and low for 1*T**
- **As long as GATE**

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 countinues to produce OUT2.



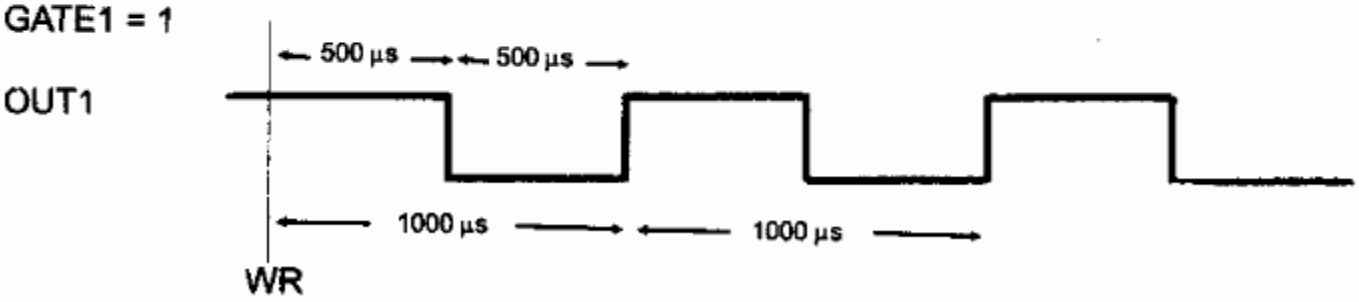
Square Wave (Mode 3)

- **Square wave rate generator**
- **Low N/2 high N/2 ((N+1)/2 if N odd)**

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

Solution:

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



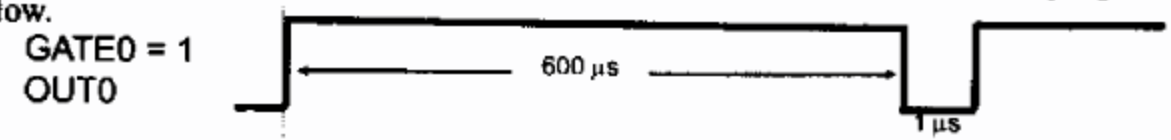
Mode 4

- **Software triggered strobe**
- **Starts upon loading the count**
- **High for N*T low for 1 and then high**

If CLK0 = 1 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 µs, after the count is loaded OUT0 will be high for 600 µs and will go low for 1 µs. Then it will go high again and stay high until the counter is reprogrammed, as shown be low.

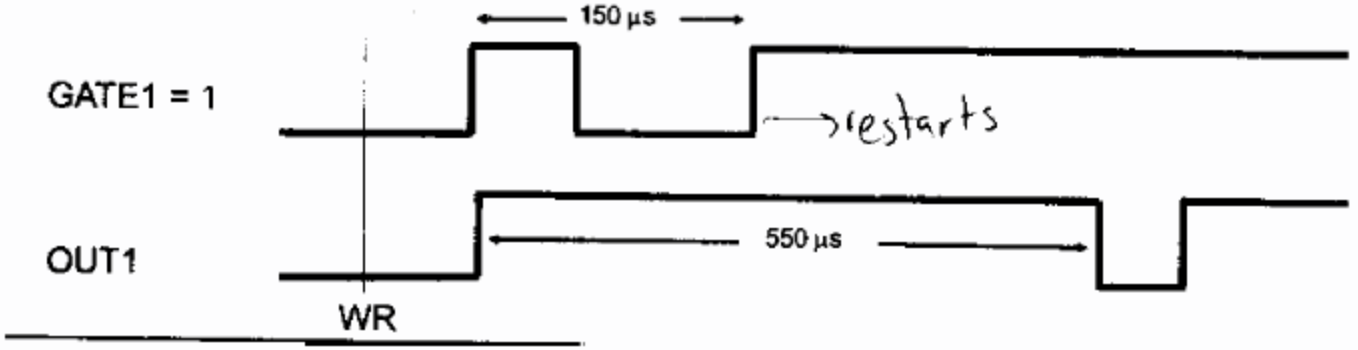


Mode 5

- **Hardware triggered strobe**
- **0 to 1 pulse on GATE**

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 µs, as shown next.



Exercises from Book

- **Text book, Vol. 2 (Page 189-191)**
- **Problems 3, 8, 18, 20, 29**