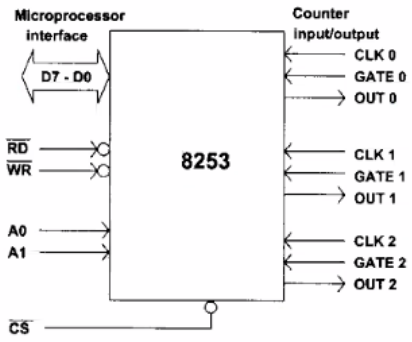
8 8253 PIT Chip

The cons of using software to do timing

- CPU doing useless things - counting, occupy more resources;
- Cannot control the delay accurately.

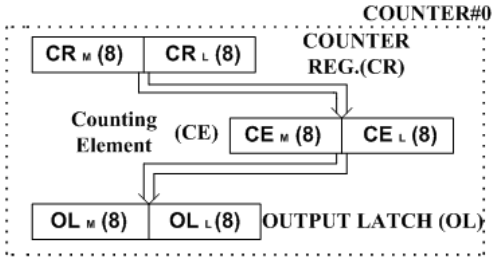
The 8253/54 Programmable Interval Timer is used to generate a lower frequency for various uses. For instance, event counter & accurate time delays.



There are totally three *independent* counters in the 8253 PIT Chip.

Counter:

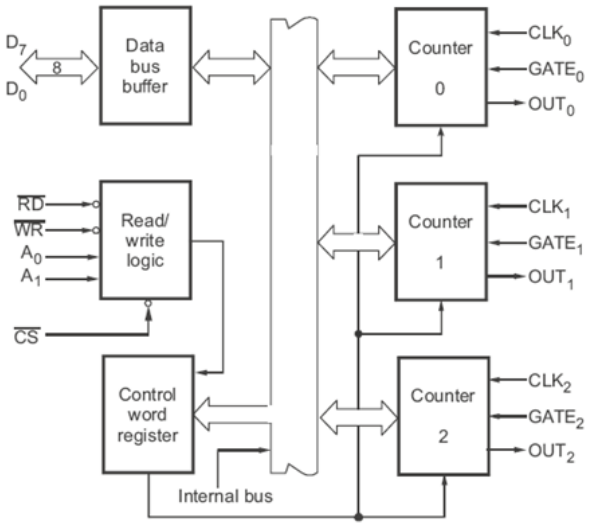
- **Gate** is used to enable (High) or disable (Low) the **counter**;
- **CLK** pin connects to the initial clock signal.
- The new clock signal is generated in **OUT** pin.
- The input frequency can be divided from 1 to 2^{16}.
- Shape of the output frequency:
  - Square-wave （方波）;
  - One-shot;
  - Square-wave with various duty cycles（占空比）.

> [*Example*] An 16-bit down counter.
>
> 
>
>
> There is a **Counting Element** inside each counter.
>
> - A 16-bit count is loaded in the CE;
> - CE begins to decrement until it reaches 0;
> - When it reaches 0, it generate a pulse that can be used to interrupt the CPU.
> - One-shot.
>
> CPU can read the value of the counter from the Output Latch (OL).

Features

- Three independent 16-bit down counters;
- 8253 can operate up to 2.6MHz;
- Three counters are identical and pre-settable, and can be programmed for either binary or BCD count;
- Counter can be programmed in 6 different modes.
- Compatible with all Intel and most other microprocessors.

Internal Structure



- **Data bus buffer**: interface the 8253 to the system data bus; bi-directional, tri-state, 8-bit.
- **Read-Write control Logic**

/CS	/RD	/WR	A1A0	FUNCTION
0	1	0	00	Write counter0 (to CR0)
0	1	0	01	Write counter1 (to CR1)
0	1	0	10	Write counter2 (to CR2)
0	1	0	11	Write control port
0	0	1	00	Read counter0 (from OL0)
0	0	1	01	Read counter1 (from OL1)
0	0	1	10	Read counter2 (from OL2)
0	0	1	11	Read control port (for *8254*)
1	X	X	XX	Not available

  - $\overline{CS}$: Tied to a decoded address;
  - $\overline{RD}$, $\overline{WR}$: In isolate I/O, the signals will be connected with $\overline{IOR}$, $\overline{IOR}$; in memory-mapped I/O, the signals will be connected with $\overline{MEMR}$, $\overline{MEMW}$.
  - A_1, A_0: select the control word register and counters, usually connected to the address lines A_1, A_0 (A_2, A_1 in 8086)

A_1	A_0	Selection
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2
1	1	Control word Register

- **Control Word Register**: Selected when $A_1 = 1, A_0 = 1$, used to specify which counter to be used, its mode, and a read or write operation.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC_1	SC_0	RW_1	RW_0	M_2	M_1	M_0	BCD

SC_1 SC_0 **SC - Select counter**

0	0	Select counter 0
0	1	Select counter 1
1	0	Select counter 2
1	1	Illegal for 8253 Read -Back command for 8254 (See Read operations)

RW_1 RW_0 **RW - Read /Write**

0	0	Counter latch command (See Read operations)
0	1	Read / Write least significant byte only
1	0	Read / Write most significant byte only
1	1	Read / write least significant byte first, then most significant byte

M_2 M_1 M_0 **M - Mode**

0	0	0	Mode 0
0	0	1	Mode 1
x	1	0	Mode 2
x	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

BCD :

0	Binary counter 16 - bits
1	Binary coded decimal (BCD) Counter (4 Decades)

- **Counters**
  - Each consists a single, 16-bit, pre-settable, down counter;
  - Can operate in either binary and BCD;
  - Input, gate and output are configured by the selection of modes;
  - Reading from a counter does not disturb the actual count in process.

Write/Read operations

- Write:
  - Write a control word into control register
  - Load the low-order byte of a count in the counter register
  - Load the high-order byte of a count in the counter register
- Read:
  - **Simple Read**: two I/O read operations, first one for low-order byte and last one for the high order byte. (reading from last latch).
  - **Counter Latch Command**: one I/O write operation used to write a control word to the control register to latch a count in the output latch, then two I/O read operations are used to read the latched count as in Simple Read. (latch and reading from the latest latch)

> [*Example*]

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

(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.

Features of 8253

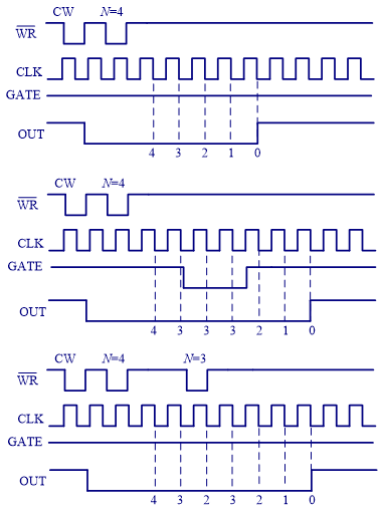
- 8253 takes one CLK pulse to convey the count from `CR` to `CE`.
- `CE` will start count only when `GATE = 1`
  - On every CLK pulse's rising edge (0-to-1), 8253 will check the `GATE`;
  - On every CLK pulse's falling edge (1-to-0), 8253 will count down.

Four main question

- When does the count start?
- Does the count repeat?
- What is the role of the `GATE`?
- What if the counter receive new value in the middle of counting process?

Mode 0: Interrupt on Terminal Count: When loading a new count N, the number of CLK pulses in OUT is $N + 1$. Does not automatically repeat. Only counts when `GATE` is high (if `GATE` is low, then the count will pause).

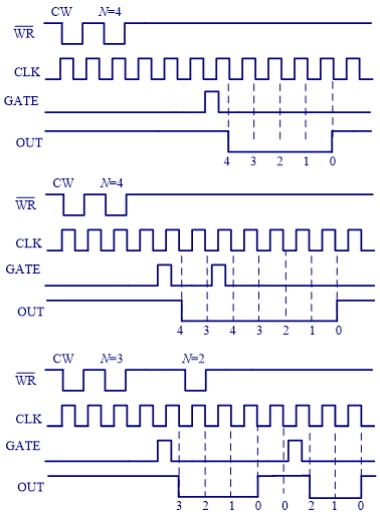
- Once the new initial value is written in the middle of counting cycle, the new value is immediately used to begin a new cycle.



- **Output: N clock pulses low and high afterwards after writing a count**.

Mode 1: Hardware Re-triggerable One-shot: When loading a new count N, the current counting will not be affected, until there is a pulse (actually rising-edge) in `GATE` (trigger). Does not automatically repeat. Re-triggerable means another pulse from the gate will cause the counter to load the initial value again and then counts.

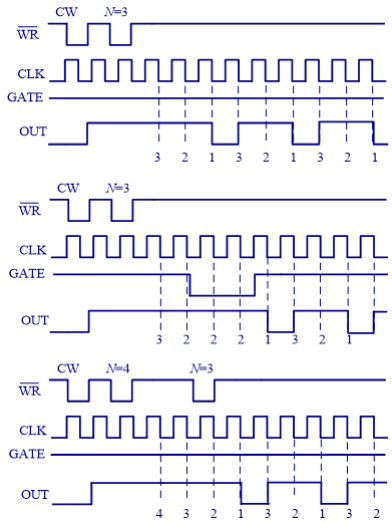
- Writing the new value in the counting process does not impact the current cycle, until the next trigger comes.



- **Output: one-shot of N clock pulses on every trigger.**

Mode 2: Rate Generator: When loading a new count N, the current counting will not be affected, until the end of the current counting. Automatically repeat on terminal count. Only count when `GATE` is high.

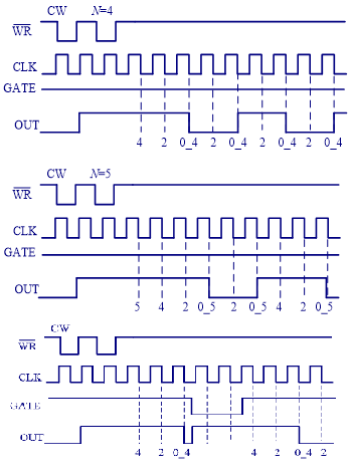
- Writing the new value in the counting process does not impact the current cycle, and the new value will be used in the new counting cycle.



- **Output: periodical signal with a period of $N-1$ clock pulses high and 1 clock pulse low.**

Mode 3: Square Wave Rate Generator: When loading a new count *N*, the current half will not be affected. Automatically repeat on terminal count. If `GATE` goes low while output is low, output is set high immediately. After this, when `GATE` goes high, the counter is loaded with the initial count on the next clock pulse and the sequence is repeated.

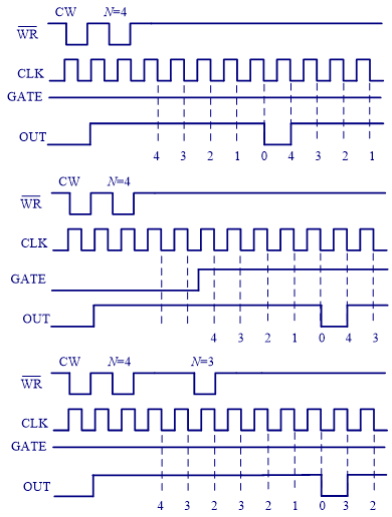
- Writing the new value in the counting process does not impact the current cycle, and the new value will be used in the new counting cycle.



- **Output: if the count is odd, the output will be high for $\frac{n+1}{2}$ clock cycles and low for $\frac{n-1}{2}$ clock cycles; if the count is even, the output will be high and low for $\frac{n}{2}$ clock cycles each**.

Mode 4: Software Triggered Strobe: When loading a new count N, the actual number of CLK pulses in OUT is $N+1$. Automatically repeat. The counter will start immediately after we write the counting to `CE`. Only counts when `GATE` is high.

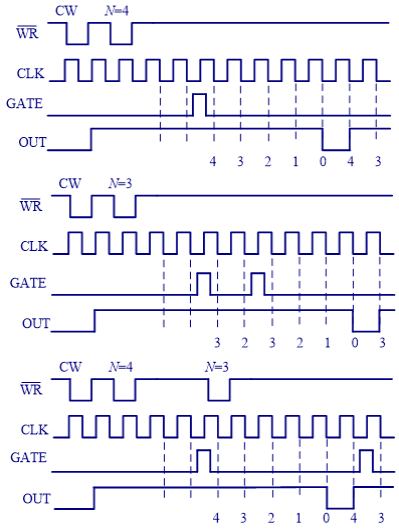
- Once the new initial value is written in the middle of counting cycle, the new value is immediately used to begin a new cycle.



- **Output: periodical signal with a period of $N + 1$ clock pulses low and high afterwards after writing a count**.

Mode 5: Hardware Triggered Strobe: When loading a new count N, the current counting will not be affected, until there is a pulse (actually rising-edge) in `GATE` (trigger). Automatically repeat on the terminal count.

- Writing the new value in the counting process does not impact the current cycle, and the new value won't be used until the next trigger.



Output: periodical signal with a period of $N + 1$ clock pulses low and high afterwards after a trigger (rising edge of `GATE`)