Timer Modes of 8254

♦ Mode 0 --- Interrupt on Terminal Count

1) When this mode is selected **OUT** pin is **initially low**.
2) The **count** value is **loaded**.
3) **GATE** pin is made **high**, so **counting is enabled**.
4) **During counting, OUT** pin remains **low**.
5) **On Terminal Count** (TC) the **OUT** pin goes **high**, and remains high.
6) **During counting** if **GATE** is made **low**, it **disables counting.**
   When **GATE** is made **high**, counting **Resumes**.
   Effect of Gate:
   Low → Disables Counting
   High → Enables (Resumes) Counting

♦ Mode 1 --- Monostable Multivibrator

1) When this mode is selected **OUT** pin is **initially high**.
2) The **count** value is **loaded**.
3) **Counting begins ONLY when** a **rising edge** is applied to the **GATE**.
4) **OUT** pin goes **low** and remains low **during counting**.
5) **On Terminal Count** (TC) the **OUT** pin goes **high**, and remains high.
6) **During counting** if **GATE** is made **low**, it **has no effect** on the Counting.
7) The **GATE pin** can be used as a **Trigger**.
   The **Counter** can be **re-triggered** by applying a **rising edge** on the **GATE**.
   This would **Restart** the **counting**, and hence re-trigger it.
   Effect of Gate:
   Low → No Effect
   High(Trigger) → Starts Counting, can also re-rtigger it.

♦ Mode 2 --- Rate Generator

1) When this mode is selected **OUT** pin is **initially high**.
2) The **count** value is **loaded**.
3) **GATE** pin is made **high**, so **counting is enabled**.
4) **During counting, OUT** pin remains **high**.
5) The OUT pin goes low for one clock cycle just before the TC.
6) The initial count is reloaded and the above process repeats.
   Thus, this mode produces a Continuous Pulse.
7) **During counting** if **GATE** is made **low**, it **disables counting.**
   When **GATE** is made **high**, counting **Restarts**.
   Effect of Gate:
   Low → Disables Counting
   High → Enables (Restarts) Counting
8) It is also called a divide by n counter, as for a count n, the **input frequency is divided** by n to produce the output frequency.

- **Mode 3** ---- Square Wave Generator

1) When this mode is selected **OUT** pin is **initially high.**
2) The **count** value is loaded.
3) **GATE** pin is made high, so **counting** is **enabled.**
4) **OUT** pin remains high for **half of the count** (n/2) and remains **low** for the **remaining half.**
5) **On TC,** the Count is **reloaded** and the **process repeats itself** producing a **continuous square wave.**
6) **During counting** if **GATE** is made **low,** it **disables counting.**
   When **GATE** is made **high,** counting **Restarts.**
   Effect of Gate:
   Low → **Disables** Counting
   High → **Enables** (Restarts) Counting
7) If the **count is ODD,** the **OUT** pin remains **high for (n+1)/2 and low for (n-1)/2.**

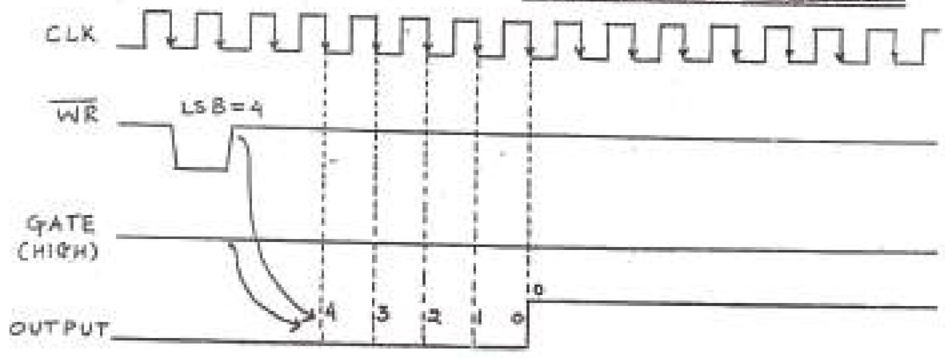
- **Mode 4** --- Software Triggered Strobe

1) When this mode is selected **OUT** pin is **initially high.**
2) The **count** value is **loaded.**
3) **GATE** pin is made **high,** so **counting** is **enabled.**
4) **During counting, OUT** pin remains **high.**
5) The **OUT** pin goes low for one clock **cycle, just after TC.**
6) Aft[illegible] pin goes high and remains high.
7) During counting if **GATE** is made **low,** it **disables counting.**
   When **GATE** is made high, counting **Restarts.**
   Effect of Gate:
   Low → Disables Counting
   High → Enables (Restarts) Counting

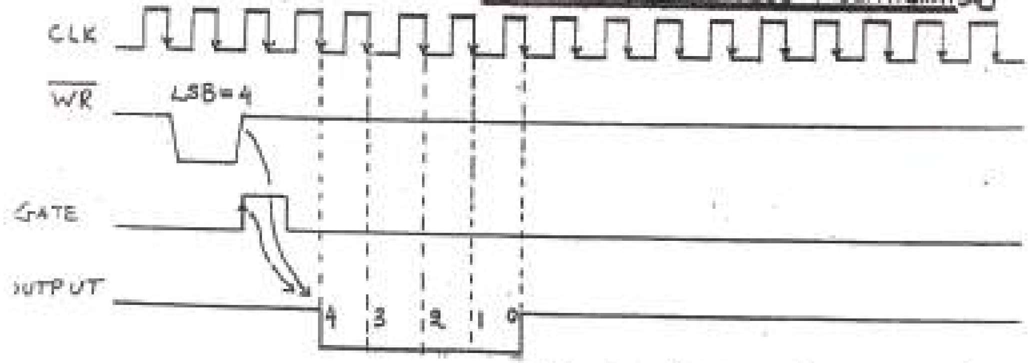
- **Mode 5** --- Hardware Triggered Strobe

1) When this mode is selected **OUT** pin is **initially high.**
2) The **count** value is loaded.
3) Counting starts ONLY after a trigger is applied to the **GATE pin.**
4) Also, the GATE pin need not remain high for the counting to continue.
5) **During counting, OUT** pin remains **high.**
6) The **OUT** pin goes low for one clock **cycle, just after TC.**
7) **After that OUT** pin goes high and remains high.
8) **Thus GATE is used as a** Trigger i.e. it has to be triggered to start counting.
   Effect of Gate:
   Low → No Effect on Counting
   High (Trigger) → Starts Counting, can also re-trigger it.

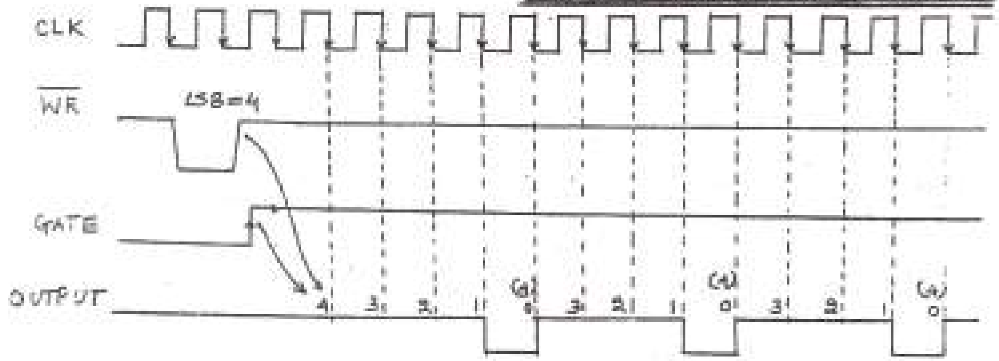
MODE 0 [INTERRUPT ON TC]



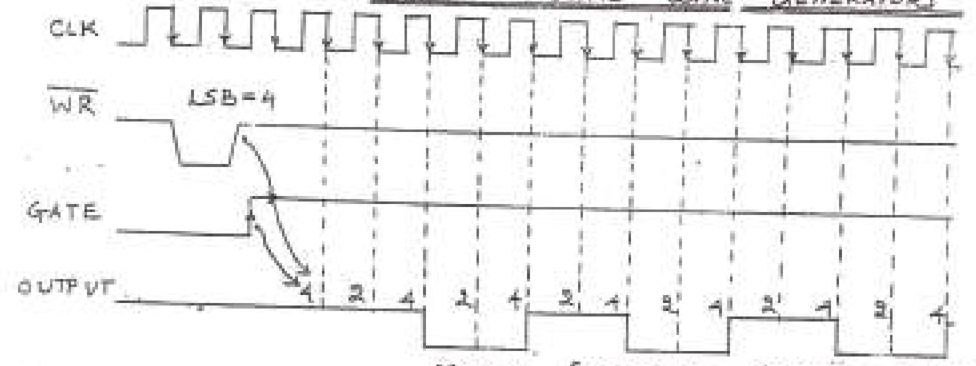
MODE 1 [MONOSTABLE MULTIVIBRATOR]



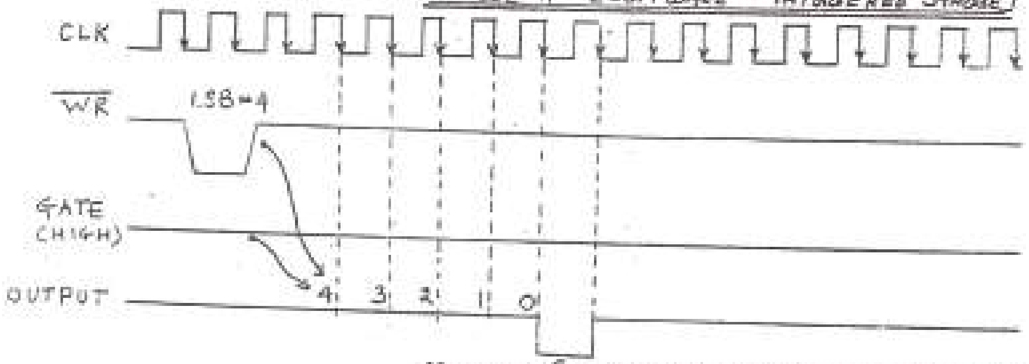
MODE 2 [RATE GENERATOR]



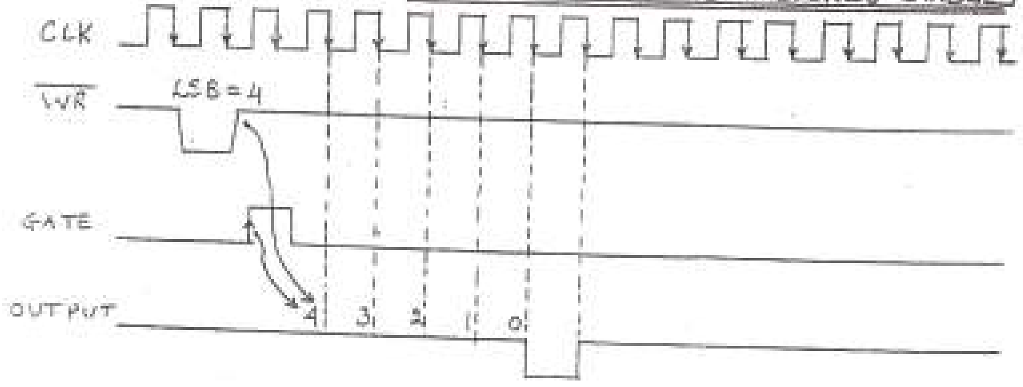
MODE 3. [SQUARE WAVE GENERATOR]



MODE 4 [SOFTWARE TRIGGERED STROBE]



MODE 5 [HARDWARE TRIGGERED STROBE]



IO Data Transfer Schemes

There are **two main schemes** of I/O Data Transfer:

1) µP Controlled Data Transfer (Software Controlled)
2) Device Controlled Data Transfer (Hardware Controlled)

1) µP Controlled Data Transfer (Software Controlled)

- In this kind of data transfer, the **speed of the peripheral device** used for the data transfer **is less then** that of **µP**.
- Thus, there is a **possibility of loss** of data.
- To prevent this loss of data the **µP controls the data transfer.**
- To do so, the µP applies various levels of conditions on the data transfer as follows:

A) Unconditional data transfer

- In this method, **no conditions are applied** on the data transfer.
- The µP **assumes** that the **peripheral is always ready** for the data transfer.
- Though it is the simplest to implement, there is a **high risk of data loss** in this method.

B) Data transfer using READY signal

- The **READY pin** is very important as it can be used to **synchronize** the **data transfer with slower peripherals**.
- **If** the **peripheral** is **NOT** ready for the transfer, **it puts a low on the READY pin.**
- The **µP checks** the READY pin **during every Machine Cycle.**
- **If** the READY pin is **low**, the µP **inserts wait states, until** the READY pin is **high again.**
- Thus, **data loss** can be prevented as the data transfer will not take place until the peripheral is ready.
- This technique is **mainly used between the µP and the memory.**

C) Data transfer using handshake signals.

- This is an advanced method of data transfer, in which **signals** are **exchanged** between the µP and the peripheral **before the actual data transfer** takes place.
- These signals are called **Handshake Signals.**
- These signals **prevent** the µP from **reading duplicate data**, and **sending new data before the previous** one is **accepted** by the receiving device.
- Thus, **reliable** data transfer can be achieved using Handshaking.
  Eg: Data transfer using **8255**.
- This kind of data transfer is implemented in two methods:

Method 1) **Data transfer with Polling (Programmed I/O)**

- Polling is a technique in which, instead of the peripheral informing the µP about the data being available, **the µP checks the status of the data.**
- This is also called Programmed I/O, as the **µP executes a program** in which **it periodically (in a loop) checks** if **the data is available** i.e. the **µP "Polls" for the data.**
- As soon as the **µP finds** that the **peripheral is ready** for the data transfer (i.e. ready to send a new byte, or already received the previous byte), **it performs the data transfer.**
- The **only disadvantage** of this method is that **valuable time of the µP is wasted** in looping **to check** the status of the data.
- **8255** both can perform Programmed I/O.

Method 2) Data transfer with Interrupts (Interrupt Controlled data transfer)

- In this method, the **time of the µP is not wasted** in checking the status of the transfer.
- Here, instead of the µP checking for the data, **the µP** executes its own program, and **is "interrupted" by the peripheral, whenever** the peripheral is **ready** for the data transfer.
- After receiving the interrupt, the **µP branches from its current program into** the **ISR**, in which the **data transfer occurs**.
- After the ISR is executed, the **µP resumes** to its **own program**.
- Thus, no time of the µP is wasted.
- **8255** can perform Interrupt Controlled data transfer.
- An **8259 is generally used** as it **increases** the **number of devices**, which can interrupt the µP.

2) Device Controlled Data Transfer (Hardware Controlled)

- In this method, an **external device** such as the **DMA Controller controls the data transfer**.
- It is mainly suitable for **high-speed data transfer** (Eg: between I/O and Memory), where the **peripheral is faster than the µP**.
- In Program Controlled I/O, Status or Interrupt driven I/O the speed of transfer is **slow** mainly **because instructions** need to be **decoded** and **then executed** for the transfer.
- **DMA trahsfer** is **software independent** and **hence much faster**.
- Here the **DMAC requests** the µP to **release the system bus**.
- Once the µP does so, the **DMAC gets the system bus** and **performs high-speed data transfer** that does not involve the µP, and hence is very fast.
- **Once** the **data transfer** is **over** the **system bus is returned to the µP**.

Eg: DMA transfer can be achieved using **8237 DMAC**.

Types of I/O mapping techniques

	Memory Mapped I/O	I/O Mapped I/O
1)	The I/O devices are **mapped into the memory**	The I/O devices are **mapped independent of the memory**.
2)	The **size of the I/O address** is the **same as** the size of the **memory address**	The **size of the I/O address** is **independent of** the **memory address**.
3)	**Only MEMR, MEMW** signals are required	**IOR, IOW, MEMR, MEMW** signals are required.
4)	**Data transfer** can take place **between any register and** an **I/O device**	Data transfer can take place **only between the Accumulator** and an I/O device.
5)	**Execution** is **slower** as **access time is more**	Execution is **faster** as **access time is less**.
6)	**Decoding 20-bit** address requires **more hardware**	Decoding **16-bit** address requires **less hardware**
7)	**Increases** the **number of I/O ports** interfaced. Eg: 2^{20}=1M **in 8086**	**Restricts** the number of I/O Ports interfaced. Eg: **2^{16} = 64K in 8086**.
8)	Most of the **instructions** like **MOV, ADD, SUB** etc can be used to access the I/O devices	**Only IN** and **OUT** instructions can be used to access the I/O devices.