Electronic Counters

An electronic counter is a flip flop circuit capable of counting the number of clock pulses applied to it. Counters are formed by connecting flip-flops together and any number of flip-flops can be connected or "cascaded" together to form a "divide-by-n" binary counter where "n" is the number of counter stages used and which is called the **Modulus**. The modulus or simply "MOD" of a counter is the number of output states the counter goes through before returning itself back to zero, i.e., one complete cycle.

Then a counter with three flip-flops like the circuit above will count from 0 to 7 i.e., 2^n -1. It has eight different output states representing the decimal numbers 0 to 7 and is called a **Modulo-8** or **MOD-8** counter. A counter with four flip-flops will count from 0 to 15 and is therefore called a **Modulo-16** counter and so on.

An example of this is given as,

3-bit Binary Counter = $2^3 = 8$ (modulo-8 or MOD-8)

4-bit Binary Counter = $2^4 = 16$ (modulo-16 or MOD-16)

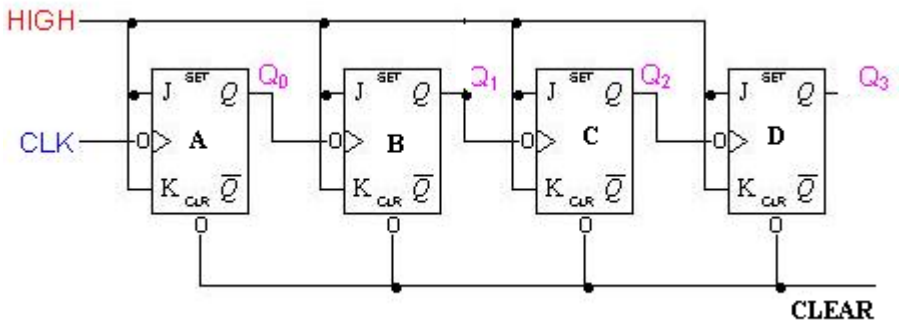
8-bit Binary Counter = $2^8 = 256$ (modulo-256 or MOD-256) and so on..

The Modulo number can be increased by adding more flip-flops to the counter and cascading is a method of achieving higher modulus counters. Then the modulo or MOD number can simply be written as:

MOD number = 2^n

Four bit binary counter (MOD – 16 counter or Ripple counter or UP counter) -

1. In this digital counter circuit set up, all flip flops are clocked by the Q outputs of previous flip flops. Thus it is called as an asynchronous counter.
2. All the J-K flip flops holds their J and K inputs at logic '1'.
3. A ripple counter comprising of N flip flops can be used to count up to 2^N pulses. That is, a circuit with 4 flip flops gives a maximum count 2^4=16.
4. This counter gives a natural binary count from 0 to 15 and resets to initial condition on the 16^{th} input pulse.
5. Since J and K inputs of all the flip flops are connected to logic '1', they act as a toggle flip flop.
6. Here the output of each flip flop toggles at the negative transition of clock input.



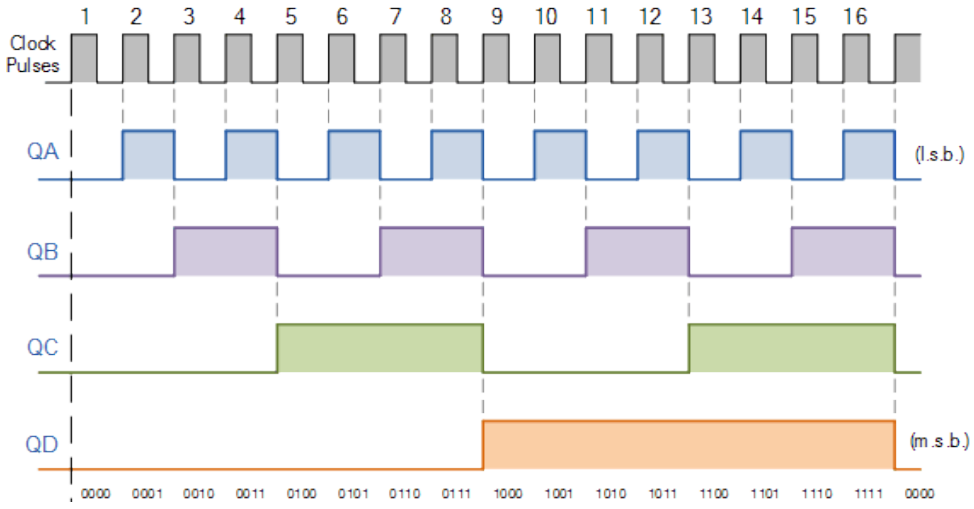
With the application of 1st clock pulse, Q_0 changes from 0 to 1, whereas Q_1, Q_2 and Q_3 remain unaffected. The output of counter $Q_3Q_2Q_1Q_0 = 0001$

On 2nd clock pulse, the flip flop A will change its state from 1 to 0 which is negative edge therefore the flip flop B will change its state from 0 to 1. The output of counter $Q_3Q_2Q_1Q_0 = 0010$. In this way each flip flop will change its state at the negative edge of clock pulse. The counter is incremented upto 1111 and at 17^{th} clock pulse it returns to its initial state 0000.

All flip flops do not trigger at the same time. Trigger flows through the flip flop like ripple on water.

Waveforms

1. First draw 16 input pulses
2. Mark negative edge of each pulse by arrow.
3. Change the state 0 to 1 and 1 to 0 at each negative edge, which is output of A flip flop.
4. Mark negative of A flip flop output with arrow and draw waveforms for flip flop B and then for flip flop C and D.

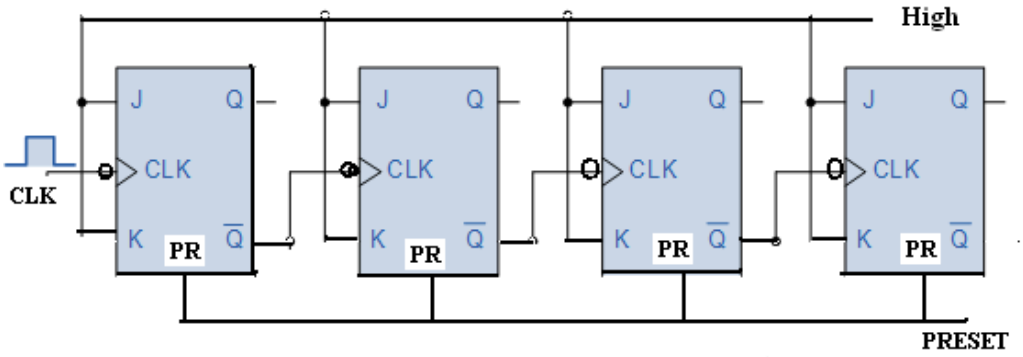


CLK	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1
16	0	0	0	0 (RESET)
17	0	0	0	1 (RECOUNTING)

Down Counter

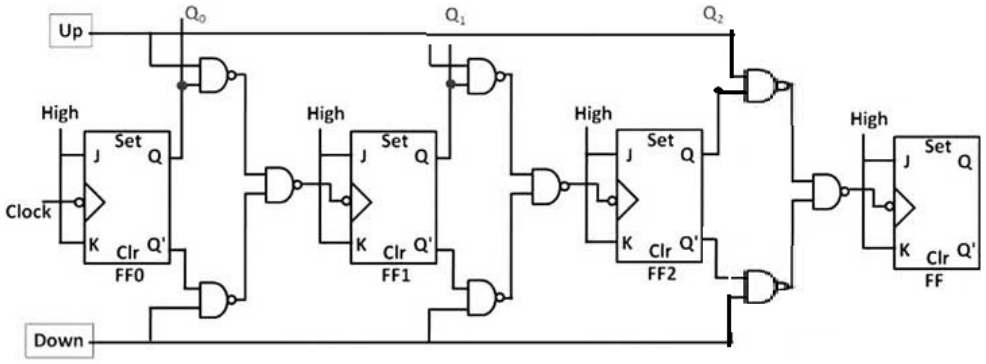
It is sometimes necessary to count "down" from a predetermined value to zero and to produce an output that activates when the zero count or other pre-set value is reached. This type of counter is normally referred to as a Down Counter.

In a Binary or BCD down counter, the count decreases by one for each external clock pulse from some preset value. In the 4-bit counter above the output of each flip-flop changes state on the falling edge (1-to-0 transition) of the CLK input which is triggered by the $\vec{Q}$ output of the previous flip-flop, rather than by the Q output as in the up counter configuration. As a result, each flip-flop will change state when the previous one changes from 0 to 1 at its output, instead of changing from 1 to 0.



UP / DOWN counter

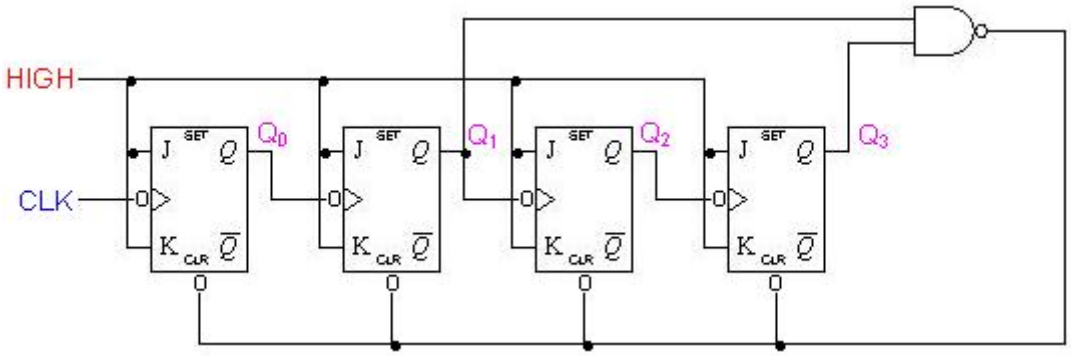
The circuit is of a simple 4-bit Up/Down synchronous counter using JK flip-flops configured to operate as toggle or T-type flip-flops giving a maximum count of zero (0000) to fifteen (1111) and back to zero again. Then the 4-Bit counter advances upward in sequence (0,1,2,3,4,5,6,7,….,15) or downwards in reverse sequence (15,14,13, …….,0).



When *Up = 1*, it connects $Q_0Q_1Q_2Q_3$ to the clock inputs through gates and generates incrementing output, it works as up counter. When *Down = 1*, it connects $\bar{Q}_0\bar{Q}_1\bar{Q}_2\bar{Q}_3$ to the clock inputs through gates and generates decrementing output, it works as down counter.

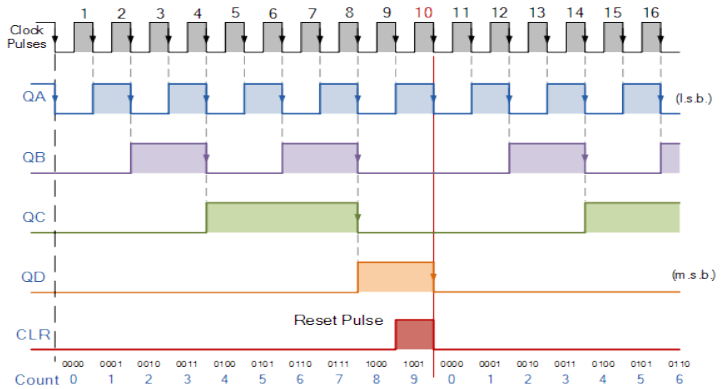
Decade counter

A decade counter requires resetting to zero when the output count reaches the decimal value of 10, ie. When, DCBA = 1010 and to do this we need to feed this condition back to the reset input. A counter with a count sequence from binary "0000" (BCD = "0") through to "1001" (BCD = "9") is generally referred to as a *BCD binary-coded-decimal counter* because its ten state sequence is that of a BCD code but binary decade counters are more common.



This type of asynchronous counter counts upwards on each trailing edge of the input clock signal starting from 0000 until it reaches an output 1001 (decimal 9). Both outputs Q_A and Q_D are now equal to logic "1". One the application of the next clock pulse, the output from the NAND gate changes state from logic "1" to a logic "0" level.

As the output of the NAND gate is connected to the CLEAR (CLR) inputs of all the J-K Flip-flops, this signal causes all of the Q outputs to be reset back to binary 0000 on the count of 10. As outputs Q_A and Q_D are now both equal to logic "0" as the flip-flop's have just been reset, the output of the NAND gate returns back to a logic level "1" and the counter restarts again from 0000. We now have a decade or Modulo-10 up-counter.

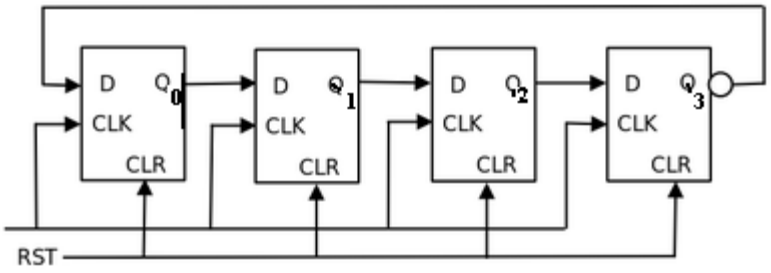


Clock Count	Output bit Pattern				Decimal Value
	Q_D	Q_C	Q_B	Q_A	
1	0	0	0	0	0
2	0	0	0	1	1
3	0	0	1	0	2
4	0	0	1	1	3
5	0	1	0	0	4
6	0	1	0	1	5

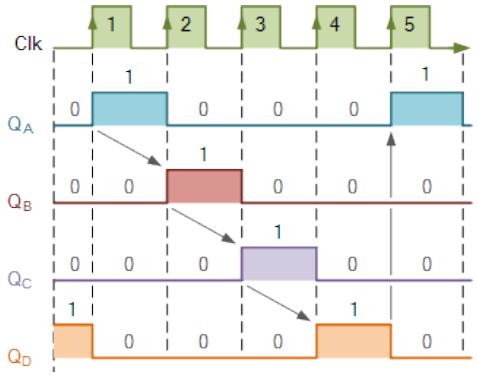
7	0	1	1	0	6
8	0	1	1	1	7
9	1	0	0	0	8
10	1	0	0	1	9
11	**Counter Resets its Outputs back to Zero**				

Ring Counter

- This counter is a circulated shift register. Here a 4 bit shift register is implemented with D flip flops. This is a synchronous counter because clock signal is common for all flip flops.
- The 'Q' output of first flip flop is directly fed to the 'D' input of following flip flop unit.
- The final 'Q' output from the last flip flop is connected to the 'D' input of 1 st unit; hence it is called as a circulating shift register.
- 'Preset' pin of 1st unit is connected to V_{cc} via a switch (Push to OFF switch). Preset pin is used to set the output of a flip flop to logical high (1) forcefully.
- The 'Presets' of other flip flops are directly coupled to V_{cc} in order to work properly because these pins are active low logic pins (This connection is not shown in above circuit. For simplicity it is removed). But don't forget to connect it while you are implementing practically.
- Similarly 'Clear' pins are also connected to V_{cc} due to the above said reason.
- We are already familiar with shift registers that for each clock pulse the data will be shifted to next register (flip flop).
- Initially all the outputs of flip flops are at low, to start ring counter you need to set the output of first flip flop manually high.
- For this purpose you just push the switch (SW2), and then the output of first flip flop goes high.
- This output will be shifted to the next flip flop during the next clock pulse and so on. At the same time the low output from last flip flop is shifted to the input of 1st flip flop.



Clock Input	Q3	Q2	Q1	Q0
1	0	0	0	1
2	0	0	1	0
3	0	1	0	0
4	1	0	0	0
5	0	0	0	1
6	0	0	1	0
7	0	1	0	0
8	1	0	0	0



Shift Register

This sequential device loads the data present on its inputs and then moves or "shifts" it to its output once every clock cycle, hence the name **Shift Register**.

A *shift register* basically consists of several single bit "D-Type Data Latches", one for each data bit, either a logic "0" or a "1", connected together in a serial type daisy-chain arrangement so that the output from one data latch becomes the input of the next latch and so on.

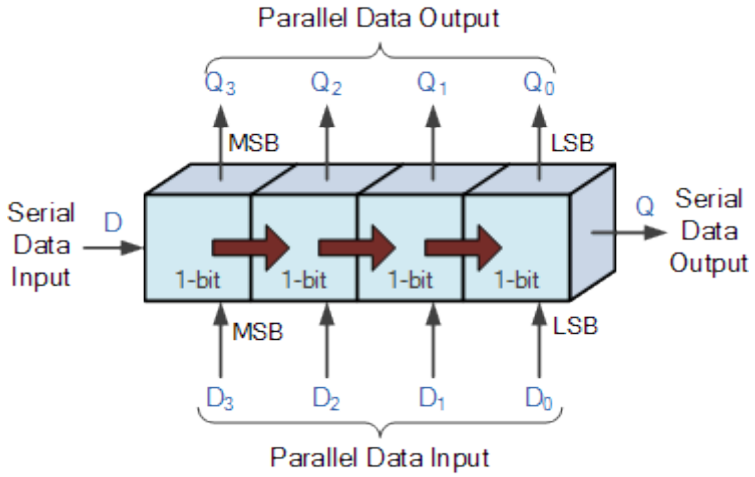
Data bits may be fed in or out of a shift register serially, that is one after the other from either the left or the right direction, or all together at the same time in a parallel configuration.

The number of individual data latches required to make up a single **Shift Register** device is usually determined by the number of bits to be stored with the most common being 8-bits (one byte) wide constructed from eight individual data latches.

***Shift Registers* are used** for data storage or for the movement of data and are therefore commonly used inside calculators or computers to store data such as two binary numbers before they are added together, or to convert the data from either a serial to parallel or parallel to serial format. They are also used in delay generation in data transmission.

The individual data latches that make up a single shift register are all driven by a common clock (Clk) signal making them synchronous devices. Generally, shift registers operate in one of four different modes with the basic movement of data through a shift register being:

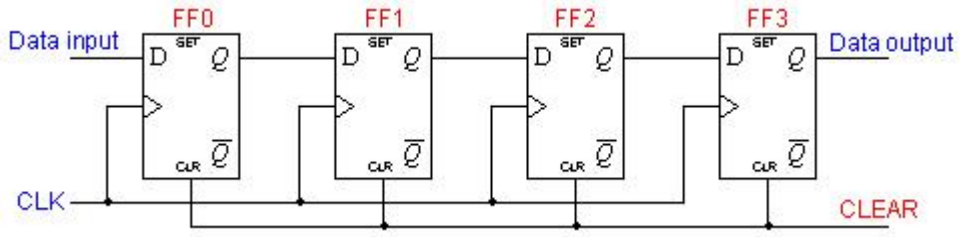
- **Serial-in to Parallel-out (SIPO) -** The register is loaded with serial data, one bit at a time, with the stored data being available at the output in parallel form.
- **Serial-in to Serial-out (SISO) -** The data is shifted serially "IN" and "OUT" of the register, one bit at a time in either a left or right direction under clock control.
- **Parallel-in to Serial-out (PISO) -** The parallel data is loaded into the register simultaneously and is shifted out of the register serially one bit at a time under clock control.
- **Parallel-in to parallel-out (PIPO) -** The parallel data is loaded simultaneously into the register, and transferred together to their respective outputs by the same clock pulse.



Right shift register

Shift register is prepared by cascading four D flip flops. As Data input sets up left flip flop, Q_3 sets up Q_2, Q_2 sets up Q_1 and so on. The flip flops are positive edge triggered type so at the positive edge of clock pulse data input moves one bit left.

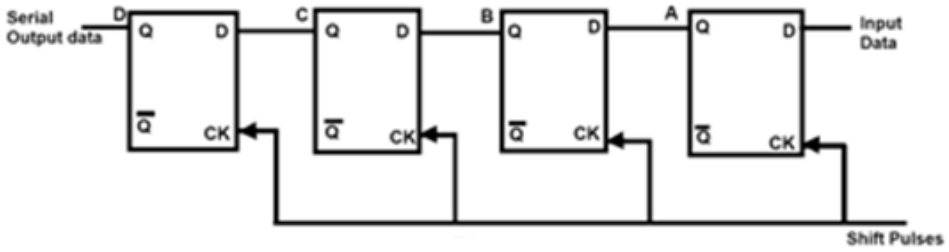
When Din = 1 then at the positive edge of first clock pulse it is shifted to right by first D flip flop and other remains as it is and output is 1000. At second clock pulse Q_2 is transferred to its left and output is 1100 by second D flip flop and so on.



Left shift register

Shift register is prepared by cascading four D flip flops. As Data input sets up right flip flop, Q_0 sets up Q_1, Q_1 sets up Q_2 and so on. The flip flops are positive edge triggered type so at the positive edge of clock pulse data input moves one bit left.

When Din = 1 then at the positive edge of first clock pulse it is shifted to left by first D flip flop and other remains as it is and output is 0001. At second clock pulse Q_0 is transferred to its left and output is 0011 by second D flip flop and so on.



- ✓ A simple **Shift Register** can be made using only D-type flip-Flops, one flip-Flop for each data bit.
- ✓ The output from each flip-Flop is connected to the D input of the flip-flop at its right.
- ✓ Shift registers hold the data in their memory which is moved or "shifted" to their required positions on each clock pulse.
- ✓ Each clock pulse shifts the contents of the register one bit position to either the left or the right.
- ✓ The data bits can be loaded one bit at a time in a series input (SI) configuration or be loaded simultaneously in a parallel configuration (PI).
- ✓ Data may be removed from the register one bit at a time for a series output (SO) or removed all at the same time from a parallel output (PO).
- ✓ One application of shift registers is in the conversion of data between serial and parallel, or parallel to serial.
- ✓ Shift registers are identified individually as SIPO, SISO, PISO, PIPO, or as a Universal Shift Register with all the functions combined within a single device.