http://www.electronics-tutorials.ws

**Sequential Logic Basics თანმიმდევრობითი ლოგიკის საფუძვლები**

Unlike [**Combinational Logic**](http://www.electronics-tutorials.ws/combination/comb_1.html) circuits that change state depending upon the actual signals being applied to their inputs at that time, **Sequential Logic** circuits have some form of inherent "Memory" built in to them as they are able to take into account their previous input state as well as those actually present, a sort of "before" and "after" is involved with sequential circuits.

In other words, the output state of a sequential logic circuit is a function of the following three states, the "present input", the "past input" and/or the "past output". *Sequential Logic circuits* remember these conditions and stay fixed in their current state until the next clock signal changes one of the states, giving sequential logic circuits "Memory".

Sequential logic circuits are generally termed as *two state* or [**Bistable**](http://www.electronics-tutorials.ws/waveforms/bistable.html) devices which can have their output or outputs set in one of two basic states, a logic level "1" or a logic level "0" and will remain "latched" (hence the name latch) indefinitely in this current state or condition until some other input trigger pulse or signal is applied which will cause the bistable to change its state once again.

**Sequential Logic Representation**

|  |
| --- |
| Sequential Logic |

The word "Sequential" means that things happen in a "sequence", one after another and in **Sequential Logic** circuits, the actual clock signal determines when things will happen next. Simple sequential logic circuits can be constructed from standard **Bistable** circuits such as Flip-flops, Latches and Counters and which themselves can be made by simply connecting together universal [**NAND Gates**](http://www.electronics-tutorials.ws/logic/logic_5.html) and/or [**NOR Gates**](http://www.electronics-tutorials.ws/logic/logic_5.html) in a particular combinational way to produce the required sequential circuit.

**Classification of Sequential Logic**

As standard logic gates are the building blocks of combinational circuits, bistable latches and flip-flops are the building blocks of Sequential Logic Circuits. Sequential logic circuits can be constructed to produce either simple edge-triggered flip-flops or more complex sequential circuits such as storage registers, shift registers, memory devices or counters. Either way sequential logic circuits can be divided into the following three main categories:

* 1. Event Driven - asynchronous circuits that change state immediately when enabled.
* 2. Clock Driven - synchronous circuits that are synchronised to a specific clock signal.
* 3. Pulse Driven - which is a combination of the two that responds to triggering pulses.

|  |
| --- |
| Types of Sequential Logic |

As well as the two logic states mentioned above logic level "1" and logic level "0", a third element is introduced that separates **sequential logic** circuits from their **combinational logic** counterparts, namely *TIME*. Sequential logic circuits that return back to their original state once reset, i.e. circuits with loops or feedback paths are said to be "cyclic" in nature.

We now know that in sequential circuits changes occur only on the application of a clock signal making it synchronous, otherwise the circuit is asynchronous and depends upon an external input. To retain their current state, sequential circuits rely on feedback and this occurs when a fraction of the output is fed back to the input and this is demonstrated as:

**Sequential Feedback Loop**



The two inverters or NOT gates are connected in series with the output at Q fed back to the input. Unfortunately, this configuration never changes state because the output will always be the same, either a "1" or a "0", it is permanently set. However, we can see how feedback works by examining the most basic sequential logic components, called the SR flip-flop.

**SR Flip-Flop**

The **SR flip-flop**, also known as a *SR Latch*, can be considered as one of the most basic sequential logic circuit possible. This simple flip-flop is basically a one-bit memory bistable device that has two inputs, one which will "SET" the device (meaning the output = "1"), and is labelled S and another which will "RESET" the device (meaning the output = "0"), labelled R. Then the SR description stands for "Set-Reset". The reset input resets the flip-flop back to its original state with an output Q that will be either at a logic level "1" or logic "0" depending upon this set/reset condition.

A basic NAND gate SR flip-flop circuit provides feedback from both of its outputs back to its opposing inputs and is commonly used in memory circuits to store a single data bit. Then the SR flip-flop actually has three inputs, Set, Reset and its current output Q relating to it's current state or history. The term "Flip-flop" relates to the actual operation of the device, as it can be "flipped" into one logic Set state or "flopped" back into the opposing logic Reset state.

**The NAND Gate SR Flip-Flop**

The simplest way to make any basic single bit set-reset SR flip-flop is to connect together a pair of cross-coupled 2-input NAND gates as shown, to form a Set-Reset Bistable also known as an active LOW SR NAND Gate Latch, so that there is feedback from each output to one of the other NAND gate inputs. This device consists of two inputs, one called the *Set*, S and the other called the *Reset*, R with two corresponding outputs Q and its inverse or complement Q (not-Q) as shown below.

**The Basic SR Flip-flop**

|  |
| --- |
| The SR Flip flop |

**The Set State**

Consider the circuit shown above. If the input R is at logic level "0" (R = 0) and input S is at logic level "1" (S = 1), the NAND gate *Y* has at least one of its inputs at logic "0" therefore, its output Q must be at a logic level "1" (NAND Gate principles). Output Q is also fed back to input "A" and so both inputs to NAND gate *X* are at logic level "1", and therefore its output Q must be at logic level "0". Again NAND gate principals. If the reset input R changes state, and goes HIGH to logic "1" with S remaining HIGH also at logic level "1", NAND gate *Y* inputs are now R = "1" and B = "0". Since one of its inputs is still at logic level "0" the output at Q still remains HIGH at logic level "1" and there is no change of state. Therefore, the flip-flop circuit is said to be "Latched" or "Set" with Q = "1" and Q = "0".

**Reset State**

In this second stable state, Q is at logic level "0", (not Q = "0") its inverse output at Q is at logic level "1", (Q = "1"), and is given by R = "1" and S = "0". As gate *X* has one of its inputs at logic "0" its output Q must equal logic level "1" (again NAND gate principles). Output Q is fed back to input "B", so both inputs to NAND gate *Y* are at logic "1", therefore, Q = "0". If the set input, S now changes state to logic "1" with input R remaining at logic "1", output Q still remains LOW at logic level "0" and there is no change of state. Therefore, the flip-flop circuits "Reset" state has also been latched and we can define this "set/reset" action in the following truth table.

**Truth Table for this Set-Reset Function**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| State | S | R | Q | Q | Description |
| Set | 1 | 0 | 1 | 0 | Set Q » 1 |
| 1 | 1 | 1 | 0 | no change |
| Reset | 0 | 1 | 0 | 1 | Reset Q » 0 |
| 1 | 1 | 0 | 1 | no change |
| Invalid | 0 | 0 | 0 | 1 | memory with Q = 0 |
| 0 | 0 | 1 | 0 | memory with Q = 1 |

It can be seen that when both inputs S = "1" and R = "1" the outputs Q and Q can be at either logic level "1" or "0", depending upon the state of inputs S or R BEFORE this input condition existed. However, input state R = "0" and S = "0" is an undesirable or invalid condition and must be avoided because this will give both outputs Q and Q to be at logic level "1" at the same time and we would normally want Q to be the inverse of Q. However, if the two inputs are now switched HIGH again after this condition to logic "1", both the outputs will go LOW resulting in the flip-flop becoming unstable and switch to an unknown data state based upon the unbalance. This unbalance can cause one of the outputs to switch faster than the other resulting in the flip-flop switching to one state or the other which may not be the required state and data corruption will exist. This unstable condition is known as its **Meta-stable** state.

Then, a bistable SR flip-flop or SR latch is activated or set by a logic "1" applied to its S input and deactivated or reset by a logic "1" applied to its R. The SR flip-flop is said to be in an "invalid" condition (Meta-stable) if both the set and reset inputs are activated simultaneously.

As well as using NAND gates, it is also possible to construct simple one-bit **SR Flip-flops** using two cross-coupled NOR gates connected in the same configuration. The circuit will work in a similar way to the NAND gate circuit above, except that the inputs are active HIGH and the invalid condition exists when both its inputs are at logic level "1", and this is shown below.

**The NOR Gate SR Flip-flop**

|  |
| --- |
| The SR Bistable Symbol |

 **Switch Debounce Circuits**

Edge-triggered flip-flops require a nice clean signal transition, and one practical use of this type of set-reset circuit is as a latch used to help eliminate mechanical switch "bounce". As its name implies, switch bounce occurs when the contacts of any mechanically operated switch, push-button or keypad are operated and the internal switch contacts do not fully close cleanly, but bounce together first before closing (or opening) when the switch is pressed. This gives rise to a series of individual pulses which can be as long as tens of milliseconds that an electronic system or circuit such as a digital counter may see as a series of logic pulses instead of one long single pulse and behave incorrectly. For example, during this bounce period the output voltage can fluctuate wildly and may register multiple input counts instead of one single count. Then set-reset SR Flip-flops or Bistable Latch circuits can be used to eliminate this kind of problem and this is demonstrated below.

**SR Bistable Switch Debounce Circuit**

|  |
| --- |
| SR Bistable Switch Debounce Circuit |

Depending upon the current state of the output, if the set or reset buttons are depressed the output will change over in the manner described above and any additional unwanted inputs (bounces) from the mechanical action of the switch will have no effect on the output at Q. When the other button is pressed, the very first contact will cause the latch to change state, but any additional mechanical switch bounces will also have no effect. The SR flip-flop can then be RESET automatically after a short period of time, for example 0.5 seconds, so as to register any additional and intentional repeat inputs from the same switch contacts, for example multiple inputs from a keyboards "RETURN" key.

Commonly available IC's specifically made to overcome the problem of switch bounce are the MAX6816, single input, MAX6817, dual input and the MAX6818 octal input switch debouncer IC's. These chips contain the necessary flip-flop circuitry to provide clean interfacing of mechanical switches to digital systems.

Set-Reset bistable latches can also be used as Monostable (one-shot) pulse generators to generate a single output pulse, either high or low, of some specified width or time period for timing or control purposes. The 74LS279 is a Quad SR Bistable Latch IC, which contains four individual NAND type bistable's within a single chip enabling switch debounce or monostable/astable clock circuits to be easily constructed.

**Quad SR Bistable Latch 74LS279**

|  |
| --- |
| Quad SR Latch 74279 |

 **Gated or Clocked SR Flip-Flop**

It is sometimes desirable in sequential logic circuits to have a bistable SR flip-flop that only changes state when certain conditions are met regardless of the condition of either the Set or the Reset inputs. By connecting a 2-input AND gate in series with each input terminal of the SR Flip-flop a Gated SR Flip-flop can be created. This extra conditional input is called an "Enable" input and is given the prefix of "EN". The addition of this input means that the output at Q only changes state when it is HIGH and can therefore be used as a clock (CLK) input making it level-sensitive as shown below.

**Gated SR Flip-flop**

|  |
| --- |
| Gated SR flip flop |

When the Enable input "EN" is at logic level "0", the outputs of the two AND gates are also at logic level "0", (AND Gate principles) regardless of the condition of the two inputs S and R, latching the two outputs Q and Q into their last known state. When the enable input "EN" changes to logic level "1" the circuit responds as a normal SR bistable flip-flop with the two AND gates becoming transparent to the Set and Reset signals. This enable input can also be connected to a clock timing signal adding clock synchronisation to the flip-flop creating what is sometimes called a "Clocked SR Flip-flop". So a **Gated Bistable SR Flip-flop** operates as a standard bistable latch but the outputs are only activated when a logic "1" is applied to its EN input and deactivated by a logic "0".

In the next tutorial about **Sequential Logic Circuits**, we will look at another type of simple edge-triggered flip-flop which is very similar to the **RS flip-flop** called a [**JK Flip-flop**](http://www.electronics-tutorials.ws/sequential/seq_2.html) named after its inventor, Jack Kilby. The JK flip-flop is the most widely used of all the flip-flop designs as it is considered to be a universal device.

JK Flip-flop **Tutorial: 2 of 6**

**The JK Flip-flop**

From the previous tutorial we now know that the basic gated SR NAND flip-flop suffers from two basic problems: number one, the S = 0 and R = 0 condition or S = R = 0 must always be avoided, and number two, if S or R change state while the enable input is high the correct latching action may not occur. Then to overcome these two fundamental design problems with the SR flip-flop, the **JK flip-Flop** was developed.

This simple **JK flip-Flop** is the most widely used of all the flip-flop designs and is considered to be a universal flip-flop circuit. The sequential operation of the JK flip-flop is exactly the same as for the previous SR flip-flop with the same "set" and "reset" inputs. The difference this time is that the JK flip-flop has no invalid or forbidden input states of the SR Latch (when S and R are both 1).

The **JK flip-flop** is basically a gated SR flip-flop with the addition of a clock input circuitry that prevents the illegal or invalid output condition that can occur when both inputs S and R are equal to logic level "1". Due to this additional clocked input, a JK flip-flop has four possible input combinations, "logic 1", "logic 0", "no change" and "toggle". The symbol for a JK flip-flop is similar to that of an [**SR Bistable Latch**](http://www.electronics-tutorials.ws/sequential/seq_1.html) as seen in the previous tutorial except for the addition of a clock input.

**The Basic JK Flip-flop**

|  |
| --- |
| JK Block Symbol |

Both the S and the R inputs of the previous SR bistable have now been replaced by two inputs called the J and K inputs, respectively after its inventor Jack Kilby. Then this equates to: J = S andK = R.

The two 2-input AND gates of the gated SR bistable have now been replaced by two 3-input NAND gates with the third input of each gate connected to the outputs at Q and Q. This cross coupling of the SR flip-flop allows the previously invalid condition of S = "1" and R = "1" state to be used to produce a "toggle action" as the two inputs are now interlocked. If the circuit is "SET" the J input is inhibited by the "0" status of Q through the lower NAND gate. If the circuit is "RESET" the K input is inhibited by the "0" status of Q through the upper NAND gate. As Q and Q are always different we can use them to control the input. When both inputs J and K are equal to logic "1", the JK flip-flop toggles as shown in the following truth table.

**The Truth Table for the JK Function**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| same asfor theSR Latch | J | K | Q | Q | Description |
| 0 | 0 | 0 | 0 | Memoryno change |
| 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | Reset Q » 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | Set Q » 1 |
| 1 | 0 | 1 | 0 |
| toggleaction | 1 | 1 | 0 | 1 | Toggle |
| 1 | 1 | 1 | 0 |

Then the JK flip-flop is basically an SR flip-flop with feedback which enables only one of its two input terminals, either SET or RESET to be active at any one time thereby eliminating the invalid condition seen previously in the SR flip-flop circuit. Also when both the J and the K inputs are at logic level "1" at the same time, and the clock input is pulsed either "HIGH", the circuit will "toggle" from its SET state to a RESET state, or visa-versa. This results in the JK flip-flop acting more like a T-type toggle flip-flop when both terminals are HIGH.

Although this circuit is an improvement on the clocked SR flip-flop it still suffers from timing problems called "race" if the output Q changes state before the timing pulse of the clock input has time to go "OFF". To avoid this the timing pulse period (T) must be kept as short as possible (high frequency). As this is sometimes not possible with modern TTL IC's the much improved **Master-Slave JK Flip-flop**was developed. This eliminates all the timing problems by using two SR flip-flops connected together in series, one for the "Master" circuit, which triggers on the leading edge of the clock pulse and the other, the "Slave" circuit, which triggers on the falling edge of the clock pulse. This results in the two sections, the master section and the slave section being enabled during opposite half-cycles of the clock signal.

The 74LS73 is a Dual JK flip-flop IC, which contains two individual JK type bistable's within a single chip enabling single or master-slave toggle flip-flops to be made. Other JK flip-flop IC's include the 74LS107 Dual JK flip-flop with clear, the 74LS109 Dual positive-edge triggered JK flip-flop and the 74LS112 Dual negative-edge triggered flip-flop with both preset and clear inputs.

**Dual JK Flip-flop 74LS73**

|  |
| --- |
| Dual JK flip-flop 7473 |

 **The Master-Slave JK Flip-flop**

The **Master-Slave Flip-Flop** is basically two gated SR flip-flops connected together in a series configuration with the slave having an inverted clock pulse. The outputs from Q and Q from the "Slave" flip-flop are fed back to the inputs of the "Master" with the outputs of the "Master" flip-flop being connected to the two inputs of the "Slave" flip-flop. This feedback configuration from the slave's output to the master's input gives the characteristic toggle of the JK flip-flop as shown below.

**The Master-Slave JK Flip-Flop**

|  |
| --- |
| Master-Slave JK Flip-flop |

The input signals J and K are connected to the gated "master" SR flip-flop which "locks" the input condition while the clock (Clk) input is "HIGH" at logic level "1". As the clock input of the "slave" flip-flop is the inverse (complement) of the "master" clock input, the "slave" SR flip-flop does not toggle. The outputs from the "master" flip-flop are only "seen" by the gated "slave" flip-flop when the clock input goes "LOW" to logic level "0". When the clock is "LOW", the outputs from the "master" flip-flop are latched and any additional changes to its inputs are ignored. The gated "slave" flip-flop now responds to the state of its inputs passed over by the "master" section. Then on the "Low-to-High" transition of the clock pulse the inputs of the "master" flip-flop are fed through to the gated inputs of the "slave" flip-flop and on the "High-to-Low" transition the same inputs are reflected on the output of the "slave" making this type of flip-flop edge or pulse-triggered.

Then, the circuit accepts input data when the clock signal is "HIGH", and passes the data to the output on the falling-edge of the clock signal. In other words, the **Master-Slave JK Flip-flop** is a "Synchronous" device as it only passes data with the timing of the clock signal.

In the next tutorial about **Sequential Logic Circuits**, we will look at [**Multivibrators**](http://www.electronics-tutorials.ws/sequential/seq_3.html) that are used as waveform generators to produce the clock signals to switch sequential circuits.

Multivibrators **Tutorial: 3 of 6**

**Multivibrators**

Individual **Sequential Logic** circuits can be used to build more complex circuits such as Multivibrators, Counters, Shift Registers, Latches and Memories etc, but for these types of circuits to operate in a "sequential" way, they require the addition of a clock pulse or timing signal to cause them to change their state. Clock pulses are generally continuous square or rectangular shaped waveform that is produced by a single pulse generator circuit such as a **Multivibrator**. This multivibrator circuit oscillates between a "HIGH" state and a "LOW" state producing a continuous output. Astable multivibrators generally have an even 50% duty cycle, that is that 50% of the cycle time the output is "HIGH" and the remaining 50% of the cycle time the output is "OFF". In other words, the duty cycle for an astable timing pulse is 1:1.

Sequential logic circuits that use the clock signal for synchronization are dependant upon the frequency and and clock pulse width to activate there switching action. Sequential circuits may also change their state on either the rising or falling edge, or both of the actual clock signal as we have seen previously with the basic flip-flop circuits. The following list are terms associated with a timing pulse or waveform.

|  |  |
| --- | --- |
| Active HIGH - if the state changes occur at the clock's rising edge or during the clock width. | Clock Pulse Waveform**Clock Signal Waveform** |
| Active LOW - if the state changes occur at the clock's falling edge. |
| Duty Cycle - is the ratio of clock width and clock period. |
| Clock Width - this is the time during which the value of the clock signal is equal to one. |
| Clock Period - this is the time between successive transitions in the same direction, i.e., between two rising or two falling edges. |
| Clock Frequency - the clock frequency is the reciprocal of the clock period, frequency = 1/clock period |

Clock pulse generation circuits can be a combination of analogue and digital circuits that produce a continuous series of pulses (these are called astable multivibrators) or a pulse of a specific duration (these are called monostable multivibrators). Combining two or more of multivibrators provides generation of a desired pattern of pulses (including pulse width, time between pulses and frequency of pulses).

There are basically three types of clock pulse generation circuits:

* Astable - A *free-running multivibrator* that has **NO** stable states but switches continuously between two states this action produces a train of square wave pulses at a fixed frequency.
* Monostable - A *one-shot multivibrator* that has only **ONE** stable state and is triggered externally with it returning back to its first stable state.
* Bistable - A *flip-flop* that has **TWO** stable states that produces a single pulse either positive or negative in value.

One way of producing a very simple clock signal is by the interconnection of logic gates. As NAND gates contains amplification, they can also be used to provide a clock signal or timing pulse with the aid of a single [**Capacitor, C**](http://www.electronics-tutorials.ws/capacitor/cap_1.html) and [**Resistor, R**](http://www.electronics-tutorials.ws/resistor/res_1.html) which provide the feedback and timing function. These timing circuits are often used because of there simplicity and are also useful if a logic circuit is designed that has un-used gates which can be utilised to create the monostable or astable oscillator. This simple type of RC Oscillator network is sometimes called a "Relaxation Oscillator".

**Monostable Circuits.**

**Monostable Multivibrators** or "one-shot" pulse generators are used to convert short sharp pulses into wider ones for timing applications. Monostable multivibrators generate a single output pulse, either "high" or "low", when a suitable external trigger signal or pulse T is applied. This trigger pulse signal initiates a timing cycle which causes the output of the monostable to change state at the start of the timing cycle, (t1) and remain in this second state until the end of the timing period, (t1) which is determined by the time constant of the timing capacitor, CT and the resistor, RT.

The monostable multivibrator now stays in this second timing state until the end of the RC time constant and automatically resets or returns itself back to its original (stable) state. Then, a monostable circuit has only one stable state. A more common name for this type of circuit is simply a "Flip-Flop" as it can be made from two cross-coupled NAND gates (or NOR gates) as we have seen previously. Consider the circuit below.

**Simple NAND Gate Monostable Circuit**

|  |
| --- |
| NAND Gate Monostable Oscillator |

Suppose that initially the trigger input T is held HIGH at logic level "1" by the resistor R1 so that the output from the first NAND gate U1 is LOW at logic level "0", (NAND gate principals). The timing resistor,RT is connected to a voltage level equal to logic level "0", which will cause the capacitor, CT to be discharged. The output of U1is LOW, timing capacitor CT is completely discharged therefore junction V1 is also equal to "0" resulting in the output from the second NANDgate U2, which is connected as an inverting NOT gate will therefore be HIGH.

The output from the second NAND gate, (U2) is fed back to one input of U1 to provide the necessary positive feedback. Since the junction V1 and the output of U1 are both at logic "0" no current flows in the capacitor CT. This results in the circuit being **Stable** and it will remain in this state until the trigger input T changes.

If a negative pulse is now applied either externally or by the action of the push-button to the trigger input of the NAND gate U1, the output of U1 will go HIGH to logic "1" (NAND gate principles). Since the voltage across the capacitor cannot change instantaneously (capacitor charging principals) this will cause the junction at V1 and also the input to U2 to also go HIGH, which inturn will make the output of the NAND gate U2 change LOW to logic "0" The circuit will now remain in this second state even if the trigger input pulse T is removed. This is known as the **Meta-stable** state.

The voltage across the capacitor will now increase as the capacitor CT starts to charge up from the output of U1 at a time constant determined by the resistor/capacitor combination. This charging process continues until the charging current is unable to hold the input of U2 and therefore junction V1 HIGH. When this happens, the output of U2 switches HIGH again, logic "1", which inturn causes the output of U1 to go LOW and the capacitor discharges into the output of U1 under the influence of resistor RT. The circuit has now switched back to its original stable state.

Thus for each negative going trigger pulse, the monostable multivibrator circuit produces a LOW going output pulse. The length of the output time period is determined by the capacitor/resistor combination ([**RC Network**](http://www.electronics-tutorials.ws/rc/rc_1.html)) and is given as the **Time Constant**T = 0.69RC of the circuit in seconds. Since the input impedance of theNAND gates is very high, large timing periods can be achieved.

As well as the NAND gate monostable type circuit above, it is also possible to build simple monostable timing circuits that start their timing sequence from the rising-edge of the trigger pulse using NOT gates, NAND gates and NOR gates connected as inverters as shown below.

**NOT Gate Monostable Circuit**

|  |
| --- |
| NOT Gate Monostable Circuit |

As with the NAND gate circuit above, initially the trigger input T is HIGH at a logic level "1" so that the output from the first NOT gate U1 is LOW at logic level "0". The timing resistor, RT and the capacitor, CT are connected together in parallel and also to the input of the second NOT gate U2. As the input to U2 is LOW at logic "0" its output at Q is HIGH at logic "1".

When a logic level "0" pulse is applied to the trigger input T of the first NOT gate it changes state and produces a logic level "1" output. The diode D1 passes this logic "1" voltage level to the RC timing network. The voltage across the capacitor, CT increases rapidly to this new voltage level, which is also connected to the input of the second NOT gate. This inturn outputs a logic "0" at Q and the circuit stays in this **Meta-stable** state as long as the trigger input T applied to the circuit remains LOW.

When the trigger signal returns HIGH, the output from the first NOT gate goes LOW to logic "0" (NOT gate principals) and the fully charged capacitor, CT starts to discharge itself through the parallel resistor, RT connected across it. When the voltage across the capacitor drops below the lower threshold value of the input to the second NOT gate, its output switches back again producing a logic level "1" at Q. The diode D1 prevents the timing capacitor from discharging itself back through the first NOT gates output.

Then, the **Time Constant** for a NOT gate **Monostable Multivibrator** is given as T = 0.8RC + Trigger in seconds.

One main disadvantage of **Monostable Multivibrators** is that the time between the application of the next trigger pulse T has to be greater than the RC time constant of the circuit.

**Astable Circuits.**

**Astable Multivibrators** are a type of free running oscillator that have no permanent "meta" or "steady" state but are continually changing there output from one state ("LOW") to the other state ("HIGH") and then back again. This continual switching action from "HIGH" to "LOW" and "LOW" to "HIGH" produces a continuous and stable square wave output that switches abruptly between the two logic levels making it ideal for timing and clock pulse applications. As with the monostable multivibrator circuit above, the timing cycle is determined by the time constant of the resistor-capacitor, [**RC Network**](http://www.electronics-tutorials.ws/rc/rc_1.html). Then the output frequency can be varied by changing the value(s) of the resistors and capacitor in the circuit.

**NAND Gate Astable Multivibrators**

|  |
| --- |
| NAND Gate Astable Oscillator |

The **astable multivibrator** circuit uses two CMOS NOT gates such as the CD4069 or the 74HC04 hex inverter ICs, or as in our simple circuit below a pair of CMOS NANDsuch as the CD4011 or the 74LS132 and an RC timing network. The two NAND gates are connected as inverting NOT gates.

Suppose that initially the output from the NAND gate U2 is HIGH at logic level "1", then the input must therefore be LOW at logic level "0" (NAND gate principles) as will be the output from the first NAND gate U1. Capacitor, C is connected between the output of the second NAND gate U2 and its input via the timing resistor, R2. The capacitor now charges up at a rate determined by the time constant of R2 and C.

As the capacitor, C charges up, the junction between the resistor R2 and the capacitor, C, which is also connected to the input of the NAND gate U1 via the stabilizing resistor, R2 decreases until the lower threshold value of U1 is reached at which point U1 changes state and the output of U1 now becomes HIGH. This causes NAND gate U2 to also change state as its input has now changed from logic "0" to logic "1" resulting in the output of NAND gate U2 becoming LOW, logic level "0".

Capacitor C is now reverse biased and discharges itself through the input of NAND gate U1. Capacitor, C charges up again in the opposite direction determined by the time constant of both R2 and C as before until it reaches the upper threshold value of NAND gate U1. This causes U1 to change state and the cycle repeats itself over again.

Then, the time constant for a NAND gate **Astable Multivibrator** is given as T = 2.2RC in seconds with the output frequency given asf = 1/T.

For example: if resistor R2 = 10kΩ and the capacitorC = 45nF, then the oscillation frequency will be given as:



then the output frequency is calculated as being 1kHz, which equates to a time constant of 1mS so the output waveform would look like:



**Bistable Circuits.**

The **Bistable Multivibrators** circuit is basically a SR flip-flop that we look at in the previous tutorials with the addition of an inverter or NOT gate to provide the necessary switching function. As with flip-flops, both states of a bistable multivibrator are stable, and the circuit will remain in either state indefinitely. This type of multivibrator circuit passes from one state to the other "only" when a suitable external trigger pulse T is applied and to go through a full "SET-RESET" cycle **two** triggering pulses are required. This type of circuit is also known as a "**Bistable Latch**", "**Toggle Latch**" or simply "**T-latch**".

**NAND Gate Bistable Multivibrator**

|  |
| --- |
| NAND Gate Bistable Multivibrator |

The simplest way to make a **Bistable Latch** is to connect together a pair of Schmitt NAND gates to form a SR latch as shown above. The two NAND gates, U2 and U3 form the bistable which is triggered by the inputNAND gate, U1. This U1NAND gate can be omitted and replaced by a single toggle switch to make a switch debounce circuit as seen previously in the [**SR Flip-flop**](http://www.electronics-tutorials.ws/sequential/seq_1.html) tutorial. When the input pulse goes "LOW" the bistable latches into its "SET" state, with its output at logic level "1", until the input goes "HIGH" causing the bistable to latch into its "RESET" state, with its output at logic level "0". The output of a bistable multivibrator will stay in this "RESET" state until another input pulse is applied and the whole sequence will start again.

Then a **Bistable Latch** or "Toggle Latch" is a two-state device in which both states either positive or negative, (logic "1" or logic "0") are stable.

**Bistable Multivibrators** have many applications such as frequency dividers, counters or as a storage device in computer memories but they are best used in circuits such as [**Latches**](http://www.electronics-tutorials.ws/sequential/seq_1.html) and [**Counters**](http://www.electronics-tutorials.ws/counter/count_1.html).

**555 Timer Circuit.**

Simple Monostable or Astable timing circuits can now be easily made using standard waveform generator IC's in the form of relaxation oscillators by connecting a few passive components to their inputs with the most commonly used waveform generator type IC being the classic **555 timer**.

The **555 Timer** is a very versatile low cost timing IC that can produce a very accurate timing periods with good stability of around 1% and which has a variable timing period from between a few micro-seconds to many hours with the timing period being controlled by a single RC network connected to a single positive supply of between 4.5 and 16 volts. The NE555 timer and its successors, ICM7555, CMOS LM1455, DUAL NE556 etc, are covered in the[**555 Oscillator**](http://www.electronics-tutorials.ws/waveforms/555_oscillator.html) tutorial and other good electronics based websites, so are only included here for reference purposes as a clock pulse generator. The 555 connected as an Astable oscillator is given below.

**NE555 Astable Multivibrator.**

|  |
| --- |
| NE555 Timer Circuit |

Here the 555 timer is connected as a basic Astable Multivibrator circuit. Pins 2 and 6 are connected together so that it will re-trigger itself on each timing cycle, thereby functioning as an Astable oscillator. Capacitor, C1 charges up through resistor,R1 and resistor, R2 but discharges only through resistor, R2 as the other side of R2 is connected to the discharge terminal, pin 7. Then the timing period of t1 and t2 is given as:

* t1 = 0.693 (R1 + R2) C1
* t2 = 0.693 (R2) C1
* T = t1 + t2

The voltage across the capacitor, C1 ranges from between 1/3 Vcc to approximately 2/3 Vcc depending upon the [**RC timing period**](http://www.electronics-tutorials.ws/rc/rc_1.html). This type of circuit is very stable as it operates from a single supply rail resulting in an oscillation frequency which is independent of the supply voltage Vcc.

In the next tutorial about **Sequential Logic Circuits**, we will look another type of clock controlled flop-flop called a [**Data Latch**](http://www.electronics-tutorials.ws/sequential/seq_4.html). Data latches are very useful sequential circuits which can be made from any standard gated SR flip-flop and used for frequency division to produce various ripple counters, frequency dividers and latches.

D Flip-Flop **Tutorial: 4 of 6**

**The D flip-flop**

One of the main disadvantages of the basic [**SR NAND Gate**](http://www.electronics-tutorials.ws/sequential/seq_1.html) bistable circuit is that the indeterminate input condition of "SET" = logic "0" and "RESET" = logic "0" is forbidden. This state will force both outputs to be at logic "1", over-riding the feedback latching action and whichever input goes to logic level "1" first will lose control, while the other input still at logic "0" controls the resulting state of the latch. In order to prevent this from happening an inverter can be connected between the "SET" and the "RESET" inputs to produce another type of flip-flop circuit called a **Data Latch**, **Delay flip-flop**, **D-type Bistable** or simply a **D-type flip-flop** as it is more generally called.

The **D flip-flop** is by far the most important of the clocked flip-flops as it ensures that ensures that inputs S and R are never equal to one at the same time. D-type flip-flops are constructed from a gated SR flip-flop with an inverter added between the S and the R inputs to allow for a single D (data) input. This single data input D is used in place of the "set" signal, and the inverter is used to generate the complementary "reset" input thereby making a level-sensitive D-type flip-flop from a level-sensitive RS-latch as now S = D and R = not D as shown.

**D flip-flop Circuit**

|  |
| --- |
| D flip-flop |

We remember that a simple SR flip-flop requires two inputs, one to "SET" the output and one to "RESET" the output. By connecting an inverter (NOT gate) to the SR flip-flop we can "SET" and "RESET" the flip-flop using just one input as now the two input signals are complements of each other. This complement avoids the ambiguity inherent in the SR latch when both inputs are LOW, since that state is no longer possible.

Thus the single input is called the "DATA" input. If this data input is HIGH the flip-flop would be "SET" and when it is LOW the flip-flop would be "RESET". However, this would be rather pointless since the flip-flop's output would always change on every data input. To avoid this an additional input called the "CLOCK" or "ENABLE" input is used to isolate the data input from the flip-flop after the desired data has been stored. The effect is that D is only copied to the output Q when the clock is active. This then forms the basis of a **D flip-flop**.

The **D flip-flop** will store and output whatever logic level is applied to its data terminal so long as the clock input is HIGH. Once the clock input goes LOW the "set" and "reset" inputs of the flip-flop are both held at logic level "1" so it will not change state and store whatever data was present on its output before the clock transition occurred. In other words the output is "latched" at either logic "0" or logic "1".

**Truth Table for the D Flip-flop**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Clk | D | Q | Q | Description |
| ↓ » 0 | X | Q | Q | Memoryno change |
| ↑ » 1 | 0 | 0 | 1 | Reset Q » 0 |
| ↑ » 1 | 1 | 1 | 0 | Set Q » 1 |

Note: ↓ and ↑ indicates direction of clock pulse as it is assumed D flip-flops are edge triggered

 **The Master-Slave JK Flip-flop**

The basic **D flip-flop** can be improved further by adding a second SR flip-flop to its output that is activated on the complementary clock signal to produce a "Master-Slave D flip-flop". On the leading edge of the clock signal (LOW-to-HIGH) the first stage, the "master" latches the input condition at D, while the output stage is deactivated. On the trailing edge of the clock signal (HIGH-to-LOW) the second "slave" stage is now activated, latching on to the output from the first master circuit. Then the output stage appears to be triggered on the negative edge of the clock pulse. "Master-Slave D flip-flops" can be constructed by the cascading together of two latches with opposite clock phases as shown.

**Master-Slave D flip-flop Circuit**

|  |
| --- |
| Master-Slave D-type Flip-flop |

We can see from above that on the leading edge of the clock pulse the master flip-flop will be loading data from the data D input, therefore the master is "ON". With the trailing edge of the clock pulse the slave flip-flop is loading data, i.e. the slave is "ON". Then there will always be one flip-flop "ON" and the other "OFF" but never both the master and slave "ON" at the same time. Therefore, the output Q acquires the value of D, only when one complete pulse, i.e. 0-1-0 is applied to the clock input.

There are many different D flip-flop IC's available in both TTL and CMOS packages with the more common being the 74LS74 which is a Dual D flip-flop IC, which contains two individual D type bistable's within a single chip enabling single or master-slave toggle flip-flops to be made. Other D flip-flop IC's include the 74LS174 HEX D flip-flop with direct clear input, the 74LS175 Quad D flip-flop with complementary outputs and the 74LS273 Octal D flip-flop containing eight D flip-flops with a clear input in one single package.

**Dual D flip-flop 74LS74**

|  |
| --- |
| Dual D flip-flop 7474 |

**Frequency Division**

One main use of a D flip-flop is as a [**Frequency Divider**](http://www.electronics-tutorials.ws/counter/count_1.html). If the Q output on a D-type flip-flop is connected directly to the D input giving the device closed loop "feedback", successive clock pulses will make the bistable "toggle" once every two clock cycles.

In the counters tutorials we saw how the **Data Latch** can be used as a "Binary Divider", or a "Frequency Divider" to produce a "divide-by-2" counter circuit, that is, the output has half the frequency of the clock pulses. By placing a feedback loop around the D flip-flop another type of flip-flop circuit can be constructed called a **T-type flip-flop** or more commonly a T-type bistable, that can be used as a divide-by-two circuit in binary counters as shown below.

**Divide-by-2 Counter**

|  |
| --- |
| Divide-by-2 Counter |

It can be seen from the frequency waveforms above, that by "feeding back" the output from Q to the input terminal D, the output pulses at Q have a frequency that are exactly one half (f/2) that of the input clock frequency, (Fin). In other words the circuit produces **frequency division** as it now divides the input frequency by a factor of two (an octave) as Q = 1 once every two clock cycles.

**Data Latches**

Another useful application of the Data Latch is to hold or remember the data present on its data input, thereby acting as a single bit memory device and IC's such as the TTL 74LS74 or the CMOS 4042 are available in Quad format for this purpose. By connecting together four, *1-bit* data latches so that all their clock terminals are connected at the same time a simple "4-bit" Data latch can be made as shown below.

**4-bit Data Latch**

|  |
| --- |
| 4-bit Data Latch |

**Transparent Data Latch**

The **Data Latch** is a very useful device in electronic and computer circuits. They can be designed to have very high output impedance at both outputs Q and its inverse or complement output Q to reduce the impedance effect on the connecting circuit when used as a buffer, I/O port, bi-directional bus driver or even a display driver. But a single "1-bit" data latch is not very practical to use on its own and instead commercially available IC's incorporate 4, 8, 10, 16 or even 32 individual data latches into one single IC package, and one such IC device is the 74LS373 Octal D-type transparent latch.

The eight individual data latches or bistables of the 74LS373 are "transparent" D-type flip-flops, meaning that when the clock (CLK) input is HIGH at logic level "1", (but can also be active low) the outputs atQ follows the data D inputs. In this configuration the latch is said to be "open" and the path from D input to Q output appears to be "transparent" as the data flows through it unimpeded, hence the name transparent latch. When the clock signal is LOW at logic level "0", the latch "closes" and the output at Q is latched at the last value of the data that was present before the clock signal changed and no longer changes in response to D.

**8-bit Data Latch**

|  |
| --- |
| 8-bit Data Latch |

Functional diagram of the 74LS373 Octal Transparent Latch

**D flip-flop Summary**

The data or **D flip-flop** can be built from a pair of back-to-back latches by connecting an inverter between the S and the R inputs to allow for a single D (data) input. The basic D-type flip-flop circuit can be improved further by adding a second SR flip-flop to its output that is activated on the complementary clock signal to produce a "Master-Slave D flip-flop". The difference between a D-type latch and a D flip-flop is that a latch does not have a clock signal, whereas a flip-flop always does. The D flip-flop is an edge triggered device which transfers input data to Q on clock rising or falling edge. Data Latches are Level sensitive devices such as the data latch and the transparent latch.

In the next tutorial about **Sequential Logic Circuits**, we will look at connecting together data latches to produce another type of sequential logic circuit called a [**Shift Register**](http://www.electronics-tutorials.ws/sequential/seq_5.html) that are used to convert parallel data into serial data and vice versa.

The Shift Register

**Tutorial: 5 of 6**

**The Shift Register**

The **Shift Register** is another type of sequential logic circuit that is used for the storage or transfer of data in the form of binary numbers and then "shifts" the data out once every clock cycle, hence the name "shift register". It basically consists of several single bit "D-Type Data Latches", one for each bit (0 or 1) connected together in a serial or daisy-chain arrangement so that the output from one data latch becomes the input of the next latch and so on. The data bits may be fed in or out of the register serially, i.e. one after the other from either the left or the right direction, or in parallel, i.e. all together. The number of individual data latches required to make up a single **Shift Register** is determined by the number of bits to be stored with the most common being 8-bits wide, i.e. eight individual data latches.

The Shift Register is used for data storage or data movement and are used in 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. The individual data latches that make up a single shift register are all driven by a common clock (Clk) signal making them synchronous devices. Shift register IC's are generally provided with a *clear* or *reset* connection so that they can be "SET" or "RESET" as required.

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 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.

The effect of data movement from left to right through a shift register can be presented graphically as:



Also, the directional movement of the data through a shift register can be either to the left, (left shifting) to the right, (right shifting) left-in but right-out, (rotation) or both left and right shifting within the same register thereby making it *bidirectional*. In this tutorial it is assumed that all the data shifts to the right, (right shifting).

**Serial-in to Parallel-out (SIPO)**

**4-bit Serial-in to Parallel-out Shift Register**

|  |
| --- |
| Shift Register |

The operation is as follows. Lets assume that all the flip-flops (FFAto FFD) have just been RESET (CLEAR input) and that all the outputs QAto QD are at logic level "0" i.e, no parallel data output. If a logic "1" is connected to the DATA input pin of FFA then on the first clock pulse the output of FFA and therefore the resulting QA will be set HIGH to logic "1" with all the other outputs still remaining LOW at logic "0". Assume now that the DATA input pin of FFA has returned LOW again to logic "0" giving us one data pulse or 0-1-0.

The second clock pulse will change the output of FFA to logic "0" and the output of FFB and QB HIGH to logic "1" as its inputD has the logic "1" level on it from QA. The logic "1" has now moved or been "shifted" one place along the register to the right as it is now at QA. When the third clock pulse arrives this logic "1" value moves to the output of FFC(QC) and so on until the arrival of the fifth clock pulse which sets all the outputs QA to QD back again to logic level "0" because the input to FFA has remained constant at logic level "0".

The effect of each clock pulse is to shift the data contents of each stage one place to the right, and this is shown in the following table until the complete data value of 0-0-0-1 is stored in the register. This data value can now be read directly from the outputs of QA to QD. Then the data has been converted from a serial data input signal to a parallel data output. The truth table and following waveforms show the propagation of the logic "1" through the register from left to right as follows.

**Basic Movement of Data through a Shift Register**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Clock Pulse No | QA | QB | QC | QD |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 1 | 0 |
| 4 | 0 | 0 | 0 | 1 |
| 5 | 0 | 0 | 0 | 0 |

|  |
| --- |
| Shift Register |

Note that after the fourth clock pulse has ended the 4-bits of data (0-0-0-1) are stored in the register and will remain there provided clocking of the register has stopped. In practice the input data to the register may consist of various combinations of logic "1" and "0". Commonly available SIPO IC's include the standard 8-bit 74LS164 or the 74LS594.

**Serial-in to Serial-out (SISO)**

This **shift register** is very similar to the SIPO above, except were before the data was read directly in a parallel form from the outputs QA to QD, this time the data is allowed to flow straight through the register and out of the other end. Since there is only one output, the DATA leaves the shift register one bit at a time in a serial pattern, hence the name **Serial-in to Serial-Out Shift Register** or **SISO**.

The SISO shift register is one of the simplest of the four configurations as it has only three connections, the serial input (SI) which determines what enters the left hand flip-flop, the serial output (SO) which is taken from the output of the right hand flip-flop and the sequencing clock signal (Clk). The logic circuit diagram below shows a generalized serial-in serial-out shift register.

**4-bit Serial-in to Serial-out Shift Register**

|  |
| --- |
| Serial-in Serial-out Shift Register |

You may think what's the point of a SISO shift register if the output data is exactly the same as the input data. Well this type of **Shift Register** also acts as a temporary storage device or as a time delay device for the data, with the amount of time delay being controlled by the number of stages in the register, 4, 8, 16 etc or by varying the application of the clock pulses. Commonly available IC's include the 74HC595 8-bit Serial-in/Serial-out Shift Register all with 3-state outputs.

**Parallel-in to Serial-out (PISO)**

The Parallel-in to Serial-out shift register acts in the opposite way to the serial-in to parallel-out one above. The data is loaded into the register in a parallel format i.e. all the data bits enter their inputs simultaneously, to the parallel input pins PA to PD of the register. The data is then read out sequentially in the normal shift-right mode from the register at Q representing the data present at PA to PD. This data is outputted one bit at a time on each clock cycle in a serial format. It is important to note that with this system a clock pulse is not required to parallel load the register as it is already present, but four clock pulses are required to unload the data.

**4-bit Parallel-in to Serial-out Shift Register**

|  |
| --- |
| Parallel-in Serial-out Shift Register |

As this type of shift register converts parallel data, such as an 8-bit data word into serial format, it can be used to multiplex many different input lines into a single serial DATA stream which can be sent directly to a computer or transmitted over a communications line. Commonly available IC's include the 74HC166 8-bit Parallel-in/Serial-out Shift Registers.

**Parallel-in to Parallel-out (PIPO)**

The final mode of operation is the Parallel-in to Parallel-out Shift Register. This type of register also acts as a temporary storage device or as a time delay device similar to the SISO configuration above. The data is presented in a parallel format to the parallel input pins PA to PD and then transferred together directly to their respective output pins QA to QA by the same clock pulse. Then one clock pulse loads and unloads the register. This arrangement for parallel loading and unloading is shown below.

**4-bit Parallel-in to Parallel-out Shift Register**

|  |
| --- |
| Parallel-in Parallel-out Shift Register |

The PIPO shift register is the simplest of the four configurations as it has only three connections, the parallel input (PI) which determines what enters the flip-flop, the parallel output (PO) and the sequencing clock signal (Clk).

Similar to the Serial-in to Serial-out shift register, this type of register also acts as a temporary storage device or as a time delay device, with the amount of time delay being varied by the frequency of the clock pulses. Also, in this type of register there are no interconnections between the individual flip-flops since no serial shifting of the data is required.

**Universal Shift Register**

Today, high speed bi-directional "universal" type **Shift Registers** such as the TTL 74LS194, 74LS195 or the CMOS 4035 are available as a 4-bit multi-function devices that can be used in either serial-to-serial, left shifting, right shifting, serial-to-parallel, parallel-to-serial, and as a parallel-to-parallel multifunction data register, hence the name "Universal". These devices can perform any combination of parallel and serial input to output operations but require additional inputs to specify desired function and to pre-load and reset the device.

**4-bit Universal Shift Register 74LS194**

|  |
| --- |
| Universal Shift Register |

Universal shift registers are very useful digital devices. They can be configured to respond to operations that require some form of temporary memory, delay information such as the SISO or PIPO configuration modes or transfer data from one point to another in either a serial or parallel format. Universal shift registers are frequently used in arithmetic operations to shift data to the left or right for multiplication or division.

**Summary of Shift Registers**

* Then to summarise.
* 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 converting between serial and parallel data.
* Shift registers are identified as SIPO, SISO, PISO, PIPO, and universal shift registers.

In the next tutorial about **Sequential Logic Circuits**, we will look at what happens when the output of the last flip-flop in a shift register is connected directly back to the input of the first flip-flop producing a closed loop circuit that constantly recirculates the data around the loop. This then produces another type of sequential logic circuit called a [**Ring Counter**](http://www.electronics-tutorials.ws/sequential/seq_6.html) that are used as decade counters and dividers.

Ring Counter

**Tutorial: 6 of 6**

**The Ring Counter**

In the previous [**Shift Register**](http://www.electronics-tutorials.ws/sequential/seq_5.html) tutorial we saw that if we apply a serial data signal to the input of a *serial-in to serial-out shift register*, the same sequence of data will exit from the last flip-flip in the register chain after a preset number of clock cycles thereby acting as a sort of time delay circuit to the original signal. But what if we were to connect the output of this shift register back to its input so that the output from the last flip-flop, QD becomes the input of the first flip-flop, DA. We would then have a closed loop circuit that "recirculates" the DATA around a continuous loop for every state of its sequence, and this is the principal operation of a **Ring Counter**. Then by looping the output back to the input, we can convert a standard shift register into a ring counter. Consider the circuit below.

**4-bit Ring Counter**

|  |
| --- |
| Ring Counter |

The synchronous **Ring Counter** example above, is preset so that exactly one data bit in the register is set to logic "1" with all the other bits reset to "0". To achieve this, a "CLEAR" signal is firstly applied to all the flip-flops together in order to "RESET" their outputs to a logic "0" level and then a "PRESET" pulse is applied to the input of the first flip-flop (FFA) before the clock pulses are applied. This then places a single logic "1" value into the circuit of the ring counter . On each successive clock pulse, the counter circulates the same data bit between the four flip-flops over and over again around the "ring" every fourth clock cycle. But in order to cycle the data correctly around the counter we must first "load" the counter with a suitable data pattern as all logic "0"'s or all logic "1"'s outputted at each clock cycle would make the ring counter invalid.

This type of data movement is called "rotation", and like the previous shift register, the effect of the movement of the data bit from left to right through a ring counter can be presented graphically as follows along with its timing diagram:

**Rotational Movement of a Ring Counter**





Since the ring counter example shown above has four distinct states, it is also known as a "modulo-4" or "mod-4" counter with each flip-flop output having a frequency value equal to one-fourth or a quarter (1/4) that of the main clock frequency.

The "MODULO" or "MODULUS" of a counter is the number of states the counter counts or sequences through before repeating itself and a ring counter can be made to output any modulo number. A "mod-n" ring counter will require "n" number of flip-flops connected together to circulate a single data bit providing "n" different output states. For example, a mod-8 ring counter requires eight flip-flops and a mod-16 ring counter would require sixteen flip-flops. However, as in our example above, only four of the possible sixteen states are used, making ring counters very inefficient in terms of their output state usage.

**Johnson Ring Counter**

The **Johnson Ring Counter** or "Twisted Ring Counters", is another shift register with feedback exactly the same as the standard *Ring Counter* above, except that this time the inverted output Q of the last flip-flop is now connected back to the input D of the first flip-flop as shown below. The main advantage of this type of ring counter is that it only needs half the number of flip-flops compared to the standard ring counter then its modulo number is halved. So a "n-stage" Johnson counter will circulate a single data bit giving sequence of 2n different states and can therefore be considered as a "mod-2n counter".

**4-bit Johnson Ring Counter**

|  |
| --- |
| Johnson Ring Counter |

This inversion of Q before it is fed back to input D causes the counter to "count" in a different way. Instead of counting through a fixed set of patterns like the normal ring counter such as for a 4-bit counter, "0001"(1), "0010"(2), "0100"(4), "1000"(8) and repeat, the Johnson counter counts up and then down as the initial logic "1" passes through it to the right replacing the preceding logic "0". A 4-bit Johnson ring counter passes blocks of four logic "0" and then four logic "1" thereby producing an 8-bit pattern. As the inverted output Q is connected to the input D this 8-bit pattern continually repeats. For example, "1000", "1100", "1110", "1111", "0111", "0011", "0001", "0000" and this is demonstrated in the following table below.

**Truth Table for a 4-bit Johnson Ring Counter**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Clock Pulse No | FFA | FFB | FFC | FFD |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 2 | 1 | 1 | 0 | 0 |
| 3 | 1 | 1 | 1 | 0 |
| 4 | 1 | 1 | 1 | 1 |
| 5 | 0 | 1 | 1 | 1 |
| 6 | 0 | 0 | 1 | 1 |
| 7 | 0 | 0 | 0 | 1 |

As well as counting or rotating data around a continuous loop, ring counters can also be used to detect or recognise various patterns or number values within a set of data. By connecting simple logic gates such as the [**AND**](http://www.electronics-tutorials.ws/logic/logic_2.html) or the [**OR**](http://www.electronics-tutorials.ws/logic/logic_3.html) gates to the outputs of the flip-flops the circuit can be made to detect a set number or value. Standard 2, 3 or 4-stage Johnson ring counters can also be used to divide the frequency of the clock signal by varying their feedback connections and divide-by-3 or divide-by-5 outputs are also available.

A 3-stage Johnson Ring Counter can also be used as a 3-phase, 120 degree phase shift square wave generator by connecting to the data outputs at A, B and NOT-B. The standard 5-stage Johnson counter such as the commonly available CD4017 is generally used as a synchronous decade counter/divider circuit. The smaller 2-stage circuit is also called a "Quadrature" (sine/cosine) Oscillator/Generator and is used to produce four individual outputs that are each "phase shifted" by 90 degrees with respect to each other, and this is shown below.

**2-bit Quadrature Generator**

|  |
| --- |
| Quatrature Generator |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Output | A | B | C | D |
| QA+QB | 1 | 0 | 0 | 0 |
| QA+QB | 0 | 1 | 0 | 0 |
| QA+QB | 0 | 0 | 1 | 0 |
| QA+QB | 0 | 0 | 0 | 1 |
| 2-bit Quadrature Oscillator, Count Sequence |

As the four outputs, A to D are phase shifted by 90 degrees with regards to each other, they can be used with additional circuitry, to drive a 2-phase full-step stepper motor for position control or the ability to rotate a motor to a particular location as shown below.

**Stepper Motor Control**

|  |
| --- |
| Stepper Motor Control |

**2-phase (unipolar) Full-Step Stepper Motor Circuit**

The speed of rotation of the [**Stepper Motor**](http://www.electronics-tutorials.ws/io/io_7.html) will depend mainly upon the clock frequency and additional circuitry would be require to drive the "power" requirements of the motor. As this section is only intended to give the reader a basic understanding of **Johnson Ring Counters** and its applications, other good websites explain in more detail the types and drive requirements of stepper motors.

**Johnson Ring Counters** are available in standard TTL or CMOS IC form, such as the CD4017 5-Stage, decade Johnson ring counter with 10 active HIGH decoded outputs or the CD4022 4-stage, divide-by-8 Johnson counter with 8 active HIGH decoded outputs.