http://www.electronics-tutorials.ws

Frequency Division **Tutorial: 1 of 4**

**Frequency Division**

In the [**Sequential Logic**](http://www.electronics-tutorials.ws/sequential/seq_4.html) tutorials we saw how D-type Flip-Flop´s work and how they can be connected together to form a Data Latch. Another useful feature of the D-type Flip-Flop is as a binary divider, for **Frequency Division** or as a "divide-by-2" counter. Here the inverted output terminal Q (NOT-Q) is connected directly back to the Data input terminal D giving the device "feedback" as shown below.

**Divide-by-2 Counter**

|  |
| --- |
| Divide-by-2 Frequency Divider |

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. In other words the circuit produces **Frequency Division** as it now divides the input frequency by a factor of two (an octave). This then produces a type of counter called a "ripple counter" and in ripple counters, the clock pulse triggers the first flip-flop whose output triggers the second flip-flop, which inturn triggers the third flip-flop and so on through the chain.

**Toggle Flip-Flop**

Another type of device that can be used for frequency division is the T-type or Toggle flip-flop. With a slight modification to a standard JK flip-flop, we can construct a new type of flip-flop called a **Toggle flip-flop** were the two inputs J and k of a JK flip-flop are connected together resulting in a device with only two inputs, the "Toggle" input itself and the controlling "Clock" input. The name "Toggle flip-flop" indicates the fact that the flip-flop has the ability to toggle between its two states, the "toggle state" and the "memory state". Since there are only two states, a T-type flip-flop is ideal for use in frequency division and counter design.

Binary ripple counters can be built using "Toggle" or "T-type flip-flops" by connecting the output of one to the clock input of the next. Toggle flip-flops are ideal for building ripple counters as it toggles from one state to the next, (HIGH to LOW or LOW to HIGH) at every clock cycle so simple frequency divider and ripple counter circuits can easily be constructed using standard T-type flip-flop circuits.

If we connect together in series, two T-type flip-flops the initial input frequency will be "divided-by-two" by the first flip-flop ( f÷2 ) and then "divided-by-two" again by the second flip-flop ( f÷2 )÷2, giving an output frequency which has effectively been divided four times, then its output frequency becomes one quarter value (25%) of the original clock frequency, ( f÷4 ). Each time we add another toggle or "T-type" flip-flop the output clock frequency is halved or divided-by-2 again and so on, giving an output frequency of 2n where "n" is the number of flip-flops used in the sequence.

Then the Toggle or T-type flip-flop is an edge triggered divide-by-2 device based upon the standard JK-type flip flop and which is triggered on the rising edge of the clock signal. The result is that each bit moves right by one flip-flop. All the flip-flops can be asynchronously reset and can be triggered to switch on either the leading or trailing edge of the input clock signal making it ideal for **Frequency Division**.

**Frequency Division using Toggle Flip-flops**

|  |
| --- |
| Frequency Division |

This type of counter circuit used for frequency division is commonly known as an **Asynchronous 3-bit Binary Counter** as the output on QA to QC, which is 3 bits wide, is a binary count from 0 to 7 for each clock pulse. In an asynchronous counter, the clock is applied only to the first stage with the output of one flip-flop stage providing the clocking signal for the next flip-flop stage and subsequent stages derive the clock from the previous stage with the clock pulse being halved by each stage.

This arrangement is commonly known as *Asynchronous* as each clocking event occurs independently as all the bits in the counter do not all change at the same time. As the counter counts sequentially in an upwards direction from 0 to 7. This type of counter is also known as an "up" or "forward" counter (**CTU**) or a **"3-bit Asynchronous Up Counter"**. The three-bit asynchronous counter shown is typical and uses flip-flops in the toggle mode. Asynchronous "Down" counters (**CTD**) are also available.

**Truth Table for a 3-bit Asynchronous Up Counter**

|  |  |  |  |
| --- | --- | --- | --- |
| Clock Cycle | Output bit Pattern | | |
| QC | QB | QA |
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 2 | 0 | 1 | 0 |
| 3 | 0 | 1 | 1 |
| 4 | 1 | 0 | 0 |
| 5 | 1 | 0 | 1 |
| 6 | 1 | 1 | 0 |
| 7 | 1 | 1 | 1 |

Then by cascading together D-type or Toggle Flip-Flops we can produce divide-by-2, 4, 8 etc, asynchronous counter circuits which divide the clock frequency 2, 4 or 8 times.

**Counters**

Then a counter is a specialised register or pattern generator that produces a specified output pattern or sequence of binary values (or states) upon the application of an input pulse called the "Clock". The clock is actually used for data in these applications. Typically, counters are logic circuits than can increment or decrement a count by one but when used as asynchronous divide-by-n counters they are able to divide these input pulses producing a clock division signal.

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, ie, one complete cycle. A counter with three flip-flops like the circuit above will count from 0 to 7 ie, 2n-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 = 23 = 8 (modulo-8 or MOD-8)
* 4-bit Binary Counter = 24 = 16 (modulo-16 or MOD-16)
* 8-bit Binary Counter = 28 = 256 (modulo-256 or MOD-256)

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 = 2n

**4-bit Modulo-16 Counter**

|  |
| --- |
| 4-bit Counter Waveform |

Multi-bit asynchronous counters connected in this manner are also called **"Ripple Counters"** or ripple dividers because the change of state at each stage appears to "ripple" itself through the counter from the LSB output to its MSB output connection. Ripple counters are available in standard IC form, from the 74LS393 Dual 4-bit counter to the 74HC4060, which is a 14-bit ripple counter with its own built in clock oscillator and produce excellent frequency division of the fundamental frequency.

**Frequency Division Summary**

For **frequency division**, toggle mode flip-flops are used in a chain as a divide by two counter. One flip-flop will divide the clock, ƒin by 2, two flip-flops will divide ƒin by 4 (and so on). One benefit of using toggle flip-flops for frequency division is that the output at any point has an exact 50% duty cycle.

The final output clock signal will have a frequency value equal to the input clock frequency divided by the MOD number of the counter. Such circuits are known as "divide-by-n" counters. Counters can be formed by connecting individual flip-flops together and are classified according to the way they are clocked. In *Asynchronous counters*, (ripple counter) the first flip-flop is clocked by the external clock pulse and then each successive flip-flop is clocked by the output of the preceding flip-flop. In *Synchronous counters*, the clock input is connected to all of the flip-flop so that they are clocked simultaneously.

In the next tutorial we will look at Asynchronous counters, and see that the main characteristic of an asynchronous counter is that each flip-flop in the chain derives its own clock from the previous flip-flop and is therefore independent of the input clock.

Asynchronous Counter **Tutorial: 2 of 4**

**Asynchronous Counter**

In the previous tutorial we saw that an **Asynchronous counter** can have 2n-1 possible counting states e.g. MOD-16 for a 4-bit counter, (0-15) making it ideal for use in [**Frequency Division**](http://www.electronics-tutorials.ws/counter/count_1.html). But it is also possible to use the basic asynchronous counter to construct special counters with counting states less than their maximum output number by forcing the counter to reset itself to zero at a pre-determined value producing a type of asynchronous counter that has truncated sequences. Then an n-bit counter that counts up to its maximum modulus (2n) is called a full sequence counter and a n-bit counter whose modulus is less than the maximum possible is called a **truncated counter**.

But why would we want to create an asynchronous truncated counter that is not a MOD-4, MOD-8, or some other modulus that is equal to the power of two. The answer is that we can by using combinational logic to take advantage of the asynchronous inputs on the flip-flop. If we take the modulo-16 asynchronous counter and modified it with additional logic gates it can be made to give a decade (divide-by-10) counter output for use in standard decimal counting and arithmetic circuits.

Such counters are generally referred to as **Decade Counters**. A decade counter requires resetting to zero when the output count reaches the decimal value of 10, ie. when DCBA = 1010and 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.

**Asynchronous Decade Counter**

|  |
| --- |
| Asynchronous Decade Counter |

This type of asynchronous counter counts upwards on each leading edge of the input clock signal starting from "0000" until it reaches an output "1010" (decimal 10). Both outputs QB andQD are now equal to logic "1" and the output from the NAND gate changes state from logic "1" to a logic "0" level and whose output is also connected to the CLEAR (CLR) inputs of all the J-K Flip-flops. This causes all of the Q outputs to be reset back to binary "0000" on the count of 10. Once QB and QD are both equal to logic "0" 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** counter.

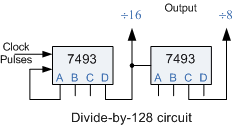
**Decade Counter Truth Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Clock Count | Output bit Pattern | | | | Decimal Value |
| QD | QC | QB | QA |
| 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 | | | | |

**Decade Counter Timing Diagram**

|  |
| --- |
| Decade Counter Timing |

Using the same idea of truncating counter output sequences, the above circuit could easily be adapted to other counting cycles be simply changing the connections to the AND gate. For example, a scale-of-twelve (modulo-12) can easily be made by simply taking the inputs to the AND gate from the outputs at "QC" and "QD", noting that the binary equivalent of 12 is "1100" and that output "QA" is the least significant bit (LSB). Since the maximum modulus that can be implemented with n flip-flops is 2n, this means that when you are designing truncated counters you should determine the lowest power of two that is greater than or equal to your desired modulus. For example, lets say you wish to count from 0 to 39, or mod-40. Then the highest number of flip-flops required would be six, n = 6 giving a maximum MOD of 64 as five flip-flops would only equal MOD-32.

Then suppose we wanted to build a "divide-by-128" counter for frequency division we would need to cascade seven flip-flops since 128 = 27. Using dual flip-flops such as the 74LS74 we would still need four IC's to complete the circuit. One easy alternative method would be to use two TTL 7493's as 4-bit ripple counter/dividers. Since 128 = 16 x 8, one 7493 could be configured as a "divide-by-16" counter and the other as a "divide-by-8" counter. The two IC's would be cascaded together to form a "divide-by-128" frequency divider as shown.

Of course standard IC asynchronous counters are available such as the TTL 74LS90 programmable ripple counter/divider which can be configured as a divide-by-2, divide-by-5 or any combination of both. The 74LS390 is a very flexible dual decade driver IC with a large number of "divide-by" combinations available ranging form divide-by-2, 4, 5, 10, 20, 25, 50, and 100.

**Frequency Dividers**

This ability of the ripple counter to truncate sequences to produce a "divide-by-n" output means that counters and especially ripple counters, can be used as frequency dividers to reduce a high clock frequency down to a more usable value for use in digital clocks and timing applications. For example, assume we require an accurate 1Hz timing signal to operate a digital clock. We could quite easily produce a 1Hz square wave signal from a standard 555 timer chip but the manufacturers data sheet tells us that it has a typical 1-2% timing error depending upon the manufacturer, and at low frequencies a 2% error at 1Hz is not good. However the data sheet also tells us that the maximum operating frequency of the 555 timer is about 300kHz and a 2% error at this high frequency would be acceptable. So by choosing a higher timing frequency of say 262.144kHz and an 18-bit ripple (Modulo-18) counter we can make a precision 1Hz timing signal as shown below.

**Simple 1Hz timing signal using an 18-bit ripple counter/divider.**

|  |
| --- |
| 1Hz Timing Counter |

This is of course a very simple example of how to produce accurate frequencies, but by using high frequency crystal oscillators and multi-bit frequency dividers, precision frequency generators can be produced for for a range of applications ranging from clocks or watches to event timing and even electronic piano/synthesizer music applications.

The main disadvantages with asynchronous counters are that there is a small delay between the arrival of the clock pulse and its output due to the internal circuitry of the gate. In asynchronous circuits this delay is called the **Propagation Delay** (giving the asynchronous ripple counter the nickname of propagation counter) and in some cases can produce false output counts. In large bit ripple counter circuits the delay of all the separate stages are added together to give a summed delay at the end of the chain which is why asynchronous counters are generally not used for in high frequency counting circuits were large numbers of bits are involved.

Also, the outputs from the counter do not have a fixed time relationship with each other and do not occur at the same time due to their clocking sequence. Then, the more flip-flops that are added to an asynchronous counter chain the lower the maximum operating frequency becomes. To overcome the problem of propagation delay Synchronous Counters were developed.

Then to summarise:

* **Asynchronous Counters** can be made from Toggle or D-type flip-flops.
* They are called asynchronous counters because the clock input of the flip-flops are not all driven by the same clock signal.
* Each output in the chain depends on a change in state from the previous flip-flops output.
* Asynchronous counters are sometimes called ripple counters because the data appears to "ripple" from the output of one flip-flop to the input of the next.
* They can be implemented using "divide-by-n" circuits.
* Truncated counters can produce any modulus number count.

Disadvantages of Asynchronous Counters:

* An extra "re-synchronizing" output flip-flop may be required.
* To count a truncated sequence not equal to 2n, extra feedback logic is required.
* Counting a large number of bits, propagation delay by successive stages may become undesirably large.
* This delay gives them the nickname of "Propagation Counters".
* Counting errors at high clocking frequencies.
* Synchronous Counters are faster using the same clock signal for all flip-flops.

In the next tutorial about Counters, we will look at the [**Synchronous Counter**](http://www.electronics-tutorials.ws/counter/count_3.html) and see that the main characteristic of an synchronous counter is that the clock input of each flip-flop in the chain is connected to all of the flip-flops so that they are clocked simultaneously.

Synchronous Counter **Tutorial: 3 of 4**

**Binary Synchronous Counter**

In the previous Asynchronous binary counter tutorial, we saw that the output of one counter stage is connected directly to the clock input of the next counter stage and so on along the chain, and as a result the asynchronous counter suffers from what is known as "Propagation Delay". However, with the **Synchronous Counter**, the external clock signal is connected to the clock input of EVERY individual flip-flop within the counter so that all of the flip-flops are clocked together simultaneously (in parallel) at the same time giving a fixed time relationship. In other words, changes in the output occur in "synchronization" with the clock signal. This results in all the individual output bits changing state at exactly the same time in response to the common clock signal with no ripple effect and therefore, no propagation delay.

**Binary 4-bit Synchronous Counter**

|  |
| --- |
| 4-bit Synchronous Counter |

It can be seen that the external clock pulses (pulses to be counted) are fed directly to each [**J-K flip-flop**](http://www.electronics-tutorials.ws/sequential/seq_2.html) in the counter chain and that both the J and K inputs are all tied together in toggle mode, but only in the first flip-flop, flip-flop A (LSB) are they connected HIGH, logic "1" allowing the flip-flop to toggle on every clock pulse. Then the synchronous counter follows a predetermined sequence of states in response to the common clock signal, advancing one state for each pulse.

The J and K inputs of flip-flop B are connected to the output "Q" of flip-flop A, but the J and K inputs of flip-flops C and D are driven from AND gates which are also supplied with signals from the input and output of the previous stage. If we enable each J-K flip-flop to toggle based on whether or not all preceding flip-flop outputs (Q) are "HIGH" we can obtain the same counting sequence as with the asynchronous circuit but without the ripple effect, since each flip-flop in this circuit will be clocked at exactly the same time. As there is no propagation delay in synchronous counters because all the counter stages are triggered in parallel the maximum operating frequency of this type of counter is much higher than that of a similar asynchronous counter.

**4-bit Synchronous Counter Waveform Timing Diagram.**

|  |
| --- |
| 4-bit Waveform Timing Diagram |

Because this 4-bit synchronous counter counts sequentially on every clock pulse the resulting outputs count upwards from 0 ( "0000" ) to 15 ( "1111" ). Therefore, this type of counter is also known as a **4-bit Synchronous Up Counter**.

As synchronous 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, the modulo's or "MOD" number still applies as it does for asynchronous counters so a Decade counter or BCD counter with counts from 0 to 2n-1can be built along with truncated sequences.

**Decade 4-bit Synchronous Counter**

A 4-bit decade synchronous counter can also be built using synchronous binary counters to produce a count sequence from 0 to 9. A standard binary counter can be converted to a decade (decimal 10) counter with the aid of some additional logic to implement the desired state sequence. After reaching the count of "1001", the counter recycles back to "0000". We now have a decade or **Modulo-10** counter.

**Decade 4-bit Synchronous Counter**

|  |
| --- |
| Decade 4-bit Synchronous Counter |

The additional AND gates detect when the sequence reaches "1001", (Binary 10) and causes flip-flop FF3 to toggle on the next clock pulse. Flip-flop FF0toggles on every clock pulse. Thus, the count starts over at "0000" producing a synchronous decade counter. We could quite easily re-arrange the additional AND gates to produce other counters such as a Mod-12 Up counter which counts 12 states from"0000" to "1011" (0 to 11) and then repeats making them suitable for clocks.

**Synchronous Counters** use edge-triggered flip-flops that change states on either the "positive-edge" (rising edge) or the "negative-edge" (falling edge) of the clock pulse on the control input resulting in one single count when the clock input changes state. Generally, synchronous counters count on the rising-edge which is the low to high transition of the clock signal and asynchronous ripple counters count on the falling-edge which is the high to low transition of the clock signal.

|  |
| --- |
| Rising and Falling Edge Clock Signals |

It may seem unusual that ripple counters use the falling-edge of the clock cycle to change state, but this makes it easier to link counters together because the most significant bit (MSB) of one counter can drive the clock input of the next. This works because the next bit must change state when the previous bit changes from high to low - the point at which a carry must occur to the next bit. Synchronous counters usually have a carry-out and a carry-in pin for linking counters together without introducing any propagation delays.

Then to summarise:

* **Synchronous Counters** can be made from Toggle or D-type flip-flops.
* They are called synchronous counters because the clock input of the flip-flops are clocked with the same clock signal.
* Due to the same clock pulse all outputs change simultaneously.
* Synchronous counters are also called parallel counters as the clock is fed in parallel to all flip-flops.
* Synchronous binary counters use both sequential and combinational logic elements.
* The memory section keeps track of the present state.
* The sequence of the count is controlled by combinational logic.

Advantages of Synchronous Counters:

* Synchronous counters are easier to design.
* With all clock inputs wired together there is no inherent propagation delay.
* Overall faster operation may be achieved compared to Asynchronous counters.

Bidirectional Counter **Tutorial: 4 of 4**

**Count Down Counter**

As well as counting "up" from zero and increase, or increment to some value, 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**, (**CTD**). In a binary or BCD down counter, the count decreases by one for each external clock pulse from some preset value. Special dual purpose i.c's such as the TTL 74LS193 or CMOS CD4510 are 4-bit binary Up or Down counters which have an additional input pin to select either the up or down count mode.

**4-bit Count Down Counter**

|  |
| --- |
| Count Down Counter |

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 Qoutput 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.

**Bidirectional Counter**

Both Synchronous and Asynchronous counters are capable of counting "Up" or counting "Down", but their is another more "Universal" type of counter that can count in both directions either Up or Down depending on the state of their input control pin and these are known as **Bidirectional Counters**. Bidirectional counters, also known as Up/Down counters, are capable of counting in either direction through any given count sequence and they can be reversed at any point within their count sequence by using an additional control input as shown below.

**Synchronous 3-bit Up/Down Counter**

|  |
| --- |
| Bidirectional Up Down Counter |

The circuit above is of a simple 3-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 (000) to seven (111) and back to zero again. Then the 3-Bit counter advances upward in sequence (0,1,2,3,4,5,6,7) or downwards in reverse sequence (7,6,5,4,3,2,1,0) but generally, bidirectional counters can be made to change their count direction at any point in the counting sequence. An additional input determines the direction of the count, either Up or Down and the timing diagram gives an example of the counters operation as this Up/Down input changes state.

Nowadays, both up and down counters are incorporated into single IC that is fully programmable to count in both an "Up" and a "Down" direction from any preset value producing a complete **Bidirectional Counter** chip. Common chips available are the 74HC190 4-bit BCD decade Up/Down counter, the 74F569 is a fully synchronous Up/Down binary counter and the CMOS 4029 4-bit Synchronous Up/Down counter.