*Larry Kahaner WB2NEL*
*4259 Bedford Avenue*
*Brooklyn NY 11229*

# Flip Flops Exposed

The purpose of this article is not only to teach the basics of flip flops, but also to help the ham builder understand the circuits which he sees every day in magazines and construction project books.

All of us try to follow the builder's explanation of operation but may fall short of full understanding due to ignorance in certain shorthand notations or unfamiliar phrases such as "the flip flop is now set," or "it operates in a master-slave configuration," or more common, "the device will be allowed to toggle."

We decide to skip the explanation and build it anyway. It works, but we miss the joy of knowing how it works. Hopefully, these intermittent knowledge gaps now can be filled once and for all.

The flip flop has an amazing talent: a memory.

This may not seem like much to you, but for a chunk of silicon it's quite an accomplishment.

This memory is the basis of all computers and counters. After all, what is counting but remembering what you had last? And the beauty of learning about flip flops is that there is no new background material with which to grapple. All that is needed is an understanding of the TTL primer in the July issue.

Take a minute and review the four basic gates and the action of the clocking input.

There are four basic types of flip flops. They are the R-S, D,T and J-K. All have two stable states, either 0 or 1. This is known as bistable. Sometimes flip flops are referred to as bistable multivibrators. Another name you might have seen is bistable latch. They are all the same thing, a form of the flip flop.

## R-S Type

Fig. 1 shows the simple R-S flip flop or latch. The S means SET and the R means RESET. Other terms used are PRESET and CLEAR, respectively.

By convention, the outputs Q and $\overline{Q}$ cannot be equal. In fact, the bar over any letter means "not". $\overline{Q}$ is called "not Q".

To show the action, let's *assume* that Q = 1 and $\overline{Q}$ = 0. Because of the feedback loops

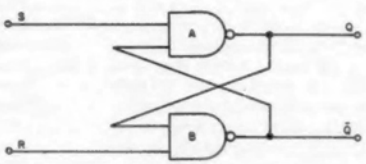


*Fig. 1. R-S flip flop.*

the B input is 1 and the A input is 0. For the initial state in the NAND flip flop (NOR gates can also be used) R and S are always 1. So we have the initial condition, as shown in Fig. 2(a).

If you recall the NAND truth table, only inputs of 1,1 yield 0; all other combinations give 1.

If we drive R to 0 the following chain takes place. The R input of 0 combines with the 1 input at B to make the output at $\overline{Q}$ = 1. This 1 feeds back to the A input and combines with the S = 1 input to make a 0 output at Q. This 0 feeds back to the input at B, as seen in Fig. 2(b).

Whew! But look what happened! Our outputs are reversed! Even if we return R to its initial state of 1, the output will still remain, as we see in Fig. 2(c).

That momentary driving of R to 0 could be a push-button or a clock pulse.

Now if we take our final condition of 2(c) and impress S to 0, the outputs will flip around again. And they will stay, even if S is returned to 1. See Figs. 2(d) and 2(e).

The name flip flop is thus shown to be quite appropriate.

A flip flop is said to be in the SET condition if the Q output is 1. The RESET condition exists if the $\overline{Q}$ is 1.

Fig. 2(a) is a SET state and 2(c) is a RESET condition. The rule: If a flip flop is SET, driving RESET to 0 will change the output. If it is RESET, driving SET to 0 will change the output.

The symmetry is beautiful. If it's SET, reset it. If it's RESET, set it.

But what happens if both R and S are impressed to 0? Well, that is the problem with the R-S latch. The answer is ambiguous. Since you never know which 0 came first, the output is unpredictable. Try it, and you'll see the fun.

A schematic diagram of a flip flop may show the actual gates or it may show a configuration such as Fig. 3(a). A personal favorite is shown in Fig. 3(b).

Either way, it indicates that something like what is shown in Fig. 3(c) may have been done with (for instance) the 7400 chip.

Not all flip flops will be made from chips containing *only* simple gates. Some ICs have

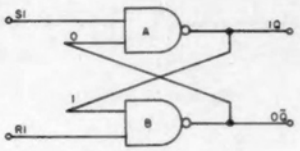


*Fig. 2(a). Initial condition.*

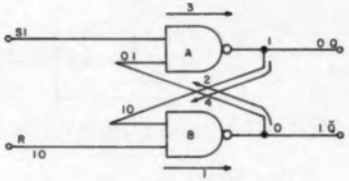


*Fig. 2(b). Follow the action!*

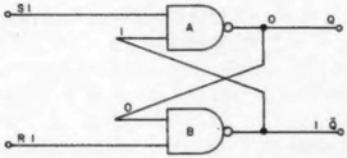


*Fig. 2(c). Outputs reverse and stay, even though R is returned to 1.*

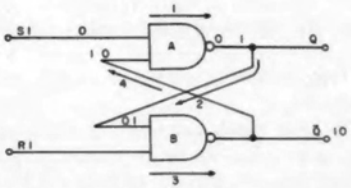


*Fig. 2(d). S goes to 0 and outputs reverse.*

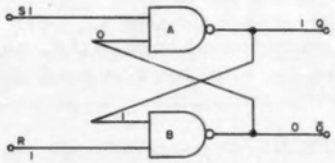


*Fig. 2(e). Outputs stay reversed, even if S is returned to 1.*


*Fig. 3(a). R-S flip flop.*

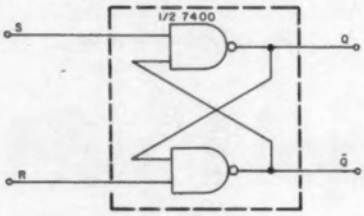


*Fig. 3(b). Flip flop showing chip it came from.*

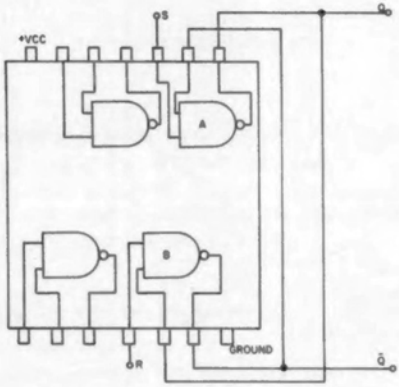


*Fig. 3(c). 7400 wired in R-S latch configuration.*

flip flops and *other* elaborate circuits within one package.

The R-S latch can also be made using NOR gates (Fig. 4). In this case the inputs are held low (0) and driven high (1) to start the flip flop function. It's just the opposite of the NAND gate flip flop, where inputs are held high (1) and driven low (0) to activate.

The NOR flip flop is not used much. It is easier to use NAND gates. They are also cheaper and more available.

In the NOR flip flop inputs of 1,1 are not allowed, just as 0,0 inputs are not allowed in the NAND configuration.

## Clocking

Two types of logic circuits are used. One is synchronous and the other is non-synchronous or asynchronous.

Asynchronous operation exists when circuits are operating independently of each other. Each individual circuit has its own input and it responds to these inputs.

Synchronous operation relies on a common input such as a clock to feed all the circuits in the system. All functions rely on the clock.

If we add a clocking input to our R-S flip flop, two additional NAND gates are used. These gates insure that the latch works only when the clock pulse (1) is present. In Fig. 5 the input of C will be 0 only if the A input is 1 and the clock pulse is 1. Remember that the R-S configuration latches only on the 0 drive – and we have supplied it.

## D Type

The second type of flip flop is the D or delay type. Since we still have the problem of the SET and RESET inputs being the same (causing an unpredictable output), the best we can do is to insure that we don't have the same inputs at the same time.

In Fig. 6 the inverter (E) negates all inputs; 0 becomes 1 and 1 becomes 0. We employ it to feed the B input. By placing it there while feeding it *and* A from the same input, we can be certain that the signals reaching the gates will be different.

Any signal going to A will remain the same and those passing through the inverter to B will change.

The D type can also have PRESET and CLEAR controls. They are superior or overriding functions. No matter what is going on, commands on these controls have priority. When either of these inputs is present (0), the output will go to 0 or 1 depending upon which function is employed.

In schematic diagrams, the letter "D" is the only indication of the flip flop type.

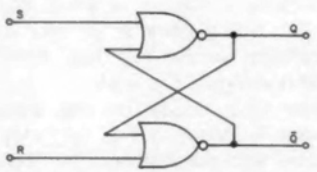


*Fig. 4. NOR gated flip flop.*

IC types which are D flip flops are 7474 (contains 2 flip flops with PRESET and CLEAR) and 7475 (contains 4 flip flops without PRESET and CLEAR).

### T Type

The toggle or T type does what its picturesque name implies – it toggles.

The logic is such that the output will change regardless of what it was prior to clocking. But this only happens if the clock is fast enough. If it is not rapid, the output will change state and then return. It will forever change and change back again.

Toggle flip flops have been assigned mixed jobs. When a clock pulse is applied, the output will change once every input cycle. Therefore, it completes one output *cycle* (not just change) for every two input cycles. This gives a divide-by-two property which is used in counters and calculators.

Another use is in random output devices like "electronic dice" and "heads-tails" circuits, which electronics magazines are so fond of printing once a year. Since it is unknown where the circuit is toggling at the moment, a stop-toggling command will produce a random output.

### J-K Type

The J-K flip flop is very widely used. The inputs J and K correspond to S and R. The gates 1-4 are the master section and 5-8 the slave section. Gate 9 is used as an inverter. (Although not shown, a constant input of 1 is kept on the other input. It will combine with the other input to invert it. 1 and 1 yield 0; 1 and 0 yield 1.)

Both sections are synchronous (use the same clock pulse) and are activated by a pulse of 1.

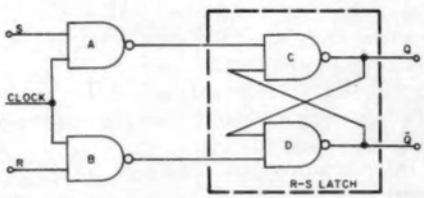


*Fig. 5. Synchronous clocking of R-S latch.*

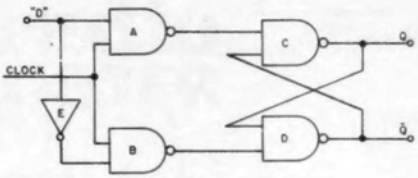


*Fig. 6(a). D type flip flop.*

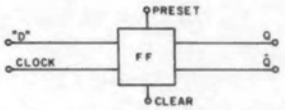


*Fig. 6(b). D type with PRESET and CLEAR.*

If the clock is 1, the master will see this and activate. However, the slave will see the inversion of this (0) because of gate 9, and not activate.

The feedback via lines a and b from slave to master will be valid until the clock pulse is 0. Then the slave will energize due to the inversion of this pulse from 0 to 1. But now the master is disabled because it sees the 0 straight from the clock.

The information is passed from master to slave. The next pulse will feed the output of the slave back to the master and so on.

A sequence of four steps takes place.

1. Isolate slave from master.
2. Enter information to master.
3. Disable master.
4. Transfer information from master to slave.

All this takes time. That is a useful property. A J-K can be used for storage of information while another circuit is doing something else. The two informations can then rendezvous at the proper time. The J-K also solves the same ambiguous input problem of the simple R-S. It's pretty good as a toggle, too.

Some examples of J-K flip flops are the 7470, 7473 and 7476. Each has a unique property such as number of flip flops, different voltage and frequency ratings, and different controls.

### How the Flip Flop Counts (or, Here's the Part We've All Been Waiting For)

We have seen that it takes two pulses to return a flip flop to its initial state. If we

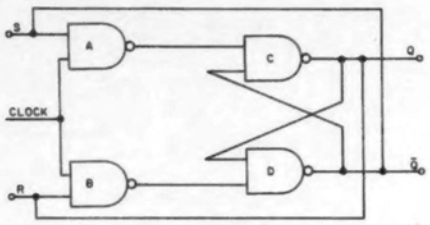


*Fig. 7(a). T type flip flop.*

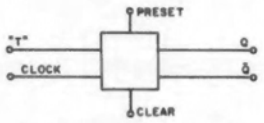


*Fig. 7(b). T type with PRESET and CLEAR.*

start with Q as 1, it will take two pulses to make it 1 again. The memory remembers the first pulse and waits for the next.

Therefore, if we monitor the output of Q we can tell when the input has completed two pulses. Since each flip flop counts by "twos", placing them together will allow the first to count by two, the second by four, the third by eight, and so on (Fig. 9).

Think of each flip flop as having two stages. Initially, all inputs and outputs are 0. Pulse 1 will energize $a_1$. Pulse 2 will return the first flip flop to its initial condition and store a pulse at $b_1$. Pulse 3 activates $a_1$

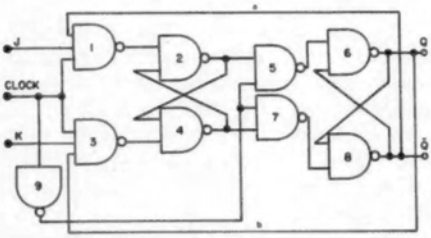


*Fig. 8(a). J-K flip flop made from R-S latches.*

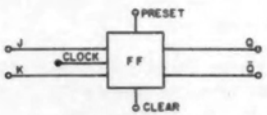


*Fig. 8(b). J-K type with PRESET and CLEAR.*

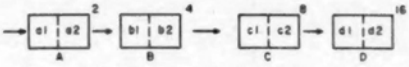


*Fig. 9. Flip flop counter.*

again. Pulse 4 flips section A and passes a pulse to b, where it combines with the pulse already there to flip section B. Our monitor at the output of B rings, buzzes or lights, and we know we have counted to 4.

After 16 pulses the entire counter is returned to 0. We have counted to 16!

## Experimentation

Many readers have asked how to convert this type of "pure" knowledge into practical application. The IC mystique still lives! Just experiment. Buy some ICs and hook them up. See what happens when you do "this" or "that". There is no great mystery. They will work – really!

Here are some suggestions to help you play around.

1) Get a 5 volt power supply. It's cheap if you build one. The October, 1974 issue of *73* has a fine one. If you like printed circuits, Radio Shack has a board which is almost the same (except for part values). Parts are very cheap from the advertisers in *73*.

2) Monitor the output with some light emitting diodes (LEDs). They tell you if the output is high (1) or low (0).

3) Get data sheets on ICs that you wish to work with. They have the circuit diagrams of the internal gates so you know how to hook them up. They also give 0 and 1 values.

4) Never be embarrassed to return non-working chips. The same mass production techniques which brought down the price also brought down the quality. Don't get worried, though: *Most* ICs *will* function properly, but if they don't, return them. I have *never* had a store *not* replace a non-working chip. Even the mail order houses exchange them without a hassle.

5) New breadboard kits have been arriving on the markets. I haven't tried them, but they look good.

6) ICs are rugged. Don't be afraid of them.

**. . . WB2NEL**