Redstone circuits/Memory S re article feedback

< Redstone circuits
v t e *This article is about a specific category of redstone circuits. For other circuits, see redstone circuit.*

> **Note: This page uses many schematics, which are loaded individually for performance reasons. [Schematic Help]**

Memory circuits are circuits that maintain their state and can be used to store data. In simple terms, a memory circuit is continuously ON or OFF until it receives a signal telling it turn OFF or ON, where ON means the circuit is producing a redstone signal, and OFF means it produces no signal.

Most circuits produce the same output for a particular input, even when combinations of these circuits are used; in electronics, this is called "combinatorial logic". The output of a memory circuit depends on the current input and the past input; the output of a memory circuit can be different each time the circuit is activated. This behavior is called "sequential logic" in electronics and allows memory circuits to be used to create counters, long term clocks, and complex memory systems.

Contents

Terminology

Memory circuits can be complex, and there are many technical terms used to describe them. Below is a list of some terms and symbols commonly encountered when discussing memory circuits.

Signals

ON and OFF
When discussing circuits, it is common to say that an input or output is "ON" when it supplies a redstone signal, and to say it is "OFF" when it does not supply a signal. For many circuits, the strength of the input signal does not affect the circuit's operation, and the output is usually a full strength redstone signal (signal strength 15). Circuits that depend on signal strength in some way usually use a data (D) input.

Rising edge
A rising edge signal is when a redstone signal goes from 0 to a positive number.

Falling edge
A falling edge signal is when a redstone signal goes from a positive number to 0.

Edge triggered
The circuit changes its output when first receiving a redstone signal, or when it stops receiving a signal. That is, the circuit reacts to a rising or falling edge signal.

Level triggered
The circuit is allowed to continually change its output while a clock (C) (sometimes called a gate) supplies a redstone signal. These circuits usually have another input called data (D) that determines what the output is.

Analog
An analog circuit uses the input signal strength in some way.

Digital
A digital circuit is only described as ON or OFF (that is, whether there is a redstone signal or not), not the strength of the signal.

Inputs and outputs (S,R,T,C,D,J,K,Q)

Set (S)
Some circuits have an input called "set" or "S" that turns the circuit ON.

Reset (R)
Some circuits have an input called "reset" or "R" that turns the circuit OFF.

Toggle (T)
Some circuits have an input called "toggle" or "T" that changes the circuit from OFF to ON, or ON to OFF.

Clock (C)
Some circuits have an input called "clock" or "C". C is a clock signal, which is a redstone signal that turns ON and OFF at regular intervals, and is used to control the circuit's behavior.

Data (D)
Some circuits have an input called "data" or "D". D is a redstone signal of a particular strength. For many circuits, the strength of the input signal does not affect the circuit's operation, and if the circuit outputs a signal, it is usually a full strength signal (signal strength 15). When a circuit with a D input produces an output, the output can be the same strength as D and is not necessarily a full strength signal. Circuits with a data input often also have a clock (C) input.

J
JK latches and flip-flops have an input called "J". J behaves similarly to set (S). J is sometimes referred to as "jump", but does not have a common name associated with it like other inputs (such as set, reset, toggle, etc.) do.

K
JK latches and flip-flops have an input called "K". K behaves similarly to reset (R). K is sometimes called "kill", but does not have a common name associated with it like other inputs (such as set, reset, toggle, etc.) do.

Q (output)
Q is the symbol commonly used to denote output. Some circuits have two outputs: Q and its opposite ($\overline{Q}$). So if Q is ON, $\overline{Q}$ is OFF, and vice versa.

Latches and flip-flops

Latches and flip-flops
Latches and flip-flops are circuits that stay ON or OFF, until a signal tells them to turn OFF or ON. In electronics, flip-flops often refer to edge-triggered circuits, and latches often refer to level-triggered circuits; however, this naming convention is not always strictly followed in electronics, and may not be followed in this article.

T flip-flops
T flip-flops only have a toggle (T) input.

RS latch
An RS latch has set (S) and reset (R) inputs. Usually, the R and S inputs are not allowed to be ON at the same time.

RS NOR latch

An RS NOR latch is an RS latch that uses the NOR logic gate in its construction. There are variations of this circuit that use different logic gates.

D flip-flops and latches

D flip-flops and latches have a data (D) input, and usually a clock (C) input that determines when the circuit's output is updated. They output can be the same signal strength as D and not necessarily a full strength signal.

Gated D Latch

A gated D latch specifically refers to a level-triggered circuit that has a data (D) input and a clock (C) input. The clock input acts as a "gate" that allows the circuit to continuously change its output while the clock is supplying a redstone signal. The output remains frozen while the clock is off. The output signal is determined by the data input.

JK latches and flip-flops

JK latches and flip-flops have J and K inputs, and usually have a clock (C) input. JK latches and flip-flops behave similarly to RS latches, with J being analogous to S, and K being analogous to R. However, JK latches and flip-flops can also be toggled when both the J and K inputs are turned on.

Overview of memory circuits

Memory circuits are modeled after circuits from real life electronics. Much of the same terminology used in electronics is used in this article to describe memory circuits in *Minecraft*. For example, circuits can be digital, where input and output are only described as ON or OFF, or analog, where input and output can vary with signal strength.

Memory circuits can usually be classified as "latches" or "flip-flops". Flip-flops and latches are circuits that can stay ON or OFF until they receive a signal that tells them to turn OFF or ON. In electronics, flip-flops are often described as circuits that change their output when the input signal changes from OFF to ON or ON to OFF; this is referred to as being "edge-triggered". Latches are often described as circuits with two inputs: one input (sometimes called a "gate") determines if the circuit is allowed to change its input, the other input determines what the output will change to. While the gate input is ON, the circuit can continuously change it output; this is called level-triggered. These naming conventions are not always strictly followed in electronics, and may not be strictly followed in this article.

There are several categories of memory circuits; they are usually named by whether they are a flip-flop or latch and by what inputs they use. For example, an RS latch is a latch that has set (S) and reset (R) inputs. Some circuits have a single output, usually denoted as Q. Some circuits are dual output and have a second output dentoted by $\overline{Q}$, which is the opposite of Q; for example, if Q is ON, $\overline{Q}$ OFF.

Common categories of memory circuits

- RS latches and flip-flops
  - two inputs: S (set) and R (reset), that turn the circuit ON and OFF; S and R are not usually allowed to be ON at the same time
- T latches and flip-flops
  - one input: T (toggle), that changes the output from OFF to ON, or from ON to OFF
- D latches and flip-flops
  - two inputs: D (data) and C (clock); C determines if the circuit can change its output, D

determines the circuit's output

- JK latches and flip-flops
  - usually three inputs: J, K, and C (clock); C determines if the circuit is allowed to change its output, J turns the circuit on, K turns the circuit off; if J and K are both ON, the circuit toggles from ON to OFF or OFF to ON.

Truth tables

A truth table serves as a simple summary of the circuit's logic by showing all of the possible combinations of inputs and outputs. Usually "1" is used to denote ON, and "0" is used to denote OFF. Truth tables are used in Boolean algebra to study systems with two states, usually true and false, or on and off.

For example, the truth table for a single output RS latch is:

S	R	Q
0	0	Q
1	0	1
0	1	0
1	1	*undefined*

When S=0, and R=0, the circuit's output is unchanged, and may be ON or OFF

When S=1, and R=0, the circuit is turned ON if it was OFF, or stays ON if it was already ON

When S=0, and R=1, the circuit is turned OFF if it was ON, or stays OFF if it was already OFF

When S=1, and R=1, the circuit doesn't have defined behavior

- undefined means the circuit wasn't designed for this combination of inputs in mind and this situation should be avoided; what actually happens when both inputs are ON depends on the specifics of how the circuit was built

RS latches

An RS latch has two inputs, S and R. The output is conventionally labeled Q, and there is often an optional "inverse output" $\overline{Q}$. (Having both Q and $\overline{Q}$ is called "dual outputs"). When a signal comes into S, Q is **s**et on and stays on until a similar signal comes into R, upon which Q is **r**eset to "off". $\overline{Q}$ indicates the opposite of Q — when Q is high, $\overline{Q}$ is low, and vice versa. Where a $\overline{Q}$ output is available, the player can often save a NOT gate by using it instead of Q.

Note that the proper name for this category of latch is "SR latch". However, in real-world electronics as in Minecraft, the classic implementation of such latches starts by inverting the inputs; such a latch is the proper "RS latch", but they're so common that the term is commonly used also for what "should" be called SR latches.

Typical uses include an alarm system in which a warning light stays on after a pressure plate is activated until a reset button is pushed, or a rail T-junction being set and reset by different detector rails. RS latches are common parts of other circuits, including other sorts of latches.

Setting both inputs high simultaneously is a "forbidden" condition, generally something to avoid. In the truth table, S=1, R=1 breaks the inverse relationship between Q and Q. If this happens, the player gets "undefined behavior" — various designs can do different things, and especially Q and $\overline{Q}$ can be high or low at the same time. If the forbidden state is co-opted to *toggle* the output, the circuit becomes a JK latch, described in its own section. If there is instead a third input that **t**oggles the output, the circuit becomes an "RST latch".

Any RS latch with dual outputs is functionally symmetrical: pulsing each input turns on "its" output, and turns off the other one. Thus R and S are interchangeable, if the outputs is swapped: The input players choose as S determines which of the outputs is Q, then the other input is R and the other output is $\overline{Q}$. (If the original circuit had only a Q output, then swapping the inputs transforms it into $\overline{Q}$.) In several designs (A, B, C, D, E, F, I) the functional symmetry is reflected by the circuit's physical symmetry, with each input energizing the torch it leads to, while turning off the other.

RS latches can be built in a number of ways:

- Two NOR gates can be linked so that whichever is lit, the other is off. The RS NOR latch is the "original" RS latch, and still among the smallest memory devices that can be made in vanilla *Minecraft*. While they can be built with just torches and redstone dust, repeaters can also be used. Many of these designs have "duplex I/O"—the same locations can be used to read or set the latch state.
- It is also possible to construct an RS NAND latch, using NAND gates instead of NOR gates. These are larger and more complex than an RS NOR latch, but may be useful for specialized purposes. Their inputs are inverted (see below for details).
- Other RS latches can be created by fitting an "input sustaining circuit" with a reset switch, say by adding a pair of NOT gates or a piston, placed so as to interrupt the circuit when triggered. Such a construction can be nearly as compact as an RS NOR latch (and often with better I/O isolation and/or timing), but they usually do not have a natural $\overline{Q}$ output.
- Other devices can also be involved. Pistons can be used to physically toggle a block's location, while hoppers or droppers can pass around an item entity. These circuits can be fast and small, with little redstone dust.

S	R	$\overline{S}$	$\overline{R}$	Q	$\overline{Q}$
1	1	0	0	Undefined	Undefined
1	0	0	1	1	0
0	1	1	0	0	1
0	0	1	1	Keep state	Keep state

RS-NOR latches

Basic RS-NOR Latches

Designs **A** and **B** are the most fundamental RS-NOR latches. In both cases, their inputs and outputs are "duplex"—the latch state can be read (Q) or set (S) on one side of the circuit, while on the other side, the latch can be reset (R), or the inverse output read ($\overline{Q}$). If separate lines for input and output are needed, opposite ends of **B** can be used, or **A** can be elaborated into **A'** with separate locations for all four lines.

Isolated RS-NOR Latches

These can be modified to provide separate, even isolated, input and output. **C** and **D** use torches and repeaters respectively to isolate the outputs, though the inputs can still be read. **E** expands the circuit slightly to isolate all four I/O lines.

Vertical RS-NOR Latches

Design **F** provides a vertical (1-wide) option; again, the I/O is duplex, though isolated outputs can be taken at alternate locations.

Design **G** takes up more room than **F**, but may be preferable, as both the set and reset are on the same side. Also, be sure to compensate for the extra tick on ($\overline{Q}$), caused by the last torch.

Design **H** is smaller than design **F** in term of height, input and output are on the same height, but it is longer and a bit slower due to the repeater.

Furthermore, it is easily stacked vertically and horizontally (with a shift of 2 blocks on the Y axis).

Design **I** is similar to design **G** as it has both set and reset on the same side,but takes up less space. The I/O is duplex, though isolated outputs can be taken at alternate locations.

Design **J** is similar to design **G** as it has both set and reset on the same side, but has no slowness due to not having any extra repeaters or torches. This may be more preferable to **G**, although the outputs (Q /$\overline{Q}$) are not level with the inputs (R/S).

RS NAND latches

An RS latch can also be designed using NAND gates. In *Minecraft*, these are less efficient than the RS NOR latch, because a single Redstone torch acts as a NOR gate, whereas several torches are required to create a NAND gate. However, they can still be useful for specialized purposes.

Such an "RS NAND latch" is equivalent to an RS NOR, but with inverters applied to all the inputs and outputs. The RS NAND is logically equivalent to the RS NOR, as the same R and S inputs give the same Q output. However, these designs take *inverse* R and S (R̅, S̅) as inputs. When S̅ and R̅ are both off, Q and $\overline{Q}$ are on. When S̅ is on, but R̅ is off, $\overline{Q}$ is on. When R̅ is on, but S̅ is off, Q is on. When S̅ and R̅ are both on, it does not change Q and $\overline{Q}$. They are the same as they were before S̅ and R̅ were both turned on.

RS-NAND Latches

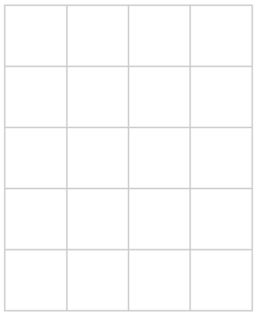
RS-Latch summary table 1

This table summarizes the resources and features of the RS latches that use only redstone dust, torches, and repeaters.

Design	A	B	A'	C	D	E	F	G	H
Size	4×2×3	3×2×3	4×4×3	2×3×3	2×3×2	2×4×2	3×**1**×4	5×3×3	6×3×3
Torches	2	2	2	2	2	2	2	4	6
Redstone wire	6	4	10	4	0	4	3	6	8
Repeaters	0	0	0	0	2	0	0	0	0
Inputs isolated?	Duplex	Duplex	Duplex	Duplex	Yes	Yes	Duplex	Yes	Yes
Outputs isolated?	Duplex	Duplex	Duplex	Yes	Yes	Yes	Duplex/ Yes	No	Yes
Input orientation	opposite	adjacent	opposite	opposite	opposite	opposite	opposite	perpendicular	perpendicular

Analog RS latch

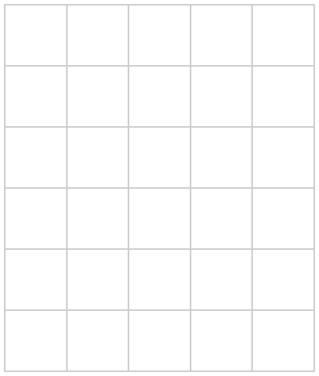
This latch maintains the highest signal level that arrived from input **S** if **R** is off, and fade (reduce memorized signal strength) by strength of **R** every two redstone ticks. For maximum strength (15) signals it behaves like any other RS latch, but it can also memorize intermediate signal levels, and because 2-tick pulses on **R** subtract their strength from its memorized state, it makes a nice element of counter or countdown circuits.


Analog RS latch

Analog counter[1]

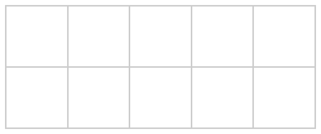
This variant of the analog RS latch adds an "increment" (**I**) and "decrement" (**D**) input. While input **I** is active, memorized signal strength is increased by 1 every two redstone ticks until reaching maximum (15) signal strength. While input **D** is active, signal strength is decreased by 1 every two redstone ticks until output **Q** is unpowered.

Output **Q** is duplex and can also be used as an input (**S**); the highest signal strength at this block is memorized.


Analog counter

Input stabilization with reset

An "Input-Stabilizing Circuit" responds to an input pulse by turning its input on and leaving it on. This can be built up into an RS Latch by adding a means to turn it off. These circuits usually don't offer a "natural" $\overline{Q}$ output. Design **J** adds a pair of NOT gates, with the reset going to the second torch. (The NOT gates can also be added to the upper redstone loop.) Design **K** uses its piston to block the circuit where it goes up onto the solid block. Design **L** shows the reverse approach, breaking the circuit by withdrawing a power-carrying block.


Input Stabilization Circuit

RS-ISR Latches

Pistons and other devices

Other RS Latches

A pair of non-sticky pistons can be used to physically push a block back and forth. This can make or break a circuit from a torch, producing an RS latch with no inverse output (**M**). If the block being pushed is a block of redstone, the circuit can be even smaller, with dual outputs (**N**). Both of these have isolated inputs and outputs. Putting *two* blocks between the pistons produces an SRT latch **O**, with an extra input to toggle the latch state. And droppers can also be pressed into service, as in design **P**: Small, tileable, but it does require a comparator.

Variations

- Expand an RS latch easily into a monostable circuit, which automatically disables itself some time after activation. To do this, split the output redstone path into two parts. The new path should run through some repeaters, and in to the reset input. When players turn on the latch, redstone feeds a signal through the delay before turning off the latch. This works not only for Q and R, but for $\overline{Q}$ and S as well. A more complex delay mechanism, such as a water clock, can replace the repeaters.
- An "Enable/Disable RS latch" can be made by adding a pair of AND gates in front of the inputs, testing each of them against a third input, E. Now if E is true, the memory cell works as normal. If E is false, the memory cell does not change state. That is, E latches (or equivalently, clocks) the RS latch itself. Note that for design **Q**, the outputs are *not* isolated, and a signal to them can set the latch regardless of E. Alternatively, repeaters could be used to latch the inputs, but this costs more and saves no space.
- As noted above, if it is possible to add a "toggle" input, the RS latch becomes an RST latch. If the "forbidden" state is used for the toggle, then it's a JK latch.

Dropper SR latch

Allows a lot of flexibility in geometry — the droppers can be read from 3 sides each and activated from 5 sides each; can be oriented vertically too and content can be read with comparators through solid blocks. However, always power it through an adjacent block; if players power the dropper directly, they activate the other dropper too and the order is unpredictable. Activates on rising edge, meaning they can apply S even while R is still active or vice versa.

Enable/Disable RS Latch

RS latch summary table 2

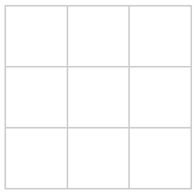
Design	J	K	L	M	N	O	P	Q
Size	2×3×3	4×3×3	4×4×2	4×3×2	4×**1×1**	5×3×3	3×**1**×2	5×5×3
Torches	2	0	1	1	0	1	0	7
Dust	7	4	6	0	9	4	0	7
Repeaters	1	1	1	1	0	1	0	0
Other devices	--	1 sticky piston	1 sticky piston	2 normal pistons	2 normal pistons	2 normal pistons	2 droppers, 2 comparators	N/A
Inputs isolated?	Yes,	No	No	Yes	Yes	No	Yes	Yes
Outputs isolated?	Yes	No	No	Yes	Yes	Yes	Yes	No
$\overline{Q}$ available?	No	No	No	No	Yes	No	Yes	Yes
Input orientation	Perpendicular	Perpendicular	Adjacent	Opposite	Opposite	Opposite	Adjacent	Adjacent

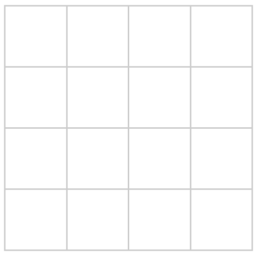
D latches and flip-flops

A D ("data") flip-flop or latch has two inputs: The data line D, and the "clock" input C. When triggered by C, the circuits set their output (Q) to D, then hold that output state between triggers. The latch form, a "gated D latch", is level triggered. It can be high- or low-triggered; either way, while the clock is in the trigger state, the output changes to match D. When the clock is in the other state, the latch holds its current state until triggered again. A D flip-flop is edge triggered; it sets the output to D only when its clock input changes from "off" to "on" (rising edge) or *vice versa* (falling edge), according to the circuit. An edge trigger can turn a gated D latch into a D flip-flop.

Building these devices with torches is fairly unwieldy, though some older designs are given below. Repeaters have a special latching ability, which drastically simplifies the problem. Now a gated D latch can be made with two repeaters, and a D flip-flop with four repeaters and a torch:

Design **G** uses the repeater's latching feature, which is added to the game in Java Edition 1.4.2. It holds its state while the clock is high, and is by far the most compact of the D latch designs. Design **H** combines two such latches, one high and one low triggered, to create a rising edge-triggered D flip-flop. The block and redstone torch can be reversed for a falling edge-triggered design. The design is based on a real life implementation of an edge-triggered D flip-flop called a "Master-Slave" configuration.

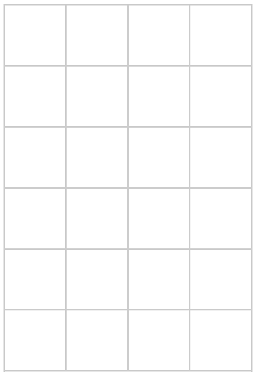

Modern Gated D Latch (G) (High level)


Modern D Flip-flop (H) (rising edge)

Analog D latch

6×4×2 (48 block volume)
flat, silent
circuit delay: 3 ticks

Earliest Known Publication: May 26, 2018[2]


Analog D latch (J) (Low level)

Design **J** is an analog version of a low-triggered D latch. The signal strength of the output Q is the same as input D when the latch is triggered.

For maximum strength (15) signals for D, this latch behaves like a normal (digital) low-triggered D latch.

Torch-based designs

For historical interest, here are several older designs, not dependent on latched repeaters, along with a table of their resource needs and other characteristics. A few of these designs also have the additional inputs and inverse output of an RS latch.

This basic level-triggered gated D latch (design **A**) sets the output to D as long as the clock is set to OFF, and ignores changes in D as long as the clock is ON. However, on a rising clock edge, if D is low, the output pulses high for 1 tick, before latching low.

Design **B** includes a rising-edge trigger and it sets the output to D only when the clock goes from OFF to ON. The torch-based edge trigger could also be replaced with one of the designs from the Pulse circuit page.

These are RS latch-based circuits with appropriately set front-ends. Directly trigger the RS latch using the R and S inputs to override the clock, forcing a certain output state. Sending signals into the Q and $\overline{Q}$ lines works similarly, because the output is not isolated. To isolate the outputs, add inverters and swap the labels.

D Latch A

D Latch B

Design **C** is a one block wide vertical version of **A**, except for using a non-inverted clock. It sets the output to D while the clock is ON (turning the torch off). This design can be repeated in parallel every other block, giving it a much smaller footprint, equal to the minimum spacing of parallel data lines. A clock signal can be distributed to all of them with a wire running perpendicularly under the data lines, allowing multiple flip-flops to share a single edge-trigger if desired. The output $\overline{Q}$ is most easily accessed in the reverse direction, toward the source of input. As in design **A**, the un-isolated Q and $\overline{Q}$ wires can do double duty as R and S inputs. Q can be inverted or repeated to isolate the latch's Set line.

D Latch C

D Latch D

Design **E** provides a more compact (but more complex) version of **A**, while still affording the same ceiling requirement. **E'** allows the latch to act on a high input.

Design **F** holds its state while the clock is high, and switches to D when the clock falls low. The repeater serves to synchronize the signals that switch out the loop and switch in D. It must be set to 1 to match the effect of the torch.

D Latch E

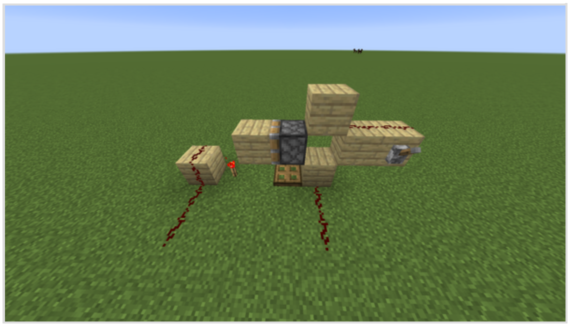
D Latch F

Design	A	B	C	D	E	E'	F	G	H
Size	7×3×3	7×7×3	6×**1**×5	5×2×6	5×3×3	5×3×3	5×3×3	2×1×2	3×2×2
Torches	4	8	5	6	4	5	4	0	1
Redstone wire	11	18	5	6	10	9	7	0	0
Repeaters	0	0	0	0	0	0	1	2	4
Trigger	Low Level	Rising Edge	High Level	High Level	Low Level	High Level	Low Level	High Level	Rising Edge
Output isolated?	No	No	No	No	No	No	Yes	Yes	Yes
Input isolated?	Yes	Yes	C Only	C Only	Yes	Yes	No	Yes	Yes

BUD-based D flip-flop

BUD-based D Flip-flop (I) (low level)

Design **I** represents an entirely different form of the D flip-flop, based on the principle of the block update detector. This flip-flop is small so it can be used multiple times at large integrated redstone circuits. Note that no blocks that are adjacent to the piston can be used as circuit components except flip-flop itself.


Piston BUD based D Flip-flop

The lever in the screenshot shown is the D input. The redstone wire in the middle is trigger signal input. The trapdoor is part of the BUD – it needs to be replaced by a note block, an activator rail, etc.

JK flip-flops and latches

A JK flip-flop is another memory element that, like the D flip-flop, changes its output state when triggered by a clock signal C. They can be edge-triggered (designs **A**, **D**, **E**) or level-triggered (**C**). Either way, the two inputs are called J and K. These names are arbitrary, and somewhat interchangeable: if a $\overline{Q}$ output is available, swapping J and K also swaps Q and $\overline{Q}$.

When the flip-flop is triggered the effect on the output Q depends on the values of the two inputs:

J	K	Q_{next}
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

- If the input J = 1 and the input K = 0, the output Q = 1.
- When J = 0 and K = 1, the output Q = 0.
- If both J and K are 0, then the JK flip-flop maintains its previous state.
- If both are 1, the output complements itself — that is, if $Q = 1$ before the clock trigger, $Q = 0$ afterward.

The table summarizes these states — note that $Q(t)$ is the new state after the trigger, while $Q(t-1)$ represents the state before the trigger.

The JK flip-flop's complement function (when J and K are 1) is meaningful only with edge-triggered JK flip-flops, because it is an instantaneous trigger condition. With level-triggered flip-flops (e.g. design C), maintaining the clock signal at 1 for too long causes a race condition on the output. Although this race condition is not fast enough to cause the torches to burn out, it makes the complement function unreliable for level-triggered flip-flops.

The JK flip-flip is a "universal flip-flop", as it can be converted to any of the other types: It's already an RS latch, with the "forbidden" input used for toggling. To make it a T flip flop, set **J = K = T**, and to make it a D flip-flop, set K to the inverse of J, that is **J = $\overline{\textbf{K}}$ = D**. In the real world, mass production makes JK latches useful and common: a single circuit to produce in bulk, that can be used as any other sort of latch. In *Minecraft*, however, JK latches are generally larger and more complex than the other types, and using their toggle function is awkward. It's almost always easier to build the specific latch type needed. Notably, an SRT Latch has all the same abilities, but gets the toggle function from a separate input.

Design **E** is a vertical JK Flip-Flop from the basis of design A.

Aside from these redstone designs, it is also possible to make a JK flip-flop by modifying a rail toggle, or with newer components such as hoppers and droppers.

JK Latch A

JK Latch C

JK Latch D

JK Latch E

Design table

Design	A	C	D	E
Size	9×2×11	7×4×5	5×2×7	14×10×1
Torches	12	11	8	10
Redstone	30	23	16	24
Repeaters	0	0	6	6
Accessible $\overline{Q}$?	No	Yes	Yes	No
Trigger	Edge	Level	Edge	Edge

T flip-flop

T flip-flops are also known as "toggles". Whenever T changes from OFF to ON, the output toggles its state. A useful way to use T flip-flops in Minecraft could, for example, be a button connected to the input. When players press the button the output toggles (a door opens or closes), and does not toggle back when the button pops out. These are also the core of all binary counters and clocks, as they function as a "period doubler", releasing one pulse for every two received.

There are many ways to build a T flip-flop, ranging from torches and dust through pistons to more exotic devices. Many designs depend on a quirk in sticky-piston behavior, namely that after pushing a block, a sticky piston lets go of it if the activating pulse was 1 tick or less. This allows short pulses to toggle the position of a block, which is useful here.

The simplest T-flip-flop design however is using a simple copper bulb with a comparator facing out of it..

Best in class TFF designs

These are designs that seem superior in various categories.

T Latch L3

T Flip-flop L4

T Flip-flop L5

T Flip-flop L6

T Flip-flop L7

L3 is a latch, which responds to a high level. Like most T latches, if the toggle line is held high too long, it oscillates, toggling repeatedly. A stone button produces a single pulse, while a wooden button's pulse is long enough to cause oscillation.

L5 is a true flip-flop with the same footprint as the **L3** (but higher), which triggers on a rising edge. Both are extremely compact, thanks to the use of latched repeaters.

L6 is a compact 1-high adaptation of D flip-flop **H**. The video shows **L6** and a similar T flip-flop.

L4 and **L7** are basically two opposite halves of the same machine — both are extremely compact and customizable tick-wise but **L4** is made for off-pulses with durations ranging from 2 to 8 redstone ticks while **L7** is made for on-pulses with durations that are 9+ redstone ticks long, which includes the 10-tick stone button. Customizing each requires changing the repeater delay or adding repeaters to match the trigger duration.

To customize **L4** for your use, adjust the top most repeater according to the duration of your trigger, as shown in the table below:

Off-pulse Duration (Redstone Ticks)	Recommended Setting for Output Repeater
1	N/A - Use a Sticky Piston
2	1
3	2
4	2
5	3
6	3
7	4
8	4
9+	N/A - Use TFF O or L7

L6 and another TFF (view on YouTube (https://youtube.com/watch?v=NkTcg3h2Ma8)) [show]

Piston TFF designs

This design doesn't use the quasi-connectivity effect, so it works in both *Bedrock* and *Java* editions. It uses a pulse generator that feeds into repeaters that power the piston through a solid block and an underground redstone dust patch. The redstone block position is the output value of the TFF. This design requires one sticky piston (for the repeater) and two non-sticky pistons, and a 6×6 area, which is linear-tilable so that the output of one TFF feeds into the next TFF.

Bedrock Edition Piston TFF

The following designs work in *Java Edition* but may present difficulties in *Bedrock Edition*.

Linear tilable TFF M

3×3 piston TFF N

2 piston QC TFF O

TFF R

Design **M** is a 1-wide dual-piston design, which can be tiled adjacent to each other for compact circuitry. (If they *don't* have to be right next to each other, dust can be used instead of the input and output repeaters.) The hidden piston forms a simple monostable circuit that cuts off the button signal (10 ticks or so) as soon as a 1-tick signal has passed through to the second repeater. Due to the piston quirk mentioned above, this 1-tick signal lets the main piston toggle the position of its mobile block, to set or unset the latch and the output. It can be made more compact by removing the last block, the repeater and the torch and replacing the block in front of the last piston with a redstone block.

That linear design can also be bent into a 3×3 square, as **N**. (The "any" blocks can be air, and that torch can just as well be on the ground.) Tiling design **N** is a little tricker, but it can be done in either horizontal direction, by mirroring adjacent copies. Note that the output can be taken from whichever side of that corner is free, but the player needs repeaters to keep adjacent outputs from cross-connecting.

Design **O**, based on the quasi-connectivity effect that works only in *Java Edition*, uses a redstone block that swaps positions when the top dust receives a signal; it is a dual piston design that uses only two pistons, two torches, two dust, and two solid blocks and a redstone block. While one of the most compact designs; using only 10 blocks of space before inputs and outputs in addition to being 1 wide and vertical, it also requires no slime balls and uses few resources aside from the redstone block while allowing for four areas to input and 4 areas to output (if repeaters are used for the output, 2 if not), in addition it can be built in the air since it doesn't have any redstone or repeaters that require placement on the ground. The design toggles on a falling edge.

Design **R** is a variation of design O, and it adds the ability to reset the output to 0, using the input R.

Observer TFF designs (*Java Edition*)

T Flip-flop O1

T Flip-flop O2

T Flip-flop O3

Those designs make use of observers and the quasi-connectivity effect. Designs **O1** and **O2** work for a rising signal, while the **O3** toggles on a falling signal.

Design table

Design	O1 horizontal	O1 vertical	O2	O3
Size	6×1×1	3×4×1	5×2×1	4×2×1
Observers	1			
Redstone blocks	1			
Sticky pistons	2			
Trigger	rising			falling
Delay	2			

Other conventional TFF designs

This section needs cleanup to comply with the style guide. [discuss]
Please help improve this page. The talk page may contain suggestions.
Reason: There are many designs here that are not well documented, and some may be redundant or broken. Any help in describing or testing circuits would be appreciated.

T Flip-flop A

T Flip-flop B

T Latch D

T Flip-flop E

T Flip-flop J

T Flip-flop K

T Flip-flop P

Design **A** demonstrates that a TFF can be made solely with redstone dust and torches, but it sprawls over 9×7×3 blocks. Design **B** is slightly unreliable for long pulses; while the input is on, the piston toggles every time the block below the piston arm is updated.

Design **D** (another torches-and-dust design, but vertical) does not have an incorporated edge trigger and toggles multiple times unless the input is passed through one first. Design **E** adds such a trigger (and a repeater).

Designs **J** and **K** make more use of repeaters, but not as latches, and they are still quite large.

T Flip-flop L1

T Flip-flop L2

Design **L2**, (also **L3**, **L4**, and **L5** above) relies on the redstone repeater locking mechanic introduced in Java Edition 1.4.2. **L4** is the smallest, but requires a piston and activates on a falling edge.

T Flip-flop Z3

T Flip-flop Z4

T Flip-flop Z5

TFF summary table

> These tables are incomplete, and need more data.

Design	A	B	D	E	J	K	M	~~O~~	P	R
Size	7×9×3	5×6×3	**1**×7×6	**1**×11×7	3×7×3	3×7×3	**1**×7×3	3×4×4	5×5×2	4×5×4
Redstone wire	28	14	9	13	11	9	0	2	8	8
Torches	10	4	7	12	5	5	1	3	7	4
Repeaters	0	0	0	1	3	2	3	0	0	1
Other Devices	none	1 SP	none	none	none	none	2 SP	2 P	none	3 SP
Input isolated?	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No
Output(s) isolated?	No	No	No	No	$\overline{Q}$ only	No	Yes	No	No	No
$\overline{Q}$ available?	No	No	No	No	Yes	No	No	No	Yes	No
Trigger	rising	rising	rising	rising	rising	rising	rising	falling	rising	falling for both T and R
Delay	4				3	4	3	1		3 for R, 1 for T
Cycle time										
Other		BUD					tilable			

Design	L1	L2	L3	L4	L5	L6
Size	3×6×3	3×5×2	3×4×2	2×3×1	3×4×3	4×4×2
Redstone wire	4	6	2	2	4	4
Torches	4	2	2	0	2	2
Repeaters	4	3	3	3	4	4
Other devices	1 SP	none	none	none	none	none
Input isolated?	Yes	Yes	Yes	No	Yes	Yes
Output(s) isolated?	Yes	Yes	Yes	Yes	$\overline{Q}$ only	No
$\overline{Q}$ available?	Yes	No	No	No	Yes	Yes
Trigger	rising	rising	high	falling	rising	rising
Delay		3	5	1/2 Trigger Duration	4 ($\overline{Q}$)	4
Cycle time				Trigger Duration	6	

Design	~~Z1~~	~~Z2~~	Z3	Z4	Z5
Size	3×3×3	3×5×3	**1**×6×5	3×5×3	**1**×5×4
Redstone wire	4	4	4	4	2
Torches	2	3	3	3	2
Repeaters	1	2	2	2	2
Other devices	1 SP	1 SP	1 SP	1 SP	1 SP
Input isolated?	Yes	Yes	Yes	Yes	Yes
Output(s) isolated?	Yes	Yes	Yes	Yes	Yes
$\overline{Q}$ available?	No	No	No	No	No
Trigger					
Delay					
Cycle time					

Size
"In a void", that includes required blocks supporting redstone.

Delay
The number of ticks from the trigger to switching the output.

Cycle time
How often the latch can toggle, including any recovery time. This is the *period* of the fastest clock that can drive it.

Other Devices
P == normal piston, **SP** == sticky piston, **C** == comparator, **H** == hopper, **D** == dropper.

Trigger
rising edge (the usual), **falling** edge, **high** or **low** level. Level-triggered TFFs oscillate on long pulses.

Rail and exotic TFFs

Basic Rail TFF (A)

Pressure-Plate Rail TFF (B)

The rail T flip-flop is a T flip-flop that uses rails and redstone. The general design uses a length of track that is stopped by a block at both ends. When the T flip-flop is in a stable state, the minecart is at either end of the track (depending on the state). An input pulse turns on powered rails at both ends of the track, causing the minecart to move to the other end.

Along the track, there are two separate detector elements (e.g. detector rails). These two detectors are each connected to an input of an RS NOR latch, and hence serve to translate minecart motion into a state transition. When the minecart moves, depending on its direction of motion, one detector turns on (and off) before the other; the second detector to be hit is what determines which input of the RS NOR latch stays on last and hence what the new state of the RS NOR latch is.

Design **A** uses detector rails, while design **B** uses pressure plates. (A minecart triggers a pressure plate on the inside of a turn, including diagonals.) Note that for B, the other side of the latch isn't a true $\overline{Q}$, as the passage of the cart turns on Q before actually switching the latch.

This type of T flip-flop is slower than traditional redstone-only circuits, but this may be desirable in certain situations. With T flip-flop designs that are level-triggered (as opposed to clocked or edge-triggered), a long input pulse causes the flip-flop to continuously switch state (oscillate) while the pulse is present. In pure redstone circuits, this is limited by the redstone circuit delays; therefore, a relatively short input pulse can cause several state transitions. Pure redstone T flip-flops usually include an edge-trigger or pulse-limiting circuit to the design, since the input pulse usually can't be guaranteed to be short enough without the use of that kind of circuit.

With rail-based designs, the speed at which the output can flip is limited by the time needed for the cart to move from one end of its rail to the other, which allows for a much longer pulse to be applied to a level-triggered input without needing an edge-trigger or pulse limiter circuit. However, the delay between the input pulse and the output transition is also longer.

Grizdale's T flip-flop

Grizdale's Compact TFF

This hopper/dropper design is not only compact, but tileable in three dimensions. The only hitch (for survival mode) is that the player needs access to nether quartz for the comparator.

The **A** variant has a size of 1×2×3. The **B** variant puts the input and output inline, but changes the footprint to 2×2×2, or 4×2×2 if players want fully powered input and output. The **B** design can also be tiled in line, side by side, vertically (by reversing alternate rows), or all three at once.

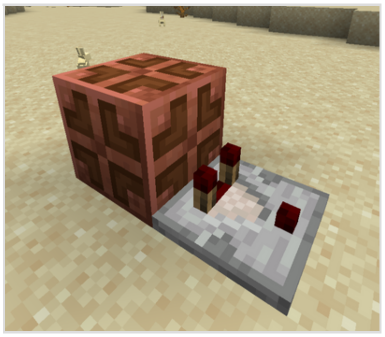
Once built, place a single item inside any of the containers and it works as a T flip-flop, with the item cycling between the two droppers. The core has a 1 tick delay between input and turning off or on, but the optional repeaters would raise this to 3.

This T Flip Flop can be turned into an SRT latch by powering the bottom dropper to set, and the top to reset. However, it isn't be as tileable as the original TFF.

Copper Bulb T flip-flop (Cop-Flop)

This 1x2 T flip-flop (dubbed the "Cop-Flop" or the "Copper Flopper" by YouTuber Mumbo Jumbo) is a

rising edge-triggered flip flop. It is made possible by using the Copper Bulb from snapshot Java Edition 23w43a. Since the Copper Bulb is a toggelable block, combining this with a comparator provides a revolutionary compact T flip-flop.


Copper Bulb T flip-flop (Cop-Flop)

Obsolete T flip-flops

T Flip-flop Z1	T Flip-flop Z2

Designs **Z1** and **Z2** do not work as of Java Edition 1.5.2 — in both cases, their pulse generator does not cause the piston to toggle its block as apparently intended.

References

1. JavaChef (March 24, 2024) "Minecraft: Analog Counter / Memory Latch " (https://youtube.com/watch?v=062T73b8uO4)
2. R.K.F. Walter (May 26, 2018) "Tutorial: Simple, Analogue Gated Memory & Long Delay Circuits" (https://youtube.com/watch?v=WGy4ObE-hzg)

Navigation

Redstone [hide]

Redstone circuits & tutorials [hide]

Redstone basics Redstone components · Redstone mechanics · Conductivity

Redstone circuits Clock circuits · Logic circuits · Memory circuits · Miscellaneous circuits · Pulse circuits · Transmission circuits

Featured tutorials Advanced redstone circuits · Block update detector (BUD) · Comparator update detector (CUD) · Flying machine · Mechanisms · Piston circuits · Quasi-connectivity · Redstone music

Redstone components [show]