

# Pulse Relay

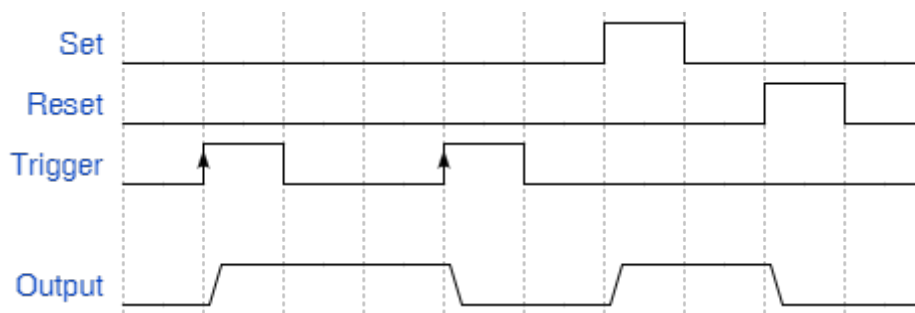


The **Pulse Relay** node sends **1** to the output when the **Set** input is set to **1** and sends **0** to the output when the **Reset** input is set to **1**.

When both **Set** and **Reset** are equal to **0**, the previous state of the output is maintained.

Whenever the **Trigger** input receives a rising edge, the **Set** and **Reset** inputs are ignored and the output state is toggled.

The **Behaviour** property determines whether **Set** or **Reset** must be dominant when both of them are equal to **1**.



Behavior of node when Set input is dominant



The Pulse Relay is used to transform a pulse input into a toggle switch.

The **Set** and **Reset** inputs are optional. If they are not connected to any other nodes, the **Trigger** input alone controls the output.

## Connectors

| Connector      | Type   | Description                                                                            |
|----------------|--------|----------------------------------------------------------------------------------------|
| <b>Set</b>     | Input  | Sets the output to <b>1</b> when activated, unless a Trigger event is being processed. |
| <b>Reset</b>   | Input  | Sets the output to <b>0</b> when activated, unless a Trigger event is being processed. |
| <b>Trigger</b> | Input  | Toggles the current output state whenever a rising edge is received.                   |
| <b>Output</b>  | Output | Current stored state of the Pulse Relay.                                               |

## Main properties

| Property         | Description                                                                                     |
|------------------|-------------------------------------------------------------------------------------------------|
| <b>Behaviour</b> | Defines whether <b>Set</b> or <b>Reset</b> is dominant when both inputs are equal to <b>1</b> . |

## Operation

The Pulse Relay combines the behaviour of a latching relay with a toggle command.

The node can therefore be controlled in two different ways:

- directly through **Set** and **Reset**;
- by toggling the current output state through **Trigger**.

When no Trigger event occurs, the Set/Reset logic is:

- if **Set** = **1** and **Reset** = **0**, the output becomes **1**;
- if **Set** = **0** and **Reset** = **1**, the output becomes **0**;
- if **Set** = **0** and **Reset** = **0**, the previous output state is maintained;
- if **Set** = **1** and **Reset** = **1**, the resulting state depends on the configured **Behaviour** property.

## Trigger

The **Trigger** input reacts to a rising edge:

**0 → 1**

Whenever a rising edge is detected, the current output state is toggled.

Conceptually:

**Output = NOT Output**

Therefore:

- if the current output is **0**, a Trigger event changes it to **1**;
- if the current output is **1**, a Trigger event changes it to **0**.



When a rising edge is received on **Trigger**, the **Set** and **Reset** inputs are ignored and the output is toggled.

## Trigger example

Assume the initial state is:

**Output = 0**

and **Set** and **Reset** are not connected.

Each rising edge on **Trigger** produces the following sequence:

| Trigger event      | Output |
|--------------------|--------|
| First rising edge  | 1      |
| Second rising edge | 0      |
| Third rising edge  | 1      |
| Fourth rising edge | 0      |

The node therefore converts repeated pulse events into a bistable toggle state.

## Set and Reset

The **Set** and **Reset** inputs can be used to force a defined output state.

When:

**Set = 1**

and:

**Reset = 0**

the output becomes:

**1**

When:

**Set = 0**

and:

**Reset = 1**

the output becomes:

**0**

When both inputs return to 0, the previous output state is maintained.

## Behaviour

The **Behaviour** property defines what happens when both **Set** and **Reset** are equal to 1 at the same time.

The two possible behaviours are conceptually:

- **Set dominant**
- **Reset dominant**

## Set dominant

When **Set** is dominant:

| Set | Reset | Output                   |
|-----|-------|--------------------------|
| 0   | 0     | Maintains previous state |
| 1   | 0     | 1                        |
| 0   | 1     | 0                        |
| 1   | 1     | 1                        |

When both inputs are active, activation has priority.

## Reset dominant

When **Reset** is dominant:

| Set | Reset | Output                   |
|-----|-------|--------------------------|
| 0   | 0     | Maintains previous state |
| 1   | 0     | 1                        |
| 0   | 1     | 0                        |
| 1   | 1     | 0                        |

When both inputs are active, deactivation has priority.

## Trigger priority

The Trigger command has a special role because it toggles the output independently from the Set/Reset logic.

For example, assume:

**Output = 0**

and:

**Set = 1**

If a rising edge is received on **Trigger**, the current state is toggled according to the Trigger command and the Set/Reset inputs are ignored for that event.

This makes Trigger suitable for push-button style control.

## Example - Push button toggle

A push button sends a short pulse every time it is pressed.

The pulse is connected to **Trigger**.

Starting from:

**Output = 0**

the sequence becomes:

- first press → **Output = 1**
- second press → **Output = 0**
- third press → **Output = 1**

This allows a momentary push button to control a persistent ON/OFF state.

## Example - Toggle with forced commands

The Pulse Relay can also combine toggle control with explicit Set and Reset commands.

For example:

- a local push button is connected to **Trigger**;
- a central ON command is connected to **Set**;
- a central OFF command is connected to **Reset**.

The push button can toggle the current state locally, while Set and Reset can force a specific state when required.

## Typical applications

The Pulse Relay is useful whenever a pulse command must control a persistent digital state.

Typical applications include:

- converting a push-button pulse into an ON/OFF toggle;
- controlling lights from momentary buttons;
- maintaining a digital state after a pulse;
- combining local toggle commands with centralized Set and Reset commands;
- implementing bistable control logic;
- controlling devices with pulse-based commands.

## Verification through simulation

The Pulse Relay can be tested using the Logic Module simulation tools.

A typical test procedure is the following:

- start with **Set = 0**, **Reset = 0** and **Trigger = 0**;
- generate a rising edge on **Trigger** and verify that the output toggles;
- return **Trigger** to 0;
- generate another rising edge and verify that the output toggles again;
- set **Set = 1** and verify that **Output = 1**;
- return **Set** to 0 and verify that the output state is maintained;
- set **Reset = 1** and verify that **Output = 0**;
- test **Set = 1** and **Reset = 1** and verify the configured **Behaviour**;
- generate a Trigger rising edge while Set or Reset is active and verify that the Trigger event takes priority.

## Configuration notes

- **Trigger** reacts to rising edges, not to the duration of the signal.
- The Trigger input must return to 0 before another rising edge can be detected.
- Each Trigger rising edge toggles the current output state.
- **Set** and **Reset** are optional.
- If Set and Reset are not connected, Trigger alone can control the output.
- The **Behaviour** property is relevant when Set and Reset are both equal to 1.
- Use Set or Reset when a defined output state must be forced.
- Use Trigger when a pulse must toggle the current state.

## Related pages

- [Logic Module](#)
- [Logic Module - Simulation management](#)
- [Latching Relay](#)

From:

<http://www.thinknx.com/wiki/> - **Learning Thinknx**

Permanent link:

[http://www.thinknx.com/wiki/doku.php?id=lm\\_logic\\_prelay](http://www.thinknx.com/wiki/doku.php?id=lm_logic_prelay)

Last update: **2026/09/21 16:12**

