Search...

redstone

Get and set redstone signals adjacent to this computer.

The **`redstone`** library exposes three "types" of redstone control:

- Binary input/output (**`setOutput`**/**`getInput`**): These simply check if a redstone wire has any input or output. A signal strength of 1 and 15 are treated the same.
- Analogue input/output (**`setAnalogOutput`**/**`getAnalogInput`**): These work with the actual signal strength of the redstone wired, from 0 to 15.
- Bundled cables (**`setBundledOutput`**/**`getBundledInput`**): These interact with "bundled" cables, such as those from Project:Red. These allow you to send 16 separate on/off signals. Each channel corresponds to a colour, with the first being **`colors.white`** and the last **`colors.black`**.

Whenever a redstone input changes, a **`redstone`** event will be fired. This may be used instead of repeativly polling.

This module may also be referred to as `rs`. For example, one may call `rs.getSides()` instead of **`getSides`**.

Usage

- Toggle the redstone signal above the computer every 0.5 seconds.

```
while true do
  redstone.setOutput("top", not redstone.getOutput("top"))
  sleep(0.5)
end
```

Run ▷

- Mimic a redstone comparator in **subtraction mode**.

```
while true do
  local rear = rs.getAnalogueInput("back")
  local sides = math.max(rs.getAnalogueInput("left"), rs.getAnalogueInput("right")
  rs.setAnalogueOutput("front", math.max(rear - sides, 0))
```

Run ▷

```
    os.pullEvent("redstone") -- Wait for a change to inputs.
  end
```

`getSides()`	Returns a table containing the six sides of the computer.
`setOutput(side, on)`	Turn the redstone signal of a specific side on or off.
`getOutput(side)`	Get the current redstone output of a specific side.
`getInput(side)`	Get the current redstone input of a specific side.
`setAnalogOutput(side, value)`	Set the redstone signal strength for a specific side.
`setAnalogueOutput(side, value)`	Set the redstone signal strength for a specific side.
`getAnalogOutput(side)`	Get the redstone output signal strength for a specific side.
`getAnalogueOutput(side)`	Get the redstone output signal strength for a specific side.
`getAnalogInput(side)`	Get the redstone input signal strength for a specific side.
`getAnalogueInput(side)`	Get the redstone input signal strength for a specific side.
`setBundledOutput(side, output)`	Set the bundled cable output for a specific side.
`getBundledOutput(side)`	Get the bundled cable output for a specific side.
`getBundledInput(side)`	Get the bundled cable input for a specific side.
`testBundledInput(side, mask)`	Determine if a specific combination of colours are on for the given side.

§ `getSides()` [Source]

Returns a table containing the six sides of the computer. Namely, "top", "bottom", "left", "right", "front" and "back".

Returns

1. { `string...` } A table of valid sides.

Changes

- **New in version 1.2**

§ `setOutput(side, on)` [Source]

Turn the redstone signal of a specific side on or off.

Parameters

1. `side` : `string` The side to set.
2. `on` : `boolean` Whether the redstone signal should be on or off. When on, a signal strength of 15 is emitted.

---

§ `getOutput(side)` [Source]

Get the current redstone output of a specific side.

Parameters

1. `side` : `string` The side to get.

Returns

1. `boolean` Whether the redstone output is on or off.

See also

- `setOutput`

---

§ `getInput(side)` [Source]

Get the current redstone input of a specific side.

Parameters

1. `side` : `string` The side to get.

Returns

1. `boolean` Whether the redstone input is on or off.

---

§ `setAnalogOutput(side, value)` [Source]

Set the redstone signal strength for a specific side.

Parameters

1. `side` : `string` The side to set.
2. `value` : `number` The signal strength between 0 and 15.

Throws

- If `value` is not between 0 and 15.

Changes

- **New in version 1.51**

---

§ `setAnalogueOutput(side, value)` [Source]

Set the redstone signal strength for a specific side.

Parameters

1. `side` : `string` The side to set.
2. `value` : `number` The signal strength between 0 and 15.

Throws

- If `value` is not between 0 and 15.

Changes

- **New in version 1.51**

---

§ `getAnalogOutput(side)` [Source]

Get the redstone output signal strength for a specific side.

Parameters

1. `side` : `string` The side to get.

Returns

1. `number` The output signal strength, between 0 and 15.

See also

- `setAnalogOutput`

Changes

- **New in version 1.51**

---

§ `getAnalogueOutput(side)` [Source]

Get the redstone output signal strength for a specific side.

Parameters

1. `side` : `string` The side to get.

Returns

1. `number` The output signal strength, between 0 and 15.

See also

- `setAnalogOutput`

Changes

- **New in version 1.51**

---

§ `getAnalogInput(side)` [Source]

Get the redstone input signal strength for a specific side.

Parameters

1. `side : string` The side to get.

Returns

1. `number` The input signal strength, between 0 and 15.

Changes

- **New in version 1.51**

---

§ `getAnalogueInput(side)` [Source]

Get the redstone input signal strength for a specific side.

Parameters

1. `side : string` The side to get.

Returns

1. `number` The input signal strength, between 0 and 15.

Changes

- **New in version 1.51**

---

§ `setBundledOutput(side, output)` [Source]

Set the bundled cable output for a specific side.

Parameters

1. `side : string` The side to set.
2. `output : number` The colour bitmask to set.

See also

- **`colors.subtract`** For removing a colour from the bitmask.
- **`colors.combine`** For adding a color to the bitmask.

---

§ `getBundledOutput(side)` [Source]

Get the bundled cable output for a specific side.

Parameters

1. `side` : **`string`** The side to get.

Returns

1. `number` The bundle cable's output.

---

§ `getBundledInput(side)` [Source]

Get the bundled cable input for a specific side.

Parameters

1. `side` : **`string`** The side to get.

Returns

1. `number` The bundle cable's input.

See also

- **`testBundledInput`** To determine if a specific colour is set.

---

§ `testBundledInput(side, mask)` [Source]

Determine if a specific combination of colours are on for the given side.

Parameters

1. `side` : **`string`** The side to test.
2. `mask` : `number` The mask to test.

Returns

1. `boolean` If the colours are on.

Usage

- Check if **`colors.white`** and **`colors.black`** are on above this block.

```
print(redstone.testBundledInput("top", colors.combine(colors.white, colo
```

Run ▷ k

See also

- `getBundledInput`

Last updated on 2025-07-06