

WiFi Module

The NodeMCU WiFi control is spread across several tables:

- `wifi` for overall WiFi configuration
- `wifi.sta` for station mode functions
- `wifi.ap` for wireless access point (WAP or simply AP) functions
- `wifi.ap.dhcp` for DHCP server control

<code>wifi.getchannel()</code>	Gets the current WiFi channel.
<code>wifi.getmode()</code>	Gets WiFi operation mode.
<code>wifi.getphymode()</code>	Gets WiFi physical mode.
<code>wifi.setmode()</code>	Configures the WiFi mode to use.
<code>wifi.setphymode()</code>	Sets WiFi physical mode.
<code>wifi.sleeptype()</code>	Configures the WiFi modem sleep type.
<code>wifi.startsmart()</code>	Starts to auto configuration, if success set up SSID and password automatically.
<code>wifi.stopsmart()</code>	Stops the smart configuring process.
<code>wifi.sta.autoconnect()</code>	Auto connects to AP in station mode.
<code>wifi.sta.config()</code>	Sets the WiFi station configuration.
<code>wifi.sta.connect()</code>	Connects to AP in station mode.
<code>wifi.sta.disconnect()</code>	Disconnects from AP in station mode.
<code>wifi.sta.eventMonReg()</code>	Registers callbacks for WiFi station status events.
<code>wifi.sta.eventMonStart()</code>	Starts WiFi station event monitor.
<code>wifi.sta.eventMonStop()</code>	Stops WiFi station event monitor.
<code>wifi.sta.getap()</code>	Scans AP list as a Lua table into callback function.
<code>wifi.sta.getbroadcast()</code>	Gets the broadcast address in station mode.
<code>wifi.sta.getconfig()</code>	Gets the WiFi station configuration.
<code>wifi.sta.gethostname()</code>	Gets current station hostname.
<code>wifi.sta.getip()</code>	Gets IP address, netmask, and gateway address in station mode.

<code>wifi.sta.getmac()</code>	Gets MAC address in station mode.
<code>wifi.sta.sethostname()</code>	Sets station hostname.
<code>wifi.sta.setip()</code>	Sets IP address, netmask, gateway address in station mode.
<code>wifi.sta.setmac()</code>	Sets MAC address in station mode.
<code>wifi.sta.status()</code>	Gets the current status in station mode.
<code>wifi.ap.config()</code>	Sets SSID and password in AP mode.
<code>wifi.ap.getbroadcast()</code>	Gets broadcast address in AP mode.
<code>wifi.ap.getclient()</code>	Gets table of clients connected to device in AP mode.
<code>wifi.ap.getip()</code>	Gets IP address, netmask and gateway in AP mode.
<code>wifi.ap.getmac()</code>	Gets MAC address in AP mode.
<code>wifi.ap.setip()</code>	Sets IP address, netmask and gateway address in AP mode.
<code>wifi.ap.setmac()</code>	Sets MAC address in AP mode.
<code>wifi.ap.dhcp.config()</code>	Configure the dhcp service.
<code>wifi.ap.dhcp.start()</code>	Starts the DHCP service.
<code>wifi.ap.dhcp.stop()</code>	Stops the DHCP service.

wifi.getchannel()

Gets the current WiFi channel.

Syntax

```
wifi.getchannel()
```

Parameters

```
nil
```

Returns

current WiFi channel

Example

```
print(wifi.getchannel())
```

wifi.getmode()

Gets WiFi operation mode.

Syntax

```
wifi.getmode()
```

Parameters

```
nil
```

Returns

The WiFi mode, as one of the `wifi.STATION`, `wifi.SOFTAP`, `wifi.STATIONAP` or `wifi.NULLMODE` constants.

See also

```
wifi.setmode()
```

wifi.getphymode()

Gets WiFi physical mode.

Syntax

```
wifi.getpymode()
```

Parameters

none

Returns

The current physical mode as one of `wifi.PHYMODE_B`, `wifi.PHYMODE_G` or `wifi.PHYMODE_N`.

See also

```
wifi.setphymode()
```

wifi.setmode()

Configures the WiFi mode to use. NodeMCU can run in one of four WiFi modes:

- Station mode, where the NodeMCU device joins an existing network
- Access point (AP) mode, where it creates its own network that others can join
- Station + AP mode, where it both creates its own network while at the same time being joined to another existing network
- WiFi off

When using the combined Station + AP mode, the same channel will be used for both networks as the radio can only listen on a single channel.

Syntax

```
wifi.setmode(mode)
```

Parameters

`mode` value should be one of

- `wifi.STATION` for when the device is connected to a WiFi router. This is often done to give the device access to the Internet.
- `wifi.SOFTAP` for when the device is acting *only* as an access point. This will allow you to see the device in the list of WiFi networks (unless you hide the SSID, of course). In this mode your computer can connect to the device, creating a local area network. Unless you change the value, the NodeMCU device will be given a local IP address of 192.168.4.1 and assign your computer the next available IP address, such as 192.168.4.2.
- `wifi.STATIONAP` is the combination of `wifi.STATION` and `wifi.SOFTAP`. It allows you to create a local WiFi connection *and* connect to another WiFi router.
- `wifi.NULLMODE` to switch off WiFi

Returns

current mode after setup

Example

```
wifi.setmode(wifi.STATION)
```

See also

```
wifi.getmode()
```

wifi.setphymode()

Sets WiFi physical mode.

- `wifi.PHYMODE_B` 802.11b, more range, low Transfer rate, more current draw
- `wifi.PHYMODE_G` 802.11g, medium range, medium transfer rate, medium current draw
- `wifi.PHYMODE_N` 802.11n, least range, fast transfer rate, least current draw (STATION ONLY) Information from the Espressif datasheet v4.3

Parameters	Typical Power Usage
Tx 802.11b, CCK 11Mbps, P OUT=+17dBm	170 mA
Tx 802.11g, OFDM 54Mbps, P OUT =+15dBm	140 mA
Tx 802.11n, MCS7 65Mbps, P OUT =+13dBm	120 mA
Rx 802.11b, 1024 bytes packet length, -80dBm	50 mA

Rx 802.11g, 1024 bytes packet length, -70dBm	56 mA
Rx 802.11n, 1024 bytes packet length, -65dBm	56 mA

Syntax

```
wifi.setphymode(mode)
```

Parameters

`mode` one of the following

- `wifi.PHYMODE_B`
- `wifi.PHYMODE_G`
- `wifi.PHYMODE_N`

Returns

physical mode after setup

See also

```
wifi.getphymode()
```

wifi.sleepype()

Configures the WiFi modem sleep type.

Syntax

```
wifi.sleepype(type_wanted)
```

Parameters

`type_wanted` one of the following:

- `wifi.NONE_SLEEP` to keep the modem on at all times
- `wifi.LIGHT_SLEEP` to allow the modem to power down under some circumstances
- `wifi.MODEM_SLEEP` to power down the modem as much as possible

Returns

The actual sleep mode set, as one of `wifi.NONE_SLEEP`, `wifi.LIGHT_SLEEP` or `wifi.MODEM_SLEEP`.

See also

- `node.dsleep()`
- `rtctime.dsleep()`

wifi.startsmart()

Starts to auto configuration, if success set up SSID and password automatically.

Intended for use with SmartConfig apps, such as Espressif's [Android & iOS app](#).

Only usable in `wifi.STATION` mode.

Syntax

```
wifi.startsmart(type, callback)
```

Parameters


- `type` 0 for ESP_TOUCH, or 1 for AIR_KISS.
- `callback` a callback function of the form `function(ssid, password) end` which gets called after configuration.

Returns

```
nil
```

Example

```
wifi.setmode(wifi.STATION)
wifi.startsmart(0,
  function(ssid, password)
    print(string.format("Success. SSID:%s ; PASSWORD:%s", ssid, password))
  end
)
```



See also

```
wifi.stopsmart()
```

wifi.stopsmart()

Stops the smart configuring process.

Syntax

```
wifi.stopsmart()
```

Parameters

none

Returns

```
nil
```

See also

```
wifi.startsmart()
```

wifi.sta Module

wifi.sta.autoconnect()

Auto connects to AP in station mode.

Syntax

```
wifi.sta.autoconnect(auto)
```

Parameters

auto 0 to disable auto connecting, 1 to enable auto connecting

Returns

```
nil
```

Example

```
wifi.sta.autoconnect(1)
```

See also

- `wifi.sta.config()`
- `wifi.sta.connect()`
- `wifi.sta.disconnect()`

wifi.sta.config()

Sets the WiFi station configuration.

Syntax

```
wifi.sta.config(ssid, password[, auto[, bssid]])
```

Parameters

- **ssid** string which is less than 32 bytes.
- **password** string which is 8-64 or 0 bytes. Empty string indicates an open WiFi access point.
- **auto** value of 0 or 1 (default)
 - 0, Disable auto connect and remain disconnected from access point
 - 1, Enable auto connect and connect to access point

- `bssid` string that contains the MAC address of the access point (optional)
 - You can set BSSID if you have multiple access points with the same SSID.
 - Note: if you set BSSID for a specific SSID and would like to configure station to connect to the same SSID only without the BSSID requirement, you MUST first configure to station to a different SSID first, then connect to the desired SSID
 - The following formats are valid:
 - "DE-C1-A5-51-F1-ED"
 - "AC-1D-1C-B1-0B-22"
 - "DE AD BE EF 7A C0"

Returns

`nil`

Example

```
--Connect to access point automatically when in range
wifi.sta.config("myssid", "password")

--Connect to Unsecured access point automatically when in range
wifi.sta.config("myssid", "")

--Connect to access point, User decides when to connect/disconnect to/from
wifi.sta.config("myssid", "mypassword", 0)
wifi.sta.connect()
-- ... do some WiFi stuff
wifi.sta.disconnect()

--Connect to specific access point automatically when in range
wifi.sta.config("myssid", "mypassword", "12:34:56:78:90:12")

--Connect to specific access point, User decides when to connect/disconnect
wifi.sta.config("myssid", "mypassword", 0, "12:34:56:78:90:12")
wifi.sta.connect()
-- ... do some WiFi stuff
wifi.sta.disconnect()
```

See also

- `wifi.sta.connect()`
- `wifi.sta.disconnect()`

wifi.sta.connect()

Connects to AP in station mode.

Syntax

```
wifi.sta.connect()
```

Parameters

none

Returns

`nil`

Example

```
wifi.sta.connect()
```

See also

- `wifi.sta.disconnect()`
- `wifi.sta.config()`

wifi.sta.disconnect()

Disconnects from AP in station mode.

Syntax

```
wifi.sta.disconnect()
```

Parameters

none

Returns

`nil`

See also

- `wifi.sta.config()`
- `wifi.sta.connect()`

wifi.sta.eventMonReg()

Registers callbacks for WiFi station status events.

Syntax

- `wifi.sta.eventMonReg(wifi_status, function([previous_state]))`
- `wifi.sta.eventMonReg(wifi.status, "unreg")`

Parameters

- `wifi_status` WiFi status you would like to set callback for, one of:
 - `wifi.STA_IDLE`
 - `wifi.STA_CONNECTING`
 - `wifi.STA_WRONGPWD`
 - `wifi.STA_APNOTFOUND`

- `wifi.STA_FAIL`
- `wifi.STA_GOTIP`
- `function` function to perform when event occurs
- `"unreg"` unregister previously registered callback
- `previous_state` previous wifi_state(0 - 5)

Returns

`nil`

Example

```
--register callback
wifi.sta.eventMonReg(wifi.STA_IDLE, function() print("STATION_IDLE") end)
wifi.sta.eventMonReg(wifi.STA_CONNECTING, function() print("STATION_CONNECTING") end)
wifi.sta.eventMonReg(wifi.STA_WRONGPWD, function() print("STATION_WRONG_PASSWORD") end)
wifi.sta.eventMonReg(wifi.STA_APNOTFOUND, function() print("STATION_NO_AP_FOUND") end)
wifi.sta.eventMonReg(wifi.STA_FAIL, function() print("STATION_CONNECT_FAIL") end)
wifi.sta.eventMonReg(wifi.STA_GOTIP, function() print("STATION_GOT_IP") end)

--register callback: use previous state
wifi.sta.eventMonReg(wifi.STA_CONNECTING, function(previous_State)
    if(previous_State==wifi.STA_GOTIP) then
        print("Station lost connection with access point\n\tAttempting to reconnect")
    else
        print("STATION_CONNECTING")
    end
end)

--unregister callback
wifi.sta.eventMonReg(wifi.STA_IDLE, "unreg")
```

See also

- `wifi.sta.eventMonStart()`
- `wifi.sta.eventMonStop()`

wifi.sta.eventMonStart()

Starts WiFi station event monitor.

Syntax

```
wifi.sta.eventMonStart([ms])
```

Parameters

`ms` interval between checks in milliseconds, defaults to 150ms if not provided

Returns

`nil`

Example

```
--start WiFi event monitor with default interval
wifi.sta.eventMonStart()

--start WiFi event monitor with 100ms interval
wifi.sta.eventMonStart(100)
```

See also

- `wifi.sta.eventMonReg()`
- `wifi.sta.eventMonStop()`
-

wifi.sta.eventMonStop()

Stops WiFi station event monitor.

Syntax

```
wifi.sta.eventMonStop(["unreg all"])
```

Parameters

`"unreg all"` unregister all previously registered functions

Returns

`nil`

Example

```
--stop WiFi event monitor
wifi.sta.eventMonStop()

--stop WiFi event monitor and unregister all callbacks
wifi.sta.eventMonStop("unreg all")
```

See also

- `wifi.sta.eventMonReg()`
- `wifi.sta.eventMonStart()`

wifi.sta.getap()

Scans AP list as a Lua table into callback function.

Syntax

```
wifi.sta.getap([[cfg], format,] callback(table))
```

Parameters

- `cfg` table that contains scan configuration
 - `ssid` SSID == nil, don't filter SSID
 - `bssid` BSSID == nil, don't filter BSSID
 - `channel` channel == 0, scan all channels, otherwise scan set channel (default is 0)
 - `show_hidden` show_hidden == 1, get info for router with hidden SSID (default is 0)
- `format` select output table format, defaults to 0
 - 0: old format (SSID : Authmode, RSSI, BSSID, Channel), any duplicate SSIDs will be discarded
 - 1: new format (BSSID : SSID, RSSI, auth mode, Channel)
- `callback(table)` a callback function to receive the AP table when the scan is done. This function receives a table, the key is the BSSID, the value is other info in format: SSID, RSSID, auth mode, channel.

Returns

`nil`

Example

[illegible]

```
scan_cfg.channel = wifi.getchannel()  
scan_cfg.show_hidden = 0  
ssid, tmp, bssid_set, bssid=nil, nil, nil, nil  
wifi.sta.getap(scan_cfg, 1, listap)
```

See also

`wifi.sta.getip()`

wifi.sta.getbroadcast()

Gets the broadcast address in station mode.

Syntax

`wifi.sta.getbroadcast()`

Parameters

`nil`

Returns

broadcast address as string, for example "192.168.0.255", returns `nil` if IP address = "0.0.0.0".

See also

`wifi.sta.getip()`

wifi.sta.getconfig()

Gets the WiFi station configuration.

Syntax

`wifi.sta.getconfig()`

Parameters

none

Returns

ssid, password, bssid_set, bssid

Note: If bssid_set is equal to 0 then bssid is irrelevant

Example

```
--Get current Station configuration
ssid, password, bssid_set, bssid=wifi.sta.getconfig()
print("\nCurrent Station configuration:\nSSID : "..ssid
.." \nPassword : "..password
.." \nBSSID_set : "..bssid_set
.." \nBSSID: "..bssid.." \n")
ssid, password, bssid_set, bssid=nil, nil, nil, nil
```

See also

- `wifi.sta.connect()`
- `wifi.sta.disconnect()`

wifi.sta.gethostname()

Gets current station hostname.

Syntax

```
wifi.sta.gethostname()
```

Parameters

none

Returns

currently configured hostname

Example

```
print("Current hostname is: \""..wifi.sta.gethostname().."\"")
```

wifi.sta.getip()

Gets IP address, netmask, and gateway address in station mode.

Syntax

```
wifi.sta.getip()
```

Parameters

none

Returns

IP address, netmask, gateway address as string, for example "192.168.0.111". Returns `nil` if IP = "0.0.0.0".

Example

```
-- print current IP address, netmask, gateway
print(wifi.sta.getip())
-- 192.168.0.111 255.255.255.0 192.168.0.1
ip = wifi.sta.getip()
print(ip)
-- 192.168.0.111
ip, nm = wifi.sta.getip()
print(nm)
-- 255.255.255.0
```

See also

```
wifi.sta.getmac()
```

wifi.sta.getmac()

Gets MAC address in station mode.

Syntax

```
wifi.sta.getmac()
```

Parameters

none

Returns

MAC address as string e.g. "18-33-44-FE-55-BB"

See also

```
wifi.sta.getip()
```

wifi.sta.sethostname()

Sets station hostname.

Syntax

```
wifi.sta.sethostname(hostname)
```

Parameters

`hostname` must only contain letters, numbers and hyphens('-') and be 32 characters or less with first and last character being alphanumeric

Returns

true if hostname was successfully set, false otherwise

Example

```
if (wifi.sta.sethostname("NodeMCU") == true) then
    print("hostname was successfully changed")
else
    print("hostname was not changed")
end
```

wifi.sta.setip()

Sets IP address, netmask, gateway address in station mode.

Syntax

```
wifi.sta.setip(cfg)
```

Parameters

`cfg` table contain IP address, netmask, and gateway

```
{
  ip = "192.168.0.111",
  netmask = "255.255.255.0",
  gateway = "192.168.0.1"
}
```

Returns

true if success, false otherwise

See also

```
wifi.sta.setmac()
```

wifi.sta.setmac()

Sets MAC address in station mode.

Syntax

```
wifi.sta.setmac(mac)
```

Parameters

MAC address in string e.g. "DE:AD:BE:EF:7A:C0"

Returns

true if success, false otherwise

Example

```
print(wifi.sta.setmac("DE:AD:BE:EF:7A:C0"))
```

See also

```
wifi.sta.setip()
```

wifi.sta.status()

Gets the current status in station mode.

Syntax

```
wifi.sta.status()
```

Parameters

```
nil
```

Returns

number : 0~5

- 0: STATION_IDLE,
- 1: STATION_CONNECTING,
- 2: STATION_WRONG_PASSWORD,
- 3: STATION_NO_AP_FOUND,
- 4: STATION_CONNECT_FAIL,
- 5: STATION_GOT_IP.

wifi.ap Module

wifi.ap.config()

Sets SSID and password in AP mode. Be sure to make the password at least 8 characters long! If you don't it will default to *no* password and not set the SSID! It will still work as an access point but use a default SSID like e.g. ESP_9997C3.

Syntax

```
wifi.ap.config(cfg)
```

Parameters

- `ssdi` SSID chars 1-32
- `pwd` password chars 8-64
- `auth` authentication one of AUTH_OPEN, AUTH_WPA_PSK, AUTH_WPA2_PSK, AUTH_WPA_WPA2_PSK, default = AUTH_OPEN
- `channel` channel number 1-14 default = 6

- `hidden` 0 = not hidden, 1 = hidden, default 0
- `max` maximal number of connections 1-4 default=4
- `beacon` beacon interval time in range 100-60000, default = 100

Returns

`nil`

Example:

```
cfg={}  
cfg.ssid="myssid"  
cfg.pwd="mypassword"  
wifi.ap.config(cfg)
```

wifi.ap.getbroadcast()

Gets broadcast address in AP mode.

Syntax

```
wifi.ap.getbroadcast()
```

Parameters

none

Returns

broadcast address in string, for example "192.168.0.255", returns `nil` if IP address = "0.0.0.0".

Example

```
bc = wifi.ap.getbroadcast()  
print(bc)  
-- 192.168.0.255
```

See also

```
wifi.ap.getip()
```

wifi.ap.getclient()

Gets table of clients connected to device in AP mode.

Syntax

```
wifi.ap.getclient()
```

Parameters

none

Returns

table of connected clients

Example

```
table={}
table=wifi.ap.getclient()
for mac,ip in pairs(table) do
    print(mac,ip)
end

-- or shorter
for mac,ip in pairs(wifi.ap.getclient()) do
    print(mac,ip)
end
```

wifi.ap.getip()

Gets IP address, netmask and gateway in AP mode.

Syntax

```
wifi.ap.getip()
```

Parameters

none

Returns

IP address, netmask, gateway address as string, for example "192.168.0.111", returns `nil` if IP address = "0.0.0.0".

Example

```
-- print current ip, netmask, gateway
print(wifi.ap.getip())
-- 192.168.4.1 255.255.255.0 192.168.4.1
ip = wifi.ap.getip()
print(ip)
-- 192.168.4.1
ip, nm = wifi.ap.getip()
print(nm)
-- 255.255.255.0
ip, nm, gw = wifi.ap.getip()
print(gw)
-- 192.168.4.1
```

See also

- `wifi.ap.getmac()`

wifi.ap.getmac()

Gets MAC address in AP mode.

Syntax

```
wifi.ap.getmac()
```

Parameters

none

Returns

MAC address as string, for example "1A-33-44-FE-55-BB"

See also

```
wifi.ap.getip()
```

wifi.ap.setip()

Sets IP address, netmask and gateway address in AP mode.

Syntax

```
wifi.ap.setip(cfg)
```

Parameters

`cfg` table contain IP address, netmask, and gateway

Returns

true if successful, false otherwise

Example

```
cfg =  
{  
    ip="192.168.1.1",  
    netmask="255.255.255.0",  
    gateway="192.168.1.1"  
}  
wifi.ap.setip(cfg)
```

See also

```
wifi.ap.setmac()
```

wifi.ap.setmac()

Sets MAC address in AP mode.

Syntax

```
wifi.ap.setmac(mac)
```

Parameters

MAC address in byte string, for example "AC-1D-1C-B1-0B-22"

Returns

true if success, false otherwise

Example

```
print(wifi.ap.setmac("AC-1D-1C-B1-0B-22"))
```

See also

```
wifi.ap.setip()
```

wifi.ap.dhcp Module

wifi.ap.dhcp.config()

Configure the dhcp service. Currently only supports setting the start address of the dhcp address pool.

Syntax

```
wifi.ap.dhcp.config(dhcp_config)
```

Parameters

`dhcp_config` table containing the start-IP of the DHCP address pool, eg. "192.168.1.100"

Returns

```
pool_startip, pool_endip
```

Example

```
dhcp_config = {}  
dhcp_config.start = "192.168.1.100"  
wifi.ap.dhcp.config(dhcp_config)
```

wifi.ap.dhcp.start()

Starts the DHCP service.

Syntax

```
wifi.ap.dhcp.start()
```

Parameters

none

Returns

boolean indicating success

wifi.ap.dhcp.stop()

Stops the DHCP service.

Syntax

```
wifi.ap.dhcp.stop()
```

Parameters

none

Returns

boolean indicating success