**NodeMCU API Instruction**

**version 0.9.5 build 2015-02-13**

**INDEX**

* [Change Log](https://github.com/nodemcu/nodemcu-firmware/wiki)
* [New GPIO Map](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#new_gpio_map)
* [Old GPIO Map](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#old_gpio_map)

**node module**

* [node.restart()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#noderestart)
* [node.dsleep()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#nodedsleep)
* [node.info()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#nodeinfo)
* [node.chipid()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#nodechipid)
* [node.flashid()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#nodeflashid)
* [node.heap()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#nodeheap)
* [node.key()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#nodekey) --deprecated
* [node.led()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#nodeled) --deprecated
* [node.input()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#nodeinput)
* [node.output()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#nodeoutput)
* [node.readvdd33()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#nodereadvdd33) --deprecated, moved to adc.readvdd33()
* [node.compile()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#nodecompile)
* [node.setcpufreq()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#nodesetcpufreq)
* [node.restore()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#noderestore) --Added on 07/04/2015

**file module**

* [file.remove()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#fileremove)
* [file.open()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#fileopen)
* [file.close()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#fileclose)
* [file.readline()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#filereadline)
* [file.writeline()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#filewriteline)
* [file.read()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#fileread)
* [file.write()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#filewrite)
* [file.flush()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#fileflush)
* [file.seek()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#fileseek)
* [file.list()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#filelist)
* [file.format()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#fileformat)
* [file.rename()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#filerename)
* [file.fsinfo()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#filefsinfo)

**wifi module**

* [wifi.setmode()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifisetmode)
* [wifi.getmode()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifigetmode)
* [wifi.getchannel()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifigetchannel) --Added on 06/16/2015
* [wifi.setphymode()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifisetphymode) --Added on 06/16/2015
* [wifi.getphymode()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifigetphymode) --Added on 06/16/2015
* [wifi.startsmart()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifistartsmart)
* [wifi.stopsmart()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifistopsmart)
* [wifi.sleeptype()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifisleeptype)

**wifi.sta sub-module**

* [wifi.sta.getconfig()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifistagetconfig) --Added on 06/16/2015
* [wifi.sta.config()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifistaconfig) --Updated on 06/16/2015
* [wifi.sta.connect()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifistaconnect)
* [wifi.sta.disconnect()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifistadisconnect)
* [wifi.sta.autoconnect()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifistaautoconnect)
* [wifi.sta.getip()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifistagetip)
* [wifi.sta.setip()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifistasetip)
* [wifi.sta.getmac()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifistagetmac) --Updated on 06/16/2015
* [wifi.sta.setmac()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifistasetmac) --Updated on 06/16/2015
* [wifi.sta.getap()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifistagetap) --Updated on 06/16/2015
* [wifi.sta.status()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifistastatus)
* [wifi.sta.getbroadcast()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifistagetbroadcast)
* [wifi.sta.eventMonReg()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifistaeventmonreg) --Added on 07/14/2015
* [wifi.sta.eventMonStart()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifistaeventmonstart) --Added on 07/14/2015
* [wifi.sta.eventMonStop()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifistaeventmonstop) --Added on 07/14/2015

**wifi.ap sub-module**

* [wifi.ap.config()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifiapconfig)
* [wifi.ap.getip()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifiapgetip)
* [wifi.ap.setip()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifiapsetip)
* [wifi.ap.getmac()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifiapgetmac)
* [wifi.ap.setmac()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifiapsetmac) --Updated on 06/16/2015
* [wifi.ap.getclient()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifiapgetclient)
* [wifi.ap.getconfig()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifiapgetconfig)
* [wifi.ap.getbroadcast()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifiapgetbroadcast)

**wifi.ap.dhcp sub-module**

* [wifi.ap.dhcp.config()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifiapdhcpconfig)
* [wifi.ap.dhcp.start()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifiapdhcpstart)
* [wifi.ap.dhcp.stop()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifiapdhcpstop)

**timer module**

* [tmr.delay()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#tmrdelay)
* [tmr.now()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#tmrnow)
* [tmr.alarm()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#tmralarm)
* [tmr.stop()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#tmrstop)
* [tmr.wdclr()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#tmrwdclr)
* [tmr.time()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#tmrtime)

**gpio module**

* [gpio.mode()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#gpiomode)
* [gpio.read()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#gpioread)
* [gpio.write()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#gpiowrite)
* [gpio.trig()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#gpiotrig)

**pwm module**

* [pwm.setup()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#pwmsetup)
* [pwm.close()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#pwmclose)
* [pwm.start()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#pwmstart)
* [pwm.stop()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#pwmstop)
* [pwm.setclock()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#pwmsetclock)
* [pwm.getclock()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#pwmgetclock)
* [pwm.setduty()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#pwmsetduty)
* [pwm.getduty()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#pwmgetduty)

**net module**

* [net.createServer()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#netcreateserver)
* [net.createConnection()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#netcreateconnection)

**net.server sub-module**

* [net.server:listen()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#netserverlisten)
* [net.server:close()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#netserverclose)

**net.socket sub-module**

* [net.socket:connect()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#netsocketconnect)
* [net.socket:send()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#netsocketsend)
* [net.socket:on()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#netsocketon)
* [net.socket:close()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#netsocketclose)
* [net.socket:dns()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#netsocketdns)

**net.dns sub-module**

* [net.dns:setdnsserver()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#netdnssetdnsserver)
* [net.dns:getdnsserver()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#netdnsgetdnsserver)
* [net.dns:resolve()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#netdnsresolve)

**i2c module**

* [i2c.setup()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#i2csetup)
* [i2c.start()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#i2cstart)
* [i2c.stop()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#i2cstop)
* [i2c.address()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#i2caddress)
* [i2c.write()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#i2cwrite)
* [i2c.read()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#i2cread)

**adc module**

* [adc.read()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#adcread)
* [adc.readvdd33()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#adcreadvdd33)

**uart module**

* [uart.setup()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#uartsetup)
* [uart.on()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#uarton)
* [uart.write()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#uartwrite)

**1-wire module**

* [ow.setup()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#owsetup)
* [ow.reset()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#owreset)
* [ow.skip()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#owskip)
* [ow.select()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#owselect)
* [ow.write()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#owwrite)
* [ow.write\_bytes()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#owwritebytes)
* [ow.read()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#owread)
* [ow.read\_bytes()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#owreadbytes)
* [ow.depower()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#owdepower)
* [ow.reset\_search()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#owresetsearch)
* [ow.target\_search()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#owtargetsearch)
* [ow.search()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#owsearch)
* [ow.crc8()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#owcrc8)
* [ow.check\_crc16()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#owcheckcrc16)
* [ow.crc16()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#owcrc16)

**bit module**

* [bit.bnot()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#bitbnot)
* [bit.band()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#bitband)
* [bit.bor()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#bitbor)
* [bit.bxor()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#bitbxor)
* [bit.lshift()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#bitlshift)
* [bit.rshift()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#bitrshift)
* [bit.arshift()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#bitarshift)
* [bit.bit()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#bitbit)
* [bit.set()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#bitset)
* [bit.clear()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#bitclear)
* [bit.isset()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#bitisset)
* [bit.isclear()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#bitisclear)

**spi module**

* [spi.setup()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#spisetup)
* [spi.send()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#spisend)
* [spi.recv()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#spirecv)
* [spi.set\_mosi()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#spisetmosi)
* [spi.get\_miso()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#spigetmiso)
* [spi.transaction()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#spitransaction)

**mqtt module**

* [mqtt.Client()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#mqttclient)

**mqtt.client sub-module**

* [mqtt.client:lwt()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#mqttclientlwt)
* [mqtt.client:connect()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#mqttclientconnect)
* [mqtt.client:close()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#mqttclientclose)
* [mqtt.client:publish()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#mqttclientpublish)
* [mqtt.client:subscribe()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#mqttclientsubscribe)
* [mqtt.client:on()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#mqttclienton)

**WS2801 module**

* [ws2801.init()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#ws2801_init)
* [ws2801.write()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#ws2801_write)

**WS2812 module**

* [ws2812.writergb()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#ws2812writergb)

**cjson module**

* [cjson.encode()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#cjsonencode)
* [cjson.decode()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#cjsondecode)

**crypto module**

* [crypto.hash()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#crypto_hash)
* [crypto.hmac()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#crypto_hmac)
* [crypto.mask()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#crypto_mask)
* [crypto.toHex()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#crypto_tohex)
* [crypto.toBase64()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#crypto_tobase64)

**u8g module**

**I2C Display Drivers**

* [u8g.sh1106\_128x64\_i2c()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_i2c) -- Only in dev branch
* [u8g.ssd1306\_128x64\_i2c()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_i2c)
* [u8g.ssd1306\_64x48\_i2c()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_i2c) -- Only in dev branch
* [u8g.ssd1309\_128x64\_i2c()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_i2c) -- Only in dev branch
* [u8g.ssd1327\_96x96\_gr\_i2c()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_i2c) -- Only in dev branch
* [u8g.uc1611\_dogm240\_i2c()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_i2c) -- Only in dev branch
* [u8g.uc1611\_dogxl240\_i2c()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_i2c) -- Only in dev branch

**SPI Display Drivers**

* [u8g.ld7032\_60x32\_hw\_spi()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_hw_spi) -- Only in dev branch
* [u8g.pcd8544\_84x48\_hw\_spi()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_hw_spi) -- Only in dev branch
* [u8g.pcf8812\_96x65\_hw\_spi()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_hw_spi) -- Only in dev branch
* [u8g.sh1106\_128x64\_hw\_spi()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_hw_spi) -- API change in dev branch
* [u8g.ssd1306\_128x64\_hw\_spi()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_hw_spi) -- Only in dev branch
* [u8g.ssd1306\_64x48\_hw\_spi()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_hw_spi) -- Only in dev branch
* [u8g.ssd1309\_128x64\_hw\_spi()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_hw_spi) -- Only in dev branch
* [u8g.ssd1322\_nhd31oled\_bw\_hw\_spi()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_hw_spi) -- Only in dev branch
* [u8g.ssd1322\_nhd31oled\_gr\_hw\_spi()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_hw_spi) -- Only in dev branch
* [u8g.ssd1325\_nhd27oled\_bw\_hw\_spi()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_hw_spi) -- Only in dev branch
* [u8g.ssd1325\_nhd27oled\_gr\_hw\_spi()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_hw_spi) -- Only in dev branch
* [u8g.ssd1327\_96x96\_gr\_hw\_spi()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_hw_spi) -- Only in dev branch
* [u8g.ssd1351\_128x128\_332\_hw\_spi()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_hw_spi) -- Only in dev branch
* [u8g.ssd1351\_128x128gh\_332\_hw\_spi()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_hw_spi) -- Only in dev branch
* [u8g.ssd1351\_128x128\_hicolor\_hw\_spi()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_hw_spi) -- Only in dev branch
* [u8g.ssd1351\_128x128gh\_hicolor\_hw\_spi()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_hw_spi) -- Only in dev branch
* [u8g.ssd1353\_160x128\_332\_hw\_spi()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_hw_spi) -- Only in dev branch
* [u8g.ssd1353\_160x128\_hicolor\_hw\_spi()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_hw_spi) -- Only in dev branch
* [u8g.st7565\_64128n\_hw\_spi()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_hw_spi) -- Only in dev branch
* [u8g.st7565\_dogm128\_hw\_spi()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_hw_spi) -- Only in dev branch
* [u8g.st7565\_dogm132\_hw\_spi()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_hw_spi) -- Only in dev branch
* [u8g.st7565\_lm6059\_hw\_spi()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_hw_spi) -- Only in dev branch
* [u8g.st7565\_lm6063\_hw\_spi()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_hw_spi) -- Only in dev branch
* [u8g.st7565\_nhd\_c12832\_hw\_spi()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_hw_spi) -- Only in dev branch
* [u8g.st7565\_nhd\_c12864\_hw\_spi()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_hw_spi) -- Only in dev branch
* [u8g.uc1601\_c128032\_hw\_spi()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_hw_spi) -- Only in dev branch
* [u8g.uc1608\_240x128\_hw\_spi()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_hw_spi) -- Only in dev branch
* [u8g.uc1608\_240x64\_hw\_spi()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_hw_spi) -- Only in dev branch
* [u8g.uc1610\_dogxl160\_bw\_hw\_spi()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_hw_spi) -- Only in dev branch
* [u8g.uc1610\_dogxl160\_gr\_hw\_spi()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_hw_spi) -- Only in dev branch
* [u8g.uc1611\_dogm240\_hw\_spi()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_hw_spi) -- Only in dev branch
* [u8g.uc1611\_dogxl240\_hw\_spi()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_hw_spi) -- Only in dev branch
* [u8g.uc1701\_dogs102\_hw\_spi()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_hw_spi) -- Only in dev branch
* [u8g.uc1701\_mini12864\_hw\_spi()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8g_generic_hw_spi) -- Only in dev branch

**u8g.disp sub-module**

* [u8g.disp:begin()](https://github.com/olikraus/u8glib/wiki/userreference#begin)
* [u8g.disp:drawBitmap()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8gdisp_drawbitmap)
* [u8g.disp:drawBox()](https://github.com/olikraus/u8glib/wiki/userreference#drawbox)
* [u8g.disp:drawCircle()](https://github.com/olikraus/u8glib/wiki/userreference#drawcircle)
* [u8g.disp:drawDisc()](https://github.com/olikraus/u8glib/wiki/userreference#drawdisc)
* [u8g.disp:drawEllipse()](https://github.com/olikraus/u8glib/wiki/userreference#drawellipse)
* [u8g.disp:drawFilledEllipse()](https://github.com/olikraus/u8glib/wiki/userreference#drawfilledellipse)
* [u8g.disp:drawFrame()](https://github.com/olikraus/u8glib/wiki/userreference#drawframe)
* [u8g.disp:drawHLine()](https://github.com/olikraus/u8glib/wiki/userreference#drawhline)
* [u8g.disp:drawLine()](https://github.com/olikraus/u8glib/wiki/userreference#drawline)
* [u8g.disp:drawPixel()](https://github.com/olikraus/u8glib/wiki/userreference#drawpixel)
* [u8g.disp:drawRBox()](https://github.com/olikraus/u8glib/wiki/userreference#drawrbox)
* [u8g.disp:drawRFrame()](https://github.com/olikraus/u8glib/wiki/userreference#drawrframe)
* [u8g.disp:drawStr()](https://github.com/olikraus/u8glib/wiki/userreference#drawstr)
* [u8g.disp:drawStr90()](https://github.com/olikraus/u8glib/wiki/userreference#drawstr90)
* [u8g.disp:drawStr180()](https://github.com/olikraus/u8glib/wiki/userreference#drawstr180)
* [u8g.disp:drawStr270()](https://github.com/olikraus/u8glib/wiki/userreference#drawstr270)
* [u8g.disp:drawTriangle()](https://github.com/olikraus/u8glib/wiki/userreference#drawtriangle)
* [u8g.disp:drawVLine()](https://github.com/olikraus/u8glib/wiki/userreference#drawvline)
* [u8g.disp:drawXBM()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8gdisp_drawXBM)
* [u8g.disp:firstPage()](https://github.com/olikraus/u8glib/wiki/userreference#firstpage)
* [u8g.disp:getColorIndex()](https://github.com/olikraus/u8glib/wiki/userreference#getcolorindex)
* [u8g.disp:getFontAscent()](https://github.com/olikraus/u8glib/wiki/userreference#getfontascent)
* [u8g.disp:getFontDescent()](https://github.com/olikraus/u8glib/wiki/userreference#getfontdescent)
* [u8g.disp:getFontLineSpacing()](https://github.com/olikraus/u8glib/wiki/userreference#getfontlinespacing)
* [u8g.disp:getHeight()](https://github.com/olikraus/u8glib/wiki/userreference#getheight)
* [u8g.disp:getMode()](https://github.com/olikraus/u8glib/wiki/userreference#getmode)
* [u8g.disp:getWidth()](https://github.com/olikraus/u8glib/wiki/userreference#getwidth)
* [u8g.disp:getStrWidth()](https://github.com/olikraus/u8glib/wiki/userreference#getstrwidth)
* [u8g.disp:nextPage()](https://github.com/olikraus/u8glib/wiki/userreference#nextpage)
* [u8g.disp:setColorIndex()](https://github.com/olikraus/u8glib/wiki/userreference#setcolortndex)
* [u8g.disp:setDefaultBackgroundColor()](https://github.com/olikraus/u8glib/wiki/userreference#setdefaultbackgroundcolor)
* [u8g.disp:setDefaultForegroundColor()](https://github.com/olikraus/u8glib/wiki/userreference#setdefaultforegroundcolor)
* [u8g.disp:setFont()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#u8gdisp_setFont)
* [u8g.disp:setFontLineSpacingFactor()](https://github.com/olikraus/u8glib/wiki/userreference#setfontlinespacingfactor)
* [u8g.disp:setFontPosBaseline()](https://github.com/olikraus/u8glib/wiki/userreference#setfontposbaseline)
* [u8g.disp:setFontPosBottom()](https://github.com/olikraus/u8glib/wiki/userreference#setfontposbottom)
* [u8g.disp:setFontPosCenter()](https://github.com/olikraus/u8glib/wiki/userreference#setfontposcenter)
* [u8g.disp:setFontPosTop()](https://github.com/olikraus/u8glib/wiki/userreference#setfontpostop)
* [u8g.disp:setFontRefHeightAll()](https://github.com/olikraus/u8glib/wiki/userreference#setfontrefheightall)
* [u8g.disp:setFontRefHeightExtendedText()](https://github.com/olikraus/u8glib/wiki/userreference#setfontrefheightextendedtext)
* [u8g.disp:setFontRefHeightText()](https://github.com/olikraus/u8glib/wiki/userreference#setfontrefheighttext)
* [u8g.disp:setRot90()](https://github.com/olikraus/u8glib/wiki/userreference#setrot90)
* [u8g.disp:setRot180()](https://github.com/olikraus/u8glib/wiki/userreference#setrot180)
* [u8g.disp:setRot270()](https://github.com/olikraus/u8glib/wiki/userreference#setrot270)
* [u8g.disp:setScale2x2()](https://github.com/olikraus/u8glib/wiki/userreference#setscale2x2)
* [u8g.disp:sleepOn()](https://github.com/olikraus/u8glib/wiki/userreference#sleepon)
* [u8g.disp:sleepOff()](https://github.com/olikraus/u8glib/wiki/userreference#sleepoff)
* [u8g.disp:undoRotation()](https://github.com/olikraus/u8glib/wiki/userreference#undorotation)
* [u8g.disp:undoScale()](https://github.com/olikraus/u8glib/wiki/userreference#undoscale)

**dht module**

* [dht.read()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#dhtread)
* [dht.read11()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#dhtread11)
* [dht.readxx()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#dhtreadxx)

**-**[Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**Summary**

* Easy to access wireless router
* Based on Lua 5.1.4, Developers are supposed to have experience with Lua Program language.
* Event-Drive programming modal.
* Build-in file, timer, pwm, i2c, net, gpio, wifi, uart, adc module.
* Serial Port BaudRate:9600
* Re-mapped GPIO pin, use the index to program gpio, i2c, pwm.
* GPIO Map Table:

**GPIO NEW TABLE ( Build 20141219 and later)**

**new\_gpio\_map**

| **IO index** | **ESP8266 pin** | **IO index** | **ESP8266 pin** |
| --- | --- | --- | --- |
| 0 [\*] | GPIO16 | 7 | GPIO13 |
| 1 | GPIO5 | 8 | GPIO15 |
| 2 | GPIO4 | 9 | GPIO3 |
| 3 | GPIO0 | 10 | GPIO1 |
| 4 | GPIO2 | 11 | GPIO9 |
| 5 | GPIO14 | 12 | GPIO10 |
| 6 | GPIO12 |  |  |

\*\* [*] D0(GPIO16) can only be used as gpio read/write. no interrupt supported. no pwm/i2c/ow supported. \**

**Example**

gpiolookup = {[0]=3,[1]=10,[2]=4,[3]=9,[4]=1,[5]=2,[10]=12,[12]=6,[13]=7,[14]=5,[15]=8,[16]=0}

pin = gpiolookup[2] -- map ESP8266 GPIO2 to NodeMCU pin 4

**-**[Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**GPIO OLD TABLE (Before build 20141212)**

**old\_gpio\_map**

| **IO index** | **ESP8266 pin** | **IO index** | **ESP8266 pin** |
| --- | --- | --- | --- |
| 0 | GPIO12 | 6 | GPIO9 |
| 1 | GPIO13 | 7 | GPIO10 |
| 2 | GPIO14 | 8 | GPIO0 |
| 3 | GPIO15 | 9 | GPIO2 |
| 4 | GPIO3 | 10 | GPIO4 |
| 5 | GPIO1 | 11 | GPIO5 |

**-**[Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**Burn/Flash Firmware**

**Address**

nodemcu\_512k.bin: 0x00000  
See NodeMCU flash tool:  
[nodemcu-flasher](https://github.com/nodemcu/nodemcu-flasher)

**node module**

**node.restart()**

**Description**

restart the chip.

**Syntax**

node.restart()

**Parameters**

* nil

**Returns**

* nil

**Example**

node.restart();

**See also**

* [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**node.dsleep()**

**Description**

Enter deep sleep mode, wake up when timed out.

**Syntax**

node.dsleep(us, option)

**Note:** This function can only be used in the condition that esp8266 PIN32(RST) and PIN8(XPD\_DCDC aka GPIO16) are connected together. Using sleep(0) will set no wake up timer, connect a GPIO to pin RST, the chip will wake up by a falling-edge on pin RST.  
option=0, init data byte 108 is valuable;  
option>0, init data byte 108 is valueless.  
More details as follows:  
0, RF\_CAL or not after deep-sleep wake up, depends on init data byte 108.  
1, RF\_CAL after deep-sleep wake up, there will belarge current.  
2, no RF\_CAL after deep-sleep wake up, there will only be small current.  
4, disable RF after deep-sleep wake up, just like modem sleep, there will be the smallest current.

**Parameters**

* us: number(Integer) or nil, sleep time in micro second. If us = 0, it will sleep forever. If us = nil, will not set sleep time.
* option: number(Integer) or nil. If option = nil, it will use last alive setting as default option.

**Returns**

* nil

**Example**

--do nothing

node.dsleep()

--sleep μs

node.dsleep(1000000)

--set sleep option, then sleep μs

node.dsleep(1000000, 4)

--set sleep option only

node.dsleep(nil,4)

**See also**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**node.info()**

**Description**

return NodeMCU version, chipid, flashid, flash size, flash mode, flash speed.

**Syntax**

node.info()

**Parameters**

* nil

**Returns**

* majorVer (number)
* minorVer (number)
* devVer (number)
* chipid (number)
* flashid (number)
* flashsize (number)
* flashmode (number)
* flashspeed (number)

**Example**

majorVer, minorVer, devVer, chipid, flashid, flashsize, flashmode, flashspeed = node.info();

print("NodeMCU "..majorVer.."."..minorVer.."."..devVer)

**See also**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**node.chipid()**

**Description**

return chip ID

**Syntax**

node.chipid()

**Parameters**

nil

**Returns**

number:chip ID

**Example**

id = node.chipid();

**See also**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**node.flashid()**

**Description**

return flashid ID

**Syntax**

node.flashid()

**Parameters**

nil

**Returns**

number:flash ID

**Example**

flashid = node.flashid();

**See also**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**node.heap()**

**Description**

return the remain HEAP size in bytes

**Syntax**

node.heap()

**Parameters**

nil

**Returns**

number: system heap size left in bytes

**Example**

heap\_size = node.heap();

**See also**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**node.key()**

**Description**

define button function, button is connected to GPIO16.

**Syntax**

node.key(type, function())

**Parameters**

type: type is either string "long" or "short". long: press the key for 3 seconds, short: press shortly(less than 3 seconds)  
function(): user defined function which is called when key is pressed. If nil, cancling the user defined function.  
Default function: long: change LED blinking rate, short: reset chip

**Returns**

nil

**Example**

node.key("long", function() print('hello world') end)

**See also**

**-** [node.led](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#nodeled)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**node.led()**

**Description**

setup the on/off time for led, which connected to GPIO16, multiplexing with node.key()

**Syntax**

node.led(low, high)

**Parameters**

Low: LED off time, LED keeps on when low=0. Unit: milliseconds, time resolution: 80~100ms  
High: LED on time. Unit: milliseconds, time resolution: 80~100ms

**Returns**

nil

**Example**

-- turn led on forever.

node.led(0);

**See also**

**-** [node.key](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#nodekey)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**node.input()**

**Description**

accept a string and put the string into Lua interpretor.  
same as pcall(loadstring(str)) but support multi seperated line.

**Syntax**

node.input(str)

**Parameters**

str: Lua chunk

**Returns**

nil

**Example**

-- never use node.input() in console. no effect.

sk:on("receive", function(conn, payload) node.input(payload) end)

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**node.output()**

**Description**

direct output from lua interpretor to a call back function.

**Syntax**

node.output(function(str), serial\_debug)

**Parameters**

function(str): a function accept every output as str, and can send the output to a socket.  
serial\_debug: 1 output also show in serial. 0: no serial output.

**Returns**

nil

**Example**

function tonet(str)

sk:send(str)

-- print(str) WRONG!!! never ever print something in this function

-- because this will cause a recursive function call!!!

end

node.ouput(tonet, 1) -- serial also get the lua output.

-- a simple telnet server

s=net.createServer(net.TCP)

s:listen(2323,function(c)

con\_std = c

function s\_output(str)

if(con\_std~=nil)

then con\_std:send(str)

end

end

node.output(s\_output, 0) -- re-direct output to function s\_ouput.

c:on("receive",function(c,l)

node.input(l) -- works like pcall(loadstring(l)) but support multiple separate line

end)

c:on("disconnection",function(c)

con\_std = nil

node.output(nil) -- un-regist the redirect output function, output goes to serial

end)

end)

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**node.readvdd33()**

**Description**

Reading vdd33 pin voltage

**Syntax**

node.readvdd33()

**Parameters**

no parameters

**Returns**

mV

**Example**

print(node.readvdd33())

output

3345

v = node.readvdd33() / 1000

print(v)

v=nil

output

3.315

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**node.compile()**

**Description**

compile lua text file into lua bytecode file, and save it as .lc file.

**Syntax**

node.compile("file.lua")

**Parameters**

lua text file end with ".lua"

**Returns**

nil

**Example**

file.open("hello.lua","w+")

file.writeline([[print("hello nodemcu")]])

file.writeline([[print(node.heap())]])

file.close()

node.compile("hello.lua")

dofile("hello.lua")

dofile("hello.lc")

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**node.setcpufreq()**

**Description**

Change the working CPU Frequency

**Syntax**

node.setcpufreq(speed)

**Parameters**

speed: node.CPU80MHZ or node.CPU160MHZ

**Returns**

return targe CPU Frequency

**Example**

node.setcpufreq(node.CPU80MHZ)

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**node.restore()**

**Description**

Restore all configuration of Wi-Fi system.

**Syntax**

node.restore()

**Parameters**

none

**Returns**

none

**Example**

node.restore()

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**file module**

**file.remove()**

**Description**

remove file from file system.

**Syntax**

file.remove(filename)

**Parameters**

filename: file to remove

**Returns**

nil

**Example**

-- remove "foo.lua" from file system.

file.remove("foo.lua")

**See also**

**-** [file.open()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#file_open)  
**-** [file.close()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#file_close)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**file.open()**

**Description**

open file.

**Syntax**

file.open(filename, mode)

**Parameters**

filename: file to be opened, directories are not supported  
mode:

* "r": read mode (the default)
* "w": write mode
* "a": append mode
* "r+": update mode, all previous data is preserved
* "w+": update mode, all previous data is erased
* "a+": append update mode, previous data is preserved, writing is only allowed at the end of file

**Returns**

nil: file not opened, or not exists. true: file opened ok.

**Example**

-- open 'init.lua', print the first line.

file.open("init.lua", "r")

print(file.readline())

file.close()

**See also**

**-** [file.close()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#file_close)  
**-** [file.readline()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#file_readline)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**file.close()**

**Description**

close the file.

**Syntax**

file.close()

**Parameters**

nil

**Returns**

nil

**Example**

-- open 'init.lua', print the first line.

file.open("init.lua", "r")

print(file.readline())

file.close()

**See also**

**-** [file.open()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#file_open)  
**-** [file.readline()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#file_readline)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**file.readline()**

**Description**

read one line of file which is opened before.

**Syntax**

file.readline()

**Parameters**

nil

**Returns**

file content in string, line by line, include EOL('\n')  
return nil when EOF.

**Example**

-- print the first line of 'init.lua'

file.open("init.lua", "r")

print(file.readline())

file.close()

**See also**

**-** [file.open()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#file_open)  
**-** [file.close()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#file_close)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**file.writeline()**

**Description**

write string to file and add a '\n' at the end.

**Syntax**

file.writeline(string)

**Parameters**

string: content to be write to file

**Returns**

true: write ok. nil: there is error

**Example**

-- open 'init.lua' in 'a+' mode

file.open("init.lua", "a+")

-- write 'foo bar' to the end of the file

file.writeline('foo bar')

file.close()

**See also**

**-** [file.open()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#file_open)  
**-** [file.write()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#file_write)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**file.read()**

**Description**

read content of file which is opened before.

**Syntax**

file.read()

**Parameters**

if nothing passed in, read all byte in file. if pass a number n, then read n byte from file, or EOF is reached. if pass a string "q", then read until 'q' or EOF is reached.

**Returns**

file content in string  
return nil when EOF.

**Example**

-- print the first line of 'init.lua'

file.open("init.lua", "r")

print(file.read('\r'))

file.close()

-- print the first 5 byte of 'init.lua'

file.open("init.lua", "r")

print(file.read(5))

file.close()

**See also**

**-** [file.open()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#file_open)  
**-** [file.close()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#file_close)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**file.write()**

**Description**

write string to file.

**Syntax**

file.write(string)

**Parameters**

string: content to be write to file.

**Returns**

true: write ok. nil: there is error

**Example**

-- open 'init.lua' in 'a+' mode

file.open("init.lua", "a+")

-- write 'foo bar' to the end of the file

file.write('foo bar')

file.close()

**See also**

**-** [file.open()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#file_open)  
**-** [file.writeline()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#file_writeline)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**file.flush()**

**Description**

flush to file.

**Syntax**

file.flush()

**Parameters**

nil

**Returns**

nil

**Example**

-- open 'init.lua' in 'a+' mode

file.open("init.lua", "a+")

-- write 'foo bar' to the end of the file

file.write('foo bar')

file.flush()

file.close()

**See also**

**-** [file.open()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#file_open)  
**-** [file.writeline()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#file_writeline)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**file.seek()**

**Description**

Sets and gets the file position, measured from the beginning of the file, to the position given by offset plus a base specified by the string whence.

**Syntax**

file.seek(whence, offset)

**Parameters**

whence:  
"set": base is position 0 (beginning of the file);  
"cur": base is current position;(default value)  
"end": base is end of file;  
offset: default 0

**Returns**

success: returns the final file position  
fail: returns nil

**Example**

-- open 'init.lua' in 'a+' mode

file.open("init.lua", "a+")

-- write 'foo bar' to the end of the file

file.write('foo bar')

file.flush()

file.seek("set")

print(file.readline())

file.close()

**See also**

**-** [file.open()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#file_open)  
**-** [file.writeline()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#file_writeline)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**file.list()**

**Description**

list all files.

**Syntax**

file.list()

**Parameters**

nil

**Returns**

a lua table which contains the {file name: file size} pairs

**Example**

l = file.list();

for k,v in pairs(l) do

print("name:"..k..", size:"..v)

end

**See also**

**-** [file.remove()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#file_remove)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**file.format()**

**Description**

format file system.

**Syntax**

file.format()

**Parameters**

nil

**Returns**

nil

**Example**

file.format()

**See also**

**-** [file.remove()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#file_remove)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**file.rename()**

**Description**

rename a file. **NOTE:** the current opened file will be closed.

**Syntax**

file.rename(oldname, newname)

**Parameters**

oldname: old file name, directories are not supported  
newname: new file name, directories are not supported

**Returns**

false: rename failed. true: rename ok.

**Example**

-- rename file 'temp.lua' to 'init.lua'.

file.rename("temp.lua","init.lua")

**See also**

**-** [file.close()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#file_close)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**file.fsinfo()**

**Description**

Get file system info

**Syntax**

* file.fsinfo()

**Parameters**

* nil

**Returns**

* remaining (number)
* used (number)
* total (number)

**Example**

-- get file system info

remaining, used, total=file.fsinfo()

print("\nFile system info:\nTotal : "..total.." Bytes\nUsed : "..used.." Bytes\nRemain: "..remaining.." Bytes\n")

**See also**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**wifi module**

**CONSTANT**

wifi.STATION, wifi.SOFTAP, wifi.STATIONAP

**wifi.setmode()**

**Description**

setup wifi operation mode.

* wifi.STATION is when the device is connected to another wifi router. This is often done to give the device access to the internet.
* wifi.SOFTAP is when the device is acting as ONLY an access point. This mode will allow you to see the device in the list of wifi networks. In this mode your computer can connect to the device creating a local area network. Unless you change the value, the ESP8266 device will be given a local IP address of 192.168.4.1 and assign your computer the next available IP, such as: 192.168.4.2.
* wifi.STATIONAP is a combination of wifi.STATION and wifi.SOFTAP. It allows you to create a local wifi connection AND connect to another wifi router.

**Syntax**

wifi.setmode(mode)

**Parameters**

mode: value should be:

* wifi.STATION
* wifi.SOFTAP
* wifi.STATIONAP

**Returns**

current mode after setup

**Example**

wifi.setmode(wifi.STATION)

**See also**

**-** [wifi.getmode()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifi_getmode)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**wifi.getmode()**

**Description**

get wifi operation mode.

**Syntax**

wifi.getmode()

**Parameters**

nil

**Returns**

wifi operation mode

1 = STATION  
2 = SOFTAP  
3 = STATIONAP

**Example**

print(wifi.getmode())

**See also**

**-** [wifi.setmode()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifi_setmode)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**wifi.getchannel()**

**Description**

get current wifi channel.

**Syntax**

wifi.getchannel()

**Parameters**

nil

**Returns**

current wifi channel

**Example**

print(wifi.getchannel())

**See also**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**wifi.setphymode()**

**Description**

Setup 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: value should be:

* wifi.PHYMODE\_B
* wifi.PHYMODE\_G
* wifi.PHYMODE\_N

**Returns**

Current physical mode after setup

**Example**

--STATION

wifi.setphymode()

**See also**

**-** [wifi.getphymode()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifi_getphymode)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**wifi.getphymode()**

**Description**

get wifi physical mode.

**Syntax**

wifi.getmode()

**Parameters**

nil

**Returns**

wifi physical mode

* 1: wifi.PHYMODE\_B
* 2: wifi.PHYMODE\_G
* 3: wifi.PHYMODE\_N

**Example**

print(wifi.getphymode())

**See also**

**-** [wifi.setphymode()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifisetphymode)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**wifi.startsmart()**

**Description**

starts to auto configuration, if success set up ssid and pwd automatically.  
in new method, must using under wifi.STATION mode.

**Syntax**

[old] wifi.startsmart(channel, function succeed\_callback())  
[new] wifi.startsmart(type, function succeed\_callback(ssid, password))

**Parameters**

[old]  
channel: 1~13, startup channel for searching, if nil, default to 6. 20 seconds for each channel.  
succeed\_callback: callback function called after configuration, which is called when got password and connected to AP.  
[new]  
type: 0~1. 0 means use ESP\_TOUCH, 1 means use AIR\_KISS. The APP (Android and iOS) can download with full source code from <https://github.com/espressifapp>. Note: dev096 branch only suppurt ESP Touch V021, please check out old version.   
succeed\_callback: with 2 arguments, ssid and password, called after configuration.

**Returns**

nil

**Example**

-- old

wifi.startsmart(6, function() end)

-- new

wifi.setmode(wifi.STATION)

wifi.startsmart(0,

function(ssid, password)

print(string.format("Success. SSID:%s ; PASSWORD:%s", ssid, password))

end

)

-- above sdk v120, can get phone ip, must use esptouch v034.

wifi.startsmart(0,function(ssid, password,phoneip) print(ssid) print(password) print(phoneip) end)

**See also**

**-** [wifi.stopsmart()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifi_stopsmart)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**wifi.stopsmart()**

**Description**

stop the configuring process.

**Syntax**

wifi.stopsmart()

**Parameters**

nil

**Returns**

nil

**Example**

wifi.stopsmart()

**See also**

**-** [wifi.startsmart()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifi_startsmart)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**wifi.sleeptype()**

**Description**

config the sleep type for wifi modem.

**Syntax**

type\_actual = wifi.sleeptype(type\_need)

**Parameters**

type\_need:  
wifi.NONE\_SLEEP, wifi.LIGHT\_SLEEP, wifi.MODEM\_SLEEP

**Returns**

type\_actual:  
wifi.NONE\_SLEEP, wifi.LIGHT\_SLEEP, wifi.MODEM\_SLEEP

**Example**

realtype = wifi.sleeptype(wifi.MODEM\_SLEEP)

**See also**

**-** [node.dsleep()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#node_dsleep)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**wifi.sta.getconfig()**

**Description**

Get wifi station configuration.  
Note: If bssid\_set is equal to 0 then bssid is irrelevant

**Syntax**

ssid, password, bssid\_set, bssid=wifi.sta.getconfig()

**Parameters**

nil

**Returns**

ssid, password, bssid\_set, bssid

**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()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifi_sta_connect)  
**-** [wifi.sta.disconnect()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifi_sta_disconnect)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**wifi.sta.config()**

**Description**

Set wifi station configuration

**Syntax**

wifi.sta.config(ssid, password)  
wifi.sta.config(ssid, password, auto)  
wifi.sta.config(ssid, password, bssid)  
wifi.sta.config(ssid, password, auto, bssid)

**Parameters**

* ssid: string which is less than 32 bytes.
* password: string which is less than 64 bytes.
* auto: value of 0 or 1 (Default is 1)  
  + 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 Access Point, User decides when to connect/disconnect to/from AP

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 to/from AP

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()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifi_sta_connect)  
**-** [wifi.sta.disconnect()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifi_sta_disconnect)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**wifi.sta.connect()**

**Description**

connect to AP in station mode.

**Syntax**

wifi.sta.connect()

**Parameters**

nil

**Returns**

nil

**Example**

wifi.sta.connect()

**See also**

**-** [wifi.sta.disconnect()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifi_sta_disconnect)  
**-** [wifi.sta.config()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifi_sta_config)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**wifi.sta.disconnect()**

**Description**

disconnect from AP in station mode.

**Syntax**

wifi.sta.disconnect()

**Parameters**

nil

**Returns**

nil

**Example**

wifi.sta.disconnect()

**See also**

**-** [wifi.sta.config()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifi_sta_config)  
**-** [wifi.sta.connect()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifi_sta_connect)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**wifi.sta.autoconnect()**

**Description**

auto connect 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()

**See also**

**-** [wifi.sta.config()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifi_sta_config)  
**-** [wifi.sta.connect()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifi_sta_connect)  
**-** [wifi.sta.disconnect()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifi_sta_disconnect)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**wifi.sta.getip()**

**Description**

get ip, netmask, gateway address in station mode.

**Syntax**

wifi.sta.getip()

**Parameters**

nil

**Returns**

ip, netmask, gateway address in string, for example:"192.168.0.111"  
return nil if ip = "0.0.0.0".

**Example**

-- print current ip, 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()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifi_sta_getmac)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**wifi.sta.setip()**

**Description**

set ip, netmask, gateway address in station mode.

**Syntax**

wifi.sta.setip(cfg)

**Parameters**

cfg: table contain ip, netmask, and gateway

{

ip="192.168.0.111",

netmask="255.255.255.0",

gateway="192.168.0.1"

}

**Returns**

true if success, false if fail.

**Example**

cfg =

{

ip="192.168.0.111",

netmask="255.255.255.0",

gateway="192.168.0.1"

}

wifi.sta.setip(cfg)

**See also**

**-** [wifi.sta.setmac()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifi_sta_setmac)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**wifi.sta.getmac()**

**Description**

get mac address in station mode.

**Syntax**

wifi.sta.getmac()

**Parameters**

nil

**Returns**

mac address in string, for example:"18-33-44-FE-55-BB"

**Example**

-- print current mac address

print(wifi.sta.getmac())

**See also**

**-** [wifi.sta.getip()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifi_sta_getip)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**wifi.sta.setmac()**

**Description**

set mac address in station mode.

**Syntax**

wifi.sta.setmac(mac)

**Parameters**

mac address in string, for example:"DE:AD:BE:EF:7A:C0"

**Returns**

true if success, false if fail.

**Example**

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

**See also**

**-** [wifi.sta.setip()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifi_sta_setip)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**wifi.sta.getap()**

**Description**

scan and get ap list as a lua table into callback function.

**Syntax**

wifi.sta.getap(function(table))  
wifi.sta.getap(cfg, function(table))  
wifi.sta.getap(format, function(table))  
wifi.sta.getap(cfg, format, function(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, 0 or 1 is valid.(0 is Default)
  + 0: Old format (SSID : Authmode, RSSI, BSSID, Channel)
    - NOTE: When using old format for table output, any duplicate SSIDs will be discarded.
  + 1: New format (BSSID : SSID, RSSI, Authmode, Channel)
* function(table): a callback function to receive ap table when scan is done  
  + This function receives a table, the key is the ssid, value is other info in format: authmode,rssi,bssid,channel
  + If you are using the new output format, the key is the bssid, value is other info in format: ssid,rssi,authmode,channel

**Returns**

nil

**Example**

-- print ap list

function listap(t)

for k,v in pairs(t) do

print(k.." : "..v)

end

end

wifi.sta.getap(listap)

-- Print AP list that is easier to read

function listap(t) -- (SSID : Authmode, RSSI, BSSID, Channel)

print("\n"..string.format("%32s","SSID").."\tBSSID\t\t\t\t RSSI\t\tAUTHMODE\tCHANNEL")

for ssid,v in pairs(t) do

local authmode, rssi, bssid, channel = string.match(v, "([^,]+),([^,]+),([^,]+),([^,]+)")

print(string.format("%32s",ssid).."\t"..bssid.."\t "..rssi.."\t\t"..authmode.."\t\t\t"..channel)

end

end

wifi.sta.getap(listap)

--NOTE: The rest of the examples use the new style output format

-- print ap list

function listap(t)

for k,v in pairs(t) do

print(k.." : "..v)

end

end

wifi.sta.getap(1, listap)

-- Print AP list that is easier to read

function listap(t) -- (SSID : Authmode, RSSI, BSSID, Channel)

print("\n\t\t\tSSID\t\t\t\t\tBSSID\t\t\t RSSI\t\tAUTHMODE\t\tCHANNEL")

for bssid,v in pairs(t) do

local ssid, rssi, authmode, channel = string.match(v, "([^,]+),([^,]+),([^,]+),([^,]+)")

print(string.format("%32.s",ssid).."\t"..bssid.."\t "..rssi.."\t\t"..authmode.."\t\t\t"..channel)

end

end

wifi.sta.getap(1, listap)

--check for specific AP

function listap(t)

print("\n\t\t\tSSID\t\t\t\t\tBSSID\t\t\t RSSI\t\tAUTHMODE\t\tCHANNEL")

for bssid,v in pairs(t) do

local ssid, rssi, authmode, channel = string.match(v, "([^,]+),([^,]+),([^,]+),([^,]+)")

print(string.format("%32.s",ssid).."\t"..bssid.."\t "..rssi.."\t\t"..authmode.."\t\t\t"..channel)

end

end

scan\_cfg={}

scan\_cfg.ssid="myssid"

scan\_cfg.bssid="AA:AA:AA:AA:AA:AA"

scan\_cfg.channel=0

scan\_cfg.show\_hidden=1

wifi.sta.getap(scan\_cfg, 1, listap)

--get RSSI for currently configured AP

function listap(t)

for bssid,v in pairs(t) do

local ssid, rssi, authmode, channel = string.match(v, "([^,]+),([^,]+),([^,]+),([^,]+)")

print("CURRENT RSSI IS: "..rssi)

end

end

ssid, tmp, bssid\_set, bssid=wifi.sta.getconfig()

scan\_cfg={}

scan\_cfg.ssid=ssid

if bssid\_set==1 then scan\_cfg.bssid=bssid else scan\_cfg.bssid=nil end

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()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifi_sta_getip)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**wifi.sta.status()**

**Description**

get 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.

**See also**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**wifi.sta.getbroadcast()**

**Description**

get getbroadcast address in station mode.

**Syntax**

wifi.sta.getbroadcast()

**Parameters**

nil

**Returns**

getbroadcast address in string, for example:"192.168.0.255"  
return nil if ip = "0.0.0.0".

**Example**

bc = wifi.sta.getbroadcast()

print(bc)

-- 192.168.0.255

**See also**

**-** [wifi.sta.getip()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifi_sta_getip)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**wifi.sta.eventMonReg()**

**Description**

Register callback for wifi station status event

**Syntax**

wifi.sta.eventMonReg(wifi\_status, function)  
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
  + Valid wifi states:
    - 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**

Nothing.

**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()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifistaeventmonstart)

**-** [wifi.sta.eventMonStop()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifistaeventmonstop)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**wifi.sta.eventMonStart()**

**Description**

Start wifi station event monitor

**Syntax**

wifi.sta.eventMonStart()  
wifi.sta.eventMonStart(ms)

**Parameters**

* ms: interval between checks in milliseconds. defaults to 150 ms if not provided.

**Returns**

Nothing.

**Example**

--start wifi event monitor with default interval

wifi.sta.eventMonStart()

--start wifi event monitor with 100 ms interval

wifi.sta.eventMonStart(100)

**See also**

**-** [wifi.sta.eventMonReg()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifistaeventmonreg)

**-** [wifi.sta.eventMonStop()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifistaeventmonstop)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**wifi.sta.eventMonStop()**

**Description**

Stop wifi station event monitor

**Syntax**

wifi.sta.eventMonStop()  
wifi.sta.eventMonStop("unreg all")

**Parameters**

* "unreg all": unregister all previously registered functions

**Returns**

Nothing.

**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()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifistaeventmonreg)

**-** [wifi.sta.eventMonStart()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifistaeventmonstart)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**wifi.ap module**

**wifi.ap.config()**

**Description**

set ssid and pwd in ap mode. Be sure to make the pwd value at least 8 characters! If you don't make the pwd value 8 characters, it will default to no password and not set the value for ssid. It will still work as an access point, but you will see a name in your wifi list like: ESP\_9997C3

**Syntax**

wifi.ap.config(cfg)

**Parameters**

* string: SSID chars 1-32
* string: pwd chars 8-64
* int: auth Authentication AUTH\_OPEN | AUTH\_WPA\_PSK | AUTH\_WPA2\_PSK | AUTH\_WPA\_WPA2\_PSK default=Open
* int: channel no. 1-13 default=6
* int: hidden 0=not hidden 1=hidden default=0
* int: max maximal connections 1-4 default=4
* int beacon interval time of beacons value 100-60000 default=100

**Example:**

cfg={}

cfg.ssid="myssid"

cfg.pwd="mypassword"

wifi.ap.config(cfg)

**Returns**

nil

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**wifi.ap.getip()**

**Description**

get ip, netmask, gateway in ap mode.

**Syntax**

wifi.ap.getip()

**Parameters**

nil

**Returns**

ip, netmask, gateway address in string, for example:"192.168.0.111"  
return nil if ip = "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()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifi_ap_getmac)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**wifi.ap.setip()**

**Description**

set ip, netmask, gateway address in ap mode.

**Syntax**

wifi.ap.setip(cfg)

**Parameters**

cfg: table contain ip, netmask, and gateway

{

ip="192.168.1.1",

netmask="255.255.255.0",

gateway="192.168.1.1"

}

**Returns**

true if success, false if fail.

**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()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifi_ap_setmac)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**wifi.ap.getmac()**

**Description**

get mac address in ap mode.

**Syntax**

wifi.ap.getmac()

**Parameters**

nil

**Returns**

mac address in string, for example:"1A-33-44-FE-55-BB"

**Example**

wifi.ap.getmac()

**See also**

**-** [wifi.ap.getip()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifi_ap_getip)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**wifi.ap.setmac()**

**Description**

set 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 if fail.

**Example**

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

**See also**

**-** [wifi.ap.setip()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifi_ap_setip)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**wifi.ap.getclient()**

**Description**

get table of clients connected to ESP in ap mode.

**Syntax**

table=wifi.ap.getclient()

**Parameters**

nil

**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

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**wifi.ap.getbroadcast()**

**Description**

get getbroadcast address in ap mode.

**Syntax**

wifi.ap.getbroadcast()

**Parameters**

nil

**Returns**

getbroadcast address in string, for example:"192.168.0.255"  
return nil if ip = "0.0.0.0".

**Example**

bc = wifi.ap.getbroadcast()

print(bc)

-- 192.168.0.255

**See also**

**-** [wifi.ap.getip()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#wifi_ap_getip)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**wifi.ap.dhcp submodule**

**wifi.ap.dhcp.config()**

**Description**

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()**

**Description**

Start the dhcp service

**Syntax**

wifi.ap.dhcp.start()

**Returns**

boolean indicating success

**wifi.ap.dhcp.stop()**

**Description**

Stop the dhcp service

**Syntax**

wifi.ap.dhcp.stop()

**Returns**

boolean indicating success

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**timer module**

**tmr.delay()**

**Description**

delay us micro seconds.

**Syntax**

tmr.delay(us)

**Parameters**

us: delay time in micro second

**Returns**

nil

**Example**

-- delay 100us

tmr.delay(100)

**See also**

**-** [tmr.now()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#tmrnow)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**tmr.now()**

**Description**

return the current value of system counter: uint31, us.

**Syntax**

tmr.now()

**Parameters**

nil

**Returns**

uint31: value of counter

**Example**

-- print current value of counter

print(tmr.now())

**See also**

**-** [tmr.delay()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#tmrdelay)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**tmr.alarm()**

**Description**

alarm time.

**Syntax**

tmr.alarm(id, interval, repeat, function do())

**Parameters**

id: 0~6, alarmer id. Interval: alarm time, unit: millisecond  
repeat: 0 - one time alarm, 1 - repeat  
function do(): callback function for alarm timed out

**Returns**

nil

**Example**

-- print "hello world" every 1000ms

tmr.alarm(0, 1000, 1, function() print("hello world") end )

**See also**

**-** [tmr.now()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#tmrnow)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**tmr.stop()**

**Description**

stop alarm.

**Syntax**

tmr.stop(id)

**Parameters**

id: 0~6, alarmer id.

**Returns**

nil

**Example**

-- print "hello world" every 1000ms

tmr.alarm(1, 1000, 1, function() print("hello world") end )

-- something else

-- stop alarm

tmr.stop(1)

**See also**

**-** [tmr.now()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#tmrnow)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**tmr.wdclr()**

**Description**

clear system watchdog counter.

**Syntax**

tmr.wdclr()

**Parameters**

nil.

**Returns**

nil

**Example**

for i=1,10000 do

print(i)

tmr.wdclr() -- should call tmr.wdclr() in a long loop to avoid hardware reset caused by watchdog.

end

**See also**

**-** [tmr.delay()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#tmrdelay)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**tmr.time()**

**Description**

return rtc time since start up in second, uint31 form.

**Syntax**

tmr.time()

**Parameters**

nil.

**Returns**

number

**Example**

**See also**

**-** [tmr.now()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#tmrnow)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**GPIO module**

**CONSTANT**

gpio.OUTPUT, gpio.INPUT, gpio.INT, gpio.HIGH, gpio.LOW

**gpio.mode()**

**Description**

initialize pin to GPIO mode, set the pin in/out mode, internal pullup.

**Syntax**

gpio.mode(pin, mode, pullup)

**Parameters**

pin: 0~12, IO index  
mode: gpio.OUTPUT or gpio.INPUT, or gpio.INT(interrupt mode) pullup: gpio.PULLUP or gpio.FLOAT, default: gpio.FLOAT.

**Returns**

nil

**Example**

-- set gpio 0 as output.

gpio.mode(0, gpio.OUTPUT)

**See also**

**-** [gpio.read()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#gpioread)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**gpio.read()**

**Description**

read pin value.

**Syntax**

gpio.read(pin)

**Parameters**

pin: 0~12, IO index

**Returns**

number:0 - low, 1 - high

**Example**

-- read value of gpio 0.

gpio.read(0)

**See also**

**-** [gpio.mode()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#gpiomode)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**gpio.write()**

**Description**

set pin value.

**Syntax**

gpio.write(pin)

**Parameters**

pin: 0~12, IO index  
level: gpio.HIGH or gpio.LOW

**Returns**

nil

**Example**

-- set pin index 1 to GPIO mode, and set the pin to high.

pin=1

gpio.mode(pin, gpio.OUTPUT)

gpio.write(pin, gpio.HIGH)

**See also**

**-** [gpio.mode()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#gpiomode)  
**-** [gpio.read()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#gpioread)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**gpio.trig()**

**Description**

set the interrupt callback function for pin.

**Syntax**

gpio.trig(pin, type, function(level))

**Parameters**

pin: **1~12**, IO index, pin D0 does not support Interrupt.  
type: "up", "down", "both", "low", "high", which represent rising edge, falling edge, both edge, low level, high level trig mode separately.  
function(level): callback function when triggered. The gpio level is the param. Use previous callback function if undefined here.

**Returns**

nil

**Example**

-- use pin 0 as the input pulse width counter

pulse1 = 0

du = 0

gpio.mode(1,gpio.INT)

function pin1cb(level)

du = tmr.now() – pulse1

print(du)

pulse1 = tmr.now()

if level == 1 then gpio.trig(1, "down") else gpio.trig(1, "up") end

end

gpio.trig(1, "down",pin1cb)

**See also**

**-** [gpio.mode()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#gpiomode)  
**-** [gpio.write()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#gpiowrite)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**PWM module**

**pwm.setup()**

**Description**

set pin to PWM mode. Only 3 pins can be set to PWM mode at the most.

**Syntax**

pwm.setup(pin, clock, duty)

**Parameters**

pin: 1~12, IO index  
clock: 1~1000, pwm frequency  
duty: 0~1023, pwm duty cycle, max 1023(10bit)

**Returns**

nil

**Example**

-- set pin index 1 as pwm output, frequency is 100Hz, duty cycle is half.

pwm.setup(1, 100, 512)

**See also**

**-** [pwm.start()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#pwmstart)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**pwm.close()**

**Description**

quit PWM mode for specified pin.

**Syntax**

pwm.close(pin)

**Parameters**

pin: 1~12, IO index

**Returns**

nil

**Example**

pwm.close(1)

**See also**

**-** [pwm.start()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#pwmstart)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**pwm.start()**

**Description**

pwm starts, you can detect the waveform on the gpio.

**Syntax**

pwm.start(pin)

**Parameters**

pin: 1~12, IO index

**Returns**

nil

**Example**

pwm.start(1)

**See also**

**-** [pwm.stop()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#pwmstop)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**pwm.stop()**

**Description**

pause the output of PWM waveform.

**Syntax**

pwm.stop(pin)

**Parameters**

pin: 1~12, IO index

**Returns**

nil

**Example**

pwm.stop(1)

**See also**

**-** [pwm.start()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#pwmstart)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**pwm.setclock()**

**Description**

set pwm frequency for pin.  
**-Note:** setup pwm frequency will synchronously change others if there are any. Only one PWM frequency can be allowed for the system.

**Syntax**

pwm.setclock(pin, clock)

**Parameters**

pin: 1~12, IO index.  
clock: 1~1000, pwm frequency.

**Returns**

nil

**Example**

pwm.setclock(1, 100)

**See also**

**-** [pwm.getclock()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#pwmgetclock)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**pwm.getclock()**

**Description**

get pwm frequency of pin.

**Syntax**

pwm.getclock(pin)

**Parameters**

pin: 1~12, IO index.

**Returns**

number:pwm frequency of pin

**Example**

print(pwm.getclock(1))

**See also**

**-** [pwm.setclock()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#pwmsetclock)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**pwm.setduty()**

**Description**

set duty clycle for pin.

**Syntax**

pwm.setduty(pin, duty)

**Parameters**

pin: 1~12, IO index  
duty: 0~1023, pwm duty cycle, max 1023(10bit).

**Returns**

nil

**Example**

pwm.setduty(1, 512)

**See also**

**-** [pwm.getduty()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#pwmgetduty)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**pwm.getduty()**

**Description**

get duty clycle for pin.

**Syntax**

pwm.getduty(pin)

**Parameters**

pin: 1~12, IO index

**Returns**

number: duty cycle, max 1023.

**Example**

-- D1 is connected to green led

-- D2 is connected to blue led

-- D3 is connected to red led

pwm.setup(1,500,512)

pwm.setup(2,500,512)

pwm.setup(3,500,512)

pwm.start(1)

pwm.start(2)

pwm.start(3)

function led(r,g,b)

pwm.setduty(1,g)

pwm.setduty(2,b)

pwm.setduty(3,r)

end

led(512,0,0) -- set led to red

led(0,0,512) -- set led to blue.

**See also**

**-** [pwm.setduty()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#pwmsetduty)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**net module**

**CONSTANT**

net.TCP, net.UDP

**net.createServer()**

**Description**

create a server.

**Syntax**

net.createServer(type, timeout)

**Parameters**

type: net.TCP or net.UDP  
timeout: for a TCP server, timeout is 1~28800 seconds, for a inactive client to disconnected.

**Returns**

net.server sub module

**Example**

net.createServer(net.TCP, 30) -- 30s timeout

**See also**

**-** [net.createConnection()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#netcreateconnection)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**net.createConnection()**

**Description**

Create a client.

**Syntax**

net.createConnection(type, secure)

**Parameters**

type: net.TCP or net.UDP  
secure: 1 or 0, 1 for ssl link, 0 for normal link

**Returns**

net.socket sub module

**Example**

net.createConnection(net.UDP, 0)

**See also**

**-** [net.createServer()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#netcreateserver)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**net.server module**

**net.server:listen()**

**Description**

listen on port from [ip] address.

**Syntax**

net.server.listen(port,[ip],function(net.socket))

**Parameters**

port: port number  
ip:ip address string, can be omitted  
function(net.socket): callback function, pass to Caller function as param if a connection is created successfully

**Returns**

nil

**Example**

-- create a server

sv=net.createServer(net.TCP, 30) -- 30s time out for a inactive client

-- server listen on 80, if data received, print data to console, and send "hello world" to remote.

sv:listen(80,function(c)

c:on("receive", function(c, pl) print(pl) end)

c:send("hello world")

end)

**See also**

**-** [net.createServer()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#netcreateserver)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**net.server:close()**

**Description**

close server.

**Syntax**

net.server.close()

**Parameters**

nil

**Returns**

nil

**Example**

-- create a server

sv=net.createServer(net.TCP, 30)

-- close server

sv:close()

**See also**

**-** [net.createServer()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#netcreateserver)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**net.socket module**

**net.socket:connect()**

**Description**

connect to remote.

**Syntax**

connect(port, ip/domain)

**Parameters**

port: port number  
ip: ip address or domain name in string

**Returns**

nil

**See also**

**-** [net.socket:on()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#netsocketon) **-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**net.socket:send()**

**Description**

send data to remote via connection.

**Syntax**

send(string, function(sent))

**Parameters**

string: data in string which will be sent to remote  
function(sent): callback function for sending string

**Returns**

nil

**See also**

**-** [net.socket:on()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#netsocketon)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**net.socket:on()**

**Description**

register callback function for event.

**Syntax**

on(event, function cb())

**Parameters**

event: string, which can be: "connection", "reconnection", "disconnection", "receive", "sent"  
function cb(net.socket, [string]): callback function. The first param is the socket.  
If event is "receive", the second param is received data in string.

**Returns**

nil

**Example**

sk=net.createConnection(net.TCP, 0)

sk:on("receive", function(sck, c) print(c) end )

sk:connect(80,"192.168.0.66")

sk:on("connection", function(sck,c)

-- Wait for connection before sending.

sk:send("GET / HTTP/1.1\r\nHost: 192.168.0.66\r\nConnection: keep-alive\r\nAccept: \*/\*\r\n\r\n")

end)

**See also**

**-** [net.createServer()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#netcreateserver)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**net.socket:close()**

**Description**

close socket.

**Syntax**

close()

**Parameters**

nil

**Returns**

nil

**See also**

**-** [net.createServer()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#netcreateserver)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**net.socket:dns()**

**Description**

get domain ip

**Syntax**

dns(domain, function(net.socket, ip))

**Parameters**

domain: domain name.  
function (net.socket, ip): callback function. The first param is the socket, the second param is the ip address in string.

**Returns**

nil

**Example**

sk=net.createConnection(net.TCP, 0)

sk:dns("www.nodemcu.com",function(conn,ip) print(ip) end)

sk = nil

**See also**

**-** [net.createServer()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#netcreateserver)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**net.dns module**

**net.dns.setdnsserver()**

**Description**

Sets the IP of the DNS server used to resolve hostnames. Default: resolver1.opendns.com(208.67.222.222). You can specify up to 2 DNS servers.

**Syntax**

net.dns.setdnsserver(dns\_ip\_addr,dns\_index)

**Parameters**

dns\_ip\_addr: The IP address to use as DNS server  
dns\_index: (range 0~1) Which DNS server to set. Supports max. 2 DNS servers.

**Returns**

nil

**Example**

See [net.dns:getdnsserver()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#netdnsgetdnsserver)

**See also**

[net.dns:getdnsserver()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#netdnsgetdnsserver)

**net.dns.getdnsserver()**

**Description**

Gets the IP address of the DNS server used to resolve domain names.

**Syntax**

net.dns.getdnsserver(dns\_index)

**Parameters**

dns\_index: (range 0~1) Which DNS server to set. Supports max. 2 DNS servers.

**Returns**

IP address of DNS server

**Example**

print(net.dns.getdnsserver(0)) -- 208.67.222.222

print(net.dns.getdnsserver(1)) -- nil

net.dns.setdnsserver("8.8.8.8",0)

net.dns.setdnsserver("192.168.1.252",1)

print(net.dns.getdnsserver(0)) -- 8.8.8.8

print(net.dns.getdnsserver(1)) -- 192.168.1.252

**See also**

[net.dns:setdnsserver()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#netdnssetdnsserver)

**net.dns.resolve()**

**Description**

Resolve a hostname to an IP address. Doesn't require a socket like net.socket.dns().

**Syntax**

net.dns.resolve( domain, function(ip) )

**Parameters**

domain: Domain name to resolve  
function(sk,ip): Callback called when the name was resolved. Don't use sk, it's a socket used internally to resolve the hostname.

**Returns**

nil

**Example**

net.dns.resolve("www.google.com", function(sk, ip)

if (ip == nil) then print("DNS fail!") else print(ip) end

end)

**See also**

[net.socket:dns()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#netsocketdns)

**i2c module**

**CONSTANT**

i2c.SLOW, i2c.TRANSMITTER, i2c. RECEIVER. FAST（400k）is not supported for now.

**i2c.setup()**

**Description**

initialize i2c.

**Syntax**

i2c.setup(id, pinSDA, pinSCL, speed)

**Parameters**

id = 0  
pinSDA: 1~12, IO index  
pinSCL: 1~12, IO index  
speed: i2c.SLOW

**Returns**

speed: the seted speed.

**See also**

**-** [i2c.read()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#i2cread)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**i2c.start()**

**Description**

start i2c transporting.

**Syntax**

i2c.start(id)

**Parameters**

id = 0

**Returns**

nil

**See also**

**-** [i2c.read()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#i2cread)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**i2c.stop()**

**Description**

stop i2c transporting.

**Syntax**

i2c.stop(id)

**Parameters**

id = 0

**Returns**

nil

**See also**

**-** [i2c.read()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#i2cread)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**i2c.address()**

**Description**

setup i2c address and read/write mode.

**Syntax**

i2c.address(id, device\_addr, direction)

**Parameters**

id=0  
device\_addr: device address.  
direction: i2c.TRANSMITTER for writing mode , i2c. RECEIVER for reading mode

**Returns**

true: get ack false: no ack get

**See also**

**-** [i2c.read()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#i2cread)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**i2c.write()**

**Description**

write data to i2c, data can be multi numbers, string or lua table.

**Syntax**

i2c.write(id, data1, data2,...)

**Parameters**

id=0  
data: data can be numbers, string or lua table.

**Returns**

number: number of bytes wrote.

**Example**

i2c.write(0, "hello", "world")

**See also**

**-** [i2c.read()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#i2cread)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**i2c.read()**

**Description**

read data for len bytes.

**Syntax**

i2c.read(id, len)

**Parameters**

id=0  
len: data length

**Returns**

string:data received.

**Example**

id=0

sda=1

scl=2

-- initialize i2c, set pin1 as sda, set pin2 as scl

i2c.setup(id,sda,scl,i2c.SLOW)

-- user defined function: read from reg\_addr content of dev\_addr

function read\_reg(dev\_addr, reg\_addr)

i2c.start(id)

i2c.address(id, dev\_addr ,i2c.TRANSMITTER)

i2c.write(id,reg\_addr)

i2c.stop(id)

i2c.start(id)

i2c.address(id, dev\_addr,i2c.RECEIVER)

c=i2c.read(id,1)

i2c.stop(id)

return c

end

-- get content of register 0xAA of device 0x77

reg = read\_reg(0x77, 0xAA)

print(string.byte(reg))

**See also**

**-** [i2c.write()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#i2cwrite)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**adc module**

**CONSTANT**

none

**adc.read()**

**Description**

read adc value of id, esp8266 has only one 10bit adc, id=0, pin TOUT

**Syntax**

adc.read(id)

**Parameters**

id = 0

**Returns**

adc value

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**adc.readvdd33()**

**Description**

Reading vdd33 pin voltage

**Syntax**

adc.readvdd33()

**Parameters**

no parameters

**Returns**

mV

**Example**

print(adc.readvdd33())

output

3345

v = adc.readvdd33() / 1000

print(v)

v=nil

output

3.315

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**uart module**

**CONSTANT**

none

**uart.setup()**

**Description**

setup uart's baud, databits, parity, stopbits, echo.

**Syntax**

uart.setup( id, baud, databits, parity, stopbits, echo )

**Parameters**

id = 0, only 1 uart supported.  
baud = 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 74880, 115200, 230400, 460800, 921600, 1843200, 2686400.   
databits = 5, 6, 7, 8.   
parity = 0(none).  
stopbits = 1(1 stopbit), 2(2 stopbit).  
echo = 0(close echo back).

**Returns**

baud.

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**uart.on()**

**Description**

set the callback function to the uart event,  
"data" event supported, means there is data input from uart.

**Syntax**

uart.on(method, [number/end\_char], [function], [run\_input])

**Parameters**

method = "data", there is data input from uart.  
number/end\_char: if pass in a number n if n=0, will receive every char in buffer.  
if pass in a one char string "c", the callback will called when "c" is encounterd, or max n=255 received.  
function: callback function, event "data" has a callback like this: function(data) end  
run\_input: 0 or 1, 0: input from uart will not go into lua interpreter, can accept binary data.  
1: input from uart will go into lua interpreter, and run.

**Returns**

nil

**Example**

-- when 4 chars is received.

uart.on("data", 4,

function(data)

print("receive from uart:", data)

if data=="quit" then

uart.on("data")

end

end, 0)

-- when '\r' is received.

uart.on("data", "\r",

function(data)

print("receive from uart:", data)

if data=="quit\r" then

uart.on("data")

end

end, 0)

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**uart.write()**

**Description**

write string to uart.

**Syntax**

uart.write( id, string1, string2... )

**Parameters**

id = 0, only 1 uart supported.  
string1..n: string write to uart.

**Returns**

nil

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**onewire module**

**CONSTANT**

none

**ow.setup()**

**Description**

set a pin in onewire mode.

**Syntax**

ow.setup(pin)

**Parameters**

pin: 1~12, IO index

**Returns**

nil

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**ow.reset()**

**Description**

Perform a 1-Wire reset cycle.

**Syntax**

ow.reset(pin)

**Parameters**

pin: 1~12, IO index

**Returns**

number: Returns 1 if a device responds with a presence pulse. Returns 0 if there is no device or the bus is shorted or otherwise held low for more than 250uS

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**ow.skip()**

**Description**

Issue a 1-Wire rom skip command, to address all on bus.

**Syntax**

ow.skip(pin)

**Parameters**

pin: 1~12, IO index

**Returns**

nil

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**ow.select()**

**Description**

Issue a 1-Wire rom select command, make sure you do the ow.reset(pin) first.

**Syntax**

ow.select(pin, rom)

**Parameters**

pin: 1~12, IO index  
rom: string value, len 8, rom code of the salve device

**Returns**

nil

**Example**

-- 18b20 Example

pin = 9

ow.setup(pin)

count = 0

repeat

count = count + 1

addr = ow.reset\_search(pin)

addr = ow.search(pin)

tmr.wdclr()

until((addr ~= nil) or (count > 100))

if (addr == nil) then

print("No more addresses.")

else

print(addr:byte(1,8))

crc = ow.crc8(string.sub(addr,1,7))

if (crc == addr:byte(8)) then

if ((addr:byte(1) == 0x10) or (addr:byte(1) == 0x28)) then

print("Device is a DS18S20 family device.")

repeat

ow.reset(pin)

ow.select(pin, addr)

ow.write(pin, 0x44, 1)

tmr.delay(1000000)

present = ow.reset(pin)

ow.select(pin, addr)

ow.write(pin,0xBE,1)

print("P="..present)

data = nil

data = string.char(ow.read(pin))

for i = 1, 8 do

data = data .. string.char(ow.read(pin))

end

print(data:byte(1,9))

crc = ow.crc8(string.sub(data,1,8))

print("CRC="..crc)

if (crc == data:byte(9)) then

t = (data:byte(1) + data:byte(2) \* 256) \* 625

t1 = t / 10000

t2 = t % 10000

print("Temperature="..t1.."."..t2.."Centigrade")

end

tmr.wdclr()

until false

else

print("Device family is not recognized.")

end

else

print("CRC is not valid!")

end

end

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**ow.write()**

**Description**

Write a byte. If 'power' is 1 then the wire is held high at the end for parasitically powered devices. You are responsible for eventually depowering it by calling depower() or doing another read or write.

**Syntax**

ow.write(pin, v, power)

**Parameters**

pin: 1~12, IO index   
v: byte to be written to salve device   
power: 1 for wire being held high for parasitically powered devices.

**Returns**

nil

**Example**

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**ow.write\_bytes()**

**Description**

Write multi bytes. If 'power' is 1 then the wire is held high at the end for parasitically powered devices. You are responsible for eventually depowering it by calling depower() or doing another read or write.

**Syntax**

ow.write\_bytes(pin, buf, power)

**Parameters**

pin: 1~12, IO index   
buf: string to be written to salve device   
power: 1 for wire being held high for parasitically powered devices.

**Returns**

nil

**Example**

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**ow.read()**

**Description**

read a byte.

**Syntax**

ow.read(pin)

**Parameters**

pin: 1~12, IO index

**Returns**

byte read from slave device.

**Example**

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**ow.read\_bytes()**

**Description**

read multi bytes.

**Syntax**

ow.read\_bytes(pin, size)

**Parameters**

pin: 1~12, IO index   
size: number of bytes to be read from slave device.

**Returns**

string: bytes read from slave device.

**Example**

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**ow.depower()**

**Description**

Stop forcing power onto the bus. You only need to do this if you used the 'power' flag to ow.write() or used a ow.write\_bytes() and aren't about to do another read or write.

**Syntax**

ow.depower(pin)

**Parameters**

pin: 1~12, IO index

**Example**

**Returns**

nil

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**ow.reset\_search()**

**Description**

Clear the search state so that it will start from the beginning again.

**Syntax**

ow.reset\_search(pin)

**Parameters**

pin: 1~12, IO index

**Returns**

nil

**Example**

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**ow.target\_search()**

**Description**

Setup the search to find the device type 'family\_code' on the next call to ow.search() if it is present.

**Syntax**

ow.target\_search(pin, family\_code)

**Parameters**

pin: 1~12, IO index   
family\_code: byte for family code.

**Returns**

nil

**Example**

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**ow.search()**

**Description**

Look for the next device.

**Syntax**

ow.search(pin)

**Parameters**

pin: 1~12, IO index

**Returns**

if succeed return a string length of 8, which contain the rom code of slave device.   
if failed in searching next device return nil.

**Example**

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**ow.crc8()**

**Description**

Compute a Dallas Semiconductor 8 bit CRC, these are used in the ROM and scratchpad registers.

**Syntax**

ow.crc8(buf)

**Parameters**

buf: string value, data to be calculated check sum in string.

**Returns**

crc result in byte.

**Example**

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**ow.check\_crc16()**

**Description**

Compute the 1-Wire CRC16 and compare it against the received CRC.

**Syntax**

ow.check\_crc16(buf, inverted\_crc0, inverted\_crc1, crc)

**Parameters**

buf: string value, data to be calculated check sum in string.   
inverted\_crc0: LSB of received CRC.   
inverted\_crc1: MSB of received CRC.   
crc: crc starting value (optional)

**Returns**

bool: true, if the CRC matches; false for dismatches.

**Example**

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**ow.crc16()**

**Description**

Compute a Dallas Semiconductor 16 bit CRC. This is required to check the integrity of data received from many 1-Wire devices. Note that the CRC computed here is **not** what you'll get from the 1-Wire network, for two reasons:  
1) The CRC is transmitted bitwise inverted.  
2) Depending on the endian-ness of your processor, the binary representation of the two-byte return value may have a different byte order than the two bytes you get from 1-Wire.

**Syntax**

ow.crc16(buf, crc)

**Parameters**

buf: string value, data to be calculated check sum in string.   
crc: crc starting value (optional)

**Returns**

return The CRC16, as defined by Dallas Semiconductor.

**Example**

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**bit module**

**CONSTANT**

none

**bit.bnot()**

**Description**

Bitwise negation, equivalent to ~value in C.

**Syntax**

bit.bnot(value)

**Parameters**

value: the number to negate.

**Returns**

number: the bitwise negated value of the number.

**Example**

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**bit.band()**

**Description**

Bitwise AND, equivalent to val1 & val2 & ... & valn in C.

**Syntax**

bit.band(val1, val2, ... valn)

**Parameters**

val1: first AND argument.  
val2: second AND argument.  
valn: nth AND argument.

**Returns**

number: the bitwise AND of all the arguments.

**Example**

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**bit.bor()**

**Description**

Bitwise OR, equivalent to val1 | val2 | ... | valn in C.

**Syntax**

bit.bor(val1, val2, ... valn)

**Parameters**

val1: first OR argument.  
val2: second OR argument.  
valn: nth OR argument.

**Returns**

number: the bitwise OR of all the arguments.

**Example**

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**bit.bxor()**

**Description**

Bitwise XOR, equivalent to val1 ^ val2 ^ ... ^ valn in C.

**Syntax**

bit.bxor(val1, val2, ... valn)

**Parameters**

val1: first XOR argument.  
val2: second XOR argument.  
valn: nth XOR argument.

**Returns**

number: the bitwise XOR of all the arguments.

**Example**

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**bit.lshift()**

**Description**

Left-shift a number, equivalent to value << shift in C.

**Syntax**

bit.lshift(value, shift)

**Parameters**

value: the value to shift.  
shift: positions to shift.

**Returns**

number: the number shifted left

**Example**

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**bit.rshift()**

**Description**

Logical right shift a number, equivalent to ( unsigned )value >> shift in C.

**Syntax**

bit.rshift(value, shift)

**Parameters**

value: the value to shift.  
shift: positions to shift.

**Returns**

number: the number shifted right (logically).

**Example**

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**bit.arshift()**

**Description**

Arithmetic right shift a number equivalent to value >> shift in C.

**Syntax**

bit.arshift(value, shift)

**Parameters**

value: the value to shift.  
shift: positions to shift.

**Returns**

number: the number shifted right (arithmetically).

**Example**

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**bit.bit()**

**Description**

Generate a number with a 1 bit (used for mask generation). Equivalent to 1 << position in C.

**Syntax**

bit.bit(position)

**Parameters**

position: position of the bit that will be set to 1.

**Returns**

number: a number with only one 1 bit at position (the rest are set to 0).

**Example**

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**bit.set()**

**Description**

Set bits in a number.

**Syntax**

bit.set(value, pos1, pos2, ..., posn)

**Parameters**

value: the base number.  
pos1: position of the first bit to set.  
pos2: position of the second bit to set.  
posn: position of the nth bit to set.

**Returns**

number: the number with the bit(s) set in the given position(s).

**Example**

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**bit.clear()**

**Description**

Clear bits in a number.

**Syntax**

bit.clear(value, pos1, pos2, ..., posn)

**Parameters**

value: the base number.  
pos1: position of the first bit to clear.  
pos2: position of the second bit to clear.  
posn: position of thet nth bit to clear.

**Returns**

number: the number with the bit(s) cleared in the given position(s).

**Example**

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**bit.isset()**

**Description**

Test if a given bit is set.

**Syntax**

bit.isset(value, position)

**Parameters**

value: the value to test.  
position: bit position to test.

**Returns**

boolean: true if the bit at the given position is 1, false otherwise.

**Example**

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**bit.isclear()**

**Description**

Test if a given bit is cleared.

**Syntax**

bit.isclear(value, position)

**Parameters**

value: the value to test.  
position: bit position to test.

**Returns**

boolean: true if the bit at the given position is 0, false othewise.

**Example**

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**spi module**

**CONSTANT**

MASTER, SLAVE, CPHA\_LOW, CPHA\_HIGH, CPOL\_LOW, CPOL\_HIGH

DATABITS\_8, DATABITS\_16 -- Deprecated

All transactions for sending and receiving are most-significant-bit first and least-significant last.

**spi.setup()**

**Description**

setup spi configuration.

**Syntax**

spi.setup( id, mode, cpol, cpha, databits, clock\_div )

**Parameters**

id: spi id number.  
mode: MASTER or SLAVE(not supported yet).  
cpol: CPOL\_LOW or CPOL\_HIGH, clock polarity.  
cpha: CPHA\_HIGH or CPHA\_LOW, clock phase.  
databits: DATABITS\_8, 1 - 32 (only in dev branch yet)  
clock\_div: spi clock divider, f(SPI) = f(CPU) / clock\_div (only in dev branch yet).

**Returns**

number: 1.

**Example**

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**spi.send()**

**Description**

send data to spi.

**Syntax**

wrote = spi.send( id, data1, [data2], ..., [datan] )

**Parameters**

id: spi id number.  
data: data can be either a string, a table or an integer number  
Each data item is considered with databits number of bits (only in dev branch yet).

**Returns**

number: bytes writen count.

**Example**

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**spi.recv()**

**Description**

recv data from spi.

**Syntax**

read = spi.recv( id, size )

**Parameters**

id: spi id number.  
size: data size want to read.

**Returns**

string: string bytes read from spi.

**Example**

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**spi.set\_mosi()**

**Description -- (Only in dev branch yet)**

Insert data items into MOSI buffer for spi.transaction().

**Syntax**

spi.set\_mosi( id, offset, bitlen, data1, [data2], ..., [datan] )

**Parameters**

id: spi id number.  
offset: bit offset into MOSI buffer for inserting data1 and subsequent items.  
bitlen: bit length of data1, data2, ...  
data: data items, bitlen number of bits considered for transaction.

**Returns**

nil

**Example**

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**spi.get\_miso()**

**Description -- (Only in dev branch yet)**

Extract data items from MISO buffer after spi.transaction().

**Syntax**

data1, [data2], ..., [datan] = spi.get\_miso( id, offset, bitlen, num )

**Parameters**

id: spi id number.  
offset: bit offset into MISO buffer for first data item.  
bitlen: bit length of data items.  
num: number of data items to retrieve.

**Returns**

num data items.

**Example**

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**spi.transaction()**

**Description -- (Only in dev branch yet)**

Start an SPI transaction, consisting of up to 5 phases:

1. Command
2. Address
3. MOSI
4. Dummy
5. MISO

**Syntax**

spi.transaction( id, cmd\_bitlen, cmd\_data, addr\_bitlen, addr\_data, mosi\_bitlen, dummy\_bitlen, miso\_bitlen )

**Parameters**

cmd\_bitlen: bit length of the command phase (0 - 16).  
cmd\_data: data for command phase.  
addr\_bitlen: bit length for address phase (0 - 32).  
addr\_data: data for command phase.  
mosi\_bitlen: bit length of the MOSI phase (0 - 512).  
dummy\_bitlen: bit length of the dummy phase (0 - 256).  
miso\_bitlen: bit length of the MISO phase (0 - 512).

**Returns**

nil

**Example**

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**mqtt module**

**CONSTANT**

**mqtt.Client()**

**Description**

Create a MQTT client. The client adheres to version 3.1.1 of the MQTT protocol, make sure that your broker supports and is correctly configured for version 3.1.1 of the MQTT protocol. The client is incompatible with brokers running version 3.1 of the MQTT protocol.

**Syntax**

mqtt.Client(clientid, keepalive, user, pass)

**Parameters**

clientid: the client id.  
keepalive: keepalive second, a number.  
user: user name, a string.  
pass: user password, a string.

**Returns**

mqtt client.

**Example**

-- init mqtt client with keepalive timer 120sec

m = mqtt.Client("clientid", 120, "user", "password")

-- setup Last Will and Testament (optional)

-- Broker will publish a message with qos = 0, retain = 0, data = "offline"

-- to topic "/lwt" if client don't send keepalive packet

m:lwt("/lwt", "offline", 0, 0)

m:on("connect", function(con) print ("connected") end)

m:on("offline", function(con) print ("offline") end)

-- on publish message receive event

m:on("message", function(conn, topic, data)

print(topic .. ":" )

if data ~= nil then

print(data)

end

end)

-- for secure: m:connect("192.168.11.118", 1880, 1)

m:connect("192.168.11.118", 1880, 0, function(conn) print("connected") end)

-- subscribe topic with qos = 0

m:subscribe("/topic",0, function(conn) print("subscribe success") end)

-- publish a message with data = hello, QoS = 0, retain = 0

m:publish("/topic","hello",0,0, function(conn) print("sent") end)

m:close();

-- you can call m:connect again

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**mqtt client module**

**mqtt.client:lwt()**

**Description**

setup Last Will and Testament (optional)  
Broker will publish a message with qos = 0, retain = 0, data = "offline"   
to topic "/lwt" if client don't send keepalive packet.

**Syntax**

mqtt:lwt(topic, message, qos, retain)

**Parameters**

topic: the topic to publish to, String.  
message: the message to publish, Buffer or String.  
qos: qos level, default 0.  
retain: retain flag, default 0.

**Returns**

nil.

**Example**

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**mqtt.client:connect()**

**Description**

Connects to the broker specified by the given host, port, and secure options

**Syntax**

mqtt:connect( host, port, secure, function(client) )

**Parameters**

host: host domain or ip, string.  
port: number, broker port.  
secure: 0 or 1, default 0.  
function(client): when connected, call this function.

**Returns**

nil.

**Example**

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**mqtt.client:close()**

**Description**

close connection to the broker.

**Syntax**

mqtt:close()

**Parameters**

nil

**Returns**

nil.

**Example**

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**mqtt.client:publish()**

**Description**

Publish a message

**Syntax**

mqtt:publish( topic, payload, qos, retain, function(client) )

**Parameters**

topic: the topic to publish to, string  
message: the message to publish, string  
qos: qos level, default 0  
retain: retain flag, default 0  
function(client): callback fired when PUBACK received.

**Returns**

nil.

**Example**

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**mqtt.client:subscribe()**

**Description**

Subscribe to a topic or topics

**Syntax**

mqtt:subscribe(topic, qos, function(client, topic, message))

**Parameters**

topic: a string topic to subscribe to  
qos: qos subscription level, default 0  
function(client, topic, message): callback fired when message received.

**Returns**

nil.

**Example**

**See also**

**-**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**mqtt.client:on()**

**Description**

register callback function to event.

**Syntax**

mqtt:on(event, function(client, [topic], [message]))

**Parameters**

event: string, which can be: "connect", "message", "offline"  
function cb(client, [topic], [message]): callback function. The first param is the client.  
If event is "message", the 2nd and 3rd param are received topic and message in string.

**Returns**

nil.

**Example**

**See also**

**-**

**WS2801 Module**

**ws2801.init()**

**Description**

Initializes the module and sets the pin configuration.

**Syntax**

ws2801.init(pin\_clk, pin\_data)

**Parameters**

pin\_clk = Pin for the clock. Supported are GPIO 0, 2, 4, 5  
pin\_data = Pin for the data. Supported are GPIO 0, 2, 4, 5

**Return**

nil

**ws2801.write()**

**Description**

Sends a string of RGB Data in 24bits to WS2801. Don't forget to call ws2801.init() before.

**Syntax**

ws2801.write(string.char(R1,G1,B1(,R2,G2,B2...)) )

**Parameters**

R1 = The first pixel's red channel's value（0-255）  
G1 = The first pixel's green channel's value（0-255）  
B1 = The first pixel's blue channel's value（0-255）  
... You can connect a lot of WS2801...  
R2,G2,B2 is the next WS2801's Red, Green and Blue channel's value

**Return**

nil

**WS2812 Module**

**ws2812.writergb()**

**Description**

Send the RGB Data in 8bits to WS2812

**Syntax**

ws2812.writergb(pin, string.char(R1,G1,B1(,R2,G2,B2...)) )

**Parameters**

pin = Supported all the PINs(0,1,2...)  
R1 = The first WS2812 though the line's Red Channel's Parameters（0-255）  
G1 = The first WS2812 though the line's Green Channel's Parameters（0-255）   
B1 = The first WS2812 though the line's Blue Channel's Parameters（0-255）   
... You can connect a lot of WS2812...  
R2,G2,B2 is the next WS2812's Red, Green and Blue Channel's Parameters

**Return**

nil

**Example**

r = 255

g = 0

b = 0

rgb = string.char(r, g, b)

ws2812.writergb(2, rgb) -- turn first WS2812B, connected to pin 2, red

**cjson.encode()**

**Description**

Encode table to json string

**Syntax**

cjson.encode(table)

**Parameters**

table = data to encode

**Return**

json string

**Example**

print(cjson.encode({key="value"}))

**cjson.decode()**

**Description**

Decode json string to table

**Syntax**

cjson.decode(s)

**Parameters**

s = string to decode

**Return**

Lua table

**Example**

t= cjson.decode("{\"key\":\"value\"}")

for k,v in pairs(t) do print(k,v) end

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**Crypto Module**

**crypto.hash()**

**Description**

Compute a cryptographic hash of a Lua string.

**Syntax**

digest = crypto.hash(algo, str)

**Parameters**

algo = hash algorithm to use, case insensitive string   
str = data to calculate the hash for

Supported hash algorithms are:

* MD2 (not available by default, has to be explicitly enabled in user\_config.h)
* MD5
* SHA1
* SHA256, SHA384, SHA512 (unless disabled in user\_config.h)

**Return**

A binary string containing the message digest. To obtain the textual version (ASCII hex characters), please use [crypto.toHex()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#crypto_tohex).

**Example**

print(crypto.toHex(crypto.hash("sha1","abc")))

**crypto.hmac()**

**Description**

Compute a HMAC (Hashed Message Authentication Code) signature for a Lua string.

**Syntax**

signature = crypto.hmac(algo, str, key)

**Parameters**

algo = hash algorithm to use, case insensitive string   
str = data to calculate the hash for  
key = key to use for signing, may be a binary string

Supported hash algorithms are:

* MD2 (not available by default, has to be explicitly enabled in user\_config.h)
* MD5
* SHA1
* SHA256, SHA384, SHA512 (unless disabled in user\_config.h)

**Return**

A binary string containing the HMAC signature. Use [crypto.toHex()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#crypto_tohex) to obtain the textual version.

**Example**

print(crypto.toHex(crypto.hmac("sha1","abc","mysecret")))

**crypto.mask()**

**Description**

Applies an XOR mask to a Lua string.

**Syntax**

masked = crypto.mask (message, mask)

**Parameters**

message = message to mask  
mask = the mask to apply, repeated if shorter than the message

**Return**

The masked message, as a binary string. Use [crypto.toHex()](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#crypto_tohex) to get a textual representation of it.

**Example**

print(crypto.toHex(crypto.mask("some message to obscure","X0Y7")))

**crypto.toHex()**

**Description**

Provides an ASCII hex representation of a (binary) Lua string. Each byte in the input string is represented as two hex characters in the output.

**Syntax**

hexstr = crypto.toHex(binary)

**Parameters**

binary = input string to get hex representation for

**Return**

An ASCII hex string.

**Example**

print(crypto.toHex(crypto.hash("sha1","abc")))

**crypto.toBase64()**

**Description**

Provides a Base64 representation of a (binary) Lua string.

**Syntax**

b64 = crypto.toBase64(binary)

**Parameters**

binary = input string to Base64 encode

**Return**

A Base64 encoded string.

**Example**

print(crypto.toBase64(crypto.hash("sha1","abc")))

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**u8g module**

**CONSTANT**

u8g.DRAW\_UPPER\_RIGHT, u8g.DRAW\_UPPER\_LEFT, u8g.DRAW\_LOWER\_RIGHT, u8g.DRAW\_LOWER\_LEFT, u8g.DRAW\_ALL, u8g.MODE\_BW, u8g.MODE\_GRAY2BIT

u8g.font\_6x10, ...

**u8g.\*\_i2c()**

**Description**

Initialize a display via I2C.

**Syntax Example**

u8g.ssd1306\_128x64\_i2c(sla)

**Parameters**

sla: I2C slave address.

**Returns**

u8g display.

**Example**

sda = 5

scl = 6

i2c.setup(0, sda, scl, i2c.SLOW)

sla = 0x3c

disp = u8g.ssd1306\_128x64\_i2c(sla)

**See also**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**u8g.\*\_hw\_spi()**

**Description**

Initialize a display via Hardware SPI.

**Syntax Example**

u8g.ssd1306\_128x64\_spi(cs, dc, res)

**Parameters**

cs: GPIO pin for /CS.  
dc: GPIO pin for DC.  
res: GPIO pin for /RES (optional).

**Returns**

u8g display.

**Example**

spi.setup(1, spi.MASTER, spi.CPOL\_LOW, spi.CPHA\_LOW, spi.DATABITS\_8, 0)

cs = 8 -- GPIO15, pull-down 10k to GND

dc = 4 -- GPIO2

res = 0 -- GPIO16, RES is optional YMMV

disp = u8g.ssd1306\_128x64\_spi(cs, dc, res)

**See also**

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**u8g display sub-module**

The Lua bindings for this library closely follow u8glib's object oriented C++ API. Visit the[u8glib wiki](https://github.com/olikraus/u8glib/wiki) for technical details.

**u8g.disp:drawBitmap()**

**Description**

Draw a bitmap at the specified x/y position (upper left corner of the bitmap). Parts of the bitmap may be outside the display boundaries. The bitmap is specified by the array bitmap. A cleared bit means: Do not draw a pixel. A set bit inside the array means: Write pixel with the current color index. For a monochrome display, the color index 0 will usually clear a pixel and the color index 1 will set a pixel.

**Syntax**

disp.drawBitmap(x, y, cnt, h, bitmap)

**Parameters**

x: X-position (left position of the bitmap).  
y: Y-position (upper position of the bitmap).  
cnt: Number of bytes of the bitmap in horizontal direction. The width of the bitmap is cnt\*8.  
h: Height of the bitmap.  
bitmap: Bitmap data supplied as string.

**Returns**

nil

**Example**

lua\_examples/u8glib/u8g\_bitmaps.lua

**See also**

[u8glib drawBitmap()](https://github.com/olikraus/u8glib/wiki/userreference#drawbitmap)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**u8g.disp:drawXBM()**

**Description**

Draw a XBM Bitmap. Position (x,y) is the upper left corner of the bitmap. XBM contains monochrome, 1-bit bitmaps. This procedure only draws pixel values 1. The current color index is used for drawing (see setColorIndex). Pixel with value 0 are not drawn (transparent).

Bitmaps and XBMs are supplied as strings to drawBitmap() and drawXBM(). This off-loads all data handling from the u8g module to generic methods for binary files. In contrast to the source code based inclusion of XBMs into u8glib, it's required to provide precompiled binary files. This can be performed online with [Online-Utility's Image Converter](http://www.online-utility.org/image_converter.jsp): Convert from XBM to MONO format and upload the binary result with [nodemcu-uploader.py](https://github.com/kmpm/nodemcu-uploader) or[ESPlorer](http://esp8266.ru/esplorer/).

**Syntax**

disp.drawXBM(x, y, w, h, bitmap)

**Parameters**

x: X-position (left position of the bitmap).  
y: Y-position (upper position of the bitmap).  
w: Width of the bitmap.  
h: Height of the bitmap.  
bitmap: XBM data supplied as string.

**Returns**

nil

**Example**

lua\_examples/u8glib/u8g\_bitmaps.lua

**See also**

[u8glib drawXBM()](https://github.com/olikraus/u8glib/wiki/userreference#drawxbm)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**u8g.disp:setFont()**

**Description**

u8glib comes with a wide range of fonts for small displays. Since they need to be compiled into the firmware image, you'd need to include them in app/include/u8g\_config.h and recompile. Simply add the desired fonts to the font table:

#define U8G\_FONT\_TABLE \

U8G\_FONT\_TABLE\_ENTRY(font\_6x10) \

U8G\_FONT\_TABLE\_ENTRY(font\_chikita)

They'll be available as u8g.<font\_name> in Lua.

**Syntax**

disp.setFont(font)

**Parameters**

font: Constant to indentify pre-compiled font.

**Returns**

nil

**Example**

disp:setFont(u8g.font\_6x10)

**See also**

[u8glib setFont()](https://github.com/olikraus/u8glib/wiki/userreference#setfont)

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)

**dht module**

**CONSTANT**

dht.OK, dht.ERROR\_CHECKSUM, dht.ERROR\_TIMEOUT

* dht.OK is 0, dht.ERROR\_CHECKSUM is 1, dht.ERROR\_TIMEOUT is 2

**dht.read()**

**Description**

Read all kinds of dht sensors, including dht11, 21, 22, 33, 44 humidity temperature combo sensor.

**Syntax**

dht.read(pin)

**Parameters**

pin: pin number of dht sensor (can't be 0), type is number

**Return**

integer of status, number of temperature, humidity, decimial of temperature, decimial of humidity.  
status is integer, temperature, humidity, decimial of temperature, decimial of humidity is number. \*Note: If using float firmware, the temperature, humidity already with decimial.

**Example**

pin = 5

status,temp,humi,temp\_decimial,humi\_decimial = dht.read(pin)

if( status == dht.OK ) then

-- Integer firmware using this example

print(

string.format(

"DHT Temperature:%d.%03d;Humidity:%d.%03d\r\n",

math.floor(temp),

temp\_decimial,

math.floor(humi),

humi\_decimial

)

)

-- Float firmware using this example

print("DHT Temperature:"..temp..";".."Humidity:"..humi)

elseif( status == dht.ERROR\_CHECKSUM ) then

print( "DHT Checksum error." );

elseif( status == dht.ERROR\_TIMEOUT ) then

print( "DHT Time out." );

end

**dht.read11()**

**Description**

Read dht11 humidity temperature combo sensor.

**Syntax**

dht.read11(pin)

**Parameters**

pin: pin number of dht sensor (can't be 0), type is number

**Return**

integer of status, number of temperature, humidity, decimial of temperature, decimial of humidity.  
status is integer, temperature, humidity, decimial of temperature, decimial of humidity is number. \*Note: If using float firmware, the temperature, humidity already with decimial.

**Example**

pin = 5

status,temp,humi,temp\_decimial,humi\_decimial = dht.read11(pin)

if( status == dht.OK ) then

-- Integer firmware using this example

print(

string.format(

"DHT Temperature:%d.%03d;Humidity:%d.%03d\r\n",

math.floor(temp),

temp\_decimial,

math.floor(humi),

humi\_decimial

)

)

-- Float firmware using this example

print("DHT Temperature:"..temp..";".."Humidity:"..humi)

elseif( status == dht.ERROR\_CHECKSUM ) then

print( "DHT Checksum error." );

elseif( status == dht.ERROR\_TIMEOUT ) then

print( "DHT Time out." );

end

**dht.readxx()**

**Description**

Read all kinds of dht sensors, except dht11.

**Syntax**

dht.readxx(pin)

**Parameters**

pin: pin number of dht sensor (can't be 0), type is number

**Return**

integer of status, number of temperature, humidity, decimial of temperature, decimial of humidity.  
status is integer, temperature, humidity, decimial of temperature, decimial of humidity is number. \*Note: If using float firmware, the temperature, humidity already with decimial.

**Example**

pin = 5

status,temp,humi,temp\_decimial,humi\_decimial = dht.readxx(pin)

if( status == dht.OK ) then

-- Integer firmware using this example

print(

string.format(

"DHT Temperature:%d.%03d;Humidity:%d.%03d\r\n",

math.floor(temp),

temp\_decimial,

math.floor(humi),

humi\_decimial

)

)

-- Float firmware using this example

print("DHT Temperature:"..temp..";".."Humidity:"..humi)

elseif( status == dht.ERROR\_CHECKSUM ) then

print( "DHT Checksum error." );

elseif( status == dht.ERROR\_TIMEOUT ) then

print( "DHT Time out." );

end

**-** [Back to Index](https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en#index)