



深圳四博智联科技有限公司 Shenzhen Four Primus Union Technology Co., Ltd  
<http://www.doit.am> <http://www.smartarduino.com> Tel: 186 7666 2425



深圳四博智联科技有限公司

## WiFiMCU Reference Book



August 2015

# Table of Contents

---

1. [Introduction](#)
2. [Lua Basic Modules](#)
3. [MCU Module](#)
4. [GPIO Module](#)
5. [TIMER Module](#)
6. [WiFi Module](#)
7. [Net Module](#)
8. [File Module](#)
9. [PWM Module](#)
10. [ADC Module](#)
11. [UART Module](#)
12. [SPI Module](#)
13. [I2C Module](#)
14. [Bit Module](#)
15. [OW Module](#)
16. [MQTT Module](#)

# Introduction

---

This reference book presents the lua function definitions of WiFiMCU.

## Lua Basic Modules

---

The Lua interpreter in Wi-Fi MCU is based on Lua 5.1.4. The following modules are supported:

lua	-
luaopen_base	Supported
luaopen_package	Supported
luaopen_string	Supported
luaopen_table	Supported
luaopen_math	Supported

'io' and 'debug' modules are not supported. The functions description in supported modules can be found at:  
<http://www.lua.org/manual/5.1/>

## Function List

Function	Definition
mcu.ver()	Get the WiFiMCU firmware version
mcu.info()	Get the mxchipWNet library version, MAC address, WLAN driver version
mcu.reboot()	Reboot WiFiMCU
mcu.mem()	Get the memory status
mcu.chipid()	Get the stm32 chip ID (96 bits)
mcu.bootreason()	Get the WiFiMCU boot reason that cause its startup

### mcu.ver()

#### Description

Get the WiFiMCU firmware version.

#### Syntax

```
nv,bd=mcu.ver()
```

#### Parameters

nil

#### Returns

nv: string type, WiFiMCU firmware version

bd: string type, build date of the firmware

#### Examples

```
-nv,bd=mcu.ver()
```

```
-print(nv,bd)
```

```
-WiFiMCU 0.9.3 build 20150818
```

### mcu.info()

#### Description

Get the mxchipWNet library version, MAC address, WLAN driver version.

#### Syntax

```
libv,mac,drv=mcu.info()
```

#### Parameters

nil

### Returns

libv: mxchipWNet library version

mac: MAC address of the module

drv: WLAN driver version

### Examples

```
-libv,mac,drv=mcu.info()
```

```
-print(libv,mac,drv)
```

```
-31620002.031 C8:93:46:50:21:4C wI0: Dec 29 2014 14:07:06 version 5.90.230.10 FWID 01-9bdaad4d
```

## mcu.reboot()

---

### Description

Reboot WiFiMCU immediately.

### Syntax

```
mcu.reboot()
```

### Parameters

nil

### Returns

nil

### Examples

```
-mcu.reboot()
```

## mcu.mem()

---

### Description

Get the memory status.

### Syntax

```
fm,tas,mtas,fc=mcu.mem()
```

### Parameters

nil

### Returns

MCU Module

fm: Total free space

tas: Total allocated space

mtas: Maximum total allocated space

fc: Number of free chunks

### Examples

```
-fm,tas,mtas,fc=mcu.mem()
```

```
-print(fm,tas,mtas,fc)
```

```
-35600 50416 86016 25
```

## mcu.chipid()

---

### Description

Get the stm32 chip ID (96 bits).

### Syntax

```
chipid= mcu.chipid()
```

### Parameters

nil

### Returns

chipid: the stm32 chip product ID

### Examples

```
-chipid= mcu.chipid()
```

```
-print(chipid)
```

```
-0200C000FDFFFAE005DFF000
```

## mcu.bootreason()

---

### Description

Get the WiFiMCU boot reason that cause its startup.

### Syntax

```
bootreason= mcu. bootreason()
```

### Parameters

nil

**Returns**

bootreason: The boot reason should be one the followings:

"NONE": Fail to get the boot reason

"SOFT\_RST": Software reset

"PWRON\_RST": Power on reset

"EXPIN\_RST": Pin reset

"WDG\_RST": Independent Watchdog reset

"WWDG\_RST": Window Watchdog reset

"LOWPWR\_RST": Low Power reset

"BOR\_RST" : POR/PDR or BOR reset

**Examples**

```
-mcu.bootreason()
```

SOFT\_RST



## Function List

Function	Definition
gpio.mode()	Define the GPIO Pin mode, set the pin to input output or interrupt mode
gpio.read()	Read the pin value
gpio.write()	Set the pin value
gpio.toggle()	Toggle the pin's output value

## Constant

gpio	Function
gpio.INPUT	Input with an internal pull-up resistor
gpio.INPUT_PULL_UP	Input with an internal pull-up resistor
gpio.INPUT_PULL_DOWN	Input with an internal pull-down resistor
gpio.INPUT_INPUT_HIGH_IMPEDANCE_DOWN	Input high impedance down
gpio.OUTPUT	Output actively driven high and actively driven low
gpio.OUTPUT_PUSH_PULL	Output actively driven high and actively driven low
gpio.OUTPUT_OPEN_DRAIN_NO_PULL	Output actively driven low but is high-impedance when set high
gpio.OUTPUT_OPEN_DRAIN_PULL_UP	Output actively driven low and is pulled high with an internal resistor when set high
gpio.INT	Interrupt
gpio.HIGH	High voltage level
gpio.LOW	Low voltage level

## GPIO Pin Table

Wi-FiMCU Index	Alternative Function	Discription
D0	GPIO/BOOT	Wi-FiMCU would enter into Bootloader Mode, if D0 goes to LOW
D1	GPIO/PWM/ADC	-
D2	GPIO	-
D3	GPIO/PWM	-
D4	GPIO	-
D5	GPIO	SWD Flash Programming Pin: swclk
D6	GPIO	SWD Flash Programming Pin: swdio
D7	GPIO	-
D8	GPIO/PWM	Uart1 rx pin: RX1
D9	GPIO/PWM	Uart1 tx pin: TX1
D10	GPIO/PWM	I2C interface: SCL

D11	GPIO/PWM	I2C interface: SDA
D12	GPIO/PWM	-
D13	GPIO/PWM/ADC	-
D14	GPIO/PWM	-
D15	GPIO/PWM/ADC	-
D16	GPIO/PWM/ADC	-
D17	GPIO/ADC	A LED is connected on Wi-FiMCU board

## gpio.mode

### Description

Define the GPIO Pin mode, set the pin to input output or interrupt mode.

### Syntax

```
gpio.mode(pin, mode)
```

```
gpio.mode(pin, gpio.INT, trigMode, func_cb)
```

### Parameters

pin: gpio ID, 0~17

mode: Should be one of the followings:

gpio.INPUT

gpio.INPUT\_PULL\_UP

gpio.INPUT\_PULL\_DOWN

gpio.INPUT\_INPUT\_HIGH\_IMPEDANCE\_DOWN

gpio.OUTPUT

gpio.OUTPUT\_PUSH\_PULL

gpio.OUTPUT\_OPEN\_DRAIN\_NO\_PULL

gpio.OUTPUT\_OPEN\_DRAIN\_PULL\_UP

gpio.INT

trigMode: if mode is gpio.INT, trigMode should be:

'rising': Interrupt triggered at input signal's rising edge

'falling': Interrupt triggered at input signal's falling edge

'both': Interrupt triggered at both rising and falling edge

func\_cb: if mode is gpio.INT, the interrupt call back function

**Note:** It's recommend that DO NOT do too much time consumption operations in the func\_cb.

### Returns

nil

### Examples

```
-gpio.mode(0, gpio.OUTPUT)
```

```
-gpio.write(0, gpio.HIGH)
```

```
-gpio.mode(1,gpio.INPUT)
```

```
-print(gpio.read(1))
```

```
-0
```

## gpio.read()

---

### Description

Read the pin value.

### Syntax

```
value=gpio.read(pin)
```

### Parameters

pin: gpio ID, 0~17

### Returns

value: 0 - low, 1 - high

### Examples

```
-gpio.mode(0, gpio.INPUT)
```

```
-print(gpio.read(0))
```

```
-0
```

## gpio.write()

---

### Description

Set the pin value.

### Syntax

```
gpio.write(pin, value)
```

### Parameters

pin: gpio ID, 0~17

value: 0 or 1 or gpio.HIGH or gpio.LOW

### Returns

nil

### Examples

```
-gpio.mode(0, gpio.OUTPUT)
```

```
-gpio.write(0,gpio.HIGH)
```

```
-gpio.write(0,0)
```

## gpio.toggle()

---

### Description

Toggle the pin's output value

### Syntax

```
gpio.toggle(pin)
```

### Parameters

pin: gpio ID, 0~17

### Returns

nil

### Examples

```
-gpio.mode(17, gpio.OUTPUT)
```

```
-gpio.toggle(17)
```

## Function List

Function	Definition
tmr.start()	Start a timer with call back function
tmr.stop()	Stop a timer
tmr.stopall()	Stop all the timer
tmr.tick()	Get the current time tick of the MCU (ms) since startup
tmr.delayms()	Delay for a assigned time in micro seconds
tmr.wdclr()	Clear the Independent watchdog counter

### tmr.start()

#### Description

Start a timer with call back function.

#### Syntax

```
tmr.start(tmrID, interval, func_cb)
```

#### Parameters

tmrID: timer ID, 0~15. 16 timers are supported at present

interval: interval time for the timer

func\_cb: Callback function for the timer

#### Returns

nil

#### Examples

```
-tmr.start(1,1000,function() print("tmr1 is called") end)
```

```
-tmr1 is called
```

```
tmr1 is called
```

```
tmr1 is called
```

### tmr.stop()

#### Description

Stop a timer

#### Syntax

```
tmr.stop(tmrID)
```

**Parameters**

tmrID: timer ID, 0~15

**Returns**

nil

**Examples**

```
-tmr.start(1,1000,function() print("tmr1 is called") end)
```

```
-tmr1 is called
```

```
tmr1 is called
```

```
tmr1 is called
```

```
-tmr. stop(1)
```

## tmr.stopall()

---

**Description**

Stop all the timer.

**Syntax**

```
tmr.stopall(tmrID)
```

**Parameters**

nil

**Returns**

nil

**Examples**

```
-tmr. stopall()
```

## tmr.tick()

---

**Description**

Get the current time tick of the MCU (ms) since startup.

**Syntax**

```
tick=tmr.tick()
```

**Parameters**

nil

### Returns

nil

### Examples

```
-print(tmr.tick())
```

1072237

## tmr.delayms()

---

### Description

Delay for a assigned time in micro seconds.

### Syntax

```
tmr.delayms(ms)
```

### Parameters

ms: The delay time in micro seconds

### Returns

nil

### Examples

```
-tmr.delayms(1000)
```

## tmr.wdclr()

---

### Description

Clear the independent watchdog counter. The default independent watchdog time is 10 seconds.

**Note:** This function should be called if some operations cost over 10 seconds.

### Syntax

```
tmr.wdclr ()
```

### Parameters

nil

### Returns

nil

### Examples

-tmr.wdclr()



## Function list

Function	Definition
wifi.startap()	Setup wifi in soft Access Point (AP) Mode, enable DHCP function
wifi.startsta()	Setup wifi in Station Mode (STA), begin to connect a AP
wifi.scan()	Scan APs
wifi.stop()	Close all the Wi-Fi connections, Both in station mode and soft ap mode
wifi.powersave()	Enable IEEE power save mode
wifi.ap.getip()	Get ip address in soft AP mode
wifi.ap.getipadv()	Get advanced net information in soft AP mode: DHCP mode, ip address, gateway, netmask, dns, MAC, broad cast address
wifi.ap.stop()	Close all the Wi-Fi connections in soft ap mode
wifi.sta.getip()	Get ip address in STA mode
wifi.sta.getipadv()	Get advanced net information in STA mode: DHCP mode, ip address, gateway, netmask, dns, MAC, broad cast address
wifi.sta.getlink()	Get the connected AP information in STA mode: Connect status, WiFi signal strength, ssid, bssid.
wifi.sta.stop()	Close all the Wi-Fi connections in STA mode

## wifi.startap()

### Description

Setup wifi in soft Access Point (AP) Mode, enable DHCP function.

### Syntax

```
wifi.startap(cfg)
```

```
wifi.startap(cfg,func_cb)
```

### Parameters

cfg: lua table, contains the configurations for soft AP mode.

cfg.ssid: soft AP's ssid

cfg.pwd: soft AP's password. It will be an open WiFi if cfg.pwd is empty

cfg.ip: optional. The local ip address of the module, It's "11.11.11.1" in default.

cfg.netmask : optional. Netmask. It's "255.255.255.0" in default.

cfg.gateway : optional. Gateway. It's "11.11.11.1" in default.

cfg.dnsSrv : optional. DNS server address. It's "11.11.11.1" in default.

cfg.retry\_interval : optional. retry interval in micro seconds. It's 1000ms in default.

**func\_cb:** The callback function when the soft AP is setup successfully or the soft AP is shut down. Function prototype is: `func_cb(info)`. "info" is the information indicates the event: 'STATION\_UP', 'STATION\_DOWN', 'AP\_UP', 'AP\_DOWN', 'ERROR'

### Returns

nil

### Examples

```
-cfg={}
```

```
-cfg.ssid="WiFiMCU_Wireless"; cfg.pwd=""
```

```
-wifi.startap(cfg)
```

## wifi.startsta()

---

### Description

Setup wifi in Station Mode (STA), begin to connect a AP.

### Syntax

```
wifi.startsta(cfg)
```

```
wifi.startsta(cfg, func_cb)
```

### Parameters

**cfg:** lua table, contains the configurations for soft AP mode.

**cfg.ssid:** AP's ssid

**cfg.pwd:** AP's password

**cfg.dhcp:** optional. Set dhcp function: 'enable' is to enable the dhcp function. WiFiMCU will get ip automatically. 'disable' is to disable the dhcp function. It's 'enable' in default.

**cfg.ip:** optional. The local ip address of the module. If **cfg.dhcp** is 'disable' this parameter must be assigned.

**cfg.netmask :** optional. Netmask. If **cfg.dhcp** is 'disable' this parameter must be assigned.

**cfg.gateway :** optional. Gateway. If **cfg.dhcp** is 'disable' this parameter must be assigned.

**cfg.dnsSrv :** optional. DNS server address. If **cfg.dhcp** is 'disable' this parameter must be assigned.

**cfg.retry\_interval :** optional. retry interval in micro seconds. If **cfg.dhcp** is 'disable' this parameter must be assigned.

**func\_cb:** The callback function when WiFiMCU had connected to the AP successfully, or WiFiMCU is disconnected to from the AP. Function prototype is: `func_cb(info)`. "info" is the information indicates the event: 'STATION\_UP', 'STATION\_DOWN', 'AP\_UP', 'AP\_DOWN', 'ERROR'

### Returns

nil

**Examples**

```
-cfg={}
```

```
-cfg.ssid="Doit"; cfg.pwd="123456789"
```

```
-wifi.startsta(cfg)
```

## wifi.scan()

---

**Description**

Scan AP list and return a Lua table contains the results.

**Syntax**

```
wifi.scan(fun_cb(t))
```

**Parameters**

func\_cb(t): The callback function when scan is finished. 't' is a Lua table in which the keys are the APs' ssid and values are strings in format (" mac, signal strength, channel, authmode")

**Returns**

nil

**Examples**

```
-function listap(t) if t then for k,v in pairs(t) do print(k.."t"..v);end else print('no ap') end end
```

```
-wifi.scan(listap)
```

```
CMCC-WEB 00:23:89:22:98:B0,90,11,OPEN
```

```
MERCURY_44B6 C0:61:18:21:44:B6,75,6,WPA2 AES
```

```
Tomato 8C:28:06:1E:01:54,100,11,WPA2 AES
```

```
ChinaNet-mALi 8C:E0:81:30:C1:95,65,10,WPA2 AES
```

```
Wireless 00:25:12:62:A6:36,57,6,OPEN
```

```
CMCC 00:23:89:22:98:B1,87,11,WPA2 AES
```

```
CMCC-FREE 00:23:89:96:02:03,60,11,OPEN
```

```
Doit BC:D1:77:32:E7:2E,100,1,WPA2 AES
```

## wifi.stop()

---

**Description**

Close all the Wi-Fi connections, Both in station mode and soft ap mode.

**Syntax**

```
wifi.stop()
```

**Parameters**

nil

**Returns**

nil

**See also**

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

```
wifi.sta.stop()
```

**Examples**

```
-wifi.stop()
```

## wifi.powersave()

---

**Description**

Enable IEEE power save mode.

**Syntax**

```
wifi.powersave ()
```

**Parameters**

nil

**Returns**

nil

**Examples**

```
-wifi.powersave ()
```

## wifi.ap.getip()

---

**Description**

Get ip address in AP mode

**Syntax**

```
ip=wifi.ap.getip()
```

**Parameters**

nil

### Returns

ip: The module ip in soft AP mode.

### Examples

```
-ip=wifi.ap.getip ()
```

```
-print(ip)
```

```
11.11.11.1
```

## wifi.ap.getipadv()

---

### Description

Get advanced net information in soft AP mode: DHCP mode, ip address, gate way, net mast, dns, MAC, broad cast address.

### Syntax

```
dhcp,ip,gw,nm,dns,mac,bip =wifi. ap.getipadv()
```

### Parameters

nil

### Returns

dhcp: DHCP mode. in soft AP mode, it will be always "DHCP\_Server"

ip: ip address.

gw: gateway address.

nm: netmask.

dns: dns address.

mac: MAC address.

bip: broadcast ip address.

### Examples

```
-dhcp,ip,gw,nm,dns,mac,bip =wifi.ap.getipadv()
```

```
-print(dhcp,ip,gw,nm,dns,mac,bip)
```

```
DHCP_Server 11.11.11.1 11.11.11.1 255.255.255.0 208.67.222.222 c89346501a62 255.255.255.255
```

## wifi.ap.stop()

---

**Description**

Close all the Wi-Fi connections in soft ap mode.

**Syntax**

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

**Parameters**

nil

**Returns**

nil

**See also**

wifi.stop()

wifi.sta.stop()

**Examples**

```
-wifi.ap.stop()
```

## wifi.sta.getip()

---

**Description**

Get ip address in STA mode.

**Syntax**

```
ip=wifi. sta.getip()
```

**Parameters**

nil

**Returns**

ip: The module ip in STA mode.

**Examples**

```
-ip=wifi.sta.getip ()
```

```
-print(ip)
```

```
192.168.1.108
```

## wifi.sta.getipadv()

---

**Description**

Get advanced net information in STA mode: DHCP mode, ip address, gateway, netmask, dns, MAC, broad cast address.

### Syntax

```
dhcp,ip,gw,nm,dns,mac,bip =wifi. sta.getipadv()
```

### Parameters

nil

### Returns

dhcp: DHCP mode. in STA mode, "DHCP\_Server" or "DHCP\_Client" or DHCP\_Disable

ip: ip address.

gw: gateway address.

nm: netmask.

dns: dns address.

mac: MAC address.

bip: broadcast ip address.

### Examples

```
-dhcp,ip,gw,nm,dns,mac,bip =wifi.sta.getipadv()
```

```
-print(dhcp,ip,gw,nm,dns,mac,bip)
```

```
DHCP_Client 192.168.1.108 192.168.1.1 255.255.255.0 192.168.1.1 c89346501a62 255.255.255.255
```

## wifi.sta.getlink()

---

### Description

Get the connected AP information in STA mode:Connect status, WiFi signal strength, ssid, bssid.

### Syntax

```
status,strength,ssid,bssid=wifi.sta.getlink()
```

### Parameters

nil

### Returns

status: The connecting status. if connected it's "connected" else it's "disconnected". It will be nil for strength/ssid/bssid if it's "disconnected".

strength: The signal strength.

ssid : The connected AP's ssid.

bssid: The connected AP's bssid.

### Examples

```
-status,strength,ssid,bssid=wifi.sta.getlink()
```

```
-print(status,strength,ssid,bssid)
```

```
connected 62 Doit BC:D1:77:32:E7:2E
```

## wifi.sta.stop()

---

### Description

Close all the Wi-Fi connections in STA mode.

### Syntax

```
wifi.sta.stop()
```

### Parameters

nil

### Returns

nil

### See also

wifi.stop()

wifi.ap.stop()

### Examples

```
-wifi.sta.stop()
```



## Function list

Function	Definition
net.new()	Create a new socket, set the socket and transmission protocol
net.start()	Start the socket, set remote port, remote ip address, or local port according to the socket and transmission protocol
net.on()	Register the callback functions for socket events
net.send()	Send data
net.close()	Close socket
net.getip()	Get the ip address and port of the client socket.

## Constant

Constant	Definition
net.TCP	TCP protocol
net.UDP	UDP protocol
net.SERVER	Server type
net.CLIENT	Client type

## net.new()

### Description

Create a new socket, set the socket and protocol type. Max 4 server and Max 4 client can be setup in WiFiMCU. If the socket type is Server, max number of 5 clients are allowed to connect.

### Syntax

```
skt=net.new(protocol,type)
```

### Parameters

protocol: The transmission protocol, must be one of the two: net.TCP, net.UDP

type: socket type, must be one of the two: net.SERVER, net.CLIENT

### Returns

skt: the handle for this socket

### Examples

```
-skt = net.new(net.TCP,net.SERVER)
```

```
-skt2 = net.new(net.UDP,net.CLIENT)
```

## net.start()

---

### Description

Start the socket, set remote port, remote ip address, or local port according to the socket and transmission protocol.

### Syntax

```
net.start(socket, localport)
```

```
net.start(socket, remoteport, "domain", [local port])
```

### Parameters

socket: The socket handle returned from net.new()

localport: If the socket type is net.SERVER, It's the local binded port for this socket.

remoteport: If the socket type is net.CLIENT, It's the remote server port.

"domain": If the socket type is net.CLIENT, it's the domain name string for remote server. The remote server's ip address can be used too.

[local port]: Optinal, if the socket type is net.CLIENT, [local port] set the local binded port for the socket. If ignored, a random port would be assigned.

### Returns

nil

### Examples

```
-skt = net.new(net.TCP,net.SERVER)
```

```
-skt2 = net.new(net.UDP,net.CLIENT)
```

```
-net.start(skt, 80)
```

```
-net.start(skt2,9000,'11.11.11.2', 8000)
```

## net.on()

---

### Description

Register the callback functions for socket events.

### Syntax

```
net.on(socket,event,func_cb)
```

### Parameters

socket: The socket handle returned from net.new()

event: If the socket type is net.SERVER, event should be one of the following:

“accept”(TCP server socket only), “receive”, “sent”, “disconnect”.

If the socket type is `net.CLIENT`, event should be one of the following:

“connect”(TCP client socket only), “receive”, “sent”, “disconnect”, “dnsfound”.

`func_cb`: Callback function for different events. The function parameters diff from events.

“accept”: TCP server socket only. If the tcp server accept a tcp client connection request, the function will be called. Function prototype is: `func_cb(clt, ip, port)`. “clt” is the tcp client socket handle, “ip” is the client ip address, “port” is the client’s port.

“receive”: If data arrived on the assigned socket, the function will be called. Function prototype is: `func_cb(clt, data)`. “clt” is the socket handle, “data” is the received data.

“sent”: When data had sent succeffuly on the assigned socket, the function will be called. Function prototype is: `func_cb(clt)`. “clt” is the socket handle.

“disconnect”: If the client socket is disconnected from server or some errors happened, the function will be called. Function prototype is: `func_cb(clt)`. “clt” is the socket handle.

“connect”: TCP Client socket only. When the client socket connects to the remote server successfully, the function will be called. Function prototype is: `func_cb(clt)`. “clt” is the socket handle.

“dnsfound”: TCP or UDP Client socket only. When the DNS operations has finished, the function will be called. Function prototype is: `func_cb(clt, ip)`. “clt” is the socket handle, “ip” is the ip address for the domain.

## Returns

nil

## Examples

```
-clt = net.new(net.TCP,net.CLIENT)

-net.on(clt,"dnsfound",function(clt,ip) print("dnsfound clt:"..clt.." ip:"..ip) end)

-net.on(clt,"connect",function(clt) print("connect:clt:"..clt) end)

-net.on(clt,"disconnect",function(clt) print("disconnect:clt:"..clt) end)

-net.on(clt,"receive",function(clt,d) print("receive:clt:"..clt.."data:"..d) end)

-net.start(clt,9003,"11.11.11.2")
```

# net.send()

## Description

Send data.

## Syntax

```
net.send(socket, data, [func_cb])
```

## Parameters

Net Module

socket: The socket handle returned from `net.new()`

data: Data to be sent.

[func\_cb]: Optional, “sent” eventcall back function. When data had sent succcessfully on the assigned socket, the function will be called. Function prototype is: `func_cb(clt)`. “clt” is the socket handle.

### Returns

nil

### Examples

```
-net.send(clt,"hello")
```

## net.close()

---

### Description

Close socket, release the resource of the socket.

### Syntax

```
net.close(socket)
```

### Parameters

socket: The socket handle returned from `net.new()`

### Returns

nil

### Examples

```
-skt = net.new(net.TCP,net.SERVER)
```

```
-net.close(skt)
```

## net.getip()

---

### Description

Get the ip address and port of the client socket.

### Syntax

```
ip, port = net.getip(socket)
```

### Parameters

socket: The socket handle returned from `net.new()`. The socket handle should be a client socket.

### Returns

ip: the ip address for the socket.

port: the port for the socket.

### **Examples**

```
-ip, port = net.getip(clt)
```

## File Module

The file system is based on spi flash embeded in WiFiMCU. The totoal storage capacity is 1280k  $[(1024+256)*1024]$  bytes.

## Function list

Function	Definition
file.format()	Format file system, all stored data will be lost after format
file.open()	Open or create a file
file.close()	Close an opened file
file.write()	Write data to an opened file
file.writeline()	Write data to an opened file, with a '\n' added at the tailed of data
file.read()	Read data from an opened file
file.readline()	Read a line data from an opened file
file.list()	Get the file name and size list in file system
file.slist()	Print the file name and size list on terminal
file.remove()	Remove file
file.seek()	Set the position of file pointer
file.flush()	Clear file buffer
file.rename()	Rename the file
file.info()	Get the file system storage status
file.state()	Get the opened file's name and size
file.compile()	Compile a Lua scripts file to lc file.
dofile()	Run a file

## file.format()

### Description

Format file system, all stored data will be lost after format. It's recommended Do not do any things while formatting.

### Syntax

```
file.format()
```

### Parameters

nil

### Returns

nil

If formatting is done successfully, "format done" will be printed, else "format error" will be printed.

### Examples

```
-file.format()
```

format done

## file.open()

---

### Description

Open or create a file.

### Syntax

```
ret = file.open(filename,mode)
```

### Parameters

filename: filename string to be created or opened. Directories are not supported yet.

mode: opened type:

"r": read mode (the default parameter)

"r+": update mode, all previous data is preserved

"w": write mode

"w+": update mode, all previous data is erased

"a": append mode

"a+": append update mode, previous data is preserved, writing is only allowed at the end of file

### Returns

ret: true if succeed, else nil.

### Examples

```
-file.open("test.lua","w+")
```

```
-file.write("This is a test")
```

```
-file.close()
```

## file.close()

---

### Description

Close an opened file.

### Syntax

```
file.close()
```

**Parameters**

nil

**Returns**

nil

**Examples**

```
-file.open("test.lua","w+")
```

```
-file.write("This is a test")
```

```
-file.close()
```

## file.write()

---

**Description**

Write data to an opened file.

**Syntax**

```
ret=file.write(data)
```

**Parameters**

data: The data to be wrote.

**Returns**

ret: true if succeed, else nil.

**Examples**

```
-file.open("test.lua","w+")
```

```
-file.write("This is a test")
```

```
-file.close()
```

## file.writeline()

---

**Description**

Write data to an opened file, with a ‘\n’ added at the tailed of data.

**Syntax**

```
ret=file.writeline(data)
```

**Parameters**



data: The data to be wrote. A char '\n' will be added at the end of data.

### Returns

ret: true if succeed, else nil.

### Examples

```
-file.open("test.lua","w+")
-file.writeline("This is a test")
-file.close()
```

## file.read()

---

### Description

Read data from an opened file.

### Syntax

```
ret=file.read()
ret=file.read(num)
ret=file.read(endchar)
```

### Parameters

if the parameter is nil, read all byte in file.

num: if a number is assigned, read the num bytes from file, or all rest data in case of end of file.

endchar: read until endchar or EOF is reached.

### Returns

ret: the file data if succeed, else nil.

### Examples

```
-file.open("test.lua","r")
-data=file.read()
-file.close()
-print(data)

This is a test

-file.open("test.lua","r")
-data=file.read(10)
```

```
-file.close()
```

```
-print(data)
```

This is a

```
-file.open("test.lua","r")
```

```
-data=file.read('e')
```

```
-file.close()
```

```
-print(data)
```

This is a te

## file.readline()

---

### Description

Read a line data from an opened file.

### Syntax

```
ret=file.readline ()
```

### Parameters

nil

### Returns

ret: the file data if succeed, else nil.

### Examples

```
-file.open ("test.lua","w+")
```

```
-file.writeline("this is a test")
```

```
-file.close()
```

```
-file.open ("test.lua","r")
```

```
-data=file.readline()
```

```
-print(data)
```

```
-This is a test
```

```
-file.close()
```

## file.list()

---

### Description

File Module

Get the file name and size list in file system.

### Syntax

```
ft=file.list()
```

### Parameters

nil

### Returns

ft: a Lua table, in which the filename is the key, file size is the value.

### Examples

```
-for k,v in pairs(file.list()) do print("name:"..k.." size(bytes):"..v) end
```

```
-name:test.lua size(bytes):15
```

## file.slist()

---

### Description

Print the file name and size list on terminal.

### Syntax

```
file.slist()
```

### Parameters

nil

### Returns

nil

### Examples

```
-file.slist()
```

```
test.lua size:15
```

## file.remove()

---

### Description

Remove file.

### Syntax

```
file.remove(filename)
```

### Parameters

File Module

filename: filename string to be removed.

### Returns

nil

### Examples

```
-file.remove ("test.lua")
```

## file.seek()

---

### Description

Set the position of file pointer.

### Syntax

```
fi = file.seek(whence, offset)
```

### Parameters

whence: should be one of the following:

"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

fi: the file pointer final position if succeed, else nil.

### Examples

```
-file.open ("test.lua","r")
```

```
-file.seek("set",10)
```

```
-data=file.read()
```

```
-file.close
```

```
-print(data)
```

```
test
```

## file.flush()

---

### Description

Clear file buffer.

**Syntax**

```
ret = file.flush()
```

**Parameters**

nil

**Returns**

ret: true if succeed, else nil.

**Examples**

```
-file.open ("test.lua","r")
```

```
-file.flush ()
```

```
-file.close()
```

## file.rename()

---

**Description**

Rename the file.

**Syntax**

```
ret=file.rename(oldname,newname)
```

**Parameters**

oldname: File name to be changed.

newname: New file name.

**Returns**

ret: true if succeed, else nil.

**Examples**

```
-file.slist()
```

```
test.lua size:14
```

```
-file.rename ('test.lua',' testNew.lua')
```

```
-file.slist()
```

```
testNew.lua size:14
```

## file.info()

---

**Description**

File Module

Get the file system storage status.

### Syntax

```
last,used,total = file.info()
```

### Parameters

nil

### Returns

last: free storage left in bytes.

used: used storage in bytes.

total: all allocated storage for file system in bytes.

### Examples

```
-last,used,total = file.info()
```

```
-print(last,used,total)
```

```
1140500 2750 1143250
```

## file.state()

---

### Description

Get the opened file's name and size

### Syntax

```
fn,sz = file.state()
```

### Parameters

nil

### Returns

fn: filename.

sz: file size in bytes.

### Examples

```
-file.open("testNew.lua","r")
```

```
-fn,sz = file.state()
```

```
-file.close()
```

```
-print(fn,sz)
```

testNew.lua 14

## file.compile()

---

### Description

Compile a Lua scripts file to lc file. The lc file will be named as the same name as the Lua file.

### Syntax

```
file.compile('filename.lua')
```

### Parameters

filename.lua: file name of the Lua scripts.

### Returns

nil.

### Examples

```
-file.open("test.lua","w+")
```

```
-file.write("print('Hello world!')")
```

```
-file.close()
```

```
-file.compile("test.lua")
```

```
-file.slist()
```

```
test.lua size:21
```

```
test.lc size:100
```

## dofile()

---

### Description

Run a file. The file can be either a Lua scripts or a lc format file.

### Syntax

```
dofile('filename.lua')
```

```
dofile('filename.lc')
```

### Parameters

filename.lua: Lua scripts file.

filename.lc: a lc file

### Returns

File Module

nil.

### **Examples**

```
-dofile("test.lua")
```

Hello world!

```
-dofile("test.lc")
```

Hello world!



## Function list

Function	definition
pwm.start()	Start pwm function at assigned gpio pin
pwm.stop()	Stop pwm

## Pin Table

Plaese refer: "GPIO Table" for detail.

## pwm.start()

### Description

Start pwm function at assigned gpio pin.

### Syntax

```
pwm.start(pin, freq, duty)
```

### Parameters

pin: gpio pin ID. There are 11 PWM ports supported in WiFiMCU: D1, D3, D4, D9, D10, D11, D12, D13, D14, D15, D16.

freq: PWM output frequency in Hz,  $0 < \text{freq} < 10\text{KHz}$

duty: Duty of PWM output, must be  $0 \leq \text{duty} \leq 100$

### Returns

nil.

### Examples

```
i=1;pin=1;

tmr.start(1,1000,function()

i=i+10;if i>=100 then i=1 end

pwm.start(pin,10000,i)

end)
```

-

## pwm.stop()

### Description

Stop pwm.

## Syntax

```
pwm.stop(pin)
```

## Parameters

pin: gpio pin ID. There are 11 PWM ports supported in WiFiMCU: D1, D3, D4, D9, D10, D11, D12, D13, D14, D15, D16.

## Returns

nil.

## Examples

```
-pwm.stop(1)
```

## Function list

Function	Definition
adc.read()	Read the ADC result at assigned pin

## Pin Table

Plaese refer: "GPIO Table" for detail.

## adc.read()

### Description

Read the ADC result at assigned pin.

### Syntax

```
data= adc.read(pin)
```

### Parameters

pin: gpio pin ID. There are 5 ADC ports supported in WiFiMCU: D1, D13, D15, D16, D17.

### Returns

data: if succeed, data between 0~4095 is returned, else nil.

**Note that: 0 presents 0V, 4095 presents 3.3V.**

### Examples

```
-=adc.read(1)
```

```
-1
```

```
-=adc.read(1)
```

```
-4095
```

Only one uart is supported in WiFiMCU so far. The GPIO pin is D8(RX1), D9(TX1).

## Function list

Function	Definition
uart.setup()	Setup uart parameters: baudrate, databits, parity, stopbits.
uart.on()	Register the callback functions for uart events
uart.send()	Send data via uart

### uart.setup()

#### Description

Setup uart parameters: baudrate, databits, parity, stopbits.

#### Syntax

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

#### Parameters

id: uart ID, always 1 at present.

baud: baudrate, such as: 4800, 9600, 115200.

parity: 'n': no parity, 'o': odd parity, 'e': even parity.

databits: data bits, '5', '6', '7', '8', '9'.

stopbits: stop bits, '1', '2'

#### Returns

nil

#### Examples

```
-uart.setup(1,9600,'n','8','1')
```

### uart.on()

#### Description

Register the callback functions for uart events.

#### Syntax

```
uart.on(id, event ,func_cb)
```

#### Parameters

id: uart ID, always 1 at present.

event: always "data".

func\_cb: Callback function for the event. When data arrived, the function will be called. Function prototype is:  
func\_cb(data). "data" is the data received.

### Returns

nil

### Examples

```
-uart.on(1, 'data',function(t) len=string.len(t) print(len.." "..t) uart.send(1,t) end)
```

## uart.send()

---

### Description

Send data via uart.

### Syntax

```
uart.send(1, string1,[number],...[stringn])
```

### Parameters

id: uart ID, always 1 at present.

string1: string ready to send.

[number]: Optional, number ready to send.

[stringn]: Optional, The nth string ready to be send.

### Returns

nil

### Examples

```
-uart.send(1,'hello wifimcu')
```

```
-uart.send(1,'hello wifimcu','hi',string.char(0x32,0x35))
```

```
-uart.send(1,string.char(0x01,0x02,0x03))
```

To be continued.

To be continued.

To be continued.



To be continued.

To be continued