WiFiMCU Lua Reference Book

DoIT / LoBo

Ver. 0.9.15_LoBo

2016-02-13

Content

Lua Basic Modules

The Lua interpreter in WiFiMCU is based on Lua 5.1.4.
The following modules are supported:

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/

MCU Module

Function List

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 caused its startup
mcu.getparams()	Get system parameters to lua table
mcu.sgetparams()	Print system parameters
mcu.setparams()	Set system parameters
mcu.random()	Returns random number

Constant

nil

System parameters & Watchdog

System parameters

There are a number of system parameters which can be modified to set the basic Lua system behavior. The parameters are saved in the parameter area of the WiFiMCU SPI Flash and are preserved between reboots/power cycles. The parameters are protected with CRC.

use_wwdg	selects IWDG or WWDG watchdog function (default: 0 = IWDG; 1 = WWDG)
wdg_tmo	watchdog timeout in seconds miliseconds (default: 10000)
stack_size	size of the Lua thread stack in bytes (default: 10KB)
inbuf_size	size of the Lua input buffer in bytes (default: 256)
baud_rate	baud rate of the Lua terminal (default: 115200)
parity	parity used in Lua terminal (default: 'n', no parity)
init_file	name of the file which is executed on system start, if the name is "", no file is executed. (default: "")

If some wrong parameters are set, and the system wont start, the parameters can be restored to the default values in **bootloader**, executing **3 -e** command.

Watchdog

The system is protected by the watchdog. If the watchdog is not reloaded before the watchdog timeout expires, the system is RESET. The watchdog is automatically refreshed during the waiting for user input. If you have some long running Lua program, you have to reload the watchdog using *tmr.wdclr()* function before the watchgog timeout expires..

There are two types of watchdog in WiFiMCU Lua:

Watchdog type 0 is *STM32F411CE* **IWDG** timer (Independent Watchdog) which is set on system start and cannot be disabled. The IWDG is enabled even in STOP mode, so you cannot use STOP power save mode if this type of watchdog is used.

Watchdog type 1 is *STM32F411CE* **WWDG** timer (Window Watchdog). It does not run in STOP mode, so it is possible to use STOP power save mode with this watchdog type.

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 0.9.7 build 20151122
```

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 wl0: 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

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
```

mcu.getparams()

Description

Get system parameters as Lua table.

Syntax

param = mcu.getparams()

Parameters

nil

Returns

param: Lua table containing the system parameters

Examples

```
> param= mcu.getparams()
> for k,v in pairs(param) do print("param: "..k.." = "..v) end
param: inbuf_size = 256
param: stack_size = 20480
param: baud_rate = 115200
param: crc = Ok
param: wdg_tmo = 10000
```

```
param: parity = NO_PARITY
param: init_file =
param: use_wwdg = 0
```

mcu.sgetparams()

Description

Print system parameters.

Syntax

mcu.sgetparams()

Parameters

nil

Returns

nil, prints parameters

Examples

```
> mcu.sgetparams()
   use_wwdg = 0
    wdg_tmo = 10000
 stack_size = 20480
 inbuf_size = 256
  init_file = ""
  baud_rate = 115200
     parity = 'n'
CRC ok.
```

mcu.setparams()

Description

Set one or more system parameters.

Syntax

mcu.setparams(paramtbl)

Parameters

paramtbl: Lua table containing one or more parameters

use_wwdg	watchdog type, 0 or 1, default = 0 (hardware IWDG)
wdg_tmo	watchdog timeout in milisec, 2000~36000000, default = 10000
stack_size	lua stack size in bytes, 5000~31000, default = 10240
inbuf_size	lua inpu buffer size in bytes, 128~1024, default = 256
init_file	name of the file executed on boot, default = "", no script executed
baud_rate	lua serial terminal baud rate, default = 115200
parity	lua serial terminal parity, ‘n’ or ‘e’ or ‘o’, default = 'n'

Note: if watchdog type is changed, the system will reboot after watchdog timeout expires.

Returns

nil, prints status

Examples

```
> =mcu.setparams({stack_size=10240,use_wwdg=1,init_file="init.lua"})
updated: soft_wdg, RESET in 10 sec!
updated: stack_size
updated: init_file
New params saved.
```

mcu.random()

Description

Return random number. Optional limits can be set.

Syntax

num = mcu.random([maxval], [minval], [seed])

Parameters

maxval	optional; maximal random number to return
minval	optional; minimal random number to return
seed	optional; reseed random number generator if seed = 1

Returns

num random number

Examples

```
> =mcu.random(320)
117
```

GPIO Module

Function List

gpio.mode()	Define the GPIP 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.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

WiFiMCU Index	Alternate function	Discription
D0	GPIO/BOOT	WiFiMCU would enter into Bootloader Mode, if D0 goes to LOW
D1	GPIO/PWM/ADC	
D2	GPIO	
D3	GPIO/PWM	Hardware I2C interface: SDA
D4	GPIO	
D5	GPIO	SWD Flash Programming Pin: swclk
D6	GPIO	SWD Flash Programming Pin: swdio
D7	GPIO/SPI5_MISO	Hardware SPI: MISO
D8	GPIO/PWM/SPI5_MOSI	UART1 Rx pin: RX1 Hardware SPI: MOSI
D9	GPIO/PWM	UART1 Tx pin: TX1
D10	GPIO/PWM	
D11	GPIO/PWM	Hardware I2C interface: SDA
D12	GPIO/PWM	
D13	GPIO/PWM/ADC	
D14	GPIO/PWM	
D15	GPIO/PWM/ADC	
D16	GPIO/PWM/ADC/SPI5_CLK	Hardware SPI: CLK
D17	GPIO/ADC	BLUE LED on WiFiMCU 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 you DO NOT do too much time consuming 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)
>gpio.toggle(17)
```

TIMER Module

Function List

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

Constant

nil

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 millisecond.

Syntax

tmr.delayms(ms)

Parameters

ms: The delay time in millisecond

Returns

nil

Examples

```
> tmr.delayms(1000)
```

tmr.delayus()

Description

Delay for a assigned time in microsecond.

Syntax

tmr.delayus(us)

Parameters

us: The delay time in microsecond

Returns

nil

Examples

```
> tmr.delayus(1000)
```

tmr.wdclr()

Description

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

Note: This function should be called if some operations takes more than 10 seconds to complete

Syntax

tmr. wdclr ()

Parameters nil

Returns nil

Examples

```
> tmr.wdclr()
```

WiFi Module

Function list

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.sta.ntptime()	Set RTC datetime from ntp server

Constant

nil

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, Default: “11.11.11.1”
cfg.netmask: optional. Default: “255.255.255.0”
cfg.gateway: optional. Dafault: “11.11.11.1”
cfg.dnsSrv: optional. DNS server address. Default: “11.11.11.1”
cfg.retry_interval: optional. retry interval in micro seconds. Default: 1 sec

func_cb: The callback function when the soft AP is setup successfully or the soft AP is shut down.

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.

Returns nil

Examples

```
>cfg={}
>cfg.ssid="Doit"; cfg.pwd="123456789"
>wifi.startsta(cfg)
```

wifi.scan()

Description

Scan AP list and return a Lua table containing 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)
> LoBoInternet      9C:C7:A6:45:B9:E7,100,11,WAP2 MIXED
hHyVEd  58:98:35:B8:3E:17,45,11,WAP2 MIXED
B.net_98796   58:23:8C:83:69:D3,70,1,WPA TKIP
B.net_11651   88:F7:C7:9A:CE:B0,72,6,WPA TKIP
3c3f1e  4C:72:B9:89:0C:FE,40,9,WPA AES
ISKONOVAC-4016CC    2A:28:5D:40:16:CC,30,1,WAP2 MIXED
```

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, broadcast 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()
```

wifi.sta.ntptime()

Description

Set RTC datetime from ntp server.

Syntax

wifi.sta.ntptime()
wifi.sta.ntptime(timezone)
wifi.sta.ntptime(timezone,ntpserver)

Parameters

timezone: optional, use specified time zone offset from UTC (-12 - +14), default=0
ntpserver: optional, specify ntp server to use, default="time1.google.com"

Returns

status: disconnected if no wifi connection detected

Examples

```
> wifi.sta.ntptime(1)
```

Net Module

Function list

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

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

socket: The socket handle returned from net.new()
data: Data to be sent.

func_cb: Optinal, “sent” eventcall back function. 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.

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 total storage capacity is ~1550K

Function list

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.
file.recv()	Receive the file using Ymodem protocol
file.send()	Send the file using Ymodem protocol
dofile()	Run a file

Constant

nil

file.format()

Description

Format file system, all stored data will be lost after format. It's recommended not to do anythings 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: open 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

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

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

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
```

file.recv()

Description

Receive the file over serial line using ymodem protocol.

Syntax file.recv(["filename"])

Parameters

filename: optional; if specified, the file is saved with that name, otherwise, file name from the sender is used

Returns nil. After receive, the directory content is printed

Examples

```
> file.recv()
Start Ymodem file transfer...
CCCCCCCCCCCCCCC
Starting ymodem transfer.  Press Ctrl+C to cancel.
Transferring test.lua...
  100%     12 KB     12 KB/sec    00:00:01      0 Errors

Received successfully, 12389

 oledtest.lua                size: 1178
 init.lua                    size: 1337
 test.lua                    size: 12389


> file.recv("new_test.lua")
Start Ymodem file transfer...
CCCCCCCCCCCCCCCCCCCCCCCCCCCC
Starting ymodem transfer.  Press Ctrl+C to cancel.
Transferring test.lua...
  100%     12 KB     12 KB/sec    00:00:01      0 Errors

Received successfully, 12389

 oledtest.lua                size: 1178
 init.lua                    size: 1337
 test.lua                    size: 12389
 new_test.lua                size: 12389
```

file.send()

Description

Send the file over serial line using ymodem protocol.

Syntax file.recv(“filename”,[“newfilename”])

Parameters

Filename:	name of the file to send
newfilename:	optional; if specified, the file is sent with that name, othewise, the original file name is used

Returns nil. After receive, the directory content is printed

Examples

```
> file.send("test.lua")
Start Ymodem file transfer...
CCCCCCCC
Starting ymodem transfer.  Press Ctrl+C to cancel.
Transferring test.lua...
  100%     12 KB     12 KB/sec    00:00:01      0 Errors

File sent successfuly.

> file.send("test.lua","old-test.lua")
sending "test.lua" as "old-test.lua"

Start Ymodem file transfer...
CCCCCCC
Starting ymodem transfer.  Press Ctrl+C to cancel.
Transferring old-test.lua...
  100%     12 KB     12 KB/sec    00:00:01      0 Errors

File sent successfuly.
```

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 nil.

Examples

```
>dofile("test.lua")
Hello world!
>dofile("test.lc")
Hello world!
```

PWM Module

Function list

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

Constant

nil

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<freq<10KHz
duty: Duty of PWM output, must be 0<=duty <=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)
```

ADC Module

Function list

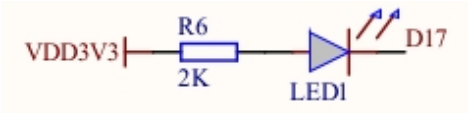
adc.read()	Read the ADC result at assigned pin
adc.readV()	Read the ADC result in V at assigned pin
adc.setref()	Set Reference voltage
adc.setautocal()	Set auto calibration on or off

Constant

nil

Pin Table

Usable pins: D1, D13, D15, D16, D17
Note: On WiFiMCU board pin D17 is connected to BLUE LED,
and is not recommended for use as ADC input.



adc.read()

Description

Returns the integer value representing ADC value at assigned pin.

Syntax

Data, std = adc.read(pin ,[samples])

Parameters

pin:	gpio pin ID (1,13,15,16,17,99). There are 5 ADC ports supported on WiFiMCU: D1, D13, D15, D16, D17. If pin=99, internal voltage reference value is returned
samples:	optional; number of samples to calculate the average (2~128)

Returns

data: if succeed, integer data between 0~4095 is returned, else nil.
std: standard deviation, floating point value
Note: 0 presents 0V, 4095 presents ~3.3V.

Examples

```
>=adc.read(1)
0   0.000000
>=adc.read(1,128)
4095    4.76398332
```

adc.readV()

Description

Returns the floating point ADC value in V (volts) at assigned pin.

Syntax

Data,std= adc.readV(pin [,samples])

Parameters

Parameters

pin: gpio pin ID (1,13,15,16,17,98,99).
There are 5 ADC ports supported on WiFiMCU: D1, D13, D15, D16, D17.
If pin=98, internal MCU temperature is returned in °C
If pin=99, internal voltage reference value is returned (typical 1,21 V)

samples: optional; number of samples to calculate the average (2~128)

Returns

data: if succeed, data between 0.000 ~ 3.300 is returned, else nil.
std: standard deviation, floating point value

Examples

```
>=adc.readV(1)
1.7693    0.00000
>=adc.readV(1, 128)
3.023873    0.004356
>=adc.readV(98, 128)
43.15285    2.14356
```

adc.setref()

Description

Sets the reference voltage value, default is 3.300 V.
Fith no parameter, calculates and sets the reference value from MCU’s internal valtage reference (1.21V),

Syntax

refval = adc.setref([value])

Parameters

value: optional; Voltage reference value (3.0 ~ 3.6 V)

Returns

refval: new voltage reference value
Note: if no parameter is given, returned value represents measured Vdd voltage

Examples

```
>=adc.setref(3.3289)
3.3289
>=adc.setref()
3.3389
```

adc.setautocal()

Description

Sets autocalibration ON or OFF.
If autocalibration is ON, voltage reference value is measured and set before every ADC reading.

Syntax

adc.setautocal(flag)

Parameters

flag: 0 = autocalibration OFF (default); 1 = autocalibration ON

Returns

nil

Examples

```
>adc.setautocal(1)
>adc.setautocal(0)
```

UART Module

Only one hardware UART is supported in WiFiMCU. The GPIO pin is D8(RX1), D9(TX1).
One software emulated UART is also supported on any GPIO pins.

Function list

uart.setup()	Setup uart parameters: buadrate, databits, parity, stopbits.
uart.on()	Register the callback functions for uart events
uart.send()	Send data via uart
uart.recv()	Get received bytes as string
uart.recvstat()	Get number of received bytes and error status.
uart.deinit()	Deinitialize UART, free used pins

Constant

null

Software emulated UART is full duplex, supported baud rates 1200~115200.
At 115200 bd, the sender must be configured for **2** stop bits!

In hardware UART mode **D8** = **RX**; **D9** = **TX**

Warning: hardware UART shares Rx pin with hardware SPI5. You can't use both at the same time.

uart.setup()

Description

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

Syntax

uart.setup(id, baud, parity, databits, stopbits [,txpin, rxpin])

Parameters

id:	uart ID: 1=hardware UART; 2=software emulated UART
baud:	baudrate, such as: 4800, 9600, 115200.
parity:	'n': no parity, 'o': odd parity, 'e': even parity.
databits:	data bits: 5~9
stopbits:	stop bits, 1~2
txpin:	only for software UART: Tx output pin; gpio ID, 0~17
rxpin:	only for software UART: Rx input pin; gpio ID, 0~17

Returns nil

Examples

```
>uart.setup(1,9600, 'n', 8, 1)
>uart.setup(2,38400, 'n', 8, 2, 3, 4)
```

uart.on()

Description

Register the callback functions for uart events.

Syntax

uart.on(id, event ,func_cb)

Parameters

id:	uart ID: 1=hardware UART; 2=software emulated UART.
event:	always "data".
func_cb:	Callback function for the event. When data arrived, the function will be called. Function prototype is: func_cb(len, data) **len** is number of bytes received; **data** is the data received.

Returns nil

Examples

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

uart.send()

Description

Send data via uart.

Syntax

uart.send(id, string1,[num],...[stringn])

Parameters

id:	uart ID: 1=hardware UART; 2=software emulated UART.
string1:	string to send.
[num]:	Optional, number (character code) to send.
[stringn]:	Optional, the nth string to 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), 0x42)
```

uart.recvstat()

Description

Check receive status.

Syntax

uart.recvstat(id)

Parameters

id: uart ID: 1=hardware UART; 2=software emulated UART.

Returns

len: number of bytes received.
err: number of errors (frame errors + parity errors)

Examples

```
>uart.recvstat(2)
0       0
>uart.recvstat(1)
40      0
```

uart.recv()

Description

Get received bytes as string.

Syntax

uart.recv(id, [len])

Parameters

id: uart ID: 1=hardware UART; 2=software emulated UART.
len: Optional: number of bytes to get; default: all bytes

Returns

recstr: string of received bytes; “[nil]” if nothing is received

Examples

```
>uart.recv(2)
Received via software uart
>uart.recvstat(1)
26      0
>uart.recv(1, 16)
Received via har
>uart.recvstat(1)
10      0
>uart.recv(1)
dware uart
```

uart.deinit()

Description

Deinit UART, free GPIO pins used

Syntax

uart.deinit(id)

Parameters

id: id=1 for hardware UART; id=2 for software UART

Returns

nil

Examples

```
>uart.deinit(1)
```

Bit Module

Function List

bit.bnot	Bitwise negation
bit.band	Bitwise AND
bit.bor	Bitwise OR
bit.bxor	Bitwise XOR
bit.lshift	Logical left shift a number
bit.rshift	Logical right shift a number
bit.arshift	Arithmetic right shift a number
bit.bit	Generate a number with a 1 bit (used for mask generation)
bit.set	Set bits in a number
bit.clear	Clear bits in a number
bit.isset	Test if a given bit is set
bit.isclear	Test if a given bit is cleared

Constant

nil

bit.bnot()

Description

Bitwise negation.

Syntax num=bit.bnot(val)

Parameters

val: the number to negation, value is 32 bit width.

Returns

num: the bitwise negated value of the number.

Examples

```
>print("result: "..bit.bnot(0x00000000))
result: -1
```

bit.band()

Description

Bitwise AND.

Syntax

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

Parameters

val1: the first number to AND val1: the second number to AND
valn: the nth number to AND

Returns

num: the bitwise AND of all the arguments.

Examples

```
> print("result: "..bit.band(0xffffffff, 0x000000ff, 0x000000f))
result: 15
```

bit.bor()

Description

Bitwise OR.

Syntax

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

Parameters

val1: the first number to OR val1: the second number to OR
valn: the nth number to OR

Returns

num: the bitwise OR of all the arguments.

Examples

```
> print("result: "..bit.bor(0x00000000, 0x000000ff, 0x000000f)) result: 255
```

bit.bxor()

Description

Bitwise XOR.

Syntax

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

Parameters

val1: the first number to XOR
val1: the second number to XOR

valn: the nth number to XOR

Returns

num: the bitwise XOR of all the arguments.

Examples

```
> print("result: "..bit.bxor(0x00000000, 0x000000ff, 0x000000f))
result: 240
```

bit.lshift()

Description

Logical left shift a number.

Syntax

num= bit.lshift(val, shift)

Parameters

val: the value to shift shift: positions to shift

Returns

num: the number shifted left.

Examples

```
> print("result: "..bit.lshift(0x00000001,8))
result: 256
```

bit.rshift()

Description

Logical right shift a number.

Syntax

num= bit.rshift(val, shift)

Parameters

val: the value to shift shift: positions to shift

Returns

num: the number shifted right.

Examples

```
> print("result: "..bit.rshift(0x00000080,1))
result: 64
```

bit.arshift()

Description

Arithmetic right shift a number.

Syntax

num= bit.arshift(val, shift)

Parameters

val: the value to shift shift: positions to shift

Returns

num: the number arithmetically shifted right.

Examples

```
> print("result: "..bit.arshift(0x00000080,1))
result: 64
```

bit.bit()

Description

Generate a number with a 1 bit (used for mask generation).

Syntax

num= bit.bit(pos)

Parameters

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

Returns

num: the number that only one bit is set to 1 and 0 for the rests.

Examples

```
> print("result: "..bit.bit(8))
result: 256
```

bit.set()

Description

Set bits in a number.

Syntax

num= bit.set(val, pos1,pos2,...,posn)

Parameters

val: the base number.

pos1: first position to be set. pos2: second position to be set. posn: nth position to be set.

Returns

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

Examples

```
> print("result: "..bit.set(0x00000000, 0, 1, 2, 3))
result: 15
```

bit.clear()

Description

Clear bits in a number.

Syntax

num= bit.clear (val, pos1,pos2,...,posn)

Parameters

val: the base number.

pos1: first position to be cleared. pos2: second position to be cleared. posn: nth position to be cleared.

Returns

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

Examples

```
> print("result: "..bit.clear(0x0000000f, 0, 1, 2, 3))
result: 0
```

bit.isset()

Description

Test if a given bit is set.

Syntax

res= bit.iset (val, pos)

Parameters

val: the value number to be test pos: bit position.

Returns

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

Examples

```
>=bit.isset(0x0000000f, 1) true >=bit.isset(0x0000000f, 5)
false
```

bit.isclear()

Description

Test if a given bit is cleared.

Syntax

res= bit.isclear (val, pos)

Parameters

val: the value number to be test

pos: bit position.

Returns

res: true if the bit at the given position is 0, false otherwise.

Examples

```
>=bit.isclear(0x0000000f, 1)
false
>=bit. isclear (0x0000000f, 5)
true
```

Sensor Module

Function List

sensor.dht11.init	Init DHT11/22, Assign the GPIO Pin for DHT11/22.
sensor.dht11.get	Get the DHT11/22 temperature and humility values
sensor.ds18b20.init	Init DS18B20, Assign the GPIO Pin for 1-wire.
sensor.ds18b20.gettemp	Start temperature measurement and get the temperature
sensor.ds18b20.search	Search for DS18B20 1-wire devices
sensor.ds18b20.setres	Set DS18B20 resolution (9,10,11,12 bit)
sensor.ds18b20.getres	Get current DS18B20 resolution (9,10,11,12 bit)
sensor.ds18b20.getrom	Get DS18B20 ROM values (returns 8 element table)
sensor.ow.init	Init 1-wire device, Assign the GPIO Pin for 1-wire.
sensor.ow.search	Search for 1-wire devices

Constant

sensor.ds18b20.DS18B20_RES9 DS18B20 9 bit resolution
sensor.ds18b20.DS18B20_RES10 DS18B20 9 bit resolution
sensor.ds18b20.DS18B20_RES11 DS18B20 9 bit resolution
sensor.ds18b20.DS18B20_RES12 DS18B20 9 bit resolution

sensor.dht11.init()

Description

Init DHT11 sensor. Assign the GPIO Pin for dht11.

Syntax

res = sensor.dht11.init(pin,type)

Parameters

pin: gpio ID, 0~17.
type: optional, DHT type: 0=DHT11; 1=DHT22 (default: DHT11)

Returns

res: true if dht11/22 initialization successful, nil otherwise.

Examples

```
>=sensor.dht11.init(7)
true
```

sensor.dht11.get()

Description

Get the DHT11/DHT22 temperature and humility value.

Syntax

temp, hum, stat = sensor.dht11.get()

Parameters nil

Returns

temp:	temperature measured by DHT (deg for DHT11; 1/10 deg for DHT22).
hum:	humidity measured by DHT (% for DHT11; 1/10 % for DHT22).
stat:	conversion status (0=OK; 1=read err; 2=csum err; 3=check err; 4=not init)

Examples

```
> =sensor.dht11.get()
26 65 0
```

sensor.ds18b20.init()

Description

Init ds18b20 sensor. Assign the GPIO Pin for 1-wire.

Syntax

res = sensor.ds18b20.init(pin)

Parameters

pin: gpio ID, 0~17.

Returns

res: true if ds18b20 initialization successfully, false otherwise.

Examples

```
>=sensor.ds18b20.init(7)
true
```

sensor.ds18b20.search()

Description

Search for DS18B20 1-wire devices.

Syntax

res = sensor.ds18b20.search()

Parameters

nil

Returns

res: Number of found DS18B20 devises.

Examples

```
>=sensor.ds18b20.search()
1
```

sensor.ds18b20.gettemp()

Description

Start temperature measurement and get the temperature.

Syntax

tmp, n = sensor.ds18b20.gettemp(dev)

Parameters

dev: ds18b20 device number, 1~num of detected devices with sensor.ds18b20.search()

Returns

tmp: temperature
n: duration of the measurement in msec (depends on current ds18b20 resolution)

Examples

```
> = sensor.ds18b20.gettemp(1)
22.1875 591
```

sensor.ds18b20.setres()

Description

Set DS18B20 resolution (9,10,11,12 bit).

Syntax

sensor.ds18b20.setres(dev, res)

Parameters

dev: ds18b20 device number, 1~num of detected devices with sensor.ds18b20.search()
res: resolution (9,10,11,12 bit)

Returns

nil

Examples

```
> = sensor.ds18b20.setres(1,10)
```

sensor.ds18b20.getres()

Description

Get DS18B20 current resolution.

Syntax

Res = sensor.ds18b20.getres(dev)

Parameters

dev: ds18b20 device number, 1~num of detected devices with sensor.ds18b20.search()

Returns

res: resolution (9,10,11,12 bit)

Examples

```
> = sensor.ds18b20.getres(1)
10
```

sensor.ds18b20.getrom()

Description

Get DS18B20 ROM values (returns 8 element table).

Syntax

rom = sensor.ds18b20.getrom(dev)

Parameters

dev: ds18b20 device number, 1~num of detected devices with sensor.ds18b20.search()

Returns

rom: Table with 8 ROM values

Examples

```
> rom=sensor.ds18b20.getrom(1); for i=1,9,1 do print(dsrom[i]) end
40
142
106
200
0
0
0
110
```

sensor.ow.init()

Description

Init 1-wire device. Assign the GPIO Pin for 1-wire.

Syntax

res = sensor.ow.init(pin)

Parameters

pin: gpio ID, 0~17.

Returns

res: true if 1-wire initialization successfully, false otherwise.

Examples

```
>=sensor.ow.init(7)
true
```

sensor.ow.search()

Description

Search for any 1-wire devices.

Syntax

res = sensor.ow.search()

Parameters

nil

Returns

res: Number of found 1-wire devises.

Examples

```
>=sensor.ds18b20.search()
1
```

SPI Module

Function List

spi.setup	Init spi, assign GPIO pins
spi.write	Write data via spi interface, data can be multi numbers, string or lua table
spi.read	Read data from spi interface
spi.deinit	Deinitializes the SPI, free gpio pins

Constant

spi.BITS_8	8 Bits data length
spi.BITS_16	16 Bits data length

In hardware SPI mode **D8** = **MOSI**; **D7** = **MISO**; **D16** = **SCK**

Warning: hardware SPI shares MOSI pin with hardware UART. You can't use both at the same time.

spi.setup()

Description

Initialize SPI. SPI module works in MASTER mode.

Syntax

spi.setup(id, config)

Parameters

id: 0 for software SPI; 2 for hardware SPI5

config: Lua table with spi configuration parameters:

mode=spi_mode	0,1,2,3
speed=spi_speed	spi clock frequency in kHz; 100~5000 for software spi (>5000 selects max possible speed) 400~50000 for hardware spi
cs=pin	gpio ID, 0~17
rw=flag	*optional*; 1 reads from MOSI while writing; 1 no read while write
The following parameters are only for software SPI:	
sck=pin	gpio ID, 0~17 used for SCK
mosi=pin	gpio ID, 0~17 used for MOSI
miso=pin	*optional*; gpio ID, 0~17 used for MISO

Returns

0 is succes; error code if not

Examples

```
-- hardware SPI5
>res = spi.setup(2,{mode=3, cs=12, speed=15000})
-- software SPI
>res = spi.setup(0,{mode=3, cs=12, speed=1000,sck=2, mosi=4})
```

spi.write()

Description

Write data via spi interface. Data can be multi numbers, string or lua table

Syntax

ret = spi.write(id, databits, data1, [data2],...,[datan])

Parameters

id: 0 for software SPI; 2 for hardware SPI5
databits: write databits. spi. BITS_8 or spi. BITS_16.
data1: should be 0<data1< 255 in spi. BITS_8 mode
or 0<data2<65535 in spi. BITS_16 mode.

data2: optional.

datan: optional.

Returns

ret: The number of data written.

Examples

```
>res = spi.setup(0,{mode=3, cs=12, speed=1000,sck=2, mosi=4})
>ret = spi.write(0, 0xAA)
```

spi.read()

Description

Read data via spi interface.

Syntax

ret = spi.read(id, databits, num)

Parameters

id: 0 for software SPI; 2 for hardware SPI5
databits: write databits. spi. BITS_8 or spi. BITS_16.
num: the number of data to read.

Returns

ret: the Lua table of read data.

Examples

```
> ret = spi.read(0, 2)
> print(ret[1]); print(ret[2])
```

spi.deinit()

Description

Deinitializes the SPI, free gpio pins.

Syntax

ret = spi.deinit(id)

Parameters

id: 0 for software SPI; 2 for hardware SPI5

Returns

ret: 0 on success; err code if error

Examples

```
> ret = spi.deinit(2)
```

I2C Module

Function List

i2c.setup	Init i2c, assign GPIO pins for software i2c
i2c.deinit	Deinit i2c, free GPIO pins
i2c.write	Write data via i2c interface, data can be multi numbers, strings or lua table
i2c.read	Read data from i2c interface

I2C pins are open drain, use pullup resistors (typical values 4.7K).

Constant

nil

i2c.setup()

Description

Software I2C:

Any GPIO pin can be set as SDA/SCL.
Standard I2C mode is used (100k)
7-bit addressing mode is used

Hardware I2C:

SDA = **D11**; **SCL** = **D3**
Standard (100k) or Fast (400k) mode can be selected.
7-bit or 10-bit addressing mode can be select

Syntax

i2c.setup(id, pinSDA, pinSCL)
I2c.setup(id, adr_mode, speed_mode)

Parameters

id:	id=0 for software I2C; id=1 for hardware I2C
pinSDA:	GPIO Pin 0~17 to be used as SDA (software I2C only)
pinSCL:	GPIO Pin 0~17 to be used as SCL (software I2C only)
adr_mode:	0=7-bit address mode; 1=10-bit address mode (hw I2C only)
speed_mode:	0=Standard (100k); 1=Fast (400k) (hw I2C only)

Returns nil

Examples

```
> i2c.setup(0, 11, 3)
> i2c.setup(1, 0, 1)
```

i2c.deinit()

Description

Deinit i2c, free GPIO pins

Syntax

i2c.deinit(id)

Parameters

id: id=0 for software I2C; id=1 for hardware I2C

Returns

nil

Examples

```
>i2c.deinit(0)
```

i2c.write()

Description

Write data via i2c interface, data can be multi numbers, strings or lua table

Syntax

ret = i2c.write(id, addr, data1, [data2],...,[datan])

Parameters

id: id=0 for software I2C; id=1 for hardware I2C

addr: device address

data1: number 0~255, lua string or lua table

data2: optional.

datan: optional.

Returns

ret: negative number if error or number of bytes written

Examples

```
> ret = i2c.write(0, 0x27, 0x0F)
> = i2c.write(1, 0x3C, “abcdefg12345”, 13, {65,50,32})
16
```

i2c.read()

Description

Read data from i2c interface

Syntax

stat,ret = i2c.read(id, addr, n)

Parameters

id:	id=0 for software I2C; id=1 for hardware I2C
addr:	device address
n:	the number of data bytes to read.

Returns

stat:	negative number if err; number of bytes read if OK.
ret:	number: byte read if n=1 Lua table containing n bytes read

Examples

```
> stat, ret = i2c.read(0, 0x27, 1)
> print(ret)
> stat, ret = i2c.read(1, 0x51, 8)
```

RTC Module

Function List

rtc.getasc	Get text representation of current date&time from RTC
rtc.get	Get Lua Table with second, minute, hour, weekday, date, month, year from RTC
rtc.getstrf	Get formated string representing the current datetime
rtc.set	Set RTC second, minute, hour, weekday, date, month, year
rtc.standby	Put CPU to standby or stop mode for specified number of seconds
rtc.standbyUntil	Put CPU to stabdby or stop mode until specified time

rtc.getasc()

Description

Get text representation of current date&time from RTC

Syntax

strtime = rtc.getasc()

Parameters

nil

Returns

string: Current date & time

Examples

```
 > =rtc.getasc()
Wed Nov  4 15:56:06 2015
 >
```

rtc.getstrf()

Description

Get formated string representing the current datetime from RTC

Syntax

strtime = rtc.getstrf(format)

Parameters

Format: format string, default: "%Y-%m-%d %H:%M:%S"

fmt	Replaced by	Example
%a	Abbreviated weekday name *	Thu
%A	Full weekday name *	Thursday
%b	Abbreviated month name *	Aug
%B	Full month name *	August
%c	Date and time representation *	Thu Aug 23 14:55:02 2001
%C	Year divided by 100 and truncated to integer (00-99)	20
%d	Day of the month, zero-padded (01-31)	23
%D	Short MM/DD/YY date, equivalent to %m/%d/%y	08/23/01
%e	Day of the month, space-padded (1-31)	23
%F	Short YYYY-MM-DD date, equivalent to %Y-%m-%d	2001-08-23
%g	Week-based year, last two digits (00-99)	01
%G	Week-based year	2001
%h	Abbreviated month name * (same as %b)	Aug
%H	Hour in 24h format (00-23)	14
%I	Hour in 12h format (01-12)	02
%j	Day of the year (001-366)	235
%m	Month as a decimal number (01-12)	08
%M	Minute (00-59)	55
%n	New-line character ('\n')	
%p	AM or PM designation	PM
%r	12-hour clock time *	02:55:02 pm
%R	24-hour HH:MM time, equivalent to %H:%M	14:55
%S	Second (00-61)	02
%t	Horizontal-tab character ('\t')	
%T	ISO 8601 time format (HH:MM:SS), equivalent to %H:%M:%S	14:55:02
%u	ISO 8601 weekday as number with Monday as 1 (1-7)	4

%x	Date representation *	08/23/01
%X	Time representation *	14:55:02
%y	Year, last two digits (00-99)	01
%Y	Year	2001
%%	A % sign	%

Returns

string: Formated current date & time

Examples

```
> =rtc.getstrf("%c")
Sun Nov 22 18:12:02 2015
> =rtc.getstrf("%H:%M:%S Date: %m/%d")
18:14:25 Date: 11/22
```

rtc.get()

Description

Get second, minute, hour, weekday, date, month, year from RTC

Syntax

curtime = rtc.get()

Parameters

nil

Returns

Curtime: Lua table with current second, minute, hour, weekday, date, month, year

Examples

```
> ct=rtc.get()
> for i=1,7,1 do print(ct[i]); end
11
59
17
0
22
11
2015
```

rtc.set()

Description

Set RTC second, minute, hour, weekday, date, month, year

Syntax

res=rtc.set(timetbl)

Parameters

timetbl: Lua table with second, minute, hour, weekday, date, month, year

Returns

res: 1 if date&time are set or 0 if error

Examples

```
> =rtc.set(53,57,15,3,4,11,15)
OK
>
```

rtc.standby()

Description

Put CPU to power save mode for specified number of seconds
Note: in STANDBY mode, CPU is RESET on wakeup.

Syntax

rtc.standby(mode, numsec)

Parameters

mode: power save mode (0 for standby; 1 for stop)
numsec: number of seconds to staj in standby

Returns

nil, after wake up CPU resets

Examples

```
> rtc.standby(0,5)
Going to STANDBY MODE...
Wake up in 5 second(s)

(RESET)
WiFiMCU Lua starting...(Free memory 65544 bytes)
Current Time: Wed Nov  4 16:11:47 2015

[ Ver. 0.9.6_lobo_0.1 WiFiMCU Team, modified by LoBo @2015 ]

Executing init.lua...
>

> rtc.standby(1,5)
Going to STOP MODE...
Wake up in 5 second(s)

Back from power save mode.
>
```

rtc.standbyUntil()

Description

Put CPU to powersave mode until specified time
Note: in STANDBY mode, CPU is RESET on wakeup.

Syntax

rtc.standbyUntil(mode, time)

Parameters

mode: power save mode (0 for standby; 1 for stop)
time: Lua table with hour, minute, second to wake up at

Returns

nil, after wake up CPU resets

Examples

```
> rtc.standbyUntil(0, {16,16,5})
Going to STANDBY MODE...
Wake up at 16:16:05

(RESET)
WiFiMCU Lua starting...(Free memory 65544 bytes)
Current Time: Wed Nov  4 16:16:05 2015

[ Ver. 0.9.6_lobo_0.1 WiFiMCU Team, modified by LoBo @2015 ]

Executing init.lua...
>

> rtc.standbyUntil(1, {16,16,5})
Going to STOP MODE...
Wake up at 16:16:05

Back from power save mode.
>
```

OLED Module

Function List

oled.init	Initialize the oled display,
oled.clear	Clear the screen
oled.write	Write strings and|or numbers to display
oled.writechar	Write one character to display
oled.fontsize	Select the font size
oled.charspace	Define the space between characters
oled.inverse	Select normal or inverted write
oled.fixedwidth	Set fixed width or proportional character printing
oled.seti2caddr	Set i2c address if using i2c interface

The module supports operations with small (0.96” ~ 1.3”) oled displays based on SSD1306 controller, using the 4-wire SPI interface or I2C interface.

oled.init()

Description

Initialize the oled display and clear the screen.
You must initialize the **SPI** or **I2C** interface first.

Syntax

res = oled.init(iid, DCpin [,init_param]) SPI interface
res = oled.init(iid [,init_param]) I2C interface

Parameters

iid: interface id:
0 software SPI interface
1 hardware SPI interface
3 software I2C interface
4 hardware I2C interface

DCpin: gpio ID, 0~17 used for DC control, **only for SPI interface**
init_param: optional; lua table containing SSD1306 initialization parameters

Returns

res: 0 on success, error code on error

Examples

```
-- hardware spi
>spi.setup(2,{mode=3, cs=12, speed=15000})
>res = oled.init(2,14)
-- software spi
>spi.setup(0,{mode=3, cs=12, mosi=9, sck=16, speed=500})
>oled.init(0,14)
-- software i2c interface
>i2c.setup(0, 11, 3)
>oled.init(3)
-- hardware i2c interface
>initp =
{0xAE,0xD5,0x80,0xA8,0x3F,0xD3,0x00,0x40,0x8D,0x14,0x20,0x00,0x22,0,7,0x21,0,
127,0xA1,0xC8,0xDA,0x12,0x81,0xCF,0xD9,0xF1,0xDB,0x40,0xA4,0xA6,0xAF}
>i2c.setup(1, 0, 1, initp)
>initp = nil
>oled.init(4)
```

oled.clear()

Description

Clear screen.

Syntax

oled.clear()

Parameters

nil

Returns

nil

Examples

```
> oled.clear()
```

oled.write()

Description

Write strings and|or numbers to display.

Syntax

oled.write(x, y, ndec, data1, [data2], ... [datan])

Parameters

x: x position (column; 0~127)
y: y positoin (row; 0~7)
ndec: number of decimal places if number data is float; 0 to print integer)
data1: number or string to write to the display
data2: optional
datan: optional

Returns

nil

Examples

```
>oled.write(0,0,0,”WiFiMCU”)
>t=2.3456
>oled.write(8,2,1,”Temp=”, t)
```

oled.writechar()

Description

Write single character to display.

Syntax

oled.write(x, y, char)

Parameters

x: x position (column; 0~127)
y: y positoin (row; 0~7)
char: character code

Returns

nil

Examples

```
>oled.writechar(16,5,0x42)
```

oled.fontsize()

Description

Set the font size (height). At the moment can be only 8 or 16

Syntax

oled.fontsize(size)

Parameters

size: new font size (8 or 16)

Returns

nil

Examples

```
>oled.fontsize(16)
```

oled.charspace()

Description

Set additional space between characters in pixels.

Syntax

oled.crarspace(chr_spc)

Parameters

chr_spc: new intercharacter space (0~8)

Returns

nil

Examples

```
>oled.charspace(1)
```

oled.inverse()

Description

Set normal (light on dark) or inverse (dark on light) display.

Syntax

oled.inverse(flag)

Parameters

flag: o for normal, 1 for inverse

Returns

nil

Examples

```
>oled.inverse(1)
```

oled.fixedwidth()

Description

Set fixed width or proportional character printing.

Syntax

oled.fixedwidth(flag)

Parameters

flag: 0 to print proportional character, 1 for fixed width

Returns

nil

Examples

```
>oled.fixedwidth(1)
>oled.write(0,0,0,"IIII\r\nMMMM")
IIII
MMMM
>oled.fixedwidth(0)
>oled.write(0,0,0,"IIII\r\nMMMM")
IIII
MMMM
```

oled.seti2caddr()

Description

Set i2c addres if using the i2c interface.

Syntax

oled.seti2caddr(addr)

Parameters

addr: i2c address, 7-bit; default 0x3C

Returns

nil

Examples

```
>oled.seti2caddr(0x3c)
```

LCD Module

Function List

lcd.init	Initialize the display
lcd.clear	Clear the screen
lcd.write	Write strings and|or numbers to display
lcd.on	Turn display on
lcd.off	Turn display off
lcd.setfont	Set the font used for write function
lcd.getscreensize	Get current screen size
lcd.getfontsize	Get current font size in pixels
lcd.getfontheight	Get current font height in pixels
lcd.fixedwidth	Set fixed width or proportional character printing
lcd.setrot	Set text rotation (angle)
lcd.setorient	Set display orientation, default PORTRAIT
lcd.setwrap	Set line wrap for lcd.write() function
lcd.setcolor	Set foreground and background colors
lcd.settransp	Set transparency for character printing
lcd.setfixed	Force fixed width printing of proportional fonts
lcd.setclipwin	Set the coordinates of the clipping window
lcd.resetclipwin	Reset clipping window to full screen
lcd.invert	Set inverted/normal colors
lcd.putpixel	Puts pixel on screen
lcd.line	Draw line
lcd.rect	Draw rectangle
lcd.triangle	Draw triangle
lcd.circle	Draw circle
lcd.image	Show image from file
lcd.hsb2rgb	Converts HSB color values to 16-bit RGB value

Constant

lcd.PORTRAIT	Default orientation
lcd.PORTRAIT_FLIP	Orientation flipped portrait
lcd.LANDSCAPE	Orientation landscape
lcd.LANDSCAPE_FLIP	Orientation flipped landscape
lcd.CENTER	Center text (write function)
lcd.RIGHT	Right allign text (write function)

lcd.LASTX	Continue writing at last X position (write function)
lcd.LASTY	Continue writing at last Y position (write function)
lcd.FONT_SMALL	Small fixed width font (8x8)
lcd.FONT_BIG	Big fixed width font (16x16)
lcd.FONT_DEJAVU12	Proportional font DejaVue 12
lcd.FONT_DEJAVU18	Proportional font DejaVue 18
lcd.FONT_DEJAVU24	Proportional font DejaVue 24
lcd.FONT_7SEG	7 segment vector font (digits,'-','.',':','deg' only
lcd.ST7735	ST7735 based display, type #0
lcd.ST7735B	ST7735 based display, type #1
lcd.ST7735G	ST7735 based display, type #2
lcd.ILI9341	ILI9341 based display
lcd.BLACK	Colors
lcd.NAVY	
lcd.DARKGREEN	
lcd.DARKCYAN	
lcd.MAROON	
lcd.PURPLE	
lcd.OLIVE	
lcd.LIGHTGREY	
lcd.DARKGREY	
lcd.BLUE	
lcd.GREEN	
lcd.CYNAN	
lcd.RED	
lcd.MAGENTA	
lcd.YELLOW	
lcd.WHITE	
lcd.ORANGE	
lcd.GREENYELLOW	
lcd.PINK	

The module supports operations with TFT SPI displays.
Displays based on ST7735 and ILI9341 controllers, using the 4-wire SPI interface are supported.

Using hardware SPI is recommended, the speed much higher.

Use the following wiring to connect the display:

WiFiMCU	Pin		Display
MOSI	any (hw spi D8)	->	SDI(MOSI)
CLK	any (hw spi D16)	->	SCK
CS	any	->	CS
DC	any	->	DC
			RESET, not used, pullup (4.7K) to 3.3V
			SDO (MISO), not used

lcd.init()

Description

Initialize the tft display and clear the screen.
You must initialize the SPI interface first.

Syntax

res = lcd.init(spi_id, DCpin, type [,orient])

Parameters

spi_id:	id of the SPI interface to be used for lcd
DCpin:	gpio ID, 0~17 used for DC (data/command) control
type:	display type, **0**,**1**,**2** (probably 1 will work best) for ST7735 or **3** for ILI9341 You can use defined constants ST7735, ST7735B, ST7735G, ILI9341
orient:	optional, display orientation (default: PORTRAIT)

Returns

res: 0 on success, error code on error

Examples

```
-- hardware spi with 50 MHz clock
>spi.setup(2,{mode=3, cs=12, speed=50000})
>res = lcd.init(2,14,1,lcd.LANDSCAPE)
-- software spi with ~5 Mhz clock
>spi.setup(0,{mode=3, cs=12, mosi=9, sck=16, speed=5000})
> res = lcd.init(0,14,1,PORTRAIT_FLIP)
-- hardware spi with 50 MHz clock, ILI9341 display
>spi.setup(2,{mode=3, cs=12, speed=50000})
>res = lcd.init(2,14,3,lcd.PORTRAIT)
```

lcd.clear()

Description

Clear screen to default or specified color.

Syntax

lcd.clear([color])

Parameters

color optional; fill the screen with color (default: BLACK)

Returns

nil

Examples

```
> lcd.clear(lcd.BLUE)
> lcd.clear()
```

lcd.off()

Description

Turns the display of, preserve power. Back light has to be turned off separately.

Syntax

lcd.off()

Parameters

nil

Returns

nil

Examples

```
> lcd.off()
```

lcd.on()

Description

Turns the display on.

Syntax

lcd.on()

Parameters

nil

Returns

nil

Examples

```
> lcd.on()
```

lcd.invert()

Description

Set inverted/normal colors.

Syntax

lcd.invert(inv)

Parameters

inv 0: inverted colors off; 1: inverted colors on

Returns

nil

Examples

```
> lcd.invert(0)
```

lcd.setorient()

Description

Set display orientation.

Syntax

lcd.setorient(orient)

Parameters

orient one of display orientation constants
PORTRAIT, PORTRAIT_FLIP, LANSCAPE, LANDSCAPE_FLIP

Returns

nil

Examples

```
> lcd.orient(lcd.LANDCSAPE)
> lcd.orient(PORTRAIT_FLIP)
```

lcd.setclipwin()

Description

Sets the clipping area coordinates. All writing to screen is clipped to that area.
Starting x & y in all functions will be adjusted to the clipping area.
This setting has no effect on lcd.image function.

Syntax

lcd.setclipwin(x1, y1, x2, y2)

Parameters

x1,y1 upper left point of the clipping area
x1,y1 bottom right point of the clipping area

Returns

nil

Examples

```
> lcd.setclipwin(20,20,220,200)
```

lcd.resetclipwin()

Description

Resets the clipping are coordinates to default full screen.

Syntax

lcd.resetclipwin()

Parameters

nil

Returns

nil

Examples

```
> lcd.resetclipwin()
```

lcd.setrot()

Description

Set text rotation (angle) for lcd.write() function. Has no effect on FONT_7SEG.

Syntax

lcd.setrot(rot)

Parameters

rot rotation angle (0~360)

Returns

nil

Examples

```
> lcd.rot(90)
> lcd.write("Ratated text")
```

lcd.settransp()

Description

Set transparency when writing the text. If transparency is on, only text foreground color is shown.

Syntax

lcd.settransp(transp)

Parameters

transp 0: transparency off; 1: transparency on

Returns

nil

Examples

```
> lcd.settransp(1)
```

lcd.setwrap()

Description

Set line wrapping writing the text. If wrapping is on, text will wrap to new line, otherwise it will be clipped.

Syntax

lcd.setwrap(wrap)

Parameters

wrap	0: line wrap off; 1: line wrap on

Returns

nil

Examples

```
> lcd.setwrap(1)
```

lcd.setfixed()

Description

Forces fixed width print of the proportional font.

Syntax

lcd.setwrap(force)

Parameters

force	0: force fixed width off; 1: force fixed width on

Returns

nil

Examples

```
> lcd.setfixed(1)
```

lcd.setcolor()

Description

Set the color used when writing characters or drawing on display.

Syntax

lcd.setcolor(color[,bgcolor])

Parameters

color	foreground color for text and drawing
bgcolor	optional; background color for writing text

Returns

nil

Examples

```
> lcd.setcolor(lcd.YELLOW)
> lcd.setcolor(lcd.ORANGE, lcd.DARKGREEN)
```

lcd.setfont()

Description

Set the font used when writing the text to display.
Six fonts are available:
FONT_SMALL (default, fixed width 8x8),
FONT_BIG (fixed width 16x16)
FONT_DEJAVU12,FONT_DEJAVU18,FONT_DEJAVU24 (proportional fonts)
FONT_7SEG (vector font, imitates 7 segment displays).

7-segment font is the vector font for which any size can be set (distance between bars and the bar width). Only characters **0**,**1**,**2**,**3**,**4**,**5**,**6**,**7**,**8**,**.**,**-**,**:**,**/** are available. Character '**/**' draws the degree sign.

Syntax

lcd.setfont(font [,size, width])

Parameters

font	one of the available fonts
size	optional; only for FONT_7SEG, distance between bars (default: 12; min=6; max=40)
width	optional; only for FONT_7SEG, bar width (default: 2; min=1; max=12 or size/2)

Returns

nil

Examples

```
> lcd.setfont(lcd.FONT_BIG)
> lcd.setfont(lcd.FONT_7SEG, 20, 4)
```

lcd.getfontsize()

Description

Get current font size in pixels. Useful if FONT_7SEG is used to get actual character width and height.

Syntax

lcd.getfontsize()

Parameters

nil

Returns

xsize width of the font character in pixels.
For the proportional fonts, maximal char width will be returned
ysize height of the font character in pixels

Examples

```
> lcd.getfontsize()
   8   12
```

lcd.getfontheight()

Description

Get current font height in pixels.

Syntax

lcd.getfontheight()

Parameters

nil

Returns

ysize height of the font character in pixels

Examples

```
> lcd.setfont(lcd.FONT_BIG)
> lcd.getfontsize()
   16
```

lcd.getscreensize()

Description

Get current screen size (width & height) in pixels.

Syntax

lcd.getscreensize()

Parameters

nil

Returns

xsize width of the screen in pixels
ysize height of the screen in pixels

Examples

```
> lcd.getscreensize()
   240   320
```

lcd.putpixel()

Description

Draws pixel on display at coordinates (x,y) using foreground or given color

Syntax

lcd.putpixel(x, y [, color])

Parameters

x, y	coordinates of pixel
color	optional: pixel color (default: current foreground color)

Returns

nil

Examples

```
> lcd.putpixel(10,10)
> lcd.putpixel(20,40,lcd.GREEN)
```

lcd.line()

Description

Draws line from (x1,y1) to (x2,y2) using foreground or given color

Syntax

lcd.line(x1, y1, x2, y2 [,color])

Parameters

x1,y1	coordinates of line start point
x1,y1	coordinates of line end point
color	optional: line color (default: current foreground color)

Returns

nil

Examples

```
> lcd.line(0,0,127,159)
> lcd.line(20,40,80,10,lcd.ORANGE)
```

lcd.rect()

Description

Draws rectangle at (x,y) w pixels wide, h pixels high, with given color. If the fill color is given, fills the rectangle.

Syntax

lcd.rect(x, y, w, h, color [,fillcolor])

Parameters

x, y	coordinates of the upper left corner of the rectangle
w	width of the rectangle
h	height of the rectangle
color	rectangle outline color
fillcolor	optional: rectangle fill color

Returns

nil

Examples

```
> lcd.rect(10,10,100,110,lcd.RED)
> lcd.rect(0,0,128,160,lcd.ORANGE,lcd.YELLOW)
```

lcd.circle()

Description

Draws circle with center at (x,y) and radius r, with given color. If the fill color is given, fills the circle.

Syntax

lcd.circle(x, y, r, color [,fillcolor])

Parameters

x, y	coordinates circle center
r	radius of the circle
color	circle outline color
fillcolor	optional: circle fill color

Returns

nil

Examples

```
> lcd.circle(64,80,20,lcd.RED)
> lcd.circle(50,60,30,lcd.ORANGE,lcd.YELLOW)
```

lcd.triangle()

Description

Draws triangle between three given points, with given color. If the fill color is given, fills the triangle.

Syntax

lcd.triangle(x1, y1, x2, y2, x3, y3, color [,fillcolor])

Parameters

x1, y1, x2, y2, x3, y3	coordinates of the 3 triangle points
color	triangle outline color
fillcolor	optional: triangle fill color

Returns

nil

Examples

```
> lcd.triangle(50,20,80,100,20,100,lcd.RED)
> lcd.triangle(50,20,80,100,20,100,lcd.RED, lcd.WHITE)
```

lcd.write()

Description

Write strings and|or numbers to display. Rotation of the displayed text can be set with lcd.setrot() function.
Two special characters are allowed in strings:
'**\r**' CR (0x0D), clears the display to EOL
'**\n**' LF (ox0A), continues to the new line, x=0

Syntax

lcd.write(x, y, data1, [data2, ... datan])

Parameters

x: x position (column; 0~screen width-1)
Special values can be entered:
lcd.CENTER, centers the text; lcd.RIGHT, right justifies the text
lcd.LASTX, continues from last X position

y: y positoin (row; 0~screen height-1)
Special values can be entered:
lcd.LASTY, continues from last Y position

data1: number or string to write to the display
If simple number is given, integer is printed. The number can be given as a table containing number (float) and number of decimal places.

data2: optional

datan: optional

Returns

nil

Examples

```
>lcd.setcolor(lcd.YELLOW)
>lcd.write(0,0,"WiFiMCU")
>t=2.3456
>lcd.write(8,16,"Temp=", {t,2})
```

lcd.image()

Description

Shows the image from file. The image file must be in raw 16bit format.
Any image can be converted with ***ImageConverter565.exe*** which can be found in *binary* directory on GitHub.
Be careful to give the right image width and height.

Syntax

lcd.image(x, y, xsize, ysize, filename)

Parameters

x:	x position of the image upper left corner
y:	y position of the image upper left corner
xsize:	image xsize (width)
ysize;	image ysize (height)
filename:	name of the row image file

Returns

nil

Examples

```
>lcd.rot(lcd.PORTRAIT)
>lcd.clear()
>lcd.image(0,0,128,96,”wifimcu_128x96.img”)
>lcd.rot(lcd.LANDSCAPE)
>lcd.image(0,0,160,128,”wifimcu_160x128.raw”)
```

lcd.hsb2rgb()

Description

Converts HSB (hue, saturation, brightness) color values to 16-bit RGB value.

Syntax

Color = lcd.hsb2rgb(hue, sat, bri)

Parameters

hue	float, hue value (0.0 ~ 359.9999)
sat	float, saturation value (0.0 ~ 1.0)
bri	brightness value (0.0 ~ 1.0)

Returns

color	16-bit RGB color value

Examples

```
> lcd.circle(50,60,30,lcd.ORANGE,lcd.hsb2rgb(90.0,1.0,0.5))
```

MQTT Module

Function List

mqtt.ver	Get mqtt client version
mqtt.new	Initialize new mqtt client
mqtt.start	Start mqtt client, connect to mqtt broker
mqtt.subscribe	Subscribe mqtt client to the topic
mqtt.unsubscribe	Unsubscribe mqtt client from the topic
mqtt.close	Stop and deinitialize mqtt client
mqtt.publish	Publish the message to the topic
mqtt.closeall	Stop and deinitialize all mqtt clients
mqtt.status	Get the status of mqtt client
mqtt.isactive	Check if mqtt client is initialized
mqtt.isconnected	Check if mqtt client is connected
mqtt.issubscribed	Check if mqtt client is subscribed to the topic
mqtt.on	Register the callback functions for mqtt events
mqtt.debug	Enable/disable mqtt debug messages
mqtt.setretry	Set number of reconnect retries

MQTT originally stood for **MQ T**elemetry **T**ransport, but is now just known as "MQTT". It is a publish/subscribe, extremely simple and lightweight messaging protocol, designed for constrained devices and low-bandwidth, high-latency or unreliable networks. The design principles are to minimize network bandwidth and device resource requirements whilst also attempting to ensure reliability and some degree of assurance of delivery. These principles also turn out to make the protocol ideal of the emerging “machine-to-machine” (M2M) or “Internet of Things” world of connected devices, and for mobile applications where bandwidth and battery power are at a premium.

This implementation has the following properties:

- up to 3 mqtt clients
- each mqtt client can subscribe to up to 5 topics
- automatic reconnect if disconnected
- automatic resubscribe to subscribed topics on reconnect
- optional debug output
- each client has separate callback functions for mqtt events
- client can publish to subscribed or unsubscribed topics

Constant

mqtt.QOS0	Quality of Service level
mqtt.QOS1	
mqtt.QOS2	
mqtt.MAX_CLIENT	Maximum number of clients
mqtt.MAX_TOPIC	Maximum number of topics per client

mqtt.ver()

Description

Get mqtt client version.

Syntax

ver = mqtt.ver()

Parameters

nil

Returns

ver: lua string, mqtt client version

Examples

```
> =mqtt.ver()
0.1.2
>
```

mqtt.new()

Description

Initialize new mqtt client.

Syntax

id = mqtt.new(clientid,user,pass[,keepalive])

Parameters

clientid: string, client id
user: string, user name (can be “” for no user name)
pass: string, password (can be “” for no password)
keepalive: optional; keepalive interval (30~300); default: 60 sec

Returns

id: 0 ~ mqtt.MAX_CLIENT-1 on success, negative number on error

Examples

```
> =mqtt.new("wifimcuclt", "wifimcu", "wifimculobo")
0
> =mqtt.new("wifimcuclient", "", "", 30)
1
```

mqtt.start()

Description

Start mqtt client, connect to mqtt broker. The client must be initialized first.

Syntax

res = mqtt.start(mqttcid, server, port)

Parameters

mqttcid:	mqtt client id
server:	mqtt server (broker) address
port:	mqtt server port (most common 1883)

Returns

res: 0 on success, negative number on error

Examples

```
> mqtt.debug(1)
> clt0=mqtt.new("wifimcuclt", "wifimcu", "wifimculobo")
[mqtt:0] Init: OK.
[mqtt: ] MQTT Thread started.
> =mqtt.start(clt0,"loboris.eu",1883)
0
[mqtt:0] Creating connection [loboris.eu:1883]
[mqtt:0] Network connection OK!
[mqtt:0] Client init OK!
         Client connecting [user: wifimcu, pass: wifimculobo]
         MQTT client connect OK!

-- if the client disconnects, it will be reconnected automatically

[mqtt:0] Yield ERROR
[mqtt:0] Client will reconnect...
         Disconnecting...
[mqtt:0] Creating connection [loboris.eu:1883]
[mqtt:0] Network connection OK!
[mqtt:0] Client init OK!
         Client connecting [user: wifimcu, pass: wifimculobo]
         Client subscribed to topic [test]
         MQTT client connect OK!
```

mqtt.subscribe()

Description

Subscribe mqtt client to the topic. The client must be started first.

Syntax

res = mqtt.subscribe(mqttcid, topic, QoS[,cb_msgarr(topic,message)])

Parameters

mqttcid:	mqtt client id
topic:	string, topic to subscribe to
QoS:	QoS (Quality of Service) level
cb_msgar:	optional; callback function for “Message arrived” mqtt event see *mqtt.on()* for detailes

Returns

res: 0 on success, negative number on error

Examples

```
> mqtt.debug(1)
> =mqtt.subscribe(clt0,"test",mqtt.QOS0)
0
> [mqtt:0] Client subscribed to topic [test]
```

mqtt.usubscribe()

Description

Unsubscribe mqtt client from the topic. The topic must be subscribed first.

Syntax

res = mqtt.unsubscribe(mqttcid, topic)

Parameters

mqttcid:	mqtt client id
topic:	string, topic to unsubscribe from

Returns

res: 0 on success, negative number on error

Examples

```
> mqtt.debug(1)
> =mqtt.unsubscribe(clt0,"test")
0
> [mqtt:0] Client unsubscribed from topic [test]
```

mqtt.publish()

Description

Publish the message to the topic. The client must be started first.

Syntax

res = mqtt.publish(mqttcid, topic, QoS, message)

Parameters

mqttcid:	mqtt client id
topic:	string, topic to subscribe to
QoS:	QoS (Quality of Service) level
message:	string, message to publish

Returns

res: 0 on success, negative number on error

Examples

```
> mqtt.debug(1)
> function cb_messagearrived(topic,len,message) print('** [Message Arrived from loboris]\r\n topic: '..topic..' \r\n message: '..message) end

> =mqtt.subscribe(clt0,"test",mqtt.QOS0)
0
> [mqtt:0] Client subscribed to topic [test]
> mqtt.publish(clt0,"test",mqtt.QOS0, 'hi from wifimcu')
0
> [mqtt:0] Client published to topic [test]
[mqtt: ] messageArrived: [topic: test] [len=15] [hi from wifimcu]
** [Message Arrived from loboris]
 topic: test
 message: hi from wifimcu
```

mqtt.status()

Description

Get the status of mqtt client.

Syntax

active,connected, ntopic = mqtt.status(mqttcid)

Parameters

mqttcid:	mqtt client id

Returns

active:	1 if active (initialized); 0 if not
connected:	1 if connected; 0 if not
ntopic:	number of subscribed topics

Examples

```
> mqtt.debug(0)
> =mqtt.status(0)
1      1      1

> mqtt.debug(1)
> =mqtt.status(0)
  Client #0: Initialized
   clientID: wifimcuclt
   username: wifimcu
   password: wifimculobo

 Message cb: assigned
 Connect cb: not assigned
 Offline cb: not assigned

  Connected: yes
     Server: loboris.eu
       port: 1883

 Subscribed:
    Topic #0: test
1      1      1
>
```

mqtt.close()

Description

Stop and deinitialize mqtt client. If it was the last active client, mqtt thread is terminated.

Syntax

mqtt.close(mqttcid)

Parameters

mqttcid: mqtt client id

Returns

res: 0 on success, negative number on error

Examples

```
> mqtt.debug(1)
> =mqtt.close(0)
> 0
[mqtt:0] Closing...
         Disconnecting...
[mqtt:0] Closed.
[mqtt: ] MQTT Thread terminated.

> =mqtt.close(0)
```

```
Client not initialized
-1
>
```

mqtt.closeall()

Description

Stop and deinitialize all mqtt clients. Mqtt thread is terminated.

Syntax

mqtt.closeall()

Parameters

nil

Returns

nil

Examples

```
> mqtt.debug(1)
> mqtt.closeall()
[mqtt:0] Closing...
         Disconnecting...
[mqtt:0] Closed.
[mqtt: ] MQTT Thread terminated.
```

mqtt.on()

Description

Register the callback functions for mqtt events.

Syntax

mqtt.on(mqttcid, event, cb_function)

Parameters

mqttcid: mqtt client id
event: 'connect' or 'offline' or 'message'

cb_function: callback function for mqtt event:

'connect'	Function prototype is: func_cb(clt). "clt" the client id.
'offline'	Function prototype is: func_cb(clt, flag). "clt" the client id, "flag" 1 if auto reconnect is pending, 2 if reconnect failed

‘message’	Function prototype is: func_cb(topic,len,msg). “topic” the message topic. “len” message length “msg” the message

Returns

nil

Examples

```
>function cb_msgar(topic,len,message) print('[Message Arrived]\r\n topic:
'..topic..' \r\n message: '..message) end
>function cb_conn(clt) print('Connected: #'..clt' \r\n’) end
>function cb_disconn(clt,f) print('Disconected ‘..clt..’,’..f) end

>mqtt.on(0,'message',cb_msgar)
>mqtt.on(0,'connect',cb_conn)
>mqtt.on(0,'message',cb_disconn)
```

mqtt.isactive()

Description

Check if mqtt client is active (initialized).

Syntax

res = mqtt.isactive(mqttcid)

Parameters

mqttcid: mqtt client id

Returns

res: 1 if the client is active, 0 if not

Examples

```
> mqtt.debug(1)
> =mqtt.isactive(0)
0
> =mqtt.new("wifimcuclt", "wifimcu", "wifimculobo")
[mqtt:0] Init: OK.
0
> [mqtt: ] MQTT Thread started.
> =mqtt.isactive(0)
1
```

mqtt.isconnected()

Description

Check if mqtt client is connected to mqtt server (broker).

Syntax

res = mqtt.isconnected(mqttcid)

Parameters

mqttcid: mqtt client id

Returns

res: 1 if the client is connected, 0 if not

Examples

```
> mqtt.debug(1)
> =mqtt.isconnected(0)
0
> =mqtt.start(clt0,"loboris.eu",1883)
0
[mqtt:0] Creating connection [loboris.eu:1883]
[mqtt:0] Network connection OK!
[mqtt:0] Client init OK!
         Client connecting [user: wifimcu, pass: wifimculobo]
         MQTT client connect OK!

> =mqtt.isconnected(0)
1
```

mqtt.issubscribed()

Description

Check if mqtt client is active (initialized).

Syntax

res = mqtt.issubscribed(mqttcid, topic)

Parameters

mqttcid: mqtt client id
topic: topic to check

Returns

res: 1 if the topic is subscribed, 0 if not

Examples

```
> mqtt.debug(1)
> =mqtt.issubscribed(0,"test")
0
> =mqtt.subscribe(clt0,"test",mqtt.QOS0)
0
[mqtt:0] Client subscribed to topic [test]
> =mqtt.issubscribed(0,"test")
1
```

```
> =mqtt.issubscribed(0,"news")
0
```

mqtt.debug()

Description

Enable or disable mqtt debug messages.

Syntax

mqtt.debug(en)

Parameters

en: 0 to disable debug messages (default); 1 to enable

Returns

nil

Examples

```
> mqtt.debug(0)
> =mqtt.start(clt0,"loboris.eu",1883)
0

> mqtt.debug(1)
> =mqtt.start(clt0,"loboris.eu",1883)

[mqtt:0] Creating connection [loboris.eu:1883]
[mqtt:0] Network connection OK!
[mqtt:0] Client init OK!
         Client connecting [user: wifimcu, pass: wifimculobo]
         MQTT client connect OK!
```

mqtt.setretry()

Description

Set number of reconnect retries.

Syntax

mqtt.setretry(n)

Parameters

n: number o reconnect retries (1~100); default 10

Returns

nil

Examples

```
> mqtt.debug(1)
```

```
> =mqtt.new("wifimcuclt", "wifimcu", "wifimculobo")
[mqtt:0] Init: OK.
0
> [mqtt: ] MQTT Thread started.

> mqtt.setretry(3)
> =mqtt.start(0,"none.com",1883)
0
> [mqtt:0] Creating connection [none.com:1883]
[mqtt:0] Network connection ERROR=-1.
[mqtt:0] Creating connection [none.com:1883]
[mqtt:0] Network connection ERROR=-1.
[mqtt:0] Creating connection [none.com:1883]
[mqtt:0] Network connection ERROR=-1.
[mqtt:0] Creating connection [none.com:1883]
[mqtt:0] Network connection ERROR=-1.
         Reconnect failed after 3 retries

>
```