A company of SIM Tech

SIM5360 LUA Application Note V0.05

Document Title:	SIM5360 LUA Application Note
Version:	0.05
Date:	2015-07-01
Status:	Release
Document Control ID:	SIM5360_LUA_Application_Note_V0.05

General Notes

SIMCom offers this information as a service to its customers, to support application and engineering efforts that use the products designed by SIMCom. The information provided is based upon requirements specifically provided to SIMCom by the customers. SIMCom has not undertaken any independent search for additional relevant information, including any information that may be in the customer's possession. Furthermore, system validation of this product designed by SIMCom within a larger electronic system remains the responsibility of the customer or the customer's system integrator. All specifications supplied herein are subject to change.

Copyright

Contents

Version History

Date	Version	Description of change	Author
2014-3-4	V0.01	New version	
2014-11-14	V0.02	Add uart2 for sendtoport os.printport atctl.setport atctl.atcget	
2014-12-29	V0.03	Add the devio lib.	
2015-04-09	V0.04	Remove the pcm lib	
2015-07-01	V0.05	Remove the getmaxmem lib	

Scope

This document presents the AT command of LUA operation and application examples. This document can apply to SIMCom 5360 modules.

1 Introduction

SIMCom LUA extension is a feature that allows customer applications control and drive the module internally and easily.

The SIMCom LUA extension is aimed at light applications where the application was usually done by a small microcontroller that managed some I/O pins and the module through the AT command interface.

By using the LUA extension APIs, customer can write applications using the nice high level LUA language very quickly.

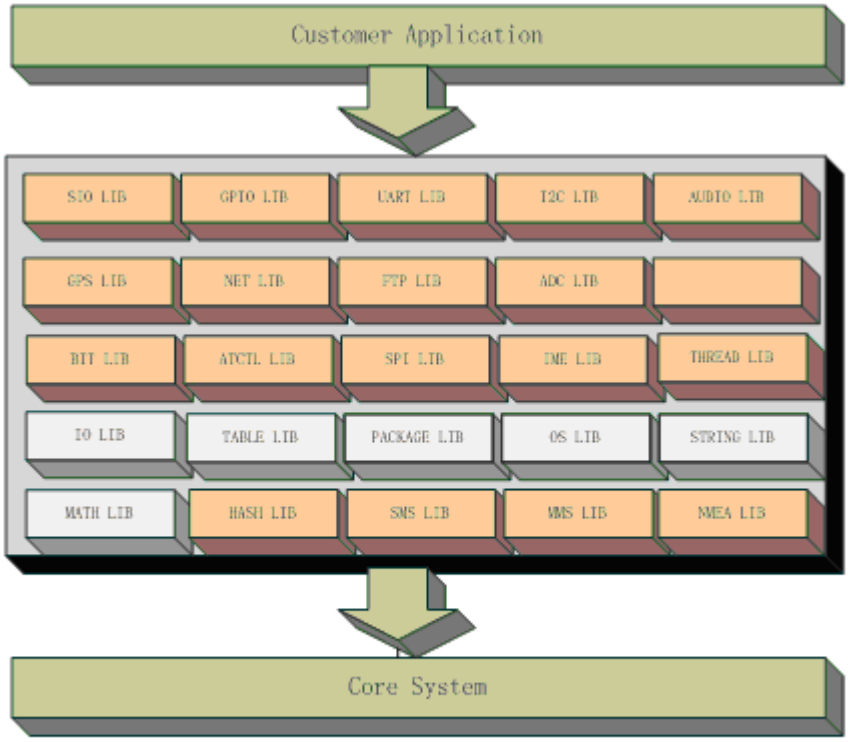
1.1 Features

1. Light script language, easy to be extended.
2. Support both procedure-oriented and object-oriented styles development.
3. Script files are saved in EFS
4. Support AT command operation in script.
5. Support GPIO/IIC/SPI/ADC/Audio/GPS/UART operation in script.
6. Support FTP operation instead of using AT commands.
7. Available memory for script is limited to 3.5M.
8. Script can be started by extern MCU using AT command
9. Script can run automatically when module powers up if the file name is “c:\autorun.lua” or “c:\autorun.out”.
10. The priority of the script task can be adjusted to HIGH, NORMAL, LOW(default).
11. High priority is not recommended to be used when external MCU is connected.
12. Support event operation.
13. Support multi-thread operation with separate thread library.
14. Support timer operation(the maximize number of timer is 30).
15. Support module entering sleep mode when LUA is idle(calling vmsleep, waitevt)

2 SOFTWARE ARCHITECTURE

2.1 Software Organization

The Embedded LUA facility is a software mechanism, it follows the software architecture as shown below:



2.2 Embedded LUA Library Information

Customer applications written using LUA scripts are text files stored in the EFS of the SIMCom Module. Before running the customer applications, user should put their scripts to the C:\ directory on the SIMCom module.

The LUA script is executed in a task inside the SIMCom module at a low priority, for not interfering the other system functions, the extended API vmsetpri is not recommended to set the LUA engine running on the high priority, except for some very special case.

The maximum RAM can be used by LUA engine is 3.5M. Two APIs (getcurmem and getpeakmem) are provided to get the current memory used and the peak memory used while running the script.

1. Libraries Added

- The SIO library is the most important one. It allows LUA script to send AT commands, receive responses and unsolicited indications. The AT command is quite the same as the usual AT sent through the serial port interface in SIMCom module. The difference is that this interface is not a real serial port but just an internal software bridge between LUA and mobile internal AT command handling engine. All AT commands working in the SIMCom module are working in this software interface as well.
- The GPIO library allows LUA script to handle general purpose input output faster than through AT commands, skipping the command parser and going directly to control the pins.
- The UART library is an implementation on the LUA core of the UART master. It allows LUA to handle UART operations instead of using AT commands.
- The I2C library is an implementation on the LUA core of the IIC bus master. It allows LUA to read and write the specified IIC registers.
- The AUDIO library is an implementation on the LUA core of the AUDIO manager.
- The GPS library allows LUA script to set and query GPS setting and geographic information.
- The NET library allows LUA script to query the wireless network information, like registration information, RSSI information and the network mode (GSM, GPRS and WCDMA, etc).
- The FTP library allows LUA script to perform simple FTP put and get operations.
- The ADC library allows LUA script to read the analogue value on the specified ADC channel.
- The BIT library allows LUA script to perform bitwise operations.
- The ATCTL library allows LUA script to handle the data received from external serial port instead of processing it using the internal AT handling engine.
- The SPI library allows LUA script to write or read SPI device.
- The IME library allows LUA script to realize chinese input method for UI.
- The THREAD library allows LUA script to create sub-threads.
- The SMS library allows LUA script to send or write sms using LUA APIs instead of using AT commands.
- The SMTP library allows LUA script to operate SMTP using LUA APIs instead of using AT commands.
- The MMS library allows LUA script to send or receive MMS using LUA APIs instead of using AT commands.
- The HASH library supports MD5, SHA1, SHA256, SHA384, SHA512 calculation.
- The MINIZIP library supports zip/unzip function using zlib
- The NMEA library supports getting NMEA information.
- The VOICE_CALL library is used to do voice call operation.

- The PHONEBOOK library is used to do phone book operation.
- The PIN library is used to do PIN management.
- The COMMON CHANNEL library allows to do SSL operation.
- The SMTPS library allows to do SMTPS sending.
- The FTPS library allows to do FTPS operation.

2. Extended Libraries

The following APIs are extended in the base library:

- printdir
- print
- reportte
- sendtoport
- vmsetpri
- vmgetpri
- vmsleep
- pathoffilename
- setevtpri
- waitevt
- setevt
- clearevts
- vmstarttimer
- vmstoptimer
- getcurmem
- getpeakmem

The following API is extended in the io library:

- file:trunc

The following APIs are extended in the os library:

- os.filelength
- os.delfile

The following APIs are extended in the string library:

- string.equal
- string.startwith
- string.endwith
- string.absfind
- string.hex2bin
- string.bin2hex

3. Removed Libraries

The following APIs are not supported in the io library:

- io.flush
- io.input
- io.lines
- io.output
- io.popen
- io.read
- io.tmpfile
- io.type
- io.write

The following APIs are not supported in the os library:

- os.execute
- os.exit
- os.getenv
- os.tmpname

SIMCom module provides LUA extension based on LUA5.1.4, which supports most original standard LUA APIs. SIMCom module also provides APIs which can access system functions and operating SIMCom hardware directly.

Table 2.1 lists SIMCom extended LUA Libraries.

Library	Functions
AT+EMAILCID	Set Email Bearer Profile Identifier
sio	The sio library is used by LUA script to send AT commands, receive responses and unsolicited indication.
gpio	The gpio library allow LUA script to handle general purpose input output directly instead of using AT commands.
uart	The uart library allows LUA script to handle UART operations directly instead of using AT commands.
i2c	The i2c library allows LUA script to handle IIC read/write operations directly instead of using AT commands.
adc	The adc library allows LUA script to read analogue value on the specified ADC channel instead of using AT commands.
audio	The audio library allows LUA script to manage AUDIO functions directly instead of using AT commands.
gps	The GPS library allows LUA script to set and query GPS setting and geographic information instead of using AT commands.
net	The net library allows LUA script to query the wireless network information directly instead of using AT commands.

ftp	The ftp library allows LUA script to put/get files to/from FTP server directly instead of using AT commands.
bit	The bit library allows LUA script to perform bitwise operations.
atctl	The atctl library allows LUA script to handle the data received from external serial port instead of processing it using the internal AT handling engine.
spi	The SPI library allows LUA script to write or read SPI device.
ime	The IME library allows LUA script to realize chinese input method for UI.
thread	The thread library allows LUA script to handle multi-thread operations.
sms	The sms library allows LUA script to manage SMS operations directly instead of using AT commands.
mms	The mms library allows LUA script to manage MMS operations directly instead of using AT commands.
smtp	The smtp library allows LUA script to manage SMTP operations directly instead of using AT commands.
socket	The socket library is used to do socket(TCP/UDP) operation.
smtps	The smtp library allows LUA script to manage SMTPS operations directly instead of using AT commands.
ftps	The ftp library allows LUA script to put/get files to/from FTPS server directly instead of using AT commands.
common_ch	This library is majorly used to do SSL operation.
sslmgr	This library is used to set SSL certificates.
voice_call	This library is used to do voice call operation.
sms	This library is used to do SMS operation.
phonebook	This library is used to do PHONEBOOK operation.
multimedia	This library is used to do multimedia operation
minizip	This library is used to do ZIP operation.
hash	This library is used to do hash operation(md5, sha1…)

Table 2.1 SIMCom extended libraries

3 LUA Build-in Custom Libraries

3.1 BASE library

3.1.1 printdir

Description

The function is used to set the direction of the print function

Prototype	void printdir ([int dir])
Parameters	dir: the direction of the print 0: not print (default) 1: print to external AT interface
Return value	None

Example

```
--D`isable print
printdir(0)
print("this is not printed\r\n")
--print to AT interface
printdir(1)
print("this is printed to AT interface\r\n")
```

3.1.2 print

Description

The function is used to print trace information to DIAG or external AT interface, this is an original LUA API, the only difference is the total print string length cannot exceed 60000 bytes. If user needs to output binary string including '\0', the os.printstr() is suggested to be used.

Prototype	void print (var1,...)
Parameters	var1... : any value that can be converted to a string
Return value	None

Example

```
prompt = "This is my prompt, count="
count = 1;
print(prompt, count, "\r\n");
```

3.1.3 sendtoport

Description

The function is used to send information to the TE.

NOTE:1. If you want use UART2,you need pull down the UART's DTR.

2. UART2 pins and SPI pins is multiplexed, so if enable SPI function, then uart2 function will be disable, and enable uart2 will be disable SPI function.

AT+CGFUNC=21,1 enable UART2 disable SPI.

AT+CGFUNC =21,0 disable UART2 ,it will be auto switch to SPI function.

Prototype	void sendtoport (int port, string data)
Parameters	port: the sio port to send data 0=to the port decided by AT+CATR 1=UART port 2=USB modem port 3=USB AT port 4=UART2 PORT data: the data to be sent
Return value	None

Example

```
prompt = “This is my prompt\r\n”
port = 1
sendtoport(port, prompt) ;
```

3.1.4 vmsetpri

Description

The function is used to set the priority of the internal task for LUA.

Prototype	void vmsetpri (int pri)
Parameters	pri : the priority of the internal task for LUA 1: low 2: medium 3: high (This value is used for some very special case. For most applications, it shouldn’t be used)
Return value	None

Example

```
LOW_PRIORITY = 1
MEDIUM_PRIORITY = 2
```

```
HIGH_PRIORITY = 3
vmsetpri(LOW_PRIORITY);
vmsetpri(MEDIUM_PRIORITY);
vmsetpri(HIGH_PRIORITY);
```

3.1.5 vmgetpri

Description

The function is used to get the priority of the internal task for LUA.

Prototype	int vmgetpri ()
Parameters	None
Return value	the priority of the internal task for LUA 1: low 2: medium 3: high

Example

```
LOW_PRIORITY = 1
MEDIUM_PRIORITY = 2
HIGH_PRIORITY = 3
pri = vmgetpri()
print("current priority is ", pri, "\r\n")
```

3.1.6 vmsleep

Description

The function is used to make the internal LUA task sleeping.

Prototype	void vmsleep (int timeout)
Parameters	timeout: the time (ms) to sleep
Return value	None

Example

```
--sleep 2000 ms
vmsleep(2000)
```

3.1.7 pathoffilename

Description

The function is used to get the directory of a full file path.

Prototype	void pathoffilename (string fullpath)
Parameters	fullpath: the full path of a file
Return value	None

Example

```
fullpath = "c:\\testdir\\myfile.txt";
dir = pathoffilename(fullpath);
```

```
--print the dir (“c:\\testdir\\”)
print(« dir = « , dir, « \r\n »)
```

3.1.8 setevtpri

Description

The function is used to set the priority of an event. For event id from 0 to 20, the default priority is 101(maximum). For event id form 21 to 40, the default priority is (100-event_id).

Prototype	setevtpri (int evt, int pri)
Parameters	evt: the event id to be set. Pri: the priority of the event.
Return value	the result of the setting: true: successful false: failed

Example

```
event_id = 7
event_priority = 100
setevtpri(evt_id, event_priority)
```

3.1.9 waitevt

Description

The function is used to wait an event to occur.

Prototype	waitevt (int timeout)
Parameters	timeout: the maximum time to wait.
Return value	There are for return values for waitevt(…). Event_id: the id of the event with the highest priority that occurred. If no event , -1 will be returned. Event_param1: the first parameter of the event. event_param2: the second parameter of the event. Event_param3: the third parameter of the event. Event_clock: the timestamp of the event

Event and parameters description

event	event_id	event_param1	event_param2	event_param3	event_clock
GPIO_EVENT	0	<gpio_param1>	<gpio_param2>	0	os.clock()
UART_EVENT	1	0	0	0	os.clock()
KEYPAD_EVENT	2	<keypad_param1>	<keypad_param2>	0	os.clock()
USB_EVENT	3	0	0	0	os.clock()
AUDIO_EVENT	4	0	0	0	os.clock()
ZIGBEE_EVENT	7	0	0	0	os.clock()
VOICECALL_EVENT	21	<vcall_param1>	<vcall_param2>	0	os.clock()

SOCKET_EVENT	22	<socket_param1>	<socket_param2>	<socket_param3>	os.clock()
SLEEP_EVENT	23	<sleep_param1>	0	0	os.clock()
COMMON_CH_EVENT	24	<comch_param1>	<comch_param2>	<comch_param2>	os.clock()
FTPS_EVENT	25	<ftps_param1>	<ftps_param2>	0	os.clock()
SMS_EVENT	26	<sms_param1>	<sms_param2>	<sms_param3>	os.clock()
TIMER_EVENT	28	<timer_param1>	<timer_param2>	<timer_param3>	os.clock()
SIO_RCVD_EVENT	29	0	0	0	os.clock()
ATCTL_EVENT	30	0	0	0	os.clock()
OUT_CMD_EVENT	31	0	0	0	os.clock()
LED_EVENT	32	<led_param1>	<led_param2>	0	os.clock()
PDP_EVENT	33	<reserved>	<reserved>		os.clock()
NMEA_EVENT	35	0	0	0	os.clock()
MULTIMEDIA_EVENT	36	<multimedia_param1>	<multimedia_param2>	0	os.clock()

Parameter defined values

<keypad_param1>
Key id
<keypad_param2>
Key state 1 – up 2 – down
<timer_param1>
Timer id, the range is 0~9
<timer_param2>
The <evt_param2> of vmstarttimer() function.
<timer_param3>
The <evt_param3> of vmstarttimer() function.
<led_param1>
Network status 0 – not registered 1 – registered
<led_param2>
Call type 0 – no call 1 – voice call 2 – CS data call 3 – PS data call
<socket_param1>
The sub-event type: 0 – network event 1 – socket event
<socket_param2>
The sub-event parameter:

If <socket_param1> is 0, this field is the ps network handle opened using network.open()
If <socket_param1> is 1, this field is the socket handle.

<socket_param3>

The sub-event parameter:
If <socket_param1> is 0, this field is the network status, 0=network closed. 1=network opened.
If <socket_param1> is 1, this field is the socket event mask(see socket.select()).

<comch_param1>

The session ID, 0 or 1

<comch_param2>

The event ID(enum value):
TRSSL_CLIENT_EVENT_ACT_OPENING_NETWORK=0,
TRSSL_CLIENT_EVENT_ACT_OPENED_NETWORK,
TRSSL_CLIENT_EVENT_ACT_OPENING_SESSION,
TRSSL_CLIENT_EVENT_ACT_OPEN_SESSION_BEGIN,
TRSSL_CLIENT_EVENT_ACT_OPEN_SESSION_END,
TRSSL_CLIENT_EVENT_ACT_CLOSING_SESSION,
TRSSL_CLIENT_EVENT_ACT_CLOSE_SESSION_BEGIN,
TRSSL_CLIENT_EVENT_ACT_CLOSE_SESSION_END,
TRSSL_CLIENT_EVENT_ACT_CLOSING_NETWORK,
TRSSL_CLIENT_EVENT_ACT_CLOSE_NETWORK_BEGIN,
TRSSL_CLIENT_EVENT_ACT_CLOSE_NETWORK_END,
TRSSL_CLIENT_EVENT_ACT_RELEASE_SSL_CLIENT_END,
TRSSL_CLIENT_EVENT_ACT_SENDING,
TRSSL_CLIENT_EVENT_ACT_SEND_BEGIN,
TRSSL_CLIENT_EVENT_ACT_SEND_END,
TRSSL_CLIENT_EVENT_ACT_RECV_BEGIN,
TRSSL_CLIENT_EVENT_ACT_RECVING,
TRSSL_CLIENT_EVENT_ACT_RECV_END,
TRSSL_CLIENT_EVENT_ACT_NEW_RECV_EVENT,
TRSSL_CLIENT_EVENT_SSL_ALERT,
TRSSL_CLIENT_EVENT_PEER_CLOSED,
TRSSL_CLIENT_EVENT_ERROR,

<comch_param3>

The event error codeD(enum value):
TRSSL_CLIENT_RESULT_OK = 0,
TRSSL_CLIENT_RESULT_ALERTING_STATE = 1,
TRSSL_CLIENT_RESULT_UNKNOWN_ERROR = 2,
TRSSL_CLIENT_RESULT_ERROR_BUSY =3,
TRSSL_CLIENT_RESULT_SERVER_CLOSED = 4,
TRSSL_CLIENT_RESULT_ERROR_TIMEOUT = 5,
TRSSL_CLIENT_RESULT_ERROR_TRANSFER_FAILED = 6,
TRSSL_CLIENT_RESULT_ERROR_MEMORY_ERROR = 7,
TRSSL_CLIENT_RESULT_ERROR_INVALID_PARAM = 8,
TRSSL_CLIENT_RESULT_NETWORK_ERROR = 9,

<ftps_param1>
The FTPS event: 7 – FTPS/FTP server closed the session.
<ftps_param2>
The error_code of FTPS event, reserved now.
<sleep_param1>
The sleep status: 0 – module enter sleep mode 1 – module exit sleep mode
<gpio_param1>
The GPIO ID.
<gpio_param2>
The GPIO status: 0 – low level voltage 1 – high level voltage
<vcall_param1>
CALL_STATE_ACTIVE = 0; CALL_STATE_HELD = 1; CALL_STATE_DIALING = 2; CALL_STATE_ALERTING = 3; CALL_STATE_INCOMING = 4; CALL_STATE_WAITING = 5; CALL_STATE_DISCONNECTED = 6; CALL_RX_DTMF = 250;
<vcall_param2>
If <vcall_param1> is 0 to 6, this parameter is the call_id If <vcall_param1> is 250, this parameter is the received DTMF code.
<sms_param1>
0=CMTI event 1=CSDI event
<sms_param2>
0=ME storage 1=SIM storage
<sms_param3>
message index
<multimedia_param1>
0=WAVE event
<multimedia_param2>
If <multimedia_param1> is 0, this parameter is the state of WAVE file playing: 0= stop 1= play

Example

```
event_id, event_param1, event_param2, event_param3, event_clock = waitevt(10000)
print(event_id, " ", event_param1, "\r\n ");
```

3.1.10 setevt

Description

The function is used to set an event.

Prototype	boolean setevt (int evt, int param1, int param2, int param3)
Parameters	evt: the id of the event to be set. param1: the first parameter of the event param2: the second parameter of the event param3: the third parameter of the event
Return value	The result of setting: TRUE: successful FALSE: failed

Example

```
event_id    = 30
setevt(event_id)
```

3.1.11 clearevts

Description

The function is used to clear all the events that occurred and contained in the event array.

Prototype	void clearevts ()
Parameters	None
Return value	None

Example

```
clearevts()
```

3.1.12 vmstarttimer

Description

The function is used to start a timer.

Prototype	boolean vmstarttimer (int timer_id, int timeout[, int timer_type[,int owner_thread_index[,int evt_param2, int evt_param3]])
Parameters	timer_id: the ID of timer (0-29) timeout: the timeout value for the timer(ms) timer_type: the timer type 0: the timer will only generate 1 timer event 1: periodic timer which generates multiple timer event until the vmstoptimer function is called. (default) int owner_thread_index: the owner thread index for this timer event with timer_id.

	int evt_param2: if TIMER event is generated, the second parameter will use this value, default is 0. int evt_param3: if TIMER event is generated, the third parameter will use this value, default is 0.
Return value	the result of starting timer: true: successful false: failed

Example

```
rst = vmstarttimer(0, 1000);
while (true) do
evt, evt_param = waitevt(10000);
if (evt ~= -1) then
print(evt, "\r\n ");
end;
end;
```

3.1.13 vmstoptimer

Description

The function is used to stop a timer.

Prototype	boolean vmstoptimer (int timer_id)
Parameters	timer_id: the ID of timer (0-29)
Return value	the result of stopping timer: true: successful false: failed

Example

```
rst = vmstarttimer(0, 1000);
count = 0;
while (true) do
count = count + 1;
if (count > 10) then
break;
end;
evt, evt_param = waitevt(10000);
if (evt ~= -1) then
print(evt, "\r\n ");
end;
end;
vmstoptimer(0);
```

3.1.14 getcurmem

Description

The function is used to get the size of the current memory used for the LUA script.

Prototype	int getcurmem ()
Parameters	None
Return value	The size of the memory used now.

Example

```
cur_mem_used = getcurmem()
print("currently ", cur_mem_used, "bytes of memory are used for this script\r\n")
```

3.1.15 getpeakmem

Description

The function is used to get the size of the peak memory used for the LUA script.

Prototype	int getpeakmem ()
Parameters	None
Return value	The peak size of the memory used for running the current script.

Example

```
peak_mem_used = getpeakmem()
print(peak_mem_used, "bytes of peak memory are used for running this script\r\n")
```

3.1.16 getchargerstate

Description

The function is used to get the charger state.

Prototype	int int getchargerstate ()
Parameters	None
Return value	charger state: the charger state 0: not connected 1: connected charge status: the charging status 0: not charging 1: charging

Example

```
state = getchargerstate()
```

3.1.17 enable_key_at_report

Description

The function is used to enable or disable “+KEY” report.

Prototype	void enable_key_at_report (cmd)
Parameters	cmd: 0: disable 1: enable

Return value	None

Example

enable_key_at_report(0)

3.1.18 sound_play_tone

Description

The function is used to play tone.

Prototype	void sound_play_tone (int dev_id, int tone_id)
Parameters	dev_id: 1: handset 2: headset 3: speaker tone_id: please refer to AT+CPTONE.
Return value	None

Example

sound_play_tone(3, 26)

3.1.19 get_usb_mode

Description

The function is used to play tone.

Prototype	int get_usb_mode ()
Parameters	None
Return value	usb_mode: 0: suspended 1: resumed 2: unconfigured 3: configured 4: disconnected 5: connected

Example

usb_mode = get_usb_mdoe()

3.1.20 image_comp_add_image

Description

The function is used to combine two images and save to one new image. Currently only BMP files of RGB565 format are supported

Prototype	int image_comp_add_image (string src_path, string add_path, string dst_path, int left_pos, int top_pos,

	int dx,int dy)
Parameters	None
Return value	src_path: the source image file path add_path: the image file path to be added dst_path: the new image file path to save left_pos: the left position for the image file to be added to the source image top_pos: the top position for the image file to be added to the source image dx: the length of the image added to the source image dy: the height of the image added to the source image

Example

image_comp_add_image("C:\\Picture\\main.bmp", "C:\\Picture\\to_add.bmp", "C:\\Picture\\new_main.bmp", 20, 15, 30, 10)

3.1.21 videophone_set_source

Description

The function is used to set the source for video phone.

Prototype	int videophone_set_source (int tx_path, string filename)
Parameters	None
Return value	tx_path: the path of the file filename: the name of the file

Example

videophone_set_source(2, "C:\\Picture\\1.bmp")

3.1.22 camera_is_initialized

Description

The function is used to check whether the camera is initialized.

Prototype	boolean camera_is_initialized ()
Parameters	None
Return value	true: the camera is initialized false: the camera is not initialized

Example

local inited = camera_is_initialized()

3.2 IO Library

3.2.1 file:trunc

Description

The function is used to truncate the file opened using io.open.

Prototype	boolean file:trunc (int pos)
Parameters	int pos: the position to truncate
Return value	true: successful false: failed

Example

```
file = io.open("c:\\test1.txt","w")
assert(file)
file:trunc(0)
file:write("test content\r\ntest\r\n")
file:close()
```

3.3 OS Library

3.3.1 os.restartscript

Description

The function is used to restart the current running script or start running a new script.

Prototype	void os.restartscript ([string path])
Parameters	path: the full path of the lua script file to run; If this parameter is nil, the lua engine will restart the current script.
Return value	None

Example

```
rst = os.restartscript("c:\\main2.lua")
rst = os.restartscript();
```

3.3.2 os.crushrestart

Description

The function is used to set whether restart the script when the current running script ends.

Prototype	void os.crushrestart (restart, report)
Parameters	restart: restart the script when current script ends. report: whether report +LUA ERROR when crush occurs
Return value	boolean: true: successful

	false: failed

Example

```
rst = os.crushrestart(1,0)
```

3.3.3 os.enable_sleep

Description

The function is used to enable the module enter sleep-mode.

Prototype	void os.enable_sleep()
Parameters	None
Return value	None

Example

```
os.enable_sleep();
```

3.3.4 os.disable_sleep

Description

The function is used to disable the module enter sleep-mode.

Prototype	void os.disable_sleep()
Parameters	None
Return value	None

Example

```
os.disable_sleep();
```

3.3.5 os.do_poweroff

Description

The function is used to power off the module.

Prototype	boolean os.do_poweroff(boolean force)
Parameters	boolean force: power off using hardware method true: use hardware method false: use software method(normal process)
Return value	boolean result: true: successful false: failed

Example

```
os.do_poweroff(false);
```

3.3.6 os.do_reset

Description

The function is used to reset the module.

Prototype	boolean os.do_reset(boolean force)
Parameters	boolean force: reset using hardware method true: use hardware method false: use software method(normal process)
Return value	boolean result: true: successful false: failed

Example

```
os.do_reset(false);
```

3.3.7 os.filelength

Description

The function is used to get the length of a file.

Prototype	int os.filelength (string path)
Parameters	path: the full path of the file
Return value	the length of the file.

Example

```
len = os.filelength("c:\\test1.txt")
print("the length of test1.txt is ", len, "bytes\r\n");
```

3.3.8 os.delfile

Description

The function is used to delete an existing file.

Prototype	boolean os.delfile (string path)
Parameters	path: the full path of the file
Return value	the result of deleting: true: successful false: failed

Example

```
rst = os.delfile("c:\\test1.txt")
```

3.3.9 os.remove

Description

The function is used to remove a file or directory.

Prototype	void os.remove(string filepath)
Parameters	string filepath: the full path of the file or directory.
Return value	boolean result: true: successful false: failed

Example

```
rst = os.remove(“c:\\t1.txt”);
rst = os.remove(“c:\\mydir\\”);
```

3.3.10 os.rename

Description

The function is used to rename a file.

Prototype	void os.rename(string old_filepath, string new_filepath)
Parameters	string old_filepath: the full path of the old file string new_filepath: the full path of the new file
Return value	boolean result: true: successful false: failed

Example

```
rst = os.rename(“c:\\old.txt”, “c:\\new.txt”);
```

3.3.11 os.lsdir

Description

The function is used to list all the directories and files in the directory.

Prototype	table table os.lsdir(string dir)
Parameters	string dir: the full directory path
Return value	table: the sub-directory list under the designated directory table: the file list under the designated directory

Example

```
local dir_list, file_list = os.lsdir("C:\\");
```

3.3.12 os.mkdir

Description

The function is used to create a directory.

Prototype	boolean table os.mkdirr(string dir)
Parameters	string dir: the full directory path
Return value	the result of creating directory: true: successful false: failed

Example

```
rst = os.mkdir("C:\\MyDir");
```

3.3.13 os.rmdir

Description

The function is used to delete a directory.

Prototype	boolean os.rmdirr(string dir)
Parameters	string dir: the full directory path
Return value	the result of deleting directory: true: successful false: failed

Example

```
rst = os.rmdir("C:\\MyDir");
```

3.3.14 os.setportmode

Description

The function is used to set the port access mode.

Prototype	void os.setportmode(mode)
Parameters	mode: the mode to set
Return value	boolean: true: successful false: failed

Example

```
rst = os.setportmode(63)
```

3.3.15 os.getportmode

Description

The function is used to get the port access mode.

Prototype	int os.getportmode()
Parameters	None
Return value	The port access mode

Example

```
mode, forced = os.getportmode()
```

3.3.16 os.setflashmode

Description

The function is used to enable or disable the accessing T-FLASH card in EFS.

Prototype	void os.setflashmode(mode)
Parameters	mode: the mode to set
Return value	boolean: true: successful

	false: failed

Example

```
rst = os.setflashmode(1)
```

3.3.17 os.getflashmode

Description

The function is used to get the mode of whether the T-FLASH card can be accessed using inner EFS functions.

Prototype	int os.getflashmode()
Parameters	None
Return value	The mode.

Example

```
mode= os.getflashmode()
```

3.3.18 os.autodog

Description

The function is used to kick dog automatically when performing heavy lua task. Usually it is not needed.

Prototype	void os.autodog(boolean kick)
Parameters	kick: kick the dog automatically true: yes false: no
Return value	none

Example

```
os.autodog(true);
```

3.3.19 os.setats0

Description

The function is used to set ATS0 parameter instead of using ATS0 command.

Prototype	void os.setats0(int port, int ring_times)
Parameters	port: the SIO port 1: UART 2: MODEM 3: USB-AT -1: NONE int ring_times: the times of ring before accept the incoming call.
Return value	boolean result: true: successful

	false: failed

Example

```
os.setats0(1, 3);
```

3.3.20 os.printstr

Description

The function is used to print string to MCU. Different from print() function, it can output raw binary string including '\0'.

Prototype	boolean os.printstr(string data[, int port])
Parameters	string data: the string to output int port: 1: UART 2: USB-MODEM 3: USB-AT Other: use the setting of os.printport()
Return value	boolean result: true: successful false: failed

Example

```
local str="test\0\201hello";
os.printstr(str);
os.printstr("hello, this is output to USB-AT\r\n", 3);
```

3.3.21 os.printport

Description

The function is used to set the port used by print(), debug.print() and os.printstr(). By default, it is set to output to all port(USB-AT, USB-MODEM, UART).

NOTE:1. If you want use UART2,you need pull down the UART's DTR.

2. UART2 pins and SPI pins is multiplexed, so if enable SPI function, then uart2 function will be disable, and enable uart2 will be disable SPI function.

AT+CGFUNC=21,1 enable UART2 disable SPI.

AT+CGFUNC =21,0 disable UART2 ,it will be auto switch to SPI function.

Prototype	void os.printport(int port)
Parameters	int port: 1: UART 2: USB-MODEM 3: USB-AT 4: UART2 PORT
Return value	None

Example

```
os.printdir(1);
os.printport(3);
print("hello, this is output to USB-AT port\r\n");
os.printport(2);
Print("hello, this is output to USB-MODEM port\r\n");
os.printport(1);
print("hello, this is output to UART port\r\n");
os.printport(0);
```

3.3.22 os.get_cscs

Description

The function is used to get the AT+CSCS setting.

Prototype	int os.get_cscs()
Parameters	None
Return value	int: the cscs value

Example

```
val = os.get_cscs();
```

3.3.23 os.set_cscs

Description

The function is used to set the CSCS setting.

Prototype	int os.set_cscs(int cscs)
Parameters	int: the cscs value
Return value	true: successful false: failed to set

Example

```
result= os.set_cscs(1);
print("hello, this is output to all ports\r\n");
```

3.4 STRING Library

3.4.1 string.equal

Description

The function is used to judge whether two strings are equal.

Prototype	boolean string.equal (string str1, string str2[, int ignore_case])
Parameters	str1: the string to compare. str2: the string to compare. ignore_case: ignore the case of the two strings. 0: not ignore the case

	1: ignore the case
Return value	the result of comparing: true: equal false: not equal

Example

```
local str1 = "test string1";
local str2 = "test string2";
local rst = string.equal(str1,str2);
print("string.equal(str1,str2)=",rst,"\r\n");
```

3.4.2 string.startwith

Description

The function is used to judge whether a string contains the same string in the header of another string.

Prototype	boolean string.startwith (string str1, string str2[, int ignorecase])
Parameters	str1: the string to compare. str2: the string to compare. ignore_case: ignore the case of the two strings.
Return value	the result of comparing: true: str1 contains the same string as str2 in the header. false: str1 doesn't contain str2 in the header.

Example

```
local str1 = "test string1, test string";
local str2 = "TEST string1";
local rst = string.startwith(str1,str2, 1);
print("string. startwith (str1,str2, 1)=",rst,"\r\n");
```

3.4.3 string.absfind

Description

The function is used to find the start position in a string that contains another string.

Prototype	boolean string.absfind (string str1, string str2[,int start_pos[, int ignorecase]])
Parameters	str1: the string that may contain str2. str2: the string to find. start_pos: the start position of finding in str1. ignore_case: ignore the case of the two strings. 0: not ignore the case 1: ignore the case
Return value	the position of finding. If failed, return nil.

Example

```
local str1 = "test string1, test string";
local str2 = ",TEST string";
local pos = string.absfind(str1,str2, 1);
print("string. absfind (str1,str2, 1)=",pos,"\r\n");
```

3.4.4 string.hex2bin

Description

The function is used to convert a hex string to binary string.

Prototype	boolean string.hex2bin (string str1)
Parameters	str1: the string to convert
Return value	The converted string

Example

```
local str1 = "0054 0049 004D 0045 0020 0053 0045 0054 0054 0049 004E 0047";
local str2 = string.hex2bin(str1);
```

3.4.5 string.bin2hex

Description

The function is used to convert a binary string to hex string.

Prototype	boolean string.hex2bin (string str1[, int add_space, int bytes_each_line])
Parameters	str1: the string to convert add space: add space character after each character bytes_each_line: how many bytes to show on each line
Return value	The converted string

Example

```
local str1 = "0054 0049 004D 0045 0020 0053 0045 0054 0054 0049 004E 0047";
local str2 = string.hex2bin(str1);
local str3 = string.bin2hex(str2, 0);
```

3.4.6 string.appendbytes

Description

The function is used to add bytes in the end of a string.

Prototype	boolean string.appendbytes (string str1, int value, int bytes[, int sizeofvalue])
Parameters	str1: the original string value: the value to be added to the end of the string bytes: how many bytes to be added sizeofvalue: the size of the value in byte

Return value	The final string

Example

```
local str1 = "0054 0049 004D 0045 0020 0053 0045 0054 0054 0049 004E 0047";
local str2 = string.hex2bin(str1);
local str3 = string.appendbytes(str2,54,2);
```

3.4.7 string.replacebytes

Description

The function is used to replace bytes in a string.

Prototype	boolean string.replacebytes (string str1, int index, int value, int bytes[, int sizeofvalue])
Parameters	str1: the original string index: the index of the string value: the value to be replaced into the string bytes: how many bytes to be added sizeofvalue: the size of the value in byte
Return value	The final string

Example

```
local str1 = "0054 0049 004D 0045 0020 0053 0045 0054 0054 0049 004E 0047";
local str2 = string.hex2bin(str1);
local str3 = string.replacebytes(str2, 2, 73, 2);
```

3.4.8 string.insertbytes

Description

The function is used to insert bytes in a string.

Prototype	boolean string.insertbytes (string str1, int index, int value, int bytes[, int sizeofvalue])
Parameters	str1: the original string index: the index of the string value: the value to be inserted to the string bytes: how many bytes to be added sizeofvalue: the size of the value in byte
Return value	The final string

Example

```
local str1 = "0054 0049 004D 0045 0020 0053 0045 0054 0054 0049 004E 0047";
local str2 = string.hex2bin(str1);
local str3 = string. insertbytes (str2, 2, 73, 2);
```

3.4.9 string.deletebytes

Description

The function is used to delete bytes from a string.

Prototype	boolean string.deletebytes (string str1, int index, int bytes)
Parameters	str1: the original string index: the index of the string bytes: how many bytes to be deleted
Return value	The final string

Example

```
local str1 = "0054 0049 004D 0045 0020 0053 0045 0054 0054 0049 004E 0047";
local str2 = string.hex2bin(str1);
local str3 = string.deletebytes(str2, 6, 2);
```

3.4.10 string.rawint

Description

The function is used to get raw integer value from a string. The length of the string must be less than or equal to 4.

Prototype	boolean string.rawint (string str1)
Parameters	str1: the original string
Return value	The final string

Example

```
local str1 = "0054 0049";
local str2 = string.hex2bin(str1);
local intValue = string.rawint(str2);
```

3.4.11 string.rawnumber

Description

The function is used to get raw number value from a string. The length of the string must equal to 4 or 8.

Prototype	boolean string.rawnumber (string str1)
Parameters	str1: the original string
Return value	The final string

Example

```
local str1 = "0054 0049";
local str2 = string.hex2bin(str1);
local doubleValue = string.rawnumber(str2);
```

3.4.12 string.fromint

Description

The function is used to convert an integer value to a string.

Prototype	boolean string.rawnumber (int value, int

	sizeofvalue)
Parameters	value: the value to be converted sizeofvalue: the size of the value in byte
Return value	The final string

Example

```
local str1 = string.fromint(32);
```

3.4.13 string.fromnumber

Description

The function is used to convert a number value to a string.

Prototype	boolean string.fromnumber (double value, int len)
Parameters	value: the value to be converted len: the size of the value in byte
Return value	The final string

Example

```
local str1 = string.fromdouble(32.5, 8);
```

3.4.14 string.endwith

Description

The function is used to judge whether a string contains the same string in the tail of another string.

Prototype	boolean string.endwith (string str1, string str2[, int ignorecase])
Parameters	str1: the string to compare. str2: the string to compare. ignore_case: ignore the case of the two strings.
Return value	the result of comparing: true: str1 contains the same string as str2 in the tail. false: str1 doesn't contain str2 in the tail.

Example

```
local str1 = "test string1, test string";
local str2 = ", test string";
local rst = string.endwith(str1,str2, 1);
print("string. endwith (str1,str2, 1)=",rst,"\r\n");
```

3.5 BIT Library

3.5.1 bit.cast

Description

The function is used to cast a variable to an internally-used integer type.

Prototype	int bit.cast(int var)
Parameters	var: the variable to be cast.
Return value	the cast result

Example

```
rst = bit.cast (-1)
```

3.5.2 bit.bnot

Description

The function is used to perform a bitwise NOT operation.

Prototype	int bit.bnot(int var)
Parameters	var: the variable to be calculated using bitwise NOT.
Return value	the bitwise NOT operation result

Example

```
rst = bit.bnot (-1)
assert (rst == bit.cast (0))
```

3.5.3 bit.band

Description

The function is used to perform a bitwise AND operation.

Prototype	int bit.band(int var1, var2)
Parameters	var1, var2: the variable to be calculated using bitwise AND.
Return value	the bitwise AND operation result

Example

```
assert (bit.band (-1, -1) == bit.cast (-1))
```

3.5.4 bit.bor

Description

The function is used to perform a bitwise OR operation.

Prototype	int bit.bor(int var1, int var2)
Parameters	var1, var2: the variable to be calculated using bitwise OR.

Return value	the bitwise OR operation result

Example

assert (bit.bor (0, -1) == bit.cast (-1))

3.5.5 bit.bxor

Description

The function is used to perform a bitwise XOR operation.

Prototype	int bit.bxor(int var1, int var2)
Parameters	var1, var2: the variable to be calculated using bitwise XOR.
Return value	the bitwise XOR operation result

Example

assert (bit.bxor (0, -1) == bit.cast (-1))

3.5.6 bit.lshift

Description

The function is used to perform a bitwise left-shift operation.

Prototype	int bit.lshift(int var)
Parameters	var: the variable to be calculated using bitwise left-shift.
Return value	the bitwise left-shift operation result

Example

assert (bit.lshift (0, 0) == bit.cast (0))

3.5.7 bit.rshift

Description

The function is used to perform a logical bitwise right-shift operation.

Prototype	int bit.rshift(int var)
Parameters	var: the variable to be calculated using logical bitwise right-shift.
Return value	the logical bitwise right-shift operation result

Example

assert (bit.rshift (-1, 0) == bit.cast (-1))

3.5.8 bit.arshift

Description

The function is used to perform an arithmetic bitwise right-shift operation.

Prototype	int bit.rshift(int var)

Parameters	var: the variable to be calculated using arithmetic bitwise right-shift.
Return value	the arithmetic bitwise right-shift operation result

Example

```
assert (bit.arshift (-1, 1) == bit.cast (-1))
```

3.6 int64 Library

3.6.1 int64.new

Description

The function is used to return a variable of type int64.

Prototype	Int64 int64.new(int value)
Parameters	int value: the value to be converted to Int64.
Return value	Int64: the return result.

Example

```
print(1);
local z=int64.new(1)
```

3.6.2 int64.min

Description

This is a property of Int64. It is used to return the minimum value of type Int64.

Prototype	Int64 int64.min
Parameters	None
Return value	Int64: the minimum value of type Int64

Example

```
print(1);
print("min"," ", int64.min, "\r\n")
```

3.6.3 int64.max

Description

This is a property of Int64. It is used to return the maximum value of type Int64.

Prototype	Int64 int64.max
Parameters	None
Return value	Int64: the maximum value of type Int64

Example

```
print(1);
print("max"," ", int64.max, "\r\n")
```

3.7 base64 Library

3.7.1 base64.encode

Description

The function is used to encode string using base64.

Prototype	string base64.encode(string str)
Parameters	string str: the string to be encoded.
Return value	string: the encoded string.

Example

```
print(1);
base64_str = base64.encode(s)
```

3.7.2 base64.decode

Description

The function is used to decode string using base64.

Prototype	string base64.encode(string base64_str)
Parameters	string base64_str: the string to bedecoded.
Return value	string: the decoded string.

Example

```
print(1);
raw_str = base64.decode(base64_str)
```

3.8 ascii85 Library

3.8.1 ascii85.encode

Description

The function is used to encode string using ascii85.

Prototype	string ascii85.encode(string str)
Parameters	string str: the string to be encoded.
Return value	string: the encoded string.

Example

```
print(1);
ascii85_str = ascii85.encode(s)
```

3.8.2 ascii85.decode

Description

The function is used to decode string using ascii85.

Prototype	string ascii85.encode(string ascii85_str)
Parameters	string ascii85_str: the string to bedecoded.
Return value	string: the decoded string.

Example

```
print(1);
raw_str = ascii85.decode(ascii85_str)
```

3.9 THREAD Library

3.9.1 thread.create

Description

The function is used to create a thread which can be launched by thread.run() function.

Prototype	thread_id thread.create(function func)
Parameters	function func: the main routine for the thread
Return value	thread_id: the identity of the thready

Example

```
function func1(a, b)
end;
thread.create(func1)
```

3.9.2 thread.run

Description

The function is used to launch a thread which is created by thread.create() function.

Prototype	boolean thread.run(thread _id t)
Parameters	thread_id t: the thread to run
Return value	true: succeeded in running the thread false: failed to run the thread

Example

```
function func1(a, b)
end;
t = thread.create(func1)
thread.run(t);
```

3.9.3 thread.stop

Description

The function is used to stop a thread launched by thread.run() function. A thread cannot use this function to stop itself and also the main thread of LUA cannot be stopped using this function.

Prototype	boolean thread.stop(thread _id t)
Parameters	thread_id t: the thread tos top

Return value	true: succeeded in stopping the thread false: failed to stop the thread

Example

```
function func1(a, b)
end;
t = thread.create(func1)
thread.run(t);
vmsleep(1000);
thread.stop(t);
```

3.9.4 thread.running

Description

The function is used to check whether a thread is running.

Prototype	boolean thread.running(thread _id t)
Parameters	thread_id t: the thread to check
Return value	true: the thread is running false: the thread is not running

Example

```
function func1(a, b)
end;
t = thread.create(func1)
thread.run(t);
is_running = thread.running(t);
```

3.9.5 thread.identity

Description

The function is used to get the identify for the current thread which calls this function.

Prototype	thread_id thread.identity()
Parameters	None
Return value	The identity of the current thread.

Example

```
t = thread.identity();
```

3.9.6 thread.setevt

Description

The function is used to set a event to a thread.

Prototype	boolean thread.setevt(thread_id t, int evt, int evt_p1, int evt_p2, int evt_p3)
Parameters	thread_id t: the thread to be set the event. int evt: the identity of the event

	int evt_p1: the first parameter of the event int evt_p2: the second parameter of the event int evt_p3: the third parameter of the event
Return value	true: succeeded in setting the event false: failed to set the event

Example

```
function func1(a, b)
end;
t = thread.create(func1)
thread.run(t);
thread.setevt(t, 3, 1,2,3);
```

3.9.7 thread.waitevt

Description

The function is used to wait an event to occur for the current thread.

Prototype	thread.waitevt (int timeout)
Parameters	timeout: the maximum time to wait.
Return value	There are five return values for waitevt(…). Event_id: the id of the event with the highest priority that occurred. If no event , -1 will be returned. Event_param1: the first parameter of the event. event_param2: the second parameter of the event. Event_param3: the third parameter of the event. Event_clock: the timestamp of the event

Example

```
event_id, event_param = waitevt(10000)
print(event_id, " ", event_param, "\r\n ");
```

3.9.8 thread.peekevt

Description

The function is used to peek whether there is an event which occured for the current thread.

Prototype	thread.peekevt (int evt_id, int evt_p1, int evt_p2, int evt_p3)
Parameters	Int evt_id: the id of the event to match Int evt_p1: the first parameter of the event Int evt_p2: the second parameter of the event Int evt_p3: the third parameter of the event
Return value	There are five return values for thread.peekevt(…). Event_id: the id of the event matched that occurred.

	If no event , -1 will be returned. Event_param1: the first parameter of the event. event_param2: the second parameter of the event. Event_param3: the third parameter of the event. Event_clock: the timestamp of the event

Example

```
event_id, event_param1, event_param2, event_param3, event_clock = thread.peekevt(10000)
print(event_id, " ", event_param, "\r\n ");
```

3.9.9 thread.setpri

Description

The function is used to set the priority of a LUA thread. When setting priority of current thread, the vmsetpri() is better.

Prototype	void thread.setpri (thread_id t, int pri)
Parameters	thread_id t: the thread to set priority. pri : the priority of the internal task for LUA 1: low 2: medium 3: high (This value is used for some very special case. For most applications, it shouldn't be used)
Return value	None

Example

```
LOW_PRIORITY = 1
MEDIUM_PRIORITY = 2
HIGH_PRIORITY = 3
thread.setpri(thread.identity(), LOW_PRIORITY);
thread.setpri(thread.identity(), MEDIUM_PRIORITY);
thread.setpri(thread.identity(), HIGH_PRIORITY);
```

3.9.10 thread.getpri

Description

The function is used to get the priority of the current thread for LUA.

Prototype	int thread.getpri ()
Parameters	None
Return value	the priority of the internal task for LUA 1: low 2: medium 3: high

Example

```
LOW_PRIORITY = 1
MEDIUM_PRIORITY = 2
```

```
HIGH_PRIORITY = 3
pri = thread.getpri()
print("current priority is ", pri, "\r\n")
```

3.9.11 thread.sleep

Description

The function is used to make thecurrent LUA thread sleeping.

Prototype	void thread.sleep (int timeout)
Parameters	timeout: the time (ms) to sleep
Return value	None

Example

```
--sleep 2000 ms
thread.sleep(2000)
```

3.9.12 thread.enter_cs

Description

The function is used to let the current thread enter a critical section

Prototype	void thread.enter_cs (int cs_no)
Parameters	cs_no: the number of the critical section to enter. The range of the cs_no is from 0 to 29.
Return value	None

Example

```
thread.enter_cs(1)
```

3.9.13 thread.leave_cs

Description

The function is used to let the current thread leave a critical section

Prototype	void thread.eave_cs (int cs_no)
Parameters	cs_no: the number of the critical section to leave. The range of the cs_no is from 0 to 29.
Return value	None

Example

```
thread.leave_cs(1)
```

3.9.14 thread.setevtowner

Description

The function is used to set the owner thread of specified events. This function doesn't affect the thread.setevt () and setevt() functions. For timer event, this function doesn't take effect, and it

should use the <owner_thread_index> parameter of vmstarttimer().

Prototype	boolean thread.setevtowner(int min_evt, int max_evt)
Parameters	int min_evt: the minimum identity of the event int max_evt: the maximum identity of the event
Return value	true: succeeded in setting the event owner false: failed to set the event owner

Example

```
thread.setevtowner(3, 6);
```

3.9.15 thread.index

Description

The function is used to get the index of a thread

Prototype	int thread.index([thread id])
Parameters	thread id: a thread created using thread.create(). If this parameter is nil, it will return the index of the current running thread.
Return value	int: the thread index.

Example

```
index = thread.index();
```

3.9.16 thread.list

Description

The function is used to get the list of the running threads.

Prototype	string thread.list(boolean tostring=true) table thread.list(boolean tostring=false)
Parameters	boolean to_string: true: the result is a string format. false: the result is a table format.
Return value	string/table: the running thread list

Example

```
list = thread.list(true);
list = thread.list(false);
```

3.9.17 thread.clearevts

Description

The function is used to clear all events for a thread

Prototype	void thread.clearevts()
Parameters	None
Return value	None

Example

thread.clearevts();

3.9.18 thread.getevtcount

Description

The function is used to statistics the count of events for a thread with specific event id and parameters.

Prototype	int thread.getevtcount(int evt, int evt_p1, int evt_p2, int evt_p3)
Parameters	int evt: the event id. If it is nil, this function will return the total count of events cached for this thread. int evt_p1: the first parameter. If it is nil, it will be ignored. int evt_p2: the second parameter.If it is nil, it will be ignored. int evt_p3: the third parameter.If it is nil, it will be ignored.
Return value	int total_count: the count of events with specific event_id and parameters.

Example

```
count = thread.getevtcount(5, nil, 2, 1);
--statistics the count of events with event id of 5, and evt_p2=2, evt_p3=1
```

3.9.19 thread.addevtfilter

Description

The function is used to add a event filter for a thread. There are maximum 10 filters for a thread.

Prototype	int thread.addevtfilter(int max_allow_count, boolean replace_oldest, int filter_event, int evt_p1, int evt_p2, int evt_p3)
Parameters	int max_allow_count: the maximum count allowd to cached in event queue. boolean replace_oldest: If it already reaches <max_allow_count>, whether to use the new event to replace the oldest one in the queue. int filter_event: the event id. It cannot be nil. int evt_p1: the first parameter. If it is nil, it will be ignored. int evt_p2: the second parameter.If it is nil, it will be ignored.

	int evt_p3: the third parameter.If it is nil, it will be ignored.
Return value	int filter_index: index of the filter.

Example

```
max_allow_count = 2;
replace_oldest = true;
filter_evt = 31;
filter_evt_p1 = 4;
filter_index = thread.addevtfilter(max_allow_count, replace_oldest, filter_evt,filter_evt_p1, nil, nil);
print("filter_index=", filter_index, "\r\n");
```

3.9.20 thread.deleteevtfilter

Description

The function is used to delete a event filter for a thread.

Prototype	boolean thread.deleteevtfilter(int filter_index)
Parameters	int filter_index: the index of the filter.
Return value	the result of deleting: true: success false: failed

Example

```
rst = thread.deleteevtfilter(filter_index);
```

3.9.21 thread.signal_clean

Description

The function is used to clean specified signals for a thread.

Prototype	void thread.signal_clean(int sig_mask)
Parameters	int sig_mask: the mask of the signals ((1, 2, 4, 8, 16, 32, 64, 128 with bor operation).
Return value	None

Example

```
thread.signal_clean(7);
thread.signal_clean(3);
thread.signal_clean(4);
```

3.9.22 thread.signal_notify

Description

The function is used to notify a thread with specified signals.

Prototype	void thread.signal_notify(thread_id t, int sig_mask)
Parameters	thread_id: the id of the thread.

	int sig_mask: the mask of the signals (1, 2, 4, 8, 16, 32, 64, 128 with bor operation).
Return value	None

Example

thread.signal_notify(t, 2);

3.9.23 thread.signal_wait

Description

The function is used to let a thread wait specified signals.

Prototype	void thread.signal_wait(int sig_mask, int timeout_val)
Parameters	int sig_mask: the mask of the signals (1, 2, 4, 8, 16, 32, 64, 128 with bor operation). int timeout: the maximum milliseconds to wait. When it is 0, wait for ever.
Return value	The waited signals

Example

waited_mask = thread.signal_wait(7, 10000);

3.9.24 thread.dog_run

Description

This function is used to start a watchdog for thread, this function must call after thread.run() function.

Prototype	boolean thread.free(thread t_id, string dog_name, int timeout_ms)
Parameters	thread tid: the thread idendity return by thread.create function dog_name : watchdog name timeout_ms : watchdog timeout value, in ms
Return value	true: successful false: failed to free

Example

result= thread.dog_run(tid, “test”, 30000);

3.9.25 thread.dog_kick

Description

This function is used to kick thread watchdog, the watchdog timer will reload.

Prototype	boolean thread.dog_kick(thread t_id)
Parameters	thread tid: the thread idendity return by thread.create function

Return value	true: successful false: failed to free

Example

result= thread.dog_kick(tid);

3.9.26 thread.dog_reboot

Description

This function is used to make a watchdog reboot.

Prototype	boolean thread.dog_reboot(thread t_id)
Parameters	thread tid: the thread idendity return by thread.create function
Return value	true: successful false: failed to free

Example

result= thread.dog_reboot(tid);

3.9.27 thread.dog_suspend

Description

This function is used to suspend thread watchdog, the watchdog will stop counter, and there is no watchdog reboot.

Prototype	boolean thread.dog_suspend(thread t_id)
Parameters	thread tid: the thread idendity return by thread.create function
Return value	true: successful false: failed to free

Example

result= thread.dog_suspend(tid);

3.9.28 thread.dog_resume

Description

This function is used to resume thread watchdog, the watchdog will continue counter, and there is a watchdog reboot if watchdog timeout.

Prototype	boolean thread.dog_resume (thread t_id)
Parameters	thread tid: the thread idendity return by thread.create function
Return value	true: successful false: failed to free

Example

result= thread.dog_resume(tid);

3.10 SIO Library(AT command)

3.10.1 sio.send

Description

The function is used to set send AT command to the LUA virtual serial port on the module.

Prototype	void sio.send(string cmd)
Parameters	cmd: the data to be sent to virtual serial port
Return value	None

Example

```
sio.send("ATI\r\n");
```

3.10.2 sio.recv

Description

The function is used to set receive data from the LUA virtual serial port on the module.

Prototype	string sio.recv(int timeout)
Parameters	timeout: the timeout value in ms to receive data.
Return value	The data received on the serial port.

Example

```
sio.send("ATI\r\n");
rst = sio.recv();
```

3.10.3 sio.clear

Description

The function is used to clear the cached data received from the LUA virtual serial port on the module.

Prototype	void sio.clear()
Parameters	None
Return value	None

Example

```
sio.clear();
```

3.10.4 sio.exclrpt

Description

The function is used to force all URC to be sent to the LUA virtual serial port on the module only. This is useful when no external MCU exists.

Prototype	void sio.exclrpt(int mode)
Parameters	mode: the mode the reporting unsolidated result

	0: the unsolidated result will be sent to the virtual serial port and other ports decided by at+catr. 1: the unsolidated result will only be sent to the virtual serial port.
Return value	None

Example

```
sio.exclrpt(1);
```

3.11 HASH Library

For detailed library information, please refer to http://www.rjek.com/luahash-0.00.tar.bz2.

3.11.1 md5.init

Description

The function is used to initialize MD5 context.

Prototype	object md5.init(void)
Parameters	None
Return value	object: the hash object.

Example

```
hash_obj = md5.init();
```

3.11.2 [md5_object]:update

Description

The function updates the message-digest context to account for the presence of each of the characters in the message whose digest is being computed..

Prototype	void [md5_object]:update(string sub_msg)
Parameters	string sub_msg: The sub-message to be computed.
Return value	None

Example

```
hash_obj:update(str);
```

3.11.3 [md5_object]:final

Description

The function terminates the message-digest computation and ends with the desired message digest ..

Prototype	string [md5_object]:final(void)
Parameters	None
Return value	string: the computed result of MD5 value.

Example

```
function tohex(s)
```

```
return (s:gsub(".", function (x)
            return ("%02x"):format(x:byte())
              end))
end
foo = md5.init()
foo:update "The quick brown fox jumps over the lazy dog"
bing = foo:final()
print(tohex(bing))
```

3.11.4 md5.sum

Description

This function is used to compute the MD5 hash value for a string quickly.

Prototype	string md5.sum(string complete_msg)
Parameters	string complete_msg: The complete message to be computed.
Return value	string hash_value: the hashed value.

Example

```
hash_value = md5.sum(str);
```

3.11.5 sha1.init

Description

The function is used to initialize SHA1 context.

Prototype	object sha1.init(void)
Parameters	None
Return value	object: the hash object

Example

```
hash_obj = sha1.init();
```

3.11.6 [sha1_object]:update

Description

The function updates the message-digest context to account for the presence of each of the characters in the message whose digest is being computed..

Prototype	void [sha1_object]:update(string sub_msg)
Parameters	string sub_msg: The sub-message to be computed.
Return value	None

Example

```
hash_obj:update(str);
```

3.11.7 [sha1_object]:final

Description

The function terminates the message-digest computation and ends with the desired message digest ..

Prototype	string [sha1_object]:final(void)
Parameters	None
Return value	string: the computed result of SHA1 value.

Example

```
function tohex(s)
    return (s:gsub(".", function (x)
        return ("%02x"):format(x:byte())
          end))
end
foo = sha1.init()
foo:update "The quick brown fox jumps over the lazy dog"
bing = foo:final()
print(tohex(bing))
```

3.11.8 sha1.sum

Description

This function is used to compute the SHA1 hash value for a string quickly.

Prototype	string sha1.sum(string complete_msg)
Parameters	string complete_msg: The complete message to be computed.
Return value	string: hash_value: the hashed value.

Example

```
hash_value = sha1.sum(str);
```

3.11.9 sha256.init

Description

The function is used to initialize SHA256 context.

Prototype	object sha256.init(void)
Parameters	None
Return value	object; the hash object

Example

```
hash_obj = sha256.init();
```

3.11.10[sha256_object]:update

Description

The function updates the message-digest context to account for the presence of each of the characters in the message whose digest is being computed..

Prototype	void [sha256_object]:update(string sub_msg)
Parameters	string sub_msg: The sub-message to be computed.
Return value	None

Example

```
hash_obj:update(str);
```

3.11.11[sha256_object]:final

Description

The function terminates the message-digest computation and ends with the desired message digest ..

Prototype	string [sha256_object]:final(void)
Parameters	None
Return value	string: the computed result of SHA256 value.

Example

```
function tohex(s)
return (s:gsub(".", function (x)
        return ("%02x"):format(x:byte())
    end))
end
foo = sha256.init()
foo:update "The quick brown fox jumps over the lazy dog"
bing = foo:final()
print(tohex(bing))
```

3.11.12sha256.sum

Description

This function is used to compute the SHA256 hash value for a string quickly.

Prototype	string sha1.sum(string complete_msg)
Parameters	string complete_msg: The complete message to be computed.
Return value	string hash_value: the hashed value.

Example

```
hash_value = sha256.sum(str);
```

3.11.13sha384.init

Description

The function is used to initialize SHA384 context.

Prototype	object sha384.init(void)
Parameters	None
Return value	object: the hash object.

Example

```
hash_obj = sha384.init();
```

3.11.14[sha384_object]:update

Description

The function updates the message-digest context to account for the presence of each of the characters in the message whose digest is being computed..

Prototype	void [sha384_object]:update(string sub_msg)
Parameters	string sub_msg: The sub-message to be computed.
Return value	None

Example

```
hash_obj:update(str);
```

3.11.15[sha384_object]:final

Description

The function terminates the message-digest computation and ends with the desired message digest ..

Prototype	string [sha384_object]:final(void)
Parameters	None
Return value	string: the computed result of SHA384 value.

Example

```
function tohex(s)
return (s:gsub(".", function (x)
        return ("%02x"):format(x:byte())
          end))
end
foo = sha384.init()
foo:update "The quick brown fox jumps over the lazy dog"
bing = foo:final()
print(tohex(bing))
```

3.11.16sha384.sum

Description

This function is used to compute the SHA384 hash value for a string quickly.

Prototype	string sha1.sum(string complete_msg)
Parameters	string complete_msg: The complete message to be computed.
Return value	string hash_value: the hashed value.

Example

```
hash_value = sha384.sum(str);
```

3.11.17sha512.init

Description

The function is used to initialize SHA384 context.

Prototype	object sha512.init(void)
Parameters	None
Return value	object: the hash object.

Example

```
hash_obj = sha512.init();
```

3.11.18[sha512_object]:update

Description

The function updates the message-digest context to account for the presence of each of the characters in the message whose digest is being computed..

Prototype	void [sha512_object]:update(string sub_msg)
Parameters	string sub_msg: The sub-message to be computed.
Return value	None

Example

```
hash_obj:update(str);
```

3.11.19[sha512_object]:final

Description

The function terminates the message-digest computation and ends with the desired message digest ..

Prototype	string [sha512_object]:final(void)
Parameters	None
Return value	string: the computed result of SHA512 value.

Example

```
function tohex(s)
```

```
return (s:gsub(".", function (x)
            return ("%02x"):format(x:byte())
              end))
end
foo = sha512.init()
foo:update "The quick brown fox jumps over the lazy dog"
bing = foo:final()
print(tohex(bing))
```

3.11.20sha512.sum

Description

This function is used to compute the SHA512 hash value for a string quickly.

Prototype	string sha1.sum(string complete_msg)
Parameters	string complete_msg: The complete message to be computed.
Return value	string: hash_value: the hashed value.

Example

```
hash_value = sha512.sum(str);
```

3.12 MINIZIP Library

This library is based on zlib-1.2.8. For reducing memory allocation, we modify MAX_WBITS from 15 to 8, MAX_MEM_LEVEL from 8 to 2, Z_BUFSIZE from 64K to 2K, UNZ_BUFSIZE from 16K to 2K. So the zip files compressed using normal zlib may not be extracted by minizip on SIMCOM LUA module. But the zip files compressed using SIMCOM LUA module can be extracted by the module itself and other normal zlib application.

3.12.1 minizip.openzip

Description

The function is used to open zip file.

Prototype	int minizip.openzip(string filepath, int option)
Parameters	string filepath: the full path of file. int option: 0: create a new file. 2: append new files to the existing zip.
Return value	int zip_handle: the handle of the opened zip file.

Example

```
printdir(1)

local zipfile = "C:\\myfile.zip";
local file1 = "C:\\1.log";
local file2 = "C:\\1024B_ASCII.txt";
```

```
local file3 = "C:\\mytest.jpg";
local file4 = "C:\\500K.txt";
--local password = nil;
local password = "123456";
local compress_level = -1;
local exclude_path = 1;

local rst;
local zip_handle = minizip.openzip(zipfile, 0);
if (not zip_handle) then
   print("failed ot open zipfile ", zipfile, "\r\n");
   return;
end;
print("minizip.openzip, zip_handle = ", zip_handle, "\r\n");

rst = minizip.addfile(zip_handle, file1, password, compress_level, exclude_path);
print("minizip.addfile(",file1,"), result = ", rst, "\r\n");

rst = minizip.addfile(zip_handle, file2, password, compress_level, exclude_path);
print("minizip.addfile(",file2,"), result = ", rst, "\r\n");

rst = minizip.closezip(zip_handle);
print("minizip.closezip, result = ", rst, "\r\n");

print("append new files to zip\r\n");
zip_handle = minizip.openzip(zipfile, 2);
if (not zip_handle) then
   print("failed ot open zipfile ", zipfile, "\r\n");
   return;
end;
print("minizip.openzip, zip_handle = ", zip_handle, "\r\n");

rst = minizip.addfile(zip_handle, file3, password, compress_level, exclude_path);
print("minizip.addfile(",file3,"), result = ", rst, "\r\n");

rst = minizip.addfile(zip_handle, file4, password, compress_level, exclude_path);
print("minizip.addfile(",file4,"), result = ", rst, "\r\n");

rst = minizip.closezip(zip_handle);
print("minizip.closezip, result = ", rst, "\r\n");
```

3.12.2 minizip.closezip

Description

The function is used to close zip file.

Prototype	int minizip.closezip(int zip_handle)
Parameters	int zip_handle: the handle of the opened zip file.
Return value	int error_code: 0: successful other: failed

Example

(refer to example of minizip.openzip)

3.12.3 minizip.addfile

Description

The function is used to add new file to opened zip file.

Prototype	int minizip.addfile(int zip_handle, string filpath, string password, int compress_level, int exclude_path)
Parameters	int zip_handle: the handle of the opened zip file. string filepath: the full path of the file to add string password: the password to crypt the file int compress_level: the compress level, 0 = store, 1=compress faster, 9=compress better int exclude_path: exclude the path
Return value	int error_code: 0: successful other: failed

Example

(refer to example of minizip.openzip)

3.12.4 miniunz.openzip

Description

The function is used to open zip file for extracting.

Prototype	int miniunz.openzip(string filepath)
Parameters	string filepath: the full path of file.
Return value	int zip_handle: the handle of the opened zip file.

Example

```
printdir(1)
local zipfile = "C:\\myfile.zip";
--local password = nil;
local password = "123456";
local output_dir = "C:";    --"C:\\";

local rst;
```

```
local zip_handle = miniunz.openzip(zipfile);
if (not zip_handle) then
   print("failed ot open zipfile ", zipfile, "\r\n");
   return;
end;
print("miniunz.openzip, zip_handle = ", zip_handle, "\r\n");

local entry_count = miniunz.number_entry(zip_handle);
print("miniunz.number_entry=", entry_count, "\r\n");

local entry_index = 0;
while (true) do
   local filename, filesize, crypted = miniunz.get_current_entry_info(zip_handle);
   if (filename) then
      print("entry[", entry_index, "]={", filename, ",",filesize," bytes,is_crypted=",crypted,"}\r\n");
   else
      print("failed to get entry[",entry_index, "] infoformation\r\n");
   end;
   rst = miniunz.extract_current_file(zip_handle, password, output_dir);
   print("miniunz.extract_current_file, rst=", rst, "\r\n");
   if (not miniunz.goto_next_entry(zip_handle)) then
      print("No next entry exist\r\n");
      break;
   end;
   entry_index = entry_index + 1;
end;

rst = miniunz.closezip(zip_handle);
print("miniunz.closezip, result = ", rst, "\r\n");
```

3.12.5 miniunz.closezip

Description

The function is used to close zip file.

Prototype	int miniunz.closezip(int zip_handle)
Parameters	int zip_handle: the handle of the opened zip file.
Return value	int error_code: 0: successful other: failed

Example

(refer to example of miniunz.openzip)

3.12.6 miniunz.number_entry

Description

The function is used to get the number of the entry in the zip file.

Prototype	int miniunz.number_entry(int zip_handle)
Parameters	int zip_handle: the handle of the opened zip file.
Return value	int error_code: >=0: the number of entries. <0: failed

Example

(refer to example of miniunz.openzip)

3.12.7 miniunz.goto_next_entry

Description

The function is used to goto next entry in the zip file.

Prototype	int miniunz.goto_next_entry(int zip_handle)
Parameters	int zip_handle: the handle of the opened zip file.
Return value	int error_code: 0: successful. other: failed

Example

(refer to example of miniunz.openzip)

3.12.8 miniunz.get_current_entry_info

Description

The function is used to get the current entry information in the zip file.

Prototype	string int boolean miniunz.get_current_entry_info(int zip_handle)
Parameters	int zip_handle: the handle of the opened zip file.
Return value	string filename: the name of the file int filesize: the size of files in byte. boolean crypted: whether the file has been crypted.

Example

(refer to example of miniunz.openzip)

3.12.9 miniunz.extract_current_file

Description

The function is used to extract the file of current entry in the zip file.

Prototype	int miniunz.extract_current_file(int zip_handle, string password, string output_directory)
Parameters	int zip_handle: the handle of the opened zip file. string password: the password to used to decrypt the file output_directory: the output directory
Return value	int error_code: 0: successful. other: failed.

Example

(refer to example of miniunz.openzip)

3.13 PIN Library

3.13.1 pin.remain_info

Description

The function is used to get the PIN status.

Prototype	table pin.remain_info(void)
Parameters	None
Return value	the result of the operation

Example

```
print(1);
function trace_pin_status()
    local info = pin.remain_info();
    if (not info) then
        print("failed to get PIN remain information\r\n");
        return;
    end;
    print("PIN1 left retry=", info.pin1_left_retries, ", left unblock retries=", info.pin1_left_unblock_retries, ", status=", info.pin1_status, "\r\n");
    print("PIN2 left retry=", info.pin2_left_retries, ", left unblock retries=", info.pin2_left_unblock_retries, ", status=", info.pin1_status, "\r\n");
end;
```

3.13.2 pin.enable

Description

The function is used to enable or disable PIN.

Prototype	boolean pin.enable(int pin_id, string pin_code, boolean enable_pin)
Parameters	int pin_id; 0: PIN1

	1: PIN2 string pin_code: the code of PIN boolean enable_pin: true: enable PIN false: disable PIN
Return value	the result of the operation true: successful false: failed

Example

```
print(1);
local result = pin.enable(pin_id, pin_code, enable);
```

3.13.3 pin.verify

Description

The function is used to verify PIN.

Prototype	boolean pin.verify(int pin_id, string pin_code)
Parameters	int pin_id; 0: PIN1 1: PIN2 string pin_code: the code of PIN
Return value	the result of the operation true: successful false: failed

Example

```
print(1);
local result = pin.verify(pin_id, pin_code);
```

3.13.4 pin.change

Description

The function is used to change PIN.

Prototype	boolean pin.change(int pin_id, string old_pin_code, string new_pin_code)
Parameters	int pin_id; 0: PIN1 1: PIN2 string old_pin_code: the old code of PIN string new_pin_code: the new code of PIN
Return value	the result of the operation true: successful false: failed

Example

```
print(1);
local result = pin.change(pin_id, old_pin_code, new_pin_code);
```

3.13.5 pin.unblock

Description

The function is used to unblock PIN.

Prototype	boolean pin.unblock(int pin_id, string puk_code, string new_pin_code)
Parameters	int pin_id; 0: PIN1 1: PIN2 string puk_code: the code of PUK string new_pin_code: the new code of PIN
Return value	the result of the operation true: successful false: failed

Example

```
print(1);
local result = pin.unblock(pin_id, puk_code, new_pin_code);
```

3.14 PHONEBOOK Library

3.14.1 phonebook.ready

Description

The function is used to whether the phone book has been initialized completely by the module.

Prototype	boolean phonebook.ready(void)
Parameters	None
Return value	the result of the operation: true: successful false: failed

Example

```
print(1);
if (not phonebook.ready()) then
    print("phonebook not ready\r\n");
    return;
end;
```

3.14.2 phonebook.info

Description

The function is used to query a phone book storage information.

Prototype	table phonebook.info(int storage)
Parameters	int storage: PB_DEV_DC = 0; PB_DEV_MC = 1; PB_DEV_RC = 2; PB_DEV_SM = 3; PB_DEV_ME = 4; PB_DEV_FD = 5; PB_DEV_ON = 6; PB_DEV_LD = 7; PB_DEV_EN = 8; PB_DEV_SN = 9;
Return value	table: the storage information

Example

```
print(1);
function get_pb_dev_info(pb_dev)
    print("get_pb_dev_info, pb_dev=", pb_dev, "\r\n");
    local pb_dev_info = phonebook.info(pb_dev);--get current phone book setting
    if (not pb_dev_info) then
        print("failed to get phonebook information\r\n");
        return;
    end;
    print("pb_dev_info={\r\n");
    print("    dev=", pb_dev_info.storage, "\r\n");
    print("    used=", pb_dev_info.used, "\r\n");
    print("    max=", pb_dev_info.max, "\r\n");
    print("}\r\n");
end;
```

3.14.3 phonebook.write

Description

The function is used to write an item to the phone book.

Prototype	boolean phonebook.write(int storage, int index, string phone_number, string name)
Parameters	int storage: PB_DEV_DC = 0; PB_DEV_MC = 1; PB_DEV_RC = 2; PB_DEV_SM = 3; PB_DEV_ME = 4; PB_DEV_FD = 5; PB_DEV_ON = 6; PB_DEV_LD = 7;

	PB_DEV_EN = 8; PB_DEV_SN = 9; int index: the store location string phone_number: the phone number. string name: the name of the owner.
Return value	the result of the operation: true: successful false: failed

Example

```
print(1);
function write_pb_item(storage, index, phone_number, name)
    print("write_pb_item, storage=", storage, ", index=", index, ", phone_number=", phone_number, ", name=", name, "\r\n");
    local result = phonebook.write(storage, index, phone_number, name);--if index is nil, write to the first empty slot.
    print("phonebook.write, result=", result, "\r\n");
end;
```

3.14.4 phonebook.read

Description

The function is used to read one or more items from the phone book.

Prototype	string phonebook.read(int storage, int start_index, int end_index)
Parameters	int storage: PB_DEV_DC = 0; PB_DEV_MC = 1; PB_DEV_RC = 2; PB_DEV_SM = 3; PB_DEV_ME = 4; PB_DEV_FD = 5; PB_DEV_ON = 6; PB_DEV_LD = 7; PB_DEV_EN = 8; PB_DEV_SN = 9; int start_index: the start store location int end_index: the end store location
Return value	string: the read list. [<storage>,<index1>,<number>,<type>,<text>[<CR><LF> <storage>, <index2>,<number>,<type>,<text>[…]]] The <storage> is the same as input.

	Other parameters are the same as AT+CPBR.

Example

```
print(1);
function read_pb_item(storage, start_index, end_index)
    print("read_pb_item, storage=", storage, ", start_index=", start_index, ", end_index=",
end_index, "\r\n");
    local item_list = phonebook.read(storage, start_index, end_index);
    if (not item_list) then
        print("failed to read pb item\r\n");
  return;
    end;
    print("item_list={\r\n");
    if (printdir()) then
        os.printstr(item_list);
    end;
    print("}\r\n");
end;
```

3.14.5 phonebook.findname

Description

The function is used to find items from the phone book using name parameter.

Prototype	string phonebook.findname(int storage, string name, int match_type)
Parameters	int storage: PB_DEV_DC = 0; PB_DEV_MC = 1; PB_DEV_RC = 2; PB_DEV_SM = 3; PB_DEV_ME = 4; PB_DEV_FD = 5; PB_DEV_ON = 6; PB_DEV_LD = 7; PB_DEV_EN = 8; PB_DEV_SN = 9; String name: the name fo find int match_type: 0 = contain(default) 1 = match exactly 3 = start with
Return value	string: the read list. [<storage>,<index1>,<number>,<type>,<text>[<CR><LF>

	<storage>, <index2>,<number>,<type>,<text>[…]]] The <storage> is the same as input. Other parameters are the same as AT+CPBF.

Example

```
print(1);
function findname_pb_item(storage, name, match_type, case_sensitive)
   print("findname_pb_item, storage=", storage, ", name=", name, ", match_type=", match_type, ",case_sensitive=", case_sensitive, "\r\n");
   --!!!ATTENTION: The phonebook.findphone()/phonebook.findname cannot be used in multiple threads concurrently.
   local item_list = phonebook.findname(storage, name, match_type, case_sensitive);
   if (not item_list) then
      print("failed to find pb item using name\r\n");
 return;
   end;
   print("item_list={\r\n");
   if (printdir()) then
      os.printstr(item_list);
   end;
   print("}\r\n");
end;
```

3.14.6 phonebook.findphone

Description

The function is used to find items from the phone book using phone number parameter.

Prototype	string phonebook.findpone(int storage, string phone_number, int match_type)
Parameters	int storage: PB_DEV_DC = 0; PB_DEV_MC = 1; PB_DEV_RC = 2; PB_DEV_SM = 3; PB_DEV_ME = 4; PB_DEV_FD = 5; PB_DEV_ON = 6; PB_DEV_LD = 7; PB_DEV_EN = 8; PB_DEV_SN = 9; string phoen_number: the phone number fo find int match_type:

	0 = contain(default) 1 = match exactly 3 = start with 4 = match intelligently
Return value	string: the read list. [<storage>,<index1>,<number>,<type>,<text>[<CR><LF> <storage>, <index2>,<number>,<type>,<text>[…]]] The <storage> is the same as input. Other parameters are the same as AT+CPBF.

Example

```
print(1);
function findphone_pb_item(storage, phone_num, match_type)
  print("findphone_pb_item, storage=", storage, ", phone_num=", phone_num, ", match_type=", match_type, "\r\n");
  --!!!ATTENTION: The phonebook.findphone()/phonebook.findname cannot be used in multiple threads concurrently.
  local item_list = phonebook.findphone(storage, phone_num, match_type);
  if (not item_list) then
    print("failed to find pb item using phone number\r\n");
 return;
  end;
  print("item_list={\r\n");
  if (printdir()) then
        os.print(item_list);
end;
print("}\r\n");
end;
```

3.15 VOICE_CALL Library

3.15.1 voice_call.initiate

Description

The function is used to initiate MO voice call. This command is equal to ATD for voice call.

Prototype	int voice_call.initiate(string phone_num[, int ss_mask])
Parameters	string phone_num: the destination phone number int ss_mask: bitwise of following values: ENABLE_CCUG = 4 --enable ccug

	DISABLE_CCUG = 8 --disable ccug ACTIVE_CLIR = 16--show calling number to called party SUPRESS_CLIR = 32--hide calling number to called party
Return value	int call_id

Example

```
print(1);
call_id = voice_call.initiate(“18652845698”);
if (not call_id) then
   print(“failed to initiate call\r\n”);
end;
```

3.15.2 voice_call.answer

Description

The function is used to answer MT voice call. This command is equal to ATA for voice call.

Prototype	table voice_call.answer([int call_id])
Parameters	Int call_id, the ID of the call to answer. If nil is input, this routine will try to answer all waiting call.
Return value	table call_ids

Example

```
print(1);
print("call voice.answer()...\r\n");
local answered_calls = voice_call.answer();
--[[answer all incoming calls. voice_call.answer(call_id) can be used to answer the call with
call_id parameter.]]
 if (answered_calls) then
    for idx = 1, table.maxn(answered_calls), 1 do
       print("call[", answered_calls[idx], "] answered\rn");
    end;
 else
    print("failed to answer call\r\n");
end;
```

3.15.3 voice_call.hangup

Description

The function is used to hangup voice call. This command is equal to AT+CHUP.

Prototype	boolean voice_call.hangup([int call_id])
Parameters	Int call_id, the ID of the call to hangup. If nil is input, this routine will try to hangup all voice calls.
Return value	the result of the operation:

	true: successful false: failed

Example

```
print(1);
print("call voice_call.hangup()...\r\n");
result = voice_call.hangup(call_id);--if no parameter input, end all calls: voice_call.huangup(nil)
if (not result) then
    print("failed to end call\r\n");
    return;
end;
```

3.15.4 voice_call.list

Description

The function is used to list all voice calls. This command is similar to AT+CLCC.

Prototype	string voice_call.list(void)
Parameters	None
Return value	string list_result: the call list <call_id>,<dir>,<stat>,<mode>,<mpty>[,<number>[<CR><LF> <call_id>,<dir>,<stat>,<mode>,<mpty>[,<number>] [...]] The <call_id> is the inner call ID. It can be used to convert to <idX> parameter of AT+CLCC using voice_call.id2seq(call_id). The other parameters are the same of AT+CLCC.

Example

```
print(1);
function trace_call_list()
  print("call voice_call.list()...\r\n");
  local call_list = voice_call.list();
  if (call_list) then
    print("call_list=\r\n{\r\n", call_list, "}\r\n");
  else
    print("no call list got\r\n");
  end;
end;
```

3.15.5 voice_call.state

Description

The function is used to get the voice call state.

Prototype	int voice_call.state(int call_id)
Parameters	int call_id: the voice call ID.
Return value	int state: the state of the voic call: CALL_STATE_ACTIVE = 0; CALL_STATE_HELD = 1; CALL_STATE_DIALING = 2; CALL_STATE_ALERTING = 3; CALL_STATE_INCOMING = 4; CALL_STATE_WAITING = 5; CALL_STATE_DISCONNECTED = 6;

Example

```
print(1);
local state = voice_call.state(call_id);
print("call state[", call_id, "]=", state, "\r\n");
```

3.15.6 voice_call.send_dtmf

Description

The function is used to send DTMF codes.

Prototype	boolean voice_call.send_dtmf(int call_id, string dtmf_str[, int durationx100])
Parameters	int call_id: the voice call ID. string dtmf_str: the string of DTMF codes. Maximum length is 28 bytes. int durationx100: the real duration for each DTMF code sending is durationx100*100(milliseconds). Default it is 1(100 ms).
Return value	the result of the operation: true: successful false: failed

Example

```
print(1);
print("call voice_call.send_dtmf(123456#)...\r\n");
result = voice_call.send_dtmf(call_id, "123456#");--maximum 28 dtmf code in string
print("voice_call.send_dtmf, result=", result, "\r\n");

print("call voice_call.send_dtmf(*)...\r\n");
result = voice_call.send_dtmf(call_id, "*");--maximum 28 dtmf code in string
print("voice_call.send_dtmf, result=", result, "\r\n");
```

3.15.7 voice_call.chld

Description

The function is used to perform operation like AT+CHLD.

Prototype	boolean voice_call.chld(int chld_cmd[, int call_id])
Parameters	int chld_cmd: the command type of CHLD operation: CALL_CHLD_CMD_0 = 0 CALL_CHLD_CMD_1 = 1 CALL_CHLD_CMD_1X = 2 CALL_CHLD_CMD_2 = 3 CALL_CHLD_CMD_2X = 4 CALL_CHLD_CMD_3 = 5 CALL_CHLD_CMD_4 = 6 for voice_call.chld(chld_cmd, call_id), usage: voicecall.chld(CALL_CHLD_CMD_0) voicecall.chld(CALL_CHLD_CMD_1) voicecall.chld(CALL_CHLD_CMD_1X, call_id) voicecall.chld(CALL_CHLD_CMD_2) voicecall.chld(CALL_CHLD_CMD_2X, call_id) voicecall.chld(CALL_CHLD_CMD_3) voicecall.chld(CALL_CHLD_CMD_4) int call_id: the voice call ID.
Return value	the result of the operation: true: successful false: failed

Example

```
print(1);
print("call voice_call.chld(1 or 1x)...\r\n");
--result = voice_call.chld(CALL_CHLD_CMD_1);
result = voice_call.chld(CALL_CHLD_CMD_1X, call_id);
if (not result) then
   print("Error! failed to end call using AT+CHLD\r\n");
end;
```

3.15.8 voice_call.id2seq

Description

The function is used to convert the inner voice call ID to external <idX> parameter in AT+CLCC.

Prototype	int voice_call.id2seq(int call_id)
Parameters	int call_id: the inner voice call ID.
Return value	int seq_no: the <idX> of AT+CLCC.

Example

```
print(1);
local seq_no = voice_call.id2seq(call_id);--get the <idX> in AT+CLCC
print("voice_call.id2seq(", call_id, ")=", seq_no,"\r\n");
```

3.15.9 voice_call.seq2id

Description

The function is used to convert external <idX> parameter in AT+CLCC to the inner voice call ID.

Prototype	int voice_call.seq2id(int seq_no)
Parameters	int seq_no: the <idX> of AT+CLCC.
Return value	int call_id: the inner voice call ID.

Example

```
print(1);
local converted_call_id = voice_call.seq2id(seq_no);--get the inner call id using <idX> in AT+CLCC
print("voice_call.seq2id(", seq_no, ")=", converted_call_id,"\r\n");
```

3.16 SMS Library

3.16.1 sms.get_cmgf

Description

The function is used to get the AT+CMGF setting.

Prototype	int sms.get_cmgf()
Parameters	None
Return value	int: the cmgf value

Example

```
val = sms.get_cmgf();
```

3.16.2 sms.set_cmgf

Description

The function is used to set the CMGF setting.

Prototype	int sms.set_cmgf(int cmgf)
Parameters	int: the cmgf value
Return value	true: successful false: failed to set

Example

```
result= sms.set_cmgf(1);
```

3.16.3 sms.get_next_msg_ref

Description

The function is used to get the next message reference for later SMS writing or sending. Usually used for long SMS. Each long SMS needs the same message reference, this function trys to return an unique message reference for each time calling.

Prototype	int sms.get_next_msg_ref()
Parameters	None
Return value	int: the new message reference value

Example

```
val = sms.get_next_msg_ref();
```

3.16.4 sms.convert_chset

Description

The function is used to convert characters of SMS from one set to another set.

Prototype	string sms.convert_chset(string str1, int in_set,int out_set, int drop_inconv)
Parameters	string str1: the string to be converted int in_set: the character set type of the input string int out_set: the character set type of the out string int drop_inconv: whether to drop the characters that cannot be converted.
Return value	string: the converted characters

Example

```
converted_str = sms.convert_character(str1, 1, 2);
```

3.16.5 sms.send

Description

The function is used to send message.

Prototype	If AT+CMGF=1, send TEXT message: boolean int sms.send(string dest_addr, string sms_data, int msg_ref, int seq_num, int total_sm) If AT+CMGF=0, send PDU message: boolean int sms.send(int len, string sms_pdu_data

	)
Parameters	string dest_addr : destination address string sms_data : the SMS data to be sent int msg_ref : the message reference int total_sm : total number of long SMS int seq_num : the sequence number of long SMS string sms_pdu_data : the SMS PDU data to be sent int len : the length of the actual TP data unit in octets. (i.e. the RP layer SMSC address octets are not counted in the length)
Return value	boolean: the result of sending int: If successful, it is the message reference, or else, it is the error code.

Example

```
local msg_content, rst, suc, msg_ref_or_err_cause;
local msg_ref, total_sm, seq_num;
----------------------------WRITE TXT SMS-----------------------------------
print("send single IRA sms\r\n");
local dest_phone_number = "18652845698";
os.set_cscs(CSCS_IRA);
sms.set_cmgf(1);
sms.set_csmp(17, 14, 0, 0);
msg_content = "test content single sms";
suc, msg_ref_or_err_cause = sms.send(dest_phone_number, msg_content);--send single sms, default is "UNSENT"
print("sms.send=", suc, ",", msg_ref_or_err_cause, "\r\n");
```

3.16.6 sms.write

Description

The function is used to write message.

Prototype	If AT+CMGF=1, write TEXT message: boolean int sms.write(string dest_addr, string sms_data, int stat, int msg_ref, int seq_num, int total_sm) If AT+CMGF=0, write PDU message: boolean int sms.write(int len, string sms_pdu_data

	)
Parameters	int len : the length of the actual TP data unit in octets. (i.e. the RP layer SMSC address octets are not counted in the length) string sms_pdu_data : the SMS PDU data to be write
Return value	boolean: the result of writing int: If successful, it is the message reference, or else, it is the error code.

Example

```
local msg_index, msg_content, rst, stat, suc, msg_ref_or_err_cause;
local msg_ref, total_sm, seq_num;
----------------------------WRITE TXT SMS-----------------------------------
print("write single IRA sms\r\n");
local dest_phone_number = "18652845698";
os.set_cscs(CSCS_IRA);
sms.set_cmgf(1);
sms.set_csmp(17, 14, 0, 0);
msg_content = "test content1";
suc, msg_ref_or_err_cause = sms.write(dest_phone_number, msg_content);--write single sms, default is "UNSENT"
print("sms.write=", suc, ",", msg_ref_or_err_cause, "\r\n");
```

3.16.7 sms.read

Description

The function is used to read message like AT+CMGRO, which does not change the state of the SMS.

Prototype	int string sms.read(int index)
Parameters	int index: the index of the message
Return value	int: the error code, 0 is successful, other value is failed. string read_content: the read content like AT+CMGRO. In order to support long SMS, when AT+CMGF=1, it includes MSG_REF, TOTAL_SM, SEQ_NUM before the SMS data.

Example

```
err_code, read_content = sms.read(1);
```

3.16.8 sms.modify_tag

Description

The function is used to modify message tag from unread to read like AT+CMGMT.

Prototype	boolean sms.modify_tag(int index)
Parameters	int index: the index of the message
Return value	the result of the accquiring: true: successful false: failed

Example

```
printdir(1);
rst = sms.modify_tag(3);--equal AT+CMGMT=3
print("sms modify msg tag, rst=", rst, "\r\n");
if (true) then
  return;
end;
```

3.16.9 sms.get_csdh

Description

The function is used to get the AT+CSDH setting.

Prototype	int sms.get_csdh()
Parameters	None
Return value	int: the csdh value

Example

```
val = sms.get_csdh();
```

3.16.10sms.set_csdh

Description

The function is used to set the CSDH setting.

Prototype	int sms.set_csdh(int csdh)
Parameters	int: the csdh value
Return value	true: successful false: failed to set

Example

```
result= sms.set_csdh(1);
```

3.16.11sms.cpms

Description

The function is used to get or set the AT+CPMS setting.

Prototype	table sms.cpms([int mem1, int mem2, int mem3])
Parameters	int mem1, mem2, mem3: CPMS_MEM_STORE_ME = 0; CPMS_MEM_STORE_MT = 1;

	CPMS_MEM_STORE_SM = 2; CPMS_MEM_STORE_SR = 3;
Return value	Table: the result of the query or set operation.

Example

```
function print_cpms_info(cpms_info)
    if (not cpms_info) then
        print("cpms information is nil\r\n");
        return;
    end;
    print("cpms information={\r\n");
    print("        {\r\n");
    print("                mem1_store=",  cpms_info.mem1_store,  "\r\n");
    print("                mem1_used=",  cpms_info.mem1_used, "\r\n");
    print("                mem1_max=",  cpms_info.mem1_max,  "\r\n");
    print("        }\r\n");
    print("        {\r\n");
    print("                mem2_store=",  cpms_info.mem2_store,  "\r\n");
    print("                mem2_used=",  cpms_info.mem2_used,  "\r\n");
    print("                mem2_max=", cpms_info.mem2_max, "\r\n");
    print("        }\r\n");
    print("        {\r\n");
    print("                mem3_store=",  cpms_info.mem3_store,  "\r\n");
    print("                mem3_used=",  cpms_info.mem3_used,  "\r\n");
    print("                mem3_max=",  cpms_info.mem3_max, "\r\n");
    print("        }\r\n");
    print("}\r\n");
end;
function query_cpms_info()
    print("query_cpms_info\r\n");
    local cpms_info = sms.cpms();
    print_cpms_info(cpms_info);
    return cpms_info;
end;
function change_cpms(mem1, mem2, mem3)
    print("change_cpms, mem1=", mem1, ", mem2=", mem2, ", mem3=", mem3, "\r\n");
    local cpms_info = sms.cpms(mem1, mem2, mem3);
    print_cpms_info(cpms_info);
    return cpms_info;
end;
function test_cpms()
    print("-------test cpms-------\r\n");
    CPMS_MEM_STORE_ME = 0;
    CPMS_MEM_STORE_MT = 1;
    CPMS_MEM_STORE_SM = 2;
```

```
  CPMS_MEM_STORE_SR = 3;

  if (not query_cpms_info()) then
    print("failed to query cpms information\r\n");
    return;
  end;

  if (not change_cpms(CPMS_MEM_STORE_ME, CPMS_MEM_STORE_SM,
CPMS_MEM_STORE_SM)) then
    print("failed to change cpms\r\n");
    return;
  end;
end;
```

3.16.12sms.get_csmp

Description

The function is used to get the AT+CSMP setting.

Prototype	table sms.get_csmp()
Parameters	None
Return value	Table: the result of the query operation.

Example

```
  function print_csmp_info(csmp_info)
    if (not csmp_info) then
      print("csmp information is nil\r\n");
      return;
    end;
    print("csmp information={\r\n");
    print("      fo=", csmp_info.fo, "\r\n");
    print("      vp=", csmp_info.vp, "\r\n");
    print("      pid=", csmp_info.pid, "\r\n");
    print("      dcs=", csmp_info.dcs, "\r\n");
    print("}\r\n");
  end;
  function query_csmp_info()
    print("query_csmp_info\r\n");
    local csmp_info = sms.get_csmp();
    print_csmp_info(csmp_info);
    return csmp_info;
  end;
```

3.16.13sms.set_csmp

Description

The function is used to set the AT+CSMP setting.

Prototype	boolean sms.set_csmp(int fo, int vp, int pid, int dcs)
Parameters	int fo: Depending on the Command or result code: first octet of GSM 03.40 SMS-DELIVER, SMS-SUBMIT (default 17), SMS-STATUS-REPORT, or SMS-COMMAND (default 2) in integer format. SMS status report is supported under text mode if <fo> is set to 49. int vp: Depending on SMS-SUBMIT <fo> setting: GSM 03.40,TP-Validity-Period either in integer format (default 167), in time-string format, or if is supported, in enhanced format (hexadecimal coded string with quotes), (<vp> is in range 0... 255). int pid: GSM 03.40 TP-Protocol-Identifier in integer format (default 0). int dcs: GSM 03.38 SMS Data Coding Scheme (default 0), or Cell Broadcast Data Coding Scheme in integer format depending on the command or result code.
Return value	boolean: the result of operation: true: successful false: failed to operate

Example

```
function change_csmp(fo, vp, pid, dcs)
    print("change_csmp, fo=", fo, ", vp=", vp, ", pid=", pid, ", dcs=", dcs, "\r\n");
    local result = sms.set_csmp(fo, vp, pid, dcs);
    print("sms.set_csmp(), result=", result, "\r\n");
    return result;
end;
```

3.16.14sms.get_cnmi

Description

The function is used to get the AT+CNMI setting.

Prototype	table sms.get_cnmi()
Parameters	None
Return value	Table: the result of the query operation.

Example

```
function print_cnmi_info(cnmi_info)
    if (not cnmi_info) then
```

```
        print("cnmi information is nil\r\n");
        return;
    end;
    print("cnmi information={\r\n");
    print("      mode=", cnmi_info.mode, "\r\n");
    print("      mt=", cnmi_info.mt, "\r\n");
    print("      bm=", cnmi_info.bm, "\r\n");
    print("      ds=", cnmi_info.ds, "\r\n");
    print("      bfr=", cnmi_info.bfr, "\r\n");
    print("}\r\n");
end;
function query_cnmi_info()
    print("query_cnmi_info\r\n");
    local cnmi_info = sms.get_cnmi();
    print_cnmi_info(cnmi_info);
    return cnmi_info;
end;
```

3.16.15sms.set_cnmi

Description

The function is used to set the AT+CNMI setting.

Prototype	boolean sms.set_cnmi(int mode, int mt, int bm, int ds, int bfr)
Parameters	int mode, mt, bm, ds, bfr: Please refer to AT+CNMI
Return value	boolean: the result of operation: true: successful false: failed to operate

Example

```
function change_cnmi(mode, mt, bm, ds, bfr)
    print("change_cnmi, mode=", mode, ", mt=", mt, ", bm=", bm, ", ds=", ds, ", bfr=",
bfr,"\r\n");
    local result = sms.set_cnmi(mode, mt, bm, ds, bfr);
    print("sms.set_cnmi(), result=", result, "\r\n");
    return result;
end;
```

3.16.16sms.get_csca

Description

The function is used to get the AT+CSCA setting. The result is always ASCII phone number or nil.

Prototype	string sms.get_csca()
Parameters	None
Return value	string: the result of the query operation.

Example

```
function print_csca_info(csca_info)
    if (not csca_info) then
        print("csca information is nil\r\n");
        return;
    end;
    print("csca = \"", csca_info, "\"\r\n");
end;
function query_csca_info()
    print("query_csca_info\r\n");
    local csca_info = sms.get_csca();
    print_csca_info(csca_info);
    return csca_info;
end;
```

3.16.17sms.set_csca

Description

The function is used to set the AT+CSCA setting. The input phone number is always ASCII format.

Prototype	boolean sms.set_sca(string csca)
Parameters	string csca: the AT+CSCA setting.
Return value	boolean: the result of operation: true: successful false: failed to operate

Example

```
function change_csca(new_csca)
    print("change_csca, csca=", new_csca,"\r\n");
    local result = sms.set_csca(new_csca);
    print("sms.set_csca(), result=", result, "\r\n");
    return result;
end;
```

3.16.18sms.delete

Description

The function is used to delete SMS.

Prototype	boolean sms.delete(int msg_index[, int delflag])
Parameters	int msg_index, the index of the message to delete. int delflag: this can be set to the combination of the

	following value: SMS_DELETE_FLAG_NONE = 0--default SMS_DELETE_FLAG_READ = 1 SMS_DELETE_FLAG_UNREAD = 2 SMS_DELETE_FLAG_SENT = 4 SMS_DELETE_FLAG_UNSENT = 8 SMS_DELETE_FLAG_SENT_ST_NOT_RECEIVED = 16 SMS_DELETE_FLAG_ALL = 0xFF
Return value	boolean: the result of operation: true: successful false: failed to operate

Example

```
SMS_DELETE_FLAG_NONE = 0--default
SMS_DELETE_FLAG_READ = 1
SMS_DELETE_FLAG_UNREAD = 2
SMS_DELETE_FLAG_SENT = 4
SMS_DELETE_FLAG_UNSENT = 8
SMS_DELETE_FLAG_SENT_ST_NOT_RECEIVED = 16
SMS_DELETE_FLAG_ALL = 0xFF

local msg_index, delflag, rst;
msg_index = 1;
rst = sms.delete(msg_index);
print("sms.delete[", msg_index, "]=", rst, "\r\n");
delflag = SMS_DELETE_FLAG_READ + SMS_DELETE_FLAG_UNSENT;
rst = sms.delete(0, delflag);
print("sms.delete[0,", delflag, "]=", rst, "\r\n");
delflag = SMS_DELETE_FLAG_ALL;
rst = sms.delete(0, delflag);
print("sms.delete[0,", delflag, "]=", rst, "\r\n");
```

3.17 NET Library

3.17.1 net.creg

Description

The function is used to get the cs-domain register result. For the corresponding AT command, please refer to AT+CREG.

Prototype	int int int net.creg ()
Parameters	None
Return value	int status: the result of the cs-domain register result.

	int lac: LAC int cell_id: cell ID

Example

```
reg_status, lac, cell_id = net.creg();
```

3.17.2 net.cgreg

Description

The function is used to get the ps-domain register result. For the corresponding AT command, please refer to AT+CGREG.

Prototype	int int int net.cgreg ()
Parameters	None
Return value	int status: the result of the ps-domain register result. int lac: LAC int cell_id: cell ID

Example

```
reg_status, lac, cell_id = net.cgreg();
```

3.17.3 net.csq

Description

The function is used to get the CSQ value. For the corresponding AT command, please refer to AT+CSQ.

Prototype	int int net.csq ()
Parameters	None
Return value	The are two return values: csq: the CSQ value err_bit: the ERROR BIT

Example

```
rst= net.csq();
print(rst,"\r\n");
```

3.17.4 net.cnsmod

Description

The function is used to query the network mode. For the corresponding AT command, please refer to AT+CNSMOD.

Prototype	int net.cnsmod ()
Parameters	None
Return value	the network mode: 0: no service 1: GSM 2: GPRS

	3: EGPRS (EDGE) 4: WCDMA 5: HSDPA only 6: HSUPA only 7: HSPA(HSDPA and HSUPA)

Example

```
rst= net.cnsmode();
print(rst,"\r\n");
```

3.17.5 net.cnti

Description

The function is used to get the AT*CNTI? value.

Prototype	int net.cnti ()
Parameters	None
Return value	the network mode: CNTI_NONE = 0 CNTI_GSM = 1 CNTI_GPRS = 2 CNTI_EGPRS = 3 CNTI_UMTS = 4 CNTI_HSDPA = 5 CNTI_HSUPA = 6 CNTI_HSPA = 7

Example

```
local cnti_value = net.cnti();
```

3.18 MultiMedia Library

3.18.1 multimedia.play_wav

Description

The function is used to play wave file. It is the same as AT+CCMXPLAYWAV.

Prototype	boolean multimedia.play_wav(string filepath, int direction)
Parameters	string filepath: the wave file path. int direction: the direction of playing: 1=play to local 2=play to remote in voice call
Return value	the result of the operation: true: successful false: failed

Example

```
print(1);
result = multimedia.play_wav(wav_path, WAVE_PLAY_TO_LOCAL);
print("multimedia.play_wav(), result=", result, "\r\n");
```

3.18.2 multimedia.stop_wav

Description

The function is used to stop playing wave file. It is the same as AT+CCMXSTOPWAV.

Prototype	boolean multimedia.stop_wav(void)
Parameters	None
Return value	the result of the operation: true: successful false: failed

Example

```
print(1);
result = multimedia.stop_wav();
print("multimedia.stop_wav=", result, "\r\n");
```

3.18.3 multimedia.wav_state

Description

The function is used to get the sate of playing wave file. It is the same as AT+CCMXWAV.STATE

Prototype	int multimedia.wav_state(void)
Parameters	None
Return value	the state of playing wave file: 0=stop 1=play

Example

```
print(1);
local state = multimedia.wav_state();
print("wave state=", state, "\r\n");
```

3.18.4 multimedia.play_tone

Description

The function is used to play tone

Prototype	boolean multimedia.play_tone(int tone_id [,int dev_id])
Parameters	int tone_id, SND_ALERT = 0;--ring for incomging call SND_WAKEUP = 1; --Wake-up/Power-up sound SND_DIAL_TONE = 2; --Dial tone

	SND_DTACO_ROAM_TONE = 3 -- DTACO roaming dial tone SND_RING_BACK = 4 -- Ring-back sound SND_INTERCEPT = 5 -- Send request intercepted locally SND_REORDER = 6 -- System busy SND_BUSY_ALERT = 7 -- Busy Signal SND_CONFIRMATION = 8 -- Confirmation Tone SND_CALL_WAITING = 9 -- Call Waiting SND_ANSWER = 10 -- 10 - Answer Tone SND_OFF_HOOK = 11 -- Off-Hook Warning SND_NORMAL_ALERT = 12 -- "Normal" Alerting SND_INTR_GROUP_ALERT = 13 -- Intergroup Alerting SND_SPCL_ALERT = 14 -- Special/Priority Alerting SND_PING_RING = 15 -- "Ping ring" SND_IS54B_LONG_H_ALERT = 16 -- IS-54B High Long SND_IS54B_SS_H_ALERT = 17 -- IS-54B High Short-short SND_IS54B_SSL_H_ALERT = 18 -- IS-54B High Short-short-long SND_IS54B_SS2_H_ALERT = 19 -- IS-54B High Short-short-2 SND_IS54B_SLS_H_ALERT = 20 -- 20 - IS-54B High Short-long-short SND_IS54B_SSSS_H_ALERT = 21 -- IS-54B High Short-short-short-short SND_IS54B_PBX_LONG_H_ALERT = 22 -- IS-54B High PBX Long SND_IS54B_PBX_SS_H_ALERT = 23 -- IS-54B High PBX Short-short SND_IS54B_PBX_SSL_H_ALERT = 24 -- IS-54B High PBX Short-short-long

	SND_IS54B_PBX_SLS_H_ALERT = 25 -- IS-54B High PBX Short-long-short SND_IS54B_PBX_SSSS_H_ALERT = 26 -- IS-54B High PBX Short-short-short-short SND_IS53A_PPPP_H_ALERT = 27 -- IS-53A High Pip-Pip-Pip-Pip Alert SND_IS54B_LONG_M_ALERT = 28 -- IS-54B Medium Long SND_IS54B_SS_M_ALERT = 29 -- IS-54B Medium Short-short SND_IS54B_SSL_M_ALERT = 30 -- 30 - IS-54B Medium Short-short-long SND_IS54B_SS2_M_ALERT = 31 -- IS-54B Medium Short-short-2 SND_IS54B_SLS_M_ALERT = 32 -- IS-54B Medium Short-long-short SND_IS54B_SSSS_M_ALERT = 33 -- IS-54B Medium Short-short-short-short SND_IS54B_PBX_LONG_M_ALERT = 34 -- IS-54B Medium PBX Long SND_IS54B_PBX_SS_M_ALERT = 35 -- IS-54B Medium PBX Short-short SND_IS54B_PBX_SSL_M_ALERT = 36 -- IS-54B Medium PBX Short-short-long SND_IS54B_PBX_SLS_M_ALERT = 37 -- IS-54B Medium PBX Short-long-short SND_IS54B_PBX_SSSS_M_ALERT = 38 -- IS-54B Medium PBX Short-short-short-short SND_IS53A_PPPP_M_ALERT = 39 -- IS-53A Medium Pip-Pip-Pip-Pip Alert SND_IS54B_LONG_L_ALERT = 40 -- 40 - IS-54B Low Long SND_IS54B_SS_L_ALERT = 41 -- IS-54B Low Short-short SND_IS54B_SSL_L_ALERT = 42 -- IS-54B Low Short-short-long SND_IS54B_SS2_L_ALERT = 43 -- IS-54B Low Short-short-2 SND_IS54B_SLS_L_ALERT = 44 -- IS-54B Low Short-long-short SND_IS54B_SSSS_L_ALERT = 45 -- IS-54B Low Short-short-short-short SND_IS54B_PBX_LONG_L_ALERT = 46 -- IS-54B Low PBX Long

	SND_IS54B_PBX_SS_L_ALERT = 47 -- IS-54B Low PBX Short-short SND_IS54B_PBX_SSL_L_ALERT = 48 -- IS-54B Low PBX Short-short-long SND_IS54B_PBX_SLS_L_ALERT = 49 -- IS-54B Low PBX Short-long-short SND_IS54B_PBX_SSSS_L_ALERT = 50 -- 50 - IS-54B Low PBX Short-short-short-shrt SND_IS53A_PPPP_L_ALERT = 51 -- IS-53A Low Pip-Pip-Pip-Pip Alert SND_FADE_TONE = 52 -- Tone to inform user of a fade SND_SVC_CHANGE = 53 -- Inform user of a service area change SND_HORN_ALERT = 54 -- Horn alert SND_ABRV_REORDER = 55 -- Abbreviated System busy SND_ABRV_INTERCEPT = 56 -- Abbrev'd Send request intercepted locally SND_ALTERNATE_REORDER = 57 -- Alternate reorder SND_MESSAGE_ALERT = 58 -- Message Waiting Signal SND_ABRV_ALERT = 59 -- Abbreviated alert SND_PIP_TONE = 60 -- 60 - Pip Tone (Voice Mail Waiting) SND_ROAM_RING = 61 -- Ringing option while roaming SND_SVC_ACQ = 62 -- Service acquired sound SND_SVC_LOST = 63 -- Service lost sound SND_SVC_CHNG_MORE_PREF = 64 -- Change to a more preferred service sound SND_SVC_CHNG_LESS_PREF = 65 -- Change to a less preferred service sound SND_EXT_PWR_ON = 66 -- External power on sound SND_EXT_PWR_OFF = 67 -- External power off sound SND_RING_1 = 68 -- User selectable ring 1

	SND_RING_2 = 69 -- User selectable ring 2 SND_RING_3 = 70 -- 70 - User selectable ring 3 SND_RING_4 = 71 -- User selectable ring 4 SND_RING_5 = 72 -- User selectable ring 5 SND_RING_6 = 73 -- User selectable ring 6 SND_RING_7 = 74 -- User selectable ring 7 SND_RING_8 = 75 -- User selectable ring 8 SND_RING_9 = 76 -- User selectable ring 9 SND_VR_HFK_CALL_RECEIVED = 77 -- Call answer sound when in HFK SND_HFK_CALL_ORIG = 78 -- Call origination sound when in HFK SND_SPECIAL_INFO = 79 -- Special info sound -- GSM tones defined in 3GPP 2.40 SND_SUBSCRIBER_BUSY = 80 -- 80 - Subscriber busy sound SND_CONGESTION = 81 -- Congestion sound SND_ERROR_INFO = 82 -- Error information sound SND_NUMBER_UNOBTAINABLE = 83 -- Number unobtainable sound SND_AUTH_FAILURE = 84 -- Authentication failure sound SND_RADIO_PATH_ACK = 85 -- Radio path acknowledgement sound SND_RADIO_PATH_NOT_AVAIL = 86 -- Radio path not available sound SND_CEPT_CALL_WAITING = 87 -- CEPT call waiting sound SND_CEPT_RING = 88 -- CEPT ringing sound SND_CEPT_DIAL_TONE = 89 -- CEPT dial tone

	SND_BRITISH_BUSY_TONE = 90 --British busy tone dev_id: 0: current device decided using AT+CSDVC 1: handset 2: headset 3: speaker
Return value	the result of the operation: true: successful false: failed

Example

```
print(1);
SND_ALERT = 0;--ring for incomging call
SND_WAKEUP = 1; --Wake-up/Power-up sound
SND_DIAL_TONE = 2; --Dial tone
SND_DTACO_ROAM_TONE = 3          -- DTACO roaming dial tone
SND_RING_BACK = 4                -- Ring-back sound
SND_INTERCEPT = 5                -- Send request intercepted locally
SND_REORDER = 6                  -- System busy
SND_BUSY_ALERT = 7               -- Busy Signal
SND_CONFIRMATION = 8             -- Confirmation Tone
SND_CALL_WAITING = 9             -- Call Waiting
SND_ANSWER = 10                  -- 10 - Answer Tone
SND_OFF_HOOK =11                 -- Off-Hook Warning
SND_NORMAL_ALERT = 12            -- "Normal" Alerting
SND_INTR_GROUP_ALERT = 13        -- Intergroup Alerting
SND_SPCL_ALERT = 14              -- Special/Priority Alerting
SND_PING_RING = 15               -- "Ping ring"
SND_IS54B_LONG_H_ALERT =16       -- IS-54B High Long
SND_IS54B_SS_H_ALERT = 17        -- IS-54B High Short-short
SND_IS54B_SSL_H_ALERT = 18       -- IS-54B High Short-short-long
SND_IS54B_SS2_H_ALERT = 19       -- IS-54B High Short-short-2
SND_IS54B_SLS_H_ALERT = 20       -- 20 - IS-54B High Short-long-short
SND_IS54B_SSSS_H_ALERT = 21      -- IS-54B High Short-short-short-short
SND_IS54B_PBX_LONG_H_ALERT = 22  -- IS-54B High PBX Long
SND_IS54B_PBX_SS_H_ALERT = 23    -- IS-54B High PBX Short-short
SND_IS54B_PBX_SSL_H_ALERT = 24   -- IS-54B High PBX Short-short-long
SND_IS54B_PBX_SLS_H_ALERT = 25   -- IS-54B High PBX Short-long-short
SND_IS54B_PBX_SSSS_H_ALERT = 26  -- IS-54B High PBX Short-short-short-short
SND_IS53A_PPPP_H_ALERT = 27      -- IS-53A High Pip-Pip-Pip-Pip Alert
SND_IS54B_LONG_M_ALERT =28       -- IS-54B Medium Long
SND_IS54B_SS_M_ALERT = 29        -- IS-54B Medium Short-short
```

```
SND_IS54B_SSL_M_ALERT = 30          -- 30 - IS-54B Medium Short-short-long
SND_IS54B_SS2_M_ALERT = 31          -- IS-54B Medium Short-short-2
SND_IS54B_SLS_M_ALERT = 32          -- IS-54B Medium Short-long-short
SND_IS54B_SSSS_M_ALERT = 33         -- IS-54B Medium Short-short-short-short
SND_IS54B_PBX_LONG_M_ALERT = 34     -- IS-54B Medium PBX Long
SND_IS54B_PBX_SS_M_ALERT = 35       -- IS-54B Medium PBX Short-short
SND_IS54B_PBX_SSL_M_ALERT = 36      -- IS-54B Medium PBX Short-short-long
SND_IS54B_PBX_SLS_M_ALERT = 37      -- IS-54B Medium PBX Short-long-short
SND_IS54B_PBX_SSSS_M_ALERT = 38     -- IS-54B Medium PBX Short-short-short-short
SND_IS53A_PPPP_M_ALERT = 39         -- IS-53A Medium Pip-Pip-Pip-Pip Alert
SND_IS54B_LONG_L_ALERT =40          -- 40 - IS-54B Low Long
SND_IS54B_SS_L_ALERT = 41           -- IS-54B Low Short-short
SND_IS54B_SSL_L_ALERT = 42          -- IS-54B Low Short-short-long
SND_IS54B_SS2_L_ALERT = 43          -- IS-54B Low Short-short-2
SND_IS54B_SLS_L_ALERT = 44          -- IS-54B Low Short-long-short
SND_IS54B_SSSS_L_ALERT = 45         -- IS-54B Low Short-short-short-short
SND_IS54B_PBX_LONG_L_ALERT = 46     -- IS-54B Low PBX Long
SND_IS54B_PBX_SS_L_ALERT = 47       -- IS-54B Low PBX Short-short
SND_IS54B_PBX_SSL_L_ALERT = 48      -- IS-54B Low PBX Short-short-long
SND_IS54B_PBX_SLS_L_ALERT = 49      -- IS-54B Low PBX Short-long-short
SND_IS54B_PBX_SSSS_L_ALERT = 50     -- 50 - IS-54B Low PBX Short-short-short-shrt
SND_IS53A_PPPP_L_ALERT = 51         -- IS-53A Low Pip-Pip-Pip-Pip Alert
SND_FADE_TONE = 52                  -- Tone to inform user of a fade
SND_SVC_CHANGE = 53                 -- Inform user of a service area change
SND_HORN_ALERT = 54                 -- Horn alert
SND_ABRV_REORDER = 55               -- Abbreviated System busy
SND_ABRV_INTERCEPT = 56             -- Abbrev'd Send request intercepted locally
SND_ALTERNATE_REORDER = 57          -- Alternate reorder
SND_MESSAGE_ALERT = 58              -- Message Waiting Signal
SND_ABRV_ALERT = 59                 -- Abbreviated alert
SND_PIP_TONE = 60                   -- 60 - Pip Tone (Voice Mail Waiting)
SND_ROAM_RING = 61                  -- Ringing option while roaming
SND_SVC_ACQ =62                     -- Service acquired sound
SND_SVC_LOST = 63                   -- Service lost sound
SND_SVC_CHNG_MORE_PREF = 64         -- Change to a more preferred service sound
SND_SVC_CHNG_LESS_PREF = 65         -- Change to a less preferred service sound
SND_EXT_PWR_ON =66                  -- External power on sound
SND_EXT_PWR_OFF = 67                -- External power off sound
SND_RING_1 = 68                     -- User selectable ring 1
SND_RING_2 = 69                     -- User selectable ring 2
SND_RING_3 = 70                     -- 70 - User selectable ring 3
SND_RING_4 = 71                     -- User selectable ring 4
SND_RING_5 = 72                     -- User selectable ring 5
SND_RING_6 = 73                     -- User selectable ring 6
```

```
SND_RING_7 = 74                    -- User selectable ring 7
SND_RING_8 = 75                    -- User selectable ring 8
SND_RING_9 = 76                    -- User selectable ring 9
SND_VR_HFK_CALL_RECEIVED = 77      -- Call answer sound when in HFK
SND_HFK_CALL_ORIG =78              -- Call origination sound when in HFK
SND_SPECIAL_INFO = 79              -- Special info sound
                                   -- GSM tones defined in 3GPP 2.40
SND_SUBSCRIBER_BUSY = 80           -- 80 - Subscriber busy sound
SND_CONGESTION = 81                -- Congestion sound
SND_ERROR_INFO = 82                -- Error information sound
SND_NUMBER_UNOBTAINABLE = 83       -- Number unobtainable sound
SND_AUTH_FAILURE =84               -- Authentication failure sound
SND_RADIO_PATH_ACK = 85            -- Radio path acknowledgement sound
SND_RADIO_PATH_NOT_AVAIL = 86      -- Radio path not available sound
SND_CEPT_CALL_WAITING = 87         -- CEPT call waiting sound
SND_CEPT_RING =88                  -- CEPT ringing sound
SND_CEPT_DIAL_TONE = 89            -- CEPT dial tone

SND_BRITISH_BUSY_TONE = 90                 --British busy tone
local tone_id =SND_ALERT;
result = multimedia.play_tone(tone_id);
print("multimedia.play_tone, result=", result, "\r\n");
```

3.18.5 multimedia.stop_tone

Description

The function is used to stop playing tone

Prototype	void multimedia.stop_tone(vooid)
Parameters	None
Return value	None

Example

```
print(1);
multimedia.stop_tone();
```

3.19 SOCKET Library

3.19.1 network.open

Description

The function is used to open the PS network.

Prototype	int network.open(int cid, int timeout)
Parameters	int cid: the cid of AT+CGSOCKCONT int timeout:

	<=0 : wait for ever >0 : the milliseconds to wait
Return value	int app_handle: .the handle to the opened network.

Example

```
printdir(1)
print("opening network...\r\n");
local cid = 1;--0=>use setting of AT+CSOCKSETPN. 1-16=>use self defined cid
local timeout = 30000;--    '<= 0' means wait for ever; '> 0' is the timeout milliseconds
local app_handle = network.open(cid, timeout);
if (not app_handle) then
    print("faield to open network\r\n");
    return;
end;
print("network.open(), app_handle=", app_handle, "\r\n");
```

3.19.2 network.close

Description

The function is used to close the PS network.

Prototype	boolean network.close(int app_handle)
Parameters	int app_handle: the handle of the PS network.
Return value	boolean result: true: successful. false: failed.

Example

```
printdir(1)
print("closing network...\r\n");
result = network.close(app_handle);
print("network.close(), result=", result, "\r\n");
```

3.19.3 network.status

Description

The function is used to query the status of the PS network.

Prototype	int network.status(int app_handle)
Parameters	int app_handle: the handle of the PS network.
Return value	int status: 0: invalid. 1: down. 2: coming up 4: up 8: going down 16: resuming

	32: going null 64: null

Example

```
printdir(1)
local status = network.status(app_handle);
print("network status = ", status, "\r\n");
```

3.19.4 network.dorm

Description

The function is used to let the PS network enter or leave dormancy state.

Prototype	void network.dorm(int app_handle, boolean operation)
Parameters	int app_handle: the handle of the PS network. Boolean operation: true: enter dormancy false: leave dormancy
Return value	boolean result: true: successful. false: failed.

Example

```
printdir(1)
local status = network.status(app_handle);
print("network status = ", status, "\r\n");
print("enter dormancy now...\r\n");
network.dorm(app_handle, true);
vmsleep(1000);
status = network.status(app_handle);
print("network status = ", status, "\r\n");
print("leave dormancy now...\r\n");
network.dorm(app_handle, false);
vmsleep(3000);
status = network.status(app_handle);
print("network status = ", status, "\r\n");
```

3.19.5 network.resolve

Description

The function is used to resolve domain name using DNS. The network.resolve() function is a standalone function for DNS domain name resolve, it is only related to AT+CIPDNSSET and AT+CSOCKSETPN, it has no relation with network.open() and network.close().

Prototype	string network.resolve(string domain_name [, cid])

Parameters	string domain_name: the domain name int cid: the profile number of CGSOCKCONT. By default it is 0, and use AT+CSOCKSETPN.
Return value	string: the IP address resolved.

Example

```
printdir(1)
print("resolving DNS address...\r\n");
local server_address = "www.baidu.com";
--[[
network.resolve() is a standalone function for DNS domain name resolve,
it is only related to AT+CIPDNSSET and AT+CSOCKSETPN,
it has no relation with network.open()
]]
local ip_address = network.resolve(server_address);
print("The IP address for ", server_address, " is ", ip_address, "\r\n");
```

3.19.6 network.local_ip

Description

The function is used to get the local IP address.

Prototype	string network.local_ip(int app_handle)
Parameters	int app_handle: the handle of the PS network.
Return value	string: the local IP address.

Example

```
printdir(1)
local local_ip_addr = network.local_ip(app_handle);
print("local ip address is ", local_ip_addr, "\r\n");
```

3.19.7 network.mtu

Description

The function is used to get the MTU setting of the opened network.

Prototype	int network.mtu(int app_handle)
Parameters	int app_handle: the handle of the PS network.
Return value	int: the MTU value. It is nil if failed.

Example

```
printdir(1)
local mtu_value = network.mtu(app_handle);
```

3.19.8 network.primary_dns

Description

The function is used to get the primary DNS address for an opened network.

Prototype	string network.primary_dns(int app_handle)
Parameters	int app_handle: the handle of the PS network.
Return value	string: the primary DNS address.

Example

```
printdir(1)
local primary_dns_addr = network.primary_dns(app_handle);
print("primary DNS address is ", primary_dns_addr, "\r\n");
```

3.19.9 network.secondary_dns

Description

The function is used to get the secondary DNS address for an opened network.

Prototype	string network.secondary_dns(int app_handle)
Parameters	int app_handle: the handle of the PS network.
Return value	string: the secondary DNS address.

Example

```
printdir(1)
local secondary_dns_addr = network.secondary_dns(app_handle);
print("secondary DNS address is ", secondary_dns_addr, "\r\n");
```

3.19.10network.change_primary_dns

Description

The function is used to set the primary DNS address for an opened network.

Prototype	boolean network.change_primary_dns(int app_handle, string address)
Parameters	int app_handle: the handle of the PS network. string address: the IP address to set
Return value	boolean: the result of operation: true: successful false: failed

Example

```
printdir(1)
local result = network.change_primary_dns(app_handle, "210.22.84.3");
print("Change primary DNS address, result=", result, "\r\n");
```

3.19.11network.change_secondary_dns

Description

The function is used to set the secondary DNS address for an opened network.

Prototype	boolean network.change_secondary_dns(int app_handle, string address)
Parameters	int app_handle: the handle of the PS network. string address: the IP address to set
Return value	boolean: the result of operation: true: successful false: failed

Example

```
printdir(1)
local result = network.change_secondary_dns(app_handle, "210.22.70.3");
print("Change secondary DNS address, result=", result, "\r\n");
```

3.19.12network.set_tcp_ka_param

Description

The function is used to set the common TCP keep alive parameters. It is the same with AT+CTCPKA.

Prototype	boolean network.set_tcp_ka_param(int max_check_times, int interval)
Parameters	int max_check_timers: maximum checking times, int interval: the checking interval in minute.
Return value	boolean result: true: successful. false: failed.

Example

```
printdir(1)
--The same with AT+CTCPKA
result = network.set_tcp_ka_param(5, 1);
--keep alive parameter, max 5 times, check every 1 minute if socket is idle.
print("network.set_tcp_ka_param()=", result, "\r\n");
```

3.19.13network.set_tcp_retran_param

Description

The function is used to set the TCP retransmit parameters. It is the same with AT+CIPCCFG.

Prototype	boolean network.set_tcp_retran_param(int max_backoff_times, int min_interval)
Parameters	int max_backoff_timers: maximum retransmit times,

	int min_interval: the minimum retransmit interval in millisecond.
Return value	boolean result: true: successful. false: failed.

Example

```
printdir(1)
--The same with AT+CIPCCFG
result = network.set_tcp_retran_param(10, 10000);
--maximum 10 retransmit times, minimum interval is 10 seconds.
print("network.set_tcp_retran_param()=", result, "\r\n");
```

3.19.14network.set_dns_timeout_param

Description

The function is used to set the DNS query timeout parameters. It is the same with AT+CIPDNSSET.

Prototype	boolean network.set_dns_timeout_param(int max_network_retry_open_times, int network_open_timeout, int max_dns_query_times)
Parameters	int max_network_retry_open_times: maximum retry times for opening PS network, int network_open_timeout: the timeout value for opening PS network in millisecond. int max_dns_query_times: maximum DNS query times.
Return value	boolean result: true: successful. false: failed.

Example

```
printdir(1)
--The same with AT+CIPDNSSET
result = network.set_dns_timeout_param(0, 30000, 5);
--[[network retry open times = 0(maximum is 3), network open timeout is 30 seconds, maximum DNS query times is 5]]
print("network.set_dns_timeout_param()=", result, "\r\n");
```

3.19.15socket.create

Description

The function is used to create a socket.

Prototype	int socket.create(int app_handle, int sock_type)
Parameters	int app_handle: the handle of the PS network.

	int sock_type: 0: TCP 1: UDP
Return value	int the created socket

Example

```
printdir(1)
local socket_fd = socket.create(app_handle, SOCK_TCP);
if (not socket_fd) then
    print("failed to create socket\r\n");
elseif (ip_address) then
    --enable keep alive
    socket.keepalive(socket_fd, true);--this depends on AT+CTCPKA command to set KEEP ALIVE interval
    print("socket_fd=", socket_fd, "\r\n");
end;
```

3.19.16socket.connect

Description

The function is used to connect the TCP server.

Prototype	boolean boolean socket.connect(int sockfd, string ip_address, int port, int timeout)
Parameters	int sockfd: the handle of the socket. string: the server IP address int port: the server listening port int timeout: <0 : wait for ever =0 : not wait >0 : the milliseconds to wait
Return value	boolean result: true: successful. false: failed. boolean socket_released: true: the socket is released false: the socket is not released(later the socket.close() must be called to release it)

Example

```
printdir(1)
print("connecting server...\r\n");
local timeout = 30000;
local connect_result, socket_released = socket.connect(socket_fd, ip_address, 80, timeout);
print("socket.connect = [result=", connect_result, ",socket_released=", socket_released, "]\r\n");
if (not connect_result) then
    print("failed to connect server\r\n");
```

```
else
   print("connect server succeeded\r\n");
end;
```

3.19.17socket.close

Description

The function is used to close the TCP connection. Also, finally it releases the socket handle.

Prototype	boolean socket.close(int sockfd)
Parameters	int sockfd: the handle of the socket.
Return value	boolean result: true: successful. false: failed.

Example

```
printdir(1)
print("closing socket...\r\n");
if (not socket.close(socket_fd)) then
   print("failed to close socket\r\n");
else
   print("close socket succeeded\r\n");
end;
```

3.19.18socket.bind

Description

The function is used to bind a port to a socket.

Prototype	boolean socket.bind(int sockfd, int port)
Parameters	int sockfd: the handle of the socket. int port: the port to bind.
Return value	boolean result: true: successful. false: failed.

Example

```
printdir(1)
socket.bind(socket_fd, listening_port)
```

3.19.19socket.listen

Description

The function is used to let a socket to listening state.

Prototype	boolean socket.listen(int sockfd, int backlog)
Parameters	int sockfd: the handle of the socket. int backlog: the maximum connections can be

	queued in listen queue.
Return value	boolean result: true: successful. false: failed.

Example

```
printdir(1)
if (not socket.bind(socket_fd, listening_port) or not socket.listen(socket_fd)) then
    print("failed to listen on port ", listening_port, "\r\n");
else
    print("listening on \"",local_ip_addr,":",listening_port,"\"...\r\n");
end;
```

3.19.20socket.accept

Description

The function is used to let a TCP socket to accept an incoming client.

Prototype	int int string int socket.accept(int sockfd, int timeout)
Parameters	int sockfd: the handle of the socket. int timeout: <0 : wait for ever =0 : not wait >0 : the milliseconds to wait
Return value	int error_code: the error code 0: OK 1: timeout 2: busy 3: parameter error 4: socket failure 5: network failure int accepted_handle the accepted socket handle string client_ip: the client IP address int client_port: the client port

Example

```
printdir(1)
local timeout = 60000;-- '< 0' means wait for ever; '0' means not wait; '> 0' is the timeout milliseconds
local err_code, accept_socket, client_ip, client_port = socket.accept(socket_fd, timeout);
print("socket.accept() = [", err_code, ",", accept_socket, ",", client_ip, ",", client_port, "]\r\n");
if (err_code == SOCK_RST_OK) then
    print("the accepted socket fd is ", accept_socket, "\r\n");
end;
```

3.19.21socket.send

Description

The function is used to let a TCP socket to send data.

Prototype	int int socket.send(int sockfd, string data, int timeout)
Parameters	int sockfd: the handle of the socket. stirng: the data to send. int timeout: <0 : wait for ever =0 : not wait >0 : the milliseconds to wait
Return value	int error_code: the error code 0: OK 1: timeout 2: busy 3: parameter error 4: socket failure 5: network failure int sent_len: the length of the sent data.

Example

```
printdir(1)
local err_code, sent_len = socket.send(socket_fd, http_req, timeout);
print("socket.send ", err_code, ", ", sent_len, "\r\n");
if (err_code and (err_code == SOCK_RST_OK)) then
   print("all data is sent\r\n");
end;
```

3.19.22socket.recv

Description

The function is used to let a TCP socket to receive data.

Prototype	int string socket.recv(int sockfd, int timeout)
Parameters	int sockfd: the handle of the socket. int timeout: <0 : wait for ever =0 : not wait >0 : the milliseconds to wait
Return value	int error_code: the error code 0: OK 1: timeout 2: busy 3: parameter error

	4: socket failure 5: network failure string: recv_data; the received data.

Example

```
printdir(1)
print("socket.recv()...\r\n");
local timeout = 15000;--  '< 0' means wait for ever; '0' means not wait; '> 0' is the timeout milliseconds
local err_code, http_rsp = socket.recv(socket_fd, timeout);
print("socket.recv(), err_code=", err_code, "\r\n");
if ((err_code == SOCK_RST_OK) and http_rsp) then
   if (printdir()) then
      os.printstr(http_rsp);--this can print string larger than 1024 bytes, and also it can print string including '\0'.
   end;
   print("\r\n");
else
   print("failed to call socket.recv\r\n");
end;
```

3.19.23socket.sendto

Description

The function is used to let a UDP socket to send data.

Prototype	int int socket.sendto(int sockfd, string ip_address, int port, string data, int timeout)
Parameters	int sockfd: the handle of the socket. string ip_address: the peer IP address int port: the peer port stirng: the data to send. int timeout: <0 : wait for ever =0 : not wait >0 : the milliseconds to wait
Return value	int error_code: the error code 0: OK 1: timeout 2: busy 3: parameter error 4: socket failure 5: network failure int sent_len: the length of the sent data.

Example

```
printdir(1)
```

```
local request = "Hello, this is LUA UDP Test\r\n";
local timeout = 30000;--  '< 0' means wait for ever; '0' means not wait; '> 0' is the timeout milliseconds
local err_code, sent_len = socket.sendto(socket_fd, peer_address,peer_port, request, timeout);
print("socket.sendto ", err_code, ", ", sent_len, "\r\n");
if (err_code and (err_code == SOCK_RST_OK)) then
   print("all data sent\r\n");
end;
```

3.19.24socket.recvfrom

Description

The function is used to let a UDP socket to receive data.

Prototype	int string int string socket.recvfrom(int sockfd, int timeout)
Parameters	int sockfd: the handle of the socket. int timeout: <0 : wait for ever =0 : not wait >0 : the milliseconds to wait
Return value	int error_code: the error code 0: OK 1: timeout 2: busy 3: parameter error 4: socket failure 5: network failure string from_address: the peer IP address int from_port: the peer port string: recv_data; the received data.

Example

```
printdir(1)
print("socket.recvfrom()...\r\n");
local timeout = 120000;
local err_code, from_address, from_port, response = socket.recvfrom(socket_fd, timeout);
print("socket.recvfrom() =[", err_code, ",", from_address, ",", from_port, "]\r\n");
if ((err_code == SOCK_RST_OK) and response) then
  if (printdir()) then
     os.printstr(response);
  end;
  print("\r\n");
end;
```

3.19.25socket.keepalive

Description

The function is used to enable the KEEP ALIVE function of a TCP socket. When enable KEEP ALIVE, the AT+CTCPKA must be used to set the interval of KEEP ALIVE.

Prototype	boolean socket.keepalive(int sockfd, boolean keepalive)
Parameters	int sockfd: the handle of the socket. boolean keepalive: true: enable false: disable.
Return value	boolean result: true: successful. false: failed.

Example

```
printdir(1)
socket.keepalive(socket_fd, true);
```

3.19.26 socket.select

Description

The function is used to select the care socket event.

Prototype	boolean socket.select(int sockfd, int event_mask)
Parameters	int sockfd: the handle of the socket. int event_mask: 1: write_event 2: read_event 4: close_event 8: accept_event
Return value	boolean result: true: successful. false: failed.

Example

```
printdir(1)
SOCK_WRITE_EVENT = 1
SOCK_READ_EVENT = 2
SOCK_CLOSE_EVENT = 4
SOCK_ACCEPT_EVENT = 8
socket.select(socket_fd, SOCK_CLOSE_EVENT);--care for close event
```

3.19.27socket.deselect

Description

The function is used to deselect the care socket event.

Prototype	boolean socket.deselect(int sockfd, int event_mask)
Parameters	int sockfd: the handle of the socket. int event_mask: 1: write_event 2: read_event 4: close_event 8: accept_event
Return value	boolean result: true: successful. false: failed.

Example

```
printdir(1)
SOCK_WRITE_EVENT = 1
SOCK_READ_EVENT = 2
SOCK_CLOSE_EVENT = 4
SOCK_ACCEPT_EVENT = 8
socket.deselect(socket_fd, SOCK_CLOSE_EVENT);--don't care for close event
```

3.20 MMS Library

3.20.1 mms.accquire

Description

The function is used to acquire MMS module for LUA.

Prototype	boolean mms.accquire()
Parameters	None
Return value	the result of the accquiring: true: successful false: failed

Example

```
rst = mms.accquire();
```

3.20.2 mms.release

Description

The function is used to release MMS module for LUA.

Prototype	void mms.release()
Parameters	None

Return value	None

Example

```
mms.release();
```

3.20.3 mms.set_mmsc

Description

The function is used to set MMSC.

Prototype	boolean mms.set_mmsc(string url)
Parameters	string url: the MMSC URL
Return value	the result of the setting: true: successful false: failed

Example

```
rst = mms.set_mmsc("http://my.mmsc.com/service");
```

3.20.4 mms.get_mmsc

Description

The function is used to set MMSC.

Prototype	string mms.get_mmsc()
Parameters	None
Return value	string: the MMSC URL

Example

```
mmsc_url = mms.get_mmsc();
```

3.20.5 mms.set_protocol

Description

The function is used to set MMS protocol and gateway address.

Prototype	boolean mms.set_protocol(int protocol, string proxy, int port)
Parameters	int protocol: the protocol used for MMS 0=WAP, 1=HTTP string proxy: the IP address of MMSC gateway int port: the port of MMSC gateway
Return value	the result of the setting: true: successful false: failed

Example

```
rst = mms.set_protocol(1, "10.0.0.172",    80);
```

3.20.6 mms.get_protocol

Description

The function is used to get MMS protocol and gateway address.

Prototype	int string int mms.get_protocol(int protocol, string proxy, int port)
Parameters	None
Return value	int: the protocol used for MMS 0=WAP, 1=HTTP string: the IP address of MMSC gateway int: the port of MMSC gateway

Example

```
protocol, proxy, port = mms.get_protocol();
```

3.20.7 mms.set_edit

Description

The function is used to set MMS edit state.

Prototype	boolean mms.set_edit(int edit_state)
Parameters	int edit_state: the state of edit 0=not editable, 1=editable
Return value	the result of the setting: true: successful false: failed

Example

```
rst = mms.set_edit(1);
```

3.20.8 mms.set_title

Description

The function is used to set MMS title.

Prototype	boolean mms.set_title(string title)
Parameters	string title: the title of the MMS
Return value	the result of the setting: true: successful false: failed

Example

```
rst = mms.set_title("test title");
```

3.20.9 mms.get_title

Description

The function is used to get MMS title.

Prototype	string mms.get_title(boolean convert_utf8_to_unicode)
Parameters	boolean convert_utf8_to_unicode: whether to convert the title to Unicode format if it is UTF-8 format true: convert false: not convert
Return value	string: the title of the MMS

Example

```
title = mms.get_title(false);
```

3.20.10mms.attach_file

Description

The function is used to add an attachment to the MMS from EFS.

Prototype	int mms.attach_file(string file_path)
Parameters	string file_path: the path of the file to attach
Return value	int: the result of the attaching: 0: successful other value: failed

Example

```
err_code= mms.attach_file("c:\\Picture\\1.jpg");
```

3.20.11mms.attach_from_memory

Description

The function is used to add an attachment to the MMS from memory.

Prototype	int mms.attach_from_memory (int source_type, string source_name, string content)
Parameters	int source_type: the type of the attachment 1=image, 2=text, 3=audio, 4=video string source_name: the attachment name string content: the attachment content
Return value	int: the result of the attaching: 0: successful other value: failed

Example

```
err_code= mms.attach_from_memory(2, "1.log", "this is the log content");
```

3.20.12mms.add_receipt

Description

The function is used to add a receipt/cc/bcc address to the MMS.

Prototype	int mms.add_receipt (string receipt, int receipt_type)
Parameters	string receipt: the receipt to add int receipt_type: the type of the receipt 0=receipt,1=cc,2=bcc
Return value	the result of the adding: true: successful false: failed

Example

rst= mms.add_receipt(“18602100071”, 0);

3.20.13mms.delete_receipt

Description

The function is used to delete a receipt/cc/bcc address to the MMS.

Prototype	int mms.delete_receipt (string receipt, int receipt_type)
Parameters	string receipt: the receipt to delete int receipt_type: the type of the receipt 0=receipt,1=cc,2=bcc
Return value	the result of the deleting: true: successful false: failed

Example

rst= mms.add_receipt(“18602100071”, 0);

3.20.14mms.get_receipts

Description

The function is used to get all the receipt/cc/bcc addresses from the MMS.

Prototype	table mms.get_receipt (int receipt_type)
Parameters	int receipt_type: the type of the receipt 0=receipt,1=cc,2=bcc
Return value	table: the string array of the addresses.

Example

receipts = mms.get_receipts(0);

3.20.15mms.save_attachment

Description

The function is used to save an attachment in the MMS to the EFS.

Prototype	int mms.save_attachment (int attach_no, string file_path)

Parameters	int attch_no: the attachment number string file_path: the path of the file
Return value	int: the result of saving 0: successful other value: failed

Example

err_code = mms.save_attachment(0, “c:\\test1.txt”);

3.20.16mms.save

Description

The function is used to save the MMS to the EFS.

Prototype	int mms.save (int box)
Parameters	int box: the box number(0 or 1)
Return value	int: the result of saving 0: successful other value: failed

Example

err_code = mms.save (0);

3.20.17mms.load

Description

The function is used to load the MMS from the EFS.

Prototype	int mms.load (int box)
Parameters	int box: the box number(0 or 1)
Return value	int: the result of loading 0: successful other value: failed

Example

err_code = mms.load (0);

3.20.18mms.get_attachment_count

Description

The function is used to get the count of the attachments in the MMS.

Prototype	int mms.get_attachment_count()
Parameters	None
Return value	int: the count of the attachments in the MMS

Example

count = mms.get_attachment_count ();

3.20.19mms.get_attachment_info

Description

The function is used to get the information of an attachment in the MMS.

Prototype	string int int mms.get_attachment_info(int attach_no)
Parameters	int attach_no: the attachment number
Return value	string: the attachment name int: the attachment type int: the size of the attachment inside the MMS

Example

```
name, type, size = mms.get_attachment_info (attach_no);
```

3.20.20mms.get_delivery_date

Description

The function is used to get the delivery date of the MMS.

Prototype	table mms.get_delivery_date()
Parameters	None
Return value	table: the delivery date in table format

Example

```
dt = mms.get_delivery_date();
print("delivery_date = ", dt.year, "-", dt.month, "-", dt.day, " ", dt.hour, ":", dt.min, ":", dt.sec, "\r\n");
```

3.20.21mms.read_attachment

Description

The function is used to read an attachment in the MMS.

Prototype	string int int mms.readt_attachment (int attach_no)
Parameters	int attach_no: the attachment number
Return value	string: the attachment content

Example

```
content = mms.read_attachment (attach_no);
```

3.21 SMTP Library

3.21.1 smtp.config

Description

The function is used to configure SMTP server and user password.

Prototype	boolean smtp.config(string server, int port, string

	username, string password)
Parameters	string server: the server IP or domain name int port: the port the server IP string username: the user name of the SMTP server string password: the password of the SMTP server
Return value	the result of the configuring: true: successful false: failed

Example

```
rst = smtp.config("smtp.163.com",25,"songjin_hello","123456");
```

3.21.2 smtp.set_from

Description

The function is used to configure sender address and name information of the email.

Prototype	boolean smtp.set_from(string sender_address, string sender_name)
Parameters	string sender_address: the address of the sender string sender_name: the name of the sender
Return value	the result of the setting: true: successful false: failed

Example

```
rst = smtp.set_from("<songjin_hello@163.com>", "");
```

3.21.3 smtp.set_rcpt

Description

The function is used to configure recipient address and name information of the email.

Prototype	boolean smtp.set_rcpt(int kind, int index, string rcpt_address, string rcpt_name)
Parameters	int kind: the type of the address 0: To normal recipient 1: CC, Carbon Copy recipient 2: BCC Bind Carbon Copy recipient int index: the index of the recipient string rcpt_address: the address of the recipient string rcpt_name: the name of the recipient
Return value	the result of the setting: true: successful false: failed

Example

```
rst = smtp.set_rcpt(0, 0, "<songjin_hello@163.com>", "");
```

3.21.4 smtp.set_subject

Description

The function is used to configure the subject information of the email.

Prototype	boolean smtp.set_subject(string subject)
Parameters	string subject: the subject of the email
Return value	the result of the setting: true: successful false: failed

Example

```
rst = smtp.set_subject("test subject");
```

3.21.5 smtp.set_body

Description

The function is used to configure the subject information of the email.

Prototype	boolean smtp.set_body(string body)
Parameters	string body: the body of the email
Return value	the result of the setting: true: successful false: failed

Example

```
rst = smtp.set_body("test body");
```

3.21.6 smtp.set_file

Description

The function is used to configure the attachment information of the email.

Prototype	boolean smtp.set_file(int attach_no, string filepath)
Parameters	int attach_no: the index of the attachment. string filepath: the full path of the file
Return value	the result of the setting: true: successful false: failed

Example

```
rst = smtp.set_file(1, "C:\\Picture\\test.jpg");
```

3.21.7 smtp.send

Description

The function is used to send the email.

Prototype	int smtp.send()
Parameters	None
Return value	the result of the sending: 1: successful Other: failed

Example

```
rst = smtp.send();
```

3.22 FTP Library

3.22.1 ftp.simpput

Description

The function is used to put a file from local EFS to the remote FTP server. For the corresponding AT command, please refer to AT+CFTPPUTFILE.

Prototype	int ftp.simpput (string address, int port, string name, string password, string remote_filepath, string local_filepath, int passive, int rest_size)
Parameters	address: FTP server address port: FTP server port name: FTP user name password: FTP user password remote_filepath: the path of the remote file . local_filepath: the path of the local EFS file passive: passive mode (1=passive mode, 0=not passive mode) rest_size : The value for FTP “REST” command which is used for broken transfer when transferring failed last time. It’s range is 0 to 2147483647.
Return value	the result of the FTP putting: 0: successful other: failed

Example

```
server = "ftp.mysite.com";
port = 21;
name = "myaccount";
pass = "password";
remote_file = "/up_normal.jpg";
uplocal_file = "c:\\Picture\\normal.jpg";
passive = 1;
rst = ftp.simpput(server, port, name, pass, remote_file, uplocal_file, passive);
print(rst,"\r\n");
```

3.22.2 ftp.simpgut

Description

The function is used to get a file from the remote FTP server to local EFS. For the corresponding AT command, please refer to AT+CFTPGETFILE.

Prototype	int ftp.simpget (string address, int port, string name, string password, string remote_filepath, string local_filepath, int passive, int rest_size)
Parameters	address: FTP server address port: FTP server port name: FTP user name password: FTP user password remote_filepath: the path of the remote file. local_filepath: the path of the local EFS file passive: passive mode (1=passive mode, 0=not passive mode) rest_size : The value for FTP “REST” command which is used for broken transfer when transferring failed last time. It’s range is 0 to 2147483647.
Return value	the result of the FTP getting: 0: successful other: failed

Example

```
server = " ftp.mysite.com ";
port = 21;
name = "myaccount";
pass = "password";
remote_file = "/up_normal.jpg";
downlocal_file = "c:\\Video\\down_normal.jpg";
passive = 1;
rst = ftp.simpget(server, port, name, pass, remote_file, downlocal_file, passive);
print(rst,"\r\n");
```

3.22.3 ftp.simplist

Description

The function is used to get the file and directory list from the remote FTP server to local EFS. For the corresponding AT command, please refer to AT+CFTPLIST.

Prototype	Int string ftp.simplist (string address, int port, string name, string password, string remote_path, int passive)
Parameters	address: FTP server address port: FTP server port

	name: FTP user name password: FTP user password remote_path: the path of the remote file. passive: passive mode (1=passive mode, 0=not passive mode)
Return value	the result of the FTP listing: int result: 0: successful other: failed string list_data: the data of the list command returns.

Example

```
server = " ftp.mysite.com ";
port = 21;
name = "myaccount";
pass = "password";
remote_path = "/MyDir/";
passive = 1;
rst, list_data= ftp.simplist(server, port, name, pass, remote_path, passive);
print(rst,"\r\n");
```

3.23 SSLMGR Library

This library is used to set SSL certificate, key and CA files used by SSL application. Currently only .der or .pem (without password protection) format certificates/key are supported.

3.23.1 sslmgr.setca

Description

The function is used to set CA file name used by SSL application.

Prototype	boolean sslmgr.setca(int chain_index, string filename)
Parameters	int chain_index: The index of CA in the chain. It's range is from 0 to 3. string filename: The name of the CA file.
Return value	boolean result: true: successful. false: failed.

Example

```
printdir(1)
local result1 = sslmgr.setca(0, "ca_cert.der");
```

3.23.2 sslmgr.setcert

Description

The function is used to set certificate file name used by SSL application.

Prototype	boolean sslmgr.setca(string filename, int sign_ca_chain_index)
Parameters	string filename: The name of the certificate file. int sign_ca_chain_index: The index of CA that is used to sign this certificate. It's range is from 0 to 3.
Return value	boolean result: true: successful. false: failed.

Example

```
printdir(1)
local result2 = sslmgr.setcert("client_cert.der", 0);
```

3.23.3 sslmgr.setkey

Description

The function is used to set key file name used by SSL application.

Prototype	boolean sslmgr.setkey(string filename[, int key_type])
Parameters	string filename: The name of the key file. int key_type: The type of the key file. 0: RSA format(default) 1: DSA format.
Return value	boolean result: true: successful. false: failed.

Example

```
printdir(1)
local result3 = sslmgr.setkey("client_key.der");
```

3.23.4 sslmgr.loadck

Description

The function is used to load the certificate/key/CA set before used by SSL application.

Prototype	boolean sslmgr.loadck(void)
Parameters	None
Return value	boolean result: true: successful. false: failed.

Example

```
printdir(1)
local result1 = sslmgr.setca(0, "ca_cert.der");
local result2 = sslmgr.setcert("client_cert.der", 0);
local result3 = sslmgr.setkey("client_key.der");
local result4 = sslmgr.loadck();
print("result=[", result1, ", ", result2, ", ", result3, ", ", result4, "]\r\n");
```

3.24 COMMON CHANNEL Library

This library is majorly used to do SSL operation.

3.24.1 common_ch.start

Description

The function is used to start common channel service, when it returns true, the PS network is activated(PDP) .

Prototype	boolean common_ch.start(void)
Parameters	None
Return value	boolean result: true: successful. false: failed.

Example

```
printdir(1)
print("call common_ch.start()...\r\n");
result = common_ch.start();
print("common_ch.start(), result=", result, "\r\n");
if (not result) then
    print("failed to call common_ch.start()\r\n");
    return;
end;
```

3.24.2 common_ch.stop

Description

The function is used to stop common channel service, when it returns true, the PS network is deactivated(PDP) .

Prototype	boolean common_ch.stop(void)
Parameters	None
Return value	boolean result: true: successful. false: failed.

Example

```
printdir(1)
print("call common_ch.stop()...\r\n");
```

```
result = common_ch.stop();
print("common_ch.stop(), result=", result, "\r\n");
```

3.24.3 common_ch.open

Description

The function is used to open a session.

Prototype	int common_ch.open(int session_id, string peer_address, int peer_port, int channel_type, int bind_port)
Parameters	int session_id: the ID of the session(0 or 1) string peer_address: the destination IPv4 address or domain name. int peer_port: the destination port. int channel_type: 0: UDP 1: TCP client 2: SSL v3.0 or TLS 1.0 client int bind_port: the local bind port for socket.
Return value	int error_code: CCH_RST_OK = 0; CCH_RST_ALERTING = 1; CCH_RST_UNKNOWN_ERROR = 2; CCH_RST_BUSY = 3; CCH_RST_PEER_CLOSED = 4; CCH_RST_TIMEOUT = 5; CCH_RST_TRANSFER_FAILED = 6; CCH_RST_MEMORY_ERROR = 7; CCH_RST_INVALID_PARAMETER = 8; CCH_RST_NETWORK_ERROR = 9;

Example

```
printdir(1)
local error_code;
local session_id = 0; --0 or 1
local peer_address = "180.166.164.118";--this can be domain name or IPv4 address
local peer_port = 990;
local channel_type = 2;--SSL
local bind_port = -1;-- -1 means no bind port

print("call common_ch.open()...\r\n");
error_code = common_ch.open(session_id, peer_address, peer_port, channel_type, bind_port);
print("common_ch.open(), error_code=", error_code, "\r\n");
```

3.24.4 common_ch.close

Description

The function is used to close a session.

Prototype	int common_ch.close(int session_id)
Parameters	int session_id: the ID of the session(0 or 1)
Return value	int error_code: CCH_RST_OK = 0; CCH_RST_ALERTING = 1; CCH_RST_UNKNOWN_ERROR = 2; CCH_RST_BUSY = 3; CCH_RST_PEER_CLOSED = 4; CCH_RST_TIMEOUT = 5; CCH_RST_TRANSFER_FAILED = 6; CCH_RST_MEMORY_ERROR = 7; CCH_RST_INVALID_PARAMETER = 8; CCH_RST_NETWORK_ERROR = 9;

Example

```
printdir(1)
print("call common_ch.close()...\r\n");
error_code = common_ch.close(session_id);
print("common_ch.close(), error_code=", error_code, "\r\n");
```

3.24.5 common_ch.send

Description

The function is used to send data using a session.

Prototype	int common_ch.close(int session_id, string data)
Parameters	int session_id: the ID of the session(0 or 1) string data: the data to send.
Return value	int error_code: CCH_RST_OK = 0; CCH_RST_ALERTING = 1; CCH_RST_UNKNOWN_ERROR = 2; CCH_RST_BUSY = 3; CCH_RST_PEER_CLOSED = 4; CCH_RST_TIMEOUT = 5; CCH_RST_TRANSFER_FAILED = 6; CCH_RST_MEMORY_ERROR = 7; CCH_RST_INVALID_PARAMETER = 8; CCH_RST_NETWORK_ERROR = 9;

Example

```
printdir(1)
```

```
print("call common_ch.send()...\r\n");
local data = "Hello, this is sent from LUA common channel";
error_code = common_ch.send(session_id, data);
print("common_ch.send(), error_code=", error_code, "\r\n");
```

3.24.6 common_ch.recv

Description

The function is used to receive data using a session.

Prototype	int common_ch.receive(int session_id, int max_len, int timeout)
Parameters	int session_id: the ID of the session(0 or 1) int max_len: the maximum bytes of the data to receive. It must be larger than 0. int timeout: the milliseconds to wait if no data in cache.
Return value	int error_code: CCH_RST_OK = 0; CCH_RST_ALERTING = 1; CCH_RST_UNKNOWN_ERROR = 2; CCH_RST_BUSY = 3; CCH_RST_PEER_CLOSED = 4; CCH_RST_TIMEOUT = 5; CCH_RST_TRANSFER_FAILED = 6; CCH_RST_MEMORY_ERROR = 7; CCH_RST_INVALID_PARAMETER = 8; CCH_RST_NETWORK_ERROR = 9;

Example

```
printdir(1)
print("call common_ch.recv()...\r\n");
local max_recv_len = 2048;
error_code, rcvd_data = common_ch.recv(session_id, max_recv_len);
print("common_ch.recv(), error_code=", error_code, "\r\n");
```

3.24.7 common_ch.get_rx_cache_len

Description

The function is used to get the length of the received data cached in the module.

Prototype	int common_ch.get_rx_cache_len(int session_id)
Parameters	int session_id: the ID of the session(0 or 1)
Return value	int len: the length of the data.

Example

```
printdir(1)
```

local len = common_ch.get_rx_cache_len(session_id);

3.24.8 common_ch.state

Description

The function is used to get the state of the common channel.

Prototype	int int int common_ch.state(void)
Parameters	None
Return value	int state: the state of the common stack. 0: NULL 1: acquired stack 2: opening network 3: closing network 4: opened network int session0_state: the state of session 0. 0: NULL 1: created session 2: closing session 3: opening session 4: opened session nt session1_state: the state of session 1. 0: NULL 1: created session 2: closing session 3: opening session 4: opened session

Example

```
printdir(1)
state, session_0_state, session_1_state = common_ch.state();
```

3.25 SMTPS Library

3.25.1 smtps.set_server

Description

The function is used to configure SMTPS server address.

Prototype	int smtps.set_server(string server, int port, int server_type)
Parameters	string server: the server IP or domain name int port: the port the server IP int server_type: 1: SMTP server 2: SMTPS server

Return value	int error_code: SMTPS_RST_OK = 0; SMTPS_RST_BUSY = 1; SMTPS_RST_OVERSIZE = 2; SMTPS_RST_DUPLICATE_FILE = 3; SMTPS_RST_TIMEOUT = 4; SMTPS_RST_TRANSFER_FAILED = 5; SMTPS_RST_MEM_ERROR = 6; SMTPS_RST_INVALID_PARAM = 7; SMTPS_RST_NETWORK_ERROR = 8; SMTPS_RST_EFS_ERROR = 9; SMTPS_RST_SERVER_ERROR = 10; SMTPS_RST_AUTH_FAIL = 11; SMTPS_RST_USER_CANCEL = 12; SMTPS_RST_UNKNOWN_ERROR = 255;

Example

```
rst = smtp.set_serv("smtp.163.com",25,1);
```

3.25.2 smtps.set_auth

Description

The function is used to configure SMTP user password.

Prototype	boolean smtps.set_auth(string username, string password)
Parameters	string username: the user name of the SMTP server string password: the password of the SMTP server
Return value	int error_code: SMTPS_RST_OK = 0; SMTPS_RST_BUSY = 1; SMTPS_RST_OVERSIZE = 2; SMTPS_RST_DUPLICATE_FILE = 3; SMTPS_RST_TIMEOUT = 4; SMTPS_RST_TRANSFER_FAILED = 5; SMTPS_RST_MEM_ERROR = 6; SMTPS_RST_INVALID_PARAM = 7; SMTPS_RST_NETWORK_ERROR = 8; SMTPS_RST_EFS_ERROR = 9; SMTPS_RST_SERVER_ERROR = 10; SMTPS_RST_AUTH_FAIL = 11; SMTPS_RST_USER_CANCEL = 12; SMTPS_RST_UNKNOWN_ERROR = 255;

Example

```
rst = smtps.set_auth("songjin_hello","123456");
```

3.25.3 smtps.set_from

Description

The function is used to configure sender address and name information of the email.

Prototype	int smtps.set_from(string sender_address, string sender_name)
Parameters	string sender_address: the address of the sender string sender_name: the name of the sender
Return value	int error_code: SMTPS_RST_OK = 0; SMTPS_RST_BUSY = 1; SMTPS_RST_OVERSIZE = 2; SMTPS_RST_DUPLICATE_FILE = 3; SMTPS_RST_TIMEOUT = 4; SMTPS_RST_TRANSFER_FAILED = 5; SMTPS_RST_MEM_ERROR = 6; SMTPS_RST_INVALID_PARAM = 7; SMTPS_RST_NETWORK_ERROR = 8; SMTPS_RST_EFS_ERROR = 9; SMTPS_RST_SERVER_ERROR = 10; SMTPS_RST_AUTH_FAIL = 11; SMTPS_RST_USER_CANCEL = 12; SMTPS_RST_UNKNOWN_ERROR = 255;

Example

```
rst = smtps.set_from("<songjin_hello@163.com>");
```

3.25.4 smtps.set_rcpt

Description

The function is used to configure recipient address and name information of the email.

Prototype	int smtps.set_rcpt(int kind, int index, string rcpt_address, string rcpt_name)
Parameters	int kind: the type of the address 0: To normal recipient 1: CC, Carbon Copy recipient 2: BCC Bind Carbon Copy recipient int index: the index of the recipient string rcpt_address: the address of the recipient string rcpt_name: the name of the recipient
Return value	int error_code: SMTPS_RST_OK = 0; SMTPS_RST_BUSY = 1; SMTPS_RST_OVERSIZE = 2;

	SMTPS_RST_DUPLICATE_FILE = 3; SMTPS_RST_TIMEOUT = 4; SMTPS_RST_TRANSFER_FAILED = 5; SMTPS_RST_MEM_ERROR = 6; SMTPS_RST_INVALID_PARAM = 7; SMTPS_RST_NETWORK_ERROR = 8; SMTPS_RST_EFS_ERROR = 9; SMTPS_RST_SERVER_ERROR = 10; SMTPS_RST_AUTH_FAIL = 11; SMTPS_RST_USER_CANCEL = 12; SMTPS_RST_UNKNOWN_ERROR = 255;

Example

```
rst = smtps.set_rcpt(0, 0, "<songjin_hello@163.com>");
```

3.25.5 smtps.set_subject

Description

The function is used to configure the subject information of the email.

Prototype	boolean smtps.set_subject(string subject, string charset)
Parameters	string subject: the subject of the email string charset: the character set of the subject. Default is “utf-8”
Return value	int error_code: SMTPS_RST_OK = 0; SMTPS_RST_BUSY = 1; SMTPS_RST_OVERSIZE = 2; SMTPS_RST_DUPLICATE_FILE = 3; SMTPS_RST_TIMEOUT = 4; SMTPS_RST_TRANSFER_FAILED = 5; SMTPS_RST_MEM_ERROR = 6; SMTPS_RST_INVALID_PARAM = 7; SMTPS_RST_NETWORK_ERROR = 8; SMTPS_RST_EFS_ERROR = 9; SMTPS_RST_SERVER_ERROR = 10; SMTPS_RST_AUTH_FAIL = 11; SMTPS_RST_USER_CANCEL = 12; SMTPS_RST_UNKNOWN_ERROR = 255;

Example

```
rst = smtps.set_subject("test subject");
```

3.25.6 smtps.set_bch

Description

The function is used to set the body character set of the email.

Prototype	int smtps.set_bch(string bch)
Parameters	string bch: the body character set, default is utf-8
Return value	int error_code: SMTPS_RST_OK = 0; SMTPS_RST_BUSY = 1; SMTPS_RST_OVERSIZE = 2; SMTPS_RST_DUPLICATE_FILE = 3; SMTPS_RST_TIMEOUT = 4; SMTPS_RST_TRANSFER_FAILED = 5; SMTPS_RST_MEM_ERROR = 6; SMTPS_RST_INVALID_PARAM = 7; SMTPS_RST_NETWORK_ERROR = 8; SMTPS_RST_EFS_ERROR = 9; SMTPS_RST_SERVER_ERROR = 10; SMTPS_RST_AUTH_FAIL = 11; SMTPS_RST_USER_CANCEL = 12; SMTPS_RST_UNKNOWN_ERROR = 255;

Example

```
rst = smtps.set_bch("gb2312");
```

3.25.7 smtps.set_body

Description

The function is used to configure the subject information of the email.

Prototype	int smtps.set_body(string body)
Parameters	string body: the body of the email
Return value	int error_code: SMTPS_RST_OK = 0; SMTPS_RST_BUSY = 1; SMTPS_RST_OVERSIZE = 2; SMTPS_RST_DUPLICATE_FILE = 3; SMTPS_RST_TIMEOUT = 4; SMTPS_RST_TRANSFER_FAILED = 5; SMTPS_RST_MEM_ERROR = 6; SMTPS_RST_INVALID_PARAM = 7; SMTPS_RST_NETWORK_ERROR = 8; SMTPS_RST_EFS_ERROR = 9; SMTPS_RST_SERVER_ERROR = 10;

	SMTPS_RST_AUTH_FAIL = 11; SMTPS_RST_USER_CANCEL = 12; SMTPS_RST_UNKNOWN_ERROR = 255;

Example

```
rst = smtps.set_body("test body");
```

3.25.8 smtps.set_file

Description

The function is used to configure the attachment information of the email.

Prototype	boolean smtps.set_file(int attach_no, string filepath)
Parameters	int attach_no: the index of the attachment. string filepath: the full path of the file
Return value	int error_code: SMTPS_RST_OK = 0; SMTPS_RST_BUSY = 1; SMTPS_RST_OVERSIZE = 2; SMTPS_RST_DUPLICATE_FILE = 3; SMTPS_RST_TIMEOUT = 4; SMTPS_RST_TRANSFER_FAILED = 5; SMTPS_RST_MEM_ERROR = 6; SMTPS_RST_INVALID_PARAM = 7; SMTPS_RST_NETWORK_ERROR = 8; SMTPS_RST_EFS_ERROR = 9; SMTPS_RST_SERVER_ERROR = 10; SMTPS_RST_AUTH_FAIL = 11; SMTPS_RST_USER_CANCEL = 12; SMTPS_RST_UNKNOWN_ERROR = 255;

Example

```
rst = smtps.set_file(1, "C:\\Picture\\test.jpg");
```

3.25.9 smtps.send

Description

The function is used to send the email.

Prototype	int smtps.send()
Parameters	None
Return value	int error_code: SMTPS_RST_OK = 0; SMTPS_RST_BUSY = 1; SMTPS_RST_OVERSIZE = 2; SMTPS_RST_DUPLICATE_FILE = 3;

	SMTPS_RST_TIMEOUT = 4; SMTPS_RST_TRANSFER_FAILED = 5; SMTPS_RST_MEM_ERROR = 6; SMTPS_RST_INVALID_PARAM = 7; SMTPS_RST_NETWORK_ERROR = 8; SMTPS_RST_EFS_ERROR = 9; SMTPS_RST_SERVER_ERROR = 10; SMTPS_RST_AUTH_FAIL = 11; SMTPS_RST_USER_CANCEL = 12; SMTPS_RST_UNKNOWN_ERROR = 255;

Example

```
rst = smtps.send();
```

3.25.10smtps.stop

Description

The function is used to stop sending the email.

Prototype	int smtps.stop()
Parameters	None
Return value	int error_code: SMTPS_RST_OK = 0; SMTPS_RST_BUSY = 1; SMTPS_RST_OVERSIZE = 2; SMTPS_RST_DUPLICATE_FILE = 3; SMTPS_RST_TIMEOUT = 4; SMTPS_RST_TRANSFER_FAILED = 5; SMTPS_RST_MEM_ERROR = 6; SMTPS_RST_INVALID_PARAM = 7; SMTPS_RST_NETWORK_ERROR = 8; SMTPS_RST_EFS_ERROR = 9; SMTPS_RST_SERVER_ERROR = 10; SMTPS_RST_AUTH_FAIL = 11; SMTPS_RST_USER_CANCEL = 12; SMTPS_RST_UNKNOWN_ERROR = 255;

Example

```
rst = smtps.stop();
```

3.26 FTPS Library

This library is majorly used to do FTP/FTPS operation.

3.26.1 ftps.start

Description

The function is used to start ftps service, when it returns true, the PS network is activated(PDP) .

Prototype	boolean ftps.start(void)
Parameters	None
Return value	boolean result: true: successful. false: failed.

Example

```
printdir(1)
print("call ftps.start()...\r\n");
result = ftps.start();
print("ftps.start() = ", result, "\r\n");
```

3.26.2 ftps.stop

Description

The function is used to stop ftps service, when it returns true, the PS network is deactivated(PDP) .

Prototype	boolean ftps.stop(void)
Parameters	None
Return value	boolean result: true: successful. false: failed.

Example

```
printdir(1)
print("call ftps.stop()...\r\n");
result = ftps.stop();
print("ftps.stop(), result=", result, "\r\n");
```

3.26.3 ftps.login

Description

The function is used to login FTPS/FTP server .

Prototype	int ftps.login(int server_type, string server_address, int port, string name, string password)
Parameters	int server_type: 0=FTP, 1=Explicit FTPS(Auth SSL), 2=Explicit FTPS(Auth TLS), 3=Implicit SSL/TLS string server_address: the address of the FTP/FTPS server int port: the server listening port

	string name: the name to login string password: the password used to login
Return value	int error_code: FTPS_RST_OK = 0; FTPS_RST_SSL_ALERT = 1; FTPS_RST_UNKNOWN_ERROR = 2; FTPS_RST_BUSY = 3; FTPS_RST_SERVER_CLOSED = 4; FTPS_RST_TIMEOUT = 5; FTPS_RST_TRANSFER_FAILED = 6; FTPS_RST_MEMORY_ERROR = 7; FTPS_RST_INVALID_PARAM = 8; FTPS_RST_REJ_BY_SERVER = 9; FTPS_RST_NETWORK_ERROR = 10;

Example

```
printdir(1)
local server = "180.166.164.118";--this can be domain name or IP address
local port = 990;
local name = "jin.song";
local pass = "123456";
local remote_path = "/";
local server_type = 2;
-- 0=FTP, 1=Explicit FTPS(Auth SSL), 2=Explicit FTPS(Auth TLS), 3=Implicit SSL/TLS
print("call ftps.login()...\r\n");
error_code = ftps.login(server_type, server, port, name, pass);
print("ftps.login() = ", error_code, "\r\n");
```

3.26.4 ftps.logout

Description

The function is used to logout the FTPS/FTP server .

Prototype	int ftps.logout(void)
Parameters	None
Return value	int error_code: FTPS_RST_OK = 0; FTPS_RST_SSL_ALERT = 1; FTPS_RST_UNKNOWN_ERROR = 2; FTPS_RST_BUSY = 3; FTPS_RST_SERVER_CLOSED = 4; FTPS_RST_TIMEOUT = 5; FTPS_RST_TRANSFER_FAILED = 6; FTPS_RST_MEMORY_ERROR = 7; FTPS_RST_INVALID_PARAM = 8; FTPS_RST_REJ_BY_SERVER = 9;

	FTPS_RST_NETWORK_ERROR = 10;

Example

```
printdir(1)
print("call ftps.logout()...\r\n");
error_code = ftps.logout();
print("ftps.logout() = ", error_code, "\r\n");
```

3.26.5 ftps.putfile

Description

The function is used to put a file to the FTPS/FTP server.

Prototype	int ftps.putfile(string remote_file_path, string local_file_path, int rest_size)
Parameters	string remote_file_path: the remote file path string local_file_path: the local file path int rest_size: the rest size, default 0
Return value	int error_code: FTPS_RST_OK = 0; FTPS_RST_SSL_ALERT = 1; FTPS_RST_UNKNOWN_ERROR = 2; FTPS_RST_BUSY = 3; FTPS_RST_SERVER_CLOSED = 4; FTPS_RST_TIMEOUT = 5; FTPS_RST_TRANSFER_FAILED = 6; FTPS_RST_MEMORY_ERROR = 7; FTPS_RST_INVALID_PARAM = 8; FTPS_RST_REJ_BY_SERVER = 9; FTPS_RST_NETWORK_ERROR = 10;

Example

```
printdir(1)
local upload_remote_file = "/upload_5120.txt";
local upload_local_file = "C:\\my_downloaded_5120.txt";
print("call ftps.putfile(\""..upload_remote_file.."\")...\r\n");
local rest_size = 0;
error_code = ftps.putfile(upload_remote_file, upload_local_file, rest_size);
print("ftps.putfile() = ", error_code, "\r\n");
```

3.26.6 ftps.getfile

Description

The function is used to get a file from the FTPS/FTP server .

Prototype	int ftps.getfile(string remote_file_path, string local_file_path, int rest_size)

Parameters	string remote_file_path: the remote file path string local_file_path: the local file path int rest_size: the rest size, default 0
Return value	int error_code: FTPS_RST_OK = 0; FTPS_RST_SSL_ALERT = 1; FTPS_RST_UNKNOWN_ERROR = 2; FTPS_RST_BUSY = 3; FTPS_RST_SERVER_CLOSED = 4; FTPS_RST_TIMEOUT = 5; FTPS_RST_TRANSFER_FAILED = 6; FTPS_RST_MEMORY_ERROR = 7; FTPS_RST_INVALID_PARAM = 8; FTPS_RST_REJ_BY_SERVER = 9; FTPS_RST_NETWORK_ERROR = 10;

Example

```
printdir(1)
local download_remote_file = "/5120.txt";
local download_local_file = "C:\\my_downloaded_5120.txt";
print("call ftps.getfile(\""..download_remote_file.."\")...\r\n");
local rest_size = 0;
error_code = ftps.getfile(download_remote_file, download_local_file, rest_size);
print("ftps.getfile() = ", error_code, "\r\n");
```

3.26.7 ftps.size

Description

The function is used to get a file size on the FTPS/FTP server .

Prototype	int int ftps.size(string remote_file_path)
Parameters	string remote_file_path: the remote file path string local_file_path: the local file path int rest_size: the rest size, default 0
Return value	int error_code: FTPS_RST_OK = 0; FTPS_RST_SSL_ALERT = 1; FTPS_RST_UNKNOWN_ERROR = 2; FTPS_RST_BUSY = 3; FTPS_RST_SERVER_CLOSED = 4; FTPS_RST_TIMEOUT = 5; FTPS_RST_TRANSFER_FAILED = 6; FTPS_RST_MEMORY_ERROR = 7; FTPS_RST_INVALID_PARAM = 8; FTPS_RST_REJ_BY_SERVER = 9; FTPS_RST_NETWORK_ERROR = 10;

	int file_size; the size of file in byte.

Example

```
printdir(1)
print("call ftps.size()\r\n");
error_code, f_bytes = ftps.size(download_remote_file);
print("ftps.size() = ", error_code, ",", f_bytes, "\r\n");
```

3.26.8 ftps.mkdir

Description

The function is used to create a directory on the FTPS/FTP server .

Prototype	int ftps.mkdir(string remote_dir_path)
Parameters	string remote_dir_path: the remote directory path
Return value	int error_code: FTPS_RST_OK = 0; FTPS_RST_SSL_ALERT = 1; FTPS_RST_UNKNOWN_ERROR = 2; FTPS_RST_BUSY = 3; FTPS_RST_SERVER_CLOSED = 4; FTPS_RST_TIMEOUT = 5; FTPS_RST_TRANSFER_FAILED = 6; FTPS_RST_MEMORY_ERROR = 7; FTPS_RST_INVALID_PARAM = 8; FTPS_RST_REJ_BY_SERVER = 9; FTPS_RST_NETWORK_ERROR = 10;

Example

```
printdir(1)
local test_mk_rm_dir = "/mytestdir";
print("call ftps.mkdir()\r\n");
error_code = ftps.mkdir(test_mk_rm_dir);
print("ftps.mkdir() = ", error_code, "\r\n");
```

3.26.9 ftps.rmdir

Description

The function is used to remove a directory on the FTPS/FTP server .

Prototype	int ftps.rmdir(string remote_dir_path)
Parameters	string remote_dir_path: the remote directory path
Return value	int error_code: FTPS_RST_OK = 0; FTPS_RST_SSL_ALERT = 1; FTPS_RST_UNKNOWN_ERROR = 2; FTPS_RST_BUSY = 3;

	FTPS_RST_SERVER_CLOSED = 4; FTPS_RST_TIMEOUT = 5; FTPS_RST_TRANSFER_FAILED = 6; FTPS_RST_MEMORY_ERROR = 7; FTPS_RST_INVALID_PARAM = 8; FTPS_RST_REJ_BY_SERVER = 9; FTPS_RST_NETWORK_ERROR = 10;

Example

```
printdir(1)
local test_mk_rm_dir = "/mytestdir";
print("call ftps.rmdir()\r\n");
error_code = ftps.rmdir(test_mk_rm_dir);
print("ftps.rmdir() = ", error_code, "\r\n");
```

3.26.10ftps.dele

Description

The function is used to delete a file on the FTPS/FTP server .

Prototype	int ftps.dele(string remote_file_path)
Parameters	string remote_file_path: the remote file path
Return value	int error_code: FTPS_RST_OK = 0; FTPS_RST_SSL_ALERT = 1; FTPS_RST_UNKNOWN_ERROR = 2; FTPS_RST_BUSY = 3; FTPS_RST_SERVER_CLOSED = 4; FTPS_RST_TIMEOUT = 5; FTPS_RST_TRANSFER_FAILED = 6; FTPS_RST_MEMORY_ERROR = 7; FTPS_RST_INVALID_PARAM = 8; FTPS_RST_REJ_BY_SERVER = 9; FTPS_RST_NETWORK_ERROR = 10;

Example

```
printdir(1)
local upload_remote_file = "/upload_5120.txt";
print("call ftps.dele()\r\n");
error_code = ftps.dele(upload_remote_file);
print("ftps.dele() = ", error_code, "\r\n");
```

3.26.11ftps.cwd

Description

The function is used to change the current directory on the FTPS/FTP server .

Prototype	int ftps.cwd(string remote_dir_path)
Parameters	string remote_dir_path: the remote directory path
Return value	int error_code: FTPS_RST_OK = 0; FTPS_RST_SSL_ALERT = 1; FTPS_RST_UNKNOWN_ERROR = 2; FTPS_RST_BUSY = 3; FTPS_RST_SERVER_CLOSED = 4; FTPS_RST_TIMEOUT = 5; FTPS_RST_TRANSFER_FAILED = 6; FTPS_RST_MEMORY_ERROR = 7; FTPS_RST_INVALID_PARAM = 8; FTPS_RST_REJ_BY_SERVER = 9; FTPS_RST_NETWORK_ERROR = 10;

Example

```
printdir(1)
local test_cwd_dir = "/testmy";
print("call ftps.cwd()\r\n");
error_code = ftps.cwd(test_cwd_dir);
print("ftps.cwd() = ", error_code, "\r\n");
```

3.26.12ftps.pwd

Description

The function is used to get the current directory on the FTPS/FTP server .

Prototype	int string ftps.pwd(void)
Parameters	None
Return value	int error_code: FTPS_RST_OK = 0; FTPS_RST_SSL_ALERT = 1; FTPS_RST_UNKNOWN_ERROR = 2; FTPS_RST_BUSY = 3; FTPS_RST_SERVER_CLOSED = 4; FTPS_RST_TIMEOUT = 5; FTPS_RST_TRANSFER_FAILED = 6; FTPS_RST_MEMORY_ERROR = 7; FTPS_RST_INVALID_PARAM = 8; FTPS_RST_REJ_BY_SERVER = 9; FTPS_RST_NETWORK_ERROR = 10; string pwd: the current directory

Example

```
printdir(1)
print("call ftps.pwd()\r\n");
error_code, pwd = ftps.pwd();
```

```
print("ftps.pwd() = ", error_code, ",", pwd, "\r\n");
```

3.26.13ftps.list

Description

The function is used to list a directory on the FTPS/FTP server .

Prototype	int string ftps.list(string remote_dir_path)
Parameters	string remote_dir_path: the remote directory path
Return value	int error_code: FTPS_RST_OK = 0; FTPS_RST_SSL_ALERT = 1; FTPS_RST_UNKNOWN_ERROR = 2; FTPS_RST_BUSY = 3; FTPS_RST_SERVER_CLOSED = 4; FTPS_RST_TIMEOUT = 5; FTPS_RST_TRANSFER_FAILED = 6; FTPS_RST_MEMORY_ERROR = 7; FTPS_RST_INVALID_PARAM = 8; FTPS_RST_REJ_BY_SERVER = 9; FTPS_RST_NETWORK_ERROR = 10; string list_data the list data

Example

```
printdir(1)
print("call ftps.list()...\r\n");
error_code, list_data = ftps.list("/");
print("ftps.list() = ", error_code, "\r\n");
if (error_code and (error_code == FTPS_RST_OK)) then
   if (list_data and printdir()) then
  print("LIST DATA:\r\n");
  os.printstr(list_data);
    end;
end;
```

3.27 EBDAT Library

This library relies on the module firmware supports LUA and EBDAT at the same time.

3.27.1 ebdat.ready

Description

The function is used to check whether the EBDAT for LUA extended API module is ready.

Prototype	boolean ebdat.ready()
Parameters	None
Return value	the result of setting:

	true: ready false: not ready

Example

```
rst = ebdat.ready();
```

3.27.2 ebdat.signal_notify

Description

The function is used to send signal to EBDAT task.

Prototype	void ebdat.signal_notify(int thread_id, int sig_mask)
Parameters	thread_id: the id of the EBDAT thread. int sig_mask: the mask of the signals (1, 2, 4, 8, 16, 32, 64, 128 with bor operation).
Return value	None

Example

```
ebdat.signal_notify(0,0x02);
```

3.27.3 ebdat.setevt

Description

The function is used to set a event to an EBDAT thread.

Prototype	void ebdat.setevt(int thread_id, int evt, int evt_p1, int evt_p2, int evt_p3)
Parameters	int thread_id: the index of thread to be set the event. int evt: the identity of the event int evt_p1: the first parameter of the event int evt_p2: the second parameter of the event int evt_p3: the third parameter of the event
Return value	None

Example

```
thread.setevt(0, 3, 1,2,3);
```

3.27.4 ebdat.callapi

Description

The function is used to call API extended using EBDAT module.

Prototype	int int int int int int int string string string ebdat.callapi(int p1, int p2, int p3, int p4, int p5, int p6, int s1, int s2, int s3)
Parameters	int p1-p6: parameter of integer type string s1-s3: parameter of string type
Return value	int error_no: error no of API return. int int int int int int: Return parameter of integer

	type. string string string: return parameter of string type.

Example

```
api_no = 2;
sub_api_no = 0;
errno, p1, p2, p3, p4, p5, p6, str1, str2, str3 = ebdat.callapi(api_no,sub_api_no, nil, nil, nil, nil,
nil, nil, "Test My Str2\r\n", nil, nil);
print("ebdat.callapi, result=", errno, ", str1=\"", str1, "\"\r\n");
```

3.28 GPIO Library

3.28.1 gpio.settrigtype

Description

The function is used to set the trigger mode of a specified GPIO. It has the same function as AT+CGPIO command.

Prototype	boolean gpio.settrigtype (int gpionum, int detect, int polarity[,int save])
Parameters	gpionum: the number of the the dynamic gpio detect: 0: LEVEL trigger mode 1: EDGE trigger mode polarity: 0: trigger when low level 1: trigger when high level save: 0: not save the setting (default) 1: save the setting
Return value	the result of setting: true: successful false: failed

Example

```
rst = gpio.settrigtype(2, 0,1);
print(rst,"\r\n");
```

3.28.2 gpio.setdrt

Description

The function is used to set the io direction of a specified GPIO. It has the same function as AT+CGDRT command.

Prototype	boolean gpio.setdrt (int gpionum, int gpio_io[,int save=0])

Parameters	gpionum: the number of the dynamic gpio gpio_io: 0: in 1: out save: 0: not save the setting (default) 1: save the setting
Return value	the result of setting: true: successful false: failed

Example

```
rst = gpio.setdrt(5,1);
print(rst,"\r\n");
```

3.28.3 gpio.setv

Description

The function is used to set the value of the specified GPIO. It has the same function as AT+CGSETV command.

Prototype	boolean gpio.setv (int gpionum, int gpio_hl[,int save=0])
Parameters	gpionum: the number of the dynamic gpio gpio_hl: 0: low 1: high save: 0: not save the setting (default) 1: save the setting
Return value	the result of setting: true: successful false: failed

Example

```
rst = gpio.setv(2,0);
print(rst,"\r\n");
```

3.28.4 gpio.getv

Description

The function is used to get the value of the specified GPIO. It has the same function as AT+CGGETV command.

Prototype	int gpio.getv (int gpionum)
Parameters	gpionum: the number of the dynamic gpio

Return value	the value of the GPIO: 0: low 1: high

Example

```
level = gpio.getdrt(5);
print(level,"\r\n");
```

3.28.5 gpio.startflash

Description

The function is used to start changing the GPIO state(pull up and pull down periodically)

Prototype	int gpio.startflash (int gpionum, int high_period, int low_period)
Parameters	gpionum: the number of the dynamic gpio high_period: the period of time to change the GPIO state to low level when in high level low_period: period of time to change the GPIO state to high level when in low level.
Return value	the flash ID of the GPIO

Example

```
gpio.startflash(5, 1000, 3000);
```

3.28.6 gpio.stopflash

Description

The function is used tostop changing the GPIO state(pull up and pull down periodically)

Prototype	int gpio.stopflash (int gpionum)
Parameters	gpionum: the number of the dynamic gpio
Return value	None

Example

```
gpio.stopflash(5);
```

3.29 UART Library

3.29.1 uart.set_uart_md

Description

The function is used to set the UART working mode. It has the same function as AT+CSUART command.

Prototype	int uart.set_uart_md (int mode)
Parameters	mode: 0: 3-line mode 1: 7-line mode

Return value	the result of setting: 1: successful 0: failed

Example

```
rst= uart.set_uart_md(0);
print(rst,"\r\n");
```

3.29.2 uart.get_uart_md

Description

The function is used to get the UART working mode. It has the same function as AT+CGDRT command.

Prototype	int uart.get_uart_md ()
Parameters	None
Return value	the UART working mode: 0: 3-line mode 1: 7-line mode

Example

```
mode= uart.get_uart_md();
print(mode,"\r\n");
```

3.29.3 uart.set_dcd_md

Description

The function is used to set the DCD mode or normal GPIO working mode for the UART. It has the same function as AT+CDCDMD command.

Prototype	int uart.set_dcd_md (int mode)
Parameters	mode: 0: DCD mode 1: GPIO mode
Return value	the result of setting: 1: successful 0: failed

Example

```
rst= uart.set_dcd_md(1);
print(rst,"\r\n");
```

3.29.4 uart.get_dcd_md

Description

The function is used to get the mode of a UART (DCD or normal GPIO mode). It has the same function as AT+CDCDMD command.

Prototype	int uart.get_dcd_md ()

Parameters	None
Return value	the mode the the GPIO: 0: DCD mode 1: normal GPIO mode

Example

```
mode= uart.get_dcd_md();
print(mode,"\r\n");
```

3.29.5 uart. dcd_setval

Description

The function is used to set the value of the GPIO when UART working under the normal GPIO mode. It has the same function as AT+CDCDVL command.

Prototype	int uart.dcd_setval (int value)
Parameters	value: 0: low 1: high
Return value	the result of setting: 1: successful 0: failed

Example

```
rst= uart. dcd_setval(0);
print(rst,"\r\n");
```

3.29.6 uart. dcd_getval

Description

The function is used to set the value of the GPIO when UART working under the normal GPIO mode. It has the same function as AT+DCDVL command.

Prototype	int uart.dcd_getval ()
Parameters	None
Return value	the value of the GPIO: 0: low 1: high

Example

```
rst= uart. dcd_getval();
print(rst,"\r\n");
```

3.30 ATCTL Library

3.30.1 atctl.setport

Description

The function is used to set the serial port for ATCTL purpose use. The print function cannot send data the the port owned by ATCTL.

NOTE:1. If you want use UART2,you need pull down the UART's DTR.

2. UART2 pins and SPI pins is multiplexed, so if enable SPI function, then uart2 function will be disable, and enable uart2 will be disable SPI function.

AT+CGFUNC=21,1 enable UART2 disable SPI.

AT+CGFUNC =21,0 disable UART2 ,it will be auto switch to SPI function.

Prototype	boolean atctl.setport(int port)
Parameters	port: the port to be used by ATCTL 1: UART PORT 2: MODEM PORT 3: USB AT PORT 4 : UART2 PORT -1: Release the port used by ATCTL
Return value	the setting result true: successful false: failed

Example

```
rst = atctl.setport (3)
rst = atctl.setport (-1)
```

3.30.2 atctl.recv

Description

The function is used to receive string from the port set by atctl.setport.

Prototype	string atctl.recv(int timeout)
Parameters	timeout: the timeout value for receiving string from the port set by atctl.setport.
Return value	the string received from the port. If failed, return nil.

Example

```
rst = atctl.recv (5000)
```

3.30.3 atctl.send

Description

The function is used to send string to the port set by atctl.setport.

Prototype	void atctl.send(string rpt)
Parameters	rpt: the content to send using the port set by atctl.setport.
Return value	None

Example

```
atctl.send ("test string1, test string")
```

3.30.4 atctl.clear

Description

The function is used to clear the cached string received from the port set by atctl.setport.

Prototype	void atctl.clear()
Parameters	None
Return value	None

Example

```
atctl.clear ()
```

3.30.5 atctl.atcadd

Description

The function is used to register an AT command header, user can use this function to extend an AT command that process by LUA script.

Prototype	boolean atctl.atcadd(string name, int mode)
Parameters	name: the string of AT command start with. mode: 0 Indicates this command can not be get by atctl.get() function when a async command is under processing. 1 Indicates this command can be get by atctl.get() function when a async command is under processing
Return value	the result of register operation: true: successful false: failed

Example

```
atctl.atcadd (“+CBT”, 0)
```

3.30.6 atctl.atcremove

Description

The function is used to deregister a AT command header that register by atctl.atcad() function.

Prototype	boolean atctl.atcremove(string name)
Parameters	name: the string of AT command start with.
Return value	the result of register operation: true: successful false: failed

Example

```
atctl.atcremove (“+CBT”)
```

3.30.7 atctl.atcremoveall

Description

The function is used to deregister all AT command header that register by atctl.atcad() function.

Prototype	boolean atctl.atcremoveall()
Parameters	None
Return value	the result of register operation: true: successful false: failed

Example

```
atctl.atcremoveall ()
```

3.30.8 atctl.atcget

Description

The function is used to get AT command information when ATCTL_EVENT(event id 30) event occur.

NOTE:1. If you want use UART2,you need pull down the UART's DTR.

2. UART2 pins and SPI pins is multiplexed, so if enable SPI function, then uart2 function will be disable, and enable uart2 will be disable SPI function.

AT+CGFUNC=21,1 enable UART2 disable SPI.

AT+CGFUNC =21,0 disable UART2 ,it will be auto switch to SPI function.

Prototype	int string int string int atctl.atcget()
Parameters	None
Return value	port: the port of AT command send by. 1: UART PORT 2: MODEM PORT 3: USB AT PORT 4: UART2 PORT name: the string of AT command register. cmdop: Syntax flags for the AT command parser: 1 Name field found 2 <=> found 3 <?> found 4 Argument field found cmdline: the command line. cmdstatus: Async command processing status 0 Indicates there is no async command processing. 1 Indicates there is a async command processing.

Example

```
local evt, evt_p1, evt_p2, evt_p3, evt_clock = thread.waitevt(9999999);
if(evt == 30) then
local cmd_port, cmd_name, cmd_op, cmd_line, cmd_status = atctl.atcget ();
    ……
    -- at command processing
    ……
end;
```

3.31 IIC Library

3.31.1 i2c.read_i2c_dev

Description

The function is used to read the value of a specified register or memory space. It has the same function as AT+CRIIC command.

Prototype	int i2c.read_i2c_dev (int device_id, int reg_addr, int reg_len)
Parameters	device_id: the id of the device reg_addr: the address of the device reg_len: the length to read
Return value	the value of the specified address. If failed, return 0.

Example

```
rst= i2c.read_i2c_dev(15,15,2);
print(rst,"\r\n");
```

3.31.2 i2c.write_i2c_dev

Description

The function is used to set the value of a specified register or memory space. It has the same function as AT+CWIIC command.

Prototype	int i2c.write_i2c_dev (int device_id, int reg_addr, int int reg_val, reg_len [,int option])
Parameters	device_id: the id of the device reg_addr: the address of the device reg_val: the value to write reg_len: the length to write option: option of write, default 0.
Return value	the result of setting: 1: successful 0: failed

Example

```
rst= i2c.write_i2c_dev(15,15,4660,2);
```

```
print(rst,"\r\n");
```

3.32 ADC Library

3.32.1 adc.readadc

Description

The function is used to read the value of the specified ADC channel. It has the same function as AT+CADC command.

Prototype	int adc.readadc (int type)
Parameters	type: the type of the ADC value to read 0: read the raw data (default) 1: read the temperature value
Return value	the value of the specified address. If failed, return 0.

Example

```
rst= adc.readadc(0);
print(rst,"\r\n");
```

3.33 AUDIO Library

3.33.1 audio.setmicamp1

Description

The function is used to set the audio path parameter – micamp1. It has the same function as AT+CMICAMP1 command.

Prototype	int audio.setmicamp1 (int gain)
Parameters	gain: the gain value (0-15)
Return value	the result of setting: 1: successful 0: failed

Example

```
rst= audio.setmicamp1(3);
print(rst,"\r\n");
```

3.33.2 audio.getmicamp1

Description

The function is used to get the audio path parameter – micamp1. It has the same function as AT+CMICAMP1 command.

Prototype	int audio.getmicamp1 ()
Parameters	None

Return value	the value of micamp1. If failed, return 0

Example

```
rst= audio.getmicamp1();
print(rst,"\r\n");
```

3.33.3 audio.setmicamp2

Description

The function is used to set the audio path parameter – micamp2.

Prototype	int audio.setmicamp2 (int gain)
Parameters	gain: the gain value (0-1)
Return value	the result of setting: 1: successful 0: failed

Example

```
rst= audio.setmicamp2(1);
print(rst,"\r\n");
```

3.33.4 audio.getmicamp2

Description

The function is used to get the audio path parameter – micamp1.

Prototype	int audio.getmicamp2 ()
Parameters	None
Return value	the value of micamp2. If failed, return 0

Example

```
rst= audio.getmicamp2();
print(rst,"\r\n");
```

3.33.5 audio.setsidetone

Description

The function is used to set digital attenuation of sidetone. It has the same function as AT+SIDET command.

Prototype	int audio.setsidetone (int gain)
Parameters	gain: the gain value (0-65535)
Return value	the result of setting: 1: successful 0: failed

Example

```
rst= audio.setsidetone(1000);
print(rst,"\r\n");
```

3.33.6 audio.getsidetone

Description

The function is used to get digital attenuation of sidetone. It has the same function as AT+SIDET command.

Prototype	int audio.getsidetone ()
Parameters	None
Return value	The digital attenuation of sidetone. If failed, return 0.

Example

```
rst= audio.getsidetone();
print(rst,"\r\n");
```

3.33.7 audio.settxgain

Description

The function is used to set the gain value of the TX direction for the current audio device. It has the same function as AT+CTXGAIN command.

Prototype	int audio.settxgain (int gain)
Parameters	gain: the gain value (0-65535)
Return value	the result of setting: 1: successful 0: failed

Example

```
rst= audio.settxgain(1000);
print(rst,"\r\n");
```

3.33.8 audio.gettxgain

Description

The function is used to get the gain value of the TX direction for the current audio device. It has the same function as AT+CTXGAIN command.

Prototype	int audio.gettxgain ()
Parameters	None
Return value	the TX gain value. If failed, return 0.

Example

```
rst= audio.gettxgain();
print(rst,"\r\n");
```

3.33.9 audio.setrxgain

Description

The function is used to set the gain value of the RX direction for the current audio device. It has the same function as AT+CRXGAIN command.

Prototype	int audio.setrxgain (int gain)
Parameters	gain: the gain value (0-65535)
Return value	the result of setting: 1: successful 0: failed

Example

```
rst= audio.setrxgain(1000);
print(rst,"\r\n");
```

3.33.10audio.getrxgain

Description

The function is used to get the gain value of the RX direction for the current audio device. It has the same function as AT+CRXGAIN command.

Prototype	int audio.getrxgain ()
Parameters	None
Return value	the RX gain value. If failed, return 0.

Example

```
rst= audio.getrxgain();
print(rst,"\r\n");
```

3.33.11audio.settxvol

Description

The function is used to set the volume value of the TX direction for the current audio device. It has the same function as AT+CTXVOL command.

Prototype	int audio.settxvol (int value)
Parameters	gain: the gain value (0-65535)
Return value	the result of setting: 1: successful 0: failed

Example

```
rst= audio.settxvol(1000);
print(rst,"\r\n");
```

3.33.12audio.gettxvol

Description

The function is used to get the volume of the TX direction for the current audio device. It has the same function as AT+CTXVOL command.

Prototype	int audio.gettxvol ()

Parameters	None
Return value	the volume of the TX direction for the current audio device. If failed, return 0.

Example

```
rst= audio.gettxvol();
print(rst,"\r\n");
```

3.33.13audio.setrxvol

Description

The function is used to set the volume value of the RX direction for the current audio device. It has the same function as AT+CRXVOL command.

Prototype	int audio.setrxvol (int value)
Parameters	value: the volume value (-100 - 100)
Return value	the result of setting: 1: successful 0: failed

Example

```
rst= audio.setrxvol(100);
print(rst,"\r\n");
```

3.33.14audio.getrxvol

Description

The function is used to get the volume of the RX direction for the current audio device. It has the same function as AT+CRXVOL command.

Prototype	int audio.getrxvol ()
Parameters	None
Return value	the volume of the RX direction for the current audio device. If failed, return 0.

Example

```
rst= audio.getrxvol();
print(rst,"\r\n");
```

3.33.15audio.settxftr

Description

The function is used to set the filter value of the TX direction for the current audio device. It has the same function as AT+CTXFTR command.

Prototype	int audio.settxftr (int filter1, int filter2, int filter3, int filter4, int filter5, int filter6, int filter7)
Parameters	filter1: the first fitler parameter value. filter2: the second fitler parameter value.

	filter3: the third fitler parameter value. filter4: the fourth fitler parameter value. filter5: the fifth fitler parameter value. filter6: the sixth fitler parameter value. filter7: the seventh fitler parameter value.
Return value	the result of setting: 1: successful 0: failed

Example

```
rst= audio.settxftr(1111,2222,3333,4444,5555,6666,7777);
print(rst,"\r\n");
```

3.33.16audio.gettxftr

Description

The function is used to get the filter value of the TX direction for the current audio device. It has the same function as AT+CTXFTR command.

Prototype	int int int int int int int audio.gettxftr()
Parameters	None
Return value	There are 7 return values for the filter parameters of the TX direction for the current audio device: filter1: the first fitler parameter value. filter2: the second fitler parameter value. filter3: the third fitler parameter value. filter4: the fourth fitler parameter value. filter5: the fifth fitler parameter value. filter6: the sixth fitler parameter value. filter7: the seventh fitler parameter value.

Example

```
filter1,filter2,filter3,filter4,filter5,filter6,filter7= audio.settxftr(1111,2222,3333,4444,5555, 6666, 7777);
print(filter1, " ", filter2, " ", filter3, " ", filter4, " ", filter5, " ", filter6, " ", filter7 ,"\r\n");
```

3.33.17audio.setrxftr

Description

The function is used to set the filter value of the RX direction for the current audio device. It has the same function as AT+CRXFTR command.

Prototype	int audio.setrxftr (int filter1, int filter2, int filter3, int filter4, int filter5, int filter6, int filter7)
Parameters	filter1: the first fitler parameter value. filter2: the second fitler parameter value. filter3: the third fitler parameter value.

	filter4: the fourth fitler parameter value. filter5: the fifth fitler parameter value. filter6: the sixth fitler parameter value. filter7: the seventh fitler parameter value.
Return value	the result of setting: 1: successful 0: failed

Example

```
rst= audio.setrxftr(1111,2222,3333,4444,5555,6666,7777);
print(rst,"\r\n");
```

3.33.18audio.getrxftr

Description

The function is used to get the filter value of the RX direction for the current audio device. It has the same function as AT+CRXFTR command.

Prototype	int int int int int int int audio.getrxftr ()
Parameters	None
Return value	There are 7 return values for the filter parameters of the RX direction for the current audio device: filter1: the first fitler parameter value. filter2: the second fitler parameter value. filter3: the third fitler parameter value. filter4: the fourth fitler parameter value. filter5: the fifth fitler parameter value. filter6: the sixth fitler parameter value. filter7: the seventh fitler parameter value.

Example

```
filter1,filter2,filter3,filter4,filter5,filter6,filter7= audio.getrxftr(1111,2222,3333,4444,5555,6666,
7777);
print(filter1, " ", filter2, " ", filter3, " ", filter4, " ", filter5, " ", filter6, " ", filter7 ,"\r\n");
```

3.33.19audio.setvollvl

Description

The function is used to set the audio path parameter – RX volume for the current audio device. It has the same function as AT+CVLVL command.

Prototype	int audio.setvollvl (int level, int value)
Parameters	int level: sound level number (1 - 4) value: sound level value (-5000 - 5000)
Return value	the result of setting: 1: successful 0: failed

Example

```
rst= audio.setvollvl(4,1000);
print(rst,"\r\n");
```

3.33.20audio.getvollvl

Description

The function is used to get the audio path parameter – RX volume for the current audio device. It has the same function as AT+CVLVL command.

Prototype	int audio.getvolval ()
Parameters	None
Return value	the value of the RX volume.

Example

```
rst= audio.getvollvl();
print(rst,"\r\n");
```

3.34 GPS Library

3.34.1 gps.start

Description

The function is used to start the GPS function. For the corresponding AT command, please refer to AT+CGPSCOLD and AT+CGPSHOT.

Prototype	int gps.start (int mode)
Parameters	mode: the start mode 1: hot start 2: code start
Return value	the result of starting GPS: 1: successful 0: failed

Example

```
rst= gps.start(1);
print(rst,"\r\n");
```

3.34.2 gps.close

Description

The function is used to stop the GPS function. For the corresponding AT command, please refer to AT+CGPS.

Prototype	int gps.close ()
Parameters	None
Return value	the result of stopping GPS: 1: successful

	0: failed

Example

```
rst= gps.close();
print(rst,"\r\n");
```

3.34.3 gps.gpsinfo

Description

The function is used to get the reported GPS information. For the corresponding AT command, please refer to AT+CGPSINFO.

Prototype	string gps.gpsinfo ()
Parameters	None
Return value	the GPS information.

Example

```
rst= gps.gpsinfo();
print(rst,"\r\n");
```

3.34.4 gps.gpssetmode

Description

The function is used to set the mode of the GPS. For the corresponding AT command, please refer to AT+CGPS.

Prototype	boolean gps.gpssetmode (int mode)
Parameters	mode: the mode of GPS 1: standalone 2: MSB 3: MSA
Return value	the result of setting: True: successful false: failed

Example

```
print(rst,"\r\n");
```

3.34.5 gps.gpsgetmode

Description

The function is used to get the mode of the GPS. For the corresponding AT command, please refer to AT+CGPS.

Prototype	int gps.gpsgetmode ()
Parameters	None
Return value	the mode of GPS: 1: standlone 2: MSB

	3: MSA

Example

```
rst= gps.gpsgetmode();
print(rst,"\r\n");
```

3.34.6 gps.gpsseturl

Description

The function is used to set the server URL of the GPS. For the corresponding AT command, please refer to AT+CGPSURL.

Prototype	boolean gps.gpsseturl (string url)
Parameters	url: the URL of GPS
Return value	the result of setting: True: successful false: failed

Example

```
rst= gps.gpsseturl("123.123.123.123:8888");
print(rst,"\r\n");
```

3.34.7 gps.gpsgeturl

Description

The function is used to get the mode of the GPS. For the corresponding AT command, please refer to AT+CGPSURL.

Prototype	string gps.gpsgeturl ()
Parameters	None
Return value	the server URL of the GPS

Example

```
rst= gps.gpsgeturl();
print(rst,"\r\n");
```

3.34.8 gps.gpssetssl

Description

The function is used tog set the SSL mode of the GPS. For the corresponding AT command, please refer to AT+CGPSSSL.

Prototype	string gps.gpssetssl (int ssl)
Parameters	ssl: the SSL mode 0: do not use SSL 1: use SSL
Return value	the result of setting: True: successful false: failed

Example

```
rst= gps.gpssetssl(0);
print(rst,"\r\n");
```

3.34.9 gps.gpsgetssl

Description

The function is used to get the SSL mode of the GPS. For the corresponding AT command, please refer to AT+CGPSSSL.

Prototype	int gps.gpsgetssl ()
Parameters	None
Return value	the SSL mode of the GPS 0: do not use SSL 1: use SSL

Example

```
rst= gps.gpsgetssl();
print(rst,"\r\n");
```

3.35 NMEA Library

3.35.1 nmea.getinfo

Description

The function is used to get the latest NMEA information when GPS opened.

Prototype	string nmea.getinfo([int filter])
Parameters	int filter: 1: GGA 2: RMC 43: GSV 8: GSA 16: VTG 32: PSTIS
Return value	string nmea_info: the latest NMEA packets.

Example

```
printdir(1)
local recv_count = 0;

--[[the nmea.getinfo() is a standalone function,
it has no relation with nmea.open, nmea.close, nmea.recv, nmea.clear]]

gps.gpsstart(1);
while (recv_count < 50) do
   recv_count = recv_count + 1;
```

```
    local nmea_info = nmea.getinfo();
    print("nmea_info = ", nmea_info, "\r\n");
    vmsleep(2000);
end;
gps.gpsclose();
```

3.35.2 nmea.open

Description

The function is used to enable NMEA data receiving(the same data using physical USB NMEA port).

Prototype	boolean nmea.open([int filter])
Parameters	int filter: 1: GGA 2: RMC 43: GSV 8: GSA 16: VTG 32: PSTIS
Return value	boolean result: 0: successful. other: failed.

Example

```
printdir(1)
NMEA_EVENT = 35

local recv_count = 0;

gps.gpsstart(1);
nmea.open();
while (true) do
    local evt, evt_p1, evt_p2, evt_p3, evt_clock = thread.waitevt(999999);
    if (evt and evt == NMEA_EVENT) then
        local nmea_data = nmea.recv(0);
  if (nmea_data) then
      recv_count = recv_count + 1;
      print("nmea_data, len=", string.len(nmea_data), "\r\n");
      print(nmea_data);
      if (recv_count >= 100) then
          break;
      end;
  end;
    end;
end;
```

```
nmea.close();
gps.gpsclose();
```

3.35.3 nmea.close

Description

The function is used to disable NMEA data receiving.

Prototype	void nmea.close(void)
Parameters	None
Return value	None

Example

(refer to example of nmea.open)

3.35.4 nmea.recv

Description

The function is used to receive cached NMEA data.. Currently the maximum cached data length is 8K bytes.

Prototype	string nmea.recv(int timeout)
Parameters	int timeout: the timeout value for receiving NMEA data. The unit is millisecond.
Return value	string nmea_data: The NMEA data received.

Example

(refer to example of nmea.open)

3.35.5 nmea.clear

Description

The function is used to clear cached NMEA data..

Prototype	void nmea.recv(void)
Parameters	None
Return value	None

Example

(refer to example of nmea.open)

3.36 SPI Library

3.36.1 spi.set_freq

Description

The function is used to set the frequency of SPI.

Prototype	void spi.set_freq(int min_slave_freq, int

	max_slave_freq, int deassertion_time)
Parameters	min_slave_freq: the minimum frequency of the slave device max_slave_freq: the maximum frequency of the slave device. deassertion_time: the deassertion time
Return value	None

Example

spi.set_freq(1000, 500000, 1000)

3.36.2 spi.set_clk

Description

The function is used to set the clock information of SPI.

Prototype	void spi.set_clk(int clk_mode, int clk_pol, int transfer_mode)
Parameters	clk_mode: the mode of the clock 0: normal 1: always on clk_pol: the polarity of the clock 0: low when active 1: high when active transfer_mode: transfer mode 0: input first mode 1: output first mode
Return value	None

Example

spi.set_clk(0, 1, 1)

3.36.3 spi.set_cs

Description

The function is used to set the CS information of SPI.

Prototype	void spi.set_cs (int cs_mode, int cs_pol)
Parameters	cs_mode: the mode of the cs 0: deasserted 1: keep asserted cs_pol: the polarity of the cs 0: low when active 1: high when active
Return value	None

Example

spi.set_cs(0, 0)

3.36.4 spi.set_num_bits

Description

The function is used to set the packing information of SPI.

Prototype	void spi.set_num_bits (int num_bits, int input_packing, int output_packing)
Parameters	num_bits: the number of bits to transfer each time for the SPI input_packing: the packing mode for the input 0: disabled 1: enabled output_packing: the unpacking mode for the output 0: disabled 1: enabled
Return value	None

Example

```
spi.set_num_bits(8, 0, 0)
```

3.36.5 spi.config_device

Description

The function is used to configure the device of SPI.

Prototype	void spi.config_device ()
Parameters	None
Return value	None

Example

```
spi.config_device();
```

3.36.6 spi.write

Description

The function is used to write data to the SPI device.

Prototype	boolean spi.write (int data, int reg_num, int len)
Parameters	data: the data to write reg_num: the id of the register len: the length of the data to write
Return value	Result of writing data to LCD SPI device TRUE: successful FALSE: failed

Example

```
spi.write(233, nil, 1);
```

3.36.7 spi.read

Description

The function is used to read data from the SPI device.

Prototype	int int int int spi.read (int reg_num, int len)
Parameters	reg_num: the id of the register len: the length of the data to read
Return value	The data(maximum 4 bytes) read from SPI device

Example

```
d1, d2, d3, d4 = spi.read(nil, 1);
```

3.36.8 spi.write_1_byte_multi_times

Description

The function is used to write 1 byte to the LCD SPI device for multiple times.

Prototype	int spi.write_1_byte_multi_times (int data, int start_pixel_x, int end_pixel_x, int underline, int underline_or_value, int reverse_color)
Parameters	data: data to write to the SPI device start_pixel_x: the start x-pixel position on LCD to display end_pixel_x: the end x-pixel position on LCD to display underline: the underline style nil: no underline 1: default underline 2: underline of dot line underline_or_value: the value for underline to display on LCD reverse_color: reverse color
Return value	The new x-pixel position after writing.

Example

```
spi.write_lcd_1_byte_multi_times(0, rect.left, rect.right, underline, phone_dev.underline_value, reverse_color)
```

3.37 ZIGBEE MG2455 Library

3.37.1 zbmg2455.open

Description

The function is used to open ZIGBEE device.

Prototype	void zbmg2455.open(void)

Parameters	None
Return value	None

Example

```
zbmg2455.open();
```

3.37.2 zbmg2455.close

Description

The function is used to close ZIGBEE device.

Prototype	void zbmg2455.close(void)
Parameters	None
Return value	None

Example

```
zbmg2455.close();
```

3.37.3 zbmg2455.set_mode

Description

The function is used to set the mode of ZIGBEE device.

Prototype	void zbmg2455.set_mode(int mode)
Parameters	int mode: 0: CLIENT 1: SERVER
Return value	None

Example

```
zbmg2455.set_mode(0);
```

3.37.4 zbmg2455.set_param

Description

The function is used to set the parameter of ZIGBEE device.

Prototype	boolean zbmg2455.set_param(int param_id, string param_value)
Parameters	int param_id: 0: PHYSICAL CHANNEL 1: NWK_PANID 2: NWK_ADDR 3: EXTENDED_PANID 4: IEEE_ADDR 5: NWK_SEARCHING string param_value: The parameter value.
Return value	boolean: the result of operating:

	true: successful false: failed.

Example

```
zbmg2455.set_param(param_id, param_value);
```

3.37.5 zbmg2455.send

Description

The function is used to set send data using ZIGBEE.

Prototype	void zbmg2455send(string data)
Parameters	data: the data to be sent to virtual serial port
Return value	None

Example

```
zbmg2455.send(data);
```

3.37.6 zbmg2455.recv

Description

The function is used to set receive data using ZIGBEE.

Prototype	string zbmg2455.recv(int timeout)
Parameters	timeout: the timeout value in ms to receive data.
Return value	The data received on ZIGBEE

Example

```
data = zbmg2455.recv(5000);
```

3.37.7 zbmg2455.clear

Description

The function is used to clear the cached data received using ZIGBEE.

Prototype	void zbmg2455.clear()
Parameters	None
Return value	None

Example

```
zbmg2455.clear();
```

3.38 DEBUG Library

3.38.1 debug.debug

Description

The function is used to let the LUA engine enter debug mode.

Prototype	void debug.debug()

Parameters	None
Return value	None

Example

```
debug.debug();
--[[After calling this function, the following string will be reported to external AT port:
+CSCRIPT: lua_debug> Please Enter Debug Command
Then the AT+DBC can be used to enter debug command:
AT+DBC
>print("hello")<CR>
OK
hello
AT+DBC
>cont<CR>
OK
]]
```

3.38.2 debug.getfenv()

Description

The function returns the environment of an object.

Prototype	table debug.getfenv (object o)
Parameters	object o: the object that needs to get the environment
Return value	table: the environment of the object

Example

```
environment = debug.getfenv(var1);
```

3.38.3 debug.gethook

Description

The function returns the current hook setting of the thread

Prototype	function, string, int debug.gethook ([thread t])
Parameters	thread t: the thread that needs to get the hook setting
Return value	function: the hook function string: the mask "c": The hook is called every time Lua calls a function; "r": The hook is called every time Lua returns from a function; "l": The hook is called every time Lua enters a new line of code. int: the count for calling the hook function

Example

hook, mask, count = debug.gethook();

3.38.4 debug.getinfo

Description

The function returns a table with information about a function

Prototype	table debug.getinfo ([thread t,] function func[, string what])
Parameters	thread t: the thread that needs to get the information function func: the function that needs to get the information. You can give a number as the value of function which means the function running at level function of the call stack of the given thread. Level 0 is the current function. string what: this is used to describe which fields to fill in.
Return value	table: the information.

Example

information = debug.getinfo(0, “S1”);

3.38.5 debug.getlocal

Description

The function returns the name and the value about a local variable

Prototype	string object debug.getlocal ([thread t,] int level, int local)
Parameters	thread t: the thread that is used to get the local variable. int level: the level of the thread stack int local: the variable index in the thread stack.
Return value	string: the variable name object: the variable value

Example

name, value = debug.getlocal(0, 1);

3.38.6 debug.getmetatable

Description

The function returns the metatable about a object

Prototype	metatable debug.getmetable(object o)
Parameters	object o: the object that needs to get the metatable.
Return value	metatable: the metatable of the object

Example

metatab = debug.getmetatable(var1);

3.38.7 debug.getregistry

Description

The function returns the registry table.

Prototype	table debug.getregistry ()
Parameters	None
Return value	table: the registry table

Example

reg = debug.getregistry();

3.38.8 debug.getupvalue

Description

The function returns the name and the value of the upvalue with index up of the function func.

Prototype	string object debug.getupvalue (function func, int up)
Parameters	function func: the function that needs to get the upvalue int up: the index of the upvalue.
Return value	string: the name of the upvalue object: the value of the upvalue

Example

name, value = debug.getupvalue(func1, 1);

3.38.9 debug.setfenv

Description

The function sets the environment of an object.

Prototype	object debug.setfenv (object o, table t)
Parameters	object o: the object that needs to set the environment table t: the environment table
Return value	object: the object after setting the environment

Example

obj = debug.setfenv(obj, tab);

3.38.10debug.sethook

Description

The function set a given function as a hook

Prototype	void debug.sethook ([thread t,] function func, string

	mask, int count)
Parameters	thread t: the thread that needs to get the hook setting function func: the hook function string mask: the mask "c": The hook is called every time Lua calls a function; "r": The hook is called every time Lua returns from a function; "l": The hook is called every time Lua enters a new line of code. int count: the count for calling the hook function
Return value	None

Example

debug.sethook(hook, mask, count);

3.38.11debug.setlocal

Description

The function assigns a value to a local variable.

Prototype	string debug.setlocal ([thread t,] int level, int local, object value)
Parameters	thread t: the thread that is used to get the local variable. int level: the level of the thread stack int local: the variable index in the thread stack. object value: the value to assign
Return value	string: the variable name

Example

name = debug.setlocal(0, 1, value);

3.38.12debug.setmetatable

Description

The function set the metatable for a given object

Prototype	boolean debug.setmetable(object o, table t)
Parameters	object o: the object that needs to set the metatable. table t: the metatable to set
Return value	the result of the setting: true: successful false: failed

Example

debug.setmetatable(var1, metatab);

3.38.13debug.setupvalue

Description

The function assign a value for a upvalue

Prototype	string debug.setupvalue(function func, int up, object value)
Parameters	function func: the function containing the upvalue int up: the index of the upvalue object value: the value to set.
Return value	string: the name of the value

Example

```
name = debug.setupvalue(func, 1, value);
```

3.38.14debug.traceback

Description

The function returns a string with a traceback of the call stack.

Prototype	string debug.traceback([thread t,] [string message][, int level])
Parameters	thread t: the thread that is used to trace information string message: this is an optional string that is appended at the beginning of the traceback. int level: the level of the thread stack
Return value	string: the name of the value

Example

```
name = debug.setupvalue(func, 1, value);
```

3.38.15debug.continue

Description

The function is equal to input “cont” command to let the script continue to run.

Prototype	void debug.continue()
Parameters	None
Return value	None

Example

```
debug.continue();
```

3.38.16debug.broken

Description

The function is used to judge whether the script is calling debug.debug() function now.

Prototype	boolean debug.broken()

Parameters	None
Return value	Whether it is calling debug.debug(): true: yes false: no

Example

```
rst = debug.broken();
```

3.38.17debug.hooksubthreads

Description

The function is used to set the hook function for all the sub-threads that will be launched later.

Prototype	void debug.hooksubthreads(function func, string mask, int count)
Parameters	function func: the hook function string mask: the mask "c": The hook is called every time Lua calls a function; "r": The hook is called every time Lua returns from a function; "l": The hook is called every time Lua enters a new line of code. int count: the count for calling the hook function
Return value	None

Example

```
debug.hooksubthreads(myhookfunc, "crl", 1);
```

3.38.18debug.print

Description

The function is used to print debug data. This function is similar to print() function, but it is not affected by printdir() function. The total print string length cannot exceed 1024 bytes.

Prototype	void debugprint (var1,…)
Parameters	var1… : any value that can be converted to a string
Return value	None

Example

```
prompt = "This is my prompt, count="
count = 1;
debug.print(prompt, count, "\r\n");
```

3.39 DEVIO Library

3.39.1 devio.open

Description

The function is used to open a stream device.

NOTE:1. If you want use UART2,you need pull down the UART's DTR.

2. UART2 pins and SPI pins is multiplexed, so if enable SPI function, then uart2 function will be disable, and enable uart2 will be disable SPI function.

AT+CGFUNC=21,1 enable UART2 disable SPI.

AT+CGFUNC =21,0 disable UART2 ,it will be auto switch to SPI function.

Prototype	bool devio.open (int dev_id)
Parameters	int dev_id:[IN] 0:UART 1:AT 2:MODEM 3:UART2
Return value	the result of the opening: true: successful false: failed

Example

```
ret = devio.open (0);
```

3.39.2 devio.close

Description

The function is used to close a stream device.

Prototype	bool devio.close (int dev_id)
Parameters	int dev_id:[IN] 0:UART 1:AT 2:MODEM 3:UART2

Return value	the result of the closing: true: successful false: failed

Example

ret = devio. close (0);

3.39.3 devio.read

Description

The function is used to read bytes from the stream device.

Prototype	function, string devio.read(int dev_id,int timeout)
Parameters	int dev_id:[IN] 0:UART 1:AT 2:MODEM 3:UART2 int timeout: [IN] the timeout value in millisecond.
Return value	string: [OUT]buffer to read data.

Example

ret = devio. read (0, 200);

3.39.4 devio.write

Description

The function is used to write bytes to the stream device.

Prototype	bool devio.write(int dev_id, string p_buf, int n_size)
Parameters	int dev_id:[IN] 0:UART 1:AT 2:MODEM 3:UART2 string p_buf: [IN]buffer to write data. int n_size: [IN]buffer len.
Return value	the result of the writing: true: successful false: failed

Example

ret = devio.write (0, “abcd”, 4);

4 LUA Script operations

4.1 Executing a LUA script

The steps required to have a script running by LUA engine of the module are:

1. Write the LUA script
2. Download the LUA script into the EFS of the module
3. Execute the LUA script

4.1.1 Write LUA script

Open the notepad.exe program on windows operating system, and enter the following text which prints “HELLO WORLD” to TE and then ends.

```
printdir(1)
print("HELLO WORLD!\r\n")
```

4.1.2 After entering the text, save it as “helloworld.lua”, and then close the notepad.exe program.

4.1.3 Download LUA script

Use PC tool to transfer the “helloworld.lua” file to the C:\ directory on the

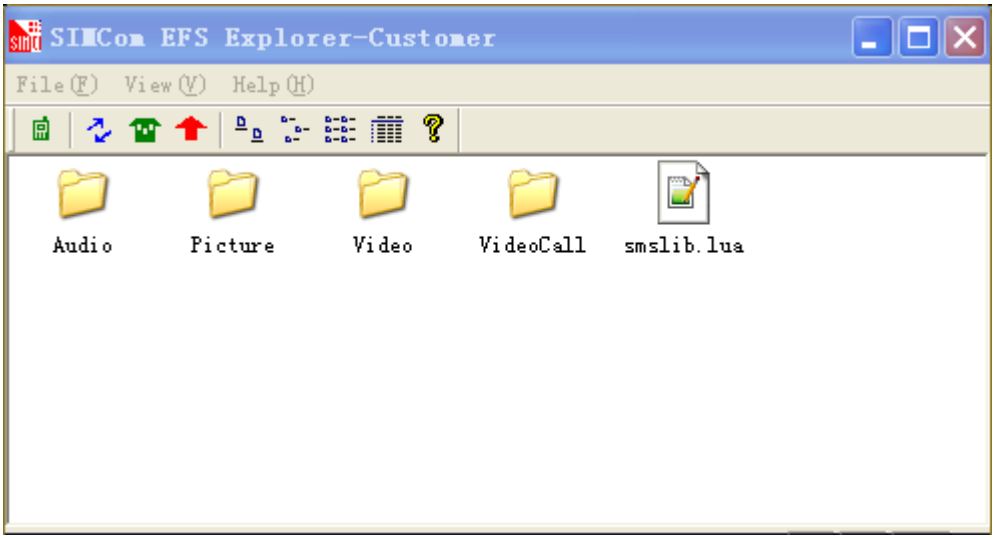

diag_lua. rar

module.

User just needs to modify the following source code in fsdiagput.lua, and the double click putfile.bat, then the file configured will be transferred to the UE.

```
43  --download files to module
44  local module_file_name = "C:\\helloworld.lua";
45  local local_pc_file_name = "helloworld.lua";
46  local result = fs_diag_transfer_file_to_ue(diag_session, module_file_name, local_pc_file_name);
```

Also another PC tool “SIMCOM EFS Explorer” can be used to transfer files:



NOTE:

When downloading multiple script files to the module, Execute AT+CSCRIPTAUTO=0 first is recommended, Following are the steps:

1. Execute AT+CSCRIPTAUTO=0
2. Download the scripts
3. If needed, execute AT+CSCRIPTAUTO=1

4.1.4 Compile LUA script

User may compile the LUA script to binary format files. Following is an example that the test1.lua is compiled to test1.out and encrypted using password “123456”.

AT+CSCRIPTPASS=””,”123456”
OK
AT+CSCRIPTCL=”test1.lua”
OK
+CSCRIPT: 0

4.1.5 Execute LUA script

Connect the module to PC, and open the AT port or UART port using hypertrm.exe, then type the following AT command to run the LUA script:

AT+CSCRIPTSTART=”helloworld.lua”, 1

After executing the script, the module will report +CSCRIPT: <result>. If the <result> equals 0, it means executing the script successfully. Other value for the <result> means failing to execute the script. Following is the executing result of the “helloworld.lua” script:

HELLO WORLD!
+CSCRIPT: 0

4.1.6 Stop the running LUA script

User may input AT+CSCRIPTSTOP? command on TE to query which script is running now inside the module. If there is an active script, the AT+CSCRIPTSTOP? command may report +CSCRIPTSTOP: <FILENAME>. User also may input AT+CSCRIPTSTOP to stop the active script.

4.1.7 Run the script automatically

User may save the script as “C:\autorun.lua” or “C:\autorun.out” on the module, and each time when the module powers on, this script will run automatically. If both files exist, the “C:\autorun.lua” file will run automatically.

4.2 Debug the active LUA script

The debug of the active LUA script needs to be done on the target, and user can use the following two methods to debug the application.

In developing stage, any source code error may cause the module reboot. So in developing stage, the AT+CSCRIPTAUTO=0 is recommended, and after reboot, the module may not crash for the same reason, or else which may cause the module in keep rebooting state.

4.2.1 Use the second parameter of AT+CSCRIPTSTART

AT+CSCRIPTSTART command provides the second parameter to support reporting error of LUA script to TE. When user is developing the script, it is very useful, and may report the grammar or running error immediately and help user to correct it. Following is an example of the error report:

AT+CSCRIPTSTART=”*myprogram.lua*”, 1

+LUA ERROR: myprogram.lua:5: 'then' expected near 'print'

+CSCRIPT: 3

Following is the “myprogram.lua” script:

```
printdir(1)
print("HELLO WORLD!\r\n")
condition = 1
if (condition == 1)
  print("conditon equals 1\r\n");
elseif (condition == 2) then
  print("conditon equals 2\r\n");
else
  print("other value\r\n");
end;
```

4.2.2 Use printdir and print functions to debug script

The printdir and print functions are mainly designed to assist user to trace the work flow of the script. When developing the script, user can set printdir(1), which may let the print function trace debug information to TE. When releasing the script, user can just set printdir(0), which then let the print function stop tracing debug information to TE.

4.3 Compile the LUA script

4.3.1 Compile the LUA script

To improve efficiency, user can compile the LUA script to binary mode. The corresponding AT command is AT+CSCRIPTCL. Following is an example of compiling "*C:\test.lua*" to "*C:\test.out*":

AT+CSCRIPTCL="*test.lua*"

Or

AT+CSCRIPTCL="*test.lua*","*test.out*"

4.3.2 Compile and Encrypt the LUA script

Also user can use password to enhance the security of the LUA script. By using the AT+CSCRIPTPASS and AT+CSCRIPTCL commands, user can compile and encrypt the script file to binary mode. Following is an example of compiling "*C:\test.lua*" to "*C:\test.out*" which will also be encoded using the password "12345678":

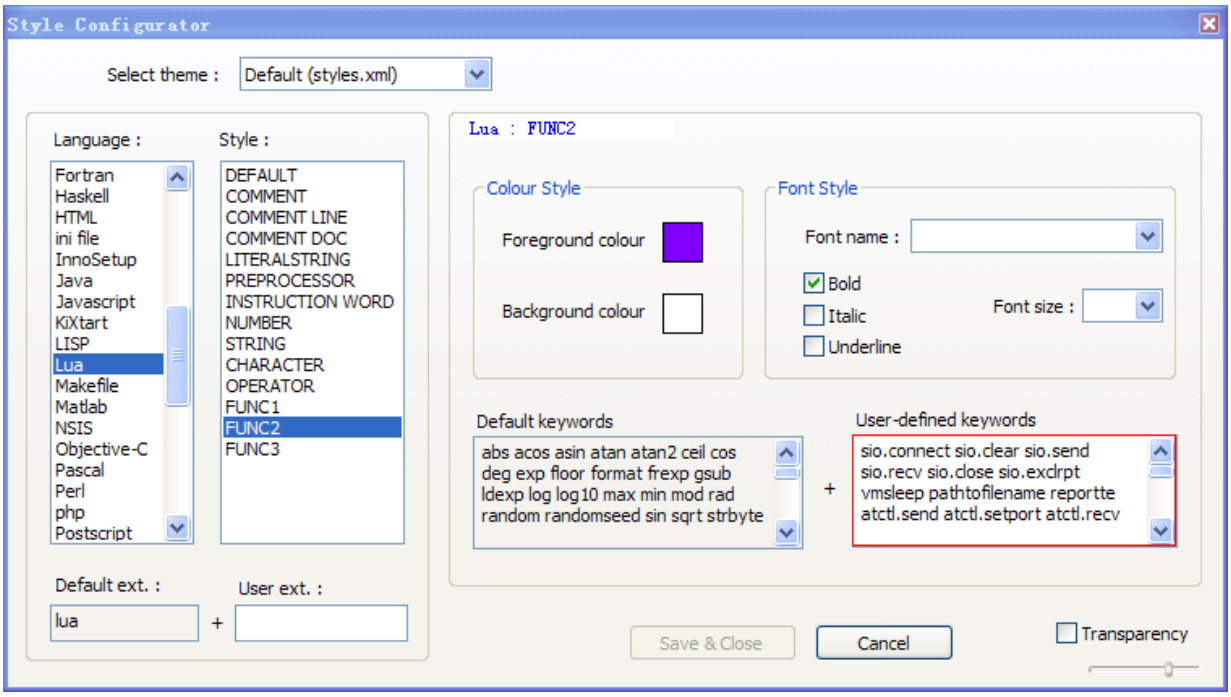
AT+CSCRIPTPASS="","12345678"

AT+CSCRIPTCL="*test.lua*"

If dofile() function is used in the script file, the parameter file name needn't to be changed to "*.out". The LUA engine will automatically try to run the "*.out" file if the "*.lua" file doesn't exist.

4.4 Use Notepad++ to edit the LUA script

Notepad++ is a light and free text editor which supports LUA language grammar lightening, and it is recommended for customers to edit the SIMCom module LUA script. User can set the SIMCom APIs by selecting the "Settings|Style configurator…" menu, and input the extended API names in the "Style Configurator" dialog like the following:



4.5 Enable or disable LUA power on check

Sometimes, the developed LUA scripts may have some bugs, and cause the module to reset frequently on each powering up(scripts use autorun feature). In such condition, customer may not able to stop or remove the LUA scripts and let the module stop reset and enter normal working mode.

New LUA module add a function to check whether the autorun.lua script can start correctly on powering up timing. If the module continuously works incorrectly(each time it cannot work for over two minutes) for 500 times, it will disable the LUA and EBDAT autorun function.

If customer does not want to use this function, The AT+CPWRONCHK=0 can be used to disable checking.

5 LUA Script Samples

This section shows some samples which are aimed to assist customer to be familiar with SIMCom LUA extended APIs.

5.1 Execute AT Commands

The following script performs the ATI query every 2 seconds.

```
printdir(1)
cmd = "ATE0\r\n"
sio.send(cmd)
--receive response within 5000 ms
rsp = sio.recv(5000)
print(rsp)

while true do
  cmd = "ATI\r\n";
  --send ATI to the AT command handling engine
  sio.send(cmd)
  --receive response from the AT command handling engine within 5000 ms
  rsp = sio.recv(5000)
  print(rsp)
  vmsleep(2000)
end
```

5.2 Perform GPIO Pins Operation

The following script gives an example of operating GPIO pins.

```
printdir(1)
if (true) then
  --Get the GPIO2 level state
  gio = gpio.getv(2);
  print(gio,"\r\n");
  --Set the GPIO2 to low level
  rst = gpio.setv(2,0);
  print(rst,"\r\n");
  --Set the GPIO0 ISR to be triggered at high level
  --and the trigger mode is LEVEL trigger.
  rst = gpio.settrigtype(0,1);
  print(rst,"\r\n");
  --Set the direction of GPIO5 to "out"
  rst = gpio.setdrt(5,1);
  print(rst,"\r\n");
end;
```

5.3 Perform GPS Operation

The following script let the GPS device run with code start for 60 seconds and then run with hot start for 30 seconds.

```
------------------------------------------------------
function show_gps_mode_string()
  mode=gps.gpsgetmode();

  if(mode == 1) then
    print("---current mode is standalone!---\r\n");
  elseif(mode == 2) then
    print("---current mode is MSB!---\r\n");
  elseif(mode == 3) then
    print("---current mode is MSA!---\r\n");
  end;
  return;
end

------------------------------------------------------
--CONFIGURATION SECTION
test_coldstart_loop_count = 60;
test_hotstart_loop_count = 30;

printdir(1)
print("------------GPS begin test----------------\r\n");
------------STANDALONE模式----------------
g_mode = gps.gpsgetmode();
if(g_mode ~= 1) then
  s_mode = gps.gpssetmode(1);
  if(not s_mode) then
    print("---GPS Mode setting error!---\r\n");
    return;
  end;
end;
show_gps_mode_string();
```

```
vmsleep(1000);

------------code start----------------
start = gps.gpsstart(2)
if(start == true) then
  print("-------GPS coldstart success!--------\r\n")
end;
count = 0;
while (count < test_coldstart_loop_count) do
  count = count+1;
  print("-----------run count=",count,"-------\r\n");
  fix = gps.gpsinfo();
  print(fix,"\r\n","\r\n");
  vmsleep(1000);
end;
close = gps.gpsclose();
if(close == true) then
  print("-------------GPS colse success!-------------\r\n");
end;

print("\r\n------waiting 10s for hotstart------\r\n\r\n");
vmsleep(10000);

------------hot start----------------
start = gps.gpsstart(1);
if(start == true) then
  print("--------GPS hotstart success!-------------\r\n")
end;
count = 0;
while (count < test_hotstart_loop_count) do
  count = count+1;
  print("-----------run count=",count,"-------\r\n");
  fix = gps.gpsinfo()
  print(fix,"\r\n","\r\n")
  vmsleep(1000)
end;
close = gps.gpsclose()
if(close == true) then
  print("-------------GPS colse success!-------------\r\n");
end;
print("----------GPS finished test--------------\r\n");
```

5.4 Perform ATCTL Operation

The following script sets the USBAT port to ATCTL mode, and control the input and output of the port instead of letting the normal AT command processing engine to control the port.

```
--supported port for atctl.setport(...)
ATCTL_UART_PORT  = 1
ATCTL_MODEM_PORT = 2
ATCTL_USBAT_PORT = 3
--  -1 is used to release the port
ATCTL_INVALID_PORT = -1

printdir(1)
--set the USBAT port to be used by ATCTL
atctl.setport(ATCTL_USBAT_PORT)
atctl.send("\r\nplease press any key in the atctl port\r\n");
--Now, user can input and data from TE, which will not be processed by
--AT command processing engine, but received by ATCTL module
count = 0;
while (count < 100) do
  --receive the data input from TE
  data = atctl.recv(15000);
  if (data) then
    count = count + 1;
    --send data to TE
    atctl.send("received data from external port:"..data.."\r\n");
  end;
end;
--Release the USBAT port, and set it as
--normal AT command processing state
atctl.setport(ATCTL_INVALID_PORT);
```

5.5 Perform FTP Operation

The following script gives an example of FTP put and get operations.

```
printdir(1)
------------------------------------------------------------
pdp_apn = "mytestapn"; server = "myftpserver"; port = 21;
name = "mytestname"; pass = "mytestpass";
remote_file = "/up_normal.jpg";
uplocal_file = "c:\\Picture\\normal.jpg";
downlocal_file = "c:\\Video\\down_normal.jpg"; passive = 0;
------------------------------------------------------------
sio.send("ATE0\r\n"); rsp = sio.recv(5000);
cmd = string.format("AT+CGSOCKCONT=1,\"IP\",\"%s\"\r\n",pdp_apn);
sio.send(cmd);
rsp = sio.recv(5000);
--upload a file to FTP server
rst = ftp.simpput(server, port, name, pass, remote_file,
                  uplocal_file, passive);
if (rst ~= 0) then
  print("ftp.simpput failed, rst = ", rst,"\r\n");
else
  print("ftp.simpput succeeded\r\n");
  put_suc = true;
end;
--download a file from FTP server
rst = ftp.simpget(server, port, name, pass, remote_file,
                  downlocal_file, passive);
if (rst ~= 0) then
  print("ftp.simpget failed, rst = ", rst,"\r\n");
else
  print("ftp.simpget succeeded\r\n");
end;
sio.send("ATE1\r\n"); rsp = sio.recv(5000);
```

5.6 Perform EFS Operation

The following script gives an example of operating files in EFS.

```
printdir(1)
--open c:\test1.txt as writable
file = io.open("c:\\test1.txt","w");
assert(file)
file:trunc();
file:write("test content\r\ntest\r\n");
file:close();
--read all data from c:\test1.txt
file = io.open("c:\\test1.txt","r");
assert(file)
cnt = file:read("*a");
print(cnt)
file:close();
--read a line from c:\test1.txt each time and print it
file = io.open("c:\\test1.txt","r");
assert(file)
for line in file:lines() do
  print("line content = ", line);
end
file:close();
--read a line from c:\test1.txt and print it
file = io.open("c:\\test1.txt","r");
assert(file)
cnt = file:read("*l");
print(cnt)
file:close();
```

5.7 Perform Bitwise Operation

The following script gives an example of bitwise operation.

```
--// script example START --------------------------
printdir(1)
print ("bit.bits = " .. bit.bits, "\r\n")
assert (bit.band (0, 0) == bit.cast (0))
assert (bit.band (0, -1) == bit.cast (0))
assert (bit.band (-1, -1) == bit.cast (-1))
assert (bit.bor (0, 0) == bit.cast (0))
assert (bit.bor (0, -1) == bit.cast (-1))
assert (bit.bor (-1, -1) == bit.cast (-1))
assert (bit.bxor (0, 0) == bit.cast (0))
assert (bit.bxor (0, -1) == bit.cast (-1))
assert (bit.bxor (-1, -1) == bit.cast (0))
assert (bit.bnot (0) == bit.cast (-1))
assert (bit.bnot (-1) == bit.cast (0))
assert (bit.lshift (0, 0) == bit.cast (0))
assert (bit.lshift (-1, 0) == bit.cast (-1))
assert (bit.rshift (0, 0) == bit.cast (0))
assert (bit.rshift (-1, 0) == bit.cast (-1))
for nb = 1, bit.bits do
 local a = 2 ^ nb - 1
 print ("nb = " .. nb .. ", a = " .. a, "\r\n")
 assert (bit.band (a, 0) == bit.cast (0))
 assert (bit.band (a, 1) == bit.cast (1))
 assert (bit.band (a, -1) == bit.cast (a))
 assert (bit.band (a, a) == bit.cast (a))
 assert (bit.bor (a, 0) == bit.cast (a))
 assert (bit.bor (a, 1) == bit.cast (a))
 assert (bit.bor (a, -1) == bit.cast (-1))
 assert (bit.bor (a, a) == bit.cast (a))
 assert (bit.bxor (a, 0) == bit.cast (a))
 assert (bit.bxor (a, 1) == bit.cast (a - 1))
 assert (bit.bxor (a, -1) == bit.cast (-a - 1))
 assert (bit.bxor (a, a) == bit.cast (0))
 assert (bit.bnot (a) == bit.cast (-1 - a))
 if nb < bit.bits then
  assert (bit.lshift (a, 1) == bit.cast (a + a))
  assert (bit.lshift (1, nb) == bit.cast (2 ^ nb))
 end
 assert (bit.rshift (a, 1) == math.floor (a / 2))
 if nb < bit.bits then
  assert (bit.rshift (a, nb) == bit.cast (0))
 end
 assert (bit.rshift (a, nb - 1) == bit.cast (1))
 assert (bit.arshift (-1, 1) == bit.cast (-1))
end
```

5.8 Use Heart Beat

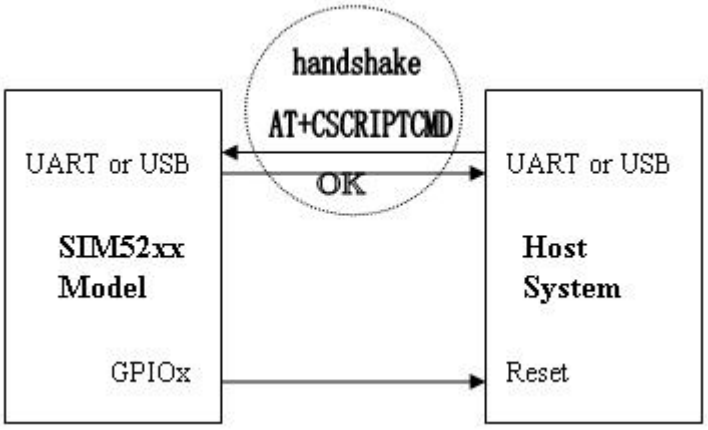
The following script gives an example of using heart heat. In this example, the external MCU should send AT+CSCRIPTCMD command to module every 1 minute, which generates event 31 to the running LUA script. Once the script receives event 31, it will reset the heart beat timer. If no

event 31 is received within 1 minute, the script will reset external MCU using reset_external_mcu() function.

```
function reset_external_mcu()
  gpio.setdrt(1);
  gpio.setv(0);
end;
printdir(1)
--const value
TIMER_EVENT = 28
OUT_CMD_EVENT = 31

HEART_BEAT_TIMEOUT = 1 * 60 * 1000;
vmstarttimer(0, HEART_BEAT_TIMEOUT);
while ( true ) do
  evt, evt_param1, evt_param2, evt_param3 = waitevt(15000);
  if (evt >= 0) then
    count = count + 1;
    print("(count=", count, ")", os.clock(), " event = ", evt, "\r\n");
    if ( evt == OUT_CMD_EVENT ) then
      vmstoptimer(0);
      sendtoport(evt_param1, "This is the confirm\r\n");
      vmstarttimer(0, HEART_BEAT_TIMEOUT);
    elseif ( (evt == TIMER_EVENT) and (evt_param1 == 0) ) then
      reset_external_mcu();
    end;
  end;
end;
```

Following is a general example of using heart beat design:



5.9 Use Event Functions

The following script gives an example of using event functions.

```
function register_evt_handler(evt, handler)
  evt_handler[evt] = handler;
end;
function default_event_handler(evt_id, evt_wparam, evt_lparam)
  return true;
end;
function event_handler_proc(evt, evt_param)
  local idx = nil;
  local handler = nil;
  if (evt <= 0) then
    return false;
  end;
  for idx, handler in pairs(evt_handler) do
    if (idx == evt) then
      if (handler) then
        return handler(evt, evt_param, 0);
      else
        break;
      end;
    end;
  end;
  return default_event_handler(evt, evt_param, 0);
end;
function timer_event_handler(evt_id, evt_wparam, evt_lparam)
  --handle timer event here, the evt_wparam is the timer id.
  return true;
end;
function sio_event_handler(evt_id, evt_wparam, evt_lparam)
  --handle sio event here, may call sio.recv() here
  return true;
end;
```

```
function lua_init()
  register_evt_handler(TIMER_EVENT,    timer_event_handler);
  register_evt_handler(SIO_RCVD_EVENT, sio_event_handler);
end;
function lua_main()
  printdir(1);
  lua_init();
  --start timer 0, which will generate a event every 1 second
  vmstarttimer(0,1000);
  --main loop of event handler
  while ( true ) do
    evt, evt_param = waitevt(15000);
    if (evt >= 0) then
      if (evt == USER_EVENT_EXIT) then
        print("exit main loop\r\n");
        break;
      else
        event_handler_proc(evt, evt_param);
      end;
    end;
  end;
end;
```

```
-------------------------------------------------
--const values for event id
GPIO_EVENT        = 0
UART_EVENT        = 1
KEYPAD_EVENT      = 2
USB_EVENT         = 3
AUDIO_EVENT       = 4
TIMER_EVENT       = 28
SIO_RCVD_EVENT    = 29
AT_CTL_EVENT      = 30
OUT_CMD_EVENT     = 31
LED_EVENT         = 32
--USER_EVENT_EXIT is extended by this script
USER_EVENT_EXIT   = 35
-------------------------------------------------
evt_handler = {}

lua_main();
```

6 QNA

6.1 Does LUA affect the sleep mode of module?

The module wouldn't enter sleep mode when LUA script is running except all LUA threads are calling vmsleep(),waitevt(), thread.signal_wait() or pending for calling other APIs with <timeout> parameter. For LUA has many events, if a new generated event is not cared by the script, the script

shouldn't do any handling, for example:

GOOD example:

```
while ( true ) do
    local evt, evt_p1, evt_p2, evt_p3, evt_clock = waitevt(9999999);
 if (evt == MYCARE_EVENT1) then
    elseif (evt == MYCARE_EVENT2) then
    end;
end;
```

BAD example:

```
while ( true ) do
local evt, evt_p1, evt_p2, evt_p3, evt_clock = waitevt(9999999);
do_many_operation_here();--bad script: the source code shouldn't do many handling when no event we care occurs, for SIMCom module has many events.
 if (evt == MYCARE_EVENT1) then
    elseif (evt == MYCARE_EVENT2) then
    end;
end;
```

Also the vmstarttimer() should only be called when necessary, or else it would affect the waitevt() function and cause the waitevt() to do be called, this may affect the sleep function of the module.

6.2 When the vmsetpri() should use high priority?

The vmsetpri() function supports three priorities. If the script uses high priority, it may affect the performing of general AT commands sent by external MCU. So when no external MCU is running and the script name is saved as "c:\autorun.lua" or "c:\autorun.out", the priority can be set to high. In all other cases, normal or low priority is recommended to be used.

6.3 How does LUA collect garbage memory?

LUA script is REQUIRED to collect garbage memory by itself, the collectgarbage() function can be used to collect garbage memory. The getcurmem() and getpeakmem() functions can be used to assist user when the memory should be colleted. For example, user can write scripts that once the getcurmem() reaches 1 MB, the scripts call the collectgarbage() function.

6.4 How to optimize the usage of memory?

Sometimes the script only uses memory about 1.3M bytes, and the script may crash for failing to allocate memory. This is because the LUA engine uses consecutive block of memory for variables with huge data, and the module cannot allocate so big consecutive block of memory. So in order to avoid this situation, local variables are recommended to use, and try not use big array or variable with huge data stored.

6.5 How to ensure downloading scripts safely?

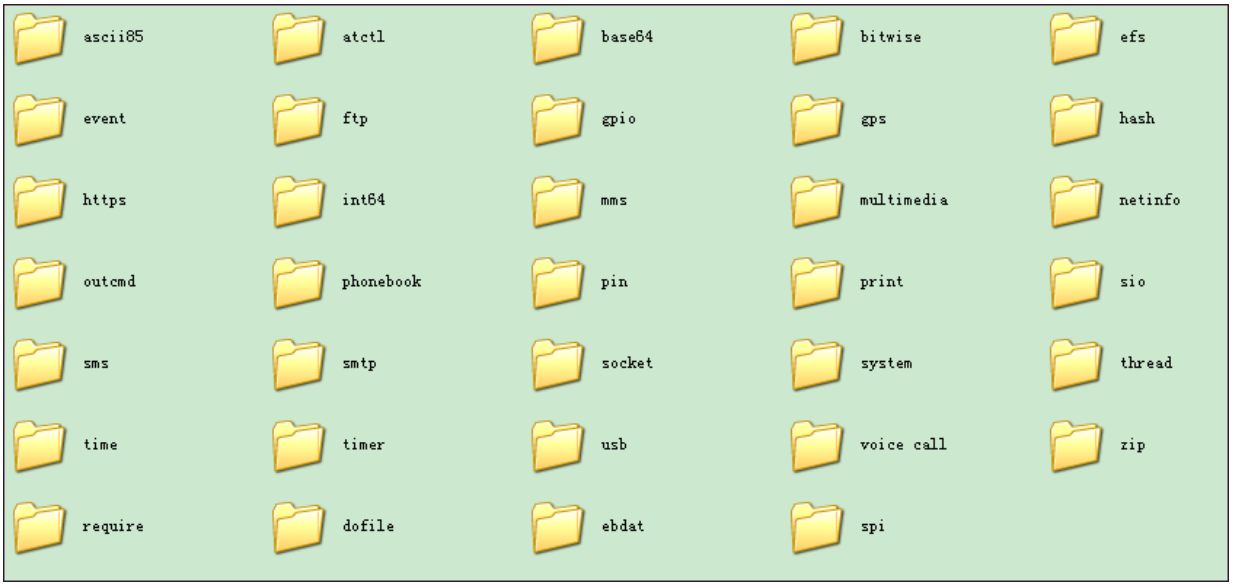
Sometimes when downloading multiple scripts to the module, if the power is cut off before

completing of download, the incomplete scripts may cause problem on next boot. So to ensure downloading scripts safely, the following steps are recommended:

- Execute AT+CSCRIPTAUTO=0
- Downloading scripts using provided tools
- If auto-run needed, execute AT+CSCRIPTAUTO=1

6.6 How to get the latest SIMCom LUA sample?

Please contact SIMCom to get the latest LUA samples, the samples may let customer easy to understand and use the LUA extended APIs.



Appendix

A Related Documents

SN	Document name	Remark
[1]	SIMCOM_SIM5360_ATC_EN_V1.0.doc	

B Terms and Abbreviations

Abbreviation	Description
API	Application Programming Interface
CPU	Central Processing Unit
LIB	Library
OS	Operating System
PDU	Protocol Data Unit
RAM	Random-Access Memory
ROM	Read-Only Memory
UMTS	Universal Mobile Telecommunications System
USIM	Universal Subscriber Identity Module
WCDMA	Wideband Code Division Multiple Access

Contact us:
Shanghai SIMCom Wireless Solutions Co.,Ltd.
Address: Building A, SIM Technology Building, No. 633, Jinzhong Road, Shanghai, P. R. China 200335
Tel: +86 21 3252 3300
Fax: +86 21 3252 3301
URL: www.sim.com/wm