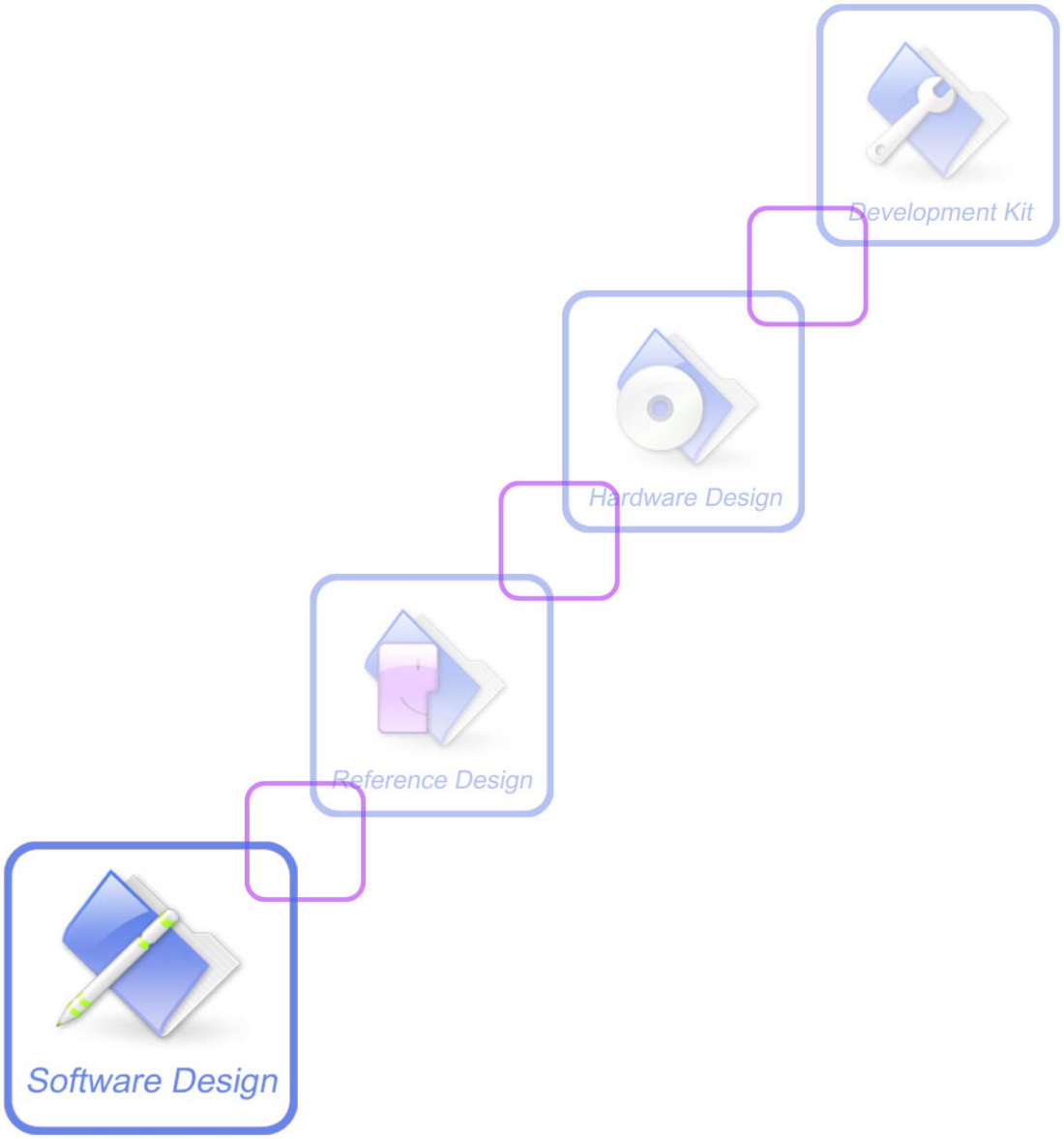
SIMCom
A company of SIM Tech

LUA Application Note



Document Title:	SIM52xx LUA Application Note
Version:	0.16
Date:	2011-12-20
Status:	Release
Document ID:	SIM52xx_LUA_Application_Note_V0.16

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

Version History

Version	Chapter	Comments
V0.01	New version	
V0.02	Add charger event Add AT+CSCRIPTCL Description	
V0.03	Add more description for heart beat part Add QNA part	
V0.04	Add AT+CSCRIPTCL	
V0.05	Add new APIs for SPI Add AT+CSCRIPTPASS description Add os.restartscript	
V0.06	Add os.restartscript	
V0.07	Add os.crushrestart	
V0.08	Modify gpio.settrigtype Add gpio.startflash Add gpio.stopflash Add image_comp_add_image Add videophone_set_source Add os.setportmode Add os.getportmode Add os.setflashmode Add os.getflashmode 4.1.2 Modified the method to transfer files to UE	
V0.09	Add thread library	
V0.10	Add camera_is_initialized	
V0.11	Modify maximum memory to 2.5M for SIM5320	
V0.12	Add ftp.simplist	
V0.13	Add sms library	
V0.14	Modify file:trunk Modify FEATURE descripton Modify QNA part, added collecting garbage description.	
V0.15	Add mms library Add debug library Add os.lsdir Add os.mkdir	

	Add os.rmdir Modify ftp.simpput Modify ftp.simpget Add thread.index Add thread.list Add chapter 4.2.3 for using debug library	
V0.16	Add thread.free Modify chapter 4.2.3.3, adding _debugger.cbpall()	

Contents

1. Introduction

1.2 Overview

SIM52XX LUA extension is a feature that allows customer applications control and drive the module internally and easily.

The SIM52XX 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.

SIM52XX LUA features:

- Light script language, easy to be extended.
- Support both procedure-oriented and object-oriented styles development.
- Script files are saved in EFS
- Support AT command operation in script.
- Support GPIO/IIC/SPI/ADC/PCM/Audio/GPS/UART operation in script.
- Support FTP operation instead of using AT commands.
- Available memory for script is limited to 3.5M (For SIM5320, it is 2.5M).
- Script can be started by extern MCU using AT command
- Script can run automatically when module powers up if the file name is “c:\autorun.lua” or “c:\autorun.out”.
- The priority of the script task can be adjusted to HIGH, NORMAL, LOW(default).
- High priority is not recommended to be used when external MCU is connected.
- Support event operation.
- Support multi-thread operation with separate thread library.
- Support timer operation(the maximize number of timer is 10).
- Support module entering sleep mode when LUA is idle(calling vmsleep, waitevt)

1.2 References

The present document is based on the following documents:

[1] SIMCOM_SIM5218_ATC_EN_V1.11.doc.

1.3 Terms and Abbreviations

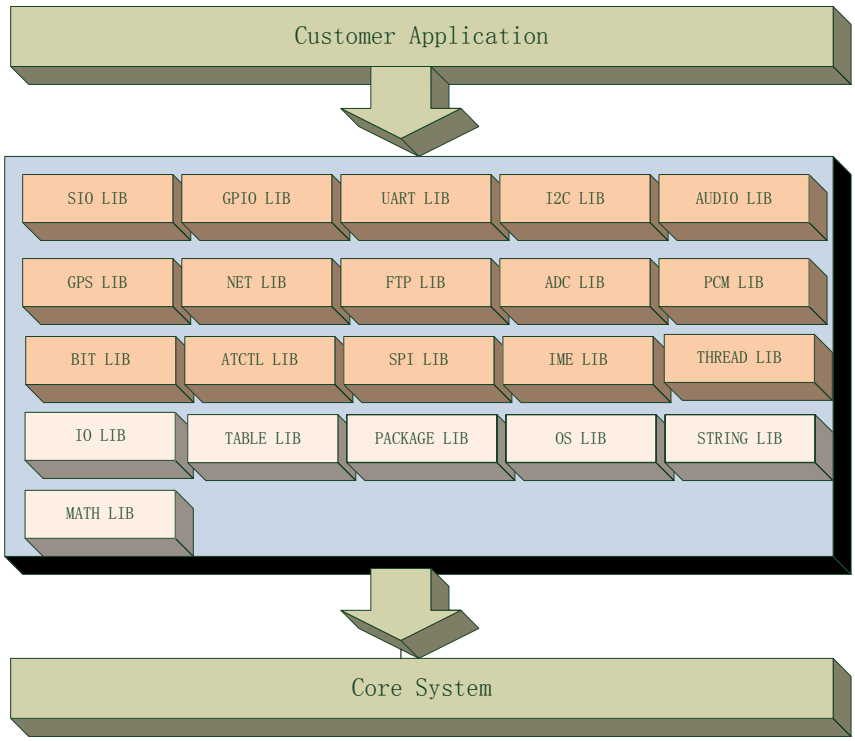
For the purposes of the present document, the following abbreviations apply:

- 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

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 SIM52XX Module. Before running the customer applications, user should put their scripts to the C:\ directory on the SIM52XX module.

The LUA script is executed in a task inside the SIM52XX 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 (For SIM5320, it is 2.5M). 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 SIM52XX 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 SIM52XX 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 PCM library allows the switch of the specified pins between command GPIO and PCM functions.
- 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.

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:

The following APIs are extended in the os library:

- os.filelength
- os.delfile

The following APIs are extended in the string library:

- string.concat
- string.equal
- string.startwith
- string.absfind

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

SIM52XX module provides LUA extension based on LUA5.1.4, which supports most original standard LUA APIs. SIM52XX module also provides APIs which can access system functions and operating SIM52XX hardware directly.

Table 2.1 lists SIM52XX extended LUA Libraries.

Library	Functions
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.
pcm	The pcm library allows LUA script to switch PCM and GPIO pins directly 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.

Table 2.1 SIM52XX 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: print to normal DIAG trace (default) 1: print to external AT interface
Return value	None

Example

```
--print to DIAG
printdir(0)
print(“this is printed to DIAG trace\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 1024 bytes.

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 reportte

Description

The function is used to send information to the TE. It is like the print function when printing to the external AT interface, and the total print string length cannot exceed 1024 bytes.

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

Example

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

3.1.4 sendtoport

Description

The function is used to send information to the TE.

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 data: the data to be sent
Return value	None

Example

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

3.1.5 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);

3.1.6 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")
```

3.1.7 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)
```

3.1.8 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 »)
```

3.1.9 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)
```

3.1.10 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. For timer event, it is the timer id. For the SCRIPTCMD event, it is the sio port of running the command. For other events, this parameter is reserved now. 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	0	0	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()
TIMER_EVENT	28	<timer_param1>	0	0	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()
CHARGER_EVENT	33	<charger_connected>	<charge_status>		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
<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
<charger_connected>
Charger status 0 – not connected 1 – connected
<charge_status>
Charge status 0 – not charging 1 – charging

Example

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

3.1.11 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)
```

3.1.12 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()
```

3.1.13 vmstarttimer

Description

The function is used to start a timer.

Prototype	boolean vmstarttimer (int timer_id, int timeout, int timer_type)
Parameters	timer_id: the ID of timer (0-19) timeout: the timeout value for the timer(ms) timer_type: the timer type 0: the timer will only generate 1 timer event 1: the timer will generate multiple timer event until the vmstoptimer function is called. (default)
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;
```

3.1.14 vmstoptimer

Description

The function is used to stop a timer.

Prototype	boolean vmstoptimer (int timer_id)
Parameters	timer_id: the ID of timer (0-9)
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);
```

3.1.15 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")
```

3.1.16 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 hookoff_init

Description

The function is used to setup hookoff ISR.

Prototype	int hookoff_init ()
Parameters	None
Return value	hookoff state: the hookoff state 0: hookoff key is on 1: hookoff key is off

Example

```
hookoff_state = hookoff_init()
```

3.1.18 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)
```

3.1.19 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)
```

3.1.20 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()
```

3.1.21 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)

3.1.22 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”)

3.1.23 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.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");
```

3.3.2 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")
```

3.3.3 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();
```

3.3.4 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)

3.3.5 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)

3.3.6 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()

3.3.7 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)
```

3.3.8 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()
```

3.3.9 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:\\");
```

3.3.10 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");
```

3.3.11 os.rmdir

Description

The function is used to delete a directory.

Prototype	boolean table 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");
```

3.4 SIO Library

3.4.1 sio.send

Description

The function is used to set send AT command to the 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");
```

3.4.2 sio.recv

Description

The function is used to set receive data from the 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();
```

3.4.3 sio.clear

Description

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

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

Example

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

3.4.4 sio.exclrpt

Description

The function is used to force the unsolidated result to be sent to the virtual serial port on the module only.

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);
```

3.5 GPIO Library

3.5.1 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");
```

3.5.2 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");
```

3.5.3 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");
```

3.5.4 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");
```

3.5.5 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);
```

3.5.5 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);
```

3.6 UART Library

3.6.1 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");
```

3.6.2 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");
```

3.6.3 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");
```

3.6.4 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");
```

3.6.5 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");
```

3.6.6 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");
```

3.7 IIC Library

3.7.1 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");
```

3.7.2 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)
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
Return value	the result of setting: 1: successful 0: failed

Example

```
rst= i2c.write_i2c_dev(15,15,4660,2);
print(rst,"\r\n");
```

3.8 ADC Library

3.8.1 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");
```

3.9 PCM Library

3.9.1 pcm.switch_gpio_and_pcm

Description

The function is used to switch GPIO and PCM mode for the specified GPIO. It has the same function as AT+CPCM command.

Prototype	int pcm.switch_gpio_and_pcm (int mode)
Parameters	mode: the mode the GPIO 0: normal GPIO (default)

	1: PCM mode
Return value	the result of setting: 1: successful 0: failed

Example

```
rst= pcm.switch_gpio_and_pcm(0);
print(rst,"\r\n");
```

3.9.2 pcm.get_cur_pcm_md

Description

The function is used to get the specified GPIO mode (normal GPIO or PCM). It has the same function as AT+CPCM command.

Prototype	int pcm.get_cur_pcm_md ((
Parameters	None
Return value	the mode of the PCM: 0: normal GPIO 1: PCM

Example

```
rst= pcm.get_cur_pcm_md();
print(rst,"\r\n");
```

3.10 AUDIO Library

3.10.1 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");
```

3.10.2 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");
```

3.10.3 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");
```

3.10.4 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");
```

3.10.5 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");
```

3.10.6 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");
```

3.10.7 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");
```

3.10.8 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");
```

3.10.9 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");
```

3.10.10 audio.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");
```

3.10.11 audio.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");
```

3.10.12 audio.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");
```

3.10.13 audio.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");
```

3.10.14 audio.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");
```

3.10.15 audio.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");
```

3.10.16 audio.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");
```

3.10.17 audio.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");
```

3.10.18 audio.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");
```

3.10.19 audio.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");
```

3.10.20 audio.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");
```

3.11 GPS Library

Note: GPS library is supported on SIM5211/SIM5218/SIM5220.

3.11.1 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");
```

3.11.2 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");
```

3.11.3 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");
```

3.11.4 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

```
rst= gps.gpssetmode(1);
print(rst,"\r\n");
```

3.11.5 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");
```

3.11.6 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");
```

3.11.7 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");
```

3.11.8 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");
```

3.11.9 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");
```

3.12 NET Library

3.12.1 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 net.creg ()
Parameters	None
Return value	the result of the cs-domain register result.

Example

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

3.12.2 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 net.cgreg ()
Parameters	None
Return value	the result of the ps-domain register result.

Example

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

3.12.3 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");
```

3.12.4 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");
```

3.13 FTP Library

3.13.1 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 = "e-device.net";
port = 21;
name = "myaccount";
pass = "password";
remote_file = "/up_normal.jpg";
uplocal_file = "c:\\Picture\\normal.jpg";
passive = 0;
rst = ftp.simpput(server, port, name, pass, remote_file, uplocal_file, passive);
print(rst,"\r\n");
```

3.13.2 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 = "e-device.net";
port = 21;
name = "myaccount";
pass = "password";
remote_file = "/up_normal.jpg";
downlocal_file = "c:\\Video\\down_normal.jpg";
passive = 0;
rst = ftp.simpget(server, port, name, pass, remote_file, downlocal_file, passive);
print(rst,"\r\n");
```

3.13.3 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 = "e-device.net";
port = 21;
name = "myaccount";
pass = "password";
remote_path = "/MyDir/";
passive = 0;
rst, list_data= ftp.simplist(server, port, name, pass, remote_path, passive);
print(rst,"\r\n");
```

3.14 STRING Library

3.14.1 string.concat

Description

The function is used to connect two strings.

Prototype	string string.concat (string str1, string str2)

Parameters	str1: the string to connect. str2: the string to connect.
Return value	the result of str1 + str2.

Example

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

3.14.2 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");
```

3.14.3 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");
```

3.14.4 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");
```

3.14.5 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);
```

3.14.6 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);
```

3.14.7 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);
```

3.14.8 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);
```

3.14.9 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);
```

3.14.10 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);
```

3.14.11 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);
```

3.14.12 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);
```

3.14.13 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);
```

3.14.14 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);
```

3.15 BIT Library

3.15.1 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)
```

3.15.2 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))
```

3.15.3 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))
```

3.15.4 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))
```

3.15.5 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))
```

3.15.6 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))
```

3.15.7 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))
```

3.15.8 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))
```

3.16 ATCTL Library

3.16.1 atctl.setport

Description

The function is used to set the serial port for ATCTL purpose use. The port used by atctl.setport shouldn't be the same as set using AT+CATR, or else which may not work correctly.

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 -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)
```

3.16.2 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)
```

3.16.3 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")
```

3.16.4 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 ()
```

3.16.5 atctl.setowner

Description

The function is used to set the thread which owns the ATCTL module.

Prototype	void atctl.setowner()
Parameters	None
Return value	None

Example

```
atctl.clear ()
```

3.17 SPI Library

3.17.1 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)
```

3.17.2 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)
```

3.17.3 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)
```

3.17.4 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)
```

3.17.5 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();
```

3.17.6 spi.config_lcd_device

Description

The function is used to configure the device of SPI.

Prototype	void spi.config_device (int reset_1st_value, int reset_2nd_value, int write_data_1st_value, int write_data_2nd_value)
Parameters	reset_1st_value: the first value for reset 0: low 1: high reset_2nd_value: the second value for reset

	0: low 1: high write_data_1st_value: the first value to set before writing data to LCD SPI device 0: low 1: high write_data_2nd_value: the second value to set after writing data to LCD SPI device 0: low 1: high
Return value	None

Example

```
spi.config_lcd_device(0, 1, 1, 0);
```

3.17.7 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);
```

3.17.8 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);
```

3.17.9 spi.write_lcd_cmd

Description

The function is used to write command to the LCD SPI device.

Prototype	boolean spi.write_lcd_cmd (int data)
Parameters	data: data to write to the SPI device
Return value	The result of the writing TRUE: successful FALSE: failed

Example

```
spi.write_lcd_cmd(240);
```

3.17.10 spi.write_lcd_data

Description

The function is used to write data to the LCD SPI device.

Prototype	boolean spi.write_lcd_data (int data)
Parameters	data: data to write to the SPI device
Return value	The result of the writing TRUE: successful FALSE: failed

Example

spi.write_lcd_data(240);

3.17.11 spi.write_lcd_hzk12_1_line

Description

The function is used to write chinese string to the LCD SPI device.

Prototype	boolean spi.write_lcd_hzk12_1_line (string ucs2_str, int is_little_endian, int line_no, int start_char_pos, int end_char_pos, int start_pixel_x, int end_pixel_x, int underline, int underline_or_value, int reverse_color, int focus_char_begin_pos, int focus_char_end_pos, int seperator, int blank_width, int fixed_fnt_width)
Parameters	ucs2_str: the string to be written to the device is_little_endian: the UCS2 mode 1: little endian 0: big endian line_no: the line number of the string to display 1: the first line to display 2: the second line to display start_char_pos: the position of the first chinese code to display end_char_pos: the position of the last chinese code to display 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 focus_char_begin_pos: the begin position of the focus char focus_char_end_pos: the end position of the focus char separator: the separator value. blank_width: the width for the blank width fixed_fnt_width: the fixed chinese font width.
Return value	The new x-pixel position after writing.

Example

spi.write_lcd_hzk12_1_line(text, little_endian, 1, nil, nil, cur_pixel_x, end_pixel_x, nil, nil, rev_color, focus_char_begin_pos, focus_char_end_pos, seperator, blank_width, fixed_char_width)

3.17.12 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)

3.17.13 spi.write_lcd_bytes

Description

The function is used to write multiple bytes to the LCD SPI device.

Prototype	int spi.write_1_byte_multi_times (string str, int start_pixel_x, int end_pixel_x, int underline, int underline_or_value, int reverse_color)
Parameters	str: 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_bytes(row_data, pixel_x, end_pixel_x, under_line_this_row, phone_dev.underline_value, reverse_color)

3.17.14 spi.get_hzk12_string_width

Description

The function is used to get the total width of a ucs2 string to display on LCD.

Prototype	int spi.get_hzk12_string_width (string str, int is_little_endian, int start_char_pos, int end_char_pos, int seperator, int blank_width, int fixed_fnt_width)
Parameters	str: data to write to the SPI device is_little_endian: the encoding type of ucs2 string 0: big endian 1: little endian start_char_pos: the first char position to calculate in the string end_char_pos: the last char position to calculate in the string separator: the separator value blank_width: the width for blank character fixed_fnt_width: the fixed font width
Return value	The total length of the string

Example

```
local total_width = spi.get_hzk12_string_width(text, little_endian, nil, nil, seperator, nil, nil)
```

3.17.15 spi.lcd_backlight

Description

The function is used to turn on or off the background light of the LCD.

Prototype	void spi.lcd_backlight (int level)
Parameters	level: the level of the light
Return value	None

Example

spi.lcd_backlight(1)

3.17.16 spi.freehand_light

Description

The function is used to turn on or off the handfree light.

Prototype	void spi.handfree_light (int level)
Parameters	level: the level of the light
Return value	None

Example

spi.handfree_light(1)

3.18 IME Library

3.18.1 ime.resetbuf

Description

The function is used to reset the buffer for IME.

Prototype	void ime.resetbuf()
Parameters	None
Return value	None

Example

ime.resetbuf()

3.18.2 ime.handle_key

Description

The function is used to handle the key pressed event for IME.

Prototype	boolean ime.handle_key(int key_code)
Parameters	key_code: the code of the key pressed

Return value	boolean: the result of the handling TRUE: successful FALSE: failed

Example

```
ime.handle_key(key_code)
```

3.18.3 ime.get_pinyin_group

Description

The function is used to get the possible pinyin combination group for IME.

Prototype	table ime.get_pinyin_group()
Parameters	None
Return value	The possible pinyin combination group

Example

```
pinyin_group = ime.get_pinyin_group()
```

3.18.4 ime.get_bihua_group

Description

The function is used to get the possible stroke combination group for IME.

Prototype	table ime.get_bihua_group()
Parameters	None
Return value	The possible stroke combination group

Example

```
bihua_group = ime.get_bihua_group()
```

3.18.5 ime.get_word_page

Description

The function is used to get the candidate words for IME.

Prototype	table ime.get_word_page()
Parameters	None
Return value	The candidate words

Example

```
word_page = ime.get_word_page()
```

3.18.5 ime.shift_spelling

Description

The function is used to shift spelling option for IME.

Prototype	boolean ime.shift_spelling(int index)
Parameters	index: the index of the spelling selected
Return value	The result of the shifting TRUE: successful FALSE: failed

Example

```
ime.shift_spelling(1)
```

3.18.6 ime.select_word

Description

The function is used to select a ucs2 word for IME.

Prototype	boolean ime.select_word(string word)
Parameters	word: the ucs2 code selected
Return value	The result of the selecting TRUE: successful FALSE: failed

Example

```
ime.select_word(word)
```

3.18.7 ime.setinputmode

Description

The function is used to set the input mode for IME.

Prototype	void ime.setinputmode(int mode)
Parameters	mode: the mode the input method 1: pinyin 2: stroke
Return value	None

Example

```
ime.setinputmode(1)
```

3.18.8 ime.ucs2_to_hzk12_data

Description

The function is used to get the hzk12 font information for a ucs2 code.

Prototype	int int table table ime.ucs2_to_hzk12_data(int ucs2_code)
Parameters	ucs2_code: the UCS2 code
Return value	CharWidth: the width of the ucs2_code to display on LCD device CharHeight: the height of the ucs2_code to display on LCD device Fnt_info_r1: the first row of the word to display on LCD device Fnt_info_r2: the second row of the word to display on LCD device

Example

```
Local fnt_width, fnt_height, fnt_data_r1, fnt_data_r2 = ime.ucs2_to_hzk12_data (ucs2_code);
```

3.19 THREAD Library

3.19.1 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)
```

3.19.2 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);
```

3.19.3 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);
```

3.19.4 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);

3.19.5 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();

3.19.6 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);
```

3.19.7 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 four 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. For timer event, it is the timer id. For the SCRIPTCMD event, it is the sio port of running the command. For other events, this parameter is reserved now. 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 ");
```

3.19.8 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 four return values for thread.peekevt(…). 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. For timer event, it is the timer id. For the SCRIPTCMD event, it is the sio port of running the command. For other events, this parameter is reserved now. 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 = thread.peekevt(10000)
    print(event_id, " ", event_param, "\r\n ");
```

3.19.9 thread.setpri

Description

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

Prototype	void thread.setpri (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
thread.setpri(LOW_PRIORITY);
thread.setpri(MEDIUM_PRIORITY);
thread.setpri(HIGH_PRIORITY);
```

3.19.10 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")
```

3.19.11 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)
```

3.19.12 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 9.
Return value	None

Example

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

3.19.13 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 9.
Return value	None

Example

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

3.19.14 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

Prototype	boolean thread.setevt(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);
```

3.19.15 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();
```

3.19.16 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);
```

3.19.17 thread.free

Description

The function is used to free the thread which is done or stoped by thread.stop().

Prototype	boolean thread.free(thread t_id)
Parameters	thread tid: the thread to be freed
Return value	true: successful false: failed to free

Example

```
result= thread.free(tid);
```

3.20 SMS Library

3.20.1 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();
```

3.20.2 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);
```

3.20.3 sms.get_cscs

Description

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

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

Example

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

3.20.4 sms.set_cscs

Description

The function is used to set the CSCS setting.

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

Example

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

3.20.5 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.

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

Example

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

3.20.6 sms.convert_chset

Description

The function is used to convert characters 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);
```

3.20.7 sms.send_txtmsg

Description

The function is send text message.

Prototype	boolean, int sms.send_txtmsg(

	string dest_addr, string toda, string sms_data, int cscs, int msg_ref, int total_sm, int seq_num, int fo, int dcs, int pid string valid_period)
Parameters	string dest_addr : destination address string toda : type of destination address string sms_data : the SMS data to be sent int cscs : the character set of SMS data int msg_ref : the message reference int total_sm : total number of long SMS int seq_num : the sequence number of long SMS int fo : first object of SMS int dcs : SMS data coding scheme int pid : TP-Protocol-Identifier string valid_period : TP-Validity-Period. See GSM 03.40
Return value	boolean: the result of sending int: If successful, it is the message reference, or else, it is the error code.

Example

```
dest_addr = "15021309668";
toda = "";
```

```
sms_data = "9650523665F695F45230FF0C5F5550CF505C6B62";
cscs = CSCS_UCS2;
msg_ref = sms.get_next_msg_ref();
total_sm = 2;
seq_num = 1;
fo = 17;
dcs = 8;
pid = 0;
vp = "";
local suc, msg_ref_or_err_cause;
print("new msg_ref=", msg_ref, "\r\n");
suc, msg_ref_or_err_cause = sms.send_txtmsg(dest_addr, toda, sms_data, cscs, msg_ref, total_sm, seq_num, fo, dcs, pid, vp);
print("suc=", suc, ", msg_ref_or_err_cause=", msg_ref_or_err_cause, "\r\n");
total_sm = 2;
seq_num = 2;
sms_data = "5F5550CF8D8565F6";
suc, msg_ref_or_err_cause = sms.send_txtmsg(dest_addr, toda, sms_data, cscs, msg_ref, total_sm, seq_num, fo, dcs, pid, vp);
print("suc=", suc, ", msg_ref_or_err_cause=", msg_ref_or_err_cause, "\r\n");
```

3.20.8 sms.write_txtmsg

Description

The function is write text message.

Prototype	boolean, int sms.write_txtmsg(string dest_addr, string toda, string sms_data, int cscs, int msg_ref,

	int total_sm, int seq_num, int fo, int dcs, int pid string valid_period)
Parameters	string dest_addr : destination address string toda : type of destination address string sms_data : the SMS data to be sent int cscs : the character set of SMS data int msg_ref : the message reference int total_sm : total number of long SMS int seq_num : the sequence number of long SMS int fo : first object of SMS int dcs : SMS data coding scheme int pid : TP-Protocol-Identifier string valid_period : TP-Validity-Period. See GSM 03.40
Return value	boolean: the result of writing int: If successful, it is the message reference, or else, it is the error code.

Example

```
dest_addr = "15021309668";
toda = "";
sms_data = "9650523665F695F45230FF0C5F5550CF505C6B62";
cscs = CSCS_UCS2;
msg_ref = sms.get_next_msg_ref();
total_sm = 2;
seq_num = 1;
```

```
fo = 17;

dcs = 8;

pid = 0;

vp = "";

local suc, msg_ref_or_err_cause;

print("new msg_ref=", msg_ref, "\r\n");

suc, msg_ref_or_err_cause = sms.write_txtmsg(dest_addr, toda, sms_data, cscs, msg_ref, total_sm, seq_num, fo, dcs, pid, vp);

print("suc=", suc, ", msg_ref_or_err_cause=", msg_ref_or_err_cause, "\r\n");

total_sm = 2;

seq_num = 2;

sms_data = "5F5550CF8D8565F6";

suc, msg_ref_or_err_cause = sms.write_txtmsg(dest_addr, toda, sms_data, cscs, msg_ref, total_sm, seq_num, fo, dcs, pid, vp);

print("suc=", suc, ", msg_ref_or_err_cause=", msg_ref_or_err_cause, "\r\n");
```

3.21 MMS Library

3.21.1 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();
```

3.21.2 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();
```

3.21.3 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");
```

3.21.4 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();

3.21.5 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);

3.21.6 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();

3.21.7 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);
```

3.21.8 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");
```

3.21.9 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);
```

3.21.10 mms.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");
```

3.21.11 mms.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”);

3.21.12 mms.add_receipt

Description

The function is used to add an 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);

3.21.13 mms.delete_receipt

Description

The function is used to delete an 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);
```

3.21.14 mms.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);
```

3.21.15 mms.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”);
```

3.21.16 mms.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);
```

3.21.17 mms.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);
```

3.21.18 mms.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 ();
```

3.21.19 mms.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);
```

3.21.20 mms.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");
```

3.21.21 mms.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);
```

3.22 DEBUG Library

3.21.1 debug.debug

Description

The function is used to acquire MMS module for LUA.

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

]]
```

3.21.2 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);
```

3.21.3 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();
```

3.21.4 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”);
```

3.21.5 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);
```

3.21.6 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);
```

3.21.7 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();
```

3.21.8 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);
```

3.21.9 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);
```

3.21.10 debug.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);
```

3.21.11 debug.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);
```

3.21.12 debug.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);
```

3.21.13 debug.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);
```

3.21.13 debug.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);
```

3.21.14 debug.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();
```

3.21.15 debug.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();
```

3.21.15 debug.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);

3.21.16 debug.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”);

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:

- Write the LUA script
- Download the LUA script into the EFS of the module
- 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")
```

After entering the text, save it as "helloworld.lua", and then close the notepad.exe program.

4.1.2 Download LUA script

diag_lua.rar

Use PC tool to transfer the "helloworld.lua" file to the C:\ directory on the 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:

4.1.3 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

4.1.4 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

4.1.5 Stop the active 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.

4.1.6 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.

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:

```
1   printdir(1)
2   print("HELLO WORLD!\r\n")
3   condition = 1
4   if (condition == 1)
5     print("conditon equals 1\r\n");
6   elseif (condition == 2) then
7     print("conditon equals 2\r\n");
8   else
9     print("other value\r\n");
10  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.

4.2.3 Use debug library and AT+DBC command

4.2.3.1 Standard debug library and AT+DBC command

SIM52XX module supports standard LUA debug library. When calling debug.debug() function in LUA script, the AT+DBC command can be used to input debug command. For 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

]]

4.2.3.2 Using step/over/finish/breakpoints method in debugging

SIM Com provides a debugging assist script file to help customer to debug the application more easily. The debugger.out script file provides step/over/finish/breakpoints debugging methods. The debugging steps are in the following:

1. Put the debugger.out file to C:\ directory on the module


Debugger.out

2. Put customer application scripts onto the C:\ directory on the module

3. Run AT+CSCRIPTSTART=”customer_app_name.lua”, 2 to run customer application script file

4. During the debugging, customer can set breakpoints and do step/over/finish debugging.

Following is a sample for debugging a script:


Thread. lua

The debugging result:


thread_debugging.
txt

4.2.3.3 Methods provided by debugger.out

_debugger.abp([thread_id,]filename, lineno): Add a breakpoint in filename:lineno for thead_id, if the thread_id is not set, this breakpoint will affect all threads.

_debugger.lbp(): List all breakpoints.

_debugger.cbp(bp_index): Remove a breakpoint indexed by bp_index.

_debugger.cbpall(): Remove all breakpoints。

_debugger.step([thread_id]): Perform a step operation. The thread_id is the thread to perform step operation. Without this parameter, the thread will be the current broken thread.

_debugger.over([thread_id]): Perform a over operation. The thread_id is the thread to perform over operation. Without this parameter, the thread will be the current broken thread.

_debugger.finish([thread_id]): Perform a finish operation which will run until the current function return. The thread_id is the thread to perform finish operation. Without this parameter, the thread will be the current broken thread.

_debugger.tt(): List all running threads.

_debugger.tb([thread_id]): Display the traceback information for thread_id. Without the parameter thread_id, all thread traceback information will be displayed.

_debugger.f([thread_id]): Display the frame information with local variables for thread_id. Without the parameter thread_id, the current broken thread stack information will be displayed.

_debugger.tv(var): Display a global variable.

4.3 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 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”:

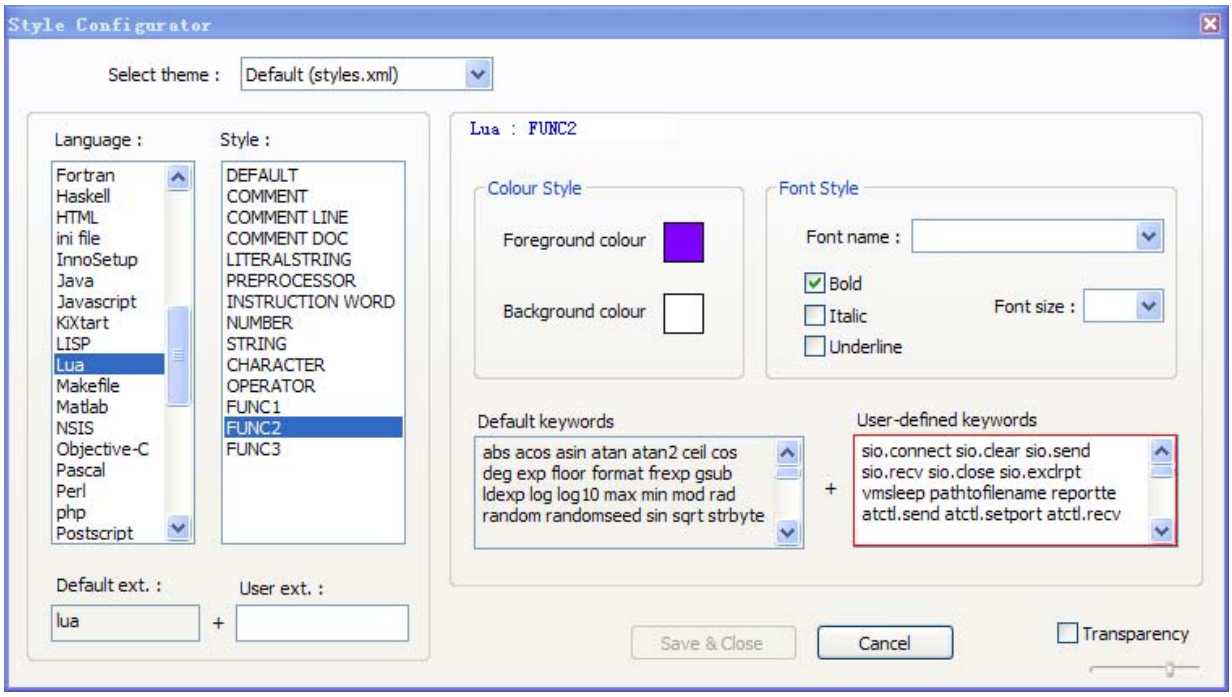
AT+CSCRIPTPASS=””,”12345678”

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 SIM52XX LUA script. User can set the SIM52XX APIs by selecting the “Settings|Style configurator…” menu, and input the extended API names in the “Style Configurator” dialog like the following:



5. LUA Script Samples

This section shows some samples which are aimed to assist customer to be familiar with SIM52XX 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();
```

```
32  vmsleep(1000);
33
34  ------------code start----------------
35  start = gps.gpsstart(2)
36  if(start == true) then
37    print("-------GPS coldstart success!--------\r\n")
38  end;
39  count = 0;
40  while (count < test_coldstart_loop_count) do
41    count = count+1;
42    print("-----------run count=",count,"-------\r\n");
43    fix = gps.gpsinfo();
44    print(fix,"\r\n","\r\n");
45    vmsleep(1000);
46  end;
47  close = gps.gpsclose();
48  if(close == true) then
49    print("-------------GPS colse success!-------------\r\n");
50  end;
51
52  print("\r\n------waiting 10s for hotstart------\r\n\r\n");
53  vmsleep(10000);
54
55  ------------hot start----------------
56  start = gps.gpsstart(1);
57  if(start == true) then
58    print("--------GPS hotstart success!-------------\r\n")
59  end;
60  count = 0;
61  while (count < test_hotstart_loop_count) do
62    count = count+1;
63    print("-----------run count=",count,"-------\r\n");
64    fix = gps.gpsinfo()
65    print(fix,"\r\n","\r\n")
66    vmsleep(1000)
67  end;
68  close = gps.gpsclose()
69  if(close == true) then
70    print("-------------GPS colse success!-------------\r\n");
71  end;
72  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

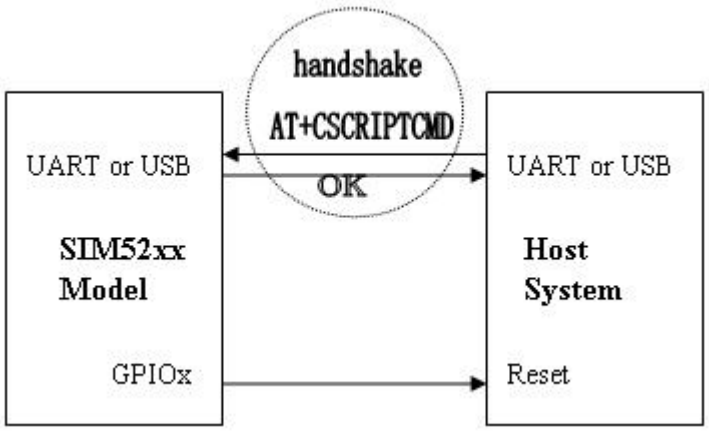
The following script gives an example of using heart heat. In this example, the external MCU should

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.

```lua
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 that LUA script is calling

vmsleep() or waitevt() functions. If the script runs threads, all the threads must also be calling thread.sleep() or thread.waitevt().

6.1 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.

6.2 When will LUA collect memory that is not used further?

When the memory used by LUA reaches the limit that can be allocated for LUA task, the LUA task will collect the garbage memory automatically. If user needs to collect memory by LUA scripts, the collectgarbage() function can be used to force to collect garbage memory.

Contact us

Shanghai SIMCom Wireless Solutions Ltd.
Add: Building A, SIM Technology Building, No.633, Jinzhong Road, Changning District 200335
Tel: +86 21 3252 3300
Fax: +86 21 3252 3301
URL: http:/www.sim.com/wm/