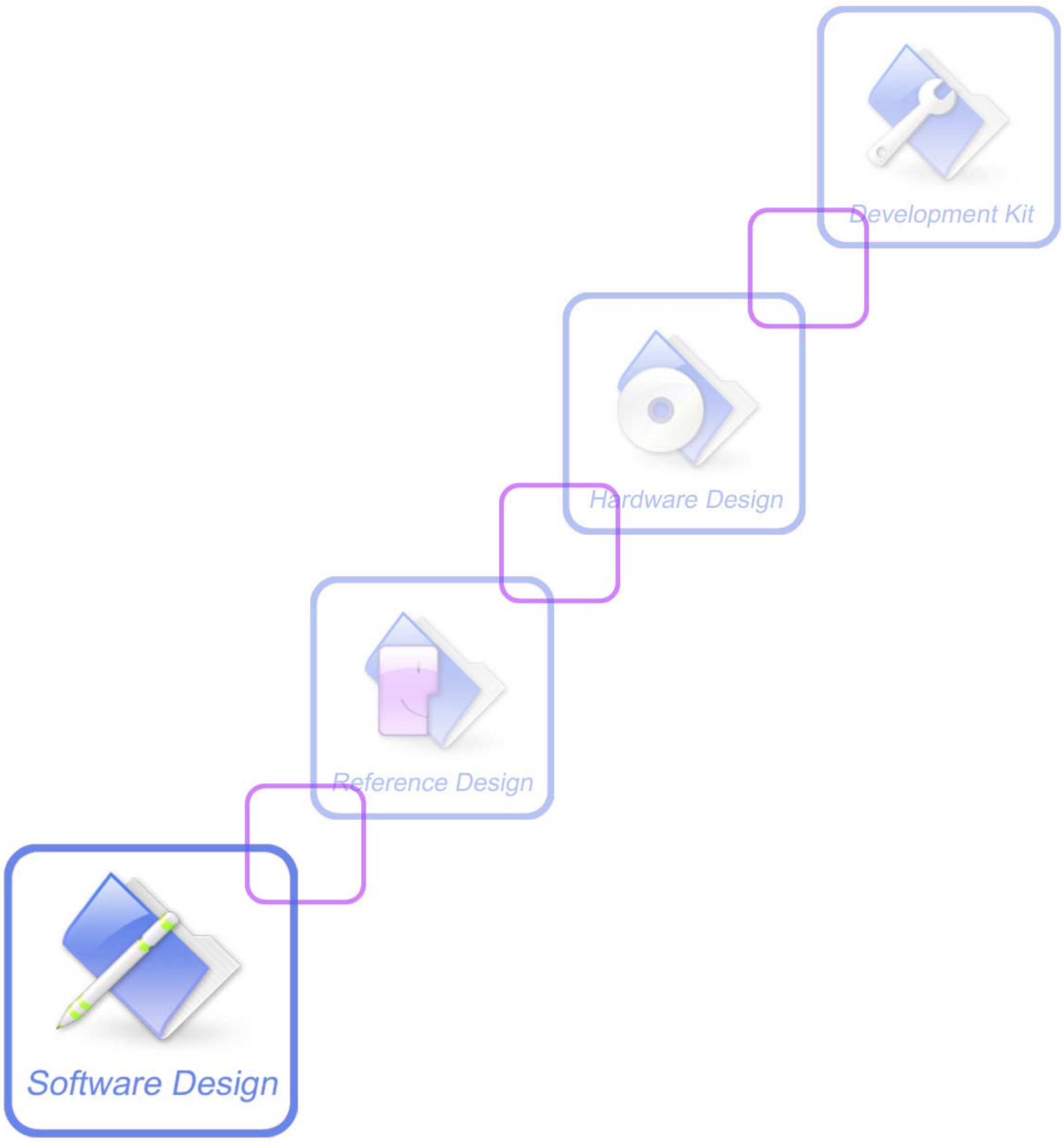
A company of SIM Tech

LUA Application Note



Document Title:	SIM52xx LUA Application Note
Version:	0.04
Date:	2010-04-29
Status:	Release
Document ID:	SIM52xx_LUA_Application_Note_V0.04

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	

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.
- 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 timer operation(the maximize number of timer is 10).
- Support XMODEM method of transferring script file between extern MCU and module.

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

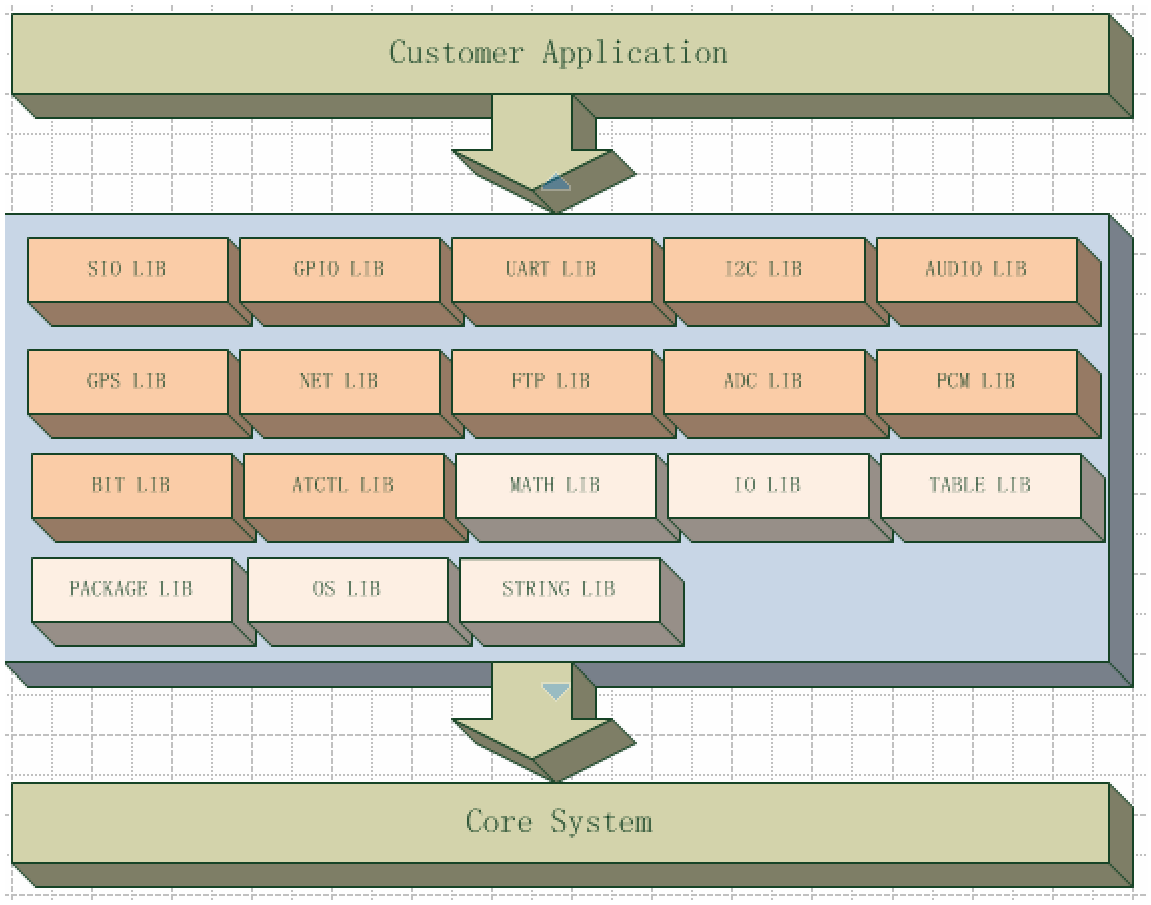
For the purposes of the present document, the following abbreviations apply:

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. 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.

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: ccur

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 library is not supported on the SIM52XX LUA engine:

- debug

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.

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	ccurs 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	ccurs ccurs 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 ccurs, -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 and parameters description

event	event_id	event_param1	event_param2	event_param3
GPIO_EVENT	0	0	0	0

UART_EVENT	1	0	0	0
KEYPAD_EVENT	2	0	0	0
USB_EVENT	3	0	0	0
AUDIO_EVENT	4	0	0	0
TIMER_EVENT	28	<timer_param1>	0	0
SIO_RCVD_EVENT	29	0	0	0
ATCTL_EVENT	30	0	0	0
OUT_CMD_EVENT	31	0	0	0
LED_EVENT	32	<led_param1>	<led_param2>	0
CHARGER_EVENT	33	<charger_connected>	<charge_status>	

Parameter defined values

<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 (0-9) 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 size of the peak memory used for the LUA script.

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

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

3.2 IO Library

3.2.1 file:trunc

Description

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

Prototype	void file:trunc ()
Parameters	None
Return value	None

Example

```
file = io.open("c:\\test1.txt","w")
assert(file)
file:trunc()
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")
```

3.4 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 detect, int polarity[,int save])
Parameters	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(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 gpio (2, 3, 5) 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 gpio (2, 3, 5)

	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 gpio (0, 1, 2, 3, 4, 5)
Return value	the value of the GPIO: 0: low 1: high

Example

```
level = gpio.getdrt(5);
print(level,"\r\n");
```

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)
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)
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)
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)
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");
```

3.14 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");
```

3.15 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 ()

4. LUA Script operations

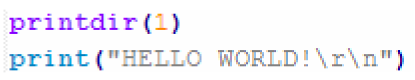
4.1 Executing a LUA script

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.

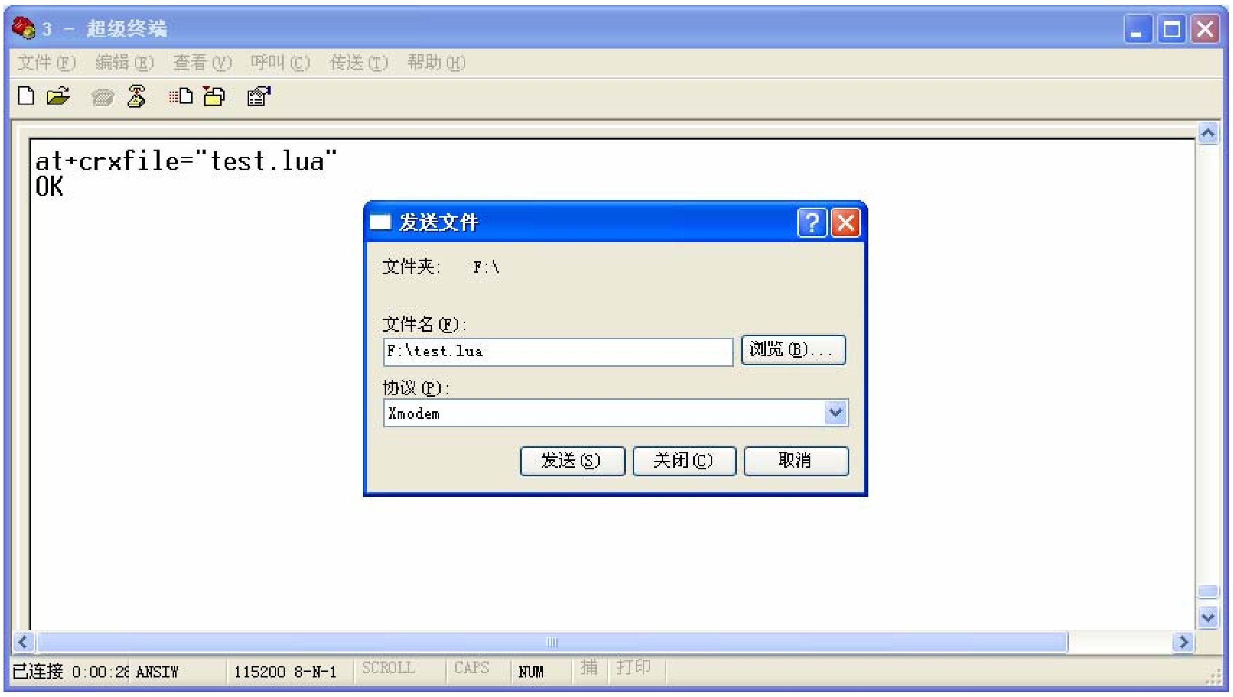


```
printdir(1)
print("HELLO WORLD!\r\n")
```

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

4.1.2 Download LUA script

Use XMODEM to transfer the "helloworld.lua" file to the C:\ directory on the module. For the XMODEM related AT command, please refer to AT+CRXFILE in [1].



```
at+crxfile="test.lua"
OK
```

3 - 超级终端
文件(F) 编辑(E) 查看(V) 呼叫(C) 传送(T) 帮助(H)
发送文件
文件夹: F:\
文件名(F): F:\test.lua
浏览(B)...
协议(P): Xmodem
发送(S) 关闭(C) 取消
已连接 0:00:28 ANSIW 115200 8-N-1 SCROLL CAPS NUM 捕 打印

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

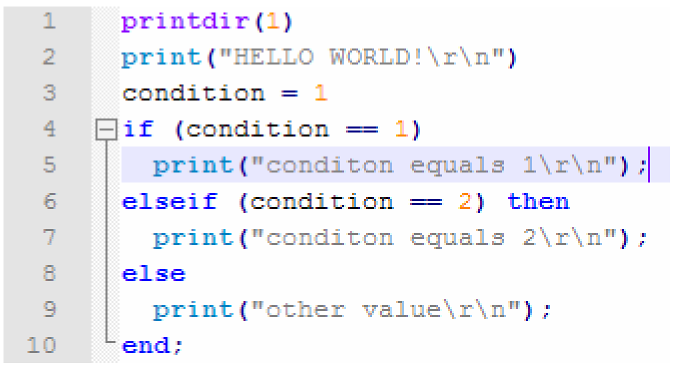
AT+CSCRIPTSTART command provides the second parameter to support reporting error of LUA script to TE. When user is developing the script, it is very useful, and may report the grammar or running error immediately and help user to correct it. Following is an example of the error report:

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.

4.3 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”:

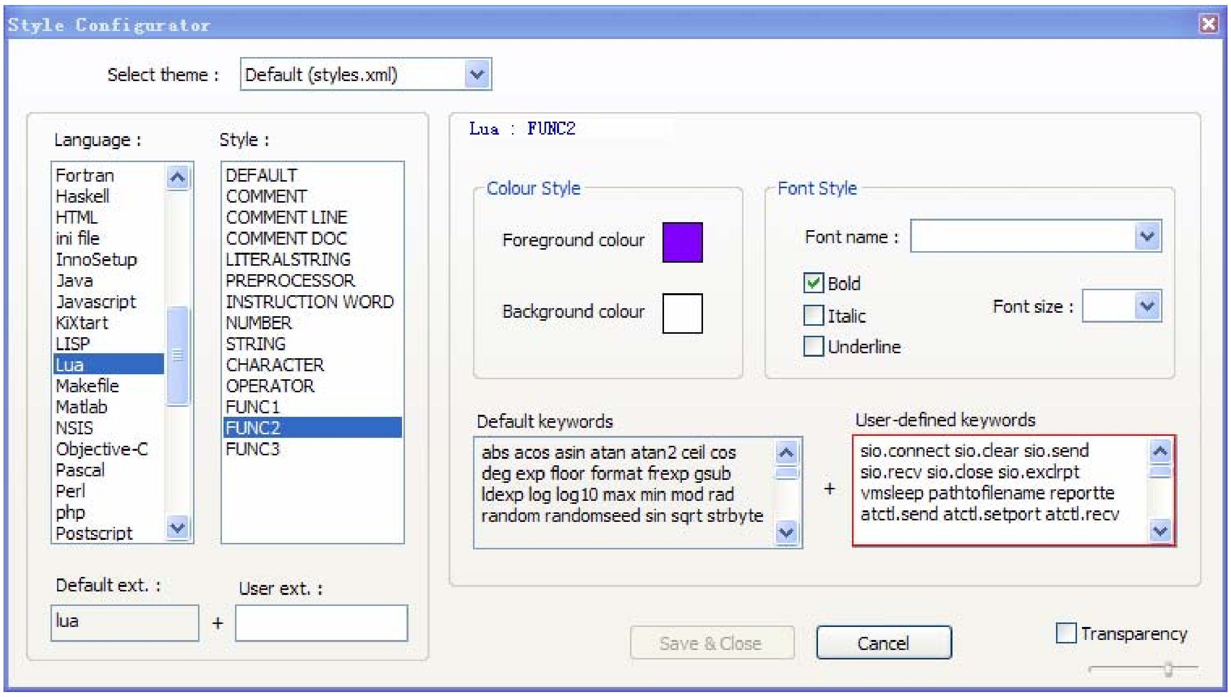
AT+CSCRIPTCL=”test.lua”

Or

AT+CSCRIPTCL=”test.lua”,”test.out”

4.4 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:


Style Configurator

Select theme : Default (styles.xml)

Language : Fortran, Haskell, HTML, ini file, InnoSetup, Java, Javascript, KiXtart, LISP, **Lua**, Makefile, Matlab, NSIS, Objective-C, Pascal, Perl, php, Postscript

Style : DEFAULT, COMMENT, COMMENT LINE, COMMENT DOC, LITERALSTRING, PREPROCESSOR, INSTRUCTION WORD, NUMBER, STRING, CHARACTER, OPERATOR, FUNC1, **FUNC2**, FUNC3

Lua : FUNC2

Colour Style
Foreground colour 🟪
Background colour ⬜

Font Style
Font name :
[✓] Bold
[ ] Italic
[ ] Underline
Font size :

Default keywords:
abs acos asin atan atan2 ceil cos deg exp floor format frexp gsub ldexp log log10 max min mod rad random randomseed sin sqrt strbyte

User-defined keywords:
sio.connect sio.clear sio.send sio.recv sio.close sio.exclrpt vmsleep pathtofilename reportte atctl.send atctl.setport atctl.recv

Default ext. : lua + User ext. :

Save & Close | Cancel | [ ] Transparency

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
```

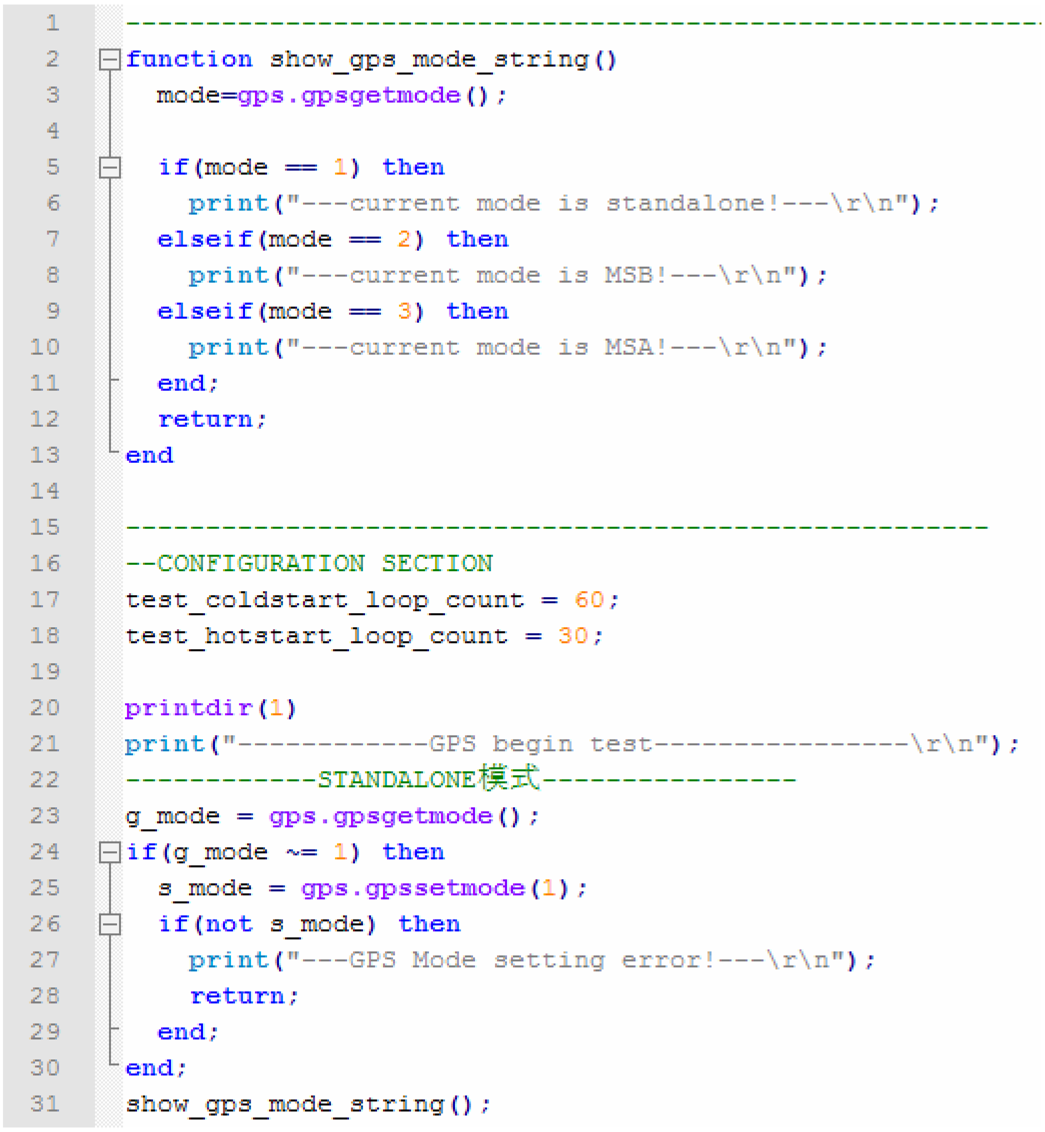
5.2 Perform GPIO Pins Operation

The following script gives an example of operating GPIO pins.

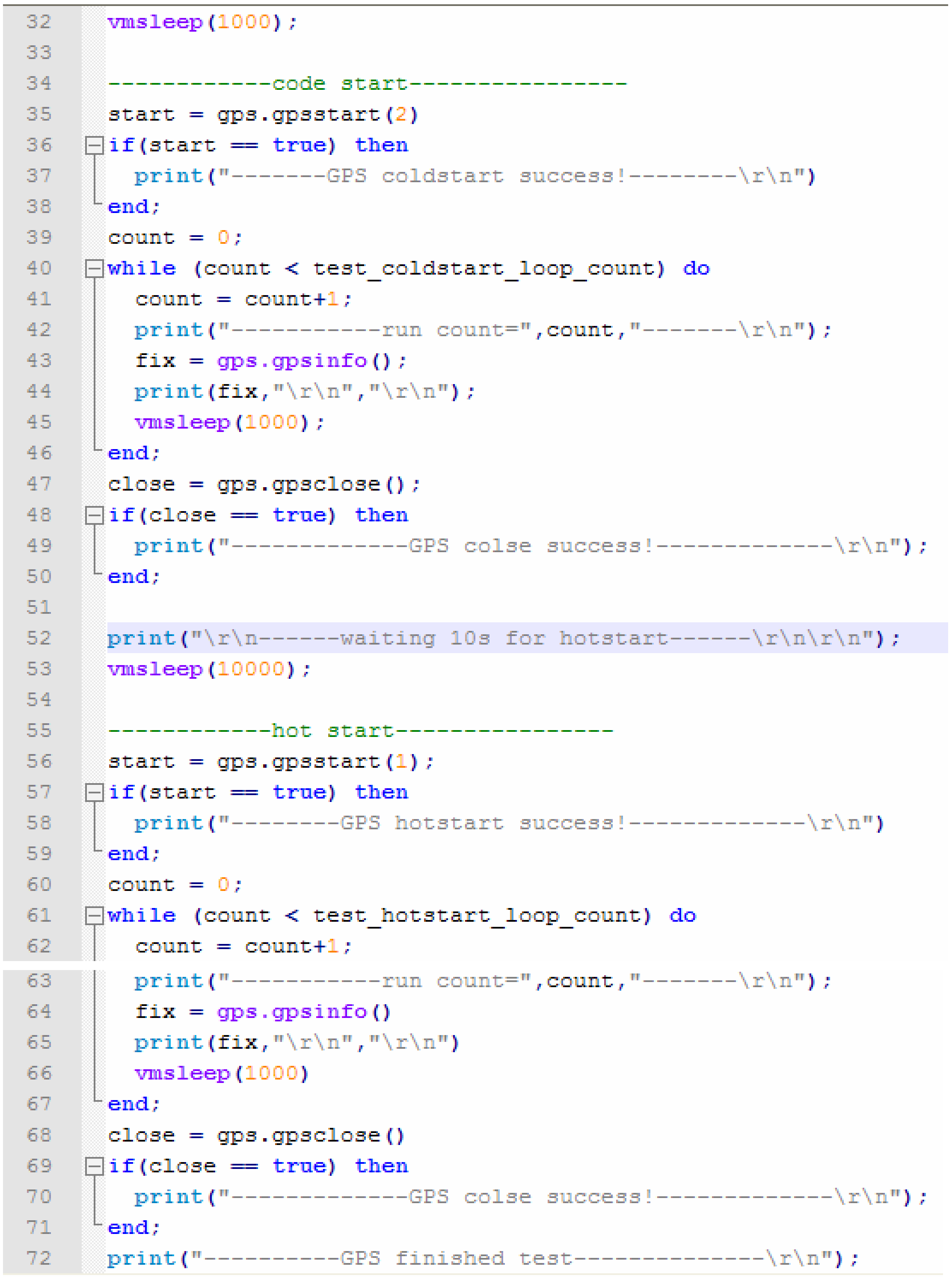
```
printdir(1)
if (true) then
  --Get the GPIO2 level state
  gio = gpio.getv(2);
  print(gio,"\r\n");
  --Set the GPIO2 to low level
  rst = gpio.setv(2,0);
  print(rst,"\r\n");
  --Set the GPIO0 ISR to be triggered at high level
  --and the trigger mode is LEVEL trigger.
  rst = gpio.settrigtype(0,1);
  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();
```



```
vmsleep(1000);

------------code start----------------
start = gps.gpsstart(2)
if(start == true) then
  print("-------GPS coldstart success!--------\r\n")
end;
count = 0;
while (count < test_coldstart_loop_count) do
  count = count+1;
  print("-----------run count=",count,"-------\r\n");
  fix = gps.gpsinfo();
  print(fix,"\r\n","\r\n");
  vmsleep(1000);
end;
close = gps.gpsclose();
if(close == true) then
  print("-------------GPS colse success!-------------\r\n");
end;

print("\r\n------waiting 10s for hotstart------\r\n\r\n");
vmsleep(10000);

------------hot start----------------
start = gps.gpsstart(1);
if(start == true) then
  print("--------GPS hotstart success!-------------\r\n")
end;
count = 0;
while (count < test_hotstart_loop_count) do
  count = count+1;
  print("-----------run count=",count,"-------\r\n");
  fix = gps.gpsinfo()
  print(fix,"\r\n","\r\n")
  vmsleep(1000)
end;
close = gps.gpsclose()
if(close == true) then
  print("-------------GPS colse success!-------------\r\n");
end;
print("----------GPS finished test--------------\r\n");
```

5.4 Perform ATCTL Operation

The following script sets the USBAT port to ATCTL mode, and control the input and output of the port instead of letting the normal AT command processing engine to control the port.

```
--supported port for atctl.setport(...)
ATCTL_UART_PORT  = 1
ATCTL_MODEM_PORT = 2
ATCTL_USBAT_PORT = 3
--  -1 is used to release the port
ATCTL_INVALID_PORT = -1

printdir(1)
--set the USBAT port to be used by ATCTL
atctl.setport(ATCTL_USBAT_PORT)
atctl.send("\r\nplease press any key in the atctl port\r\n");
--Now, user can input and data from TE, which will not be processed by
--AT command processing engine, but received by ATCTL module
count = 0;
while (count < 100) do
  --receive the data input from TE
  data = atctl.recv(15000);
  if (data) then
    count = count + 1;
    --send data to TE
    atctl.send("received data from external port:"..data.."\r\n");
  end;
end;
--Release the USBAT port, and set it as
--normal AT command processing state
atctl.setport(ATCTL_INVALID_PORT);
```

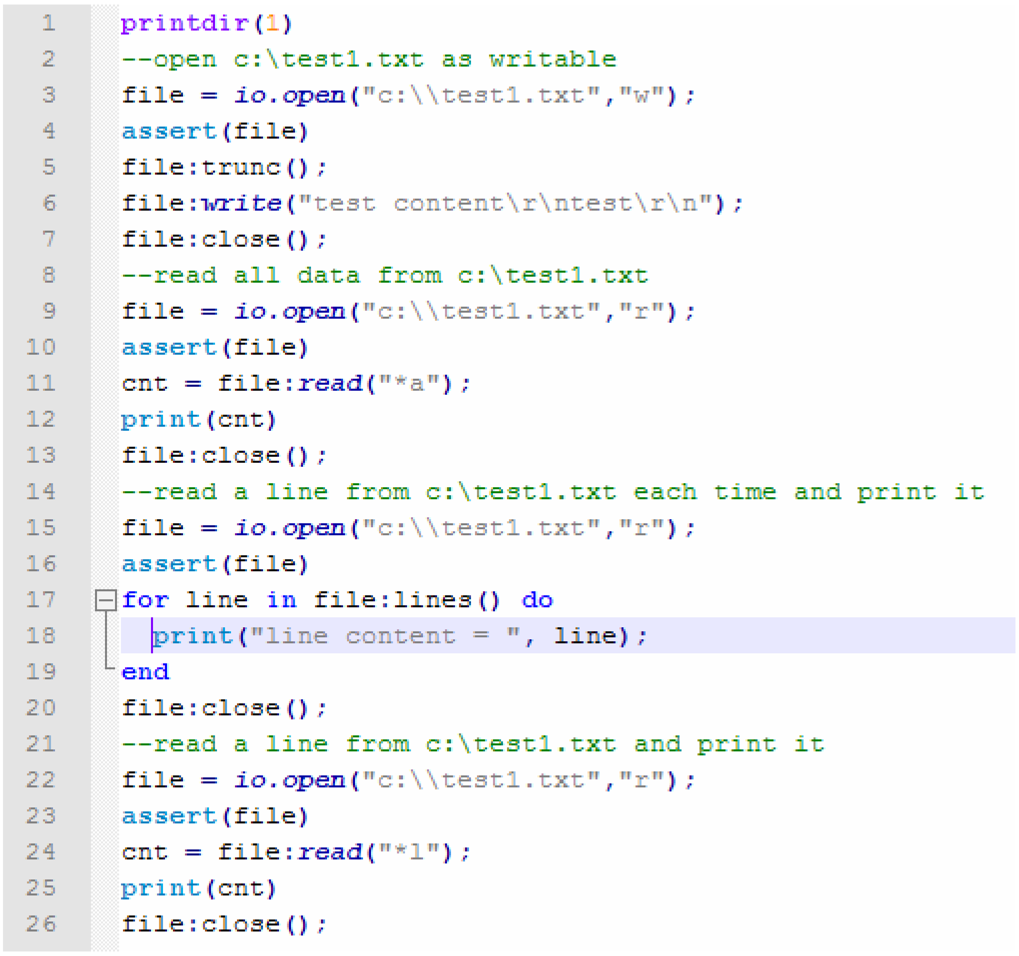
5.5 Perform FTP Operation

The following script gives an example of FTP put and get operations.

```
printdir(1)
-----------------------------------------------------------
pdp_apn = "mytestapn"; server = "myftpserver"; port = 21;
name = "mytestname"; pass = "mytestpass";
remote_file = "/up_normal.jpg";
uplocal_file = "c:\\Picture\\normal.jpg";
downlocal_file = "c:\\Video\\down_normal.jpg"; passive = 0;
------------------------------------------------------------
sio.send("ATE0\r\n"); rsp = sio.recv(5000);
cmd = string.format("AT+CGSOCKCONT=1,\"IP\",\"%s\"\r\n",pdp_apn);
sio.send(cmd);
rsp = sio.recv(5000);
--upload a file to FTP server
rst = ftp.simpput(server, port, name, pass, remote_file,
            uplocal_file, passive);
if (rst ~= 0) then
  print("ftp.simpput failed, rst = ", rst,"\r\n");
else
  print("ftp.simpput succeeded\r\n");
  put_suc = true;
end;
--download a file from FTP server
rst = ftp.simpget(server, port, name, pass, remote_file,
            downlocal_file, passive);
if (rst ~= 0) then
  print("ftp.simpget failed, rst = ", rst,"\r\n");
else
  print("ftp.simpget succeeded\r\n");
end;
sio.send("ATE1\r\n"); rsp = sio.recv(5000);
```

5.6 Perform EFS Operation

The following script gives an example of operating files in EFS.



```
printdir(1)
--open c:\test1.txt as writable
file = io.open("c:\\test1.txt","w");
assert(file)
file:trunc();
file:write("test content\r\ntest\r\n");
file:close();
--read all data from c:\test1.txt
file = io.open("c:\\test1.txt","r");
assert(file)
cnt = file:read("*a");
print(cnt)
file:close();
--read a line from c:\test1.txt each time and print it
file = io.open("c:\\test1.txt","r");
assert(file)
for line in file:lines() do
  print("line content = ", line);
end
file:close();
--read a line from c:\test1.txt and print it
file = io.open("c:\\test1.txt","r");
assert(file)
cnt = file:read("*l");
print(cnt)
file:close();
```

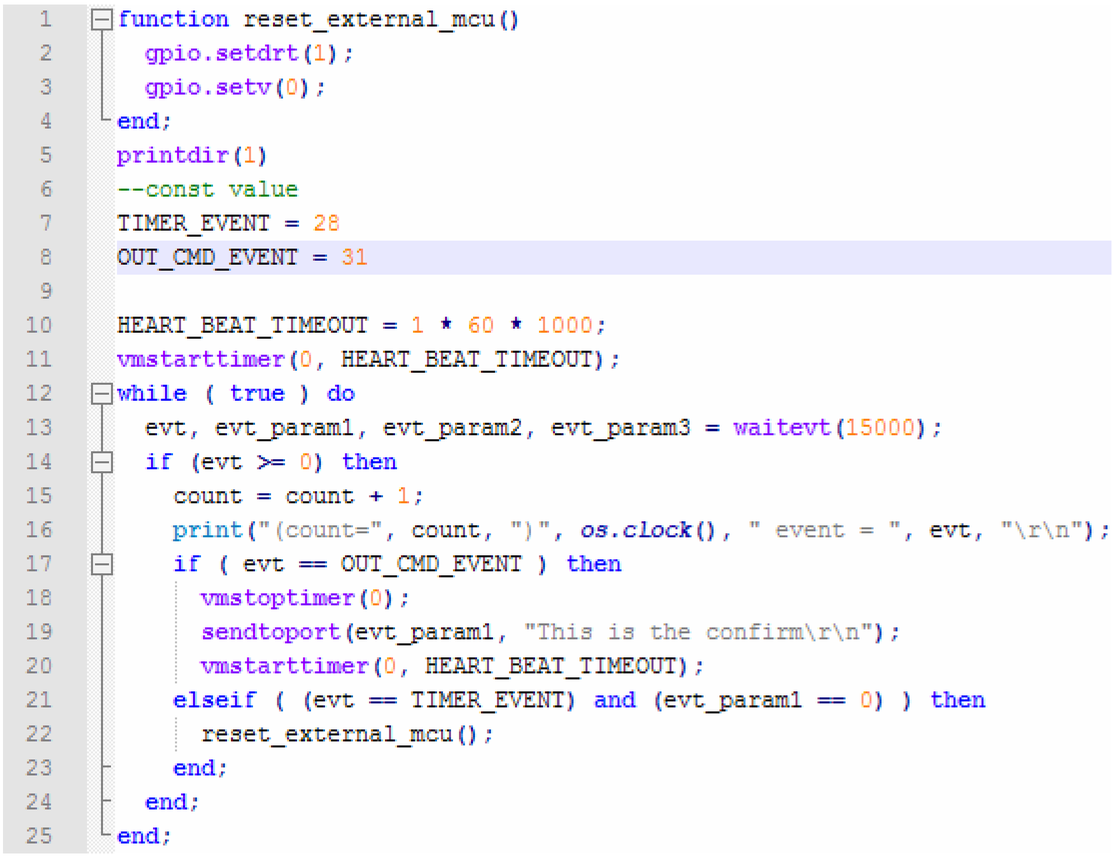
5.7 Perform Bitwise Operation

The following script gives an example of bitwise operation.

```
--// script example START ---------------------------
printdir(1)
print ("bit.bits = " .. bit.bits, "\r\n")
assert (bit.band (0, 0) == bit.cast (0))
assert (bit.band (0, -1) == bit.cast (0))
assert (bit.band (-1, -1) == bit.cast (-1))
assert (bit.bor (0, 0) == bit.cast (0))
assert (bit.bor (0, -1) == bit.cast (-1))
assert (bit.bor (-1, -1) == bit.cast (-1))
assert (bit.bxor (0, 0) == bit.cast (0))
assert (bit.bxor (0, -1) == bit.cast (-1))
assert (bit.bxor (-1, -1) == bit.cast (0))
assert (bit.bnot (0) == bit.cast (-1))
assert (bit.bnot (-1) == bit.cast (0))
assert (bit.lshift (0, 0) == bit.cast (0))
assert (bit.lshift (-1, 0) == bit.cast (-1))
assert (bit.rshift (0, 0) == bit.cast (0))
assert (bit.rshift (-1, 0) == bit.cast (-1))
for nb = 1, bit.bits do
 local a = 2 ^ nb - 1
 print ("nb = " .. nb .. ", a = " .. a, "\r\n")
 assert (bit.band (a, 0) == bit.cast (0))
 assert (bit.band (a, 1) == bit.cast (1))
 assert (bit.band (a, -1) == bit.cast (a))
 assert (bit.band (a, a) == bit.cast (a))
 assert (bit.bor (a, 0) == bit.cast (a))
 assert (bit.bor (a, 1) == bit.cast (a))
 assert (bit.bor (a, -1) == bit.cast (-1))
 assert (bit.bor (a, a) == bit.cast (a))
 assert (bit.bxor (a, 0) == bit.cast (a))
 assert (bit.bxor (a, 1) == bit.cast (a - 1))
 assert (bit.bxor (a, -1) == bit.cast (-a - 1))
 assert (bit.bxor (a, a) == bit.cast (0))
 assert (bit.bnot (a) == bit.cast (-1 - a))
 if nb < bit.bits then
  assert (bit.lshift (a, 1) == bit.cast (a + a))
  assert (bit.lshift (1, nb) == bit.cast (2 ^ nb))
 end
 assert (bit.rshift (a, 1) == math.floor (a / 2))
 if nb < bit.bits then
  assert (bit.rshift (a, nb) == bit.cast (0))
 end
 assert (bit.rshift (a, nb - 1) == bit.cast (1))
 assert (bit.arshift (-1, 1) == bit.cast (-1))
end
```

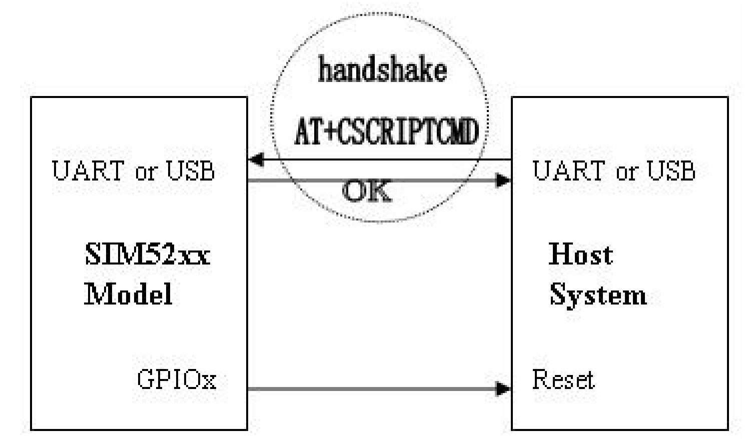
5.8 Use Heart Beat

The following script gives an example of using heart heat. In this example, the external MCU should send AT+CSCRIPTCMD command to module every 1 minute, which generates event 31 to the running LUA script. Once the script receives event 31, it will reset the heart beat timer. If no event 31 is received within 1 minute, the script will reset external MCU using reset_external_mcu() function.



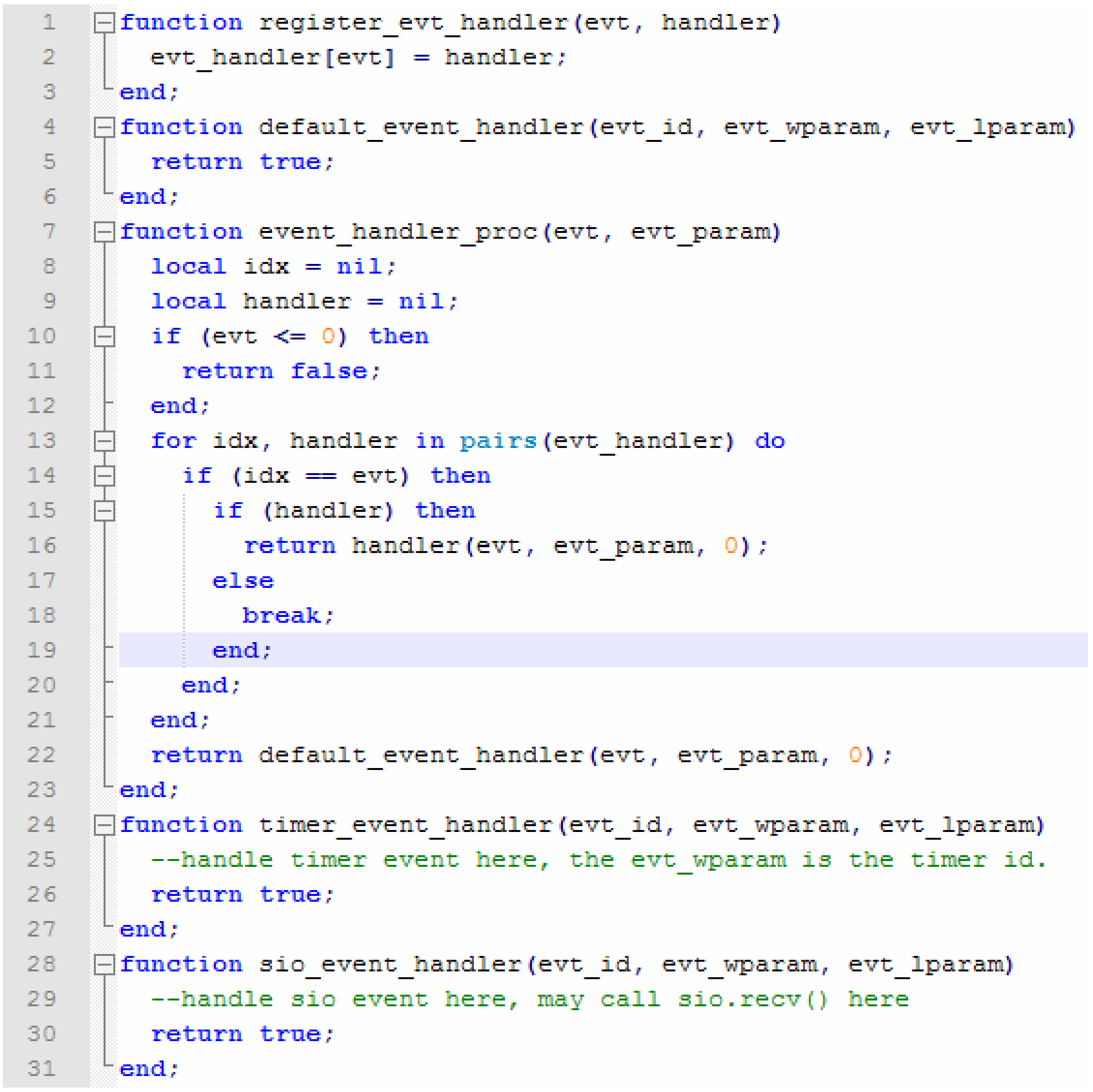
```
function reset_external_mcu()
  gpio.setdrt(1);
  gpio.setv(0);
end;
printdir(1)
--const value
TIMER_EVENT = 28
OUT_CMD_EVENT = 31

HEART_BEAT_TIMEOUT = 1 * 60 * 1000;
vmstarttimer(0, HEART_BEAT_TIMEOUT);
while ( true ) do
  evt, evt_param1, evt_param2, evt_param3 = waitevt(15000);
  if (evt >= 0) then
    count = count + 1;
    print("(count=", count, ")", os.clock(), " event = ", evt, "\r\n");
    if ( evt == OUT_CMD_EVENT ) then
      vmstoptimer(0);
      sendtoport(evt_param1, "This is the confirm\r\n");
      vmstarttimer(0, HEART_BEAT_TIMEOUT);
    elseif ( (evt == TIMER_EVENT) and (evt_param1 == 0) ) then
      reset_external_mcu();
    end;
  end;
end;
```

Following is a general example of using heart beat design:

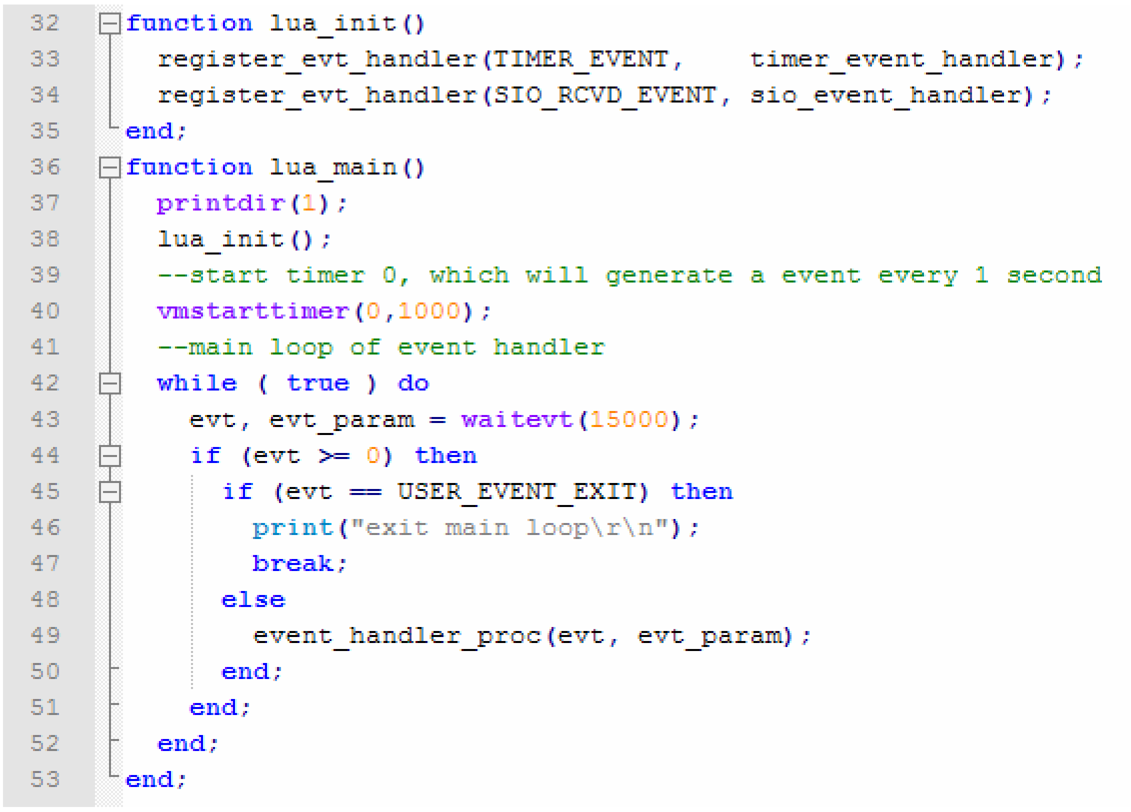


5.9 Use Event Functions

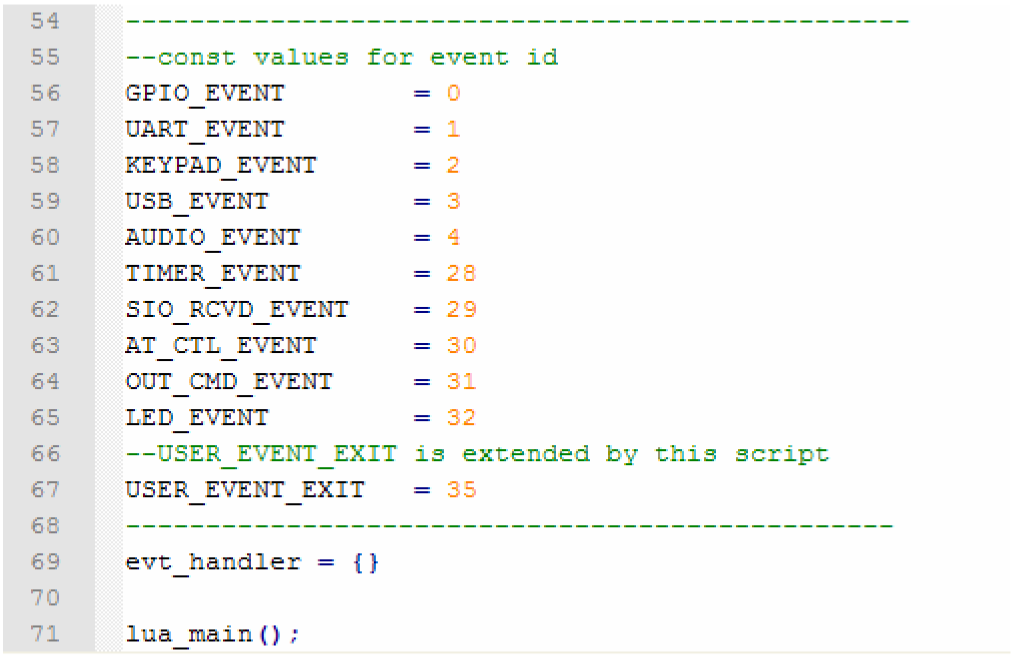
The following script gives an example of using event functions.



```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;
```



```lua
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;
```



```lua
-------------------------------------------------
--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.

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.

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/