SIM7600 Series_Open Linux_Development Guide

LTE Module

SIMCom Wireless Solutions Limited
Building B, SIM Technology Building, No.633, Jinzhong Road
Changning District, Shanghai P.R. China
Tel: 86-21-31575100
support@simcom.com
www.simcom.com

Document Title:	SIM7600 Series_Open Linux_Development Guide
Version:	2.00
Date:	2020.8.6
Status:	Released

SIMCom Wireless Solutions Limited
Building B, SIM Technology Building, No.633 Jinzhong Road, Changning District, Shanghai P.R. China
Tel: +86 21 31575100
Email: simcom@simcom.com

For more information, please visit:
https://www.simcom.com/download/list-863-en.html

For technical support, or to report documentation errors, please visit:
https://www.simcom.com/ask/ or email to: support@simcom.com

Version History

Version	Date	Owner	What is new
V2.00	2019.04.01		Update document format

This document applies to the following products

The document just appliesfor SIM7600E-H/SIM7600SA-H series

Contents

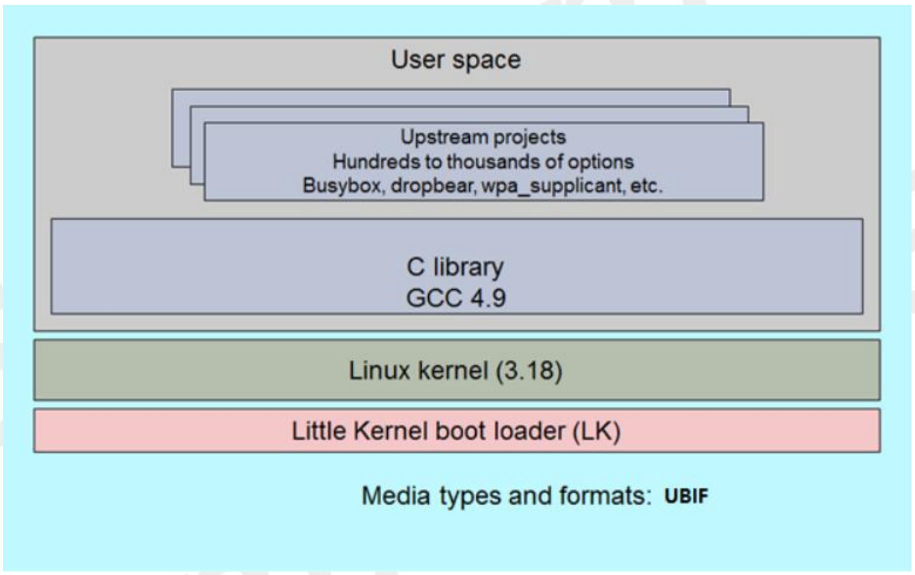
Abbreviation

AT	ATtention; the two-character abbreviation is used to start a command line to be sent from TE/DTE to TA/DCE
DCE	Data Communication Equipment; Data Circuit terminating Equipment
DCS	Digital Cellular Network
DTE	Data Terminal Equipment
DTMF	Dual Tone Multi–Frequency
EDGE	Enhanced Data GSM Environment
EGPRS	Enhanced General Packet Radio Service
GPIO	General–Purpose Input/Output
GPRS	General Packet Radio Service
GSM	Global System for Mobile communications
HSDPA	High Speed Downlink Packet Access
HSUPA	High Speed Uplink Packet Access
I2C	Inter–Integrated Circuit
IMEI	International Mobile station Equipment Identity
IMSI	International Mobile Subscriber Identity
ME	Mobile Equipment
MO	Mobile–Originated
MS	Mobile Station
MT	Mobile–Terminated; Mobile Termination
PCS	Personal Communication System
PIN	Personal Identification Number
PUK	Personal Unlock Key
SIM	Subscriber Identity Module
SMS	Short Message Service
SMS–SC	Short Message Service–ServiceCenter
TA	Terminal Adaptor; e.g. a data card (equal to DCE)
TE	Terminal Equipment; e.g. a computer (equal to DTE)
UE	User Equipment
UMTS	Universal Mobile Telecommunications System
USIM	Universal Subscriber Identity Module
WCDMA	Wideband Code Division Multiple Access
FTP	File Transfer Protocol
HTTP	Hyper Text Transfer Protocol
RTC	Real Time Clock
NAS	Network Access Service
WDS	Wireless Data Service
QMI	Qualcomm Messaging Interface

1 SIM7600 Development Platform Overview

1.1 System Overview

The development platform of SIM7600 module is Linux system, its frame is as follows：


User space
Upstream projects
Hundreds to thousands of options
Busybox, dropbear, wpa_supplicant, etc.
C library
GCC 4.9
Linux kernel (3.18)
Little Kernel boot loader (LK)
Media types and formats: UBIF

The SIM7600 is based on the ARM Cotex-A7 1.3GHz CPU and runs the Linux operating system. The kernel version is 3.18.20. The file system uses the UBIFS file system. The Linux-managed ubi file system contains three logical partitions:
-ubi0:rootfs：
The rootfs logical partition is read-only and holds the Linux code.
-ubi0:usrfs：
Usrfs stores the Linux file system, which is generally used for Open Linux of user applications on this partition.
-ubi0:cachefs:
Cachefs is generally used for FOTA upgrades. If there is not enough cache space during the upgrade, the upgrade program will release some of the space for FOTA upgrade by deleting unopened files. So for security reasons, it is best not to put the user's data on the cache partition.

The three partitions of SIM7600E-H are detailed in the following table：

Filesystem	Size	Used	Available	Use%	Mounted on

ubi0:rootfs	50.2M	36.1M	14.1M	72%	/
ubi0:usrfs	10.5M	364K	10.1M	3%	/data
Ubi0:cachefs	40.8M	24K	38.6M	0%	/cache

1.2 Open Linux selection and function

According to the needs of the market, we have introduced several models of 4G modules that support Open Linux. Customers can choose the best cost-effective solution based on their own product definition. Specific support functions are as follows：

	SIM7600E-H
Platform	MDM9x07
Memory(bit)	2+2
Protocols	TCP/IP/IPV4/IPV6/Multi PDP/FTPS/HTTPS/DNS/COAP/MQTT
CALL	.
ECALL	.
SMS	.
TLS1.2	.
Audio Record/Play	.
TTS	.
DTMF	.
LBS	.
FOTA	.
Security	
Routing policy	4G,WIFI and Ehternet port
NDIS/RNDIS	.
Bluetooth	BLE4.2（External w58 module）
WIFI	2.4G（External w58 module）
GNSS	GPS/GLONASS/BEIDOU
SGMII	HSCI turn to LAN9730(100M)
UART	2* High UARTs
USB2.0	.
OTG	.
HSIC	.
Audio PCM	1*PCM
Audio Analog	1input+1output
	（Common access with PCM）
GPIO	At least 5*GPIO

SDIO	SDIO3.0（200MHz Max）
SD Card	SD3.0（128G Max）
SPI	Optional
I2C	1*I2C

1.3 Open Linux related PIN definition

SIM7600E-H

PIN No.	PINNAME	SYS GPIO No.	Default Function	Function 1	Function2	Pull	Wakeup Interrupt
6	SPI_CLK	--	UART1_RTS	--	--	B-PD	
7	SPI_MISO	--	UART1_RX	--	--	B-PD	
8	SPI_MOSI	--	UART1_TX	--	--	B-PD	
9	SPI_CS	--	UART1_CTS	--	--	B-PD	
21	SD_CMD	--	SD Card	--	--	B-PD	
22	SD_DATA0	--	SD Card	--	--	B-PD	
23	SD_DATA1	--	SD Card	--	--	B-PD	
24	SD_DATA2	--	SD Card	--	--	B-PD	
25	SD_DATA3	--	SD Card	--	--	B-PD	
26	SD_CLK	--	SD Card	--	--	B-NP	
27	SDIO_DATA1	--	WLAN	--	--	B-PD	
28	SDIO_DATA2	--	WLAN	--	--	B-PD	
29	SDIO_CMD	--	WLAN	--	--	B-PD	
30	SDIO_DATA0	--	WLAN	--	--	B-PD	

31	SDIO_DATA3	--	WLAN	--	--	B-PD	
32	SDIO_CLK	--	WLAN	--	--	B-NP	
33	GPIO3	GPIO_1020	MIFI_POWER_EN	GPIO	MIFI_POWER_EN	B-PU	
34	GPIO6	GPIO_1023	MIFI_SLEEP_CLK	GPIO	MIFI_SLEEP_CLK	B-PD	
46	ADC2		ADC			--	
47	ADC1		ADC			B-PU	
48	SD_DET	GPIO_26	GPIO	GPIO	SD_DET	B-PD	
49	STATUS	GPIO_52	Status	GPIO	Status	B-PD	
50	GPIO43	GPIO_36	MIFI_COEX	GPIO	MIFI_COEX	B-PD	
52	GPIO41	GPIO_79	BT	GPIO	BT	B-PD	*
55	SCL	--	I2C_SCL	--	--	B-PD	
56	SDA	--	I2C_SDA	--	--	B-PU	
66	RTS	--	UART2_RTS	--	--	B-PD	
67	CTS	--	UART2_CTS	--	--	B-PD	
68	RXD	--	UART2_RX	--	--	B-PD	
69	RI	--	GPIO(RI)	--	--	B-PD	
70	DCD	--	GPIO	--	--	B-PD	
71	TXD	--	UART2_TX	--	--	B-PD	
72	DTR	--	GPIO(DTR)	--	--	B-PD	*
73	PCM_OUT	--	PCM	--	--	B-PD	
74	PCM_IN	--	PCM	--	--	B-PD	
75	PCM_SYNC	--	PCM	--	--	B-PD	
76	PCM_CLK	--	PCM	--	--	B-P	

						U	
87	GPIO77	GPIO_77	BT	GPIO	BT	B-P D	

2. The environment of Open Linux

Open Linux has both Windows and Linux environments. Customers can choose their own familiar environment for development. If you only develop one application using Windows will be faster. If you are concerned with Kernel changes, it is must use the Linux environment.

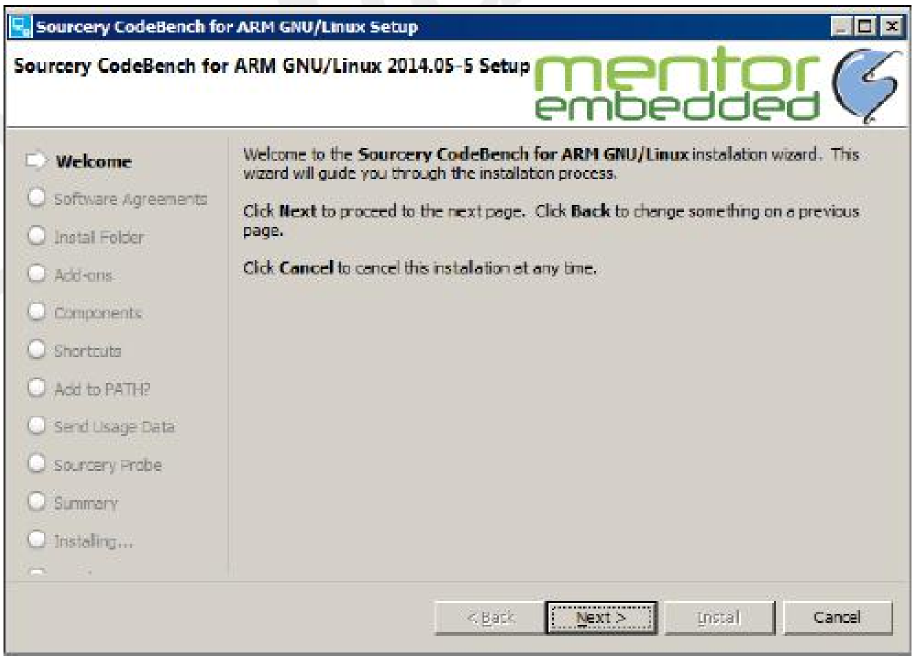
2.1 Install Windows Embedded Compiler

2.1.1 Install ARM GNU/LINUX

The development of SIM7600 is actually the development of embedded Linux. You can choose to build an ARM-Linux compiler compatible with SIM7600 platform for compiling applications under the Windows operating system.
Tools for installing applications for compiling SIM7600 under Windows system is Sourcery CodeBench for ARM GNU/Linux

Download link：https://sourcery.mentor.com/public/gnu_toolchain/

Installation method:

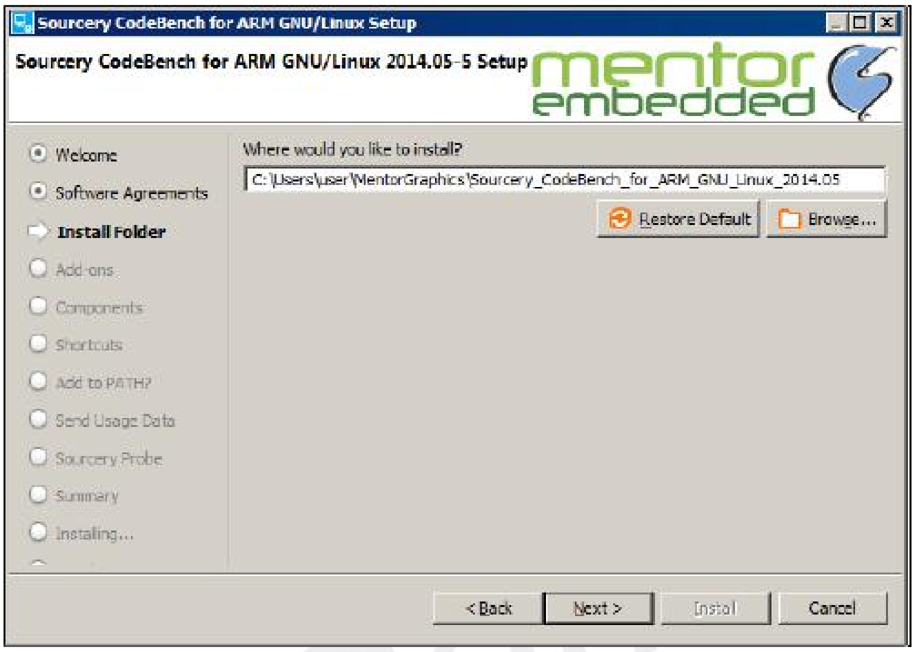

Sourcery CodeBench for ARM GNU/Linux Setup

Sourcery CodeBench for ARM GNU/Linux 2014.05-5 Setup

Welcome to the **Sourcery CodeBench for ARM GNU/Linux** installation wizard. This wizard will guide you through the installation process.

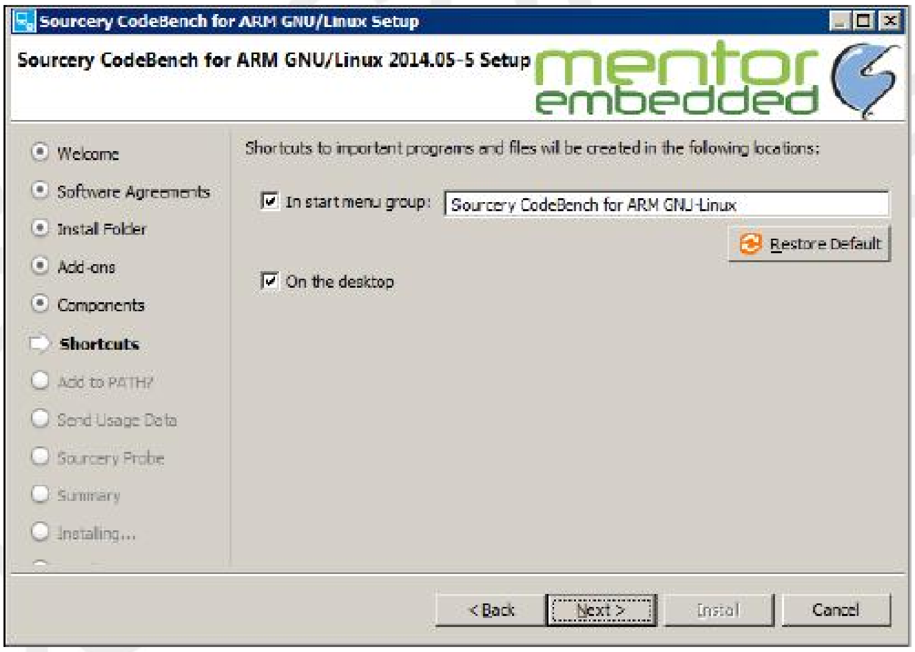
Click **Next** to proceed to the next page. Click **Back** to change something on a previous page.

Click **Cancel** to cancel this installation at any time.



Where would you like to install?

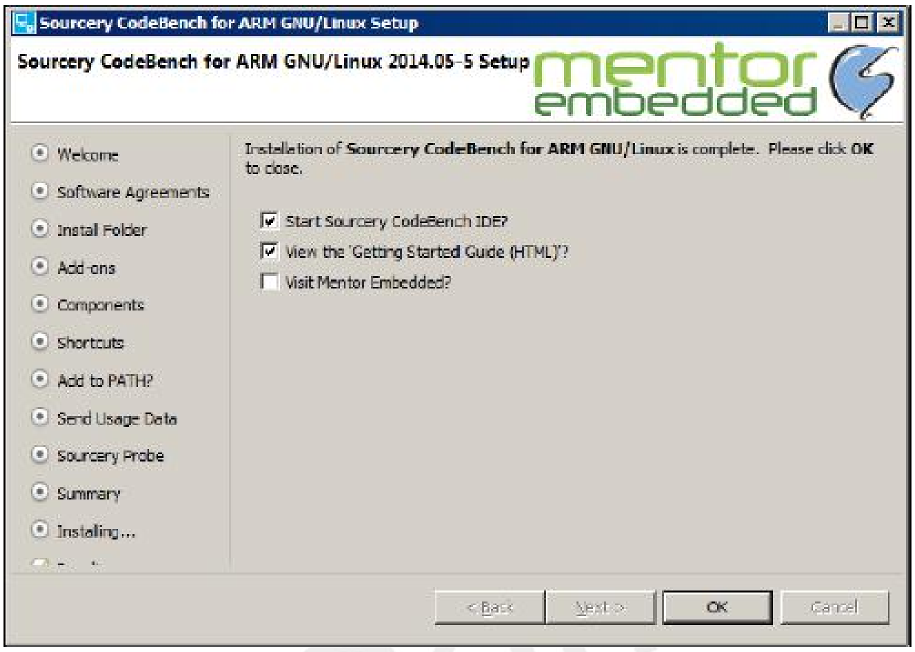
C:\Users\user\MentorGraphics\Sourcery_CodeBench_for_ARM_GNU_Linux_2014.05



Shortcuts to important programs and files will be created in the following locations:

☑ In start menu group: Sourcery CodeBench for ARM GNU-Linux

☑ On the desktop



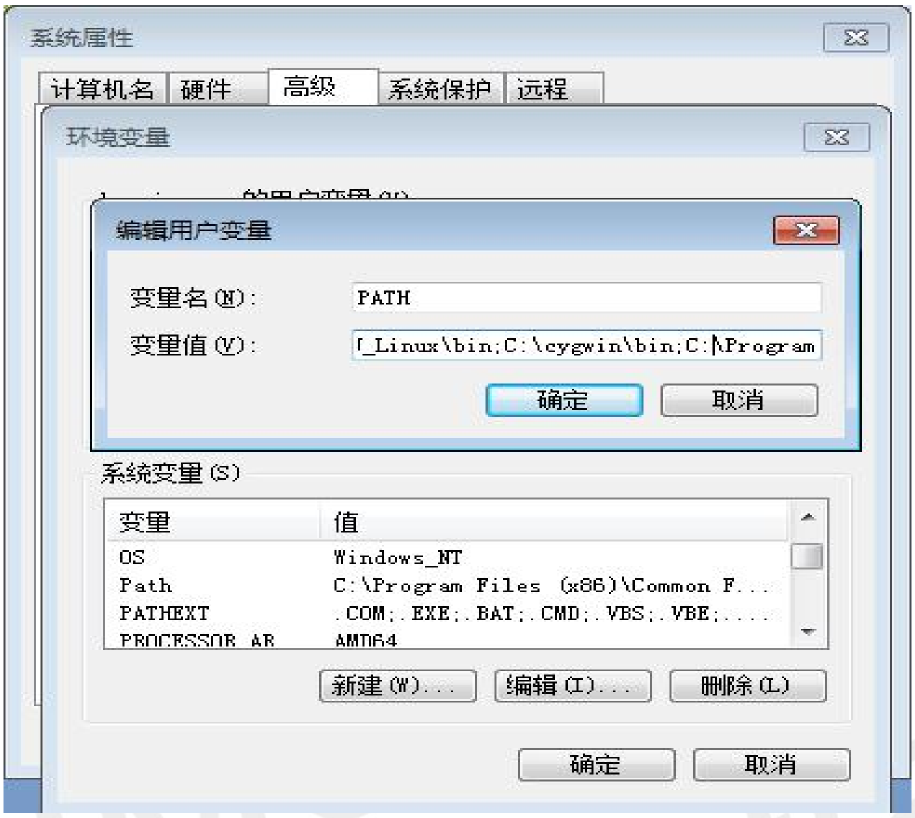
Installation of Sourcery CodeBench for ARM GNU/Linux is complete. Please click OK to close.

- [✓] Start Sourcery CodeBench IDE?
- [✓] View the 'Getting Started Guide (HTML)'?
- [ ] Visit Mentor Embedded?

2.1.2 Install Cygwin

In order to simplify the compilation of the application and facilitate the management of the application's code engineering, it is recommended to compile the program using make. Cygwin's make program can be used under Windows environment. Customers can download the free Cygwin from cygwin's official website.

After the installation, then add the Cygwin/bin/ path to the environment variable. As shown in the following figure:



变量名(N): PATH

变量值(V): [_Linux\bin;C:\cygwin\bin;C:\Program

系统变量(S)

变量	值
OS	Windows_NT
Path	C:\Program Files (x86)\Common F...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;....
PROCESSOR_AR	AMD64

2.2 Configure the Linux compilation environment

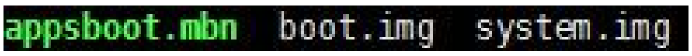
It is recommended to use Ubuntu12.04 64-bit as a compiler system.

2.2.1 Compilation method

Using sudo tar command to extract sdk archive, such as extracting to the current directory command as follow:
sudo tar xzf sim_open_sdk.tar.gz
Then initialize the environment variable in the sim_open_sdk directory, the command is as follows:
source sim_crosscompile/sim-crosscompile-env-init

2.2.1.1 Compile all

Executing make command in the sim_open_sdk directory. All the images generated after the compilation is completed are in the sim_open_sdk/output directory, as follows.



```
appsboot.mbn  boot.img  system.img
```

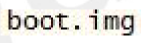
2.2.1.2 Compile bootloader

Executing 'make aboot' in the sim_open_sdk directory.
The image is in the sim_open_sdk/output directory as follows.

```
appsboot.mbn
```

2.2.1.3 Compile kernel

1）Executing make kernel_menuconfig in sim_open_sdk directory to configure the kernel, and then pop up the graphical configuration interface. If you need to add new configuration optionsselect the corresponding configuration in the graphical configuration interface, then save and exit. If you do not add new configuration just quit. After the configuration is complete, the .config file will be generated in the sim_kernel/build/ directory.
2）Executing make kernel to compile the kernel. The generated kernel image is in the sim_open_sdk/output directory.



```
boot.img
```

2.2.1.4 Generate rootfs file system image

Executing make rootfs in the sim_open_sdk directory and the generated image is in the sim_open_sdk/output directory.

```
system.img
```

2.2.1.5 Compile driver module

Executing make kernel_module to compile the driver module. After the compilation is successful, it will be automatically installed in the corresponding directory of sim_rootfs, so then execute make rootfs to regenerate the rootfs file system image.

2.2.1.6 Compile demo

Executingmake demo to compile the demo program and the generated image is in the sim_open_sdk/output directory.

demo_app

2.2.1.7 Clear the generated image

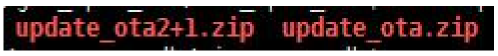
1）Clear all and execute make clean.
2）Clear bootloader and execute make aboot_clean.
3）Clear kernel and execute make kernel_clean.
4）Clear rootfs and execute make rootfs_clean.
5）Clear demo and execute make demo_clean.

2.2.1.8 Question

1) If you execute make kernel_menuconfig to configure the kernel, the following error occurs.
error： ../scripts/kconfig/lxdialog/dialog.h:: fatal error: curses.h: No such file or directory
Executing sudo apt-get install libncurses5-dev.

2.2.2 Making OTA upgrade package command

It needs to provide both the current version package source and the target version package target to make ota upgrade package. Both source and target need to be placed under the sim_open_sdk/sim_ota directory.
1) generation of target
When the application is ready, the bootloader, kernel and system are all compiled. That is after appsboot.mbn, boot.img and system.img have been generated in the output directory, make ota command will be executed in the sim_open_sdk directory, and then the prepared target can be seen in the sim_open_sdk/sim_ota directory.
2) The source prepared by the customer is put into the sim_open_sdk/sim_ota directory, and the source structure is the same as the target structure.
3) Executing ota.sh under the sim_open_sdk directory will generate the ota upgrade package, which is under the sim_open_sdk/output directory, as follows:

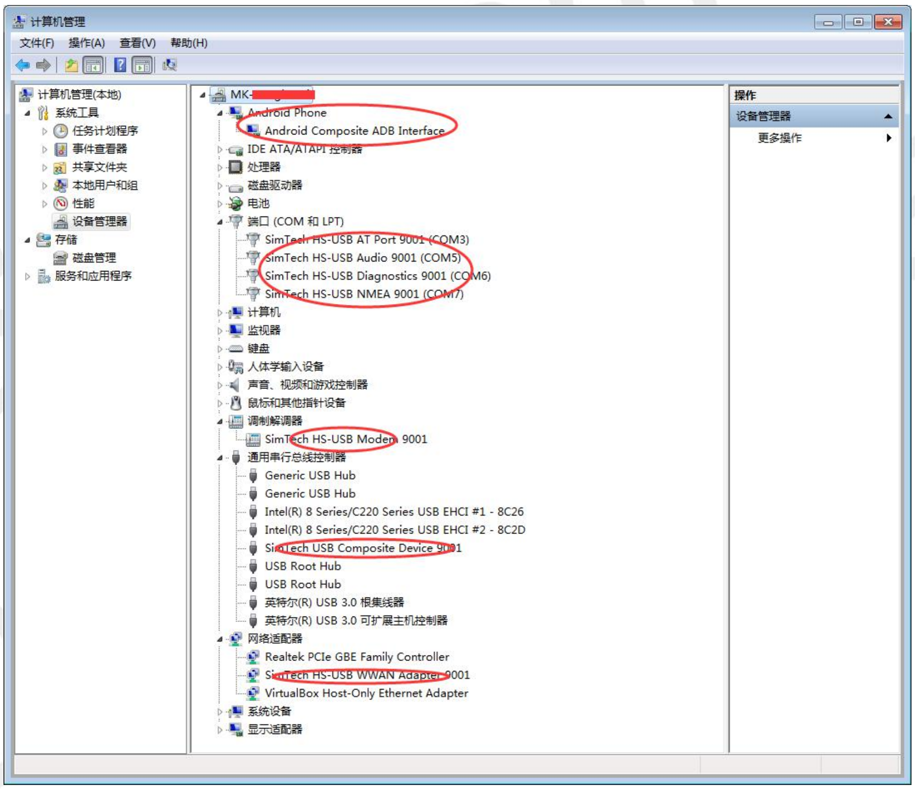


```
update_ota2+1.zip  update_ota.zip
```

The differential modes oftwo packages are different. When update please using update_ota2+1.zip.

2.3 Install Windows driver

After connecting the SIM7600's USB to the computer, some USB virtual devices will appear and drivers will need to be installed. Please use the corresponding Windows driver installation package to install. After installation will as shown below:



计算机管理

文件(F) 操作(A) 查看(V) 帮助(H)

- 计算机管理(本地)
  - 系统工具
    - 任务计划程序
    - 事件查看器
    - 共享文件夹
    - 本地用户和组
    - 性能
    - 设备管理器
  - 存储
    - 磁盘管理
  - 服务和应用程序

- MK-[redacted]
  - Android Phone
    - Android Composite ADB Interface
  - IDE ATA/ATAPI 控制器
  - 处理器
  - 磁盘驱动器
  - 电池
  - 端口 (COM 和 LPT)
    - SimTech HS-USB AT Port 9001 (COM3)
    - SimTech HS-USB Audio 9001 (COM5)
    - SimTech HS-USB Diagnostics 9001 (COM6)
    - SimTech HS-USB NMEA 9001 (COM7)
  - 计算机
  - 监视器
  - 键盘
  - 人体学输入设备
  - 声音、视频和游戏控制器
  - 鼠标和其他指针设备
  - 调制解调器
    - SimTech HS-USB Modem 9001
  - 通用串行总线控制器
    - Generic USB Hub
    - Generic USB Hub
    - Intel(R) 8 Series/C220 Series USB EHCI #1 - 8C26
    - Intel(R) 8 Series/C220 Series USB EHCI #2 - 8C2D
    - SimTech USB Composite Device 9001
    - USB Root Hub
    - USB Root Hub
    - 英特尔(R) USB 3.0 根集线器
    - 英特尔(R) USB 3.0 可扩展主机控制器
  - 网络适配器
    - Realtek PCIe GBE Family Controller
    - SimTech HS-USB WWAN Adapter 9001
    - VirtualBox Host-Only Ethernet Adapter
  - 系统设备
  - 显示适配器

操作

设备管理器

更多操作

Each device corresponds to the function as shown below：

Interface number	windows name	Linux name	Interface description
0	SimTech HS-USB Diagnostics	USB serial	Diagnostic Interface
1	SimTech HS-USB NMEA	USB serial	GPS NMEA Interface
2	SimTech HS-USB AT Port	USB serial	AT port Interface
3	SimTech HS-USB Modem	USB serial	Modem port Interface
4	SimTech HS-USB Audio	USB serial	USB Audio Interface
5	SimTech HS-USB WWAN Adapter	USB Net	NDIS wwan interface
6	Android Composite ADB Interface	USB adb	Android add debug port

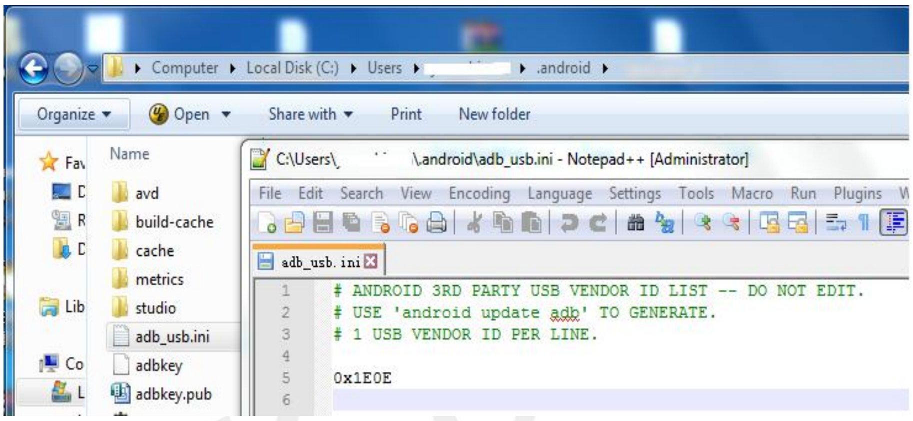
The adb device of SIM7600 is closed by default. It needs to be opened by AT command setting, and then restart the module to take effect.The command is AT+CUSBADB=1

After the command returns OK, you need to restart the module manually. You can see an Android

Composite ADB Interface device from the Windows device manager, which indicates that the adb device can be used.

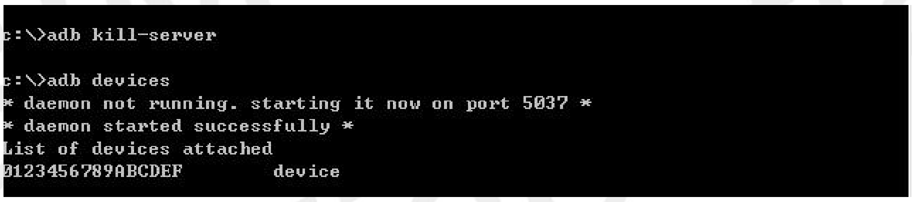
Note：

If the computer is installed with the ADB device for the first time, then add the adb_usb.ini file in the ‘.android’ directory under the current login account in the ‘user’ on the C drive, and add the VID of the SIM7600 to 0x1E0E in the file.



```
# ANDROID 3RD PARTY USB VENDOR ID LIST -- DO NOT EDIT.
# USE 'android update adb' TO GENERATE.
# 1 USB VENDOR ID PER LINE.

0x1E0E
```

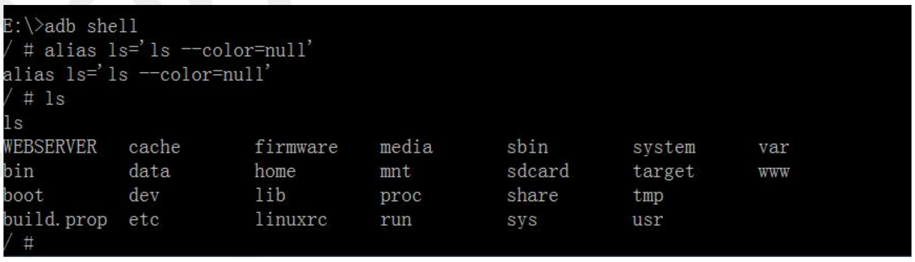
Adb tool can be tested in the Windows cmd window.



```
c:\>adb kill-server

c:\>adb devices
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
List of devices attached
0123456789ABCDEF        device
```

adb shell can access the SIM7600 Linux system console.

Use alias ls=’ls-color=null’ to align the system's display, as shown below：



```
E:\>adb shell
/ # alias ls=’ls --color=null’
alias ls=’ls --color=null’
/ # ls
ls
WEBSERVER   cache     firmware   media    sbin     system   var
bin         data      home       mnt      sdcard   target   www
boot        dev       lib        proc     share    tmp
build.prop  etc       linuxrc    run      sys      usr
/ #
```

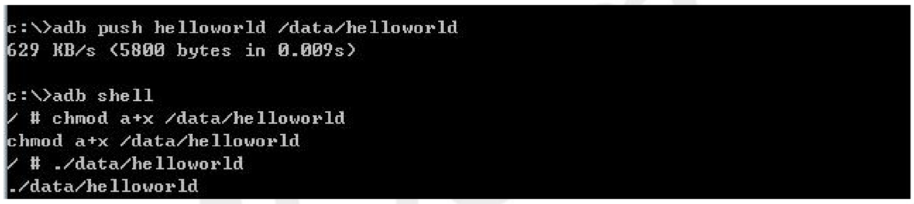
2.4 Debug download tool

NOTE:

Module software version must be open linux development version can only be in accordance with the instructions below to use fastboot to download each partitions. If not the open linux development version, it

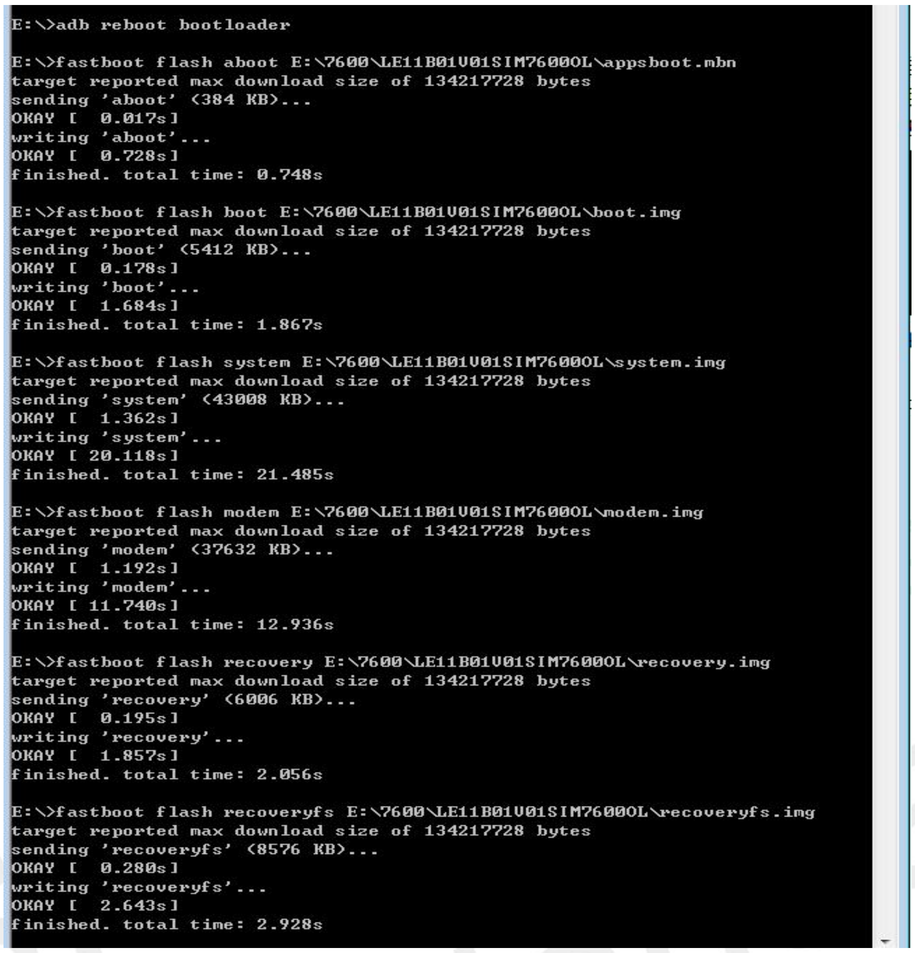
needs to upgrade to an open linux development version before the operation. Through the at command ati, you can check whether it is open linux development version or not. If the version information contains keywords OL, it is an open linux development version.
You must use the tool to download a full open linux development version when download version for the first time. It may cause some function can't be normal used if update part version files using fastboot.

Regardless of the application or compiled image file, it is recommended to use windows to download and debug.
Application debugging uses the adb tool and image download uses the fastboot tool. These two tools can be downloaded from the Internet or use the environment compression package provided by us.
The adb tool can push the app program into the module. The app runs in the foreground and can print all app's log. Same as standard linuxdmesg can print all the kernel's logs.



```
c:\>adb push helloworld /data/helloworld
629 KB/s (5800 bytes in 0.009s)

c:\>adb shell
/ # chmod a+x /data/helloworld
chmod a+x /data/helloworld
/ # ./data/helloworld
./data/helloworld
```

Fastboot tool can download appsboot.mbn, modem.img, boot.img, system.img, recovery.img and recoveryfs.img.
Use AT command at+bootldr or adb command adb reboot bootloader to enter fastboot mode, you can download the above image.

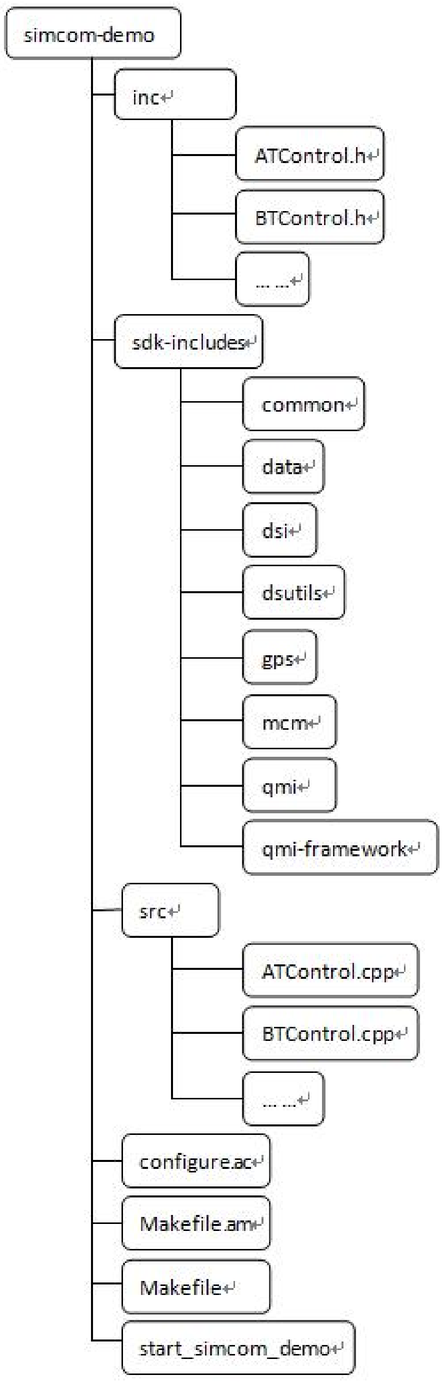


```
E:\>adb reboot bootloader

E:\>fastboot flash aboot E:\7600\LE11B01V01SIM7600OL\appsboot.mbn
target reported max download size of 134217728 bytes
sending 'aboot' (384 KB)...
OKAY [  0.017s]
writing 'aboot'...
OKAY [  0.728s]
finished. total time: 0.748s

E:\>fastboot flash boot E:\7600\LE11B01V01SIM7600OL\boot.img
target reported max download size of 134217728 bytes
sending 'boot' (5412 KB)...
OKAY [  0.178s]
writing 'boot'...
OKAY [  1.684s]
finished. total time: 1.867s

E:\>fastboot flash system E:\7600\LE11B01V01SIM7600OL\system.img
target reported max download size of 134217728 bytes
sending 'system' (43008 KB)...
OKAY [  1.362s]
writing 'system'...
OKAY [ 20.118s]
finished. total time: 21.485s

E:\>fastboot flash modem E:\7600\LE11B01V01SIM7600OL\modem.img
target reported max download size of 134217728 bytes
sending 'modem' (37632 KB)...
OKAY [  1.192s]
writing 'modem'...
OKAY [ 11.740s]
finished. total time: 12.936s

E:\>fastboot flash recovery E:\7600\LE11B01V01SIM7600OL\recovery.img
target reported max download size of 134217728 bytes
sending 'recovery' (6006 KB)...
OKAY [  0.195s]
writing 'recovery'...
OKAY [  1.857s]
finished. total time: 2.056s

E:\>fastboot flash recoveryfs E:\7600\LE11B01V01SIM7600OL\recoveryfs.img
target reported max download size of 134217728 bytes
sending 'recoveryfs' (8576 KB)...
OKAY [  0.280s]
writing 'recoveryfs'...
OKAY [  2.643s]
finished. total time: 2.928s
```

2.5 Application Compilation and Run

2.5.1 Demo application

Demo's directory structure



Note：All ‘demo’mentioned in this article are in the SDK. The directory is simcom-demo.

1） Demo program compiled under windows

Copy the Makefile_win file in the tools directory to the simcom-demo directory and rename it to Makefile. Next compile directly with make command and generate demo_app application in the simcom-demo/bin directory. The compiled executable file is imported into the data directory of the SIM7600 Linux system by adb push, and the execute permission is added. Thenthe adb shell can be used to enter the SIM7600 Linux console to manually debug the execution program.During the debugging phase, the client starts and debugs the application through the control terminal of the adb shell. The code can use the printf method to debug the log and information.

2） Demo program compiled under linux

See section 2.2.1.6 for the compilation method.

2.5.2 Helloworld application

1） Compile helloworld

Executing make helloworld, and the generated application is in the sim_open_sdk/output directory.

2） Helloword Self-starting settings

Copy the application helloworld compiled above to the sim_open_sdk/sim_usrfs directory. Then copy the sim_open_sdk/helloworld/start_helloworld file to the sim_open_sdk/sim_rootfs/etc/init.d/ directory and executes the following command in the sim_open_sdk directory.

sudo ln -sf ../init.d/start_simcom_demosim_rootfs/etc/rc5.d/S99start_simcom_demo

Then regenerate the rootfs image according to Section 2.2.1.4.

2.6 System partition and file protection

In system partition, system, data and cacheare divided into one physical partition. The size of the three logical partitions can be dynamically adjusted. The adjusted file is sim_open_sdk/tools/ubinize_system_userdata.cfg

```
[sysfs_volume]
mode=ubi
image=./mdm9607-perf-sysfs.ubifs
vol_id=0
vol_type=dynamic
vol_name=rootfs
vol_size=55MiB // Adjust the size of the system partition here
[usrfs_volume]
mode=ubi
image=./usrfs/mdm9607-perf-usrfs.ubifs
vol_id=1
vol_type=dynamic
vol_name=usrfs
vol_flags = autoresize//Data partition size is set automatically
[cache_volume]
mode=ubi
vol_id=2
vol_type=dynamic
vol_name=cachefs
vol_size=45MiB //Adjust the size of the cache partition here
```

All necessary files in the system must be placed in the system directory. The temporarily generated files are placed in the data directory. Note that system is a read-only directory. The data files stored as temporary files may be lost under certain extreme conditions.

When the system is running, in order to ensure the security of the process the scheme of /data and /cache

backup is adopted. (Refer to helloworld/helloworld.c and helloworld/helloworld_auto_restore.sh in the open linux environment.)
Application under module /data: /data/helloworld
The version identifier of the application under module /data: /data/helloworld_ver.txt
Application under module /cache: /cache/helloworld_bak
The version identifier of the application under module/cache: /cache/helloworld_bak_ver.txt

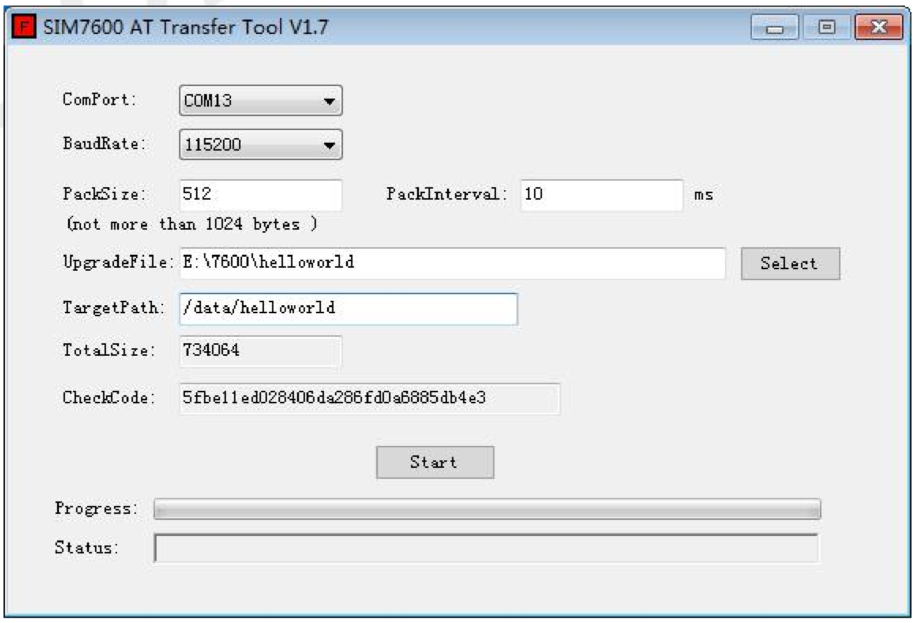
(1) Execute the detection script /etc/init.d/helloworld_auto_restore.sh when the module is powered on.
(2) The first boot will copy /data/helloworld to /cache/helloworld_bak and execute /data/helloworld
(3) If /data/helloworld does not exist, copy /cache/helloworld_bak to /data/helloworld.
    if /cache/helloworld_bak does not exist, copy /data/helloworld to /cache/helloworld_bak.
(4) If both exist, the version identifier corresponding to the two is determined. If the two are inconsistent, the higher version of the program is overwritten to the lower version of the program.
(5) Application upgrade strategy
    1) /data/helloworlddeletes /cache/helloworld_bak_ver.txt and /cache/helloworld_bak in order.
    2) /data/helloworld downloads the target program to /cache/helloworld_bak.
    3) /data/helloworld starts /cache/helloworld_bak update &
    4) /cache/helloworld_bak writes its own version ID.
    5) Reboot

Note:
Due to the caching principle of the Linux system, the file operation will not be immediately written to the flash, and abnormal power failure will result in file corruption or loss. Therefore, for important file operations, you need to execute the sync command to ensure the operation is complete.

2.7 Production line production mode

2.7.1 Download APP separately

SIMCom provides a tool for the production line to download customer developed applications. As shown below：

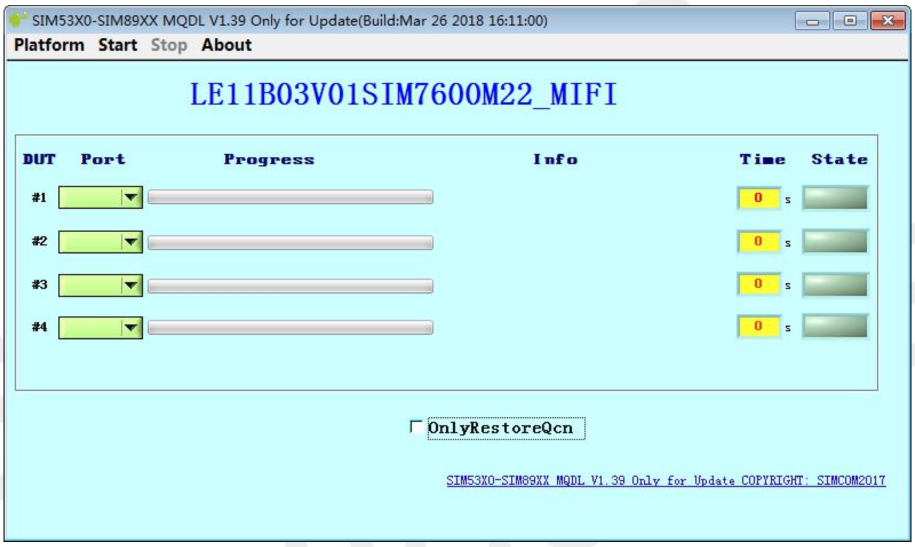

SIM7600 AT Transfer Tool V1.7

ComPort: COM13
BaudRate: 115200
PackSize: 512 PackInterval: 10 ms
(not more than 1024 bytes)
UpgradeFile: E:\7600\helloworld
TargetPath: /data/helloworld
TotalSize: 734064
CheckCode: 5fbe11ed028406da286fd0a6885db4e3
Progress:
Status:

The basic principle of this tool is that the customer selects his own compiled application file and then selects Down File. The tool will place this program in the module /data and /usr/bin directory and rename it 'helloworld', and then automatically modify it to executable permissions. The next time the module is booted, it will automatically check if there is helloworld under data. If so, it will automatically execute and complete the client's program startup.

2.7.2 Download the compiled image

Providing production line tools for customers to download and use directly. You can download them in four places at the same time. After the tools are installedthere will be instructions for use.


SIM53X0-SIM89XX MQDL V1.39 Only for Update(Build:Mar 26 2018 16:11:00)
Platform
Start
Stop
About
LE11B03V01SIM7600M22_MIFI
DUT
Port
Progress
Info
Time
State
#1
#2
#3
#4
0 s
OnlyRestoreQcn
SIM53X0-SIM89XX MQDL V1.39 Only for Update COPYRIGHT: SIMCOM2017

3. Programming Guide

3.1 System basic API

The basic API of the SIM7600 Linux system is the same as all Linux systems. But timer and RTC timer are different. The RTC timer is still accurate after the system sleeps. TCP (socket) demo with the use of Linux can refer to the specific code DEMO.

3.2 Embedded AT transceiver

The Open Linux application can use the functions of all AT commands. AT command communication is supported by using the tty device node /dev/smd8 in Linux.

3.2.1 Send AT command interface

Interface	int sendATCmd(char* atCmd, char* finalRsp, char *buff, uint32_t buffSize, long timeout_ms)
Input	atCmd:AT command finalRsp: Expected return value with string buff：**AT** returns value all content pointer buffSize：AT returns value All content pointer size, at least 100 bytes timeout_ms：AT timeout
Output	None
Return vaule	0：success -1：failureOther：AT returns content length
NOTE	

3.3 UART

SIM7600E-H：Dual high-speed serial interfaceUART1 and UART2.Device nodes are UART1：/dev/ttyHS0（It can be configured as Bluetooth）and UART2：/dev/ttyHS1.All serial ports do not support the use of AT commands.

The specific parameters are as follows：

Parameters	Value ranges	Defaults	Setting method
Start bit	1bit	1bit	ioctl
Data bit	7bit,8bit	8bit	
Stop bit	1bit,2bit	1bit	
Check digit	Odd,Even,Space, None	None	
Baud rate	300,600,1200,2400,4800,9600,19200,38400,57600,115200, 230400,460800,921600,1000000,1152000,1500000,3000000,3200000,3686400	115200	ioctl
Single frame send size	512B	512B	
Single frame response delay	（1-10）ms （The time that Host needs to wait after each single frame is sent synchronously）	1ms(recommend)	
Maximum buffer	10KB	10KB	

SIM7600E-H Its PIN definition is as follows：

PIN No	PINNAME	Default Function	Pull	Wakeup Interrupt
6	SPI_CLK	UART1_RTS	B-PD	
7	SPI_MISO	UART1_RX	B-PD	
8	SPI_MOSI	UART1_TX	B-PD	
9	SPI_CS	UART1_CTS	B-PD	
66	RTS	UART2_RTS	B-PD	
67	CTS	UART2_CTS	B-PD	
68	RXD	UART2_RX	B-PD	
71	TXD	UART2_TX	B-PD	

3.4 GPIO

The following is a description of the SIM7600E-T GPIO controllable PIN:

PIN No	PINNAME	SYS GPIO No.	Default Function	Function 0	Function 1	Pull	Wakeup Interrupt
33	GPIO3	GPIO_1020	MIFI_POWER_EN	GPIO	MIFI_POWER_EN	B-PU	
34	GPIO6	GPIO_102	MIFI_SLEEP_CL	GPIO	MIFI_SLEEP_CL	B-P	

		3	K		K	D	
48	SD_DET	GPIO_26	GPIO	GPIO	SD_DET	B-PD	*
49	STATUS	GPIO_52	Status	GPIO	Status	B-PD	*
50	GPIO43	GPIO_36	MIFI_COEX	GPIO	MIFI_COEX	B-PD	
51	NETLIGHT	GPIO_18	NETLIGHT	GPIO	NETLIGHT	B-PD	
52	GPIO41	GPIO_79	BT_PCM	GPIO	BT_PCM	B-PD	*
54	FLIGHTMODE	GPIO_76	BT_PCM	GPIO	BT_PCM	B-PD	*
69	UART_RI	GPIO_50	GPIO(RI)	GPIO	GPIO	B-PD	
70	UART_DCD	GPIO_51	GPIO	GPIO	GPIO	B-PD	
72	UART_DTR	GPIO_74	GPIO(DTR)	GPIO	GPIO	B-PD	*
86	COEX3	GPIO_78	BT_PCM	GPIO	BT_PCM	B-PD	
87	GPIO77	GPIO_77	BT_PCM	GPIO	BT_PCM	B-PD	

Note：

- ➢ If it doesn't need the BT_PCM function, you can export the corresponding SYS GPIO No to using as GPIO.
- ➢ If the UART is configured to low speed serial ports, the CTS and RTS can be using as GPIO.
- ➢ The following is a description of the SIM7600E-H GPIO controllable PIN:
- ➢ The following is a description of the SIM7600E-H GPIO controllable PIN:

The following is a description of the SIM7600E-H GPIO controllable PIN:

PIN No	PINNAME	SYS GPIO No.	Default Function	Function 0	Function 1	Pull	Wakeup Interrupt
33	GPIO3	GPIO_1020	MIFI_POWER_EN	GPIO	MIFI_POWER_EN	B-PU	
34	GPIO6	GPIO_1023	MIFI_SLEEP_CLK	GPIO	MIFI_SLEEP_CLK	B-PD	
48	SD_DET	GPIO_26	GPIO	GPIO	SD_DET	B-PD	*
49	STATUS	GPIO_52	Status	GPIO	Status	B-PD	*
52	GPIO41	GPIO_79	BT	GPIO	BT	B-P	*

						D	
50	GPIO43	GPIO_36	MIFI_COEX	GPIO	MIFI_COEX	B-PD	
69	UART_RI	GPIO_50	GPIO(RI)	GPIO	GPIO	B-PD	
70	UART_DCD	GPIO_51	GPIO	GPIO	GPIO	B-PD	
72	UART_DTR	GPIO_74	GPIO(DTR)	GPIO	GPIO	B-PD	*
87	GPIO77	GPIO_77	BT	GPIO	BT	B-PD	

Note：
- PIN50, PIN52, and PIN87 boot phases are used as BOOT_CFG. So if you want to use these GPIOs for development, they only serve as outputs.
- PIN69 is fixed as the PIN pin for waking up the MCU and PIN72 is fixed as the PIN pin for the MCU wake-up module.

Instance:

1）GPIOConfiguration
About GPIO operation method:
Via the /sys/class/gpio/*** file node. This method requires attention to the correspondence between PIN No. and SYS GPIO No.
1)). Dynamically establish GPIO file node and write GPIO number to /sys/class/gpio/export file.For example using PIN50:
```
echo 36 > /sys/class/gpio/export
```
2)). Pull up or pull down GPIO and set GPIO as output first, then set output high and low.
For example pull up PIN50:
```
echo 36 >/sys/class/gpio/export
echo out> /sys/class/gpio/gpio36/direction
echo 1> /sys/class/gpio/gpio36/value
```
3)). Read GPIO-first set GPIO as input and then read the status.
```
  echo in> /sys/class/gpio/gpio36/direction
  cat /sys/class/gpio/gpio36/value
```

2） GPIO Interrupt configuration
Methods:
```
    echo "in" > /sys/class/gpio/gpio36/direction
    echo "both" > /sys/class/gpio/gpio36/edge
```
The state of the poll value node is required to trigger the response in real time.

3.5 ADC

The module provides two ADC analog-to-digital conversion interfaces corresponding to the module PIN46.PIN47. They correspond to ADC2 and ADC1, respectively. The developer can read the corresponding device file to obtain the voltage value.

Interface	int read_adc(int ch)
Input	ch :aisle
Output	None
Return value	0：success -1：failure
NOTE	

3.6 I2C

SIM7600E-H：SDA and SCL correspond to the PIN55 and PIN56 of the module, respectively. The developer first opens the /dev/i2c-5 devices to obtain the handle, and then calls the read-write function to operate the device.

3.6.1 Write I2C interface

Interface	int sim_i2c_write(uint8_t slave_address, uint16_t reg, uint8_t *buf, int len)	
Input	slave_address	Device slave address
	reg	Device register
	buf	Data bufferto be written
	len	Data length
Output	None	
Return value	0：success -1：failure Other：data length	
Interface	int sim_i2c_write(uint8_t slave_address, uint16_t reg, uint8_t *buf, int len)	

3.6.2 Read I2C interface

Interface	int sim_i2c_read(uint8_t slave_address, uint16_t reg, uint8_t *buf, int len)	
Input	slave_address	Device slave address

	reg Device register len Data length
Output	buf Data bufferto be read
Return value	0：success -1：failure
NOTE	

3.7 SD Card/EMMC flash

Through the SDIO PIN (21-26), external SD Card or EMMC, the default mount / etc / card shortcut is / sdcard. Partitioning and formatting need to be done once on the production line and using the AT command AT + CFDISK. After the system default mount only the first partition is / sdcard, other partitions need to use the command to do mount in the APP.

3.7.1 Partition

Using AT command ‘AT+CFDISK=<num>[,<size>]’ to finish partition.

3.7.2 Format

After the first partition, use the AT command ‘AT+CFDISK’ to format.

3.7.3 Mount

The module will automatically mount /dev/mmcblk0p1 to /media/card. For other partitions, users need to mount.
For example mounting /dev/mmcblk0p2 to /mnt：
mount–t auto /dev/mmcblk0p2 /mnt

3.7.4 CFDISK Command

AT+CFDISK

Test Command **AT+ CFDISK=?**	Response a)If successfully: **+CFDISK: (1-4)[…]** **OK** b)If failed: **ERROR**
Read Command **AT+ CFDISK?**	Response a)If successfully: **+CFDISK: <num>,<size>** **OK** b)If failed: **ERROR**
Write Command **AT+ CFDISK=<num>[,<size>,…]**	Response a)If successfully: **OK** b)If failed: **ERROR**
Write Command(Formatting all partitions) **AT+ CFDISK**	Response a)If successfully: **OK** b)If failed: **ERROR**

Defined values

<num>	Partition number
<size>	Partition size.The unit is KB. NOTE:The last partition size does not need to be set. The size of the last partition is the size of the disk remaining.

Examples

```
AT+CFDISK=?
OK+CFDISK: (1-4)[...]

OK
AT+CFDISK=4,50000,50000,50000
OK
AT+CFDISK
OK
AT+CFDISK?
+CFDISK: 1,50040
+CFDISK: 2,50048
+CFDISK: 3,50048
+CFDISK: 4,3708288

OK
```

3.8 UIM

Customers can use UIM's related functions to obtain some information about the SIM card, such as ICCID and IMSI.

3.8.1 Check SIM card status

Interface	int getSimCardStatus(SimCard_Status_type *simStatus);
Input	None
Output	simStatus card_status: 0: No SIM card detected 1: SIM card detected2: Unknown mistake app_type: types ofSIM card1: SIM 2: USIM 3: RUIM 4: CSIM 5: ISIM app_state:1: detected 2: pin block 3: puk block 4: person check 5:Permanently blocked 6: illgeal pin_state:1: unknown 2: Enabled and not verified 3: Enabled and verified 4: Disabled 5: Blocked 6:Permanently blocked) pin_retries:The number of remaining solutions pin puk_retries:The number of remaining solutions puk
Return value	0：success -1：failure
NOTE	

3.8.2 Query SIM card ICCID

Interface	int get_iccid(char *piccid);
Input	None
Output	pIccid： ICCID
Return value	0：success -1：failure
NOTE	pIccid is a pointer or an array of at least 21 bytes in length and the content has been initialized to 0.

3.8.3 Query SIM card IMSI

Interface	int get_imsi(char *imsi);
Input	None
Output	imsi： IMSI
Return value	0：success -1：failure
NOTE	Imsi is a pointer or an array of at least 16 bytes in length, and the content has been initialized to 0.

3.9 SMS

Send and receive SMS.

3.9.1 SMS initialization

Interface	int sms_init(sms_ind_cb_fcn sms_ind_cb);
Input	sms_ind_cb：For receiving SMS messages
Output	None
Return value	0：success -1：failure
NOTE	After initialization to perform another SMS operation.

3.9.2 Set receive SMS format

Interface	void sms_set_format(sms_format format);
Input	format: 1:ASCII not use 2:UTF8 default 3:UCS2
Output	None
Return value	None
NOTE	The format is ASCII for English SMS, the default format is UTF8 for Chines SMS, you can call this API to swich the format to UCS2.

3.9.3 Send messages

Interface	int send_message(int smsMode, char *phoneNumber, unsigned char *content, int content_len);
Input	smsMode: 1:ASCII 2:UTF8 3:UCS2 phoneNumber:Destination of phone number content：message content content_len：length of content
Output	None
Return value	0：success -1：failure
NOTE	

3.9.4 Callback function handles message reception

Interface	void process_simcom_ind_message(simcom_event_e event,void *cb_usr_data);
Input	None

Output	event=SIMCOM_EVENT_SMS_PP_IND
	cb_usr_data:Corresponding structure
	typedef struct {
	    boolean is_concate;
	    uint8 msg_ref;
	    uint8 seq_num;
	    uint8 total_sm;
	    sms_format format;
	    int message_len;
	    char message_content[SMS_MAX_CONTENT_LENGTH];
	    char source_address[SMS_MAX_ADDRESS_LENGTH];
	}sms_info_type;
	is_concate: long message flag, 1 indicates long message
	msg_ref: message ID for long message, the same message ID indicates the same one message.
	seq_num: the messge index in the whole long message
	total_num: the total message number for a whole long message
	format: 1:ASCII 2:UTF8 3: UCS2
	message_len: message length
	message_content：message content
	source_address：phone number

Return value	None
NOTE	This function is passed in Init as a parameter.

3.10 Voice Call

Make and receive voice calls while monitoring the status of voice calls.

3.10.1 Phone initialization

Interface	int Init();
Input	None
Output	None
Return value	0：success -1：failure
NOTE	After initialization to perform other operations on the phone.

3.10.2 Dial number

Interface	int voice_dial(char *phoneNum);
Input	Phone Num：The number you need to dial
Output	None
Return value	0：success -1：failure
NOTE	

3.10.3 Handle current call

Interface	int voice_mtcall_process(voice_call_option option, unsigned char call_id);
Input	Option： 1：Hang up the phone, only one call for the current 2：Answer, only one call for the current 3：Keep the call, keep the current call and pick up the new call 4：Hang up all calls 5：Hang up hold call 6：Hang up current call Call_id:optionEquivalent to 1 or 2
Output	None
Return value	0：success -1：failure

NOTE	

3.10.4 Get the status of the specified call

Interface	int get_call_info(uint8_t callid, call_info2_type *pcall_info);
Input	callid Non-zero value
Output	pcall_infoStructure contains parameters： 1 call_id, - same as input 2 call_state： 1：Start a call 2：Incoming call 3：Call establishment 5：Bell 6：Call holding 7：Call waiting 8：Hanging 9：Hang up 3 Direction： 1：Exhale 2：Call in 4 Number
Return value	0：success -1：failure
NOTE	

3.10.5 Get all call states

Interface	int get_all_call_info(call_info_type *pcall_info_list);
Input	
Output	pcall_info_list Structure contains parameters： 1. call_num - There are currently several calls 2. call_info： 2.1 call_id, - Same as input 2.2 call_state： 1：Start a call 2：Incoming call 3：Call establishment 5：Bell 6：Call holding 7：Call waiting

	8：Hanging 9：Hang up 2.3 Direction： 1：Exhale 2：Call in 2.4 Number
Return value	0：success -1：failure
NOTE	

3.10.6 Callback

Interface	void process_simcom_ind_message(simcom_event_e event,void *cb_usr_data)
Input	
Output	Event = SIMCOM_EVENT_VOICE_CALL_IND cb_usr_dataStructure contains parameters： 1. call_num - There are currently several calls 2. call_info： 2.1 call_id, - Same as input 2.2 call_state： 1：Start a call 2：Incoming call 3：Call establishment 5：Bell 6：Call holding 7：Call waiting 8：Hanging 9：Hang up 2.3 Direction： 1：Exhale 2：Call in 2.4 Number
Return value	0：success -1：failure
NOTE	

3.11 NAS

NAS interface is mainly used to obtain some information about the status of the network, such as registering network types and signals.

3.11.1 Query registration network status

Interface	int get_NetworkType(nas_serving_system_type_v01 *serving_system);
Input	None
Output	serving_system registration_state： - 0x00 –unregistered - 0x01 –registered - 0x02 –Search network - 0x03 –Registration refused - 0x04 –Unknown state cs_attach_state：CS domain status, whether the attachment is successful or not - 0x00 –Unknown - 0x01 –Attached successfully - 0x02 –Attached unsuccessfully ps_attach_state：PS domain status, whether it is attached successfully determines whether it can dial up the Internet. - 0x00 --Unknown - 0x01 --Attached successfully（Note：Success does not mean that you can access the Internet, but also need to dial） - 0x02 --Attached unsuccessfully selected_network;network types - 0x00 –Unknown - 0x01 –3GPP2 - 0x02 –3GPP radio_if_len：Number of registered networks（The effective number is one. However, there may be two for telecom cards and they may register on LTE and CDMA at the same time.） radio_if[];Registered network array -0x00 –No service -0x01 –CDMA_1X -0x02 –CDMA_EVDO -0x04 -- GSM -0x05 -- UMTS -0x08 -- LTE
Return value	0：success -1：failure
NOTE	

3.11.2 Query signal

Interface	int get_SignalStrength(int * signalStrength);
Input	None
Output	signalStrength：Signal value
Return value	0：success -1：failure
NOTE	

3.12 WDS

Wds interface is mainly used to query and set APN

3.12.1 Query APN

Interface	int get_apnInfo(int profile_index, int* pdp_type, char* apn_str, char *username, char *password);
Input	None
Output	profile_index： APN index pdp_type： pdp types - 0 -- IPv4 - 2 -- IPv6 - 3 -- IPv4 and IPv6 apn_str： APN string username： password：
Return value	0：success -1：failure
NOTE	

3.12.2 Set APN

Interface	int set_apnInfo(int profile_index, int pdp_type, char* apn_str, char *username, char *password);
Input	profile_index：（1~16） : APN index to be queried

	pdp_type： pdp types，General choice 0 - 0 -- IPv4 - 2 -- IPv6 - 3 -- IPv4 and IPv6 apn_str： APN string username：If not, set to NULL password：If not, set to NULL
Output	None
Return value	0：success -1：failure
NOTE	

3.13 Data Call

It is used to complete setting/inquiry for APN, dialing username and password under CDMA network. With a connection to the network, disconnect the network connection and other functions. Up to 8 links are supported at the same time. The data link used by the APP suggests starting with the 7th profile.

3.13.1 Initialize the network

Interface	int Init();
Input	None
Output	None
Return value	None
NOTE	

3.13.2 Create data link

Interface	int start_dataCall(int profile);
Input	profile: profile serial number
Output	None
Return value	0：success-1：failure
NOTE	

3.13.3 Get data link parameters

Interface	int get_datacall_info(int profile, datacall_info_type *pcallinfo);
Input	profile: profile serial number

Output	datacall_info_type *pcallinfo
	typedef struct {
	    dsi_hndl_t handle; //Link handle
	    enum app_tech_e tech; //Use standard
	    int family; //IP type
	    int profile; //profile serial
	    datacall_status_type status; //Current link status
	    char if_name[32]; //Interface name
	    char ip_str[16]; //IP address
	    char pri_dns_str[16];//primaryDNS
	    char sec_dns_str[16];//secondaryDNS
	    char gw_str[16]; //Gateway
	    unsigned int mask; //Mask
	}datacall_info_type;

Return value	0：success -1：failure
NOTE	

3.13.4 Release network resources

Interface	int DeInit();
Input	None
Output	None
Return value	None
NOTE	

3.14 GNSS

It enables the opening and closing of GNSS, the use of XTRA, and the output of NMEA and latitude and longitude.

3.14.1 Initialize gnss

Interface	static boolean gps_init();
Input	None
Output	None
Return value	TRUE: success FALE: failure
NOTE	This function must be executed before all other operations.

3.14.2 EnableXTRA

Interface	static void gps_xtra_enable();
Input	None
Output	None
Return Value	None
NOTE	If you need to use XTRA you must be ENABLE before turning on GNSS, and need to be connected to the module.

3.14.3 Prohibit XTRA

Interface	static void gps_xtra_disable();
Input	None
Output	None
Return value	None
NOTE	

3.14.4 GNSS Cold Start

Interface	static boolean gps_coldstart();
Input	None
Output	None
Return value	TRUE: success FALE: failure
NOTE	When the signal is good, the cold start is about 35 seconds

3.14.5 GNSS Hot Start

Interface	static boolean gps_hotstart();
Input	None
Output	None
Return value	TRUE: success FALE: failure
NOTE	

3.14.6 GPS Stop

Interface	static boolean gps_stop();
Input	None
Output	None
Return value	TRUE: success FALE: failure
NOTE	

3.14.7 Callback function output brief location information

Interface	void process_simcom_ind_message(simcom_event_e event,void *cb_usr_data)
Input	None
Output	None
Return value	

NOTE	After turning on GNSS and positioning it will output latitude, longitude, time, height, speed and etc.
	Event = SIMCOM_EVENT_LOC_IND
	cb_usr_datastructure type：
	
	typedef struct {
	    double latitude;
	    double longitude;
	    double altitude;
	 float speed;
	    float bearing;
	float accuracy;
	//time[0] = year-2000
	//time[1] = month

	//time[2] = day
	//time[3] = time
	//time[4] = minute
	//time[5] = second
	    uint8_t time[6];
	} GpsInfo;

3.14.8 Callback function output NMEA statement

Interface	void process_simcom_ind_message(simcom_event_e event,void *cb_usr_data)
Input	None
Output	None
Return value	None
NOTE	Turn on NMEA statements every second after opening GNSS Event = SIMCOM_EVENT_NMEA_IND cb_usr_data type char

3.15 WIFI

Description of function:
Recommend customers choose W58 module, which encapsulates Qualcomm QCA9377 WIFI chip.
Specifications of W58 module：

- SDIO 3.0 interface，RelatedPIN（27-32）
- 20MHz/40MHz（2.4GHz）
- As AP mode, it connects 30 nodes.
- 1X1 Single antenna design so has low cost solution.

Software function：

- Support WIFI AP/station mixed mode
- Supports communication between devices connected to WIFI hotspots and connected Module devices
- Support hidden WIFI hotspot function
- Provides AT commands for configuring WIFI and routing protocols

Special Instructions:

- get_wifi_mode and set_wifi_mode apply toW58, W58L cannot use.
- sta_init and get_sta_status apply to W58L, W58 cannot use.

3.15.1 Get current WIFI mode settings (for W58)

Interface	wifi_mode_type get_wifi_mode()
Input	None
Output	None
Return value	wifi_mode_type： 0：Single AP mode 1：Dual AP mode 2：AP+STA mode
NOTE	

3.15.2 Set WIFI mode (for W58)

There are three modes of WIFI：
0：Single AP modeit provides a hot spot for mobile Internet
1：Dual AP modeit provides two hotspots for mobile Internet
2：AP+STA mode it provides a hotspot for mobile Internet access, while providing a WIFI client to allow the module to connect to other routers

In general, if the WIFI mode of the device design is fixed, it is better to preset the WIFI mode.
Preset WIFI mode methods：

a. Modify the nodes in sim_open_sdk/sim_usrfs/mobileap_cfg.xml and sim_open_sdk/sim_rootfs/etc/default_mobileap_cfg.xml:
b. <WlanMode>AP-AP</WlanMode> value：

AP mode： AP
Dual APmode：AP-AP
AP+STA mode： AP-STA

c. modify sim_open_sdk/sim_usrfs/mobileap_enable_cfgvalue

AP mode：0
AT+AP mode： 1
AP+STA mode： 2
Note：aand b models must be consistent.
If the WIFI mode of the device needs to be dynamically switched according to the scenario, you can call the following interface to adjust：

Interface	wifi_mode_type set_wifi_mode(wifi_mode_type mode)
Input	mode：wifi_mode_type 0：single AP mode 1：Dual AP mode

	2：AP+STA mode
Output	None
Return value	Boolean type： TRUE or FALSE
NOTE	

3.15.3 WIFI Power

Interface	boolean wifi_power(int act)
Input	act：int type 1：open 0：close
Output	None
Return value	Boolean type： TRUE or FALSE
NOTE	

3.15.4 Get WIFI status

Interface	boolean get_wifi_status(uint8 *flag)
Input	None
Output	flag：uint8 *type： 1：open 0：close
Return value	Boolean type： TRUE or FALSE
NOTE	

3.15.5 Set WIFI hotspot name

Interface	oolean set_ssid(char *SSID, ap_index_type ap_index)
Input	1. SSID：Char*type Hotpot name string 2. ap_index：ap_index_type AP mode，1

	AP-AP mode，0 or 1 STA-AP mode，2
Output	
Return value	Boolean type： TRUE or FALSE
NOTE	

3.15.6 Get WIFI hotspot name

Interface	boolean get_ssid(char *str_SSID, ap_index_type ap_index)
Input	ap_index：ap_index_type AP mode，1 AP-AP mode，0 or 1 STA-AP mode，2
Output	SSID：Char*type Hotpot name string
Return value	Boolean type： TRUE or FALSE
NOTE	

3.15.7 Set AP auth type, encrypt mode, password

Interface	boolean set_auth(char *str_pwd, int auth_type, int encrypt_mode, ap_index_type ap_index)
Input	1. str_pwd：char*type password 2. auth_type：int type auth type 3. encrypt_mode：int type encrypt mode 4. ap_index：ap_index_type AP mode，1 AP-AP mode，0 or 1 STA-AP mode，2
Output	None
Return value	Boolean type： TRUE or FALSE
NOTE	1.When the auth type input is 0 or 1, the input value of the encrypt mode is 0 or 1；

	2.When the auth type input is 2, the input value of encrypt mode can only be 1；
	3.When the input value of the auth type is greater than 3, the input value of the encrypt mode must be greater than or equal to 2；
	4.When the encrypt mode input is 0, it does not need to enter the password；
	5.When the encrypt mode input is 1, the password must be entered.
	The format of the password input must satisfy: an ASCIIencoded string of length 5 or 13 or a hexadecimal encoded string of length 10 or 26.
	6.When the input value of the encrypt mode is greater than or equal to 2, the password must be input.
	The format of the password input must satisfy: an ASCII encoded string of length 8 to 63 or a hexadecimal encoded string of length 64.
	
	Default value :
	int authType = 5;
	int encryptMode = 4;
	ap_index_type: 0-> ap
	0-> ap & ap
	2-> ap & sta

3.15.8 Get AP auth type, encrypt mode, password

Interface	boolean get_auth(int *auth_type_ptr, int *encrypt_mode_ptr, char *pwd_str, ap_index_type ap_index)
Input	ap_index：ap_index_type AP mode，1 AP-AP mode，0 or 1 STA-AP mode，2
Output	1. authType：int*type 2.encrypt_mode_ptr：int*type 3. pwd_str：char*type
Return value	Boolean type： TRUE or FALSE
NOTE	

3.15.9 Set up WIFI broadcast switch

Interface	boolean set_bcast(int broadcast, ap_index_type ap_index)
Input	1. Broadcast：int type

	2. ap_index：ap_index_type AP mode，1 AP-AP mode，0 or 1 STA-AP mode，2
Output	None
Return value	Boolean type： TRUE or FALSE
NOTE	

3.15.10 Get WIFI broadcast settings

Interface	boolean get_bcast(int *broadcast,ap_index_type ap_index)
Input	ap_index：ap_index_type AP mode，1 AP-AP mode，0 or 1 STA-AP mode，2
Output	Broadcast：int type
Return value	Boolean type： TRUE or FALSE
NOTE	

3.15.11 Get DHCP settings

Interface	boolean get_dhcp(char *host_ip_str, char *start_ip_str, char *end_ip_str, char *time_str)
Input	None
Output	1. host_ip_str：host address 2. start_ip_str：start address 3. end_ip_str：end address 4. time_str：Authorization time
Return value	Boolean type： TRUE or FALSE
NOTE	

3.15.12 Get the number of connected clients

Interface	int get_client_count(ap_index_type ap_index)
Input	ap_index：ap_index_type AP mode，1 AP-AP mode，0 or 1 STA-AP mode，2
Output	None
Return value	Int type，Number of clients connected to AP
NOTE	

3.15.13 Get IP Address in STA Mode

Interface	Booleanget_sta_ip(char *ip_str, int len)
Input	None
Output	Ip_str：IP address
Return value	Boolean type： TRUE or FALSE
NOTE	

3.15.14 Get WIFI MACaddress

Interface	boolean get_mac_addr(char *mac_addr, ap_index_type ap_index)
Input	ap_index：ap_index_type AP mode，1 AP-AP mode，0 or 1 STA-AP mode，2
Output	mac_addr：char*type，mac address The first one is HOST MAC
Return value	Boolean type： TRUE or FALSE
NOTE	

3.15.15 Set IP obtained after the STA connects to the external hotspot

Interface	boolean get_sta_ip(char *ip_str, int len)
Input	None
Output	Ip_str：ip address

Return value	Boolean type： TRUE or FALSE
NOTE	

3.15.16 SetSTA's SSID and Password for Connecting to an External AP

Interface	boolean set_sta_cfg(char *ssid_str, char *psk_value)
Input	1. ssid_str：hotspot name 2. psk_value：password
Output	None
Return value	Boolean type： TRUE or FALSE
NOTE	

3.15.17 Get SSID and password set by the STA

Interface	boolean get_sta_cfg(char *ssid_str, char *psk_value)
Input	None
Output	1. ssid_str：hotspot name 2. psk_value：password
Return value	Boolean type： TRUE or FALSE
NOTE	

3.15.18 WIFI sta Scan available hotspots

Interface	boolean sta_scan(char *list_str)
Input	None
Output	list_str：list of available hotspots
Return value	Boolean type： TRUE or FALSE
NOTE	

3.15.19 Set username and password for dialing in cdma mode

Interface	boolean set_user_name_pwd(char *sz_usrname, char *sz_usrpwd)
Input	1. sz_usrname：username 2. sz_usrpwd：password
Output	None
Return value	Boolean type： TRUE or FALSE
NOTE	

3.15.20 Get username and passwordfor dialing in cdma mode

Interface	boolean get_user_name_pwd(char *sz_usrname, int len_name, char *sz_usrpwd, int len_pwd)
Input	None
Output	1. sz_usrname：username 2. sz_usrpwd：password
Return value	Boolean type： TRUE or FALSE
NOTE	

3.15.21 Get network status

Interface	boolean get_net_status(char *net_enable_str)
Input	None
Output	net_enable_str：network status
Return value	Boolean type： TRUE or FALSE
NOTE	

3.15.22 Restore wifi settings

Interface	void restore_wifi()
Input	None

Output	None
Return value	None
NOTE	

3.15.23 Set hotpot name, auth type, encrypt mode, password

Interface	boolean set_ssid_and_auth(char *SSID, char *str_pwd, int auth_type, int encrypt_mode, ap_index_type ap_index)
Input	1. SSID：Char* type Hotpot name string 2.str_pwd：char* type Password 3.auth_type：int type Auth type 4.encrypt_mode：int type Encrypt mode 5.ap_index：ap_index_type type AP mode，the value is 1 AP-AP mode，the value is 0 or 1 STA-AP mode，the value is 2
Output	None
Return value	Boolean type： TRUE or FALSE
Interface	boolean set_ssid_and_auth(char *SSID, char *str_pwd, int auth_type, int encrypt_mode, ap_index_type ap_index)

3.15.24 Open/close STA mode (for W58L)

Interface	boolean sta_init(int sta_enable)
Input	sta_enable: int type 0–close STA mode 1-open STA mode
Output	None
Return value	Boolean type： TRUE or FALSE
NOTE	

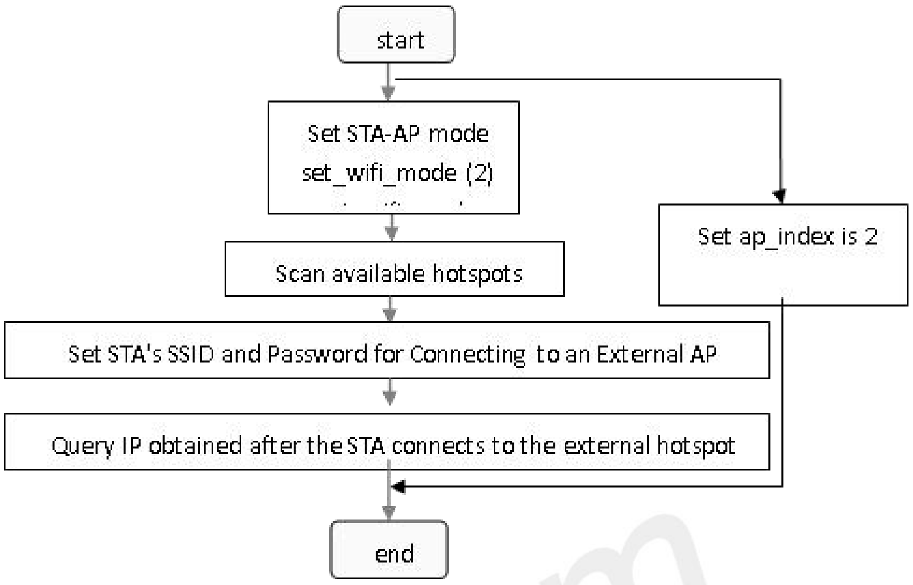
3.15.25 Get the state of STA mode (for W58L)

Interface	boolean get_sta_status(uint8 *flag)
Input	None
Output	Flag: uint8 * type 1 – opened 0 – closed
Return value	Boolean type： TRUE or FALSE
NOTE	

3.15.26 Get the operation result code

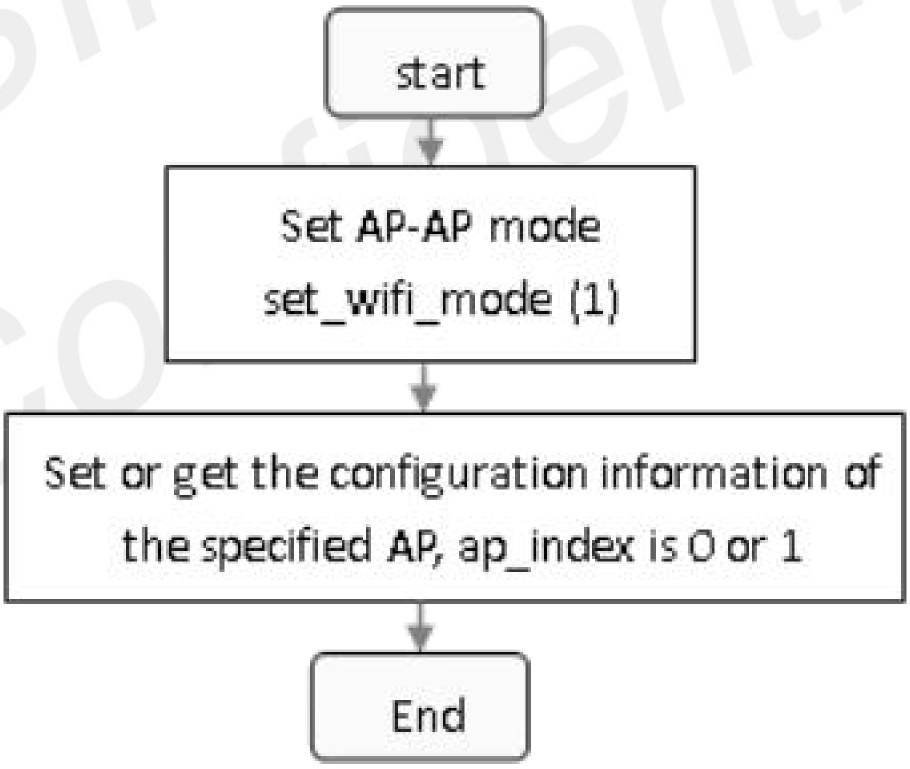
Interface	Uint8wifi_get_err_code()
Input	None
output	None
Return value	Uint8 type： 0– success 1– invalid parameter 2–malloc memory error 3–send/receive message error 4–open file error 5–read/write file error 6–invalid return value 7–invalid AP ID 8–get information error 9–not W58L 10–other error
NOTE	The result code retain the last operation result. You could call the API to get the current operation result code if call a API which return failed.

3.15.27 Description of STA-AP function


start
Set STA-AP mode
set_wifi_mode (2)
Scan available hotspots
Set ap_index is 2
Set STA's SSID and Password for Connecting to an External AP
Query IP obtained after the STA connects to the external hotspot
end

3.15.28 Description of AP-AP function

AP-AP function description refer to the following flow chart：


start
Set AP-AP mode
set_wifi_mode (1)
Set or get the configuration information of the specified AP, ap_index is 0 or 1
End

3.16 SPI

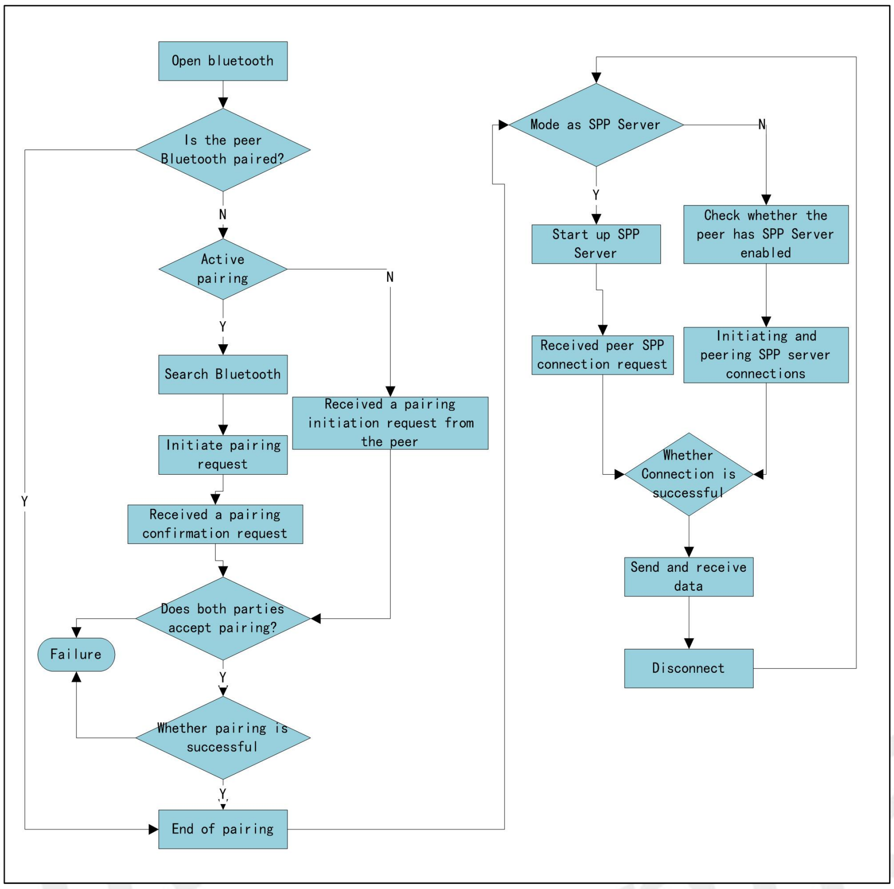
SIM7600E-H:SPI interface cannot be used on the standard version. You need to update the version that supports SPI. You can configure UART1 or UART2 as SPI. For details, please refer to the SIM7600 Open Linux UART & SPI Documentation. The module's SPI supports only the Master mode and does not support the Slave mode. After the SPI version is started, the /dev/spidev2.0 device node is generated. The default SPI interface can only connect to a single peripheral device.

3.17 USB OTG

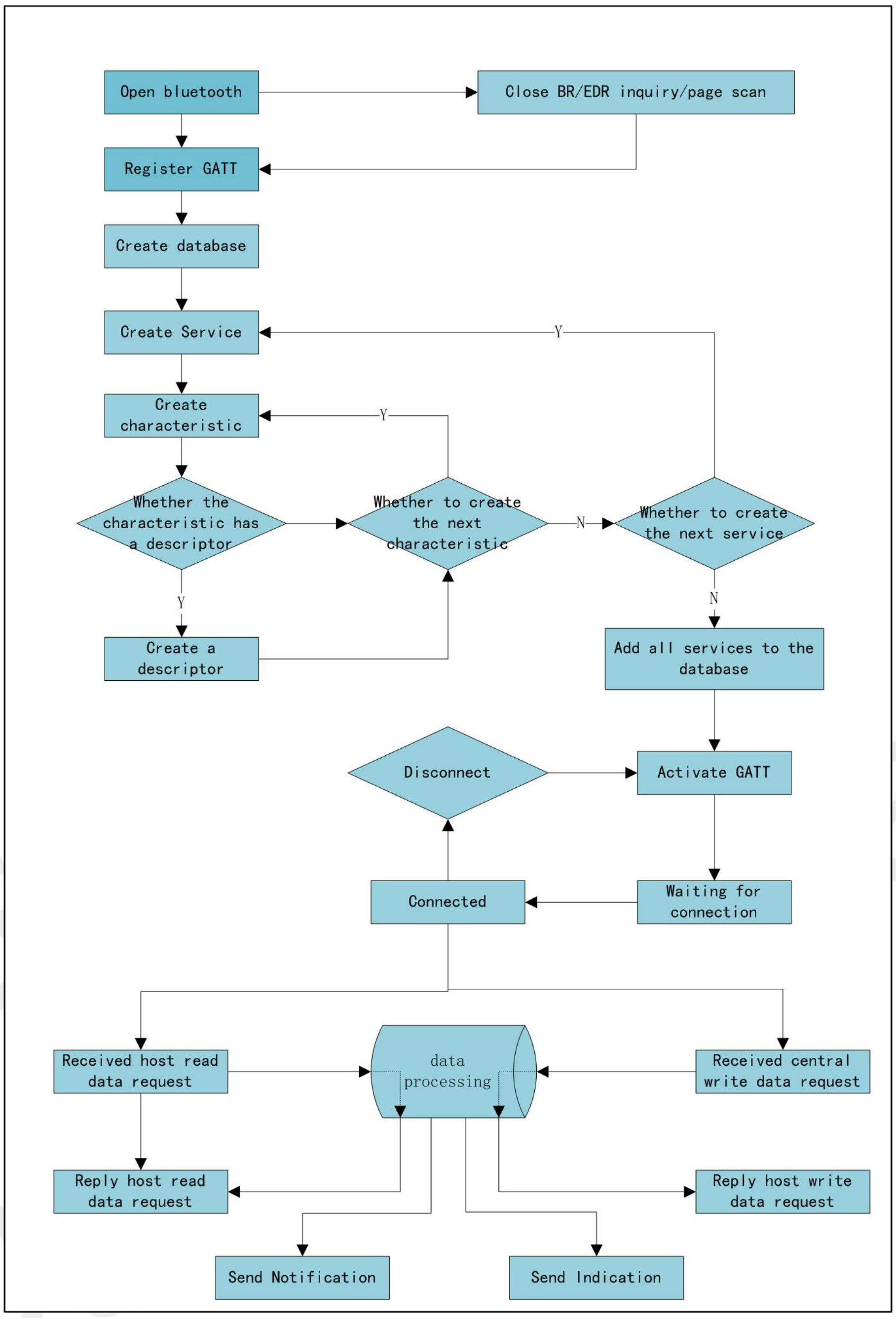
It requires peripheral circuit design. Currently it only supports U disk. SDK is default support.

3.18 Bluetooth

It is recommended that customers select the W58 module and package the Qualcomm QCA9377 Bluetooth chip.
This chapter describes how to implement SPP data sending and receiving, GATT data transmission and etc. by calling the Bluetooth API.
SPP data send and receive data. First turn on Bluetooth. If you do not pair with the peer Bluetooth, you need to pair first. After the pairing is completed, start the SPP server for the peer Bluetooth connection or directly connect the peer Bluetooth SPP server to complete the data transfer.


Open bluetooth
Is the peer Bluetooth paired?
N
Active pairing
Y
N
Search Bluetooth
Initiate pairing request
Received a pairing initiation request from the peer
Received a pairing confirmation request
Does both parties accept pairing?
Failure
Y
Whether pairing is successful
Y
End of pairing
Y
Mode as SPP Server
Y
N
Start up SPP Server
Check whether the peer has SPP Server enabled
Received peer SPP connection request
Initiating and peering SPP server connections
Whether Connection is successful
Send and receive data
Disconnect

Bluetooth SPP pairing and SPP data transceiver flow chart


Open bluetooth
Close BR/EDR inquiry/page scan
Register GATT
Create database
Create Service
Y
Create characteristic
Y
Whether the characteristic has a descriptor
Whether to create the next characteristic
N
Whether to create the next service
Y
Create a descriptor
N
Add all services to the database
Disconnect
Activate GATT
Connected
Waiting for connection
Received host read data request
data processing
Received central write data request
Reply host read data request
Reply host write data request
Send Notification
Send Indication

GATT Data transceiver flow chart

The following is the relevant API interface in SPP function：

3.18.1 Bluetooth interface initialization

Interface	int Init(bt_ind_cb_fcn handle);
Input	typedef void (*bt_ind_cb_fcn)(void *pData); callback
Output	None

Return value	0：success -1： failure
NOTE	

3.18.2 Bluetooth power

Interface	int simcom_bt_power_on(int bPower);
Input	bPower 0: close 1: open
Output	None
Return value	0：success -1： failure
NOTE	

3.18.3 Get paired list

Interface	int get_bonded_device(void);
Input	None
Output	Return the paired list in the callback function with the messageBT_BONDED_COMMAND The specific parameters of the message refer to callback function description
Return value	0：success -1：failure
NOTE	

3.18.4 Search Bluetooth

Interface	int search_devices(int subcmd, int mode, int timeout);
Input	None
Output	The search list is returned in the callback function with the message BT_SEARCH_COMMAND The specific parameters of the message refer to callback function description
Return value	0：success -1：failure
NOTE	

3.18.5 Bluetooth pairing

Interface	int search_devices(int subcmd, int mode, int timeout);
Input	index：Search index number in Bluetooth list
Output	The pairing confirmed notification is returned in the callback function with the message BT_ACCEPT_COMMAND or return the pairing result with message BT_BOND_COMMAND The specific parameters of the message refer to callback function description.
Return value	0：success -1：failure
NOTE	

3.18.6 Pairing confirmation request

Interface	Init registered callback function
Input	None
Output	Report BT_ACCEPT_COMMAND message The specific parameters of the message refer to callback function description.
Return value	
NOTE	

3.18.7 Accept pairing

Interface	int accept_bond(int isAccept, accept_mode mode);
Input	isAccept：Whether to receive pairing mode：Pairing mode ACCEPT_MODE_NONE = 0 mode can be ignored when not accepted. ACCEPT_MODE_COMPARE = 1 Compare pairing password ACCEPT_MODE_PASSKEY = 2 Need to enter password ACCEPT_MODE_REBOND = 3 Re-pairing ACCEPT_MODE_NOTIFICATION = 4Notice only, no reply required ACCEPT_MODE_JUSTWORK = 5Pairing directly CCEPT_MODE_PINCODE = 6PIN pairing mode
Output	The pairing result is returned in the callback function with the message BT_BOND_COMMAND The specific parameters of the message refer to callback function

	description.
Return value	0：success -1：failure
NOTE	This interface is executed when the callback function BT_ACCEPT_COMMAND message is received.

3.18.8 Pairing results

Interface	Init registered callback function
Input	None
Output	Report BT_BOND_COMMAND message The specific parameters of the message refer to callback function description.
Return value	
NOTE	

3.18.9 Open SPP Server

Interface	int spp_server(int bActive);
Input	bActive： 1：open 0：close
Output	Return value
Return value	0：success -1：failure
NOTE	Determine whether to open or close successfully by the return value

3.18.10 Query whether the peer Bluetooth is enabled with SPP Server

Interface	int bt_get_profile(int index, int *isSupportSPP);
Input	index：Paired list index number
Output	isSupportSPP： 1：opened 0：Unopened
Return value	0：success -1：failure
NOTE	

3.18.11 Initiate SPP connection

Interface	int spp_conncet_device(int index);
Input	index：Paired list index number
Output	Return value
Return value	0：success　　-1：failure
NOTE	Determine whether to open or close successfully by the return value

3.18.12 Cut SPP connection

Interface	int spp_disconncet_device(void);
Input	None
Output	Return value
Return value	0：success　　-1：failure
NOTE	Determine whether to open or close successfully by the return value

3.18.13 Non-active connect SPP

Interface	Init registered callback function
Input	None
Output	Report BT_CONNECT_COMMANDmessage The specific parameters of the message refer to callback function description.
Return value	
NOTE	

3.18.14 Non-active cut SPP connections

Interface	Init registered callback function
Input	None
Output	Report BT_DISCONNECT_COMMAND message The specific parameters of the message refer to callback function description.
Return value	

NOTE	

3.18.15 Query the status of the local SPP server

Interface	int bt_get_server_status(int *isActived);
Input	None
Output	IsActived 0: close 1：open
Return value	0：success -1：failure
NOTE	

3.18.16 Query SPP connection status

Interface	int bt_get_connect_status(int *isConnected);
Input	None
Output	isConnected： 0：disconnect 1：connect
Return value	0：success -1：failure
NOTE	

3.18.17 SPP send data

Interface	int spp_send_data(uint8_t *data, uint16_t data_len);
Input	Data data data_len length of data
Output	Return value
Return value	0：success -1：failure
NOTE	

3.18.18 SPP receive data

Interface	Init registered callback function
Input	None
Output	Report BT_SPP_RECV_COMMAND message

	The specific parameters of the message refer to callback function description.
Return value	
NOTE	

3.18.19 Query local Bluetooth name and MAC address

Interface	int host_device(char *device_name, SIMCOM_BT_ADDR_T *paddr);
Input	None
Output	device_namename of bluetooth paddr MAC address
Return value	0：success -1：failure
NOTE	This interface is an overloaded function. Parameters are shown in this section as queries. If there is only one char * parameterset the Bluetooth address.

3.18.20 Set local Bluetooth name

Interface	int host_device(char *device_name);
Input	device_name name of bluetooth
Output	None
Return value	0： success -1：failure
NOTE	This interface is overloaded and parameters settings as shown in this section.

3.18.21 Set PIN code

Interface	int pin_set(char *pincode);
Input	pincode： pin code4~16 digits Arabic string
Output	None
Return value	0：success -1：failure
NOTE	The PIN code is a relatively unsafe mode that is easy to crack so it is not recommended. It is turned off by default. If the customer has special requirements, please refer to the demo start part of the source code to modify the Bluetooth boot parameters.

3.18.22 Read PIN code

Interface	int pin_read(char *pincode);
Input	None
Output	pincode
Return value	0：success -1：failure
NOTE	

3.18.23 Set BR/EDR inquiry scan and page scan

Interface	int bt_set_scan_enable(int inquiry_scan_status, int page_scan_status);
Input	inquiry_scan_status 0: close 1：open page_scan_status 0: close 1：open
Output	None
Return value	0：success -1：failure
NOTE	inquiry_scan：If closed, other Bluetooth devices will not find this Bluetooth page_scan：If closed, other Bluetooth will not be able to connect to this Bluetooth

3.18.24 Query BR/EDR inquiry scan and page scan settings

Interface	int bt_set_scan_enable(int inquiry_scan_status, int page_scan_status);
Input	None
Output	inquiry_scan_status 0: close 1：open page_scan_status 0: close 1：open
Return value	0：success -1：failure
NOTE	inquiry_scan：If closed, other Bluetooth devices will not find this Bluetooth page_scan：If closed, other Bluetooth will not be able to connect to this Bluetooth

3.18.25 GATT registration

Interface	int bt_gatt_register(int bRegister);
Input	bRegister:　　1：register　　0：Unregister
Output	None
Return value	0：success　　-1：failure
NOTE	

3.18.26 Create database

Interface	int bt_gatt_createdatabase(int bCreateDb);
Input	bCreateDb　　1：create　　0：delete
Output	None
Return value	0：success　　-1：failure
NOTE	

3.18.27 Create 16-bit UUID service

Interface	int bt_gatt_create_service(int uuid16);
Input	uuid16　　16bit UUID
Output	None
Return value	0：success　　-1：failure
NOTE	This interface is a heavy load interface.

3.18.28 Create 128-bit UUID service

Interface	int bt_gatt_create_service(unsigned char *uuid128);
Input	uuid128　　128bit UUID
Output	None
Return value	0：success　　-1：failure
NOTE	This interface is a heavy load interface.

3.18.29 Create 16-bit characteristics

Interface	int bt_gatt_create_characteristic(int uuid16, int pro, int permission, uint32_t *attrHandle);
Input	uuid16 16-bit UUID pro property permission authority
Output	None
Return value	0：success -1：failure
NOTE	This interface is a heavy load interface.

3.18.30 Create 128-bit characteristics

Interface	int bt_gatt_create_characteristic(unsigned char *uuid128, int pro, int permission, uint32_t *attrHandle);
Input	uuid128 128-bit UUID pro property permission authority
Output	None
Return value	0：success -1：failure
NOTE	This interface is a heavy load interface.

3.18.31 Create a descriptor

Interface	int bt_gatt_create_descriptor();
Input	None
Output	None
Return value	0：success -1：failure
NOTE	This interface is currently configured for perfection so use the default configuration temporarily.

3.18.32 Add the created service to the database

Interface	int bt_gatt_add_service_2_db();
Input	None
Output	None
Return value	0：success -1：failure

NOTE	

3.18.33 Send notification

Interface	int bt_gatt_notification(uint16_t attrhandle,char *data, uint32_t data_len);
Input	attrhandle attribute handle data data data_len data length
Output	None
Return value	0：success -1：failure
NOTE	

3.18.34 Send indication

Interface	int bt_gatt_indication (uint16_t attrhandle,char *data, uint32_t data_len);
Input	attrhandleattribute handle data data data_len data length
Output	None
Return value	0：success -1：failure
NOTE	

3.18.35 Return host reads data from local request

Interface	int bt_gatt_read_cfm(uint16_t attrHandle, int rspCode, uint8_t *data, uint32_t data_len);
Input	attrhandleattribute handle rspCodeerror code data data data_len data length
Output	None
Return value	0：success -1：failure
NOTE	

3.18.36 Return the host to write data requests from the local

Interface	int bt_gatt_write_cfm(uint16_t attrHandle, int rspCode);
Input	attrhandleattribute handle rspCodeerror code
Output	None
Return value	0：success -1：failure

3.18.37 Callback

```
static void simcom_process_response(bt_msg *recv_msg)
{
switch(recv_msg->command)
{
case BT_SEARCH_COMMAND：
Scan result output
param1: search status: search end
param2: search result index
param3: rssi;
address: device mac address
data_len:   data name length
data:     device name
break;
case BT_BOND_COMMAND:
Pairing returns results
param1: bond result 0: FAILED   1:SUCCESS
address: device mac address
  data_len   device name length
data:      if data_len > 0   deviceName
break;
case BT_ACCEPT_COMMAND:
Prompt the user to confirm the pairing
param1:   accept mode
param2:   if ((param1 == ACCEPT_MODE_NOTIFICATION) ||
                              (param1 == ACCEPT_MODE_COMPARE) ||
                              (param1 == ACCEPT_MODE_PASSKEY))
param2 is passkey value
else
ignore
address: device address
  data_len:     data length
data:       device name    (maybe null)
break;
case BT_GET_BONDED_COMMAND:
Get a list of paired devices
param1 = total bonded number
param2 = current index;
param3 :    if total bonded unmber more than 0,    1: last one
address = devices;                     break;

case BT_CONNECT_COMMAND:
                  Report the message that other Bluetooth devices via SPP connection module.
break;

case BT_DISCONNECT_COMMAND:
                  Report the message that non-module actively disconnect SPP connection.
break;

case BT_SPP_RECV_COMMAND:
SPP      Receive data
data_len:     receive data len
data:            receive data
break;

case BT_GATT_CONNECT_COMMAND:
                  Report the message thatGATT has connected.
break;
case BT_GATT_READ_IND_COMMAND:
                  Report the message that other devices try to read module data via Bluetooth.
break;
case BT_GATT_WRITE_IND_COMMAND:
                  Report the message that other devices try to write module data via Bluetooth.
break;
            }
}
```

3.19 ETH

3.19.1 Network card mode settings

Ethernet is divided into three modes:

0: ETH_LAN Peripherals access the module through the Internet and the peripheral IP is allocated via the HDCP module.
1: ETH_WAN The module accesses the Internet via Ethernet.
2：ETH_LAN_STATIC Peripherals access the Internet through the module. Module ETH and peripheral IP set to static IP

In general, the functionality of Ethernet is pre-defined at design time. So you can set the mode before compiling the SDK.Modify the preset mode by modifying the value of the SDK source mdm-init/wlan/lan_mode. The corresponding value of the pattern is the above list value.

If the client is using windows development, downloading the default file to the system using the tool requires downloading two files: Copy 'lan_mode' and change the name to 'default_lan_mode'.

Download 'default_lan_mode' to 'etc' directory and download 'lan_mode' to 'data' directory.
'Default_lan_mode' is used to restore when the file in the data directory is destroyed.

The dynamic switching mode can set the network mode with AT+CLANMODE=0/1/2 command. However, this setting only modifies the lan_mode value in the data directory. If the lan_mode in the data directory is destroyed, it will revert to the default value (default_lan_mode value). Therefore, if the ETH mode designed by the customer is fixed, it is recommended to directly preset the initial mode.

3.19.2 Network card mode selection

Interface	Init(ethernet_type_info type)
Input	type 1 BCM898XX 2 AT803X
Output	None
Return value	0：success -1：failure

3.19.3 Driver install

Interface	int Install_driver()
Input	None
Output	None
Return value	0：success -1：failure

3.19.4 Driver uninstall

Interface	int Uninstall_driver()
Input	None
Output	None
Return value	0：success -1：failure
NOTE	If sleep needs to uninstall Ethernet driver.

3.19.5 Read preset MAC address from NV

Interface	int get_mac_address_from_nv(dms_device_mac_enum_v01 device_type, uint8_t *mac_addr);

Input	DMS_DEVICE_MAC_WLAN_V01 = 0 ---- WIFI MAC address
	DMS_DEVICE_MAC_BT_V01 = 1 ---- Bluetooth MAC address
	DMS_DEVICE_MAC_LAN_V01 = 2 ---- ETH MAC address

Output	None
Return value	0：success -1：failure
NOTE	

3.19.6 Set MAC address

Interface	int set_eth_mac_address(char *mac_address)
Input	mac_address: MAC address stringFor example：64:00:6A:04:65:A1
Output	None
Return value	0：success -1：failure
NOTE	If the MAC address is not read in NV, a random value can be set.

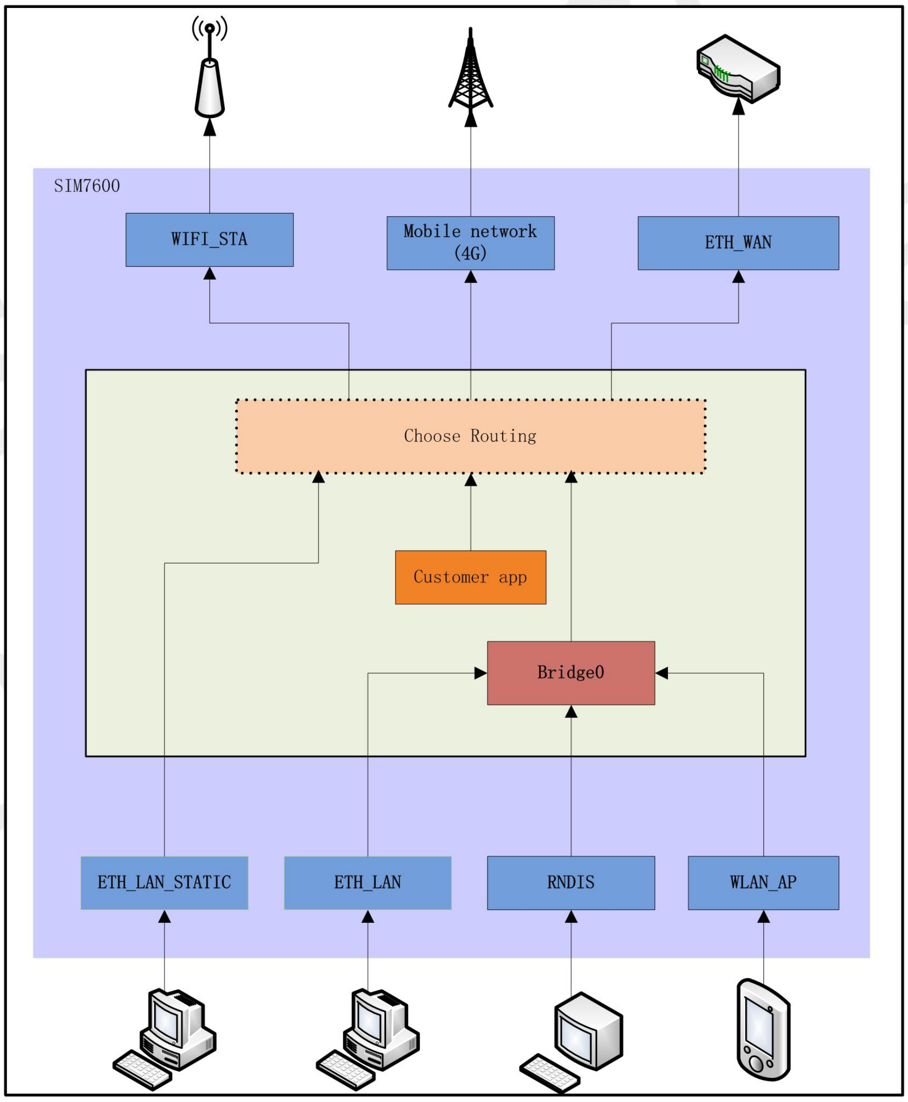
3.19.7 Set IP

Interface	int set_eth_ip_address(char *ip_address)
Input	ip_address: IP address： 192.168.*.*
Output	None
Return value	None
NOTE	Module sleep needs to uninstall ETH driver.

3.20 Network settings

This section mainly introduces the network access modes and network settings supported by the module.

3.21 Network access methods



There are three main ways to access the external network module：

1. Mobile network
2. WIFI_STAWIFI opens STA
3. ETH_WANEthernet as WAN port

There are four ways to connect external devices:
1. WIFI_AP
2. RNDIS
3. ETH_LANEthernet is connected to the internal bridge0 of the module. The module automatically assigns IP to the device.
4. ETH_LAN_STATIC Both the Ethernet module and the Ethernet port on the master device are set static IP.

The first three modes：The connection between the master device and the module is connected in the form of a local area network. A 192.168.225.* IP will be obtained from module HDCP. After the IP is successfully obtained the master can connect to the external network through the module, and can also communicate with the internal process of the module (APP) through the LAN IP.

The last mode：It needs to be set by the customer APP and the master to set the IP, gateway, NDS.

IP:	192.168.*.* (Same network segment as IP set by APP to ETH_LAN_STATIC)
Gateway：	192.168.*.* (Same as the IP set by the APP for ETH_LAN_STATIC)
DNSserver：	192.168.225.1 （regular）

3.21.1 Default route priority preset

Modifythe nodes in sim_open_sdk/sim_usrfs/mobileap_cfg.xml and sim_open_sdk/sim_rootfs/etc/default_mobileap_cfg.xml：

```
<FirstPreferredBackhaul>usb_cradle</FirstPreferredBackhaul>
<SecondPreferredBackhaul>wlan</SecondPreferredBackhaul>
    <ThirdPreferredBackhaul>wwan</ThirdPreferredBackhaul>
    usb_cradle:     Ethernet (ETH_WAN)
    wlan：WLAN_STA
    wwan：Mobile network
```

The module selects a default route based on the current module status and profile settings.
For example：
In the current setting, when the module has both WLAN_STA and ETH_WAN connected, the default route will be set to ETH_WAN. This means that accessing the external network will be accessed via ETH.
The setting does not change. If the Ethernet cable is unplugged, the default route will become WLAN. Insert the Ethernet cable. After the Ethernet connection is successful, the default route will be switched to ETH_WAN.
If the customer wants to use ETH_WAN preferentially when both WLAN_STA and ETH_WAN are connected, their order can be adjusted. Similarly, if the customer wants to pass 4G network when the 4G network dials successfully, then jump the 4G priority to the highest.

3.22 ALSA

3.22.1 Set the volume of inner speaker

Interface	int set_clvl_value(int clvl_value)
Input	clvl_value：volume value (0-5)
Output	None
Return value	0：success -1：failure
NOTE	

3.22.2 Get the volume fo inner speaker

Interface	int get_clvl_value(void)
Input	None
Output	None
Return value	-1：failure volume value：success
NOTE	

3.22.3 Set the mic gain

Interface	int set_micgain_value(int micgain_value)
Input	micgain_value：the gain value to be set(0-8)
Output	None
Return value	0：success -1：failure
NOTE	

3.22.4 Get the mic gain

Interface	int get_micgain_value(void)
Input	None
Output	None

Return value	The mic gain value：success　　-1：failure
Interface	int get_micgain_value(void)

3.22.5 Switch voice channel

Interface	int set_csdvc_value(int csdvc_value)
Input	csdvc_value 1: handset 3: speaker
Output	None
Return value	0：success　　-1：failure
NOTE	

3.22.6 Query the current voice channel

Interface	int get_csdvc_value(void)
Input	None
Output	None
Return value	The integer fo current voice channel：success　　-1：failure
NOTE	

3.23 Device Control

3.23.1 Enter the recovery mode

Interface	int exec_cdelta_cmd(const char *path)
Input	Path specifies the path of the upgrade package. It will use the default path(/cache/update_ota.zip) when path is NULL.
output	None
Return value	0：success　　-1：failure
NOTE	

3.23.2 adb setting

Interface	int exec_cusbadb_cmd(bool value)
Input	value 1：open adb port 0：close adb port
Output	None
Return value	0：success　　-1：failure
NOTE	

3.24 DMS

3.24.1 Initialization

Interface	int dms_init()
Input	None
Output	None
Return value	0：success　　-1：failure
NOTE	

3.24.2 Get imei

Interface	int get_imei(char *pImei)
Input	pImei ：the buffer poiter to cache the imei
Output	None
Return value	0：success　　-1：failure
NOTE	

3.24.3 Get meid

Interface	int get_meid(char *pMeid);
Input	pMeid：the buffer pointer to cache the meid
Output	None
Return value	0：success -1：failure
NOTE	

3.24.4 Get the firmware version identification code

Interface	int get_rev_id(char *pRev_id)
Input	pRev_id：the buffer pointer to cache the Rev_id
Output	None
Return value	0：success
Interface	int get_rev_id(char *pRev_id)

3.24.5 Set the UE work mode

Interface	int dms_set_operating_mode(unsigned char mode)
Input	mode: 0 Online 1 Low power 2 Factory Test mode 3 Offline 4 Resetting 5 Shutting down 6 Persistent low power 7 Mode-only low power
Output	None
Return value	0：success -1：failure
NOTE	

3.24.6 Release

Interface	void dms_deinit()
Input	None
output	None
Return value	none
NOTE	

4. Customer version mataince

The customer version is saved in the file /etc/simcom_ap_ver.ini; the max length of the version string is 35 bytes. It can querry the version information throuth the following methods:

1) Cat the file content.

2) Call AT+CSUB=1.