Build a Smarter World

EC2X&AG35 QuecOpen Quick Start

www.quectel.com

Our aim is to provide customers with timely and comprehensive service. For any assistance, please contact our company headquarters:

Quectel Wireless Solutions Co., Ltd.
7th Floor, Hongye Building, No.1801 Hongmei Road, Xuhui District, Shanghai 200233, China
Tel: +86 21 5108 6236
Email: info@quectel.com

Or our local office. For more information, please visit:
http://quectel.com/support/sales.htm

For technical support, or to report documentation errors, please visit:
http://quectel.com/support/technical.htm
Or email to: support@quectel.com

About the Document

This document applies to the EC2X and AG35 platforms.

History

Revision	Date	Author	Description
1.0	2017-11-25	Running Gale	Initial
1.1	2018-02-01	Running	Add fax chapter
1.2	2018-02-10	Running	Add auto start app
1.3	2018-02-23	Gale	Add the selection and modification of kernel configuration file Add the production of usrdata.ubi
1.4	2018-03-01	Running Jackson	Change this docment's name Add document reading instruction Add the production of debug version

Contents

1. Document reading instruction

1.1 QuecOpen introduction

Understand the technical architecture and hardware and software resources of the product.

(1) EC2X product, non-automobile, please read:
《KBA_QuecOpen_EC2X_ technology and resources overview》

(2) AG35 products, automobile, please read:
《KBA_QuecOpen_ vehicle AG35_ technology and resources overview》

1.2 hardware development

1. Functional PIN selection.

According to the functional requirements and the pin resource definition document: 《Quectel_EC20 R2.1_QuecOpen_GPIO_Assignment_Speadsheet》

2. SCH and PCB design.

Refer to 《Quectel_EC20_R2.0-QuecOpen_ hardware design manual》

And EVB schematic diagram 《EC20&EC21&EC25-TE-A_SCH. PDF》

3. Please contact the relevant personnel of the company in the process of design.

1.3 Software development

After reading this document directly, read the corresponding functional documents under the software development folder.

2. Introduction of development process

2.1 Requirements for developers

(1) Familiar with standard GNU/Linux application development, and common Linux system commands;
(2) Grasp the basic knowledge of some driving and network protocols.
(3) Understand some AT command knowledge and refer to the Quectel at command manual.

2.2 QuecOpen development process

(1) Ubuntu1404 or 1604 system, 4GB memory or above, 4 core CPU or more. If using virtual machine, the memory is allocated to the virtual machine not less than 4GB
(2) Install development tools, drivers, and SDK as the second chapter listed.
(3) Get familiar with the development process of QuecOpen SDK by writing a simple APP according chapter 3.1;
(4) Reimport the customer APP into the root file system according to the chapter 3.4 and regenerate the file system image;
(5) Advance development can refer to other relevant documents.

3. Development Environment Preparation

3.1 Install required tools

Ubuntu USB driver installation and burning tool installation please refer to 《Quectel_WCDMA&LTE_Linux_USB_Driver_User_Guide》

3.2 ADB installation

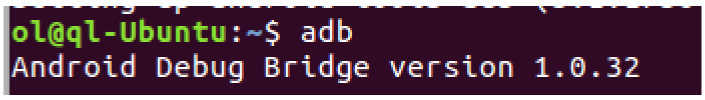
（1）Install ADB driver

```
sudo apt-get update
sudo apt-get install android-tools-adb
```

If above command fails, please try following ones:

```
sudo add-apt-repository ppa:nilarimogard/webupd8
sudo apt-get update
sudo apt-get install android-tools-adb
```

You can check it via “adb” command after it installed successfully:



```
ol@ql-Ubuntu:~$ adb
Android Debug Bridge version 1.0.32
```

（2）Add module USB VID

Query device VID:

```
lsusb
```

Modify configure file:

```
sudo vi .android/adb_usb.ini
```

（3）List ADB devices:

```
sudo adb kill-server
sudo adb devices
```

3.3 Firmware Update

When we got the latest firmware and SDK, we need to update the device first, by this way, we can verify the installation of the 2.1 tools, as well as the upgrade of the device.

About how to update firmware please refer to introduction of KBA_QuecOpen_ software upgrade downloading mode"

3.4 Download one file to the module

This chapter will show you how to download one common file or one App to the Linux file system.

3.4.1 Using ADB

Command basic format：

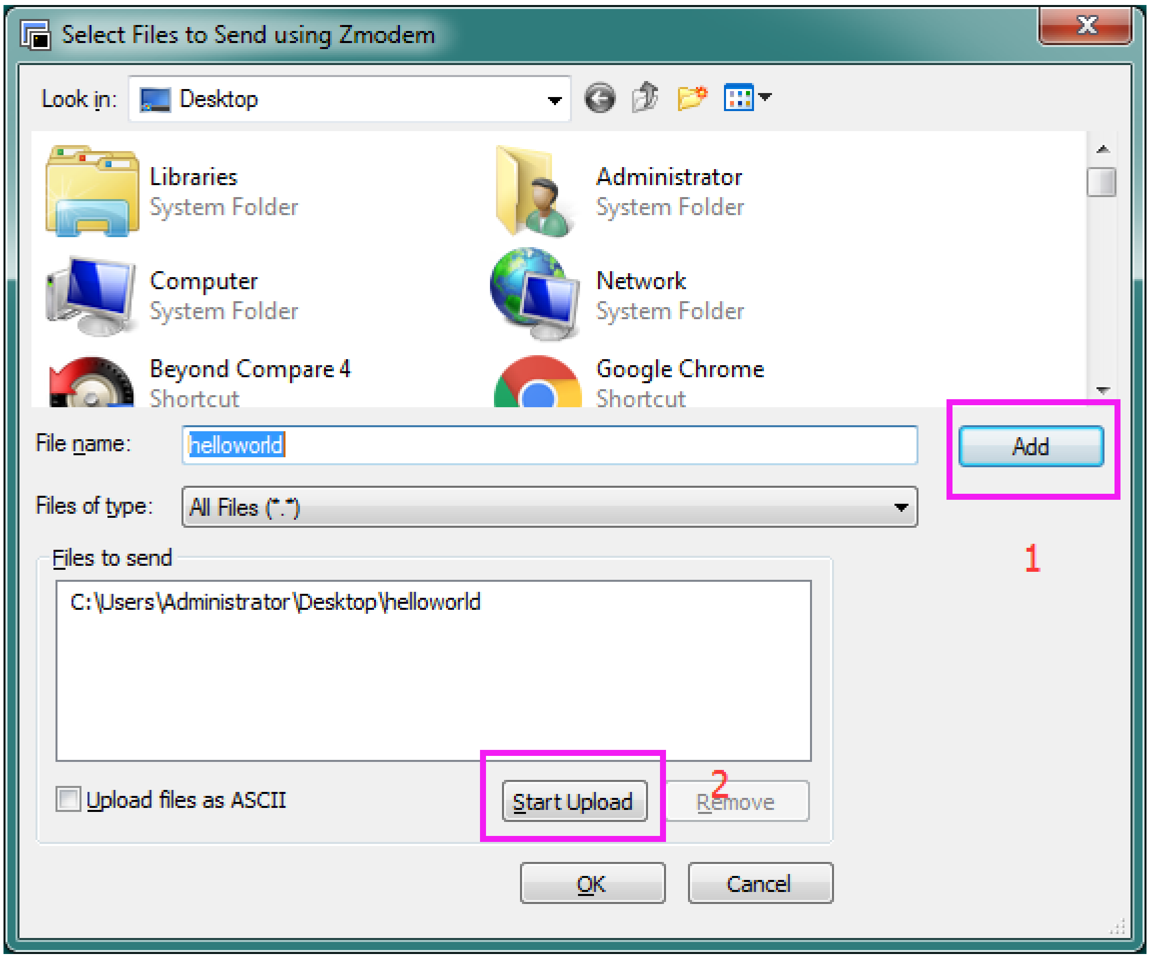
```
sudo adb push <local path> <module path>
```

for example:

```
adb push   ~/ql-ol-sdk/ql-ol-extsdk/example/helloWorld/hellolworld   /usrdata
```

3.4.2 Using Serial Port

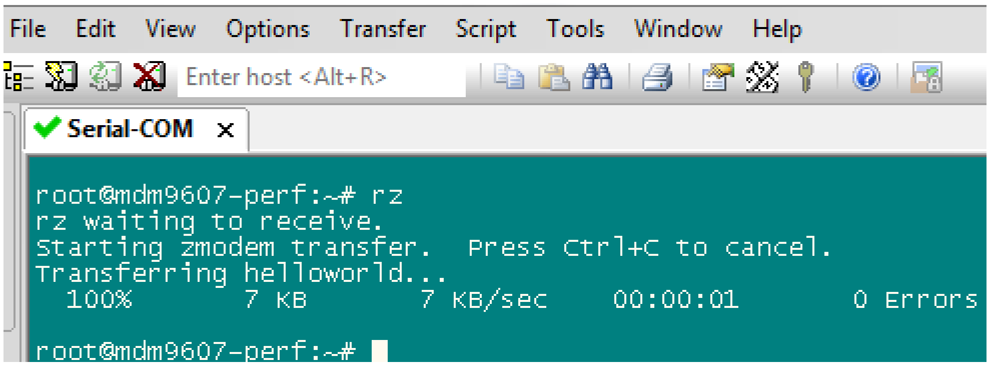
The following picture is generated by secureCRT in Windows when choose Zmodem to send file.



File name: helloworld

Files of type: All Files (*.*)

Files to send: C:\Users\Administrator\Desktop\helloworld



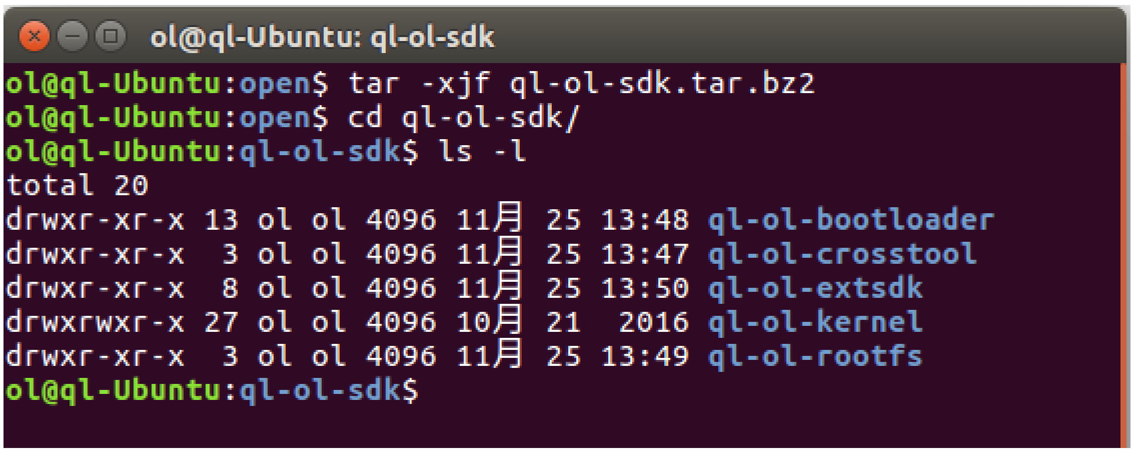
```
root@mdm9607-perf:~# rz
rz waiting to receive.
Starting zmodem transfer.  Press Ctrl+C to cancel.
Transferring helloworld...
  100%       7 KB       7 KB/sec    00:00:01       0 Errors

root@mdm9607-perf:~#
```

3.6 SDK Installation

（1） **Unzip SDK file**

SDK tar ball unzip procedure must be done under non-root Ubuntu environment.

```
tar -jxvf ql-ol-sdk.tar.bz2
```



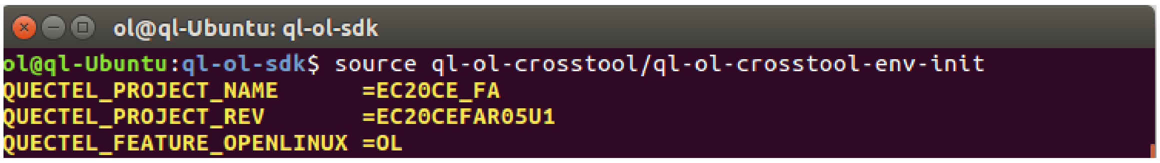
```
ol@ql-Ubuntu:open$ tar -xjf ql-ol-sdk.tar.bz2
ol@ql-Ubuntu:open$ cd ql-ol-sdk/
ol@ql-Ubuntu:ql-ol-sdk$ ls -l
total 20
drwxr-xr-x 13 ol ol 4096 11月 25 13:48 ql-ol-bootloader
drwxr-xr-x  3 ol ol 4096 11月 25 13:47 ql-ol-crosstool
drwxr-xr-x  8 ol ol 4096 11月 25 13:50 ql-ol-extsdk
drwxrwxr-x 27 ol ol 4096 10月 21  2016 ql-ol-kernel
drwxr-xr-x  3 ol ol 4096 11月 25 13:49 ql-ol-rootfs
ol@ql-Ubuntu:ql-ol-sdk$
```

（2） **List of document content**

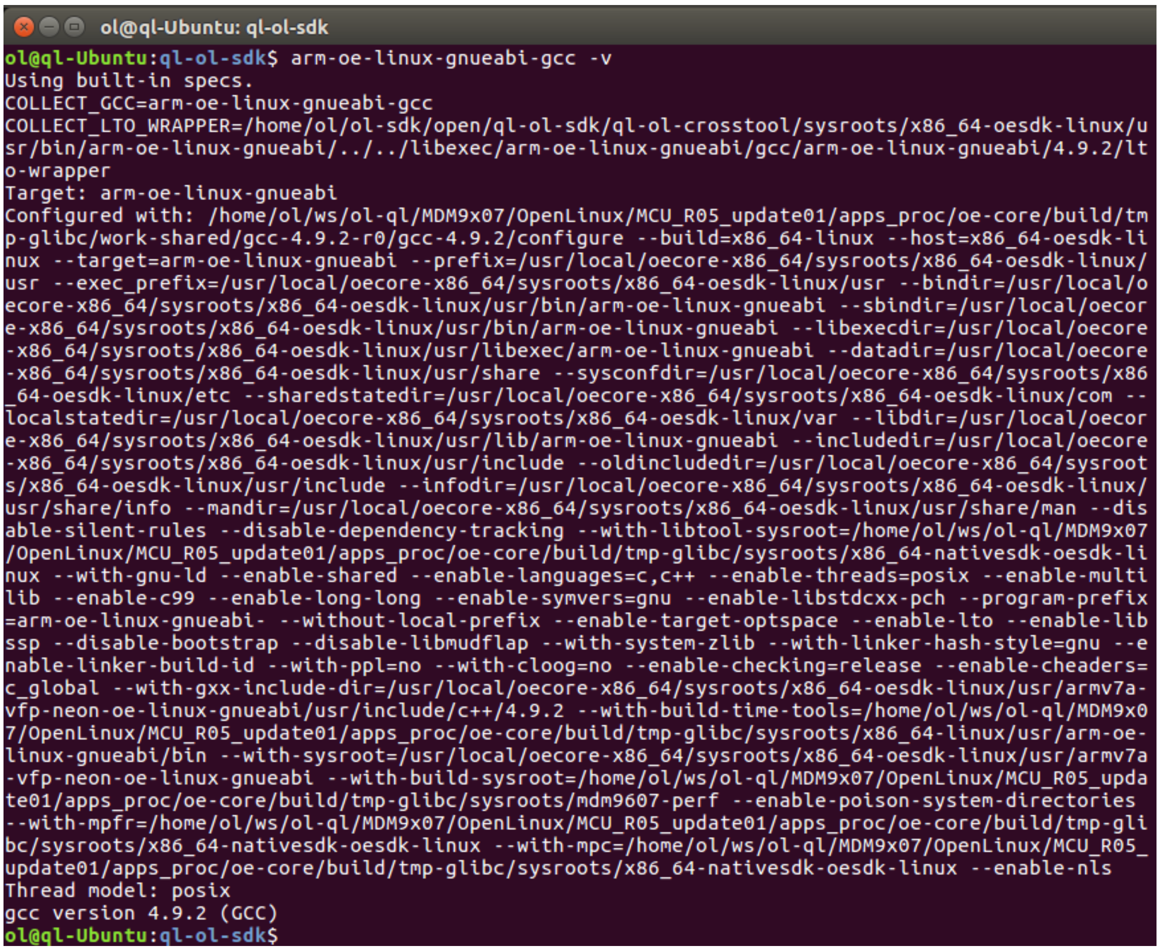
Directory	content
ql-ol-crosstool	Cross tool chain. Include QTI lib and header files.
ql-ol-bootloader	QTI bootloader source code（Only available as customer specified）
ql-ol-kernel	Linux kernel source code（Only available as customer specified）
ql-ol-rootfs	Root file system for platform runtime
ql-ol-extsdk	Include API， example and tools

（3） **Setup build environment**

```
cd ql-ol-sdk
source ql-ol-crosstool/ql-ol-crosstool-env-init
```



```
ol@ql-Ubuntu:ql-ol-sdk$ source ql-ol-crosstool/ql-ol-crosstool-env-init
QUECTEL_PROJECT_NAME      =EC20CE_FA
QUECTEL_PROJECT_REV       =EC20CEFAR05U1
QUECTEL_FEATURE_OPENLINUX =OL
```



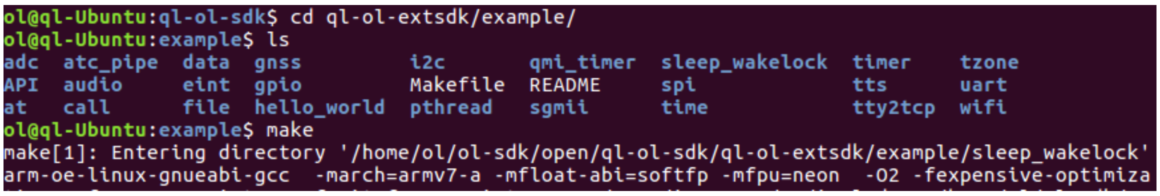
```
ol@ql-Ubuntu:ql-ol-sdk$ arm-oe-linux-gnueabi-gcc -v
Using built-in specs.
COLLECT_GCC=arm-oe-linux-gnueabi-gcc
COLLECT_LTO_WRAPPER=/home/ol/ol-sdk/open/ql-ol-sdk/ql-ol-crosstool/sysroots/x86_64-oesdk-linux/u
sr/bin/arm-oe-linux-gnueabi/../../libexec/arm-oe-linux-gnueabi/gcc/arm-oe-linux-gnueabi/4.9.2/lt
o-wrapper
Target: arm-oe-linux-gnueabi
Configured with: /home/ol/ws/ol-ql/MDM9x07/OpenLinux/MCU_R05_update01/apps_proc/oe-core/build/tm
p-glibc/work-shared/gcc-4.9.2-r0/gcc-4.9.2/configure --build=x86_64-linux --host=x86_64-oesdk-li
nux --target=arm-oe-linux-gnueabi --prefix=/usr/local/oecore-x86_64/sysroots/x86_64-oesdk-linux/
usr --exec_prefix=/usr/local/oecore-x86_64/sysroots/x86_64-oesdk-linux/usr --bindir=/usr/local/o
ecore-x86_64/sysroots/x86_64-oesdk-linux/usr/bin/arm-oe-linux-gnueabi --sbindir=/usr/local/oecor
e-x86_64/sysroots/x86_64-oesdk-linux/usr/bin/arm-oe-linux-gnueabi --libexecdir=/usr/local/oecore
-x86_64/sysroots/x86_64-oesdk-linux/usr/libexec/arm-oe-linux-gnueabi --datadir=/usr/local/oecore
-x86_64/sysroots/x86_64-oesdk-linux/usr/share --sysconfdir=/usr/local/oecore-x86_64/sysroots/x86
_64-oesdk-linux/etc --sharedstatedir=/usr/local/oecore-x86_64/sysroots/x86_64-oesdk-linux/com --
localstatedir=/usr/local/oecore-x86_64/sysroots/x86_64-oesdk-linux/var --libdir=/usr/local/oecor
e-x86_64/sysroots/x86_64-oesdk-linux/usr/lib/arm-oe-linux-gnueabi --includedir=/usr/local/oecore
-x86_64/sysroots/x86_64-oesdk-linux/usr/include --oldincludedir=/usr/local/oecore-x86_64/sysroot
s/x86_64-oesdk-linux/usr/include --infodir=/usr/local/oecore-x86_64/sysroots/x86_64-oesdk-linux/
usr/share/info --mandir=/usr/local/oecore-x86_64/sysroots/x86_64-oesdk-linux/usr/share/man --dis
able-silent-rules --disable-dependency-tracking --with-libtool-sysroot=/home/ol/ws/ol-ql/MDM9x07
/OpenLinux/MCU_R05_update01/apps_proc/oe-core/build/tmp-glibc/sysroots/x86_64-nativesdk-oesdk-li
nux --with-gnu-ld --enable-shared --enable-languages=c,c++ --enable-threads=posix --enable-multi
lib --enable-c99 --enable-long-long --enable-symvers=gnu --enable-libstdcxx-pch --program-prefix
=arm-oe-linux-gnueabi- --without-local-prefix --enable-target-optspace --enable-lto --enable-lib
ssp --disable-bootstrap --disable-libmudflap --with-system-zlib --with-linker-hash-style=gnu --e
nable-linker-build-id --with-ppl=no --with-cloog=no --enable-checking=release --enable-cheaders=
c_global --with-gxx-include-dir=/usr/local/oecore-x86_64/sysroots/x86_64-oesdk-linux/usr/armv7a-
vfp-neon-oe-linux-gnueabi/usr/include/c++/4.9.2 --with-build-time-tools=/home/ol/ws/ol-ql/MDM9x0
7/OpenLinux/MCU_R05_update01/apps_proc/oe-core/build/tmp-glibc/sysroots/x86_64-linux/usr/arm-oe-
linux-gnueabi/bin --with-sysroot=/usr/local/oecore-x86_64/sysroots/x86_64-oesdk-linux/usr/armv7a
-vfp-neon-oe-linux-gnueabi --with-build-sysroot=/home/ol/ws/ol-ql/MDM9x07/OpenLinux/MCU_R05_upda
te01/apps_proc/oe-core/build/tmp-glibc/sysroots/mdm9607-perf --enable-poison-system-directories
--with-mpfr=/home/ol/ws/ol-ql/MDM9x07/OpenLinux/MCU_R05_update01/apps_proc/oe-core/build/tmp-gli
bc/sysroots/x86_64-nativesdk-oesdk-linux --with-mpc=/home/ol/ws/ol-ql/MDM9x07/OpenLinux/MCU_R05_
update01/apps_proc/oe-core/build/tmp-glibc/sysroots/x86_64-nativesdk-oesdk-linux --enable-nls
Thread model: posix
gcc version 4.9.2 (GCC)
ol@ql-Ubuntu:ql-ol-sdk$
```

（4） **Verify**

Build all the examples

cd ql-ol-extsdk/example

make

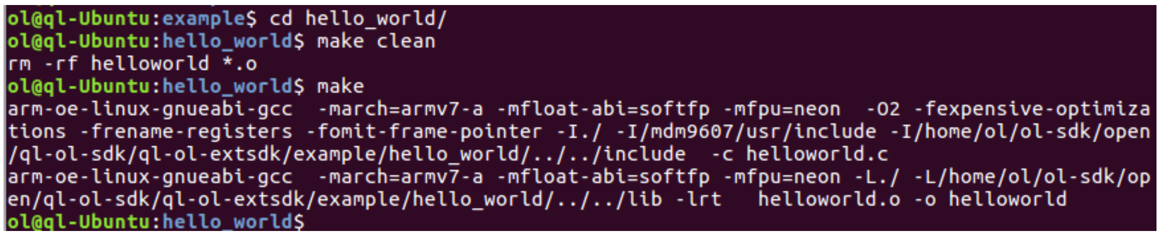


```
ol@ql-Ubuntu:ql-ol-sdk$ cd ql-ol-extsdk/example/
ol@ql-Ubuntu:example$ ls
adc  atc_pipe  data  gnss         i2c       qmi_timer  sleep_wakelock  timer    tzone
API  audio     eint  gpio         Makefile  README     spi             tts      uart
at   call      file  hello_world  pthread   sgmii      time            tty2tcp  wifi
ol@ql-Ubuntu:example$ make
make[1]: Entering directory '/home/ol/ol-sdk/open/ql-ol-sdk/ql-ol-extsdk/example/sleep_wakelock'
arm-oe-linux-gnueabi-gcc  -march=armv7-a -mfloat-abi=softfp -mfpu=neon  -O2 -fexpensive-optimiza
```

Build single example

cd hello_world

make



```
ol@ql-Ubuntu:example$ cd hello_world/
ol@ql-Ubuntu:hello_world$ make clean
rm -rf helloworld *.o
ol@ql-Ubuntu:hello_world$ make
arm-oe-linux-gnueabi-gcc  -march=armv7-a -mfloat-abi=softfp -mfpu=neon  -O2 -fexpensive-optimiza
tions -frename-registers -fomit-frame-pointer -I./ -I/mdm9607/usr/include -I/home/ol/ol-sdk/open
/ql-ol-sdk/ql-ol-extsdk/example/hello_world/../../include  -c helloworld.c
arm-oe-linux-gnueabi-gcc  -march=armv7-a -mfloat-abi=softfp -mfpu=neon -L./ -L/home/ol/ol-sdk/op
en/ql-ol-sdk/ql-ol-extsdk/example/hello_world/../../lib -lrt   helloworld.o -o helloworld
ol@ql-Ubuntu:hello_world$
```

4. Linux Develop and Debug

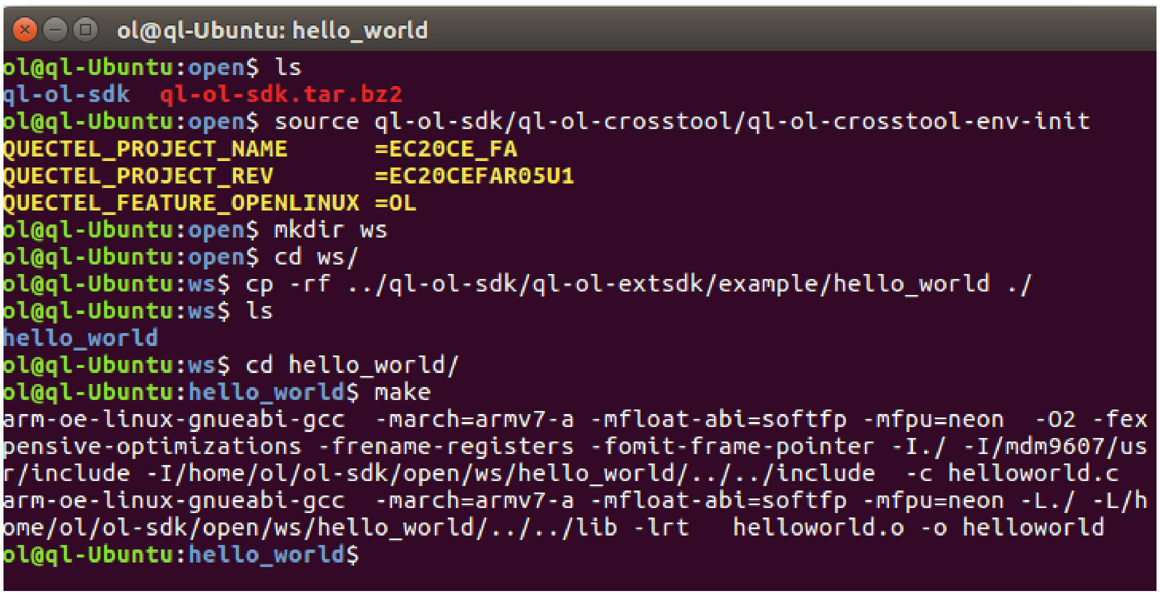
You must execute *“source ql-ol-crosstool/ql-ol-crosstool-env-init”* first of all !
This is just the beginning…

4.1 Linux APP development

QuecOpen Linux standard APP development is same as traditional embedded ARM-Linux development process, the requirements for customers, so long as there are basic Linux application development experience. The following document, from HelloWorld to single dial dial-up, guides customers to experience the QuecOpen Linux development process.

4.1.1 Helloworld

（1）**Create workspace**
Here we create ws folder for example.
（2）**Copy demo**
Copy ql-ol-sdk/ql-ol-extsdk/example/hello_world to the ws folder.
（3）**Build**



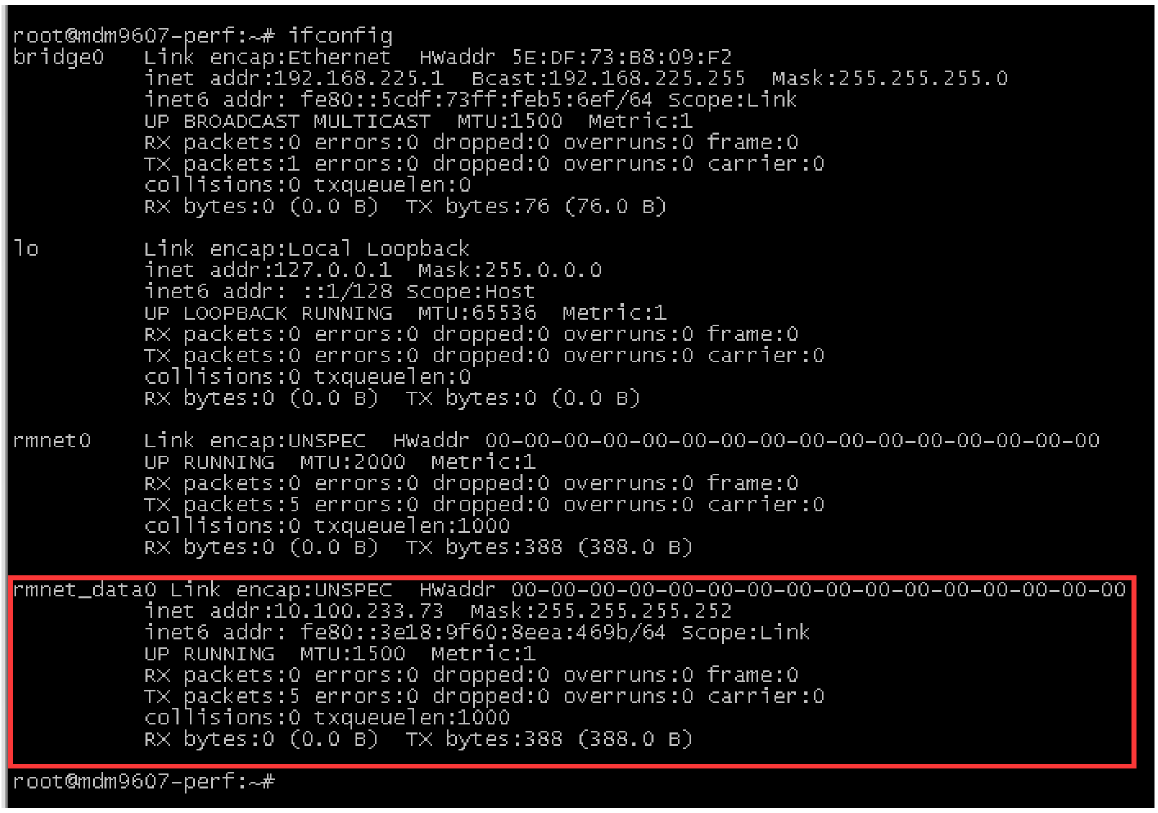
ol@ql-Ubuntu: hello_world

```
ol@ql-Ubuntu:open$ ls
ql-ol-sdk  ql-ol-sdk.tar.bz2
ol@ql-Ubuntu:open$ source ql-ol-sdk/ql-ol-crosstool/ql-ol-crosstool-env-init
QUECTEL_PROJECT_NAME      =EC20CE_FA
QUECTEL_PROJECT_REV       =EC20CEFAR05U1
QUECTEL_FEATURE_OPENLINUX =OL
ol@ql-Ubuntu:open$ mkdir ws
ol@ql-Ubuntu:open$ cd ws/
ol@ql-Ubuntu:ws$ cp -rf ../ql-ol-sdk/ql-ol-extsdk/example/hello_world ./
ol@ql-Ubuntu:ws$ ls
hello_world
ol@ql-Ubuntu:ws$ cd hello_world/
ol@ql-Ubuntu:hello_world$ make
arm-oe-linux-gnueabi-gcc  -march=armv7-a -mfloat-abi=softfp -mfpu=neon  -O2 -fex
pensive-optimizations -frename-registers -fomit-frame-pointer -I./ -I/mdm9607/us
r/include -I/home/ol/ol-sdk/open/ws/hello_world/../../include  -c helloworld.c
arm-oe-linux-gnueabi-gcc  -march=armv7-a -mfloat-abi=softfp -mfpu=neon -L./ -L/h
ome/ol/ol-sdk/open/ws/hello_world/../../lib -lrt   helloworld.o -o helloworld
ol@ql-Ubuntu:hello_world$
```

（3）**Download and run APP**
1）Refer to charpter 2.5 to download file to the module;
2）Modify the file permissions to be executable
3）Run helloworld

4.1.2 Single APN Data Call

（1） Copy demo
Copy ql-ol-sdk/ql-ol-extsdk/example/data to your folder.
（2） Compile
（3） download and run
After dial successfully, you can check it via following command:



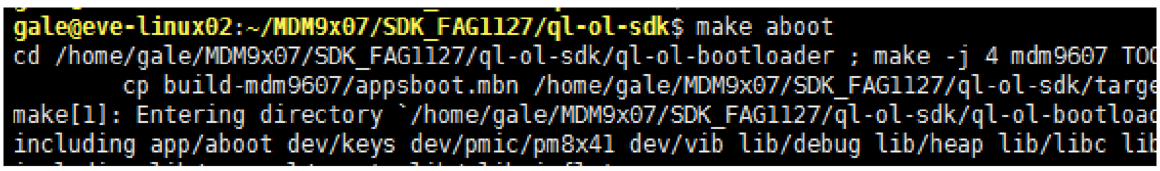
```
root@mdm9607-perf:~# ifconfig
bridge0   Link encap:Ethernet  HWaddr 5E:DF:73:B8:09:F2
          inet addr:192.168.225.1  Bcast:192.168.225.255  Mask:255.255.255.0
          inet6 addr: fe80::5cdf:73ff:feb5:6ef/64 Scope:Link
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:76 (76.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

rmnet0    Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          UP RUNNING  MTU:2000  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:388 (388.0 B)

rmnet_data0 Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet addr:10.100.233.73  Mask:255.255.255.252
          inet6 addr: fe80::3e18:9f60:8eea:469b/64 Scope:Link
          UP RUNNING  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:388 (388.0 B)

root@mdm9607-perf:~#
```

4.1.3 Advanced application development

Please refer to the guidance document in our application development file and the example in SDK.

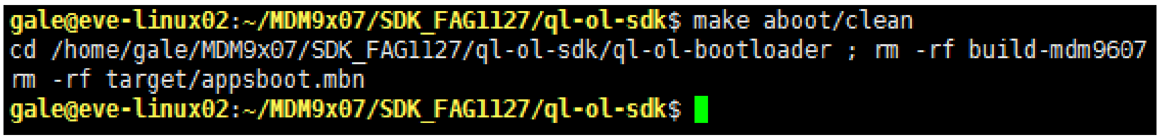
4.2 Bootloader Development

make aboot //Build bootloader, and generate appsboot.mbn in target/ folder of current path.



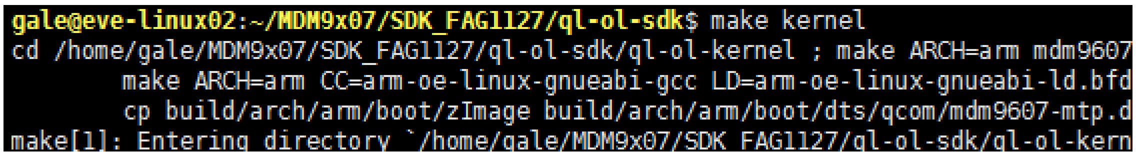
```
gale@eve-linux02:~/MDM9x07/SDK_FAG1127/ql-ol-sdk$ make aboot
cd /home/gale/MDM9x07/SDK_FAG1127/ql-ol-sdk/ql-ol-bootloader ; make -j 4 mdm9607 TOO
        cp build-mdm9607/appsboot.mbn /home/gale/MDM9x07/SDK_FAG1127/ql-ol-sdk/targe
make[1]: Entering directory `/home/gale/MDM9x07/SDK_FAG1127/ql-ol-sdk/ql-ol-bootload
including app/aboot dev/keys dev/pmic/pm8x41 dev/vib lib/debug lib/heap lib/libc lib
```

make aboot/clean **//clean up mid-file of last command**

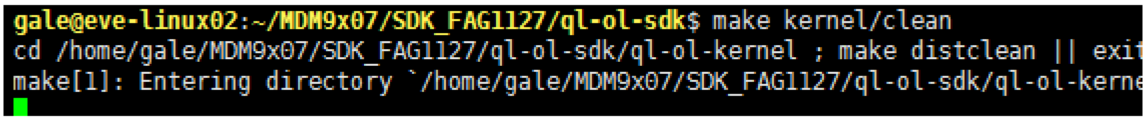

```
gale@eve-linux02:~/MDM9x07/SDK_FAG1127/ql-ol-sdk$ make aboot/clean
cd /home/gale/MDM9x07/SDK_FAG1127/ql-ol-sdk/ql-ol-bootloader ; rm -rf build-mdm9607
rm -rf target/appsboot.mbn
gale@eve-linux02:~/MDM9x07/SDK_FAG1127/ql-ol-sdk$
```

4.3 Kernel development

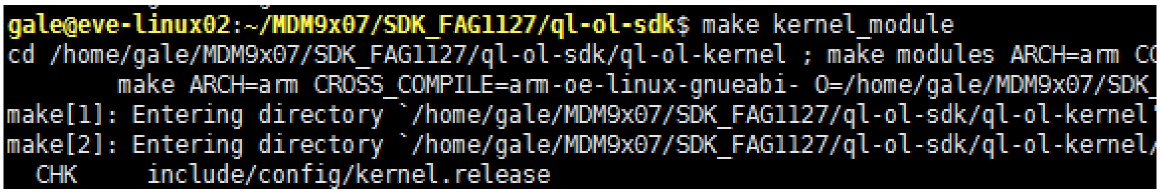
make kernel **//build will generate boot.img in the target/ folder.**


```
gale@eve-linux02:~/MDM9x07/SDK_FAG1127/ql-ol-sdk$ make kernel
cd /home/gale/MDM9x07/SDK_FAG1127/ql-ol-sdk/ql-ol-kernel ; make ARCH=arm mdm9607
        make ARCH=arm CC=arm-oe-linux-gnueabi-gcc LD=arm-oe-linux-gnueabi-ld.bfd
        cp build/arch/arm/boot/zImage build/arch/arm/boot/dts/qcom/mdm9607-mtp.d
make[1]: Entering directory `/home/gale/MDM9x07/SDK_FAG1127/ql-ol-sdk/ql-ol-kern
```

make kernel/clean // clean up mid-file of last command

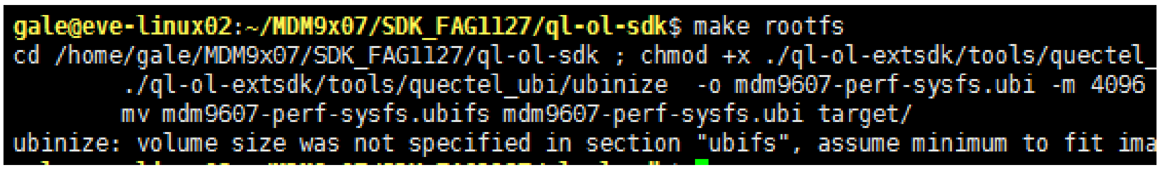

```
gale@eve-linux02:~/MDM9x07/SDK_FAG1127/ql-ol-sdk$ make kernel/clean
cd /home/gale/MDM9x07/SDK_FAG1127/ql-ol-sdk/ql-ol-kernel ; make distclean || exit
make[1]: Entering directory `/home/gale/MDM9x07/SDK_FAG1127/ql-ol-sdk/ql-ol-kerne
```

make kernel_module **// Build kernel module only if you have modified related kmod，and it will be installed to rootfs folder automatically, then you need rebuild sysfs.ubi.**

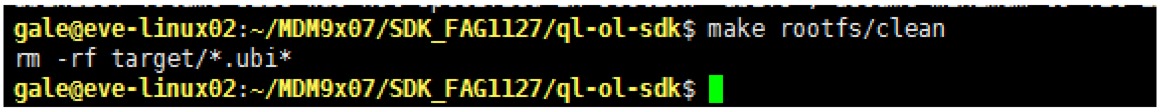

```
gale@eve-linux02:~/MDM9x07/SDK_FAG1127/ql-ol-sdk$ make kernel_module
cd /home/gale/MDM9x07/SDK_FAG1127/ql-ol-sdk/ql-ol-kernel ; make modules ARCH=arm CC
        make ARCH=arm CROSS_COMPILE=arm-oe-linux-gnueabi- O=/home/gale/MDM9x07/SDK_
make[1]: Entering directory `/home/gale/MDM9x07/SDK_FAG1127/ql-ol-sdk/ql-ol-kernel'
make[2]: Entering directory `/home/gale/MDM9x07/SDK_FAG1127/ql-ol-sdk/ql-ol-kernel/
  CHK     include/config/kernel.release
```

4.4 Make file system

make rootfs **//This will generate sysfs.ubi in the target/ folder.**


```
gale@eve-linux02:~/MDM9x07/SDK_FAG1127/ql-ol-sdk$ make rootfs
cd /home/gale/MDM9x07/SDK_FAG1127/ql-ol-sdk ; chmod +x ./ql-ol-extsdk/tools/quectel_
        ./ql-ol-extsdk/tools/quectel_ubi/ubinize  -o mdm9607-perf-sysfs.ubi -m 4096
        mv mdm9607-perf-sysfs.ubifs mdm9607-perf-sysfs.ubi target/
ubinize: volume size was not specified in section "ubifs", assume minimum to fit ima
```

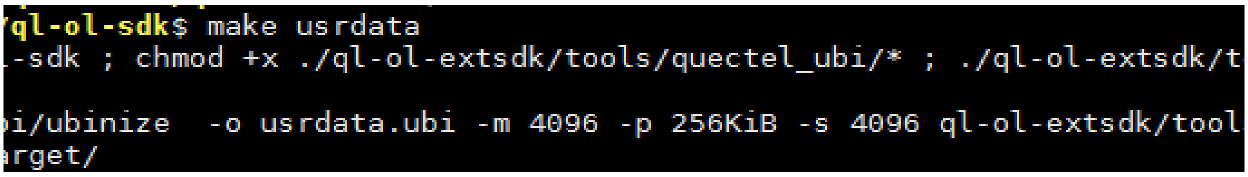
make rootfs/clean // clean up mid-file of last command；


```
gale@eve-linux02:~/MDM9x07/SDK_FAG1127/ql-ol-sdk$ make rootfs/clean
rm -rf target/*.ubi*
gale@eve-linux02:~/MDM9x07/SDK_FAG1127/ql-ol-sdk$
```

4.6 make usrdata.ubi

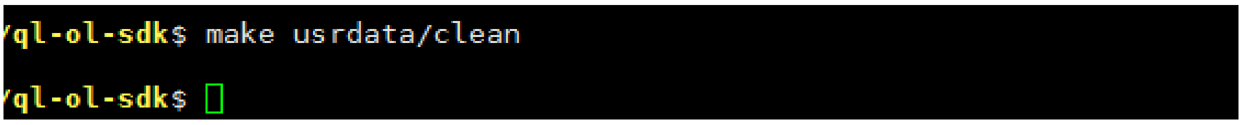
System has usr_data partition, which can be used to store users' files, and to upgrade and use DFOTA.

make usrdata **//gen usrdata.ubi**



```
/ql-ol-sdk$ make usrdata
-sdk ; chmod +x ./ql-ol-extsdk/tools/quectel_ubi/* ; ./ql-ol-extsdk/t

i/ubinize  -o usrdata.ubi -m 4096 -p 256KiB -s 4096 ql-ol-extsdk/tool
rget/
```

make usrdata/clean

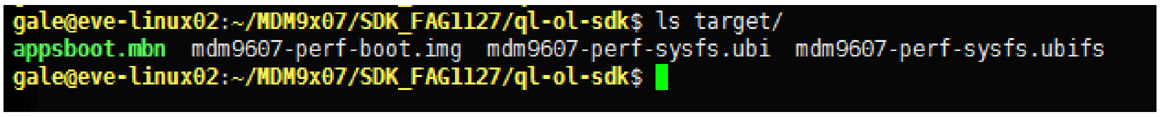


```
/ql-ol-sdk$ make usrdata/clean

/ql-ol-sdk$
```

4.6 One key compilation

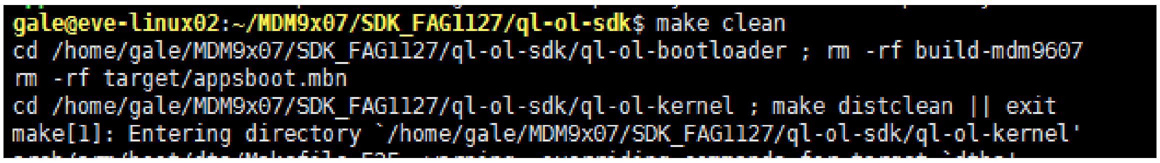
QuecOpen SDK provides one key to build all of aboot, kernel, kernel_module, rootfs。

make **//build all and put the output to target/ folder.**



```
gale@eve-linux02:~/MDM9x07/SDK_FAG1127/ql-ol-sdk$ ls target/
appsboot.mbn  mdm9607-perf-boot.img  mdm9607-perf-sysfs.ubi  mdm9607-perf-sysfs.ubifs
gale@eve-linux02:~/MDM9x07/SDK_FAG1127/ql-ol-sdk$
```

make clean **//clean up**



```
gale@eve-linux02:~/MDM9x07/SDK_FAG1127/ql-ol-sdk$ make clean
cd /home/gale/MDM9x07/SDK_FAG1127/ql-ol-sdk/ql-ol-bootloader ; rm -rf build-mdm9607
rm -rf target/appsboot.mbn
cd /home/gale/MDM9x07/SDK_FAG1127/ql-ol-sdk/ql-ol-kernel ; make distclean || exit
make[1]: Entering directory `/home/gale/MDM9x07/SDK_FAG1127/ql-ol-sdk/ql-ol-kernel'
```

Then you can copy all the files in the target folder to the upgrade package and download them to the module.

4.7 Generate debug firmware

When you need to open the kernel debugging log, you need to compile the debug version. The compilation method is as follows:

1. The configuration

Make debug_kernel_menuconfig

2. Compile file system and kernel.

Make debug_version

5. Module Startup Check

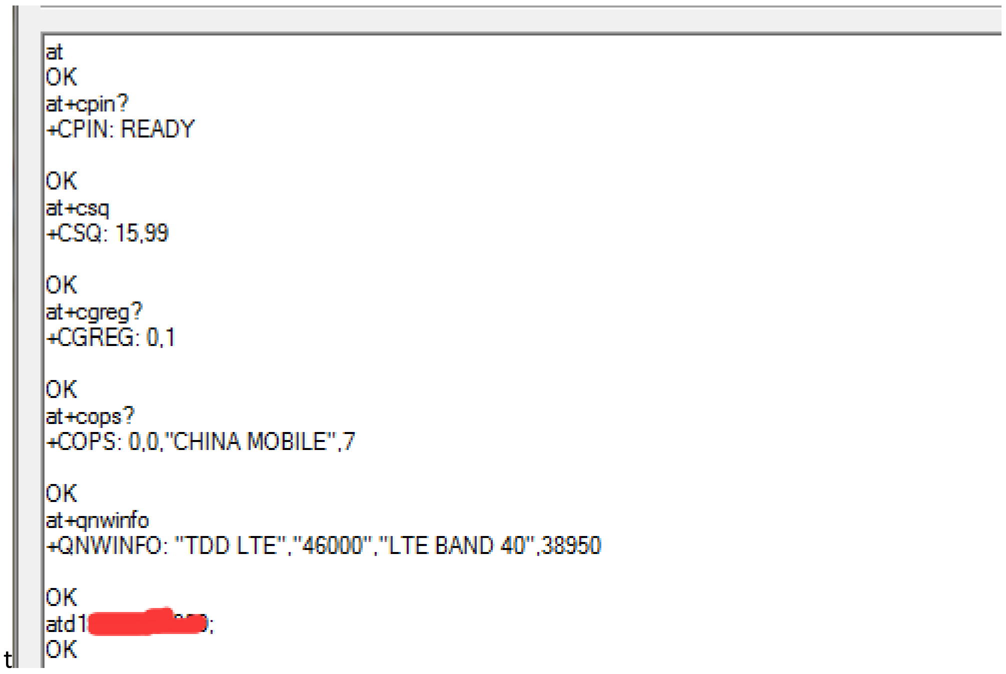
This chapter will show you how to check whether system run up successfully by using AT command. Details as following:

（1）Connect PC with the module's MAIN UART port or USB AT port through the USB turn serial port cable;
（2）Insert SIM card, connect antenna and turn on the power;
（3）Send following AT command via QCOM tool:

AT : check AT port connected or not.
AT+CPIN? : check sim card insert and valid.
AT+CSQ : check signal strength
AT+CGREG? : Check module register to local network successfully.
AT+COPS? : Check current service provider
AT+QNWINFO : Query current network mode

ATDxxx; :Dial and audio test, here xxx means phone number.

The following picture shows the startup check when insert a general China Mobile SIM card.

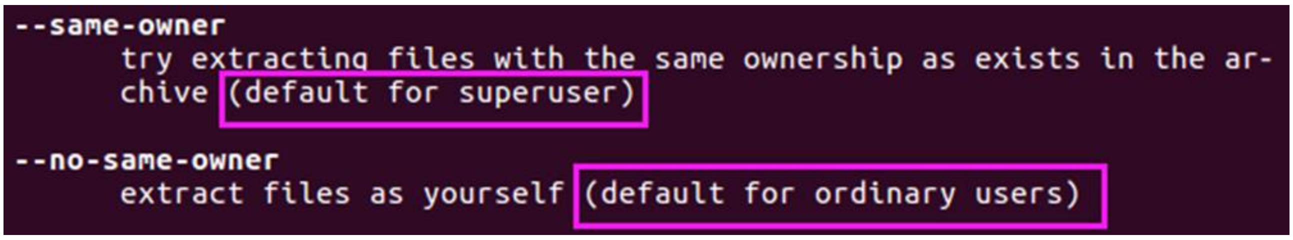


```
at
OK
at+cpin?
+CPIN: READY

OK
at+csq
+CSQ: 15,99

OK
at+cgreg?
+CGREG: 0,1

OK
at+cops?
+COPS: 0,0,"CHINA MOBILE",7

OK
at+qnwinfo
+QNWINFO: "TDD LTE","46000","LTE BAND 40",38950

OK
atd1[redacted];
OK
```

FAQ

1. Why does the compression package have to be decompressed in a normal user environment?

You can query the tar command manual to see:



```
--same-owner
       try extracting files with the same ownership as exists in the ar-
       chive (default for superuser)

--no-same-owner
       extract files as yourself (default for ordinary users)
```