MSM8626 LA Software User Manual

MSM8626 LA 软件用户手册

User Manual 用户手册

SP80-ND928-4 E

June 13, 2013

Submit technical questions at:
https://support.cdmatech.com/

Confidential and Proprietary – Qualcomm Technologies, Inc.
机密和专有信息—高通技术股份有限公司

NO PUBLIC DISCLOSURE PERMITTED: Please report postings of this document on public servers or web sites to: DocCtrlAgent@qualcomm.com. 禁止公开：如在公共服务器或网站上发现本文档，请报告至: DocCtrlAgent@qualcomm.com.

Restricted Distribution: Not to be distributed to anyone who is not an employee of either Qualcomm or its subsidiaries without the express approval of Qualcomm's Configuration Management. 限制分发：未经高通配置管理部门的明示批准，不得发布给任何非高通或高通子公司员工的人。

Not to be used, copied, reproduced, or modified in whole or in part, nor its contents revealed in any manner to others without the express written permission of Qualcomm Technologies, Inc. 未经高通科技股份有限公司明示的书面允许，不得使用、复印、 复制、或修改全部或部分文档，不得以任何形式向他人透露其内容。

The user of this documentation acknowledges and agrees that any Chinese text and/or translation herein shall be for reference purposes only and that in the event of any conflict between the English text and/or version and the Chinese text and/or version, the English text and/or version shall be controlling. 本文档的用户知悉并同意中文文本和/或翻译仅供参考之目的，如英文 文本和/或版本和中文文本和/或版本之间存在冲突，以英文文本和/或版本为准。 This document contains confidential and proprietary information and must be shredded when discarded. 未经高通明示的书面允 许，不得使用、复印、复制全部或部分文档，不得以任何形式向他人透露其内容。本文档含有高通机密和专有信息，丢 弃时必须粉碎销毁。

Qualcomm is a trademark of QUALCOMM Incorporated, registered in the United States and other countries. All QUALCOMM Incorporated trademarks are used with permission. Other product and brand names may be trademarks or registered trademarks of their respective owners. Qualcomm 是高通公司在美国及其它国家注册的商标。所有高通公司的商标皆获得使用许可。 其它产品和品牌名称可能为其各自所有者的商标或注册商标。

This technical data may be subject to U.S. and international export, re-export, or transfer (“export”) laws. Diversion contrary to U.S. and international law is strictly prohibited. 本文档及所含技术资料可能受美国和国际出口、再出口或转移出口法律的 限制。严禁违反或偏离美国和国际的相关法律。

Qualcomm Technologies, Inc. 5775 Morehouse Drive San Diego, CA 92121 U.S.A.
高通技术股份有限公司，美国加利福尼亚州圣地亚哥市莫豪斯路 5775 号，邮编 92121

©2013 Qualcomm Technologies, Inc. All rights reserved.
©2013 高通技术股份有限公司版权所有，并保留所有权利。

Contents 目录

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

QUALCOMM®
2013.07.09 at 23:19:43 PDT
liyh-gionee.com
108.171.248.77

Figures 图目录

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

QUALCOMM®
2013.07.09 at 23:19:43 PDT
liyh-gionee.com
108.171.248.77

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

Tables 表目录

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

Revision history

Revision	Date	Description
A	April 2013	Initial release
B	May 2013	Update build commands in 3.3.1 and update QPST version in 4.1.
C	May 2013	This is a bilingual version.
D	June 2013	The layout of this template is changed. And the content of 4.3.1 has been updated.
E	June 2013	5.2 DSDS/DSDA NV Settings is newly added.

Note: There is no Rev. I, O, Q, S, X, or Z per Mil. standards.

QUALCOMM
2013.07.09 at 23:19:43 PDT
liyh-gionee.com
108.171.248.77

1 Introduction

1.1 Purpose

This document introduces how to obtain, build and program software, and the general operations which are applicable to the MSM8626 LA reference platform.

1.2 Scope

This document introduces how to set up a development environment, obtain the software, and install it into the development environment.

It also covers software operation, such as how to rebuild the software "as-is" and program the resultant build products (firmware) into a reference platform. It provides the needed information to program and reprogram the firmware devices in the system.

It describes the firmware devices, the equipment, software, and procedures about how to erase firmware devices, load software into the firmware devices, and verify the load process. It also describes general operations, such as call, gps and multimedia configuration.

1.3 Conventions

Function declarations, function names, type declarations, and code samples appear in a different font, e.g., `#include`.

Code variables appear in angle brackets, e.g., `<number>`.

Commands to be entered appear in a different font, e.g., **`copy a:*.* b:`**.

Button and key names appear in bold font, e.g., click **Save** or press **Enter**.

If you are viewing this document using a color monitor, or if you print this document to a color printer, red typeface indicates data types, blue typeface indicates attributes, and green typeface indicates system attributes.

Parameter types are indicated by arrows:

- → Designates an input parameter
- ← Designates an output parameter
- ↔ Designates a parameter used for both input and output

Shading indicates content that has been added or changed in this revision of the document.

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

1.4 References

Reference documents are listed in Table 1-1. Reference documents that are no longer applicable are deleted from this table; therefore, reference numbers may not be sequential.

Table 1-1 Reference documents and standards

Ref.	Document	
Qualcomm Technologies		
Q1	*Application Note: Software Glossary for Customers*	CL93-V3077-1
Q2	*Hexagon™ Tools Installation Guide*	80-VB419-25
Q3	*Hexagon™ Development Tools Overview*	80-VB419-74
Q4	*Presentation: SCONS Build System Overview*	80-N3601-1
Q5	*SCons Build System Migration Guide*	80-N3605-1
Q6	*Application Note: SCons External Dependencies*	80-N2077-2
Q7	*Application Note: Make vs SCons Differences*	80-N2077-3
Q8	*Application Note: Modifying Default SCons Environment*	80-N2077-4
Q9	*Application Note: SCons Modifications*	80-N2077-5
Q10	*Application Note: SCons Warning Messages about pywin32 Extensions*	80-N2077-6
Q11	*USB Host Driver for Windows 2000/Windows XP User Guide*	80-V4609-1
Q12	*USB Host Driver Installation Instructions for Microsoft Windows*	80-VP092-1
Q13	*gpsOne® Gen 8 Engineer RF Development and Mobile Station Time Calibration Test Procedures*	80-VM522-2
Q14	*Introduction to Qualcomm ChipCenter and Qualcomm ChipCode*	80-NC193-1
Q15	*Qualcomm ChipCenter and Qualcomm ChipCode User Guide*	80-NC193-2
Resources		
R1	*Android™ Open Source Project Page*	http://source.android.com/
R2	*Android™ Developer Resources*	http://developer.android.com/index.html
R3	*Android™ Source Download and System Setup*	http://source.android.com/source/index.html
R4	*Code Aurora Forum*	https://www.codeaurora.org/
R5	*Installing Repo*	http://source.android.com/source/downloading.html
R6	*Qualcomm ChipCode Website*	https://chipcode.qti.qualcomm.com/
R7	*Qualcomm ChipCode Help Wiki*	https://chipcode.qti.qualcomm.com/projects/help/wiki

1.5 Technical assistance

For assistance or clarification on information in this document, submit a case to Qualcomm Technologies, Inc. (QTI) at https://support.cdmatech.com/.

If you do not have access to the CDMATech Support Service website, register for access or send email to support.cdmatech@qti.qualcomm.com.

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

1.6 Acronyms

For definitions of terms and abbreviations, see [Q1].

QUALCOMM®
2013.07.09 at 23:19:43 PDT
liyh-gionee.com
108.171.248.77

2 Software Environment

2.1 Equipment and software

Table 2-1 identifies the hardware, software and other equipment which is needed for a user to install and run the software.

Table 2-1 Required hardware, software, and other equipment

	Item description	Version	Source/vendor	Purpose
1	Linux development workstation exceeds minimum desktop system requirements for running Ubuntu 64-bit OS. A powerful PC to speed up compile times is recommended. Google recommends the following items in the android forum link. ▪ 16GB RAM minimum recommended ▪ Quad Core CPU (e.g., Intel i7-2600 @3.4 Ghz) equivalent or better ▪ 1 TB Hard Drive (SSD accelerated preferred)	—	—	Android build machine
2	Windows 7 or Windows XP workstation	Windows 7 or Windows XP	Microsoft	Alternate Non-HLOS build machine and Windows-based programming tools
3	Ubuntu 10.04 LTS Linux distribution for 64-bit architecture	10.04 LTS	Ubuntu Community/ Canonical, Ltd.	Android build host OS
4	Java SE JDK for Linux x64	6	Oracle	Required for building Android
5	repo	—	Android Open Source Project	Android source management tool
6	ARM® toolchain	ARM compiler Tools 5.01 update 3 (build 94)	ARM Ltd.	Toolchain for building boot images, RPM, TrustZone and SDI

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

	Item description	Version	Source/vendor	Purpose
7	Hexagon™ toolchain	5.0.07 and 5.0.09	Qualcomm/ Gnu	Toolchain for building Modem Processor Subsystem (MPSS) and Applications Digital Signal Processor Subsystem (ADSP) 5.0.07 for MPSS and 5.0.09 for ADSP
8	Python	2.6.2	Python.org	Toolchain for building boot images, RPM, ADSP, and MPSS.

2.2 Installation and setup

2.2.1 Ubuntu installation

The following instructions introduce how to install, update, and configure an Ubuntu 10.04 (64 bit) system, you can get the similar instruction at Android source website [R3] as well. You must be able to log in as root or use sudo to have root permissions during the installation.

How to install Ubuntu:

1. Create an installation CD, and install it into the computer according to the instructions at http://releases.ubuntu.com.
2. After installation, perform a software update by using one of the following options:
   - Using the GUI, select System→Administration→Update Manager
   - Using the shell command line
     i. Edit the source config file directly, as follows:

```
sudo vi /etc/apt/sources.list
```

     ii. Edit the file to enable the universe and multiverse sources and disable the Ubuntu installation CD source.
     iii. From the command line, perform the package list update and package upgrades using:

```
sudo apt-get update
sudo apt-get upgrade
```

3. Use apt-get to install the additional required packages.

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

```
$ sudo apt-get install git-core gnupg flex bison gperf build-essential zip
curl zlib1g-dev libc6-dev lib32ncurses5-dev ia32-libs x11proto-core-dev
libx11-dev lib32readline5-dev lib32z-dev libgl1-mesa-dev g++-multilib
mingw32 tofrodos python-markdown libxml2-utils xsltproc
```

4. Make bash as the default shell (Android build scripts contain bash shell dependencies which require the system default shell /bin/sh to invoke bash) using one of the following options:
   - Reconfigure the package:
     iv. Use the command:

     `sudo dpkg-reconfigure dash`

     v. Answer no.
   - Manually change the symlink /bin/sh→dash to /bin/sh→bash using the following commands:

     `sudo rm /bin/sh`

     `sudo ln -s /bin/bash /bin/sh`

NOTE: See the Ubuntu Wiki page at https://wiki.ubuntu.com/DashAsBinSh for more information.

2.2.2 Samba configuration for Windows sharing (optional)

How to configure Samba for Windows sharing:

1. Use the following command to install the Samba server and configuration manager for Windows sharing:

   `sudo apt-get install samba system-config-samba`

2. Configure the Samba server using:

```
System->Administration->Samba
     preferences->server settings:
     vmgroup, security=user authentication
     encrypt pw=yes, guest accnt=no guest accnt
add share directory=/, share name=root, description=root directory
```


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

2.2.3 JDK installation

The Sun JDK is no longer in Ubuntu's main package repository. In order to download it, you need to add the appropriate repository and indicate to the system which JDK should be used.

```
sudo add-apt-repository "deb http://archive.canonical.com/ lucid partner"
sudo apt-get update
sudo apt-get install sun-java6-jdk
```

2.2.4 Repo installation

The repo tool is a source code configuration management tool used by the Android project (see [R5]). It is a front end to git written in Python, which uses a manifest file to aid downloading code organized as a set of projects stored in different git repositories.

How to install repo:

1. Create a ~/bin directory in your home directory, or, if you have root or sudo access, install for all system users under a common location, such as `/usr/local/bin or somewhere under /opt`.
2. Download the repo script.

   ```
   $ curl https://dl-ssl.google.com/dl/googlesource/git-repo/repo
   >~/bin/repo
   ```

3. Set the repo script attributes to executable.

   ```
   $ chmod a+x ~/bin/repo
   ```

4. Make sure that the installed directory location for repo is included in your PATH.

   ```
   $ export PATH=~/bin:$PATH
   ```

5. Try running **`repo --help`** to verify installation; you should see a message similar to the following:

   ```
   $ repo --help
   usage: repo COMMAND [ARGS]
   repo is not yet installed.  Use "repo init" to install it here.
   The most commonly used repo commands are:
     init      Install repo in the current working directory
     help      Display detailed help on a command
   ```

6. For access to the full online help, install repo (**repo init**).

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

2.2.5 ARM compiler tools installation

Building the non-HLOS images requires the specific version of the ARM Compiler Tools which is indicated in Table 2-1. Linux is the recommended build environment for building all software images. However, either Windows or Linux-hosted versions will work for building the non-HLOS images. For more information about the ARM Developer Suite and toolchains, go to the ARM support website at http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.subset.swdev.coretools/index.html.

2.2.5.1 Installing ARM compiler tools on Linux host

How to install ARM compiler tools on a Linux host:

1. Obtain the required ARM toolchain identified in Table 2-1 from your ARM vendor.
2. Follow the vendor instructions to install the toolchain and flex license manager onto your Linux build system.

2.2.5.2 Installing ARM compiler tools on Windows host

If use a Windows-based build machine for creating non-HLOS builds, it provides additional details about installing the ARM toolchain on a Windows build system.

Obtain the required ARM toolchain identified in Table 2-1 from your ARM vendor, and follow the vendor instructions to install the toolchain and flex license manager onto your Windows build system.

How to install ARM compiler tools on a Windows host:

1. Access the software from https://silver.arm.com/download/download.tm?pv=1245960
2. The default installing location is `C:\Program Files (x86)\ARM_Compiler5\`
3. If necessary, change the directory where the files will be extracted to match the location where you have installed the tools. For example, the installing directory for Qualcomm is `C:\Program Files (x86)\ARM_Compiler5\bin`.
4. Confirm that the updated tools are installed by opening a DOS command prompt window and checking the versions for the compilers, linker, assembler, and fromelf.
5. Run **armcc –vsn** to check the versions. It should return the following information:

```
ARM/Thumb C/C++ Compiler, 5.01 [Build 94]
For support contact support-sw@arm.com
Software supplied by: ARM Limited

armar –-vsn
armlink –-vsn
armasm –-vsn
fromelf --vsn
```

The returned version should be build 94 for all.

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

2.2.6 Hexagon toolchain installation

Building the non-HLOS images requires the specific version of the Hexagon toolchain which is indicated in Table 2-1. Linux is the recommended build environment for building all software images. However, either Windows or Linux-hosted versions will work for building the non-HLOS images.

See [Q2] for detailed procedures to download and install the Hexagon toolchain software. Additional documentation for using the Hexagon tools can be found in [Q3].

.

QUALCOMM®
2013.07.09 at 23:19:43 PDT
liyh-gionee.com
108.171.248.77

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

3 Software Product Information

3.1 Software product identification

The software for this product line is divided into different release packages, each of these packages must be downloaded separately and combined according to the downloading instructions below. Then it can have a complete product line software set.

- From chipcode.qti.qualcomm.com [R6]:
  - Proprietary non-HLOS software releases (contains proprietary source and firmware images for all nonapps processors)
  - Proprietary HLOS release (contains proprietary source and firmware images for the apps processor HLOS)
- From codeaurora.org [R4]:
  - Open source HLOS release (contains open source for apps processor HLOS)

The proprietary non-HLOS package is an umbrella package built from a combined set of individual component releases which have already been integrated. The proprietary and open source HLOS packages need to be obtained from separate sources then combined according to the downloading instructions given in 3.3.2. Each package is identified by a unique build identification (build ID) code followed by a release version number.

Figure 3-1 illustrates how to decode the software release build ID (numbers indicate character position in the build ID).

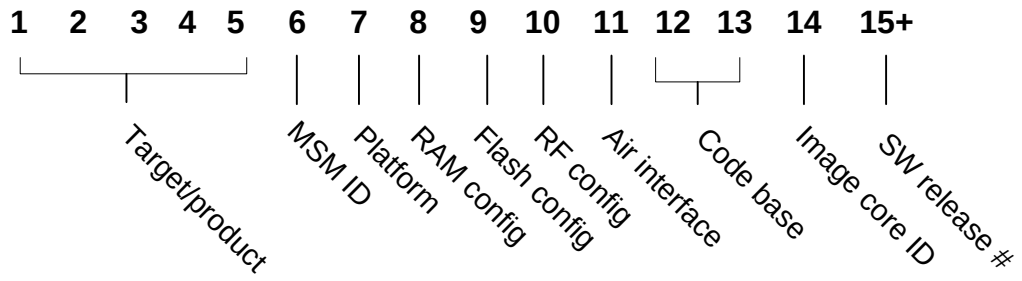

Figure 3-1 Decoding the software release build ID

Character positions 11, 12 and 13, and 14 in the build ID can take the following values:

- Character position 11 – Air interface
  - N
  - A – UMTS

- B – Multimode
- C – C2K only

- Character positions 12 and 13 – Code base
  - AZ – L4
  - LY – Linux Android
- Character position 14 – Image core ID
  - A – Apps (proprietary HLOS)
  - N – Non-HLOS umbrella package, integrated package containing each of the following:
    - M –MPSS
    - B – Boot images
    - L – Low Power Audio Subsystem and Sensors (ADSP)
    - W – Wireless Connectivity Networking Subsystem (WCNSS)
    - R – Resource Power Manager (RPM)
    - T – Trust Zone (TZ)
    - I – System Debug Image (SDI)

QUALCOMM® 2013.07.09 at 23:19:43 PDT liyh-gionee.com 108.171.248.77

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

Figure 3-2 illustrates the combined software release packages.

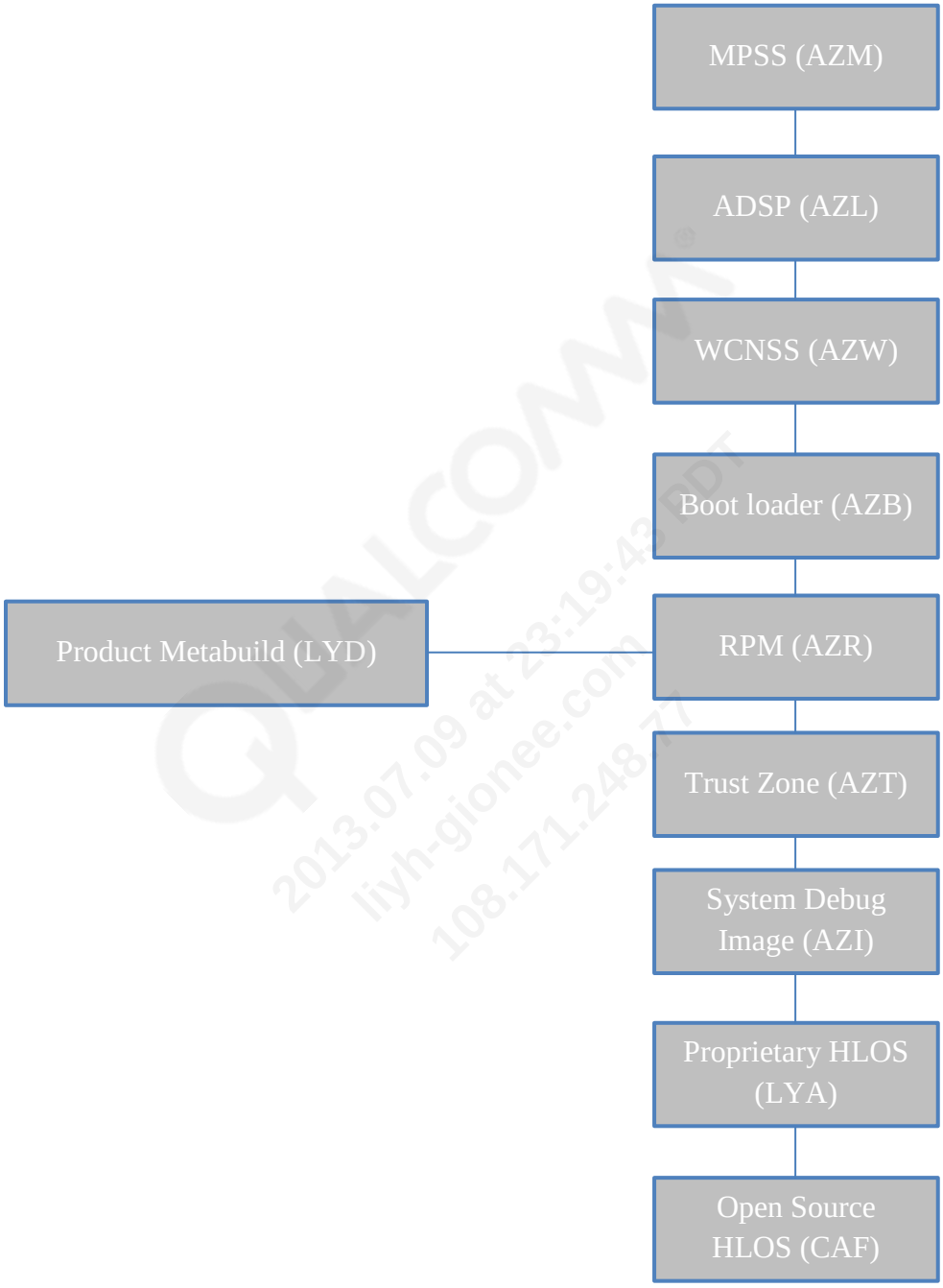

Figure 3-2 Combined software release packages

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

Table 3-1 gives the component release build properties. The compiler, Python, Perl, and Cygwin version information for each of the non-HLOS build modules is also provided. Make sure the build PC has the correct versions for each tool.

Table 3-1 Component release build properties

Component build release	Source or binary only	Toolchain required for building source	Python version	Perl version	Cygwin	Supported build hosts
Android HLOS (LYA)	Source	Android gnu toolchain	—	—	—	Linux only
MPSS (AZM)	Source	Hexagon 5.0.07	Python 2.6.2	Perl 5.8.x Linux builds only	Windows builds only; only needs tee.exe	Linux, Windows XP, and Windows 7
ADSP (AZL)	Binary	—	—	—	—	—
Boot loaders (AZB)	Source	ARM Compiler Tools 5.01 update 3 (build 94)	Python 2.6.2	Perl 5.8.x Linux builds only	Windows builds only; only needs tee.exe	Linux, Windows XP, and Windows 7
RPM (AZR)	Source	ARM Compiler Tools 5.01 update 3 (build 94)	Python 2.6.2	—	Windows builds only; only needs tee.exe	Linux, Windows XP and Windows 7 only
TZ (AZT)	Source	ARM Compiler Tools 5.01 update 3 (build 94)	Python 2.6.2	—	Windows builds only; only needs tee.exe	Linux, Windows XP, and Windows 7 only
WCNSS (AZW)	Binary	ARM Compiler Tools 5.01 update 3 (build 94)	Python 2.6.2	Not required unless you're using some tools to dump extraction. Recommended to use 5.8.x and above.	Windows builds only; only needs tee.exe	Linux, Windows XP, and Windows 7 only
SDI (AZI)	Source	ARM Compiler Tools 5.01 update 3 (build 94)	Python 2.6.2	—	Windows builds only; only needs tee.exe	Linux, Windows XP and Windows 7 only

3.2 Access to the software

This section provides instructions to obtain (download) software from the designated authoritative distribution sources. Once the software download is completed, go directly to 3.3 for building the software.

3.2.1 Downloading Qualcomm Technologies, Inc. proprietary software

The non-HLOS and proprietary HLOS software releases are distributed on the Qualcomm ChipCode website [R6], Qualcomm ChipCode website is part of the QUALCOMM new distribution system, which replaces existed Documents and Downloads for source code distribution.

Designated contact points at customer sites are given access to download software for the customers who have licenses.

For customers who are new to Qualcomm ChipCode website, please read [Q14] and [Q15], which provide Chipcode website overview and instructions on installing Git on your system and performing basic and advanced methods of downloading software code, documents, and software tools. We also provide online help wiki at [R7], there are all current, known issues, hot topics and a tutorial video link.

To make the build process smooth, it is desirable to create a top-level directory in build PC and unzip each archive file to generate the below directory structure. In the following example, <target_root> is the top-level directory.

```
<target_root>          /common/
                       /adsp_proc/
                       /boot_images/
                       /common/
                       /debug_image/
                       /modem_proc/
                       /rpm_proc/
                       /trustzone_images/
                       /wcnss_proc/Q
                       /LINUX/
                       contents.xml
```

NOTE: It is very important to ensure that the `contents.xml` file is located in the root folder as shown.

3.2.2 Downloading open source HLOS software

The Linux Board Support Package (BSP) release will be obtained in two parts, a proprietary release from the Qualcomm ChipCode website [R6] and an open source release from the Code Aurora Forum (CAF) site.

How to download the open source software:

1. Look in your Release Notes and note the build ID indicated on the title page.
2. Go to https://www.codeaurora.org/xwiki/bin/QAEP/ and find the release branch containing the matching build ID in the branch releases table. The release area that lists all builds is at: https://www.codeaurora.org/xwiki/bin/QAEP/release
3. In an empty workspace directory, issue the repo init command with the correct branch and manifest as indicated in the branch releases table.

```
$ repo init -u git://codeaurora.org/platform/manifest.git -b release -m
[manifest] –repo-url=git://codeaurora.org/tools/repo.git
```

4. Type the repo sync command.

```
$ repo sync
```

5. After the repo sync finishes, copy the vendor/qcom/proprietary directory tree from the proprietary HLOS release into the open source HLOS source tree contained in your workspace.

```
$cp -r <LYA_build_location>/HY11-<build_id>/LINUX/android/* .
```

3.3 Compilation/build procedures

3.3.1 Non-HLOS

3.3.1.1 Setting build shell environment

Before issuing the non-HLOS build commands, certain shell environment settings must be set to ensure the correct executable path and toolchain configuration. The specific environment settings will vary somewhat based upon your host software installation, but it might be similar to the example “myenviron_amss.sh” script below (for Linux), which sets the path to point to the ARM toolchain lib, include, bin, and license file configuration.

```
#
# myenviron_amss_8626
#
export ARMLMD_LICENSE_FILE=<mylicense_file>@<mylicense_server>

ARM_COMPILER_PATH=/pkg/qct/software/arm/RVDS/5.01bld94/bin64
PYTHON_PATH=/usr/local/python/2.6.2/bin
MAKE_PATH=/pkg/gnu/make/3.81/bin

export ARMROOT=/pkg/qct/software/arm/RVDS/5.01bld94
export ARMTOOLS=ARMCT5.01
export ARMLIB=$ARMROOT/lib
export ARMINCLUDE=$ARMROOT/include
export ARMINC=$ARMINCLUDE
export ARMBIN=$ARMROOT/bin64

export PATH=$MAKE_PATH:$PYTHON_PATH:$ARMBIN:$ARM_COMPILER_PATH:$PATH
export ARMHOME=$ARMROOT
export HEXAGON_ROOT=/pkg/qct/software/hexagon/releases/tools
```


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

```
export HEXAGON_RTOS_RELEASE=5.0.01
export HEXAGON_Q6VERSION=v4
export HEXAGON_IMAGE_ENTRY=0x08400000
```

3.3.1.2 Building MPSS

To build MPSS (<target_root> is the top-level directory that is created in 3.2.1):

1. Verify that the environment variables below have been set as indicated in 3.3.1.1.
   - HEXAGON_ROOT
   - HEXAGON_RTOS_RELEASE
   - HEXAGON_Q6VERSION
   - HEXAGON_IMAGE_ENTRY
2. Navigate to the following directory:

   cd <target_root>/modem_proc/build/ms

3. Depending on your build environment, choose one of the following options:

Table 3-2 Build MPSS instruction

Build environment	Command
Linux	**build.sh 8x26.gen BUILD_ID=AAAAANAZ BUILD_VER=<version_number>**
Windows	**build.cmd 8x26.gen BUILD_ID=AAAAANAZ BUILD_VER=<version_number>**

NOTE: In the above commands, <version_number> is determined by the last four (numerical) characters in the build ID; for example, for a release with build ID M8626AAAAANLYD1010, <version_number> is 1010.

3.3.1.3 Building boot loaders

How to build the boot loaders:

1. For Linux, verify that the paths below have been set. Refer to **setenv.sh** in your boot build: `<target_root>\boot_images\build\ms\setenv.sh`.

```
export ARMLMD_LICENSE_FILE=<LICENSE FILE INFO>
   export ARM_COMPILER_PATH=/<Path to compiler>/arm/RVDS/5.01bld94/bin64
   export PYTHON_PATH=/<Path to python>/python/2.6.6/bin
   export MAKE_PATH=/<Path to make>/gnu/make/3.81/bin
   export ARMTOOLS=ARMCT5.01
   export ARMROOT=/<Path to compiler>/arm/RVDS/5.01bld94
   export ARMLIB=$ARMROOT/lib
```


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

```
export ARMINCLUDE=$ARMROOT/include
export ARMINC=$ARMINCLUDE
export ARMBIN=$ARMROOT/bin64
export PATH=$MAKE_PATH:$PYTHON_PATH:$ARM_COMPILER_PATH:$PATH
export ARMHOME=$ARMROOT
export_armlmd_license
```

2. For Windows, verify that the following paths have been set:

```
set ROOT_DRIVE=C:
set APPS_DIR=Apps
set APPS_PATH=%ROOT_DRIVE%\%APPS_DIR%
set UTILROOT=%ROOT_DRIVE%\util
set GNUPATH=%UTILROOT%\cygwin\bin
set CRMPERL=C:\CRMApps\apps\Perl\bin
set PERLPATH=C:\perl\bin
set PYTHON_PATH=C:\CRMApps\apps\Python262
set ARMTOOLS=ARMCT501
set ARMROOT=%APPS_PATH%\ARMCT5.01\94
set ARMCC5BIN=%ARMROOT%\bin
set ARMCC5INC=%ARMROOT%\include
set ARMCC5LIB=%ARMROOT%\lib
set
path=%PERLPATH%;%CRMPERL%;%PYTHON_PATH%;%GNUPATH%;%ARMCC5BIN%;%UTILROOT%
;%path%
```

3. Verify that the ARM license path (ARMLMD_LICENSE_FILE) is updated as indicated in 3.3.1.1.
4. Navigate to the following directory:

   cd <target_root>/boot_images/build/ms

   Where <target_root> is the top-level directory that is created in 3.2.1.

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

5. Depending on your build environment/release, choose one of the following build command options:

Table 3-3 Build boot loader instruction

Build environment/release	Command
Linux build environment	
MSM8626 (Rel 1010 and later)	Build boot images **$ ./build.sh TARGET_FAMILY=8x26 BUILD_ID=FAAAANAZ** Cleaning the build **$ ./build.sh -c TARGET_FAMILY=8x26 BUILD_ID=FAAAANAZ** Depending on your configuration, you may need to edit the script boot_images/build/ms/build_8626.sh to change **if [-e "setenv.sh"]; then** **- source setenv.sh** **+ source ./setenv.sh** **fi**
Windows build environment	
MSM8626 (Rel 1010and later)	Building boot images **build.cmd TARGET_FAMILY=8x26 BUILD_ID=FAAAANAZ** Cleaning the build **build.cmd -c TARGET_FAMILY=8x26 BUILD_ID=FAAAANAZ**

3.3.1.4 Building TrustZone images

To build the MSM8x26 TrustZone images:

1. Navigate to the following directory:

```
cd <target_root>/trustzone_images/build/ms
```

2. Run the following command to build all images:

Table 3-4 Build trustzone instruction

Build environment	Command
Linux	**build.sh CHIPSET=msm8x26 tz**
Windows	**build.cmd CHIPSET=msm8x26 tz**

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

3.3.1.5 Building RPM

Use the following commands to build RPM (<target_root> is the top-level directory). Ensure that your tools are the versions specified in Table 3-1.

Note that building under Linux may not work for early ES releases.

1. Open a command prompt and change to the following directory:

```
cd <target_root>\rpm_proc\build
```

2. Depending on your build environment, choose one of the following options:

Table 3-5 Build RPM instruction

Build environment	Command
Linux	`./build_8x26.sh`
Windows	`build_8x26.bat`

3. To clean the build, choose one of the following options:

Table 3-6 Clean RPM instruction

Build environment	Command
Linux	`./build_8x26.sh –c`
Windows	`build._8x26.bat –c`

NOTE: rpm.mbn can be found at rpm_proc/build/ms/bin/AAAAANAA.

3.3.1.6 Building SDI

Use the following commands to build SDI (<target_root> is the top-level directory). Ensure that your tools are the versions specified in Table 3-1.

1. Open a command prompt and change to the following directory:

```
cd <target_root>\build_image\build\ms
```

2. Depending on your build environment, choose one of the following options:

Table 3-7 Build SDI instruction

Build environment	Command
Linux	`./b8x26.sh TARGET_FAMILY=8x26 sdi BUILD_VER=<major version> BUILD_ID=AAAAANAZ BUILD_MIN=<minor version>`
Windows	`b8x26.cmd TARGET_FAMILY=8x26 sdi BUILD_VER=<major version> BUILD_ID=AAAAANAZ BUILD_MIN=<minor version>`

NOTE: In the above commands, major and minor versions are identical to the Qualcomm release number if possible. For example. if M8626AAAAANLYD1010 is the release, keep BUILD_VER=1 and BUILD_MIN=10.

NOTE: sdi.mbn can be found at build_image/build/ms/bin/AAAAANAAZ

3.3.1.7 Building ADSP

It provides ADSP images, don't need to build it from source codes, while in some case, use the following commands to build ADSP (<target_root> is the top-level directory). Ensure that the tools are the versions specified in Table 3-1.

1. Open a command prompt and change to the following directory:

```
cd <target_root>\adsp_proc
```

2. Depending on the build environment, choose one of the following options:

Table 3-8 Build ADSP instruction

Build environment	Command
Linux	`./build_adsp_image.sh "0x70ffffff" CHIPSET=msm8x26 all`
Windows	`build.cmd 0x70ffffff "CHIPSET=msm8x26" all`

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

3.3.1.8 Updating NON-HLOS.bin

If any of MPSS, ADSP, or WCNSS is recompiled, use the following commands to update the NON-HLOS.bin file with the new images (<target_root> is the top-level directory that is created in 3.2.1):

1. Navigate to the following directory:

```
cd <target_root>/common/build
```

2. Enter the command:

```
python update_common_info.py
```

3.3.2 Apps processor Android HLOS

How to build the apps processor Android HLOS:

1. In a BASH shell, navigate to the Android source tree base directory.

```
cd <build id>/LINUX/android
```

2. Enter the following command to configure the build environment shell settings.

```
source build/envsetup.sh
```

NOTE: You must use the source command, so the environment settings are defined in the current shell.

3. Enter the `choosecombo` command to select the build configuration, or enter with no parameters to see an interactive menu for making selections.

```
choosecombo 1 msm8226 userdebug
```

4. Run `make` to start the build (The command shows below with –j 4 option to run parallel builds for faster build times on a multicore build machine).

```
make -j4
```


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

4 Firmware Programming

4.1 Equipment and software

In addition to the items listed in Table 2-1 for software environment requirements, Table 4-1 lists the additional equipment and software required for programming firmware images into a target device.

Table 4-1 Equipment and software required for programming firmware images

	Item description	Version	Source/vendor	Purpose
1	Qualcomm QPST	2.7.405 or later	Qualcomm, Inc.	Programming firmware images using QPST
2	QXDM Professional™	3.14.447 or later	Qualcomm, Inc.	Programming NV Item values, reading diagnostic, etc
2	Lauterbach TRACE32 (T32) CortexA8/Krait License Extension	LA-7843X or LA-7843	Lauterbach GmbH	Programming firmware images using JTAG and applications processor debugging
3	Lauterbach T32 QDSP6 License Extension	LA-3741A	Lauterbach GmbH	Modem software processor, firmware processor,ADSP debugging using JTAG
4	Lauterbach T32 Cortex-M3 License Extension	LA-7844X or LA-7844	Lauterbach GmbH	Programming firmware images using JTAG and RPM debugging using JTAG
5	Lauterbach T32 ARM9™ License Extension	LA-7742X	Lauterbach GmbH	Venus and WCNSS debugging using JTAG
6	Lauterbach T32 Windows	Aug 2012 Software Version: R.2012.08.000040902 Build: 38589--40902.	Lauterbach GmbH	Programming firmware images and debugging using JTAG
7	Android SDK tools (Host USB drivers, adb, fastboot)	r10 or higher ADB 1.0.29 or later	Android Open Source Project	Windows host USB driver for adb and fastboot; adb and fastboot tools for Windows
8	Qualcomm USB Network Driver Combo	1.0.80 or later	Qualcomm, Inc.	Windows host USB drivers for Qualcomm composite devices

4.2 Installation and setup

This section describes step-by-step procedures that the user must follow to perform the installation and configuration for all equipment and software. The equipment and software are

needed to create a working environment which is able to program each firmware image and device.

4.2.1 Installing T32

It's recommended to use QPST for firmwork download. T32 is necessary when QPST download doesn't work. The June 2012 Build 37825 version of T32 is the mandatory minimum revision which is needed for binary download and debugging. The T32 links under `common\t32\t32_dap\` should be used for binary download and debugging.

By default, these files assume the T32 installing directory is `C:\T32`. If T32 is installed in a different directory, the .lnk shortcut files must be modified.

How to modify the.lnk shortcut files:

1. Locate the .lnk shortcut files.
2. Right-click the mouse and select **Properties**.
3. In the Target field, change the path to the proper path of t32marm.exe. The default path is `C:\t32\t32marm.exe`.

4.2.2 Installing Android adb, fastboot, and host USB interface

4.2.2.1 USB setup in Windows

Android CDP support requires the following USB device support:

- Android USB Driver (android_winusb.inf)
  - Android ADB Interface
  - Android Boot Loader Interface (fastboot)
- Qualcomm Composite USB Modem/Serial Driver (qcmdm.inf, qcser.inf)
  - Qualcomm HS-USB Android DIAG
  - Qualcomm HS-USB Android Modem
  - Qualcomm HS-USB Android GPS (NMEA)
- Qualcomm Composite USB Network Combo driver (qcnet.inf)
  - Qualcomm Wireless HS-USB Ethernet Adapter

Before installing the drivers, it is necessary to edit the **qcmdm.inf** and **qcser.inf** files to make sure they contain support for the Android SURF VID/PID with appropriate entries in each section as indicated in 4.2.2.2.

Also see [Q11] and [Q12] for additional details on Windows USB driver installation, troubleshooting, and for rebuilding the Windows USB host driver from source.

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

4.2.2.2 Installing Android SDK for adb, fastboot, and USB driver for Windows

How to install the Android SDK platform and USB driver components on a Windows machine:

1. Go to http://developer.android.com/sdk/win-usb.html.
2. Follow the instructions to install the SDK and USB driver.
3. Right-click **My Computer,** and select **Properties→Advanced→Environment Variables**, then set the path to include the `c:\android-sdk-windows\tools` directory.
4. The Android USB driver for adb and fastboot needs to add the Qualcomm SURF VID/PID which supports the connection to Sthe URF. Edit the file `android-sdk-windows\usb_driver\android_winusb.inf` to add the Qualcomm VID/PID lines to each section.

```
android_winusb.inf
[Google.NTx86]
;Qualcomm SURF/FFA
%SingleAdbInterface%        = USB_Install, USB\VID_05C6&PID_9025
%CompositeAdbInterface%     = USB_Install, USB\VID_05C6&PID_9025&MI_01
%SingleBootLoaderInterface% = USB_Install, USB\VID_18D1&PID_D00D

[Google.NTamd64]
;Qualcomm SURF/FFA
%SingleAdbInterface%        = USB_Install, USB\VID_05C6&PID_9025
%CompositeAdbInterface%     = USB_Install, USB\VID_05C6&PID_9025&MI_01
%SingleBootLoaderInterface% = USB_Install, USB\VID_18D1&PID_D00D
```

In addition, make sure that there are matching entries under the **[Strings]** section.

```
[Strings]
SingleAdbInterface          = "Android ADB Interface"
CompositeAdbInterface       = "Android Composite ADB Interface"
SingleBootLoaderInterface   = "Android Bootloader Interface"
```

5. The adb client (adb.exe) supports a built-in list of recognized USB VID/PID devices. To add the SURF or another device to the list of recognized devices which is not included in the built-in support list, create a `%USERPROFILE%\.android` directory if it does not exist.
6. Navigate to the `%USERPROFILE%\.android` directory.
7. In the `%USERPROFILE%\.android` directory, create `/edit the adb_usb.ini` file. If the file already exists, it will contain a **DO NOT EDIT** message. Disregard this message and edit the file anyway. Add a line containing 0x05C6 to the end of the file.

NOTE: Do not run **android update adb** or it will reset the contents of this file and overwrite the line just added.

After editing, the **adb_usb.ini** file should look like this:

```
# ANDROID 3RD PARTY USB VENDOR ID LIST -- DO NOT EDIT.
# USE 'android update adb' TO GENERATE.
# 1 USB VENDOR ID PER LINE.
0x05C6
```

8. Obtain the latest version of the Qualcomm Composite USB driver from Documents and Downloads. (To include network interface support, use the Qualcomm Composite USB Network Combo driver.)

Android debugging is enabled/disabled in user space with composition 9025/9026 respectively.

```
qcmdm.inf
[Models]
%QUALCOMM90252% = Modem2, USB\VID_05C6&PID_9025&MI_02
%QUALCOMM90261% = Modem2, USB\VID_05C6&PID_9026&MI_01

[Models.NTamd64]
%QUALCOMM90252% = Modem2, USB\VID_05C6&PID_9025&MI_02
%QUALCOMM90261% = Modem2, USB\VID_05C6&PID_9026&MI_01

[Models.NTia64]
%QUALCOMM90252% = Modem2, USB\VID_05C6&PID_9025&MI_02
%QUALCOMM90261% = Modem2, USB\VID_05C6&PID_9026&MI_01

[Strings]
QUALCOMM90252 = "Qualcomm Android Modem 9025"
QUALCOMM90261 = "Qualcomm HS-USB Android Modem 9026"

qcser.inf
[QcomSerialPort]
%QcomDevice90250%  = QportInstall00, USB\VID_05C6&PID_9025&MI_00
%QcomDevice90253%  = QportInstall00, USB\VID_05C6&PID_9025&MI_03
%QcomDevice90260%  = QportInstall00, USB\VID_05C6&PID_9026&MI_00
%QcomDevice90262%  = QportInstall00, USB\VID_05C6&PID_9026&MI_02

[QcomSerialPort.NTia64]
%QcomDevice90250%  = QportInstall00, USB\VID_05C6&PID_9025&MI_00
%QcomDevice90253%  = QportInstall00, USB\VID_05C6&PID_9025&MI_03
%QcomDevice90260%  = QportInstall00, USB\VID_05C6&PID_9026&MI_00
%QcomDevice90262%  = QportInstall00, USB\VID_05C6&PID_9026&MI_02
```


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

```
[QcomSerialPort.NTamd64]
%QcomDevice90250%  = QportInstall00, USB\VID_05C6&PID_9025&MI_00
%QcomDevice90253%  = QportInstall00, USB\VID_05C6&PID_9025&MI_03
%QcomDevice90260%  = QportInstall00, USB\VID_05C6&PID_9026&MI_00
%QcomDevice90262%  = QportInstall00, USB\VID_05C6&PID_9026&MI_02

[Strings]
QcomDevice90250 = "Qualcomm HS-USB Android DIAG 9025"
QcomDevice90253 = "Qualcomm HS-USB Android GPS (NMEA)9025"
QcomDevice90260 = "Qualcomm HS-USB Android DIAG 9026"
QcomDevice90262 = "Qualcomm HS-USB Android GPS (NMEA)9026"

qcnet.inf
[QCOM]
qcwwan.DeviceDesc90254 = qcwwan.ndi, USB\VID_05C6&PID_9025&MI_04
qcwwan.DeviceDesc90263 = qcwwan.ndi, USB\VID_05C6&PID_9026&MI_03

[QCOM.NTia64]
qcwwan.DeviceDesc90254 = qcwwan.ndi, USB\VID_05C6&PID_9025&MI_04
qcwwan.DeviceDesc90263 = qcwwan.ndi, USB\VID_05C6&PID_9026&MI_03

[QCOM.NTamd64]
qcwwan.DeviceDesc90254 = qcwwan.ndi, USB\VID_05C6&PID_9025&MI_04
qcwwan.DeviceDesc90263 = qcwwan.ndi, USB\VID_05C6&PID_9026&MI_03

[Strings]
qcwwan.DeviceDesc90254      = "Qualcomm Wireless HS-USB Ethernet Adapter
9025"
qcwwan.DeviceDesc90263      = "Qualcomm Wireless HS-USB Ethernet Adapter
9026"
```

4.2.2.3 USB driver setup in Linux

USB driver modifications are required to set up the Android Debug Bridge (ADB) in Linux.

How to set up ADB:

1. Navigate to the following directory:

   cd /etc/udev/rules.d/

2. Enter the command:

   sudo vi 50-android.rules

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

The result should be similar to the following:

```
#Sooner low-level bootloader
SUBSYSTEM=="usb", SYSFS{idVendor}=="18d1", SYSFS{idProduct}=="d00d",
MODE="0664", GROUP="plugdev"
# adb composite interface device 9025
SUBSYSTEM=="usb", SYSFS{idVendor}=="05C6", SYSFS{idProduct}=="9025",
MODE="0664", GROUP="plugdev"
```

3. After editing the file, see the list of target devices connected to the Linux box, type:

```
lsusb
```

4.2.2.4 Installing adb and fastboot in Linux

The adb and fastboot executable for Linux are located in the `android\out\host\linux-x86\bin` directory in the Android software release after a build is complete. This executable is built as part of the standard compile process. To run adb or fastboot, sudo or root access on the Linux machine may be required.

1. If the `android\out\host\linux-x86\bin` directory is not in the executable search path, add it. If it's already in the executable search path, skip to Step 2:
   a. Type the command:

      `source build/envsetup.sh`

   b. Type the command:

      `choosecombo 1 msm8226 userdebug`

2. Verify that fastboot has properly flashed the Android images to the target, type the command:

   `sudo fastboot devices`

3. Verify that device is displayed by fastboot in response to typing in the fastboot devices command.

4.3 Programming procedures

This section describes the procedures for programming and reprogramming each firmware image and device.

4.3.1 Programming eMMC with QPST

If no image is Flashed to eMMC (this is the situation for the first-time binary download in the factory line), the PBL enumerates USB as the Qualcomm Sahara download interface and waits

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

for the download command from the host PC. In this case, the minimum initial binary is needed to be Flashed so that the rest of the Android images can be downloaded.

NOTE: On the MSM8626 CDP, you can force the device into this mode by erasing the device entirely or using the dip switch S7 pin 3.

NOTE: Your version of QPST meets the minimum requirement mentioned in 4.1.

1. Create an XML file named **sahara.xml** with the contents as below. It must contain the entire path to the emmcbld programmer.

```
<?xml version="1.0" encoding="utf-8"?>
<sahara_config>
<images>
<image image_id="13" image_path="<boot_build_path>
\boot_images\build\ms\bin\EMMCBLD\MPRG8626.mbn" programmer="true" />
</images>
</sahara_config>
```

2. Open QPST configuration. You should see the device enumerated in Download Mode.

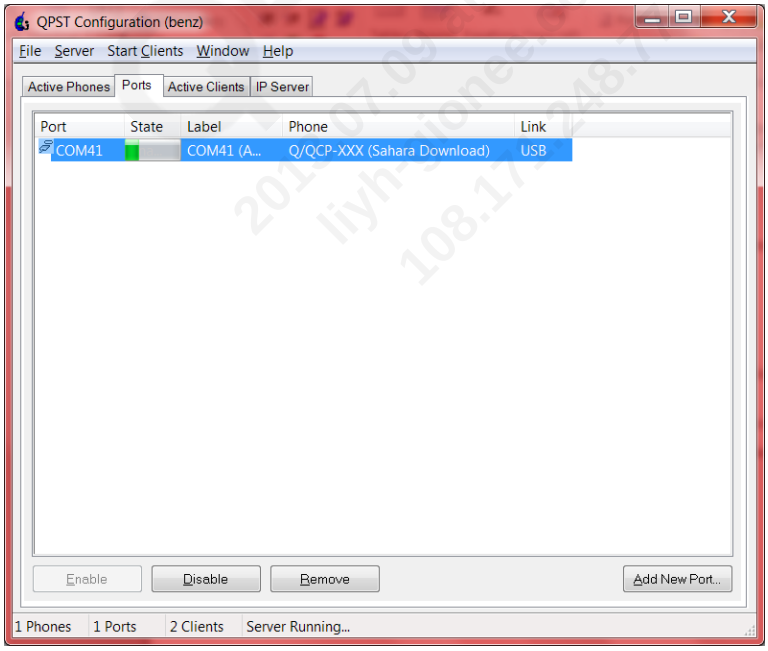

Figure 4-1 QPST configuration

3. Open QPST eMMC Software Download Application.
4. Browse to select the port on which the device enumerated in Q/QCP-XXX (Sahara Download).
5. Make sure Program Boot Loaders and Program MMC device are checked.

6. Select the **sahara.xml** in the top box for Sahara XML file. The configuration should be as shown below.

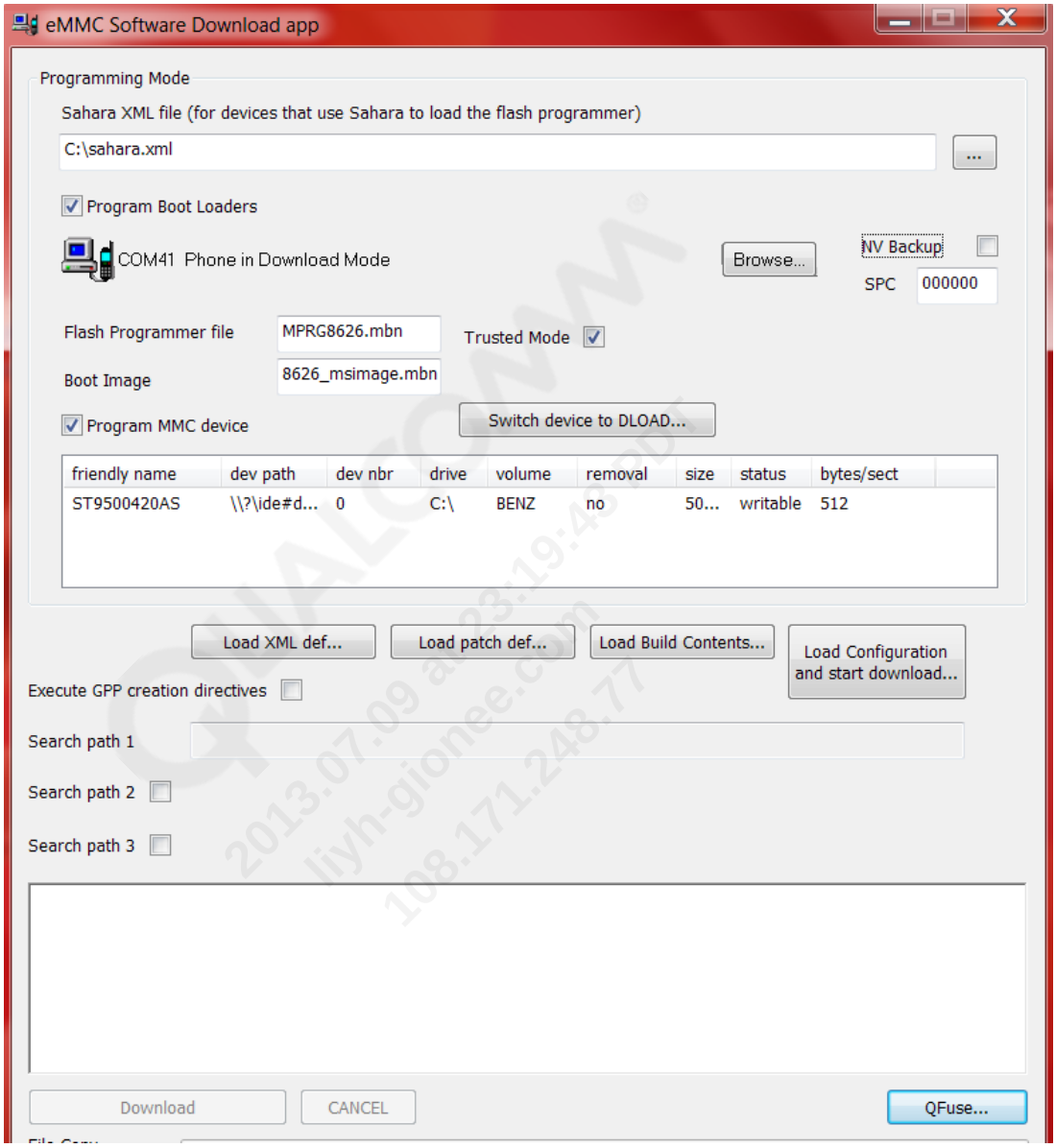

Figure 4-2 eMMC software download application

7. Select the **contents.xml** from the metabuild for Load XML definition.

8. Click **Load Build Contents**. It should look as shown below.

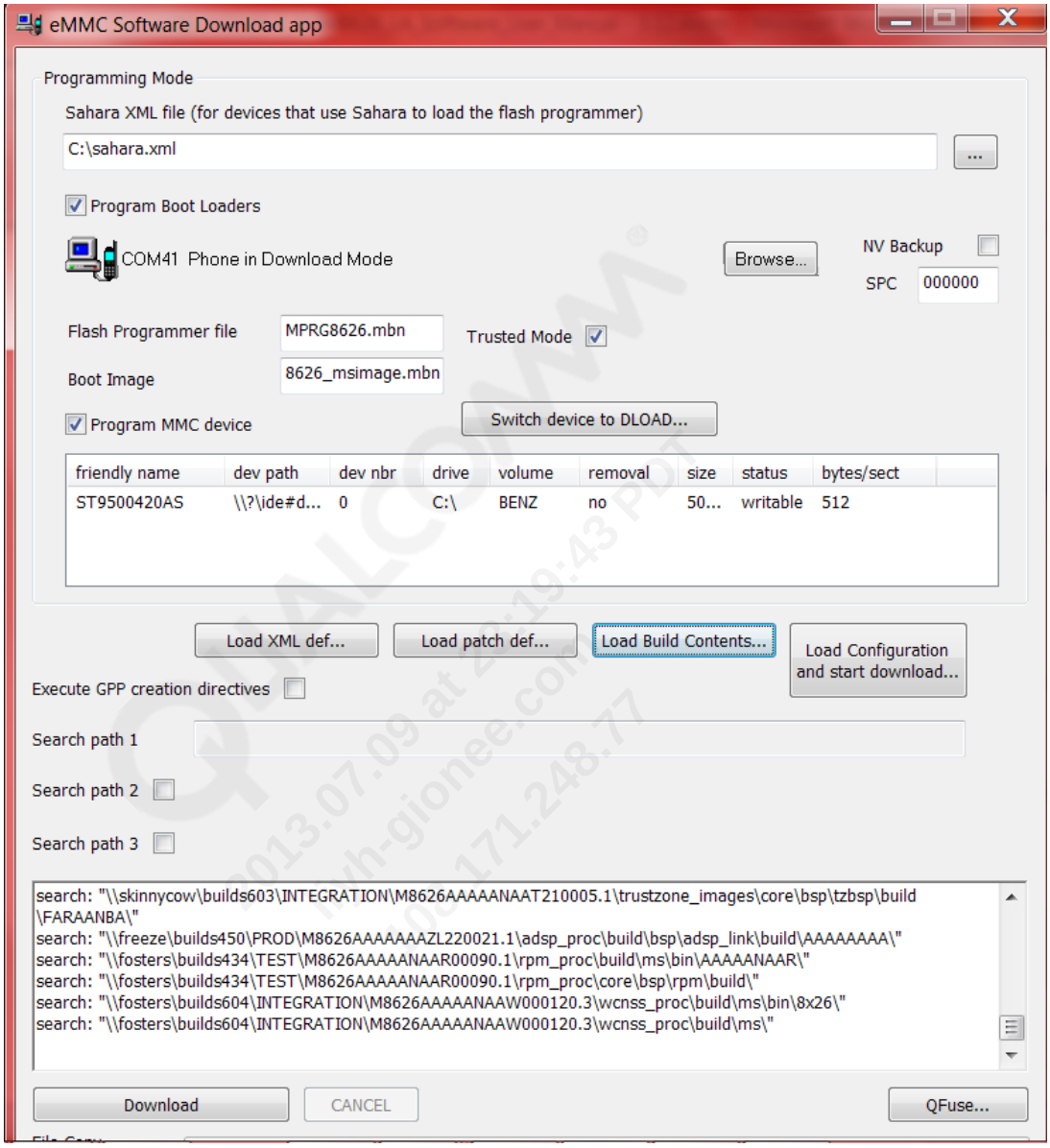

Figure 4-3 eMMC software download emmcbld images

9. Click **Download**.

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

10. Once the download of emmcbld images completes, the device should reset into mass storage mode, and QPST should continue on to load the rest of the build. It should look as shown below.

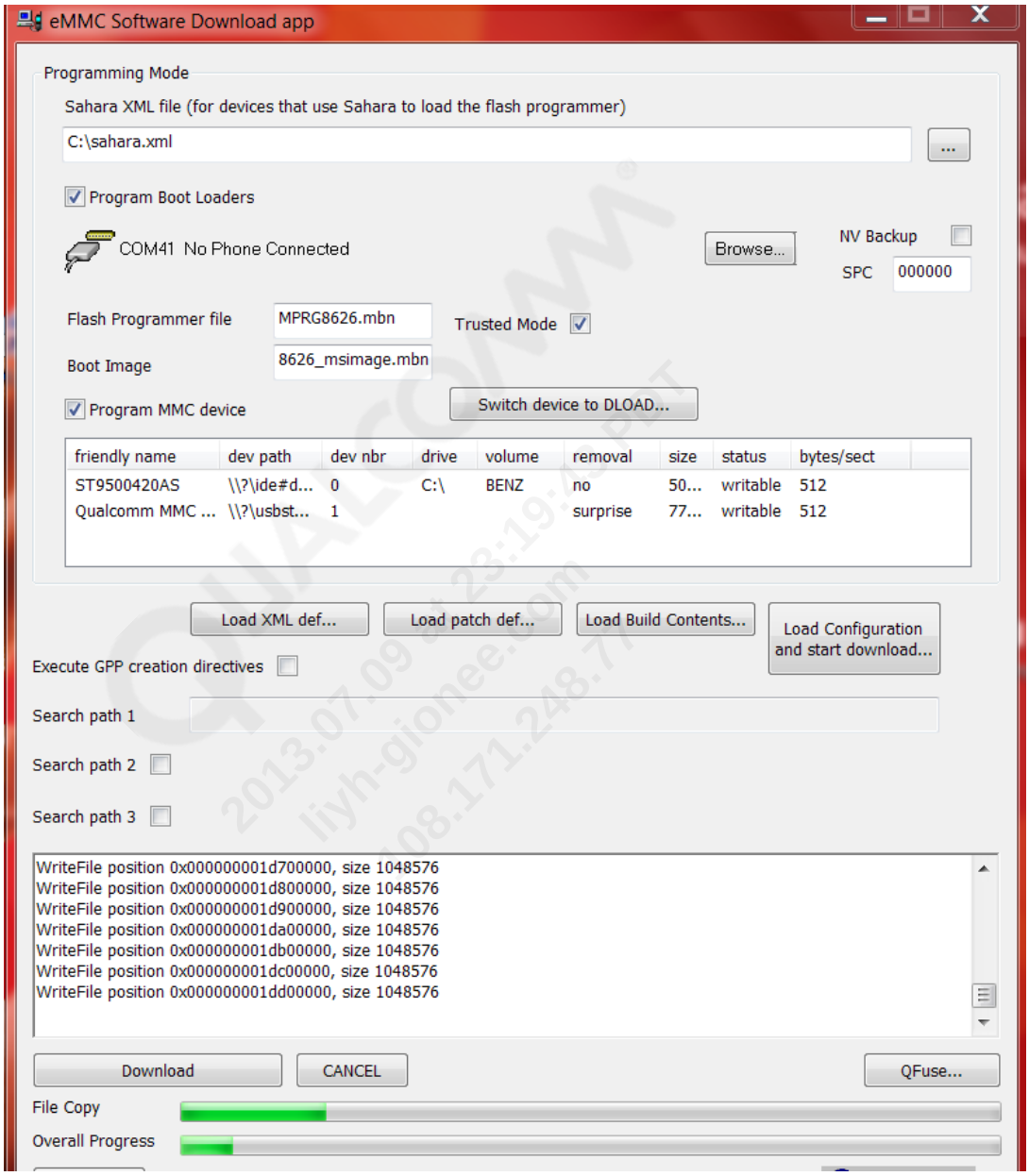

Figure 4-4 eMMC software download rest images

11. After the download is complete, the device resets and the ‘format disks window’ pops up. Cancel all the pop-ups and reset the device (disconnecting the USB cable) to boot-up to Android.

NOTE: If you use the dip switch S7-3 to force emergency mode, you must switch it back at this point and power cycle the device. Otherwise, the device will go back into emergency download mode.

NOTE: It's recommended flash CDT if it's first download on a blank eMMC. CDT provides platform/device-dependent data, such as Platform ID, DDR hardware parameters, etc. If sbl1 succeeds read the CDT from eMMC, will update the default `config_data_table` array which linked into the build at compile time. Below is methods of programming CDT to eMMC boot partition:

CDT Programming from QPST tool:

1. Needs the following to program the CDT with QPST tool,

   Flash programmer: MPRG826.mbn

   XML file: to load the flash programmer, use sahara.xml

   XML file: to load the configuration file that specifies the search paths and file names for rawprogram and patch xml files. use qrd_prog_cfg.xml with the contents as below:

```
<?xml version="1.0"?>
<configuration>
<options>
<!-- NOTE: Defaults are shown here -->
VERBOSE_FEEDBACK        = true
SEARCH_CWD_LAST         = true
HEX_PROGRAMMER          = MPRG8626.mbn
</options>
<search_paths>
<!-- NOTE: Search paths IN ORDER, as in first look here, then there,
then there etc -->
<!-- NOTE: CWD is an implied search path, and it is always last -->
emmc_cdt_program_script\cdt_prog_xml
emmc_cdt_program_script\cdt_bin
</search_paths>
<rawprogram>
rawprogram2_qrd.xml
</rawprogram>
<patch>
patch2.xml
</patch>
</configuration>
```

   XML file: these files can be found in release or apply for customer support services. The rawprogram2_qrd.xml, patch2.xml used to load the CDT binary which created by CDT cdt_generator.py and CDT xml:

   `cd boot_images\core\boot\secboot3\scripts`

   `python cdt_generator.py qrd_1.0_platform_jedec_lpddr2_single_channel.xml qrd_1.0_platform_jedec_lpddr2_single_channel.bin`

2. Make sure phone in Emergency Download mode

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

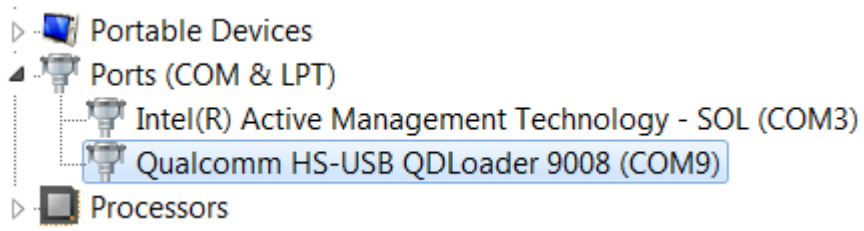

Figure 4-5 Emergency Download Port

3. Open eMMC Software Download app (run as administrator) , provide the sahara xml file , uncheck "Program Boot Loaders", "Program MMC device" and "NV Backup".
4. Click on "Load Configuration and start download..." and select qrd_prog_cfg.xml to program the CDT as shown below:

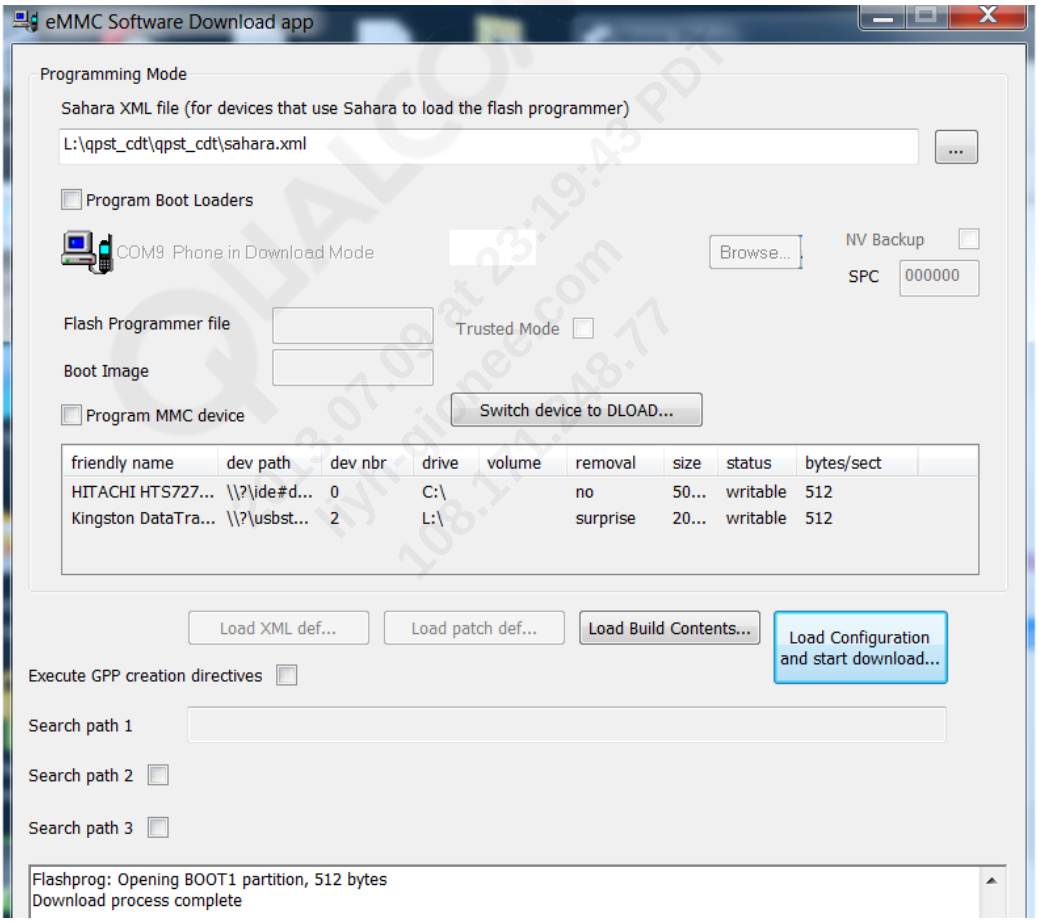

Figure 4-6 Flash CDT with QPST

Programming CDT Using JTAG:

1. Setup T32 debug environment first(refer to 4.3.4)
2. Run the CDT script in APPS0 session, you will see the success information as below.

```
cd.do
C:\emmc_cdt_program_script\qrd_1.0_lpddr2_single_channel_emmc_cdt_program.cmm
```


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

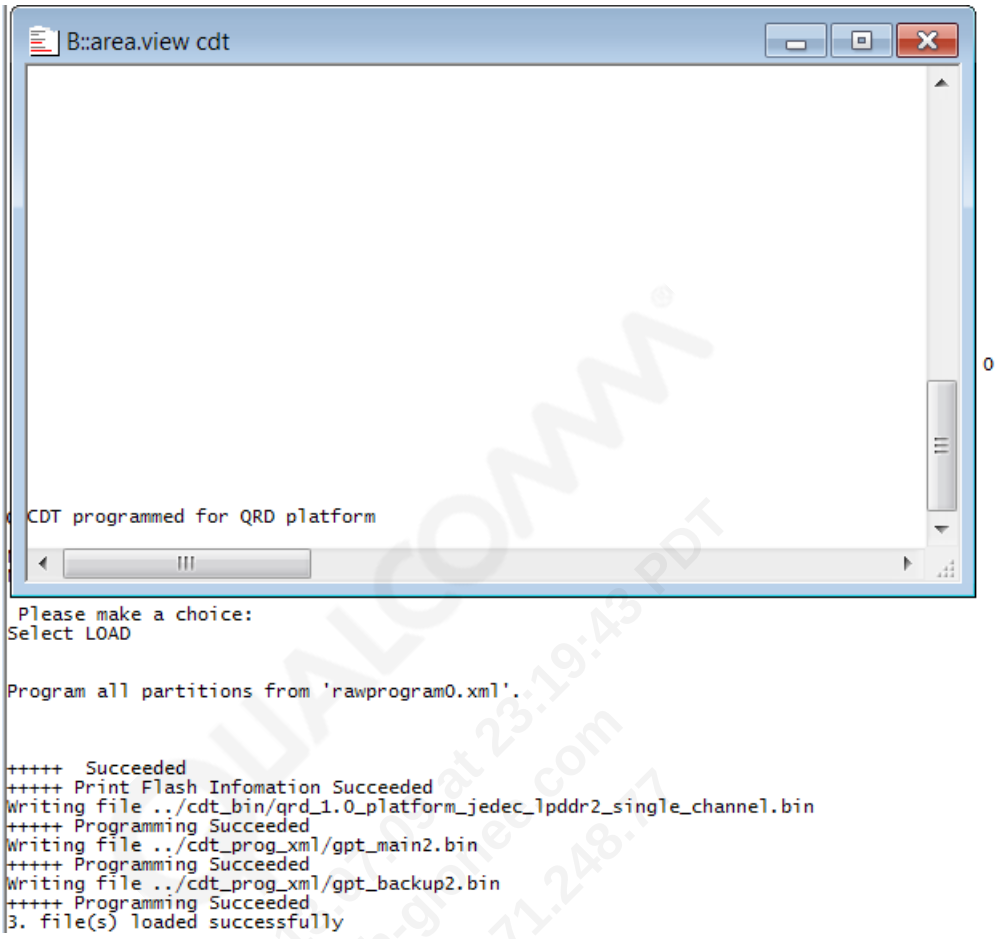

Figure 4-7 Flash CDT with T32

4.3.2 Programming system images using fastboot

Before programming the system images, use Fastboot. The Android boot loaders must already be flashed on the target:

1. Plug the USB cable into the target.
2. Depending on your build environment, choose one of the following options:
   - From the Windows command shell, run:

     ```
     fastboot devices
     ```

   - From Linux:
     i. Navigate to the following directory:

     ```
     cd <AndroidRoot>/LINUX/device/out/host/linux-x86/bin
     ```

     ii. Run:

```
sudo fastboot devices
```

The list of registered devices is shown.

3. Once the device is detected, flash the binaries to the target. The following commands will run all of the Fastboot steps at once.

```
cd <target_root>/common/build
fastboot_all.py
```

Each binary can also be flashed selectively through the following fastboot command options:

```
fastboot flash modem  <path to NON-HLOS.bin> or <path to APQ.bin>
fastboot flash sbl1 <path to sbl1.mbn>
fastboot flash rpm <path to rpm.mbn>
fastboot flash tz <path to tz.mbn>
fastboot flash aboot <path to emmc_appsboot.mbn >
fastboot flash boot <path to boot.img>
fastboot flash system <path to system.img>
fastboot flash userdata <path to userdata.img>
fastboot flash persist <path to persist.img>
fastboot flash recovery <path to recovery.img>
```

To derive a list of all fastboot partitions supported by fastboot programming, refer to the source code in `LINUX/android/bootable/bootloader/lk/platform/msm_shared/mmc.c`.

4.3.3 Flashing applications to Android using ADB

How to flash applications to Android using ADB:

1. Plug the USB cable into the target.
2. Navigate to the following directory:

   ```
   cd <root>/LINUX/device/out/host/linux-x86/bin
   ```

3. Enter the command:

   ```
   sudo adb devices
   ```

   A device should register.

4. Navigate to the following directory:

   ```
   cd <root>/LINUX/device/out/target/product/surf/obj/
   APPS/AppName_intermediates/
   ```

5. Copy the files:

   ```
   cp package.apk AppName.apk
   ```

6. Push the files as follows:

   `adb push AppName.apk /system/app/.`

NOTE: In general, the syntax is: adb push <file_name> <location_on_the_target>

4.3.4 Programming eMMC boot loaders with T32

How to program boot loaders with T32

1. Start T32 by using the **t32start.cmd** in the `<meta build>\common\t32` folder.
2. Navigate to **Configuration Tree → MSM8X26 → DAP → Podbus Device Chain → Power Trace Ethernet**.

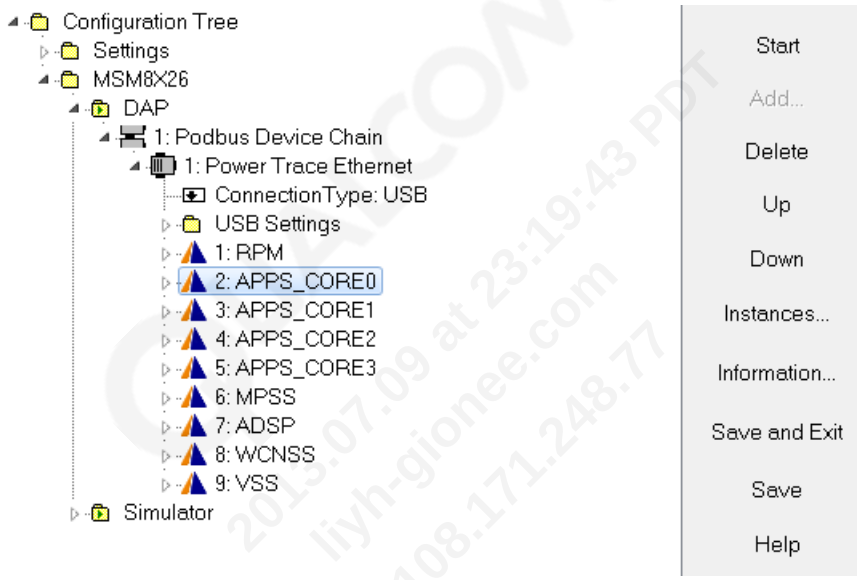

Figure 4-8 T32 configuration

3. Select RPM and click **Start**.
4. Select APPS_CORE0 and click **Start**.
5. In RPM T32 window, select **RPM commands → Build Options**, then select **asic** flavor in "Product Flavor" field.
6. Click **Load** in the Build Options window.

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

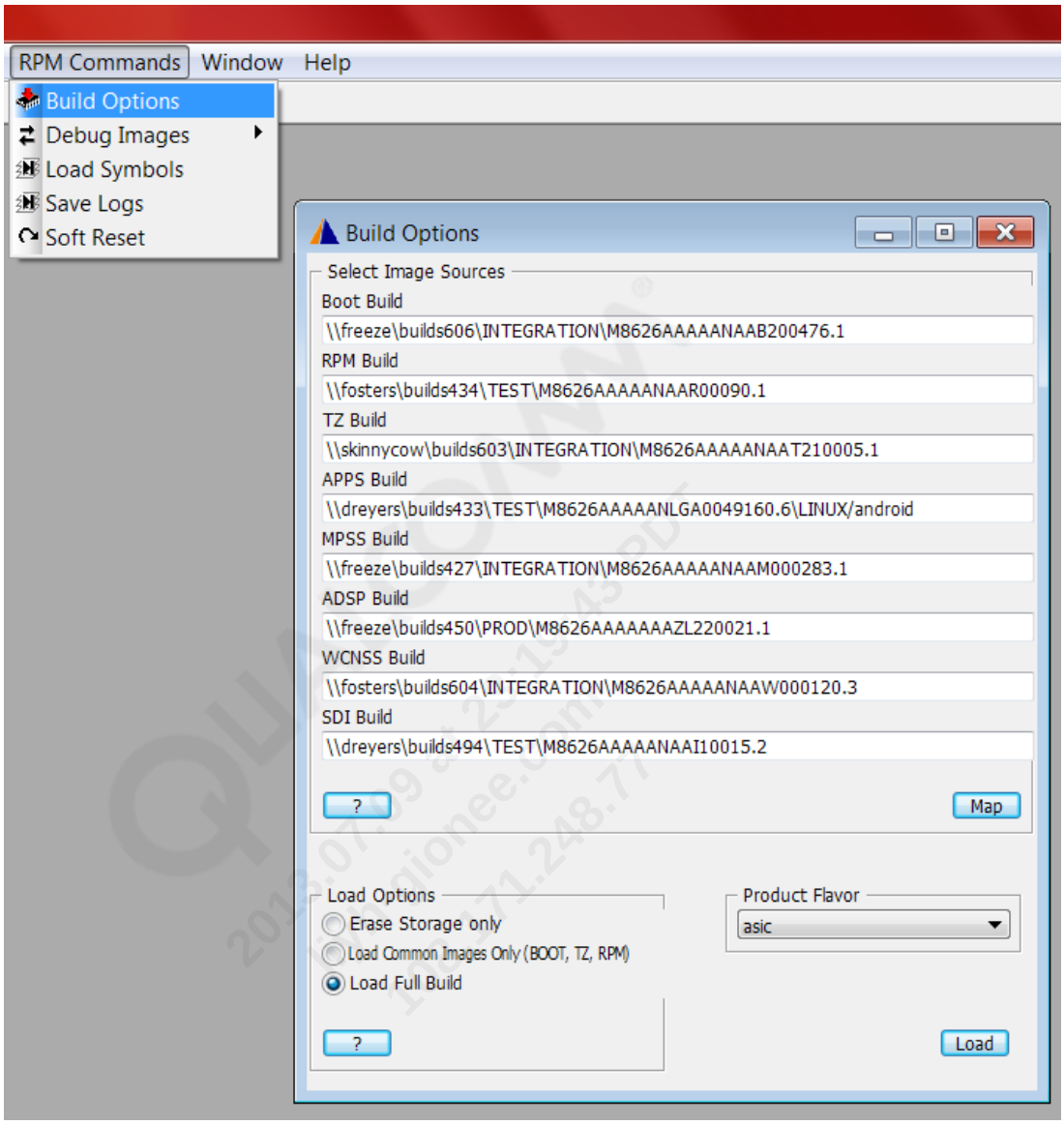

Figure 4-9 T32 load images

7. Keep your USB connecting to PC, and it will use fastboot to flash **NON-HLOS.bin** and apps binaries. Fastboot download process will start automatically by script after T32 finishes programming bootloaders.

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

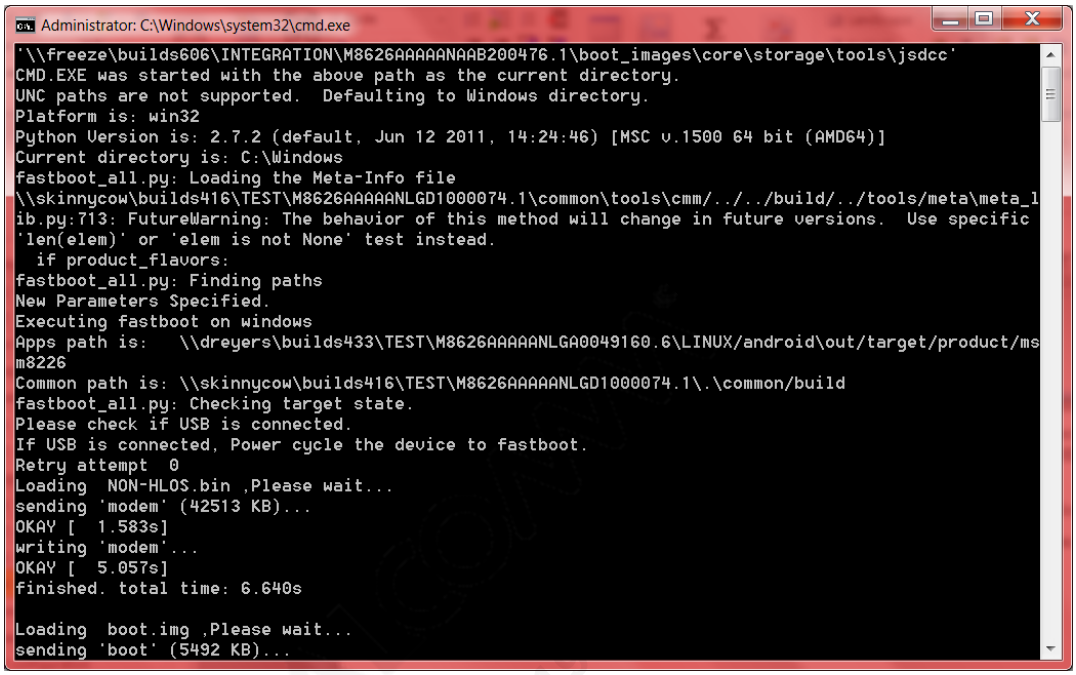

Figure 4-10 Fastboot flash NON-HLOS and apps images

8. After Fastboot completes flashing the binaries, power cycle the device.

5 Operational Guide

5.1 Initial bringup

For initial bringup, refer to the following sections to locate the SIM slots, primary antenna, and USB port.

5.1.1 MSM8626 Core Development Platform (CDP) UIM configuration

Refer to Figure 5-1~ Figure 5-3 for CDP SIM slots configuration:

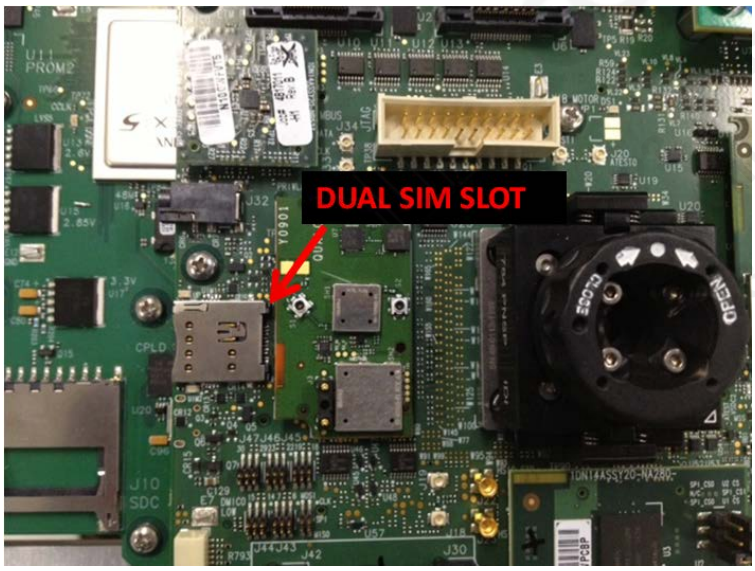

Figure 5-1 DUAL SIM slots on the baseband card

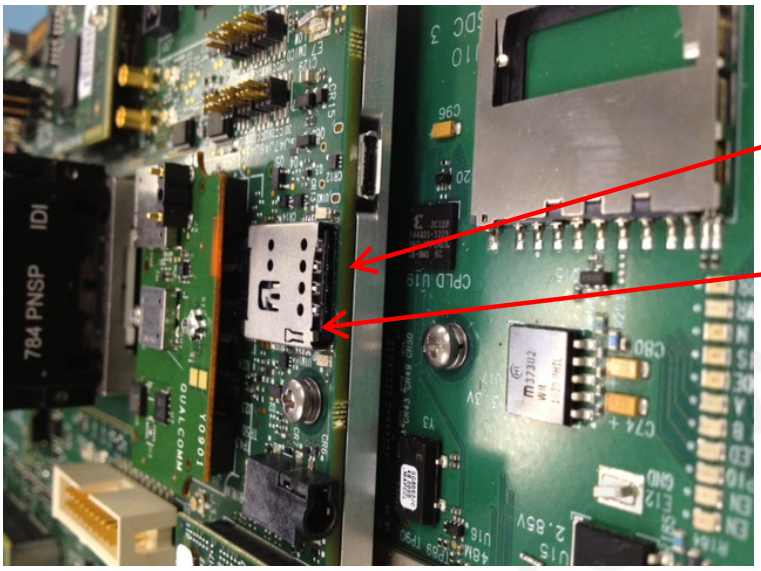

Figure 5-2 DUAL SIM slots (UIM1/UIM2) on the baseband card

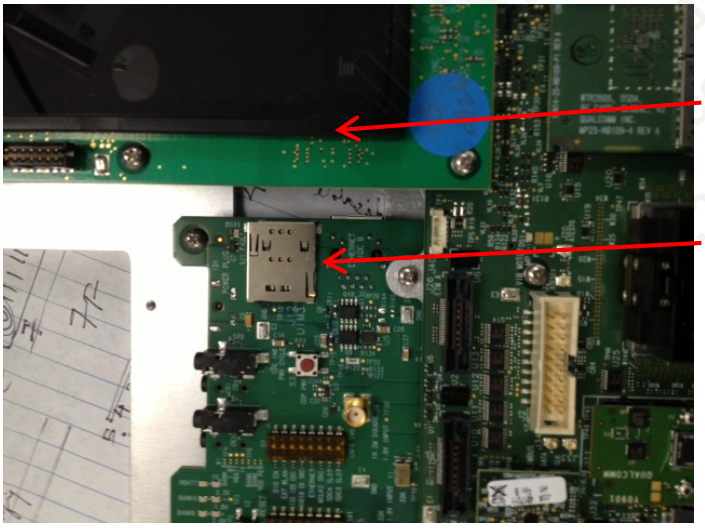

Figure 5-3 UIM3 on the baseband card

5.1.2 MSM8626 Modem Test Platform (MTP) UIM configuration

Refer to Figure 5-4~ Figure 5-6 for MTP SIM slots configuration:

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

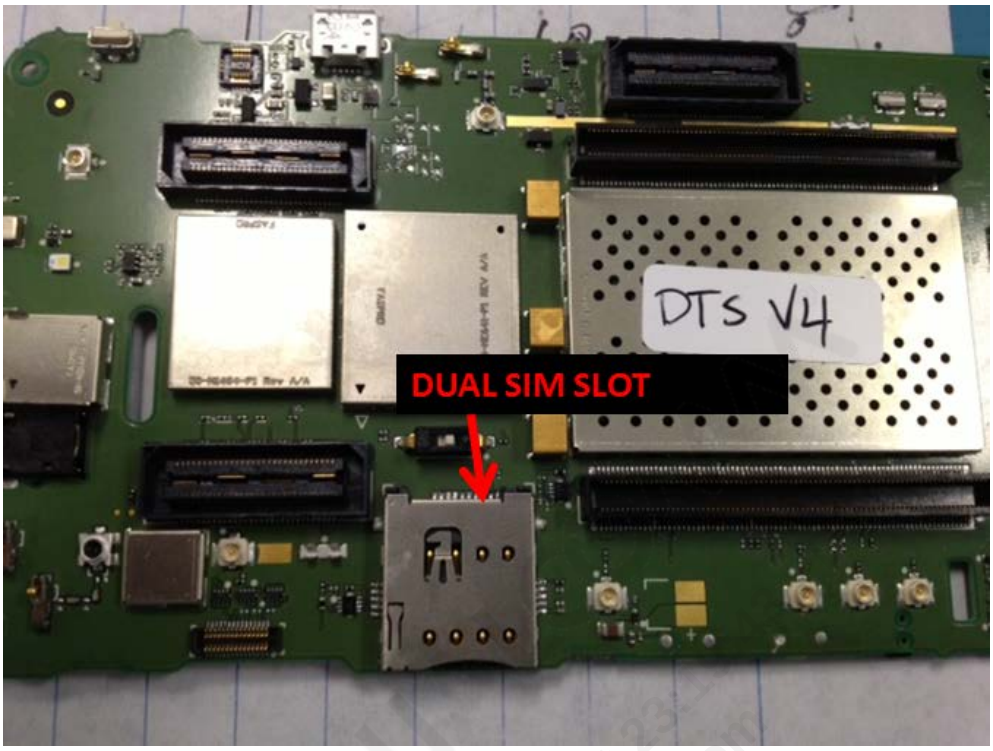

Figure 5-4 DUAL SIM slots on the baseband card

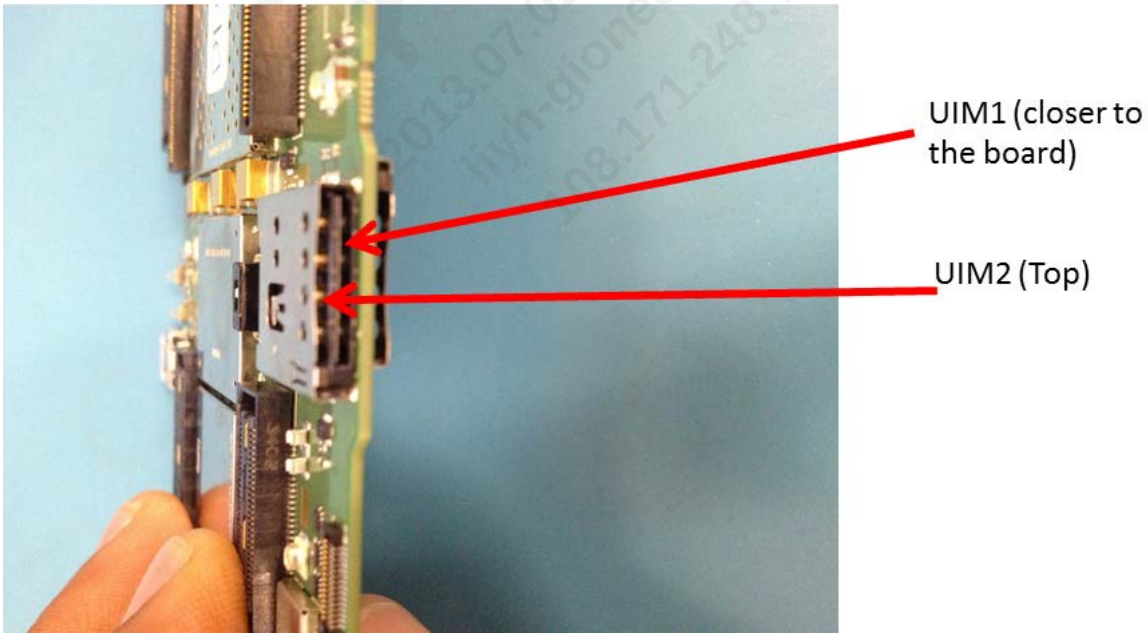

Figure 5-5 DUAL SIM slots (UIM1/UIM2) on the baseband card

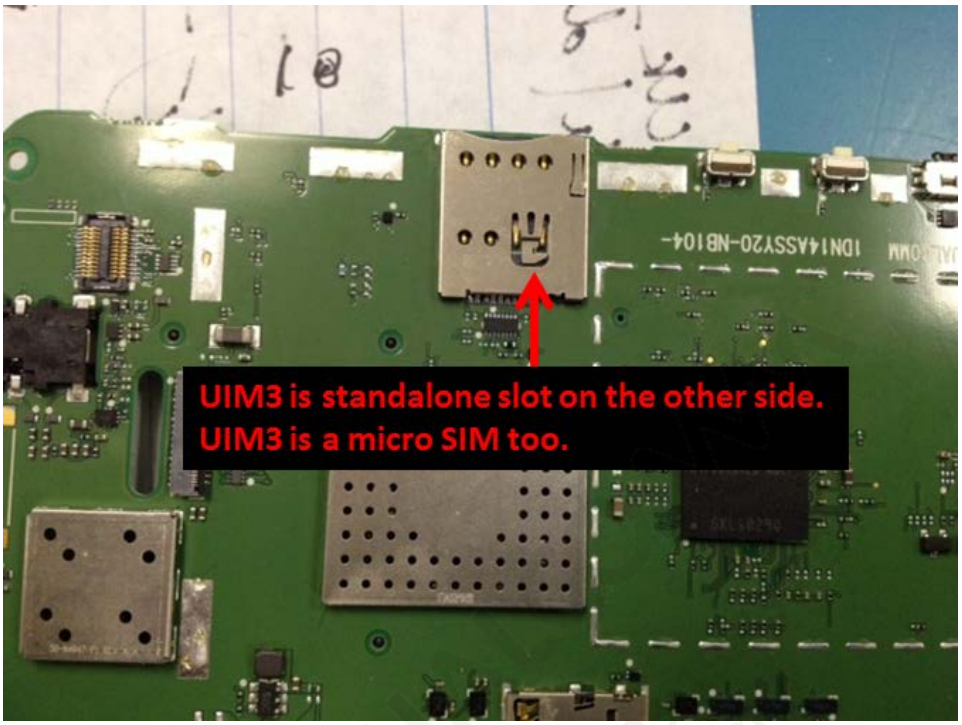

Figure 5-6 UIM3 on the baseband card

5.1.3 Antenna configuration

Refer to Figure 5-9 for CDP Antenna configuration

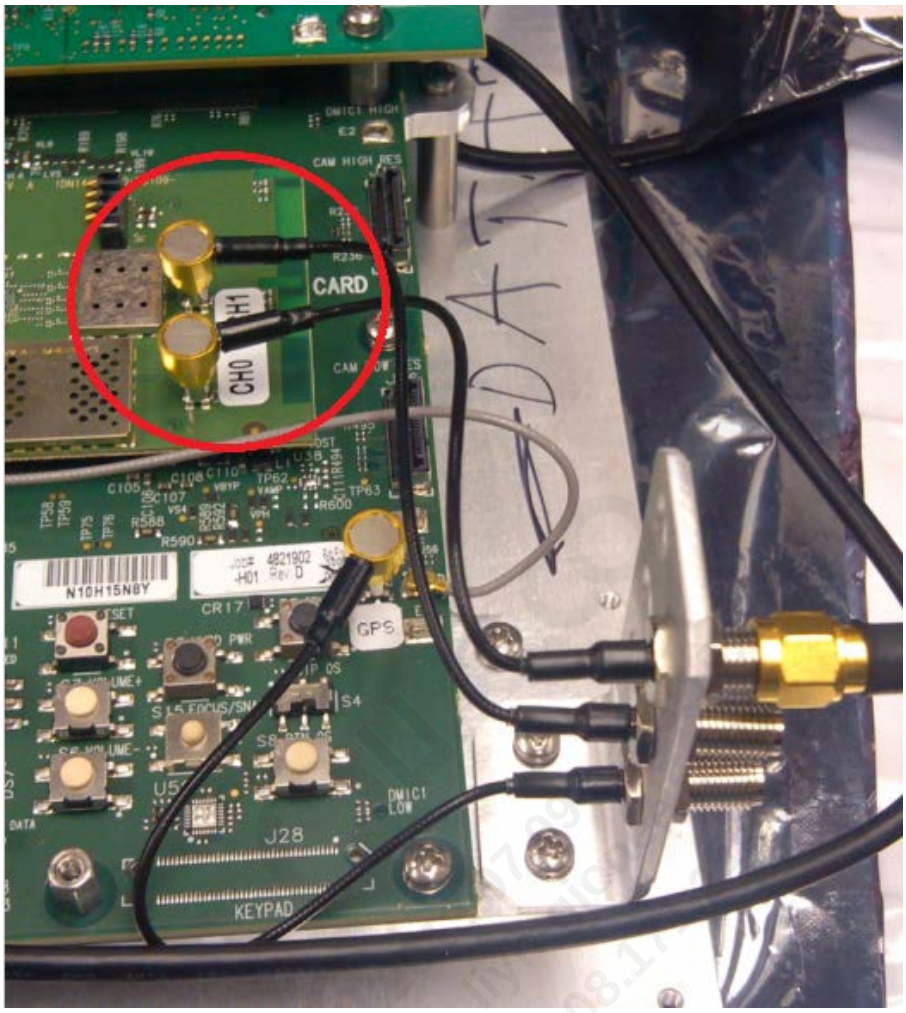

Figure 5-7 CDP antenna configuration

5.1.4 USB/JTAG configuration

Refer to Figure 5-8 for CDP USB/JTAG configuration:

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

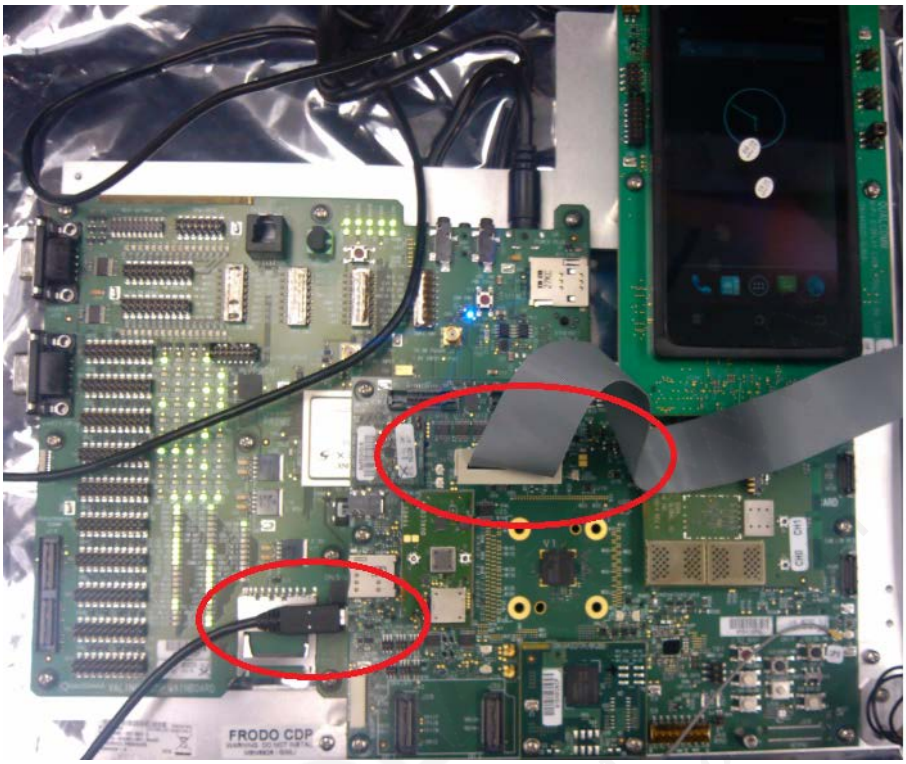

Figure 5-8 CDP USB/JTAG Configuration

5.2 DSDS/DSDA NV settings

5.2.1 NV configuration for WCDMA/GSM + GSM

Refer to Table 5-1 for WCDMA/GSM + GSM DSDS NV configuration

Table 5-1 WCDMA/GSM + GSM DSDS NV Settings

NV item	Subscription1 (WCDMA/GSM) value	Subscription2 (GSM) value
00010	17: Auto (WCDMA or GSM) 14: WCDMA only	13: GSM only
00441	0xFFFF (0x380 or 0x387 for specific bands)	Same as SUB1
00850	0x02 CS and PS	0x00 CS only
00855	0 RTRE configuration for WCDMA/GSM + GSM (Before setting this item send spc 000000 through QXDM command window)	Same as SUB1
00880	0x01 integrity enable	Same as SUB1
00881	0x01 ciphering enable	Same as SUB1
00882	0 fake security enable	Same as SUB1

00905	0x0000 fatal error option	Same as SUB1
00946	0x0040 IMT band(0CE0 US PCS Band)	0x0040
03649(RRC Version)	Inactive or 0→R99; 1→R5; 2→R6; 3→R7; 4→R8	Same as SUB1
03851	0 RxD control	Same as SUB1
04118 (HSDPA Cat)	Inactive or 24→DC	Same as SUB1
04210 (HSUPA Cat)	Inactive or 6→EUL 2ms; 5→EUL 10 ms	Same as SUB1
04398	0→DSDS; 1→SS	Same as SUB1
04399	1 detect hardware reset	Same as SUB1
06876	00005→WCDMA/GSM to GSM Tune away 00006→DSDS	Same as SUB1
06907	1 Dual SIM hardware 0 Single SIM hardware	Same as SUB1

For WCDMA/GSM + GSM DSDA NV configuration, the same as Table 5-1, except NV70266 (Dual standby preference) should be set to 2.

5.2.2 NV configuration for CDMA + GSM

Refer to Table 5-2 for CDMA + GSM DSDS NV configuration

Table 5-2 CDMA + GSM DSDS NV Settings

NV item	Subscription1 (CDMA+ HDR) value	Subscription2 (GSM) value
10 (Mode Preference)	4-> : Automatic mode 19: CDMA and HDR only	13: GSM only
475 (HDR SCP Session Status)	0-> Inactive	Same as SUB1
850 (Service Domine Preference)	0x02 CS and PS	0x00 CS only
905 (Fatal Error Option)	0x0000 fatal error option	Same as SUB1
4204 (HDR SCP Force	0-> HDR SCP Force Release 0 Session Configuration	Same as SUB1
4964 (HDR SCP Force At Configuration)	0-> HDR rev 0 , 1->HDR rev A, 3->HDR rev B.	Same as SUB1

03446 (TRM Configuration)	2, 0	Same as SUB1
4398 (UIM Select Default USIM Application	0→DSDS; 1-> SS	Same as SUB1
4399 (detect hardware reset)	1 detect hardware reset	Same as SUB1
6874 (ASID 1 Data)	255	Same as SUB1
6875 (ASID 2 Data)	255	Same as SUB1
562 (preference Hybrid Mode)	1 -> Hybrid operation allowed	0 -> Hybrid operation not allowed
6876(Dual standby config Items)	00002 (Dual standby preference)	Same as SUB1
6907 (NV_UIM_HW_SIM_CONFIG)	1-> Dual SIM 0-> Single SIM	Same as SUB1
855 (RTRE configuration)	0 (Before setting this item send spc 000000 through QXDM command window)	Same as SUB1

For CDMA + GSM DSDA NV configuration, the same as Table 5-2, except NV70266 (Dual standby preference) should be set to 2.

5.3 Call configuration

This section details configuration information for making calls.

5.3.1 1X voice call

Prerequisites

- NV setting (set NV # 10 to 4 for automatic mode)
- QCN (be sure to perform RF calibration on the device; do not use a golden QCN…, also note that 1X will be on the SV chain by default)
- PRL (must match the band/channel being tested, and SID & NID must also match)

Callbox setup

- Current testing has been done on Agilent 8960
- Callbox setup instructions
  - In System Config/Application Setup, select CDMA 2000 Lab App B (version should be B or above);
  - In Call Setup/Call Control screen
    - Set the Operating Mode to Active Cell;

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

- Set the System Type to IS-2000;
- Click "**More**" to move to page 2 of 5, select Cell Info/Cell Parameters. Configure the SID/NID(match with PRL, or set it as wildcard: SID=0, NID=65535);

□ In Call Setup/Call Params screen

- Set Cell 1 Power to a proper value (between -45 ~ -65dBm);
- Set Cell Band and Channel, match the PRL setting;
- Set Protocol Rev to 6(IS-2000-0);
- Set Radio Config(RC) to (3,3) - (Fwd3, Rvs3);
- Set FCH Service Option Setup for (Fwd3, Rvs3) to SO3 (Voice);

Device setup

■ Step-by-step instructions to make the call

a. Use the Call Manager screen in QXDM Professional (QXDM Pro).

b. Set the phone number to something like 1234 and make sure the service option matches the callbox setting.

c. Click **Call** to start a call. For MT, originate the call from the test box.

NOTE: For Dual SIM device, the default subscription is 0. If subscription 1 is needed, check the **Dual SIM**, then select the **Subscription ID**, refer to Figure 5-5.

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

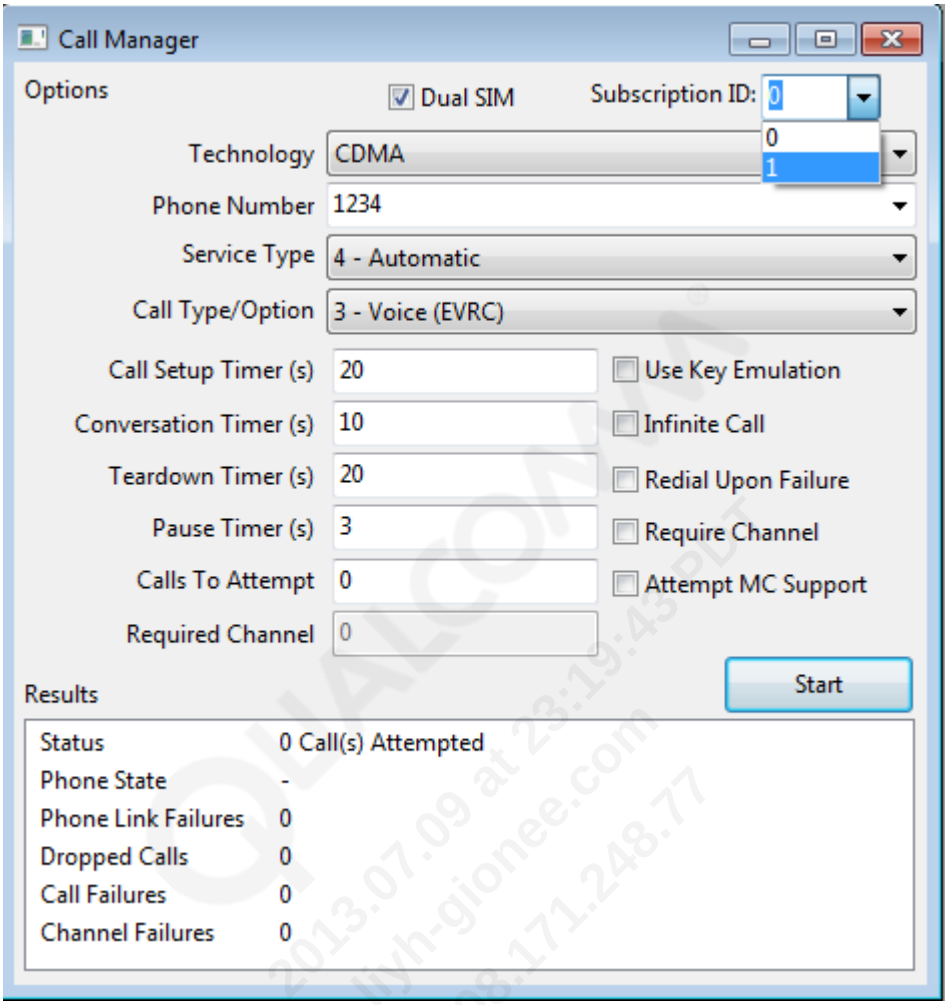

Figure 5-9 MSM8626 call manager

5.3.2 1X data call

Prerequisites

- NV setting (set NV #10 to 4 for Automatic mode)
- QCN (be sure to perform RF calibration on the device; do not use a "golden" QCN…; also note that 1X will be on the SV chain by default)
- PRL (must match the band/channel being tested, and SID & NID must also match)

Callbox setup

- Current testing has been done on Agilent 8960 or Anritsu MT8820;
- In Call Setup/Call Params screen
  - Set Radio Config(RC) to (3,3) - (Fwd3, Rvs3);
  - Set FCH Service Option Setup for (Fwd3, Rvs3) to SO32 (TDSO);

- Callbox setup instructions – Make sure band/channel and SID/NID match those specified in PRL; cell power should be set somewhere between -45 and -65 dB

Device setup

- Step-by-step instructions to make the call
  a. Use the Call Manager screen in QDXM Pro.
  b. Set the phone number to something like 1234 and make sure the service option matches the callbox setting.
  c. Click **Call** to start a call; for MT, originate the call from the test box.

5.3.3 HDR call

NV settings

- NV setting (set NV # 10 to 4 for Automatic mode)
- For DO Rev A calls, NV #4964 should be set to Rev A mode and the callbox has to be put in Rev A mode. For example, Set NV #4964 = NV_HDRSCP_REVA_PROTOCOLS_WITH_MFPA.

RF calibration

- Ensure that the device has been RF-calibrated
- Roaming List – A preferred roaming list with HDR channels needs to be loaded and subnet ID in PRL must match the callbox setting
- Roaming list is loaded via QPST Service Programming; do the following:
  a. Select the Roam tab
  b. Enter prl path in the Preferred Roaming area
  c. Select Write to Phone

Callbox setup

- For DO RevA calls, the callbox must be put in Rev A mode

Device setup

- Step-by-step instructions to make the call
  a. Power cycle the device
  b. Enter "**mode online**" in the QXDM Pro "Command Bar"; the device should attempt to acquire HDR channel and negotiate a session

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

5.3.4 GSM voice call

Prerequisites

- NV settings
  a. Set NV #10(Mode Preference) to 13 for GSM only operation
  b. Set NV #441(Band Class Preference) to 0x200 for GSM900 band
  c. For Multimode build, the Mode preference must be changed via the UI. Otherwise Multimode will use the default setting in Android (defaults to 1X only). And it will use this to overwrite the NV item, so the modem does not go into GSM.
- QCN (be sure to perform RF calibration on the device; do not use a golden QCN)

Callbox setup

- Current testing has been done on Agilent 8960 or Anritsu MT8820
- Set band to GSM900; cell power should be set somewhere between -45 and -65 dBm
- Set Channel mode to TCH/F (full rate TCH)

Device setup

- Step-by-step instructions to make the call
  a. Use the Call Manager screen in QDXM Pro.
  b. Set the phone number to something like 1234 and make sure the service option matches the callbox setting.
  c. Click **Call** to start a call; for MT, originate the call from the test box.

5.3.5 GPRS data call

RF calibration

- Ensure device has been RF-calibrated

Callbox setup (Agilent 8960)

- Select call setup screen
- Callbox setup
  - BCCH parameters→cell power = -75 dBm
  - BCCH parameters→cell band = EGSM
  - BCCH parameters→Broadcast chan = 20
  - PDTCH parameters→Multislot config = 1 Down 1 Up
  - Operating mode = Active mode GPRS
  - Data Conn = Type ETSI Type A

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

Device setup

- Step-by-step instructions to make the call
  a. Insert a SIM (test SIM is good enough)
  b. Power up device; Enter "**mode online**" in QXDM Pro "Command Bar" if required
  c. UE should camp on GPRS cell and ATTACH
  d. Initiate test mode A data call; hit Start Data Connection; bottom of screen should display TRANSFERRING

5.3.6 WCDMA voice call

Prerequisites

- QCN – Ensure device has been calibrated

Callbox setup

- Agilent 8960
  - Cal box setup
    - Call Control/Security Info/Security Parameters/Security Operations – None
    - Call Parms/Cell Power – -50.00 dBm
    - Call Parms/Channel Type – 12.2 KRMC
    - Call Parms/Paging Service – AMR Voice
  - Test results – MO and MT passed 5/5
- Anritsu 8480C
  - Callbox setup
    - Station Globals – Spec_Release – 3; HSDPA – FALSE; EUL – FALSE
- Anritsu 8820
  - Callbox setup
    - Call Processing – On
    - Test Loop Mode – Off
    - Signal/Channel Coding – Voice

Device setup

- Step-by-step instructions to make the call (MO)
  a. After starting firmware and software, attach USB and then do a "mode online" from QXDM Pro "Command Bar"
  b. Allow the mobile to acquire and register to network
  c. Start a call using the Call Manager dialog in QXDM Pro
  d. Set Technology to WCDMA

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

e. Set phone number to 1234
f. Check the “infinite call” box
g. Start the call
h. Phone state will show “Originating call” and then “Conversation”

5.3.7 WCDMA data call

Prerequisites

- QCN – Ensure device has been calibrated

Callbox

- Agilent 8960
  - Callbox setup
    - Call Parms/Channel Type – HSPA
  - Test results – DUN data calls passed with ~384 kbps throughput; QMI not tried
- Anritsu 8480C
  - Callbox setup
    - Station Globals – Spec_Release – 5; HSDPA – TRUE; EUL – FALSE
  - Test results – DUN data calls was up but crashed later; QMI not tried
- Anritsu 8820
  - Callbox setup
    - Call processing – On
    - Test Loop Mode – Mode 1
    - Signal/Chanel Coding – Fixed Reference Channel

Device setup

- Step-by-step instructions to make the call
  a. Allow UE register to network
  b. Click “**Connect/Dial**” on USB (DUN) or QMICM to initiate data call
  c. Run iPerf or FTP applications to test throughput (we tested with FTP)

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

5.4 GPS configuration

Prerequisites

- Obtaining GNSS SubSysGNSS DLL Ver 1.0.44 or higher is required to perform offline RF dev

Operation procedures

- Running Offline RF Dev requires QPSR, see [Q13] for complete details

5.5 Multimedia configuration

This section will be added in future version.

QUALCOMM 2013.07.09 at 23:19:43 PDT liyh-gionee.com 108.171.248.77

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

A Android Build Reference

A.1 Android device tree structure

The Android device tree structure, for example, the <Android device tree root>, is laid out as follows:

- build/ – Build environment setup and makefiles
- bionic/ – Android C library
- dalvik/ – Android JVM
- kernel/ – Linux kernel
- framework/ – Android platform layer (system libraries and Java components)
- system/ – Android system (utilities and libraries, fastboot, logcat, liblog)
- external/ – Non-Android-specific Open Source projects required for Android
- prebuilt/ – Precompiled binaries for building Android, e.g., cross-compilers
- packages/ – Standard Android Java applications and components
- development/ – Android reference applications and tools for developers
- hardware/ – Hardware abstraction layer (audio, sensors) and Qualcomm specific hardware wrappers
- vendor/qcom/ – Qualcomm target definitions, e.g., msm7201a_surf
- vendor/qcom-proprietary – Qualcomm-proprietary components, e.g. MM, QCRIL, etc.
- out/ – Built files created by user
  - out/host/ – Host executables created by the Android build
  - out/target/product/<product> – Target files
    - appsboot*.mbn – Applications boot loader
    - boot.img – Android boot image (Linux kernel + root FS)
    - system.img – Android components (/system)
    - userdata.img – Android development applications and database
    - root/ – Root FS directory, which compiles into ramdisk.img and merged into boot.img
    - system/ – System FS directory, which compiles into system.img

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

- obj/ – Intermediate object files
  - include/ – Compiled include files from components
  - lib/
  - STATIC_LIBRARIES/
  - SHARED_LIBRARIES/
  - EXECUTABLES/
  - APPS/
- symbols/ – Symbols for all target binaries

A.2 Android target tree structure

The Android target tree structure is laid out as follows:

- / – Root directory (ramdisk.img, read-only)
  - init.rc – Initialization config files (device config, service startups) init.qcom.rc
  - dev/ – Device nodes
  - proc/ – Process information
  - sys/ – System/kernel configuration
  - sbin/ – System startup binaries (ADB daemon; read-only)
  - system/ – From system.img (read-write)
    - bin/ – Android system binaries
    - lib/ – Android system libraries
    - xbin/ – Nonessential binaries
    - framework/ – Android framework components (Java)
    - app/ – Android applications (Java)
    - etc/ – Android configuration files
  - sdcard/ – Mount point for SD card
  - data/ – From userdata.img (read-write)
    - app/ – User installed Android applications
    - tombstones/ – Android crash logs

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

A.3 Building Tiny Android

Tiny Android, or TINY_ANDROID, is a build variant that creates only a superminimal build configuration used for board bringup and very low-level debugging. The TINY_ANDROID configuration consists of only the Android Linux kernel and a system root file system containing a minimal set of system utilities.

To build Tiny Android, use:

```
$ make BUILD_TINY_ANDROID=true -j4
```

A.4 Building Linux kernel manually

How to build the Linux kernel manually:

1. Set up the Android build environment (envsetup.sh/choosecombo).
2. Change to the kernel directory (kernel/).
3. Set up the correct kernel config with the command:

   ```
   make ARCH=arm CROSS_COMPILE=arm-eabi- msm8974_defconfig
   ```

4. Build the kernel image with the command:

   ```
   make -j3 ARCH=arm CROSS_COMPILE=arm-eabi- zImage
   ```

5. If desired, build the optional kernel modules with the command:

   ```
   make -j3 ARCH=arm CROSS_COMPILE=arm-eabi- modules
   ```

   The resulting kernel image will appear in kernel/arch/arm/boot/zImage.

NOTE: In theory, it is possible to use –jn as long as ‘n’ is smaller than the number of processors in the server where the build is being made.

6. To start with a clean tree, use the following commands:
   a. To remove object files:

      ```
      make clean
      ```

   b. To remove all generated files:

      ```
      make distclean
      ```


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

A.5 Build Android manually

To build Android manually:

1. Set up the Android build environment (envsetup.sh/choosecombo).
2. Change to the main Android directory.
3. Build with the command:

```
make -j4
```

4. To build individual components, choose one of the following options:
   - To run make from the top of the tree, use the command:

```
m <component name>  # E.g. m libril-qc-1
```

   - To build all of the modules in the current directory, change to the component directory and use the command:

```
mm
```

5. To delete individual component object files, choose one of the following options:
   - To delete a particular module, use the command:

```
m clean-<module name>
```

   - To delete a module within a given path, use the commands:

```
rm -rf out/target/product/*/obj/STATIC_LIBRARIES/
<module name>_intermediates

rm -rf out/target/product/*/obj/SHARED_LIBRARIES/
<module name>_intermediates

rm -rf out/target/product/*/obj/EXECUTABLES/
<module name>_intermediates
```


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

A.6 Other important Android build commands

Other important Android build commands are:

- printconfig – Prints the current configuration as set by the choosecombo commands.
- m – Runs make from the top of the tree. This is useful because the user can run make from within subdirectories. If you have the TOP environment variable set, that is what this command will use. If you do not have the TOP variable set, it looks up the tree from the current directory, trying to find the top of the tree.
- - mm – Builds all of the modules in the current directory.
- - mmm – Builds all of the modules in the supplied directories.
- croot – cd to the top of the tree.
- sgrep – grep for the regex you provide in all .c, .cpp, .h, .java, and .xml files below the current directory.
- clean-$(LOCAL_MODULE) and clean-$(LOCAL_PACKAGE_NAME)
  - Let you selectively clean one target. For example, you can type make clean-libutils, and it will delete libutils.so and all of the intermediate files, or you can type make clean-Home and it will clean just the Home application.
- make clean – Makes clean deletes of all of the output and intermediate files for this configuration. This is the same as rm -rf out/<configuration>/.

Android makefiles (Android.mk) have the following properties:

- Similar to regular GNU makefiles; some differences are:
  - Predefined variables to assign for source files, include paths, compiler flags, library includes, etc.
  - Predefined action for compiling executables, shared libraries, static libraries, Android packages, using precompiled binaries, etc.
- Variables
  - LOCAL_SRC_FILES – List of all source files to include
  - LOCAL_MODULE – Module name (used for “m”)
  - LOCAL_CFLAGS – C compiler flags override
  - LOCAL_SHARED_LIBRARIES – Shared libraries to include
- Action
  - include $(CLEAR_VARS) – Clears LOCAL* variables for the following sections:
    - include $(BUILD_EXECUTABLE)
    - include $(BUILD_SHARED_LIBRARIES)
    - include $(BUILD_STATIC_LIBRARIES)

NOTE: Paths in Android.mk are always relative to the Android device tree root directory.

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

To add a new module to the Android source tree, perform the following steps:

1. Create a directory to contain the new module source files and Android.mk file.
2. In the Android.mk file, define the LOCAL_MODULE variable with the name of the new module name to be generated from your Android.mk.

NOTE: For Applications modules, use LOCAL_PACKAGE_NAME instead.

Local path in your new module will be LOCAL_PATH. This is the directory your Android.mk file is in. You can set it by inserting the following as the first line in your Android.mk file:

```
LOCAL_PATH := $(call my-dir).
LOCAL_SRC_FILES
```

The build system looks at LOCAL_SRC_FILES to find out which source files to compile, .cpp, .c, .y, .l, and/or .java. For .lex and .yacc files, the intermediate .h and .c/.cpp files are generated automatically. If the files are in a subdirectory of the one containing the Android.mk file, it is necessary to prefix them with the directory name:

```
LOCAL_SRC_FILES := \
   file1.cpp \
   dir/file2.cpp
```

The new module can be configured with the following:

- LOCAL_STATIC_LIBRARIES – These are the static libraries that you want to include in your module.

  ```
  LOCAL_STATIC_LIBRARIES := \
     libutils \
     libtinyxml
  ```

- LOCAL_MODULE_PATH – Instructs the build system to put the module somewhere other than what is normal for its type. If you override this, make sure that you also set LOCAL_UNSTRIPPED_PATH if it is an executable or a shared library so the unstripped binary also has somewhere to go; otherwise, an error will occur.

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

B SCons

SCons Ver 2.0.0 or higher is a software construction tool required for building all non-HLOS source code releases. More information on SCons can be obtained at:

- SCons – http://www.scons.org
  - User manual – http://www.scons.org/doc/HTML/scons-user/book1.html
  - Online manual pages – http://www.scons.org/doc/HTML/scons-man.html
  - SCons overview – http://www.humanized.com/presentations/sconsChapter Name

QUALCOMM 2013.07.09 at 23:19:43 liyh-gionee.com 108.171.248.77

1 简介

1.1 目的

本文档介绍了如何获取、编译、下载软件,以及适用于 MSM8626 LA 参考平台的通用操作。

1.2 范围

本文档说明了如何设置开发环境、获取软件、并将软件安装到开发环境中。同时涵盖了软件操作,例如：重建现有软件并将重建产品(固件)编译到参考平台；也说明了如何编制和重编系统的固件设备。

本文档描述了固件设备以及删除固件程序、加载软件至固件设备、验证加载流程所需的设备、软件和步骤。也描述了通用操作,例如通话、gps 和多媒体的配置。

1.3 约定

函数声明，函数名称，类型声明以及代码举例需要使用不同的字体，例如`#include`。

代码变量应出现在角括号中，例如`<number>`。

命令在输入时应使用不同的字体，例如 **`copy a:*.* b:`**。

按钮和关键字应使用粗体字体，例如 click **Save** or press **Enter**。

如果你正在使用彩色显示器查看此文件，或使用彩色打印机打印文档时，红色字体显示的数据类型，蓝色字体显示属性，而绿色字体显示系统属性。

参数类型是用箭头表示：

→ 指定一个输入参数

← 指定一个输出参数

↔ 指定一个用于输入和输出参数

1.4 参考

参考文件，其中可能包括高通的文档、标准和资源，具体列在表 Table 1-1。对于不再适用参考的文档已从本表中删除，因此，参考文档的引用顺序可能不连续。

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

Table 1-1 参考文档和标准

编号	文档	
高通文档		
Q1	*Application Note: Software Glossary for Customers*	CL93-V3077-1
Q2	*Hexagon™ Tools Installation Guide*	80-VB419-25
Q3	*Hexagon™ Development Tools Overview*	80-VB419-74
Q4	*Presentation: SCONS Build System Overview*	80-N3601-1
Q5	*SCons Build System Migration Guide*	80-N3605-1
Q6	*Application Note: SCons External Dependencies*	80-N2077-2
Q7	*Application Note: Make vs SCons Differences*	80-N2077-3
Q8	*Application Note: Modifying Default SCons Environment*	80-N2077-4
Q9	*Application Note: SCons Modifications*	80-N2077-5
Q10	*Application Note: SCons Warning Messages about pywin32 Extensions*	80-N2077-6
Q11	*USB Host Driver for Windows 2000/Windows XP User Guide*	80-V4609-1
Q12	*USB Host Driver Installation Instructions for Microsoft Windows*	80-VP092-1
Q13	*gpsOne® Gen 8 Enginer RF Development and Mobile Station Time Calibration Test Procedures*	80-VM522-2
Q14	*Introduction to Qualcomm ChipCenter and Qualcomm ChipCode*	80-NC193-1
Q15	*Qualcomm ChipCenter and Qualcomm ChipCode User Guide*	80-NC193-2
资源		
R1	*Android™ Open Source Project Page*	http://source.android.com/
R2	*Android™ Developer Resources*	http://developer.android.com/index.html
R3	*Android™ Source Download and System Setup*	http://source.android.com/source/index.html
R4	*Code Aurora Forum*	https://www.codeaurora.org/
R5	*Installing Repo*	http://source.android.com/source/downloading.html
R6	*Qualcomm ChipCode Website*	https://chipcode.qti.qualcomm.com/
R7	*Qualcomm ChipCode Help Wiki*	https://chipcode.qti.qualcomm.com/projects/help/wiki

1.5 技术支持

如果您想得到关于本指南的支持细节，请登陆至 https://support.cdmatech.com/。

如果您无法登陆 CDMA 技术支持服务网站，请注册或发送电子邮件至 support.cdmatech@qualcomm.com。

1.6 缩写词

术语和缩略词的定义请参考文档[Q1]。

2 软件环境

2.1 设备和软件

Table 2-1 列出了安装及运行软件所需的硬件、软件和其他设备。

Table 2-1 所需硬件、软件和其他设备

	描述	版本	来源/供应商	用途
1	可以运行 64 位 Ubuntu 操作系统的 Linux 开发工作站。 推荐能加快编译时间的 PC。谷歌安卓论坛推荐如下： ▪ 推荐最低 16GB 的 RAM ▪ 高于或等同于 Quad Core CPU（例如：Intel i7-2600 @3.4 Ghz） ▪ 1TB 硬盘驱动器（首选 SSD 加速器）	—	—	构建机器
2	Windows 7 或 Windows XP 工作站	Windows XP 或 Windows 7	微软	替代 Non-HLOS 构建机器及基于 windows 的编程工具
3	基于 64 位架构的 Ubuntu 10.04 LTS Linux 发行版	10.04 LTS	Ubuntu 社区 /Canonical	Android 构建主机的操作系统
4	适用于 Linux x64 的 Java SE JDK	6	Oracle	构建 Android 系统
5	repo	—	Android 开源项目	Android 源代码管理工具
6	ARM® 工具链	ARM 编译工具 5.01 update 3 (build 94)	ARM Ltd.	构建引导映像、RPM、TrustZone 和 SDI 的工具链
7	Hexagon™工具链	5.0.07 和 5.0.09	高通/Gnu	构建 Modem Processor Subsystem (MPSS) 和 Applications Digital Signal Processor Subsystem (ADSP) 的工具链 用于 MPSS 的 5.0.07 版本以及用于 ADSP 的 5.0.09 版本

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

	描述	版本	来源/供应商	用途
8	Python	2.6.2	Python.org	构建引导映像、RPM、ADSP 和 MPSS 的工具链

2.2 安装和设置

2.2.1 安装 Ubuntu

以下介绍了如何安装、更新和配置 Ubuntu 10.04 (64 位)系统，也可参考 Android source get started website [R3]。安装时，用户必须作为 root 用户登陆，或使用 sudo 得到 root 许可。

安装步骤如下：

1. 创建一张安装盘，并按照 http://releases.ubuntu.com 的说明进行安装。
2. 安装后，按以下方法中的一种更新软件。
   - 使用 GUI, 选择 System→Administration→Update Manager。
   - 使用 shell 命令行。
     i. 直接编辑如下源配置文件。

        sudo vi /etc/apt/sources.list

     ii. 编辑文件以使用 universe 及 multiverse 源，并关闭 Ubuntu 安装盘源。

     iii. 使用以下命令更新包列表并升级软件包。

        sudo apt-get update

        sudo apt-get upgrade
3. 使用 `apt-get` 安装所需的其他软件包。

```
$ sudo apt-get install git-core gnupg flex bison gperf build-essential
zip curl zlib1g-dev libc6-dev lib32ncurses5-dev ia32-libs x11proto-core-
dev libx11-dev lib32readline5-dev lib32z-dev libgl1-mesa-dev g++-
multilib mingw32 tofrodos python-markdown libxml2-utils xsltproc
```

4. 按以下方法中的一种将 bash 设为默认 shell (Android 构建脚本包含了 bash shell 的依赖，需要系统默认 shell /bin/sh 调用 bash)。
   - 重新配置程序包：
     i. 使用如下命令：

        sudo dpkg-reconfigure dash

     ii. 回答 no。
   - 使用以下命令手动更改 symlink/bin/sh→dash 为/bin/sh→bash。

        sudo rm /bin/sh

        sudo ln -s /bin/bash /bin/sh

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

NOTE: 更多信息，请登陆 https://wiki.ubuntu.com/DashAsBinSh。

2.2.2 Windows共享的Samba配置 （可选）

步骤如下：

1. 通过以下命令安装 Windows 共享的 Samba 服务器及配置管理器。

```
sudo apt-get install samba system-config- samba
```

2. 通过以下命令配置 Samba 服务器。

```
System->Administration->Samba
          preferences->server settings:
          vmgroup, security=user authentication
          encrypt pw=yes, guest accnt=no guest accnt
   add share directory=/, share name=root, description=root directory
```

2.2.3 安装JDK

Ubuntu 主要的包存储库不再包含 Sun JDK。用户需要添加适当的存储库并告知系统应使用的 JDK，以下载 Sun JDK。

```
sudo add-apt-repository "deb http://archive.canonical.com/ lucid partner"
sudo apt-get update
sudo apt-get install sun-java6-jdk
```

2.2.4 安装Repo

Repo 工具是 Android 项目的源代码配置管理工具(参见[R5])，是用 Python 语言写的 git 的前端。Repo 工具通过清单文件下载代码，这些代码按照项目集的方式进行组织并存储在不同的 git 存储库中。

安装步骤如下：

1. 在 home 目录下创建~/bin 目录；或者，如果有 root 或 sudo 访问权限，将 repo 安装在所有系统用户都可以访问的共同路径下，例如`/usr/local/bin` 或者是`/opt` 下的某个位置。

2. 下载 repo 脚本。

```
$ curl https://dl-ssl.google.com/dl/googlesource/git-repo/repo
>~/bin/repo
```

3. 设置 repo 脚本属性为 executable。

```
$ chmod a+x ~/bin/repo
```

4. 确认 PATH 中包含了 repo 的安装目录位置。

```
$ export PATH=~/bin:$PATH
```

5. 试运行 **repo –help** 进行确认，应能看到如下的信息：

```
$ repo --help
usage: repo COMMAND [ARGS]
repo is not yet installed.  Use "repo init" to install it here.
The most commonly used repo commands are:
  init      Install repo in the current working directory
  help      Display detailed help on a command
```

6. 安装 repo(repo init)以访问完整的在线帮助。

2.2.5 安装ARM编译工具

构建 non-HLOS 映像需使用 Table 2-1 列出的特定版本的 ARM 编译工具。推荐使用 Linux 环境构建所有软件映像；不过，Windows 或 Linux 主机版本可用于构建 non-HLOS 映像。

更多关于 ARM 开发套件和工具链的信息,请登陆 ARM 支持网站：
http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.subset.swdev.coretools/index.html

2.2.5.1 在Linux主机上安装ARM编译工具

在 Linux 主机上安装 ARM 编译工具，步骤如下：

1. 根据 Table 2-1 列出的 ARM 供应商获取所需的 ARM 工具链。
2. 按照供应商的要求将工具链以及 flex 许可管理器安装到 Linux 构建系统。

2.2.5.2 在Windows主机上安装ARM编译工具

如使用 Windows 构建机器创建 non-HLOS builds,以下提供了更多在 Windows 主机上安装 ARM 编译器的细节描述。

根据 Table 2-1 列出的 ARM 供应商获取所需的 ARM 工具链。按照供应商的要求将工具链以及 flex 许可管理器安装到 Windows 构建系统。

在 Windows 主机上安装 ARM 编译工具，步骤如下：

1. 登陆 https://silver.arm.com/download/download.tm?pv=1245960 并访问软件。
2. 默认安装位置为 `C:\Program Files (x86)\ARM_Compiler5\`。
3. 如需要，把文件解压到安装工具的目录下。例如：高通的安装目录是 `C:\Program Files (x86)\ARM_Compiler5\bin`。
4. 打开 DOS 命令提示窗并查看编译器、链接器、汇编器和 fromelf 的版本，确认已安装更新的工具。
5. 运行 **armcc –vsn** 查看版本。返回以下命令。

```
ARM/Thumb C/C++ Compiler, 5.01 [Build 94]
For support contact support-sw@arm.com
Software supplied by: ARM Limited
```

```
armar –-vsn
armlink –-vsn
armasm –-vsn
fromelf --vsn
```

返回的所有版本应该是 build94。

2.2.6 安装Hexagon工具链

构建 non-HLOS 映像需使用 Table 2-1 列出的特定版本的 Hexagon 工具链。推荐使用 Linux 环境构建所有软件映像；不过，Windows 或 Linux 主机版本可用于构建 non-HLOS 映像。

关于下载安装 Hexagon 工具链软件的具体步骤，参考[Q2]。更多关于 Hexagon 工具的使用文档，参考[Q3]。

QUALCOMM® 2013.07.09 at 23:19:43 PDT liyh-gionee.com 108.171.248.77

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

3 软件产品信息

3.1 软件产品识别

这条产品线所用的软件被分为不同的发布包,每个包必须根据以下说明单独下载并整合为一个完整的产品线软件集：

- From chipcode.qti.qualcomm.com [R6]:
  - 专有 non-HLOS 软件发布 (包含所有 nonapps 处理器的专有源代码和固件映像)
  - 专有 HLOS 发布 (包含 apps 处理器 HLOS 的专有源代码和固件映像)
- From codeaurora.org [R4]:
  - 开源 HLOS 版本 (包含 apps 处理器 HLOS 的开源代码)

专有的 non-HLOS 包是由单个组件版本的集合构建而成的伞形结构，组件版本已整合。专有和开源 HLOS 包需从独立源中获取并根据 3.2.2 描述的下载说明进行合并。每个包通过唯一的 build ID 及发布版本号识别。

Figure 3-1 显示如何解码软件发布版本 ID(数字表示 build ID 中的字符位置)。

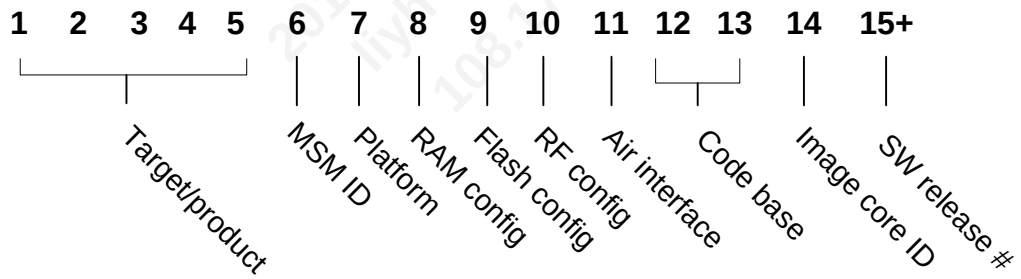

Figure 3-1 解码软件发布版本 ID

Build ID 的字符 11、12、13、14 可取下列值：

- 字符 11–空口
  - N
  - A – UMTS
  - B –多模
  - C – 只支持 C2K

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

- 字符 12 和 13–码基
  - AZ – L4
  - LY – Linux Android
- 字符 14-映像核 ID
  - A – Apps (HLOS 专有)
  - N – NON-HLOS 伞形结构、集成包包含如下内容：
    - M –MPSS
    - B – 引导映像
    - L – 低功率音频子系统和传感器(ADSP)
    - W – 无线连接组网子系统(WCNSS)
    - R – Resource Power Manager (RPM)
    - T – Trust Zone (TZ)
    - I –系统调试映像 (SDI)

QUALCOMM 2013.07.09 at 23:19:43 PDT liyh-gionee.com 108.171.248.77

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

Figure 3-2 显示了整合的软件发布包。

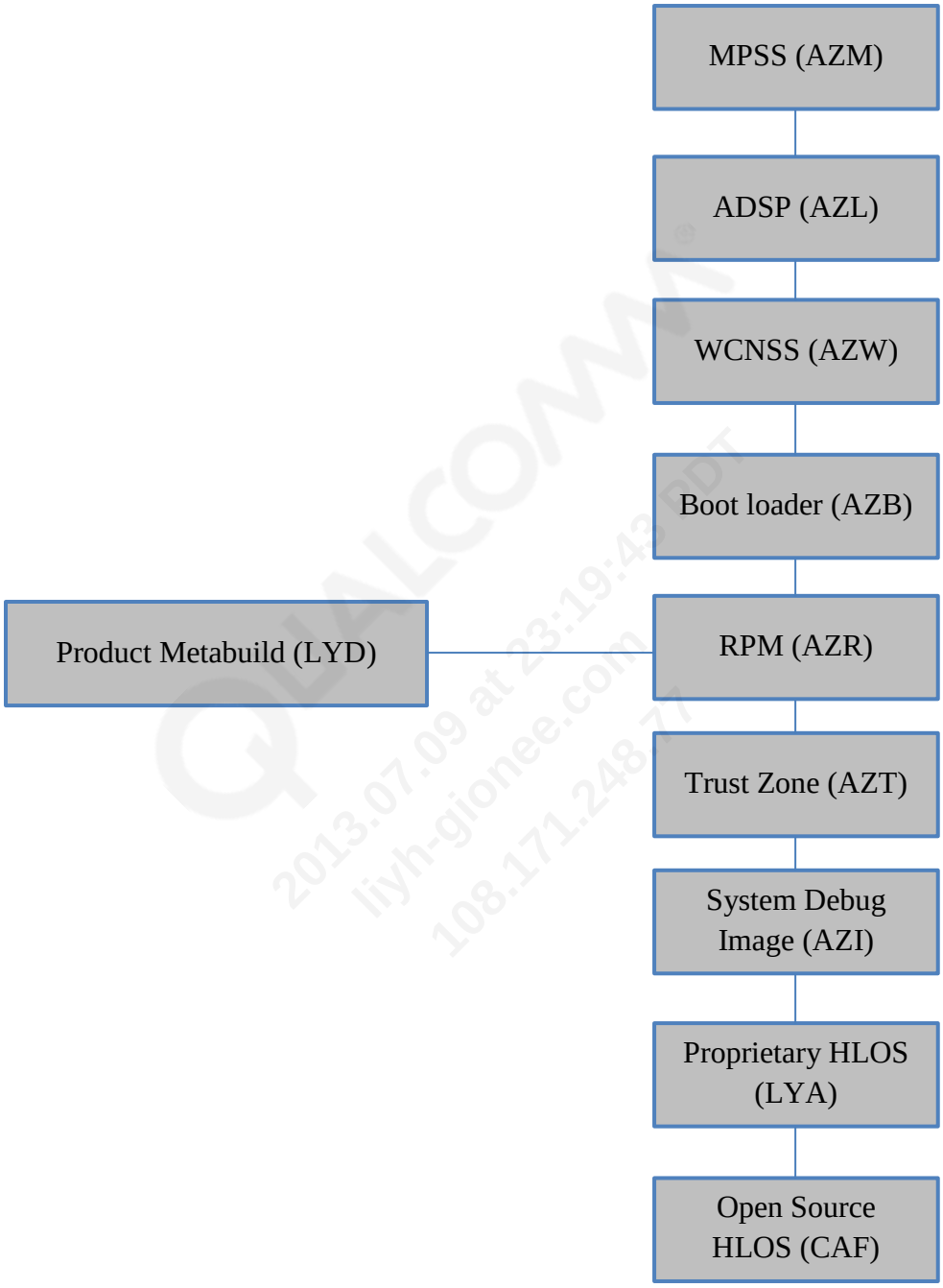

Figure 3-2 整合的软件发布包

Table 3-1 列出了组件发布版本属性，并提供了每个 non-HLOS 构建模块的编译器、Python、Perl 及 Cygwin 版本信息。确保每个工具的构建 PC 的版本是正确的。

Table 3-1 组件发布版本属性

组件发布版本	源代码或二进制文件	构建源代码所需的工具链	Python 版本	Perl 版本	Cygwin	支持的构建主机
Android HLOS (LYA)	源代码	Android gnu 工具链	—	—	—	仅适用于 Linux
MPSS (AZM)	源代码	Hexagon 5.0.07	Python 2.6.2	Perl 5.8.x 仅适用于 Linux 构建	仅适用于 Windows 构建；只需要 tee.exe	Linux, Windows XP, and Windows 7
ADSP (AZL)	二进制文件	—	—	—	—	—
Boot loaders (AZB)	源代码	ARM 编译工具 5.01 update 3 (build 94)	Python 2.6.2	Perl 5.8.x 仅适用于 Linux 构建	仅适用于 Windows 构建；只需要 tee.exe	Linux, Windows XP, and Windows 7
RPM (AZR)	源代码	ARM 编译工具 5.01 update 3 (build 94)	Python 2.6.2	—	仅适用于 Windows 构建；只需要 tee.exe	仅适用于 Linux、Windows XP 和 Windows 7
TZ (AZT)	源代码	ARM 编译工具 5.01 update 3 (build 94)	Python 2.6.2	—	仅适用于 Windows 构建；只需要 tee.exe	仅适用于 Linux、Windows XP 和 Windows 7
WCNSS (AZW)	二进制文件	ARM 编译工具 5.01 update 3 (build 94)	Python 2.6.2	通常情况下不需要,除非使用某些工具进行提取。推荐使用 5.8.x 及以上版本	仅适用于 Windows 构建；只需要 tee.exe	仅适用于 Linux、Windows XP 和 Windows 7
SDI (AZI)	源代码	ARM 编译工具 5.01 update 3 (build 94)	Python 2.6.2	—	仅适用于 Windows 构建；只需要 tee.exe	仅适用于 Linux、Windows XP 和 Windows 7

3.2 访问软件

本节描述了如何从指定的授权发布源获取(下载)软件。一旦软件下载完成,将直接转到 Compilation/build procedures 进行软件构建。

3.2.1 下载高通专有软件

non-HLOS 和专有 HLOS 软件版本发布在高通 ChipCode 网站(R6)。高通 ChipCode 网站是高通新发布系统的一部分，它替代了已有的源代码发布使用的 Documents and Downloads。

客户可在网站的指定联络点下载已授权的软件。

高通 ChipCode 网站的新用户，请参阅 [Q14] 和 [Q15]。 [Q14] 和 [Q15]提供了 ChipCode 网站概述，说明了如何在系统安装 Git 及进行软件代码、文档和软件工具的基本和高级下载。所有已知问题和热门主题，及视频教程链接，请登录在线帮助 Wiki [R7]。

为保证平稳的构建过程,需在构建 PC 中创建一个顶级目录并解压每个存档文件以生成以下目录结构。在以下示例中， <target_root>是顶级目录。

```
<target_root>          /common/
                       /adsp_proc/
                       /boot_images/
                       /common/
                       /debug_image/
                       /modem_proc/
                       /rpm_proc/
                       /trustzone_images/
                       /wcnss_proc/Q
                       /LINUX/
                       contents.xml
```

NOTE: 确保 contents.xml 文件位于所示根文件夹。

3.2.2 下载开源HLOS软件

可从两方面获取 Linux Board Support Package (BSP)发布版本: 从高通 ChipCode 网站[R6]获得专有发布版本，及从 Code Aurora Forum (CAF)网站获取开源版本。

按以下步骤下载开源软件：

1. 查看发布说明，注意标题页显示的 build ID。
2. 登陆 https://www.codeaurora.org/xwiki/bin/QAEP/，在发布分支表里找到包含匹配构建 ID 的发布分支。登陆 https://www.codeaurora.org/xwiki/bin/QAEP/release 查看所有 builds。
3. 在空的工作目录下,运行 repo init 命令，该命令带有发布分支表列出的正确分支和清单。

   ```
   $ repo init -u git://codeaurora.org/platform/manifest.git -b release -m
   [manifest] –repo-url=git://codeaurora.org/tools/repo.git
   ```

4. 输入 repo sync 命令。

   ```
   $ repo sync
   ```

5. Repo sync 结束后，将 vendor/qcom/proprietary 目录树从专有 HLOS 发布中复制到工作区中的开源 HLOS 树中。

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

3.3 Compilation/build步骤

3.3.1 Non-HLOS

3.3.1.1 设置构建的shell环境

运行 non-HLOS build 命令之前,必须设置特定的 shell 环境,以确保正确的执行路径和工具链配置。具体的环境设置由于主机的软件安装会略有差异，但可能和以下“myenviron_amss.sh”脚本示例类似（供 Linux 系统参考），其中路径设置为指向 ARM 工具链 lib，include，bin 及许可文件配置。

```
#
# myenviron_amss_8626
#
export ARMLMD_LICENSE_FILE=<mylicense_file>@<mylicense_server>

ARM_COMPILER_PATH=/pkg/qct/software/arm/RVDS/5.01bld94/bin64
PYTHON_PATH=/usr/local/python/2.6.2/bin
MAKE_PATH=/pkg/gnu/make/3.81/bin

export ARMROOT=/pkg/qct/software/arm/RVDS/5.01bld94
export ARMTOOLS=ARMCT5.01
export ARMLIB=$ARMROOT/lib
export ARMINCLUDE=$ARMROOT/include
export ARMINC=$ARMINCLUDE
export ARMBIN=$ARMROOT/bin64

export PATH=$MAKE_PATH:$PYTHON_PATH:$ARMBIN:$ARM_COMPILER_PATH:$PATH
export ARMHOME=$ARMROOT
export HEXAGON_ROOT=/pkg/qct/software/hexagon/releases/tools
export HEXAGON_RTOS_RELEASE=5.0.01
export HEXAGON_Q6VERSION=v4
export HEXAGON_IMAGE_ENTRY=0x08400000
```

3.3.1.2 构建MPSS

构建 MPSS 步骤如下 （<target_root>是在 3.2.1 中创建的顶级目录）：

1. 验证以下环境变量已按 3.3.1.1 的描述设置。
   - HEXAGON_ROOT
   - HEXAGON_RTOS_RELEASE
   - HEXAGON_Q6VERSION
   - HEXAGON_IMAGE_ENTRY

2. 导航到以下目录：

   ```
   cd <target_root>/modem_proc/build/ms
   ```

3. 根据构建环境，选择以下选项之一：

Table 3-2 构建MPSS的指令

构建环境	命令
Linux	**build.sh 8x26.gen BUILD_ID=AAAAANAZ BUILD_VER=<version_number>**
Windows	**build.cmd 8x26.gen BUILD_ID=AAAAANAZ BUILD_VER=<version_number>**

NOTE: 以上命令中, <version_number>是由构建 ID 的最后四位(数值)字符决定；例如：构建 ID 为 M8626AAAAANLYD1010 的发布版本, 其<version_number>是 1010。

3.3.1.3 构建boot loaders

步骤如下：

1. 如是 linux 系统，验证以下路径已设置。参考引导构建中的 setenv.sh：<target_root>\boot_images\build\ms\setenv.sh。

```
export ARMLMD_LICENSE_FILE=<LICENSE FILE INFO>
export ARM_COMPILER_PATH=/<Path to compiler>/arm/RVDS/5.01bld94/bin64
export PYTHON_PATH=/<Path to python>/python/2.6.6/bin
export MAKE_PATH=/<Path to make>/gnu/make/3.81/bin
export ARMTOOLS=ARMCT5.01
export ARMROOT=/<Path to compiler>/arm/RVDS/5.01bld94
export ARMLIB=$ARMROOT/lib
export ARMINCLUDE=$ARMROOT/include
export ARMINC=$ARMINCLUDE
export ARMBIN=$ARMROOT/bin64
export PATH=$MAKE_PATH:$PYTHON_PATH:$ARM_COMPILER_PATH:$PATH
export ARMHOME=$ARMROOT
export_armlmd_license
```

2. 如是 Window 系统，验证以下路径已设置：

```
set ROOT_DRIVE=C:
set APPS_DIR=Apps
set APPS_PATH=%ROOT_DRIVE%\%APPS_DIR%
set UTILROOT=%ROOT_DRIVE%\util
set GNUPATH=%UTILROOT%\cygwin\bin
set CRMPERL=C:\CRMApps\apps\Perl\bin
set PERLPATH=C:\perl\bin
```


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

```
set PYTHON_PATH=C:\CRMApps\apps\Python262
set ARMTOOLS=ARMCT501
set ARMROOT=%APPS_PATH%\ARMCT5.01\94
set ARMCC5BIN=%ARMROOT%\bin
set ARMCC5INC=%ARMROOT%\include
set ARMCC5LIB=%ARMROOT%\lib
set
path=%PERLPATH%;%CRMPERL%;%PYTHON_PATH%;%GNUPATH%;%ARMCC5BIN%;%UTILROOT%
;%path%
```

3. 验证 ARM 许可的路径(ARMLMD_LICENSE_FILE)已按 3.3.1.1 中的描述更新。
4. 导航至以下目录：

   ```
   cd <target_root>/boot_images/build/ms
   ```

   其中最高等级目录<target_root>已在 3.2.1 中创建。
5. 根据构建环境或版本，选择以下命令之一:

Table 3-3 构建bootloader的指令

构建环境/版本	命令
Linux 构建环境	
MSM8626 (Rel 1010 及后续版本)	构建引导映像 **$ ./build.sh TARGET_FAMILY=8x26 BUILD_ID=FAAAANAZ** 清除构建 **$ ./build.sh -c TARGET_FAMILY=8x26 BUILD_ID=FAAAANAZ** 根据具体配置，可能需要修改脚本 boot_images/build/ms/build_8626.sh 为： **if [-e "setenv.sh"]; then** **- source setenv.sh** **+ source ./setenv.sh** **fi**
Windows 构建环境	
MSM8626 (Rel 1010and later) MSM8626 (Rel 1010 及后续版本)	构建引导印象 **build.cmd TARGET_FAMILY=8x26 BUILD_ID=FAAAANAZ** 清除构建 **build.cmd -c TARGET_FAMILY=8x26 BUILD_ID=FAAAANAZ**

3.3.1.4 构建TrustZone映像

构建 MSM8x26 TrustZone 映像，步骤如下：

1. 导航至以下目录:

   ```
   cd <target_root>/trustzone_images/build/ms
   ```

2. 运行以下命令构建所有映像:

Table 3-4 构建 TrustZone 的指令

构建环境	命令
Linux	**build.sh CHIPSET=msm8x26 tz**
Windows	**build.cmd CHIPSET=msm8x26 tz**

3.3.1.5 构建RPM

使用以下命令构建 RPM (<target_root>为最高级别目录)。确保工具版本为 Table 3-1 中指定的版本。

注意 Linux 环境下的构建可能对于早期 ES 版本不起作用。

1. 打开命令提示符，并更改到以下目录：

   ```
   cd <target_root>\rpm_proc\build
   ```

2. 根据构建环境， 选择下列选项之一：

Table 3-5 构建 PRM 的指令

构建环境	命令
Linux	**./build_8x26.sh**
Windows	**build_8x26.bat**

3. 选择下列选项之一用于清除构建：

Table 3-6 解除 RPM 的指令

构建环境	命令
Linux	**./build_8x26.sh –c**
Windows	**build._8x26.bat –c**

NOTE: rpm.mbn 可在 rpm_proc/build/ms/bin/AAAAANAA 中找到。

3.3.1.6 构建SDI

使用以下命令构建 SDI (<target_root>为最高级别目录)。确保工具版本为 Table 3-1 中指定的版本。

1. 打开命令提示符，更改到以下目录：

   ```
   cd <target_root>\build_image\build\ms
   ```

2. 根据构建环境， 选择以下选项之一：

Table 3-7 构建SDI的指令

构建环境	命令
Linux	`./b8x26.sh TARGET_FAMILY=8x26 sdi BUILD_VER=<major version>  BUILD_ID=AAAAANAZ BUILD_MIN=<minor version>`
Windows	`b8x26.cmd TARGET_FAMILY=8x26 sdi BUILD_VER=<major version> BUILD_ID=AAAAANAZ BUILD_MIN=<minor version>`

在以上命令中，如可能，主次版本和高通版本号相同。例如：如发布 M8626AAAAANLYD1005，保持 BUILD_VER=1 以及 BUILD_MIN=1。

NOTE: sdi.mbn 可在 build_image/build/ms/bin/AAAAANAAZ 中找到。

3.3.1.7 构建ADSP

ADSP 映像已提供，因此无需从源代码进行构建。不过在某些情况下，可以使用以下命令构建 ADSP (<target_root>为最高级别目录)。确保工具版本为 Table 3-1 中指定的版本。

1. 打开命令提示符，更改到以下目录。
2. 根据构建环境， 选择以下选项之一：

Table 3-8 构建ADSP的指令

构建环境	命令
Linux	`./build_adsp_image.sh "0x70ffffff" CHIPSET=msm8x26 all`
Windows	`build.cmd 0x70ffffff “CHIPSET=msm8x26” all`

3.3.1.8 更新NON-HLOS.bin

如果重新编译了 MPSS、ADSP 或 WCNSS，使用以下命令更新具有新映像的 NON-HLOS.bin 文件（<target_root>是在 3.2.1 中创建的顶级目录）。

1. 导航到以下目录：

```
cd <target_root>/common/build
```

2. 输入以下命令：

```
python update_common_info.py
```

3.3.2 应用处理器Android HLOS

构建应用处理器 Android HLOS, 步骤如下：

1. 在 BASH shell 中，导航到 Android 源代码树基目录。

```
cd <build id>/LINUX/android
```

2. 输入以下命令，配置构建环境的 shell 设置。

```
source build/envsetup.sh
```

NOTE: 必须使用 source 命令，这样环境设置定义在当前 shell 中。

3. 输入命令 `choosecombo`，选择构建配置。或者不输入参数，查看交互菜单进行选择。

   ```
   choosecombo 1 msm8226 userdebug
   ```

4. 运行 `make` 命令，开始构建（按如下所示，在多核构建机上可使用-j4 选项进行并行构建，以加快构建时间）。

   ```
   make -j4
   ```

QUALCOMM®
2013.07.09 at 23:19:43 PDT
liyh-gionee.com
108.171.248.77

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

4 固件编程

4.1 设备及软件

除了 Table 2-1 列出的软件环境需求，Table 4-1 还列出了其他设备和软件，用于将固件映像编制到目标设备上。

Table 4-1 固件映像编程所需设备和软件

	描述	版本	来源/供应商	用途
1	Qualcomm QPST	2.7.405 或后续版本	Qualcomm, Inc.	使用 QPST 编制固件映像
2	QXDM Professional™	3.14.447 或后续版本	Qualcomm, Inc.	编制 NV 项值，阅读诊断等
2	Lauterbach TRACE32 (T32) CortexA8/Krait 许可扩展	LA-7843X 或 LA-7843	Lauterbach GmbH	使用 JTAG 编制固件映像，进行应用处理器调试
3	Lauterbach T32 QDSP6 许可扩展	LA-3741A	Lauterbach GmbH	Modem 软件处理器、固件处理器，使用 JTAG 进行 ADSP 调试
4	Lauterbach T32 Cortex-M3 许可扩展	LA-7844X or LA-7844	Lauterbach GmbH	使用 JTAG 编制固件映像及进行 RPM 调试
5	Lauterbach T32 ARM9™ 许可扩展	LA-7742X	Lauterbach GmbH	使用 JTAG 进行 Venus 和 WCNSS 调试
6	Lauterbach T32 Windows	Aug 2012 软件版本： R.2012.08.0000 40902 构建：38589--40902.	Lauterbach GmbH	使用 JTAG 编制固件映像及进行调试
7	Android SDK 工具 （主机的 USB 驱动程序、abd、fastboot）	r10 或更高版本 ADB1.0.29 或后续版本	Android 开源项目	Windows adb 和 fastboot 的 Windows 主机 USB 驱动程序和工具
8	高通 USB 网络驱动 Combo	1.0.80 或后续版本	Qualcomm, Inc.	高通组合设备的 Windows 主机 USB 驱动程序

4.2 安装及设置

本节描述了用户必须遵循的所有设备及软件的安装和配置步骤，这些设备和软件可用来创建用于编制每个固件映像和设备的工作环境。

4.2.1 安装T32

推荐使用 QPST 下载固件,当 QPST 下载不起作用时，必须安装 T32。2012 年 6 月构建的 T32 37825 版本是二进制下载及调试所必须的最低修订版本要求。二进制文件的下载和调试应使用 `common\t32\t32_dap\`下的 T32 链接。

这些文件默认 T32 的安装目录是 `C:\T32`。如果 T32 安装在不同的目录下,必须修改.lnk 快捷文件。

更改.lnk 快捷文件的步骤如下：

1. 定位.lnk 快捷文件。
2. 单击鼠标右键，选择 **Properties。**
3. 在目标字段，更改路径到 t32marm.exe。默认路径为 `C:\t32\t32marm.exe`。

4.2.2 安装Android adb, Fastboot以及主机USB接口

4.2.2.1 在Windows系统中设置USB

Android CDP 需以下 USB 设备文件支持：

- Android USB 驱动器（android_winusb.inf）
  - Android ADB 接口
  - Android Boot Loader 接口（fastboot）
- 高通组合 USB Modem/串口驱动器（qcmdm.inf, qcser.inf）
  - Qualcomm HS-USB Android DIAG
  - Qualcomm HS-USB Android Modem
  - Qualcomm HS-USB Android GPS (NMEA)
- 高通组合 USB 网络 Combo 驱动（qcnet.inf）
  - 高通无线 HS-USB 以太网适配器

安装驱动器之前，必须编辑 qcmdm.inf 和 qcser.inf 文件，确保文件中的每一段都包含了如 4.2.2.2 所述的支持 Android Surf VID / PID 的合适条目。

其他关于在 Windows 系统安装 USB 驱动器，故障排查，重建 Windows USB 主机驱动器的细节，请查看[Q11]和 [Q12]。

4.2.2.2 在Windows系统下安装adb和fastboot的Android SDK以及USB驱动程序

在 Windows 机器上安装 Android SDK 平台和 USB 启动器组件，步骤如下：

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

1. 访问 http://developer.android.com/sdk/win-usb.html。
2. 按照说明安装 SDK 和 USB 驱动器。
3. 右键单击 **My Computer**, 选择 **Properties→Advanced→Environment Variables**。设置包含 `c:\android-sdk-windows\tools` 的路径。
4. Adb 和 fastboot 的 Android USB 驱动器需添加高通 SURF VID/PID, 以便支持连接到 Sthe URF。编辑 `android-sdk-windows\usb_driver\android_winusb.inf`，在每段中添加高通 VID/PID 行。

```
android_winusb.inf
[Google.NTx86]
;Qualcomm SURF/FFA
%SingleAdbInterface%        = USB_Install, USB\VID_05C6&PID_9025
%CompositeAdbInterface%     = USB_Install, USB\VID_05C6&PID_9025&MI_01
%SingleBootLoaderInterface% = USB_Install, USB\VID_18D1&PID_D00D

[Google.NTamd64]
;Qualcomm SURF/FFA
%SingleAdbInterface%        = USB_Install, USB\VID_05C6&PID_9025
%CompositeAdbInterface%     = USB_Install, USB\VID_05C6&PID_9025&MI_01
%SingleBootLoaderInterface% = USB_Install, USB\VID_18D1&PID_D00D
```

此外，确定[Strings]段中包含匹配的条目。

```
[Strings]
SingleAdbInterface          = "Android ADB Interface"
CompositeAdbInterface       = "Android Composite ADB Interface"
SingleBootLoaderInterface   = "Android Bootloader Interface"
```

5. Adb 客户端（adb.exe）支持内置的可识别的 USB VID/PID 设备表。创建 `%USERPROFILE%\.android` 目录（如果此目录不存在）以添加 SURF 或其他设备至可识别设备表。
6. 导航至`%USERPROFILE%\.android` 目录。
7. 在`%USERPROFILE%\.android` 目录下，创建或编辑 `adb_usb.ini` 文件。如果此文件已存在，它应该包含 **DO NOT EDIT** 信息。忽略该信息，在文件结尾处添加包含 0x05C6 的命令行。

NOTE: 不要运行 **android update adb** 命令，此命令会重置文件内容并且覆盖刚添加的命令行。

编辑后的 adb_usb.ini 文件应如下所示：

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

```
# ANDROID 3RD PARTY USB VENDOR ID LIST -- DO NOT EDIT.
# USE 'android update adb' TO GENERATE.
# 1 USB VENDOR ID PER LINE.
0x05C6
```

8. 从 Documents and Downloads 中获取高通组合 USB 驱动器的最新版本。（使用高通组合 USB 网络 combo 驱动，以包含网络接口的支持。）

   Android 调试可分别通过 composition 9025 和 9026 在用户空间中启用和禁用。

```
[Models]
%QUALCOMM90252% = Modem2, USB\VID_05C6&PID_9025&MI_02
%QUALCOMM90261% = Modem2, USB\VID_05C6&PID_9026&MI_01

[Models.NTamd64]
%QUALCOMM90252% = Modem2, USB\VID_05C6&PID_9025&MI_02
%QUALCOMM90261% = Modem2, USB\VID_05C6&PID_9026&MI_01

[Models.NTia64]
%QUALCOMM90252% = Modem2, USB\VID_05C6&PID_9025&MI_02
%QUALCOMM90261% = Modem2, USB\VID_05C6&PID_9026&MI_01

[Strings]
QUALCOMM90252 = "Qualcomm Android Modem 9025"
QUALCOMM90261 = "Qualcomm HS-USB Android Modem 9026"

qcser.inf
[QcomSerialPort]
%QcomDevice90250%  = QportInstall00, USB\VID_05C6&PID_9025&MI_00
%QcomDevice90253%  = QportInstall00, USB\VID_05C6&PID_9025&MI_03
%QcomDevice90260%  = QportInstall00, USB\VID_05C6&PID_9026&MI_00
%QcomDevice90262%  = QportInstall00, USB\VID_05C6&PID_9026&MI_02

[QcomSerialPort.NTia64]
%QcomDevice90250%  = QportInstall00, USB\VID_05C6&PID_9025&MI_00
%QcomDevice90253%  = QportInstall00, USB\VID_05C6&PID_9025&MI_03
%QcomDevice90260%  = QportInstall00, USB\VID_05C6&PID_9026&MI_00
%QcomDevice90262%  = QportInstall00, USB\VID_05C6&PID_9026&MI_02

[QcomSerialPort.NTamd64]
%QcomDevice90250%  = QportInstall00, USB\VID_05C6&PID_9025&MI_00
%QcomDevice90253%  = QportInstall00, USB\VID_05C6&PID_9025&MI_03
%QcomDevice90260%  = QportInstall00, USB\VID_05C6&PID_9026&MI_00
%QcomDevice90262%  = QportInstall00, USB\VID_05C6&PID_9026&MI_02
```


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

```
[Strings]
QcomDevice90250 = "Qualcomm HS-USB Android DIAG 9025"
QcomDevice90253 = "Qualcomm HS-USB Android GPS (NMEA)9025"
QcomDevice90260 = "Qualcomm HS-USB Android DIAG 9026"
QcomDevice90262 = "Qualcomm HS-USB Android GPS (NMEA)9026"

qcnet.inf
[QCOM]
qcwwan.DeviceDesc90254 = qcwwan.ndi, USB\VID_05C6&PID_9025&MI_04
qcwwan.DeviceDesc90263 = qcwwan.ndi, USB\VID_05C6&PID_9026&MI_03

[QCOM.NTia64]
qcwwan.DeviceDesc90254 = qcwwan.ndi, USB\VID_05C6&PID_9025&MI_04
qcwwan.DeviceDesc90263 = qcwwan.ndi, USB\VID_05C6&PID_9026&MI_03

[QCOM.NTamd64]
qcwwan.DeviceDesc90254 = qcwwan.ndi, USB\VID_05C6&PID_9025&MI_04
qcwwan.DeviceDesc90263 = qcwwan.ndi, USB\VID_05C6&PID_9026&MI_03

[Strings]
qcwwan.DeviceDesc90254     = "Qualcomm Wireless HS-USB Ethernet Adapter
9025"
qcwwan.DeviceDesc90263     = "Qualcomm Wireless HS-USB Ethernet Adapter
9026"
```

4.2.2.3 Linux环境下安装USB驱动

修改 USB 驱动需要在 Linux 环境下安装 Android 调试桥（ADB）。

安装步骤如下：

1. 进入下列路径。

   ```
   cd /etc/udev/rules.d/
   ```

2. 输入下列命令：

   ```
   sudo vi 50-android.rules
   ```

   运行结果应与下面结果类似：

```
#Sooner low-level bootloader
SUBSYSTEM=="usb", SYSFS{idVendor}=="18d1", SYSFS{idProduct}=="d00d",
MODE="0664", GROUP="plugdev"
# adb composite interface device 9025
SUBSYSTEM=="usb", SYSFS{idVendor}=="05C6", SYSFS{idProduct}=="9025",
MODE="0664", GROUP="plugdev"
```


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

3. 文件编辑完成后，若要查看与 Linux box 相连的目标设备名单，输入下列命令：

```
lsusb
```

4.2.2.4 在Linux环境下安装ADB和fastboot

一个 build 完成后，Linux 的 ADB 和 fastboot 可执行文件位于 Android 软件版本的 android\out\host\linux-x86\bin 目录下。这些可执行文件是标准的编译流程的一部分。要运行 ADB 或者 fastboot，Linux 机器可能需要有 sudo 或者根访问权限。

1. 如果 android\out\host\linux-x86\bin 目录不在可执行的搜索路径下，需要将该目录加进去。如果已经存在，跳到第二步。
   a. 输入下列命令：

   ```
   source build/envsetup.sh
   ```

   b. 输入下列命令：

   ```
   choosecombo 1 msm8226 userdebug
   ```

2. 输入下列命令来验证 fastboot 已经将 Android 映像刷到目标版本：

   ```
   sudo fastboot devices
   ```

3. 验证在输入 fastboot 设备命令后，设备通过 fastboot 显示出来。

4.3 编程步骤

该章节描述了对于每个 firmware 图像和设备进行编程和再编程的步骤。

4.3.1 用QPST对eMMC进行编程

如果没有图片刷到 eMMC （生产线上第一次下载二级制文件会出现这种情况），PBL 会将 USB 列为高通 Sahara 下载界面并等待主机 PC 的下载命令。这种情况下，需要刷最小化初始二进制文件来下载剩下的 Android 图像。

NOTE: 在 MSM8626 CDP 上，你可以通过清除该设备，或者使用 dip switch S7 pin 3 来强制设备进入该模式。

NOTE: 你的 QPST 版本必须要符合 4.1 章节的最低要求。

1. 创建一个包含下列内容的 XML 文件，命名为 sahara.xml。该文件必须包含到 **emmcbld programmer** 的整个路径。

```
<?xml version="1.0" encoding="utf-8"?>
<sahara_config>
<images>
<image image_id="13" image_path="<boot_build_path>
\boot_images\build\ms\bin\EMMCBLD\MPRG8626.mbn" programmer="true" />
```


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

```
</images>
</sahara_config>
```

2. 打开 QPST 配置。你应该可以看到设备列出来，工作在 Download Mode。

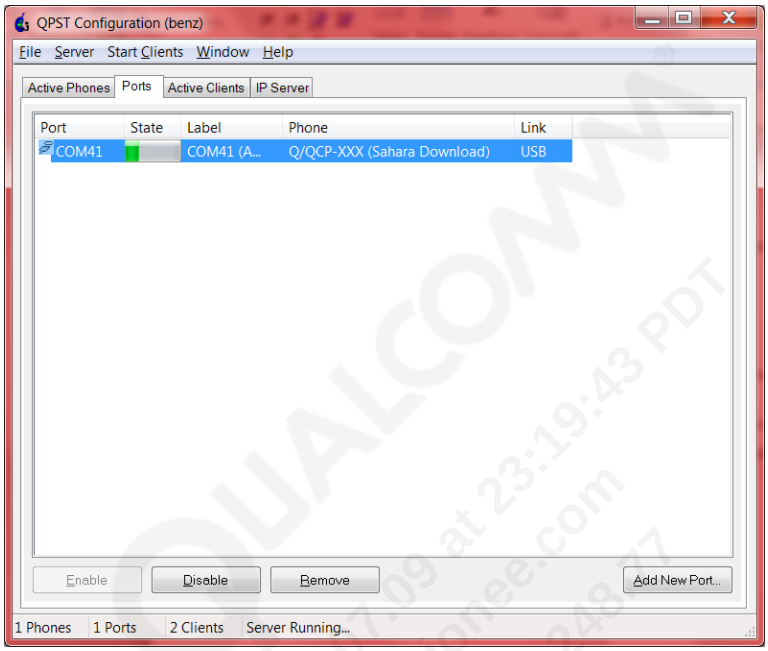

Figure 4-1 QPST 配置

3. 打开 QPST eMMC Software Download 应用。
4. 浏览选择 Q/QCP-XXX (Sahara Download)下列出的设备所在的接口。
5. 确保已经检测过 Program Boot Loaders 和 Program MMC 设备。
6. 在最上面的的空白处选择 **sahara.xml** 作为 Sahara XML 文件。配置如下图所示。

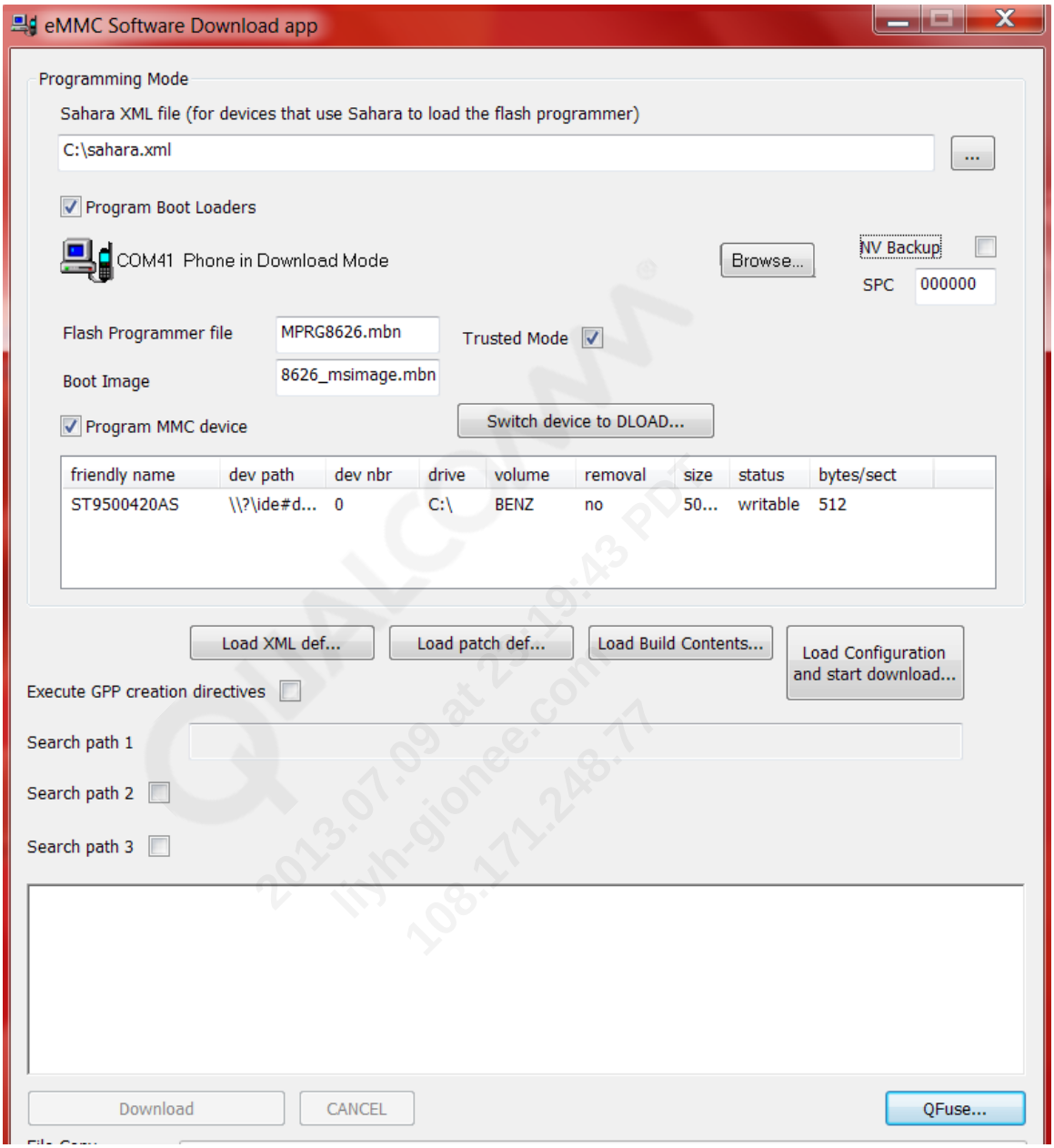

Figure 4-2 eMMC软件下载应用

7. 从 metabuild 当中选择 contents.xml 作为 Load XML definition。
8. 点击 Load Build Contents，界面显示如下图所示。

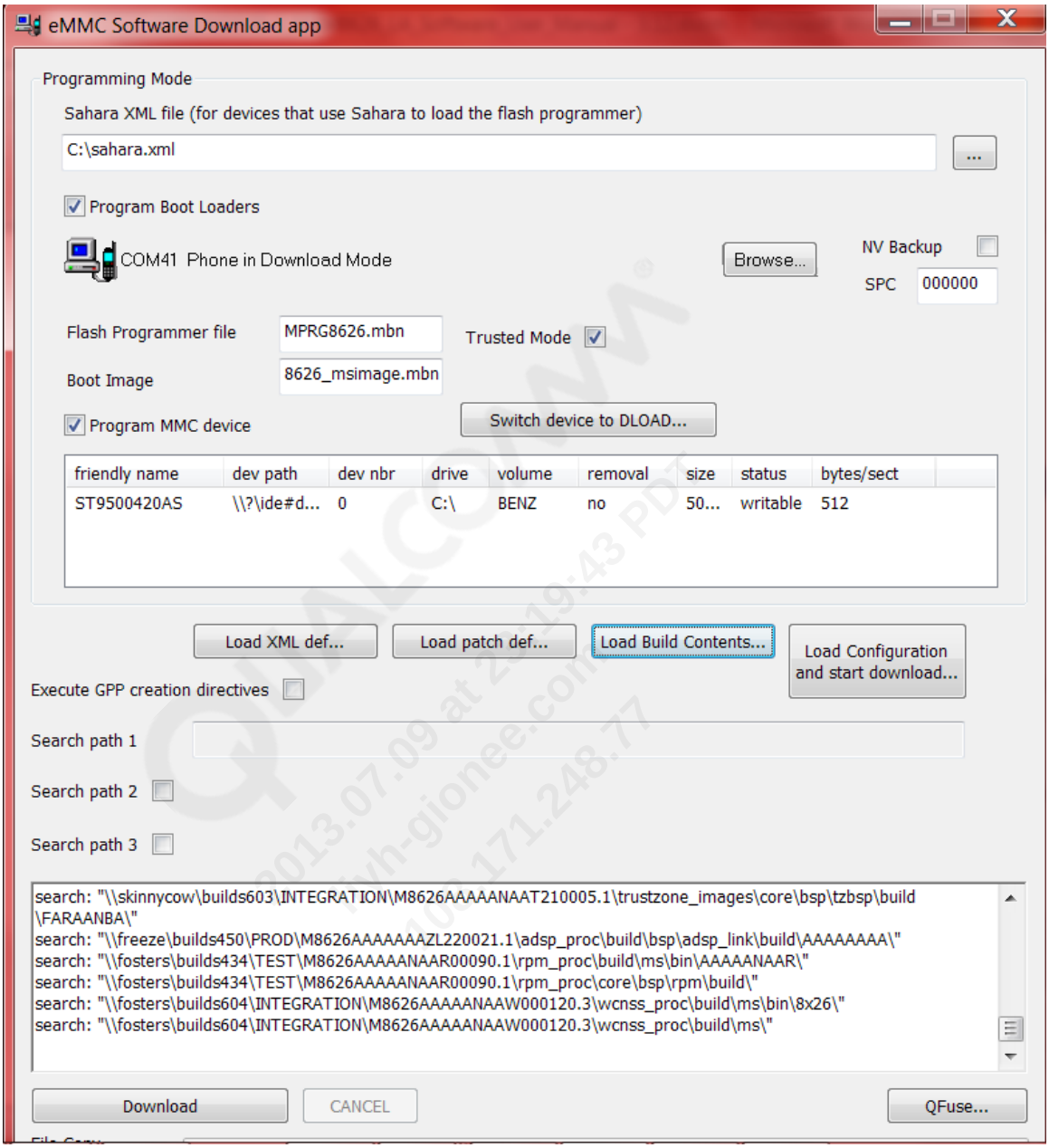

Figure 4-3 eMMC 软件下载 emmcbld 映像

9. 点击 Download。
10. 一旦 emmcbld 图片下载完毕，设备应该重置为 mass storage 模式并且 QPST 应该可以继续载入 build 的剩余部分。如下图所示。

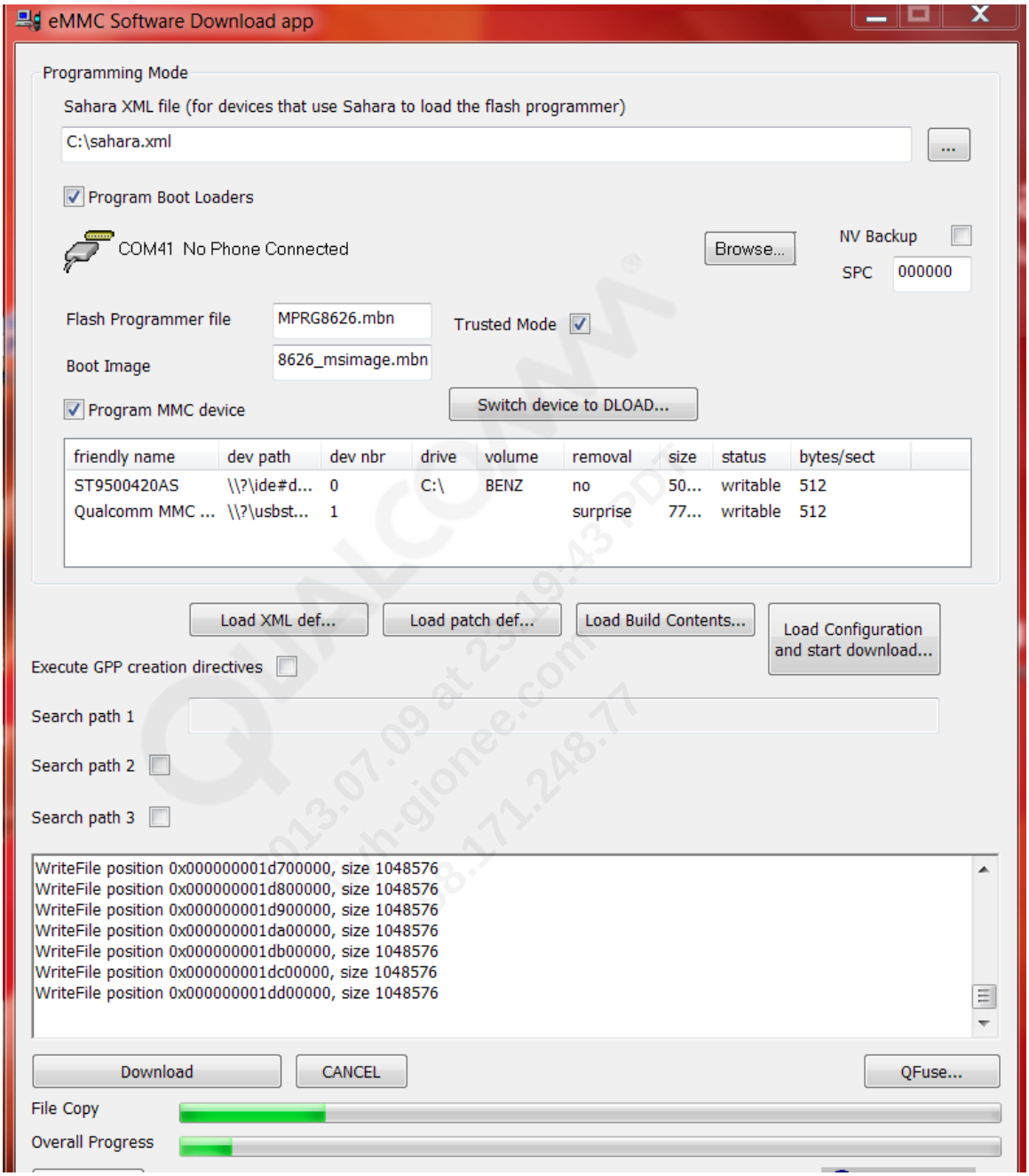

Figure 4-4 用 eMMC 软件下载剩下的映像

11. 下载完成后，设备重置，‘format disks window’弹出。关闭所有弹出的窗口并断开 USB 线重置设备 bootup 到 Android。

NOTE: 如果之前你使用 dip switch S7-3 来强制使用 emergency 模式，你必须此时切换回去并重启设备。否则该设备会返回 emergency 下载模式。

NOTE: 如果是在一个空白的 EMMC 上第一次下载 CDT，建议对其进行刷机。CDT 提供了依赖于平台/设备的数据，比如平台 ID, DDR 硬件参数等等。如果 sbl1 成功从 eMMC 上读取 CDT，

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

将会更新默认的 config_data_table 阵列。该阵列在编译时会链接到 bulid 里。下面是将 CDT 编程到 eMMC 启动分区里的方法：

用 QPST 工具进行 CDT 编程

1. 用 QPST 工具进行 CDT 编程需要：

   Flash programmer: MPRG826.mbn

   XML 文件: 用来加载 flash programmer, 使用 sahara.xml

   XML 文件: 用来加载配置文件。 这些配置文件制定了 rawprogram 和补丁 xml 文件的查找路径以及文件名称。使用如下所示内容的 qrd_prog_cfg.xml 文件：

```
<?xml version="1.0"?>
<configuration>
   <options>
      <!-- NOTE: Defaults are shown here -->
      VERBOSE_FEEDBACK        = true
      SEARCH_CWD_LAST         = true
      HEX_PROGRAMMER          = MPRG8626.mbn
   </options>
   <search_paths>
      <!-- NOTE: Search paths IN ORDER, as in first look here, then there, then there etc -->
      <!-- NOTE: CWD is an implied search path, and it is always last -->
   emmc_cdt_program_script\cdt_prog_xml
   emmc_cdt_program_script\cdt_bin
   </search_paths>
   <rawprogram>
      rawprogram2_qrd.xml
   </rawprogram>
   <patch>
      patch2.xml
   </patch>
   </configuration>
```

   XML 文件：可以在发布文档里找到，或者在客户支持中心申请。rawprogram2_qrd.xml 和 patch2.xml 文件用来加载 CDT cdt_generator.py 和 CDT xml 生成的 CDT 二进制文件:

```
cd boot_images\core\boot\secboot3\scripts
python cdt_generator.py qrd_1.0_platform_jedec_lpddr2_single_channel.xml qrd_1.0_platform_jedec_lpddr2_single_channel.bin
```

2. 确保手机设为紧急下载模式

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

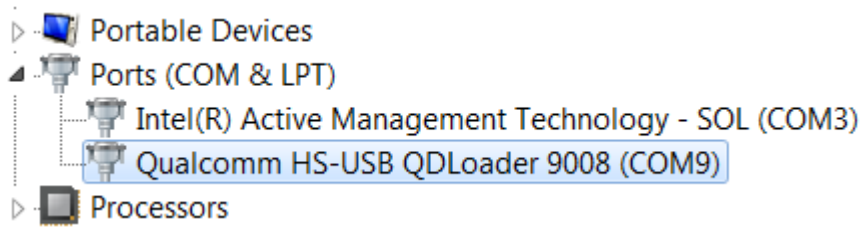

Figure 4-5 紧急下载接口

3. 打开 eMMC Software Download 应用， (作为管理员运行), 提供 sahara xml file, 去勾选 "Program Boot Loaders", "Program MMC device" 和 "NV Backup"。
4. 点击 "Load Configuration and start download..." 并选择 qrd_prog_cfg.xml 对 CDT 进行编程， 具体如下所示：

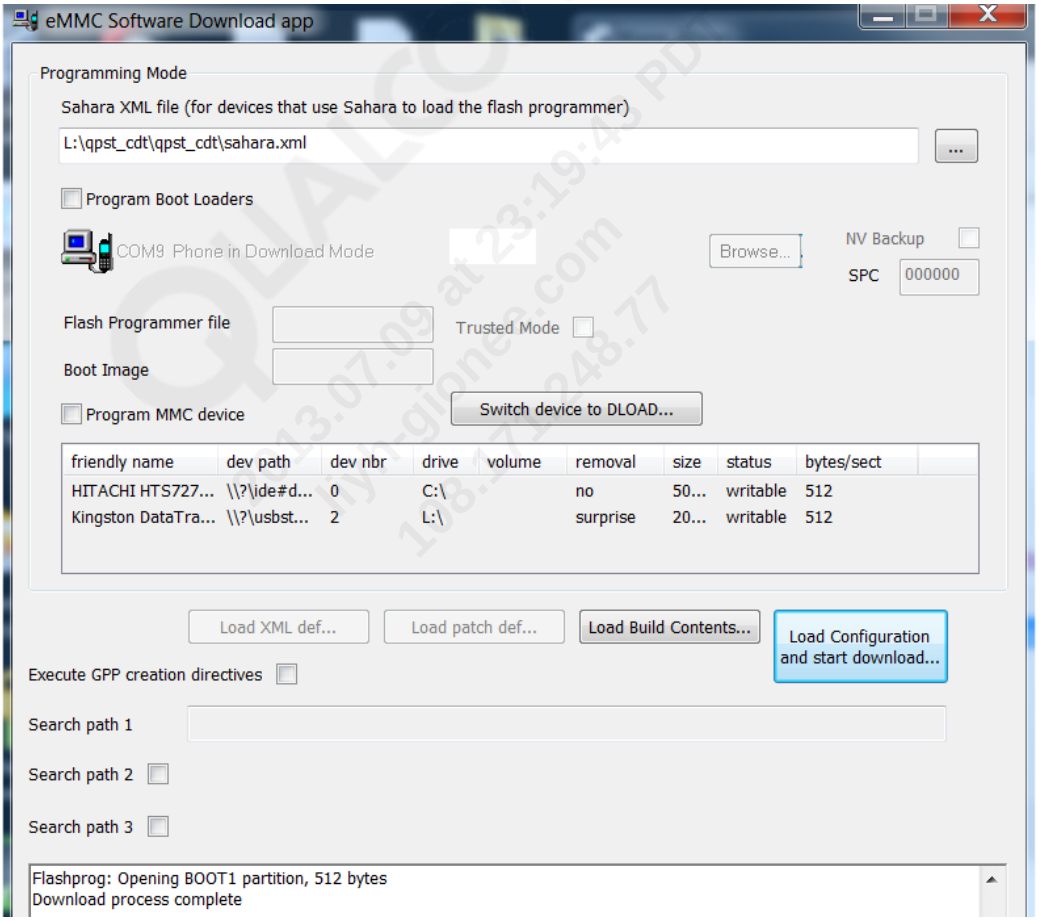

Figure 4-6 用 QPST 来 Flash CDT

用 JTAG 对 CDT 编程:

1. 首先安装 T32 调试环境。（参考 4.3.4）。
2. 在 APPS0 会话中运行 CDT 脚本，成功信息如下所示：

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

```
cd.do
C:\emmc_cdt_program_script\qrd_1.0_lpddr2_single_channel_emmc_cdt_progra
m.cmm
```

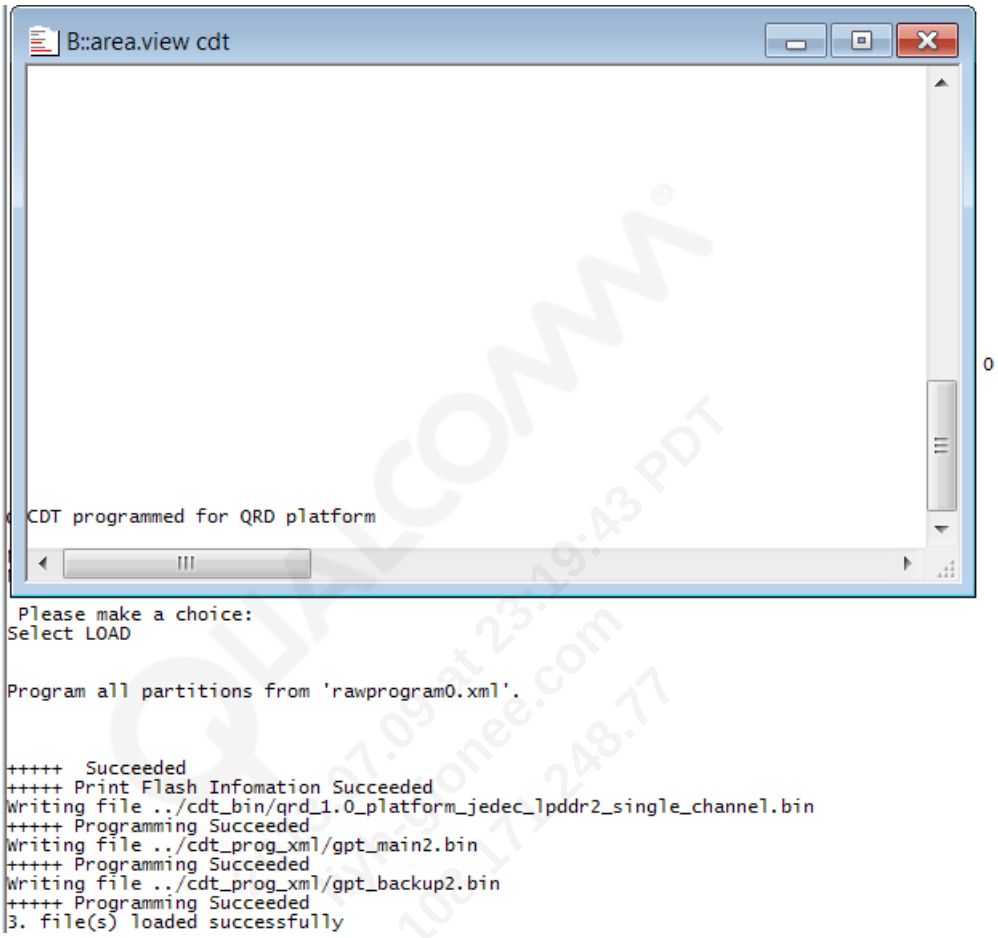

Figure 4-7 用 T32 来 Flash CDT

4.3.2 用fastboot对系统图片进行编程

在使用 Fastboot 对系统图片进行编程之前，Android boot loaders 必须已经 flash 到目标设备。

1. 将 USB 线插到目标设备上。
2. 根据你的 build 环境，选择下列选项之一：
   - 从 Windows 命令 shell 里，运行下面的命令：

     ```
     fastboot devices
     ```
   - 从 Linux，之行下列操作：
     i. 进入下列路径：

     ```
     cd <AndroidRoot>/LINUX/device/out/host/linux-x86/bin
     ```

     ii. 运行下列命令：

     ```
     sudo fastboot devices
     ```

显示注册设备列表

3. 一旦检测出设备，立即将 binaries 刷到目标设备。下列命令将会同时运行所有的 Fastboot 步骤。

```
cd <target_root>/common/build
    fastboot_all.py
```

也可以通过下面的fastboot 命令选项对每个binary选择性地进行

```
fastboot flash modem  <path to NON-HLOS.bin> or <path to APQ.bin>
fastboot flash sbl1 <path to sbl1.mbn>
fastboot flash rpm <path to rpm.mbn>
fastboot flash tz <path to tz.mbn>
fastboot flash aboot <path to emmc_appsboot.mbn >
fastboot flash boot <path to boot.img>
fastboot flash system <path to system.img>
fastboot flash userdata <path to userdata.img>
fastboot flash persist <path to persist.img>
fastboot flash recovery <path to recovery.img>
```

若要生成 fastboot 编程所支持的所有 fastboot 分区列表，参考 LINUX/android/bootable/bootloader/lk/platform/msm_shared/mmc.c 路径下的源代码。

4.3.3 ADB使用ADB将应用刷到Android

按照下列步骤将应用刷到 Android。

1. 将 USB 线插到目标设备上。
2. 进入下列目录：

```
cd <root>/LINUX/device/out/host/linux-x86/bin
```

3. 输入下列命令：

```
sudo adb devices
```

设备需要注册。

4. 进入下列目录：

```
cd <root>/LINUX/device/out/target/product/surf/obj/
APPS/AppName_intermediates/
```

5. 复制下列文件：

```
cp package.apk AppName.apk
```

6. 如下 push 上述文件。

```
adb push AppName.apk /system/app/.
```


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

NOTE: 总的来说，句法是 adb push <file_name> <location_on_the_target>。

4.3.4 用T32对eMMC boot loaders进行编程

按照下列步骤对 boot loaders 进行编程。

1. 在<meta build>\common\t32 文件夹里使用 t32start.cmd 启动 T32。
2. 如下图所示，进入下列路径 Configuration Tree → MSM8X26 → DAP → Podbus Device Chain → Power Trace Ethernet。

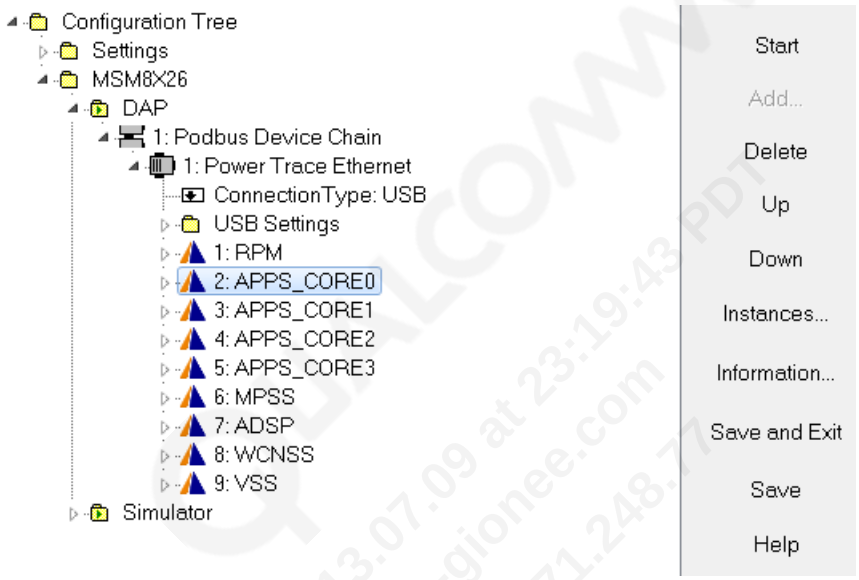

Figure 4-8 T32 配置

3. 选择 RPM 并且点击 Start。
4. 选择 APPS_CORE0 并且点击 Start。
5. 在弹出的 RPM T32 窗口上，选择 **RPM commands → Build Options**，然后选择在 Product Flavor 下选择 asic。
6. 在弹出的 Build Options 窗口，点击 Load。

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

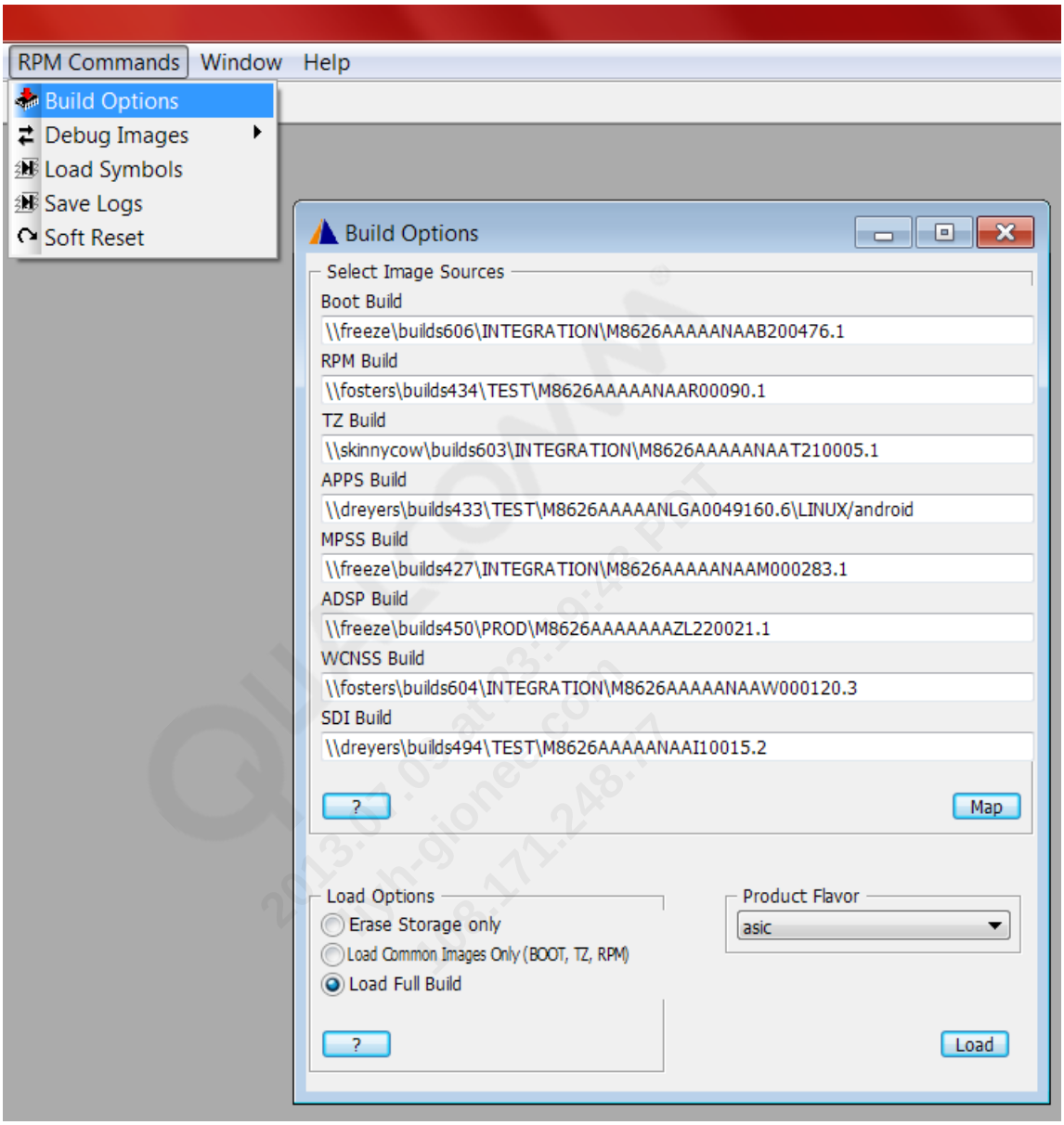

Figure 4-9 T32 加载映像

7. 在 PC 机上用 fastboot 来刷 NON-HLOS.bin 和 apps binaries 时，要保持 USB 与 PC 机连接。Fastboot 下载在 T32 完成对 Bootloader 的编程后自动由脚本启动。

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

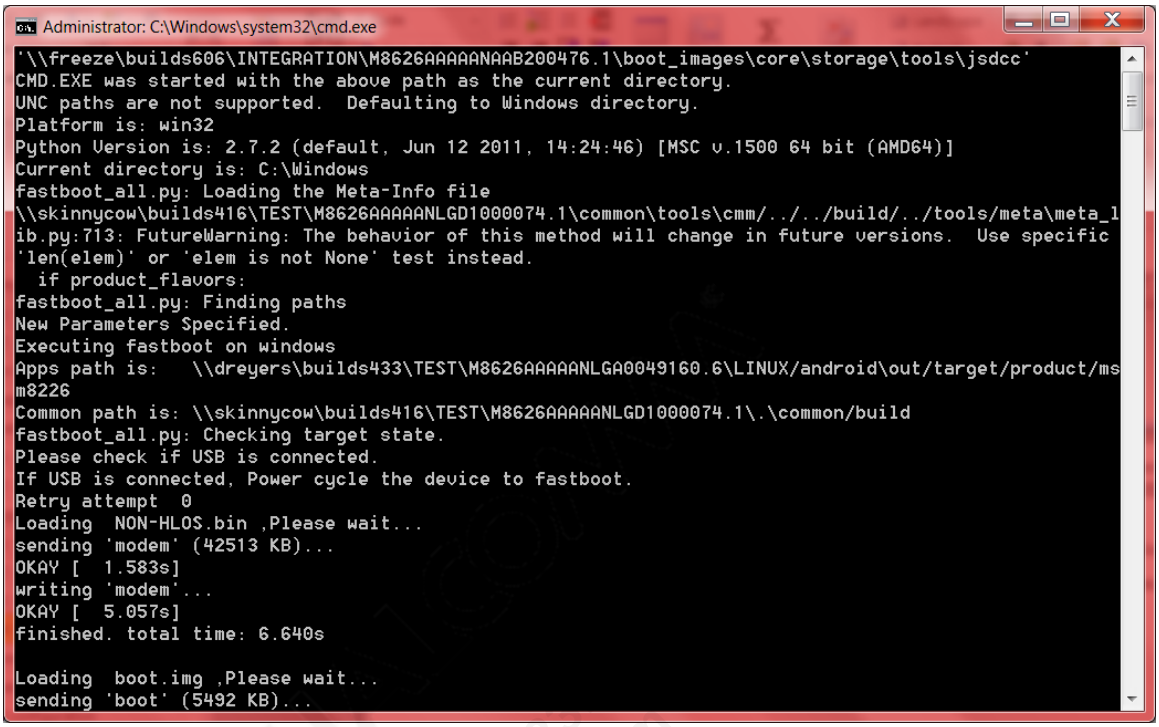

Figure 4-10 刷 NON-HLOS 和应用映像

8. Fastboot 完成二进制文件的刷机后，重启设备。

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

5 操作指导

5.1 初始Bringup

关于初始 Bringup，参考下面的章节内容了解如何安装 SIM 卡槽，主天线和 USB 端口。

5.1.1 MSM8626 CDP UIM配置

参照 Figure 5-1 至 Figure 5-3 了解 CDP SIM 卡槽配置。

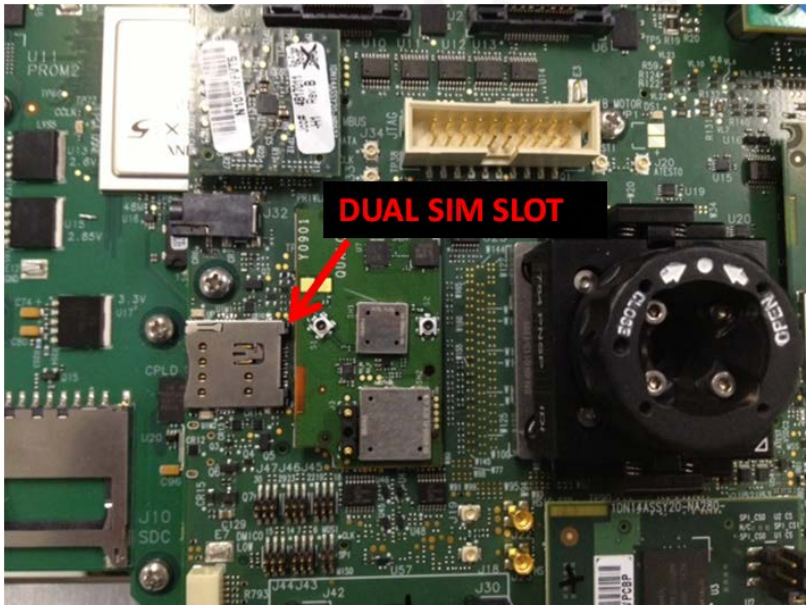

Figure 5-1 基带卡上的双卡槽

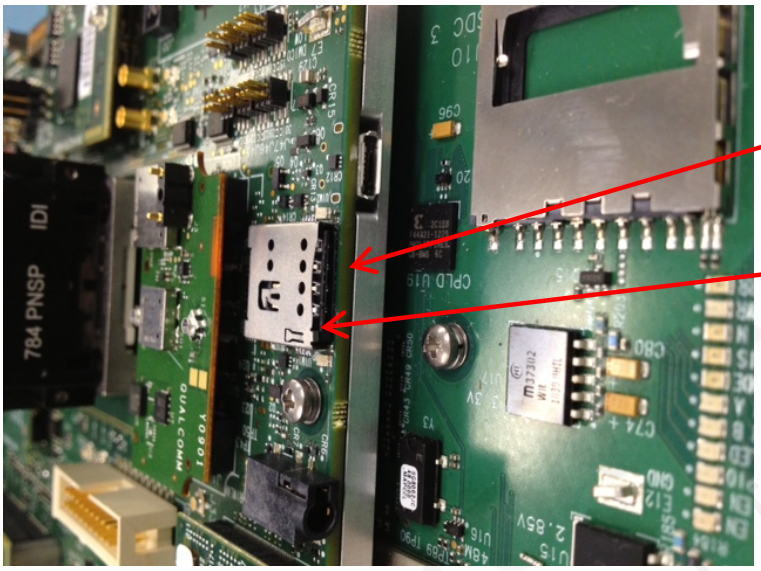

Figure 5-2 基带卡上的双卡槽 （UIM1/UIM2）

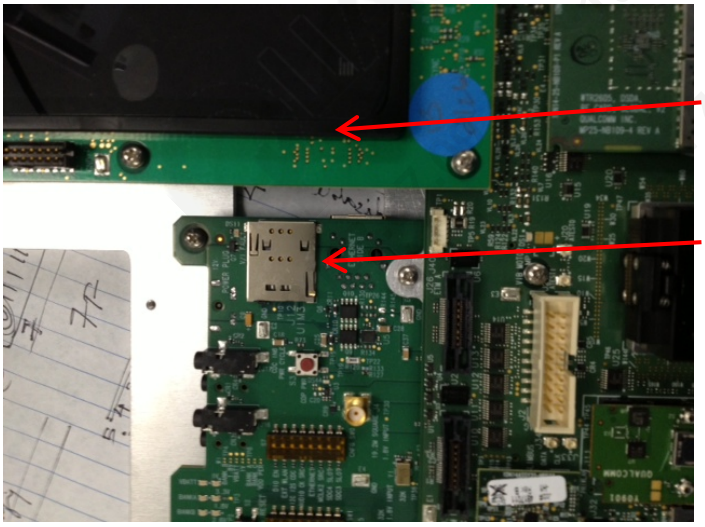

Figure 5-3 基带卡上的 UIM3

5.1.2 MSM8626 MTP UIM 配置

参照 Figure 5-4 至 Figure 5-6 了解 MTP SIM 卡槽配置。

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

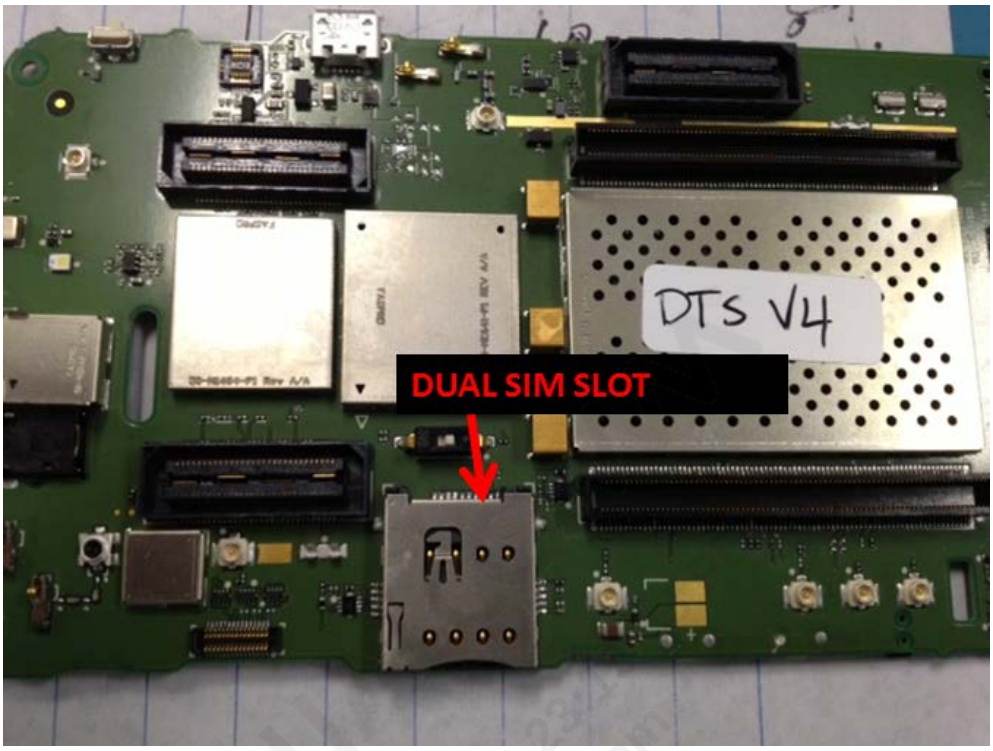

Figure 5-4 基带卡上的双卡槽

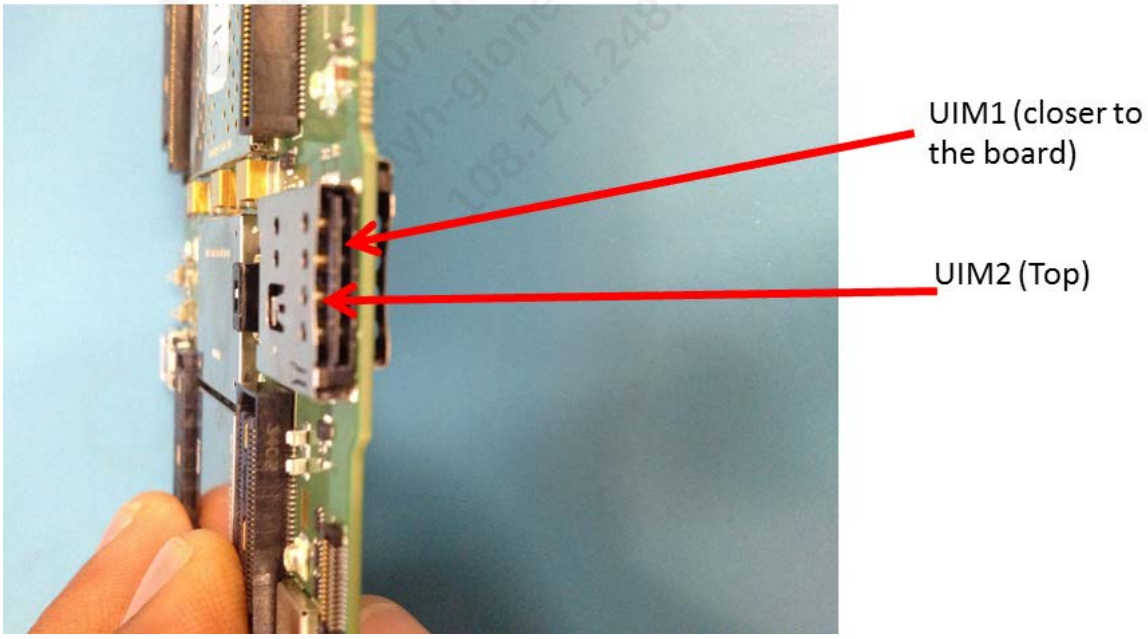

Figure 5-5 基带卡上的双卡槽 （UIM1/UIM2）

Figure 5-6 基带卡上的 UIM3

5.1.3 天线配置

参照 Figure 5-7 了解 CDP 天线配置。

2015.07.09 at 23
liyh-gionee.com
108.171.248.77

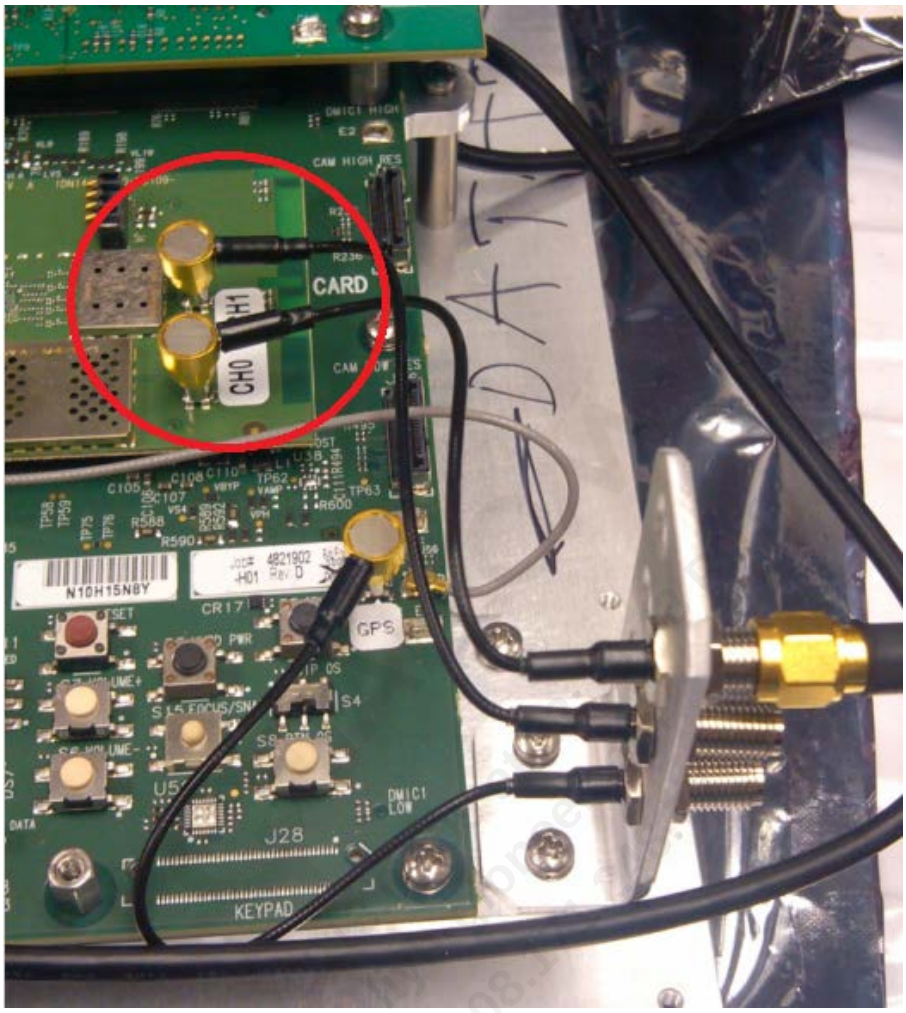

Figure 5-7 CDP 天线配置

5.1.4 USB/JTAG配置

参照 Figure 5-8 了解 CDP USB/JTAG 配置。

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

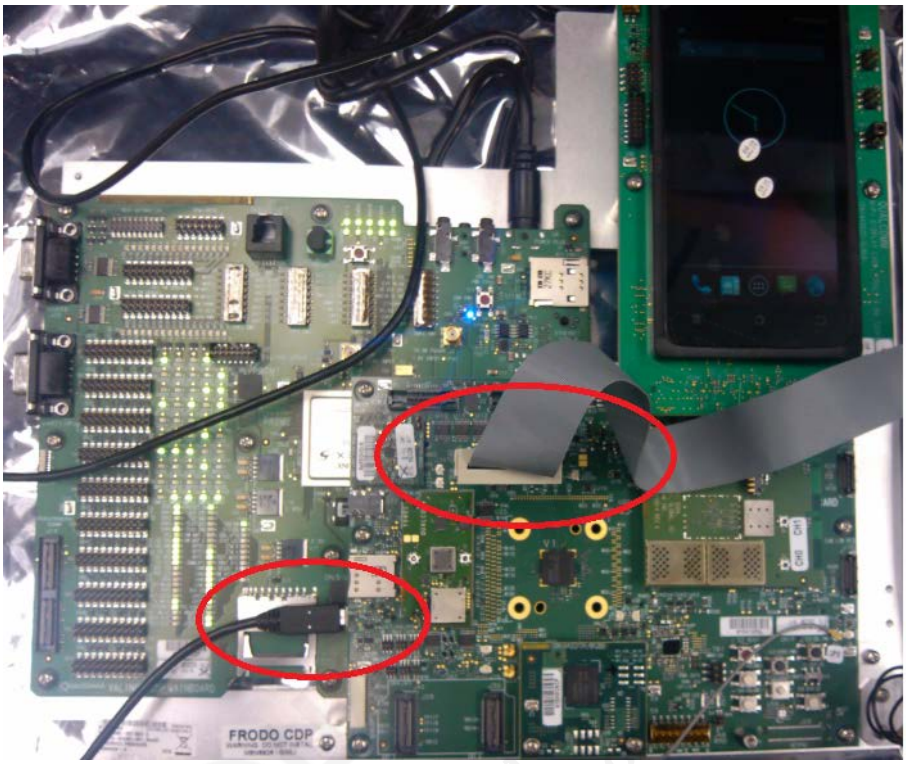

Figure 5-8 CDP USB/JTAG 配置

5.2 DSDS/DSDA NV设置

5.2.1 WCDMA/GSM + GSM的NV配置

关于 WCDMA/GSM + GSM DSDS 的 NV 配置，参见 Table 5-1。

Table 5-1 WCDMA/GSM + GSM DSDS 的 NV 配置

NV 项	卡 1 (WCDMA/GSM) 配置值	卡 2 (GSM) 配置值
00010	17: Auto (WCDMA or GSM) 14: WCDMA only	13: GSM only
00441	0xFFFF (对于特殊频段，设置为 0x380 或 0x387)	与卡 1 值相同
00850	0x02 CS and PS	0x00 CS only
00855	0 RTRE configuration for WCDMA/GSM + GSM （在设置该 NV 前，需要在 QXDM 命令窗口运行 spc 000000 命令）	与卡 1 值相同
00880	0x01 integrity enable	与卡 1 值相同
00881	0x01 ciphering enable	与卡 1 值相同

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

00882	0 fake security enable	与卡 1 值相同
00905	0x0000 fatal error option	与卡 1 值相同
00946	0x0040 IMT band(0CE0 US PCS Band)	0x0040
03649(RRC Version)	Inactive or 0→R99; 1→R5; 2→R6; 3→R7; 4→R8	与卡 1 值相同
03851	0 RxD control	与卡 1 值相同
04118 (HSDPA Cat)	Inactive or 24→DC	与卡 1 值相同
04210 (HSUPA Cat)	Inactive or 6→EUL 2ms; 5→EUL 10 ms	与卡 1 值相同
04398	0→DSDS; 1→SS	与卡 1 值相同
04399	1 detect hardware reset	与卡 1 值相同
06876	00005→WCDMA/GSM to GSM Tune away 00006→DSDS	与卡 1 值相同
06907	1 Dual SIM hardware (双卡硬件) 0 Single SIM hardware (单卡硬件)	与卡 1 值相同

WCDMA/GSM + GSM DSDA 的 NV 配置, 除了 NV70266 (Dual standby preference，双待优选项) 需设置为 2 外，其它项与上表 Table 5-1 相同。

5.2.2 CDMA + GSM NV设置

关于 CDMA + GSM DSDS 的 NV 配置，参见 Table 5-2。

Table 5-2 CDMA + GSM DSDS 的 NV 配置

NV 项	卡 1 (CDMA+ HDR) 配置值	卡 2 (GSM) 配置值
10 (Mode Preference)	4-> : Automatic mode 19: CDMA and HDR only	13: GSM only
475 (HDR SCP Session Status)	0-> Inactive	与卡 1 值相同
850 (Service Domine Preference)	0x02 CS and PS	0x00 CS only
905 (Fatal Error Option)	0x0000 fatal error option	与卡 1 值相同
4204 (HDR SCP Force	0-> HDR SCP Force Release 0 Session	与卡 1 值相同

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

	Configuration	
4964 (HDR SCP Force At Configuration)	0-> HDR rev 0 , 1->HDR rev A, 3->HDR rev B.	与卡 1 值相同
03446 (TRM Configuration)	2, 0	与卡 1 值相同
4398 (UIM Select Default USIM Application	0→DSDS; 1-> SS	与卡 1 值相同
4399 (detect hardware reset)	1 detect hardware reset	与卡 1 值相同
6874 (ASID 1 Data)	255	与卡 1 值相同
6875 (ASID 2 Data)	255	与卡 1 值相同
562 (preference Hybrid Mode)	1 -> Hybrid operation allowed	0 -> Hybrid operation not allowed
6876(Dual standby config Items	00002 (Dual standby preference)	与卡 1 值相同
6907 (NV_UIM_HW_SIM_CONFIG)	1-> Dual SIM 0-> Single SIM	与卡 1 值相同
855 (RTRE configuration)	0 (在设置该 NV 前，需要在 QXDM 命令窗口运行 spc 000000 命令)	与卡 1 值相同

CDMA + GSM DSDA 的 NV 配置, 除了 NV70266 (Dual standby preference，双待优选项) 需设置为 2 外，其它项与上表 Table 5-2 相同。

5.3 呼叫配置

该节详细介绍打电话的配置信息。

5.3.1 1X Voice语音呼叫

预置条件：

- NV 设置 （将 NV # 10 设为 4，自动模式）
- QCN (确保在设备上进行 RF 校准，不要使用金 QCN，并且注意 1X 默认在 SV 链上。
- PRL (必须跟测试的频段/信道匹配，SID 和 NID 也必须匹配。)

Callbox 安装：

- 目前测试已经在 Agilent 8960 上完成。
- 安装指导：
  - 在 System Config/Application Setup 上，选择 CDMA 2000 Lab App B（B 版本或者更高版本）

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

- 在 Call Setup/Call Control 界面上
  - 将 Operating Mode 设为 Active Cell
  - 将 System Type 设为 IS-2000
  - 点击“More”移至五页中的第二页，选择 Cell Info/Cell Parameters。配置 SID/NID。（与 PRL 匹配，或者设置为通配符：SID=0, NID=65535）
- 在 Call Setup/Call Params 界面
  - 设置 Cell 1 Power 为合适的值 （介于-45 ~ -65dBm 之间）
  - 设置 Cell Band 和 Channel， 与 PRL setting 匹配
  - 设置 Protocol Rev 为 6(IS-2000-0)
  - 设置 Radio Config(RC)为(3,3) - (Fwd3, Rvs3)
  - 设置(Fwd3, Rvs3)的 FCH Service Option Setup 为 SO3 (Voice)

设备安装：

- 打电话的分步指导

a. 使用 QXDM Professional (QXDM Pro)的 Call Manager 界面。

b. 将电话号码设置为类似 1234 这样的数字，确保业务选项与 Callbox 设置匹配。

c. 点击 **Call** 开始打电话，如果是 MT，从 test box 里发起呼叫。

NOTE: 如果是双 SIM 卡设备，默认的 Subscription ID 是 0，如果需要设为 1，则勾选 **Dual SIM**, 将 **Subscripition ID** 设为 1，如 Figure 5-5 所示。

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

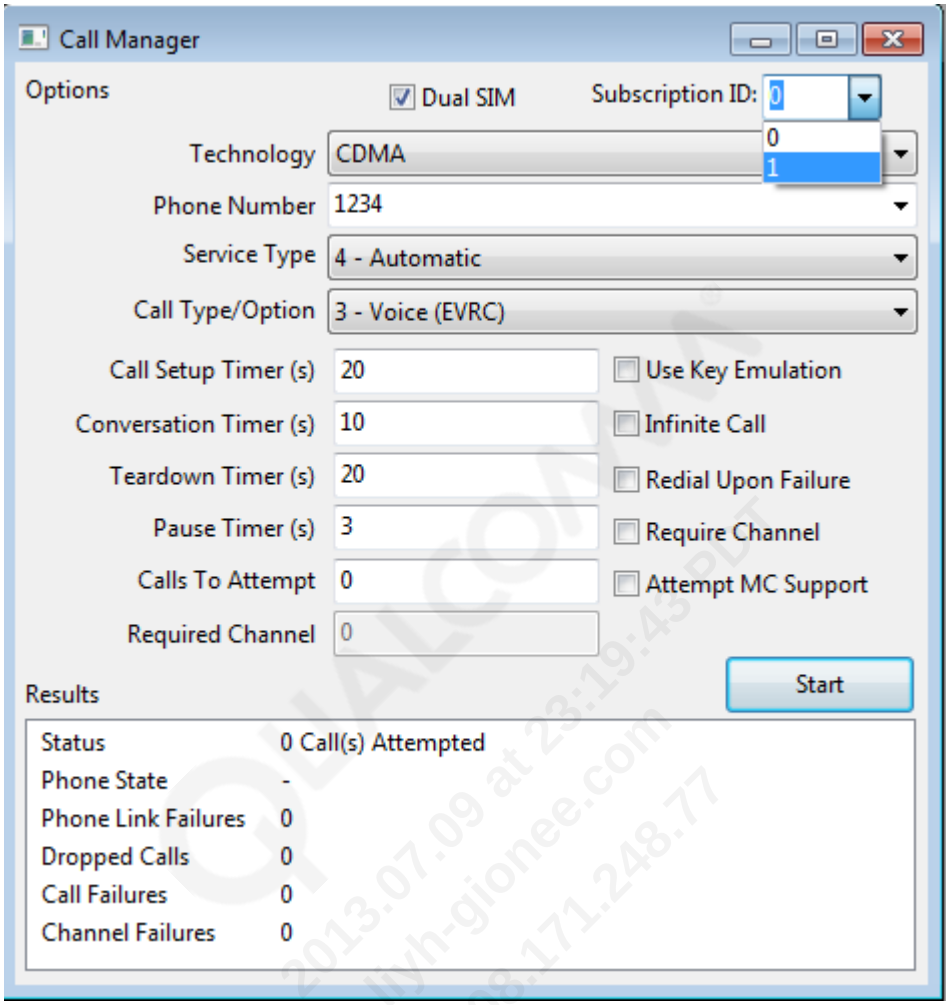

Figure 5-9 MSM8626 呼叫管理

5.3.2 1X 数据呼叫

预置条件：

- NV 设置 （将 NV # 10 设为 4，自动模式）
- QCN (确保在设备上进行 RF 校准，不要使用金 QCN，并且注意 1X 默认在 SV 链上。
- PRL (必须跟测试的频段/信道匹配，SID 和 NID 也必须匹配。)

Callbox 安装：

- 目前测试已经在 Agilent 8960 或者 Anritsu MT8820 上完成。
- 在 Call Setup/Call Params 界面：
  - 将 **Radio Config(RC)**设为(3,3) - (Fwd3, Rvs3)；
  - 将**(Fwd3, Rvs3)**的 FCH Service Option Setup 设为 **SO32 (TDSO)；**

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

- Callbox 安装指导–确保频段/信道和 SID/NID 设置于 PRL 里的设置匹配；小区功率值应该介于-45 ~ -65dBm 之间。

设备安装：

- 打电话的分步指导：
  a. 使用 QXDM Professional (QXDM Pro)的 Call Manager 界面
  b. 将电话号码设置为类似 1234 这样的数字，确保业务选项与 Callbox 设置匹配
  c. 点击 **Call** 开始打电话，如果是 MT，从 test box 里发起呼叫

5.3.3 HDR 呼叫

NV 设置：

- NV 设置 （将 NV # 10 设为 4，自动模式）。
- 对于 DO Rev A 呼叫，NV #4964 应该设为 Rev A 模式，callbox 也必须设为 Rev A 模式。比如，将 NV #4964 设为 NV_HDRSCP_REVA_PROTOCOLS_WITH_MFPA。

RF 校准：

- 确保设备完成 RF 校准。
- 漫游列表 – 包含 HDR 信道的 PRL 需要加载，并且 PRL 里的子网 ID 必须和 callbox 设置匹配。
- 按照下列步骤通过 QPST Service Programming 加载漫游列表：
  a. 选择 Roam 页签。
  b. 在 Preferred Roaming 区域选择 PRL 路径。
  c. 选择 Write to Phone。

Callbox 安装：

- 针对 DO RevA 呼叫，callbox 必须设置为 DO RevA 模式。

设备安装：

- 打电话的分步指导：
  a. 重启设备。
  b. 在 QXDM Pro 的 **Command Bar** 里输入 **mode online**，之后设备应该可以尝试获取 HDR 信道并协商会话。

5.3.4 GSM语音呼叫

预置条件：

- NV 设置
  a. 对于 GSM 制式的操作，将 NV #10(Mode Preference)设为 13。

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

b. 对于 GSM900 频段，将 NV #441(Band Class Preference)设为 **0x200**。

c. 对于多模 build，优先模式需要通过 UI 进行修改。否则，多模的 build 会使用 Android 里的默认设置 （缺省支持 1X）。并且会用此默认设置覆盖 NV 项，导致调制解调器无法进入 GSM 制式。

- QCN （确保已经在设备上进行了 RF 校准，不要使用金 GCN）

Callbox 安装：

- 目前测试已经在 Agilent 8960 或者 Anritsu MT8820 上完成。
- 将频段设置为 GSM900，小区功率值介于-45 ~ -65dBm 之间。
- 将信道模式设置为 TCH/F（全速率业务信道）

设备安装：

- 打电话的分步指导：

a. 使用 QXDM Professional (QXDM Pro)的 Call Manager 界面。

b. 将电话号码设置为类似 1234 这样的数字，确保业务选项与 Callbox 设置匹配。

c. 点击 **Call** 开始打电话，如果是 MT，从 test box 里发起呼叫 GPRS data call。

5.3.5 GPRS 数据呼叫

RF 校准：

确保设备已经进行 RF 校准

Callbox 安装 (Agilent 8960)：

- 选择 call setup 界面。
- Callbox setup
  - BCCH parameters→cell power = -75 dBm
  - BCCH parameters→cell band = EGSM
  - BCCH parameters→Broadcast chan = 20
  - PDTCH parameters→Multislot config = 1 Down 1 Up
  - Operating mode = Active mode GPRS
  - Data Conn = Type ETSI Type A

设备安装：

- 打电话的分步指导：

a. 插入 SIM 卡（测试确保 SIM 卡是好的）。

b. 开机；如果需要，在 QXDM Pro 的 Command Bar 里输入 mode online。

c. UE 应该驻留在 GPRS 小区并进行 ATTACH。

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

d. 发起测试模式 A 的数据呼叫，点击 Start Data Connection，屏幕底部会显示 TRANSFERRING。

5.3.6 WCDMA 语音呼叫

预置条件：

- QCN –确保设备已经校准。

Callbox 安装：

- Agilent 8960
  - Call box 安装
    - Call Control/Security Info/Security Parameters/Security Operations – None
    - Call Parms/Cell Power – -50.00 dBm
    - Call Parms/Channel Type – 12.2 KRMC
    - Call Parms/Paging Service – AMR Voice
  - 测试结果 – MO and MT passed 5/5
- Anritsu 8480C
  - Callbox 安装
    - Station Globals – Spec_Release – 3; HSDPA – FALSE; EUL – FALSE
- Anritsu 8820
  - Callbox 安装
    - Call Processing – On
    - Test Loop Mode – Off
    - Signal/Channel Coding – Voice

设备安装：

- 打电话的分步指导(主叫电话)：
  a. 启动固件和软件后，连上 USB，在 QXDM Pro 的 **Command Bar** 里输入 **mode online**）。
  b. 允许移动设备获取和注册网络。
  c. 通过 **QXDM Pro** 的 Call Manager 对话框发起呼叫。
  d. 将 Technology 设置为 WCDMA。
  e. 将电话号码设置为 **1234**。
  f. 勾选 infinite call。
  g. 发起呼叫。
  h. 电话状态先显示 Originating call，然后显示 Conversation。

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

5.3.7 WCDMA数据呼叫

预置条件：

- QCN – 确保设备已经校准。

Callbox：

- Agilent 8960
  - Callbox 安装。
    - Call Parms/Channel Type – HSPA
  - 测试结果– DUN 数据呼叫通过，吞吐量为~384 kbps， QMI 未尝试。
- Anritsu 8480C
  - Callbox 安装。
    - Station Globals – Spec_Release – 5; HSDPA – TRUE; EUL – FALSE
  - 测试结果– DUN 呼叫建立但是之后崩溃，QMI 未尝试。
- Anritsu 8820
  - Callbox 安装。
    - Call processing – On
    - Test Loop Mode – Mode 1
    - Signal/Chanel Coding – Fixed Reference Channel

设备安装：

- 打电话分步指导：

a. 允许 UE 注册网络。

b. 在拨号连接应用程序或者 QMICM 上点击 Connect/Dial 发起数据呼叫。

c. 运行 iPerf 或者 FTP 来测试吞吐量 （我们用 FTP 进行的测试）。

5.4 GPS配置

预置条件：

- 需要获取 GNSS SubSysGNSS DLL Ver 1.0.44 或者更高版本来进行线下 RF 开发。

操作步骤：

- 需要用 QPSR 进行线下 RF 开发。具体参见[Q13]。

5.5 多媒体配置

该节内容将在后续版本合入。

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

A Android Build 参考

A.1 Android设备树状结构

Android 设备的树状结构，比如<Android device tree root>, 展开如下所示：

- build/ – Build 环境建立和 makefiles 生成
- bionic/ – Android C 库
- dalvik/ – Android Java 虚拟机
- kernel/ – Linux 内核
- framework/ – Android 平台层 (系统库和 Java 部件)
- system/ – Android 系统 (应用程序和库, fastboot, logcat, liblog)
- external/ –Android 所需的非 Android 开源项目
- prebuilt/ – 构建 Android 之前编译好的二进制文件，比如，cross-compilers
- packages/ – 标准的 Android Java 应用和部件
- development/ – 开发所需 Android 参考程序和工具
- hardware/ – 硬件抽象层 (音频，感应器)和高通专属硬件包
- vendor/qcom/ – 高通特定目标设备定义，比如， e.g., msm7201a_surf
- vendor/qcom-proprietary – 高通所有的部件，例如， MM, QCRIL, 等等
- out/ – 用户创建的 built 文件
  - out/host/ – Android build 生成的主机可执行文件
  - out/target/product/<product> – 目标文件
    - appsboot*.mbn – 应用 boot loader
    - boot.img – Android 开机画面 (Linux 内核 + root FS)
    - system.img – Android 部件 (/system)
    - userdata.img – Android 开发应用程序和数据库
    - root/ – Root FS 文件夹,编译后成成 ramdisk.img 文件并合并到 boot.img 文件中。
    - system/ – 系统 FS 文件夹, 编译生成 system.img 文件。
    - obj/ – 中间对象文件。
      - 从部件编译的 include 文件

- lib/
- STATIC_LIBRARIES/
- SHARED_LIBRARIES/
- EXECUTABLES/
- APPS/

- symbols/ – 所有目标二进制文件符号

A.2 Android目标树结构

Android 目标树结构如下所示：

- / – 根文件夹 （ramdisk.img,只读）
  - init.rc – 初始配置文件 (设备配置，服务启动) init.qcom.rc
  - dev/ – 设备节点
  - proc/ – 进程信息
  - sys/ – 系统/内核配置
  - sbin/ – 系统启动二进制文件 (ADB 守护进程;只读）
  - system/ – 来自 system.img (读-写）
    - bin/ – Android 系统二进制文件
    - lib/ – Android 系统库
    - xbin/ – 非必须二进制文件
    - framework/ – Android 框架部件 (Java)
    - app/ – Android 应用(Java)
    - etc/ – Android 配置文件
  - sdcard/ – SD 卡挂载点
  - data/ – 来自 userdata.img (读-写)
    - app/ – 用户安装的 Android 应用
    - tombstones/ – Android 崩溃日志

A.3 构建小Android

小 Android，也叫 TINY_ANDROID, 是另一种 build，只产生一个超简 build 配置，用来进行单板初始化和非常底层的调试。 TINY_ANDROID 配置只包含 Android Linux 内核和由最小化的系统程序组成的系统根文件。

按照下面的命令来构建小 Android：

```
$ make BUILD_TINY_ANDROID=true -j4
```


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

A.4 手动构建Linux内核

按照下列方法手动进行 Linux 内核的构建。

1. 创建 Android build 环境。(envsetup.sh/choosecombo)。
2. 转到内核目录 (kernel/)。
3. 运行如下命令进行正确的内核配置：

   ```
   make ARCH=arm CROSS_COMPILE=arm-eabi- msm8974_defconfig
   ```

4. 运行如下命令构建内核图片：

   ```
   make -j3 ARCH=arm CROSS_COMPILE=arm-eabi- zImage
   ```

5. 如果需要，运行如下命令构建可选的内核模块：

   ```
   make -j3 ARCH=arm CROSS_COMPILE=arm-eabi- modules
   ```

   产生的内核图片会存在in kernel/arch/arm/boot/zImage文件夹里。

NOTE: 理论上，只要“n”小于 build 所在的服务器的处理器个数，就可以使用–jn。

6. 运行下面的命令从一个干净的 build 开始。
   a. 运行下列命令去除对象文件：

      ```
      make clean
      ```

   b. 运行下列命令去除所有产生的文件：

      ```
      make distclean
      ```

A.5 手动构建Android

按照下列步骤手动构建 Android：

1. 创建 Android build 环境。（envsetup.sh/choosecombo）。
2. 转到 Android 主目录。
3. 运行下列命令进行构建：

   ```
   make -j4
   ```

4. 创建独立的部件，有如下两种方法可以选择：
   - 运行下列命令从树冠运行 make

     ```
     m <component name>  # E.g. m libril-qc-1
     ```

   - 要构建当前目录下所有的模块，转到部件目录并运行下列命令

     ```
     mm
     ```

5. 要删除独立的部件对象文件，有如下两种方法可以选择：
   - 运行下列命令删除某个模块：

     ```
     m clean-<module name>
     ```

   - 运行下列命令删除某指定路径下的模块：

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

```
rm -rf out/target/product/*/obj/STATIC_LIBRARIES/
<module name>_intermediates

rm -rf out/target/product/*/obj/SHARED_LIBRARIES/
<module name>_intermediates

rm -rf out/target/product/*/obj/EXECUTABLES/
<module name>_intermediates
```

A.6 其它重要的Android build命令

其它重要的 Android build 命令如下：

- printconfig –按照 choosecombo 命令设置打印当前配置。
- 从树冠运行 make。这个命令可以让用户在子目录下运行 make。如果你有 TOP 环境变量集，该命令使用此变量集。如果你没有 TOP 环境变量集，该命令从当前目录向上寻找，试图找到树冠。
- - mm –构建当前目录下的所有模块。
- - mmm – 构建上层目录下的所有模块。
- croot – cd 到树冠。
- sgrep – grep 你在当前目录下所有 .c, .cpp, .h, .java,和 .xml 文件。
- clean-$(LOCAL_MODULE) and clean-$(LOCAL_PACKAGE_NAME)。
  - 让你选择性地清除一个目标。比如，你可以输入 make clean-libutils，则 libutils.so 和所有的中间文档都被清除了；或者你可以输入 make clean-Home，则只有 Home 应用被清除。
- 彻底清除该配置所有的输出和中间文件。与 rm -rf out/<configuration>/相同。

Android 的makefile(Android.mk)具有下列属性：

- 跟常规 GNU makefile 类似，但是具有下列差别：
  - 分给源文件的预先定义的变量，包括路径，编译器标志，和库文件。
  - 用来编译可执行文件，可分享库，静态库，Android 包，和预先编译二进制文件的预定义动作。
- 变量
  - LOCAL_SRC_FILES – 所有包含的源文件。
  - LOCAL_MODULE – 模块名称 （used for “m”）。
  - LOCAL_CFLAGS – 覆盖缺省的 C 编译器标志。
  - LOCAL_SHARED_LIBRARIES – 所要包含的共享库。
- 动作

- include $(CLEAR_VARS) – Clears LOCAL* variables for the following sections:
  - include $(BUILD_EXECUTABLE)
  - include $(BUILD_SHARED_LIBRARIES)
  - include $(BUILD_STATIC_LIBRARIES)

NOTE: Android.mk 里的路径通常与 Android 设备树根目录相关。

执行下列步骤增加一个新的模块到 Anaroid 源树上。

1. 创建一个新的目录存放新的模块源文件和 Android.mk 文件。
2. 在 Android.mk 文件里，用该文件产生的新的模块名来定义 LOCAL_MODULE 变量。

NOTE: 如果是应用模块，用 LOCAL_PACKAGE_NAME 文件。

新模块的本地路径为 LOCAL_PATH。该路径下有 Android.mk 文件。你可以插入下列命令到 Android.mk 文件的第一行来设置该路径。

```
LOCAL_PATH := $(call my-dir).
LOCAL_SRC_FILES
```

build 系统查看 LOCAL_SRC_FILES 寻找需要编译的源文件，判断是.cpp, .c, .y, .l,和/或者 .Java。如果是 lex 和 yacc 文件，.h 和 .c/.cpp 中间文件会自动生成。如果文件是 xx 文件所在的目录的子目录，那么文件名前缀要加上目录名。

```
LOCAL_SRC_FILES := \
   file1.cpp \
   dir/file2.cpp
```

下列命令可以用来配置新的模块：

- LOCAL_STATIC_LIBRARIES – 你需要放在模块里的静态库。

```
LOCAL_STATIC_LIBRARIES := \
   libutils \
   libtinyxml
```

- LOCAL_MODULE_PATH – 指导 build 系统将模块放在该类型非通常路径。如果你忽略这一点，确认你也设置了 LOCAL_UNSTRIPPED_PATH 如果这是个可执行文件或者是一个可分享库，这，unstripped 二进制文件也有存放的位置，否则会报错

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息

B SCons

SCons Ver 2.0.0 或者更高版本是用来构建所有非 HLOS 源代码 release 的软件构建工具。具体参见下面的网页。

- SCons – http://www.scons.org
  - 用户指南– http://www.scons.org/doc/HTML/scons-user/book1.html
  - 在线手册– http://www.scons.org/doc/HTML/scons-man.html
  - SCons 概览– http://www.humanized.com/presentations/scons

QUALCOMM® 2013.07.09 at 23:19:43 PDT liyh-gionee.com 108.171.248.77

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION 可能含有美国和国际出口受限信息