MSM8909 Linux Android Software User Manual

Software Product Document

SP80-NR964-4 C

December 17, 2014

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

QUALCOMM®
yangzheman@hipad.com

Confidential and Proprietary – Qualcomm Technologies, Inc.

NO PUBLIC DISCLOSURE PERMITTED: Please report postings of this document on public servers or websites to: 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.

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.

Qualcomm reserves the right to make changes to the product(s) or information contained herein without notice. No liability is assumed for any damages arising directly or indirectly by their use or application. The information provided in this document is provided on an "as is" basis.

This document contains confidential and proprietary information and must be shredded when discarded.

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.

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.

Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

© 2014 Qualcomm Technologies, Inc.
All rights reserved.

Contents

QUALCOMM®
wanzen...@...com

QUALCOMM®
yanzhennan@hipad.com

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Tables

Figures

QUALCOMM®
yanzhonnan@hipad.com

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Revision history

Revision	Date	Description
A	Oct 2014	Initial release
B	Nov 2014	▪ Added Sections 4.5 and 5.1 ▪ Updated Sections 2.4, 3.3.8, and 4.2 ▪ Updated Tables 2-1, 3-2, and 4-1
C	Dec 2014	▪ Updated Sections 3.2, 3.3.2, 3.3.3, 3.3.4, and 3.3.6 ▪ Updated Tables 2-1 and 3-2

QUALCOMM®
yanzhennan@hipad.com

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

1 Introduction

This document describes how to obtain, build, and program software applicable to the MSM8909 Linux Android Software Product (SP) “as-is” into a reference platform including:

- Setting up a development environment and installing the software
- Building the software and flashing it onto a reference platform
- Configuration and bringup of call, GPS, multimedia, etc.

1.1 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>`.

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

1.2 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		
Q2	*Hexagon Tools Installation Guide*	80-VB419-25
Q3	*Hexagon Development Tools Overview*	80-VB419-74
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	*IZat Gen 8 Engine Family RF Development Test Procedures*	80-VM522-2
Q15	*Qualcomm Createpoint User Guide*	80-NC193-2
Q16	*Qualcomm Flash Image Loader (QFIL) User Guide*	80-NN120-1
Q17	*Hexagon LLVM C/C++ Compiler User Guide*	80-VB419-89
Q18	*Presentation: LLVM Compiler for Hexagon Processor Deployment Plan*	80-VB419-87
Q20	*Presentation: MSM8x09 Chipset Software Overview*	80-NR964-2
Q24	*MSM8916+WTR1605 RF Calibration Software/XTT/DLL Revision Guide*	80-NK201-10
Q25	*DSI Programing Guide for B-Family Android Devices*	80-NA157-174
Q26	*Linux Android Display Driver Porting Guide*	80-NN766-1
Q27	*WCN36x0 WLAN/BT/FM FTM Guide with QRCT Test Example*	80-WL300-27

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Ref.	Document	
Q28	*QDART-MFG Installer*	80-NH711-3
Q29	*TPP User Guide*	80-NF136-1
Q30	*Application Note: MSM8916 External MI2S Interface Overview*	80-NL239-42
Q31	*Application Note: Sensor Driver Development and Troubleshooting*	80-NL239-32
Q32	*Linux Camera Debugging Guide*	80-NL239-33
Q33	*Application Note: Widevine DRM*	80-N9340-1
Q34	*Subsystem Restart User Guide*	80-N5609-2
Q35	*Presentation: MSM8x10 Android Subsystem Restart Overview*	80-NC839-21
Q36	*Presentation: MSM8974 Android Subsystem Restart*	80-NA157-31
Q37	*Application Note: SGLTE Device Configuration*	80-NJ017-11
Q38	*Application Note: Segment Loading Feature*	80-NL239-46
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/

QUALCOMM®
yanzhen.an@hipad.com

1.3 Debugging the software

Licensees can debug the entire software release by using [Q22].

1.4 Porting the software

For porting procedure, see [Q23].

2 Installation and Setup

2.1 Required equipment and software

Table 2-1 identifies the equipment and software needed to install and run the software.

NOTE: Due to a possibility of version change for the tools listed in the following table with software releases, refer to MSM8909 software release note for the exact version of the tools.

Table 2-1 Required equipment and software

	Item description	Version	Source/vendor	Purpose
1	Linux development workstation that exceeds minimum requirements for running Ubuntu 64-bit OS	—	—	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 12.0.4 LTS Linux distribution for 64-bit architecture	12.0.4 LTS	Ubuntu Community/ Canonical, Ltd.	Android build host OS
4	Java SE JDK for Linux x64	6 7 (for Android L)	Oracle	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.	Building boot images, RPM, and TrustZone (TZ)
7	Hexagon™ toolchain	6.4.04	Qualcomm/ GNU	Building Modem Processor Subsystem (MPSS)
8	Python	2.7.6	Python.org	Building boot, subsystem, etc. ▪ Boot – Ver 2.6.6 ▪ RPM – Ver 2.6.6 ▪ TZ – Ver 2.6.6 ▪ Meta – Ver 2.7.5 ▪ MPSS – Ver 2.7.6
9	SCons Ver 2.0.0 or higher		scons.org	Building all non-HLOS source code releases

QUALCOMM® wanzhennan@hmd.com

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

2.2 Installing Ubuntu

Log in as root or use sudo to have root permissions during the installation.

1. Create an installation CD and install it onto the computer by following the instructions at http://releases.ubuntu.com.
2. After installation, perform a software update using one of the following options:
   - Using the GUI, select System→Administration→Update Manager

     or
   - Using the shell command line
     1. Edit the source config file directly, as follows:

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

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

        ```
        sudo apt-get update

        sudo apt-get upgrade
        ```

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

   ```
   $ 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. IMPORTANT! Make bash the default shell (Android build scripts contain bash shell dependencies that require the system default shell /bin/sh to invoke bash) using one of the following options:
   - Reconfigure the package:
     1. Use the command:

        ```
        sudo dpkg-reconfigure dash
        ```

     2. 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.

QUALCOMM® wangchenman@hipad.com

2.3 Configuring Samba for Windows sharing (optional)

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
   ```

2.4 Installing JDK

The Sun JDK is no longer in the main package repository of Ubuntu. To download it, add the appropriate repository and indicate to the system about the use of JDK.

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

If Android L version is used, install the JDK 7:

```
# get package openjdk-7-jdk
sudo apt-get install openjdk-7-jdk
```

2.5 Installing Repo

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

To install repo:

1. Create a ~/bin directory in 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
   ```

QUALCOMM® wenzhenna@hipad.com

4. Include the installed directory location for repo in the PATH.

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

5. Run **repo --help** to verify installation; a message similar to the following should appear:

```
$ 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
```

NOTE: For access to the full online help, install repo (repo init).

2.6 Installing the ARM Compiler Tools

Building the non-HLOS images requires the specific version of the ARM Compiler Tools indicated in Section 2.1. Linux is the recommended build environment for building all software images; however, either Windows or Linux-hosted versions 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.

QUALCOMM
anzhenna@hipad.com

Installing on a Linux host

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

Installing on a Windows host

1. Obtain the required ARM toolchain from the ARM vendor.
2. Follow the vendor instructions to install the toolchain and flex license manager onto the Windows build system.
3. Access the software from https://silver.arm.com/download/download.tm?pv=1245960.
4. The default install location is C:\Program Files (x86)\ARM_Compiler5\.
5. If necessary, change the directory where the files are extracted to match the location where the tools are installed. For example, the installing directory for QTI is C:\Program Files (x86)\ARM_Compiler5\bin.
6. 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.

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

7. To check the versions, run **armcc –vsn**. It must return the following:

   ```
   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 must be `Build 94` for all.

2.7 Installing the Hexagon toolchain

Linux is the recommended build environment for building all software images; however, either Windows or Linux-hosted versions work for building the non-HLOS images.

See [Q2] for detailed procedures to download and install the Hexagon toolchain software. See [Q3] for more documentation on using the Hexagon tools.

See [Q17] and [Q18] for a detailed explanation of the LLVM compiler in the Hexagon toolchain. LLVM compilers work with Hexagon software development tools and utilities to provide a complete programming system for developing high-performance software.

QUALCOMM®
yanzhennan@hotmail.com

3 Downloading and Building the Software

Table 3-1 describes the software for this product line divided into the release packages that need to be downloaded separately and combined to have a complete product line software set.

Table 3-1 Release packages

From chipcode.qti.qualcomm.com [R6]	From codeaurora.org [R4]
Proprietary non-HLOS software Contains proprietary source and firmware images for all non-Apps processors. This software is an umbrella package built from a combined set of individual component releases that are already integrated.	**Open source HLOS software** Contains open source for Apps processor HLOS
Proprietary HLOS software Contains proprietary source and firmware images for the apps processor HLOS.	–

QUALCOMM wangzhiming@hipad.com

The proprietary and open source HLOS packages must be obtained from separate sources and then combined according to the downloading instructions given in Section 3.3.8. Each package is identified by a unique build identification (build ID) code, which follows this naming convention:

<PL_Image>-<Version>-<Chipset>

Where:

- PL_Image – LA.BR64.Branch for Linux Android
- Version – Variable number of digits used to represent the build ID version.
- Chipset – 8909 for MSM8909

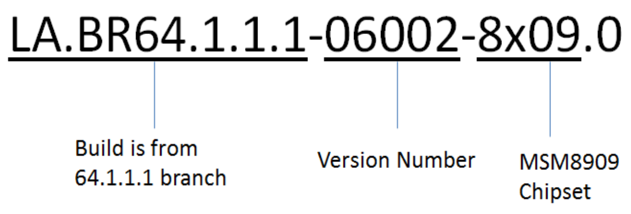
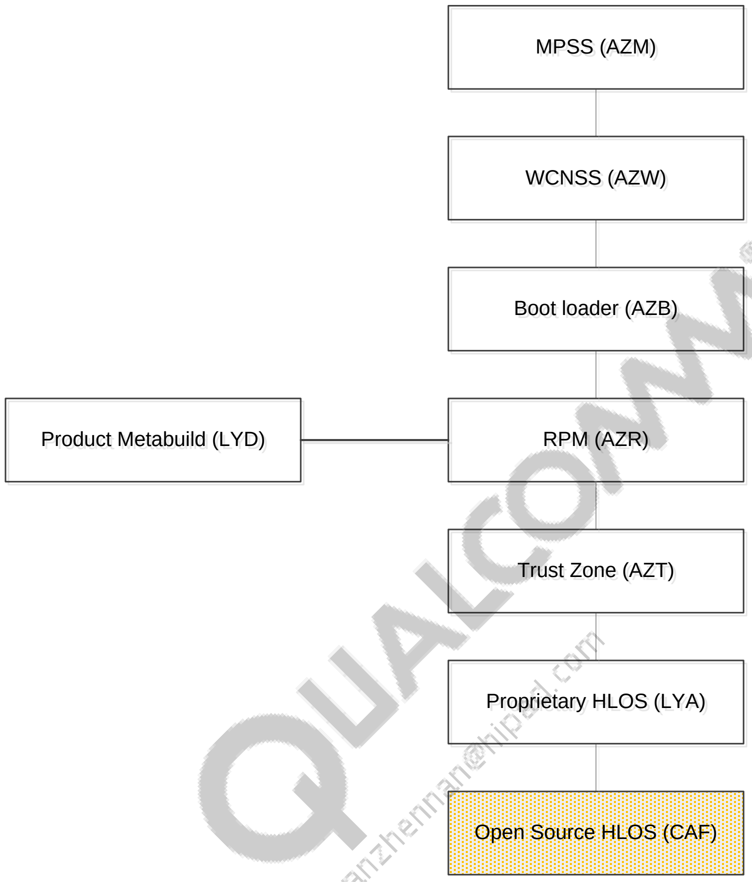


Figure 3-1 Build ID naming convention

Figure 3-2 shows the combined software release packages.



QUALCOMM®
yanzhennan@hipad.com

Figure 3-2 Combined software release packages

Table 3-2 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. Ensure that the build PC has the correct versions for each tool.

NOTE: Due to a possibility of version change for the tools listed in the following table with software releases, refer to MSM8909 software release note for the exact version of the tools.

Table 3-2 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 6.4.04	Python 2.7.6 (64-bit)	Perl 5.14.2	Windows builds only; Needs tee.exe	Linux, Windows XP, and Windows 7

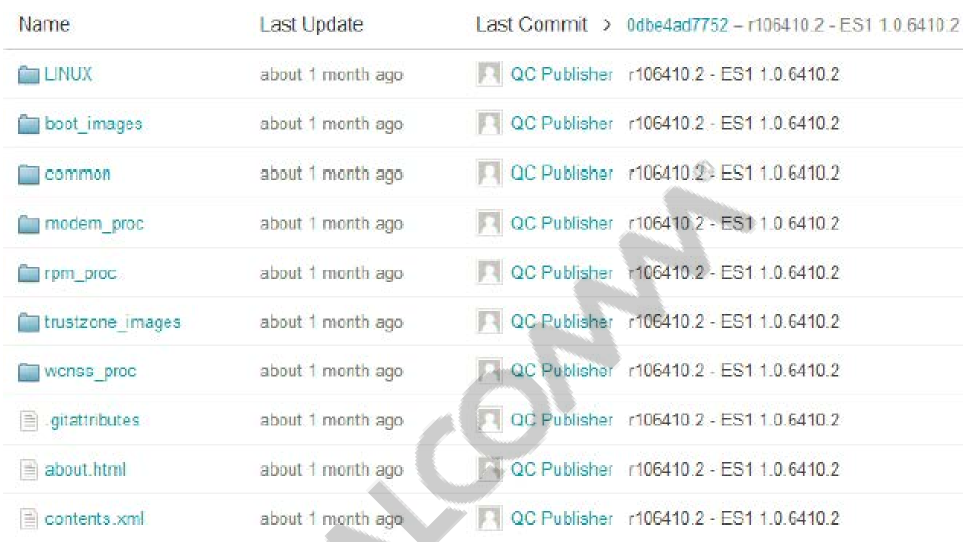
Component build release	**Source or binary only**	**Toolchain required for building source**	**Python version**	**Perl version**	**Cygwin**	**Supported build hosts**
Boot loaders (AZB)	Source	ARM Compiler Tools 5.01 update 3 (build 94)	Python 2.6.6	Perl 5.8.x Linux builds only	Windows builds only; Needs tee.exe	Linux, Windows XP, and Windows 7
RPM (AZR)	Source	ARM Compiler Tools 5.01 update 3 (build 94) RPM - (ARM compiler Tools Q4 Full version)	Python 2.6.6	Perl 5.6.1	Windows builds only; Needs tee.exe	Linux, Windows XP and Windows 7 only
TZ (AZT)	Source	ARM Compiler Tools 5.01 update 3 (build 94)	Python 2.6.6	Perl 5.10.1	Windows builds only; Needs tee.exe	Linux, Windows XP, and Windows 7 only
WCNSS (AZW)	Binary	—	—	—	—	—

yanzhennan@hipad.com

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

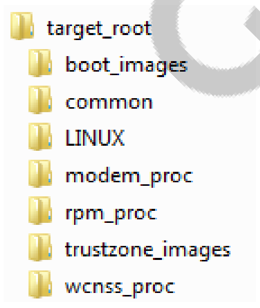
3.1 Downloading QTI software from ChipCode

1. The software distribution package (distro) is composed of multiple subsystem image files. Download the distro components to the build PC.



Name	Last Update	Last Commit > 0dbe4ad7752 – r106410.2 - ES1 1.0.6410.2	
LINUX	about 1 month ago	QC Publisher	r106410.2 - ES1 1.0.6410.2
boot_images	about 1 month ago	QC Publisher	r106410.2 - ES1 1.0.6410.2
common	about 1 month ago	QC Publisher	r106410.2 - ES1 1.0.6410.2
modem_proc	about 1 month ago	QC Publisher	r106410.2 - ES1 1.0.6410.2
rpm_proc	about 1 month ago	QC Publisher	r106410.2 - ES1 1.0.6410.2
trustzone_images	about 1 month ago	QC Publisher	r106410.2 - ES1 1.0.6410.2
wcnss_proc	about 1 month ago	QC Publisher	r106410.2 - ES1 1.0.6410.2
.gitattributes	about 1 month ago	QC Publisher	r106410.2 - ES1 1.0.6410.2
about.html	about 1 month ago	QC Publisher	r106410.2 - ES1 1.0.6410.2
contents.xml	about 1 month ago	QC Publisher	r106410.2 - ES1 1.0.6410.2

2. Create a top-level directory on the build PC and unzip each of the subsystem images to generate the following directory structure. In this example, <target_root> is the top-level directory.



```
target_root
  boot_images
  common
  LINUX
  modem_proc
  rpm_proc
  trustzone_images
  wcnss_proc
```

contents.xml

about.html

3. Ensure to download the `about.html` and `contents.xml` files. These files contain the build ID that is used to download the open source software from CAF.

QUALCOMM
yanzhennan@hipad.com

3.2 Downloading open source Android HLOS software from CAF

1. See the release notes for the link to access the open source software from CAF.

   Alternatively, go to https://www.codeaurora.org/xwiki/bin/QAEP/release and find the release branch containing the matching build ID in the branch releases table.

2. In an empty directory, use 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
<build_id>.xml --repo-url=git://codeaurora.org/tools/repo.git --repo-
branch=caf-stable
```

3. Type the repo sync command.

```
$ repo sync
```

4. Copy the vendor/qcom/proprietary directory tree into the open source HLOS source tree contained in the workspace.

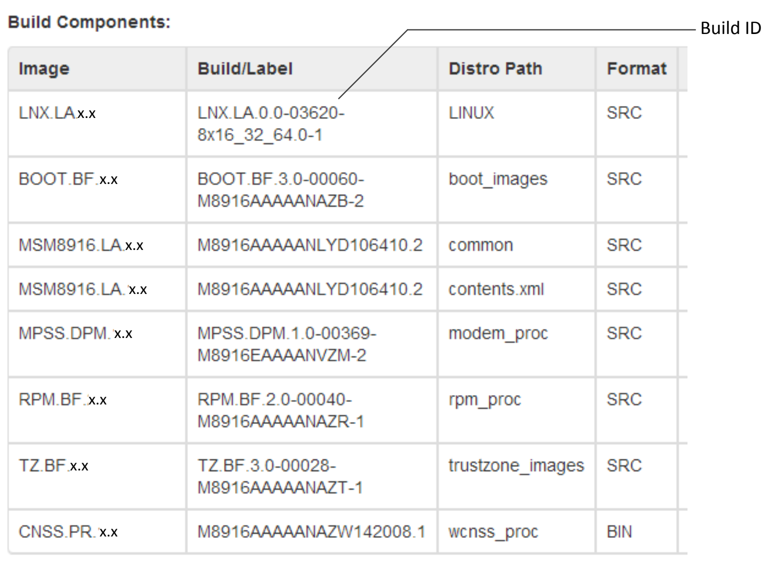
```
$cp -r <LYA_build_location>/HY11-<build_id>/LINUX/android/* .
```

QUALCOMM

Finding the Linux build ID for other releases

To obtain the Linux build ID for another release for which there is no release notes, the build ID can be obtained in the following ways:

- From ChipCode, on the distro page, the about.html section displays the build ID.



Build Components:

Image	Build/Label	Distro Path	Format
LNX.LA.x.x	LNX.LA.0.0-03620-8x16_32_64.0-1	LINUX	SRC
BOOT.BF.x.x	BOOT.BF.3.0-00060-M8916AAAAANAZB-2	boot_images	SRC
MSM8916.LA.x.x	M8916AAAAANLYD106410.2	common	SRC
MSM8916.LA. x.x	M8916AAAAANLYD106410.2	contents.xml	SRC
MPSS.DPM. x.x	MPSS.DPM.1.0-00369-M8916EAAAANVZM-2	modem_proc	SRC
RPM.BF.x.x	RPM.BF.2.0-00040-M8916AAAAANAZR-1	rpm_proc	SRC
TZ.BF.x.x	TZ.BF.3.0-00028-M8916AAAAANAZT-1	trustzone_images	SRC
CNSS.PR. x.x	M8916AAAAANAZW142008.1	wcnss_proc	BIN

Figure 3-3 Build ID information from about.html

- From the `about.html` file that was downloaded to the build PC.
- Search for `<name>apps</name>`, then locate the `build_id`, from the contents.xml file that was downloaded to the build PC.

```
<build>
<name>apps</name>
<role>apps</role>
<chipset>msm8909</chipset>
<build_id>LA.64.1.1.1-06002-8x09.2-1</build_id>
```

3.3 Compiling the Non-HLOS software

3.3.1 Setting build Windows environment

Before issuing the non-HLOS build commands, certain command environment settings must be set to ensure the correct executable path and toolchain configuration. The specific environment settings vary based on the host software installation, but it is similar to the example “myenviron_amss.cmd” script shown below (for Windows), which sets the path to point to the ARM toolchain lib, include, bin, and license file configuration.

```
#
# myenviron_amss_8916
#
SET ARMLMD_LICENSE_FILE=<mylicense_file>@<mylicense_server>
set ARM_COMPILER_PATH=C:\apps\ARMCT5.01\94\bin64
set PYTHON_PATH=C:\Python26
set PYTHONPATH=C:\Python26
set MAKE_PATH=C:\apps\ARMCT5.01\94\bin64
set GNUPATH=C:\cygwin\bin
set CRMPERL=C:\Perl64\bin
set PERLPATH=C:\Perl64\bin

set ARMHOME=C:\Apps\ARMCT5.01\94
set ARMINC=C:\Apps\ARMCT5.01\94\include
set ARMLIB=C:\Apps\ARMCT5.01\94\lib
set ARMBIN=C:\Apps\ARMCT5.01\94\bin
set ARMPATH=C:\Apps\ARMCT5.01\94\bin
set ARMINCLUDE=C:\Apps\ARMCT5.01\94\include
set ARMTOOLS=ARMCT5.01
set
PATH=.;C:\Python26;C:\Apps\ARMCT5.01\94\bin;C:\apps\ARMCT5.01\94\bin64;C:\c
ygwin\bin;%PATH%
set HEXAGON_ROOT=C:\Qualcomm\HEXAGON_Tools
set HEXAGON_RTOS_RELEASE=5.0.07
set HEXAGON_Q6VERSION=v4
```

QUALCOMM® zhennan@hipad.com

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

```
set HEXAGON_IMAGE_ENTRY=0x08400000
```

3.3.2 Building MPSS

To build MPSS (<target_root> is the top-level directory that is created in Section 3.1):

1. For Linux, verify that the paths shown below are set. Refer to setenv.sh in the boot build.

```
<target_root>\modem_proc\build\ms\setenv.sh
export ARMLMD_LICENSE_FILE=<LICENSE FILE INFO>
ARM_COMPILER_PATH=/pkg/qct/software/arm/RVDS/2.2BLD593/RVCT/Programs/2.2
/593/linux-pentium
PYTHON_PATH=/pkg/qct/software/python/2.6.6/bin
MAKE_PATH=/pkg/gnu/make/3.81/bin
export ARMTOOLS=RVCT221
export ARMROOT=/pkg/qct/software/arm/RVDS/2.2BLD593
export ARMLIB=$ARMROOT/RVCT/Data/2.2/349/lib
export ARMINCLUDE=$ARMROOT/RVCT/Data/2.2/349/include/unix
export ARMINC=$ARMINCLUDE
export ARMCONF=$ARMROOT/RVCT/Programs/2.2/593/linux-pentium
export ARMDLL=$ARMROOT/RVCT/Programs/2.2/593/linux-pentium
export ARMBIN=$ARMROOT/RVCT/Programs/2.2/593/linux-pentium
export PATH=$MAKE_PATH:$PYTHON_PATH:$ARM_COMPILER_PATH:$PATH
export ARMHOME=$ARMROOT
export HEXAGON_ROOT=/pkg/qct/software/hexagon/releases/tools
```

2. Navigate to the following directory:

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

QUALCOMM yanzhinnanehijaca.com

3. Depending on the build environment, use one of the following commands:

Build environment	MSM8909 1.0 PL
Linux	**Build images** – ./build.sh 8909.gen.prod -k **Clean the build –**./build.sh 8909.gen.prod -c **Notices:** For CMCC – EAAAANWZM: ./build.sh 8909.lwg.prod -k For CU – EAAAANWZM: ./build.sh 8909.lwg.prod -k For CT – EAAAANCZM: ./build.sh 8909.lcg.prod -k For open market to have all the RATs support: EAAAANVZ – ./build.sh 8909.gen.prod -k 1X compiled out with segment loading enabled: EAAAANWZM: ./build.sh 8909.lwg.prod -k All RATs/kitchen sink segment loading enabled: EAAAANVZ: ./build.sh 8909.gen.prod -k
Windows	**Build images** – build.cmd 8909.gen.prod -k **Clean the build** – build.cmd 8909.gen.prod -c

QUALCOMM®
yanzhennan@zipod.com

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

3.3.3 Building boot loaders

To build the boot loaders:

1. For Linux, verify that the paths shown below are set. Refer to setenv.sh in the 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=$ARMROOT/lib
   export ARMINCLUDE=$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. Navigate to the following directory:

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

   Where <target_root> is the top-level directory that is created in Section 3.1.

QUALCOMM vanchennan@hipad.com

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

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

Build environment	MSM8909 1.0 PL

Linux	Build boot images
	`$ ./build.sh  TARGET_FAMILY=8909 --prod`
	Cleaning the build
	`$ ./build.sh TARGET_FAMILY=8909 –prod -c`
	
	Depending on the configuration, edit the script boot_images/build/ms/build_8909.sh to change
	if [-e “setenv.sh”]; then
	- source setenv.sh
	+ source ./setenv.sh
	fi

Windows	Building boot images build.cmd TARGET_FAMILY=8909 --prod Cleaning the build build.cmd TARGET_FAMILY=8909 –prod -c

NOTE: Currently NAND compilation is not enabled.

QUALCOMM® yanzhennuo@hipad.com

3.3.4 Building TrustZone images

To build the MSM8909 TrustZone images:

1. Navigate to the following directory:

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

2. Run the following command to build all images:

Build environment	MSM8909 1.0 PL
Linux	**Build images** – ./build.sh CHIPSET=msm8909 tz sampleapp tzbsp_no_xpu playready widevine isdbtmm aostlm securitytest keymaster commonlib **Clean the build** – ./build.sh CHIPSET=msm8909 tz sampleapp tzbsp_no_xpu playready widevine isdbtmm aostlm securitytest keymaster commonlib -c
Windows	**Build images** – build.cmd CHIPSET=msm8909 tz sampleapp tzbsp_no_xpu playready widevine isdbtmm aostlm securitytest keymaster commonlib **Clean the build** – build.cmd CHIPSET=msm8909 tz sampleapp tzbsp_no_xpu playready widevine isdbtmm aostlm securitytest keymaster commonlib -c

3.3.5 Building RPM

Use the following commands to build RPM (<target_root> is the top-level directory). Ensure that the tools used are of versions specified in Table 3-2.

NOTE: Building on 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 the build environment, use one of the following commands:

Build environment	MSM8909 1.0 PL
Linux	**Build images** – ./build_8909.sh **Clean the build** – ./build_8909.sh -c
Windows	**Build images** – build_8909.bat **Clean the build** – build_8909.bat -c

NOTE: rpm.mbn are found at rpm_proc\build\ms\bin\8909

3.3.6 Building WCNSS

NOTE: Since WCNSS is shipped as a binary, WCNSS compilation is not required. However, OEMs need to pick this binary from the Qualcomm chipcode release.

3.3.7 Updating NON-HLOS.bin

If any 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 Section 3.1):

1. Navigate to the following directory:

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

2. Enter the command:

```
python update_common_info.py
```

NOTE: If you are creating the whole sparse_images (which is required when contents.xml is burned to be a whole image using Qualcomm Product Support Tool (QPST)), the image compiled from Linux/Android must be copied into the LINUX/android/out/target/product/MSM8909/ directory. For details about the required files, refer to apps-related configuration information in the contents.xml file. In Linux, you are advised to copy Linux/android code downloaded from

QUALCOMM® wuzhennan@hipad.com

codeaurora.org to the LINUX/android directory downloaded from Qualcomm ChipCode for compilation.

3.3.8 Building 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: Use the source command so that the environment settings are defined in the current shell.

QUALCOMM®
yanzhennan@hipad.com

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

3. Enter the lunch command to select the build configuration, or enter with no parameters to see an interactive menu for making selections.
   a. For MSM8909 1.0 PL:
      ```
      lunch msm8909-userdebug
      ```
   b. For 512 MB DDR memory:
      ```
      lunch msm8909_512-userdebug
      ```
4. Run `make` to start the build. To run parallel builds for faster build times on a multicore build machine, run the following command:

   ```
   make –j4
   ```

QUALCOMM®
yanzhennan@hipad.com

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4 Firmware Programming

4.1 Required software

Table 4-1 lists the software required for programming firmware images into a target device.

Table 4-1 Required software

	Item description	Version	Source/vendor	Purpose
1	QPST	2.7.421 or later	Qualcomm Technologies, Inc.	Programming firmware images using QPST
2	QXDM Professional™	3.14.827 or later	Qualcomm Technologies, Inc.	Programming NV Item values, reading diagnostic, etc.
3	Lauterbach T32 ARMv7 license extension	LA-7843X or LA-7843	Lauterbach GmbH	Programming firmware images using JTAG and applications processor debugging
4	Lauterbach T32 Hexagon license extension	LA-3741A	Lauterbach GmbH	Modem software processor, firmware processor, ADSP debugging using JTAG
5	Lauterbach T32 Cortex-M3 license extension	LA-7844X or LA-7844	Lauterbach GmbH	Programming firmware images using JTAG and RPM debugging using JTAG
6	Lauterbach T32 ARM9™ license extension	LA-7742X or LA-7742	Lauterbach GmbH	Venus and WCNSS debugging using JTAG
7	Lauterbach T32 Windows	Feb 2014 R.2014.02.000053364 Build: 51144--53364	Lauterbach GmbH	Programming firmware images and debugging using JTAG

QUALCOMM® yanzhennan@hinad.com

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

	Item description	Version	Source/vendor	Purpose
8	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
9	Qualcomm USB network driver combo	1.00.29 or later	Qualcomm Technologies, Inc.	Windows host USB drivers for Qualcomm composite devices

4.2 Installing T32

QPST is used for the firmware download; however, T32 can also be used when the QPST download does not work. The February 2014 Build 51144--53364 version of T32 is the mandatory minimum revision that is needed for binary download and debugging. The T32 links under common\t32\t32_dap\ are used for binary download and debugging.

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

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.

4.3 Installing Android ADB, Fastboot, and USB driver for 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®
yanzhenman@hipad.com

- Qualcomm Composite USB network combo driver (qcnet.inf)
  - Qualcomm wireless HS-USB Ethernet adapter

Installing Android ADB, Fastboot, and USB driver for Windows

1. Before installing the drivers, edit the qcmdm.inf and qcser.inf files. This ensures that they contain support for the Android SURF VID/PID with appropriate entries in each section as indicated in Step 4.
2. Go to http://developer.android.com/sdk/win-usb.html, and follow the instructions to install the SDK and USB driver.
3. Right-click **My Computer,** and select **Properties → Advanced → Environment Variables**, and set the path to include the c:\android-sdk-windows\tools directory.
4. The Android USB driver for ADB and Fastboot must add the Qualcomm SURF VID/PID, which supports the connection to the 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, ensure 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.

QUALCOMM® yanzhenhan@hipad.com

7. In the %USERPROFILE%\.android directory, create /edit the adb_usb.ini file. If the file exists, it contains 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 the Android update ADB as it resets the contents of this file and overwrites the line just added.

After editing, the adb_usb.ini file must look like the following:

```
# 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
```

QUALCOMM®
anzhenhan@ipad.com

```
%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
[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”
```

QUALCOMM wanzhennan@hipad.com

4.4 Installing adb and fastboot in Linux

1. Before installing adb, modify the USB driver by navigating to the following directory:

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

2. Enter the command:

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

   The result must 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, to see the list of target devices connected to the Linux box, type:

   ```
   Lsusb
   ```

4. The adb and fastboot executables for Linux are located in the android\out\host\linux-x86\bin directory in the Android software release after a build is complete. If the android\out\host\linux-x86\bin directory is not in the executable search path, use the following steps to add it. If it is already in the executable search path, skip to Step 5:

   a. Type the command:

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

   b. Type the command:

      ```
      choosecombo 1 msm8909 userdebug
      ```

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

   ```
   sudo fastboot devices
   ```

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

NOTE: To run adb or fastboot, sudo or root access on the Linux machine may be required.

QUALCOMM®
vazizieman@hipad.com

4.5 Programming procedures

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

4.5.1 Programming eMMC with QPST

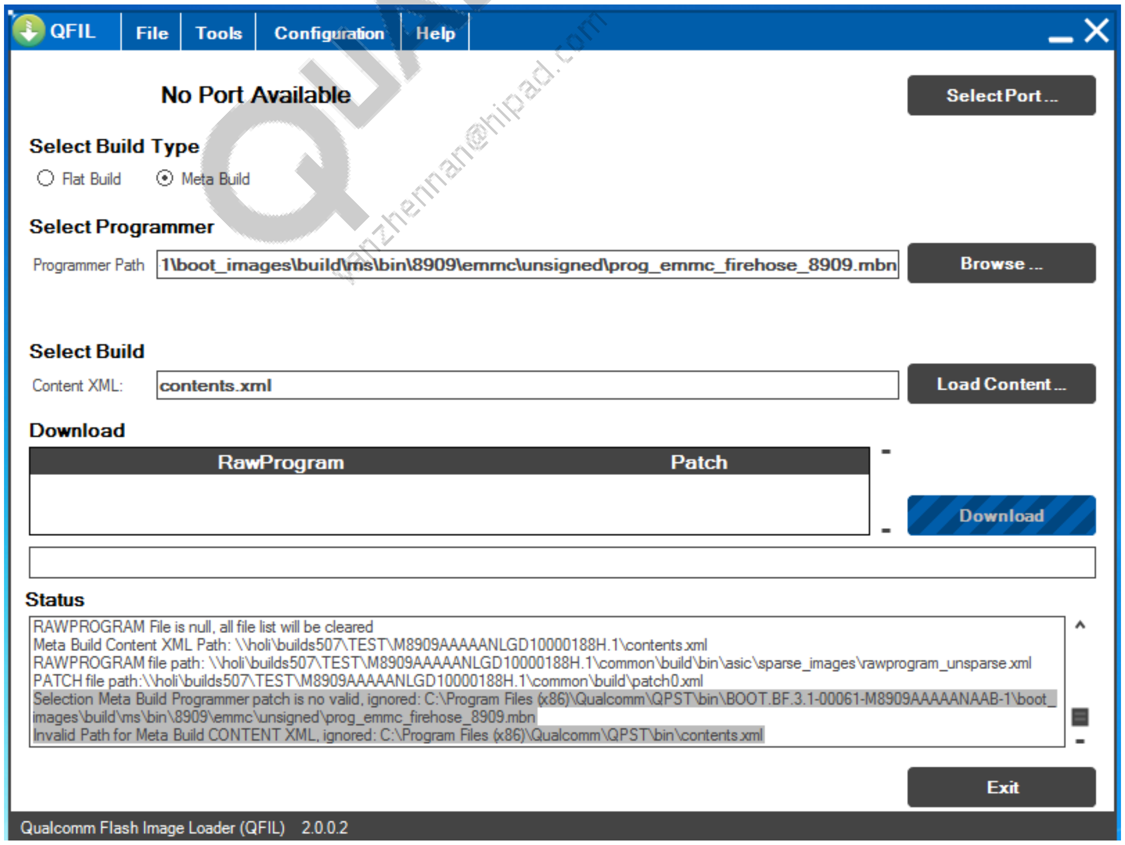
If no image is flashed to eMMC (this is the situation for the first-time binary download in the factory line), the PBL enumerates USB, as the Qualcomm Sahara download interface waits for the download command from the host PC. In this case, the minimum initial binary is needed to be flashed so that the rest of the Android images can be downloaded.

NOTE: On the MSM8909 CDP, you can force the device into the binary download mode by erasing the device entirely or using the dip switch S7 pin 3.

NOTE: The QPST version must meet the minimum requirement specified in Section 4.1.

1. Launch QFIL from QPST. If USB 9091 port is displayed in the Windows Device Manager, it is automatically detected.

QUALCOMM
wanzhennan@hipad.com



Select Build Type: ○ Flat Build [✓ Meta Build]

Programmer Path: 1\boot_images\build\ms\bin\8909\emmc\unsigned\prog_emmc_firehose_8909.mbn

Content XML: contents.xml

```
RAWPROGRAM File is null, all file list will be cleared
Meta Build Content XML Path: \\holi\builds507\TEST\M8909AAAAANLGD10000188H.1\contents.xml
RAWPROGRAM file path: \\holi\builds507\TEST\M8909AAAAANLGD10000188H.1\common\build\bin\asic\sparse_images\rawprogram_unsparse.xml
PATCH file path:\\holi\builds507\TEST\M8909AAAAANLGD10000188H.1\common\build\patch0.xml
Selection Meta Build Programmer patch is no valid, ignored: C:\Program Files (x86)\Qualcomm\QPST\bin\BOOT.BF.3.1-00061-M8909AAAAANAAB-1\boot_
images\build\ms\bin\8909\emmc\unsigned\prog_emmc_firehose_8909.mbn
Invalid Path for Meta Build CONTENT XML, ignored: C:\Program Files (x86)\Qualcomm\QPST\bin\contents.xml
```

2. Select the contents.xml from the root directory of the build, it fills the Programmer Path field is populated automatically.
3. Click Download. The image is downloaded to the device and the eMMC is programmed.

4.5.2 Programming CDT using QPST/QFIL

To program CDT using the QFIL tool, ensure that you have the following:

- Device programmer – `prog_emmc_firehose_8909.mbn`
- CDT xmls
  - Example: 8909\emmc_cdt_program_script\cdt_prog_xml\rawprogram2_cdp.xml
  - 8909\emmc_cdt_program_script\cdt_prog_xml\patch2.xml
- CDT binary corresponding to the CDT that is programmed (present in the rawprogram2_cdp.xml)
  - Example: 8909\emmc_cdt_program_script\cdt_bin\cdp_1.0_platform_jedec_lpddr2_single_channel .bin

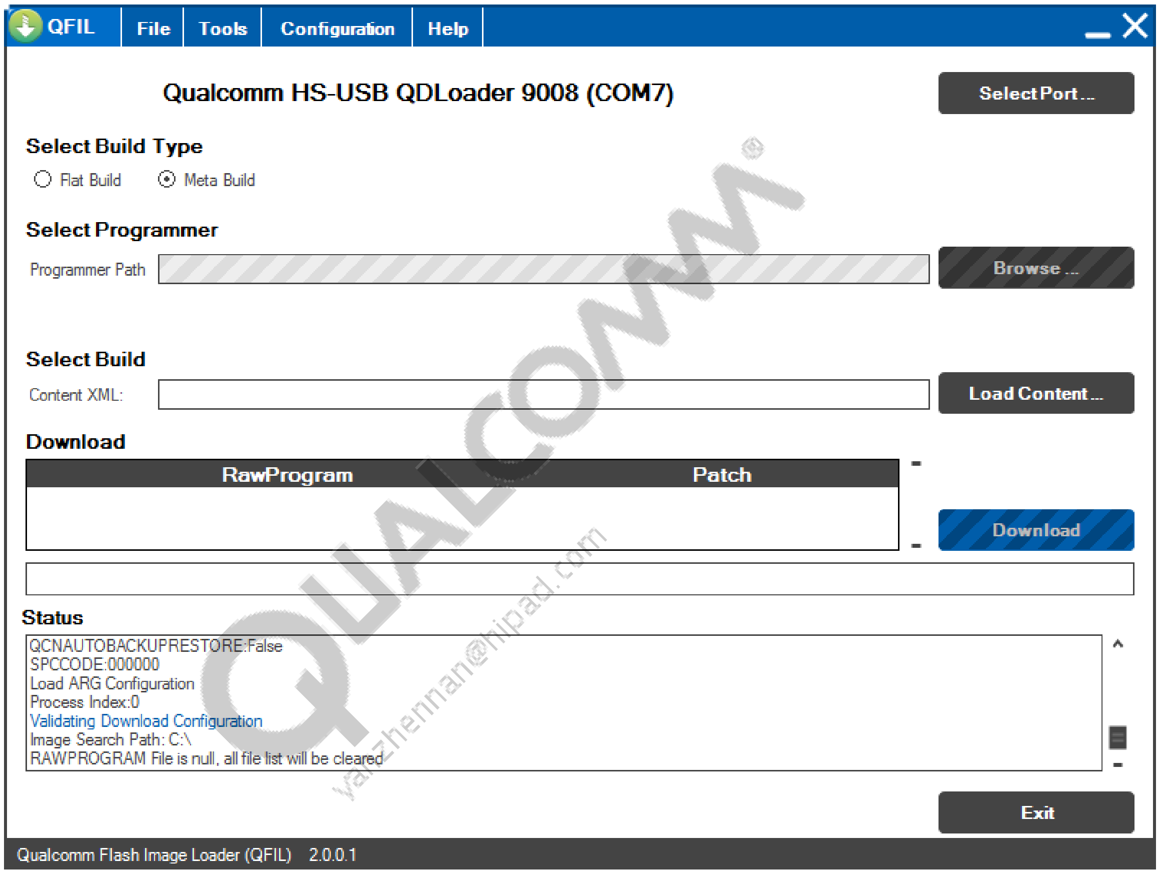
1. Copy the above into one folder
2. Launch the QFIL application.
3. Select the programmer from the Boot build location similar to 4.5.1.
4. Select the CDT xmls (rawprogram2_cdp.xml and the patch2.xml) from the folder in which it has been copied.
5. Select download to program the CDT.

QUALCOMM® vanzhenneo@hmad.com

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Flashing the CDT file

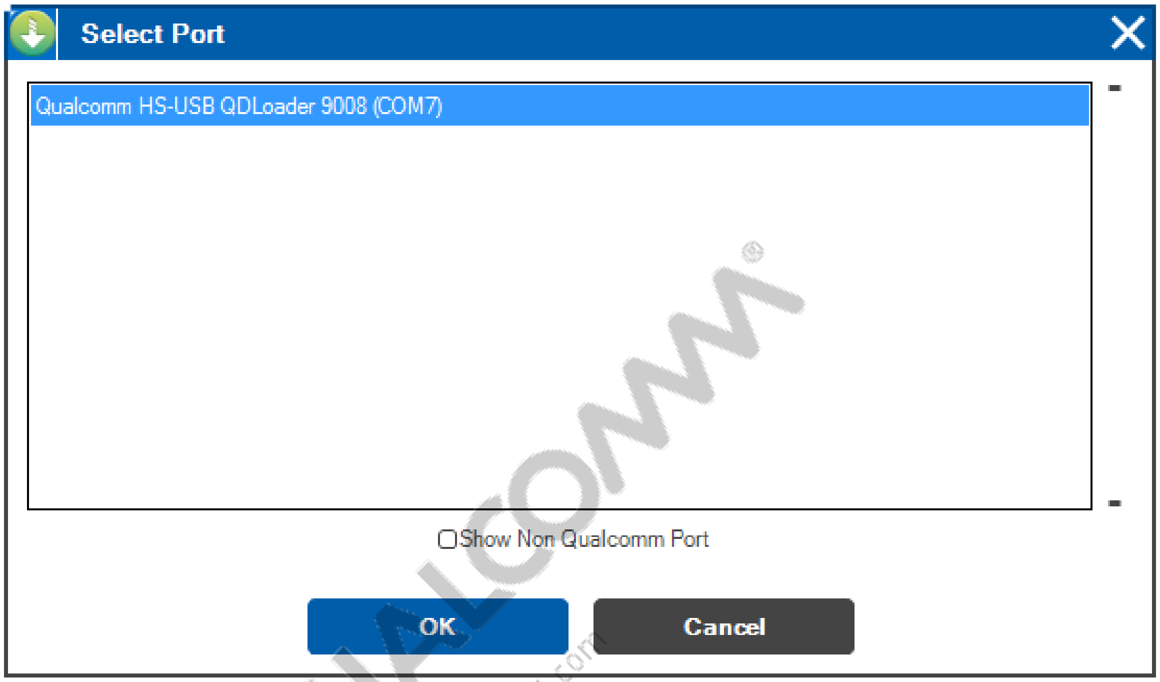
1. Launch QFIL from QPST. If USB 9008 port is displayed in the Windows Device Manager, it is automatically detected.



```
QCNAUTOBACKUPRESTORE:False
SPCCODE:000000
Load ARG Configuration
Process Index:0
Validating Download Configuration
Image Search Path: C:\
RAWPROGRAM File is null, all file list will be cleared
```

QUALCOMM
yanzhennan@hipad.com

2. Click OK. The 9008 port window appears.



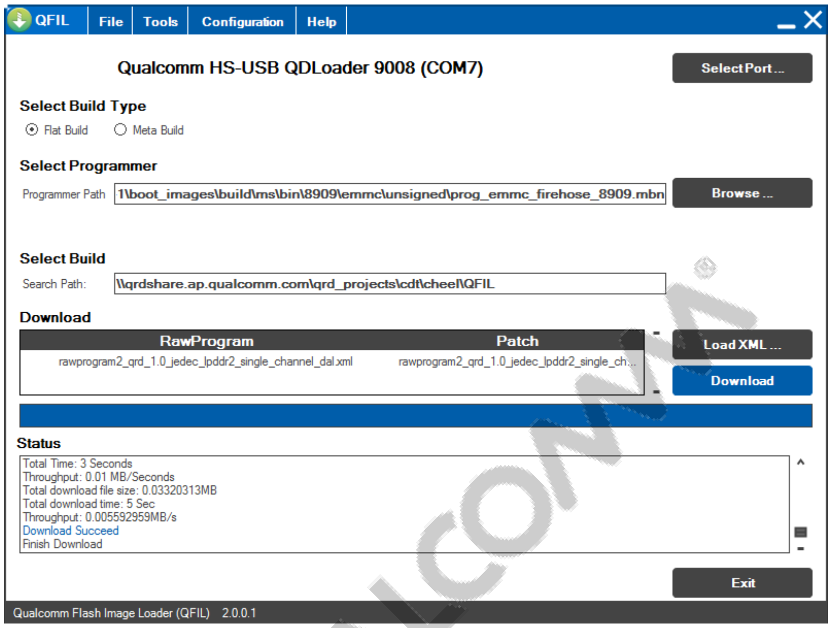
Qualcomm HS-USB QDLoader 9008 (COM7)

3. Select Build Type as Flat Build.
4. Select the programmer path from the Programmer Path field.

QUALCOMM®
yanzhennan@hipad.com

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

5. Click Load XML.



Qualcomm HS-USB QDLoader 9008 (COM7)

Select Build Type

(● Flat Build) ○ Meta Build

Select Programmer

Programmer Path: 1\boot_images\build\ms\bin\8909\emmc\unsigned\prog_emmc_firehose_8909.mbn

Select Build

Search Path: \\qrdshare.ap.qualcomm.com\qrd_projects\cdt\cheel\QFIL

Download

RawProgram	Patch
rawprogram2_qrd_1.0_jedec_lpddr2_single_channel_dal.xml	rawprogram2_qrd_1.0_jedec_lpddr2_single_ch...

Status

```
Total Time: 3 Seconds
Throughput: 0.01 MB/Seconds
Total download file size: 0.03320313MB
Total download time: 5 Sec
Throughput: 0.005592959MB/s
Download Succeed
Finish Download
```

Qualcomm Flash Image Loader (QFIL) 2.0.0.1

6. Click Download to start the CDT file download.

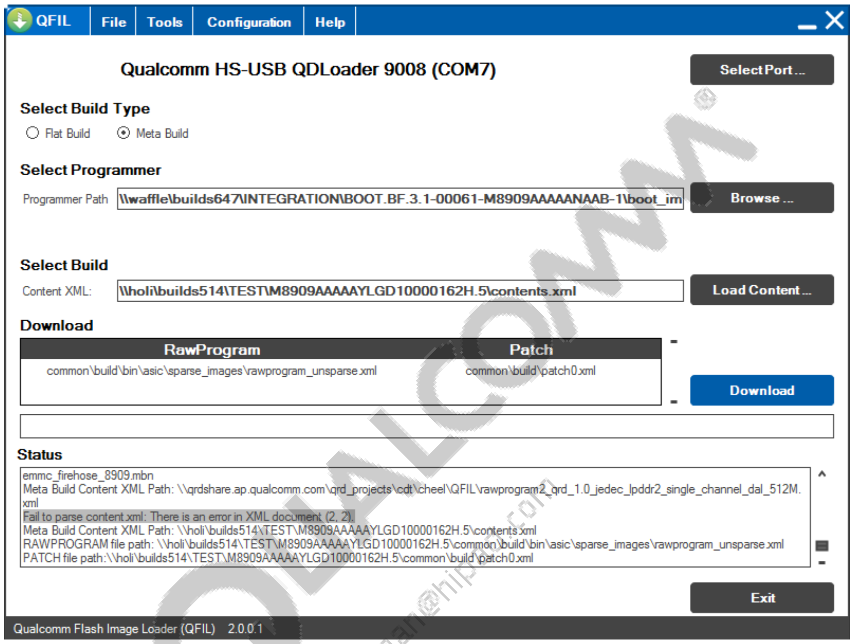
QUALCOMM
yanzhennan@hipad.com

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Download image by QFIL

1. Select Build Type as Meta Build.
2. Load contents.xml file.

   The Programmer Path field is populated automatically.



Qualcomm HS-USB QDLoader 9008 (COM7)

Select Build Type
Flat Build [✓ Meta Build]

Select Programmer
Programmer Path: \\waffle\builds647\INTEGRATION\BOOT.BF.3.1-00061-M8909AAAAANAAB-1\boot_im

Select Build
Content XML: \\holi\builds514\TEST\M8909AAAAAYLGD10000162H.5\contents.xml

Download

RawProgram	Patch
common\build\bin\asic\sparse_images\rawprogram_unsparse.xml	common\build\patch0.xml

Status

```
emmc_firehose_8909.mbn
Meta Build Content XML Path: \\qrdshare.ap.qualcomm.com\qrd_projects\cdt\cheel\QFIL\rawprogram2_qrd_1.0_jedec_lpddr2_single_channel_dal_512M.
xml
Fail to parse content.xml: There is an error in XML document (2, 2).
Meta Build Content XML Path: \\holi\builds514\TEST\M8909AAAAAYLGD10000162H.5\contents.xml
RAWPROGRAM file path: \\holi\builds514\TEST\M8909AAAAAYLGD10000162H.5\common\build\bin\asic\sparse_images\rawprogram_unsparse.xml
PATCH file path:\\holi\builds514\TEST\M8909AAAAAYLGD10000162H.5\common\build\patch0.xml
```

Qualcomm Flash Image Loader (QFIL) 2.0.0.1

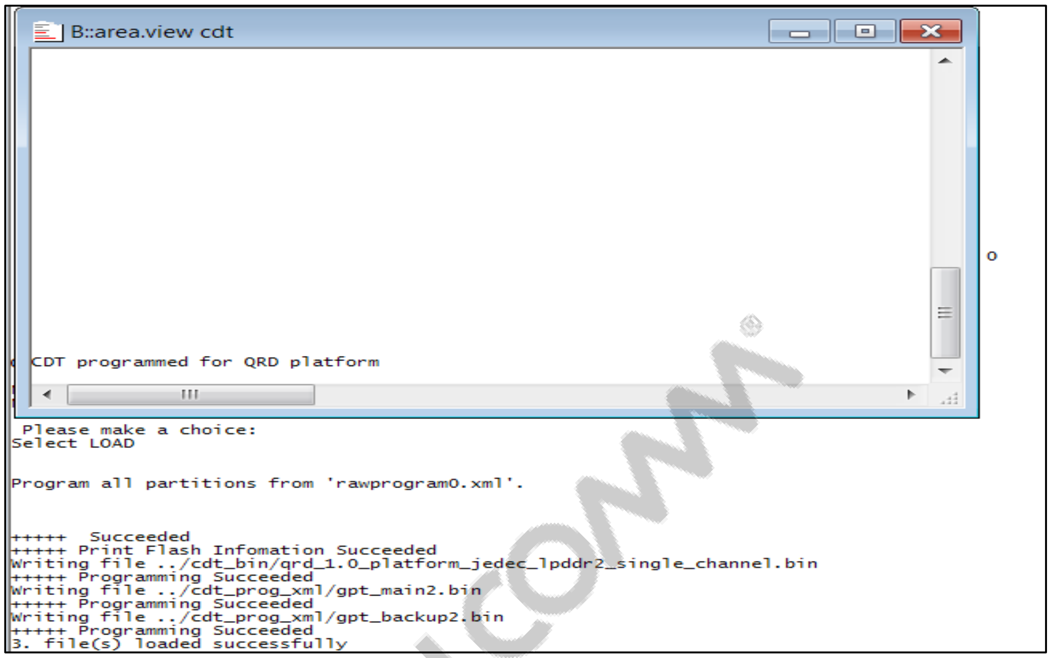
3. Click Download.

4.5.3 Programming CDT using JTAG

1. Set up T32 debug environment first (see section 4.5.4)
2. Run the CDT script in CortexA53Core0 session. The following information is shown if the script succeeds:

```
cd.do C:\emmc_cdt_program_script\
mtp_1.0_lpddr2_single_channel_emmc_cdt_program.cmm
```

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION



```
CDT programmed for QRD platform
```

```
 Please make a choice:
Select LOAD

Program all partitions from 'rawprogram0.xml'.


+++++  Succeeded
+++++ Print Flash Infomation Succeeded
Writing file ../cdt_bin/qrd_1.0_platform_jedec_lpddr2_single_channel.bin
+++++ Programming Succeeded
Writing file ../cdt_prog_xml/gpt_main2.bin
+++++ Programming Succeeded
Writing file ../cdt_prog_xml/gpt_backup2.bin
+++++ Programming Succeeded
3. file(s) loaded successfully
```

4.5.4 Programming system images using Fastboot

Before programming the system images, using 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
     ```

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

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

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

  - ii Run:

    ```
    sudo fastboot devices
    ```

A list of registered devices is shown on the console.

3. Once the device is detected, Flash the binaries to the target. The following commands run all 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 QSEE <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.

4.5.5 Flashing 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 following command to register a device:

```
sudo adb devices
```

QUALCOMM®

4. Navigate to the following directory:

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

5. Copy the following files:

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

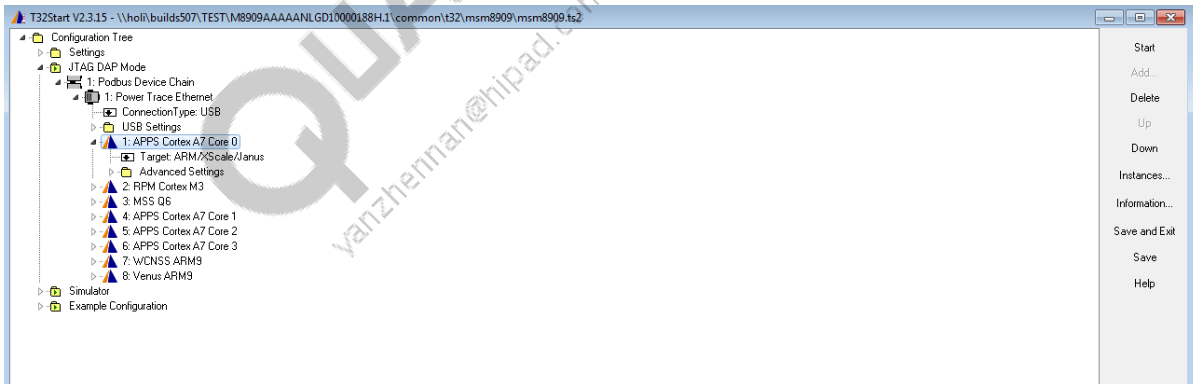
6. Push the files as follows:

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

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

4.5.6 Programming eMMC boot loaders with T32

1. Initiate T32 by using the t32start.cmd in the <meta build>\common\t32 folder.
2. Navigate to Configuration Tree → JTAG DAP Mode → Podbus Device Chain → Power Trace ethernet



```
T32Start V2.3.15 - \\holi\builds507\TEST\M8909AAAAANLGD10000188H.1\common\t32\msm8909\msm8909.ts2
Configuration Tree
  Settings
  JTAG DAP Mode
    1: Podbus Device Chain
      1: Power Trace Ethernet
        ConnectionType: USB
        USB Settings
        1: APPS Cortex A7 Core 0
          Target: ARM/XScale/Janus
          Advanced Settings
        2: RPM Cortex M3
        3: MSS Q6
        4: APPS Cortex A7 Core 1
        5: APPS Cortex A7 Core 2
        6: APPS Cortex A7 Core 3
        7: WCNSS ARM9
        8: Venus ARM9
  Simulator
  Example Configuration
```

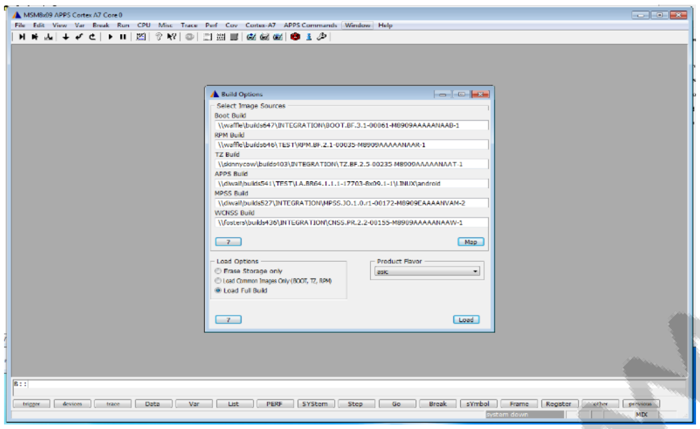
QUALCOMM®
yanzhennan@hipad.com

3. Select CortexA7 Core0 and click **Start**.
4. In CortexA7 Core0 T32 window, click **APPS COMMANDS** → **Build Options**, then select **ASIC** flavor in “Product Flavor” field and click **MAP**.

   The Build Options window displays.

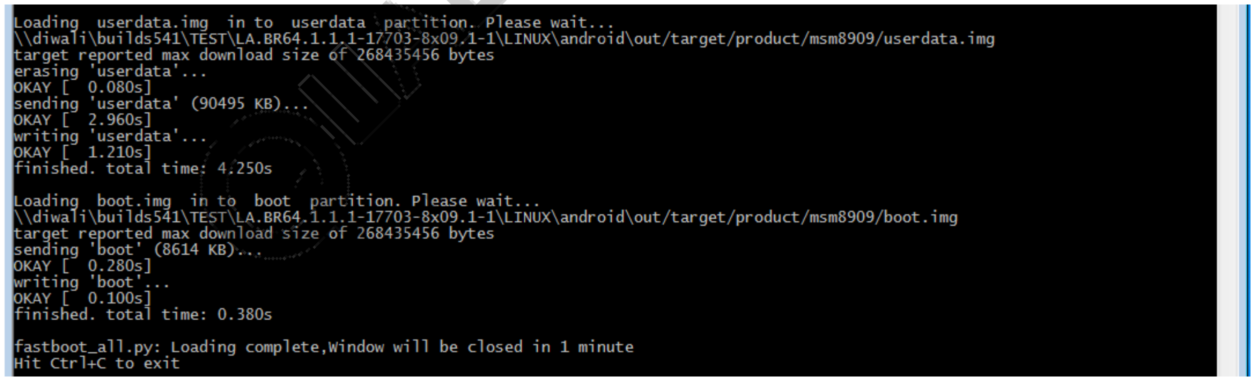
MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

5. Click **Load** in the Build Options window.



```
Build Options
Select Image Sources
Boot Build
\\waffle\builds647\INTEGRATION\BOOT.BF.3.1-00061-M8909AAAAANAAB-1
RPM Build
\\waffle\builds646\TEST\RPM.BF.2.1-00035-M8909AAAAANAAR-1
TZ Build
\\skinnycow\builds403\INTEGRATION\TZ.BF.2.5-00235-M8909AAAAANAAT-1
APPS Build
\\diwali\builds541\TEST\LA.BR64.1.1.1-17703-8x09.1-1\LINUX\android
MPSS Build
\\diwali\builds527\INTEGRATION\MPSS.JO.1.0.r1-00172-M8909EAAAANVAM-2
WCNSS Build
\\fosters\builds436\INTEGRATION\CNSS.PR.2.2-00155-M8909AAAAANAAW-1
Load Options
( ) Erase Storage only
( ) Load Common Images Only (BOOT, TZ, RPM)
(•) Load Full Build
Product Flavor
asic
```

6. Keep the USB connected to PC, as it uses fastboot to flash **NON-HLOS.bin** and apps binaries. Fastboot download process starts automatically by script after T32 finishes programming boot loaders.



```
Loading  userdata.img  in to  userdata  partition. Please wait...
\\diwali\builds541\TEST\LA.BR64.1.1.1-17703-8x09.1-1\LINUX\android\out/target/product/msm8909/userdata.img
target reported max download size of 268435456 bytes
erasing 'userdata'...
OKAY [  0.080s]
sending 'userdata' (90495 KB)...
OKAY [  2.960s]
writing 'userdata'...
OKAY [  1.210s]
finished. total time: 4.250s

Loading  boot.img  in to  boot  partition. Please wait...
\\diwali\builds541\TEST\LA.BR64.1.1.1-17703-8x09.1-1\LINUX\android\out/target/product/msm8909/boot.img
target reported max download size of 268435456 bytes
sending 'boot' (8614 KB)...
OKAY [  0.280s]
writing 'boot'...
OKAY [  0.100s]
finished. total time: 0.380s

fastboot_all.py: Loading complete,Window will be closed in 1 minute
Hit Ctrl+C to exit
```

7. After fastboot completes flashing the binaries, power cycle the device.

5 Operational Guide

5.1 CDP and MTP SIM slots, primary antenna, and USB ports

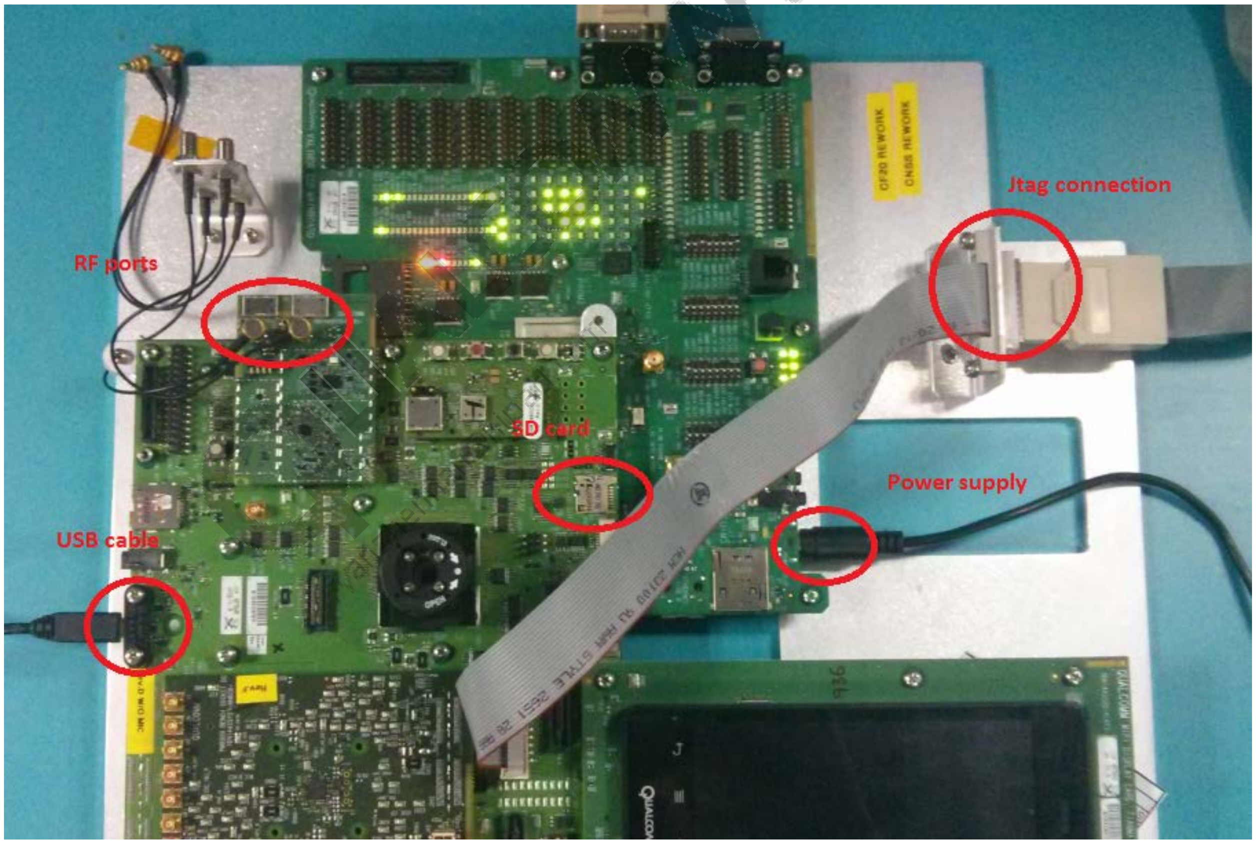


Figure 5-1 CDP USB/JTAG/RF/SD card/power configuration

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

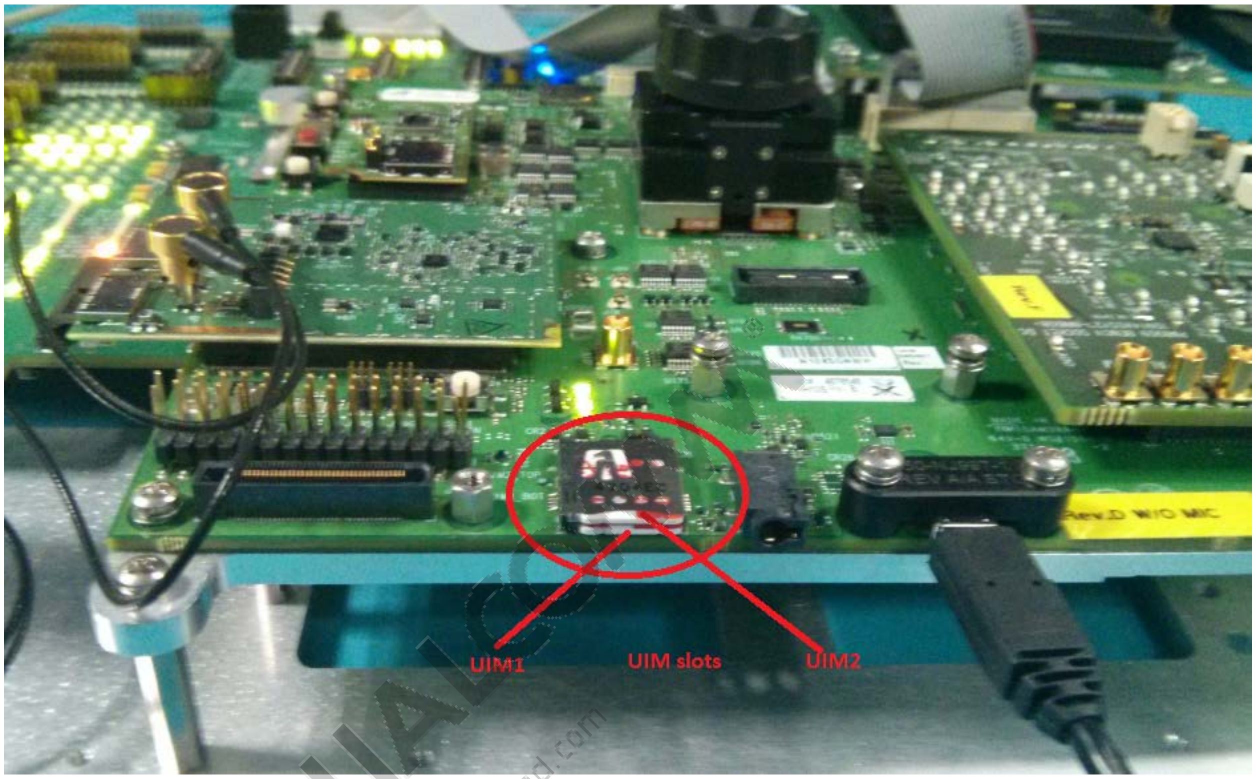


Figure 5-2 CDP UIM configuration

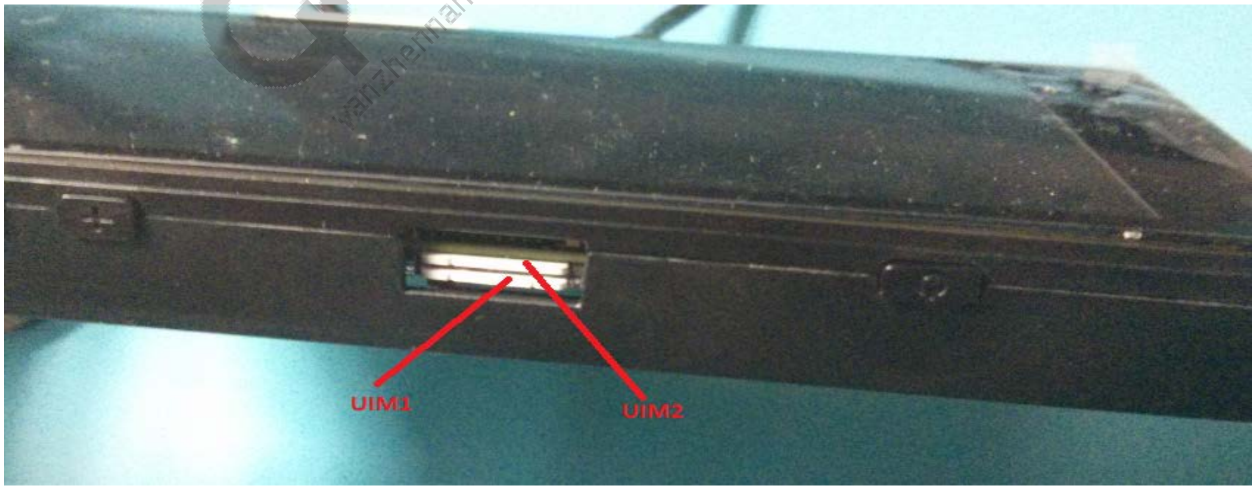


Figure 5-3 MTP UIM configuration

QUALCOMM yanzheman@hipad.com

5.2 Common NV settings

5.2.1 RF NV settings

1. Static QCN is generated from the XML shipped within the MPSS build. Pick the appropriate xml file from the following path:

   <MPSS ROOT>\modem_proc\rftarget_dimepm\common \rftarget_dimepm\common\qcn\

2. Choose an RF card per the requirements.
3. Run QRCT to generate a static QCN, select the RF configuration to be used and then convert it to QCN.
4. Navigate to Tool → NV tools and pick the master xml file from the following build path:

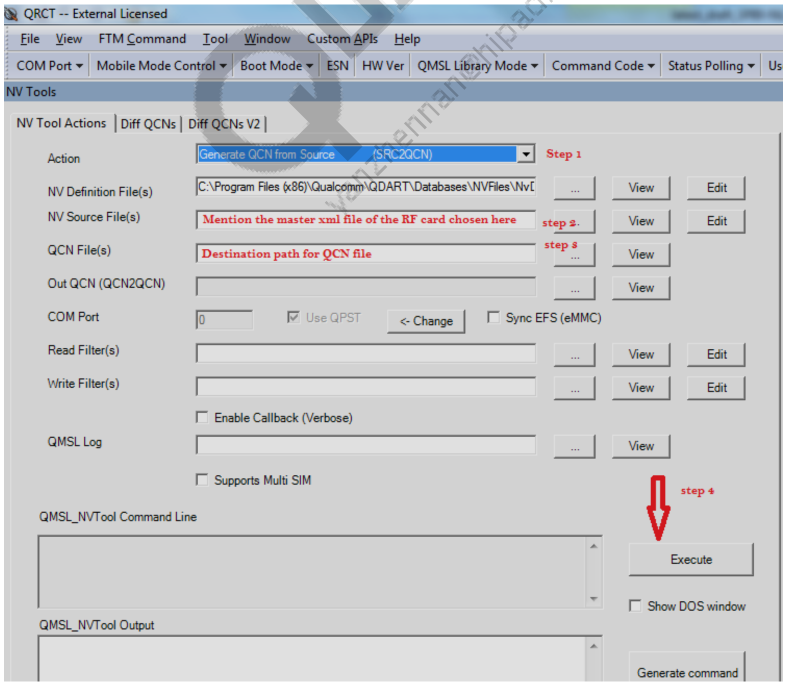
   <MPSS ROOT>\modem_proc\rftarget_dimepm\common \rftarget_dimepm\common\qcn\

   For example, for RF card: RFC_WTR1605_CHILE_SVDSDA, the build path must be as follows:

      \modem_proc\rftarget_dimepm\common\qcn\wtr1605_chile_svdsda\etc
      MSM8909_WTR1605_CHILE_SVDSDA_MASTERFILE.

5. Click Execute.

QUALCOMM
wangzhennan@shipad.com


QRCT -- External Licensed
File View FTM Command Tool Window Custom APIs Help
COM Port Mobile Mode Control Boot Mode ESN HW Ver QMSL Library Mode Command Code Status Polling Us
NV Tools
NV Tool Actions | Diff QCNs | Diff QCNs V2
View
Edit
...
Read Filter(s)
Write Filter(s)
QMSL_NVTool Command Line
QMSL_NVTool Output
Execute
Generate command

Action: Generate QCN from Source (SRC2QCN) — Step 1

NV Definition File(s): C:\Program Files (x86)\Qualcomm\QDART\Databases\NVFiles\NvD

NV Source File(s): Mention the master xml file of the RF card chosen here — step 2

QCN File(s): Destination path for QCN file — step 3

Out QCN (QCN2QCN):

COM Port: 0 [✓ Use QPST] <- Change ☐ Sync EFS (eMMC)

☐ Enable Callback (Verbose)

QMSL Log:

☐ Supports Multi SIM

step 4

☐ Show DOS window

Figure 5-4 QRCT NV tool for generating QCN

NOTE: The MPSS build has a reference QCN as a deliverable. The QCN contains RF-only NV items and does not have XO TRIM NV item in it. Hence, for basic functional testing, this QCN is loaded followed by XO-only calibration. The reference QCNs are found at \modem_proc\rftarget_dimepm\common

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

5.2.2 NV configuration for WCDMA/GSM + GSM

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 2 ms; 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

QUALCOMM® wangzhenkan@hipad.com

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

5.2.3 NV configuration for CDMA + GSM

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 domain preference)	0x02 CS and PS	0x00 CS only
905 (Fatal error option)	0x0000 fatal error option	Same as SUB1

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

NV item	Subscription1 (CDMA+ HDR) value	Subscription2 (GSM) value
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 configuration 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 Pro command window)	Same as SUB1

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

5.2.4 LTE + GSM DSDS settings

The following changes are required to ensure that the device is correctly configured for LTE + GSM DSDS mode of operation.

VoLTE/IMS/eMBMS and CSG features are not supported with LTE + GSM DSDS. This section explains all the required NV items for disabling VoLTE/IMS/eMBMS as well as other parameters for LTE + GSM mode of DSDS operation.

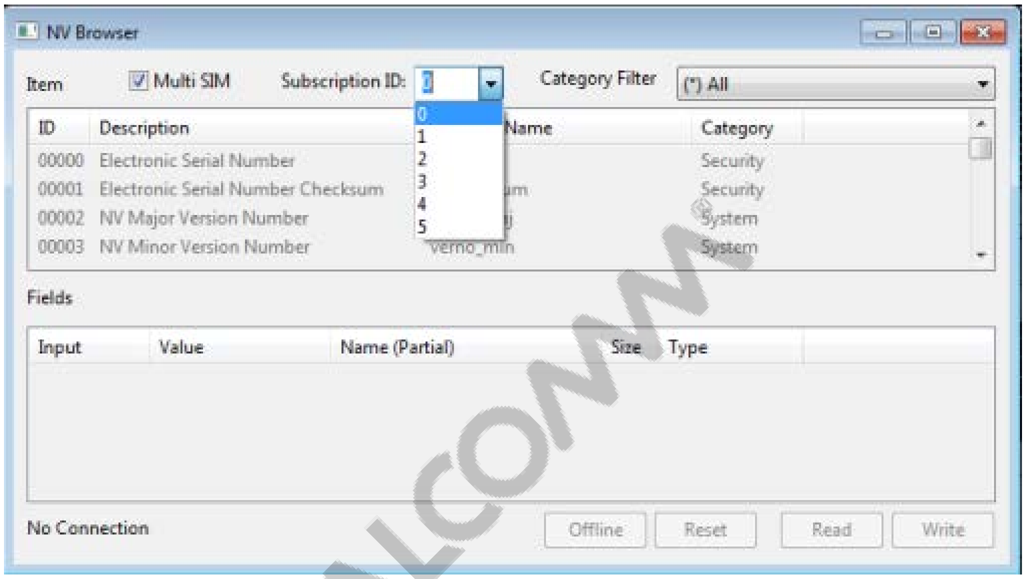
Table 5-3 LTE + GSM DSDS settings

Feature name	How to disable
VoLTE/IMS	Refer [Q37]
eMBMS	EFS file embms_feature_status (with value set as 0) must be copied under the path /nv/item_files/modem/lte/rrc
CSG	Create an EFS file viz. csg_control at the path "/nv/item_files/modem/lte/rrc/csg/". A 16-byte hexadecimal value is written into it 01 01 01 00 01 00 00 00 E0 93 04 00 00 00 00 00

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Enable DSDS NV settings

1. Set the following NVs in QXDM Pro NV browser.


NV Browser
Item
Multi SIM
Subscription ID:
Category Filter
(*) All
ID
Description
Name
Category
0
1
2
3
4
5
00000 Electronic Serial Number Security
00001 Electronic Serial Number Checksum Security
00002 NV Major Version Number System
00003 NV Minor Version Number verno_min System
Fields
Input
Value
Name (Partial)
Size
Type
No Connection
Offline
Reset
Read
Write

QUALCOMM yanzhennan@hipad.com

Subscription 0	
NV item number	**NV item value**
00010	31 (GWL)
00850	0x01
65777	1
70210	hw_config.UIM[1].DISABLE_UIM set to FALSE
06876	5
06907	1
04398	0

2. Perform “Spc 000000” in QXDM Pro and then set the following:

NV item number	NV item value
00855	0 (for both single SIM and dual SIM)
70266	1 (for dual SIM)

Subscription 1	
NV item number	**NV item value**
00010	13 (GSM-only)
00850	0x00
65777	0

3. Set the NV settings for IRAT (NV settings are required only on Subscription 0).

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Subscription 0	
NV item number	**NV Item value**
00010	34 (G+L)
00850	0x02
65777	0
00946	1F
02954	0

4. Restart the UE and run the dual SIM commands from ADB shell:

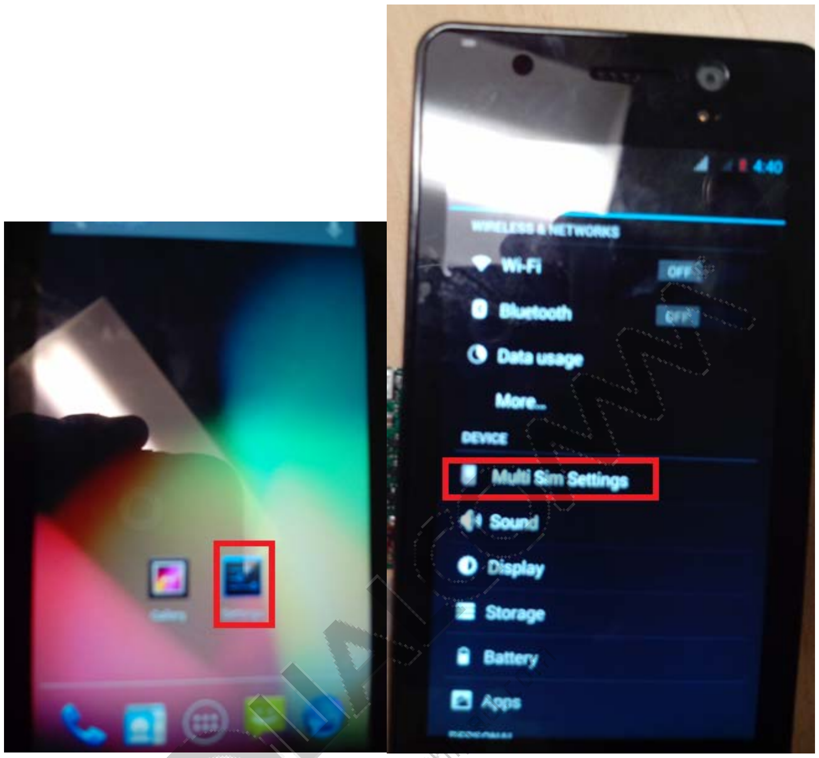
```
adb devices
adb root
adb shell
setprop persist.radio.multisim.config dsds
getprop persist.multisim.config (it must show up as DSDS)
```

5. Restart the UE.

QUALCOMM®
yanzhennan@hipad.com

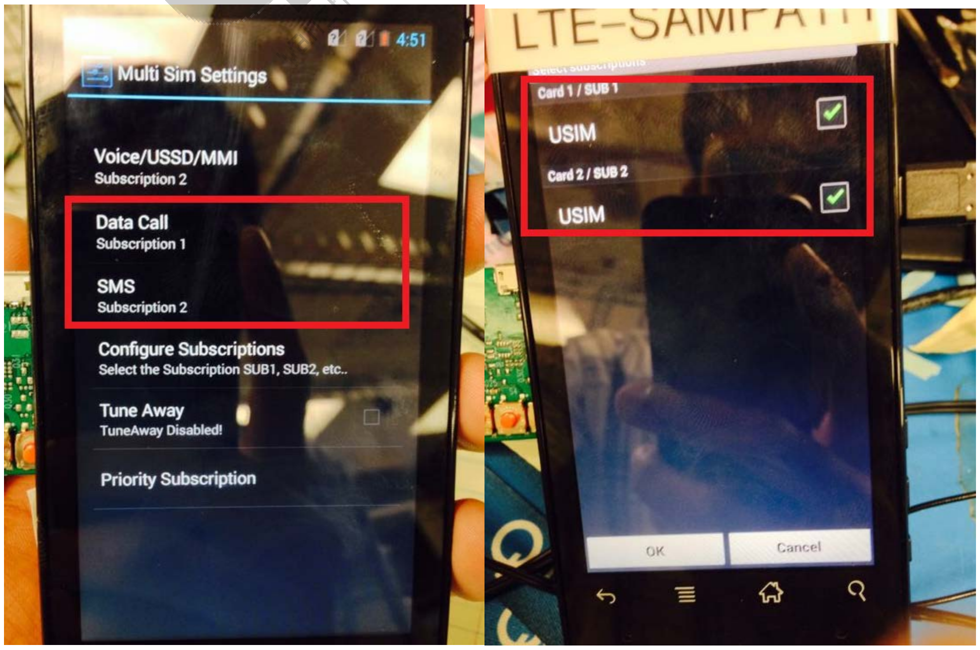
MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

6. Go to Settings on the UI, and perform the below setting.



7. Select the Multi SIM settings on the UI.

8. Configure the UIM cards to respective Subs by selecting the Configure Subscriptions.



Multi Sim Settings

Voice/USSD/MMI
Subscription 2

Data Call
Subscription 1

SMS
Subscription 2

Configure Subscriptions
Select the Subscription SUB1, SUB2, etc..

Tune Away
TuneAway Disabled!

Priority Subscription

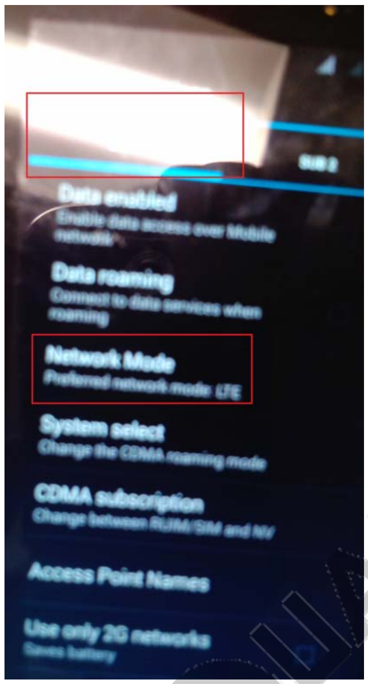
Select subscriptions

Card 1 / SUB 1
USIM ✓

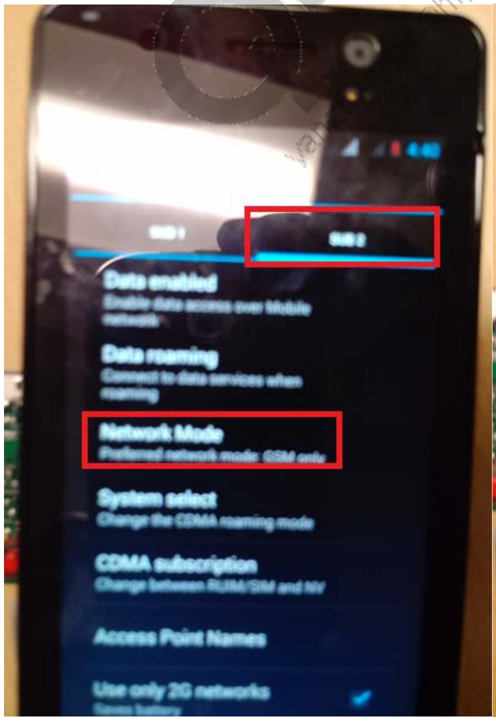
Card 2 / SUB 2
USIM ✓

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

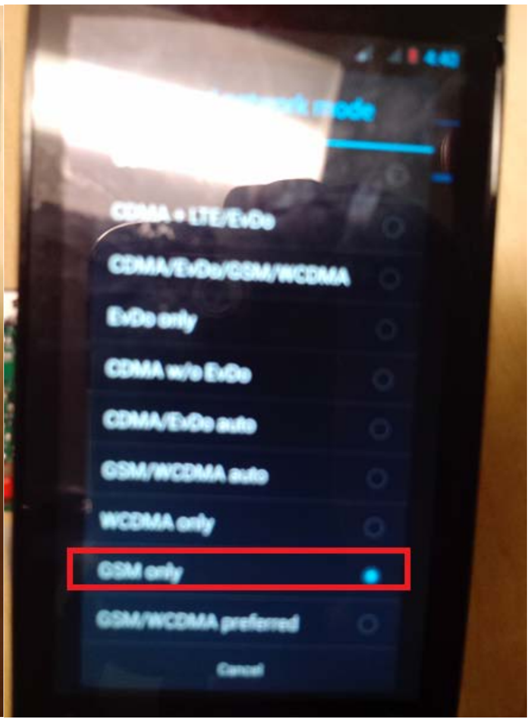
9. Go back to settings, select Mobile Networks, and select subscriptions. Set the respective subscriptions to LTE and GSM.


SUB 2
Data enabled
Enable data access over Mobile network
Data roaming
Connect to data services when roaming
System select
Change the CDMA roaming mode
CDMA subscription
Change between RUIM/SIM and NV
Access Point Names
Use only 2G networks
Saves battery

Network Mode
Preferred network mode: LTE


4:40
SUB 1
SUB 2
Data enabled
Enable data access over Mobile network
Data roaming
Connect to data services when roaming
System select
Change the CDMA roaming mode
CDMA subscription
Change between RUIM/SIM and NV
Access Point Names
Use only 2G networks
Saves battery

Network Mode
Preferred network mode: GSM only



[✓ GSM only]

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

10. Run the below command after power-up, to check if both RILs are activated for DSDS.

```
C:\>adb shell ps rild
USER   PID  PPID  VSIZE  RSS   WCHAN    PC          NAME
radio  243   1    17968  2900  ffffffff 00000000 S  /system/bin/rild
radio  247   1    17952  2932  ffffffff 00000000 S  /system/bin/rild
```

11. Check whether the below command returns only one radio as output, and if it returns only one radio as output then the RIL still is in single SIM mode.

```
C:\>adb shell ps rild
USER   PID   PPID  VSIZE  RSS     WCHAN    PC          NAME
radio  166   1     16928  2828  ffffffff 00000000 S  /system/bin/rild
```

5.2.5 1xSRLTE settings

The following settings need to be done to configure the device in 1xSRLTE mode of operation.

Table 5-4 1xSRLTE settings

NV item	Value	Description
72539	1	ESR support/CSFB support for dual Rx UEs 1 – Supported 2 – Not supported
72550	500 (ms)	LTE NAS 1xSRLTE ESR delay timer

5.3 Segment loading configuration

As part of memory optimizations, there is a segment loading feature that assumes that there will not be any use case where the network supports both the modes (WCDMA and TD-SCDMA) at one place. The segment loading feature is controlled by FEATURE_SEGMENT_LOADING and when enabled in MPPS build and based on the UE supported modes, it supports two segments i.e., (LTE/GSM/WCDMA/CDMA) or (LTE/GSM/TD-SCDMA/CDMA). The segment loading works in two modes i.e. Automatic or Manual modes, and by default it is set to Manual mode.

Manual mode

For Manual mode:

- On a fresh load of the build, the UE is by default configured for Manual mode of segment loading operation.
  - NV72542 – 2 (default) → The UE works in WCDMA Segment mode.
  - NV72542 – 1 → The UE works in TD-SCDMA Segment mode.

wanzhennan@hipad.com

- In Manual mode, there are no switches between WCDMA and TD-SCDMA segments, as the Manual mode means that the UE is set to operate as determined by the NV72542 setting.
    - Changing the NV72452 value between 1 and 2, the UE can be set to operate with the WCDMA segment or TD-SCDMA segment.

At power-up for the first time and based on the NV 72542 setting, the UE loads the said NV image initially and starts searching for the network as per the configured RAT priority order. If the UE fails to get the coverage on the network in Manual mode, change the NV72542 explicitly to search for other networks. To avoid the user intervention for NV switch and finally network search, set the segment loading to Automatic mode.

Automatic mode

1. To take the UE out of the Manual mode and to put it in Automatic mode, load the segment_loading.xml file as below:
2. Copy from <build_root>\modem_proc\mmcp\policyman\configurations\SegLoad\ segment_loading.xml to \..\policyman\ on the device.
3. Change NV 10 to “auto”.

- While in Automatic mode, the UE loads the image based on the NV 72542 setting.
    - Segment switching occurs as per the logic outlined in the segment_loading.xml file.
    - While in service on MCCs is listed in segment_loading.xml file, TDS segment is loaded.
    - While in service on MCCs not listed in segment_loading.xml file, WCDMA segment is loaded.
    - While in complete OOS, segment switching occurs as determined by the timers listed in segment_loading.xml.
    - Logic/comments in “<build_root>\modem_proc\mmcp\policyman\configurations\SegLoad\ segment_loading.xml” file offer detailed steps.
    - Segment switching occurs with modem subsystem reset.
    - The segment_loading.xml file works independent of other policy manager configuration files.

NOTE: Do not set NV 72542 to 0. It causes device runtime to crash due to a heap exhaustion

5.4 Call configuration

5.4.1 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 is on the SV chain by default)
- PRL (must match the band/channel being tested, and SID & NID must also match)

yanzhinnan@hipad.com

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;
    - 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 ~ -65 dBm);
    - 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);

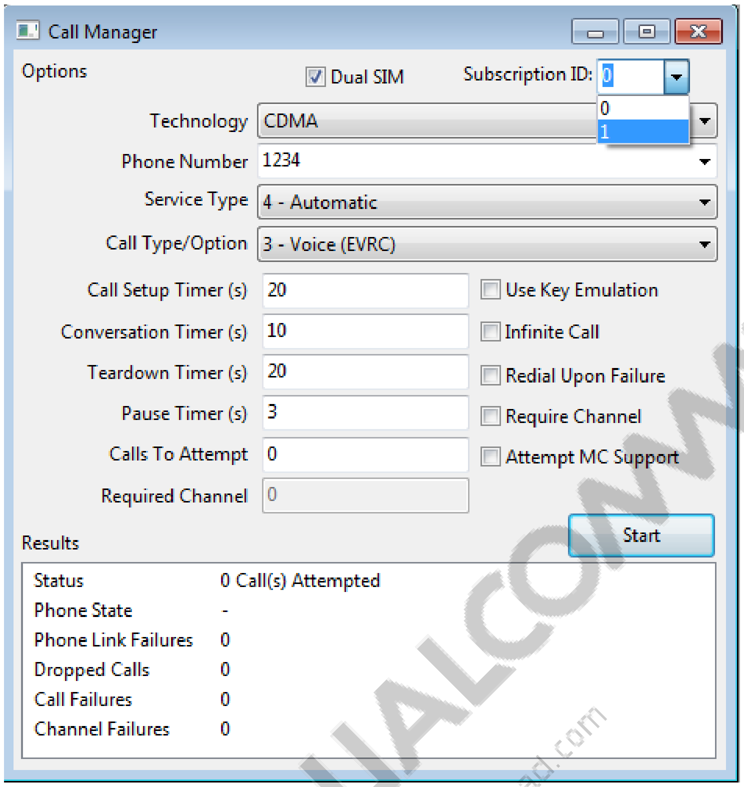
Device setup

To make the call:

1. Use the Call Manager screen in QXDM Professional (QXDM Pro).
2. Set the phone number to something like 1234 and make sure that the service option matches the callbox setting.
3. 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** and then select **Subscription ID**.

wenzhennan@hipad.com



Options	
Dual SIM	✓
Subscription ID:	0 (options: 0, 1)
Technology	CDMA
Phone Number	1234
Service Type	4 - Automatic
Call Type/Option	3 - Voice (EVRC)
Call Setup Timer (s)	20
Conversation Timer (s)	10
Teardown Timer (s)	20
Pause Timer (s)	3
Calls To Attempt	0
Required Channel	0
Use Key Emulation	☐
Infinite Call	☐
Redial Upon Failure	☐
Require Channel	☐
Attempt MC Support	☐

Results

Status	0 Call(s) Attempted
Phone State	-
Phone Link Failures	0
Dropped Calls	0
Call Failures	0
Channel Failures	0

5.4.2 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 is 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 must be set somewhere between -45 and -65 dB

QUALCOMM
tanzhennan@hipad.com

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Device setup

To make the call:

1. Use the Call Manager screen in QDXM Pro.
2. Set the phone number to something like 1234 and make sure that the service option matches the callbox setting.
3. Click **Call** to start a call; for MT, originate the call from the test box.

5.4.3 HDR call

NV settings

- NV setting (set NV # 10 to 4 for Automatic mode)
- For DO Rev A calls, NV #4964 must 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 must be loaded and subnet ID in PRL must match the callbox setting
- Roaming list is loaded via QPST Service Programming; perform 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 Rev A calls, the callbox must be placed in Rev A mode

Device setup

To make a call:

1. Power-cycle the device
2. Enter **mode online** in the QXDM Pro command bar; the device must attempt to acquire HDR channel and negotiate a session

QUALCOMM®
yanzhenhan@hipad.com

5.4.4 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 is changed via the UI. Otherwise, multimode uses the default setting in Android (defaults to 1X only) and it uses 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 is performed on Agilent 8960 or Anritsu MT8820
- Set band to GSM900; cell power must be set somewhere between -45 and -65 dBm
- Set Channel mode to TCH/F (full rate TCH)

Device setup

To make a call:

1. Use the Call Manager screen in QDXM Pro.
2. Set the phone number, e.g., to 1234, and make sure that the service option matches the callbox setting.
3. Click **Call** to start a call; for MT, originate the call from the test box.

QUALCOMM
vanzhennan@hnad.com

5.4.5 GPRS data call

RF calibration

- Ensure the device is 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

Device setup

To make a call:

1. Insert a SIM (test SIM can also be used)
2. Power up the device; Enter **mode online** in QXDM Pro command bar if necessary
3. The UE camps on GPRS cell and ATTACH
4. Initiate Test mode A data call; hit Start Data Connection; bottom of screen must display TRANSFERRING

5.4.6 WCDMA voice call

Prerequisites

- QCN – Ensure that the device is calibrated

Callbox setup

- Agilent 8960
  - Call 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

To make a call (MO):

1. After starting firmware and software, attach the USB and then do a “mode online” from QXDM Pro command bar
2. Allow the mobile to acquire and register to network
3. Start a call using the Call Manager dialog in QXDM Pro
4. Set Technology to WCDMA

QUALCOMM® vanherman@shipsd.com

5. Set phone number to 1234
6. Check the **infinite call** box
7. Start the call
8. Phone state shows “Originating call” and then “Conversation”

5.4.7 WCDMA data call

Prerequisites

- QCN – Ensure device is 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

To make a call:

1. Allow UE register to network
2. Click “**Connect/Dial**” on USB (DUN) or QMICM to initiate data call
3. Run iPerf or FTP applications to test throughput (we tested with FTP)

5.4.8 TD-SCDMA call

Prerequisites

- QCN – Ensure device is calibrated
- NV_PREF_MODE_1(10) – Set to 53 TD-SCDMA only
- Set NV 00850 = 0x2 (CS PS)

QUALCOMM® yanzhenna@hipad.com

- TDS RRC integrity protection enabled (66011) – Set to 0; set to 1 for a live network
- TDS RRC ciphering enabled (66012) – Set to 0; set to 1 for a live network
- TDS RRC fake security status (66013) – Set to 0; set to 1 for a live network
- TDS RRC special frequency enabled (66014) – Set to 0
- TDS RRC PDCP disabled (66016) – Set to 1
- TDS RRC version (66017) – Set to 3
- TDS RRC HSDPA category (66020) – Set to 15
- TDS RRC NV version (66023) – Set to 2
- TDS RRC optional feature bitmasks (66024) – Set to 0x3F
- TDS RRC Special Test Setting Enabled (69731) – Set to 0; set to 1 when entering MTNet and CMCC tests

Callbox

- Agilent 8960
  - Callbox setup
    - Cell Power – -45.00 dBm
    - Channel Type – 12.2K RMC SC
    - Paging Service – AMR Voice
    - Channel – Desired band/channel
    - Screen2 – Cell Info → Cell Parameters → PS Domain Information → Present
- Anritsu 8820
  - Callbox setup
    - Call processing – On
    - Test Loop Mode – Mode 1
    - Signal/Chanel Coding – Fixed reference channel

Device setup

To make a call:

1. Configure the test equipment using the settings mentioned in Section 5.4.8 and check that the signal is acquired.
   a. MO and MT Calls - Initiate the MO call from device and MT call from the test equipment.
2. Send/receive SMS from/to device and test equipment.
3. DUN call using a dial-up connection “PS on USB” setup a DUN data Call.
4. Browse data from web browser to data session.

QUALCOMM® yanzhenhang@hipad.com

5.4.9 LTE data call

Prerequisites

- QCN – Ensure device is calibrated
- NV Settings – Set NV 00010 = 30 (LTE-only) and NV 00850 = 0x1 (PS-only) and NV 65777 = 0x1

Callbox setup

- Aeroflex
  - Cal box setup
    - Under home screen, select Mode as Callbox Mode
    - Select Parameter Config Tab.Parameter Group → Select System → Band ID Ex: For TDD Testing:38 and For FDD Tsting:1
    - Under Parameter Group select Layer 3 → MCC 001 and MNC 01 based on the SIM configuration.
- Anritsu 8820
  - Callbox setup
    - Call Processing – On
    - Test Loop Mode – Off
    - Signal/Channel Coding – Required LTE band/channel
    - Start Call

Device setup

To make a call:

1. Connect primary and secondary RFs to the respective ports.
2. Run mode LPM in QXDM.
3. Click Registry software ICON.
4. Click Application software ICON.
5. In Apps software, open TC and run.
6. Click Analyzer → Run. A message appears, “Please power on the phone” in application software.
7. Run mode online in QXDM.

   The UE registers to the network with status in Conversation state.
8. Verify browsing by opening a web page.

QUALCOMM® yanzhenhan@hipad.com

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

5.5 GPS configuration

Prerequisites

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

5.6 Multimedia configuration

This section describes the bringup of multimedia features like audio, display, video, and camera. For further details, see [Q22].

5.6.1 Audio configuration

This section will be included in a future revision of this document.

5.6.2 Display

For display panel bring-up and driver porting-related information, see [Q25]. [Q25] describes application usage of the Display Serial Interface (DSI) panel bring-up for the Android OS. It also provides sample code and PLL calculation pertaining to the DSI Mobile Industry Processor Interface (MIPI) panel bring-up. For details on Linux Android display driver porting, see [Q26].

5.6.3 Camera

To verify the camera features, the Android default camera application is used. For details related to sensor driver porting and migration, refer to [Q31].

For techniques on debugging different kinds of camera errors, refer to [Q32].

This section will be included in a future revision of this document.

5.6.4 Video

To verify the playback of various video formats, the Android default video player application is used.

This section will be included in a future revision of this document.

5.7 WCNSS configuration

WCNSS functionality does not require any specific configuration as everything is built-in. The basic functionality is verified using either FTM tool or GUI (default Android settings application). For FTM tool usage, see [Q27].

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

5.8 Subsystem Restart (SSR)

Subsystem Restart is a feature designed to give a seamless end-user experience when restarting after a system malfunction. The SoC is considered to be divided into individual subsystems (for example, Modem, WCNSS, etc.), and a central root, the Applications Processor (AP). Th**e** clients of these individual subsystems that are running on the AP receive notification from the kernel about a particular subsystem shutting down. The clients must be able to handle this notification in a graceful manner. The clients can expect to receive another notification when the subsystems are in back up. Examples of such clients are EFS sync, remote storage, etc.

The use case for this feature is a catastrophic restart, i.e., a software malfunction on the modem, or any other subsystem that could cause the phone to be dysfunctional. In this instance, the AP is expected to restart the respective subsystems, to restore them to normal operation.

The core restart module, powers up/down the registered subsystems when they crash and sends appropriate notifications.

Compile options to enable SSR – CONFIG_MSM_SUBSYSTEM_RESTART

Following are some useful adb commands for SSR:

- To find a specific subsystem
  ```
  cat /sys/bus/msm_subsys/devices/subsysX/name (here X = 0,1,2 for
  different subsystems modem,wcnss etc.)
  ```
- To know if SSR is enabled on a specific subsystem
  ```
  cat /sys/bus/msm_subsys/devices/subsysX/restart_level
  ```
- To enable SSR for various subsystems
  ```
  echo related > /sys/bus/msm_subsys/devices/subsysX/restart_level
  ```
- To disable SSR for various subsystems
  ```
  echo system > /sys/bus/msm_subsys/devices/subsysX/restart_level
  ```

For further information on SSR, refer Q34, Q35, and Q36.

QUALCOMM
yanzhen.me@hmat.com

6 Factory Tools

6.1 QDART-MFG and TPP

The QDART-MFG installer is a set of factory tools designed to manufacture, reduce the setup steps, and optimize installation size and installation time. It provides GoNoGo UI, QSPR test framework, test solution for all test stations, including, software download and upgrade, RF calibration and verify, Bluetooth and WLAN, MMI, radiated, and service programming. For more details, see [Q28] and [Q29]

6.2 FactoryKit

The FactoryKit is an Android application used for MMI test. It functions similar to FastMMI, but with longer boot-up time. The FactoryKit is used on customer devices and all QRD SKU devices.

QUALCOMM®
yanzhennan@hipad.com

A 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/` – HAL (audio, sensors) and Qualcomm specific hardware wrappers
- `vendor/qcom/` – Qualcomm target definitions, e.g., msm7201a_surf
- `vendor/qcom-proprietary/` – Qualcomm-proprietary components, for example, 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
    - `obj/` – Intermediate object files
      - `include/` – Compiled include files from components
      - `lib/`
      - `STATIC_LIBRARIES/`
      - `SHARED_LIBRARIES/`
      - `EXECUTABLES/`
      - `APPS/`
- `symbols/` – Symbols for all target binaries

QUALCOMM

A.1 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

QUALCOMM®
wanzhenmeng@hipad.com

A.2 Building Tiny Android

Tiny Android, or TINY_ANDROID, is a build variant that creates only a superminimal build configuration used for board bringup and 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.3 Building 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:

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

```
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 appears 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 created.

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
   ```

A.4 Building 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
   ```

QUALCOMM®
yanzhennan@hipad.com

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
     ```

A.5 Other important Android build commands

Table A-1 lists other important Android build commands.

Table A-1 Other important Android build commands

Android build command	Description
printconfig	Prints the current configuration as set by the choosecombo commands.
m	Runs make from the top of the tree. This is useful because you can run make from within subdirectories. If you have the TOP environment variable set, the commands use it. If you do not have the TOP variable set, the commands look 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-$(LOCAL_MODULE) clean-$(LOCAL_PACKAGE_NAME)	Allows you to selectively clean one target. For example, you can type make clean-libutils, and it deletes libutils.so and all of the intermediate files, or you can type make clean-Home and it cleans just the Home application.
make clean	Deletes all of the output and intermediate files for this configuration. This is the same as `rm -rf out/<configuration>/`

QUALCOMM yanzhennan@hipad.com

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.
    - 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
- Predefined action for compiling executables, shared libraries, static libraries, Android packages, using precompiled binaries, etc.
    - 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.

Adding a module

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 the Android.mk.

   For Applications modules, use LOCAL_PACKAGE_NAME instead.
3. Set the local path in the new module by inserting the following as the first line in the Android.mk file:

```

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

NOTE: The local path in the new module is LOCAL_PATH. This is the directory where the Android.mk file is available.

QUALCOMM

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4. Prefix them with the directory name:

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

NOTE: 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

5. Configure the new module with the following:

   LOCAL_STATIC_LIBRARIES – These are the static libraries that are included in the 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, ensure to 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 occurs.

QUALCOMM®
yanzhenhan@mat.com

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION