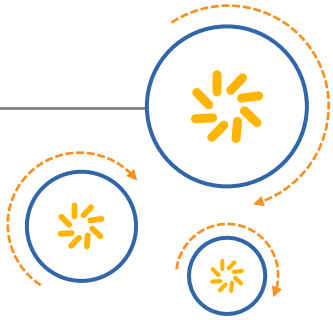
Qualcomm Atheros, Inc.



IPQ4019.ILQ.1.1 CSU1

Release Notes

80-Y9592-4 Rev. C

March 15, 2016

Qualcomm Atheros, Inc.
1700 Technology Drive
San Jose, CA 95110
U.S.A.

Revision history

Revision	Date	Description
A	February 2016	Initial release
B	March 2016	Section 9.3: Updated firmware build instructions.
C	March 2016	Section 9.3.2.1: Removed legacy information.

Contents

1 Introduction 5

1.1 Purpose 5
1.2 Identification 6
1.3 Related documentation 6

2 Features 7

2.1 Features 7
2.2 Release contents 7

3 Supported Hardware 8

3.1 AP.DK01 (IPQ4018/IPQ4028), AP.DK04 (IPQ4019/IPQ4029) 8
3.2 Hardware reference guides 8

4 Restrictions 9

5 Power Management 10

6 System Frequencies 11

7 Performance Data 12

7.1 AP.DK04 WLAN Performance 12
7.1.1 WLAN performance results 12
7.1.2 DBDC performance test setup 15
7.1.3 WLAN setup steps 16
7.2 AP.DK04 wired performance 19
7.2.1 Wired and SAMBA performance results 19
7.2.2 SFE/HNAT Routing test setup 20
7.2.3 IPsec Ethernet-Ethernet performance setup 22
7.2.4 Samba/Storage wired performance setup 22
7.2.5 Raw crypto performance 22

8 Known Issues 23

9 Download and Build Instructions 24

9.1 QSDK profiles 24
9.2 Download code 25
9.2.1 Download packages available through ChipCode 25
9.2.2 Packages downloaded from external websites 25
9.3 Firmware generation 29
9.3.1 Reassemble the code 29
9.3.2 Build QSDK 30
9.3.3 Generate a complete firmware image 31
9.4 Flashing instructions 33
9.4.1 Flashing commands 33
9.4.2 Upgrade the firmware 35
9.5 Procedure to load Ramdisk image 36

A Minimizing Build Times and Avoiding Multiple Downloads 37

B QDART_Connectivity 39

B.1 Overview 39
B.2 Changes 40
B.3 Known limitations 40

C Supported Features 41

Figures

Figure 7-1 Ethernet to WLAN 802.11ac + 802.11n 16
Figure 7-2 Example two-node test setup 16
Figure 7-3 Spirent/IXIA traffic generator 20
Figure 7-4 IPsec Ethernet – Ethernet performance setup 22
Figure 7-5 File operations over LAN 22

Tables

Table 3-1 AP.DK01 and AP.DK04 8
Table 4-1 Restrictions on the software 9
Table 6-1 System frequencies 11
Table 7-1 AP.DK04 wireless throughput in bridging mode (WDS) 12
Table 7-2 AP.DK04 wireless throughput in wired QWRAP mode 13
Table 7-3 WDS – TCP 11ACVHT80 14
Table 7-4 WDS – TCP 11NGVHT40 15
Table 7-5 RFC2544 peak throughput for wired use case using SFE 19
Table 7-6 Peak RFC2544 Bi-Di throughput using HNAT for all frame sizes 19
Table 7-7 Storage KPI 19
Table 7-8 IPSec KPI 20
Table 8-1 Open issues for this release 23
Table 8-2 Fixed issues for this release 23
Table 9-1 Source code top-level structure 25
Table 9-2 Packages available from external sites 25
Table B-1 Changes for QDART-Conn 40

1 Introduction

1.1 Purpose

This document provides details on the IPQ4019.ILQ.1.1 CSU1 release.

Despite being downloaded from the Qualcomm ChipCode™ portal, the Qualcomm Atheros Support site, or embedded on Equipment received from Qualcomm Atheros, Inc. ("QCA"), the Qualcomm Atheros IPQ4019.ILQ.1.1 CSU1 software release ("SW Package") shall be considered Deliverables and is subject to the terms and conditions of the Qualcomm Atheros, Inc. Limited Use Agreement ("Agreement"). The applicable Use Period, as that term is defined in the Agreement, for the SW Package starts on the Effective Date of your Agreement or the date you received the SW Package, whichever is later, and expires on February 28, 2017 (unless a different Use Period for the SW Package is specified in the Agreement, in which case the Use Period in the Agreement shall prevail). By receiving and/or using the SW Package, you acknowledge and agree that your use of the SW Package is subject to the terms and conditions of the signed Agreement. If you do not agree to the terms of the Agreement, have not signed such Agreement, or have not received the written approval from QCA set forth below, you shall immediately delete the SW Package from all storage media and destroy any and all copies made.

If you wish to utilize the SW Package commercially, you must obtain written approval from QCA, which may be obtained by contacting QCA by email at QCA.Contracts@qualcomm.com or via facsimile to (408) 516-5825, that the SW Package is approved for use under the terms of an applicable definitive agreement in effect between you and QCA or you must enter into a new definitive license agreement relating to the SW Package.

Information published by QCA regarding any third-party information does not constitute a license to use such information or endorsement thereof. QCA provides any such third party information as-is, without any representation, warranty, or indemnity, either express or implied. Use of such information may require a license from a third party under the intellectual property rights of such third party, or a license from QCA or its affiliates under the intellectual property rights of QCA or its affiliates. Users assume all risk of any use of such third party information.

The release provides support for AP.DK01 and AP.DK04 reference design that works in conjunction with the Qualcomm® Atheros Software Development Kit (QSDK) CSU1 release.

- This release is to support IPQ4018, IPQ4028, IPQ4019 and IPQ4029 Version 1.0
- This release also includes an update to the Factory Test Mode driver (aka UTF) for calibration support
- This software release is applicable to APs with any flavor of IPQ4018, IPQ4028, IPQ4019, and IPQ4029 Version 1.0 board regardless of band or front-end

1.2 Identification

This release is:	**IPQ4019.ILQ.1.1 CSU1**
The release version is:	IPQ4019.ILQ.1.1.r1-00056-P-2 This distro can be used to build either Premium or Standard profile.
The Linux Foundation hosted open source label (the CAF_TAG) that corresponds to this release is:	caf_AU_LINUX_QSDK_RELEASE_ARUGULA_BB_CS_TARGET_ALL.3.0.1264.125.xml
Qualcomm ChipCode™ distribution tag (Use this tag to check out code from the git repository):	r00056.2

1.3 Related documentation

In addition to the release notes, refer to the documents for more information on using this release.

Doc Num	Document
80-Y9347-18	*IPQ4018 Access Point SoC Device Specification*
80-Y9347-28	*IPQ4028 Access Point SoC Device Specification*
80-Y9348-28	*IPQ40x8 Device Revision Guide*
80-Y9347-19	*IPQ4019 Access Point SoC Device Specification*
80-Y9347-29	*IPQ4029 Access Point SoC Device Specification*
80-Y9348-29	*IPQ40x9 Device Revision Guide*
80-Y9112-1	*QCA8075 Five-Port 10/100/1000 Mbps Ethernet Transceiver Device Specification*
80-Y9112-2	*QCA8075 5-Port 10/100/1000 Mbps Transceiver Hardware Programming Reference*
80-Y9112-3	*QCA8075 Device Revision Guide*
80-Y9112-4	*QCA8072 Device Revision Guide*
80-Y9112-5	*QCA8075 QCA8072 Co-Layout Design App Note*
80-Y9700-3	*IPQ4018/IPQ4028/IPQ4019/IPQ4029 RF Test User Guide*
80-Y9700-4	*IPQ4018/IPQ4028 Thermal Design Guidelines App Note*
80-Y9700-7	*IPQ4019/IPQ4029 Thermal Design Guidelines*
80-Y9700-5	*IPQ4019/4029 AP.DK04 Hardware Reference Guide*
80-Y9700-6	*IPQ4019/4029 AP.DK04 Setup Guide*
80-Y9798-1	*WLAN AP Test and Calibration User Guide*
80-Y8052-1	*AP 10.4 CLI User Guide*
80-Y8053-1	*AP10.4 Programmers Guide*
80-Y9571-2	*IPQ4018/IPQ4019/IPQ4028/IPQ4029 Profiling with Perf Application Note*
80-Y9571-4	*IPQ40x8 SW User Guide (CoreBSP input, comb EDMA, QRFS, shortcut-fe)*
80-Y9571-5	*IPQ4018/IPQ4019/IPQ4028/IPQ4029 Lauterbach Application Note*
80-Y9571-6	*IPQ4018/IPQ4019/IPQ4028/IPQ4029 Perf Management App Note*
80-Y9571-7	*IPQ4018/IPQ4019/IPQ4028/IPQ4029 Home Switch SDK User Guide*
80-Y9571-8	*IPQ4018/IPQ4019/IPQ4028/IPQ4029 Switch SDK Reference Manual*
80-Y9571-9	*IPQ4018/IPQ4019/IPQ4028/IPQ4029 SSDK Diagnostic Shell User Guide*
80-Y9571-10	*IPQ4018/IPQ4019/IPQ4028/IPQ4029 Switch UCI Command User Guide*
80-Y6399-3	*IPQ40xx QSDK Setup User Guide*
80-Y9572-1	*IPQ4019.ILQ.1.0 Firmware Build Guide*
80-Y8950-19	*IPQ40XX CDT and Flash Partition Configuration Guide*

2 Features

2.1 Features

This release is made up of these major components:

- Quad core ARM A7 processor:

  Quad core ARM A7 CPU 710 MHz
- Offload engines:
  - Crypto acceleration
- Switch
- Wi-Fi 5 GHz
  - 2x2 802.11ac 5 GHz
- Wi-Fi 2 GHz
  - 2x2 802.11n 2.4 GHz
- Comprehensive networking and high speed I/O
  - 1x USB3, 1xUSB2
  - DDR3L 667 MHz clock rate; 1333 Mbps data rate, per bit in BGA package
  - DDR3L 533 MHz clock rate; 1066 Mbps data rate per bit in QFN package
- Standard programming model
  - Linux 3.14.43
  - QSDK

2.2 Release contents

- This release supports Wi-Fi AP, router, repeater, or other designs using the IPQ4018, IPQ4019, IPQ4028, or IP4029 chipsets.

3 Supported Hardware

3.1 AP.DK01 (IPQ4018/IPQ4028), AP.DK04 (IPQ4019/IPQ4029)

This release supports both the APDK01 and APDK04 platforms.

Table 3-1 AP.DK01 and AP.DK04

Hardware Board	Supported	Comments
AP.DK01.1	Yes	IPQ4018 Based
AP.DK01.2	Yes	IPQ4028 Based
AP.DK04.1	Yes	IPQ4019 Based
AP.DK04.2	Yes	IPQ4029 Based

3.2 Hardware reference guides

See the hardware reference guides for more information:

Doc Number	Title
80-Y9700-1	*IPQ4018/4028 AP.DK01 Hardware Reference Guide*
80-Y9700-2	*IPQ4018/4028 AP.DK01 setup guide*
80-Y9700-5	*IPQ4019/4029 AP.DK04 Hardware Reference Guide*
80-Y9700-6	*IPQ4019/4029 AP.DK04 setup guide*

4 Restrictions

Table 4-1 lists the restrictions on the software while using it for testing.

Table 4-1 Restrictions on the software

<table>
<tr><td rowspan="6">Channel Conditions</td><td colspan="2">For peak performance results, the AP, STA, and channel conditions must be configured to:</td></tr>
<tr><td>Attribute</td><td>Value</td></tr>
<tr><td>Image</td><td>See section 1.2 for the release version</td></tr>
<tr><td>AP</td><td>AP-DK04</td></tr>
<tr><td>RSSI of AP at STA</td><td>43 - 53</td></tr>
<tr><td>RSSI of STA at AP</td><td>43 - 53</td></tr>
<tr><td></td><td>Attenuation</td><td>32 dB per chain</td></tr>
<tr><td>ESS</td><td colspan="2">▪ IPQ40xx Ethernet subsystem (ESS) LAN and WAN groups to be tagged with different VLAN IDs. Two VLAN IDs (1 and 2) are reserved for the LAN and WAN groups, respectively. Different VLAN IDs (such as X and Y) can be used through the following configuration:
<code>echo X > /proc/sys/net/edma/edma_default_wtag</code> (for WAN)
<code>echo Y > /proc/sys/net/edma/edma_default_ltag</code> (for LAN)
▪ Jumbo frame is not supported.
▪ Enhanced DMA (EDMA) driver supports S/G through paged array of fragments (nr_frags). It does not support fraglist.
▪ Auto scaling needs to be disabled and CPU operating at highest frequency for performance KPI measurements. In bidirectional KPI case, two flows should have two different RSS hash and different core mapping in order to run in multiple cores.
▪ GRO on GMAC interfaces needs to be disabled for performance KPI measurements. (It is done through a script automatically)
▪ IGMP snooper supports IGMPv3/MLDv2 but it has the report suppression issue when it works in MLDv1. And it does not support the IGMP server in the LAN side.
▪ Shortcut Forwarding Engine (SFE) can only accelerate UDP IPSEC downlink case.</td></tr>
<tr><td>Wi-Fi</td><td colspan="2">Qualcomm Wireless Repeater AP (QWRAP) design limitations
▪ In this release, the maximum number of Ethernet/Wireless clients currently supported is 26. Any attempt to initialize more than 26 QWRAP clients will cause the system to crash. This limitation is expected to be addressed in future releases
▪ Supported security modes: Open, WPA2-PSK, WEP, AES, TKIP
▫ Other modes such as WDS are not supported when QWRAP/Proxy mode is enabled.
▫ Dynamic switching of modes between Proxy STA and other modes is not supported.
▪ Enterprise authentication modes are not supported.
▪ Roaming supports STA roams among WRAPs, but not between WRAP and Root AP.
▪ For a list of protocols supported in payload when MAT is used, see the AP 10.4 Programmers Guide – Wireless LAN.
▪ Cascading QWRAP is supported only with a unique MAC address defined for each proxy station in each QWRAP AP level. More than one QWRAP level is not extensively deployed and tested.
▪ QWRAP + WAPI encryption mode is not supported</td></tr>
</table>

5 Power Management

ARM A7 CPU Frequency Scaling	CPU frequency can be scaled up and down based on CPU load. ▪ OnDemand governor is the default CPU Freq governor ▪ Following are supported CPU Frequencies for this release ▫ 710 MHz ▫ 500 MHz ▫ 200 MHz ▫ 48 MHz
Dynamic clock gating is enabled	Software detects the inactivity in parts of the system and shuts down the clock to these parts accordingly
DDR and NOC	DDR runs at 533 MHz or 667 MHz depending on IP40xx package SNOC at 200 MHz PCNOC at 100 MHz Frequency Scaling is not supported for DDR and NOC

Use the following commands, as example, to switch governor and CPU frequency:

```
echo userspace > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
echo 710000 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_setspeed
```

6 System Frequencies

Table 6-1 shows the system frequencies for this release:

Table 6-1 System frequencies

	Min	Max
Cortex A7	48 MHz[1]	710 MHz
SNOC	200 MHz	200 MHz
PCNOC	100 MHz	100 MHz
DDR		
IPQ40x9	667 MHz	667 MHz
IPQ40x8	533 MHz	533 MHz

As indicated in section 5, the power management features change these frequencies as per-system load. Table 6-1 indicates the frequencies at which system operates when system load is at minimum and maximum respectively.

- Cortex A7 frequencies are changed based on CPU load
- DDR frequency always in the max frequency through the system ON state

[1] scaling_min_freq is set to 200 MHz

7 Performance Data

7.1 AP.DK04 WLAN Performance

7.1.1 WLAN performance results

Table 7-1 AP.DK04 wireless throughput in bridging mode (WDS)

Traffic	5 GHz - 2x2 VHT80 Throughput (Mbps)	CPU %	CPU0	CPU1	CPU2	CPU3
TCP UL	689	23.60	0.91	9.49	0.78	83.23
TCP DL	687	20.06	0.22	1.97	0.04	78
TCP Bi-Di	706	21.36	0.23	5.63	0.58	78.98
UDP UL	720	21.72	0.38	9.42	0.57	76.5
UDP DL	718	23.87	0.2	0.89	0.59	93.79
UDP Bi-Dir	727	23.70	0.75	5.61	0.24	88.2

Traffic	2.4 GHz - 2x2 VHT40 Throughput (Mbps)	CPU %	CPU0	CPU1	CPU2	CPU3
TCP UL	309	12.02	4.44	0.03	42.49	1.13
TCP DL	311	10.12	1.11	0.25	38.72	0.41
TCP Bi-Di	314	10.65	2.68	0	39.22	0.7
UDP UL	331	10.29	4.3	0.51	36.33	0.01
UDP DL	331	13.85	0.53	0.13	54.74	0
UDP Bi-Dir	333	16.26	1.13	0.22	63.67	0

Traffic	DBDC Throughput (Mbps)	CPU %	CPU0	CPU1	CPU2	CPU3
TCP UL	1001	37.04	6.32	10.66	44.98	86.18
TCP DL	1004	30.83	0.96	0.8	40.8	80.74
TCP Bi-Di	1019	33.56	3.6	5.77	42.12	82.74
UDP UL	1052	34.14	5.84	11.08	37.77	81.86
UDP DL	1051	42.11	1.35	0.49	71.79	94.79
UDP Bi-Dir	1053	36.86	3.98	5.24	53.08	85.15

Hardware and Software Details	
AP, STA: 2 GHz 5 GHz STA Hardware	IPQ40xx 2x2 2 GHz and 2x2 5 GHz
SW	IPQ4019.ILQ.1.1.r1-00056-P-2
SETUP	Cabled
Mode	WDS
2 GHz Channel	6, 11NGHT40+
5 GHz Channel	149, 11ACVHT80
Security	WPA2 PSK
No of Ethernet backbones on AP	2
Traffic Duration	120 seconds

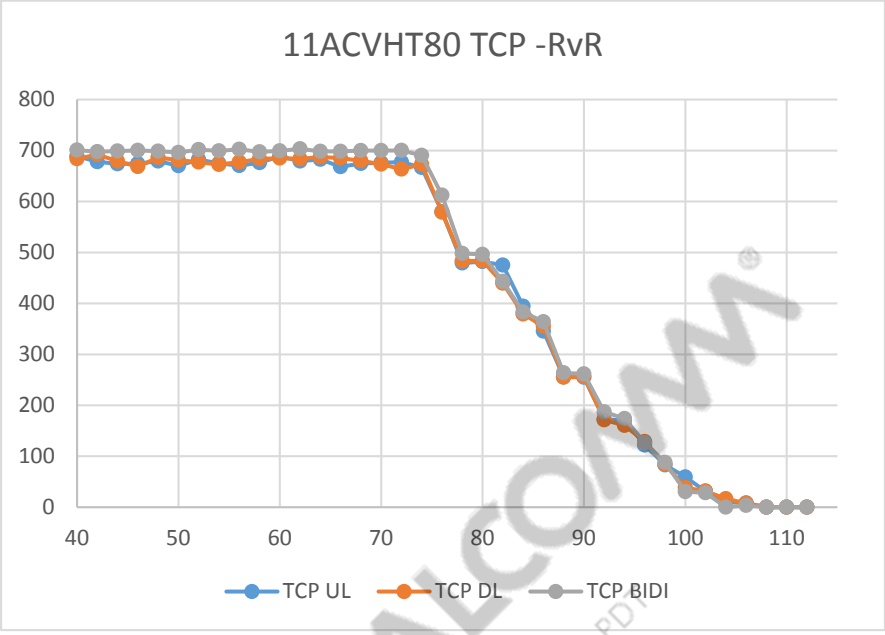
Table 7-2 AP.DK04 wireless throughput in wired QWRAP mode

Traffic	5 GHz - 2x2 VHT80 Throughput (Mbps)	CPU %	CPU0	CPU1	CPU2	CPU3
TCP DL	689	55.41	57.73	38.03	41.14	84.73
TCP UL	684	60.70	49.24	55.98	41.88	95.7
TCP Bi-Di	705	58.01	64.02	43.84	34.28	89.9
UDP DL	720	58.70	54.27	43.24	37.33	99.95
UDP UL	717	58.46	53.06	53.85	37.22	89.71
UDP Bi-Dir	726	58.29	50.33	57.23	31.24	94.35

Traffic	2.4 GHz - 2x2 VHT40 Throughput (Mbps)	CPU %	CPU0	CPU1	CPU2	CPU3
TCP DL	318	42.77	44.36	46.87	48.78	31.06
TCP UL	317	47.50	54.06	43.43	59.35	33.14
TCP Bi-Di	321	43.72	53.96	40.26	55.22	25.43
UDP DL	331	58.07	60.57	43.25	97.99	30.47
UDP UL	331	44.61	40.34	52.71	57.78	27.61
UDP Bi-Dir	333	53.28	45.84	55.28	79.5	32.51

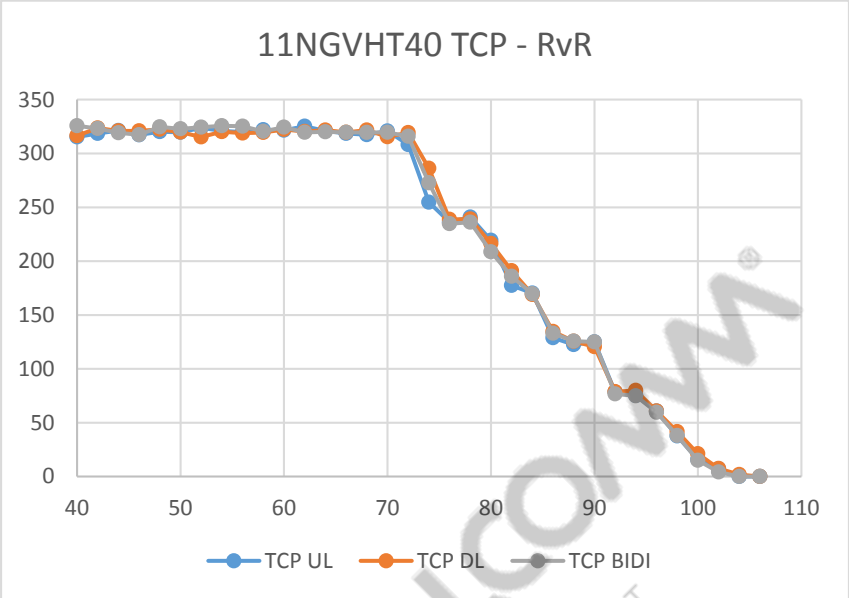
Hardware and Software Details	
AP, STA: 2 GHz 5 GHz STA Hardware	IPQ40xx 2x2 2 GHz and 2x2 5 GHz
SW	IPQ4019.ILQ.1.1.r1-00056-P-2
SETUP	Cabled
Mode	QWRAP
2 GHz Channel	6, 11NGHT40+
5 GHz Channel	149, 11ACVHT80
Security	WPA2 PSK
No of Ethernet backbones on AP	2
Traffic Duration	120 seconds

Table 7-3 WDS – TCP 11ACVHT80



Hardware and Software Details	
AP, STA: 2 GHz 5 GHz STA Hardware	IPQ40xx 2x2 2 GHz and 2x2 5 GHz
SW	IPQ4019.ILQ.1.1.r1-00056-P-2
SETUP	Cabled
Mode	WDS
5 GHz Channel	149, 11ACVHT80
Security	Open
No of Ethernet backbones on AP	2
Traffic Duration	120 seconds

Table 7-4 WDS – TCP 11NGVHT40



Hardware and Software Details	
AP, STA: 2 GHz 5 GHz STA Hardware	IPQ40xx 2x2 2 GHz and 2x2 5 GHz
SW	IPQ4019.ILQ.1.1.r1-00056-P-2
SETUP	Cabled
Mode	WDS
2 GHz Channel	6, 11NGVHT40
Security	Open
No of Ethernet backbones on AP	2
Traffic Duration	120 seconds

7.1.2 DBDC performance test setup

Throughput measurement tool:	Ixia Chariot application with high performance throughput script (default file size).	
Test Duration:	2 minutes	
Hardware:	AP	AP.DK04 (APUT)
	STA1	AP.DK04 (WDS 2G STA)
	STA2	AP.DK04 (WDS 5G STA)
Channel:	5 GHz	149 BW80
	2.4 GHz	6 BW40

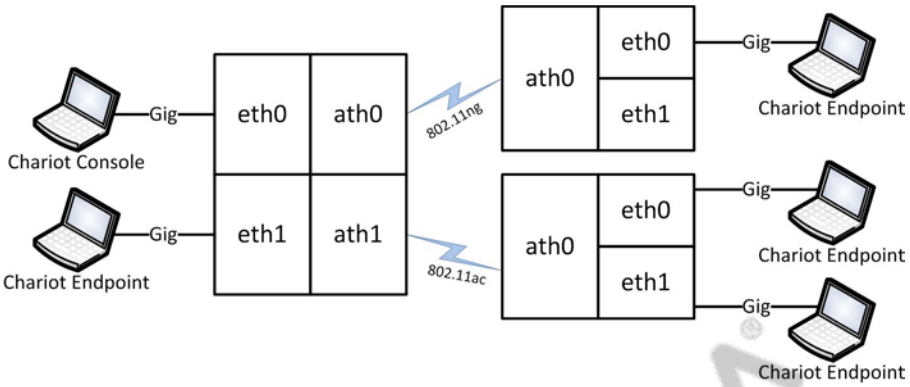


Figure 7-1 Ethernet to WLAN 802.11ac + 802.11n

For the 5 GHz interface, either two laptops or a desktop PC with two 1-GHz Ethernet NIC cards may be used to pump traffic through both LAN and WAN ports. Both eth0 and eth1 must be added to the BR-LAN bridge using this command:

```
# brctl addif br-lan eth0
```

Test Setup The test results were obtained using Dakota AP.DK04 in cabled shield boxes (with attenuation between transmitter and receiver). The total attenuation for each chain was ~ 30 dB. The cabled tests results are based on 11NGHT40 for 2.4 GHz and the channel used is:

- 5G: Channel 149 (5745 MHz)
- 2G: Channel 6 (2437 MHz)

See the following figure for an example based on the internal Qualcomm Atheros test lab setup.

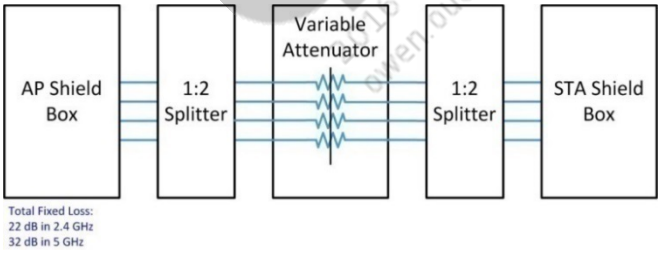


Figure 7-2 Example two-node test setup

7.1.3 WLAN setup steps

AP Setup

1. Flash the images to the AP according to the instructions described in section 9.4.
2. Type **iwconfig** to check that no wireless interfaces are up by default.
3. Use these configuration commands to the get wireless interface up and running:

AP - DUT

```
wifi detect > /etc/config/wireless
uci commit wireless
uci revert -P /var/state wireless
wifi
```

```
uci delete wireless.@wifi-iface[1]
uci delete wireless.@wifi-iface[0]
uci add wireless wifi-iface
uci add wireless wifi-iface
uci set wireless.@wifi-iface[0].network=lan
uci set wireless.@wifi-iface[0].wds=1
uci set wireless.@wifi-iface[0].ssid=UCI2G
uci set wireless.@wifi-iface[0].encryption=none
uci set wireless.@wifi-iface[0].device=wifi0
uci set wireless.@wifi-iface[0].nss=2
uci set wireless.@wifi-iface[0].mode=ap
usi set wireless.@wifi-iface[0].disablecoext='1'
uci set wireless.@wifi-iface[1].network=lan
uci set wireless.@wifi-iface[1].mode=ap
uci set wireless.@wifi-iface[1].wds=1
uci set wireless.@wifi-iface[1].ssid=UCI5G
uci set wireless.@wifi-iface[1].encryption=none
uci set wireless.@wifi-iface[1].device=wifi1
uci set wireless.@wifi-iface[1].nss=2
uci set wireless.@wifi-device[0].hwmode=11ng
uci set wireless.@wifi-device[0].disabled=0
uci set wireless.@wifi-device[0].htmode=HT40
uci set wireless.@wifi-device[0].channel=6
uci set wireless.@wifi-device[0].txchainmask=3
uci set wireless.@wifi-device[0].rxchainmask=3
uci set wireless.@wifi-device[1].hwmode=11ac
uci set wireless.@wifi-device[1].disabled=0
uci set wireless.@wifi-device[1].htmode=VHT80
uci set wireless.@wifi-device[1].channel=149
uci set wireless.@wifi-device[1].txchainmask=3
uci set wireless.@wifi-device[1].rxchainmask=3
uci commit wireless
wifi
```

4. If encryption is to be turned ON, add these commands:
   ```
   uci set wireless.@wifi-iface[0].encryption=psk2
   uci set wireless.@wifi-iface[0].key=abcdefgh
   uci set wireless.@wifi-iface[0].wpa_group_rekey=600
   uci commit wireless
   wifi down; wifi up
   ```
5. The contents of the **etc/config/wireless** file should be:
   ```
   *** device - type, macaddr, channel, hwmode, htmode ****
   *** iface - ssid, wds, encryption, txchainmask, rxchainmask, mode, disabled, device, XX ****
   ```
6. On network restart, run the **iwconfig** command to check for the following:
   a. For 5 GHz radio: Mode: Master, Bit rate: 866.7 GBps, Freq: 5.5 GHz
   b. For 2 GHz boards: Mode: Master, Bit rate: 400 Mbps, Freq: 2.437 GHz
7. Change the ipaddress using the command **ifconfig br-lan 192.168.1.15**.

At this point the AP is up.

Client Setup

1. Flash the image to the client according to the instructions described in section 9.4.

2. Type **iwconfig** to verify that no wireless interfaces are up by default.
3. Copy these configuration commands to get the wireless interface up and running:

2.4 GHz WDS Client	5 GHz WDS Client
wifi detect > /etc/config/wireless uci commit wireless uci revert -P /var/state wireless wifi uci delete wireless.@wifi-iface[1] uci delete wireless.@wifi-iface[0] uci add wireless wifi-iface uci set wireless.@wifi-device[0].hwmode=11ng uci set wireless.@wifi-iface[0].network=lan uci set wireless.@wifi-iface[0].mode=sta uci set wireless.@wifi-iface[0].wds=1 uci set wireless.@wifi-iface[0].ssid=UCI2G uci set wireless.@wifi-iface[0].encryption=none uci set wireless.@wifi-iface[0].device=wifi0 uci set wireless.@wifi-iface[0].disablecoext='1' uci set wireless.@wifi-device[0].txchainmask=3 uci set wireless.@wifi-device[0].rxchainmask=3 uci set wireless.@wifi-device[0].htmode=HT40 uci set wireless.@wifi-device[0].disabled=0 uci set wireless.@wifi-device[0].channel=6 uci commit wireless wifi	wifi detect > /etc/config/wireless uci commit wireless uci revert -P /var/state wireless wifi uci delete wireless.@wifi-iface[1] uci delete wireless.@wifi-iface[0] uci add wireless wifi-iface uci set wireless.@wifi-device[1].hwmode=11ac uci set wireless.@wifi-device[1].disabled=0 uci set wireless.@wifi-device[1].txchainmask=3 uci set wireless.@wifi-device[1].rxchainmask=3 uci set wireless.@wifi-iface[0].mode=sta uci set wireless.@wifi-iface[0].wds=1 uci set wireless.@wifi-iface[0].ssid=UCI5G uci set wireless.@wifi-iface[0].encryption=none uci set wireless.@wifi-iface[0].device=wifi1 uci commit wireless uci set network.lan.ipaddr=192.168.1.30 uci set network.lan.netmask=255.255.255.0 uci commit network wifi

4. If encryption is to be turned ON, add the following commands:

   uci set wireless.@wifi-iface[0].encryption=psk2
   uci set wireless.@wifi-iface[0].key=abcdefgh
   uci set wireless.@wifi-iface[0].wpa_group_rekey=600
   uci commit wireless
   wifi down; wifi up

5. The contents of the **etc/config/wireless** file should be:

   ** device - type, macaddr, channel, hwmode, htmode **
   ** iface - ssid, wds, encryption, txchainmask, rxchainmask, mode(sta), disabled, device, XX **

6. On restart, type the **iwconfig** command to check for the following:
   - For 5 GHz radio: Mode: Master, Bit rate: 866.7 GBps, Freq: 5.5 GHz
   - For 2 GHz radio: Mode: Master, Bit rate: 400 Mbps, Freq: 2.437 GHz

NOTE: At 5.5 GHz, two scan cycles are happening, so it comes up initially at 2.412 GHz, then moves to 5.5 GHz, and finally associates at 5.5 GHz. This association takes approximately 45 seconds.

7. After network restart, change the IP address using the command **ifconfig br-lan 192.168.1.17**. At this point the STA is up and associated.
8. Open cmd and ping **192.168.1.22** from 192.168.1.11.
9. Open Chariot, run uplink downlink - TCP and then UDP.

7.2 AP.DK04 wired performance

Measurements are taken with IPQ4019.ILQ.1.1.r1-00056-P-2.

7.2.1 Wired and SAMBA performance results

Table 7-5 RFC2544 peak throughput for wired use case using SFE

DHCP/ PPPoE	IPv4/ IPv6	TCP/UDP	Frame Size (Bytes)	Direction	Routing/ Bridge	Throughput Observed (Mbps)
DHCP	IPv4	TCP	1518	Bi-Di	Routing + NAT	1856
DHCP	IPv4	UDP	1518	Bi-Di	Routing + NAT	1973
DHCP	IPv4	UDP	74	Bi-Di	Routing + NAT	126
DHCP	IPv6	TCP	1518	Bi-Di	Routing	1827
DHCP	IPv6	UDP	1518	Bi-Di	Routing	1973
DHCP	IPv6	UDP	94	Bi-Di	Routing	118

Table 7-6 Peak RFC2544 Bi-Di throughput using HNAT for all frame sizes

Bytes	Achieved Throughput (Mbps)	CPU (Average SMP %)
74	1574	<2%
128	1729	<2%
256	1855	<2%
512	1924	<2%
1024	1961	<2%
1280	1969	<2%
1518	1973	<2%

Table 7-7 Storage KPI

Client	File System	Read/ Write	Device	Throughput (MBps)	Quad Core Utilization (% SMP) CS
Samba - Linux client	FAT32	Read	USB3	77	21
Samba - Linux client	FAT32	Write	USB3	29	30
Samba - Win client	FAT32	Read	USB3	45	16
Samba - Win client	FAT32	Write	USB3	18	27

Table 7-8 IPSec KPI

	Auth	Cipher	Pkt Size	Throughput (Mbps)	CPU Utilization
IPsec 2xWLAN to 2xWAN Max SAs: 16 (cached + uncached)	SHA1	AES128	1438	170	44.00%
IPsec 2xWLAN to 2xWAN Max SAs: 16 (cached + uncached)	SHA1	AES256	1438	169	42.00%

7.2.2 SFE/HNAT Routing test setup

Figure 7-3 shows the IXIA traffic generator.

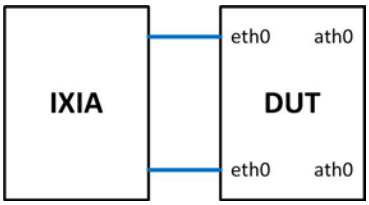


Figure 7-3 Spirent/IXIA traffic generator

Test description

- The SFE performance can be measured using the IXIA traffic generator. The IXIA traffic generator supports tools to measure TCP/UDP performance.
- The performance (as defined in RFC 2544) can be measured using RFC 2544 measurement tool available as part of the IXIA IxNetwork utility. RFC 2544 measurement tool searches for FPS/Mbps rate at which DUT does not drop a single packet for the duration of packet forwarding performance test. This measurement is done for several packet sizes − 1514, 1280, 1024, 512, 256, 128 and 64.

NOTE: The IxNetwork allows a least packet size of 74 bytes for IPv4 and 86 bytes for IPv6 instead of 64 bytes.

NOTE: Do a small run initially to ensure that the fast path rule has been pushed, before doing a full RFC 2544 run.

Routing mode − Test setup configuration

```
(IXIA/SPIRENT) Port1---------eth1-DUT-eth0-------Port2(IXIA/SPIRENT)
IXIA/SPIRENT ) Port1  IPv4  - 192.168.1.2/24,  IPv6 - 4aaa::2/64
IXIA /SPIRENT) Port2  IPv4  - 172.16.10.20/16, IPv6 - 5aaa::2/64
```

CPU scaling configuration

```
echo performance > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
echo performance > /sys/devices/system/cpu/cpu1/cpufreq/scaling_governor
echo performance > /sys/devices/system/cpu/cpu2/cpufreq/scaling_governor
echo performance > /sys/devices/system/cpu/cpu3/cpufreq/scaling_governor
```

CPU frequency configuration

```
echo 710000 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_min_freq
echo 710000 > /sys/devices/system/cpu/cpu1/cpufreq/scaling_min_freq
echo 710000 > /sys/devices/system/cpu/cpu2/cpufreq/scaling_min_freq
echo 710000 > /sys/devices/system/cpu/cpu3/cpufreq/scaling_min_freq
```

DUT configuration − Routing mode

```
uci set network.loopback=interface
uci set network.loopback.ifname=lo
uci set network.loopback.proto=static
uci set network.loopback.ipaddr=127.0.0.1
uci set network.loopback.netmask=255.0.0.0
uci set network.lan=interface
uci set network.lan.ifname='eth1'
uci set network.lan.type=bridge
uci set network.lan.proto=static
uci set network.lan.ipaddr=192.168.1.1
uci set network.lan.netmask=255.255.255.0
uci set network.lan.ip6addr=4aaa::1/64
uci set network.wan=interface
uci set network.wan.ifname=eth0
uci set network.wan.proto=static
uci set network.wan.ipaddr=172.16.10.10
uci set network.wan.netmask=255.255.255.0
uci set network.wan.ip6addr=5aaa::1/64
uci commit
/etc/init.d/network restart
```

Firewall configuration

```
uci add firewall redirect
uci set firewall.@redirect[0]=redirect
uci set firewall.@redirect[0].target=DNAT
uci set firewall.@redirect[0].src=wan
uci set firewall.@redirect[0].dest=lan
uci set firewall.@redirect[0].proto='tcp udp'
uci set firewall.@redirect[0].name=DNAT
uci set firewall.@redirect[0].src_ip=172.16.10.20
uci set firewall.@redirect[0].src_dip=172.16.10.10
uci set firewall.@redirect[0].dest_ip=192.168.1.2
uci commit
/etc/init.d/firewall restart
```

7.2.3 IPsec Ethernet-Ethernet performance setup

Figure 7-4 shows the IPsec Ethernet-Ethernet performance setup.

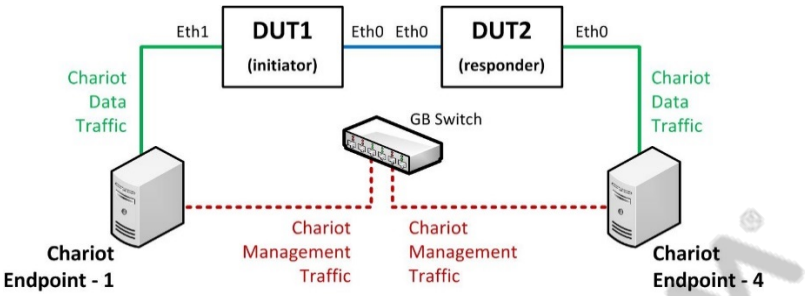


Figure 7-4 IPsec Ethernet – Ethernet performance setup

Where the Chariot Endpoint is a HP Z220 Workstation with:

- Windows 7
- Intel Core i7
- Multiple Ethernet Ports

7.2.4 Samba/Storage wired performance setup

Figure 7-5 shows the file operations over LAN.

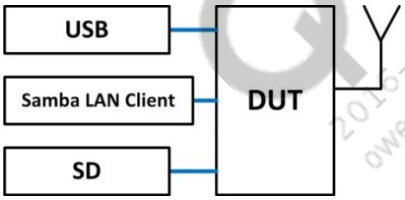


Figure 7-5 File operations over LAN

7.2.5 Raw crypto performance

For raw crypto testing, an internal tool is used where the user can configure various parameters to encrypt or decrypt packets. Based on the information provided by the user, the crypto tool prepares the packets for encryption/decryption, sends them to the SFE crypto driver and receives completions. Time taken to complete this process is used to determine the raw crypto throughput.

Key parameters configured for the measurements

Number of requests in the first pass:	200
Number of loops of request pass:	100
Cipher algorithm:	AES128, AES256
Authentication algorithm:	SHA1, SHA256
Cipher length and Auth length used:	1500, 1024, 256 bytes
Cipher skip and Auth skip:	16 bytes
Key_len:	16 bytes for AES128, 32 bytes for AES256
Hash_len:	12 bytes for SHA1, 20 bytes for SHA256

8 Known Issues

This section lists the open issues and fixed issues for this release.

Table 8-1 Open issues for this release

	Title	Customer Impact
1.	5 GHz WF bump observed at -40dBm and CM=3,especially for FF skew part	Debug in progress.
2.	Observed Backtrace (insmod failed) during module insertion	Insmod failures – Caused by repeated Wi-Fi up/down sequence during long hour regression test. Not seen in normal operation.
3.	Observing page allocation failure while running regression	Observed on low memory footprint Standard Profile with 128 MBytes RAM. It is a result of memory fragmentation caused by doing repeated Wi-Fi up/down during the 8 – 12 hour regression test run

Table 8-2 Fixed issues for this release

	Title	Customer Impact
1.	DUT crashes upon running continuous channel change test	Not a typical customer use case. Root caused to a race condition caused when the channel change command is issued in a loop. Issue happens specifically in DFS channels. Fix will be available in update release

9 Download and Build Instructions

Released software will be distributed via ChipCode portal for proprietary based code, and through the Linux Foundation for open source.

Install QSDK and generate the image:	See section:
2. Obtain the Qualcomm Atheros proprietary code from ChipCode; other components are obtained from external websites by QSDK while building the default configuration	9.2
3. Generate the firmware by reassembling the code, building QSDK, and generating a complete firmware image	9.3
4. Flash the target	9.4

9.1 QSDK profiles

QSDK supports two profiles: **Premium** and **Standard.** The premium profile is targeted for 32 MB flash and 256 MB DDR configuration, whereas the standard profile is a subset of the premium profile features to support 16 MB flash and 128 MB DDR configuration. To fit the image in 16 MB, the following features have been disabled compared to the premium profile:

- CD Router certification
- Bluetooth
- Audio
- Video
- IPSec

You will be receiving the Premium distro for this release. Although this Premium distro can build either Premium or Standard profiles, DK01 boards are configured to boot from the 32Mbyte flash; therefore board changes are needed before a standard profile image can be flashed and booted.

9.2 Download code

9.2.1 Download packages available through ChipCode

Qualcomm Atheros proprietary code is available from ChipCode. The top-level structure of the source code is given in Table 9-1.

Table 9-1 Source code top-level structure

Images	SRC/BIN	Source/Image Path
NHSS	SRC	apss_proc\out\proprietary
BOOT	BIN	boot_images\build\ms\bin\40xx\emmc boot_images\build\ms\bin\40xx\misc boot_images\build\ms\bin\40xx\nand boot_images\build\ms\bin\40xx\norplusnand boot_images\build\ms\bin\40xx\nor
TZ	BIN	trustzone_images\build\ms\bin\ MAZAANAZ
CNSS.BL	BIN	cnss_proc\bin cnss_proc\src

NOTE: A web/GUI interface and a secure git server both allow access to this code. Browse available packages and obtain the download URL at https://chipcode.qti.qualcomm.com/ (see https://chipcode.qti.qualcomm.com/helpki/cloning-code-from-a-repository for more information).

Customers are recommended to obtain the proprietary code with 'git clone'.

See https://chipcode.qti.qualcomm.com/helpki for more information on installation and configuration of the correct version of git and OpenSSL on both Windows and Linux platforms that is required to support the authentication methods used by ChipCode.

9.2.2 Packages downloaded from external websites

The following components are downloaded by QSDK while building the default configuration for the profiles defined in section 9.3. QSDK may be further customized to download additional components; this list only contains the components that are necessary for at least one of the QSDK 2.0 default profiles. This list does not include the packages obtained from ChipCode as described in section 9.3 (firmware generation).

Table 9-2 Packages available from external sites

Package
alljoyn-14.12.00a-src.tar.gz
alljoyn-services-14.12.00-src.tar.gz
alljoyn-services-14.12.00-src.tar.gz
alljoyn-services-14.12.00-src.tar.gz
alljoyn-services-14.12.00-src.tar.gz
alsa-lib-1.0.27.2.tar.bz2
alsa-utils-1.0.27.2.tar.bz2
arptables-v0.0.4.tar.gz

Package
binutils-.tar.bz2
bluez-5.27.tar.xz
bridge-utils-1.5.tar.gz
busybox-1.19.4.tar.bz2
bzip2-1.0.6.tar.gz
dbus-1.8.8.tar.gz
dosfstools-3.0.12.tar.gz
dropbear-2014.63.tar.bz2
e2fsprogs-1.42.4.tar.gz
elfutils-0.155.tar.bz2
ethtool-3.7.tar.xz
expat-2.0.1.tar.gz
file-5.11.tar.gz
firewall-2014-03-20-93aea77092b0c178fefe3ab95fc040534eda90a3.tar.gz
fstools-2014-04-07-a1f48fc0444f5c3c44ee6ef1005cd8da65decefd.tar.gz
glib-2.37.7.tar.xz
gmp-5.1.3.tar.bz2
i2c-tools-2013-12-15-r6204.tar.gz
iperf-2.0.5.tar.gz
iproute2-3.11.0.tar.xz
iptables-1.4.21.tar.bz2
iputils-s20101006.tar.bz2
dhcp-4.2.4.tar.gz
libdaemon-0.14.tar.gz
libffi-3.0.11.tar.gz
json-c-0.11.tar.gz
libnfnetlink-1.0.1.tar.bz2
libnl-bf-g7d47666eb3c414feb8901970d35b96461214c2bf.tar.gz
readline-6.2.tar.gz
libubox-2014-03-18-4f44401ae8d23465261cef80b87630ffccd5a864.tar.gz
libusb-1.0.9.tar.bz2
libusb-compat-0.1.4.tar.bz2
linux-firmware-20150106.tar.gz
lua-5.1.5.tar.gz
lzo-2.06.tar.gz
mcproxy-1.1.0.y.tar.bz2
mdadm-3.2.5.tar.xz
miniupnpd-1.8.20130426.tar.gz
mtd-utils-1.5.0.tar.gz
ncurses-5.9.tar.gz
netifd-2014-04-07-5df59a6cf9fd307378bd5cbea809a22f6de9f33d.tar.gz
ntfs-3g_ntfsprogs-2011.4.12.tgz

Package
odhcp6c-2014-03-31.tar.bz2
odhcpd-2014-04-06.tar.bz2
openssl-1.0.1j.tar.gz
openswan-2.6.41.tar.gz
opkg-9c97d5ecd795709c8584e972bfdf3aee3a5b846d.tar.gz
perl-5.10.0.tar.gz
pm-utils-1.4.1.tar.gz
ppp-2.4.5.tar.gz
procd-2015-01-25-7e6045fe4558e442e97e4a8967f08689d53f2cc4.tar.gz
pure-ftpd-1.0.36.tar.bz2
quagga-0.99.21.tar.gz
radvd-1.9.1.tar.gz
rp-pppoe-3.11.tar.gz
samba-3.6.25.tar.gz
sysstat-10.1.7.tar.bz2
tftp-hpa-0.48.tar.gz
trace-cmd-v2.2.1.tar.gz
u-boot-2013.10.tar.bz2
ubox-2014-03-27-1d9d2e6ae99c9ba72d1bc40e554d5f422c9b9196.tar.gz
ubus-2014-03-18-1d5ac421a5b3dca60562e876ba70d0c2fe46b3d2.tar.gz
uci-2014-02-18.1.tar.gz
uClibc++-0.2.4.tar.bz2
udev-173.tar.bz2
uhttpd-2014-03-22-31b459cb1d0ba3280cbc3fc06ce6fab903c07da6.tar.gz
util-linux-2.24.1.tar.xz
wide-dhcpv6-20080615.tar.gz
wireless_tools.29.tar.gz
xl2tpd-1.3.1.tar.gz
xtables-addons-1.45.tar.xz
zlib-1.2.8.tar.gz
alljoyn-14.12.00a-src.tar.gz
alljoyn-services-14.12.00-src.tar.gz
alljoyn-services-14.12.00-src.tar.gz
alljoyn-services-14.12.00-src.tar.gz
alljoyn-services-14.12.00-src.tar.gz
alsa-lib-1.0.27.2.tar.bz2
alsa-utils-1.0.27.2.tar.bz2
arptables-v0.0.4.tar.gz
binutils-.tar.bz2
bluez-5.27.tar.xz
bridge-utils-1.5.tar.gz
busybox-1.19.4.tar.bz2

Package
bzip2-1.0.6.tar.gz
dbus-1.8.8.tar.gz
dosfstools-3.0.12.tar.gz
dropbear-2014.63.tar.bz2
e2fsprogs-1.42.4.tar.gz
elfutils-0.155.tar.bz2
ethtool-3.7.tar.xz
expat-2.0.1.tar.gz
file-5.11.tar.gz
firewall-2014-03-20-93aea77092b0c178fefe3ab95fc040534eda90a3.tar.gz
fstools-2014-04-07-a1f48fc0444f5c3c44ee6ef1005cd8da65decefd.tar.gz
glib-2.37.7.tar.xz
gmp-5.1.3.tar.bz2
i2c-tools-2013-12-15-r6204.tar.gz
iperf-2.0.5.tar.gz
iproute2-3.11.0.tar.xz
iptables-1.4.21.tar.bz2
iputils-s20101006.tar.bz2
dhcp-4.2.4.tar.gz
libdaemon-0.14.tar.gz
libffi-3.0.11.tar.gz
json-c-0.11.tar.gz
libnfnetlink-1.0.1.tar.bz2
libnl-bf-g7d47666eb3c414feb8901970d35b96461214c2bf.tar.gz
readline-6.2.tar.gz
libubox-2014-03-18-4f44401ae8d23465261cef80b87630ffccd5a864.tar.gz
libusb-1.0.9.tar.bz2
libusb-compat-0.1.4.tar.bz2
linux-firmware-20150106.tar.gz
lua-5.1.5.tar.gz
lzo-2.06.tar.gz
mcproxy-1.1.0.y.tar.bz2
mdadm-3.2.5.tar.xz
miniupnpd-1.8.20130426.tar.gz
mtd-utils-1.5.0.tar.gz
ncurses-5.9.tar.gz
netifd-2014-04-07-5df59a6cf9fd307378bd5cbea809a22f6de9f33d.tar.gz
ntfs-3g_ntfsprogs-2011.4.12.tgz
odhcp6c-2014-03-31.tar.bz2
odhcpd-2014-04-06.tar.bz2
openssl-1.0.1j.tar.gz
openswan-2.6.41.tar.gz

Package
opkg-9c97d5ecd795709c8584e972bfdf3aee3a5b846d.tar.gz
perl-5.10.0.tar.gz
pm-utils-1.4.1.tar.gz
ppp-2.4.5.tar.gz
procd-2015-01-25-7e6045fe4558e442e97e4a8967f08689d53f2cc4.tar.gz
pure-ftpd-1.0.36.tar.bz2
quagga-0.99.21.tar.gz
radvd-1.9.1.tar.gz
rp-pppoe-3.11.tar.gz
samba-3.6.25.tar.gz
sysstat-10.1.7.tar.bz2
tftp-hpa-0.48.tar.gz
trace-cmd-v2.2.1.tar.gz
u-boot-2013.10.tar.bz2
ubox-2014-03-27-1d9d2e6ae99c9ba72d1bc40e554d5f422c9b9196.tar.gz
ubus-2014-03-18-1d5ac421a5b3dca60562e876ba70d0c2fe46b3d2.tar.gz
uci-2014-02-18.1.tar.gz
uClibc++-0.2.4.tar.bz2

9.3 Firmware generation

9.3.1 Reassemble the code

The first step is to generate the QSDK framework by re-assembling the code from ChipCode and The Linux Foundation. The example given in this section assumes that all packages listed in section 9.2.1 have been obtained via eg git clone and placed in the top-level directory:

All Packages:

These files are fetched from ChipCode using git and should be copied to the working the QSDK top-level directory:	Local directory path to files fetched by git from ChipCode:
qsdk-qca-wifi-10.4-3.0.1264.125 qsdk-qca-wlan-10.4-3.0.1264.125 qsdk-whc-3.0.1264.125 qsdk-ieee1905-security-3.0.1264.125	apss_proc\out\proprietary\Wifi
qca-lib-3.0.1264.125.tar.bz2 qca-mcs-apps-3.0.1264.125.tar.bz2	apss_proc\out\proprietary\QSDK-Base
qca-wifi-fw-IPQ4019_hw_1-WLAN.BL.3.1-00084-S-1.tar.bz2	cnss_proc\bin\IPQ4019/hw.1
qca-wifi-fw-src-component-cmn-WLAN.BL.3.1-00084-S-1.tgz qca-wifi-fw-src-component-halphy_tools-WLAN.BL.3.1-00084-S-1.tgz	cnss_proc\src\components

Generate the QSDK framework:

```
$ repo init -u git://codeaurora.org/quic/qsdk/releases/manifest/qstak -b
release -m
caf_AU_LINUX_QSDK_RELEASE_ARUGULA_BB_CS_TARGET_ALL.3.0.1264.125.xml --repo-
url=git://codeaurora.org/tools/repo.git --repo-branch=caf-stable
$ repo sync
$ mkdir -p qsdk/dl
```

```
$ cp -rf apss_proc/out/proprietary/Wifi/qsdk-qca-wifi-10.4-3.0.1264.125/* qsdk/
$ cp -rf apss_proc/out/proprietary/Wifi/qsdk-qca-wlan-10.4-3.0.1264.125/* qsdk/
$ cp -rf apss_proc/out/proprietary/Wifi/qsdk-ieee1905-security-3.0.1264.125/*
   qsdk/
$ tar -xjf apss_proc/out/proprietary/QSDK-Base/qca-lib-3.0.1264.125.tar.bz2 -C
   qsdk
$ tar -xjf apss_proc/out/proprietary/QSDK-Base/qca-mcs-apps-3.0.1264.125.tar.bz2
   -C qsdk
$ tar -xzvf qca-wifi-fw-src-component-cmn-WLAN.BL.3.1-00084-S-1.tgz
$ mv include qsdk/qca/src/qca-wifi-10.4/fwcommon
$ tar -xzvf qca-wifi-fw-src-component-halphy_tools-WLAN.BL.3.1-00084-S-1.tgz
$ mv wlan/halphy_tools qsdk/qca/src/qca-wifi-10.4
$ cp cnss_proc/bin/IPQ4019/hw.1/qca-wifi-fw-IPQ4019_hw_1-WLAN.BL.3.1-00084-S-
   1.tar.bz2 qsdk/dl
$ cp -rf cnss_proc/src/components/* qsdk/dl
```

For WHC Customers:

```
$ cp -rf apss_proc/out/proprietary/Wifi/qsdk-whc-3.0.1264.125/* qsdk
```

NOTE: The local directory **qsdk** is created by these repo steps as a sub-directory of the current working directory, from which repo was executed. This is the working QSDK top level directory.

If the CONFIG_DOWNLOAD_FOLDER="" item in the default .config file is changed, the files qca-wifi-fw-*.tgz in cnss_proc/src/components must be copied to both the qsdk/dl folder as well as the newly configured download folder, in order for the builds on third party machines to complete. The file qca-wifi-fw-IPQ4019_hw_1-WLAN.BL.3.1-00084-S-1.tar.bz2 must be located in the newly configured download folder.

9.3.2 Build QSDK

9.3.2.1 Set up the environment

This framework has been developed using Ubuntu (from version 12.04 to version 13.04), and Debian. However, QSDK framework regenerates critical tools required to compile firmware at build-time. In that sense, the framework is independent from the host environment; although it is developed using the distributions above, it is expected to work on others such as RedHat, Mint, or Fedora. This command is for Debian/Ubuntu; it must be customized for other distributions:

```
$ sudo apt-get install gcc g++ binutils patch bzip2 flex make gettext \
  pkg-config unzip zlib1g-dev libc6-dev subversion libncurses5-dev gawk \
  sharutils curl libxml-parser-perl ocaml-nox ocaml-nox ocaml ocaml-findlib \
  libpcre3-dev binutils-gold python-yaml
```

Because the framework automatically downloads the open source components, make sure an internet connection is active on the build host while creating the build.

9.3.2.2 Create the build

1. Move to the qsdk dir
   ```
   $ cd qsdk
   ```
2. Install the different feeds in the build framework:
   ```
   $./scripts/feeds update –a
   $./scripts/feeds install –a -f
   ```
3. Copy the base configuration to use for the build; choose either the standard or premium profile :

Premium	`$ cp qca/configs/qsdk/ipq806x_premium.config .config`
Standard	`$ cp qca/configs/qsdk/ipq806x_standard.config .config`

4. Regenerate a complete configuration file and start the build:
   ```
   $ make defconfig
   $ for pkg_num in 2 3 4 6 9;do sed 's/CONFIG_PACKAGE_qca-wifi-fw-hw'${pkg_num}'-10.4-asic=y/# CONFIG_PACKAGE_qca-wifi-fw-hw'${pkg_num}'-10.4-asic is not set/g' -i .config;done
   $ sed 's/CONFIG_PACKAGE_kmod-wil6210=y/# CONFIG_PACKAGE_kmod-wil6210 is not set/g' -i .config
   $ sed 's/CONFIG_PACKAGE_wigig-firmware=y/# CONFIG_PACKAGE_wigig-firmware is not set/g' -i .config
   $ make V=s
   ```

These instructions download the packages required for the corresponding profile and create the image. Once the build is complete, these files should be available in the **qsdk/bin/ipq806x** directory:

- openwrt-ipq40xx-u-boot-stripped.elf (Bootloader)
- openwrt-ipq806x-qcom-ipq40xx-ap.dkxx-fit-uImage.itb (Kernel + dtb)
- openwrt-ipq806x-squashfs-root.img (SquashFS)
- openwrt-ipq806x-ipq40xx-ubi-root.img (UBIFS)

9.3.3 Generate a complete firmware image

The IPQ40xx flash image includes multiple components. These include DTB, SBL1, TZ, CDT, DDR, MIBIB, Kernel, Filesystem, etc. To simplify the flashing of images, they have been combined into a single Flattened Image Tree (FIT) image. The FIT image can be flashed into the respective partition based on user configuration information. Additional tools required on the Ubuntu 12.04 64-bit machine:

1. Install mkimage: sudo apt-get install uboot-mkimage
2. Install DTC: sudo apt-get install device-tree-compiler
3. Install Python 2.7
4. Some systems may require installing/updating the U-Boot tools: **sudo apt-get install u-boot-tools**
5. Copy the flash config files to **common/build/ipq**; choose the either standard or premium profile:

Premium	`$ cp meta-scripts/ipq40xx_premium/* common/build/ipq`

Standard	`$ cp meta-scripts/ipq40xx_standard/* common/build/ipq`

6. Copy **pack.py** to the **apss_proc/out/** directory:
   `$ cp -rf qsdk/qca/src/uboot-1.0/tools/pack.py  apss_proc/out/`
7. Copy trustzone files to common/build/ipq
   `$ cp –rf trustzone_images/build/ms/bin/MAZAANAZ/* common/build/ipq`
8. Copy the **openwrt*** images built to the **common/build/ipq** folder:
   `$ cp –rf qsdk/bin/ipq806x/openwrt* common/build/ipq`
9. Copy the **boardconfig*** files to the **common/build/ipq** folder; choose either premium or standard profile depending on your requirement:

Standard	`$ cp boot_images/build/ms/bin/40xx/misc/tools/config/boardconfig_standard common/build/ipq`
Premium	`$ cp boot_images/build/ms/bin/40xx/misc/tools/config/boardconfig_premium common/build/ipq`

10. Run these commands to create a single image; choose either premium or standard profile depending on your requirement:

    `$ cd common/build`

Standard	`$ python update_common_info_standard.py`
Premium	`$ python update_common_info.py`

The commands create **nor-ipq40xx-single.img, nand-ipq40xx-single.img, nornand-ipq40xx-single.img, emmc-ipq40xx-single.img, norplusemmc-single.img, and nor-ipq40xx-standard-single.img** single images in the bin folder. The binary images are copied to the **ipq** directory either by the user or by the update_common_info.py command to create the FIT image:

```
cdt-AP.DK04.1-C1.bin
cdt-AP.DK04.1-C2.bin
cdt-AP.DK04.1-C3.bin
cdt-AP.DK04.1-S1.bin
gpt_backup0.bin
gpt_main0.bin
nor-system-partition-ipq40xx-
  s.bin
nand-system-partition-ipq40xx.bin
nor-system-partition-ipq40xx.bin
norplusnand-system-partition-
  ipq40xx.bin
openwrt-ipq40xx-u-boot-
  stripped.elf
openwrt-ipq806x-ipq40xx-ubi-root.img
openwrt-ipq806x-qcom-ipq40xx-
  ap.dkxx-fit-uImage.itb
openwrt-ipq806x-squashfs-root.img
sbl1_emmc.mbn
sbl1_nand.mbn
sbl1_nor.mbn
tz.mbn
nand-flash.conf
nor-flash.conf
emmc-flash.conf
norplusnand-flash.conf
packages
```

Flash Type	Single Image
SPI NOR (32 MBytes)	nor-ipq40xx-single.img
SPI NOR + QPIC NAND	norplusnand-ipq40xx-single.img
SPI NOR (16 MBytes) and DDR 128 MB	nor-ipq40xx-standard-single.img
ONFI NAND	nand-ipq40xx-single.img
eMMC	emmc-ipq40xx-single.img
SPI NOR (16 MBytes) + eMMC	norplusemmc-single.img

9.4 Flashing instructions

1. As a preliminary step, ensure that the board console port is connected to the PC using these RS232 parameters:
   - 115200bps
   - 8N1
2. Confirm that the PC is connected to the board using one of the Ethernet ports. The PC should have a TFTP server launched and listening on the interface to which the board is connected. At this stage power up the board and, after a few seconds, press any key during the countdown.

9.4.1 Flashing commands

The xxxx-ipq40xx-single.img is already a packed image and does not need any further packing. Start by copying the xxxx-ipq40xx-single.img to the TFTP server root directory.

1. Check hardware jumper Configuration according to reference board and configuration.

   About jumper configuration, refer to *AP.DK04 Setup Guide* (80-Y9700-6) and *IPQ4018-IPQ4028 AP.DK01 Setup Guide* (80-Y9700-2).

2. Confirm machine ID, Meta version, profile and single image.

Board/ Configuration	Machine ID	Flash	Vendor/Part #	Meta/ profile Support	Image Name	MCN#
AP.DK01.1-C1	8010000	NOR 32 MB	Macronix/MX25L25635FMI	1.1 CSU1 Premium profile	nor-ipq40xx-single.img	65-Y9431-451 (IPQ4018)
AP.DK01.1-C2	8010100	NOR 16 MB + SPI NAND(128 MB)	Gigadevice/GD25Q128CSIG(NOR) Gigadevice/GD5F1GQ4UCYIG(SPI NAND)	1.1 CSU1 Premium profile	nornand-ipq40xx-single.img	65-Y9431-451 (IPQ4018)
AP.DK01.1-C2	8010100	NOR 2 MB + SPI NAND(128 MB)	Gigadevice/GD25Q16CSIG(NOR) Gigadevice/GD5F1GQ4UCYIG(SPI NAND)	1.1 CSU1 Premium profile	nornand-ipq40xx-single.img	65-Y9431-451 (IPQ4018)
AP.DK01.1-S1	8010200	NOR 16 MB	Gigadevice/GD25Q128CSIG (NOR)	1.1 CSU1 Standard profile	nor-ipq40xx-standard-single.img	65-Y9431-451 (IPQ4018)
AP.DK01.2-C1	8010000	NOR 32 MB	Macronix/MX25L25635FMI	1.1 CSU1 Premium profile	nor-ipq40xx-single.img	65-Y9433-451 (IPQ4028)
AP.DK01.2-C2	8010100	NOR 16 MB + SPI NAND(128 MB)	Gigadevice/GD25Q128CSIG(NOR) Gigadevice/GD5F1GQ4UCYIG(SPI NAND)	1.1 CSU1 Premium profile	nornand-ipq40xx-single.img	65-Y9433-451 (IPQ4028)
AP.DK01.2-C2	8010100	NOR 2 MB + SPI NAND(128 MB)	Gigadevice/GD25Q16CSIG(NOR) Gigadevice/GD5F1GQ4UCYIG(SPI NAND)	1.1 CSU1 Premium profile	nornand-ipq40xx-single.img	65-Y9433-451 (IPQ4028)

Board/ Configuration	Machine ID	Flash	Vendor/Part #	Meta/ profile Support	Image Name	MCN#
AP.DK01.2-S1	8010200	NOR 16 MB	Gigadevice/GD25Q128CSIG (NOR)	1.1 CSU1 Standard profile	nor-ipq40xx-standard-single.img	65-Y9433-451 (IPQ4028)
AP.DK04.1-C1	8010001	NOR 16 MBytes	Gigadevice/GD25Q128CSIG(NOR)	1.1 CSU1 Standard profile	nor-ipq40xx-standard-single.img	65-Y9443-351
AP.DK04.1-C1	8010001	QPIC NAND (128 MBytes)	Micron /MT29F1G08	1.1 CSU1 Premium profile	nand-ipq40xx-single.img	65-Y9443-351
AP.DK04.1-C1	8010001	eMMC		1.1 CSU1 Premium profile	emmc-single.img	65-Y9443-351
AP.DK04.1-C1	8010001	NOR 16 MB + QPIC NAND (128 MBytes)	Gigadevice/GD25Q128CSIG(NOR) Micron /MT29F1G08	1.1 CSU1 Premium profile	nornand-ipq40xx-single.img	65-Y9443-351
AP.DK04.1-C2	8010101	NOR 16 MBytes – Audio	Gigadevice/GD25Q128CSIG(NOR)	1.1 CSU1 Standard profile	nor-ipq40xx-standard-single.img	65-Y9443-351
AP.DK04.1-C3	8010201	NOR 16 MBytes + eMMC	Gigadevice/GD25Q128CSIG (NOR)	1.1 CSU1 Premium profile	norplusemmc-single.img	65-Y9443-351

3. Commands for setting machine ID

   ```
   set machid 0xABCDEYZ;
   ```

   Example for AP.DK01.1-C1

   ```
   set machid 0x8010000
   ```

4. Commands for the TFTP process:

   ```
   set ipaddr 192.168.1.11
   set serverip 192.168.1.xx
   ```
   (This should be the address of the TFTP server)
   ```
   ping ${serverip}
   tftpboot 0x84000000 xxxx-ipq40xx-single.img
   ```

5. Flash the image with this command:

   ```
   imgaddr=0x84000000 && source $imgaddr:script
   ```

   The procedures for changing machine ID in u-boot is complex which requires more steps. For example: changing AP.DK01.1-C1 to AP.DK01.1-C2 on a DK01 board. AP.DK01 includes a 32 MBytes NOR flash and a 2 MBytes NOR flash. For a new AP.DK01 board, the 2 MBytes NOR flash is empty.

- To flash the SPI NOR+NAND singe image:

a. Check hardware jumper configuration. Because the 2M NOR flash is empty, the jumper settings for AP.DK01.1-C2 cannot be used to bring up the board. Use the jumper settings for AP.DK01.1-C1 to bring up to u-boot shell. Ensure the machine ID is 0x8010000.

b. Confirm machine ID, meta version, profile, and single image name of image to be flashed into the 2M NOR + NAND flash.

c. Set the machine ID to 0x8010100 (AP.DK01.1-C2).

d. Upload nornand-ipq40xx-single.img by TFTP.

e. Flash the image.

i Make sure the jumper settings for AP.DK01.1-C2 is used. Otherwise the 2 MBytes NOR and NAND cannot be accessed.

ii imgaddr=0x84000000 && source $imgaddr:script

iii An error occurs with the log:

```
"NAND erase: Size exceeds partition or device limit"
```

Root cause of failure: If the running u-boot uses machine ID 0x801000, it cannot access NAND. In this case, reset the board and redo Step 4 and Step 5 to write the second part of the image to NAND.

iv After board reset, new software is read and booted from the 2 MBytes NOR flash. The running u-boot uses machine ID 0x8010100. No need to redo Step 3.

v Once done, Step 5 will be successful.

- To flash image to the NOR flash:

```
sf probe && imgaddr=0x84000000 && source $imgaddr:script
```

- If the source version is standard profile, use the low 128 MByte DDR address space.

  For example, when changing machine ID to 8010000(AP.DK01.1-C1) from 8010200(AP.DK01.1-S1), do not use 0x88000000 DDR address because the AP.DK01.1-S1 version supports only 128 MByte DDR.

  Wrong command

```
tftpboot 0x88000000 nor-ipq40xx-single.img&& sf probe
&&imgaddr=0x88000000&&source $imgaddr:script
```

  Correct command

```
tftpboot 0x84000000 nor-ipq40xx-single.img&& sf probe
&&imgaddr=0x84000000&&source $imgaddr:script
```

6. Reset the board once the image flashing is successful.

9.4.2 Upgrade the firmware

This release has a feature to upgrade images from the OpenWrt web interface without the need for a TFTP server. After flashing the first image, any future upgrades can be done from the web interface. Use single image to upgrade the image.

9.5 Procedure to load Ramdisk image

1. Halt the auto-boot process at u-boot prompt by keyboard keying during the auto-boot count-down phase.
2. Execute set fdt_high 0x87000000.
3. Download ramdisk image to address 0x88000000 based on board type.

   `tftp 0x88000000 openwrt-ipq806x-qcom-ipq40xx-<xxx>-fit-uImage.itb`

Board/Configuration	Machine ID	Image Name
AP.DK04.1-C1	8010001	openwrt-ipq806x-qcom-ipq40xx-ap.dk04.1-c1-fit-uImage-initramfs.itb
AP.DK04.1-C2	8010101	openwrt-ipq806x-qcom-ipq40xx-ap.dk04.1-c2-fit-uImage-initramfs.itb
AP.DK04.1-C3	8010201	openwrt-ipq806x-qcom-ipq40xx-ap.dk04.1-c3-fit-uImage-initramfs.itb

4. Execute bootm 0x88000000.

A Minimizing Build Times and Avoiding Multiple Downloads

Total QSDK install and build time can be reduced by eliminating all additional internet downloads normally performed during re-runs of the install/make steps described in section 9.3, due to unpredictable bandwidth and server availability/loading of the remote repository host servers.

To reduce the time needed to obtain all files prior to the **make** and to avoid repeated downloads of the data, create a local mirror repository that users at the same company can access via an internal network using these commands when initially cloning the ChipCode repository (this example uses the distribution name *ipq_distro*; the distribution name will be something like *<your_company_id>/ipq4019-ilq-1-0_qca_oem-standard*).

NOTE: This example shows a git clone URL to the git server located in San Diego. You can use a different URL as advised on the ChipCode web interface that points to a Qualcomm git server located in your region.

1. Create the local mirror copy of the repository for proprietary code:
   ```
   cd /path/to/local/mirror/access/for/all
   git clone https://chipcode.qti.qualcomm.com/ipq_distro.git
   ```
2. Keep the ChipCode mirror copy up-to-date by running:
   ```
   cd /path/to/local/mirror/access/for/all/ipq_distro
   git fetch --all
   ```
3. Create a local mirror copy of the CodeAurora repositories:
   ```
   cd /path/to/local/mirror/access/for/all
   repo init –u git://codeaurora.org/quic/qsdk/releases/manifest/qstak –b
   release –m caf_TAG –-mirror
   repo sync –j4
   ```
4. Keep the CodeAurora mirror copy MANIFESTS up to date simply by running:
   ```
   cd /path/to/local/mirror/access/for/all
   repo sync –j 4
   ```
   If a new Open Source release label (eg caf_NEW_tag) has been published on the CodeAurora server, a minimal set of files corresponding to this new release label can be downloaded to the local mirror copy by executing the following command AFTER and IN ADDITION to the repo sync described above:
   ```
   repo sync –j 4 –m caf_NEW_tag
   ```
   See section 1.2 for the caf_TAG.

Each user can create their own local working copy of the QSDK code using these git and repo commands to access the mirror repositories in place of the corresponding commands in section 9.3:

- For the ChipCode repository:
  ```
  git clone /path/to/local/mirror/access/for/all/ipq_version.git
  ```
- For the Code Aurora repositories:
  ```
  repo init –-reference /path/to/local/mirror/access/for/all/ -u
  git://codeaurora.org/quic/qsdk/releases/manifest/qstak –b release –m
  caf_TAG
  repo sync –j4
  ```

See section 1.2 for the caf_TAG.

B QDART_Connectivity

B.1 Overview

This section describes the QDART_Connectivity 1.0.36. Refer to the *IPQ4018/ IPQ4019/ IPQ4028/ IPQ4029 RF Test User Guide* (80-Y9700-3) for more information.

The following functionality is supported in this build:

- QDART_Connectivity components running on a Windows 7 PC
- DK04 configuration support
- Tx power calibration
- Instrument support: NI MIMO, LP MIMO
- Single instrument: LP IQXel 80 or 160
- FTM support with 1x1 LP
- Cal data saved into Flash (NOR) on DK04
- Able to load board data file
- All chain mask combinations including 2x2
- Tx single tone at carrier frequency
- Tx99 support
- Concurrent Tx of 2 GHz and 5 GHz cards on the DK04
- Link test (DUT to DUT)
- Tx power accuracy test
- Tx mask test
- Target power vs Tx EVM sweep
- Rx waterfall sweep
- Backup/restore of calibration data
- Provide capability to get MAC from board data
- Include several Microsoft re-distributable DLLs to eliminate installation difficulties that may be experienced on some PCs
- Crystal calibration
- Frequency offset test

B.2 Changes

Table B-1 Changes for QDART-Conn

1.0.34	Added support for reporting noise floor and RSSI in dBm for QCA99xx
	Added support for enabling heavy clip register during hardware calibration for QCA98xx
	Added support to enable band edge for heavy clipping for QCA99xx and IPQ4018/19
	Fixed PER issue affecting Litepoint IQxel80 and IQxel160 when running without QTI WLAN SCPI
	Added support for transmit duty cycle parameter for QCA99xx
	Added support for 160MHz wide channel spurious and band edge testing
	Fixed receive status synchronization issue affecting QCA99xx and IPQ4018/19
	Added support for embedding spaces in board data file directory names
	Added flexibility in selecting board data file directories for AR9561/AR9531/AR9550
	Updated QCA9886 register map
	Added LDPC and STBC options in SISO receive test
	Fixed timeout issue affecting HPE44XXB spectrum analyzer
1.0.35	Added support for QCA9377 customer OTP stream read/write access
	Added support to include channel bonding when testing 20/40 MHz data rates with 40/80 MHz wide channels
	Fixed an issue with minimum acceptable power for the PER test which resulted in displaying an incorrect value
1.0.36	Fixed the installation issue associated with the QDART Connectivity 1.0.35

B.3 Known limitations

This build has these known limitations:

- For the PC station configuration, the Windows system must be set up to run in test mode before the driver can be installed
- Instrument support: LP MIMO is for a configuration of two IQXel 160s
- Crystal calibration does not support 2.4 GHz radio

C Supported Features

These features are supported as part of this IPQ4019.1.0 release. For commands, please refer to the *AP 10.4 Command Line Interface (CLI) User Guide* (80-Y8052-1) and the *AP 10.4 Programmer's Guide Wireless LAN* (80-Y8053-1).

IEEE 802.11a
IEEE 802.11b
IEEE 802.11g
IEEE 802.11n (2 GHz and 5 GHz)
IEEE 802.11ac
IEEE 802.11i, WEP, WPA, WPA2, WPA-WPA2 Hybrid mode
IEEE 802.11e
IEEE 802.11h
IEEE 802.11w
AES-128 CCMP
IEEE 802.1X
AP
802.11n frequency support
802.11n BW support
802.11n number of SS
802.11n MCS support
802.11n A-MSDU RX
802.11n A-MSDU TX
802.11n A-MPDU RX
802.11n A-MPDU TX
802.11n Block-ACK support
802.11n A-MPDUs with A-MSDUs
802.11ac BW Support
802.11ac number of SS
802.11ac MCS Support
802.11ac 40/80 TX Mask
802.11ac CCA detection
802.11ac A-MSDU RX
802.11ac A-MSDU TX
802.11ac A-MSDU – MSDU Min Size
802.11ac A-MPDU RX

802.11ac A-MPDU TX
802.11ac Single MPDU Delimiter
802.11ac A-MPDUs with A-MSDUs
Support for 802.11n-only clients
Support for 802.11ac-only clients
WDS – (WDS and Repeater AP)
Number of clients in BSS mode
Wave 1 WDS and repeater AP
Configurable RTS/CTS threshold
Configurable DTIM interval (1-10)
Configurable beacon interval
Tx power configuration
Security mode configuration
Set group cipher
Proxy STA/QWRAP
802.11ac MU-MIMO
Spectral analysis
DFS
Bursted and staggered beacons
Tx99/Rx99
256-QAM in 2.4G
TPC, rate, retries, Tx chain mask
Adaptive noise immunity
Green AP
U-APSD
MIMO power save
Manual channel selection
Open security
Tx scheduling/queuing
IP header in packet to be DWORD aligned
2x2 chain configuration
Country code configuration
SSID configuration
Manual channel select
SU Tx beamforming
Legacy power save support
Multi-SSID
HW support for at least 1 LED
IEEE 802.11d, k, m, r, u, v
NAWDS, Extender AP
Congestion control
PMK Caching
Time-based retry /discard

Quick STA kick out
Target crash recovery mechanism
Tx beamforming in 2.4 GHz
Support for Wi-Fi positioning
Inserting custom IE’s inside the beacon
Trustzone
OpenWrt BB
Standard profile support
IQUE (Multicast to Unicast conversion)
Hidden SSID
MBSSID support with different security modes
WAPI WPS2.0
AES-CCMP/GCMP (128/256), BIP-CMAC (256)
FIPS support
Multi-adapter support
Concurrent support of different mode clients
TCP/IP, UDP checksum offload for Ipv4 and Ipv6
TCP segmentation offload (TSO)
Raw mode
Packet classification accelerator
Set regulatory domain
Set user defined channel power limits
Blocking/allowing STAs (access control list)
Bandsteering
Channel Range Selection
Intelligent channel manager
DCS
EACS
IQUE – IGMPv3, Video Stream protection, congestion control
Aggregation scaling
Format encapsulation/decapsulation
Chain configuration
Driver load/unload
Enhanced client mode (ECM) support for NAWDS
Support for Enterprise APIs
Off-channel change and transmission
Watchdog
USB Storage
Linux – Preemption support
Software/Firmware Upgrade
NOR boot
NAND(ONFI) boot
eMMC boot

Audio – (I2S/TDM/SPDIF)
Secure boot
Emergency Download through USB Interface