QCS-AX

Channel State Information (CSI) Extraction

ON Semiconductor®

www.onsemi.com

USER MANUAL

Introduction

This document covers how QCS extracts Channel State Information (CSI) from the QCS−AX chipset family. Channel state information is the channel properties between the two wireless devices in operation. This channel state information will be communicated between the two devices (AP and STA) on the current operating channel, and this channel characteristic will be extracted and passed to upper layer software. This is supported on software release R6.3.x and later releases.

ABBREVIATION AND ACRONYM

Abbreviation and Acronym	Description
AP	Access Point
CDK	Customer Development Kit
CSI	Channel State Information
CSM	Client Steering Manager
CSMD	Client Steering Manager Daemon
HT	High Throughput
LTF	Long Training Field information of HT/VHT packets
NPU	Network Processing Unit
QSPDIA	QCS Spatial Diagnostics Application
RPE	Radio Peer Entity, bidirectional netlink−based interface
SDK	Software Development Kit
STA	Station/Client
TLV	Tag Length Value
VHT	Very High Throughput
QCS	Quantenna Connectivity Solutions Division
QCS−AX	QSR10GU−AX and QSR5GU−AX family of chipsets

References

- QSR10GU−AX−AN−BBIC5−SDK−Build−and−Installation−Guide.pdf from CDK
- Qdock package from the addon directory from CDK

Overview

QCS CSI extraction feature provides the low−level channel state information from QCS−AX Wi−Fi devices to upper layer software. 3rd−party application software can use CSI data to perceive the Wi−Fi channel environment and implement advanced features. For example, motion detection and indoor localization. CSI data extraction is supported in both 2.4 GHz and 5 GHz bands with 802.11n and newer STAs.

The figure below illustrates a typical functional diagram of a use case with motion detection using the QCS CSI extraction framework through Qdock. To support CSI, Qdock framework is a mandatory requirement.

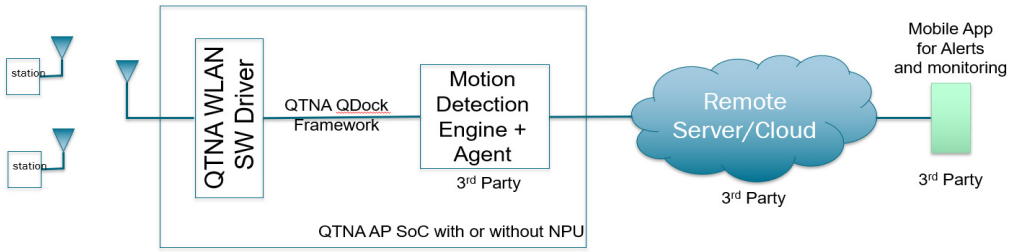

Figure 1. Typical Functional Diagram of CSI Use Case

By monitoring and calculating from HT/VHT LTFs in the data frames between AP and STAs, QCS WLAN driver can provide the channel matrices (as a part of the CSI API) to upper layer software. This channel information plus some additional information about the current PHY state (HW noise, RSSI, etc., see API details below) can be passed to 3rd−party applications through the Qdock framework. Qdock framework including RPE and CSM is responsible for the communication between 3rd−party applications and the low−level Wi−Fi driver. With Qdock framework, 3rd−party applications can run on QCS−AX SoC or on a connected NPU. CSI extraction can be supported for standalone or NPU based solutions without changing the control and data interfaces. The below two pictures illustrate the software architecture of standalone and NPU based CSI extraction solutions. QCS SDK also includes an example application called QSPDIA (QCS's spatial diagnostics application) which is not a part of the CSI extraction framework. It is merely an example to get started with creating the 3rd−party applications using the CSI API. The example application collects and dumps the CSI data onto a TCP socket.

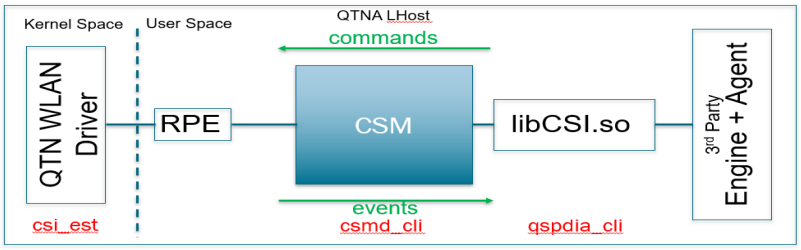

Figure 2. Software Block Diagram for QCS SoC Standalone Solution

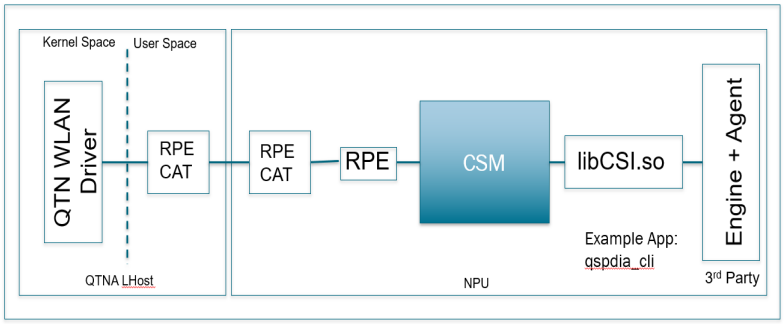

Figure 3. Software Block Diagram for NPU based Solution

Integration Steps

Data API supported by Qdock framework can be used by 3rd−party application to extract and receive CSI data. Qdock package, Qdock−X.Y.Z.tar.gz is packed into Qdock−QSR10G−x.y.z.zip, which is in the folder Source/Addon directory of the CDK. To build an image from the SDK with CSI and Qdock support, place the QCS Qdock package, Qdock−X.Y.Z.tar.gz tarball into the SDK's root folder and untar it. Once package is untarred into SDK's root folder, follow the SDK build procedure to build the runtime image. For more information on SDK build procedure, refer to the QSR10GU−AX−AN−BBIC5−SDK−Build−and−Installation−Guide.pdf from the CDK.

CSI extraction requires the CSMD to be initialized with a valid configuration file. An example configuration file is included, SPDIA.csmd.json located at qdock−x.y.z\src\csm. CSI extraction is initiated by configuring the interface using the qspdia_cli application. Follow these steps to run the SPDIA example application:

- Load in the image built with Qdock/CSI from above to the QCS−AX platform.
- Add “qsteer=1” in wireless_conf.txt and reboot the QCS−AX platform.
- Run “start_qdock stop” to make sure daemon is stopped.
  NOTE: An error message about start_soniq script is missing. Please ignore.
- Run ”csmd −c /etc/SPDIA.csmd.json &” to start the SPDIA daemon. Check by using ”ps”.
- Associate a WiFi client to the AP and check this station with ”csmd_cli show sta assoc”.
- Add the MAC address of client to monitor by using “qspdia_cli diag add xx:xx:xx:xx:xx:xx 30” (where xx:xx:xx:xx:xx:xx is the MAC of the monitored STA).

CSI Plugin Example

CSI extraction feature (sample code) is running as a plugin in the Qdock framework. The sample code can be found in buildroot\package\qdock\qdock−x.x.xx\src\csm\spdia. This example supports configuring the mode/interval/reorder setting for station and getting the CSI information dump of each stations’ data.

- **spdia_qtn.c**: the main C file for how to implement as a Qdock CSI plugin
- **spdia_dump.c**: some example functions to dump the spdia data. Example plugin can dump all the data to /tmp/.spdia_dump_NN file
- **spdia_ctrl.c**: an example to configure the plugin and driver
- **spdia_cli.c**: command line tool. Standalone tool for user to send commands and get results from the spdia CSI plugin

Sample spdia command line to support different configuration options shown below:

```
quantenna # qspdia_cli
Usage: qspdia_cli <command> [<parameter>] [<value>]
Commands:
  set <param> <value>          : set the run−time param
  get [<param>]                : get the run−time param
  diag show                    : show all diagnosed STAs
  diag add <mac> [key=val]     : Add diagnosed STA with
                                 specified parameters by
                    (KEY, it can be: period/mode/reorder/NG)
  diag del <mac>    : del the given STA from diagnosed list
  dbg level <level> : <level>:off/error/warn/notice/info/debug
```

To check the CSI data of each station from example plugin, user can:

- Local dump: dump information into local file (/tmp/.spdia_dump_NN). This is enabled by setting the debug type and level in the SPDIA.csmd.json configuration file.
- Remote TCP dump: run command “nc board_IP 50005 > csi_mat.log” on a ubuntu PC which connects to the test board through ethernet. This is because the example application dumps the CSI information onto a TCP socket.

Configuration and Data API

Configuration

CSI extraction plugin supports configuring low level WiFi driver through Qdock. The example plugin supports the following configurations:

- period=<val> : set the CSI monitoring period in ms
- mode=<ndp/data/mixed/none> : set the CSI monitoring operation mode
- reorder=<0/1> : enable/disable CSI tone reordering
- NG=<0/1/2> : set CSI decimation level

NOTE: “mode” setting: “ndp” means only calculate for ndp frames, “data” means only calculate for data frames, “mixed” means both ndp and data frames are used, and “none” means do not calculate.

Data API

The SPDIA module receives the CSI data from RPE through the function spdia_deal_info_event(). This received CSI data is internally parsed in function spdia_parse_info_event(). The message format in which the RPE delivers the CSI information is described below:

Table 1.

Field	Type	Value
ID	UINT16	0x0101
Message Type	UINT8	0x1
API_VER	UINT8	Version
IF_MAC	UINT8[6]	MAC address of the CSI source that this event refers to
PAYLOAD_LEN	UINT16	Length of payload
PAYLOAD	UINT8	CSI information

The payload contains the CSI information in TLV format. The TLV format details are described in Table 2 below:

Table 2.

Field	TLV ID	Value & Length
MAC address	TLV 503	TLV ID = 503 TLV Length = 6 bytes TLV Value: 6 bytes of MAC address
RSSI Vector	TLV 601	TLV ID = 601 TLV Length = multiple of 4 bytes TLV Value (padded if needed): One RSSI value (dBm) per receive antenna
HW Noise Vector	TLV 602	TLV ID = 602 TLV Length = 4 bytes TLV Value (padded if needed): One noise value (units of 0.1 dBm) per antenna
Timestamp	TLV 506	TLV ID = 506 TLV Length = 8 bytes TLV Value: Timestamp of last RX packet
Configuration	TLV 603	TLV ID = 603 TLV Length = 4 bytes TLV Value: BYTE0: NC (Number of columns in SPDIA matrix) BYTE1: NR (Number of rows in SPDIA matrix) BYTE2: NG (Grouping parameter) BYTE3: reserved
Tones	TLV 604	TLV ID = 604 TLV Length = 4 bytes TLV Value: Number of tones for resolution table
RX Packet Details	TLV 605	TLV ID = 605 TLV Length = 4 bytes TLV Value (decimal) BYTE0: Channel (encoded as in DSS Param Set IE) BYTE1: Bandwidth and Mode as BIT0−3: 0: 20MHz; 1: 40MHz; 2: 80MHz; 3: 160MHz; 4:80+80MHz BIT4−7: 0: 11n; 1: 11ac BYTE2: MCS (encoded as per 11ac numbering) BYTE3: NSS (number of Spatial Streams the MCS is sent on)
SPDIA Data	TLV 606	TLV ID = 606 TLV Length = padded to multiple of 4 bytes TLV Value (padded if needed): SPDIA/CSI Data

The CSI information can be dumped to TCP port 5005, after parsing the CSI information from the TLV. The data present in the TCP dump is described below:

Table 3.

Field	Type	Description
MAGIC_WORD	UINT8[8]	set to QCETAPI1 for QCS−AX and QCETAPI0 for QSR1000
Report Type	UINT32	For CSI it is always 1
Payload Size	UINT32	Payload size in bytes
Timestamp	UINT64	Timestamp
MAC Address	UINT8[6]	MAC address of the CSI source that this report refers to
RSSI Vector	INT32[8]	Per − chain RSSI (up to 8 values)
HW Noise Vector	INT32	Receiver noise estimate (divide by 10 to get the correct value)
Beamforming Mode	UINT8	1 − 11ac 0 − 11n
Number of columns	UINT8	Number of columns in CSI matrix (0 − 3, add 1 to get correct value 1 − 4)
Number of rows	UINT8	Number of rows in CSI matrix (0 − 3, add 1 to get correct value 1 − 4)

Table 3. (continued)

Field	Type	Description
Bandwidth	UINT8	BW of current frame (0 − 20MHz, 1 − 40MHz, 2 − 80MHz)
Decimation	UINT8	Grouping parameter (see Decimation Table for 802.11n (HT) mode with and without reordering and Decimation Table for 802.11ac (VHT) mode with and without reordering for more info)
Channel	UINT8	Channel of current transmission
MCS	UINT8	MCS of current packet
Spatial Streams	UINT8	Number of streams (1 − 4)
Tones	UINT32	Number of subcarriers
H−matrix	UINT32	CSI H−matrix data

- The “HW Noise Vector” value is scaled, so divide by 10 to get the correct value in dBm.
- For the NC/NR values, add 1 to get correct value 1−4. The NC value representing the number of non−zero columns in the H matrix is equal to the number of spatial streams in the packet. The NR value representing the number of rows in the H matrix is equal to the number of antennas at the receiver. Irrespective of the NC and NR values, the output H matrix is always of size 4x4. For example, if the frame uses 2 spatial streams and the receiver has 3 antennas, NC=2, NR=3, then the H matrix will be of size 4x4 with a 3x2 sub−matrix with non−zero values. Rest of the matrix will be zero packed.
- The H matrix, presented in code with a 8 bit array named CSI. Format example of the payload for the 4x4 H matrix is:

  H(s) = [h11(s), h12(s), h13(s), h14(s);
  h21(s), h22(s), h23(s), h24(s);
  h31(s), h32(s), h33(s), h34(s);
  h41(s), h42(s), h43(s), h44(s)];

  where:

  hii(s) = hii_real(s) + j*hii_imag(s);

  and s = subcarrier index (1 to 122 for 11ac/80MHz with NG=1)

```
csi[0] = h11_real(s=1);
csi[1] = h11_imag(s=1);
csi[2] = h12_real(s=1);
csi[3] = h12_imag(s=1);
...
csi[30] = h44_real(s=1);
csi[31] = h44_imag(s=1);
...
csi[3872] = h11_real(s=122);
csi[3873] = h11_imag(s=122);
csi[3874] = h12_real(s=122);
csi[3875] = h12_imag(s=122);
...
csi[3902] = h44_real(s=122);
csi[3903] = h44_imag(s=122);
```

Resolution of each of the real and imaginary parts are 16 bits. Hence, for the above matrix example, the size will be 122 (subcarriers) * 16 (matrix coefficients) * 2 (real + imag) * 2 (bytes) = 7808 bytes. With the header, the total size will be 7808 + 62 = 7870 bytes.

The grouping parameter NG reflects the decimation index. Following are the number of subcarriers present in the CSI data when different decimation parameters are used for different channel bandwidths.

Table 4. DECIMATION TABLE FOR 802.11n (HT) MODE WITH AND WITHOUT REORDERING

	Number of Subcarriers					
	Without Reordering			With Reordering		
BW	NG = 0	NG = 1	NG = 2	NG = 0	NG = 1	NG = 2
20 MHz	56	30	16	48	26	14
40 MHz	114	58	30	96	52	28

Table 5. DECIMATION TABLE FOR 802.11ac (VHT) MODE WITH AND WITHOUT REORDERING

	Number of Subcarriers					
	Without Reordering			With Reordering		
BW	NG = 0	NG = 1	NG = 2	NG = 0	NG = 1	NG = 2
20 MHz	52	30	16	48	26	14
40 MHz	108	58	30	96	52	28
80 MHz	234	122	62	192	104	56

EXAMPLE

Test Setup

The figure below illustrates a test setup which is used for CSI extraction. QCS−AX platforms can be used for AP side and station side. The PC can be a Linux Ubuntu PC or WIN10 PC with nc (netcat) installed. Netcat is a simple networking tool which can read/write data across network connections using TCP/IP protocol.

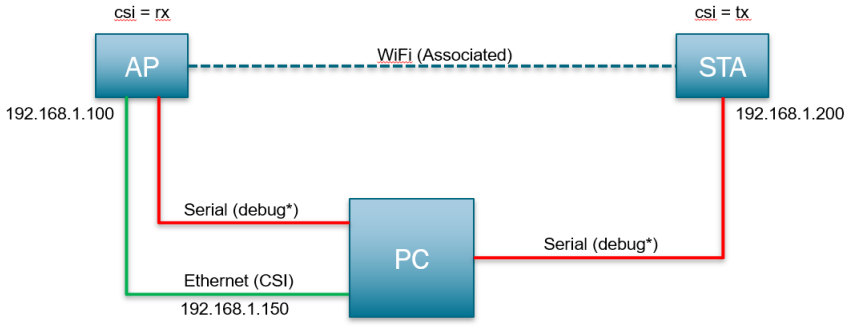

Figure 4. Test Setup Block Diagram

Test Setup Configuration

AP

- Load image generated with Qdock support to the QCS−AX platform.
- Add qsteer=1 into /mnt/jffs2/wireless_config.txt and reboot.

```
quantenna # cat /mnt/jffs2/wireless_conf.txt
region=none&staticip=1&start_down=0&maui=0&autostart=1&bsa=0&qsteer=1&
interface=wifi0_0&mode=ap&bw=80&channel=0&bf=1&pwr=19&scs=0&band=11ax&pmf=0
interface=wifi2_0&mode=ap&bw=40&channel=0&bf=1&pwr=19&scs=0&band=11axng
```

- Setup SSID and password for normal operation.
- Run “start_qdock stop” to make sure csmd daemon is stopped.

NOTE: An error message about start_soniq script is missing. Please ignore.

- Run “csmd −c /etc/SPDIA.csmd.json &” to start the SPDIA daemon. Check by using command “ps | grep csmd”.

```
quantenna # ps | grep csmd
1814 root      0:02 csmd −c /etc/SPDIA.csmd.json
```

- SPDIA.csmd.json should have the correct debug port. Snippet of the SPDIA.csmd.json for reference with dump_port set to 50005.

```
quantenna # cat /etc/SPDIA.csmd.json
……
“networks”: [
  {
  “MDID”: ”00:00”,
  “logics”: [
      {
       “name”: “spdia.qtn”,
       “parameters”:
       {
         “stations”: [
           {“mac”:”00:26:86:f0:86:04”, “period”:50, “mode”:”ndp”, “reorder”:1, “ng”:1},
           {“mac”:”00:26:86:f0:84:12”, “period”:50, “mode”:”data”, “reorder”:1, “ng”:0}
           ],
           “dump_level”:0,
           “dump_interval”:1000,
           “dump_burst”:1,
           “dump_kbytes”:500,
           “dump_port”:50005,
           “log_level”:”warn”
       }
      }
        ]
  }
     ],
```

- TCP dump_port value can be checked at runtime using the QSPDIA command:

```
quantenna # qspdia_cli get dump_port
50005
```

STA

- Load image generated with Qdock support to the QCS−AX platform.
- Add qsteer=1 in /mnt/jffs2/wireless_config.txt and reboot.

```
quantenna # cat /mnt/jffs2/wireless_conf.txt
region=none&staticip=1&start_down=0&maui=0&autostart=1&bsa=0&qsteer=1&
interface=wifi0_0&mode=sta&bw=80&channel=0&bf=1&pwr=19&scs=0&band=11ac&pmf=0
#interface=wifi2_0&mode=sta&bw=40&channel=11&bf=1&pwr=19&scs=0&band=11ac
```

Linux PC

- Set IP address 192.168.1.150 (or base on the IP setup of the network).
- Make sure the ethernet connection to AP is correct by pinging AP from PC.
- Connect serial cable to AP board to access QCS−AX serial console.
- Install nc (netcat).

Test Steps

Associate STA

- Associate station with the AP, confirm by show_assoc in serial console.
  Example: QCS−AX STA associated to AP with MAC address 00:26:86:f1:34:48.

```
quantenna # show_assoc
MAC               Idx  AID  Type Mode   Vendor   BW   Assoc   Auth   BA State      VAP    PS   SUBF
00:26:86:f1:38:54   6    0   vap   −      qtn   80      0      0   00000000   wifi0_0     0     0
00:26:86:f1:34:48   8    1   sta   ac     qtn   80  781545     1   00004363   wifi0_0     0     3
```

Start CSMD Daemon

- Confirm CSMD is running (ps | grep csmd) and configuration file is correct.
- SPDIA.csmd.json should have the correct debug port.
- NOTE: CSMD daemon is not needed on the side that is “TX”.

CSI Data Extraction

- Use command qspdia_cli to setup the MAC address of the STA associated and configure other CSI parameters. Set non−zero interval to start collecting CSI data.
  Example: Adding QCS−AX STA with time interval 50, NG set to 0 and reorder set to 0.

```
quantenna # qspdia_cli diag add 00:26:86:f1:34:48 mode=data interval=50 ng=0 reorder=0
success
```

- Confirm STA is added to CSI table to get diagnostic CSI data using qspdia_cli command.
  Example: Confirm QCS−AX STA is added to CSI table list.

```
quantenna # qspdia_cli diag show
[00:26:86:f0:86:04]: period 50
[00:26:86:f0:84:12]: period 50
[00:26:86:f1:34:48]: period 50
```

- Periodic data to station can be sent using different tools hping, ping etc. The below example sends periodic data to the STA from the Linux machine connected to AP using fast ping.
  - Use sudo ping −i 0.05 <STA IP>

```
linuxpc # sudo ping −i 0.05 192.168.1.100
```

- Confirm CSI data extraction engine is processing the CSI data using the below command. The rx_csi_data_processed counter value should be incrementing with time. This value indicates the CSI data packets processing count.

```
quantenna # stats muc_rx wifi0 | grep csi
rx_csi_data_processed       5338
rx_csi_data_no_capt         11922
rx_csi_data_unlock          4634
rx_csi_jiff_unlock          13862
rx_csi_interval_mismatch    4800
```

NOTE: Use wifi0 for 5GHz stats and wifi2 for 2.4 GHz stats.

- Collect CSI extracted data for analysis
  - CSI extracted data can be dumped into local file or collected over remote TCP dump on port 50005. As local data dump consumes CPU cycles, it is always recommended to collect data over remote TCP dump.
  - **Remote Collection**:
    Collect data using nc on Linux PC with root user. File can be extracted over TCP port 50005.
    - timeout 60 nc −q 3600 <AP IP> 50005 > test_csi.log

```
linuxpc # sudo su
linuxpc # timeout 60 nc −q 3600 192.168.1.100 50005 > test_csi.log
```

- **Local Collection**:
  - Local collection of dump is enabled by default
  - Collect data using local dump to temporary directory. Dumping of local data can be disabled by changing dump_level to 0 in the SPDIA.csmd.json configuration file or using qspdia_cli command to change log level.
- Details about dump_level
  - 0 − none;
  - 1 − the compatibility data−structure without H matric payload;
  - 2 − the compatibility data−structure with H matric payload;
  - 3 − the RPE_EVENT_SPDIA_STATS event

```
quantenna #  ls −la /tmp/ | grep spd
−rw−r−−r−−   1 root        564447 Jan 10 02:09 .spdia_dump_00
−rw−r−−r−−   1 root        225352 Jan 10 02:09 .spdia_dump_01
```

Limitation

- Since CSI extraction is using either “ndp” and/or “data” frames, CSI relies on these types of traffic to be present on the link. It is the responsibility of the application implementer to make sure such traffic exist. For example, if the mode is set to data, and interval is set to 50 ms, the implementer must ensure there is at least one data frame from the CSI sender to the CSI collection node every 50 ms.
- Currently, a total of 3 clients are supported for the CSI data extraction. These 3 clients can be all 2.4 GHz, or all 5 GHz, or a combination of both.

PUBLICATION ORDERING INFORMATION

LITERATURE FULFILLMENT:
Email Requests to: orderlit@onsemi.com

ON Semiconductor Website: www.onsemi.com

TECHNICAL SUPPORT
North American Technical Support:
Voice Mail: 1 800−282−9855 Toll Free USA/Canada
Phone: 011 421 33 790 2910

Europe, Middle East and Africa Technical Support:
Phone: 00421 33 790 2910
For additional information, please contact your local Sales Representative