BROADCOM®

Technical Information

MoCA

MoCA 2.0 Diagnostics

MoCA-TI108-R

Corporate Headquarters: San Jose, CA

April 12, 2016

Revision History

Revision	*Date*	*Change Description*
MOCA-TI108-R	04/12/16	**Updated:** • Table 2: “Link-down Causes and Common Warnings,” on page 24: Added 6526/6527 ID. • Appendix D, “Continuous TX Usage”, on page 35: Added note on RBW and VBW. • “API” on page 39: Added “-p” to API syntax and example. • “Tuning Procedure” on page 40: Added 0xc nibble value.
MOCA-TI107-R	02/08/16	**Added:** • Appendix F: “Tuning Impedances of Broadcom BCM6802C0 Devices,” on page 47
MOCA-TI106-R	11/20/15	**Updated:** • “TX Power Tune” on page 23
MOCA-TI105-R	11/02/15	**Updated:** • Table 2: “Link-down Causes and Common Warnings” • “PER between Two Nodes in the Networks”
MOCA-TI104-R	09/08/15	**Updated:** • The last subbullet in “PER On Egress Node Only”
MOCA-TI103-R	08/06/15	**Updated:** • “TX Power Tune” • Moved edited Appendix E and portions of “Table 2: Link-Down Causes” to a new section, “Link-Down Causes and Common Warnings”
MOCA-TI102-R	07/01/15	**Added:** • Appendix F: “List of Common Warnings”
MOCA-TI101-R	04/12/15	**Updated:** • Moved all General Items to appendices A–E.
MOCA-TI100-R	09/03/14	Initial TechPubs release.
Previous Updates:		
2.4.5	07/08/14	Added bonding commands, ECB boards handling, TPCAP, CTX and TX power printing sections
2.4.4	05/28/14	Added link down causes
2.4.3	04/29/14	Added LMO report analysis

Information furnished by Broadcom is believed to be accurate and reliable. However, Broadcom does not assume any liability arising out of the application or use of this information, nor the application or use of any product or circuit described herein, neither does it convey any license under its patent rights nor the rights of others.

Revision	***Date***	***Change Description***
2.4.2	04/16/14	Added FC analysis comment
2.4.1	07/11/13	Added Troubleshooting section
2.4.0	04/14/13	Added command alias listing
2.2.1	11/15/12	Added loopback command info
2.2.0	10/26/12	Added ‘do’ command section
2.1.0	03/02/12	–
2.0.0	02/15/12	Initial Release

Table of Contents

List of Tables

About This Document

Purpose and Audience

This document is a manual for using the test application *mocap*, provided with Broadcom's MoCA 2.0 driver software. This manual also includes useful information for testing and troubleshooting common MoCA issues.

This document is intended for hardware design, application, and OEM engineers.

Acronyms and Abbreviations

In most cases, acronyms and abbreviations are defined on first use.

For a comprehensive list of acronyms and other terms used in Broadcom® documents, go to: http://www.broadcom.com/press/glossary.php.

Document Conventions

The following conventions may be used in this document:

Convention	***Description***
Bold	User input and actions: for example, type **exit**, click **OK**, press **Alt+C**
`Monospace`	Code: `#include <iostream>` HTML: `<td rowspan = 3>` Command line commands and parameters: `wl [-l] <command>`
< >	Placeholders for *required* elements: enter your <username> or `wl <command>`
[ ]	Indicates *optional* command-line parameters: `wl [-l]` Indicates bit and byte ranges (inclusive): [0:3] or [7:0]

Technical Support

Broadcom provides customer access to a wide range of information, including technical documentation, schematic diagrams, product bill of materials, PCB layout information, and software updates through its customer support portal (https://support.broadcom.com). For a CSP account, contact your Sales or Engineering support representative.

In addition, Broadcom provides other product support through its Downloads and Support site (http://www.broadcom.com/support/).

Introduction

The mocap application provides a command line interface to configure the MoCA interface on Broadcom MoCA 2.0-supported platforms. The application has diagnostic capabilities for status and statistics information on the MoCA interface.

Mocap is built upon and serves as an extension to the MoCA 2.0 software API. The command names and options map directly to functions and structures available via the API. Built-in help information is included in the application, which is the same information found in the API documentation. This makes the mocap application a useful tool as well as an informational resource.

Using mocap

With No Parameters

Invoking *mocap* without any parameters shows the basic syntax and list of available commands. In the following example, the list of commands has been edited due to its length.

```
# mocap
mocap version: 2.2.0
Usage: mocap <option> <sub-options> [parameters]
Type 'mocap <option> <sub-option> help' for more information about a particular sub-option.

Options       Sub-options
-------------------------------------------------
get
    --adc_mode --aes_exchange_interval --aes_mm_key --aes_pm_key
    --aes_pmk_initial_key --amp_type --arpl_th_100 --arpl_th_50 --assertText
    --beacon_channel --beacon_pwr_reduction --beacon_pwr_reduction_en --bo_mode
    --bonding --brcmtag_enable --bw --c4_moca20_en --cap_phy_rate_en

        [lines removed]

    --target_phy_rate_20_turbo_vlper_sec_ch --target_phy_rate_qam128
    --target_phy_rate_qam256 --tek_exchange_interval --tlp_mode --tpc_en --trace
    --turbo_en --uc_fwd --verbose --wdog_enable --wom_ip --wom_magic_enable
    --wom_magic_mac --wom_mode --wom_pattern
set
    --adc_mode --aes_exchange_interval --aes_mm_key --aes_pm_key
    --aes_pmk_initial_key --amp_type --arpl_th_100 --arpl_th_50 --assertText
    --beacon_channel --beacon_channel_set --beacon_pwr_reduction
    --beacon_pwr_reduction_en --bo_mode --bonding --brcmtag_enable --bw

        [lines removed]

    --restart --restore_defaults --rf_band --rlapm_en --rlapm_table_100
    --rlapm_table_50 --rtr_config --rx_power_tune --rx_tx_packets_per_qm
    --rxd_lmo_request --sapm_en --sapm_table_100 --sapm_table_50 --sapm_table_sec
    --schop --sco --selective_rr --sigma2_prints
```

```
do
    --dd_init --fmr_20 --fmr_init --moca_reset --pqos_create_flow
    --pqos_delete_flow --pqos_list --pqos_query --pqos_status
    --pqos_update_flow --ps_cmd
```

There are *get*, *set*, and *do* commands for mocap. The *get* commands retrieve information from the MoCA driver. The *set* commands configure parameters in the MoCA driver. The *do* commands typically have input and output parameters and may require a transaction to occur on the MoCA network before completing.

Accessing Help Information

At nearly every level of the hierarchy of mocap, it is possible to type “help” to obtain the next values to input or to obtain information regarding a value or parameter.

Below, is an example for a *get* command. Information about the command is listed as well as the syntax on how to invoke the command.

```
# mocap get --taboo_channels help

   Set and Get taboo channel configuration. The fixed mask parameters are used to set specific
   frequencies as taboo regardless of the operating frequency. The left and right mask values are
   used to set frequencies relative to the operating frequency as taboo.

mocap get --taboo_channels
```

Below is an example for a *set* command. Information about the command is listed as well as the syntax on how to invoke the command. In this example, there are several options for parameters that can be input.

```
# mocap set --taboo_channels help

   Set and Get taboo channel configuration. The fixed mask parameters are used to set specific
   frequencies as taboo regardless of the operating frequency. The left and right mask values are
   used to set frequencies relative to the operating frequency as taboo.

mocap set --taboo_channels <options>

options:
   taboo_left_mask          <uint32>
   taboo_fixed_mask_start   <uint32>
   taboo_fixed_channel_mask <uint32>
   taboo_right_mask         <uint32>
```

Each option for the command also has help information.

```
# mocap set --taboo_channels taboo_left_mask help

taboo_left_mask:
========================================
Left side mask for adjacent channels taboo, relative to the LOF.

Values:
   Only 24 lsb are relevant.
Default:
   0x00ffffff
Note: Parameter will take effect after next MoCA core initialization.
```

Basic *get* Command

For basic *get* commands that do not require any input parameters, the syntax is simple:

```
mocap get --parameter_name

For example:
# mocap get --lof

lof
========================================
val: 1200  ( 0x4b0 )
```

Single Parameter *get* Command

Some *get* commands require an input parameter such as a *node ID*. If a required input parameter is missing, mocap will issue a warning message indicating the missing parameter. When only a single input parameter is required, it follows the `--option` word in the syntax.

For example, a *get* command with a missing parameter:

```
# mocap get --node_stats
Missing index parameter.
```

The output indicates that the *index* parameter is missing. Running the help command, it is clear that the index is the only parameter required. The index specifies the node ID for which the stats will be shown.

```
# mocap get --node_stats help
Nodes Statistics
```

The following table is maintained for each MoCA destination node on the MoCA network.

```
mocap get --node_stats <uint32 index>
index:
========================================
Node ID of the destination node
Minimum:
   0
Maximum:
   15
```

Finally, executing the command with a valid node ID shows the desired information.

```
# mocap get --node_stats 0
node_stats
========================================
tx_packets       : 0  ( 0x0 )
rx_packets       : 0  ( 0x0 )
rx_cw_unerror    : 0  ( 0x0 )
rx_cw_corrected  : 0  ( 0x0 )
rx_cw_uncorrected: 0  ( 0x0 )
rx_no_sync       : 0  ( 0x0 )
```

If an out of range input parameter is supplied, an error message is displayed.

```
# mocap get --node_stats 100
eth1: NODE_STATS error, out of range IE 100
Error -3
```

Multiple Parameter *get* Commands

Some *get* commands require more than one parameter in order to obtain the desired output information. Again, mocap will indicate a warning if a required input parameter is missing. In this example, there are two parameters required: *index* and *profile_type*. Since there is more than one, the suboption names must be given on the command line.

A warning is shown indicating that the *index* parameter is missing.

```
# mocap get --gen_node_ext_status
Missing index parameter.
```

Running *help* shows that there are two input options for this command.

```
# mocap get --gen_node_ext_status help
Nodes Extended Status (PHY Parameters)
```

The following table is maintained for each MoCA destination node on the MoCA network. This table is also maintained for the various profile types.

```
mocap get --gen_node_ext_status <options>
options:
   index          <uint32>
   profile_type   <uint32>
```

If only the *index* parameter is specified, a warning indicating the *profile_type* parameter is missing is shown.

```
# mocap get --gen_node_ext_status index 1
Missing profile_type parameter.
```

Running *help* for *profile_type* shows the possible values for this parameter.

```
# mocap get --gen_node_ext_status index 1 profile_type help
profile_type:
========================================
```

The profile type of the corresponding table is to be retrieved.

MoCA 2.0 profiles start with profile_type 6.

```
Values:
   0 = RX Unicast
   1 = RX Broadcast
   2 = RX Map
   3 = TX Unicast
   4 = TX Broadcast
   5 = TX Map
   6 = RX Unicast VLPER
   7 = RX Unicast NPER
   8 = RX Broadcast VLPER
   9 = RX Broadcast NPER
   10 = RX Map 2.0
   11 = RX OFDMA
   12 = TX Unicast VLPER
   13 = TX Unicast NPER
   14 = TX Broadcast VLPER
   15 = TX Broadcast NPER
   16 = TX Map 2.0
   17 = TX OFDMA

Minimum:
   0
Maximum:
   5
```

Finally, running the command with all options specified provides the desired output.

```
# mocap get --gen_node_ext_status index 1 profile_type 7
== gen_node_ext_status ===============================
node           : 1  02:10:18:38:82:01
profile_type   : RX Unicast NPER
nbas           : 4256  ( 0x10a0 )
preamble_type  : 6  ( 0x6 )
cp             : 26  ( 0x1a )
tx_power       : 0 dBm
rx_power       : -39.750 dBm
bit_loading[64]:
   256 - 287:  00000000000008888888888888888888
   288 - 319:  89899999999999999999999999999999
   320 - 351:  99999999999999999999999999999999
   352 - 383:  99999999999999999999999999999999
   384 - 415:  89999999999999999999999999999999
   416 - 447:  99999999999999999999999999999999
   448 - 479:  99999999999999999999999999999999
   480 - 511:  99999999999999999999999988887000
   000 - 031:  00088889999999999999999999999999
   032 - 063:  99999999999999999999999999999999
   064 - 095:  99999999999999999999999999999999
   096 - 127:  99999999999999999999999999999999
   128 - 159:  99999999999999999999999999999999
   160 - 191:  99999999999999999999999999999999
   192 - 223:  99999999999999999999988888888888
   224 - 255:  88888888888888887888000000000000
avg_snr        : 33.957
phy_rate       : 639 Mbps
turbo_status   : 0  ( 0x0 )
== end gen_node_ext_status ===========================
```

Single Parameter *set* Commands

Similar to their *get* counterparts, single parameter *set* commands require only the parameter following the main option. If the command is successful, no output is given. Some parameters are only sent to the MoCA interface at initialization time—a note is shown for such parameters.

```
# mocap set --lof 1200

Note: Parameter will take effect after next MoCA core initialization.
```

Multiple Parameter *set* Commands

Some *set* commands can accept several parameters. In this case, each suboption name must be specified on the command line in order to set that parameter.

Using `--taboo_channels` as an example, there are four possible suboptions that can be set.

```
# mocap set --taboo_channels help
   Set and Get taboo channel configuration. The fixed mask parameters are used to set specific
   frequencies as taboo regardless of the operating frequency. The left and right mask values are
   used to set frequencies relative to the operatingfrequency as taboo.

mocap set --taboo_channels <options>
options:
   taboo_left_mask            <uint32>
   taboo_fixed_mask_start     <uint32>
   taboo_fixed_channel_mask   <uint32>
   taboo_right_mask           <uint32>
```

In this example, only the left and right mask values are updated. The fixed mask start and fixed channel mask values are unchanged.

```
# mocap set --taboo_channels taboo_left_mask 0xffff taboo_right_mask 0xFFFF0000

Note: Parameter will take effect after next MoCA core initialization.
```

Setting Arrays

The syntax for setting an array requires at least two parameters: the *value* to set and the *index* to set. Multiple array entries can be set to the same value by also specifying an *end index*.

In this example, the AES MM key can be set. It is an array of four 32-bit values.

```
# mocap set --aes_mm_key help
AES MAC Management Key

mocap set --aes_mm_key <options>

options:
   val   <uint32 value> <index> <end index (optional)>

# mocap get --aes_mm_key
aes_mm_key
========================================
val[4]: 0x0 0x0 0x0 0x0
```

In this example, entries 0, 1, and 2 are set to 0xabcd and entry 3 is set to 0x1234.

```
# mocap set --aes_mm_key val 0xabcd 0 2 val 0x1234 3

Note: Parameter will take effect after next MoCA core initialization.

# mocap get --aes_mm_key
aes_mm_key
========================================
val[4]: 0xabcd 0xabcd 0xabcd 0x1234
```

Combining Commands

Multiple options can be specified in one command.

For example, obtain the value of several parameters in one command:

```
# mocap get --lof --single_channel_operation --interface_status --taboo_channels

lof
========================================
val: 1200  ( 0x4b0 )

single_channel_operation
========================================
val: 0  ( 0x0 )

interface_status
========================================
link_status: 0  ( 0x0 )
rf_channel : 58  ( 0x3a )

taboo_channels
========================================
taboo_fixed_mask_start  : 41  ( 0x29 )
taboo_fixed_channel_mask: 16252928  ( 0xf80000 )
taboo_left_mask         : 65535  ( 0xffff )
taboo_right_mask        : 4294901760  ( 0xffff0000 )
```

In this example, several parameters can be set in the same command:

```
# mocap set --privacy_en 0 --moca_core_trace_enable 0 --verbose 0
```

In this example, the *get* and *set* commands are combined. The commands are executed in order from left to right.

```
# mocap get --lof set --lof 1200 get --lof
lof
========================================
val: 1300  ( 0x514 )

Note: Parameter will take effect after next MoCA core initialization.

lof
========================================
val: 1200  ( 0x4b0 )
```

The *do* Command

The *do* command typically has input and output parameters. Most L2 MoCA operations are *do* commands. These operations require message transactions to occur between the nodes of the MoCA network in order to complete. The status of the operation is typically reported in the output of the *do* command.

Creating a PQOS flow is an L2 MoCA operation that is accessible via a mocap *do* command.

```
# mocap do --pqos_create_flow help
   Creating a new PQoS flow. The flowid field must be unique to the network. The Ingress side is
   configured by entering the ingress node MAC address. The Egress side is configured by entering
   the egress node MAC address.

mocap do --pqos_create_flow <options>
options:
   ingress_node           <macaddr>
   egress_node            <macaddr>
   flow_id                <macaddr>
   packet_da              <macaddr>
   packet_size            <uint32>
   flow_tag               <uint32>
   peak_data_rate         <uint32>
   lease_time             <uint32>
   burst_size             <uint32>
   vlan_id                <uint32>
   vlan_prio              <uint32>
   max_latency            <uint32>
   short_term_avg_ratio   <uint32>
   max_retry              <uint32>
   flow_per               <uint32>
   in_order_delivery      <uint32>
   traffic_protocol       <uint32>
   ingr_class_rule        <uint32>
   vlan_tag               <uint32>
   dscp_moca              <uint32>
```

Help information is available for each input parameter. If a parameter has a default value, it is not necessary to specify a value for it on the command line. The default value will be used.

```
# mocap do --pqos_create_flow ingress_node help
ingress_node:
========================================
MAC address of the ingress node of the flow.

# mocap do --pqos_create_flow peak_data_rate help
peak_data_rate:
========================================
Peak data rate in kbps
Default:
   1000
Minimum:
   0
Maximum:
   0xFFFFFE
```

Here we see that the response code and decision output fields indicate a successful transaction. The flow has been created.

```
# mocap do --pqos_create_flow ingress_node 02:10:18:32:41:01 egress_node 02:10:18:38:82:01 flow_id
01:00:5e:00:11:22 packet_da 00:11:22:33:44:55 vlan_id 0 vlan_prio 5 peak_data_rate 2500 packet_size
1500
== pqos_create_flow  ==================================
flow_id                 : 01:00:5e:00:11:22
flowda                  : 00:11:22:33:44:55
response_code           : Success (0)
decision                : DECISION_SUCCESS (1)
flow_tag                : 0  ( 0x0 )
peak_data_rate          : 2500  ( 0x9c4 )
packet_size             : 1500  ( 0x5dc )
burst_size              : 2  ( 0x2 )
lease_time              : 0  ( 0x0 )
total_stps              : 709856  ( 0xad4e0 )
total_txps              : 366  ( 0x16e )
flow_stps               : 455202  ( 0x6f222 )
flow_txps               : 209  ( 0xd1 )
dest_flow_id            : 129  ( 0x81 )
maximum_latency         : 0  ( 0x0 )
short_term_avg_ratio    : 0  ( 0x0 )
max_number_retry        : 0  ( 0x0 )
flow_per                : 0  ( 0x0 )
in_order_delivery       : 0  ( 0x0 )
ingr_class_rule         : 0  ( 0x0 )
traffic_protocol        : 0  ( 0x0 )
vlan_tag                : 5  ( 0x5 )
dscp_moca               : 0  ( 0x0 )
max_short_term_avg_ratio: 0  ( 0x0 )
bw_limit_info           : 129  ( 0x81 )
== end pqos_create_flow  ==============================
```

Groups

When diagnosing an issue, it is often useful to obtain as much debug information as possible. The various parameters accessible to mocap are categorized into different groups of data. Each group of parameters can be retrieved in one command, or all groups can be retrieved together.

To see the available group types, run the following command.

```
# mocap get --group help
Group name:
========================================
debug
forwarding
intfc
lab
mac_layer
network
node
phy
power_mgmt
security
```

Specify the name of the group to obtain its information:

```
# mocap get --group security
GROUP:  security
================================================
privacy_en: 0  ( 0x0 )
pmk_exchange_interval: 39600000 (11:00:00.000)
tek_exchange_interval: 540000 (00:09:00.000)
aes_exchange_interval: 21600000 (06:00:00.000)

== mmk_key  ===========================================
mmk_key_hi: 0  ( 0x0 )
mmk_key_lo: 0  ( 0x0 )
== end mmk_key  =======================================

== pmk_initial_key  ===================================
pmk_initial_key_hi: 0  ( 0x0 )
pmk_initial_key_lo: 0  ( 0x0 )
== end pmk_initial_key  ===============================

aes_mm_key: 0x0 0x0 0x0 0x0
aes_pm_key: 0x0 0x0 0x0 0x0

== current_keys  ======================================
pmk_even_key[2]    : 00000000 00000000
pmk_odd_key[2]     : 00000000 00000000
tek_even_key[2]    : 00000000 00000000
tek_odd_key[2]     : 00000000 00000000
aes_pmk_even_key[4]: 00000000 00000000 00000000 00000000
aes_pmk_odd_key[4] : 00000000 00000000 00000000 00000000
aes_tek_even_key[4]: 00000000 00000000 00000000 00000000
aes_tek_odd_key[4] : 00000000 00000000 00000000 00000000
== end current_keys  ==================================

password: 99999999988888888
```

To obtain the information from all groups, use the following command.

```
# mocap get --groupall
```

Common Commands

Basic Configuration

To stop and start the MoCA interface, use commands `mocap set --stop` and `mocap set --start`.

```
# mocap set --stop
eth1: Stopping MoCA interface

# mocap set --start
eth1: Starting MoCA interface
eth1: Loading Moca Core image...(9)
# eth1: Loading Moca Core image done.
eth1: THIS Node MAC address: 00:10:18:3d:54:bf
eth1: Last Operational Frequency = 1200 Mhz
eth1: MoCA Startup Successful.
eth1: MoCA Version
eth1: -----------------------
eth1: firmware version  : 2.0.0
eth1: mocad version     : 2.0.0
eth1: HW version        : 0x742900a0
eth1: bmoca version     : 4.0.20110831
eth1: MoCA self version : 0x20
eth1: -----------------------
```

Use `--lof` to set the last operating frequency and use `--single_channel_operation` to set the frequency scanning mode.

```
# mocap set --lof 1200 --single_channel_operation 0
```

For debugging information, traces from the MoCA core can be enabled and disabled using `--moca_core_trace_enable`.

To enable:

```
# mocap set --moca_core_trace_enable 1
```

To disable:

```
# mocap set --moca_core_trace_enable 0
```

Status Information

`Interface_status` shows whether the MoCA interface currently has a link and which channel number is being used.

```
# mocap get --interface_status
interface_status
========================================
link_status: 1  ( 0x1 )
rf_channel : 46  ( 0x2e )
```

`Network_status` shows information about the current MoCA network.

```
# mocap get --network_status
network_status
========================================
network_moca_version: 32  ( 0x20 )
connected_nodes     : 3  ( 0x3 )
node_id             : 0  ( 0x0 )
nc_node_id          : 0  ( 0x0 )
backup_nc_id        : 1  ( 0x1 )
bw_status           : 0  ( 0x0 )
nodes_usable_bitmask: 2  ( 0x2 )
network_taboo_mask  : 0  ( 0x0 )
network_taboo_start : 41  ( 0x29 )
```

`Node_status` shows information about the self node.

```
# mocap get --node_status
node_status
========================================
vendor_id            : 32  ( 0x20 )
moca_hw_version      : 1713512960  ( 0x66222200 )
moca_sw_version_major: 2  ( 0x2 )
moca_sw_version_minor: 1  ( 0x1 )
moca_sw_version_rev  : 22662  ( 0x5886 )
self_moca_version    : 32  ( 0x20 )
qam_256_support      : 1  ( 0x1 )
```

`Gen_node_status` shows information about other nodes on the network.

```
# mocap get --gen_node_status 1
gen_node_status
========================================
eui             : 00:10:18:3d:54:c0
freq_offset     : 0x44e82  ( 282242 )
node_tx_backoff : 10  ( 0xa )
protocol_support: 536871255  ( 0x20000157 )
```

`Gen_node_ext_status` shows extra PHY information about the link with other nodes.

```
# mocap get --gen_node_ext_status index 1 profile_type 12
gen_node_ext_status
=========================================
nbas           : 4600  ( 0x11f8 )
preamble_type  : 6  ( 0x6 )
cp             : 30  ( 0x1e )
tx_power       : -17 dBm
rx_gain        : 0.000 dBm
bit_loading[64]: 000099aa  aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa
                 aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa
                 aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa
                 aaaaaaa9  aaa9a999  99999999  99999999  98888888  88888877  77770000  00000000
                 00000000  00000777  77888888  88888889  89899899  89899999  99999999  99999999
                 99999999  9999999a  99999999  a9a9999a  a99aa999  9a999aa9  99a9aaaa  aa9aa9a9
                 aa9a9a9a  9aaa9aaa  aaaaaaaa  9aaa9aaa  aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa
                 aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa  aaa99000

avg_snr        : 0.000
turbo_status   : 0  ( 0x0 )
```

Statistics Information

`Gen_stats` shows general statistics pertaining to the self node.

```
# mocap get --gen_stats
== gen_stats  =========================================
ecl_tx_total_pkts         : 15014571
ecl_tx_ucast_pkts         : 14981799
ecl_tx_bcast_pkts         : 8
ecl_tx_mcast_pkts         : 32764
ecl_tx_ucast_unknown      : 0
ecl_tx_mcast_unknown      : 0
ecl_tx_ucast_drops        : 0
ecl_tx_mcast_drops        : 0
ecl_tx_total_bytes        : 19153582012
ecl_tx_buff_drop_pkts     : 222994
ecl_rx_total_pkts         : 8406523
ecl_rx_ucast_pkts         : 8342538
ecl_rx_bcast_pkts         : 1
ecl_rx_mcast_pkts         : 63984
ecl_rx_ucast_drops        : 0
ecl_rx_total_bytes        : 571987116
ecl_rx_mcast_filter_pkts  : 0
ecl_fc_bg                 : 0
ecl_fc_low                : 0
ecl_fc_medium             : 0
ecl_fc_high               : 0
ecl_fc_pqos               : 0
ecl_fc_bp_all             : 0
mac_tx_low_drop_pkts      : 0
mac_rx_buff_drop_pkts     : 0
mac_channel_usable_drop   : 0
mac_remove_node_drop      : 0
```

```
mac_loopback_pkts         : 0
mac_loopback_drop_pkts    : 0
aggr_pkt_stats_rx_max     : 0
aggr_pkt_stats_rx_count   : 0
aggr_pkt_stats_tx[20]     : 64802 38735 38280 39448 39936 39497 38842 38762
                            1305940 82161 15013 6827 7311 5369 3943 2675
                            928 1830 1028 9774
link_down_count           : 0
link_up_count             : 1
nc_handoff_counter        : 0
nc_backup_counter         : 0
resync_attempts_to_network: 0
tx_beacons                : 6134612
tx_map_packets            : 81073069
tx_rr_packets             : 0
tx_ofdma_rr_packets       : 0
tx_control_uc_packets     : 75015
tx_control_bc_packets     : 305178
tx_protocol_ie            : 0
rx_beacons                : 6272404
rx_map_packets            : 81451287
rx_rr_packets             : 81072742
rx_ofdma_rr_packets       : 0
rx_control_uc_packets     : 183381
rx_control_bc_packets     : 1
rx_protocol_ie            : 0
== end gen_stats  =====================================
```

`Error_stats` shows error counts statistics pertaining to the self node.

```
# mocap get --error_stats
== error_stats  =======================================
rx_uc_crc_error        : 717
rx_uc_timeout_error    : 0
rx_bc_crc_error        : 1
rx_bc_timeout_error    : 0
rx_map_crc_error       : 0
rx_map_timeout_error   : 0
rx_beacon_crc_error    : 0
rx_beacon_timeout_error: 0
rx_rr_crc_error        : 130
rx_ofdma_rr_crc_error  : 0
rx_rr_timeout_error    : 195
rx_lc_uc_crc_error     : 2
rx_lc_bc_crc_error     : 0
rx_lc_uc_timeout_error : 0
rx_lc_bc_timeout_error : 0
rx_probe1_error        : 0
rx_probe2_error        : 0
rx_probe3_error        : 0
rx_probe1_gcd_error    : 0
rx_plp_crc_error       : 10
rx_plp_timeout_error   : 0
== end error_stats  ===================================
```

`Node_stats` shows information regarding statistics with another node on the network.

```
# mocap get --node_stats 1
Node 1  02:10:18:38:82:01
== node_stats  ========================================
tx_packets       : 14758803
rx_packets       : 8342538
rx_cw_unerror    : 0
rx_cw_corrected  : 78578296
rx_cw_uncorrected: 653
rx_no_sync       : 0
== end node_stats  ====================================
```

`Node_stats_ext` shows extended statistics information with another node on the network.

```
# mocap get --node_stats_ext 1
Node 1  02:10:18:38:82:01
== node_stats_ext  ====================================
rx_uc_crc_error        : 717  ( 0x02cd )
rx_uc_timeout_error    : 0  ( 0x0000 )
rx_bc_crc_error        : 1  ( 0x0001 )
rx_bc_timeout_error    : 0  ( 0x0000 )
rx_map_crc_error       : 0  ( 0x0000 )
rx_map_timeout_error   : 0  ( 0x0000 )
rx_beacon_crc_error    : 0  ( 0x0000 )
rx_beacon_timeout_error: 0  ( 0x0000 )
rx_rr_crc_error        : 130  ( 0x0082 )
rx_ofdma_rr_crc_error  : 0  ( 0x0000 )
rx_rr_timeout_error    : 200  ( 0x00c8 )
rx_lc_uc_crc_error     : 2  ( 0x0002 )
rx_lc_bc_crc_error     : 0  ( 0x0000 )
rx_lc_uc_timeout_error : 0  ( 0x0000 )
rx_lc_bc_timeout_error : 0  ( 0x0000 )
rx_probe1_error        : 0  ( 0x0000 )
rx_probe2_error        : 0  ( 0x0000 )
rx_probe3_error        : 0  ( 0x0000 )
rx_probe1_gcd_error    : 0  ( 0x0000 )
rx_plp_crc_error       : 10  ( 0x000a )
rx_plp_timeout_error   : 0  ( 0x0000 )
padding                : 0  ( 0x0000 )
== end node_stats_ext  ================================
```

Loopback Configuration

The MoCA interface can be configured in *loopback* mode such that all packets received from the MoCA network are sent back to the MoCA network with the destination and source MAC addresses swapped in the packet.

The command to enable loopback is as follows:

```
   # mocap set --loopback_en 1
```

To disable loopback, set the *loopback_en* parameter to 0:

```
   # mocap set --loopback_en 0
```

TX Power Tune

There are three parameters that control the maximum TX power level: `max_tx_power`, `max_tx_power_tune`, `max_tx_power_tune_sec_ch`, and `beacon_pwr_reduction`.

The reduction of the TX power for beacons will be a combination of the above three parameters. For other bursts, it will be a combination of the first two parameters. Use them in the following way:

- There is a backoff (power reduction) table, containing 57 indexes, starting from +3 dB to –53 dB.
- `max_tx_power` sets the index in the table (not offset). That is, setting `max_tx_power` to 0 uses the 4th index.
- `max_tx_power_tune` ([dB]) sets offsets from the above selection.
- `beacon_pwr_reduction` sets offsets from the above selection. In 2.10.7.10 and above, the units are (*3 dB). In 2.10.6.x and earlier, the units are [dB].

  Example: If `max_tx_power` = –30, `max_tx_power_tune` = 1, and `beacon_pwr_reduction` = 10, the overall backoff will be:

  - For beacons: –30 – 1 – 10 = –41 dBm
  - For other bursts: –30 – 1 = –31 dBm

  The configuration MUST not exceed the boundaries of this table. Doing so will generate assert `152506`.

Broadcom MoCA box vendors should test their boards for maximum power on every allowed frequency of the device. In case the power is above the desired power, the vendor needs to tune this device.

The desired reduction should be experimented with in the lab, using the `mocap set --max_tx_power_tune` command. If vendors want to use nondefault values, they can store them in NVRAM by using the `-p` option.

Example: `mocap -p set --max_tx_power_tune offset 3 1150`

The `-p` option makes parameters persistent, assuming the standard mocacfg startup script is used. Otherwise, customers will need to set these values in their own startup script.

After all values are determined, they should be hard-coded into the MoCA startup of the device. The hard-coding might be done via scripting, an API call to the MoCA driver, or by adding these values to /etc/moca/moca.conf.default if the mocacfg script is used.

Appendix E: “PHY TX Power Configuration Presentation” provides more information about the TX power amplifiers.

RX Power Tune

The rx_power_tune setting is for per-frequency RX tuning. This fine tuning is required for accurate RX power reports.

The RX power tuning could be done according the following command:

```
mocap set --rx_power_tune offset 3 1150
```

This adjusts the reported offset by 3 dB when operating at 1150 MHz. Once the user has determined the settings, they can add them to their board's startup scripts. If using the BRCM-supplied startup script, *mocacfg*, the settings can be added to /etc/moca/moca.conf.default.

Example: `INIT_TIME = --rx_power_tune offset 1 1150 offset 3 1175 offset 2 1200`

The procedure to measure the power offsets is as follows:

1. Connect a node to a power meter
2. Put this node into constant power TX mode: `mocap set --continuous_power_tx_mode 1 --lof 1150`
3. Measure the reading from the power meter, compensating for any matching pads if necessary
4. Disconnect the node from the power meter and, using the same cabling, connect it to the customer board (DUT).
5. Restart both nodes, disabling TPC. `mocap set --restart --tpc_en 0 --lof 1150 --single_channel_operation 1 --continuous_power_tx_mode 0`. Wait for a network to form.
6. From the DUT, read back the RX power: `mocap get --gen_node_ext_status index 0 profile_type 7`. Change index ‘0’ to the other node’s id (for example, ‘1’).
7. Subtract the reported value in Step 6 from the measured value in Step 3. Use this value as the offset.
8. Repeat Steps 1-7 for all frequencies

Command Aliases

Some commonly used CLI commands have alias commands to make typing the commands easier. The aliases are shorter versions of the original CLI commands. The original CLI commands continue to function.

For example, `mocap set --moca_core_trace_enable 1` still works, but `mocap set --trace 1` now has the same effect.

Table 1 shows a list of old commands with new aliases:

Table 1: Command Aliases

Old Command	*Alias*
bandwidth	bw
const_tx_params	ctxparms
continuous_power_tx_mode	ctx
current_keys	keys
dd_init	dd
error_stats	errors
gen_node_ext_status	nodephy
gen_node_status	nodeinfo
gen_stats	stats
interface_status	link
moca_core_trace_enable	trace
moca_reset	mr
network_status	net
pqos_create_flow	pqosc
pqos_delete_flow	pqosd
pqos_list	pqosl
pqos_query	pqosq
pqos_status	pqoss
pqos_update_flow	pqosu
primary_ch_offset	pco
secondary_ch_offset	sco
single_channel_operation	schop
taboo_channels	taboo

Bonding Commands

Basic Commands

Enable/disable bonding is done with the bonding command:

- Usage: `mocap set --bonding [0|1]`

Configuring the secondary channel offset:

- Usage: `mocap set --sec_ch_offs [0|1|2]`
  - `0 = 0 MHz offset`
  - `1 = –125 MHz offset`
  - `2 = +125 MHz offset`

PHY Configuration Commands

The following commands configure the SNR tables and base margin for the primary channel when used in bonded connection:

- `snr_margin_table_pri_ch`
- `snr_margin_ldpc_pri_ch`

The following commands are the counterparts of the regular PHY commands, for the secondary channel.

- `snr_margin_table_ldpc_sec_ch`
- `snr_margin_ldpc_sec_ch`
- `target_phy_rate_20_sec_ch`
- `target_phy_rate_20_turbo_sec_ch`
- `target_phy_rate_20_turbo_vlper_sec_ch`

PER Calculation Based on Statistic Counters

PER between Two Nodes in the Networks

In order to get the total dropped packets, counters are needed from both ingress and egress nodes, and PER would be:

ingress_node_tx = ecl_tx_total_pkts

egress_node_rx = ecl_rx_total_pkts – ecl_rx_ucast_drops

per = (ingress_node_tx – egress_node_rx) / ingress_node_tx

PER On Ingress Node Only

The following formula shows the PER of packets that the ingress dropped internally and did not transmit in the MoCA network.

ecl_total_drop = ecl_tx_ucast_drops + ecl_tx_mcast_drops + ecl_tx_buff_drop_pkts

mac_total_drop = mac_tx_low_drop_pkts + mac_channel_usable_drop + mac_remove_node_drop

ingress_per = (ecl_total_drop + mac_total_drop) / ecl_tx_total_pkts

PER On Egress Node Only

The following formulas show the PER of packets that arrived from a specific node and were dropped because of CRC errors or internally by the node because of overflow or the user’s filtering configuration.

- At multinodes network:

  rx_packet_error_count = rx_uc_crc_error + rx_uc_timeout_error + rx_bc_crc_error + rx_bc_timeout_error

  egress_per = rx_packet_error_count / (rx_packet_error_count + rx_packets)
- At two nodes network:

  rx_packet_error_count = rx_uc_crc_error + rx_uc_timeout_error + rx_bc_crc_error + rx_bc_timeout_error + mac_rx_buff_drop_pkts

  egress_per = rx_packet_error_count / (rx_packet_error_count + rx_packets)

Note: rx_uc_crc_error, rx_uc_timeout_error, rx_bc_crc_error, rx_bc_timeout_error count loss of full aggregated packets as 1.

Using ECB Boards

Board Handling

Information on all commands is available by typing “help” on the CLI.

Loading New Image

Images that are used by the ECB boards:

- *flashimage.bin* – Loaded via TFTP.
- *bootplusimage.bin* – Loaded via BBS (Broadcom internal use only).

Note: The same image is used for running in single and bonding modes.

There are two ways to load an image via *tftp*:

1. Ethernet port – available in 680xB0 and 680xC0.
   When this method is used, make sure that the MoCA network is down but the node is running (either disconnect the coax cable from the loaded node, or make sure that the MoCA network is not available).
2. MoCA port – available in 680xC0 only.

Loading Process

1. Loading of the *flashimage.bin* is done via *tftp*, with the command:
   ```
   tftp <ip address> flashimage.bin
   ```
2. Once file download completes, it is required to reboot the board to start using the new image.

Troubleshooting:

- If *tftp* does not start, make sure that the ECB got a valid IP address.
  Use: `ifconfig auto` to get IP address from the DHCP server.
- It is possible to revert to the former image by using the `bootimage` CLI command (see help).

Web Access

Web access is via this address: http://10.x.x.x

Troubleshooting

This section contains a list of common issues that may be encountered when testing MoCA.

Issue Types

There are several different types of issues that may arise related to MoCA:

- Assertions
- Errors
- Warnings
- Host Issues

Assertions

Assertions are conditions that are detected by the MoCA firmware from which there is no path to recovery. Following an assertion, the MoCA firmware requires a restart from mocad. Some assertions may be due to configuration issues. Other assertions may require firmware modifications in order to be fixed.

The string "`-----------| MoCA Core Assertion !!! |----------`" indicates the beginning of the output for an assertion. Included in the output are several fields, most notably is the *Error Code,* which uniquely identifies the location in firmware of the assertion. If enabled, a long dump of hexadecimal numbers will follow the initial assertion info. This is the *RTT Dump* and can be very useful for identifying the cause of the assertion during postmortem analysis. All of the assertion data should be collected when reporting such errors.

Errors

Error messages occur when an unexpected event is detected. Errors can be recovered from but they typically indicate an issue of some kind and require investigation.

Errors can be identified by the *MoCA_E:* tag. A number will follow which identifies the error. If MoCA core traces are enabled, a string may also be printed out which can provide some details about the error.

Warnings

Warning messages occur when an atypical event is detected. Warnings may or may not indicate actual issues. Usually, warnings indicate noteworthy events that may correlate to other issues occurring at the same time.

Warnings can be identified by the *MoCA_W:* tag. A number will follow which identifies the warning. If MoCA core traces are enabled or if the warning bit of the `--verbose` parameter is set, a string may also be printed out which can provide some details about the warning.

Host Issues

Host issues will originate from one of the host applications such as mocad or mocap. There is no standard format for host issues and there may be several causes. These issues may stem from compilation settings, configuration settings, or user actions.

Common Issues

Link-Down Causes and Common Warnings

Table 2: Link-down Causes and Common Warnings

Warning ID	*Description*	*Possible Reason*
1001	Host pool empty.	MoCA firmware generating too many messages to host. Host CPU is being overloaded with other tasks, such as packet processing.
6515	Unusable channel.	Very low (GCD 100) PHY rate. Bad link.
6516	Unusable channel.	Very low (GCD VLPER) PHY rate. Bad link.
6517	Unusable RX channel.	Very low PHY rate. Bad link.
6518/ 6519	Unusable channel	Very low receive PHY rate with another node.
6526/ 6527	Unusable channel	Very low receive PHY rate with another node. Possible misalignment of brcm_bonding_seed configuration.
6535	PHY rate degraded by more than 15%.	Link got worsen. If RLAPM is enabled, this can be caused by crossing the “steps.”
9002	Not enough probes received	Severe link issues.
9012	High throughput is disabled.	Link characteristics are not sufficient.
9502	Too many consecutive lost LCs from node X.	Severe link issues.
9505	Too many consecutive lost Beacons.	Severe link issues or node was disconnected.
14000	NC doesn't support MoCA 2.0.	Connecting to MoCA 1.1 NC.
14002	Device Discovery failed.	Connecting to MoCA 1.1 NC.
15208	EN has received Admission Response retry when NC, NN or Network Version = 1.x.	Connecting to MoCA 1.1 NC.
15209/ 15427/ 15624	Admission has started but failed on retransmission count or T2	–
15402	Probe Transmission Request process has failed. Abort admission.	Severe link issues.
15417	EVM probe report was not received after five ACF retransmissions. Abort admission.	Severe link issues.
15604	Probe Transmission Request process has failed. Abort admission.	Severe link issues.
15608	No receive of admission response for too long.	Link issues, CRCs, privacy issue (enable/ disable or nonmatched keys).
15610	Different Turbo mode at NC and NN	–

Table 2: Link-down Causes and Common Warnings (Cont.)

Warning ID	***Description***	***Possible Reason***
15611	Network is full	–
15613	Persistent failure to receive Discovery or Admission Response.	Link issues, CRCs.
15614	Receive of admission response with CRC.	Link issues, CRCs.
15615	NN has unusable RX channel with the NC. Admission aborted.	Very low PHY rate. Bad link.
15618	Aborting admission due to connectivity problems with the NC	NC dropped from network or Severe link issues.
15619	Unusable channel for NN. Abort admission	Very low receive PHY rate between the NN and the NC.
18209	Unusable channel.	Very low transmit PHY rate with another node
18600	Error—too many 50M cons map losses—Aborting.	Severe link issues.
18601	Error—too many consecutive MAP frame losses —Aborting.	Link issues, CRCs.
18601	Error—too many 100M cons map losses—Aborting.	Severe link issues.
19212	Bad CIR calculation.	Very hard link.
19400	Unusable channel.	Very low (TX) PHY rate. Bad link.
21000	Link down, reason unknown (coax removals, other nodes went down, etc.)	–
21200	Node was dropped off the network or left alone on this channel—Restarting.	Disconnections, restarts.

Duplicate MAC Address

Type: Error

How to identify: MoCA_E: 14514

Possible Cause: The MoCA mac_addr parameter has not been configured properly and multiple nodes on the network are using a default value for the MAC address.

Empty Message Pool

Type: Warning

How to Identify: MoCA_W: 1001 or MoCA_W: 1002

Possible Cause: The MoCA firmware has temporarily exhausted its pool of messages for sending traps to the Host. This is most likely due to the Host not responding to the firmware trap messages fast enough. This can happen if the Host process responsible for responding to MoCA traps is starved of CPU resources.

QOS & MultiQ Configuration (Linux Kernel)

Type: Host Issue

How to Identify: ***Caution:*** MoCA cannot configure host-side transmit queues for QOS. Continuing with non-compliant queuing behavior. Some CTPs may fail. The kernel must have QOS and MULTIQ configured, and the tc utility must be in the path for spec-compliant operation. Refer to the document *MoCA_Diagnostics.pdf* for more details.

Possible Cause: The Linux kernel has not been configured with QOS and MultiQ enabled. The following Linux build configuration settings are required:

```
CONFIG_NET_SCHED=y
CONFIG_NET_SCH_CBQ=y
CONFIG_NET_SCH_PRIO=y
CONFIG_NET_SCH_MQPRIO=y
CONFIG_NET_CLS=y
CONFIG_NET_CLS_U32=y
CONFIG_CLS_U32_PERF=y
CONFIG_CLS_U32_MARK=y
CONFIG_NET_EMATCH=y
CONFIG_NET_EMATCH_CMP=y
CONFIG_NET_EMATCH_NBYTE=y
CONFIG_NET_EMATCH_U32=y
CONFIG_NET_CLS_ACT=y
CONFIG_NET_ACT_GACT=y
CONFIG_NET_ACT_PEDIT=y
CONFIG_NET_ACT_SKBEDIT=y
CONFIG_NET_SCH_FIFO=y
CONFIG_NET_EMATCH_STACK=32
CONFIG_NET_SCH_MULTIQ=y

tc -- include iproute2 package
```

QOS & MultiQ Configuration (Linux Kernel)

Possible Cause: No MoCA Link

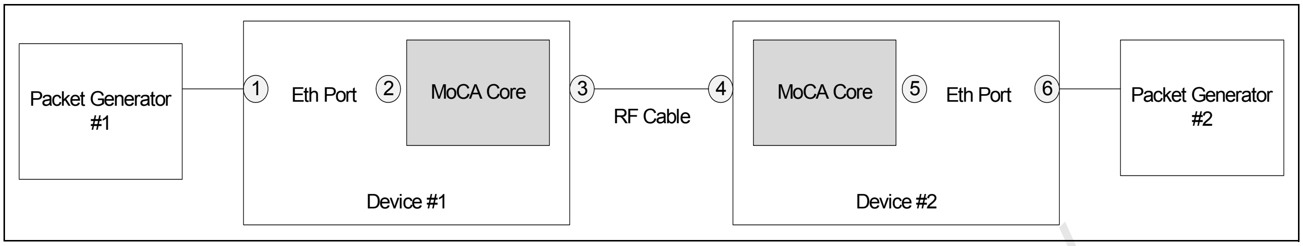
If unable to form a MoCA link, the first debugging steps are the following:

- Enable core traces (`mocap set --trace 1`) and see if anything odd appears in the logs.
- Make sure all nodes are configured in a compatible way.
  Run `mocap get --privacy_en --password --lof --rf_band --single_channel_operation`
  If privacy is enabled, it must be enabled on all nodes that wish to join the network, and all nodes must have the same password configured. Nodes must be operating on compatible RF bands. If nodes are configured with `single_channel_operation` enabled, they must have the same lof configured as well.
- Try running `mocap set --restore_defaults --restart` to see if default parameters will allow a link to be formed.
- Try adding or remove attenuation on the RF link.

Unable To Pass Traffic

If traffic does not flow through the MoCA interface, it is important to identify the location where packets are being dropped. A typical 2-node system will look something like Figure 1.

Figure 1: Typical 2-Node System



If we analyze the flow of traffic from *Packet Generator #1* to *Packet Generator #2*, there are typically six locations to identify where packets might be dropped. Using a Linux-based system as an example, locations (1) and (2) on *Device #1* will correspond to different interfaces. Packet counts for these interfaces can be obtained using the command `ifconfig`. First, verify that the number of received packets at location (1) is as expected. Then, verify that the number of transmitted packets at location (2) is as expected. At location (3), the command `mocap get --stats` will show the packets received and transmitted through the MoCA interface. In this direction, the MoCA interface of *Device #1* is transmitting to the MoCA network so the stats field *ecl_tx_total_pkts* should be checked. If the number of *ecl_tx_total_pkts* is not as expected, the MoCA stat fields that contain the word "drop" should be checked for any non-zero values.

If the location of packet drops is still not found, the analysis should continue on *Device #2*. For location (4), the field *ecl_rx_total_pkts* of the `mocap get --stats` output should be checked first. If the packet count is not as expected, the output of the command `mocap get --errors` should be checked. This will show whether packets have been dropped due to CRCs or timeouts. From there, `ifconfig` should be used again to verify packet counts at locations (5) and (6). Continuing to use Linux as an example, the received packet count should be analyzed at location (5) and the transmitted packet count should be analyzed at location (6).

Traffic Rate is Lower than Expected

If the traffic rate is lower than expected, there could be multiple causes.

- Is traffic being sent in both directions?

  A MoCA interface will limit the packet rate for traffic being sent to an unknown destination MAC address. The MoCA interface will learn a destination MAC address if it receives a packet from another MoCA node with the corresponding source MAC address of the packet.
- There may be an issue with the RF channel being used, such as noise.
  Try forcing the nodes to use different RF frequencies to determine whether or not the packet loss is occurring across all frequencies.
- There may be a configuration problem causing packet drops.

  Try running the command

```
mocap set --restore_defaults --restart on all nodes and send traffic again.
```

Appendix A: Flow Control Analysis

There are two reports that can show whether flow control signals were asserted:

- Running the command `mocap get --stats`.
  The output will contain the following information:

```
ecl_fc_bg                    : 0  // background priority counter
ecl_fc_low                   : 0  // low priority counter
ecl_fc_medium                : 0  // med priority counter
ecl_fc_high                  : 0  // high priority counter
ecl_fc_pqos                  : 0  // PQoS priority counter
ecl_fc_bp_all                : 0  // A counter which sums all the above
```

- Periodic reports:
  - In MoCA 2.0 versions 2.10.3 and above, the LMO report will include FC statistics and will appear as follows:

    ```
    CORE1: @ FC        : S=0-0-0-0-0, C=0-0-0-0-0, BP=0
    ```

    - ‘S’ stands for *Current status*
    - ‘C’ stands for *Count since last LMO*
    - ‘BP’ stands for *Backpressure Current status*
  - In MoCA 2.0 versions prior to 2.10.3, the FC statistics will be printed outside the LMO report and will appear as follows:

    ```
    FC: Status BP_ALL=0, [16..0]=0x00000. PQoS=000 High=000 Medium=000 Low=122 BG=000 isRun=1
    ```

    - First two values are the current status.
    - Last five values are counters since last LMO
    - `isRun` is the firmware flow control feature *Enable*.

Appendix B: MoCA 2.0 LMO Report Descriptions

MoCA 2.0 LMO report descriptions are listed in Table 3.

Table 3: MoCA 2.0 LMO Report Description

Name	***Description***
`LMO: LnID`	Node ID of the LMO node
`LMO: LnVer`	MoCA version of the LMO node
`LMO: LMO_Type`	Link state when LMO stated (Internal)
`LMO: Dur`	Duration in seconds of the last LMO cycle
`LMO: LMO_Cnt`	Counter for the number of LMO cycles since link up
`LMO: LF`	Indication if LMO_Flag was up during the LMO cycle
`LMO: Slf`	‘1’ if the LMO was on this specific node (self). O.W ‘0’
`NODE: ID`	Node id of the LMO node
`NODE: MAC`	MAC address of the LMO node
`NODE: Ver`	MoCA version (10/11/20) of the LMO node
`NODE: PrfNC`	’1’ if the LMO node is the preferred NC
`NODE2: TR50`	Target PHY rate for 50M transmissions for this node
`NODE2: TR100N`	Target PHY rate for 100M NPER transmissions for this node
`NODE2: TR100V`	Target PHY rate for 100M VLPER transmissions for this node
`NODE2: TR100N_B`	Target PHY rate for 100M NPER Bonding transmissions for this node
`NODE2: TR100V_B`	Target PHY rate for 100M VLPER Bonding transmissions for this node
`NODE3: Brcm`	‘1’ if the LN is Broadcom identified chip
`NODE3: VendId`	Vendor Id of the LN by MoCA spec
`NODE3: HW`	HW type of the LN
`NODE3: HW_Ver`	Internal
`NODE3: Rel`	Release version of the LN
`NODE3: Rev`	SVN version of the LN
`SELF: SelfID`	Self node ID
`SELF: Self_MAC`	Self MAC address
`SELF: UnusableBM`	Unusable nodes bitmask. ‘1’ means the link is unusable to the node
`SELF: TPC`	‘1’ means TPC is on
`SELF: Prvc`	‘1’ means privacy is on
`SELF: Star`	‘1’ means star topology is on
`NC: NcID`	NC node ID
`NC: NcVer`	NC MoCA version
`NC: NC_MAC`	NC MAC address
`NC: BkNc`	Node id of the backup NC
`NC: IF2`	Internal

Table 3: MoCA 2.0 LMO Report Description (Cont.)

Name	Description
NET: LOF	Last operational frequency
NET: ANB	Active node bitmask
NET: LPB	Low power nodes bitmask
NET: NetVer	Network version
NET: NumNodes	Number of active nodes in the network
NET: TP	Network topology #numNodesByVersion (2.0+, 2.0, 1.1, 1.0)
NET: Tab	Taboo
NET2: Nodes	16 node print of the nodes version (2 = 2.0,1 = 1.1, 0 = 1.0, - = Nonexisting, b = bonding node)
NET2: LUT	Link-up time—time passed from link-up
NET2: TCT	Topology change time—time passed since last node added/removed
NET2: Turbo	‘1’ means Turbo feature is on
PM: NetPwrState	16 node print of the nodes power state
PM: NodePwrMode	Self-power mode
PM: M1TxPwrVar	Self-M1 TX Power variation
SW_VENDOR:	16 node print of the nodes vendor information (B = Broadcom, ? = unknown)
SW_VENDOR: Rel	Release number
SW_VENDOR: SW	SW revision (Major.minor.rev)
SW_VENDOR: SA	‘1’ means stand-alone chip
SW_VENDOR: LT	‘1’ means limit traffic mode is on
HW: HW	Chip HW version
HW: PC	PHY clock (MHz)
HW: SC	System clock (MHz)
HW: AMP	Power amplifier type
HW: FFTW	Internal
HW: PDB	Internal
HW: AW	Internal
HW: Gen	Internal
CLKS: MAC_CLK	MAC clock in the end of the LMO cycle
CLKS: CPU_0	% of CPU 0 utilization during the LMO cycle
CLKS: CPU_1	% of CPU 1 utilization during the LMO cycle
CLKS: CRRAbort	Internal
Channel: BcnChnl	The working beacon channel
Channel: PrimChnl	The working primary channel
Channel: Offset	Primary channel offset
Channel: SecChnl	The working secondary channel
Channel: Taboo	Taboo mask start/ taboo channel mask
MapStat: Aus	Number of AUs in LMO maps (MAX,AVG,MIN)
MapStat: MG	Number of Good (valid CRC) maps received (Leg, ELM, Map2, Dual)

Table 3: MoCA 2.0 LMO Report Description (Cont.)

Name	Description
MapStat: MB	Number of Bad (valid CRC) maps received (Leg, ELM, Map2, Dual)
MapStat: Sz	Number of map 2 in different sizes (400, 800, 1200, 1600)
TimeStat: TauPct	Percent of TAU in the LMO cycle
TimeStat: Tau	Sizes of TAU in the LMO cycle (MAX,AVG,MIN)
TimeStat: MapCycle	Size of the map frames in the LMO cycle (MAX,AVG,MIN)
PM_USG: ANLG	Percent of the time the Analog module was down during the LMO cycle
PM_USG: PLL	Percent of the time the PLL module was down during the LMO cycle
PM_USG: 3451	Percent of the time the 3451 module was down during the LMO cycle
PM_USG: ANLG	Percent of the time the Analog module was down during the LMO cycle
R/Tx 50/100: Pre	Preamble type of the profile
R/Tx 50/100: nBas	Number of bits in a sync symbol of the profile
R/Tx 50/100: cp	cp of the profile
R/Tx 50/100: PR	PHY rate of the profile
R/Tx 50/100: Seq	Sequence id of the profile (1/0)
R/Tx 50/100: BO	TX/RX Backoff of the profile
R/Tx 50/100: Ldcw	LDPC code word length of the profile
R/Tx 50/100: CU	‘1’ means the profile represents a usable channel
Rx 50/100_2: SNR	SNR of the channel between the two nodes
Rx 50/100_2: TI	Automatic gain control table index
Rx 50/100_2: AGC	Automatic gain control address used in the profile
Rx 50/100_2: CTI	Coarse AGC table index
Rx 50/100_2: CAGC	Coarse AGC address
Rx 50/100_2: TAGC	Internal
Rx 50/100_2: RS	RSSI value discovered
Rx 50/100_2: Pwr	RX Power level detected in the link
CSFO: SF50/100/Bu	Sample frequency offset for 50/100M/Bonding bursts
CSFO: CF50/100/Bu	Center frequency offset for 50/100M/Bonding bursts
OFFST_050:	Internal
CTC: RxOffset	The number of timeslot the node is requested to offset its transmission
DATA: RxDmaDrp	Internal
DATA: TxDmaDrp	Internal
DATA: TxLowDrp	Internal
DATA: NtfOvf	Internal
DATA_2: Rx	Number of received packets from all the nodes during this LMO cycle
DATA_2: Tx	Number of transmitted packets to all the nodes during this LMO cycle
DATA_2: FragBrst	Number of fragmented MoCA frames received during this LMO cycle
DATA_2: RTR	Number of Block ACK RTR frames
DATA_2: TxPrio	Number of transmitted packets per priority BG, LOW, MED, HIGH, PQOS
FC:	Flow control counters—Described in “Flow Control Analysis” on page 28

Table 3: MoCA 2.0 LMO Report Description (Cont.)

Name	Description
`OFDMA_i`	OFDMA information of OFDMA group #i; The OFDMA information is printed by the LN that runs OFDMA process.
`OFDMA_i: ANB`	The bitmask of the nodes that are belong to this OFDMA group.
`OFDMA_i: CP`	The Cyclic Prefix of this OFDMA group.
`OFDMA_i: NInum`	The number of OFDMA slots that this OFDMA group contains.
`OFDMA_i: NInbas`	Number of bits of each OFDMA slot in this OFDMA group.
`OFDMA_i: seq`	Sequence ID of this OFDMA group

Remarks:

1. When sending debug logs to Broadcom, please try to send logs from all the nodes.
2. Timestamps for each line is needed for efficient debugging. Use a terminal program that can also record the timing of the console prints. One such program is *Tera Term*.
   a. Download the latest version from: http://ttssh2.sourceforge.jp/index.html.en
   b. Install the program, and connect to the device.
   c. To save logs with timestamp, go to “**File** → “**Log**”, and on the bottom left corner you will see a checkbox labeled “Timestamp”; please make sure it is checked.

Appendix C: TPCAP Usage

TPCAP (test-port capture) is a mechanism embedded within the MoCA HW for collecting information about the HW procedure.

TPCAP dump is combined with RTT dump as well as test ports information. Both dumps should be provided for complete analysis.

1. Load *tpcap* file to the device under test (DUT). The file is part of every release.
2. When MoCA driver runs, start running the *tpcap* with the command as described below.
3. Upload the *tpcapCapture.txt* file as well as the RTT dumps (from file or console).

TPCAP common configurations:

1. `tpcap --wait` – capture (defaults) the first CRC of any burst type, at PHY probe 0 (ADC).
2. `tpcap --wait --stoptype 2 phy_probe 4 --burst_type 18` – stops only when a timeout occurs, captures the secondary PHY (bonding usage) profile using a windowing probe.
3. Packet dump is triggered by user: this case is used for capturing an ordinary packet that does not necessarily show any issue (not waiting for a specific defined trigger).
   a. `tpcap --phy_probe 0 --burst_type 1 --nbursts 100000` – defines that only beacon at ADC is captured. The number of bursts is a *must* at this step.
   b. `tpcap dump --filename` – once this command is provided, the `tpcap` is triggered, and the defined packet is captured.
4. Additional definitions/cases and examples are available via `tpcap --help`. Attached below.
5. Troubleshooting:
   a. In order to abort TPCAP, use ^C.
   b. After TPCAP is captured, need to reload the DUT's MoCA driver. DUT reboot is mostly recommended.
   c. MoCA Standalone users – since there is no platform to save the dump file, need to dump the *tpcapCapture* to the CLI.

TPCAP formats:

```
tpcap [--bursttype <burst_type>] [--stoptype <0-no, 1-CRC, 2-timeout>] [--nsamples <val>]
[--port <val>] [--direction <0-Tx, 1-Rx, 3-loopback, 4-All>] [--memalloc] [--continue]
[--phy_probe <cap_phy_probe>] [--wait] [--node <node_id>] [--lpbk <val>] [--phy_num <val>]
[--noAssert] [--nbursts <val>] [--testPortClockSel <val>] [--testPortAdcRate <val>]
[--testPortDataSel <val>] [--testPortControlSel <val>]

--nbursts - Used to record specific number of bursts
--bursttype
   0 - BT_UNUSED0
   1 - BT_BEACON
   2 - BT_DIVERSITY          Diversity Mode profile in 50 MHz
   3 - BT_PROBE_I            50M Probe I
   4 - BT_PROBE_II
   5 - BT_PROBE_III
   6 - BT_MAP                50M Map
   7 - BT_UNICAST            Unicast 50M profile
   8 - BT_BROADCAST          50M Broadcast
   9 - BT_20_EVM_PROBE
   10- BT_BRCM_CALIBRATION-BROADCOM PROPRIETARY: Calibration burst
   11- BT_UNUSED2
   12- BT_20_MAP_100         MAP profile in MoCA 2.0 PHY
   13- BT_20_UC_NPER         NPER Unicast profile in MoCA 2.0 PHY (1e-6)
   14- BT_20_GCD_NPER        NPER GCD profile in MoCA 2.0 PHY     (1e-6)
   15- BT_20_OFDMA           OFDMA profile in MoCA 2.0 PHY
   16- BT_20_UC_VLPER        VLPER Unicast profile in MoCA 2.0 PHY. (1e-8)
   17- BT_20_GCD_VLPER       VLPER GCD profile in MoCA 2.0 PHY.   (1e-8)
   18- BT_CB_UC_NPER         NPER Unicast profile in channel bonding
   19- BT_CB_EVM_PROBE       EVM Probe profile in channel bonding
   20- BT_CB_UC_VLPER        VLPER Unicast profile in channel bonding
   21- BT_DIVERSITY_100      Diversity Mode profile in MoCA 2.0 100 MHz channel
   22- BT_DIVERSITY_2        Diversity Mode profile in Secondary Channel of a bonded link
   23- BT_UNUSED3
   24- BT_BRCM_EVM_PROBE_ALL_ZERO - BROADCOM PROPRIETARY: OFDMA - 0 NBAS in all sub-carriers
                             (used in Tx Only)

--phy_probe
   0 - ADC out (ADC clock of 200 MHz)
   1 - Slicer in (PHY_CLK)
   2 - MPC (SYS_CLK)
   3 - Autocorrelation out (PHY_CLK)
   4 - Phase Rotator out (PHY_CLK) - WIN
   5 - MPD (line_clk_derived)
   6 - Tx out @ 200 MHz (DAC_CLK)
   7 - FFT out (PHY_CLK)

tpcap dump [--file <name>] [--display <0,1,2>]

--display
   0 - Normal print (default) 04X 04X
   1 - Printing with sampling index
   2 - Print for 'port=1' option. Prints the file in waveform format
```

Examples:

```
   Port=0 (ADC)  : tpcap –wait

   Port=0 (MPD)  : tpcap --wait --phy_probe 5

   Port=1 (Logic): tpcap --port 1 --testPortDataSel 13 --testPortControlSel 4
   --testPortClockSel 5 --nsamples 4194304 --wait --display 2
```

Appendix D: Continuous TX Usage

The CLI allows several variances of configuring the continuous TX mode.

```
mocap set --ctx [0..6] const_tx_params ??? --bonding [0|1] --rf_band [0..7] --lof [channel]
--prim_ch_offs [0..2] --sec_ch_offs [0..2] --bandwidth [0|1] --max_tx_power [0..56]
```

Note: This configuration is not persistent.

Note: The spectrum settings of the Resolution Bandwidth (RBW) and Video Bandwidth (VBW) on a spectrum analyzer should follow CTP 405 (Tx Power, Frequency, Spectral Mask, and Noise test).

Table 4: Continuous TX Usage

	Command	*Values*
Clean	restore_defaults	–
Mode	continuous_power_tx_mode Alias: ctx	0 = Normal operation 1 = Continuous power TX mode 2 = Continuous RX mode 5 = Continuous power TX mode Secondary (bonded chips only) 6 = Continuous power TX mode Bonded (bonded chips only)
	const_tx_params	const_tx_submode: 0 = Single tone 1 = Normal probe I 2 = Continuous wave mode 3 = Band mode
		const_tx_sc1
		const_tx_sc2
		const_tx_band[16]
	bonding	def: 0

Table 4: Continuous TX Usage (Cont.)

	Command	***Values***
Freqs	rf_band	0 = D-Low, support all MoCA channels in sub-band D-Low 1 = D-High, support all MoCA channels in sub-band D-High 2 = ExD, support all MoCA channels in band D 3 = E, support all MoCA channels in band E 4 = F, support all MoCA channels in band F 5 = C4, support single MoCA channel C4 (1000 MHz) 6 = H, support all MoCA channels in band H 7 = Generic, support all MoCA channels in single channel mode only
	lof	–
	prim_ch_offs	0 = 0 MHz offset 1 = +25 MHz offset 2 = –25 MHz offset
	sec_ch_offs	0 = 0 MHz offset (bonding off) 1 = –125 MHz offset 2 = +125 MHz offset
	bandwidth	def: 0 min: 0 max: 1
Power	max_tx_power	def: 3 min: –31 max: 3

Appendix E: PHY TX Power Configuration Presentation

Monitor current PHY TX power configurations:

```
mocap get --tx_power_params channelMode [0..2] txTableIndex [index]

channelMode:
0 – beacon
1 – primary
2 – secondary
txTableIndex:
0x0  - Single mode, table index 0
0x10 - Bonded mode, table index 0
0x11 - Bonded mode, table index 1
0x12 - Bonded mode, table index 2
0x13 - Bonded mode, table index 3
0x14 - Bonded mode, table index 4

                                 TX Power Parameters
                             ==========================

SEC channel: 1525
Table Mode: Single
Table Index: 0
User defined max TX power: 0
Max TX channel tune: -2

SEC channel: 1300
Table Mode: Single
Table Index: 0
User defined max TX power: 0
Max TX channel tune: -2

                3450                                       Soc
==========    ============   =====================================================================
|Back off|    |3450 PA   |   | PA Driver Gain Control | PA Driver Max Gain |    TX Digital Gain |
==========    ============   =====================================================================
| 0000   |    | 0x0007   |   |      0x007a             |     0x000d         |     0x00c2        |
==========    ============   =====================================================================
| 0001   |    | 0x0007   |   |      0x006f             |     0x000d         |     0x00c2        |
==========    ============   =====================================================================
| 0002   |    | 0x0007   |   |      0x0064             |     0x000d         |     0x00c2        |
==========    ============   =====================================================================
| 0003   |    | 0x0007   |   |      0x005a             |     0x000d         |     0x00c2        |
==========    ============   =====================================================================
| 0004   |    | 0x0007   |   |      0x0052             |     0x000d         |     0x00c2        |
==========    ============   =====================================================================
...
```

```
Monitor current Rx gain configurations:

mocap get --rx_gain_params

                                 RX Gain Parameters
                             =========================
3450 Fields
============
LNA IBIAS:                0x0004
LNA STG1 GM:              0x0004
LNA STG1 CASC BASE:       0x0004
LNA STG2 AMP:             0x0004
LNA STG2 BUF:             0x0004
Table Index:              0
              3450                                       Soc
========  ===========     ======================================================================
|index |  | 3450 LNA|     |RF PGA  GAIN   | IF LPF GAIN |       IF PGA  GAIN   |Total Gain (dB)
========  ===========     ======================================================================
| 0000 |  | 0x0007  |     |  0x0000       |  0x0004     |       0x0022         |  66.9636
========  ===========     ======================================================================
| 0001 |  | 0x0007  |     |  0x0000       |  0x0004     |       0x0022         |  66.9636
========  ===========     ======================================================================
| 0002 |  | 0x0007  |     |  0x0000       |  0x0004     |       0x0021         |  66.0231
========  ===========     ======================================================================
| 0003 |  | 0x0007  |     |  0x0000       |  0x0004     |       0x0021         |  66.0231
========  ===========     ======================================================================
| 0004 |  | 0x0007  |     |  0x0000       |  0x0004     |       0x0020         |  64.8945
...
```

Appendix F: Tuning Impedances of Broadcom BCM6802C0 Devices

Overview

Bonding devices have two PHY blocks that are hardware combined on the board for transmitting and receiving two combined channels: Primary channel (PHY0) and Secondary channel (PHY1).

The quality of the transmitted bonded signal is affected by the board layout, such as trace lengths, vias, and components between the SoC and the 3451. Both PHYs have output impedance tuning options for best matching optimization between the SoC and the 3451, thus enabling the flexibility to adjust the transmitter performance despite the layout differences compared to the reference board.

API

```
mocap -p set --impendance_mode_bonding 0xZYX
```

Each parameter, “X”,”Y”, and “Z”, represents a nibble (4 bits) in the configuration word, having a value between 0 and F.

- X represents 4 bits for configuring the Primary channel transmission (PHY0), in single channel bursts, where only the Primary channel is transmitting, e.g., when linking with a non-bonding 2.0 device.
- Y represents 4 bits for configuring the Secondary channel transmission (PHY1), in single channel bursts, where only the Secondary channel is transmitting. This type of transmission is rarely used, and then only as part of the MoCA control protocol, not for data.
- Z represents 4 bits for configuring the Bonding transmission.

The values of X, Y, and Z are tuned separately, and each one serves a different mode of transmission.

For example, `mocap -p set --impendance_mode_bonding  0xd3f` would mean:

- Bonding setting is “d” (1101 in binary bits).
- Secondary setting is “3” (0011 in binary bits).
- Primary setting is “5” (0101 in binary bits).

Tuning Procedure

Choose a specific frequency to work on (set the desired LOF, PCO, and SCO). The LOF should not affect the conclusions, hence any LOF may be chosen.

Repeat the following procedure three times, for tuning (1) Primary, (2) Secondary, and (3) Bonding nibbles, and obtaining the impedance value (0xZYX).

- Connect a spectrum analyzer to the device.
- Repeat the below for each possible value between “0” and “f”.
  - Set the spectrum analyzer to the frequency and bandwidth under test.
  - Turn on the device in a continuous mode as follows.
    - For Primary:
      ```
      mocap set --restart --bonding 1 --bw 1 --ctx 1 --lof <LOF> --pco <0/+25/-25> --sco <+/-125>
      ```
    - For Secondary:
      ```
      mocap set --restart --bonding 1 --bw 1 --ctx 5 --lof <LOF> --pco <0/+25/-25> --sco <+/-125>
      ```
    - For Bonding:
      ```
      mocap set --restart --bonding 1 --bw 1 --ctx 6 --lof <LOF> --pco <0/+25/-25> --sco <+/-125>
      ```
  - Measure the integrated power value and record the PSD graph. Ideally, power should meet 3–4 dBm. Mask quality should be assessed for flatness, symmetry between channels, and carrier to noise (C/N) ratio.

Some of the permutations of the 4 bits are identical, as listed below:

0x1=0x2
0x4=0x8
0x5=0x6=0xa=0x9
0x7=0xb
0xd=0xe

Therefore, instead of testing 16 times, only eight tests are needed. You will need to fill the table below per each nibble tuning.

Table 5: Nibble Tuning-Assist Table

Nibble Value	*Integrated Power Measured Value*	*Notes Describing the Quality of the TX Signal*	*Link to a Spectrum Analyzer Screen Capture*
0x0			
0x1			
0x3			
0x4			
0x5			
0x7			
0xc			
0xd			
0xf			

Example

For optimizing the Secondary channel, we need to tune the second nibble: 0xZYX. “Z” and “X” are don’t care in this tuning.

1. Set PHY1 to 1200.

   For example:

   ```
   mocap set --restart --bonding 1 --bw 1 --ctx 5 --lof 1300 --pco 25 --sco -125
   ```

2. Set the spectrum analyzer accordingly while Center freq = 1200 MHz.
3. Try the following eight configurations: 0x000, 0x010, 0x030, 0x040, 0x050, 0x070, 0x0c0, 0x0d0, and 0x0f0

   Use the command:

   ```
   mocap -p set --impedance_mode_bonding 0xZYX --restart
   ```

Measuring Power and PSD on the spectrum analyzer on each of the eight configurations will show the best configuration for the Secondary channel.

Configuration

The best values 0xZYX should be configured during device initialization. They are stored in NV RAM, so will persist after power cycles.

Broadcom® reserves the right to make changes without further notice to any products or data herein to improve reliability, function, or design.

Information furnished by Broadcom is believed to be accurate and reliable. However, Broadcom does not assume any liability arising out of the application or use of this information, nor the application or use of any product or circuit described herein, neither does it convey any license under its patent rights nor the rights of others.

BROADCOM®

Broadcom

Web: www.broadcom.com
Corporate Headquarters: San Jose, CA