MoCA 2.0 API
(Draft Version)

Users Guide

Version 2.10.6.110

Broadcom Corporation
5300 California Avenue
Irvine, California, USA 92677
Phone: 949-926-5000
Fax: 949-926-5203
www.broadcom.com

Table of Contents

moca_set_c4_moca20_en
moca_get_mac_addr
moca_set_mac_addr
moca_get_node_status
moca_set_beacon_channel_set
moca_get_fw_version
moca_get_max_tx_power_tune
moca_set_max_tx_power_tune
moca_get_rx_power_tune
moca_set_rx_power_tune
moca_set_mocad_forwarding_rx_mac
moca_set_mocad_forwarding_rx_ack
moca_get_mocad_forwarding_tx_alloc
moca_set_mocad_forwarding_tx_send
moca_register_core_ready_cb
moca_register_power_up_status_cb
moca_register_new_lof_cb
moca_register_admission_completed_cb
moca_register_tpcap_done_cb
moca_register_mocad_forwarding_rx_packet_cb
moca_register_mocad_forwarding_tx_ack_cb
moca_register_pr_degradation_cb
moca_set_start
moca_set_stop
moca_get_drv_info
moca_get_miscval
moca_set_miscval
moca_get_en_capable
moca_set_en_capable
moca_set_restore_defaults
moca_get_mocad_version
moca_set_restart
moca_get_lof_update
moca_set_lof_update
moca_get_primary_ch_offset
moca_set_primary_ch_offset
moca_get_assertText
moca_set_assertText
moca_get_wdog_enable
moca_set_wdog_enable
moca_get_miscval2
moca_set_miscval2
moca_get_mr_seq_num
moca_set_mr_seq_num
moca_get_secondary_ch_offset
moca_set_secondary_ch_offset
moca_set_cof
moca_get_amp_type
moca_set_amp_type

moca_get_rlapm_en
moca_set_rlapm_en
moca_get_freq_shift
moca_set_freq_shift
moca_get_max_phy_rate
moca_set_max_phy_rate
moca_get_bandwidth
moca_set_bandwidth
moca_get_arpl_th_100
moca_set_arpl_th_100
moca_get_adc_mode
moca_set_adc_mode
moca_get_max_phy_rate_turbo
moca_set_max_phy_rate_turbo
moca_get_max_constellation
moca_set_max_constellation
moca_get_rlapm_table_50
moca_set_rlapm_table_50
moca_get_phy_status
moca_get_rlapm_table_100
moca_set_rlapm_table_100
moca_get_rx_gain_params
moca_get_tx_power_params
moca_get_nv_cal_enable
moca_set_nv_cal_enable
moca_get_rlapm_cap_50
moca_set_rlapm_cap_50
moca_get_snr_margin_rs
moca_set_snr_margin_rs
moca_get_snr_margin_ldpc
moca_set_snr_margin_ldpc
moca_get_snr_margin_ldpc_sec_ch
moca_set_snr_margin_ldpc_sec_ch
moca_get_snr_margin_ldpc_pre5
moca_set_snr_margin_ldpc_pre5
moca_get_snr_margin_ofdma
moca_set_snr_margin_ofdma
moca_get_rlapm_cap_100
moca_set_rlapm_cap_100
moca_get_sapm_table_50
moca_set_sapm_table_50
moca_get_sapm_table_100
moca_set_sapm_table_100
moca_set_nv_cal_clear
moca_get_sapm_table_sec
moca_set_sapm_table_sec
moca_get_amp_reg
moca_set_amp_reg
moca_get_snr_margin_ldpc_pri_ch

List of Figures

Introduction

This document describes the MoCA 2.0 API for applications using the BRCM MoCA core.

Please note that this document is still considered a draft version. *This means that the functions and structures listed in this document are likely to change and evolve. Broadcom will make an effort to preserve the currently described functions and structures however as more testing and development is completed, changes may be required.*

The overall system interaction between the various layers is shown in the figure below.

In general the user applications will use the MoCA API functions that are described in this document. An example of a user application is the BRCM mocap application. This application provides a command line interface to control and configure the MoCA core. Another example would be the GCAP commands. For each MoCA interface that exists, an instance of the MoCA daemon (mocad) must be created.

The MoCA 2.0 APIs communicate through the MoCA daemon, which uses IOCTL calls to the MoCA kernel driver. The MoCA kernel driver communicates directly with the MoCA core through hardware registers.

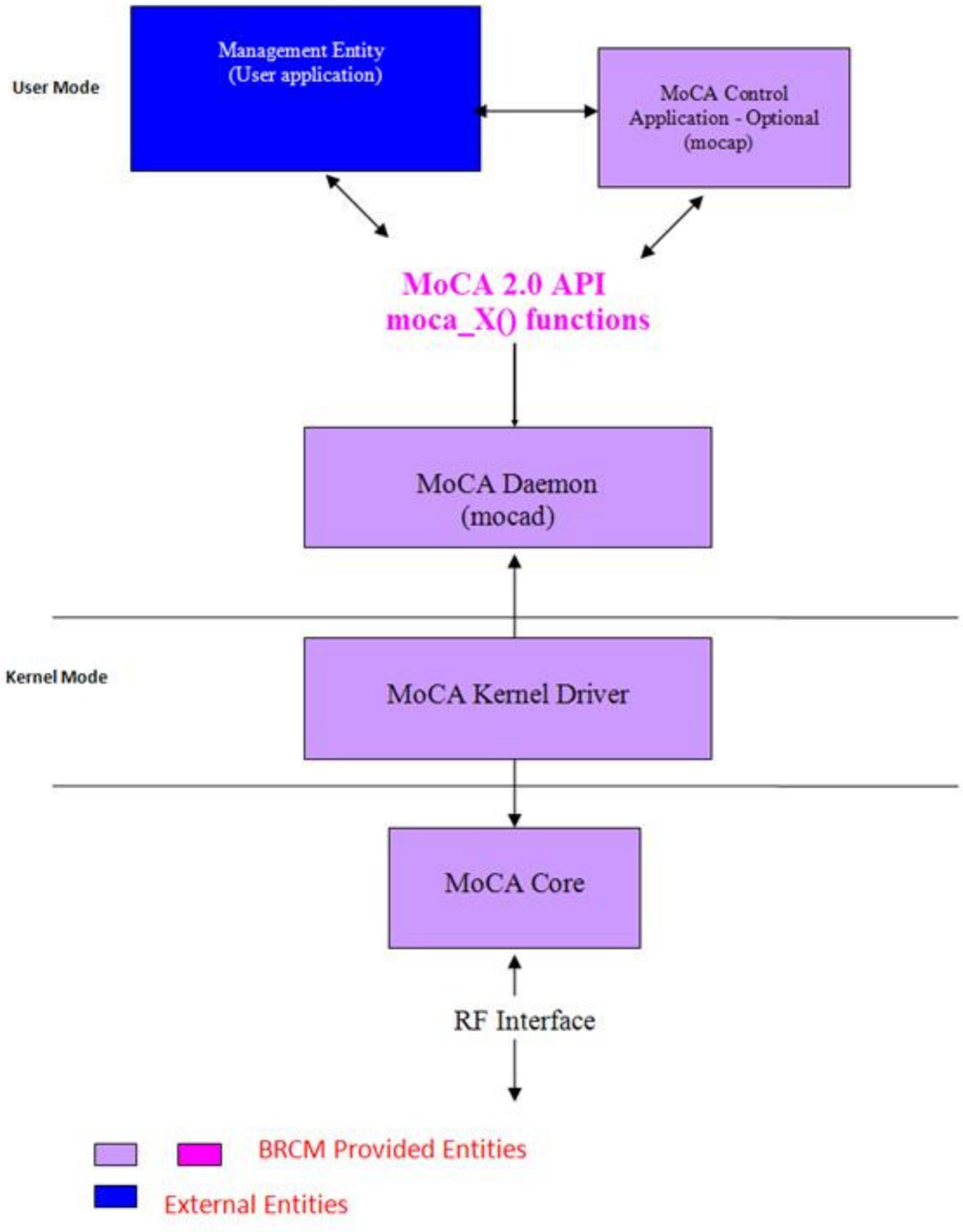

Figure 1 - MoCA System Layer Interaction

Typical Usage

The following sections describe some of the basic functions needed for MoCA operation. The files mocap.c and mocalib-cli-gen.c should be consulted for detailed examples on how to use the functions in this API.

Initialization

The application will first need to open a handle to the MoCA interface it wishes to control, this is done using moca_open(). Function moca_open() will allocate memory which is freed when moca_close() is called.

To start the MoCA interface, there are three steps to be taken.

1. Set initialization parameters
   The initialization parameters are set using several functions such as moca_set_flow_control_en(), moca_set_privacy_en() and moca_set_taboo_channels(). If the application is written to allow multiple MoCA core binary files to be used, the function moca_set_fw_file() is used to set the binary file. Functions that set initialization parameters have the following comment in their description: *"This function can be invoked at any time however the setting will only take effect when the MoCA interface is started."*

2. Set configuration parameters
   Configuration parameters may be set before starting the MoCA core. These parameters are set by several functions (e.g. moca_set_max_constellation(),moca_set_max_frame_size()).

3. Start the MoCA core
   This is done by calling function moca_set_start(). This will send a message to mocad instructing it to load the firmware and configure the core with the appropriate parameters.

To stop the MoCA core, the function moca_set_stop() is used.

Run-time Management

The application may want to change certain configuration parameters on-the-fly or may want to obtain status data and statistics from the MoCA interface.

The “moca_set” functions used to configure parameters also have “moca_get” counterparts for retrieving the current settings.

Status information can primarily be retrieved using functions from the Interface and Network groups. Specific node status information can be retrieved using functions from the Network group.

Statistics information can be retrieved using functions moca_get_gen_stats(), moca_get_ext_stats() and moca_get_ext_octet_count(). Statistics pertaining to a specific node on the MoCA network can be obtained using moca_get_node_stats() and moca_get_node_stats_ext_acc(). The statistics counters can be reset using function moca_set_reset_stats().

Functions and Structures

This section lists the functions and structures available via the MoCA 2.0 API. The functions and structures are divided into logical groups in order to make it easier to see which parmeters relate to one another.

There are "get" functions and "set" functions. The "get" functions are used to retrieve data from the MoCA module. The "set" functions are used to configure parameters or initiate operations within the MoCA module.

There are also callback register functions (moca_register_XYZ_cb). These functions allow an application to register a callback function for notification of asynchronous MoCA module events. The callback register functions accept a 'userarg' parameter which will be passed to the callback function upon the occurence of the specific event. The function moca_event_loop() must be running in order for callback functions to be called. The function moca_event_loop() can be running as a separate thread.

The "moca_do_XYZ" functions generally perform complex operations where communication with other nodes in the MoCA network is required. Because of this, the "do" functions may take longer to return in comparison with "set" or "get" functions.

Unless otherwise specified, functions return a value of 0 when successful.

General functions

moca_open

Prototype:

```
void *moca_open(char *ifname)
```

Description:

Open a connection to the MoCA module. This function returns the context handle to be passed as an argument to other API functions.

moca_close

Prototype:

```
void moca_close(void *vctx)
```

Description:

Close a connection to the MoCA module and free associated memory with the connection.

moca_event_loop

Prototype:

```
int moca_event_loop(void *vctx)
```

Description:

This function is needed to trigger callback functions for asynchronous MoCA module events. This function can be run as a separate thread.

moca_cancel_event_loop

Prototype:

```
void moca_cancel_event_loop(void *vctx)
```

Description:

Cancel an event loop that was started with moca_event_loop(). This will cause the moca_event_loop () function to return.

NODE Group

NODE Group

The Node group of parameters contains configurable fields that are specific to this node.

Structures

struct moca_core_ready

Fields:

uint8_t	chip_type	Chip type identification code
uint8_t	compatibility	Backwards compatibility bit. A running index. Current value is 1
uint8_t	phy_freq_mhz	PHY freq MHz.
uint8_t	reserved	reserved for future use
uint32_t	syncVersion	Synchronization version of the mocad_strings.h

struct moca_drv_info

Fields:

uint32_t	build_number
uint32_t	chip_id
uint32_t	core_uptime
char	devname[64]
uint32_t	hw_rev
char	ifname[16]
uint32_t	link_uptime

uint32_t	rf_band	
uint32_t	uptime	
uint32_t	version	

struct moca_fw_version
Fields:

uint32_t	version_major	Major version
uint32_t	version_minor	Minor version
uint32_t	version_moca	MoCA version
uint32_t	version_patch	Patch level

struct moca_mac_addr
Fields:

macaddr_t	val	

struct moca_max_tx_power_tune
Fields:

int8_t	offset[86]	*Default:* 0 *Minimum:* 0 *Maximum:* 56
uint16_t	padding	

struct moca_mocad_forwarding_rx_ack
Fields:

uint32_t	offset	Offset of packet from start of packet-ram
uint32_t	size	size of packet

struct moca_mocad_forwarding_rx_packet
Fields:

uint32_t	length	Length of packet
uint32_t	offset	Offset of packet relative to start of packet-ram

struct moca_mocad_forwarding_tx_alloc
Fields:

uint32_t	count	Number of buffers
uint32_t	offset	Offset of buffers from start of packet-ram
uint32_t	size	Size of each buffer

struct moca_mocad_forwarding_tx_send

Fields:

uint32_t	dest_if	Interface to send packet on. 0: GMII/Ethernet, 1: MoCA RF
uint32_t	offset	Offset of packet, from start of packet-ram
uint32_t	size	Size of packet to send

struct moca_mocad_version

Fields:

uint32_t	mocad_version_major	Major version
uint32_t	mocad_version_minor	Minor version
uint32_t	mocad_version_moca	MoCA version
uint32_t	mocad_version_patch	Patch level

struct moca_node_status

Fields:

uint32_t	moca_hw_version	Version of the MoCA Core HW (VLSI version). This version identifies the MoCA core hardware block and NOT the whole integrated chip. Therefore, two different Broadcom chips may have the same MOCA_HW_VERSION value, like 7420B0, 7340A0 and 7342A0
uint32_t	moca_sw_version_major	Software Major version of the MoCA Core.
uint32_t	moca_sw_version_minor	Software Minor version of the MoCA Core.
uint32_t	moca_sw_version_rev	Software revision number of the MoCA Core.
uint32_t	qam_256_support	Indicates whether or not the MoCA Core supports QAM256
uint32_t	self_moca_version	Self MoCA version
uint32_t	vendor_id	MoCA vendor ID

struct moca_rx_power_tune

Fields:

int8_t	offset[86]	*Default:* 0 *Minimum:* -120 *Maximum:* 120

uint16_t padding

Functions

moca_get_preferred_nc
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_preferred_nc(void *vctx, uint32_t *val)
```

Description:
In MoCA 1.1, Preferred NC nodes have a preference over the other nodes in the MoCA Network to become the NC node.
(GCAP.37)

Parameters:
val
Default:
1 (BAND_E)
1 (BAND_F)
0

moca_set_preferred_nc
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_preferred_nc(void *vctx, uint32_t val)
```

Description:
In MoCA 1.1, Preferred NC nodes have a preference over the other nodes in the MoCA Network to become the NC node.
(GCAP.37) This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

Parameters:
val
Default:
1 (BAND_E)
1 (BAND_F)
0

moca_get_single_channel_operation

Prototype:

```
MOCALIB_GEN_GET_FUNCTION int moca_get_single_channel_operation(void *vctx,
uint32_t *val)
```

Description:

This is the Single Channel Operation indication.
Enable the MoCA for automatic Network Search, using the LOF and RF_TYPE parameters, or use the OSP Single Channel Operation.

Parameters:

val
Default:
1 (BAND_GENERIC)
0

moca_set_single_channel_operation

Prototype:

```
MOCALIB_GEN_SET_FUNCTION int moca_set_single_channel_operation(void *vctx,
uint32_t val)
```

Description:

This is the Single Channel Operation indication.
Enable the MoCA for automatic Network Search, using the LOF and RF_TYPE parameters, or use the OSP Single Channel Operation. This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

Parameters:

val
Default:
1 (BAND_GENERIC)
0

moca_get_continuous_power_tx_mode

Prototype:

```
MOCALIB_GEN_GET_FUNCTION int moca_get_continuous_power_tx_mode(void *vctx,
uint32_t *val)
```

Description:

Ability to transmit in a constant power mode as defined by the spec. It is used only for lab testing. The transmit channel will be the LOF.

Parameters:
val
Default: 0

moca_set_continuous_power_tx_mode
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_continuous_power_tx_mode(void *vctx,
uint32_t val)
```

Description:
Ability to transmit in a constant power mode as defined by the spec. It is used only for lab testing. The transmit channel will be the LOF. This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

Parameters:
val
Default: 0

moca_get_continuous_rx_mode_attn
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_continuous_rx_mode_attn(void *vctx, int32_t
*val)
```
Parameters:
val
Default: 0
Minimum: -1
Maximum: 63

moca_set_continuous_rx_mode_attn
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_continuous_rx_mode_attn(void *vctx, int32_t
val)
```

Description:
This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

Parameters:
val
Default: 0
Minimum: -1
Maximum: 63

moca_get_lof
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_lof(void *vctx, uint32_t *val)
```

Description:
Last Operation Frequency. RF frequency to which the MoCA interface was tuned when last operational.
(GCAP.8)
This field is used also for setting required frequency of operation, when not in Network Search mode.

Parameters:
val
Default:
1000 (BAND_C4)
1400 (BAND_D_HIGH)
1150 (BAND_D_LOW)
575 (BAND_E)
1150 (BAND_EX_D)
800 (BAND_F)
1150 (BAND_GENERIC)
975 (BAND_H)
0

moca_set_lof
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_lof(void *vctx, uint32_t val)
```

Description:
Last Operation Frequency. RF frequency to which the MoCA interface was tuned when last operational.
(GCAP.8)

This field is used also for setting required frequency of operation, when not in Network Search mode.

Parameters:
val
Default:
1000 (BAND_C4)
1400 (BAND_D_HIGH)
1150 (BAND_D_LOW)
575 (BAND_E)
1150 (BAND_EX_D)
800 (BAND_F)
1150 (BAND_GENERIC)
975 (BAND_H)
0

moca_get_bonding
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_bonding(void *vctx, uint32_t *val)
```

Description:
Enables bonding on chips that support it.

Parameters:
val
Default:
0 (BONDING_SUPPORTED)
0
Minimum: 0
Maximum:
1 (BONDING_SUPPORTED)
0

moca_set_bonding
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_bonding(void *vctx, uint32_t val)
```

Description:
Enables bonding on chips that support it. This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

Parameters:
val
Default:
0 (BONDING_SUPPORTED)
0
Minimum: 0
Maximum:
1 (BONDING_SUPPORTED)
0

moca_get_listening_freq_mask
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_listening_freq_mask(void *vctx, uint32_t
*val)
```

Description:
Bit mask for specifying which frequencies should be scanned during the listening phase of network search. Depending on the RF band of operation, the MSB of this parameter corresponds to the lowest frequency channel of the band. Each subsequent bit of this parameter represents the next highest 25MHz channel. The base channels for each RF band are as follows:
Band D-Low : 46 (1150 MHz)
Band D-High: 56 (1400 MHz)
Band Ext-D : 46 (1150 MHz)
Band C4 : 40 (1000 MHz)
Band E : 20 (500 MHz)
Band F : 27 (675 MHz)
Band H : 39 (975 MHz)

Parameters:
val
Default: 0xFFFFFFFF

moca_set_listening_freq_mask
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_listening_freq_mask(void *vctx, uint32_t
val)
```

Description:
Bit mask for specifying which frequencies should be scanned during the listening phase of network search. Depending on the RF band of operation, the MSB of this parameter corresponds to the

lowest frequency channel of the band. Each subsequent bit of this parameter represents the next highest 25MHz channel. The base channels for each RF band are as follows:
Band D-Low : 46 (1150 MHz)
Band D-High: 56 (1400 MHz)
Band Ext-D : 46 (1150 MHz)
Band C4 : 40 (1000 MHz)
Band E : 20 (500 MHz)
Band F : 27 (675 MHz)
Band H : 39 (975 MHz) This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

Parameters:
val
Default: 0xFFFFFFFF

moca_get_listening_duration
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_listening_duration(void *vctx, uint32_t
*val)
```

Description:
The duration in milliseconds that should be spent listening for beacons on each channel during the network search listening phase.

Parameters:
val
Default: 1050
Minimum: 100

moca_set_listening_duration
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_listening_duration(void *vctx, uint32_t val)
```

Description:
The duration in milliseconds that should be spent listening for beacons on each channel during the network search listening phase. This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

Parameters:
val
Default: 1050

Minimum: 100

moca_get_limit_traffic
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_limit_traffic(void *vctx, uint32_t *val)
```

Description:
Limit traffic for extra power save mode.

Parameters:
val
Default: 0
Minimum: 0
Maximum: 2

moca_set_limit_traffic
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_limit_traffic(void *vctx, uint32_t val)
```

Description:
Limit traffic for extra power save mode. This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

Parameters:
val
Default: 0
Minimum: 0
Maximum: 2

moca_get_remote_man
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_remote_man(void *vctx, uint32_t *val)
```

Description:
Remote management mode

Parameters:

val
Default:
1 (STANDALONE,6802B0)
2 (STANDALONE,6802C0)
0
Minimum: 0
Maximum: 2

moca_set_remote_man
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_remote_man(void *vctx, uint32_t val)
```

Description:
Remote management mode This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

Parameters:
val
Default:
1 (STANDALONE,6802B0)
2 (STANDALONE,6802C0)
0
Minimum: 0
Maximum: 2

moca_get_c4_moca20_en
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_c4_moca20_en(void *vctx, uint32_t *val)
```

Description:
Enables MoCA 2.0 also on C4 band.

Parameters:
val
Default: 0
Minimum: 0
Maximum: 1

moca_set_c4_moca20_en
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_c4_moca20_en(void *vctx, uint32_t val)
```

Description:
Enables MoCA 2.0 also on C4 band. This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

Parameters:
val
Default: 0
Minimum: 0
Maximum: 1

moca_get_mac_addr
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_mac_addr(void *vctx, struct moca_mac_addr
*out)
```

Description:
Unique IDentifier (IEEE 48-bit Extended Unique Identifier) of a MoCA Node on the MoCA network. This MAC address is the MAC address of the ONT MoCA interface port.

moca_set_mac_addr
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_mac_addr(void *vctx, struct moca_mac_addr
*in)
```

Description:
Unique IDentifier (IEEE 48-bit Extended Unique Identifier) of a MoCA Node on the MoCA network. This MAC address is the MAC address of the ONT MoCA interface port. This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

moca_get_node_status
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_node_status(void *vctx, struct
moca_node_status *out)
```

Description:
Retrieve general status information about this MoCA node.

moca_set_beacon_channel_set
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_beacon_channel_set(void *vctx, uint32_t
channel)
```

Description:
This is part of a user command to change channel (!) This IE is the first step in Channel Selection process. This IE will flag the MoCA Core to prepare for Channel Selection. The process of CS will be initiated by a user CLI/API, and the host function will do: 1) Send down this IE MMP message. 2) start a MR transaction using the MR_REQUEST command 3) After the success of [2] (receive of MR_RESPONSE OK trap) this Assigned Channel number should be stored in NV init_param BEACON_CHANNEL field for future reboots.

Parameters:
channel

moca_get_fw_version
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_fw_version(void *vctx, struct
moca_fw_version *out)
```

Description:
The MoCA firmware release version

moca_get_max_tx_power_tune
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_max_tx_power_tune(void *vctx, struct
moca_max_tx_power_tune *out)
```

Description:
tx power per frequency

moca_set_max_tx_power_tune
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_max_tx_power_tune(void *vctx, struct
moca_max_tx_power_tune *in)
```

Description:
tx power per frequency This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

moca_get_rx_power_tune
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_rx_power_tune(void *vctx, struct
moca_rx_power_tune *out)
```

Description:
rx power tuning per frequency

moca_set_rx_power_tune
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_rx_power_tune(void *vctx, struct
moca_rx_power_tune *in)
```

Description:
rx power tuning per frequency

moca_set_mocad_forwarding_rx_mac
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_mocad_forwarding_rx_mac(void *vctx,
macaddr_t * mac_addr)
```

Description:
Forward packets to mocad\n

Parameters:
mac_addr
MAC address to filter on

moca_set_mocad_forwarding_rx_ack
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_mocad_forwarding_rx_ack(void *vctx, const
struct moca_mocad_forwarding_rx_ack *in)
```

Description:
Allow firmware to free a packet (see IE_MOCAD_FIRWARDING_PACKET)

moca_get_mocad_forwarding_tx_alloc
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_mocad_forwarding_tx_alloc(void *vctx, struct moca_mocad_forwarding_tx_alloc *out)
```

Description:
Request firmware allocate buffers used for TX

moca_set_mocad_forwarding_tx_send
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_mocad_forwarding_tx_send(void *vctx, const struct moca_mocad_forwarding_tx_send *in)
```

Description:
Request firmware send a packet

moca_register_core_ready_cb
Prototype:
```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_core_ready_cb(void *vctx, void (*callback)(void *userarg, struct moca_core_ready *out), void *userarg)
```

moca_register_power_up_status_cb
Prototype:
```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_power_up_status_cb(void *vctx, void (*callback)(void *userarg, uint32_t status), void *userarg)
```

moca_register_new_lof_cb
Prototype:
```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_new_lof_cb(void *vctx, void (*callback)(void *userarg, uint32_t lof), void *userarg)
```

moca_register_admission_completed_cb

Prototype:

```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_admission_completed_cb(void
*vctx, void (*callback)(void *userarg, uint32_t lof), void *userarg)
```

moca_register_tpcap_done_cb

Prototype:

```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_tpcap_done_cb(void *vctx, void
(*callback)(void *userarg), void *userarg)
```

moca_register_mocad_forwarding_rx_packet_cb

Prototype:

```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_mocad_forwarding_rx_packet_cb
(void *vctx, void (*callback)(void *userarg, struct
moca_mocad_forwarding_rx_packet *out), void *userarg)
```

Description:

Trap indicating a packet matching IE_MOCAD_FORWADING_MAC has arrived

moca_register_mocad_forwarding_tx_ack_cb

Prototype:

```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_mocad_forwarding_tx_ack_cb(void
*vctx, void (*callback)(void *userarg, uint32_t offset), void *userarg)
```

Description:

Trap indicating a packet has been sent and the associated buffer is available

moca_register_pr_degradation_cb

Prototype:

```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_pr_degradation_cb(void *vctx,
void (*callback)(void *userarg), void *userarg)
```

Description:

Trap indicating PHY rate degradation by more than 15%

moca_set_start

Prototype:

```
MOCALIB_GEN_SET_FUNCTION int moca_set_start(void *vctx)
```

Description:
Instruct the MoCA daemon to load and start the MoCA core.

moca_set_stop
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_stop(void *vctx)
```

Description:
Instruct the MoCA daemon to stop the MoCA core.

moca_get_drv_info
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_drv_info(void *vctx, struct moca_drv_info
*out)
```

moca_get_miscval
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_miscval(void *vctx, uint32_t *val)
```
Parameters:
val

moca_set_miscval
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_miscval(void *vctx, uint32_t val)
```
Parameters:
val

moca_get_en_capable
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_en_capable(void *vctx, uint32_t *enable)
```
Parameters:
enable
Default: 1

moca_set_en_capable
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_en_capable(void *vctx, uint32_t enable)
```
Parameters:
enable
Default: 1

moca_set_restore_defaults
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_restore_defaults(void *vctx)
```

moca_get_mocad_version
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_mocad_version(void *vctx, struct
moca_mocad_version *out)
```

Description:
The mocad release version

moca_set_restart
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_restart(void *vctx)
```

Description:
Instruct the MoCA daemon to restart the MoCA core.

moca_get_lof_update
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_lof_update(void *vctx, uint32_t *val)
```

Description:
This parameter controls whether the LOF is updated when joining a network. If set to 'enabled' the LOF will be updated to the channel of the network that this node is currently linked on. If set to 'disabled' the LOF will not be updated.

Parameters:
val
Default: 1
Minimum: 0
Maximum: 1

moca_set_lof_update
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_lof_update(void *vctx, uint32_t val)
```

Description:
This parameter controls whether the LOF is updated when joining a network. If set to 'enabled' the LOF will be updated to the channel of the network that this node is currently linked on. If set to 'disabled' the LOF will not be updated.

Parameters:
val
Default: 1
Minimum: 0
Maximum: 1

moca_get_primary_ch_offset
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_primary_ch_offset(void *vctx, int32_t *val)
```

Description:
For a MoCA 2.0 network, this parameter specifies the frequency offset of the primary channel relative to the beacon channel. This parameter is relevant when the node is NC.

Parameters:
val
Default: 1
Valid Values: -25, 0, 1, 25

moca_set_primary_ch_offset

Prototype:

```
MOCALIB_GEN_SET_FUNCTION int moca_set_primary_ch_offset(void *vctx, int32_t val)
```

Description:

For a MoCA 2.0 network, this parameter specifies the frequency offset of the primary channel relative to the beacon channel. This parameter is relevant when the node is NC. This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

Parameters:

val

Default: 1

Valid Values: -25, 0, 1, 25

moca_get_assertText

Prototype:

```
MOCALIB_GEN_GET_FUNCTION int moca_get_assertText(void *vctx, uint32_t *assertText)
```

Parameters:

assertText

Default: 0

moca_set_assertText

Prototype:

```
MOCALIB_GEN_SET_FUNCTION int moca_set_assertText(void *vctx, uint32_t assertText)
```

Parameters:

assertText

Default: 0

moca_get_wdog_enable

Prototype:

```
MOCALIB_GEN_GET_FUNCTION int moca_get_wdog_enable(void *vctx, uint32_t *enable)
```

Parameters:

enable

Default: 1

moca_set_wdog_enable
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_wdog_enable(void *vctx, uint32_t enable)
```
Parameters:
enable
Default: 1

moca_get_miscval2
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_miscval2(void *vctx, uint32_t *val)
```
Parameters:
val

moca_set_miscval2
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_miscval2(void *vctx, uint32_t val)
```
Parameters:
val

moca_get_mr_seq_num
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_mr_seq_num(void *vctx, uint32_t *val)
```

Description:
The sequence number used by the MR transaction.

Parameters:
val
Default: 0
Minimum: 0
Maximum: 0xFFFF

moca_set_mr_seq_num
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_mr_seq_num(void *vctx, uint32_t val)
```

Description:
The sequence number used by the MR transaction.

Parameters:
val
Default: 0
Minimum: 0
Maximum: 0xFFFF

moca_get_secondary_ch_offset
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_secondary_ch_offset(void *vctx, int32_t
*val)
```

Description:
For a MoCA 2.0 network, this parameter specifies the frequency offset of the secondary channel relative to the beacon channel in bonded mode. This parameter is relevant when the node is NC.

Parameters:
val
Default:
125 (BAND_GENERIC)
1
Valid Values: -125, 0, 1, 125

moca_set_secondary_ch_offset
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_secondary_ch_offset(void *vctx, int32_t val)
```

Description:
For a MoCA 2.0 network, this parameter specifies the frequency offset of the secondary channel relative to the beacon channel in bonded mode. This parameter is relevant when the node is NC. This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

Parameters:
val

Default:
125 (BAND_GENERIC)
1
Valid Values: -125, 0, 1, 125

moca_set_cof
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_cof(void *vctx, uint32_t val)
```

Description:
Current operating frequency. This parameter sets the LOF for MoCA on the next MoCA start/restart without saving the frequency in NVRAM. This parameter has no 'get' function. The interface_status rf_channel field should be read to obtain the actual operating frequency. Once a link is established, this parameter will have no effect unless it is set again followed by a MoCA start/restart.

Parameters:
val
Default: 0

moca_get_amp_type
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_amp_type(void *vctx, uint32_t *val)
```

Description:
Specifies the revision of PA/LNA. This setting is used if mocad cannot auto-detect (e.g. out-of-date bmoca kernel module)

Parameters:
val
Default: 1

moca_set_amp_type
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_amp_type(void *vctx, uint32_t val)
```

Description:

Specifies the revision of PA/LNA. This setting is used if mocad cannot auto-detect (e.g. out-of-date bmoca kernel module) This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

Parameters:
val
Default: 1

PHY Group

The PHY group of parameters control the physical layer of the MoCA interface.

Structures

struct moca_amp_reg
Fields:

uint32_t	addr	The address of the register to get or set. This function will only succeed when used with kernels that support register access to the PA/LNA chip.
uint32_t	value	For set operations, this the value to set the register to.\n For get operations, this is the value of specified register. *Default:* 0

struct moca_max_constellation
Fields:

uint32_t	gcd_limit_100	This is the GCD (broadcast) constellation limit for 100 MHz profiles. *Default:* 10 *Minimum:* 1 *Maximum:* 10
uint32_t	gcd_limit_50	This is the GCD (broadcast) constellation limit for 50 MHz profiles. *Default:* 10 *Minimum:* 1 *Maximum:* 10
uint32_t	node_id	*Minimum:* 0 *Maximum:* 15
uint32_t	p2p_limit_100	This is the point-to-point (unicast) constellation limit for 100 MHz profiles. *Default:* 10 *Minimum:* 1 *Maximum:* 10

uint32_t	p2p_limit_50	This is the point-to-point (unicast) constellation limit for 50 MHz profiles. *Default:* 10 *Minimum:* 1 *Maximum:* 10

struct moca_rlapm_table_100

Fields:

uint8_t	reserved_0	
uint8_t	reserved_1	
uint8_t	rlapmtable[66]	*Default:* 0 *Minimum:* 0 *Maximum:* 60

struct moca_rlapm_table_50

Fields:

uint8_t	reserved_0	
uint8_t	reserved_1	
uint8_t	rlapmtable[66]	*Default:* 0 *Minimum:* 0 *Maximum:* 60

struct moca_rx_gain_params

Fields:

uint32_t	is3451	
uint32_t	lna_ctrl_reg	

struct moca_sapm_table_100

Fields:

uint8_t	val[512]	Array of values indexed by sub-carrier number. *Default:* 0 *Minimum:* 0 *Maximum:* 120

struct moca_sapm_table_50

Fields:

uint8_t	val[256]	Array of values indexed by sub-carrier number.

Default: 0
Minimum: 0
Maximum: 120

struct moca_sapm_table_sec
Fields:

uint8_t	val[512]	Array of values indexed by sub-carrier number. *Default:* 0 *Minimum:* 0 *Maximum:* 120

struct moca_snr_margin_ldpc
Fields:

int32_t	base_margin	A base SNR Margin value adjustment in dB which applies to all constellations from BPSK (index 0) to 1024QAM (index 9). SNR base margin value ranges from -3 to +25dB in steps of 1/256 dB. *Default:* 0x0 *Minimum:* -768 *Maximum:* 6400
int16_t	offsets[10]	A table of 10 values representing the SNR Margin value adjustments in dB per constellation from BPSK (index 0) to 1024QAM (index 9). Offset values range from -3 to +25dB in steps of 1/256 dB. *Default:* 0 *Minimum:* -768 *Maximum:* 6400

struct moca_snr_margin_ldpc_pre5
Fields:

int32_t	base_margin	A base SNR Margin value adjustment in dB which applies to all constellations from BPSK (index 0) to 1024QAM (index 9). SNR base margin value ranges from -3 to +25dB in steps of 1/256 dB. *Default:* 0 *Minimum:* -768 *Maximum:* 6400
int16_t	offsets[10]	A table of 10 values representing the SNR Margin value adjustments in dB per constellation from BPSK (index 0) to 1024QAM (index 9). Offset values range from -3 to +25dB in steps of 1/256 dB. *Default:* 0 *Minimum:* -768 *Maximum:* 6400

struct moca_snr_margin_ldpc_pri_ch

Fields:

int32_t	base_margin	A base SNR Margin value adjustment in dB which applies to all constellations from BPSK (index 0) to 1024QAM (index 9). SNR base margin value ranges from -3 to +25dB in steps of 1/256 dB. *Default:* 0x100 *Minimum:* -768 *Maximum:* 6400
int16_t	offsets[10]	A table of 10 values representing the SNR Margin value adjustments in dB per constellation from BPSK (index 0) to 1024QAM (index 9). Offset values range from -3 to +25dB in steps of 1/256 dB. *Default:* 0 *Minimum:* -768 *Maximum:* 6400

struct moca_snr_margin_ldpc_sec_ch

Fields:

int32_t	base_margin	A base SNR Margin value adjustment in dB which applies to all constellations from BPSK (index 0) to 1024QAM (index 9). SNR base margin value ranges from -3 to +25dB in steps of 1/256 dB. *Default:* 0x100 *Minimum:* -768 *Maximum:* 6400
int16_t	offsets[10]	A table of 10 values representing the SNR Margin value adjustments in dB per constellation from BPSK (index 0) to 1024QAM (index 9). Offset values range from -3 to +25dB in steps of 1/256 dB. *Default:* 0 *Minimum:* -768 *Maximum:* 6400

struct moca_snr_margin_ofdma

Fields:

int32_t	base_margin	A base SNR Margin value adjustment in dB which applies to all constellations from BPSK (index 0) to 1024QAM (index 9). SNR base margin value ranges from -3 to +25dB in steps of 1/256 dB. *Default:* 0 *Minimum:* -768 *Maximum:* 6400
int16_t	offsets[10]	

A table of 10 values representing the SNR Margin value adjustments in dB per constellation from BPSK (index 0) to 1024QAM (index 9). Offset values range from -3 to +25dB in steps of 1/256 dB.
Default: 0
Minimum: -768
Maximum: 6400

struct moca_snr_margin_rs
Fields:

int32_t	base_margin	A base SNR Margin value adjustment in dB which applies to all constellations from BPSK (index 0) to 1024QAM (index 9). SNR base margin value ranges from -3 to +25dB in steps of 1/256 dB. *Default:* 0 *Minimum:* -768 *Maximum:* 6400
int16_t	offsets[10]	A table of 10 values representing the SNR Margin value adjustments in dB per constellation from BPSK (index 0) to 1024QAM (index 9). Offset values range from -3 to +25dB in steps of 1/256 dB. *Default:* 0 *Minimum:* -768 *Maximum:* 6400

struct moca_tx_power_params
Fields:

uint32_t	channel
uint32_t	channelMode
uint32_t	channel_reduce_tune
uint32_t	is3451
uint32_t	pa_ctrl_reg
uint32_t	pad_ctrl_deg
uint32_t	table_max_index
uint32_t	tx_digital_gain
uint16_t	tx_table[62]
uint32_t	user_reduce_power

struct moca_tx_power_params_in
Fields:

uint32_t	channelMode
uint32_t	txTableIndex

Functions

moca_get_tpc_en
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_tpc_en(void *vctx, uint32_t *val)
```

Description:
Enable Transmit Power Control (TPC).
When enabled, the transmit power level is adjusted to a setting that will achieve the maximum target PHY bit rate. The adjusted power setting will be less than or equal to 'Tx Power'.
When disabled, the transmit power level is set to .

Parameters:
val
Default:
0 (BAND_E)
0 (BAND_F)
0 (BAND_H)
1

moca_set_tpc_en
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_tpc_en(void *vctx, uint32_t val)
```

Description:
Enable Transmit Power Control (TPC).
When enabled, the transmit power level is adjusted to a setting that will achieve the maximum target PHY bit rate. The adjusted power setting will be less than or equal to 'Tx Power'.
When disabled, the transmit power level is set to . This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

Parameters:
val
Default:
0 (BAND_E)
0 (BAND_F)
0 (BAND_H)
1

moca_get_max_tx_power
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_max_tx_power(void *vctx, int32_t *val)
```

Description:
Indicates the max transmitter power level allowed.

Parameters:
val
Default: 3
Minimum: -31
Maximum: 3

moca_set_max_tx_power
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_max_tx_power(void *vctx, int32_t val)
```

Description:
Indicates the max transmitter power level allowed. This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

Parameters:
val
Default: 3
Minimum: -31
Maximum: 3

moca_get_beacon_pwr_reduction
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_beacon_pwr_reduction(void *vctx, uint32_t
*val)
```

Description:
Amount of power reduction for beacons vs other transmissions.
Beacon power reduction must be disabled for Bands E and F.

Parameters:
val

Default: 0
Minimum: 0
Maximum:
0 (BAND_E)
0 (BAND_F)
30

moca_set_beacon_pwr_reduction

Prototype:

```
MOCALIB_GEN_SET_FUNCTION int moca_set_beacon_pwr_reduction(void *vctx, uint32_t
val)
```

Description:

Amount of power reduction for beacons vs other transmissions.
Beacon power reduction must be disabled for Bands E and F. This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

Parameters:

val
Default: 0
Minimum: 0
Maximum:
0 (BAND_E)
0 (BAND_F)
30

moca_get_beacon_pwr_reduction_en

Prototype:

```
MOCALIB_GEN_GET_FUNCTION int moca_get_beacon_pwr_reduction_en(void *vctx, uint32_t
*val)
```

Description:

Enable/Disable BEACON_PWR_REDUCTION.
Beacon power reduction must be disabled for Bands E and F.

Parameters:

val
Default:
0 (6816)
1 (7xxx)

0 (BAND_E)
0 (BAND_F)
Minimum: 0
Maximum:
0 (BAND_E)
0 (BAND_F)
1

moca_set_beacon_pwr_reduction_en

Prototype:

```
MOCALIB_GEN_SET_FUNCTION int moca_set_beacon_pwr_reduction_en(void *vctx, uint32_t val)
```

Description:

Enable/Disable BEACON_PWR_REDUCTION.
Beacon power reduction must be disabled for Bands E and F. This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

Parameters:

val
Default:
0 (6816)
1 (7xxx)
0 (BAND_E)
0 (BAND_F)
Minimum: 0
Maximum:
0 (BAND_E)
0 (BAND_F)
1

moca_get_bo_mode

Prototype:

```
MOCALIB_GEN_GET_FUNCTION int moca_get_bo_mode(void *vctx, uint32_t *val)
```

Description:

This flag enables two modes of operation, introducing tradeoff between fast Back Off convergence and better noise robustness of the system.

Parameters:

val

Default: 0

moca_set_bo_mode
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_bo_mode(void *vctx, uint32_t val)
```

Description:
This flag enables two modes of operation, introducing tradeoff between fast Back Off convergence and better noise robustness of the system. This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

Parameters:
val
Default: 0

moca_get_qam256_capability
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_qam256_capability(void *vctx, uint32_t *val)
```

Description:
This fields specifies the QAM256 ability in Admission Res/Req negotiations (NODE_PROTOCOL_SUPPORT field).

Parameters:
val
Default: 1

moca_set_qam256_capability
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_qam256_capability(void *vctx, uint32_t val)
```

Description:
This fields specifies the QAM256 ability in Admission Res/Req negotiations (NODE_PROTOCOL_SUPPORT field). This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

Parameters:

val
Default: 1

moca_get_otf_en
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_otf_en(void *vctx, uint32_t *val)
```

Description:
Enables/Disables On The Fly calibration.
This feature calibrates the Tx Power periodically, and is used for overcoming max power change in temperatures.

Parameters:
val
Default: 0

moca_set_otf_en
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_otf_en(void *vctx, uint32_t val)
```

Description:
Enables/Disables On The Fly calibration.
This feature calibrates the Tx Power periodically, and is used for overcoming max power change in temperatures. This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

Parameters:
val
Default: 0

moca_get_star_topology_en
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_star_topology_en(void *vctx, uint32_t *val)
```

Description:
Enable support for star topology, which allows new nodes to admit to a network as long as the link to the NC is usable. The channel between ENs does not need to be usable in this mode.

Parameters:
val
Default: 0
Minimum: 0
Maximum: 1

moca_set_star_topology_en
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_star_topology_en(void *vctx, uint32_t val)
```

Description:
Enable support for star topology, which allows new nodes to admit to a network as long as the link to the NC is usable. The channel between ENs does not need to be usable in this mode. This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

Parameters:
val
Default: 0
Minimum: 0
Maximum: 1

moca_get_ofdma_en
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_ofdma_en(void *vctx, uint32_t *val)
```

Description:
Enable support for OFDMA PHY Frames

Parameters:
val
Default: 1
Minimum: 0
Maximum: 1

moca_set_ofdma_en
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_ofdma_en(void *vctx, uint32_t val)
```

Description:
Enable support for OFDMA PHY Frames This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

Parameters:
val
Default: 1
Minimum: 0
Maximum: 1

moca_get_min_bw_alarm_threshold
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_min_bw_alarm_threshold(void *vctx, uint32_t
*mbps)
```

Description:
Indicates a user configured threshold for PHY link bandwidth between two nodes that will raise an alarm. This configurable threshold shouldn't be confused with a different alarm below a fixed threshold of 358 bits per symbol (~57Mbps), which is the minimum PHY rate to allow a connection between any two nodes, according to MoCA spec.

Parameters:
mbps
Default: 100
Minimum: 50
Maximum: 3200

moca_set_min_bw_alarm_threshold
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_min_bw_alarm_threshold(void *vctx, uint32_t
mbps)
```

Description:
Indicates a user configured threshold for PHY link bandwidth between two nodes that will raise an alarm. This configurable threshold shouldn't be confused with a different alarm below a fixed threshold of 358 bits per symbol (~57Mbps), which is the minimum PHY rate to allow a connection between any two nodes, according to MoCA spec.

Parameters:
mbps
Default: 100
Minimum: 50
Maximum: 3200

moca_get_en_max_rate_in_max_bo
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_en_max_rate_in_max_bo(void *vctx, uint32_t
*val)
```

Description:
1 - enable. If max backoff is reached (30dB) then the device will try to increase the PHY rate beyond the target PHY rate, as long as the backoff stays maximal. As always, increasing PHY rate will reduce SNR margin, but the margin will still be under the allowed configuration. 0 - disable. If max backoff is reached then target PHY rate will be reached (if possible) and not more.

Parameters:
val
Default: 0

moca_set_en_max_rate_in_max_bo
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_en_max_rate_in_max_bo(void *vctx, uint32_t
val)
```

Description:
1 - enable. If max backoff is reached (30dB) then the device will try to increase the PHY rate beyond the target PHY rate, as long as the backoff stays maximal. As always, increasing PHY rate will reduce SNR margin, but the margin will still be under the allowed configuration. 0 - disable. If max backoff is reached then target PHY rate will be reached (if possible) and not more.

Parameters:
val
Default: 0

moca_get_target_phy_rate_qam128

Prototype:

```
MOCALIB_GEN_GET_FUNCTION int moca_get_target_phy_rate_qam128(void *vctx, uint32_t
*mbps)
```

Description:

Target PHY rate in Mbps, according to MoCA spec. Target PHY rate may be changed only before Admission time. Otherwise, the expected results are not guaranteed.

Parameters:

mbps

Default: 245

Maximum: 500

moca_set_target_phy_rate_qam128

Prototype:

```
MOCALIB_GEN_SET_FUNCTION int moca_set_target_phy_rate_qam128(void *vctx, uint32_t
mbps)
```

Description:

Target PHY rate in Mbps, according to MoCA spec. Target PHY rate may be changed only before Admission time. Otherwise, the expected results are not guaranteed.

Parameters:

mbps

Default: 245

Maximum: 500

moca_get_target_phy_rate_qam256

Prototype:

```
MOCALIB_GEN_GET_FUNCTION int moca_get_target_phy_rate_qam256(void *vctx, uint32_t
*mbps)
```

Description:

Target PHY rate in Mbps, according to MoCA spec. Target PHY rate may be changed only before Admission time. Otherwise, the expected results are not guaranteed.

Parameters:

mbps

Default: 275

Maximum: 500

moca_set_target_phy_rate_qam256

Prototype:

```
MOCALIB_GEN_SET_FUNCTION int moca_set_target_phy_rate_qam256(void *vctx, uint32_t
mbps)
```

Description:

Target PHY rate in Mbps, according to MoCA spec. Target PHY rate may be changed only before Admission time. Otherwise, the expected results are not guaranteed.

Parameters:

mbps

Default: 275

Maximum: 500

moca_get_sapm_en

Prototype:

```
MOCALIB_GEN_GET_FUNCTION int moca_get_sapm_en(void *vctx, uint32_t *bool_val)
```

Description:

Enabling the usage SNR Margin adjustments per sub carrier

Parameters:

bool_val

Default: 0

moca_set_sapm_en

Prototype:

```
MOCALIB_GEN_SET_FUNCTION int moca_set_sapm_en(void *vctx, uint32_t bool_val)
```

Description:

Enabling the usage SNR Margin adjustments per sub carrier

Parameters:

bool_val

Default: 0

moca_get_arpl_th_50
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_arpl_th_50(void *vctx, int32_t *arpl)
```

Description:
Aggregate Received Power Level (ARPL) Threshold for 50 MHz (MoCA 1.1) transmissions which MUST be specified from 0 to 65 dBm in steps of 1 dB to be used with the SAPM feature.
See also: sapm_table_50

Parameters:
arpl
Default: -50
Minimum: -65
Maximum: 0

moca_set_arpl_th_50
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_arpl_th_50(void *vctx, int32_t arpl)
```

Description:
Aggregate Received Power Level (ARPL) Threshold for 50 MHz (MoCA 1.1) transmissions which MUST be specified from 0 to 65 dBm in steps of 1 dB to be used with the SAPM feature.
See also: sapm_table_50

Parameters:
arpl
Default: -50
Minimum: -65
Maximum: 0

moca_get_rlapm_en
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_rlapm_en(void *vctx, uint32_t *bool_val)
```

Description:
Enabling the usage of SNR Margin adjustments according to Rx Power

Parameters:
bool_val
Default:
1 (BAND_E)

1 (BAND_F)
0

moca_set_rlapm_en
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_rlapm_en(void *vctx, uint32_t bool_val)
```

Description:
Enabling the usage of SNR Margin adjustments according to Rx Power

Parameters:
bool_val
Default:
1 (BAND_E)
1 (BAND_F)
0

moca_get_freq_shift
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_freq_shift(void *vctx, uint32_t *direction)
```

Description:
XtalPull test for midRF CTP, used by ICAP.110. This configuration is relevant only after TX continuous mode is activated, and the host should prevent sending it in regular mode.

Parameters:
direction
Default: 0

moca_set_freq_shift
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_freq_shift(void *vctx, uint32_t direction)
```

Description:
XtalPull test for midRF CTP, used by ICAP.110. This configuration is relevant only after TX continuous mode is activated, and the host should prevent sending it in regular mode.

Parameters:
direction
Default: 0

moca_get_max_phy_rate
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_max_phy_rate(void *vctx, uint32_t *mbps)
```

Description:
The maximum PHY rate supported in non-turbo mode.

Parameters:
mbps
Default:
630 (7425)
670

moca_set_max_phy_rate
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_max_phy_rate(void *vctx, uint32_t mbps)
```

Description:
The maximum PHY rate supported in non-turbo mode.

Parameters:
mbps
Default:
630 (7425)
670

moca_get_bandwidth
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_bandwidth(void *vctx, uint32_t *bandwidth)
```

Description:
Configure the MoCA interface to operate using a 50MHz or 100MHz bandwidth.

Parameters:
bandwidth
Default: 0
Minimum: 0
Maximum: 1

moca_set_bandwidth
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_bandwidth(void *vctx, uint32_t bandwidth)
```

Description:
Configure the MoCA interface to operate using a 50MHz or 100MHz bandwidth. This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

Parameters:
bandwidth
Default: 0
Minimum: 0
Maximum: 1

moca_get_arpl_th_100
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_arpl_th_100(void *vctx, int32_t *arpl)
```

Description:
Aggregate Received Power Level (ARPL) Threshold for 100 MHz PHY transmissions which MUST be specified from 0 to 65 dBm in steps of 1 dB to be used with the SAPM feature.
See also: sapm_table_100

Parameters:
arpl
Default: -50
Minimum: -65
Maximum: 0

moca_set_arpl_th_100
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_arpl_th_100(void *vctx, int32_t arpl)
```

Description:
Aggregate Received Power Level (ARPL) Threshold for 100 MHz PHY transmissions which MUST be specified from 0 to 65 dBm in steps of 1 dB to be used with the SAPM feature.
See also: sapm_table_100

Parameters:
arpl
Default: -50
Minimum: -65
Maximum: 0

moca_get_adc_mode
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_adc_mode(void *vctx, uint32_t *val)
```

Description:
ADC clock mode

Parameters:
val
Default: 0

moca_set_adc_mode
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_adc_mode(void *vctx, uint32_t val)
```

Description:
ADC clock mode This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

Parameters:
val
Default: 0

moca_get_max_phy_rate_turbo

Prototype:

```
MOCALIB_GEN_GET_FUNCTION int moca_get_max_phy_rate_turbo(void *vctx, uint32_t
*mbps)
```

Description:

The maximum PHY rate supported in turbo mode.

Parameters:

mbps

Default:

680 (7425)

670

moca_set_max_phy_rate_turbo

Prototype:

```
MOCALIB_GEN_SET_FUNCTION int moca_set_max_phy_rate_turbo(void *vctx, uint32_t
mbps)
```

Description:

The maximum PHY rate supported in turbo mode.

Parameters:

mbps

Default:

680 (7425)

670

moca_get_max_constellation

Prototype:

```
MOCALIB_GEN_GET_FUNCTION int moca_get_max_constellation(void *vctx, uint32_t
node_id, struct moca_max_constellation *out)
```

Description:

Set/Get max constellation on all carriers in the receive from a specified node (GCAP.32).

Parameters:

node_id

Minimum: 0

Maximum: 15

moca_set_max_constellation
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_max_constellation(void *vctx, const struct
moca_max_constellation *in)
```

Description:
Set/Get max constellation on all carriers in the receive from a specified node (GCAP.32).

moca_get_rlapm_table_50
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_rlapm_table_50(void *vctx, struct
moca_rlapm_table_50 *out)
```

Description:
An array of Margin Adjustments per RX power for 50 MHz channel (MoCA 1.1) transmissions.
First value is for 0dBm received power.
Last value is for -65dBm received power.

The dB values for adjustments are multiplied by 2, in order to allow 1/2 dB resolution.

moca_set_rlapm_table_50
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_rlapm_table_50(void *vctx, struct
moca_rlapm_table_50 *in)
```

Description:
An array of Margin Adjustments per RX power for 50 MHz channel (MoCA 1.1) transmissions.
First value is for 0dBm received power.
Last value is for -65dBm received power.

The dB values for adjustments are multiplied by 2, in order to allow 1/2 dB resolution.

moca_get_phy_status
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_phy_status(void *vctx, uint32_t
*tx_gcd_power_reduction)
```

Description:

Retrieve status information about the MoCA PHY layer.

Parameters:
tx_gcd_power_reduction
  The Transmit Power Control back-off used for broadcast transmissions from this node (mocaIfTxGcdPowerReduction)
  Minimum: 0
  Maximum: 35

moca_get_rlapm_table_100
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_rlapm_table_100(void *vctx, struct moca_rlapm_table_100 *out)
```

Description:
An array of Margin Adjustments per RX power for 100 MHz channel (MoCA 2.0) transmissions.
First value is for 0dBm received power.
Last value is for -65dBm received power.

The dB values for adjustments are multiplied by 2, in order to allow 1/2 dB resolution.

moca_set_rlapm_table_100
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_rlapm_table_100(void *vctx, struct moca_rlapm_table_100 *in)
```

Description:
An array of Margin Adjustments per RX power for 100 MHz channel (MoCA 2.0) transmissions.
First value is for 0dBm received power.
Last value is for -65dBm received power.

The dB values for adjustments are multiplied by 2, in order to allow 1/2 dB resolution.

moca_get_rx_gain_params
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_rx_gain_params(void *vctx, uint32_t table_index, struct moca_rx_gain_params *out)
```

Description:
This function returns the RX gain general parameters

Parameters:
table_index

moca_get_tx_power_params
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_tx_power_params(void *vctx, struct
moca_tx_power_params_in *in, struct moca_tx_power_params *out)
```

Description:
This function returns the TX power parameters for the specified channel.\n channel mode <0-beacon, 1-primary, 2- secondary>\n txTableIndex <0x0-Single index 0, 0x10-Bonded index 0, 0x11-Bonded index 1,0x12-Bonded index 2,0x13- Bonded index 3,0x14-Bonded index 4>

moca_get_nv_cal_enable
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_nv_cal_enable(void *vctx, uint32_t *val)
```

Description:
Enable the mechanism whereby the MoCA firmware will reuse RF calibration data and Probe II results from previous boots. The data is stored by the host for use on subsequent MoCA core boots. This mechanism decreases the MoCA firmware boot time.

Parameters:
val
Default: 0
Minimum: 0
Maximum: 1

moca_set_nv_cal_enable
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_nv_cal_enable(void *vctx, uint32_t val)
```

Description:
Enable the mechanism whereby the MoCA firmware will reuse RF calibration data and Probe II results from previous boots. The data is stored by the host for use on subsequent MoCA core boots. This mechanism decreases the MoCA firmware boot time.

Parameters:
val
Default: 0
Minimum: 0
Maximum: 1

moca_get_rlapm_cap_50
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_rlapm_cap_50(void *vctx, uint32_t *val)
```
Parameters:
val
Default: 0

moca_set_rlapm_cap_50
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_rlapm_cap_50(void *vctx, uint32_t val)
```
Parameters:
val
Default: 0

moca_get_snr_margin_rs
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_snr_margin_rs(void *vctx, struct
moca_snr_margin_rs *out)
```

Description:
This parameter is used to configure the RS SNR margin on the MoCA interface. The snr_margin feature is intended for use only by advanced lab users. Values range from -3 to +25dB in steps of 1/256 dB. The resulting table entries must have similar or increasing values as the constellation increases.

moca_set_snr_margin_rs
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_snr_margin_rs(void *vctx, struct
```

```
moca_snr_margin_rs *in)
```

Description:
This parameter is used to configure the RS SNR margin on the MoCA interface. The snr_margin feature is intended for use only by advanced lab users. Values range from -3 to +25dB in steps of 1/256 dB. The resulting table entries must have similar or increasing values as the constellation increases. This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

moca_get_snr_margin_ldpc
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_snr_margin_ldpc(void *vctx, struct
moca_snr_margin_ldpc *out)
```

Description:
This parameter is used to configure the LDPC SNR margin on the MoCA interface. The snr_margin feature is intended for use only by advanced lab users. Values range from -3 to +25dB in steps of 1/256 dB. The resulting table entries must have similar or increasing values as the constellation increases.

moca_set_snr_margin_ldpc
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_snr_margin_ldpc(void *vctx, struct
moca_snr_margin_ldpc *in)
```

Description:
This parameter is used to configure the LDPC SNR margin on the MoCA interface. The snr_margin feature is intended for use only by advanced lab users. Values range from -3 to +25dB in steps of 1/256 dB. The resulting table entries must have similar or increasing values as the constellation increases. This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

moca_get_snr_margin_ldpc_sec_ch
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_snr_margin_ldpc_sec_ch(void *vctx, struct
moca_snr_margin_ldpc_sec_ch *out)
```

Description:
This parameter is used to configure the LDPC SNR margin on the secondary channel of the MoCA interface when bonded operation is in effect. The snr_margin feature is intended for use only by

advanced lab users. Values range from -3 to +25dB in steps of 1/256 dB. The resulting table entries must have similar or increasing values as the constellation increases.

moca_set_snr_margin_ldpc_sec_ch

Prototype:

```
MOCALIB_GEN_SET_FUNCTION int moca_set_snr_margin_ldpc_sec_ch(void *vctx, struct
moca_snr_margin_ldpc_sec_ch *in)
```

Description:

This parameter is used to configure the LDPC SNR margin on the secondary channel of the MoCA interface when bonded operation is in effect. The snr_margin feature is intended for use only by advanced lab users. Values range from -3 to +25dB in steps of 1/256 dB. The resulting table entries must have similar or increasing values as the constellation increases. This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

moca_get_snr_margin_ldpc_pre5

Prototype:

```
MOCALIB_GEN_GET_FUNCTION int moca_get_snr_margin_ldpc_pre5(void *vctx, struct
moca_snr_margin_ldpc_pre5 *out)
```

Description:

This parameter is used to configure the LDPC Preamble 5 SNR margin on the MoCA interface. The snr_margin feature is intended for use only by advanced lab users. Values range from -3 to +25dB in steps of 1/256 dB. The resulting table entries must have similar or increasing values as the constellation increases.

moca_set_snr_margin_ldpc_pre5

Prototype:

```
MOCALIB_GEN_SET_FUNCTION int moca_set_snr_margin_ldpc_pre5(void *vctx, struct
moca_snr_margin_ldpc_pre5 *in)
```

Description:

This parameter is used to configure the LDPC Preamble 5 SNR margin on the MoCA interface. The snr_margin feature is intended for use only by advanced lab users. Values range from -3 to +25dB in steps of 1/256 dB. The resulting table entries must have similar or increasing values as the constellation increases. This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

moca_get_snr_margin_ofdma

Prototype:

```
MOCALIB_GEN_GET_FUNCTION int moca_get_snr_margin_ofdma(void *vctx, struct
moca_snr_margin_ofdma *out)
```

Description:

This parameter is used to configure the OFDMA SNR margin on the MoCA interface. The snr_margin feature is intended for use only by advanced lab users. Values range from -3 to +25dB in steps of 1/256 dB. The resulting table entries must have similar or increasing values as the constellation increases.

moca_set_snr_margin_ofdma

Prototype:

```
MOCALIB_GEN_SET_FUNCTION int moca_set_snr_margin_ofdma(void *vctx, struct
moca_snr_margin_ofdma *in)
```

Description:

This parameter is used to configure the OFDMA SNR margin on the MoCA interface. The snr_margin feature is intended for use only by advanced lab users. Values range from -3 to +25dB in steps of 1/256 dB. The resulting table entries must have similar or increasing values as the constellation increases. This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

moca_get_rlapm_cap_100

Prototype:

```
MOCALIB_GEN_GET_FUNCTION int moca_get_rlapm_cap_100(void *vctx, uint32_t *val)
```

Parameters:

val

Default: 0

moca_set_rlapm_cap_100

Prototype:

```
MOCALIB_GEN_SET_FUNCTION int moca_set_rlapm_cap_100(void *vctx, uint32_t val)
```

Parameters:

val

Default: 0

moca_get_sapm_table_50

Prototype:

```
MOCALIB_GEN_GET_FUNCTION int moca_get_sapm_table_50(void *vctx, struct
moca_sapm_table_50 *out)
```

Description:

Sub-carrier Added PHY Margin table for 50 MHz (MoCA 1.1) transmission profiles. These arrays allows differentiation in SNR margin values per sub carrier.

The values will be passed multiplied by 2, for allowing 0.5dB values.

moca_set_sapm_table_50

Prototype:

```
MOCALIB_GEN_SET_FUNCTION int moca_set_sapm_table_50(void *vctx, struct
moca_sapm_table_50 *in)
```

Description:

Sub-carrier Added PHY Margin table for 50 MHz (MoCA 1.1) transmission profiles. These arrays allows differentiation in SNR margin values per sub carrier.

The values will be passed multiplied by 2, for allowing 0.5dB values.

moca_get_sapm_table_100

Prototype:

```
MOCALIB_GEN_GET_FUNCTION int moca_get_sapm_table_100(void *vctx, struct
moca_sapm_table_100 *out)
```

Description:

Sub-carrier Added PHY Margin table for 100 MHz (MoCA 2.0) transmission profiles. These arrays allows differentiation in SNR margin values per sub carrier.

The values will be passed multiplied by 2, for allowing 0.5dB values.

moca_set_sapm_table_100

Prototype:

```
MOCALIB_GEN_SET_FUNCTION int moca_set_sapm_table_100(void *vctx, struct
moca_sapm_table_100 *in)
```

Description:

Sub-carrier Added PHY Margin table for 100 MHz (MoCA 2.0) transmission profiles. These arrays allows differentiation in SNR margin values per sub carrier.

The values will be passed multiplied by 2, for allowing 0.5dB values.

moca_set_nv_cal_clear
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_nv_cal_clear(void *vctx)
```

Description:
Clear the NVRAM RF Calibration data. This will force the MoCA core to perform calibration upon the next initialization if NV Calibration is enabled (see nv_cal_enable).

moca_get_sapm_table_sec
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_sapm_table_sec(void *vctx, struct
moca_sapm_table_sec *out)
```

Description:
Sub-carrier Added PHY Margin table for 100 MHz (MoCA 2.0) secondary channel transmission profiles. These arrays allows differentiation in SNR margin values per sub carrier.

The values will be passed multiplied by 2, for allowing 0.5dB values.

moca_set_sapm_table_sec
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_sapm_table_sec(void *vctx, struct
moca_sapm_table_sec *in)
```

Description:
Sub-carrier Added PHY Margin table for 100 MHz (MoCA 2.0) secondary channel transmission profiles. These arrays allows differentiation in SNR margin values per sub carrier.

The values will be passed multiplied by 2, for allowing 0.5dB values.

moca_get_amp_reg
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_amp_reg(void *vctx, uint32_t addr, struct
moca_amp_reg *out)
```

Description:
Read and write registers in the PA/LNA 345x chip.

Parameters:
addr
The address of the register to get or set. This function will only succeed when used with kernels that support register access to the PA/LNA chip.

moca_set_amp_reg
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_amp_reg(void *vctx, const struct
moca_amp_reg *in)
```

Description:
Read and write registers in the PA/LNA 345x chip.

moca_get_snr_margin_ldpc_pri_ch
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_snr_margin_ldpc_pri_ch(void *vctx, struct
moca_snr_margin_ldpc_pri_ch *out)
```

Description:
This parameter is used to configure the LDPC SNR margin on the primary channel of the MoCA interface when bonded operation is in effect. The snr_margin feature is intended for use only by advanced lab users. Values range from -3 to +25dB in steps of 1/256 dB. The resulting table entries must have similar or increasing values as the constellation increases.

moca_set_snr_margin_ldpc_pri_ch
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_snr_margin_ldpc_pri_ch(void *vctx, struct
moca_snr_margin_ldpc_pri_ch *in)
```

Description:
This parameter is used to configure the LDPC SNR margin on the primary channel of the MoCA interface when bonded operation is in effect. The snr_margin feature is intended for use only by advanced lab users. Values range from -3 to +25dB in steps of 1/256 dB. The resulting table entries must have similar or increasing values as the constellation increases. This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

MAC_LAYER Group

The MAC group of parameters control the Media Access Control layer of the MoCA interface.

Structures

struct moca_rtr_config

Fields:

uint8_t	bg	The number of retries allowed for background priority flows. *Default:* 0 *Minimum:* 0 *Maximum:* 3
uint8_t	high	The number of retries allowed for high priority flows. *Default:* 0 *Minimum:* 0 *Maximum:* 3
uint8_t	low	The number of retries allowed for low priority flows. *Default:* 0 *Minimum:* 0 *Maximum:* 3
uint8_t	med	The number of retries allowed for medium priority flows. *Default:* 0 *Minimum:* 0 *Maximum:* 3

Functions

moca_get_max_frame_size

Prototype:

```
MOCALIB_GEN_GET_FUNCTION int moca_get_max_frame_size(void *vctx, uint32_t *bytes)
```

Description:

Maximum frame size allowed to be transmitted through the MoCA physical interface. Used for limiting aggregation. Note: the frame size includes payloads headers and CRC's.

Parameters:

bytes

Default: 32768

Minimum: 2048
Maximum: 32768

moca_set_max_frame_size
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_max_frame_size(void *vctx, uint32_t bytes)
```

Description:
Maximum frame size allowed to be transmitted through the MoCA physical interface. Used for limiting aggregation. Note: the frame size includes payloads headers and CRC's.

Parameters:
bytes
Default: 32768
Minimum: 2048
Maximum: 32768

moca_get_min_aggr_waiting_time
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_min_aggr_waiting_time(void *vctx, uint32_t
*val)
```

Description:
When the GCAP.34 is ON the MIN_AGGR_WAITING_TIME = MAX_AGGR_PACKETS*1/ (RATE/8/PACKET_SIZE).

RATE=1 Mbps, PACKET_SIZE=64 Bytes => MAX_AGGR_PACKETS * 512 usec (=3072 for max aggregation of 6; and 5120 for max aggregation of 10)

RATE=64 Mbps, PACKET_SIZE=800 bytes => MAX_AGGR_PACKETS * 100 usec (=600 for max aggregation of 6; and 1000 for max aggregation of 10)

When the GCAP.34 is OFF the MIN_AGGR_WAITING_TIME = 0.

Parameters:
val
Default: 0

moca_set_min_aggr_waiting_time

Prototype:

```
MOCALIB_GEN_SET_FUNCTION int moca_set_min_aggr_waiting_time(void *vctx, uint32_t
val)
```

Description:

When the GCAP.34 is ON the MIN_AGGR_WAITING_TIME = MAX_AGGR_PACKETS*1/ (RATE/8/PACKET_SIZE).

RATE=1 Mbps, PACKET_SIZE=64 Bytes => MAX_AGGR_PACKETS * 512 usec (=3072 for max aggregation of 6; and 5120 for max aggregation of 10)

RATE=64 Mbps, PACKET_SIZE=800 bytes => MAX_AGGR_PACKETS * 100 usec (=600 for max aggregation of 6; and 1000 for max aggregation of 10)

When the GCAP.34 is OFF the MIN_AGGR_WAITING_TIME = 0.

Parameters:

val

Default: 0

moca_get_selective_rr

Prototype:

```
MOCALIB_GEN_GET_FUNCTION int moca_get_selective_rr(void *vctx, uint32_t *val)
```

Parameters:

val

Default: 3

moca_set_selective_rr

Prototype:

```
MOCALIB_GEN_SET_FUNCTION int moca_set_selective_rr(void *vctx, uint32_t val)
```

Parameters:

val

Default: 3

moca_get_max_transmit_time

Prototype:

```
MOCALIB_GEN_GET_FUNCTION int moca_get_max_transmit_time(void *vctx, uint32_t
```

```
*usec)
```

Description:
Maximum transmission time allowed to transmit a frame through the MoCA physical interface. Used for limiting aggregation

Parameters:
usec
Default: 400
Minimum: 300
Maximum: 1000

moca_set_max_transmit_time
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_max_transmit_time(void *vctx, uint32_t usec)
```

Description:
Maximum transmission time allowed to transmit a frame through the MoCA physical interface. Used for limiting aggregation

Parameters:
usec
Default: 400
Minimum: 300
Maximum: 1000

moca_get_max_pkt_aggr
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_max_pkt_aggr(void *vctx, uint32_t *pkts)
```

Description:
Max allowed packets for aggregated transmissions (enhanced GCAP.34)

Parameters:
pkts
Default: 20
Minimum: 1
Maximum: 20

moca_set_max_pkt_aggr
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_max_pkt_aggr(void *vctx, uint32_t pkts)
```

Description:
Max allowed packets for aggregated transmissions (enhanced GCAP.34)

Parameters:
pkts
Default: 20
Minimum: 1
Maximum: 20

moca_get_rtr_config
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_rtr_config(void *vctx, struct
moca_rtr_config *out)
```

Description:
Configure retransmission behavior in the node for non-PQOS flows.

moca_set_rtr_config
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_rtr_config(void *vctx, struct
moca_rtr_config *in)
```

Description:
Configure retransmission behavior in the node for non-PQOS flows.

moca_get_tlp_mode
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_tlp_mode(void *vctx, uint32_t *mode)
```

Description:
TLP mode (GCAP.107)

Parameters:
mode
Default: 1
Minimum: 1

Maximum: 2

moca_set_tlp_mode
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_tlp_mode(void *vctx, uint32_t mode)
```

Description:
TLP mode (GCAP.107)

Parameters:
mode
Default: 1
Minimum: 1
Maximum: 2

moca_get_max_pkt_aggr_bonding
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_max_pkt_aggr_bonding(void *vctx, uint32_t
*pkts)
```

Description:
Max allowed packets for aggregated transmissions for bonding

Parameters:
pkts
Default: 27
Minimum: 1
Maximum: 30

moca_set_max_pkt_aggr_bonding
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_max_pkt_aggr_bonding(void *vctx, uint32_t
pkts)
```

Description:
Max allowed packets for aggregated transmissions for bonding

Parameters:

pkts
Default: 27
Minimum: 1
Maximum: 30

FORWARDING Group

The Forwarding group of parameters are used to control how data traffic is handled by the MoCA core.

Structures

struct moca_egr_mc_addr_filter

Fields:

macaddr_t	addr	
uint32_t	entryid	*Minimum:* 0 *Maximum:* 31
uint16_t	reserved_0	
uint32_t	valid	

struct moca_egr_mc_addr_filter_set

Fields:

macaddr_t	addr	
uint32_t	entryid	*Minimum:* 0 *Maximum:* 31
uint32_t	valid	

struct moca_mc_fwd

Fields:

uint32_t	dest_node_id	MoCA node ID of the Destination *Minimum:* 0 *Maximum:* 15
macaddr_t	multicast_mac_addr	MAC address mapped to the MoCA Dest node ID
uint16_t	reserved_0	

struct moca_mc_fwd_set
Fields:

macaddr_t	dest_mac_addr1	To Add/Update entry: 1st destination MAC address in this MC group. To delete the entry: Set to zero
macaddr_t	dest_mac_addr2	To Add/Update entry: 2nd destination MAC address in this MC group. To delete entry: reserved
macaddr_t	dest_mac_addr3	To Add/Update entry: 3rd destination MAC address in this MC group. To delete entry: reserved
macaddr_t	dest_mac_addr4	To Add/Update entry: 4th destination MAC address in this MC group. To delete entry: reserved
macaddr_t	multicast_mac_addr	The multicast address as learned from the IGMP snooping

struct moca_mcfilter_addentry
Fields:

macaddr_t	addr

struct moca_mcfilter_delentry
Fields:

macaddr_t	addr

struct moca_mcfilter_table
Fields:

macaddr_t	addr[48]

struct moca_pqos_create_flow_in
Fields:

uint32_t	burst_size	Number of packets per burst. *Default:* 2 *Minimum:* 0 *Maximum:* 10
uint32_t	dscp_moca	The value of the three MSB of the DSCP Type of Service field used for MSDU classification when ingr_class_ryle is set to 1 or 3. *Default:* 0 *Minimum:* 0 *Maximum:* 7
macaddr_t	egress_node	MAC address of the egress node of the flow. If more than 1 node will be on the egress of this flow, set this field to broadcast MAC address (FF:FF:FF:FF:FF:FF).
macaddr_t	flow_id	Flow identifier in the form of a multicast MAC address *Default:* 01:00:5e:00:01:00
uint32_t	flow_per	Used to specify whether the flow should use the Nominal packet error rate (PER) PHY profile or the Very Low PER PHY profile. *Default:* 0 *Minimum:* 0 *Maximum:* 1
uint32_t	flow_tag	Optional identifier for application use. *Default:* 0
uint32_t	in_order_delivery	Indication of recommendation for Egress Node to deliver retransmitted MSDUs belonging to this PQoS Flow in order. *Default:* 0 *Minimum:* 0 *Maximum:* 2
uint32_t	ingr_class_rule	Ingress classification rule for assigning MSDUs to the PQOS flow. *Default:* 0 *Valid Values:* 0, 4, 5, 6, 7
macaddr_t	ingress_node	MAC address of the ingress node of the flow.
uint32_t	lease_time	Lease time in seconds. Infinite lease time if set to zero. *Default:* 0
uint32_t	max_latency	The maximum latency of the flow. *Default:* 0 *Minimum:* 0 *Maximum:* 255
uint32_t	max_retry	Maximum number of retransmission attempts for each MSDU of the PQoS Flow. *Default:* 0 *Minimum:* 0 *Maximum:* 3
macaddr_t	packet_da	Destination MAC address of the actual traffic to be sent on this flow.

		Default: 01:00:5e:00:01:00
uint32_t	packet_size	Packet size in bytes, including the VLAN header but not including the FCS. *Default:* 800 *Minimum:* 59 *Maximum:* 1519
uint32_t	peak_data_rate	Peak data rate in kbps *Default:* 1000 *Minimum:* 1 *Maximum:* 0xFFFFFF
uint32_t	short_term_avg_ratio	The ratio of the short term average rate of the flow compared to the peak rate over the interval of max_latency. This value plus one serves as the numerator of the ratio. The denominator is 256. This value is only applicable when the max_latency value is greater than or equal to 10 ms. *Default:* 255 *Minimum:* 0 *Maximum:* 255
uint32_t	traffic_protocol	The type of traffic carried over this PQOS flow. *Default:* 0 *Minimum:* 0 *Maximum:* 5
uint32_t	vlan_id	VLAN ID of this flow (optional, set to 0xFFFFFFFF if unused). *Default:* 0xFFFFFFFF
uint32_t	vlan_tag	The VLAN priority used for MSDU classification when ingr_class_rule is set to 6 or 7. *Default:* 5 *Minimum:* 0 *Maximum:* 7

struct moca_pqos_create_flow_out

Fields:

uint32_t	burst_size
uint32_t	bw_limit_info
uint32_t	decision
uint32_t	dest_flow_id
uint32_t	dscp_moca
macaddr_t	flow_id
uint32_t	flow_per
uint32_t	flow_stps
uint32_t	flow_tag
uint32_t	flow_txps

macaddr_t	flowda
uint32_t	in_order_delivery
uint32_t	ingr_class_rule
uint32_t	lease_time
uint32_t	max_number_retry
uint32_t	max_short_term_avg_ratio
uint32_t	maximum_latency
uint32_t	packet_size
uint32_t	peak_data_rate
uint32_t	response_code
uint32_t	short_term_avg_ratio
uint32_t	total_stps
uint32_t	total_txps
uint32_t	traffic_protocol
uint32_t	vlan_tag

struct moca_pqos_delete_flow_out

Fields:

macaddr_t	flowid
uint32_t	response_code

struct moca_pqos_list_in

Fields:

uint16_t	flow_max_return	The maximum number of flows to be returned in this operation. *Default:* 32 *Minimum:* 0 *Maximum:* 32
uint32_t	flow_start_index	The index of the first flow to be returned from the requested node. *Default:* 0
uint32_t	ingr_node_id	Node ID of the ingress node. Only used if ingress_node_mac is set to zero.
macaddr_t	ingr_node_mac	MAC address of the ingress node. Set to 00:00:00:00:00:00 to specify the node using the ingress_node_id parameter. *Default:* 00:00:00:00:00:00

struct moca_pqos_list_out

Fields:

uint32_t	flow_update_count

macaddr_t	flowid[32]	
uint32_t	num_ret_flow_ids	This is the total number of valid ingress flows that were returned in this operation. This value is a count of the non-zero flowid entries.
uint32_t	response_code	
uint32_t	total_flow_id_count	This is the total number of ingress flows that exist on this node.

struct moca_pqos_maintenance_complete

Fields:

uint32_t	allocatedstps
uint32_t	allocatedtxps
uint32_t	iocovercommit

struct moca_pqos_query_out

Fields:

uint32_t	burst_size	Number of packets per burst.
uint32_t	dest_flow_id	The destination flow ID of this flow if it exists, zero otherwise.
uint32_t	dscp_moca	The value of the three MSB of the DSCP Type of Service field used for MSDU classification when ingr_class_rule is set to 1 or 3.
macaddr_t	egress_node	MAC address of the egress node of the flow.
macaddr_t	flow_id	Flow identifier in the form of a multicast MAC address
uint32_t	flow_per	Used to specify whether the flow should use the Nominal packet error rate (PER) PHY profile or the Very Low PER PHY profile.
uint32_t	flow_tag	Optional identifier for application use.
uint32_t	in_order_delivery	Indication of recommendation for Egress Node to deliver retransmitted MSDUs belonging to this PQoS Flow in order.
uint32_t	ingr_class_rule	Ingress classification rule for assigning MSDUs to the PQOS flow.
macaddr_t	ingress_node	MAC address of the ingress node of the flow.
uint32_t	lease_time	Original lease time of the flow in seconds. Infinite lease time if set to zero.
uint32_t	lease_time_left	Number of seconds remaining in the lease of the flow. Infinite lease time if set to zero.
uint32_t	max_latency	Maximum latency of the flow in units of ms.
uint32_t	max_retry	Maximum number of retransmission attempts for each MSDU of the PQoS Flow.
macaddr_t	packet_da	Destination MAC address of the actual traffic to be sent on this flow.
uint32_t	packet_size	Packet size in bytes.

uint32_t	peak_data_rate	Peak data rate in kbps
uint32_t	response_code	
uint32_t	short_term_avg_ratio	The ratio of the short term average rate of the flow compared to the peak rate over the interval of max_latency. This value plus one serves as the numerator of the ratio. The denominator is 256. This value is only applicable when the max_latency value is greater than or equal to 10 ms.
uint32_t	traffic_protocol	The type of traffic carried over this PQOS flow.
uint32_t	vlan_tag	The VLAN tag used for MSDU classification when ingr_class_rule is set to 2 or 3.

struct moca_pqos_status_out

Fields:

uint32_t	flow_stps	The number of available PQOS slot times per second in this node.
uint32_t	flow_txps	The number of available PQOS transmissions per second in the network.
uint32_t	total_stps	The summed up number of PQOS slot times per second in the network.
uint32_t	total_txps	The summed up number of PQOS transmissions per second in the network.

struct moca_pqos_update_flow_in

Fields:

uint32_t	burst_size	Number of packets per burst. *Minimum:* 1 *Maximum:* 9
macaddr_t	egress_mac	MAC address of the egress node. This can be obtained for the given Flow ID from a PQOS Query operation.
macaddr_t	flow_id	Flow identifier in the form of a multicast MAC address *Default:* 01:00:5e:00:00:00
uint32_t	flow_per	Used to specify whether the flow should use the Nominal packet error rate (PER) PHY profile or the Very Low PER PHY profile. *Default:* 0 *Minimum:* 0 *Maximum:* 1
uint32_t	flow_tag	Optional identifier for application use.
uint32_t	in_order_delivery	Indication of recommendation for Egress Node to deliver retransmitted MSDUs belonging to this PQoS Flow in order. *Default:* 0 *Minimum:* 0 *Maximum:* 2
macaddr_t	ingress_mac	MAC address of the ingress node. This can be obtained for the given Flow ID from a PQOS Query operation.

uint32_t	lease_time	Lease time in seconds. Infinite lease time if set to zero.
uint32_t	max_latency	The maximum latency of the flow. *Default:* 0 *Minimum:* 0 *Maximum:* 255
uint32_t	max_retry	Maximum number of retransmission attempts for each MSDU of the PQoS Flow. *Default:* 0 *Minimum:* 0 *Maximum:* 3
uint32_t	packet_size	Packet size in bytes. *Minimum:* 59 *Maximum:* 1518
uint32_t	peak_data_rate	Peak data rate in kbps *Minimum:* 1 *Maximum:* 0xFFFFFE
uint16_t	reserved	*Default:* 0
uint32_t	short_term_avg_ratio	The ratio of the short term average rate of the flow compared to the peak rate over the interval of max_latency. This value plus one serves as the numerator of the ratio. The denominator is 256. This value is only applicable when the max_latency value is greater than or equal to 10 ms. *Default:* 255 *Minimum:* 0 *Maximum:* 255
uint32_t	traffic_protocol	The type of traffic carried over this PQOS flow. *Default:* 0 *Minimum:* 0 *Maximum:* 5

struct moca_pqos_update_flow_out

Fields:

uint32_t	burst_size
uint32_t	bw_limit_info
uint32_t	decision
uint32_t	dscp_moca
uint32_t	flow_per
uint32_t	flow_stps
uint32_t	flow_tag
uint32_t	flow_txps
macaddr_t	flowda
macaddr_t	flowid

uint32_t	in_order_delivery
uint32_t	ingr_class_rule
uint32_t	lease_time
uint32_t	max_number_retry
uint32_t	max_short_term_avg_ratio
uint32_t	maximum_latency
uint32_t	packet_size
uint32_t	peak_data_rate
uint32_t	response_code
uint32_t	short_term_avg_ratio
uint32_t	total_stps
uint32_t	total_txps
uint32_t	traffic_protocol
uint32_t	vlan_tag

struct moca_src_addr

Fields:

macaddr_t	mac_addr	MAC address sourced from this device
uint16_t	moca_node_id	Self Node Id

struct moca_stag_priority

Fields:

uint32_t	enable	*Default:* 0 *Minimum:* 0 *Maximum:* 1
uint32_t	moca_priority_0	*Default:* 0 *Minimum:* 0 *Maximum:* 4
uint32_t	moca_priority_1	*Default:* 0 *Minimum:* 0 *Maximum:* 4
uint32_t	moca_priority_2	*Default:* 0 *Minimum:* 0 *Maximum:* 4
uint32_t	moca_priority_3	*Default:* 0 *Minimum:* 0 *Maximum:* 4
uint32_t	moca_priority_4	*Default:* 0 *Minimum:* 0

		Maximum: 4
uint32_t	moca_priority_5	*Default:* 0 *Minimum:* 0
uint32_t	moca_priority_6	*Default:* 0 *Minimum:* 0 *Maximum:* 4
uint32_t	moca_priority_7	*Default:* 0 *Minimum:* 0 *Maximum:* 4
uint32_t	tag_mask	*Default:* 0 *Minimum:* 0 *Maximum:* 0xFFFF
uint32_t	tag_priority_0	*Default:* 0 *Minimum:* 0 *Maximum:* 4
uint32_t	tag_priority_1	*Default:* 0 *Minimum:* 0 *Maximum:* 4
uint32_t	tag_priority_2	*Default:* 0 *Minimum:* 0 *Maximum:* 4
uint32_t	tag_priority_3	*Default:* 0 *Minimum:* 0 *Maximum:* 4
uint32_t	tag_priority_4	*Default:* 0 *Minimum:* 0 *Maximum:* 4
uint32_t	tag_priority_5	*Default:* 0 *Minimum:* 0 *Maximum:* 4
uint32_t	tag_priority_6	*Default:* 0 *Minimum:* 0 *Maximum:* 4
uint32_t	tag_priority_7	*Default:* 0 *Minimum:* 0 *Maximum:* 4

struct moca_stag_removal

Fields:

uint32_t	enable	*Default:* 0 *Minimum:* 0 *Maximum:* 1

uint32_t	mask_0	*Default:* 0xffffffff
uint32_t	mask_1	*Default:* 0xffffffff
uint32_t	mask_2	*Default:* 0xffffffff
uint32_t	mask_3	*Default:* 0xffffffff
uint32_t	valid_0	*Default:* 0 *Minimum:* 0 *Maximum:* 1
uint32_t	valid_1	*Default:* 0 *Minimum:* 0 *Maximum:* 1
uint32_t	valid_2	*Default:* 0 *Minimum:* 0 *Maximum:* 1
uint32_t	valid_3	*Default:* 0 *Minimum:* 0 *Maximum:* 1
uint32_t	value_0	*Default:* 0
uint32_t	value_1	*Default:* 0
uint32_t	value_2	*Default:* 0
uint32_t	value_3	*Default:* 0

struct moca_uc_fwd
Fields:

macaddr_t	mac_addr	MAC address mapped to the MoCA Dest node ID
uint16_t	moca_dest_node_id	MoCA node ID of the Destination *Minimum:* 0 *Maximum:* 15

Functions

moca_get_multicast_mode
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_multicast_mode(void *vctx, uint32_t *val)
```

Description:
Selecting a Mode of operation for MC. Normal mode is when the host has IGMP snooping ability. In CTP testing, the BC mode should be used.

Parameters:

val
Default: 1

moca_set_multicast_mode
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_multicast_mode(void *vctx, uint32_t val)
```

Description:
Selecting a Mode of operation for MC. Normal mode is when the host has IGMP snooping ability. In CTP testing, the BC mode should be used. This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

Parameters:
val
Default: 1

moca_get_egr_mc_filter_en
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_egr_mc_filter_en(void *vctx, uint32_t *val)
```

Description:
Enables/Disables Egress Eth MC Packet Filtering mode.
When enabled, only the Eth MC packets with MAC Address that are match to an entry in the MC filter table are delivered through the GMII interface.

Parameters:
val
Default: 0

moca_set_egr_mc_filter_en
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_egr_mc_filter_en(void *vctx, uint32_t val)
```

Description:
Enables/Disables Egress Eth MC Packet Filtering mode.
When enabled, only the Eth MC packets with MAC Address that are match to an entry in the MC

filter table are delivered through the GMII interface. This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

Parameters:
val
Default: 0

moca_get_fc_mode
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_fc_mode(void *vctx, uint32_t *val)
```

Description:
Set or get the flow control mode.

Parameters:
val
Default:
0 (FC_CAPABLE_CHIP)
1
Minimum: 0
Maximum: 1

moca_set_fc_mode
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_fc_mode(void *vctx, uint32_t val)
```

Description:
Set or get the flow control mode. This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

Parameters:
val
Default:
0 (FC_CAPABLE_CHIP)
1
Minimum: 0
Maximum: 1

moca_get_per_mode
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_per_mode(void *vctx, uint32_t *mode)
```

Description:
Controls which transmission PER mode the Node uses for MPDUs not belonging to PQoS Flows

Parameters:
mode
Default: 0
Minimum: 0
Maximum: 1

moca_set_per_mode
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_per_mode(void *vctx, uint32_t mode)
```

Description:
Controls which transmission PER mode the Node uses for MPDUs not belonging to PQoS Flows

Parameters:
mode
Default: 0
Minimum: 0
Maximum: 1

moca_get_policing_en
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_policing_en(void *vctx, uint32_t *enable)
```

Description:
Controls whether policing of PQoS Flows is enabled or disabled

Parameters:
enable
Default: 0
Minimum: 0
Maximum: 1

moca_set_policing_en
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_policing_en(void *vctx, uint32_t enable)
```

Description:
Controls whether policing of PQoS Flows is enabled or disabled

Parameters:
enable
Default: 0
Minimum: 0
Maximum: 1

moca_get_pqos_egress_numflows
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_pqos_egress_numflows(void *vctx, uint32_t
*pqos_egress_numflows)
```

Description:
Retrieve the number of PQoS Flows in which this node is an Egress node.

Parameters:
pqos_egress_numflows
Number of PQoS Flows in which this node is an Egress node (mocaIfEgressNodeNumFlows)
Minimum: 0
Maximum: 12

moca_get_orr_en
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_orr_en(void *vctx, uint32_t *enable)
```

Description:
Controls whether or not Opportunistic Reservation Requests are to be used for MoCA 2.0 PQoS flows.

Parameters:
enable
Default: 0
Minimum: 0
Maximum: 1

moca_set_orr_en
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_orr_en(void *vctx, uint32_t enable)
```

Description:
Controls whether or not Opportunistic Reservation Requests are to be used for MoCA 2.0 PQoS flows.

Parameters:
enable
Default: 0
Minimum: 0
Maximum: 1

moca_get_brcmtag_enable
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_brcmtag_enable(void *vctx, uint32_t *enable)
```

Description:
Enable BRCM tag usage on packets from switch

Parameters:
enable
Default: 0
Minimum: 0
Maximum: 1

moca_set_brcmtag_enable
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_brcmtag_enable(void *vctx, uint32_t enable)
```

Description:
Enable BRCM tag usage on packets from switch

Parameters:
enable
Default: 0

Minimum: 0
Maximum: 1

moca_get_egr_mc_addr_filter
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_egr_mc_addr_filter(void *vctx, uint32_t
entryid, struct moca_egr_mc_addr_filter *out)
```

Description:
Get Multicast MAC Address filtering entry.

Parameters:
entryid
Minimum: 0
Maximum: 31

moca_set_egr_mc_addr_filter
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_egr_mc_addr_filter(void *vctx, struct
moca_egr_mc_addr_filter_set *in)
```

Description:
Set Multicast MAC Address filtering entry.

moca_set_pqos_maintenance_start
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_pqos_maintenance_start(void *vctx)
```

Description:
Activating an internal maintenance process

moca_get_uc_fwd
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_uc_fwd(void *vctx, struct moca_uc_fwd *out,
int max_out_len)
```

Description:

UC Forwarding Table
Note: The UC Forwarding Table is a Read-only table from the host.

moca_get_mc_fwd
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_mc_fwd(void *vctx, struct moca_mc_fwd *out,
int max_out_len)
```

Description:
MC Forwarding Table
The MC FWD table is special in the sense that reading READ and WRITE parameters are different. When writing to the table, the input parameters are unicast MAC addresses. When reading the table, the output parameter is Node ID.
Note: This IE is applicable only if MULTICAST_MODE = Normal.

moca_set_mc_fwd
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_mc_fwd(void *vctx, struct moca_mc_fwd_set
*in)
```

moca_get_src_addr
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_src_addr(void *vctx, struct moca_src_addr
*out, int max_out_len)
```

Description:
SRC Addresses Table
Note: The SRS Addresses Table is a Read-only table from the host

moca_get_loopback_en
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_loopback_en(void *vctx, uint32_t *en)
```

Description:
In loopback mode, the data traffic coming from the Coax will be looped back into the Coax. In 6816 the loopback is a SW loopback, at the MAC level (no supporting HW available) In 7xxx the loopback is by HW at the ECL level. For UC, SA and DA will be flipped. For MC, the SA will be local MAC device, and DA will be a new address according to spec.

Parameters:
en
Default: 0

moca_set_loopback_en
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_loopback_en(void *vctx, uint32_t en)
```

Description:
In loopback mode, the data traffic coming from the Coax will be looped back into the Coax. In 6816 the loopback is a SW loopback, at the MAC level (no supporting HW available) In 7xxx the loopback is by HW at the ECL level. For UC, SA and DA will be flipped. For MC, the SA will be local MAC device, and DA will be a new address according to spec.

Parameters:
en
Default: 0

moca_get_mcfilter_enable
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_mcfilter_enable(void *vctx, uint32_t *val)
```

Description:
Enables/Disables multicast Filter mode or enable on DFID only.

Parameters:
val
Default: 0

moca_set_mcfilter_enable
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_mcfilter_enable(void *vctx, uint32_t val)
```

Description:
Enables/Disables multicast Filter mode or enable on DFID only.

Parameters:
val
Default: 0

moca_set_mcfilter_addentry
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_mcfilter_addentry(void *vctx, struct
moca_mcfilter_addentry *in)
```

Description:
Add Multicast MAC Address filtering entry.

moca_set_mcfilter_delentry
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_mcfilter_delentry(void *vctx, struct
moca_mcfilter_delentry *in)
```

Description:
Delete Multicast MAC Address filtering entry.

moca_get_pause_fc_en
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_pause_fc_en(void *vctx, uint32_t *val)
```

Description:
Enable/Disable Pause Frame Flow Control for packets destined for the MoCA network, if available. Not all MoCA chips support Pause frames.

Parameters:
val
Default:
1 (STANDALONE)
0
Minimum: 0
Maximum: 1

moca_set_pause_fc_en
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_pause_fc_en(void *vctx, uint32_t val)
```

Description:
Enable/Disable Pause Frame Flow Control for packets destined for the MoCA network, if available. Not all MoCA chips support Pause frames.

Parameters:
val
Default:
1 (STANDALONE)
0
Minimum: 0
Maximum: 1

moca_get_stag_priority
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_stag_priority(void *vctx, struct
moca_stag_priority *out)
```

Description:
Mapping between stag priority and MoCA priority

moca_set_stag_priority
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_stag_priority(void *vctx, const struct
moca_stag_priority *in)
```

Description:
Mapping between stag priority and MoCA priority

moca_get_stag_removal
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_stag_removal(void *vctx, struct
moca_stag_removal *out)
```

Description:
Tag reference table, used for tag removal

moca_set_stag_removal
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_stag_removal(void *vctx, const struct
moca_stag_removal *in)
```

Description:
Tag reference table, used for tag removal

moca_register_ucfwd_update_cb
Prototype:
```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_ucfwd_update_cb(void *vctx, void
(*callback)(void *userarg), void *userarg)
```

moca_register_pqos_maintenance_complete_cb
Prototype:
```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_pqos_maintenance_complete_cb(void
*vctx, void (*callback)(void *userarg, struct moca_pqos_maintenance_complete
*out), void *userarg)
```

moca_register_pqos_create_flow_cb
Prototype:
```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_pqos_create_flow_cb(void *vctx,
void (*callback)(void *userarg, struct moca_pqos_create_flow_out *out), void
*userarg)
```

Description:
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.

moca_do_pqos_create_flow
Prototype:
```
MOCALIB_GEN_DO_FUNCTION int moca_do_pqos_create_flow(void *vctx, struct
moca_pqos_create_flow_in *in, struct moca_pqos_create_flow_out *out)
```

Description:

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.

moca_register_pqos_update_flow_cb

Prototype:

```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_pqos_update_flow_cb(void *vctx,
void (*callback)(void *userarg, struct moca_pqos_update_flow_out *out), void
*userarg)
```

Description:

Update the parameters of an existing PQoS flow

moca_do_pqos_update_flow

Prototype:

```
MOCALIB_GEN_DO_FUNCTION int moca_do_pqos_update_flow(void *vctx, struct
moca_pqos_update_flow_in *in, struct moca_pqos_update_flow_out *out)
```

Description:

Update the parameters of an existing PQoS flow

moca_register_pqos_delete_flow_cb

Prototype:

```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_pqos_delete_flow_cb(void *vctx,
void (*callback)(void *userarg, struct moca_pqos_delete_flow_out *out), void
*userarg)
```

Description:

Deleting an existing PQoS flow

moca_do_pqos_delete_flow

Prototype:

```
MOCALIB_GEN_DO_FUNCTION int moca_do_pqos_delete_flow(void *vctx, macaddr_t
flow_id, struct moca_pqos_delete_flow_out *out)
```

Description:

Deleting an existing PQoS flow

moca_register_pqos_list_cb
Prototype:
```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_pqos_list_cb(void *vctx, void
(*callback)(void *userarg, struct moca_pqos_list_out *out), void *userarg)
```

Description:
Retrieving the list of flow IDs for a specific ingress node. The node can be selected by its MAC Addr or its node ID.
A maximum of 32 PQOS flows will be returned.

moca_do_pqos_list
Prototype:
```
MOCALIB_GEN_DO_FUNCTION int moca_do_pqos_list(void *vctx, struct moca_pqos_list_in
*in, struct moca_pqos_list_out *out)
```

Description:
Retrieving the list of flow IDs for a specific ingress node. The node can be selected by its MAC Addr or its node ID.
A maximum of 32 PQOS flows will be returned.

moca_register_pqos_query_cb
Prototype:
```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_pqos_query_cb(void *vctx, void
(*callback)(void *userarg, struct moca_pqos_query_out *out), void *userarg)
```

Description:
Retrieving a specific PQoS flow parameters

moca_do_pqos_query
Prototype:
```
MOCALIB_GEN_DO_FUNCTION int moca_do_pqos_query(void *vctx, macaddr_t flow_id,
struct moca_pqos_query_out *out)
```

Description:
Retrieving a specific PQoS flow parameters

moca_register_pqos_status_cb
Prototype:
```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_pqos_status_cb(void *vctx, void
```

```
(*callback)(void *userarg, struct moca_pqos_status_out *out), void *userarg)
```

Description:
Perform a PQOS Status operation to obtain the available PQOS resources on this node.

moca_do_pqos_status
Prototype:
```
MOCALIB_GEN_DO_FUNCTION int moca_do_pqos_status(void *vctx, uint32_t unused,
struct moca_pqos_status_out *out)
```

Description:
Perform a PQOS Status operation to obtain the available PQOS resources on this node.

moca_set_mcfilter_clear_table
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_mcfilter_clear_table(void *vctx)
```

Description:
Clear Multicast filtering table.

moca_get_mcfilter_table
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_mcfilter_table(void *vctx, struct
moca_mcfilter_table *out)
```

Description:
Get Multicast MAC Address filtering entry.

moca_get_host_qos
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_host_qos(void *vctx, uint32_t *enable)
```

Description:
Controls whether or not mocad will automatically create filters for prioritizing MoCA traffic, including PQOS traffic.

Parameters:
enable

Default:
1 (FC_CAPABLE_CHIP)
0

moca_set_host_qos
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_host_qos(void *vctx, uint32_t enable)
```

Description:
Controls whether or not mocad will automatically create filters for prioritizing MoCA traffic, including PQOS traffic.

Parameters:
enable
Default:
1 (FC_CAPABLE_CHIP)
0

NETWORK Group
The Network group of parameters provide information about the MoCA network.

Structures

struct moca_aca_in
Fields:

uint32_t	channel	The channel number to perform the ACA on.
uint32_t	dest_nodemask	The bitmask of the destination nodes for the ACA. *Minimum:* 1 *Maximum:* 0xFFFF
uint32_t	num_probes	The number of probes to be used in the ACA operation. *Maximum:* 8
uint32_t	src_node	The Node ID of the source node for the ACA. *Minimum:* 0 *Maximum:* 15
uint32_t	type	The type of ACA to perform, either EVM or Quiet. *Default:* 1 *Minimum:* 1

Maximum: 2

struct moca_aca_out

Fields:

uint32_t	aca_status	
uint32_t	aca_type	
uint32_t	num_elements	The number of valid elements found in the power_profile array.
uint8_t	power_profile[512]	
int32_t	relative_power	
uint32_t	rx_status	
int32_t	total_power	
uint32_t	tx_status	

struct moca_dd_init_out

Fields:

uint32_t	ae_number[16]	The maximum number of allocation elements in one MAP that each node can process, as reported via the device discovery protocol.
uint32_t	aggr_pdus[16]	The maximum number of aggregated PDUs that can be received by each node, as reported via the device discovery protocol.
uint32_t	aggr_size[16]	The maximum number of aggregated bytes supported by each node, as reported via the device discovery protocol.
uint32_t	egress_pqos_flows [16]	The maximum number of egress PQOS flows supported by each node, as reported via the device discovery protocol.
uint32_t	ingress_pqos_flows [16]	The maximum number of ingress PQOS flows supported by each node, as reported via the device discovery protocol.
uint32_t	responded_nodemask	A bit mask of the nodes that responded to the DD request.
uint32_t	responsecode	

struct moca_error_stats

Fields:

uint32_t	rx_acf_crc_error	This is a counter of the total number of ACF frames received with a CRC error.
uint32_t	rx_bc_crc_error	This is a counter of the total broadcast packets received that have a CRC error.
uint32_t	rx_bc_timeout_error	This is a counter of the total broadcast packets that were lost due to timeout.
uint32_t	rx_beacon_crc_error	

		This is a counter of the total beacons received that have a CRC error.
uint32_t	rx_beacon_timeout_error	This is a counter of the total beacons that were lost due to timeout.
uint32_t	rx_lc_bc_crc_error	This is a counter of the total broadcast link control packets received that have a CRC error.
uint32_t	rx_lc_bc_timeout_error	This is a counter of the total broadcast link control packets that were lost due to timeout.
uint32_t	rx_lc_uc_crc_error	This is a counter of the total unicast link control packets received that have a CRC error.
uint32_t	rx_lc_uc_timeout_error	This is a counter of the total unicast link control packets that were lost due to timeout.
uint32_t	rx_map_crc_error	This is a counter of the total map packets received that have a CRC error.
uint32_t	rx_map_timeout_error	This is a counter of the total map packets that were lost due to timeout.
uint32_t	rx_ofdma_rr_crc_error	This is a counter of the total OFDMA RR packets received that have a CRC error.
uint32_t	rx_plp_crc_error	This is a counter of the total periodic link packets received that have a CRC error.
uint32_t	rx_plp_timeout_error	This is a counter of the total periodic link packets that were lost due to timeout.
uint32_t	rx_probe1_error	This is a counter of the total probe I packets received that have a CRC error on primary channel.
uint32_t	rx_probe1_error_sec_ch	This is a counter of the total probe I packets received that have a CRC error on secondary channel.
uint32_t	rx_probe1_gcd_error	This is a counter of the total probe I GCD packets received that have a CRC error.
uint32_t	rx_probe2_error	This is a counter of the total probe II packets received that have a CRC error.
uint32_t	rx_probe3_error	This is a counter of the total probe III packets received that have a CRC error.
uint32_t	rx_rr_crc_error	This is a counter of the total RR packets received that have a CRC error.
uint32_t	rx_rr_timeout_error	This is a counter of the total RR packets that were lost due to timeout.
uint32_t	rx_uc_crc_error	This is a counter of the total unicast packets received that have a CRC error on primary channel.
uint32_t	rx_uc_crc_error_sec_ch	This is a counter of the total unicast packets received that have a CRC error on secondary channel when bonded mode is active.
uint32_t	rx_uc_timeout_error	

		This is a counter of the total unicast packets that were lost due to timeout on primary channel.
uint32_t	rx_uc_timeout_error_sec_ch	This is a counter of the total unicast packets that were lost due to timeout on secondary channel when bonded mode is active.

struct moca_fmr_20_out

Fields:

uint8_t	node0_gap_gcd	In a mixed mode network, this is the inter-symbol time gap for the GCP PHY profile on the beacon channel, otherwise 0.
uint8_t	node0_gap_nper[16]	If index is the node ID of a MoCA 1.x node this is the inter-symbol time gap for the unicast profile. If index is the node ID of the entry node this is the inter-symbol time gap for the NPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the inter-symbol time gap for NPER unicast PHY Profile.
uint8_t	node0_gap_vlper[16]	If index is the node ID of a MoCA 1.x node this is 0. If index is the node ID of the entry node this is the inter-symbol time gap for the VLPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the inter-symbol time gap for VLPER Unicast PHY Profile.
uint8_t	node0_ofdma_def_tab_num	Number of valid OFDMA definition tables with responding node as the receiver of the OFDMA frames.
uint16_t	node0_ofdma_tab_bps[4]	The number of bits per symbol of each OFDMA subchannel for each of the OFDMA tables.
uint8_t	node0_ofdma_tab_gap[4]	The inter-symbol time gap for each of the OFDMA tables.
uint32_t	node0_ofdma_tab_node_bitmask [4]	The node bitmask for each of the OFDMA tables.
uint8_t	node0_ofdma_tab_num_subchan [4]	The number of subchannels for each of the OFDMA tables.
uint16_t	node0_ofdmb_gcd	In a mixed mode network, this is the number of bits per OFDM symbol for the GCP PHY profile on the beacon channel, otherwise 0.
uint16_t	node0_ofdmb_nper[16]	If index is the node ID of a MoCA 1.x node this is the number of bits per OFDM symbol for the unicast profile. If index is the node ID of the entry node this is the number of bits per OFDM symbol for the NPER GCD

		PHY Profile. If index is the node ID of a MoCA 2.0 node this is the number of bits per OFDM symbol for NPER unicast PHY Profile.
uint16_t	node0_ofdmb_vlper[16]	If index is the node ID of a MoCA 1.x node this is 0. If index is the node ID of the entry node this is the number of bits per OFDM symbol for the VLPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the number of bits per OFDM symbol for VLPER Unicast PHY Profile.
uint8_t	node10_gap_gcd	In a mixed mode network, this is the inter-symbol time gap for the GCP PHY profile on the beacon channel, otherwise 0.
uint8_t	node10_gap_nper[16]	If index is the node ID of a MoCA 1.x node this is the inter-symbol time gap for the unicast profile. If index is the node ID of the entry node this is the inter-symbol time gap for the NPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the inter-symbol time gap for NPER unicast PHY Profile.
uint8_t	node10_gap_vlper[16]	If index is the node ID of a MoCA 1.x node this is 0. If index is the node ID of the entry node this is the inter-symbol time gap for the VLPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the inter-symbol time gap for VLPER Unicast PHY Profile.
uint8_t	node10_ofdma_def_tab_num	Number of valid OFDMA definition tables with responding node as the receiver of the OFDMA frames.
uint16_t	node10_ofdma_tab_bps[4]	The number of bits per symbol of each OFDMA subchannel for each of the OFDMA tables.
uint8_t	node10_ofdma_tab_gap[4]	The inter-symbol time gap for each of the OFDMA tables.
uint32_t	node10_ofdma_tab_node_bitmask [4]	The node bitmask for each of the OFDMA tables.
uint8_t	node10_ofdma_tab_num_subchan [4]	The number of subchannels for each of the OFDMA tables.
uint16_t	node10_ofdmb_gcd	In a mixed mode network, this is the number of bits per OFDM symbol for the GCP PHY profile on the beacon channel, otherwise 0.
uint16_t	node10_ofdmb_nper[16]	If index is the node ID of a MoCA 1.x node this is the number of bits per OFDM symbol for the unicast profile. If index is the node ID of the entry node this is the

		number of bits per OFDM symbol for the NPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the number of bits per OFDM symbol for NPER unicast PHY Profile.
uint16_t	node10_ofdmb_vlper[16]	If index is the node ID of a MoCA 1.x node this is 0. If index is the node ID of the entry node this is the number of bits per OFDM symbol for the VLPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the number of bits per OFDM symbol for VLPER Unicast PHY Profile.
uint8_t	node11_gap_gcd	In a mixed mode network, this is the inter-symbol time gap for the GCP PHY profile on the beacon channel, otherwise 0.
uint8_t	node11_gap_nper[16]	If index is the node ID of a MoCA 1.x node this is the inter-symbol time gap for the unicast profile. If index is the node ID of the entry node this is the inter-symbol time gap for the NPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the inter-symbol time gap for NPER unicast PHY Profile.
uint8_t	node11_gap_vlper[16]	If index is the node ID of a MoCA 1.x node this is 0. If index is the node ID of the entry node this is the inter-symbol time gap for the VLPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the inter-symbol time gap for VLPER Unicast PHY Profile.
uint8_t	node11_ofdma_def_tab_num	Number of valid OFDMA definition tables with responding node as the receiver of the OFDMA frames.
uint16_t	node11_ofdma_tab_bps[4]	The number of bits per symbol of each OFDMA subchannel for each of the OFDMA tables.
uint8_t	node11_ofdma_tab_gap[4]	The inter-symbol time gap for each of the OFDMA tables.
uint32_t	node11_ofdma_tab_node_bitmask [4]	The node bitmask for each of the OFDMA tables.
uint8_t	node11_ofdma_tab_num_subchan [4]	The number of subchannels for each of the OFDMA tables.
uint16_t	node11_ofdmb_gcd	In a mixed mode network, this is the number of bits per OFDM symbol for the GCP PHY profile on the beacon channel, otherwise 0.
uint16_t	node11_ofdmb_nper[16]	If index is the node ID of a MoCA 1.x node this is the number of bits per OFDM symbol for the unicast profile.

		If index is the node ID of the entry node this is the number of bits per OFDM symbol for the NPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the number of bits per OFDM symbol for NPER unicast PHY Profile.
uint16_t	node11_ofdmb_vlper[16]	If index is the node ID of a MoCA 1.x node this is 0. If index is the node ID of the entry node this is the number of bits per OFDM symbol for the VLPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the number of bits per OFDM symbol for VLPER Unicast PHY Profile.
uint8_t	node12_gap_gcd	In a mixed mode network, this is the inter-symbol time gap for the GCP PHY profile on the beacon channel, otherwise 0.
uint8_t	node12_gap_nper[16]	If index is the node ID of a MoCA 1.x node this is the inter-symbol time gap for the unicast profile. If index is the node ID of the entry node this is the inter-symbol time gap for the NPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the inter-symbol time gap for NPER unicast PHY Profile.
uint8_t	node12_gap_vlper[16]	If index is the node ID of a MoCA 1.x node this is 0. If index is the node ID of the entry node this is the inter-symbol time gap for the VLPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the inter-symbol time gap for VLPER Unicast PHY Profile.
uint8_t	node12_ofdma_def_tab_num	Number of valid OFDMA definition tables with responding node as the receiver of the OFDMA frames.
uint16_t	node12_ofdma_tab_bps[4]	The number of bits per symbol of each OFDMA subchannel for each of the OFDMA tables.
uint8_t	node12_ofdma_tab_gap[4]	The inter-symbol time gap for each of the OFDMA tables.
uint32_t	node12_ofdma_tab_node_bitmask [4]	The node bitmask for each of the OFDMA tables.
uint8_t	node12_ofdma_tab_num_subchan [4]	The number of subchannels for each of the OFDMA tables.
uint16_t	node12_ofdmb_gcd	In a mixed mode network, this is the number of bits per OFDM symbol for the GCP PHY profile on the beacon channel, otherwise 0.
uint16_t	node12_ofdmb_nper[16]	If index is the node ID of a MoCA 1.x node this is the number of bits per OFDM symbol for the unicast

		profile. If index is the node ID of the entry node this is the number of bits per OFDM symbol for the NPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the number of bits per OFDM symbol for NPER unicast PHY Profile.
uint16_t	node12_ofdmb_vlper[16]	If index is the node ID of a MoCA 1.x node this is 0. If index is the node ID of the entry node this is the number of bits per OFDM symbol for the VLPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the number of bits per OFDM symbol for VLPER Unicast PHY Profile.
uint8_t	node13_gap_gcd	In a mixed mode network, this is the inter-symbol time gap for the GCP PHY profile on the beacon channel, otherwise 0.
uint8_t	node13_gap_nper[16]	If index is the node ID of a MoCA 1.x node this is the inter-symbol time gap for the unicast profile. If index is the node ID of the entry node this is the inter-symbol time gap for the NPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the inter-symbol time gap for NPER unicast PHY Profile.
uint8_t	node13_gap_vlper[16]	If index is the node ID of a MoCA 1.x node this is 0. If index is the node ID of the entry node this is the inter-symbol time gap for the VLPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the inter-symbol time gap for VLPER Unicast PHY Profile.
uint8_t	node13_ofdma_def_tab_num	Number of valid OFDMA definition tables with responding node as the receiver of the OFDMA frames.
uint16_t	node13_ofdma_tab_bps[4]	The number of bits per symbol of each OFDMA subchannel for each of the OFDMA tables.
uint8_t	node13_ofdma_tab_gap[4]	The inter-symbol time gap for each of the OFDMA tables.
uint32_t	node13_ofdma_tab_node_bitmask[4]	The node bitmask for each of the OFDMA tables.
uint8_t	node13_ofdma_tab_num_subchan[4]	The number of subchannels for each of the OFDMA tables.
uint16_t	node13_ofdmb_gcd	In a mixed mode network, this is the number of bits per OFDM symbol for the GCP PHY profile on the beacon channel, otherwise 0.
uint16_t	node13_ofdmb_nper[16]	

		If index is the node ID of a MoCA 1.x node this is the number of bits per OFDM symbol for the unicast profile. If index is the node ID of the entry node this is the number of bits per OFDM symbol for the NPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the number of bits per OFDM symbol for NPER unicast PHY Profile.
uint16_t	node13_ofdmb_vlper[16]	If index is the node ID of a MoCA 1.x node this is 0. If index is the node ID of the entry node this is the number of bits per OFDM symbol for the VLPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the number of bits per OFDM symbol for VLPER Unicast PHY Profile.
uint8_t	node14_gap_gcd	In a mixed mode network, this is the inter-symbol time gap for the GCP PHY profile on the beacon channel, otherwise 0.
uint8_t	node14_gap_nper[16]	If index is the node ID of a MoCA 1.x node this is the inter-symbol time gap for the unicast profile. If index is the node ID of the entry node this is the inter-symbol time gap for the NPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the inter-symbol time gap for NPER unicast PHY Profile.
uint8_t	node14_gap_vlper[16]	If index is the node ID of a MoCA 1.x node this is 0. If index is the node ID of the entry node this is the inter-symbol time gap for the VLPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the inter-symbol time gap for VLPER Unicast PHY Profile.
uint8_t	node14_ofdma_def_tab_num	Number of valid OFDMA definition tables with responding node as the receiver of the OFDMA frames.
uint16_t	node14_ofdma_tab_bps[4]	The number of bits per symbol of each OFDMA subchannel for each of the OFDMA tables.
uint8_t	node14_ofdma_tab_gap[4]	The inter-symbol time gap for each of the OFDMA tables.
uint32_t	node14_ofdma_tab_node_bitmask [4]	The node bitmask for each of the OFDMA tables.
uint8_t	node14_ofdma_tab_num_subchan [4]	The number of subchannels for each of the OFDMA tables.
uint16_t	node14_ofdmb_gcd	

		In a mixed mode network, this is the number of bits per OFDM symbol for the GCP PHY profile on the beacon channel, otherwise 0.
uint16_t	node14_ofdmb_nper[16]	If index is the node ID of a MoCA 1.x node this is the number of bits per OFDM symbol for the unicast profile. If index is the node ID of the entry node this is the number of bits per OFDM symbol for the NPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the number of bits per OFDM symbol for NPER unicast PHY Profile.
uint16_t	node14_ofdmb_vlper[16]	If index is the node ID of a MoCA 1.x node this is 0. If index is the node ID of the entry node this is the number of bits per OFDM symbol for the VLPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the number of bits per OFDM symbol for VLPER Unicast PHY Profile.
uint8_t	node15_gap_gcd	In a mixed mode network, this is the inter-symbol time gap for the GCP PHY profile on the beacon channel, otherwise 0.
uint8_t	node15_gap_nper[16]	If index is the node ID of a MoCA 1.x node this is the inter-symbol time gap for the unicast profile. If index is the node ID of the entry node this is the inter-symbol time gap for the NPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the inter-symbol time gap for NPER unicast PHY Profile.
uint8_t	node15_gap_vlper[16]	If index is the node ID of a MoCA 1.x node this is 0. If index is the node ID of the entry node this is the inter-symbol time gap for the VLPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the inter-symbol time gap for VLPER Unicast PHY Profile.
uint8_t	node15_ofdma_def_tab_num	Number of valid OFDMA definition tables with responding node as the receiver of the OFDMA frames.
uint16_t	node15_ofdma_tab_bps[4]	The number of bits per symbol of each OFDMA subchannel for each of the OFDMA tables.
uint8_t	node15_ofdma_tab_gap[4]	The inter-symbol time gap for each of the OFDMA tables.
uint32_t	node15_ofdma_tab_node_bitmask [4]	The node bitmask for each of the OFDMA tables.
uint8_t		

	node15_ofdma_tab_num_subchan [4]	The number of subchannels for each of the OFDMA tables.
uint16_t	node15_ofdmb_gcd	In a mixed mode network, this is the number of bits per OFDM symbol for the GCP PHY profile on the beacon channel, otherwise 0.
uint16_t	node15_ofdmb_nper[16]	If index is the node ID of a MoCA 1.x node this is the number of bits per OFDM symbol for the unicast profile. If index is the node ID of the entry node this is the number of bits per OFDM symbol for the NPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the number of bits per OFDM symbol for NPER unicast PHY Profile.
uint16_t	node15_ofdmb_vlper[16]	If index is the node ID of a MoCA 1.x node this is 0. If index is the node ID of the entry node this is the number of bits per OFDM symbol for the VLPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the number of bits per OFDM symbol for VLPER Unicast PHY Profile.
uint8_t	node1_gap_gcd	In a mixed mode network, this is the inter-symbol time gap for the GCP PHY profile on the beacon channel, otherwise 0.
uint8_t	node1_gap_nper[16]	If index is the node ID of a MoCA 1.x node this is the inter-symbol time gap for the unicast profile. If index is the node ID of the entry node this is the inter-symbol time gap for the NPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the inter-symbol time gap for NPER unicast PHY Profile.
uint8_t	node1_gap_vlper[16]	If index is the node ID of a MoCA 1.x node this is 0. If index is the node ID of the entry node this is the inter-symbol time gap for the VLPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the inter-symbol time gap for VLPER Unicast PHY Profile.
uint8_t	node1_ofdma_def_tab_num	Number of valid OFDMA definition tables with responding node as the receiver of the OFDMA frames.
uint16_t	node1_ofdma_tab_bps[4]	The number of bits per symbol of each OFDMA subchannel for each of the OFDMA tables.
uint8_t	node1_ofdma_tab_gap[4]	The inter-symbol time gap for each of the OFDMA tables.
uint32_t		The node bitmask for each of the OFDMA tables.

	node1_ofdma_tab_node_bitmask [4]	
uint8_t	node1_ofdma_tab_num_subchan [4]	The number of subchannels for each of the OFDMA tables.
uint16_t	node1_ofdmb_gcd	In a mixed mode network, this is the number of bits per OFDM symbol for the GCP PHY profile on the beacon channel, otherwise 0.
uint16_t	node1_ofdmb_nper[16]	If index is the node ID of a MoCA 1.x node this is the number of bits per OFDM symbol for the unicast profile. If index is the node ID of the entry node this is the number of bits per OFDM symbol for the NPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the number of bits per OFDM symbol for NPER unicast PHY Profile.
uint16_t	node1_ofdmb_vlper[16]	If index is the node ID of a MoCA 1.x node this is 0. If index is the node ID of the entry node this is the number of bits per OFDM symbol for the VLPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the number of bits per OFDM symbol for VLPER Unicast PHY Profile.
uint8_t	node2_gap_gcd	In a mixed mode network, this is the inter-symbol time gap for the GCP PHY profile on the beacon channel, otherwise 0.
uint8_t	node2_gap_nper[16]	If index is the node ID of a MoCA 1.x node this is the inter-symbol time gap for the unicast profile. If index is the node ID of the entry node this is the inter-symbol time gap for the NPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the inter-symbol time gap for NPER unicast PHY Profile.
uint8_t	node2_gap_vlper[16]	If index is the node ID of a MoCA 1.x node this is 0. If index is the node ID of the entry node this is the inter-symbol time gap for the VLPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the inter-symbol time gap for VLPER Unicast PHY Profile.
uint8_t	node2_ofdma_def_tab_num	Number of valid OFDMA definition tables with responding node as the receiver of the OFDMA frames.
uint16_t	node2_ofdma_tab_bps[4]	The number of bits per symbol of each OFDMA subchannel for each of the OFDMA tables.
uint8_t	node2_ofdma_tab_gap[4]	

		The inter-symbol time gap for each of the OFDMA tables.
uint32_t	node2_ofdma_tab_node_bitmask [4]	The node bitmask for each of the OFDMA tables.
uint8_t	node2_ofdma_tab_num_subchan [4]	The number of subchannels for each of the OFDMA tables.
uint16_t	node2_ofdmb_gcd	In a mixed mode network, this is the number of bits per OFDM symbol for the GCP PHY profile on the beacon channel, otherwise 0.
uint16_t	node2_ofdmb_nper[16]	If index is the node ID of a MoCA 1.x node this is the number of bits per OFDM symbol for the unicast profile. If index is the node ID of the entry node this is the number of bits per OFDM symbol for the NPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the number of bits per OFDM symbol for NPER unicast PHY Profile.
uint16_t	node2_ofdmb_vlper[16]	If index is the node ID of a MoCA 1.x node this is 0. If index is the node ID of the entry node this is the number of bits per OFDM symbol for the VLPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the number of bits per OFDM symbol for VLPER Unicast PHY Profile.
uint8_t	node3_gap_gcd	In a mixed mode network, this is the inter-symbol time gap for the GCP PHY profile on the beacon channel, otherwise 0.
uint8_t	node3_gap_nper[16]	If index is the node ID of a MoCA 1.x node this is the inter-symbol time gap for the unicast profile. If index is the node ID of the entry node this is the inter-symbol time gap for the NPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the inter-symbol time gap for NPER unicast PHY Profile.
uint8_t	node3_gap_vlper[16]	If index is the node ID of a MoCA 1.x node this is 0. If index is the node ID of the entry node this is the inter-symbol time gap for the VLPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the inter-symbol time gap for VLPER Unicast PHY Profile.
uint8_t	node3_ofdma_def_tab_num	Number of valid OFDMA definition tables with responding node as the receiver of the OFDMA frames.
uint16_t	node3_ofdma_tab_bps[4]	

		The number of bits per symbol of each OFDMA subchannel for each of the OFDMA tables.
uint8_t	node3_ofdma_tab_gap[4]	The inter-symbol time gap for each of the OFDMA tables.
uint32_t	node3_ofdma_tab_node_bitmask [4]	The node bitmask for each of the OFDMA tables.
uint8_t	node3_ofdma_tab_num_subchan [4]	The number of subchannels for each of the OFDMA tables.
uint16_t	node3_ofdmb_gcd	In a mixed mode network, this is the number of bits per OFDM symbol for the GCP PHY profile on the beacon channel, otherwise 0.
uint16_t	node3_ofdmb_nper[16]	If index is the node ID of a MoCA 1.x node this is the number of bits per OFDM symbol for the unicast profile. If index is the node ID of the entry node this is the number of bits per OFDM symbol for the NPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the number of bits per OFDM symbol for NPER unicast PHY Profile.
uint16_t	node3_ofdmb_vlper[16]	If index is the node ID of a MoCA 1.x node this is 0. If index is the node ID of the entry node this is the number of bits per OFDM symbol for the VLPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the number of bits per OFDM symbol for VLPER Unicast PHY Profile.
uint8_t	node4_gap_gcd	In a mixed mode network, this is the inter-symbol time gap for the GCP PHY profile on the beacon channel, otherwise 0.
uint8_t	node4_gap_nper[16]	If index is the node ID of a MoCA 1.x node this is the inter-symbol time gap for the unicast profile. If index is the node ID of the entry node this is the inter-symbol time gap for the NPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the inter-symbol time gap for NPER unicast PHY Profile.
uint8_t	node4_gap_vlper[16]	If index is the node ID of a MoCA 1.x node this is 0. If index is the node ID of the entry node this is the inter-symbol time gap for the VLPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the inter-symbol time gap for VLPER Unicast PHY Profile.
uint8_t	node4_ofdma_def_tab_num	

		Number of valid OFDMA definition tables with responding node as the receiver of the OFDMA frames.
uint16_t	node4_ofdma_tab_bps[4]	The number of bits per symbol of each OFDMA subchannel for each of the OFDMA tables.
uint8_t	node4_ofdma_tab_gap[4]	The inter-symbol time gap for each of the OFDMA tables.
uint32_t	node4_ofdma_tab_node_bitmask [4]	The node bitmask for each of the OFDMA tables.
uint8_t	node4_ofdma_tab_num_subchan [4]	The number of subchannels for each of the OFDMA tables.
uint16_t	node4_ofdmb_gcd	In a mixed mode network, this is the number of bits per OFDM symbol for the GCP PHY profile on the beacon channel, otherwise 0.
uint16_t	node4_ofdmb_nper[16]	If index is the node ID of a MoCA 1.x node this is the number of bits per OFDM symbol for the unicast profile. If index is the node ID of the entry node this is the number of bits per OFDM symbol for the NPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the number of bits per OFDM symbol for NPER unicast PHY Profile.
uint16_t	node4_ofdmb_vlper[16]	If index is the node ID of a MoCA 1.x node this is 0. If index is the node ID of the entry node this is the number of bits per OFDM symbol for the VLPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the number of bits per OFDM symbol for VLPER Unicast PHY Profile.
uint8_t	node5_gap_gcd	In a mixed mode network, this is the inter-symbol time gap for the GCP PHY profile on the beacon channel, otherwise 0.
uint8_t	node5_gap_nper[16]	If index is the node ID of a MoCA 1.x node this is the inter-symbol time gap for the unicast profile. If index is the node ID of the entry node this is the inter-symbol time gap for the NPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the inter-symbol time gap for NPER unicast PHY Profile.
uint8_t	node5_gap_vlper[16]	If index is the node ID of a MoCA 1.x node this is 0. If index is the node ID of the entry node this is the inter-symbol time gap for the VLPER GCD PHY Profile.

		If index is the node ID of a MoCA 2.0 node this is the inter-symbol time gap for VLPER Unicast PHY Profile.
uint8_t	node5_ofdma_def_tab_num	Number of valid OFDMA definition tables with responding node as the receiver of the OFDMA frames.
uint16_t	node5_ofdma_tab_bps[4]	The number of bits per symbol of each OFDMA subchannel for each of the OFDMA tables.
uint8_t	node5_ofdma_tab_gap[4]	The inter-symbol time gap for each of the OFDMA tables.
uint32_t	node5_ofdma_tab_node_bitmask [4]	The node bitmask for each of the OFDMA tables.
uint8_t	node5_ofdma_tab_num_subchan [4]	The number of subchannels for each of the OFDMA tables.
uint16_t	node5_ofdmb_gcd	In a mixed mode network, this is the number of bits per OFDM symbol for the GCP PHY profile on the beacon channel, otherwise 0.
uint16_t	node5_ofdmb_nper[16]	If index is the node ID of a MoCA 1.x node this is the number of bits per OFDM symbol for the unicast profile. If index is the node ID of the entry node this is the number of bits per OFDM symbol for the NPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the number of bits per OFDM symbol for NPER unicast PHY Profile.
uint16_t	node5_ofdmb_vlper[16]	If index is the node ID of a MoCA 1.x node this is 0. If index is the node ID of the entry node this is the number of bits per OFDM symbol for the VLPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the number of bits per OFDM symbol for VLPER Unicast PHY Profile.
uint8_t	node6_gap_gcd	In a mixed mode network, this is the inter-symbol time gap for the GCP PHY profile on the beacon channel, otherwise 0.
uint8_t	node6_gap_nper[16]	If index is the node ID of a MoCA 1.x node this is the inter-symbol time gap for the unicast profile. If index is the node ID of the entry node this is the inter-symbol time gap for the NPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the inter-symbol time gap for NPER unicast PHY Profile.
uint8_t	node6_gap_vlper[16]	If index is the node ID of a MoCA 1.x node this is 0. If index is the node ID of the entry node this is the inter-symbol time gap for the VLPER GCD PHY

		Profile. If index is the node ID of a MoCA 2.0 node this is the inter-symbol time gap for VLPER Unicast PHY Profile.
uint8_t	node6_ofdma_def_tab_num	Number of valid OFDMA definition tables with responding node as the receiver of the OFDMA frames.
uint16_t	node6_ofdma_tab_bps[4]	The number of bits per symbol of each OFDMA subchannel for each of the OFDMA tables.
uint8_t	node6_ofdma_tab_gap[4]	The inter-symbol time gap for each of the OFDMA tables.
uint32_t	node6_ofdma_tab_node_bitmask [4]	The node bitmask for each of the OFDMA tables.
uint8_t	node6_ofdma_tab_num_subchan [4]	The number of subchannels for each of the OFDMA tables.
uint16_t	node6_ofdmb_gcd	In a mixed mode network, this is the number of bits per OFDM symbol for the GCP PHY profile on the beacon channel, otherwise 0.
uint16_t	node6_ofdmb_nper[16]	If index is the node ID of a MoCA 1.x node this is the number of bits per OFDM symbol for the unicast profile. If index is the node ID of the entry node this is the number of bits per OFDM symbol for the NPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the number of bits per OFDM symbol for NPER unicast PHY Profile.
uint16_t	node6_ofdmb_vlper[16]	If index is the node ID of a MoCA 1.x node this is 0. If index is the node ID of the entry node this is the number of bits per OFDM symbol for the VLPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the number of bits per OFDM symbol for VLPER Unicast PHY Profile.
uint8_t	node7_gap_gcd	In a mixed mode network, this is the inter-symbol time gap for the GCP PHY profile on the beacon channel, otherwise 0.
uint8_t	node7_gap_nper[16]	If index is the node ID of a MoCA 1.x node this is the inter-symbol time gap for the unicast profile. If index is the node ID of the entry node this is the inter-symbol time gap for the NPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the inter-symbol time gap for NPER unicast PHY Profile.
uint8_t	node7_gap_vlper[16]	If index is the node ID of a MoCA 1.x node this is 0. If index is the node ID of the entry node this is the

		inter-symbol time gap for the VLPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the inter-symbol time gap for VLPER Unicast PHY Profile.
uint8_t	node7_ofdma_def_tab_num	Number of valid OFDMA definition tables with responding node as the receiver of the OFDMA frames.
uint16_t	node7_ofdma_tab_bps[4]	The number of bits per symbol of each OFDMA subchannel for each of the OFDMA tables.
uint8_t	node7_ofdma_tab_gap[4]	The inter-symbol time gap for each of the OFDMA tables.
uint32_t	node7_ofdma_tab_node_bitmask [4]	The node bitmask for each of the OFDMA tables.
uint8_t	node7_ofdma_tab_num_subchan [4]	The number of subchannels for each of the OFDMA tables.
uint16_t	node7_ofdmb_gcd	In a mixed mode network, this is the number of bits per OFDM symbol for the GCP PHY profile on the beacon channel, otherwise 0.
uint16_t	node7_ofdmb_nper[16]	If index is the node ID of a MoCA 1.x node this is the number of bits per OFDM symbol for the unicast profile. If index is the node ID of the entry node this is the number of bits per OFDM symbol for the NPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the number of bits per OFDM symbol for NPER unicast PHY Profile.
uint16_t	node7_ofdmb_vlper[16]	If index is the node ID of a MoCA 1.x node this is 0. If index is the node ID of the entry node this is the number of bits per OFDM symbol for the VLPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the number of bits per OFDM symbol for VLPER Unicast PHY Profile.
uint8_t	node8_gap_gcd	In a mixed mode network, this is the inter-symbol time gap for the GCP PHY profile on the beacon channel, otherwise 0.
uint8_t	node8_gap_nper[16]	If index is the node ID of a MoCA 1.x node this is the inter-symbol time gap for the unicast profile. If index is the node ID of the entry node this is the inter-symbol time gap for the NPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the inter-symbol time gap for NPER unicast PHY Profile.
uint8_t	node8_gap_vlper[16]	

		If index is the node ID of a MoCA 1.x node this is 0. If index is the node ID of the entry node this is the inter-symbol time gap for the VLPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the inter-symbol time gap for VLPER Unicast PHY Profile.
uint8_t	node8_ofdma_def_tab_num	Number of valid OFDMA definition tables with responding node as the receiver of the OFDMA frames.
uint16_t	node8_ofdma_tab_bps[4]	The number of bits per symbol of each OFDMA subchannel for each of the OFDMA tables.
uint8_t	node8_ofdma_tab_gap[4]	The inter-symbol time gap for each of the OFDMA tables.
uint32_t	node8_ofdma_tab_node_bitmask [4]	The node bitmask for each of the OFDMA tables.
uint8_t	node8_ofdma_tab_num_subchan [4]	The number of subchannels for each of the OFDMA tables.
uint16_t	node8_ofdmb_gcd	In a mixed mode network, this is the number of bits per OFDM symbol for the GCP PHY profile on the beacon channel, otherwise 0.
uint16_t	node8_ofdmb_nper[16]	If index is the node ID of a MoCA 1.x node this is the number of bits per OFDM symbol for the unicast profile. If index is the node ID of the entry node this is the number of bits per OFDM symbol for the NPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the number of bits per OFDM symbol for NPER unicast PHY Profile.
uint16_t	node8_ofdmb_vlper[16]	If index is the node ID of a MoCA 1.x node this is 0. If index is the node ID of the entry node this is the number of bits per OFDM symbol for the VLPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the number of bits per OFDM symbol for VLPER Unicast PHY Profile.
uint8_t	node9_gap_gcd	In a mixed mode network, this is the inter-symbol time gap for the GCP PHY profile on the beacon channel, otherwise 0.
uint8_t	node9_gap_nper[16]	If index is the node ID of a MoCA 1.x node this is the inter-symbol time gap for the unicast profile. If index is the node ID of the entry node this is the inter-symbol time gap for the NPER GCD PHY Profile.

		If index is the node ID of a MoCA 2.0 node this is the inter-symbol time gap for NPER unicast PHY Profile.
uint8_t	node9_gap_vlper[16]	If index is the node ID of a MoCA 1.x node this is 0. If index is the node ID of the entry node this is the inter-symbol time gap for the VLPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the inter-symbol time gap for VLPER Unicast PHY Profile.
uint8_t	node9_ofdma_def_tab_num	Number of valid OFDMA definition tables with responding node as the receiver of the OFDMA frames.
uint16_t	node9_ofdma_tab_bps[4]	The number of bits per symbol of each OFDMA subchannel for each of the OFDMA tables.
uint8_t	node9_ofdma_tab_gap[4]	The inter-symbol time gap for each of the OFDMA tables.
uint32_t	node9_ofdma_tab_node_bitmask [4]	The node bitmask for each of the OFDMA tables.
uint8_t	node9_ofdma_tab_num_subchan [4]	The number of subchannels for each of the OFDMA tables.
uint16_t	node9_ofdmb_gcd	In a mixed mode network, this is the number of bits per OFDM symbol for the GCP PHY profile on the beacon channel, otherwise 0.
uint16_t	node9_ofdmb_nper[16]	If index is the node ID of a MoCA 1.x node this is the number of bits per OFDM symbol for the unicast profile. If index is the node ID of the entry node this is the number of bits per OFDM symbol for the NPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the number of bits per OFDM symbol for NPER unicast PHY Profile.
uint16_t	node9_ofdmb_vlper[16]	If index is the node ID of a MoCA 1.x node this is 0. If index is the node ID of the entry node this is the number of bits per OFDM symbol for the VLPER GCD PHY Profile. If index is the node ID of a MoCA 2.0 node this is the number of bits per OFDM symbol for VLPER Unicast PHY Profile.
uint32_t	responsecode	

struct moca_fmr_init_out

Fields:

uint16_t	fmrinfo_node_0[16]	

		The FMR Information for node ID responded_node_0 transmitting to each other node on the network. OFDMb info stored in bits 0-10, GAP info in bits 11-15.
uint16_t	fmrinfo_node_10 [16]	
uint16_t	fmrinfo_node_11 [16]	
uint16_t	fmrinfo_node_12 [16]	
uint16_t	fmrinfo_node_13 [16]	
uint16_t	fmrinfo_node_14 [16]	
uint16_t	fmrinfo_node_15 [16]	
uint16_t	fmrinfo_node_1[16]	
uint16_t	fmrinfo_node_2[16]	
uint16_t	fmrinfo_node_3[16]	
uint16_t	fmrinfo_node_4[16]	
uint16_t	fmrinfo_node_5[16]	
uint16_t	fmrinfo_node_6[16]	
uint16_t	fmrinfo_node_7[16]	
uint16_t	fmrinfo_node_8[16]	
uint16_t	fmrinfo_node_9[16]	
uint32_t	responded_node_0	Node ID for information in fmrinfo_node_0[].
uint32_t	responded_node_1	
uint32_t	responded_node_10	
uint32_t	responded_node_11	
uint32_t	responded_node_12	
uint32_t	responded_node_13	
uint32_t	responded_node_14	
uint32_t	responded_node_15	
uint32_t	responded_node_2	
uint32_t	responded_node_3	
uint32_t	responded_node_4	
uint32_t	responded_node_5	
uint32_t	responded_node_6	
uint32_t	responded_node_7	
uint32_t	responded_node_8	
uint32_t	responded_node_9	
uint32_t	responsecode	

struct moca_gen_node_ext_status

Fields:

uint32_t	avg_snr	A dB measure of the Signal to Noise Ratio (SNR) based on the Type 1 probe from per node.
uint32_t	bit_loading [64]	512 sub carriers. Each sub carrier has a constellation value between 2 to 8 in a 4 bits field. Total is 256 bytes, i.e. 64 words.
uint32_t	cp	*Minimum:* 10 *Maximum:* 128
uint32_t	nbas	The total bits per one ACMT symbol *Minimum:* 224 *Maximum:* 4480
uint32_t	phy_rate	The PHY rate in Mbps.
uint32_t	preamble_type	*Minimum:* 0 *Maximum:* 10
int32_t	rx_backoff	Receive power Adjusment. Relevant only for RX
int32_t	rx_power	Receive power level. Relevant only for RX
uint32_t	turbo_status	Indicated whether the Turbo enabled or disabled.
int32_t	tx_backoff	Transmit power Adjusment. Relevant only for TX
uint32_t	tx_power	Transmit power level. Relevant only for TX

struct moca_gen_node_ext_status_in

Fields:

uint32_t	index	Node ID of the destination node *Minimum:* 0 *Maximum:* 15
uint32_t	profile_type	The profile type of which the corresponding table is to be retrieved. MoCA 2.0 profiles start with profile_type 6. *Minimum:* 0 *Maximum:* 21

struct moca_gen_node_status

Fields:

uint32_t	active_moca_version	Indicates whether the node is MoCA 1.x or MoCA 2.0 node
uint32_t	ae_number	The maximum number of allocation elements per MAP that this node can support. The value for this setting was obtained from a Device Discovery transaction. Set to 0 if no Device Discovery transaction has occurred.
macaddr_t	eui	The Node's MAC Address.
int32_t	freq_offset	

		The frequency offset of the node. This parameter is signed integer and can get negative values.
uint32_t	max_aggr_kb	The maximum number of kB this node can receive in one aggregated transmission. The value for this setting was obtained from a Device Discovery transaction. Set to 0 if no Device Discovery transaction has occurred.
uint32_t	max_aggr_pdus	The maximum number of PDUs that this node can receive in one aggregated transmission. The value for this setting was obtained from a Device Discovery transaction. Set to 0 if no Device Discovery transaction has occurred.
uint32_t	max_egress_pqos	The maximum number of egress PQOS flows that this node can support. The value for this setting was obtained from a Device Discovery transaction. Set to 0 if no Device Discovery transaction has occurred.
uint32_t	max_ingress_pqos	The maximum number of ingress PQOS flows that this node can support. The value for this setting was obtained from a Device Discovery transaction. Set to 0 if no Device Discovery transaction has occurred.
uint32_t	node_tx_backoff	The unicast back off value of the other node transmission to this node *Minimum:* 0 *Maximum:* 35
uint32_t	protocol_support	The protocol support field that the other node published during Admission
uint16_t	reserved_0	

struct moca_lmo_info

Fields:

uint32_t	is_lmo_success	
uint32_t	lmo_anb	The active node bitmask of the LMO
uint32_t	lmo_duration_dsec	The duration of the LMO in deciseconds
uint32_t	lmo_initial_ls	
uint32_t	lmo_node_id	

struct moca_moca_reset_in

Fields:

uint32_t	node_mask	To include node with ID 'x' in the request, set bit (1 << 'x'). For example, set node_mask to 0x8D to include node IDs 0, 2, 3 and 7.

uint32_t	non_def_seq_num	The non-default sequence number to use in the MR submit message. The default value of 0x10000 indicates that the node will use its internal sequence value for the transaction (see mr_seq_num). *Default:* 0x10000 *Minimum:* 0 *Maximum:* 0x10000
uint32_t	reset_timer	*Default:* 1 *Maximum:* 255

struct moca_moca_reset_out

Fields:

uint8_t	n00ResetStatus
uint8_t	n00RspCode
uint8_t	n01ResetStatus
uint8_t	n01RspCode
uint8_t	n02ResetStatus
uint8_t	n02RspCode
uint8_t	n03ResetStatus
uint8_t	n03RspCode
uint8_t	n04ResetStatus
uint8_t	n04RspCode
uint8_t	n05ResetStatus
uint8_t	n05RspCode
uint8_t	n06ResetStatus
uint8_t	n06RspCode
uint8_t	n07ResetStatus
uint8_t	n07RspCode
uint8_t	n08ResetStatus
uint8_t	n08RspCode
uint8_t	n09ResetStatus
uint8_t	n09RspCode
uint8_t	n10ResetStatus
uint8_t	n10RspCode
uint8_t	n11ResetStatus
uint8_t	n11RspCode
uint8_t	n12ResetStatus
uint8_t	n12RspCode
uint8_t	n13ResetStatus
uint8_t	n13RspCode
uint8_t	n14ResetStatus

uint8_t	n14RspCode
uint8_t	n15ResetStatus
uint8_t	n15RspCode
uint32_t	non_def_seq_num
uint32_t	reset_status
uint32_t	response_code

struct moca_moca_reset_request

Fields:

uint32_t	cause
uint32_t	mr_seq_num

struct moca_network_status

Fields:

uint32_t	backup_nc_id	The Node ID of the selected Back-up NC. (GCAP.5) *Minimum:* 0 *Maximum:* 15
uint32_t	bonded_nodes_bitmask	Bitmask of nodes on bonded channel
uint32_t	bw_status	BW status with the other nodes in the Network
uint32_t	connected_nodes	16_bit bitmask indicating the Node ID of the MoCA nodes active on the MoCA network, including nodes in admission process. The bit position within the bitmask is a direct mapping of the node ID of the active node: bit[15..0] = Node_ID[15..0] (GCAP.13)
uint32_t	nc_node_id	Indicate the node ID of node currently selected as the Network Coordinator (NC) of the network. (GCAP.4) *Minimum:* 0 *Maximum:* 15
uint32_t	network_moca_version	Beacon MOCA_VERSION (GCAP.26)
uint32_t	network_taboo_mask	This is the Beacon Taboo mask according to spec. e.g. 32 is corresponding to 800MHz center frequency. (GCAP.09)
uint32_t	network_taboo_start	This is the Beacon Taboo start according to spec. According to the MoCA spec, this field is 24 bits only. Only the 24 lsb are relevant. i.e 0x00HHHHHH where H represents any Hex number. (GCAP.09)
uint32_t	node_id	node ID of the current device *Minimum:* 0 *Maximum:* 15
uint32_t	nodes_usable_bitmask	The GCD_BITMASK field reported in Type I Probe Report of the recent LMO. This field bit mask represents the nodes that this

node communicates to in the MoCA network. The number of '1' may be smaller than the number of nodes reported by the NC node.

struct moca_node_stats

Fields:

uint32_t	primary_ch_rx_cw_corrected	Primary channel: Number of code word received with error and corrected
uint32_t	primary_ch_rx_cw_uncorrected	Primary channel: Number of code word received with error and uncorrected
uint32_t	primary_ch_rx_cw_unerror	Primary channel: Number of code word received without error
uint32_t	primary_ch_rx_no_sync	Primary channel: Number of timeouts on receive bursts with a 'no sync' error, i.e. there was a signal but PHY couldn't sync on it. Not implemented for 68xx chips.
uint32_t	rx_packets	Number of correct data packets received from the Node
uint32_t	secondary_ch_rx_cw_corrected	Secondary channel: Number of code word received with error and corrected
uint32_t	secondary_ch_rx_cw_uncorrected	Secondary channel: Number of code word received with error and uncorrected
uint32_t	secondary_ch_rx_cw_unerror	Secondary channel: Number of code word received without error
uint32_t	secondary_ch_rx_no_sync	Secondary channel: Number of timeouts on receive bursts with a 'no sync' error, i.e. there was a signal but PHY couldn't sync on it. Not implemented for 68xx chips.
uint32_t	tx_packets	Number of data packets transmitted to the Node.

struct moca_node_stats_ext

Fields:

uint16_t	reserved_0	
uint16_t	rx_acf_crc_error	This is a counter of the ACF packets received from the node specified by 'index' that have a CRC error.
uint16_t	rx_bc_crc_error	This is a counter of the broadcast packets received from the node specified by 'index' that have a CRC error.
uint16_t	rx_bc_timeout_error	

		This is a counter of the broadcast packets from the node specified by 'index' that were lost due to timeout.
uint16_t	rx_beacon_crc_error	This is a counter of the beacons received from the node specified by 'index' that have a CRC error.
uint16_t	rx_beacon_timeout_error	This is a counter of the beacons from the node specified by 'index' that were lost due to timeout.
uint16_t	rx_broken_packet_error	This is a counter of broken fragmented packets on primary channel, received from the node specified by 'index'
uint16_t	rx_broken_packet_error_sec_ch	This is a counter of broken fragmented packets on secondary channel, received from the node specified by 'index'
uint16_t	rx_lc_bc_crc_error	This is a counter of the broadcast link control packets received from the node specified by 'index' that have a CRC error.
uint16_t	rx_lc_bc_timeout_error	This is a counter of the broadcast link control packets from the node specified by 'index' that were lost due to timeout.
uint16_t	rx_lc_uc_crc_error	This is a counter of the unicast link control packets received from the node specified by 'index' that have a CRC error.
uint16_t	rx_lc_uc_timeout_error	This is a counter of the unicast link control packets from the node specified by 'index' that were lost due to timeout.
uint16_t	rx_map_crc_error	This is a counter of the map packets received from the node specified by 'index' that have a CRC error.
uint16_t	rx_map_timeout_error	This is a counter of the map packets from the node specified by 'index' that were lost due to timeout.
uint16_t	rx_ofdma_rr_crc_error	This is a counter of the OFDMA RR packets received from the node specified by 'index' that have a CRC error.
uint16_t	rx_plp_crc_error	This is a counter of the periodic link packets received from the node specified by 'index' that have a CRC error.
uint16_t	rx_plp_timeout_error	This is a counter of the periodic link packets received from the node specified by 'index' that were lost due to timeout.
uint16_t	rx_probe1_error	This is a counter of the probe I packets received from the node specified by 'index' that have a CRC error on primary channel
uint16_t	rx_probe1_error_sec_ch	This is a counter of the probe I packets received from the node specified by 'index' that have a CRC error on secondary channel

uint16_t	rx_probe1_gcd_error	This is a counter of the probe I GCD packets received from the node specified by 'index' that have a CRC error.
uint16_t	rx_probe2_error	This is a counter of the probe II packets received from the node specified by 'index' that have a CRC error.
uint16_t	rx_probe3_error	This is a counter of the probe III packets received from the node specified by 'index' that have a CRC error.
uint16_t	rx_rr_crc_error	This is a counter of the RR packets received from the node specified by 'index' that have a CRC error.
uint16_t	rx_rr_timeout_error	This is a counter of the RR packets from the node specified by 'index' that were lost due to timeout.
uint16_t	rx_uc_crc_error	This is a counter of the unicast packets received from the node specified by 'index' that have a CRC error on primary channel.
uint16_t	rx_uc_crc_error_sec_ch	This is a counter of the unicast packets received from the node specified by 'index' that have a CRC error on secondary channel when bonded mode is active.
uint16_t	rx_uc_timeout_error	This is a counter of the unicast packets from the node specified by 'index' that were lost due to timeout on primary channel.
uint16_t	rx_uc_timeout_error_sec_ch	This is a counter of the unicast packets from the node specified by 'index' that were lost due to timeout on secondary channel when bonded mode is active.

struct moca_node_stats_ext_in

Fields:

uint32_t	index	Node ID of the destination node *Minimum:* 0 *Maximum:* 15
uint32_t	reset_stats	Reset the statistics following the read. *Default:* 0

struct moca_node_stats_in

Fields:

uint32_t	index	Node ID of the destination node *Minimum:* 0 *Maximum:* 15
uint32_t	reset_stats	Reset the statistics following the read. *Default:* 0

struct moca_ofdma_assignment_table

Fields:

uint32_t	cp_length[4]
uint32_t	node_bitmask[4]
uint32_t	num_Subchannels[4]
uint32_t	ofdmaDefTabNum
uint32_t	ofdmaFrameId[4]
uint32_t	subchannelDefId[4]

struct moca_ofdma_definition_table

Fields:

uint32_t	node_bitmask[4]
uint32_t	ofdmaDefTabNum
uint32_t	subchannelDefId[4]
uint32_t	subchannel_NBAS[4]

struct moca_rxd_lmo_request

Fields:

uint32_t	channel_id	Primary (0), secondary (1) or both (2) channels *Default:* 0 *Minimum:* 0 *Maximum:* 2
uint32_t	node_id	node ID of the current device *Minimum:* 0 *Maximum:* 15
uint32_t	probe_id	Predefined probe configuration *Minimum:* 1 *Maximum:* 2

struct moca_start_ulmo

Fields:

uint32_t	node_id	The ID of the node to send the unsolicited LMO to. *Minimum:* 0 *Maximum:* 0x3F
uint32_t	ofdma_node_mask	The bitmask of nodes to send the unsolicited OFDMA LMO to. *Minimum:* 0 *Maximum:* 0xFFFF
uint32_t	report_type	The unsolicited LMO type (GCD,UC,OFDMA). *Minimum:* 0

		Maximum: 2
uint32_t	subcarrier[16]	Bitmask of the valid 0 - 511 sub-carriers E.g. subcarrier[0] corresponds to SCs 0-31 Setting subcarrier[0] to 0x8000000F will enable SCs 0,28-31 Setting subcarrier[15] to 0x00000001 will enable SC 511

struct moca_taboo_channels

Fields:

uint32_t	taboo_fixed_channel_mask	Ability to change 'fixed' Taboo Frequency Mask (GCAP.9) The MSB corresponds to the lowest Taboo channel as identified in the TABOO_MASK_START field. Each consecutive bit then corresponds to channels offset by multiples of 25MHz. A bit in the TABOO_CHANNEL_MASK field is set to '1' if the corresponding channel is Taboo. This is the 'fixed' taboo, i.e. The node will publish this taboo irrelevant to the LOF value. *Default:* 0x0
uint32_t	taboo_fixed_mask_start	RF channel number of the lowest frequency channel covered by the Taboo Channel Mask field. - This is the 'fixed' taboo, i.e. The node will publish this taboo irrelevant to the LOF value. *Default:* 0
uint32_t	taboo_left_mask	Left side mask for adjacent channels taboo, relative to the LOF. *Default:* 0x00ffffff
uint32_t	taboo_right_mask	Right side mask for adjacent channels taboo, relative to the LOF *Default:* 0xffffff00

Functions

moca_get_taboo_channels

Prototype:

```
MOCALIB_GEN_GET_FUNCTION int moca_get_taboo_channels(void *vctx, struct
moca_taboo_channels *out)
```

Description:

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.

moca_set_taboo_channels

Prototype:

```
MOCALIB_GEN_SET_FUNCTION int moca_set_taboo_channels(void *vctx, const struct
moca_taboo_channels *in)
```

Description:

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. This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

moca_get_gen_node_status

Prototype:

```
MOCALIB_GEN_GET_FUNCTION int moca_get_gen_node_status(void *vctx, uint32_t index,
struct moca_gen_node_status *out)
```

Description:

Nodes Status Parameters

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

Parameters:

index

Node ID of the destination node

Minimum: 0

Maximum: 15

moca_get_gen_node_ext_status

Prototype:

```
MOCALIB_GEN_GET_FUNCTION int moca_get_gen_node_ext_status(void *vctx, struct
moca_gen_node_ext_status_in *in, struct moca_gen_node_ext_status *out)
```

Description:

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.

moca_get_node_stats
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_node_stats(void *vctx, struct
moca_node_stats_in *in, struct moca_node_stats *out)
```

Description:
Nodes Statistics

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

moca_get_node_stats_ext
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_node_stats_ext(void *vctx, struct
moca_node_stats_ext_in *in, struct moca_node_stats_ext *out)
```

Description:
Nodes Extended Statistics

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

moca_get_network_status
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_network_status(void *vctx, struct
moca_network_status *out)
```

Description:
Retrieve status information about the MoCA network.

moca_set_ooo_lmo
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_ooo_lmo(void *vctx, uint32_t node_id)
```

Description:
A Request for an Out-of-Order LMO to any node (GCAP.27)

Parameters:
node_id
Minimum: 0
Maximum: 15

moca_get_start_ulmo
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_start_ulmo(void *vctx, struct
moca_start_ulmo *out)
```

Description:
A Request for an unsolicited LMO to any node.

moca_set_start_ulmo
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_start_ulmo(void *vctx, const struct
moca_start_ulmo *in)
```

Description:
A Request for an unsolicited LMO to any node.

moca_set_rxd_lmo_request
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_rxd_lmo_request(void *vctx, const struct
moca_rxd_lmo_request *in)
```

Description:
A Request for a Receiver-Determined Probe LMO (GCAP.119)

moca_get_ofdma_definition_table
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_ofdma_definition_table(void *vctx, struct
moca_ofdma_definition_table *out)
```

Description:
Display selected values from OFDMA Subchannel Definition Table (GCAP.130)

moca_get_ofdma_assignment_table
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_ofdma_assignment_table(void *vctx, struct
moca_ofdma_assignment_table *out)
```

Description:
Display selected values from OFDMA Subchannel Assignment Table (GCAP.131)

moca_register_admission_status_cb
Prototype:
```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_admission_status_cb(void *vctx,
void (*callback)(void *userarg, uint32_t status), void *userarg)
```

moca_register_limited_bw_cb
Prototype:
```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_limited_bw_cb(void *vctx, void
(*callback)(void *userarg, uint32_t bw_status), void *userarg)
```

moca_register_lmo_info_cb
Prototype:
```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_lmo_info_cb(void *vctx, void
(*callback)(void *userarg, struct moca_lmo_info *out), void *userarg)
```

moca_register_topology_changed_cb
Prototype:
```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_topology_changed_cb(void *vctx,
void (*callback)(void *userarg, uint32_t nodemask), void *userarg)
```

moca_register_moca_version_changed_cb
Prototype:
```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_moca_version_changed_cb(void
*vctx, void (*callback)(void *userarg, uint32_t new_version), void *userarg)
```

moca_register_moca_reset_request_cb
Prototype:
```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_moca_reset_request_cb(void *vctx,
void (*callback)(void *userarg, struct moca_moca_reset_request *out), void
*userarg)
```

moca_register_nc_id_changed_cb
Prototype:
```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_nc_id_changed_cb(void *vctx, void
```

```
(*callback)(void *userarg, uint32_t new_nc_id), void *userarg)
```

moca_register_mr_event_cb

Prototype:

```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_mr_event_cb(void *vctx, void
(*callback)(void *userarg, uint32_t status), void *userarg)
```

Description:

This is a MoCA Reset event used in the case where this node is the entry node for a MoCA Reset operation that attempts to reset all other nodes on the network.

moca_register_aca_cb

Prototype:

```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_aca_cb(void *vctx, void
(*callback)(void *userarg, struct moca_aca_out *out), void *userarg)
```

Description:

Perform an Alternate Channel Assessment operation.

moca_do_aca

Prototype:

```
MOCALIB_GEN_DO_FUNCTION int moca_do_aca(void *vctx, struct moca_aca_in *in, struct
moca_aca_out *out)
```

Description:

Perform an Alternate Channel Assessment operation.

moca_register_fmr_init_cb

Prototype:

```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_fmr_init_cb(void *vctx, void
(*callback)(void *userarg, struct moca_fmr_init_out *out), void *userarg)
```

Description:

A trigger for initiating a full mesh rate operation. The request can be for a specific node or for group of nodes by setting the appropriate bits in the node_mask field.

moca_do_fmr_init

Prototype:

```
MOCALIB_GEN_DO_FUNCTION int moca_do_fmr_init(void *vctx, uint32_t node_mask,
struct moca_fmr_init_out *out)
```

Description:

A trigger for initiating a full mesh rate operation. The request can be for a specific node or for group of nodes by setting the appropriate bits in the node_mask field.

moca_register_moca_reset_cb

Prototype:

```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_moca_reset_cb(void *vctx, void
(*callback)(void *userarg, struct moca_moca_reset_out *out), void *userarg)
```

Description:

Order a MoCA Reset operation on the MoCA network. Specify the nodes to be reset using the node_mask field.

moca_do_moca_reset

Prototype:

```
MOCALIB_GEN_DO_FUNCTION int moca_do_moca_reset(void *vctx, struct
moca_moca_reset_in *in, struct moca_moca_reset_out *out)
```

Description:

Order a MoCA Reset operation on the MoCA network. Specify the nodes to be reset using the node_mask field.

moca_register_dd_init_cb

Prototype:

```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_dd_init_cb(void *vctx, void
(*callback)(void *userarg, struct moca_dd_init_out *out), void *userarg)
```

Description:

A trigger for initiating a Device Discovery operation. The request can be for a specific node or for group of nodes by setting the appropriate bits in the node_mask field.
This operation is for MoCA 2.0 nodes only.
The output arrays are indexed by node ID. Nodes that are not included in the DD transaction or that don't respond to the DD transaction will have values of zero.

moca_do_dd_init

Prototype:

```
MOCALIB_GEN_DO_FUNCTION int moca_do_dd_init(void *vctx, uint32_t node_mask, struct
moca_dd_init_out *out)
```

Description:

A trigger for initiating a Device Discovery operation. The request can be for a specific node or for group of nodes by setting the appropriate bits in the node_mask field.
This operation is for MoCA 2.0 nodes only.
The output arrays are indexed by node ID. Nodes that are not included in the DD transaction or that don't respond to the DD transaction will have values of zero.

moca_register_fmr_20_cb

Prototype:

```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_fmr_20_cb(void *vctx, void
(*callback)(void *userarg, struct moca_fmr_20_out *out), void *userarg)
```

Description:

A trigger for initiating a MoCA 2.0 full mesh rate operation. The request can be for a specific node or for group of nodes by setting the appropriate bits in the node_mask field.

moca_do_fmr_20

Prototype:

```
MOCALIB_GEN_DO_FUNCTION int moca_do_fmr_20(void *vctx, uint32_t node_mask, struct
moca_fmr_20_out *out)
```

Description:

A trigger for initiating a MoCA 2.0 full mesh rate operation. The request can be for a specific node or for group of nodes by setting the appropriate bits in the node_mask field.

moca_get_error_stats

Prototype:

```
MOCALIB_GEN_GET_FUNCTION int moca_get_error_stats(void *vctx, struct
moca_error_stats *out)
```

Description:

Error Statistics

The following table is a sum of the node_stats_ext counters for each node on the MoCA network.

moca_register_hostless_mode_cb
Prototype:
```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_hostless_mode_cb(void *vctx, void
(*callback)(void *userarg, uint32_t status), void *userarg)
```

Description:
Put firmware into host-less mode of operation. Firmware will not send any traps to mocad, and mocad will disable the watchdog.

moca_do_hostless_mode
Prototype:
```
MOCALIB_GEN_DO_FUNCTION int moca_do_hostless_mode(void *vctx, uint32_t enable,
uint32_t *status)
```

Description:
Put firmware into host-less mode of operation. Firmware will not send any traps to mocad, and mocad will disable the watchdog.

moca_register_wakeup_node_cb
Prototype:
```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_wakeup_node_cb(void *vctx, void
(*callback)(void *userarg, uint32_t status), void *userarg)
```

Description:
Wake up a remote node (request that it change to M0)

moca_do_wakeup_node
Prototype:
```
MOCALIB_GEN_DO_FUNCTION int moca_do_wakeup_node(void *vctx, uint32_t node,
uint32_t *status)
```

Description:
Wake up a remote node (request that it change to M0)

INTFC Group
The Interface group of parameters provide statistics and status about the MoCA interface of this device.

Structures

struct moca_ext_octet_count
Fields:

uint32_t	in_octets_hi
uint32_t	in_octets_lo
uint32_t	out_octets_hi
uint32_t	out_octets_lo

struct moca_gen_stats
Fields:

uint32_t	aggr_pkt_stats_rx_count	
uint32_t	aggr_pkt_stats_rx_max	
uint32_t	aggr_pkt_stats_tx[30]	Statistics about the max TX aggregated packets (enhanced GCAP.24)
uint32_t	ecl_fc_bg	Counter of the number of times that flow control was enabled at the ECL for packets of Background level priority.
uint32_t	ecl_fc_bp_all	Counter of the number of times that flow control was enabled at the ECL for all packets of any level priority.
uint32_t	ecl_fc_high	Counter of the number of times that flow control was enabled at the ECL for packets of High level priority.
uint32_t	ecl_fc_low	Counter of the number of times that flow control was enabled at the ECL for packets of Low level priority.
uint32_t	ecl_fc_medium	Counter of the number of times that flow control was enabled at the ECL for packets of Medium level priority.
uint32_t	ecl_fc_pqos	Counter of the number of times that flow control was enabled at the ECL for packets of PQOS level priority.
uint32_t	ecl_rx_bcast_pkts	Counter of broadcast egress packets received from the MoCA network, to be sent to the ECL.
uint32_t	ecl_rx_mcast_filter_pkts	Counter of the number of multicast packets received from the MoCA network that were filtered at the ECL.
uint32_t	ecl_rx_mcast_pkts	Counter of multicast egress packets received from the MoCA network, to be sent to the ECL.
uint64_t	ecl_rx_total_bytes	Counter of total egress bytes of data received from the MoCA network, to be sent to the ECL.
uint32_t	ecl_rx_total_pkts	Counter of total egress packets received from the MoCA network, to be sent to the ECL.
uint32_t	ecl_rx_ucast_drops	Counter of unicast egress packets received from the MoCA network, dropped at the ECL.

uint32_t	ecl_rx_ucast_pkts	Counter of unicast egress packets received from the MoCA network, to be sent to the ECL.
uint32_t	ecl_tx_bcast_pkts	Counter of broadcast ingress packets received at the ECL, to be transmitted to the MoCA network.
uint32_t	ecl_tx_buff_drop_pkts	Counter of the number of packets destined for the MoCA network that were dropped at the ECL due to buffer overflow.
uint32_t	ecl_tx_error_drop_pkts	Counter of the number of packets destined for the MoCA network that were dropped at the ECL due to errors in the packets. E.g. FCS errors.
uint32_t	ecl_tx_mcast_drops	Counter of multicast ingress packets dropped at the ECL.
uint32_t	ecl_tx_mcast_pkts	Counter of multicast ingress packets received at the ECL, to be transmitted to the MoCA network.
uint32_t	ecl_tx_mcast_unknown	Counter of unknown multicast ingress packets at the ECL, to be transmitted to the MoCA network.
uint64_t	ecl_tx_total_bytes	Counter of total ingress bytes of data received at the ECL, to be transmitted to the MoCA network.
uint32_t	ecl_tx_total_pkts	Counter of total ingress packets received at the ECL, to be transmitted to the MoCA network.
uint32_t	ecl_tx_ucast_drops	Counter of unicast ingress packets dropped at the ECL.
uint32_t	ecl_tx_ucast_pkts	Counter of unicast ingress packets received at the ECL, to be transmitted to the MoCA network.
uint32_t	ecl_tx_ucast_unknown	Counter of unknown unicast ingress packets at the ECL, to be transmitted to the MoCA network.
uint32_t	link_down_count	Count of the number of times the MoCA link went down.
uint32_t	link_up_count	Count of the number of times the MoCA link went up.
uint32_t	mac_channel_usable_drop	Counter of the number of packets destined for the MoCA interface that are dropped because of low PHY rates.
uint32_t	mac_frag_mpdu_rx	Counter of fragmented MPDUs received from the MoCA Network
uint32_t	mac_frag_mpdu_tx	Counter of fragmented MPDUs transmitted to the MoCA Network
uint32_t	mac_loopback_drop_pkts	Counter of the number of MoCA loopback packets dropped.
uint32_t	mac_loopback_pkts	Counter of the number of MoCA loopback packets processed.
uint32_t	mac_pqos_policing_drop	Counter of PQOS MPDUs dropped due to policing on the MoCA Network
uint32_t	mac_pqos_policing_tx	

		Counter of PQOS MPDUs transmitted with policing enabled on the MoCA Network
uint32_t	mac_remove_node_drop	Counter of the number of packets destined for the MoCA interface that are dropped because the destination node was no longer on the network.
uint32_t	mac_rx_buff_drop_pkts	Counter of the number of packets from the MoCA interface that are dropped due to buffer overflow.
uint32_t	mac_tx_low_drop_pkts	Counter of number the of packets destined for the MoCA interface that are dropped due to priority.
uint32_t	nc_backup_counter	Counter of NC-Backups in the Network
uint32_t	nc_became_backup_nc_counter	Count of the number of times the node became NC-Backup
uint32_t	nc_became_nc_counter	Count of the number of times the node became NC
uint32_t	nc_handoff_counter	Counter of NC-Handoffs in the Network
uint32_t	resync_attempts_to_network	Counts the number of enterings into re-sync mode, i.e. loosing a MAP and catching on a Beacon.
uint32_t	rx_beacons	Counter of Beacons received from the MoCA Network
uint32_t	rx_buffer_full_counter	Count of the number of times the rx buffer became full
uint32_t	rx_control_bc_packets	Counter of broadcast Link Control packets received from the MoCA Network
uint32_t	rx_control_uc_packets	Counter of unicast Link Control packets received from the MoCA Network
uint32_t	rx_map_packets	Counter of MAP packets received from the MoCA Network
uint32_t	rx_ofdma_rr_packets	Counter of OFDMA RR packets received from the MoCA Network
uint32_t	rx_protocol_ie	Counter of Protocol IEs received from the MoCA Network
uint32_t	rx_rr_packets	Counter of RR packets received from the MoCA Network
uint32_t	tx_beacons	Counter of Beacons transmitted to the MoCA Network
uint32_t	tx_control_bc_packets	Counter of broadcast Link Control packets transmitted to the MoCA Network
uint32_t	tx_control_uc_packets	Counter of unicast Link Control packets transmitted to the MoCA Network
uint32_t	tx_map_packets	Counter of MAPs transmitted to the MoCA Network
uint32_t	tx_ofdma_rr_packets	Counter of OFDMA RR frames transmitted to the MoCA Network
uint32_t	tx_protocol_ie	Counter of Protocol IEs transmitted to the MoCA Network
uint32_t	tx_rr_packets	Counter of RR frames transmitted to the MoCA Network

struct moca_if_access_table
Fields:

macaddr_t	mac_addr[16]	The MAC addresses of the devices allowed to join the network.

struct moca_interface_status
Fields:

uint32_t	link_status	Indicates the link status of the MoCA node in the Network.
uint32_t	primary_channel	The MoCA 2.0 primary channel.
uint32_t	rf_channel	RF frequency to which the MoCA interface is currently tuned to. *Minimum:* 20 *Maximum:* 73
uint32_t	secondary_channel	The MoCA 2.0 secondary channel.

Functions

moca_get_rf_band
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_rf_band(void *vctx, uint32_t *val)
```

Description:
Defines one or multiple bands or sub-bands of operation of the Node among all the supported bands and sub-bands.

Parameters:
val

moca_set_rf_band
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_rf_band(void *vctx, uint32_t val)
```

Description:
Defines one or multiple bands or sub-bands of operation of the Node among all the supported bands and sub-bands. This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

Parameters:

val

moca_get_if_access_en
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_if_access_en(void *vctx, uint32_t *val)
```

Description:
Configures the firmware to use the if_access_table when deciding whether or not to admit nodes to the network. This setting will only have an effect when the self node is the NC. Nodes currently joined to the network will not be affected, only new nodes attempting to join the network will be affected.

Parameters:
val
Default: 0
Minimum: 0
Maximum: 1

moca_set_if_access_en
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_if_access_en(void *vctx, uint32_t val)
```

Description:
Configures the firmware to use the if_access_table when deciding whether or not to admit nodes to the network. This setting will only have an effect when the self node is the NC. Nodes currently joined to the network will not be affected, only new nodes attempting to join the network will be affected.

Parameters:
val
Default: 0
Minimum: 0
Maximum: 1

moca_get_led_mode
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_led_mode(void *vctx, uint32_t *val)
```

Description:
Configure the firmware to control the MoCA LED according to the rules described by the mode value.

Parameters:
val
Default: 0
Minimum: 0
Maximum:
1 (STANDALONE)
2

moca_set_led_mode
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_led_mode(void *vctx, uint32_t val)
```

Description:
Configure the firmware to control the MoCA LED according to the rules described by the mode value. This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

Parameters:
val
Default: 0
Minimum: 0
Maximum:
1 (STANDALONE)
2

moca_get_gen_stats
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_gen_stats(void *vctx, uint32_t reset_stats,
struct moca_gen_stats *out)
```

Description:
Retrieve statistics from the MoCA interface
ECL_INGR = received at the ECL layer from the Ethernet interface and destined for the MoCA RF interface
ECL_EGR = received at the ECL layer from the MoCA RF interface and destined for the Ethernet interface

IN = ingress = into the MoCA coax network (Switch -> MoCA core -> Coax)
OUT = Egress = out of the MoCA coax network (Coax -> MoCA core -> Switch)

Parameters:
reset_stats
Reset the statistics following the read.
Default: 0

moca_get_interface_status
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_interface_status(void *vctx, struct
moca_interface_status *out)
```

Description:
Retrieve general status information about the MoCA interface.

moca_get_if_access_table
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_if_access_table(void *vctx, struct
moca_if_access_table *out)
```

Description:
When if_access_en is enabled and this node is the NC, only nodes with MAC addresses that are listed in this table will be allowed to join the network.

moca_set_if_access_table
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_if_access_table(void *vctx, struct
moca_if_access_table *in)
```

Description:
When if_access_en is enabled and this node is the NC, only nodes with MAC addresses that are listed in this table will be allowed to join the network.

moca_register_link_up_state_cb
Prototype:
```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_link_up_state_cb(void *vctx, void
```

```
(*callback)(void *userarg, uint32_t status), void *userarg)
```

moca_get_ext_octet_count
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_ext_octet_count(void *vctx, struct
moca_ext_octet_count *out)
```

moca_set_reset_stats
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_reset_stats(void *vctx)
```

POWER_MGMT Group

The Power Management group of parameters allow the MoCA device to change power states.

Structures

struct moca_node_power_state
Fields:

uint32_t	pwr	
uint32_t	state	A list from the range M0-M3

struct moca_wom_ip
Fields:

uint32_t	index	Index of filter to modify. 0-4 *Minimum:* 0 *Maximum:* 4
uint32_t	ipaddr	IP address of packet to match

struct moca_wom_magic_mac
Fields:

macaddr_t	val	

struct moca_wom_pattern
Fields:

uint8_t		First 16 bytes of the packet

	bytes [16]	
uint8_t	mask [16]	Mask for each byte of the bitmask. Setting a bit to '1' will force the corresponding bit in 'bytes' to be ignored.

struct moca_wom_pattern_set
Fields:

uint8_t	bytes [16]	First 14 bytes of the packet
uint32_t	index	Index of filter to modify. 0-4 *Minimum:* 0 *Maximum:* 4
uint8_t	mask [16]	Mask for each byte of the bitmask. Setting a bit to '1' will force the corresponding bit in 'bytes' to be ignored. *Default:* 0xFF

Functions

moca_get_m1_tx_power_variation
Prototype:

```
MOCALIB_GEN_GET_FUNCTION int moca_get_m1_tx_power_variation(void *vctx, uint32_t
*state)
```

Description:
Set m1_tx_power_variation

Parameters:
state
Default: 0
Minimum: 0
Maximum: 6

moca_set_m1_tx_power_variation
Prototype:

```
MOCALIB_GEN_SET_FUNCTION int moca_set_m1_tx_power_variation(void *vctx, uint32_t
state)
```

Description:

Set m1_tx_power_variation

Parameters:
state
Default: 0
Minimum: 0
Maximum: 6

moca_get_nc_listening_interval
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_nc_listening_interval(void *vctx, uint32_t
*val)
```

Description:
NC listening interval, units of Beacon interval BSI

Parameters:
val
Default: 10
Minimum: 1
Maximum: 10

moca_set_nc_listening_interval
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_nc_listening_interval(void *vctx, uint32_t
val)
```

Description:
NC listening interval, units of Beacon interval BSI This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

Parameters:
val
Default: 10
Minimum: 1
Maximum: 10

moca_get_nc_heartbeat_interval

Prototype:

```
MOCALIB_GEN_GET_FUNCTION int moca_get_nc_heartbeat_interval(void *vctx, uint32_t
*val)
```

Description:

NC hearbeat interval, in seconds

Parameters:

val

Default: 10
Minimum: 1
Maximum: 255

moca_set_nc_heartbeat_interval

Prototype:

```
MOCALIB_GEN_SET_FUNCTION int moca_set_nc_heartbeat_interval(void *vctx, uint32_t
val)
```

Description:

NC hearbeat interval, in seconds This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

Parameters:

val

Default: 10
Minimum: 1
Maximum: 255

moca_get_wom_magic_enable

Prototype:

```
MOCALIB_GEN_GET_FUNCTION int moca_get_wom_magic_enable(void *vctx, uint32_t *val)
```

Description:

Enables magic-packet filtering for WoM

Parameters:

val

Default: 0
Minimum: 0
Maximum: 1

moca_set_wom_magic_enable
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_wom_magic_enable(void *vctx, uint32_t val)
```

Description:
Enables magic-packet filtering for WoM

Parameters:
val
Default: 0
Minimum: 0
Maximum: 1

moca_get_pm_restore_on_link_down
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_pm_restore_on_link_down(void *vctx, uint32_t
*val)
```

Description:
Resets power mode when link goes down and back up again

Parameters:
val
Default: 0
Minimum: 0
Maximum: 1

moca_set_pm_restore_on_link_down
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_pm_restore_on_link_down(void *vctx, uint32_t
val)
```

Description:
Resets power mode when link goes down and back up again

Parameters:
val

Default: 0
Minimum: 0
Maximum: 1

moca_get_power_state
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_power_state(void *vctx, uint32_t *state)
```

Description:
For GET operations, reports current power state.
To SET the power state, use the 'do ps_cmd' operation.

Refer to power_state_capabilities to learn the states that may be transitioned to from the current state.

Parameters:
state
A list from the range M0-M3
Minimum: 0
Maximum: 3

moca_get_hostless_mode_request
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_hostless_mode_request(void *vctx, uint32_t
*enable)
```

Description:
Enable midRF Power Saving State. In this mode, MoCA will not transmit any data traffic.

Parameters:
enable

moca_set_hostless_mode_request
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_hostless_mode_request(void *vctx, uint32_t
enable)
```

Description:

Enable midRF Power Saving State. In this mode, MoCA will not transmit any data traffic.

Parameters:
enable

moca_set_wakeup_node_request
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_wakeup_node_request(void *vctx, uint32_t
node)
```

Description:
Request that a node mode to M0

Parameters:
node

moca_get_node_power_state
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_node_power_state(void *vctx, uint32_t node,
struct moca_node_power_state *out)
```

Description:
Get power state and m1_tx_power_variation (GCAP.124)

Parameters:
node
Node ID to report

moca_get_filter_m2_data_wakeUp
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_filter_m2_data_wakeUp(void *vctx, uint32_t
*mode)
```

Description:
Force node to wake up

Parameters:
mode

Default: 0

moca_set_filter_m2_data_wakeUp

Prototype:

```
MOCALIB_GEN_SET_FUNCTION int moca_set_filter_m2_data_wakeUp(void *vctx, uint32_t
mode)
```

Description:

Force node to wake up

Parameters:

mode

Default: 0

moca_get_wom_pattern

Prototype:

```
MOCALIB_GEN_GET_FUNCTION int moca_get_wom_pattern(void *vctx, struct
moca_wom_pattern *out, int max_out_len)
```

Description:

Defines a WoM packet filter. MoCA will trigger a wakeup interrupt if it receives a packet matching the filter, and wom_mode is enabled. Up to 5 filters can be configured.

moca_set_wom_pattern

Prototype:

```
MOCALIB_GEN_SET_FUNCTION int moca_set_wom_pattern(void *vctx, struct
moca_wom_pattern_set *in)
```

Description:

Defines a WoL packet filter. MoCA will trigger a wakeup interrupt if it receives a packet matching the filter, and wom_mode is enabled. Up to 5 filters can be configured. Set Mask to all 0xFF to invalidate an entry

moca_get_wom_ip

Prototype:

```
MOCALIB_GEN_GET_FUNCTION int moca_get_wom_ip(void *vctx, uint32_t *out, int
max_out_len)
```

Description:
Defines a WoL packet filter. MoCA will trigger a wakeup interrupt if it receives an ARP packet matching the ipaddress, and wom_mode is enabled. Up to 5 IP addresses can be configured.

moca_set_wom_ip
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_wom_ip(void *vctx, const struct moca_wom_ip
*in)
```

Description:
Defines a WoL packet filter. MoCA will trigger a wakeup interrupt if it receives an ARP packet matching the ipaddress, and wom_mode is enabled. Up to 5 IP addresses can be configured.

moca_get_wom_magic_mac
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_wom_magic_mac(void *vctx, struct
moca_wom_magic_mac *out)
```

Description:
Defines the MAC address to be used in magic-packet filtering. This feature needs to be enabled via wom_magic_enable. MoCA will trigger a wakeup interrupt if it receives a magic-packet with this MAC address

moca_set_wom_magic_mac
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_wom_magic_mac(void *vctx, struct
moca_wom_magic_mac *in)
```

Description:
Defines the MAC address to be used in magic-packet filtering. This feature needs to be enabled via wom_magic_enable. MoCA will trigger a wakeup interrupt if it receives a magic-packet with this MAC address

moca_get_standby_power_state
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_standby_power_state(void *vctx, uint32_t
*state)
```

Description:
For GET operations, reports current standby power state.
For SET operations, set the power state of the core during system standby

Refer to power_state_capabilities to learn the supported power states.

Parameters:
state
A list from the range M0-M3
Default: 2
Minimum: 0
Maximum: 3

moca_set_standby_power_state
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_standby_power_state(void *vctx, uint32_t
state)
```

Description:
For GET operations, reports current standby power state.
For SET operations, set the power state of the core during system standby

Refer to power_state_capabilities to learn the supported power states.

Parameters:
state
A list from the range M0-M3
Default: 2
Minimum: 0
Maximum: 3

moca_get_wom_mode
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_wom_mode(void *vctx, uint32_t *val)
```

Description:
Enables WoM mode via packet filtering in the MoCA core in system suspend. See wom_ip and wom_pattern to configure the packet filtering

Parameters:

val
Default:
2 (SWITCH)
0
Minimum: 0
Maximum: 2

moca_set_wom_mode
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_wom_mode(void *vctx, uint32_t val)
```

Description:
Enables WoM mode via packet filtering in the MoCA core in system suspend. See wom_ip and wom_pattern to configure the packet filtering

Parameters:
val
Default:
2 (SWITCH)
0
Minimum: 0
Maximum: 2

moca_register_power_state_rsp_cb
Prototype:
```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_power_state_rsp_cb(void *vctx,
void (*callback)(void *userarg, uint32_t rsp_code), void *userarg)
```

Description:
Power state response message following a SET of power_state variable

moca_register_power_state_event_cb
Prototype:
```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_power_state_event_cb(void *vctx,
void (*callback)(void *userarg, uint32_t event_code), void *userarg)
```

Description:
Power state related events

moca_register_power_state_cap_cb
Prototype:
```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_power_state_cap_cb(void *vctx,
void (*callback)(void *userarg, uint32_t power_modes), void *userarg)
```

Description:
Power state capability message which announces which power saving modes are supported.

moca_get_wol
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_wol(void *vctx, uint32_t *val)
```
Parameters:
val

moca_set_wol
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_wol(void *vctx, uint32_t val)
```
Parameters:
val

moca_register_ps_cmd_cb
Prototype:
```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_ps_cmd_cb(void *vctx, void
(*callback)(void *userarg, uint32_t rsp_code), void *userarg)
```

Description:
Execute a power state change command and report the results of the command.

moca_do_ps_cmd
Prototype:
```
MOCALIB_GEN_DO_FUNCTION int moca_do_ps_cmd(void *vctx, uint32_t new_state,
uint32_t *rsp_code)
```

Description:
Execute a power state change command and report the results of the command.

moca_get_power_state_capabilities
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_power_state_capabilities(void *vctx,
uint32_t *power_modes)
```
Parameters:
power_modes

SECURITY Group
The Security group of parameters.

Structures

struct moca_aes_mm_key
Fields:

uint32_t	val[4]	

struct moca_aes_pm_key
Fields:

uint32_t	val[4]	

struct moca_aes_pmk_initial_key
Fields:

uint32_t	val[4]	

struct moca_current_keys
Fields:

uint32_t	aes_pmk_even_key[4]	Current AES PMK even key
uint32_t	aes_pmk_odd_key[4]	Current AES PMK odd key
uint32_t	aes_tek_even_key[4]	Current AES TEK even key

uint32_t	aes_tek_odd_key[4]	Current AES TEK odd key
uint32_t	pmk_even_key[2]	Current PMK even key
uint32_t	pmk_odd_key[2]	Current PMK odd key
uint32_t	tek_even_key[2]	Current TEK even key
uint32_t	tek_odd_key[2]	Current TEK odd key

struct moca_key_changed

Fields:

uint32_t	even_odd
uint32_t	key_type

struct moca_key_times

Fields:

uint32_t	aes_pmk_even_odd
uint32_t	aes_pmk_last_interval
uint32_t	aes_pmk_time
uint32_t	aes_tek_even_odd
uint32_t	aes_tek_last_interval
uint32_t	aes_tek_time
uint32_t	pmk_even_odd
uint32_t	pmk_last_interval
uint32_t	pmk_time
uint32_t	tek_even_odd
uint32_t	tek_last_interval
uint32_t	tek_time

struct moca_mmk_key

Fields:

uint32_t	mmk_key_hi	mmk_key_hi holds 32 msb.
uint32_t	mmk_key_lo	mmk_key_hi holds 32 lsb.

struct moca_password

Fields:

char	password [32]	The network password used to generate privacy keys. This string must be between 12 and 17 characters long with each character being a decimal number (0-9).

Default: 0

struct moca_permanent_salt
Fields:

uint32_t	aes_salt[3]

struct moca_pmk_initial_key
Fields:

uint32_t	pmk_initial_key_hi	pmk_initial_key_hi holds 32 msb.
uint32_t	pmk_initial_key_lo	pmk_initial_key_lo holds 32 lsb.

Functions

moca_get_privacy_en
Prototype:

```
MOCALIB_GEN_GET_FUNCTION int moca_get_privacy_en(void *vctx, uint32_t *val)
```

Description:
Enable the MoCA Link Privacy

Parameters:
val
Default: 0

moca_set_privacy_en
Prototype:

```
MOCALIB_GEN_SET_FUNCTION int moca_set_privacy_en(void *vctx, uint32_t val)
```

Description:
Enable the MoCA Link Privacy This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

Parameters:
val
Default: 0

moca_get_pmk_exchange_interval

Prototype:

```
MOCALIB_GEN_GET_FUNCTION int moca_get_pmk_exchange_interval(void *vctx, uint32_t
*msec)
```

Description:

PMK interval time. This configuration will take effect only after the next key change.

Parameters:

msec

Default: 39600000

Minimum: 20000

moca_set_pmk_exchange_interval

Prototype:

```
MOCALIB_GEN_SET_FUNCTION int moca_set_pmk_exchange_interval(void *vctx, uint32_t
msec)
```

Description:

PMK interval time. This configuration will take effect only after the next key change.

Parameters:

msec

Default: 39600000

Minimum: 20000

moca_get_tek_exchange_interval

Prototype:

```
MOCALIB_GEN_GET_FUNCTION int moca_get_tek_exchange_interval(void *vctx, uint32_t
*msec)
```

Description:

TEK intervals time. This configuration will take effect only after the next key change.

Parameters:

msec

Default: 540000

Minimum: 20000

moca_set_tek_exchange_interval

Prototype:

```
MOCALIB_GEN_SET_FUNCTION int moca_set_tek_exchange_interval(void *vctx, uint32_t
msec)
```

Description:

TEK intervals time. This configuration will take effect only after the next key change.

Parameters:

msec

Default: 540000

Minimum: 20000

moca_get_aes_exchange_interval

Prototype:

```
MOCALIB_GEN_GET_FUNCTION int moca_get_aes_exchange_interval(void *vctx, uint32_t
*msec)
```

Description:

AES PMK and TEK intervals time. This configuration will take effect only after the next key change.

Parameters:

msec

Default: 25200000

Minimum: 20000

moca_set_aes_exchange_interval

Prototype:

```
MOCALIB_GEN_SET_FUNCTION int moca_set_aes_exchange_interval(void *vctx, uint32_t
msec)
```

Description:

AES PMK and TEK intervals time. This configuration will take effect only after the next key change.

Parameters:

msec

Default: 25200000

Minimum: 20000

moca_get_mmk_key
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_mmk_key(void *vctx, struct moca_mmk_key
*out)
```

Description:
64-bit MAC Management Key, derived from a user input of 17 ASCII character password. (derived from GCAP.16)

moca_get_pmk_initial_key
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_pmk_initial_key(void *vctx, struct
moca_pmk_initial_key *out)
```

Description:
64 bits Privacy Management Key Initial, derived from a user input of 17 ASCII chars password. (derived from GCAP.16)

moca_get_aes_mm_key
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_aes_mm_key(void *vctx, struct
moca_aes_mm_key *out)
```

Description:
AES MAC Management Key

moca_set_aes_mm_key
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_aes_mm_key(void *vctx, const struct
moca_aes_mm_key *in)
```

Description:
AES MAC Management Key This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

moca_get_aes_pm_key
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_aes_pm_key(void *vctx, struct
moca_aes_pm_key *out)
```

Description:
AES Privacy Management Key

moca_set_aes_pm_key
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_aes_pm_key(void *vctx, const struct
moca_aes_pm_key *in)
```

Description:
AES Privacy Management Key This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

moca_get_current_keys
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_current_keys(void *vctx, struct
moca_current_keys *out)
```

Description:
Retrieve the various current MoCA key values of this device.

moca_get_permanent_salt
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_permanent_salt(void *vctx, struct
moca_permanent_salt *out)
```

Description:
Retrieve the AES permanent salt of this device.

moca_get_aes_pmk_initial_key
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_aes_pmk_initial_key(void *vctx, struct
moca_aes_pmk_initial_key *out)
```

Description:

128-bit Privacy Management Key Initial.

moca_set_aes_pmk_initial_key
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_aes_pmk_initial_key(void *vctx, const struct
moca_aes_pmk_initial_key *in)
```

Description:
128-bit Privacy Management Key Initial. This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

moca_register_key_changed_cb
Prototype:
```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_key_changed_cb(void *vctx, void
(*callback)(void *userarg, struct moca_key_changed *out), void *userarg)
```

Description:
This event provides notification that privacy keys have been updated.

moca_get_key_times
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_key_times(void *vctx, struct moca_key_times
*out)
```

moca_get_password
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_password(void *vctx, struct moca_password
*out)
```

moca_set_password
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_password(void *vctx, struct moca_password
*in)
```

DEBUG Group
The Debug group of parameters is used for testing by advanced users only.

Structures

struct moca_const_tx_params

Fields:

uint32_t	const_tx_band[16]	Bitmask of the valid 0 - 511 sub-carriers E.g. const_tx_band[0] corresponds to SCs 0-31 Setting const_tx_band[0] to 0x8000000F will enable SCs 0,28-31 Setting const_tx_band[15] to 0x00000001 will enable SC 511
uint32_t	const_tx_sc1	The first SC tone for single tone only
uint32_t	const_tx_sc2	The second SC tone for single tone only
uint32_t	const_tx_submode	*Default:* 1 *Minimum:* 0 *Maximum:* 3

struct moca_error

Fields:

uint32_t	err_id
uint32_t	num_params
uint32_t	string_id

struct moca_error_lookup

Fields:

uint32_t	err_id
uint32_t	num_params
uint32_t	string_id

struct moca_error_to_mask

Fields:

int32_t	error1
int32_t	error2
int32_t	error3

struct moca_fw_file

Fields:

uint8_t	fw_file[128]

struct moca_gmii_trap_header
Fields:

uint8_t	dest_mac[6]	Destination MAC address
uint8_t	dscp_ecn	Differentiated Services Code Point and Explicit Congestion Notification fields *Default:* 0
uint8_t	dst_ip_addr[4]	Destination IP address.
uint16_t	dst_port	Destination UDP port.
uint16_t	id	Identification field. *Default:* 0
uint16_t	ip_checksum	IP checksum, to be initialized by the host using an IP length of zero.
uint8_t	prot	Protocol field. *Default:* 17
uint8_t	source_mac [6]	Source MAC address
uint8_t	src_ip_addr[4]	Source IP address.
uint16_t	src_port	Source UDP port.
uint8_t	ttl	Time to live field. *Default:* 32

struct moca_mocad_printf_out
Fields:

int8_t	msg[240]

Functions

moca_get_mtm_en
Prototype:

```
MOCALIB_GEN_GET_FUNCTION int moca_get_mtm_en(void *vctx, uint32_t *val)
```

Description:
Enable/Disable (manufacturing Test Mode)

Parameters:
val
Default: 0

moca_set_mtm_en
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_mtm_en(void *vctx, uint32_t val)
```

Description:
Enable/Disable (manufacturing Test Mode) This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

Parameters:
val
Default: 0

moca_get_cir_prints
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_cir_prints(void *vctx, uint32_t *bool_val)
```

Description:
Enabling or disabling the CIR prints.
To enable these prints moca_core_trace_enable must also be set to 1.

Parameters:
bool_val
Default: 0

moca_set_cir_prints
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_cir_prints(void *vctx, uint32_t bool_val)
```

Description:
Enabling or disabling the CIR prints.
To enable these prints moca_core_trace_enable must also be set to 1.

Parameters:
bool_val
Default: 0

moca_get_snr_prints
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_snr_prints(void *vctx, uint32_t *bool_val)
```

Description:
Enabling or disabling the SNR prints.
To enable these prints moca_core_trace_enable must also be set to 1.

Parameters:
bool_val
Default: 0

moca_set_snr_prints
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_snr_prints(void *vctx, uint32_t bool_val)
```

Description:
Enabling or disabling the SNR prints.
To enable these prints moca_core_trace_enable must also be set to 1.

Parameters:
bool_val
Default: 0

moca_get_sigma2_prints
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_sigma2_prints(void *vctx, uint32_t
*bool_val)
```

Description:
Enabling or disabling the Sigma II prints of Probe I results.
To enable these prints moca_core_trace_enable must also be set to 1.

Parameters:
bool_val
Default: 0

moca_set_sigma2_prints
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_sigma2_prints(void *vctx, uint32_t bool_val)
```

Description:
Enabling or disabling the Sigma II prints of Probe I results.
To enable these prints moca_core_trace_enable must also be set to 1.

Parameters:
bool_val
Default: 0

moca_get_const_tx_params
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_const_tx_params(void *vctx, struct
moca_const_tx_params *out)
```

Description:
Continuous TX mode debug parameters

moca_set_const_tx_params
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_const_tx_params(void *vctx, const struct
moca_const_tx_params *in)
```

Description:
Continuous TX mode debug parameters This function can be invoked at any time however the setting will only take effect when the MoCA interface is started.

moca_set_gmii_trap_header
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_gmii_trap_header(void *vctx, struct
moca_gmii_trap_header *in)
```

Description:
GMII Trap Header.

This structure allows the host to configure the GMII trap buffer Ethernet, IP and UDP headers. When the destination MAC address is non-zero, the firmware will send certain traps over the GMII interface using the specified header.

The host is responsible for ensuring that the header contains valid fields. The firmware will update the length fields and checksum values.

To disable GMII traps, the host should set this structure to all zeroes.

moca_get_led_status
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_led_status(void *vctx, uint32_t *led_status)
```

Description:
Retrieve the current status of the MoCA LED.

Parameters:
led_status
A bitfield indicating the current MoCA LED status

moca_get_moca_core_trace_enable
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_moca_core_trace_enable(void *vctx, uint32_t
*bool_val)
```

Description:
Enabling or disabling the MoCA core trace to the host via MMP traps.
When measuring performance, the trace should be turned off.

Parameters:
bool_val
Default: 0

moca_set_moca_core_trace_enable
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_moca_core_trace_enable(void *vctx, uint32_t
bool_val)
```

Description:
Enabling or disabling the MoCA core trace to the host via MMP traps.
When measuring performance, the trace should be turned off.

Parameters:
bool_val
Default: 0

moca_register_error_cb
Prototype:
```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_error_cb(void *vctx, void
(*callback)(void *userarg, struct moca_error *out), void *userarg)
```

moca_register_error_lookup_cb
Prototype:
```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_error_lookup_cb(void *vctx, void
(*callback)(void *userarg, struct moca_error_lookup *out), void *userarg)
```

moca_get_error_to_mask
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_error_to_mask(void *vctx, struct
moca_error_to_mask *out)
```

moca_set_error_to_mask
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_error_to_mask(void *vctx, const struct
moca_error_to_mask *in)
```

moca_set_fw_file
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_fw_file(void *vctx, struct moca_fw_file *in)
```

moca_get_verbose
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_verbose(void *vctx, uint32_t *level)
```

Description:
This parameter controls which prints are displayed by the moca daemon. This is a bit field where each bit enables or disables the printings of a specific log level.
By default, Error, Warning and Informational messages are printed.

Parameters:
level

moca_set_verbose
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_verbose(void *vctx, uint32_t level)
```

Description:
This parameter controls which prints are displayed by the moca daemon. This is a bit field where each bit enables or disables the printings of a specific log level.
By default, Error, Warning and Informational messages are printed.

Parameters:
level

moca_get_dont_start_moca
Prototype:
```
MOCALIB_GEN_GET_FUNCTION int moca_get_dont_start_moca(void *vctx, uint32_t
*dont_start_moca)
```

Description:
This parameter is used to tell the MoCA Daemon not to boot the MoCA core upon start-up. The MoCA daemon will wait for this field to be set to 0 before starting the MoCA core after it has been set to 1.

Parameters:
dont_start_moca
Default: 0

moca_set_dont_start_moca
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_dont_start_moca(void *vctx, uint32_t
dont_start_moca)
```

Description:

This parameter is used to tell the MoCA Daemon not to boot the MoCA core upon start-up. The MoCA daemon will wait for this field to be set to 0 before starting the MoCA core after it has been set to 1.

Parameters:
dont_start_moca
Default: 0

moca_set_no_rtt
Prototype:
```
MOCALIB_GEN_SET_FUNCTION int moca_set_no_rtt(void *vctx)
```

Description:
This parameter is used to disable RTT prints by turning off bit 7 of the verbose field.

moca_register_mocad_printf_cb
Prototype:
```
MOCALIB_GEN_REGISTER_FUNCTION void moca_register_mocad_printf_cb(void *vctx, void
(*callback)(void *userarg, struct moca_mocad_printf_out *out), void *userarg)
```

Description:
One trap is sent for every core trace

moca_do_mocad_printf
Prototype:
```
MOCALIB_GEN_DO_FUNCTION int moca_do_mocad_printf(void *vctx, struct
moca_mocad_printf_out *out)
```

Description:
One trap is sent for every core trace