ULP Node Host Message Specification

DCN: 014-0020-00
System Release: 1.2.5.x
Date: April 6, 2011

On-Ramp Wireless Incorporated
10920 Via Frontera, Suite 200
San Diego, CA 92127
U.S.A.

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 CAL_CONFIG_FlashCalibration_t Struct Reference

Data Fields

- uint32_t **version**
- uint32_t **hardware_version**
- uint32_t **serial_num**
- uint32_t **node_id**
- uint32_t **k_phy**
- uint32_t **maxTxVGALow**
- uint32_t **maxTxVGAMid**
- uint32_t **maxTxVGAHigh**
- uint32_t **maxTxVGAOutputPwr**
- uint32_t **max_tx_pwr_low_freq**
- uint32_t **max_tx_pwr_mid_freq**
- uint32_t **max_tx_pwr_high_freq**
- uint32_t **tx_vga35_pwr_low_freq**
- uint32_t **tx_vga35_pwr_mid_freq**
- uint32_t **tx_vga35_pwr_high_freq**
- uint32_t **lna_high_gain_low_freq**
- uint32_t **lna_mid_gain_low_freq**
- uint32_t **lna_low_gain_low_freq**
- uint32_t **lna_high_gain_mid_freq**
- uint32_t **lna_mid_gain_mid_freq**
- uint32_t **lna_low_gain_mid_freq**
- uint32_t **lna_high_gain_high_freq**
- uint32_t **lna_mid_gain_high_freq**
- uint32_t **lna_low_gain_high_freq**
- uint32_t **noise_pwr_1mhz**
- uint32_t **i_offset**
- uint32_t **q_offset**
- uint32_t **osc_26mhz**
- uint32_t **osc_32khz**
- uint32_t **pa_temp_mcomp**

- uint32_t **pa_temp_bcomp**
- uint32_t **aux_a2d_temp_m**
- uint32_t **aux_a2d_temp_b**
- uint32_t **t_cal**
- uint32_t **aux_a2d_chan1_m**
- uint32_t **aux_a2d_chan1_b**
- uint32_t **aux_a2d_chan2_m**
- uint32_t **aux_a2d_chan2_b**

The documentation for this struct was generated from the following file:

- cal_config.h

3.2 CAL_CONFIG_FlashConfig_t Struct Reference

Data Fields

- uint32_t **version**
- uint32_t **bcastGoldCode** [(6 *4)]
- uint8_t **channel** [(6 *4)]
- uint32_t **tcxoFreq**
- uint16_t **countryCode**
- uint16_t **sysSelMinSleepTimer**
- uint16_t **fieldTestNumPdusPerFrame**
- uint16_t **fieldTestNumFramePeriod**
- uint8_t **fieldTestUlRssiMargin**
- uint8_t **autorun**
- uint8_t **operatingMode**
- uint8_t **dlBcastSpreading**
- uint8_t **maxTxPwrLimit**
- uint8_t **joinType**
- uint8_t **joinBackoffType**
- uint8_t **otaDwnldNodeType_InitSeq**
- uint32_t **sysSelMaxSleepTimer**
- int16_t **sysSelImmediateJoinThreshold**
- uint8_t **sysSelMaxFreqOptimizedPasses**
- uint8_t **reserved**

The documentation for this struct was generated from the following file:

- cal_config.h

3.3 host_msg_ack_t Struct Reference

An ACK message.

`#include <host_customer_msg.h>`

Collaboration diagram for host_msg_ack_t:

Data Fields

- host_msg_header_t header
- uint32_t footer

3.3.1 Detailed Description

An ACK message. Used by the eNode to acknowledge reception from the Host of a Host Interface Message. This message is sent by the eNode in response to every Host Interface Message that it receives. The Host should wait until it receives an ACK from the previously sent message before it sends its next message to the eNode.

See also

HOST_MSG_TYPE_ACK

3.3.2 Field Documentation

3.3.2.1 uint32_t host_msg_ack_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.3.2.2 host_msg_header_t host_msg_ack_t::header

2-byte Message Length followed by 2-byte Message Type

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.4 host_msg_beginSwUpgrade_t Struct Reference

A Begin SW Upgrade message.

```
#include <host_customer_msg.h>
```

Collaboration diagram for host_msg_beginSwUpgrade_t:

Data Fields

- host_msg_header_t header
- uint32_t numChunks
- uint32_t checksum
- uint32_t footer

3.4.1 Detailed Description

A Begin SW Upgrade message. Used by the Host to start the process of upgrading the eNode software. Caution should be exercised when sending this message as this will cause the eNode to erase and start to overwrite the software section of the flash memory device. If this process is not completed with a valid eNode software image, then the eNode may not be able to boot up at all.

See also

HOST_MSG_TYPE_BEGIN_SW_UPGR

3.4.2 Field Documentation

3.4.2.1 uint32_t host_msg_beginSwUpgrade_t::checksum

Expected checksum over entirety of SW upgrade

3.4.2.2 uint32_t host_msg_beginSwUpgrade_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.4.2.3 host_msg_header_t host_msg_beginSwUpgrade_t::header

2-byte Message Length followed by 2-byte Message Type

3.4.2.4 uint32_t host_msg_beginSwUpgrade_t::numChunks

Number of data chunks in the SW upgrade

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.5 host_msg_beginSwUpgradeRsp_t Struct Reference

A Begin Software Upgrade Response message.

`#include <host_customer_msg.h>`

Collaboration diagram for host_msg_beginSwUpgradeRsp_t:

Data Fields

- host_msg_header_t header
- uint32_t result
- uint32_t footer

3.5.1 Detailed Description

A Begin Software Upgrade Response message. This is used by the eNode to respond to a request to upgrade eNode software. The contents of this message indicate whether is it OK to continue with the software upgrade process or not.

See also

HOST_MSG_TYPE_BEGIN_SW_UPGR_RSP

3.5.2 Field Documentation

3.5.2.1 uint32_t host_msg_beginSwUpgradeRsp_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.5.2.2 host_msg_header_t host_msg_beginSwUpgradeRsp_t::header

2-byte Message Length followed by 2-byte Message Type

3.5.2.3 uint32_t host_msg_beginSwUpgradeRsp_t::result

The result of the request to begin SW upgrade process.

0 = ok, 1 = invalid state, 2 = bad size

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.6 host_msg_blackoutEndInd_t Struct Reference

Collaboration diagram for host_msg_blackoutEndInd_t:

Data Fields

- host_msg_header_t header
- uint8_t wasUpdateIntervalSkipped
- uint8_t **pad** [3]
- uint32_t footer

3.6.1 Field Documentation

3.6.1.1 uint32_t host_msg_blackoutEndInd_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.6.1.2 host_msg_header_t host_msg_blackoutEndInd_t::header

2-byte Message Length followed by 2-byte Message Type

3.6.1.3 uint8_t host_msg_blackoutEndInd_t::wasUpdateIntervalSkipped

0 if no update interval fell in blackout, 1 if UI was squished

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.7 host_msg_blackoutStartInd_t Struct Reference

Collaboration diagram for host_msg_blackoutStartInd_t:

Data Fields

- host_msg_header_t header
- uint32_t secUntilStart
- uint32_t durationInSec
- uint32_t footer

3.7.1 Field Documentation

3.7.1.1 uint32_t host_msg_blackoutStartInd_t::durationInSec

Duration of the blackout.

3.7.1.2 uint32_t host_msg_blackoutStartInd_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.7.1.3 host_msg_header_t host_msg_blackoutStartInd_t::header

2-byte Message Length followed by 2-byte Message Type

3.7.1.4 uint32_t host_msg_blackoutStartInd_t::secUntilStart

Seconds until the blackout period begins. Set to 0 to signal immediate blackout.

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.8 host_msg_broadcastDataReq_t Struct Reference

Collaboration diagram for host_msg_broadcastDataReq_t:

Data Fields

- host_msg_header_t header
- uint32_t bcastId
- uint32_t offset
- uint32_t length
- uint32_t footer

3.8.1 Field Documentation

3.8.1.1 uint32_t host_msg_broadcastDataReq_t::bcastId

Broadcast id.

3.8.1.2 uint32_t host_msg_broadcastDataReq_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.8.1.3 host_msg_header_t host_msg_broadcastDataReq_t::header

2-byte Message Length followed by 2-byte Message Type

3.8.1.4 uint32_t host_msg_broadcastDataReq_t::length

Length in bytes requested. Valid range is 1 - 256.

3.8.1.5 uint32_t host_msg_broadcastDataReq_t::offset

Offset into the broadcast data requested.

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.9 host_msg_broadcastDataRsp_t Struct Reference

Collaboration diagram for host_msg_broadcastDataRsp_t:

Data Fields

- host_msg_header_t header
- host_msg_broadcastStatus_t status
- uint16_t **reserved**
- uint32_t bcastId
- uint32_t offset
- uint32_t length
- uint8_t payload [256]
- uint32_t footer

3.9.1 Field Documentation

3.9.1.1 uint32_t host_msg_broadcastDataRsp_t::bcastId

Broadcast id.

3.9.1.2 uint32_t host_msg_broadcastDataRsp_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.9.1.3 host_msg_header_t host_msg_broadcastDataRsp_t::header

2-byte Message Length followed by 2-byte Message Type

3.9.1.4 uint32_t host_msg_broadcastDataRsp_t::length

Length in bytes requested. Valid range is 1 - 256.

3.9.1.5 uint32_t host_msg_broadcastDataRsp_t::offset

Offset into the broadcast data requested.

3.9.1.6 uint8_t host_msg_broadcastDataRsp_t::payload[256]

Broadcast data payload. If length is not 256, remaining bytes are undefined.

3.9.1.7 host_msg_broadcastStatus_t host_msg_broadcastDataRsp_t::status

Status.

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.10 host_msg_broadcastEndInd_t Struct Reference

Collaboration diagram for host_msg_broadcastEndInd_t:

Data Fields

- host_msg_header_t header
- uint32_t bcastId
- uint32_t length
- uint32_t footer

3.10.1 Field Documentation

3.10.1.1 uint32_t host_msg_broadcastEndInd_t::bcastId

Broadcast id.

3.10.1.2 uint32_t host_msg_broadcastEndInd_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.10.1.3 host_msg_header_t host_msg_broadcastEndInd_t::header

2-byte Message Length followed by 2-byte Message Type

3.10.1.4 uint32_t host_msg_broadcastEndInd_t::length

Length in bytes of the broadcast.

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.11 host_msg_broadcastStartCnf_t Struct Reference

Collaboration diagram for host_msg_broadcastStartCnf_t:

Data Fields

- host_msg_header_t header
- uint32_t bcastId
- uint32_t acceptBroadcast
- uint32_t footer

3.11.1 Field Documentation

3.11.1.1 uint32_t host_msg_broadcastStartCnf_t::acceptBroadcast

1 if the host wants the eNode to continue receiving the broadcast, 0 otherwise.

3.11.1.2 uint32_t host_msg_broadcastStartCnf_t::bcastId

Broadcast id.

3.11.1.3 uint32_t host_msg_broadcastStartCnf_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.11.1.4 host_msg_header_t host_msg_broadcastStartCnf_t::header

2-byte Message Length followed by 2-byte Message Type

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.12 host_msg_broadcastStartInd_t Struct Reference

Collaboration diagram for host_msg_broadcastStartInd_t:

Data Fields

- host_msg_header_t header
- uint32_t bcastId
- uint8_t payload [256]
- uint32_t length
- uint32_t footer

3.12.1 Field Documentation

3.12.1.1 uint32_t host_msg_broadcastStartInd_t::bcastId

Unique id generated by eNode to refer to this broadcast.

3.12.1.2 uint32_t host_msg_broadcastStartInd_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.12.1.3 host_msg_header_t host_msg_broadcastStartInd_t::header

2-byte Message Length followed by 2-byte Message Type

3.12.1.4 uint32_t host_msg_broadcastStartInd_t::length

Length in bytes of the broadcast.

3.12.1.5 uint8_t host_msg_broadcastStartInd_t::payload[256]

Customer-specific broadcast data identifier.

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.13 host_msg_connect_t Struct Reference

A CONNECT message.

`#include <host_customer_msg.h>`

Collaboration diagram for host_msg_connect_t:

Data Fields

- host_msg_header_t header
- uint32_t connected
- uint32_t footer

3.13.1 Detailed Description

A CONNECT message. Used by the Host to indicate to the eNode over which bus to communicate. This is typically the SPI bus. This message must be sent to the eNode before other messages can be communicated from the eNode to the Host.

See also

HOST_MSG_TYPE_CONNECT

3.13.2 Field Documentation

3.13.2.1 uint32_t host_msg_connect_t::connected

Specifies eNode/Host connection bus.

See also

host_msg_host_t

3.13.2.2 uint32_t host_msg_connect_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.13.2.3 host_msg_header_t host_msg_connect_t::header

2-byte Message Length followed by 2-byte Message Type

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.14 host_msg_err_t Struct Reference

An ERR message.

`#include <host_customer_msg.h>`

Collaboration diagram for host_msg_err_t:

Data Fields

- host_msg_header_t header
- uint32_t errCode
- uint32_t footer

3.14.1 Detailed Description

An ERR message. Used by the eNode to indicate an error has occurred. This message is not currently implemented.

See also

HOST_MSG_TYPE_ERR

3.14.2 Field Documentation

3.14.2.1 uint32_t host_msg_err_t::errCode

Error code

3.14.2.2 uint32_t host_msg_err_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.14.2.3 host_msg_header_t host_msg_err_t::header

2-byte Message Length followed by 2-byte Message Type

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.15 host_msg_flushTxSduQueue_t Struct Reference

Requests all queued uplink sdu to be dropped.

`#include <host_customer_msg.h>`

Collaboration diagram for host_msg_flushTxSduQueue_t:

Data Fields

- host_msg_header_t header
- uint32_t includeInProgressSdus
- uint32_t footer

3.15.1 Detailed Description

Requests all queued uplink sdu to be dropped. Used by the Host to drop any TX SDUs waiting for transmission.

See also

HOST_MSG_TYPE_FLUSH_TXSDU_QUEUE

3.15.2 Field Documentation

3.15.2.1 uint32_t host_msg_flushTxSduQueue_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.15.2.2 host_msg_header_t host_msg_flushTxSduQueue_t::header

2-byte Message Length followed by 2-byte Message Type

3.15.2.3 uint32_t host_msg_flushTxSduQueue_t::includeInProgressSdus

A value of 1 will also flush in progress SDUs, a value of 0 will not flush in progress SDUs

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.16 host_msg_flushTxSduQueueRsp_t Struct Reference

Relays the result from a flush TXSDU queue message.

`#include <host_customer_msg.h>`

Collaboration diagram for host_msg_flushTxSduQueueRsp_t:

Data Fields

- host_msg_header_t header
- uint32_t flushSucceeded
- uint32_t footer

3.16.1 Detailed Description

Relays the result from a flush TXSDU queue message. Used by the Node to convey the result of a flush TXSDU queue message

See also

HOST_MSG_TYPE_FLUSH_TXSDU_QUEUE
HOST_MSG_TYPE_FLUSH_TXSDU_QUEUE_RSP

3.16.2 Field Documentation

3.16.2.1 uint32_t host_msg_flushTxSduQueueRsp_t::flushSucceeded

0 if flush operation could not be executed, 1 if flush operation was completed successfully

3.16.2.2 uint32_t host_msg_flushTxSduQueueRsp_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.16.2.3 host_msg_header_t host_msg_flushTxSduQueueRsp_t::header

2-byte Message Length followed by 2-byte Message Type

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.17 host_msg_frameStats_t Struct Reference

A Frame Statistics Message.

`#include <host_customer_msg.h>`

Collaboration diagram for host_msg_frameStats_t:

Data Fields

- host_msg_header_t header
- int16 center_freq_offset
- uint16 failedFrameCnt
- uint16 fingerTimingOffsetParity [10]
- int16 fingerCAFC [10]
- uint16 fingerEnergy [10]
- uint8 fingerFineAFCs [10]
- uint16 lowTimingOffset
- uint16 highTimingOffset
- int16 lowCAFC
- int16 highCAFC
- int16 RSSI
- uint16 frameDelaySymbols
- int16 hammingWeight [10]
- uint8 winningFineAFC [10]
- uint16 boostedFineAFCMetric [10]
- uint16 demodType
- uint16 subslot
- int32 txFreqStride
- int32 txTimeTrackingStride
- int32 freqOffset
- uint16 lastTxSpreading
- uint16 lastTxSubslot
- uint32 oscCal32k
- uint32 oscCal26m
- uint16 digitalTruncation
- uint16 txVGA
- int16 rssi_high
- int16 rssi_low
- uint32 sfn
- uint16 lastDchSpreading
- uint8 channel
- uint8 numLoggingMsgsDropped
- int32 latitude
- int32 longitude
- int32 altitude
- uint16 heading
- uint16 velocity
- uint16 **tempAdc**
- int16 **tempEst**
- uint32 fingerPower [10]
- uint32_t footer

3.17.1 Detailed Description

A Frame Statistics Message. Used by the eNode to report Frame Statistics to the Host.

See also

HOST_MSG_TYPE_FRAME_STATS

3.17.2 Field Documentation

3.17.2.1 int32 host_msg_frameStats_t::altitude

GPS Altitude in meters

3.17.2.2 uint16 host_msg_frameStats_t::boostedFineAFCMetric[10]

Fine AFC

3.17.2.3 int16 host_msg_frameStats_t::center_freq_offset

Center Frequency Offset

3.17.2.4 uint8 host_msg_frameStats_t::channel

Last demoded channel

3.17.2.5 uint16 host_msg_frameStats_t::demodType

Demod type

See also

host_msg_frameStatsType_t

3.17.2.6 uint16 host_msg_frameStats_t::digitalTruncation

Number of 6dB adjustments for TX AFC

3.17.2.7 uint16 host_msg_frameStats_t::failedFrameCnt

Number of consecutive Failed Frames

3.17.2.8 int16 host_msg_frameStats_t::fingerCAFC[10]

Finger Course AFC

3.17.2.9 uint16 host_msg_frameStats_t::fingerEnergy[10]

Finger Energy

3.17.2.10 uint8 host_msg_frameStats_t::fingerFineAFCs[10]

Finger Fine AFC

3.17.2.11 uint32 host_msg_frameStats_t::fingerPower[10]

Finger Power

3.17.2.12 uint16 host_msg_frameStats_t::fingerTimingOffsetParity[10]

bits 0-1: Timing parity bits 2-15: Timing offset

3.17.2.13 uint32_t host_msg_frameStats_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.17.2.14 uint16 host_msg_frameStats_t::frameDelaySymbols

Start of acquisition to first RX symbol time (in symbols)

3.17.2.15 int32 host_msg_frameStats_t::freqOffset

Absolute Frequency Offset

3.17.2.16 int16 host_msg_frameStats_t::hammingWeight[10]

Hamming Weight

3.17.2.17 host_msg_header_t host_msg_frameStats_t::header

2-byte Message Length followed by 2-byte Message Type

3.17.2.18 uint16 host_msg_frameStats_t::heading

GPS Heading in 12.4 degrees format

3.17.2.19 int16 host_msg_frameStats_t::highCAFC

high end of Course AFC in rectangle

3.17.2.20 uint16 host_msg_frameStats_t::highTimingOffset

high end of Timing Offset in rectangle

3.17.2.21 uint16 host_msg_frameStats_t::lastDchSpreading

DCH Spreading Factor.

Spreading Factor = 2^lastDchSpreading

valid range = 4-14 (16-8192)

3.17.2.22 uint16 host_msg_frameStats_t::lastTxSpreading

Last Tx Spreading Factor, stored in log(2) form.

Spreading Factor = 2^lastTxSpreading

valid range = 4-14 (16-8192)

3.17.2.23 uint16 host_msg_frameStats_t::lastTxSubslot

Last Tx Subslot

3.17.2.24 int32 host_msg_frameStats_t::latitude

GPS Latitude in 9.23s degrees format

3.17.2.25 int32 host_msg_frameStats_t::longitude

GPS Longitude in 9.23s degrees format

3.17.2.26 int16 host_msg_frameStats_t::lowCAFC

low end of Course AFC in rectangle

3.17.2.27 uint16 host_msg_frameStats_t::lowTimingOffset

low end of Timing Offset in rectangle

3.17.2.28 uint8 host_msg_frameStats_t::numLoggingMsgsDropped

Count of dropped messages

3.17.2.29 uint32 host_msg_frameStats_t::oscCal26m

26MHz Oscillator Cal

3.17.2.30 uint32 host_msg_frameStats_t::oscCal32k

32KHz Oscillator Cal

3.17.2.31 int16 host_msg_frameStats_t::RSSI

RSSI $*$ 16

3.17.2.32 int16 host_msg_frameStats_t::rssi_high

RSSI $*$ 4

3.17.2.33 int16 host_msg_frameStats_t::rssi_low

RSSI $*$ 16

3.17.2.34 uint32 host_msg_frameStats_t::sfn

Frame Number

3.17.2.35 uint16 host_msg_frameStats_t::subslot

Subslot

3.17.2.36 int32 host_msg_frameStats_t::txFreqStride

TX AFC Frequency Stride

3.17.2.37 int32 host_msg_frameStats_t::txTimeTrackingStride

Time Tracking Stride

3.17.2.38 uint16 host_msg_frameStats_t::txVGA

TX VGA

3.17.2.39 uint16 host_msg_frameStats_t::velocity

GPS Velocity in Km/Hr

3.17.2.40 uint8 host_msg_frameStats_t::winningFineAFC[10]

Fine AFC

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.18 host_msg_getExceptionBufferReq_t Struct Reference

A Get Exception Buffer request message.

`#include <host_customer_msg.h>`

Collaboration diagram for host_msg_getExceptionBufferReq_t:

Data Fields

- host_msg_header_t header
- uint8_t clearBuffer
- uint8_t chunk
- uint16_t **reserved2**
- uint32_t footer

3.18.1 Detailed Description

A Get Exception Buffer request message. Used by the Host to request Reset Exception Information from the eNode. Format of exception data is variable release to release and used internally to Onramp for support.

See also

HOST_MSG_TYPE_GET_EXCEPTION_BUFFER_REQ

3.18.2 Field Documentation

3.18.2.1 uint8_t host_msg_getExceptionBufferReq_t::chunk

Specifies the chunk to retrieve. Valid chunks are 0 through 10.

3.18.2.2 uint8_t host_msg_getExceptionBufferReq_t::clearBuffer

If set to 1 the exception buffer will be cleared.

3.18.2.3 uint32_t host_msg_getExceptionBufferReq_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.18.2.4 host_msg_header_t host_msg_getExceptionBufferReq_t::header

2-byte Message Length followed by 2-byte Message Type

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.19 host_msg_getExceptionBufferRsp_t Struct Reference

A Get Exception Buffer response message.

`#include <host_customer_msg.h>`

Collaboration diagram for host_msg_getExceptionBufferRsp_t:

Data Fields

- host_msg_header_t header
- uint8_t buffer [256]
- uint32_t footer

3.19.1 Detailed Description

A Get Exception Buffer response message. Used by the eNode to report Reset Exception Information to the Host. This is sent in response to a HOST_MSG_TYPE_GET_EXCEPTION_BUFFER_REQ. This message contains information on the conditions of the last reset.

See also

HOST_MSG_TYPE_GET_EXCEPTION_BUFFER_RSP

3.19.2 Field Documentation

3.19.2.1 uint8_t host_msg_getExceptionBufferRsp_t::buffer[256]

Buffer containing sw exception data from last reboot

3.19.2.2 uint32_t host_msg_getExceptionBufferRsp_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.19.2.3 host_msg_header_t host_msg_getExceptionBufferRsp_t::header

2-byte Message Length followed by 2-byte Message Type

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.20 host_msg_getParamRsp_t Struct Reference

A Get Params Rsp message.

```
#include <host_customer_msg.h>
```

Collaboration diagram for host_msg_getParamRsp_t:

Data Fields

- host_msg_header_t header
- uint16 channelBW
- uint16 channelNum
- uint16 numNCAccum
- int16 rssiMargin
- uint16 demodChannel
- uint16 bcastSlot
- uint16 dataSubslot
- uint16 pad
- uint32 bcastGoldCode
- uint32 dlDataGoldCode
- uint16 bcastSpreading
- uint16 dlDataSpreading
- uint16 ulSpreading
- uint16 systemState
- uint16 cid
- int8 listenInterval
- int8 slotInterval
- uint32 nodeId
- int16 maxTxPwrLimit
- int16 maxTxPwrLimitHeadRoom
- uint32_t footer

3.20.1 Detailed Description

A Get Params Rsp message. Used by the eNode to communicate to the Host the current configuration of various control parameters.

See also

HOST_MSG_TYPE_GET_PARAMS_RSP

3.20.2 Field Documentation

3.20.2.1 uint32 host_msg_getParamRsp_t::bcastGoldCode

Broadcast Gold Code NOTE: Test parameter for internal debug purposes only.

3.20.2.2 uint16 host_msg_getParamRsp_t::bcastSlot

Broadcast slot NOTE: Test parameter for internal debug purposes only.

3.20.2.3 uint16 host_msg_getParamRsp_t::bcastSpreading

Broadcast Spreading Factor, stored in log(2) form.

Spreading Factor = 2^bcastSpreading

valid range = 4-14 (16-8192)

NOTE: Test parameter for internal debug purposes only.

3.20.2.4 uint16 host_msg_getParamRsp_t::channelBW

Channel Bandwidth PHY_REGS_BandWidth_t: 0=2000KHz, 1=1000KHz, 2=500KHz NOTE: Test parameter for internal debug purposes only.

3.20.2.5 uint16 host_msg_getParamRsp_t::channelNum

Channel number (1.99 mhz steps from 2402)

3.20.2.6 uint16 host_msg_getParamRsp_t::cid

Connection ID NOTE: Test parameter for internal debug purposes only.

3.20.2.7 uint16 host_msg_getParamRsp_t::dataSubslot

Data Subslot NOTE: Test parameter for internal debug purposes only.

3.20.2.8 uint16 host_msg_getParamRsp_t::demodChannel

Which channel to demod: Broadcast or Data Channel NOTE: Test parameter for internal debug purposes only.

3.20.2.9 uint32 host_msg_getParamRsp_t::dlDataGoldCode

Downlink Data Gold Code NOTE: Test parameter for internal debug purposes only.

3.20.2.10 uint16 host_msg_getParamRsp_t::dlDataSpreading

Downlink Data Spreading Factor, stored in log(2) form.

Spreading Factor = 2^dlDataSpreading

valid range = 4-14 (16-8192)

NOTE: Test parameter for internal debug purposes only.

3.20.2.11 uint32_t host_msg_getParamRsp_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.20.2.12 host_msg_header_t host_msg_getParamRsp_t::header

2-byte Message Length followed by 2-byte Message Type

3.20.2.13 int8 host_msg_getParamRsp_t::listenInterval

Listen Interval.

0 - Every Frame (only allowed for continuous mode) 1 - Every Update Interval 2 - Every Second Update Interval 3 - Every Third Update Interval ... 10 - Every Tenth Update Interval

3.20.2.14 int16 host_msg_getParamRsp_t::maxTxPwrLimit

Max Tx Power Limit

NOTE: Test parameter for internal debug purposes only.

3.20.2.15 int16 host_msg_getParamRsp_t::maxTxPwrLimitHeadRoom

Max Tx Power Limit Headroom

NOTE: Test parameter for internal debug purposes only.

3.20.2.16 uint32 host_msg_getParamRsp_t::nodeId

Node ID

3.20.2.17 uint16 host_msg_getParamRsp_t::numNCAccum

Number of Chips used to correlate against Gold Code NOTE: Test parameter for internal debug purposes only.

3.20.2.18 uint16 host_msg_getParamRsp_t::pad

Reserved for future use.

3.20.2.19 int16 host_msg_getParamRsp_t::rssiMargin

Uplink Margin in dB NOTE: Test parameter for internal debug purposes only.

3.20.2.20 int8 host_msg_getParamRsp_t::slotInterval

Slot Interval (also known as Update Interval).

0 - 4.8 minutes (continuous mode) 1 - 4.8 minutes 2 - 7.2 minutes 3 - 9.6 minutes 4 - 12 minutes 5 - 24 minutes 6 - 36 minutes 7 - 48 minutes 8 - 60 minutes 9 - 120 minutes 10 - 180 minutes 11 - 240 minutes 12 - 360 minutes 13 - 480 minutes 14 - 720 minutes 15 - 1440 minutes

3.20.2.21 uint16 host_msg_getParamRsp_t::systemState

Current Over The Air Link state.

See also

sys_mgr_state_t

3.20.2.22 uint16 host_msg_getParamRsp_t::ulSpreading

Uplink Spreading Factor, stored in log(2) form.

Spreading Factor = 2^ulSpreading

valid range = 4-14 (16-8192)

NOTE: Test parameter for internal debug purposes only.

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.21 host_msg_getParams_t Struct Reference

Get Params message.

`#include <host_customer_msg.h>`

Collaboration diagram for host_msg_getParams_t:

Data Fields

- host_msg_header_t header
- uint32_t footer

3.21.1 Detailed Description

Get Params message. Used by the Host to query the Parameters of the eNode.

See also

HOST_MSG_TYPE_GET_PARAMS

3.21.2 Field Documentation

3.21.2.1 uint32_t host_msg_getParams_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.21.2.2 host_msg_header_t host_msg_getParams_t::header

2-byte Message Length followed by 2-byte Message Type

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.22 host_msg_getState_t Struct Reference

A Get State message.

```
#include <host_customer_msg.h>
```

Collaboration diagram for host_msg_getState_t:

Data Fields

- host_msg_header_t header
- uint32_t footer

3.22.1 Detailed Description

A Get State message. Used by the Host to query the current state of the eNode Over The Air link.

See also

HOST_MSG_TYPE_GET_STATE

3.22.2 Field Documentation

3.22.2.1 uint32_t host_msg_getState_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.22.2.2 host_msg_header_t host_msg_getState_t::header

2-byte Message Length followed by 2-byte Message Type

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.23 host_msg_getStateRsp_t Struct Reference

Get State response message.

`#include <host_customer_msg.h>`

Collaboration diagram for host_msg_getStateRsp_t:

Data Fields

- host_msg_header_t header
- uint32_t state
- uint32_t footer

3.23.1 Detailed Description

Get State response message. A response to the GET_STATE query, indicating the current state of the Over The Air link.

See also

HOST_MSG_TYPE_GET_STATE_RSP

3.23.2 Field Documentation

3.23.2.1 uint32_t host_msg_getStateRsp_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.23.2.2 host_msg_header_t host_msg_getStateRsp_t::header

2-byte Message Length followed by 2-byte Message Type

3.23.2.3 uint32_t host_msg_getStateRsp_t::state

Over The Air Link state.

See also

sys_mgr_state_t

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.24 host_msg_header_t Struct Reference

The host interface message header.

```
#include <host_customer_msg.h>
```

Data Fields

- uint16_t msgLen
- host_msg_type_t msgType

3.24.1 Detailed Description

The host interface message header. This header is common to all Host Interface Messages. It preceeds the payload of each message.

3.24.2 Field Documentation

3.24.2.1 uint16_t host_msg_header_t::msgLen

The length of the message in bytes, not including this header.

3.24.2.2 host_msg_type_t host_msg_header_t::msgType

The message type.

This identifies what type of Host Interface Message is being sent and the structure of the payload that follows this header.

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.25 host_msg_hostIdReq_t Struct Reference

Collaboration diagram for host_msg_hostIdReq_t:

Data Fields

- host_msg_header_t header
- uint8_t hostId [16]
- uint32_t footer

3.25.1 Field Documentation

3.25.1.1 uint32_t host_msg_hostIdReq_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.25.1.2 host_msg_header_t host_msg_hostIdReq_t::header

2-byte Message Length followed by 2-byte Message Type

3.25.1.3 uint8_t host_msg_hostIdReq_t::hostId[16]

Arbitrary 128-bit identifier.

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.26 host_msg_nodeSwUpgradeCnf_t Struct Reference

Collaboration diagram for host_msg_nodeSwUpgradeCnf_t:

Data Fields

- host_msg_header_t header
- uint32_t footer

3.26.1 Field Documentation

3.26.1.1 uint32_t host_msg_nodeSwUpgradeCnf_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.26.1.2 host_msg_header_t host_msg_nodeSwUpgradeCnf_t::header

2-byte Message Length followed by 2-byte Message Type

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.27 host_msg_nodeSwUpgradeInd_t Struct Reference

Collaboration diagram for host_msg_nodeSwUpgradeInd_t:

Data Fields

- host_msg_header_t header
- uint32_t footer

3.27.1 Field Documentation

3.27.1.1 uint32_t host_msg_nodeSwUpgradeInd_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.27.1.2 host_msg_header_t host_msg_nodeSwUpgradeInd_t::header

2-byte Message Length followed by 2-byte Message Type

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.28 host_msg_otaDiagInd_t Struct Reference

Indicates whether OTA diag mode is enabled.

```
#include <host_customer_msg.h>
```

Collaboration diagram for host_msg_otaDiagInd_t:

Data Fields

- host_msg_header_t header
- uint32_t state
- uint32_t footer

3.28.1 Detailed Description

Indicates whether OTA diag mode is enabled. Will be sent to the host whenever the state changes. When the state is enabled this indicates to the host that the node is enqueuing its own SDU's and the host's throughput and ability to queue SDU's may be degraded.

See also

HOST_MSG_TYPE_OTA_DIAG_IND

3.28.2 Field Documentation

3.28.2.1 uint32_t host_msg_otaDiagInd_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.28.2.2 host_msg_header_t host_msg_otaDiagInd_t::header

2-byte Message Length followed by 2-byte Message Type

3.28.2.3 uint32_t host_msg_otaDiagInd_t::state

The current OTA diag state.

0 = disabled, 1 = enabled.

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.29 host_msg_preUpdateNotificationInd_t Struct Reference

HOST_MSG_TYPE_PRE_UPDATE_NOTIFICATION_IND message.

`#include <host_customer_msg.h>`

Collaboration diagram for host_msg_preUpdateNotificationInd_t:

Data Fields

- host_msg_header_t header
- uint32_t **reserved**
- uint32_t footer

3.29.1 Detailed Description

HOST_MSG_TYPE_PRE_UPDATE_NOTIFICATION_IND message. Sent by the node prior to an update interval (if configured to do so).

See also

HOST_MSG_TYPE_PRE_UPDATE_NOTIFICATION_IND

3.29.2 Field Documentation

3.29.2.1 uint32_t host_msg_preUpdateNotificationInd_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.29.2.2 host_msg_header_t host_msg_preUpdateNotificationInd_t::header

2-byte Message Length followed by 2-byte Message Type

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.30 host_msg_provisionKeysReq_t Struct Reference

Provisions security keys in the node.

`#include <host_customer_msg.h>`

Collaboration diagram for host_msg_provisionKeysReq_t:

Data Fields

- host_msg_header_t header
- uint8_t rootKey [16]
- uint8_t gatewayKey [24]
- uint8_t gatewayCdldKey [16]
- uint32_t footer

3.30.1 Detailed Description

Provisions security keys in the node. After this message is received the node will lock out JTAG access.

See also

HOST_MSG_TYPE_PROVISION_KEYS_REQ

3.30.2 Field Documentation

3.30.2.1 uint32_t host_msg_provisionKeysReq_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.30.2.2 uint8_t host_msg_provisionKeysReq_t::gatewayCdldKey[16]

The gateway-wide code download 128-bit key.

3.30.2.3 uint8_t host_msg_provisionKeysReq_t::gatewayKey[24]

The gateway-wide 168-bit (+ 24 parity bits) key.

3.30.2.4 host_msg_header_t host_msg_provisionKeysReq_t::header

2-byte Message Length followed by 2-byte Message Type

3.30.2.5 uint8_t host_msg_provisionKeysReq_t::rootKey[16]

The root (node-specific) 128-bit key.

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.31 host_msg_provisionKeysRsp_t Struct Reference

Provisions security keys in the node.

`#include <host_customer_msg.h>`

Collaboration diagram for host_msg_provisionKeysRsp_t:

Data Fields

- host_msg_header_t header
- uint32_t footer

3.31.1 Detailed Description

Provisions security keys in the node.

See also

HOST_MSG_TYPE_PROVISION_KEYS_RSP

3.31.2 Field Documentation

3.31.2.1 uint32_t host_msg_provisionKeysRsp_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.31.2.2 host_msg_header_t host_msg_provisionKeysRsp_t::header

2-byte Message Length followed by 2-byte Message Type

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.32 host_msg_readFlashConf_t Struct Reference

A READ FLASH CONF message.

`#include <host_customer_msg.h>`

Collaboration diagram for host_msg_readFlashConf_t:

Data Fields

- host_msg_header_t header
- uint32_t footer

3.32.1 Detailed Description

A READ FLASH CONF message. Used by the Host to request the Configuration file data from the flash memory device.

See also

HOST_MSG_TYPE_READ_FLASH_CONF

3.32.2 Field Documentation

3.32.2.1 uint32_t host_msg_readFlashConf_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.32.2.2 host_msg_header_t host_msg_readFlashConf_t::header

2-byte Message Length followed by 2-byte Message Type

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.33 host_msg_readFlashConfRsp_t Struct Reference

A READ FLASH CONF RSP message.

`#include <host_customer_msg.h>`

Collaboration diagram for host_msg_readFlashConfRsp_t:

Data Fields

- host_msg_header_t header
- CAL_CONFIG_FlashConfig_t **config**
- uint32_t footer

3.33.1 Detailed Description

A READ FLASH CONF RSP message. Used by the eNode to report the Configuration file data from the flash device. This is in response to a HOST_MSG_TYPE_READ_FLASH_CONF message. For details of the config block contents, see the Node Provisioning Tool documentation (README.NPT.txt as a starting point).

See also

HOST_MSG_TYPE_READ_FLASH_CONF_RSP

3.33.2 Field Documentation

3.33.2.1 uint32_t host_msg_readFlashConfRsp_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.33.2.2 host_msg_header_t host_msg_readFlashConfRsp_t::header

2-byte Message Length followed by 2-byte Message Type

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.34 host_msg_rxSdu_t Struct Reference

A RX SDU message.

```
#include <host_customer_msg.h>
```

Collaboration diagram for host_msg_rxSdu_t:

Data Fields

- host_msg_header_t header
- uint16_t size
- uint16_t pad
- uint8_t payload [464]
- uint32_t footer

3.34.1 Detailed Description

A RX SDU message. Used by the eNode to inform the Host of reception of an SDU from the ULP network.

See also

HOST_MSG_TYPE_RXSDU

3.34.2 Field Documentation

3.34.2.1 uint32_t host_msg_rxSdu_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.34.2.2 host_msg_header_t host_msg_rxSdu_t::header

2-byte Message Length followed by 2-byte Message Type

3.34.2.3 uint16_t host_msg_rxSdu_t::pad

Reserved for future use.

3.34.2.4 uint8_t host_msg_rxSdu_t::payload[464]

The SDU. Variable size

3.34.2.5 uint16_t host_msg_rxSdu_t::size

The SDU size, in bytes.

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.35 host_msg_setChannel_t Struct Reference

A SET CENTER FREQ message.

`#include <host_customer_msg.h>`

Collaboration diagram for host_msg_setChannel_t:

Data Fields

- host_msg_header_t header
- uint32_t channelNum
- uint32_t footer

3.35.1 Detailed Description

A SET CENTER FREQ message. Used by the Host to specify to the eNode the Center Frequency.

See also

HOST_MSG_TYPE_SET_CHANNEL

3.35.2 Field Documentation

3.35.2.1 uint32_t host_msg_setChannel_t::channelNum

channel number.

3.35.2.2 uint32_t host_msg_setChannel_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.35.2.3 host_msg_header_t host_msg_setChannel_t::header

2-byte Message Length followed by 2-byte Message Type

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.36 host_msg_setGoldCode_t Struct Reference

A SET GOLD CODE message.

`#include <host_customer_msg.h>`

Collaboration diagram for host_msg_setGoldCode_t:

Data Fields

- host_msg_header_t header
- uint32_t bcastGoldCode
- uint32_t dataGoldCode
- uint32_t footer

3.36.1 Detailed Description

A SET GOLD CODE message. Used by the Host to specify to the eNode the Gold Codes for Broadcast and for Data.

See also

HOST_MSG_TYPE_SET_GOLD_CODES

3.36.2 Field Documentation

3.36.2.1 uint32_t host_msg_setGoldCode_t::bcastGoldCode

Broadcast gold code.

3.36.2.2 uint32_t host_msg_setGoldCode_t::dataGoldCode

Data gold code.

3.36.2.3 uint32_t host_msg_setGoldCode_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.36.2.4 host_msg_header_t host_msg_setGoldCode_t::header

2-byte Message Length followed by 2-byte Message Type

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.37 host_msg_setPreUpdateNotificationReq_t Struct Reference

HOST_MSG_TYPE_SET_PRE_UPDATE_NOTIFICATION_REQ message.

```
#include <host_customer_msg.h>
```

Collaboration diagram for host_msg_setPreUpdateNotificationReq_t:

Data Fields

- host_msg_header_t header
- uint32_t timeInMs
- uint32_t footer

3.37.1 Detailed Description

HOST_MSG_TYPE_SET_PRE_UPDATE_NOTIFICATION_REQ message. Used by the host to request to be notified prior to an update interval. This is intended to allow the host to queue an SDU in time for an upcoming update cycle while minimizing the amount of time the node must be awake.

See also

HOST_MSG_TYPE_SET_PRE_UPDATE_NOTIFICATION_REQ

3.37.2 Field Documentation

3.37.2.1 uint32_t host_msg_setPreUpdateNotificationReq_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.37.2.2 host_msg_header_t host_msg_setPreUpdateNotificationReq_t::header

2-byte Message Length followed by 2-byte Message Type

3.37.2.3 uint32_t host_msg_setPreUpdateNotificationReq_t::timeInMs

The amount of time in milliseconds before an update interval that the host will be notified. Set to 0 to disable.

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.38 host_msg_setPreUpdateNotificationRsp_t Struct Reference

HOST_MSG_TYPE_SET_PRE_UPDATE_NOTIFICATION_RSP message.

`#include <host_customer_msg.h>`

Collaboration diagram for host_msg_setPreUpdateNotificationRsp_t:

Data Fields

- host_msg_header_t header
- uint32_t result
- uint32_t footer

3.38.1 Detailed Description

HOST_MSG_TYPE_SET_PRE_UPDATE_NOTIFICATION_RSP message.

See also

HOST_MSG_TYPE_SET_PRE_UPDATE_NOTIFICATION_RSP

3.38.2 Field Documentation

3.38.2.1 uint32_t host_msg_setPreUpdateNotificationRsp_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.38.2.2 host_msg_header_t host_msg_setPreUpdateNotificationRsp_t::header

2-byte Message Length followed by 2-byte Message Type

3.38.2.3 uint32_t host_msg_setPreUpdateNotificationRsp_t::result

0 indicates the request was successful.

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.39 host_msg_setSpreading_t Struct Reference

A SET SPREADING message.

`#include <host_customer_msg.h>`

Collaboration diagram for host_msg_setSpreading_t:

Data Fields

- host_msg_header_t header
- uint32_t dlBcastSpreading
- uint32_t ulSpreading
- uint32_t footer

3.39.1 Detailed Description

A SET SPREADING message. Used by the Host to specify the Spreading Factor of the Downlink Broadcast channel and the Uplink.

See also

HOST_MSG_TYPE_SET_SPREADING

3.39.2 Field Documentation

3.39.2.1 uint32_t host_msg_setSpreading_t::dlBcastSpreading

Downlink Broadcast spreading.

3.39.2.2 uint32_t host_msg_setSpreading_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.39.2.3 host_msg_header_t host_msg_setSpreading_t::header

2-byte Message Length followed by 2-byte Message Type

3.39.2.4 uint32_t host_msg_setSpreading_t::ulSpreading

Uplink spreading.

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.40 host_msg_startFrameStats_t Struct Reference

A START FRAME STATS message.

```
#include <host_customer_msg.h>
```

Collaboration diagram for host_msg_startFrameStats_t:

Data Fields

- host_msg_header_t header
- uint32_t footer

3.40.1 Detailed Description

A START FRAME STATS message. Used by the Host to instruct the eNode to start to report frame statistics to the Host.

See also

HOST_MSG_TYPE_START_FRAME_STATS

3.40.2 Field Documentation

3.40.2.1 uint32_t host_msg_startFrameStats_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.40.2.2 host_msg_header_t host_msg_startFrameStats_t::header

2-byte Message Length followed by 2-byte Message Type

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.41 host_msg_stopFrameStats_t Struct Reference

A STOP FRAME STATS message.

```
#include <host_customer_msg.h>
```

Collaboration diagram for host_msg_stopFrameStats_t:

Data Fields

- host_msg_header_t header
- uint32_t footer

3.41.1 Detailed Description

A STOP FRAME STATS message. Used by the Host to instruct the eNode to stop reporting frame statistics to the Host.

See also

HOST_MSG_TYPE_STOP_FRAME_STATS

3.41.2 Field Documentation

3.41.2.1 uint32_t host_msg_stopFrameStats_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.41.2.2 host_msg_header_t host_msg_stopFrameStats_t::header

2-byte Message Length followed by 2-byte Message Type

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.42 host_msg_swUpgrade2BeginReq_t Struct Reference

A Begin SW Upgrade2 message.

```
#include <host_customer_msg.h>
```

Collaboration diagram for host_msg_swUpgrade2BeginReq_t:

Data Fields

- host_msg_header_t header
- uint32_t numChunks
- uint32_t checksum
- uint32_t footer

3.42.1 Detailed Description

A Begin SW Upgrade2 message. Used by the Host to start the process of upgrading the eNode software. This method is preferred to the '1' type of upgrade but is only supported on enodes with 2nd flash banks (enode r8 and later) and micronodes.

See also

HOST_MSG_TYPE_SW_UPGR2_BEGIN_REQ

3.42.2 Field Documentation

3.42.2.1 uint32_t host_msg_swUpgrade2BeginReq_t::checksum

Expected checksum over entirety of SW upgrade

3.42.2.2 uint32_t host_msg_swUpgrade2BeginReq_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.42.2.3 host_msg_header_t host_msg_swUpgrade2BeginReq_t::header

2-byte Message Length followed by 2-byte Message Type

3.42.2.4 uint32_t host_msg_swUpgrade2BeginReq_t::numChunks

Number of data chunks in the SW upgrade

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.43 host_msg_swUpgrade2BeginRsp_t Struct Reference

A Begin Software Upgrade2 Response message.

`#include <host_customer_msg.h>`

Collaboration diagram for host_msg_swUpgrade2BeginRsp_t:

Data Fields

- host_msg_header_t header
- uint32_t result
- uint32_t footer

3.43.1 Detailed Description

A Begin Software Upgrade2 Response message. This is used by the eNode to respond to a request to upgrade eNode software. The contents of this message indicate whether is it OK to continue with the software upgrade process or not.

See also

HOST_MSG_TYPE_SW_UPGR2_BEGIN_RSP

3.43.2 Field Documentation

3.43.2.1 uint32_t host_msg_swUpgrade2BeginRsp_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.43.2.2 host_msg_header_t host_msg_swUpgrade2BeginRsp_t::header

2-byte Message Length followed by 2-byte Message Type

3.43.2.3 uint32_t host_msg_swUpgrade2BeginRsp_t::result

The result of the request to begin SW upgrade process.

0 = ok, 1 = invalid state, 2 = bad size

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.44 host_msg_swUpgrade2ChunkReq_t Struct Reference

A SW upgrade chunk.

`#include <host_customer_msg.h>`

Collaboration diagram for host_msg_swUpgrade2ChunkReq_t:

Data Fields

- host_msg_header_t header
- uint32_t num
- uint32_t checksum
- uint8_t chunk [256]
- uint32_t footer

3.44.1 Detailed Description

A SW upgrade chunk.

See also

HOST_MSG_TYPE_SW_UPGR2_CHUNK_REQ

3.44.2 Field Documentation

3.44.2.1 uint32_t host_msg_swUpgrade2ChunkReq_t::checksum

Expected checksum over this chunk.

3.44.2.2 uint8_t host_msg_swUpgrade2ChunkReq_t::chunk[256]

The chunk data.

3.44.2.3 uint32_t host_msg_swUpgrade2ChunkReq_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.44.2.4 host_msg_header_t host_msg_swUpgrade2ChunkReq_t::header

2-byte Message Length followed by 2-byte Message Type

3.44.2.5 uint32_t host_msg_swUpgrade2ChunkReq_t::num

Which chunk this is.

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.45 host_msg_swUpgrade2ChunkRsp_t Struct Reference

A chunk response message.

```
#include <host_customer_msg.h>
```

Collaboration diagram for host_msg_swUpgrade2ChunkRsp_t:

Data Fields

- host_msg_header_t header
- uint32_t result
- uint32_t footer

3.45.1 Detailed Description

A chunk response message.

See also

HOST_MSG_TYPE_SW_UPGR2_CHUNK_RSP

3.45.2 Field Documentation

3.45.2.1 uint32_t host_msg_swUpgrade2ChunkRsp_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.45.2.2 host_msg_header_t host_msg_swUpgrade2ChunkRsp_t::header

2-byte Message Length followed by 2-byte Message Type

3.45.2.3 uint32_t host_msg_swUpgrade2ChunkRsp_t::result

The result of the chunk request

0 = ok, 1 = failed chunk checksum.

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.46 host_msg_swUpgrade2EndReq_t Struct Reference

Sent to end the SW upgrade and boot to the new image.

`#include <host_customer_msg.h>`

Collaboration diagram for host_msg_swUpgrade2EndReq_t:

Data Fields

- host_msg_header_t header
- uint32_t footer

3.46.1 Detailed Description

Sent to end the SW upgrade and boot to the new image.

See also

HOST_MSG_TYPE_SW_UPGR2_END_REQ

3.46.2 Field Documentation

3.46.2.1 uint32_t host_msg_swUpgrade2EndReq_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.46.2.2 host_msg_header_t host_msg_swUpgrade2EndReq_t::header

2-byte Message Length followed by 2-byte Message Type

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.47 host_msg_swUpgrade2EndRsp_t Struct Reference

Response to host_msg_chunkSwUpgrade2Req_t.

```
#include <host_customer_msg.h>
```

Collaboration diagram for host_msg_swUpgrade2EndRsp_t:

Data Fields

- host_msg_header_t header
- uint32_t result
- uint32_t footer

3.47.1 Detailed Description

Response to host_msg_chunkSwUpgrade2Req_t.

See also

HOST_MSG_TYPE_SW_UPGR2_END_RSP

3.47.2 Field Documentation

3.47.2.1 uint32_t host_msg_swUpgrade2EndRsp_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.47.2.2 host_msg_header_t host_msg_swUpgrade2EndRsp_t::header

2-byte Message Length followed by 2-byte Message Type

3.47.2.3 uint32_t host_msg_swUpgrade2EndRsp_t::result

The result of the chunk request

0 = ok, 1 = failed image checksum.

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.48 host_msg_systemSetState_t Struct Reference

A SYSTEM SET STATE message.

```
#include <host_customer_msg.h>
```

Collaboration diagram for host_msg_systemSetState_t:

Data Fields

- host_msg_header_t header
- uint32_t state
- uint32_t footer

3.48.1 Detailed Description

A SYSTEM SET STATE message. Used by the Host to tell the eNode to turn on or off its Over The Air link.

See also

HOST_MSG_TYPE_SYSTEM_SET_STATE

3.48.2 Field Documentation

3.48.2.1 uint32_t host_msg_systemSetState_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.48.2.2 host_msg_header_t host_msg_systemSetState_t::header

2-byte Message Length followed by 2-byte Message Type

3.48.2.3 uint32_t host_msg_systemSetState_t::state

Over The Air system state.

See also

host_msg_systemAirlinkState_t

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.49 host_msg_systemState_t Struct Reference

A SYSTEM STATE message.

```
#include <host_customer_msg.h>
```

Collaboration diagram for host_msg_systemState_t:

Data Fields

- host_msg_header_t header
- uint32_t state
- uint32_t footer

3.49.1 Detailed Description

A SYSTEM STATE message. Send by the eNode to indicate that a Over The Air link state change has occurred. This message is originated by the eNode and sent only when the state changes. To poll this state from the Host, use HOST_MSG_TYPE_GET_STATE and HOST_MSG_TYPE_GET_STATE_RSP messages.

See also

HOST_MSG_TYPE_SYSTEM_STATE

3.49.2 Field Documentation

3.49.2.1 uint32_t host_msg_systemState_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.49.2.2 host_msg_header_t host_msg_systemState_t::header

2-byte Message Length followed by 2-byte Message Type

3.49.2.3 uint32_t host_msg_systemState_t::state

Over The Air Link state.

See also

sys_mgr_state_t

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.50 host_msg_timeSyncReq_t Struct Reference

HOST_MSG_TYPE_TIME_SYNC_REQ message.

```
#include <host_customer_msg.h>
```

Collaboration diagram for host_msg_timeSyncReq_t:

Data Fields

- host_msg_header_t header
- uint32_t footer

3.50.1 Detailed Description

HOST_MSG_TYPE_TIME_SYNC_REQ message. Used by the Host to Request Time Synchronization with the eNode. This results in the eNode scheduling some time in the future that will be communicated back to the Host. When this time occurs, it drives a GPIO to the Host such that the eNode and Host can be approximately time aligned.

See also

HOST_MSG_TYPE_TIME_SYNC_REQ

3.50.2 Field Documentation

3.50.2.1 uint32_t host_msg_timeSyncReq_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.50.2.2 host_msg_header_t host_msg_timeSyncReq_t::header

2-byte Message Length followed by 2-byte Message Type

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.51 host_msg_timeSyncRsp_t Struct Reference

HOST_MSG_TYPE_TIME_SYNC_REQ message.

```
#include <host_customer_msg.h>
```

Collaboration diagram for host_msg_timeSyncRsp_t:

Data Fields

- host_msg_header_t header
- uint32_t time_of_day_whole
- uint32_t time_of_day_frac
- uint16_t year
- uint8_t month
- uint8_t day
- uint8_t valid
- uint8_t rsv1
- uint16_t **rsv2**
- uint32_t footer

3.51.1 Detailed Description

HOST_MSG_TYPE_TIME_SYNC_REQ message. Used by the eNode to communicate to the Host a future time stamp when the eNode will drive the TOUT signal to the Host high. The idea is that the Host will sense this TOUT rising edge and be able to use this information to be approximately time aligned with the eNode.

See also

HOST_MSG_TYPE_TIME_SYNC_RSP

3.51.2 Field Documentation

3.51.2.1 uint8_t host_msg_timeSyncRsp_t::day

The day that corresponds to the next host interrupt.

3.51.2.2 uint32_t host_msg_timeSyncRsp_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.51.2.3 host_msg_header_t host_msg_timeSyncRsp_t::header

2-byte Message Length followed by 2-byte Message Type

3.51.2.4 uint8_t host_msg_timeSyncRsp_t::month

The month that corresponds to the next host interrupt.

3.51.2.5 uint8_t host_msg_timeSyncRsp_t::rsv1

Reserved for future use

3.51.2.6 uint32_t host_msg_timeSyncRsp_t::time_of_day_frac

The time of day that corresponds to the next host interrupt in seconds, fractional part.

3.51.2.7 uint32_t host_msg_timeSyncRsp_t::time_of_day_whole

The time of day that corresponds to the next host interrupt in seconds, whole part.

Combined with the next field this forms a <17.32> fixed point format number describing the time of day in fixed point seconds.

3.51.2.8 uint8_t host_msg_timeSyncRsp_t::valid

Indicates whether the time reported is valid. If this field is zero then there will be no interrupt to the host.

3.51.2.9 uint16_t host_msg_timeSyncRsp_t::year

The year that corresponds to the next host interrupt.

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.52 host_msg_txProgrammed_t Struct Reference

A TX PROGRAMMED message.

```
#include <host_customer_msg.h>
```

Collaboration diagram for host_msg_txProgrammed_t:

Data Fields

- host_msg_header_t header
- int32 txFreqStride
- int32 txTimeTrackingStride
- int32 freqOffset
- uint16 spreading
- uint16 startingSubslot
- uint16 digitalTruncation
- uint16 txVGA
- uint16 numSubslots
- uint8 numLoggingMsgsDropped
- uint8 **pad**
- uint32_t footer

3.52.1 Detailed Description

A TX PROGRAMMED message. Used by eNode to indicate that something has been programmed to be transmitted on the radio. This message is not useful to customers and should be hidden.

See also

HOST_MSG_TYPE_TX_PROGRAMMED

3.52.2 Field Documentation

3.52.2.1 uint16 host_msg_txProgrammed_t::digitalTruncation

Number of 6dB adjustments for TX AFC

3.52.2.2 uint32_t host_msg_txProgrammed_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.52.2.3 int32 host_msg_txProgrammed_t::freqOffset

Absolute Frequency Offset

3.52.2.4 host_msg_header_t host_msg_txProgrammed_t::header

2-byte Message Length followed by 2-byte Message Type

3.52.2.5 uint8 host_msg_txProgrammed_t::numLoggingMsgsDropped

Count of dropped messages

3.52.2.6 uint16 host_msg_txProgrammed_t::numSubslots

number of subslots

3.52.2.7 uint16 host_msg_txProgrammed_t::spreading

Tx Spreading Factor, stored in log(2) form.

Spreading Factor = 2^spreading

valid range = 4-14 (16-8192)

3.52.2.8 uint16 host_msg_txProgrammed_t::startingSubslot

Last Tx Subslot

3.52.2.9 int32 host_msg_txProgrammed_t::txFreqStride

TX AFC Frequency Stride

3.52.2.10 int32 host_msg_txProgrammed_t::txTimeTrackingStride

Time Tracking Stride

3.52.2.11 uint16 host_msg_txProgrammed_t::txVGA

TX VGA

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.53 host_msg_txSdu_t Struct Reference

A TX SDU message.

```
#include <host_customer_msg.h>
```

Collaboration diagram for host_msg_txSdu_t:

Data Fields

- host_msg_header_t header
- uint16_t size
- uint16_t host_tag
- host_msg_sduFlags_t flags
- uint16_t pad
- uint8_t payload [464]
- uint32_t footer

3.53.1 Detailed Description

A TX SDU message. Used by the Host to command the eNode to transmit an SDU on the ULP network.

See also

HOST_MSG_TYPE_TXSDU

3.53.2 Field Documentation

3.53.2.1 host_msg_sduFlags_t host_msg_txSdu_t::flags

Delivery options.

3.53.2.2 uint32_t host_msg_txSdu_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.53.2.3 host_msg_header_t host_msg_txSdu_t::header

2-byte Message Length followed by 2-byte Message Type

3.53.2.4 uint16_t host_msg_txSdu_t::host_tag

Arbitrary, host-chosen identifier.

An identifier that the Host passes, which can be used to correlate the responses.

3.53.2.5 uint16_t host_msg_txSdu_t::pad

Reserved for future use.

3.53.2.6 uint8_t host_msg_txSdu_t::payload[464]

The SDU.

3.53.2.7 uint16_t host_msg_txSdu_t::size

The SDU size, in bytes.

Includes the number of bytes in the SDU buffer. SDU size must be in multiples of 8, with a maximum of HOST_MSG_MAX_SDU_SIZE bytes.

Note also that in the 1.2 system, best effort SDUs must be exactly 8 bytes in length, and are not supported in security enabled networks.

See also

HOST_MSG_MIN_SDU_SIZE
HOST_MSG_MAX_SDU_SIZE

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.54 host_msg_txSduResult_t Struct Reference

A TX SDU Result message.

`#include <host_customer_msg.h>`

Collaboration diagram for host_msg_txSduResult_t:

Data Fields

- host_msg_header_t header
- uint16_t host_tag
- host_msg_txsdu_result_sdustatus_t sduStatus
- uint32_t footer

3.54.1 Detailed Description

A TX SDU Result message. Used by the eNode to notify the eHost of the status of an SDU to be transmitted. This is sent shortly after the eNode receives a TX-SDU command.

See also

HOST_MSG_TYPE_TXSDU_RESULT

3.54.2 Field Documentation

3.54.2.1 uint32_t host_msg_txSduResult_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.54.2.2 host_msg_header_t host_msg_txSduResult_t::header

2-byte Message Length followed by 2-byte Message Type

3.54.2.3 uint16_t host_msg_txSduResult_t::host_tag

Arbitrary, host-chosen identifier.

An identifier that the Host originally has associated with a particular SDU.

3.54.2.4 host_msg_txsdu_result_sdustatus_t host_msg_txSduResult_t::sduStatus

indicates the delivery status of an sdu.

A bitmap that is used to indicate the status of an SDU that was sent to the eNode to be transmitted.

See also

HOST_MSG_SDU_STATUS_BITS_TRANSMITTED
HOST_MSG_SDU_STATUS_BITS_ACK_SUCCESS
HOST_MSG_SDU_STATUS_BITS_ACK_FAIL
HOST_MSG_SDU_STATUS_BITS_REPLACED
HOST_MSG_SDU_STATUS_BITS_BUFFER_FULL
HOST_MSG_SDU_STATUS_BITS_OTHER_ERROR
HOST_MSG_SDU_STATUS_BITS_DROPPED_NET_EXIT
HOST_MSG_SDU_STATUS_BITS_DROPPED_HOST
HOST_MSG_SDU_STATUS_BITS_DROPPED_MAINTENANCE
HOST_MSG_SDU_STATUS_BITS_DROPPED_CDLD
HOST_MSG_SDU_STATUS_BITS_DROPPED_NOT_JOINED

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.55 host_msg_txSduRsp_t Struct Reference

A TX SDU RSP message.

```
#include <host_customer_msg.h>
```

Collaboration diagram for host_msg_txSduRsp_t:

Data Fields

- host_msg_header_t header
- uint16_t host_tag
- uint16_t isEnqueued
- uint32_t footer

3.55.1 Detailed Description

A TX SDU RSP message. Used by the eNode to notify the eHost of the status of an SDU to be transmitted. This is sent shortly after the eNode receives a TX-SDU command.

See also

HOST_MSG_TYPE_TXSDU_RSP

3.55.2 Field Documentation

3.55.2.1 uint32_t host_msg_txSduRsp_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.55.2.2 host_msg_header_t host_msg_txSduRsp_t::header

2-byte Message Length followed by 2-byte Message Type

3.55.2.3 uint16_t host_msg_txSduRsp_t::host_tag

Arbitrary, host-chosen identifier.

An identifier that the Host originally has associated with a particular SDU.

3.55.2.4 uint16_t host_msg_txSduRsp_t::isEnqueued

bool is true if last request was enqueued.

Indicates whether the SDU was enqueued. True means that the SDU was enqueued, False means that the SDU was not enqueued. In the event that it was not enqueued, the TXSDU_RESULT will contain more information on the reason that it was not.

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.56 host_msg_uptimeStatsReq_t Struct Reference

A Uptime Stats request message.

`#include <host_customer_msg.h>`

Collaboration diagram for host_msg_uptimeStatsReq_t:

Data Fields

- host_msg_header_t header
- uint32_t footer

3.56.1 Detailed Description

A Uptime Stats request message. Used by the Host to query the eNode for Uptime Statistics. These statistics include data on watchdog and amount of time since last boot.

See also

HOST_MSG_TYPE_UPTIME_STATS_REQ

3.56.2 Field Documentation

3.56.2.1 uint32_t host_msg_uptimeStatsReq_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.56.2.2 host_msg_header_t host_msg_uptimeStatsReq_t::header

2-byte Message Length followed by 2-byte Message Type

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.57 host_msg_uptimeStatsRsp_t Struct Reference

A Uptime Stats response message.

```
#include <host_customer_msg.h>
```

Collaboration diagram for host_msg_uptimeStatsRsp_t:

Data Fields

- host_msg_header_t header
- uint32_t numWdogResets
- uint32_t lastBootWasWatchdog
- uint32_t secondsSinceLastBoot
- uint32_t footer

3.57.1 Detailed Description

A Uptime Stats response message. Used by the eNode to report Uptime Statistics to the Host. This is in reponse to a HOST_MSG_TYPE_UPTIME_STATS_REQ query from the Host.

See also

HOST_MSG_TYPE_UPTIME_STATS_RSP

3.57.2 Field Documentation

3.57.2.1 uint32_t host_msg_uptimeStatsRsp_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.57.2.2 host_msg_header_t host_msg_uptimeStatsRsp_t::header

2-byte Message Length followed by 2-byte Message Type

3.57.2.3 uint32_t host_msg_uptimeStatsRsp_t::lastBootWasWatchdog

The last boot was the result of a watchdog reset.

3.57.2.4 uint32_t host_msg_uptimeStatsRsp_t::numWdogResets

The number of watchdog resets since the node was deployed.

3.57.2.5 uint32_t host_msg_uptimeStatsRsp_t::secondsSinceLastBoot

How long, in seconds, since the node last booted.

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.58 host_msg_version_t Struct Reference

A VERSION message.

`#include <host_customer_msg.h>`

Collaboration diagram for host_msg_version_t:

Data Fields

- host_msg_header_t header
- uint32_t footer

3.58.1 Detailed Description

A VERSION message. Used by the Host to request the version information from the eNode.

See also

HOST_MSG_TYPE_VERSION

3.58.2 Field Documentation

3.58.2.1 uint32_t host_msg_version_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.58.2.2 host_msg_header_t host_msg_version_t::header

2-byte Message Length followed by 2-byte Message Type

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.59 host_msg_versionRsp_t Struct Reference

A VERSION RSP message.

```
#include <host_customer_msg.h>
```

Collaboration diagram for host_msg_versionRsp_t:

Data Fields

- host_msg_header_t header
- uint32_t swRev
- uint32_t phyRev
- uint32_t footer

3.59.1 Detailed Description

A VERSION RSP message. Used by the eNode to report to the Host the version information.

See also

HOST_MSG_TYPE_VERSION_RSP

3.59.2 Field Documentation

3.59.2.1 uint32_t host_msg_versionRsp_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.59.2.2 host_msg_header_t host_msg_versionRsp_t::header

2-byte Message Length followed by 2-byte Message Type

3.59.2.3 uint32_t host_msg_versionRsp_t::phyRev

Phy revision.

3.59.2.4 uint32_t host_msg_versionRsp_t::swRev

Software revision.

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.60 host_msg_writeFlashConf_t Struct Reference

A WRITE FLASH CONF message.

```
#include <host_customer_msg.h>
```

Collaboration diagram for host_msg_writeFlashConf_t:

Data Fields

- host_msg_header_t header
- CAL_CONFIG_FlashConfig_t **config**
- uint32_t footer

3.60.1 Detailed Description

A WRITE FLASH CONF message. Used by the Host to instruct the eNode to write a new Configuration file to the flash memory device. Caution should be exercised when sending this message as this destroys all old Configuration file values and overwrites them with the values specified in this message.

See also

HOST_MSG_TYPE_WRITE_FLASH_CONF

3.60.2 Field Documentation

3.60.2.1 uint32_t host_msg_writeFlashConf_t::footer

This footer will be placed, word aligned, following the variable sized payload.

See also

HOST_MSG_END_MARKER

3.60.2.2 host_msg_header_t host_msg_writeFlashConf_t::header

2-byte Message Length followed by 2-byte Message Type

The documentation for this struct was generated from the following file:

- host_customer_msg.h

3.61 SpiProtoCmd Struct Reference

Two byte SPI transfer header - see doc "SPI slave node interface".

```
#include <spi_common_proto.h>
```

Data Fields

- uint8 **byte_1**
- uint8 **byte_2**

3.61.1 Detailed Description

Two byte SPI transfer header - see doc "SPI slave node interface".

The documentation for this struct was generated from the following file:

- spi_common_proto.h

Chapter 4

File Documentation

4.1 cal_config.h File Reference

Structures defining the layout of flash calibration and configuration tables.

Data Structures

- struct CAL_CONFIG_FlashConfig_t
- struct CAL_CONFIG_FlashCalibration_t

Defines

- #define **CAL_CONFIG_CAL_STRUCT_VERSION** 2
- #define **CAL_CONFIG_CONFIG_STRUCT_VERSION** 3
- #define **CAL_CONFIG_NUM_SCAN_SYSTEMS** (6∗4)
- #define **CAL_CONFIG_LAST_VALID_CHANNEL** 50

4.1.1 Detailed Description

Structures defining the layout of flash calibration and configuration tables. ==

DESCRIPTION: Used to parse blocks from flash, and for constructing set/get host messages for updating during calibration & commissioning.

Copyright 2010 OnRamp Wireless, Inc.

==

4.2 host_customer_msg.h File Reference

Host interface messaging interface for customer.

```
#include "system.h"
```

Include dependency graph for host_customer_msg.h:This graph shows which files directly or indirectly include this file:

Data Structures

- struct host_msg_header_t

  The host interface message header.

- struct host_msg_txSdu_t

  A TX SDU message.

- struct host_msg_txSduRsp_t

  A TX SDU RSP message.

- struct host_msg_txSduResult_t

  A TX SDU Result message.

- struct host_msg_flushTxSduQueue_t

  Requests all queued uplink sdu to be dropped.

- struct host_msg_flushTxSduQueueRsp_t

  Relays the result from a flush TXSDU queue message.

- struct host_msg_rxSdu_t

  A RX SDU message.

- struct host_msg_startFrameStats_t

  A START FRAME STATS message.

- struct host_msg_stopFrameStats_t

  A STOP FRAME STATS message.

- struct host_msg_uptimeStatsReq_t

  A Uptime Stats request message.

- struct host_msg_uptimeStatsRsp_t

  A Uptime Stats response message.

- struct host_msg_getExceptionBufferReq_t

  A Get Exception Buffer request message.

- struct host_msg_getExceptionBufferRsp_t

  A Get Exception Buffer response message.

- struct host_msg_getState_t

A Get State message.

- struct host_msg_getStateRsp_t

  Get State response message.

- struct host_msg_timeSyncReq_t

  HOST_MSG_TYPE_TIME_SYNC_REQ message.

- struct host_msg_timeSyncRsp_t

  HOST_MSG_TYPE_TIME_SYNC_REQ message.

- struct host_msg_setPreUpdateNotificationReq_t

  HOST_MSG_TYPE_SET_PRE_UPDATE_NOTIFICATION_REQ message.

- struct host_msg_setPreUpdateNotificationRsp_t

  HOST_MSG_TYPE_SET_PRE_UPDATE_NOTIFICATION_RSP message.

- struct host_msg_preUpdateNotificationInd_t

  HOST_MSG_TYPE_PRE_UPDATE_NOTIFICATION_IND message.

- struct host_msg_getParamRsp_t

  A Get Params Rsp message.

- struct host_msg_frameStats_t

  A Frame Statistics Message.

- struct host_msg_txProgrammed_t

  A TX PROGRAMMED message.

- struct host_msg_systemSetState_t

  A SYSTEM SET STATE message.

- struct host_msg_systemState_t

  A SYSTEM STATE message.

- struct host_msg_getParams_t

  Get Params message.

- struct host_msg_ack_t

  An ACK message.

- struct host_msg_err_t

  An ERR message.

- struct host_msg_connect_t

  A CONNECT message.

- struct host_msg_setSpreading_t

  A SET SPREADING message.

- struct host_msg_setGoldCode_t

  A SET GOLD CODE message.
- struct host_msg_setChannel_t

  A SET CENTER FREQ message.
- struct host_msg_version_t

  A VERSION message.
- struct host_msg_versionRsp_t

  A VERSION RSP message.
- struct host_msg_readFlashConf_t

  A READ FLASH CONF message.
- struct host_msg_readFlashConfRsp_t

  A READ FLASH CONF RSP message.
- struct host_msg_writeFlashConf_t

  A WRITE FLASH CONF message.
- struct host_msg_beginSwUpgrade_t

  A Begin SW Upgrade message.
- struct host_msg_beginSwUpgradeRsp_t

  A Begin Software Upgrade Response message.
- struct host_msg_swUpgrade2BeginReq_t

  A Begin SW Upgrade2 message.
- struct host_msg_swUpgrade2BeginRsp_t

  A Begin Software Upgrade2 Response message.
- struct host_msg_swUpgrade2ChunkReq_t

  A SW upgrade chunk.
- struct host_msg_swUpgrade2ChunkRsp_t

  A chunk response message.
- struct host_msg_swUpgrade2EndReq_t

  Sent to end the SW upgrade and boot to the new image.
- struct host_msg_swUpgrade2EndRsp_t

  Response to host_msg_chunkSwUpgrade2Req_t.
- struct host_msg_otaDiagInd_t

  Indicates whether OTA diag mode is enabled.
- struct host_msg_provisionKeysReq_t

  Provisions security keys in the node.

- struct host_msg_provisionKeysRsp_t

  Provisions security keys in the node.

- struct host_msg_blackoutStartInd_t
- struct host_msg_blackoutEndInd_t
- struct host_msg_broadcastStartInd_t
- struct host_msg_broadcastStartCnf_t
- struct host_msg_broadcastEndInd_t
- struct host_msg_broadcastDataReq_t
- struct host_msg_broadcastDataRsp_t
- struct host_msg_nodeSwUpgradeInd_t
- struct host_msg_nodeSwUpgradeCnf_t
- struct host_msg_hostIdReq_t

Defines

- #define HOST_MSG_MIN_SDU_SIZE 8
- #define HOST_MSG_MAX_SDU_SIZE 464
- #define HOST_MSG_MAX_HOST_INTF_SDU_SIZE HOST_MSG_MAX_SDU_SIZE
- #define HOST_MSG_OVERHEAD_LEN (sizeof(host_msg_header_t) + 4)
- #define HOST_MSG_END_MARKER 0xA5A5F0F0
- #define HOST_MSG_DIR_HOST_TO_NODE 0x4000
- #define HOST_MSG_DIR_NODE_TO_HOST 0x0000
- #define HOST_MSG_SDU_STATUS_BITS_TRANSMITTED (1<<0)
- #define HOST_MSG_SDU_STATUS_BITS_ACK_SUCCESS (1<<1)
- #define HOST_MSG_SDU_STATUS_BITS_ACK_FAIL (1<<2)
- #define HOST_MSG_SDU_STATUS_BITS_BUFFER_FULL (1<<4)
- #define HOST_MSG_SDU_STATUS_BITS_OTHER_ERROR (1<<5)
- #define HOST_MSG_SDU_STATUS_BITS_DROPPED_NET_EXIT (1<<6)
- #define HOST_MSG_SDU_STATUS_BITS_DROPPED_HOST (1<<7)
- #define HOST_MSG_SDU_STATUS_BITS_DROPPED_MAINTENANCE (1<<8)
- #define HOST_MSG_SDU_STATUS_BITS_DROPPED_CDLD (1<<9)
- #define HOST_MSG_SDU_STATUS_BITS_DROPPED_NOT_JOINED (1<<10)

Typedefs

- typedef uint16_t host_msg_txsdu_result_sdustatus_t

  Results of a TX SDU process.

Enumerations

- enum host_msg_sduFlags_t { HOST_MSG_SDU_FLAGS_ACKED = (1 << 3), **HOST_MSG_-SDU_FLAGS_MAKE_TWO_BYTES_LONG** = 65535 }

  SDU delivery options.

- enum host_msg_host_t {

  HOST_MSG_HOST_NULL, HOST_MSG_HOST_UART, HOST_MSG_HOST_SPI, HOST_-MSG_HOST_INTERNAL,

  HOST_MSG_HOST_OTA_DIAG, HOST_MSG_HOST_CDLD, HOST_MSG_HOST_MAC_-INTERNAL }

  The various host interfaces we support.

- enum host_msg_joinType_t { HOST_MSG_JOIN_NORMAL = 0, HOST_MSG_JOIN_TEST = 0xF }
- enum host_msg_joinBackoffType_t { HOST_MSG_JOIN_BACKOFF_TYPE_NONE, HOST_-MSG_JOIN_BACKOFF_TYPE_RAND_10_20 }
- enum host_msg_type_t {

  HOST_MSG_TYPE_START_FRAME_STATS = 0x4000 | 0x00, HOST_MSG_TYPE_STOP_-FRAME_STATS = 0x4000 | 0x01, HOST_MSG_TYPE_FRAME_STATS = 0x0000 | 0x02, HOST_-MSG_TYPE_TX_PROGRAMMED = 0x0000 | 0x03,

  HOST_MSG_TYPE_UPTIME_STATS_REQ = 0x4000 | 0x04, HOST_MSG_TYPE_UPTIME_-STATS_RSP = 0x0000 | 0x04, HOST_MSG_TYPE_GET_EXCEPTION_BUFFER_REQ = 0x4000 | 0x05, HOST_MSG_TYPE_GET_EXCEPTION_BUFFER_RSP = 0x0000 | 0x05,

  HOST_MSG_TYPE_SYSTEM_SET_STATE = 0x4000 | 0x10, HOST_MSG_TYPE_SYSTEM_-STATE = 0x0000 | 0x11, HOST_MSG_TYPE_SET_SPREADING = 0x4000 | 0x12, HOST_MSG_-TYPE_SET_GOLD_CODES = 0x4000 | 0x13,

  HOST_MSG_TYPE_SET_CHANNEL = 0x4000 | 0x14, HOST_MSG_TYPE_VERSION = 0x4000 | 0x15, HOST_MSG_TYPE_VERSION_RSP = 0x0000 | 0x15, HOST_MSG_TYPE_GET_-PARAMS = 0x4000 | 0x16,

  HOST_MSG_TYPE_GET_PARAMS_RSP = 0x0000 | 0x16, HOST_MSG_TYPE_READ_-FLASH_CONF = 0x4000 | 0x17, HOST_MSG_TYPE_READ_FLASH_CONF_RSP = 0x0000 | 0x17, HOST_MSG_TYPE_WRITE_FLASH_CONF = 0x4000 | 0x18,

  HOST_MSG_TYPE_GET_STATE = 0x4000 | 0x19, HOST_MSG_TYPE_GET_STATE_RSP = 0x0000 | 0x19, HOST_MSG_TYPE_BEGIN_SW_UPGR = 0x4000 | 0x1A, HOST_MSG_TYPE_-BEGIN_SW_UPGR_RSP = 0x0000 | 0x1A,

  HOST_MSG_TYPE_OTA_DIAG_IND = 0x0000 | 0x1B, HOST_MSG_TYPE_PROVISION_-KEYS_REQ = 0x4000 | 0x1C, HOST_MSG_TYPE_PROVISION_KEYS_RSP = 0x0000 | 0x1C, HOST_MSG_TYPE_SW_UPGR2_BEGIN_REQ = 0x4000 | 0x1D,

  HOST_MSG_TYPE_SW_UPGR2_BEGIN_RSP = 0x0000 | 0x1D, HOST_MSG_TYPE_SW_-UPGR2_CHUNK_REQ = 0x4000 | 0x1E, HOST_MSG_TYPE_SW_UPGR2_CHUNK_RSP = 0x0000 | 0x1E, HOST_MSG_TYPE_SW_UPGR2_END_REQ = 0x4000 | 0x1F,

  HOST_MSG_TYPE_SW_UPGR2_END_RSP = 0x0000 | 0x1F, HOST_MSG_TYPE_TXSDU = 0x4000 | 0x20, HOST_MSG_TYPE_TXSDU_RSP = 0x0000 | 0x20, HOST_MSG_TYPE_-TXSDU_RESULT = 0x0000 | 0x21,

  HOST_MSG_TYPE_RXSDU = 0x0000 | 0x22, HOST_MSG_TYPE_FLUSH_TXSDU_QUEUE = 0x4000 | 0x23, HOST_MSG_TYPE_FLUSH_TXSDU_QUEUE_RSP = 0x0000 | 0x23, HOST_-MSG_TYPE_ACK = 0x0000 | 0x30,

  HOST_MSG_TYPE_ERR = 0x0000 | 0x31, HOST_MSG_TYPE_CONNECT = 0x4000 | 0x32, HOST_MSG_TYPE_TIME_SYNC_REQ = 0x4000 | 0x33, HOST_MSG_TYPE_TIME_SYNC_-RSP = 0x0000 | 0x33,

  HOST_MSG_TYPE_SET_PRE_UPDATE_NOTIFICATION_REQ = 0x4000 | 0x34, HOST_-MSG_TYPE_SET_PRE_UPDATE_NOTIFICATION_RSP = 0x0000 | 0x34, HOST_MSG_TYPE_-PRE_UPDATE_NOTIFICATION_IND = 0x0000 | 0x35, HOST_MSG_TYPE_BLACKOUT_-START_IND = 0x0000 | 0x40, HOST_MSG_TYPE_BLACKOUT_END_IND = 0x0000 | 0x41, HOST_MSG_TYPE_-BROADCAST_START_IND = 0x0000 | 0x42, HOST_MSG_TYPE_BROADCAST_START_CNF = 0x4000 | 0x42, HOST_MSG_TYPE_BROADCAST_END_IND = 0x0000 | 0x43,

HOST_MSG_TYPE_BROADCAST_DATA_REQ = 0x4000 | 0x44, HOST_MSG_TYPE_-BROADCAST_DATA_RSP = 0x0000 | 0x44, HOST_MSG_TYPE_NODE_SW_UPGRADE_IND = 0x0000 | 0x45, HOST_MSG_TYPE_NODE_SW_UPGRADE_CNF = 0x4000 | 0x45,

HOST_MSG_TYPE_SET_HOST_ID_REQ = 0x4000 | 0x46, **HOST_MSG_TYPE_SIZE** = 65535 }

- enum host_msg_errCode_t { HOST_MSG_ERR_INVALID_CMD = 1 }

  The host command error codes we support.

- enum host_msg_frameStatsType_t { HOST_MSG_FRAME_STATS_WARM_DEMOD, HOST_-MSG_FRAME_STATS_PREAMBLE, HOST_MSG_FRAME_STATS_PREAMBLE_PLUS_-WARM_DEMOD }

  Type of demodulation.

- enum host_msg_broadcastStatus_t { HOST_MSG_BROADCAST_STATUS_SUCCESS, HOST_-MSG_BROADCAST_STATUS_FAILURE_OUT_OF_RANGE, HOST_MSG_BROADCAST_-STATUS_FAILURE_INVALID_BCAST_ID, **HOST_MSG_BROADCAST_STATUS_MAKE_-TWO_BYTES_LONG** = 65535 }

  Status codes for the Broadcast Data Response.

- enum host_msg_systemAirlinkState_t { **HOST_MSG_SYSTEM_SET_STATE_AIRLINK_OFF**, **HOST_MSG_SYSTEM_SET_STATE_AIRLINK_ON** }
- enum sys_mgr_state_t {

  SYS_MGR_STATE_NIL, SYS_MGR_STATE_STARTUP, SYS_MGR_STATE_IDLE, SYS_-MGR_STATE_SCANNING,

  SYS_MGR_STATE_TRACK, SYS_MGR_STATE_JOINED }

4.2.1 Detailed Description

Host interface messaging interface for customer.

4.2.2 Define Documentation

4.2.2.1 #define HOST_MSG_DIR_HOST_TO_NODE 0x4000

Bit 30 is used to indicate the direction of the Host Interface Message: 1=Host to Node, 0=Node to Host

4.2.2.2 #define HOST_MSG_DIR_NODE_TO_HOST 0x0000

Bit 30 is used to indicate the direction of the Host Interface Message: 1=Host to Node, 0=Node to Host

4.2.2.3 #define HOST_MSG_END_MARKER 0xA5A5F0F0

The constant trailing sequence that is at the end of each Host Interface Message

4.2.2.4 #define HOST_MSG_MAX_HOST_INTF_SDU_SIZE HOST_MSG_MAX_SDU_SIZE

Maximum size of SDU in Host Interface Message

4.2.2.5 #define HOST_MSG_MAX_SDU_SIZE 464

Size in bytes of maximum sized SDU.

4.2.2.6 #define HOST_MSG_MIN_SDU_SIZE 8

Size in bytes of minimum sized SDU

4.2.2.7 #define HOST_MSG_OVERHEAD_LEN (sizeof(host_msg_header_t) + 4)

Size of overhead of each Host Interface Message.

The overhead size includes the header and the footer.

4.2.2.8 #define HOST_MSG_SDU_STATUS_BITS_ACK_FAIL (1<<2)

SDU Status = ACK Fail. If this bit is set, then the eNode requested an acknowledgement of SDU reception by the network, but this acknowledgement has not been received by the eNode.

4.2.2.9 #define HOST_MSG_SDU_STATUS_BITS_ACK_SUCCESS (1<<1)

SDU Status = ACK Success.

If this bit is set, then the SDU has been acknowledged as being received by the network.

4.2.2.10 #define HOST_MSG_SDU_STATUS_BITS_BUFFER_FULL (1<<4)

SDU Status = Buffer Full. If this bit is set, then the transmit buffer is full and the SDU was not queued.

4.2.2.11 #define HOST_MSG_SDU_STATUS_BITS_DROPPED_CDLD (1<<9)

SDU Status = Dropped Due Code Download. If this bit is set, a code download mode has caused this SDU to be dropped.

4.2.2.12 #define HOST_MSG_SDU_STATUS_BITS_DROPPED_HOST (1<<7)

SDU Status = Dropped Due To Host Request. If this bit is set, a request from the host has caused this SDU to be dropped.

4.2.2.13 #define HOST_MSG_SDU_STATUS_BITS_DROPPED_MAINTENANCE (1<<8)

SDU Status = Dropped Due Maintenance. If this bit is set, a maintenance mode has caused this SDU to be dropped.

4.2.2.14 #define HOST_MSG_SDU_STATUS_BITS_DROPPED_NET_EXIT (1<<6)

SDU Status = Dropped Due To Network Exit. If this bit is set, a loss of association with an AP has caused this SDU to be dropped.

4.2.2.15 #define HOST_MSG_SDU_STATUS_BITS_DROPPED_NOT_JOINED (1<<10)

SDU Status = Dropped Due No Network Conecetvity. If this bit is set, the node not being in the joined state has caused this SDU to be dropped.

4.2.2.16 #define HOST_MSG_SDU_STATUS_BITS_OTHER_ERROR (1<<5)

SDU Status = Other Error. If this bit is set, then some other event caused this SDU to be dropped.

4.2.2.17 #define HOST_MSG_SDU_STATUS_BITS_TRANSMITTED (1<<0)

SDU Status = Transmission success.

If this bit is set, then the SDU has been transmitted over the air.

4.2.3 Typedef Documentation

4.2.3.1 typedef uint16_t host_msg_txsdu_result_sdustatus_t

Results of a TX SDU process.

Returned in a TX SDU Result message to report the outcome of the TX request. A bitfield of status indicators chosen from:

See also

HOST_MSG_SDU_STATUS_BITS_TRANSMITTED
HOST_MSG_SDU_STATUS_BITS_ACK_SUCCESS
HOST_MSG_SDU_STATUS_BITS_ACK_FAIL
HOST_MSG_SDU_STATUS_BITS_BUFFER_FULL
HOST_MSG_SDU_STATUS_BITS_OTHER_ERROR
HOST_MSG_SDU_STATUS_BITS_DROPPED_NET_EXIT
HOST_MSG_SDU_STATUS_BITS_DROPPED_HOST
HOST_MSG_SDU_STATUS_BITS_DROPPED_MAINTENANCE
HOST_MSG_SDU_STATUS_BITS_DROPPED_CDLD
HOST_MSG_SDU_STATUS_BITS_DROPPED_NOT_JOINED

4.2.4 Enumeration Type Documentation

4.2.4.1 enum host_msg_broadcastStatus_t

Status codes for the Broadcast Data Response.

Enumerator:

HOST_MSG_BROADCAST_STATUS_SUCCESS	Request successful, data is valid.

HOST_MSG_BROADCAST_STATUS_FAILURE_OUT_OF_RANGE Request failure: offset + length is out of valid range.

HOST_MSG_BROADCAST_STATUS_FAILURE_INVALID_BCAST_ID Request failure: bcast id is invalid.

4.2.4.2 enum host_msg_errCode_t

The host command error codes we support.

Used in ERROR message to specify which error detected.

See also

HOST_MSG_TYPE_ERR
host_msg_err_t

Enumerator:

HOST_MSG_ERR_INVALID_CMD The SDU should be acked (not best effort).

4.2.4.3 enum host_msg_frameStatsType_t

Type of demodulation.

See also

host_msg_frameStats_t

Enumerator:

HOST_MSG_FRAME_STATS_WARM_DEMOD Warm Demod

HOST_MSG_FRAME_STATS_PREAMBLE Preamble

HOST_MSG_FRAME_STATS_PREAMBLE_PLUS_WARM_DEMOD Preamble plus Warm Demod

4.2.4.4 enum host_msg_host_t

The various host interfaces we support.

Used in CONNECT message to specify how the Host and eNode communicate.

See also

HOST_MSG_TYPE_CONNECT
host_msg_connect_t

Enumerator:

HOST_MSG_HOST_NULL No connection between Host/eNode.

HOST_MSG_HOST_UART The UART serial control interface.

HOST_MSG_HOST_SPI The SPI slave interface.

HOST_MSG_HOST_INTERNAL The onboard host interface.

HOST_MSG_HOST_OTA_DIAG The OTA diagnostics.

HOST_MSG_HOST_CDLD The code download.

HOST_MSG_HOST_MAC_INTERNAL Join more, etc.

4.2.4.5 enum host_msg_joinBackoffType_t

Enumerator:

HOST_MSG_JOIN_BACKOFF_TYPE_NONE Use this for small-network, high-mobility deployments.

HOST_MSG_JOIN_BACKOFF_TYPE_RAND_10_20 Use this for large deployments.

4.2.4.6 enum host_msg_joinType_t

Enumerator:

HOST_MSG_JOIN_NORMAL CMAC 32 authentication - no diagnostic on rejoin reason

HOST_MSG_JOIN_TEST CMAC 28 authentication - includes reason for rejoin. Recommended.

4.2.4.7 enum host_msg_sduFlags_t

SDU delivery options.

A bitmap of use to specify details about the type of SDU.

Enumerator:

HOST_MSG_SDU_FLAGS_ACKED The SDU should be acked (not best effort).

4.2.4.8 enum host_msg_systemAirlinkState_t

Turn the Over The Air Link On or Off.

Indicates whether the Air Link should be On of Off. Used for message HOST_MSG_TYPE_SYSTEM_-SET_STATE.

See also

HOST_MSG_TYPE_SYSTEM_SET_STATE
host_msg_systemSetState_t

4.2.4.9 enum host_msg_type_t

The opcode of a Host Interface Message.

This value is used to specify what message is being communicated over Host Interface. This message opcode is used to determine the format of the data in the rest of the Host Interface Message.

See also

host_msg_header_t::msgType;

Enumerator:

HOST_MSG_TYPE_START_FRAME_STATS From Host to Node: enables frame stats that are sent by Node

HOST_MSG_TYPE_STOP_FRAME_STATS From Host to Node: disables frame stats that are sent by Node

HOST_MSG_TYPE_FRAME_STATS From Node to Host: contains statistics for debug use

HOST_MSG_TYPE_TX_PROGRAMMED Informs Host that a TX is programmed to be sent over the air

HOST_MSG_TYPE_UPTIME_STATS_REQ From Host to Node: uptime message request

HOST_MSG_TYPE_UPTIME_STATS_RSP From Node to Host: uptime message request response

HOST_MSG_TYPE_GET_EXCEPTION_BUFFER_REQ From Host to Node: exception buffer message request

HOST_MSG_TYPE_GET_EXCEPTION_BUFFER_RSP From Node to Host: exception buffer message request response

HOST_MSG_TYPE_SYSTEM_SET_STATE Host to Node: turn Node Over-The-Air interface On or Off

HOST_MSG_TYPE_SYSTEM_STATE Node to Host: current status of Node (off, scanning, or tracking)

HOST_MSG_TYPE_SET_SPREADING Host to Node: set Broadcast Spreading Factor

HOST_MSG_TYPE_SET_GOLD_CODES Host to Node: set Broadcast Gold Code

HOST_MSG_TYPE_SET_CHANNEL Host to Node: set channel that Node tracks to

HOST_MSG_TYPE_VERSION Host to Node: request Software and Hardware version

HOST_MSG_TYPE_VERSION_RSP Node to Host: contains Software and Hardware version

HOST_MSG_TYPE_GET_PARAMS Host to Node: request configuration parameters

HOST_MSG_TYPE_GET_PARAMS_RSP Node to Host: contains configuration parameters

HOST_MSG_TYPE_READ_FLASH_CONF Host to Node: request flash configuration

HOST_MSG_TYPE_READ_FLASH_CONF_RSP Node to Host: contains flash configuration

HOST_MSG_TYPE_WRITE_FLASH_CONF Host to Node: specifies flash configuration to be programmed into flash

HOST_MSG_TYPE_GET_STATE Host to Node: State query.

HOST_MSG_TYPE_GET_STATE_RSP Node to Host: State response.

HOST_MSG_TYPE_BEGIN_SW_UPGR Host to Node: State query.

HOST_MSG_TYPE_BEGIN_SW_UPGR_RSP Node to Host: State response.

HOST_MSG_TYPE_OTA_DIAG_IND Node to Host: OTA diag indication.

HOST_MSG_TYPE_PROVISION_KEYS_REQ Host to Node: provision keys request.

HOST_MSG_TYPE_PROVISION_KEYS_RSP Node to Host: provision keys response.

HOST_MSG_TYPE_SW_UPGR2_BEGIN_REQ Host to Node: begin SW upgrade.

HOST_MSG_TYPE_SW_UPGR2_BEGIN_RSP Node to Host: begin SW upgrade response.

HOST_MSG_TYPE_SW_UPGR2_CHUNK_REQ Host to Node: chunk for a SW upgrade.

HOST_MSG_TYPE_SW_UPGR2_CHUNK_RSP Node to Host: chunk for a SW upgrade response.

HOST_MSG_TYPE_SW_UPGR2_END_REQ Host to Node: end SW upgrade.

HOST_MSG_TYPE_SW_UPGR2_END_RSP Node to Host: end SW upgrade response.

HOST_MSG_TYPE_TXSDU Host to Node: A MAC-bound (uplink) SDU.

HOST_MSG_TYPE_TXSDU_RSP Node to Host: tx feedback messages

HOST_MSG_TYPE_TXSDU_RESULT Node to Host: contains success/failure information about SDU transmission

HOST_MSG_TYPE_RXSDU Node to Host: A host-bound (downlink) SDU.

HOST_MSG_TYPE_FLUSH_TXSDU_QUEUE Host to Node: Requests all queued uplink SDU to be dropped.

HOST_MSG_TYPE_FLUSH_TXSDU_QUEUE_RSP Node to Host: Flush TXSDU queue response.

HOST_MSG_TYPE_ACK Node to Host: ack sent for every Host to Node msg

HOST_MSG_TYPE_ERR Node to Host: Error condition

HOST_MSG_TYPE_CONNECT Host to node: enables node to host messages

HOST_MSG_TYPE_TIME_SYNC_REQ Host to node: request time synchronization

HOST_MSG_TYPE_TIME_SYNC_RSP Node to host: time synchronization response

HOST_MSG_TYPE_SET_PRE_UPDATE_NOTIFICATION_REQ Host to node: request set pre-update-interval notification

HOST_MSG_TYPE_SET_PRE_UPDATE_NOTIFICATION_RSP Node to host: set pre-update-interval notification response

HOST_MSG_TYPE_PRE_UPDATE_NOTIFICATION_IND Node to host: pre-update-interval notification indication

HOST_MSG_TYPE_BLACKOUT_START_IND Node to host: indication containing time to blackout period start

HOST_MSG_TYPE_BLACKOUT_END_IND Node to host: indication of blackout period end

HOST_MSG_TYPE_BROADCAST_START_IND Node to host: broadcast starting indication

HOST_MSG_TYPE_BROADCAST_START_CNF Host to node: decides whether this image should be received

HOST_MSG_TYPE_BROADCAST_END_IND Node to host: image is received, available locally

HOST_MSG_TYPE_BROADCAST_DATA_REQ Host to node: request image chunk

HOST_MSG_TYPE_BROADCAST_DATA_RSP Node to host: transfer image chunk to host

HOST_MSG_TYPE_NODE_SW_UPGRADE_IND Node to host: node starting upgrade

HOST_MSG_TYPE_NODE_SW_UPGRADE_CNF Host to node: confirm node starting upgrade

HOST_MSG_TYPE_SET_HOST_ID_REQ Host to node: configure unique host ID for diag purposes (optional)

4.2.4.10 enum sys_mgr_state_t

Over The Air System State.

Used for messages HOST_MSG_TYPE_GET_STATE_RSP, HOST_MSG_TYPE_SYSTEM_STATE, HOST_MSG_TYPE_GET_PARAMS_RSP

See also

host_msg_getStateRsp_t
host_msg_systemState_t
host_msg_getParamRsp_t

Enumerator:

SYS_MGR_STATE_NIL	System Manager not started.
SYS_MGR_STATE_STARTUP	System Manager has been started.
SYS_MGR_STATE_IDLE	Startup sequence complete. Waiting for network enter command.
SYS_MGR_STATE_SCANNING	Scanning for network.
SYS_MGR_STATE_TRACK	Scan successful. Trying to join network.
SYS_MGR_STATE_JOINED	Network joined successfully.

4.3 spi_common_proto.h File Reference

common SPI master/slave protocol definitions

Data Structures

- struct SpiProtoCmd

  Two byte SPI transfer header - see doc "SPI slave node interface".

Defines

- #define **SPI_PROTO_MAX_PAYLOAD_BYTES** 510
- #define **SPI_PROTO_SLAVE** 0x1
- #define **SPI_PROTO_MASTER** 0x2
- #define **SPI_PROTO_OP_ARB** 0x1
- #define **SPI_PROTO_OP_VAL** 0x2
- #define **SPI_PROTO_OP_MMSG** 0x9
- #define **SPI_PROTO_OP_MHDR** 0xA
- #define **SPI_PROTO_OP_SMSG** 0xB
- #define **SPI_PROTO_OP_SHDR** 0xC
- #define **SPI_PROTO_CREATE_CMD**(cmd, src, len, op) (cmd).byte_1 = ((((src) & 0x3) << 6) | (((len) & 0x3) << 4) | ((op) & 0xF))
- #define **SPI_PROTO_OPCODE**(cmd) ((cmd).byte_1 & 0xF)
- #define **SPI_PROTO_SOURCE**(cmd) (((cmd).byte_1 >> 6) & 0x3)

4.3.1 Detailed Description

common SPI master/slave protocol definitions ===

DESCRIPTION: Message header definitions for SPI master/slave protocol

Copyright 2009 OnRamp Wireless, Inc.

==

Index