

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.

Copyright © 2011 On-Ramp Wireless Incorporated.
All Rights Reserved.

The information disclosed in this document is proprietary to On-Ramp Wireless Inc., and is not to be used or disclosed to unauthorized persons without the written consent of On-Ramp Wireless. The recipient of this document shall respect the security of this document and maintain the confidentiality of the information it contains. The master copy of this document is stored in electronic format, therefore any hard or soft copy used for distribution purposes must be considered as uncontrolled. Reference should be made to On-Ramp Wireless to obtain the latest version. By accepting this material the recipient agrees that this material and the information contained therein is to be held in confidence and in trust and will not be used, copied, reproduced in whole or in part, nor its contents revealed in any manner to others without the express written permission of On-Ramp Wireless Incorporated.

On-Ramp Wireless Incorporated reserves the right to make changes to the product(s) or information contained herein without notice. No liability is assumed for any damages arising directly or indirectly by their use or application. The information provided in this document is provided on an as is basis.

This document contains On-Ramp Wireless proprietary information and must be shredded when discarded.

This documentation and the software described in it are copyrighted with all rights reserved. This documentation and the software may not be copied, except as otherwise provided in your software license or as expressly permitted in writing by On-Ramp Wireless, Incorporated.

Any sample code herein is provided for your convenience and has not been tested or designed to work on any particular system configuration. It is provided AS IS and your use of this sample code, whether as provided or with any modification, is at your own risk. On-Ramp Wireless undertakes no liability or responsibility with respect to the sample code, and disclaims all warranties, express and implied, including without limitation warranties on merchantability, fitness for a specified purpose, and infringement. On-Ramp Wireless reserves all rights in the sample code, and permits use of this sample code only for educational and reference purposes.

This technology and technical data may be subject to U.S. and international export, re-export or transfer (export) laws. Diversion contrary to U.S. and international law is strictly prohibited.

Ultra-Link Processing™ is a trademark of On-Ramp Wireless.

Other product and brand names may be trademarks or registered trademarks of their respective owners.

Contents

1	Data Structure Index	1
1.1	Data Structures	1
2	File Index	3
2.1	File List	3
3	Data Structure Documentation	5
3.1	CAL_CONFIG_FlashCalibration_t Struct Reference	5
3.2	CAL_CONFIG_FlashConfig_t Struct Reference	7
3.3	host_msg_ack_t Struct Reference	8
3.3.1	Detailed Description	8
3.3.2	Field Documentation	8
3.3.2.1	footer	8
3.3.2.2	header	8
3.4	host_msg_beginSwUpgrade_t Struct Reference	9
3.4.1	Detailed Description	9
3.4.2	Field Documentation	9
3.4.2.1	checksum	9
3.4.2.2	footer	9
3.4.2.3	header	9
3.4.2.4	numChunks	9
3.5	host_msg_beginSwUpgradeRsp_t Struct Reference	10
3.5.1	Detailed Description	10
3.5.2	Field Documentation	10
3.5.2.1	footer	10
3.5.2.2	header	10
3.5.2.3	result	10
3.6	host_msg_blackoutEndInd_t Struct Reference	11
3.6.1	Field Documentation	11

3.6.1.1	footer	11
3.6.1.2	header	11
3.6.1.3	wasUpdateIntervalSkipped	11
3.7	host_msg_blackoutStartInd_t Struct Reference	12
3.7.1	Field Documentation	12
3.7.1.1	durationInSec	12
3.7.1.2	footer	12
3.7.1.3	header	12
3.7.1.4	secUntilStart	12
3.8	host_msg_broadcastDataReq_t Struct Reference	13
3.8.1	Field Documentation	13
3.8.1.1	bcastId	13
3.8.1.2	footer	13
3.8.1.3	header	13
3.8.1.4	length	13
3.8.1.5	offset	13
3.9	host_msg_broadcastDataRsp_t Struct Reference	14
3.9.1	Field Documentation	14
3.9.1.1	bcastId	14
3.9.1.2	footer	14
3.9.1.3	header	14
3.9.1.4	length	14
3.9.1.5	offset	14
3.9.1.6	payload	14
3.9.1.7	status	15
3.10	host_msg_broadcastEndInd_t Struct Reference	16
3.10.1	Field Documentation	16
3.10.1.1	bcastId	16
3.10.1.2	footer	16
3.10.1.3	header	16
3.10.1.4	length	16
3.11	host_msg_broadcastStartCnf_t Struct Reference	17
3.11.1	Field Documentation	17
3.11.1.1	acceptBroadcast	17
3.11.1.2	bcastId	17
3.11.1.3	footer	17

3.11.1.4	header	17
3.12	host_msg_broadcastStartInd_t Struct Reference	18
3.12.1	Field Documentation	18
3.12.1.1	bcastId	18
3.12.1.2	footer	18
3.12.1.3	header	18
3.12.1.4	length	18
3.12.1.5	payload	18
3.13	host_msg_connect_t Struct Reference	19
3.13.1	Detailed Description	19
3.13.2	Field Documentation	19
3.13.2.1	connected	19
3.13.2.2	footer	19
3.13.2.3	header	19
3.14	host_msg_err_t Struct Reference	20
3.14.1	Detailed Description	20
3.14.2	Field Documentation	20
3.14.2.1	errCode	20
3.14.2.2	footer	20
3.14.2.3	header	20
3.15	host_msg_flushTxSduQueue_t Struct Reference	21
3.15.1	Detailed Description	21
3.15.2	Field Documentation	21
3.15.2.1	footer	21
3.15.2.2	header	21
3.15.2.3	includeInProgressSdus	21
3.16	host_msg_flushTxSduQueueRsp_t Struct Reference	22
3.16.1	Detailed Description	22
3.16.2	Field Documentation	22
3.16.2.1	flushSucceeded	22
3.16.2.2	footer	22
3.16.2.3	header	22
3.17	host_msg_frameStats_t Struct Reference	23
3.17.1	Detailed Description	24
3.17.2	Field Documentation	24
3.17.2.1	altitude	24

3.17.2.2	boostedFineAFCMetric	24
3.17.2.3	center_freq_offset	24
3.17.2.4	channel	24
3.17.2.5	demodType	24
3.17.2.6	digitalTruncation	24
3.17.2.7	failedFrameCnt	24
3.17.2.8	fingerCAFC	24
3.17.2.9	fingerEnergy	25
3.17.2.10	fingerFineAFCs	25
3.17.2.11	fingerPower	25
3.17.2.12	fingerTimingOffsetParity	25
3.17.2.13	footer	25
3.17.2.14	frameDelaySymbols	25
3.17.2.15	freqOffset	25
3.17.2.16	hammingWeight	25
3.17.2.17	header	25
3.17.2.18	heading	25
3.17.2.19	highCAFC	25
3.17.2.20	highTimingOffset	26
3.17.2.21	lastDchSpreading	26
3.17.2.22	lastTxSpreading	26
3.17.2.23	lastTxSubslot	26
3.17.2.24	latitude	26
3.17.2.25	longitude	26
3.17.2.26	lowCAFC	26
3.17.2.27	lowTimingOffset	26
3.17.2.28	numLoggingMsgsDropped	26
3.17.2.29	oscCal26m	26
3.17.2.30	oscCal32k	27
3.17.2.31	RSSI	27
3.17.2.32	rsi_high	27
3.17.2.33	rsi_low	27
3.17.2.34	sfn	27
3.17.2.35	subslot	27
3.17.2.36	txFreqStride	27
3.17.2.37	txTimeTrackingStride	27

3.17.2.38 txVGA	27
3.17.2.39 velocity	27
3.17.2.40 winningFineAFC	27
3.18 host_msg_getExceptionBufferReq_t Struct Reference	28
3.18.1 Detailed Description	28
3.18.2 Field Documentation	28
3.18.2.1 chunk	28
3.18.2.2 clearBuffer	28
3.18.2.3 footer	28
3.18.2.4 header	28
3.19 host_msg_getExceptionBufferRsp_t Struct Reference	29
3.19.1 Detailed Description	29
3.19.2 Field Documentation	29
3.19.2.1 buffer	29
3.19.2.2 footer	29
3.19.2.3 header	29
3.20 host_msg_getParamRsp_t Struct Reference	30
3.20.1 Detailed Description	30
3.20.2 Field Documentation	30
3.20.2.1 bcastGoldCode	30
3.20.2.2 bcastSlot	30
3.20.2.3 bcastSpreading	31
3.20.2.4 channelBW	31
3.20.2.5 channelNum	31
3.20.2.6 cid	31
3.20.2.7 dataSubslot	31
3.20.2.8 demodChannel	31
3.20.2.9 dlDataGoldCode	31
3.20.2.10 dlDataSpreading	31
3.20.2.11 footer	31
3.20.2.12 header	32
3.20.2.13 listenInterval	32
3.20.2.14 maxTxPwrLimit	32
3.20.2.15 maxTxPwrLimitHeadRoom	32
3.20.2.16 nodeId	32
3.20.2.17 numNCAccum	32

3.20.2.18	pad	32
3.20.2.19	rssMargin	32
3.20.2.20	slotInterval	32
3.20.2.21	systemState	33
3.20.2.22	ulSpreading	33
3.21	host_msg_getParams_t Struct Reference	34
3.21.1	Detailed Description	34
3.21.2	Field Documentation	34
3.21.2.1	footer	34
3.21.2.2	header	34
3.22	host_msg_getState_t Struct Reference	35
3.22.1	Detailed Description	35
3.22.2	Field Documentation	35
3.22.2.1	footer	35
3.22.2.2	header	35
3.23	host_msg_getStateRsp_t Struct Reference	36
3.23.1	Detailed Description	36
3.23.2	Field Documentation	36
3.23.2.1	footer	36
3.23.2.2	header	36
3.23.2.3	state	36
3.24	host_msg_header_t Struct Reference	37
3.24.1	Detailed Description	37
3.24.2	Field Documentation	37
3.24.2.1	msgLen	37
3.24.2.2	msgType	37
3.25	host_msg_hostIdReq_t Struct Reference	38
3.25.1	Field Documentation	38
3.25.1.1	footer	38
3.25.1.2	header	38
3.25.1.3	hostId	38
3.26	host_msg_nodeSwUpgradeCnf_t Struct Reference	39
3.26.1	Field Documentation	39
3.26.1.1	footer	39
3.26.1.2	header	39
3.27	host_msg_nodeSwUpgradeInd_t Struct Reference	40

3.27.1	Field Documentation	40
3.27.1.1	footer	40
3.27.1.2	header	40
3.28	host_msg_otaDiagInd_t Struct Reference	41
3.28.1	Detailed Description	41
3.28.2	Field Documentation	41
3.28.2.1	footer	41
3.28.2.2	header	41
3.28.2.3	state	41
3.29	host_msg_preUpdateNotificationInd_t Struct Reference	42
3.29.1	Detailed Description	42
3.29.2	Field Documentation	42
3.29.2.1	footer	42
3.29.2.2	header	42
3.30	host_msg_provisionKeysReq_t Struct Reference	43
3.30.1	Detailed Description	43
3.30.2	Field Documentation	43
3.30.2.1	footer	43
3.30.2.2	gatewayCdldKey	43
3.30.2.3	gatewayKey	43
3.30.2.4	header	43
3.30.2.5	rootKey	44
3.31	host_msg_provisionKeysRsp_t Struct Reference	45
3.31.1	Detailed Description	45
3.31.2	Field Documentation	45
3.31.2.1	footer	45
3.31.2.2	header	45
3.32	host_msg_readFlashConf_t Struct Reference	46
3.32.1	Detailed Description	46
3.32.2	Field Documentation	46
3.32.2.1	footer	46
3.32.2.2	header	46
3.33	host_msg_readFlashConfRsp_t Struct Reference	47
3.33.1	Detailed Description	47
3.33.2	Field Documentation	47
3.33.2.1	footer	47

3.33.2.2	header	47
3.34	host_msg_rxSdu_t Struct Reference	48
3.34.1	Detailed Description	48
3.34.2	Field Documentation	48
3.34.2.1	footer	48
3.34.2.2	header	48
3.34.2.3	pad	48
3.34.2.4	payload	48
3.34.2.5	size	49
3.35	host_msg_setChannel_t Struct Reference	50
3.35.1	Detailed Description	50
3.35.2	Field Documentation	50
3.35.2.1	channelNum	50
3.35.2.2	footer	50
3.35.2.3	header	50
3.36	host_msg_setGoldCode_t Struct Reference	51
3.36.1	Detailed Description	51
3.36.2	Field Documentation	51
3.36.2.1	bcastGoldCode	51
3.36.2.2	dataGoldCode	51
3.36.2.3	footer	51
3.36.2.4	header	51
3.37	host_msg_setPreUpdateNotificationReq_t Struct Reference	52
3.37.1	Detailed Description	52
3.37.2	Field Documentation	52
3.37.2.1	footer	52
3.37.2.2	header	52
3.37.2.3	timeInMs	52
3.38	host_msg_setPreUpdateNotificationRsp_t Struct Reference	53
3.38.1	Detailed Description	53
3.38.2	Field Documentation	53
3.38.2.1	footer	53
3.38.2.2	header	53
3.38.2.3	result	53
3.39	host_msg_setSpreading_t Struct Reference	54
3.39.1	Detailed Description	54

3.39.2	Field Documentation	54
3.39.2.1	dlBcastSpreading	54
3.39.2.2	footer	54
3.39.2.3	header	54
3.39.2.4	ulSpreading	54
3.40	host_msg_startFrameStats_t Struct Reference	55
3.40.1	Detailed Description	55
3.40.2	Field Documentation	55
3.40.2.1	footer	55
3.40.2.2	header	55
3.41	host_msg_stopFrameStats_t Struct Reference	56
3.41.1	Detailed Description	56
3.41.2	Field Documentation	56
3.41.2.1	footer	56
3.41.2.2	header	56
3.42	host_msg_swUpgrade2BeginReq_t Struct Reference	57
3.42.1	Detailed Description	57
3.42.2	Field Documentation	57
3.42.2.1	checksum	57
3.42.2.2	footer	57
3.42.2.3	header	57
3.42.2.4	numChunks	57
3.43	host_msg_swUpgrade2BeginRsp_t Struct Reference	58
3.43.1	Detailed Description	58
3.43.2	Field Documentation	58
3.43.2.1	footer	58
3.43.2.2	header	58
3.43.2.3	result	58
3.44	host_msg_swUpgrade2ChunkReq_t Struct Reference	59
3.44.1	Detailed Description	59
3.44.2	Field Documentation	59
3.44.2.1	checksum	59
3.44.2.2	chunk	59
3.44.2.3	footer	59
3.44.2.4	header	59
3.44.2.5	num	60

3.45	host_msg_swUpgrade2ChunkRsp_t Struct Reference	61
3.45.1	Detailed Description	61
3.45.2	Field Documentation	61
3.45.2.1	footer	61
3.45.2.2	header	61
3.45.2.3	result	61
3.46	host_msg_swUpgrade2EndReq_t Struct Reference	62
3.46.1	Detailed Description	62
3.46.2	Field Documentation	62
3.46.2.1	footer	62
3.46.2.2	header	62
3.47	host_msg_swUpgrade2EndRsp_t Struct Reference	63
3.47.1	Detailed Description	63
3.47.2	Field Documentation	63
3.47.2.1	footer	63
3.47.2.2	header	63
3.47.2.3	result	63
3.48	host_msg_systemSetState_t Struct Reference	64
3.48.1	Detailed Description	64
3.48.2	Field Documentation	64
3.48.2.1	footer	64
3.48.2.2	header	64
3.48.2.3	state	64
3.49	host_msg_systemState_t Struct Reference	65
3.49.1	Detailed Description	65
3.49.2	Field Documentation	65
3.49.2.1	footer	65
3.49.2.2	header	65
3.49.2.3	state	65
3.50	host_msg_timeSyncReq_t Struct Reference	66
3.50.1	Detailed Description	66
3.50.2	Field Documentation	66
3.50.2.1	footer	66
3.50.2.2	header	66
3.51	host_msg_timeSyncRsp_t Struct Reference	67
3.51.1	Detailed Description	67

3.51.2	Field Documentation	67
3.51.2.1	day	67
3.51.2.2	footer	67
3.51.2.3	header	67
3.51.2.4	month	68
3.51.2.5	rsv1	68
3.51.2.6	time_of_day_frac	68
3.51.2.7	time_of_day_whole	68
3.51.2.8	valid	68
3.51.2.9	year	68
3.52	host_msg_txProgrammed_t Struct Reference	69
3.52.1	Detailed Description	69
3.52.2	Field Documentation	69
3.52.2.1	digitalTruncation	69
3.52.2.2	footer	69
3.52.2.3	freqOffset	69
3.52.2.4	header	70
3.52.2.5	numLoggingMsgsDropped	70
3.52.2.6	numSubslots	70
3.52.2.7	spreading	70
3.52.2.8	startingSubslot	70
3.52.2.9	txFreqStride	70
3.52.2.10	txTimeTrackingStride	70
3.52.2.11	txVGA	70
3.53	host_msg_txSdu_t Struct Reference	71
3.53.1	Detailed Description	71
3.53.2	Field Documentation	71
3.53.2.1	flags	71
3.53.2.2	footer	71
3.53.2.3	header	71
3.53.2.4	host_tag	71
3.53.2.5	pad	72
3.53.2.6	payload	72
3.53.2.7	size	72
3.54	host_msg_txSduResult_t Struct Reference	73
3.54.1	Detailed Description	73

3.54.2	Field Documentation	73
3.54.2.1	footer	73
3.54.2.2	header	73
3.54.2.3	host_tag	73
3.54.2.4	sduStatus	73
3.55	host_msg_txSduRsp_t Struct Reference	75
3.55.1	Detailed Description	75
3.55.2	Field Documentation	75
3.55.2.1	footer	75
3.55.2.2	header	75
3.55.2.3	host_tag	75
3.55.2.4	isEnqueued	75
3.56	host_msg_uptimeStatsReq_t Struct Reference	77
3.56.1	Detailed Description	77
3.56.2	Field Documentation	77
3.56.2.1	footer	77
3.56.2.2	header	77
3.57	host_msg_uptimeStatsRsp_t Struct Reference	78
3.57.1	Detailed Description	78
3.57.2	Field Documentation	78
3.57.2.1	footer	78
3.57.2.2	header	78
3.57.2.3	lastBootWasWatchdog	78
3.57.2.4	numWdogResets	78
3.57.2.5	secondsSinceLastBoot	79
3.58	host_msg_version_t Struct Reference	80
3.58.1	Detailed Description	80
3.58.2	Field Documentation	80
3.58.2.1	footer	80
3.58.2.2	header	80
3.59	host_msg_versionRsp_t Struct Reference	81
3.59.1	Detailed Description	81
3.59.2	Field Documentation	81
3.59.2.1	footer	81
3.59.2.2	header	81
3.59.2.3	phyRev	81

3.59.2.4	swRev	81
3.60	host_msg_writeFlashConf_t Struct Reference	82
3.60.1	Detailed Description	82
3.60.2	Field Documentation	82
3.60.2.1	footer	82
3.60.2.2	header	82
3.61	SpiProtoCmd Struct Reference	83
3.61.1	Detailed Description	83
4	File Documentation	85
4.1	cal_config.h File Reference	85
4.1.1	Detailed Description	85
4.2	host_customer_msg.h File Reference	86
4.2.1	Detailed Description	91
4.2.2	Define Documentation	91
4.2.2.1	HOST_MSG_DIR_HOST_TO_NODE	91
4.2.2.2	HOST_MSG_DIR_NODE_TO_HOST	91
4.2.2.3	HOST_MSG_END_MARKER	91
4.2.2.4	HOST_MSG_MAX_HOST_INTF_SDU_SIZE	92
4.2.2.5	HOST_MSG_MAX_SDU_SIZE	92
4.2.2.6	HOST_MSG_MIN_SDU_SIZE	92
4.2.2.7	HOST_MSG_OVERHEAD_LEN	92
4.2.2.8	HOST_MSG_SDU_STATUS_BITS_ACK_FAIL	92
4.2.2.9	HOST_MSG_SDU_STATUS_BITS_ACK_SUCCESS	92
4.2.2.10	HOST_MSG_SDU_STATUS_BITS_BUFFER_FULL	92
4.2.2.11	HOST_MSG_SDU_STATUS_BITS_DROPPED_CDLD	92
4.2.2.12	HOST_MSG_SDU_STATUS_BITS_DROPPED_HOST	92
4.2.2.13	HOST_MSG_SDU_STATUS_BITS_DROPPED_MAINTENANCE	92
4.2.2.14	HOST_MSG_SDU_STATUS_BITS_DROPPED_NET_EXIT	93
4.2.2.15	HOST_MSG_SDU_STATUS_BITS_DROPPED_NOT_JOINED	93
4.2.2.16	HOST_MSG_SDU_STATUS_BITS_OTHER_ERROR	93
4.2.2.17	HOST_MSG_SDU_STATUS_BITS_TRANSMITTED	93
4.2.3	Typedef Documentation	93
4.2.3.1	host_msg_txsdu_result_sdustatus_t	93
4.2.4	Enumeration Type Documentation	93
4.2.4.1	host_msg_broadcastStatus_t	93
4.2.4.2	host_msg_errCode_t	94

4.2.4.3	host_msg_frameStatsType_t	94
4.2.4.4	host_msg_host_t	94
4.2.4.5	host_msg_joinBackoffType_t	95
4.2.4.6	host_msg_joinType_t	95
4.2.4.7	host_msg_sduFlags_t	95
4.2.4.8	host_msg_systemAirlinkState_t	95
4.2.4.9	host_msg_type_t	95
4.2.4.10	sys_mgr_state_t	97
4.3	spi_common_proto.h File Reference	99
4.3.1	Detailed Description	99

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

CAL_CONFIG_FlashCalibration_t	5
CAL_CONFIG_FlashConfig_t	7
host_msg_ack_t (An ACK message)	8
host_msg_beginSwUpgrade_t (A Begin SW Upgrade message)	9
host_msg_beginSwUpgradeRsp_t (A Begin Software Upgrade Response message)	10
host_msg_blackoutEndInd_t	11
host_msg_blackoutStartInd_t	12
host_msg_broadcastDataReq_t	13
host_msg_broadcastDataRsp_t	14
host_msg_broadcastEndInd_t	16
host_msg_broadcastStartCnf_t	17
host_msg_broadcastStartInd_t	18
host_msg_connect_t (A CONNECT message)	19
host_msg_err_t (An ERR message)	20
host_msg_flushTxSduQueue_t (Requests all queued uplink sdu to be dropped)	21
host_msg_flushTxSduQueueRsp_t (Relays the result from a flush TXSDU queue message)	22
host_msg_frameStats_t (A Frame Statistics Message)	23
host_msg_getExceptionBufferReq_t (A Get Exception Buffer request message)	28
host_msg_getExceptionBufferRsp_t (A Get Exception Buffer response message)	29
host_msg_getParamRsp_t (A Get Params Rsp message)	30
host_msg_getParams_t (Get Params message)	34
host_msg_getState_t (A Get State message)	35
host_msg_getStateRsp_t (Get State response message)	36
host_msg_header_t (The host interface message header)	37
host_msg_hostIdReq_t	38
host_msg_nodeSwUpgradeCnf_t	39
host_msg_nodeSwUpgradeInd_t	40
host_msg_otaDiagInd_t (Indicates whether OTA diag mode is enabled)	41
host_msg_preUpdateNotificationInd_t (HOST_MSG_TYPE_PRE_UPDATE_NOTIFICATION_IND message)	42
host_msg_provisionKeysReq_t (Provisions security keys in the node)	43
host_msg_provisionKeysRsp_t (Provisions security keys in the node)	45
host_msg_readFlashConf_t (A READ FLASH CONF message)	46

host_msg_readFlashConfRsp_t (A READ FLASH CONF RSP message)	47
host_msg_rxSdu_t (A RX SDU message)	48
host_msg_setChannel_t (A SET CENTER FREQ message)	50
host_msg_setGoldCode_t (A SET GOLD CODE message)	51
host_msg_setPreUpdateNotificationReq_t (HOST_MSG_TYPE_SET_PRE_UPDATE_- NOTIFICATION_REQ message)	52
host_msg_setPreUpdateNotificationRsp_t (HOST_MSG_TYPE_SET_PRE_UPDATE_- NOTIFICATION_RSP message)	53
host_msg_setSpreading_t (A SET SPREADING message)	54
host_msg_startFrameStats_t (A START FRAME STATS message)	55
host_msg_stopFrameStats_t (A STOP FRAME STATS message)	56
host_msg_swUpgrade2BeginReq_t (A Begin SW Upgrade2 message)	57
host_msg_swUpgrade2BeginRsp_t (A Begin Software Upgrade2 Response message)	58
host_msg_swUpgrade2ChunkReq_t (A SW upgrade chunk)	59
host_msg_swUpgrade2ChunkRsp_t (A chunk response message)	61
host_msg_swUpgrade2EndReq_t (Sent to end the SW upgrade and boot to the new image)	62
host_msg_swUpgrade2EndRsp_t (Response to host_msg_chunkSwUpgrade2Req_t)	63
host_msg_systemSetState_t (A SYSTEM SET STATE message)	64
host_msg_systemState_t (A SYSTEM STATE message)	65
host_msg_timeSyncReq_t (HOST_MSG_TYPE_TIME_SYNC_REQ message)	66
host_msg_timeSyncRsp_t (HOST_MSG_TYPE_TIME_SYNC_REQ message)	67
host_msg_txProgrammed_t (A TX PROGRAMMED message)	69
host_msg_txSdu_t (A TX SDU message)	71
host_msg_txSduResult_t (A TX SDU Result message)	73
host_msg_txSduRsp_t (A TX SDU RSP message)	75
host_msg_uptimeStatsReq_t (A Uptime Stats request message)	77
host_msg_uptimeStatsRsp_t (A Uptime Stats response message)	78
host_msg_version_t (A VERSION message)	80
host_msg_versionRsp_t (A VERSION RSP message)	81
host_msg_writeFlashConf_t (A WRITE FLASH CONF message)	82
SpiProtoCmd (Two byte SPI transfer header - see doc "SPI slave node interface")	83

Chapter 2

File Index

2.1 File List

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

cal_config.h (Structures defining the layout of flash calibration and configuration tables)	85
host_customer_msg.h (Host interface messaging interface for customer)	86
spi_common_proto.h (Common SPI master/slave protocol definitions)	99
system.h	??

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 **maxTxVGA**Low
- uint32_t **maxTxVGA**Mid
- uint32_t **maxTxVGA**High
- uint32_t **maxTxVGA**OutputPwr
- 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 **txoFreq**
- uint16_t **countryCode**
- uint16_t **sysSelMinSleepTimer**
- uint16_t **fieldTestNumPdusPerFrame**
- uint16_t **fieldTestNumFramePeriod**
- uint8_t **fieldTestUIRssiMargin**
- 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 = $2^{\text{lastDchSpreading}}$

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 = $2^{\text{lastTxSpreading}}$

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 = $2^{\text{bcastSpreading}}$

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 = $2^{\text{dlDataSpreading}}$

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 = $2^{\text{ulSpreading}}$

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_getStateResp_t Struct Reference

Get State response message.

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

Collaboration diagram for host_msg_getStateResp_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_getStateResp_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_getStateResp_t::header

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

3.23.2.3 [uint32_t](#) host_msg_getStateResp_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 precedes 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 enqueueing 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 = $2^{\text{spreading}}$

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 response 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 Conecctvity. 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 or 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

- acceptBroadcast
  - host\_msg\_broadcastStartCnf\_t, [17](#)
- altitude
  - host\_msg\_frameStats\_t, [24](#)
- bcastGoldCode
  - host\_msg\_getParamRsp\_t, [30](#)
  - host\_msg\_setGoldCode\_t, [51](#)
- bcastId
  - host\_msg\_broadcastDataReq\_t, [13](#)
  - host\_msg\_broadcastDataRsp\_t, [14](#)
  - host\_msg\_broadcastEndInd\_t, [16](#)
  - host\_msg\_broadcastStartCnf\_t, [17](#)
  - host\_msg\_broadcastStartInd\_t, [18](#)
- bcastSlot
  - host\_msg\_getParamRsp\_t, [30](#)
- bcastSpreading
  - host\_msg\_getParamRsp\_t, [30](#)
- boostedFineAFCMetric
  - host\_msg\_frameStats\_t, [24](#)
- buffer
  - host\_msg\_getExceptionBufferRsp\_t, [29](#)
- cal\_config.h, [85](#)
- CAL\_CONFIG\_FlashCalibration\_t, [5](#)
- CAL\_CONFIG\_FlashConfig\_t, [7](#)
- center\_freq\_offset
  - host\_msg\_frameStats\_t, [24](#)
- channel
  - host\_msg\_frameStats\_t, [24](#)
- channelBW
  - host\_msg\_getParamRsp\_t, [31](#)
- channelNum
  - host\_msg\_getParamRsp\_t, [31](#)
  - host\_msg\_setChannel\_t, [50](#)
- checksum
  - host\_msg\_beginSwUpgrade\_t, [9](#)
  - host\_msg\_swUpgrade2BeginReq\_t, [57](#)
  - host\_msg\_swUpgrade2ChunkReq\_t, [59](#)
- chunk
  - host\_msg\_getExceptionBufferReq\_t, [28](#)
  - host\_msg\_swUpgrade2ChunkReq\_t, [59](#)
- cid
  - host\_msg\_getParamRsp\_t, [31](#)
- clearBuffer
  - host\_msg\_getExceptionBufferReq\_t, [28](#)
- connected
  - host\_msg\_connect\_t, [19](#)
- dataGoldCode
  - host\_msg\_setGoldCode\_t, [51](#)
- dataSubslot
  - host\_msg\_getParamRsp\_t, [31](#)
- day
  - host\_msg\_timeSyncRsp\_t, [67](#)
- demodChannel
  - host\_msg\_getParamRsp\_t, [31](#)
- demodType
  - host\_msg\_frameStats\_t, [24](#)
- digitalTruncation
  - host\_msg\_frameStats\_t, [24](#)
  - host\_msg\_txProgrammed\_t, [69](#)
- dlBcastSpreading
  - host\_msg\_setSpreading\_t, [54](#)
- dlDataGoldCode
  - host\_msg\_getParamRsp\_t, [31](#)
- dlDataSpreading
  - host\_msg\_getParamRsp\_t, [31](#)
- durationInSec
  - host\_msg\_blackoutStartInd\_t, [12](#)
- errCode
  - host\_msg\_err\_t, [20](#)
- failedFrameCnt
  - host\_msg\_frameStats\_t, [24](#)
- fingerCAFC
  - host\_msg\_frameStats\_t, [24](#)
- fingerEnergy
  - host\_msg\_frameStats\_t, [24](#)
- fingerFineAFCs
  - host\_msg\_frameStats\_t, [25](#)
- fingerPower
  - host\_msg\_frameStats\_t, [25](#)
- fingerTimingOffsetParity
  - host\_msg\_frameStats\_t, [25](#)
- flags
  - host\_msg\_txSdu\_t, [71](#)
- flushSucceeded
  - host\_msg\_flushTxSduQueueRsp\_t, [22](#)

## footer

- [host\\_msg\\_ack\\_t, 8](#)
- [host\\_msg\\_beginSwUpgrade\\_t, 9](#)
- [host\\_msg\\_beginSwUpgradeRsp\\_t, 10](#)
- [host\\_msg\\_blackoutEndInd\\_t, 11](#)
- [host\\_msg\\_blackoutStartInd\\_t, 12](#)
- [host\\_msg\\_broadcastDataReq\\_t, 13](#)
- [host\\_msg\\_broadcastDataRsp\\_t, 14](#)
- [host\\_msg\\_broadcastEndInd\\_t, 16](#)
- [host\\_msg\\_broadcastStartCnf\\_t, 17](#)
- [host\\_msg\\_broadcastStartInd\\_t, 18](#)
- [host\\_msg\\_connect\\_t, 19](#)
- [host\\_msg\\_err\\_t, 20](#)
- [host\\_msg\\_flushTxSduQueue\\_t, 21](#)
- [host\\_msg\\_flushTxSduQueueRsp\\_t, 22](#)
- [host\\_msg\\_frameStats\\_t, 25](#)
- [host\\_msg\\_getExceptionBufferReq\\_t, 28](#)
- [host\\_msg\\_getExceptionBufferRsp\\_t, 29](#)
- [host\\_msg\\_getParamRsp\\_t, 31](#)
- [host\\_msg\\_getParams\\_t, 34](#)
- [host\\_msg\\_getState\\_t, 35](#)
- [host\\_msg\\_getStateRsp\\_t, 36](#)
- [host\\_msg\\_hostIdReq\\_t, 38](#)
- [host\\_msg\\_nodeSwUpgradeCnf\\_t, 39](#)
- [host\\_msg\\_nodeSwUpgradeInd\\_t, 40](#)
- [host\\_msg\\_otaDiagInd\\_t, 41](#)
- [host\\_msg\\_preUpdateNotificationInd\\_t, 42](#)
- [host\\_msg\\_provisionKeysReq\\_t, 43](#)
- [host\\_msg\\_provisionKeysRsp\\_t, 45](#)
- [host\\_msg\\_readFlashConf\\_t, 46](#)
- [host\\_msg\\_readFlashConfRsp\\_t, 47](#)
- [host\\_msg\\_rxSdu\\_t, 48](#)
- [host\\_msg\\_setChannel\\_t, 50](#)
- [host\\_msg\\_setGoldCode\\_t, 51](#)
- [host\\_msg\\_setPreUpdateNotificationReq\\_t, 52](#)
- [host\\_msg\\_setPreUpdateNotificationRsp\\_t, 53](#)
- [host\\_msg\\_setSpreading\\_t, 54](#)
- [host\\_msg\\_startFrameStats\\_t, 55](#)
- [host\\_msg\\_stopFrameStats\\_t, 56](#)
- [host\\_msg\\_swUpgrade2BeginReq\\_t, 57](#)
- [host\\_msg\\_swUpgrade2BeginRsp\\_t, 58](#)
- [host\\_msg\\_swUpgrade2ChunkReq\\_t, 59](#)
- [host\\_msg\\_swUpgrade2ChunkRsp\\_t, 61](#)
- [host\\_msg\\_swUpgrade2EndReq\\_t, 62](#)
- [host\\_msg\\_swUpgrade2EndRsp\\_t, 63](#)
- [host\\_msg\\_systemSetState\\_t, 64](#)
- [host\\_msg\\_systemState\\_t, 65](#)
- [host\\_msg\\_timeSyncReq\\_t, 66](#)
- [host\\_msg\\_timeSyncRsp\\_t, 67](#)
- [host\\_msg\\_txProgrammed\\_t, 69](#)
- [host\\_msg\\_txSdu\\_t, 71](#)
- [host\\_msg\\_txSduResult\\_t, 73](#)
- [host\\_msg\\_txSduRsp\\_t, 75](#)
- [host\\_msg\\_uptimeStatsReq\\_t, 77](#)

- [host\\_msg\\_uptimeStatsRsp\\_t, 78](#)
- [host\\_msg\\_version\\_t, 80](#)
- [host\\_msg\\_versionRsp\\_t, 81](#)
- [host\\_msg\\_writeFlashConf\\_t, 82](#)

## frameDelaySymbols

- [host\\_msg\\_frameStats\\_t, 25](#)

## freqOffset

- [host\\_msg\\_frameStats\\_t, 25](#)
- [host\\_msg\\_txProgrammed\\_t, 69](#)

## gatewayCddKey

- [host\\_msg\\_provisionKeysReq\\_t, 43](#)

## gatewayKey

- [host\\_msg\\_provisionKeysReq\\_t, 43](#)

## hammingWeight

- [host\\_msg\\_frameStats\\_t, 25](#)

## header

- [host\\_msg\\_ack\\_t, 8](#)
- [host\\_msg\\_beginSwUpgrade\\_t, 9](#)
- [host\\_msg\\_beginSwUpgradeRsp\\_t, 10](#)
- [host\\_msg\\_blackoutEndInd\\_t, 11](#)
- [host\\_msg\\_blackoutStartInd\\_t, 12](#)
- [host\\_msg\\_broadcastDataReq\\_t, 13](#)
- [host\\_msg\\_broadcastDataRsp\\_t, 14](#)
- [host\\_msg\\_broadcastEndInd\\_t, 16](#)
- [host\\_msg\\_broadcastStartCnf\\_t, 17](#)
- [host\\_msg\\_broadcastStartInd\\_t, 18](#)
- [host\\_msg\\_connect\\_t, 19](#)
- [host\\_msg\\_err\\_t, 20](#)
- [host\\_msg\\_flushTxSduQueue\\_t, 21](#)
- [host\\_msg\\_flushTxSduQueueRsp\\_t, 22](#)
- [host\\_msg\\_frameStats\\_t, 25](#)
- [host\\_msg\\_getExceptionBufferReq\\_t, 28](#)
- [host\\_msg\\_getExceptionBufferRsp\\_t, 29](#)
- [host\\_msg\\_getParamRsp\\_t, 32](#)
- [host\\_msg\\_getParams\\_t, 34](#)
- [host\\_msg\\_getState\\_t, 35](#)
- [host\\_msg\\_getStateRsp\\_t, 36](#)
- [host\\_msg\\_hostIdReq\\_t, 38](#)
- [host\\_msg\\_nodeSwUpgradeCnf\\_t, 39](#)
- [host\\_msg\\_nodeSwUpgradeInd\\_t, 40](#)
- [host\\_msg\\_otaDiagInd\\_t, 41](#)
- [host\\_msg\\_preUpdateNotificationInd\\_t, 42](#)
- [host\\_msg\\_provisionKeysReq\\_t, 43](#)
- [host\\_msg\\_provisionKeysRsp\\_t, 45](#)
- [host\\_msg\\_readFlashConf\\_t, 46](#)
- [host\\_msg\\_readFlashConfRsp\\_t, 47](#)
- [host\\_msg\\_rxSdu\\_t, 48](#)
- [host\\_msg\\_setChannel\\_t, 50](#)
- [host\\_msg\\_setGoldCode\\_t, 51](#)
- [host\\_msg\\_setPreUpdateNotificationReq\\_t, 52](#)
- [host\\_msg\\_setPreUpdateNotificationRsp\\_t, 53](#)
- [host\\_msg\\_setSpreading\\_t, 54](#)

- host\_msg\_startFrameStats\_t, 55
- host\_msg\_stopFrameStats\_t, 56
- host\_msg\_swUpgrade2BeginReq\_t, 57
- host\_msg\_swUpgrade2BeginRsp\_t, 58
- host\_msg\_swUpgrade2ChunkReq\_t, 59
- host\_msg\_swUpgrade2ChunkRsp\_t, 61
- host\_msg\_swUpgrade2EndReq\_t, 62
- host\_msg\_swUpgrade2EndRsp\_t, 63
- host\_msg\_systemSetState\_t, 64
- host\_msg\_systemState\_t, 65
- host\_msg\_timeSyncReq\_t, 66
- host\_msg\_timeSyncRsp\_t, 67
- host\_msg\_txProgrammed\_t, 69
- host\_msg\_txSdu\_t, 71
- host\_msg\_txSduResult\_t, 73
- host\_msg\_txSduRsp\_t, 75
- host\_msg\_uptimeStatsReq\_t, 77
- host\_msg\_uptimeStatsRsp\_t, 78
- host\_msg\_version\_t, 80
- host\_msg\_versionRsp\_t, 81
- host\_msg\_writeFlashConf\_t, 82
- heading
  - host\_msg\_frameStats\_t, 25
- highCAFC
  - host\_msg\_frameStats\_t, 25
- highTimingOffset
  - host\_msg\_frameStats\_t, 25
- host\_customer\_msg.h
  - HOST\_MSG\_BROADCAST\_STATUS\_-FAILURE\_INVALID\_BCAST\_ID, 94
  - HOST\_MSG\_BROADCAST\_STATUS\_-FAILURE\_OUT\_OF\_RANGE, 93
  - HOST\_MSG\_BROADCAST\_STATUS\_-SUCCESS, 93
  - HOST\_MSG\_ERR\_INVALID\_CMD, 94
  - HOST\_MSG\_FRAME\_STATS\_PREAMBLE, 94
  - HOST\_MSG\_FRAME\_STATS\_-PREAMBLE\_PLUS\_WARM\_DEMOD, 94
  - HOST\_MSG\_FRAME\_STATS\_WARM\_-DEMOD, 94
  - HOST\_MSG\_HOST\_CDLD, 95
  - HOST\_MSG\_HOST\_INTERNAL, 94
  - HOST\_MSG\_HOST\_MAC\_INTERNAL, 95
  - HOST\_MSG\_HOST\_NULL, 94
  - HOST\_MSG\_HOST\_OTA\_DIAG, 95
  - HOST\_MSG\_HOST\_SPI, 94
  - HOST\_MSG\_HOST\_UART, 94
  - HOST\_MSG\_JOIN\_BACKOFF\_TYPE\_-NONE, 95
  - HOST\_MSG\_JOIN\_BACKOFF\_TYPE\_-RAND\_10\_20, 95
  - HOST\_MSG\_JOIN\_NORMAL, 95
  - HOST\_MSG\_JOIN\_TEST, 95
  - HOST\_MSG\_SDU\_FLAGS\_ACKED, 95
  - HOST\_MSG\_TYPE\_ACK, 97
  - HOST\_MSG\_TYPE\_BEGIN\_SW\_UPGR, 96
  - HOST\_MSG\_TYPE\_BEGIN\_SW\_UPGR\_-RSP, 96
  - HOST\_MSG\_TYPE\_BLACKOUT\_END\_-IND, 97
  - HOST\_MSG\_TYPE\_BLACKOUT\_START\_-IND, 97
  - HOST\_MSG\_TYPE\_BROADCAST\_DATA\_-REQ, 97
  - HOST\_MSG\_TYPE\_BROADCAST\_DATA\_-RSP, 97
  - HOST\_MSG\_TYPE\_BROADCAST\_END\_-IND, 97
  - HOST\_MSG\_TYPE\_BROADCAST\_-START\_CNF, 97
  - HOST\_MSG\_TYPE\_BROADCAST\_-START\_IND, 97
  - HOST\_MSG\_TYPE\_CONNECT, 97
  - HOST\_MSG\_TYPE\_ERR, 97
  - HOST\_MSG\_TYPE\_FLUSH\_TXSDU\_-QUEUE, 97
  - HOST\_MSG\_TYPE\_FLUSH\_TXSDU\_-QUEUE\_RSP, 97
  - HOST\_MSG\_TYPE\_FRAME\_STATS, 96
  - HOST\_MSG\_TYPE\_GET\_EXCEPTION\_-BUFFER\_REQ, 96
  - HOST\_MSG\_TYPE\_GET\_EXCEPTION\_-BUFFER\_RSP, 96
  - HOST\_MSG\_TYPE\_GET\_PARAMS, 96
  - HOST\_MSG\_TYPE\_GET\_PARAMS\_RSP, 96
  - HOST\_MSG\_TYPE\_GET\_STATE, 96
  - HOST\_MSG\_TYPE\_GET\_STATE\_RSP, 96
  - HOST\_MSG\_TYPE\_NODE\_SW\_-UPGRADE\_CNF, 97
  - HOST\_MSG\_TYPE\_NODE\_SW\_-UPGRADE\_IND, 97
  - HOST\_MSG\_TYPE\_OTA\_DIAG\_IND, 96
  - HOST\_MSG\_TYPE\_PRE\_UPDATE\_-NOTIFICATION\_IND, 97
  - HOST\_MSG\_TYPE\_PROVISION\_KEYS\_-REQ, 96
  - HOST\_MSG\_TYPE\_PROVISION\_KEYS\_-RSP, 96
  - HOST\_MSG\_TYPE\_READ\_FLASH\_CONF, 96
  - HOST\_MSG\_TYPE\_READ\_FLASH\_-CONF\_RSP, 96
  - HOST\_MSG\_TYPE\_RXSDU, 97
  - HOST\_MSG\_TYPE\_SET\_CHANNEL, 96

- HOST\_MSG\_TYPE\_SET\_GOLD\_CODES, 96
- HOST\_MSG\_TYPE\_SET\_HOST\_ID\_REQ, 97
- HOST\_MSG\_TYPE\_SET\_PRE\_UPDATE\_NOTIFICATION\_REQ, 97
- HOST\_MSG\_TYPE\_SET\_PRE\_UPDATE\_NOTIFICATION\_RSP, 97
- HOST\_MSG\_TYPE\_SET\_SPREADING, 96
- HOST\_MSG\_TYPE\_START\_FRAME\_STATS, 96
- HOST\_MSG\_TYPE\_STOP\_FRAME\_STATS, 96
- HOST\_MSG\_TYPE\_SW\_UPGR2\_BEGIN\_REQ, 96
- HOST\_MSG\_TYPE\_SW\_UPGR2\_BEGIN\_RSP, 96
- HOST\_MSG\_TYPE\_SW\_UPGR2\_CHUNK\_REQ, 96
- HOST\_MSG\_TYPE\_SW\_UPGR2\_CHUNK\_RSP, 96
- HOST\_MSG\_TYPE\_SW\_UPGR2\_END\_REQ, 97
- HOST\_MSG\_TYPE\_SW\_UPGR2\_END\_RSP, 97
- HOST\_MSG\_TYPE\_SYSTEM\_SET\_STATE, 96
- HOST\_MSG\_TYPE\_SYSTEM\_STATE, 96
- HOST\_MSG\_TYPE\_TIME\_SYNC\_REQ, 97
- HOST\_MSG\_TYPE\_TIME\_SYNC\_RSP, 97
- HOST\_MSG\_TYPE\_TX\_PROGRAMMED, 96
- HOST\_MSG\_TYPE\_TXSDU, 97
- HOST\_MSG\_TYPE\_TXSDU\_RESULT, 97
- HOST\_MSG\_TYPE\_TXSDU\_RSP, 97
- HOST\_MSG\_TYPE\_UPTIME\_STATS\_REQ, 96
- HOST\_MSG\_TYPE\_UPTIME\_STATS\_RSP, 96
- HOST\_MSG\_TYPE\_VERSION, 96
- HOST\_MSG\_TYPE\_VERSION\_RSP, 96
- HOST\_MSG\_TYPE\_WRITE\_FLASH\_CONF, 96
- SYS\_MGR\_STATE\_IDLE, 98
- SYS\_MGR\_STATE\_JOINED, 98
- SYS\_MGR\_STATE\_NIL, 98
- SYS\_MGR\_STATE\_SCANNING, 98
- SYS\_MGR\_STATE\_STARTUP, 98
- SYS\_MGR\_STATE\_TRACK, 98
- HOST\_MSG\_BROADCAST\_STATUS\_FAILURE\_INVALID\_BCAST\_ID host\_customer\_msg.h, 94
- HOST\_MSG\_BROADCAST\_STATUS\_FAILURE\_OUT\_OF\_RANGE host\_customer\_msg.h, 93
- HOST\_MSG\_BROADCAST\_STATUS\_SUCCESS host\_customer\_msg.h, 93
- HOST\_MSG\_ERR\_INVALID\_CMD host\_customer\_msg.h, 94
- HOST\_MSG\_FRAME\_STATS\_PREAMBLE host\_customer\_msg.h, 94
- HOST\_MSG\_FRAME\_STATS\_PREAMBLE\_PLUS\_WARM\_DEMOD host\_customer\_msg.h, 94
- HOST\_MSG\_FRAME\_STATS\_WARM\_DEMOD host\_customer\_msg.h, 94
- HOST\_MSG\_HOST\_CDLD host\_customer\_msg.h, 95
- HOST\_MSG\_HOST\_INTERNAL host\_customer\_msg.h, 94
- HOST\_MSG\_HOST\_MAC\_INTERNAL host\_customer\_msg.h, 95
- HOST\_MSG\_HOST\_NULL host\_customer\_msg.h, 94
- HOST\_MSG\_HOST\_OTA\_DIAG host\_customer\_msg.h, 95
- HOST\_MSG\_HOST\_SPI host\_customer\_msg.h, 94
- HOST\_MSG\_HOST\_UART host\_customer\_msg.h, 94
- HOST\_MSG\_JOIN\_BACKOFF\_TYPE\_NONE host\_customer\_msg.h, 95
- HOST\_MSG\_JOIN\_BACKOFF\_TYPE\_RAND\_10\_20 host\_customer\_msg.h, 95
- HOST\_MSG\_JOIN\_NORMAL host\_customer\_msg.h, 95
- HOST\_MSG\_JOIN\_TEST host\_customer\_msg.h, 95
- HOST\_MSG\_SDU\_FLAGS\_ACKED host\_customer\_msg.h, 95
- HOST\_MSG\_TYPE\_ACK host\_customer\_msg.h, 97
- HOST\_MSG\_TYPE\_BEGIN\_SW\_UPGR host\_customer\_msg.h, 96
- HOST\_MSG\_TYPE\_BEGIN\_SW\_UPGR\_RSP host\_customer\_msg.h, 96
- HOST\_MSG\_TYPE\_BLACKOUT\_END\_IND host\_customer\_msg.h, 97
- HOST\_MSG\_TYPE\_BLACKOUT\_START\_IND host\_customer\_msg.h, 97
- HOST\_MSG\_TYPE\_BROADCAST\_DATA\_REQ host\_customer\_msg.h, 97
- HOST\_MSG\_TYPE\_BROADCAST\_DATA\_RSP host\_customer\_msg.h, 97
- HOST\_MSG\_TYPE\_BROADCAST\_END\_IND host\_customer\_msg.h, 97
- HOST\_MSG\_TYPE\_BROADCAST\_START\_CNF

- host\_customer\_msg.h, 97
- HOST\_MSG\_TYPE\_BROADCAST\_START\_IND
  - host\_customer\_msg.h, 97
- HOST\_MSG\_TYPE\_CONNECT
  - host\_customer\_msg.h, 97
- HOST\_MSG\_TYPE\_ERR
  - host\_customer\_msg.h, 97
- HOST\_MSG\_TYPE\_FLUSH\_TXSDU\_QUEUE
  - host\_customer\_msg.h, 97
- HOST\_MSG\_TYPE\_FLUSH\_TXSDU\_QUEUE\_-RSP
  - host\_customer\_msg.h, 97
- HOST\_MSG\_TYPE\_FRAME\_STATS
  - host\_customer\_msg.h, 96
- HOST\_MSG\_TYPE\_GET\_EXCEPTION\_-BUFFER\_REQ
  - host\_customer\_msg.h, 96
- HOST\_MSG\_TYPE\_GET\_EXCEPTION\_-BUFFER\_RSP
  - host\_customer\_msg.h, 96
- HOST\_MSG\_TYPE\_GET\_PARAMS
  - host\_customer\_msg.h, 96
- HOST\_MSG\_TYPE\_GET\_PARAMS\_RSP
  - host\_customer\_msg.h, 96
- HOST\_MSG\_TYPE\_GET\_STATE
  - host\_customer\_msg.h, 96
- HOST\_MSG\_TYPE\_GET\_STATE\_RSP
  - host\_customer\_msg.h, 96
- HOST\_MSG\_TYPE\_NODE\_SW\_UPGRADE\_-CNF
  - host\_customer\_msg.h, 97
- HOST\_MSG\_TYPE\_NODE\_SW\_UPGRADE\_-IND
  - host\_customer\_msg.h, 97
- HOST\_MSG\_TYPE\_OTA\_DIAG\_IND
  - host\_customer\_msg.h, 96
- HOST\_MSG\_TYPE\_PRE\_UPDATE\_-NOTIFICATION\_IND
  - host\_customer\_msg.h, 97
- HOST\_MSG\_TYPE\_PROVISION\_KEYS\_REQ
  - host\_customer\_msg.h, 96
- HOST\_MSG\_TYPE\_PROVISION\_KEYS\_RSP
  - host\_customer\_msg.h, 96
- HOST\_MSG\_TYPE\_READ\_FLASH\_CONF
  - host\_customer\_msg.h, 96
- HOST\_MSG\_TYPE\_READ\_FLASH\_CONF\_RSP
  - host\_customer\_msg.h, 96
- HOST\_MSG\_TYPE\_RXSDU
  - host\_customer\_msg.h, 97
- HOST\_MSG\_TYPE\_SET\_CHANNEL
  - host\_customer\_msg.h, 96
- HOST\_MSG\_TYPE\_SET\_GOLD\_CODES
  - host\_customer\_msg.h, 96
- HOST\_MSG\_TYPE\_SET\_HOST\_ID\_REQ
  - host\_customer\_msg.h, 97
- HOST\_MSG\_TYPE\_SET\_PRE\_UPDATE\_-NOTIFICATION\_REQ
  - host\_customer\_msg.h, 97
- HOST\_MSG\_TYPE\_SET\_PRE\_UPDATE\_-NOTIFICATION\_RSP
  - host\_customer\_msg.h, 97
- HOST\_MSG\_TYPE\_SET\_SPREADING
  - host\_customer\_msg.h, 96
- HOST\_MSG\_TYPE\_START\_FRAME\_STATS
  - host\_customer\_msg.h, 96
- HOST\_MSG\_TYPE\_STOP\_FRAME\_STATS
  - host\_customer\_msg.h, 96
- HOST\_MSG\_TYPE\_SW\_UPGR2\_BEGIN\_REQ
  - host\_customer\_msg.h, 96
- HOST\_MSG\_TYPE\_SW\_UPGR2\_BEGIN\_RSP
  - host\_customer\_msg.h, 96
- HOST\_MSG\_TYPE\_SW\_UPGR2\_CHUNK\_REQ
  - host\_customer\_msg.h, 96
- HOST\_MSG\_TYPE\_SW\_UPGR2\_CHUNK\_RSP
  - host\_customer\_msg.h, 96
- HOST\_MSG\_TYPE\_SW\_UPGR2\_END\_REQ
  - host\_customer\_msg.h, 97
- HOST\_MSG\_TYPE\_SW\_UPGR2\_END\_RSP
  - host\_customer\_msg.h, 97
- HOST\_MSG\_TYPE\_SYSTEM\_SET\_STATE
  - host\_customer\_msg.h, 96
- HOST\_MSG\_TYPE\_SYSTEM\_STATE
  - host\_customer\_msg.h, 96
- HOST\_MSG\_TYPE\_TIME\_SYNC\_REQ
  - host\_customer\_msg.h, 97
- HOST\_MSG\_TYPE\_TIME\_SYNC\_RSP
  - host\_customer\_msg.h, 97
- HOST\_MSG\_TYPE\_TX\_PROGRAMMED
  - host\_customer\_msg.h, 96
- HOST\_MSG\_TYPE\_TXSDU
  - host\_customer\_msg.h, 97
- HOST\_MSG\_TYPE\_TXSDU\_RESULT
  - host\_customer\_msg.h, 97
- HOST\_MSG\_TYPE\_TXSDU\_RSP
  - host\_customer\_msg.h, 97
- HOST\_MSG\_TYPE\_UPTIME\_STATS\_REQ
  - host\_customer\_msg.h, 96
- HOST\_MSG\_TYPE\_UPTIME\_STATS\_RSP
  - host\_customer\_msg.h, 96
- HOST\_MSG\_TYPE\_VERSION
  - host\_customer\_msg.h, 96
- HOST\_MSG\_TYPE\_VERSION\_RSP
  - host\_customer\_msg.h, 96
- HOST\_MSG\_TYPE\_WRITE\_FLASH\_CONF
  - host\_customer\_msg.h, 96
- host\_customer\_msg.h, 86
  - host\_msg\_broadcastStatus\_t, 93
- HOST\_MSG\_DIR\_HOST\_TO\_NODE, 91



- HOST\_MSG\_DIR\_NODE\_TO\_HOST, 91
- HOST\_MSG\_END\_MARKER, 91
- host\_msg\_errCode\_t, 94
- host\_msg\_frameStatsType\_t, 94
- host\_msg\_host\_t, 94
- host\_msg\_joinBackoffType\_t, 95
- host\_msg\_joinType\_t, 95
- HOST\_MSG\_MAX\_HOST\_INTF\_SDU\_SIZE, 91
- HOST\_MSG\_MAX\_SDU\_SIZE, 92
- HOST\_MSG\_MIN\_SDU\_SIZE, 92
- HOST\_MSG\_OVERHEAD\_LEN, 92
- HOST\_MSG\_SDU\_STATUS\_BITS\_ACK\_FAIL, 92
- HOST\_MSG\_SDU\_STATUS\_BITS\_ACK\_SUCCESS, 92
- HOST\_MSG\_SDU\_STATUS\_BITS\_BUFFER\_FULL, 92
- HOST\_MSG\_SDU\_STATUS\_BITS\_DROPPED\_CDLD, 92
- HOST\_MSG\_SDU\_STATUS\_BITS\_DROPPED\_HOST, 92
- HOST\_MSG\_SDU\_STATUS\_BITS\_DROPPED\_MAINTENANCE, 92
- HOST\_MSG\_SDU\_STATUS\_BITS\_DROPPED\_NET\_EXIT, 92
- HOST\_MSG\_SDU\_STATUS\_BITS\_DROPPED\_NOT\_JOINED, 93
- HOST\_MSG\_SDU\_STATUS\_BITS\_OTHER\_ERROR, 93
- HOST\_MSG\_SDU\_STATUS\_BITS\_TRANSMITTED, 93
- host\_msg\_sduFlags\_t, 95
- host\_msg\_systemAirlinkState\_t, 95
- host\_msg\_txsdu\_result\_sdustatus\_t, 93
- host\_msg\_type\_t, 95
- sys\_mgr\_state\_t, 97
- host\_msg\_ack\_t, 8
  - footer, 8
  - header, 8
- host\_msg\_beginSwUpgrade\_t, 9
  - checksum, 9
  - footer, 9
  - header, 9
  - numChunks, 9
- host\_msg\_beginSwUpgradeRsp\_t, 10
  - footer, 10
  - header, 10
  - result, 10
- host\_msg\_blackoutEndInd\_t, 11
  - footer, 11
  - header, 11
  - wasUpdateIntervalSkipped, 11
- host\_msg\_blackoutStartInd\_t, 12
  - durationInSec, 12
  - footer, 12
  - header, 12
  - secUntilStart, 12
- host\_msg\_broadcastDataReq\_t, 13
  - bcastId, 13
  - footer, 13
  - header, 13
  - length, 13
  - offset, 13
- host\_msg\_broadcastDataRsp\_t, 14
  - bcastId, 14
  - footer, 14
  - header, 14
  - length, 14
  - offset, 14
  - payload, 14
  - status, 14
- host\_msg\_broadcastEndInd\_t, 16
  - bcastId, 16
  - footer, 16
  - header, 16
  - length, 16
- host\_msg\_broadcastStartCnf\_t, 17
  - acceptBroadcast, 17
  - bcastId, 17
  - footer, 17
  - header, 17
- host\_msg\_broadcastStartInd\_t, 18
  - bcastId, 18
  - footer, 18
  - header, 18
  - length, 18
  - payload, 18
- host\_msg\_broadcastStatus\_t
  - host\_customer\_msg.h, 93
- host\_msg\_connect\_t, 19
  - connected, 19
  - footer, 19
  - header, 19
- HOST\_MSG\_DIR\_HOST\_TO\_NODE
  - host\_customer\_msg.h, 91
- HOST\_MSG\_DIR\_NODE\_TO\_HOST
  - host\_customer\_msg.h, 91
- HOST\_MSG\_END\_MARKER
  - host\_customer\_msg.h, 91
- host\_msg\_err\_t, 20
  - errCode, 20
  - footer, 20
  - header, 20
- host\_msg\_errCode\_t
  - host\_customer\_msg.h, 94
- host\_msg\_flushTxSduQueue\_t, 21
  - footer, 21

- header, 21
- includeInProgressSdus, 21
- host\_msg\_flushTxSduQueueRsp\_t, 22
  - flushSucceeded, 22
  - footer, 22
  - header, 22
- host\_msg\_frameStats\_t, 23
  - altitude, 24
  - boostedFineAFCMetric, 24
  - center\_freq\_offset, 24
  - channel, 24
  - demodType, 24
  - digitalTruncation, 24
  - failedFrameCnt, 24
  - fingerCAFC, 24
  - fingerEnergy, 24
  - fingerFineAFCs, 25
  - fingerPower, 25
  - fingerTimingOffsetParity, 25
  - footer, 25
  - frameDelaySymbols, 25
  - freqOffset, 25
  - hammingWeight, 25
  - header, 25
  - heading, 25
  - highCAFC, 25
  - highTimingOffset, 25
  - lastDchSpreading, 26
  - lastTxSpreading, 26
  - lastTxSubslot, 26
  - latitude, 26
  - longitude, 26
  - lowCAFC, 26
  - lowTimingOffset, 26
  - numLoggingMsgsDropped, 26
  - oscCal26m, 26
  - oscCal32k, 26
  - RSSI, 27
  - rss\_i\_high, 27
  - rss\_i\_low, 27
  - sfn, 27
  - subslot, 27
  - txFreqStride, 27
  - txTimeTrackingStride, 27
  - txVGA, 27
  - velocity, 27
  - winningFineAFC, 27
- host\_msg\_frameStatsType\_t
  - host\_customer\_msg.h, 94
- host\_msg\_getExceptionBufferReq\_t, 28
  - chunk, 28
  - clearBuffer, 28
  - footer, 28
  - header, 28
- host\_msg\_getExceptionBufferRsp\_t, 29
  - buffer, 29
  - footer, 29
  - header, 29
- host\_msg\_getParamRsp\_t, 30
  - bcastGoldCode, 30
  - bcastSlot, 30
  - bcastSpreading, 30
  - channelBW, 31
  - channelNum, 31
  - cid, 31
  - dataSubslot, 31
  - demodChannel, 31
  - dldataGoldCode, 31
  - dldataSpreading, 31
  - footer, 31
  - header, 32
  - listenInterval, 32
  - maxTxPwrLimit, 32
  - maxTxPwrLimitHeadRoom, 32
  - nodeId, 32
  - numNCAccum, 32
  - pad, 32
  - rssMargin, 32
  - slotInterval, 32
  - systemState, 33
  - ulSpreading, 33
- host\_msg\_getParams\_t, 34
  - footer, 34
  - header, 34
- host\_msg\_getState\_t, 35
  - footer, 35
  - header, 35
- host\_msg\_getStateRsp\_t, 36
  - footer, 36
  - header, 36
  - state, 36
- host\_msg\_header\_t, 37
  - msgLen, 37
  - msgType, 37
- host\_msg\_host\_t
  - host\_customer\_msg.h, 94
- host\_msg\_hostIdReq\_t, 38
  - footer, 38
  - header, 38
  - hostId, 38
- host\_msg\_joinBackoffType\_t
  - host\_customer\_msg.h, 95
- host\_msg\_joinType\_t
  - host\_customer\_msg.h, 95
- HOST\_MSG\_MAX\_HOST\_INTF\_SDU\_SIZE
  - host\_customer\_msg.h, 91
- HOST\_MSG\_MAX\_SDU\_SIZE
  - host\_customer\_msg.h, 92

- HOST\_MSG\_MIN\_SDU\_SIZE
  - host\_customer\_msg.h, 92
- host\_msg\_nodeSwUpgradeCnf\_t, 39
  - footer, 39
  - header, 39
- host\_msg\_nodeSwUpgradeInd\_t, 40
  - footer, 40
  - header, 40
- host\_msg\_otaDiagInd\_t, 41
  - footer, 41
  - header, 41
  - state, 41
- HOST\_MSG\_OVERHEAD\_LEN
  - host\_customer\_msg.h, 92
- host\_msg\_preUpdateNotificationInd\_t, 42
  - footer, 42
  - header, 42
- host\_msg\_provisionKeysReq\_t, 43
  - footer, 43
  - gatewayCldKey, 43
  - gatewayKey, 43
  - header, 43
  - rootKey, 43
- host\_msg\_provisionKeysRsp\_t, 45
  - footer, 45
  - header, 45
- host\_msg\_readFlashConf\_t, 46
  - footer, 46
  - header, 46
- host\_msg\_readFlashConfRsp\_t, 47
  - footer, 47
  - header, 47
- host\_msg\_rxSdu\_t, 48
  - footer, 48
  - header, 48
  - pad, 48
  - payload, 48
  - size, 48
- HOST\_MSG\_SDU\_STATUS\_BITS\_ACK\_FAIL
  - host\_customer\_msg.h, 92
- HOST\_MSG\_SDU\_STATUS\_BITS\_ACK\_-  
SUCCESS
  - host\_customer\_msg.h, 92
- HOST\_MSG\_SDU\_STATUS\_BITS\_BUFFER\_-  
FULL
  - host\_customer\_msg.h, 92
- HOST\_MSG\_SDU\_STATUS\_BITS\_DROPPED\_-  
CDLD
  - host\_customer\_msg.h, 92
- HOST\_MSG\_SDU\_STATUS\_BITS\_DROPPED\_-  
HOST
  - host\_customer\_msg.h, 92
- HOST\_MSG\_SDU\_STATUS\_BITS\_DROPPED\_-  
MAINTENANCE
  - host\_customer\_msg.h, 92
- HOST\_MSG\_SDU\_STATUS\_BITS\_DROPPED\_-  
NET\_EXIT
  - host\_customer\_msg.h, 92
- HOST\_MSG\_SDU\_STATUS\_BITS\_DROPPED\_-  
NOT\_JOINED
  - host\_customer\_msg.h, 93
- HOST\_MSG\_SDU\_STATUS\_BITS\_OTHER\_-  
ERROR
  - host\_customer\_msg.h, 93
- HOST\_MSG\_SDU\_STATUS\_BITS\_-  
TRANSMITTED
  - host\_customer\_msg.h, 93
- host\_msg\_sduFlags\_t
  - host\_customer\_msg.h, 95
- host\_msg\_setChannel\_t, 50
  - channelNum, 50
  - footer, 50
  - header, 50
- host\_msg\_setGoldCode\_t, 51
  - bcastGoldCode, 51
  - dataGoldCode, 51
  - footer, 51
  - header, 51
- host\_msg\_setPreUpdateNotificationReq\_t, 52
  - footer, 52
  - header, 52
  - timeInMs, 52
- host\_msg\_setPreUpdateNotificationRsp\_t, 53
  - footer, 53
  - header, 53
  - result, 53
- host\_msg\_setSpreading\_t, 54
  - dlBcastSpreading, 54
  - footer, 54
  - header, 54
  - ulSpreading, 54
- host\_msg\_startFrameStats\_t, 55
  - footer, 55
  - header, 55
- host\_msg\_stopFrameStats\_t, 56
  - footer, 56
  - header, 56
- host\_msg\_swUpgrade2BeginReq\_t, 57
  - checksum, 57
  - footer, 57
  - header, 57
  - numChunks, 57
- host\_msg\_swUpgrade2BeginRsp\_t, 58
  - footer, 58
  - header, 58
  - result, 58
- host\_msg\_swUpgrade2ChunkReq\_t, 59
  - checksum, 59

- chunk, 59
  - footer, 59
  - header, 59
  - num, 59
- host\_msg\_swUpgrade2ChunkRsp\_t, 61
  - footer, 61
  - header, 61
  - result, 61
- host\_msg\_swUpgrade2EndReq\_t, 62
  - footer, 62
  - header, 62
- host\_msg\_swUpgrade2EndRsp\_t, 63
  - footer, 63
  - header, 63
  - result, 63
- host\_msg\_systemAirlinkState\_t
  - host\_customer\_msg.h, 95
- host\_msg\_systemSetState\_t, 64
  - footer, 64
  - header, 64
  - state, 64
- host\_msg\_systemState\_t, 65
  - footer, 65
  - header, 65
  - state, 65
- host\_msg\_timeSyncReq\_t, 66
  - footer, 66
  - header, 66
- host\_msg\_timeSyncRsp\_t, 67
  - day, 67
  - footer, 67
  - header, 67
  - month, 67
  - rsv1, 68
  - time\_of\_day\_frac, 68
  - time\_of\_day\_whole, 68
  - valid, 68
  - year, 68
- host\_msg\_txProgrammed\_t, 69
  - digitalTruncation, 69
  - footer, 69
  - freqOffset, 69
  - header, 69
  - numLoggingMsgsDropped, 70
  - numSubslots, 70
  - spreading, 70
  - startingSubslot, 70
  - txFreqStride, 70
  - txTimeTrackingStride, 70
  - txVGA, 70
- host\_msg\_txsdu\_result\_sdustatus\_t
  - host\_customer\_msg.h, 93
- host\_msg\_txSdu\_t, 71
  - flags, 71
  - footer, 71
  - header, 71
  - host\_tag, 71
  - pad, 71
  - payload, 72
  - size, 72
- host\_msg\_txSduResult\_t, 73
  - footer, 73
  - header, 73
  - host\_tag, 73
  - sduStatus, 73
- host\_msg\_txSduRsp\_t, 75
  - footer, 75
  - header, 75
  - host\_tag, 75
  - isEnqueued, 75
- host\_msg\_type\_t
  - host\_customer\_msg.h, 95
- host\_msg\_uptimeStatsReq\_t, 77
  - footer, 77
  - header, 77
- host\_msg\_uptimeStatsRsp\_t, 78
  - footer, 78
  - header, 78
  - lastBootWasWatchdog, 78
  - numWdogResets, 78
  - secondsSinceLastBoot, 78
- host\_msg\_version\_t, 80
  - footer, 80
  - header, 80
- host\_msg\_versionRsp\_t, 81
  - footer, 81
  - header, 81
  - phyRev, 81
  - swRev, 81
- host\_msg\_writeFlashConf\_t, 82
  - footer, 82
  - header, 82
- host\_tag
  - host\_msg\_txSdu\_t, 71
  - host\_msg\_txSduResult\_t, 73
  - host\_msg\_txSduRsp\_t, 75
- hostId
  - host\_msg\_hostIdReq\_t, 38
- includeInProgressSdus
  - host\_msg\_flushTxSduQueue\_t, 21
- isEnqueued
  - host\_msg\_txSduRsp\_t, 75
- lastBootWasWatchdog
  - host\_msg\_uptimeStatsRsp\_t, 78
- lastDchSpreading
  - host\_msg\_frameStats\_t, 26

- lastTxSpreading
  - host\_msg\_frameStats\_t, 26
- lastTxSubslot
  - host\_msg\_frameStats\_t, 26
- latitude
  - host\_msg\_frameStats\_t, 26
- length
  - host\_msg\_broadcastDataReq\_t, 13
  - host\_msg\_broadcastDataRsp\_t, 14
  - host\_msg\_broadcastEndInd\_t, 16
  - host\_msg\_broadcastStartInd\_t, 18
- listenInterval
  - host\_msg\_getParamRsp\_t, 32
- longitude
  - host\_msg\_frameStats\_t, 26
- lowCAFC
  - host\_msg\_frameStats\_t, 26
- lowTimingOffset
  - host\_msg\_frameStats\_t, 26
- maxTxPwrLimit
  - host\_msg\_getParamRsp\_t, 32
- maxTxPwrLimitHeadRoom
  - host\_msg\_getParamRsp\_t, 32
- month
  - host\_msg\_timeSyncRsp\_t, 67
- msgLen
  - host\_msg\_header\_t, 37
- msgType
  - host\_msg\_header\_t, 37
- nodeId
  - host\_msg\_getParamRsp\_t, 32
- num
  - host\_msg\_swUpgrade2ChunkReq\_t, 59
- numChunks
  - host\_msg\_beginSwUpgrade\_t, 9
  - host\_msg\_swUpgrade2BeginReq\_t, 57
- numLoggingMsgsDropped
  - host\_msg\_frameStats\_t, 26
  - host\_msg\_txProgrammed\_t, 70
- numNCAccum
  - host\_msg\_getParamRsp\_t, 32
- numSubslots
  - host\_msg\_txProgrammed\_t, 70
- numWdogResets
  - host\_msg\_uptimeStatsRsp\_t, 78
- offset
  - host\_msg\_broadcastDataReq\_t, 13
  - host\_msg\_broadcastDataRsp\_t, 14
- oscCal26m
  - host\_msg\_frameStats\_t, 26
- oscCal32k
  - host\_msg\_frameStats\_t, 26
- pad
  - host\_msg\_getParamRsp\_t, 32
  - host\_msg\_rxSdu\_t, 48
  - host\_msg\_txSdu\_t, 71
- payload
  - host\_msg\_broadcastDataRsp\_t, 14
  - host\_msg\_broadcastStartInd\_t, 18
  - host\_msg\_rxSdu\_t, 48
  - host\_msg\_txSdu\_t, 72
- phyRev
  - host\_msg\_versionRsp\_t, 81
- result
  - host\_msg\_beginSwUpgradeRsp\_t, 10
  - host\_msg\_setPreUpdateNotificationRsp\_t, 53
  - host\_msg\_swUpgrade2BeginRsp\_t, 58
  - host\_msg\_swUpgrade2ChunkRsp\_t, 61
  - host\_msg\_swUpgrade2EndRsp\_t, 63
- rootKey
  - host\_msg\_provisionKeysReq\_t, 43
- RSSI
  - host\_msg\_frameStats\_t, 27
- rsi\_high
  - host\_msg\_frameStats\_t, 27
- rsi\_low
  - host\_msg\_frameStats\_t, 27
- rsiMargin
  - host\_msg\_getParamRsp\_t, 32
- rsv1
  - host\_msg\_timeSyncRsp\_t, 68
- sduStatus
  - host\_msg\_txSduResult\_t, 73
- secondsSinceLastBoot
  - host\_msg\_uptimeStatsRsp\_t, 78
- secUntilStart
  - host\_msg\_blackoutStartInd\_t, 12
- sfn
  - host\_msg\_frameStats\_t, 27
- size
  - host\_msg\_rxSdu\_t, 48
  - host\_msg\_txSdu\_t, 72
- slotInterval
  - host\_msg\_getParamRsp\_t, 32
- spi\_common\_proto.h, 99
- SpiProtoCmd, 83
- spreading
  - host\_msg\_txProgrammed\_t, 70
- startingSubslot
  - host\_msg\_txProgrammed\_t, 70
- state
  - host\_msg\_getStateRsp\_t, 36

- host\_msg\_otaDiagInd\_t, [41](#)
- host\_msg\_systemSetState\_t, [64](#)
- host\_msg\_systemState\_t, [65](#)
- status
  - host\_msg\_broadcastDataRsp\_t, [14](#)
- subslot
  - host\_msg\_frameStats\_t, [27](#)
- swRev
  - host\_msg\_versionRsp\_t, [81](#)
- SYS\_MGR\_STATE\_IDLE
  - host\_customer\_msg.h, [98](#)
- SYS\_MGR\_STATE\_JOINED
  - host\_customer\_msg.h, [98](#)
- SYS\_MGR\_STATE\_NIL
  - host\_customer\_msg.h, [98](#)
- SYS\_MGR\_STATE\_SCANNING
  - host\_customer\_msg.h, [98](#)
- SYS\_MGR\_STATE\_STARTUP
  - host\_customer\_msg.h, [98](#)
- SYS\_MGR\_STATE\_TRACK
  - host\_customer\_msg.h, [98](#)
- sys\_mgr\_state\_t
  - host\_customer\_msg.h, [97](#)
- systemState
  - host\_msg\_getParamRsp\_t, [33](#)
- time\_of\_day\_frac
  - host\_msg\_timeSyncRsp\_t, [68](#)
- time\_of\_day\_whole
  - host\_msg\_timeSyncRsp\_t, [68](#)
- timeInMs
  - host\_msg\_setPreUpdateNotificationReq\_t, [52](#)
- txFreqStride
  - host\_msg\_frameStats\_t, [27](#)
  - host\_msg\_txProgrammed\_t, [70](#)
- txTimeTrackingStride
  - host\_msg\_frameStats\_t, [27](#)
  - host\_msg\_txProgrammed\_t, [70](#)
- txVGA
  - host\_msg\_frameStats\_t, [27](#)
  - host\_msg\_txProgrammed\_t, [70](#)
- ulSpreading
  - host\_msg\_getParamRsp\_t, [33](#)
  - host\_msg\_setSpreading\_t, [54](#)
- valid
  - host\_msg\_timeSyncRsp\_t, [68](#)
- velocity
  - host\_msg\_frameStats\_t, [27](#)
- wasUpdateIntervalSkipped
  - host\_msg\_blackoutEndInd\_t, [11](#)
- winningFineAFC
  - host\_msg\_frameStats\_t, [27](#)
- year
  - host\_msg\_timeSyncRsp\_t, [68](#)