



EmberZNet 5.7.2 API Reference:

For the STM32F103RET Host

April 29, 2016
120-3025-000-5720

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
Tel: 1+(512) 416-8500
Fax: 1+(512) 416-9669
Toll Free: 1+(877) 444-3032
www.silabs.com



Disclaimer

The information in this document is believed to be accurate in all respects at the time of publication but is subject to change without notice. Silicon Laboratories assumes no responsibility for errors or omissions, and disclaims responsibility for the functioning of undescribed features or parameters. Silicon Laboratories makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Silicon Laboratories assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. Silicon Laboratories products are not designed, intended, or authorized for use in applications intended to support or sustain life, or for any other application in which the failure of the Silicon Laboratories product could create a situation where personal injury or death may occur. Should Buyer purchase or use Silicon Laboratories products for any such unintended or unauthorized application, Buyer shall indemnify and hold Silicon Laboratories harmless against all claims and damages.

Silicon Laboratories, Silicon Labs, and Ember are trademarks of Silicon Laboratories Inc.

Other products or brandnames mentioned herein are trademarks or registered trademarks of their respective holders.

About This Guide

Purpose

This document is a unified collection of API reference documentation covering EmberZNet PRO Stack.

Silicon Labs recommends that you use this document as a searchable reference. It includes all of the information contained in the html version of these materials that are provided as an online reference for developers of EmberZNet-based ZigBee wireless applications. There are three key advantages that this document provides over the online html versions:

- Everything is contained in this single document.
- This document is fully searchable using the Adobe Acrobat search engine that is part of the free Acrobat Reader (available from www.adobe.com).
- This document can be easily printed.

Audience

This document is intended for use by programmers and designers developing ZigBee wireless networking products based on the EmberZNet PRO Stack Software. This document assumes that the reader has a solid understanding of embedded systems design and programming in the C language. Experience with networking and radio frequency systems is useful but not expected.

Getting Help

Development kit customers are eligible for training and technical support. You can use the Silicon Labs web site www.silabs.com/zigbee to obtain information about all Ember products and services.

You can also contact customer support at www.silabs.com/zigbee-support.html.

Chapter 1

Introduction

Note

Document 120-3024-000A, *EmberZNet API Reference: For the EM35x Network Co-Processor*, has been obsoleted and superseded by this document with respect to the STM32F103RET Host functionality. PC Host functionality is now documented in 120-3026-000.

The EmberZNet API Reference documentation for the STM32F103RET Host includes the following API sets:

- [Ember Common](#)
- [Hardware Abstraction Layer \(HAL\) API Reference](#)
- [Application Utilities API Reference](#)

Chapter 2

Deprecated List

File [ami-inter-pan-host.h](#)

The ami-inter-pan library is deprecated and will be removed in a future release. Similar functionality is available in the Inter-PAN plugin in Application Framework.

File [ami-inter-pan.h](#)

The ami-inter-pan library is deprecated and will be removed in a future release. Similar functionality is available in the Inter-PAN plugin in Application Framework.

File [fragment-host.h](#)

The fragment library is deprecated and will be removed in a future release. Similar functionality is available in the Fragmentation plugin in Application Framework.

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

Ember Common	13
Ember Common Data Types	14
Sending and Receiving Messages	64
Ember Status Codes	67
Smart Energy Security	90
Configuration	93
Hardware Abstraction Layer (HAL) API Reference	105
HAL Configuration	106
Common PLATFORM_HEADER Configuration	107
STM32F103RET IAR Specific PLATFORM_HEADER Configuration	114
Microcontroller General Functionality	121
STM32F103RET General Functionality	126
ST Microcontroller Standard Peripherals Library Inclusions and Definitions	128
SPI Protocol	129
STM32F103RET Specific SPI Protocol	133
System Timer	137
Sample APIs for Peripheral Access	139
Serial UART Communication	140
STM32F103RET Specific UART	143
ADC Control	144
STM32F103RET Specific ADC	145
Button Control	147
STM32F103RET Specific Button	149
Buzzer Control	152
STM32F103RET Specific Buzzer	153
LED Control	161
STM32F103RET Specific LED	163
Bootloader EEPROM Control	165
HAL Utilities	171
Cyclic Redundancy Code (CRC)	172
Application Utilities API Reference	174

Forming and Joining Networks	175
Bootloading	181
Stand-Alone Bootloader for EZSP	182
Stand-Alone Bootloader Library	186
Command Interpreters	193
Command Interpreter 2	194
ZigBee Device Object (ZDO) Information	202
Message Fragmentation	213
Network Manager	217
Serial Communication	220
Deprecated Files	232

Chapter 4

Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

EmberAesMmoHashContext	This data structure contains the context data when calculating an AES MMO hash (message digest)	233
EmberApsFrame	An in-memory representation of a ZigBee APS frame of an incoming or outgoing message	234
EmberBindingTableEntry	Defines an entry in the binding table	235
EmberCertificate283k1Data	This data structure contains the certificate data that is used for Certificate Based Key Exchange (CBKE) in SECT283k1 Elliptical Cryptography	236
EmberCertificateData	This data structure contains the certificate data that is used for Certificate Based Key Exchange (CBKE)	237
EmberCommandEntry	Command entry for a command table	237
EmberCurrentSecurityState	This describes the security features used by the stack for a joined device	239
EmberEventControl	Control structure for events	240
EmberEventData_S	Complete events with a control and a handler procedure	240
EmberInitialSecurityState	This describes the Initial Security features and requirements that will be used when forming or joining the network	241
EmberKeyData	This data structure contains the key data that is passed into various other functions	243
EmberKeyStruct	This describes a one of several different types of keys and its associated data	243

EmberMacFilterMatchStruct	
This structure indicates a matching raw MAC message has been received by the application configured MAC filters	245
EmberMessageDigest	
This data structure contains an AES-MMO Hash (the message digest)	245
EmberMfgSecurityStruct	
This structure is used to get/set the security config that is stored in manufacturing tokens	246
EmberMulticastTableEntry	
Defines an entry in the multicast table	246
EmberNeighborTableEntry	
Defines an entry in the neighbor table	247
EmberNetworkInitStruct	
Defines the network initialization configuration that should be used when ::emberNetworkInitExtended() is called by the application	249
EmberNetworkParameters	
Holds network parameters	249
EmberPrivateKey283k1Data	
This data structure contains the private key data that is used for Certificate Based Key Exchange (CBKE) in SECT283k1 Elliptical Cryptography	251
EmberPrivateKeyData	
This data structure contains the private key data that is used for Certificate Based Key Exchange (CBKE)	251
EmberPublicKey283k1Data	
This data structure contains the public key data that is used for Certificate Based Key Exchange (CBKE) in SECT283k1 Elliptical Cryptography	252
EmberPublicKeyData	
This data structure contains the public key data that is used for Certificate Based Key Exchange (CBKE)	253
EmberReleaseTypeStruct	
A structure relating version types to human readable strings	253
EmberRouteTableEntry	
Defines an entry in the route table	254
EmberSignature283k1Data	
This data structure contains a DSA signature used in SECT283k1 Elliptical Cryptography. It is the bit concatenation of the 'r' and 's' components of the signature	255
EmberSignatureData	
This data structure contains a DSA signature. It is the bit concatenation of the 'r' and 's' components of the signature	256
EmberSmacData	
This data structure contains the Shared Message Authentication Code (SMAC) data that is used for Certificate Based Key Exchange (CBKE)	256
EmberTaskControl	
Control structure for tasks	257
EmberVersion	
Version struct containing all version information	257
EmberZigbeeNetwork	
Defines a ZigBee network and the associated parameters	258
HalEepromInformationType	
This structure defines a variety of information about the attached external EEPROM device	259

InterPanHeader	
A struct for keeping track of all of the header info	261

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

_STM32F103RET_Host_API.top	Starting page for the Ember API documentation for the STM32F103R-ET Host, exclusively for building documentation	263
adc.h		264
ami-inter-pan-host.h	Utilities for sending and receiving ZigBee AMI InterPAN messages. See Sending and Receiving Messages for documentation	265
ami-inter-pan.h	Utilities for sending and receiving ZigBee AMI InterPAN messages. See Sending and Receiving Messages for documentation	266
bootload-ezsp-utils.h	Utilities used for performing stand-alone bootloading over EZSP. See Bootloading for documentation	268
bootload-utils.h	Utilities used for performing stand-alone bootloading. See Bootloading for documentation	270
bootloader-eeeprom.h		272
button-common.h		274
button-specific.h		275
buzzer.h		277
cbke-crypto-engine.h	EmberZNet Smart Energy security API. See Smart Energy Security for documentation	280
command-interpret2.h	Processes commands coming from the serial port. See Command Interpreter 2 for documentation	286
crc.h		289
ember-configuration-defaults.h	User-configurable stack memory allocation defaults	291
ember-types.h	Ember data type definitions	310
error-def.h	Return-code definitions for EmberZNet stack API functions	333

error.h	Return codes for Ember API functions and module definitions	346
ezsp-host-configuration-defaults.h	User-configurable parameters for host applications	347
form-and-join.h	Utilities for forming and joining networks	348
fragment-host.h	Fragmented message support for EZSP Hosts. Splits long messages into smaller blocks for transmission and reassembles received blocks. See Message Fragmentation for documentation	349
hal.h	Generic set of HAL includes for all platforms	351
iar-st.h	355
led-common.h	358
led-specific.h	359
micro-common.h	360
micro-specific.h	362
network-manager.h	Utilities for use by the ZigBee network manager. See Network Manager for documentation	363
platform-common.h	365
hal/host/serial.h	368
app/util/serial/serial.h	High-level serial communication functions	370
spi-protocol-common.h	372
spi-protocol-specific.h	373
stm32f10x_conf.h	375
system-timer.h	376
uart.h	377
zigbee-device-common.h	ZigBee Device Object (ZDO) functions available on all platforms. See ZigBee Device Object (ZDO) Information for documentation	379
zigbee-device-host.h	ZigBee Device Object (ZDO) functions not provided by the stack. See ZigBee Device Object (ZDO) Information for documentation	382

Chapter 6

Module Documentation

6.1 Ember Common

Modules

- [Ember Common Data Types](#)
- [Sending and Receiving Messages](#)
- [Ember Status Codes](#)
- [Smart Energy Security](#)
- [Configuration](#)

6.1.1 Detailed Description

6.2 Ember Common Data Types

Data Structures

- struct [EmberReleaseTypeStruct](#)
A structure relating version types to human readable strings.
- struct [EmberVersion](#)
Version struct containing all version information.
- struct [EmberZigbeeNetwork](#)
Defines a ZigBee network and the associated parameters.
- struct [EmberNetworkInitStruct](#)
Defines the network initialization configuration that should be used when `::emberNetworkInitExtended()` is called by the application.
- struct [EmberNetworkParameters](#)
Holds network parameters.
- struct [EmberApsFrame](#)
An in-memory representation of a ZigBee APS frame of an incoming or outgoing message.
- struct [EmberBindingTableEntry](#)
Defines an entry in the binding table.
- struct [EmberNeighborTableEntry](#)
Defines an entry in the neighbor table.
- struct [EmberRouteTableEntry](#)
Defines an entry in the route table.
- struct [EmberMulticastTableEntry](#)
Defines an entry in the multicast table.
- struct [EmberEventControl](#)
Control structure for events.
- struct [EmberEventData_S](#)
Complete events with a control and a handler procedure.
- struct [EmberTaskControl](#)
Control structure for tasks.
- struct [EmberKeyData](#)
This data structure contains the key data that is passed into various other functions.
- struct [EmberCertificateData](#)
This data structure contains the certificate data that is used for Certificate Based Key Exchange (CBKE).
- struct [EmberPublicKeyData](#)
This data structure contains the public key data that is used for Certificate Based Key Exchange (CBKE).
- struct [EmberPrivateKeyData](#)
This data structure contains the private key data that is used for Certificate Based Key Exchange (CBKE).
- struct [EmberSmacData](#)
This data structure contains the Shared Message Authentication Code (SMAC) data that is used for Certificate Based Key Exchange (CBKE).
- struct [EmberSignatureData](#)
This data structure contains a DSA signature. It is the bit concatenation of the 'r' and 's' components of the signature.

- struct [EmberMessageDigest](#)
This data structure contains an AES-MMO Hash (the message digest).
- struct [EmberAesMmoHashContext](#)
This data structure contains the context data when calculating an AES MMO hash (message digest).
- struct [EmberCertificate283k1Data](#)
This data structure contains the certificate data that is used for Certificate Based Key Exchange (CBKE) in SECT283k1 Elliptical Cryptography.
- struct [EmberPublicKey283k1Data](#)
This data structure contains the public key data that is used for Certificate Based Key Exchange (CBKE) in SECT283k1 Elliptical Cryptography.
- struct [EmberPrivateKey283k1Data](#)
This data structure contains the private key data that is used for Certificate Based Key Exchange (CBKE) in SECT283k1 Elliptical Cryptography.
- struct [EmberSignature283k1Data](#)
This data structure contains a DSA signature used in SECT283k1 Elliptical Cryptography. It is the bit concatenation of the 'r' and 's' components of the signature.
- struct [EmberInitialSecurityState](#)
This describes the Initial Security features and requirements that will be used when forming or joining the network.
- struct [EmberCurrentSecurityState](#)
This describes the security features used by the stack for a joined device.
- struct [EmberKeyStruct](#)
This describes a one of several different types of keys and its associated data.
- struct [EmberMfgSecurityStruct](#)
This structure is used to get/set the security config that is stored in manufacturing tokens.
- struct [EmberMacFilterMatchStruct](#)
This structure indicates a matching raw MAC message has been received by the application configured MAC filters.

Macros

- #define [EMBER_MIN_BROADCAST_ADDRESS](#)
- #define [emberIsZigbeeBroadcastAddress\(address\)](#)
- #define [EMBER_JOIN_DECISION_STRINGS](#)
- #define [EMBER_DEVICE_UPDATE_STRINGS](#)
- #define [emberInitializeNetworkParameters\(parameters\)](#)
- #define [EMBER_COUNTER_STRINGS](#)
- #define [EMBER_STANDARD_SECURITY_MODE](#)
- #define [EMBER_TRUST_CENTER_NODE_ID](#)
- #define [EMBER_NO_TRUST_CENTER_MODE](#)
- #define [EMBER_GLOBAL_LINK_KEY](#)
- #define [EMBER_MFG_SECURITY_CONFIG_MAGIC_NUMBER](#)
- #define [EMBER_MAC_FILTER_MATCH_ENABLED_MASK](#)
- #define [EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_MASK](#)
- #define [EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_MASK](#)
- #define [EMBER_MAC_FILTER_MATCH_ON_DEST_MASK](#)
- #define [EMBER_MAC_FILTER_MATCH_ON_SOURCE_MASK](#)
- #define [EMBER_MAC_FILTER_MATCH_ENABLED](#)

- #define `EMBER_MAC_FILTER_MATCH_DISABLED`
- #define `EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_NONE`
- #define `EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_LOCAL`
- #define `EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_BROADCAST`
- #define `EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_NONE`
- #define `EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_NON_LOCAL`
- #define `EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_LOCAL`
- #define `EMBER_MAC_FILTER_MATCH_ON_DEST_BROADCAST_SHORT`
- #define `EMBER_MAC_FILTER_MATCH_ON_DEST_UNICAST_SHORT`
- #define `EMBER_MAC_FILTER_MATCH_ON_DEST_UNICAST_LONG`
- #define `EMBER_MAC_FILTER_MATCH_ON_SOURCE_LONG`
- #define `EMBER_MAC_FILTER_MATCH_ON_SOURCE_SHORT`
- #define `EMBER_MAC_FILTER_MATCH_ON_SOURCE_NONE`
- #define `EMBER_MAC_FILTER_MATCH_END`
- #define `WEAK_TEST`

Typedefs

- typedef uint8_t `EmberTaskId`
- typedef PGM struct `EmberEventData_S` `EmberEventData`
- typedef uint16_t `EmberMacFilterMatchData`
- typedef uint8_t `EmberLibraryStatus`

Enumerations

- enum `EmberNodeType` {
`EMBER_UNKNOWN_DEVICE`, `EMBER_COORDINATOR`, `EMBER_ROUTER`,
`EMBER_END_DEVICE`,
`EMBER_SLEEPY_END_DEVICE`, `EMBER_MOBILE_END_DEVICE`, `EMBER-`
`RF4CE_TARGET`, `EMBER_RF4CE_CONTROLLER` }
- enum `EmberEndDeviceConfiguration` { `EMBER_END_DEVICE_CONFIG_NON-`
`E`, `EMBER_END_DEVICE_CONFIG_PERSIST_DATA_ON_PARENT` }
- enum `EmberNetworkInitBitmask` { `EMBER_NETWORK_INIT_NO_OPTIONS`, `E-`
`MBER_NETWORK_INIT_PARENT_INFO_IN_TOKEN` }
- enum `EmberApsOption` {
`EMBER_APS_OPTION_NONE`, `EMBER_APS_OPTION_DSA_SIGN`, `EMBER-`
`_APS_OPTION_ENCRYPTION`, `EMBER_APS_OPTION_RETRY`,
`EMBER_APS_OPTION_ENABLE_ROUTE_DISCOVERY`, `EMBER_APS_OPTI-`
`ON_FORCE_ROUTE_DISCOVERY`, `EMBER_APS_OPTION_SOURCE_EUI64`,
`EMBER_APS_OPTION_DESTINATION_EUI64`,
`EMBER_APS_OPTION_ENABLE_ADDRESS_DISCOVERY`, `EMBER_APS_O-`
`PTION_POLL_RESPONSE`, `EMBER_APS_OPTION_ZDO_RESPONSE_REQU-`
`IRED`, `EMBER_APS_OPTION_FRAGMENT` }
- enum `EmberIncomingMessageType` {
`EMBER_INCOMING_UNICAST`, `EMBER_INCOMING_UNICAST_REPLY`, `E-`
`MBER_INCOMING_MULTICAST`, `EMBER_INCOMING_MULTICAST_LOOP-`
`BACK`,
`EMBER_INCOMING_BROADCAST`, `EMBER_INCOMING_BROADCAST_LO-`
`OPBACK` }

- enum EmberOutgoingMessageType {
EMBER_OUTGOING_DIRECT, EMBER_OUTGOING_VIA_ADDRESS_TABLE,
EMBER_OUTGOING_VIA_BINDING, EMBER_OUTGOING_MULTICAST,
EMBER_OUTGOING_MULTICAST_WITH_ALIAS, EMBER_OUTGOING_BROADCAST_WITH_ALIAS,
EMBER_OUTGOING_BROADCAST }
- enum EmberZigbeeCommandType {
EMBER_ZIGBEE_COMMAND_TYPE_MAC, EMBER_ZIGBEE_COMMAND_TYPE_NWK,
EMBER_ZIGBEE_COMMAND_TYPE_APS, EMBER_ZIGBEE_COMMAND_TYPE_ZDO,
EMBER_ZIGBEE_COMMAND_TYPE_ZCL, EMBER_ZIGBEE_COMMAND_TYPE_BEACON }
- enum EmberNetworkStatus {
EMBER_NO_NETWORK, EMBER_JOINING_NETWORK, EMBER_JOINED_NETWORK,
EMBER_JOINED_NETWORK_NO_PARENT, EMBER_LEAVING_NETWORK }
- enum EmberNetworkScanType { EMBER_ENERGY_SCAN, EMBER_ACTIVE_SCAN }
- enum EmberBindingType { EMBER_UNUSED_BINDING, EMBER_UNICAST_BINDING,
EMBER_MANY_TO_ONE_BINDING, EMBER_MULTICAST_BINDING }
- enum EmberJoinDecision { EMBER_USE_PRECONFIGURED_KEY, EMBER_SEND_KEY_IN_THE_CLEAR,
EMBER_DENY_JOIN, EMBER_NO_ACTION }
- enum EmberDeviceUpdate {
EMBER_STANDARD_SECURITY_SECURED_REJOIN, EMBER_STANDARD_SECURITY_UNSECURED_JOIN,
EMBER_DEVICE_LEFT, EMBER_STANDARD_SECURITY_UNSECURED_REJOIN,
EMBER_HIGH_SECURITY_SECURED_REJOIN, EMBER_HIGH_SECURITY_UNSECURED_JOIN,
EMBER_HIGH_SECURITY_UNSECURED_REJOIN }
- enum EmberRejoinReason {
EMBER_REJOIN_REASON_NONE, EMBER_REJOIN_DUE_TO_NWK_KEY_UPDATE,
EMBER_REJOIN_DUE_TO_LEAVE_MESSAGE, EMBER_REJOIN_DUE_TO_NO_PARENT,
EMBER_REJOIN_DUE_TO_ZLL_TOUCHLINK, EMBER_REJOIN_DUE_TO_APP_EVENT_5,
EMBER_REJOIN_DUE_TO_APP_EVENT_4, EMBER_REJOIN_DUE_TO_APP_EVENT_3,
EMBER_REJOIN_DUE_TO_APP_EVENT_2, EMBER_REJOIN_DUE_TO_APP_EVENT_1 }
- enum EmberClusterListId { EMBER_INPUT_CLUSTER_LIST, EMBER_OUTPUT_CLUSTER_LIST }
- enum EmberEventUnits {
EMBER_EVENT_INACTIVE, EMBER_EVENT_MS_TIME, EMBER_EVENT_QS_TIME,
EMBER_EVENT_MINUTE_TIME, EMBER_EVENT_ZERO_DELAY }
- enum EmberJoinMethod { EMBER_USE_MAC_ASSOCIATION, EMBER_USE_NWK_REJOIN,
EMBER_USE_NWK_REJOIN_HAVE_NWK_KEY, EMBER_USE_NWK_COMMISSIONING }
- enum EmberCounterType {
EMBER_COUNTER_MAC_RX_BROADCAST, EMBER_COUNTER_MAC_TX_BROADCAST,
EMBER_COUNTER_MAC_RX_UNICAST, EMBER_COUNTER_MAC_TX_UNICAST_SUCCESS,
EMBER_COUNTER_MAC_TX_UNICAST_RETRY, EMBER_COUNTER_MAC_TX_UNICAST_FAILED,
EMBER_COUNTER_APS_DATA_RX_BROADCAST }

```

ST, EMBER_COUNTER_APS_DATA_TX_BROADCAST,
EMBER_COUNTER_APS_DATA_RX_UNICAST, EMBER_COUNTER_APS_D-
ATA_TX_UNICAST_SUCCESS, EMBER_COUNTER_APS_DATA_TX_UNICA-
ST_RETRY, EMBER_COUNTER_APS_DATA_TX_UNICAST_FAILED,
EMBER_COUNTER_ROUTE_DISCOVERY_INITIATED, EMBER_COUNTER-
_NEIGHBOR_ADDED, EMBER_COUNTER_NEIGHBOR_REMOVED, EMBE-
R_COUNTER_NEIGHBOR_STALE,
EMBER_COUNTER_JOIN_INDICATION, EMBER_COUNTER_CHILD_REMO-
VED, EMBER_COUNTER_ASH_OVERFLOW_ERROR, EMBER_COUNTER_-
ASH_FRAMING_ERROR,
EMBER_COUNTER_ASH_OVERRUN_ERROR, EMBER_COUNTER_NWK_F-
RAME_COUNTER_FAILURE, EMBER_COUNTER_APS_FRAME_COUNTER-
_FAILURE, EMBER_COUNTER_ASH_XOFF,
EMBER_COUNTER_APS_LINK_KEY_NOT_AUTHORIZED, EMBER_COUN-
TER_NWK_DECRYPTION_FAILURE, EMBER_COUNTER_APS_DECRYPTI-
ON_FAILURE, EMBER_COUNTER_ALLOCATE_PACKET_BUFFER_FAILU-
RE,
EMBER_COUNTER_RELAYED_UNICAST, EMBER_COUNTER_PHY_TO_M-
AC_QUEUE_LIMIT_REACHED, EMBER_COUNTER_PACKET_VALIDATE_-
LIBRARY_DROPPED_COUNT, EMBER_COUNTER_TYPE_NWK_RETRY_O-
VERFLOW,
EMBER_COUNTER_PHY_CCA_FAIL_COUNT, EMBER_COUNTER_BROAD-
CAST_TABLE_FULL, EMBER_COUNTER_TYPE_COUNT }
• enum EmberInitialSecurityBitmask {
    EMBER_DISTRIBUTED_TRUST_CENTER_MODE, EMBER_TRUST_CENTE-
    R_GLOBAL_LINK_KEY, EMBER_PRECONFIGURED_NETWORK_KEY_MO-
    DE, EMBER_HAVE_TRUST_CENTER_EUI64,
    EMBER_TRUST_CENTER_USES_HASHED_LINK_KEY, EMBER_HAVE_PR-
    ECONFIGURED_KEY, EMBER_HAVE_NETWORK_KEY, EMBER_GET_LIN-
    K_KEY_WHEN_JOINING,
    EMBER_REQUIRE_ENCRYPTED_KEY, EMBER_NO_FRAME_COUNTER_R-
    ESET, EMBER_GET_PRECONFIGURED_KEY_FROM_INSTALL_CODE }
• enum EmberExtendedSecurityBitmask { EMBER_JOINER_GLOBAL_LINK_KE-
    Y, EMBER_EXT_NO_FRAME_COUNTER_RESET, EMBER_NWK_LEAVE_R-
    EQUEST_NOT_ALLOWED }
• enum EmberCurrentSecurityBitmask {
    EMBER_STANDARD_SECURITY_MODE_, EMBER_DISTRIBUTED_TRUST-
    _CENTER_MODE_, EMBER_TRUST_CENTER_GLOBAL_LINK_KEY_, EM-
    BER_HAVE_TRUST_CENTER_LINK_KEY,
    EMBER_TRUST_CENTER_USES_HASHED_LINK_KEY_ }
• enum EmberKeyStructBitmask {
    EMBER_KEY_HAS_SEQUENCE_NUMBER, EMBER_KEY_HAS_OUTGOING-
    _FRAME_COUNTER, EMBER_KEY_HAS_INCOMING_FRAME_COUNTER, E-
    MBER_KEY_HAS_PARTNER_EUI64,
    EMBER_KEY_IS_AUTHORIZED, EMBER_KEY_PARTNER_IS_SLEEPY }
• enum EmberKeyType {
    EMBER_TRUST_CENTER_LINK_KEY, EMBER_TRUST_CENTER_MASTER-
    _KEY, EMBER_CURRENT_NETWORK_KEY, EMBER_NEXT_NETWORK_K-
    EY,
    EMBER_APPLICATION_LINK_KEY, EMBER_APPLICATION_MASTER_KE-
    Y }
• enum EmberKeyStatus {
    EMBER_KEY_STATUS_NONE, EMBER_APP_LINK_KEY_ESTABLISHED, E-

```

- EMBER_APP_MASTER_KEY_ESTABLISHED, EMBER_TRUST_CENTER_LINK_KEY_ESTABLISHED,
- EMBER_KEY_ESTABLISHMENT_TIMEOUT, EMBER_KEY_TABLE_FULL, EMBER_TC_RESPONDED_TO_KEY_REQUEST, EMBER_TC_APP_KEY_SENT_TO_REQUESTER,
- EMBER_TC_RESPONSE_TO_KEY_REQUEST_FAILED, EMBER_TC_REQUEST_KEY_TYPE_NOT_SUPPORTED, EMBER_TC_NO_LINK_KEY_FOR_REQUESTER, EMBER_TC_REQUESTER_EUI64_UNKNOWN,
- EMBER_TC_RECEIVED_FIRST_APP_KEY_REQUEST, EMBER_TC_TIMEOUT_WAITING_FOR_SECOND_APP_KEY_REQUEST, EMBER_TC_NON_MATCHING_APP_KEY_REQUEST_RECEIVED, EMBER_TC_FAILED_TO_SEND_APP_KEYS,
- EMBER_TC_FAILED_TO_STORE_APP_KEY_REQUEST, EMBER_TC_REJECTED_APP_KEY_REQUEST, EMBER_TC_FAILED_TO_GENERATE_NEW_KEY, EMBER_TC_FAILED_TO_SEND_TC_KEY,
- EMBER_TRUST_CENTER_IS_PRE_R21, EMBER_TC_REQUESTER_VERIFY_KEY_TIMEOUT, EMBER_TC_REQUESTER_VERIFY_KEY_FAILURE, EMBER_TC_REQUESTER_VERIFY_KEY_SUCCESS,
- EMBER_VERIFY_LINK_KEY_FAILURE, EMBER_VERIFY_LINK_KEY_SUCCESS }
- enum EmberLinkKeyRequestPolicy { EMBER_DENY_KEY_REQUESTS, EMBER_ALLOW_KEY_REQUESTS, EMBER_GENERATE_NEW_TC_LINK_KEY }
- enum EmberKeySettings { EMBER_KEY_PERMISSIONS_NONE, EMBER_KEY_PERMISSIONS_READING_ALLOWED, EMBER_KEY_PERMISSIONS_HASHING_ALLOWED }
- enum EmberMacPassthroughType { EMBER_MAC_PASSTHROUGH_NONE, EMBER_MAC_PASSTHROUGH_SE_INTERPAN, EMBER_MAC_PASSTHROUGH_EMBERNET, EMBER_MAC_PASSTHROUGH_EMBERNET_SOURCE, EMBER_MAC_PASSTHROUGH_APPLICATION, EMBER_MAC_PASSTHROUGH_CUSTOM }

Functions

- uint8_t * emberKeyContents (EmberKeyData *key)
- uint8_t * emberCertificateContents (EmberCertificateData *cert)
- uint8_t * emberPublicKeyContents (EmberPublicKeyData *key)
- uint8_t * emberPrivateKeyContents (EmberPrivateKeyData *key)
- uint8_t * emberSmacContents (EmberSmacData *key)
- uint8_t * emberSignatureContents (EmberSignatureData *sig)
- uint8_t * emberCertificate283k1Contents (EmberCertificate283k1Data *cert)
- uint8_t * emberPublicKey283k1Contents (EmberPublicKey283k1Data *key)
- uint8_t * emberPrivateKey283k1Contents (EmberPrivateKey283k1Data *key)
- uint8_t * ember283k1SignatureContents (Ember283k1SignatureData *sig)

Miscellaneous Ember Types

- enum EmberVersionType { EMBER_VERSION_TYPE_PRE_RELEASE, EMBER_VERSION_TYPE_ALPHA_1, EMBER_VERSION_TYPE_ALPHA_2, EMBER_VERSION_TYPE_ALPHA_3 }

```

A_3,
EMBER_VERSION_TYPE_BETA_1, EMBER_VERSION_TYPE_BETA_2, EMBER_VERSION_TYPE_BETA_3, EMBER_VERSION_TYPE_GA }
• enum EmberLeaveRequestFlags { EMBER_ZIGBEE_LEAVE_AND_REJOIN, EMBER_ZIGBEE_LEAVE_AND_REMOVE_CHILDREN }
• enum EmberLeaveReason {
    EMBER_LEAVE_REASON_NONE, EMBER_LEAVE_DUE_TO_NWK_LEAVE_MESSAGE, EMBER_LEAVE_DUE_TO_APS_REMOVE_MESSAGE, EMBER_LEAVE_DUE_TO_ZDO_LEAVE_MESSAGE,
    EMBER_LEAVE_DUE_TO_ZLL_TOUCHLINK, EMBER_LEAVE_DUE_TO_APP_EVENT_1 }
• typedef uint8_t EmberStatus
• typedef uint8_t EmberEUI64 [EUI64_SIZE]
• typedef uint8_t EmberMessageBuffer
• typedef uint16_t EmberNodeId
• typedef uint16_t EmberMulticastId
• typedef uint16_t EmberPanId
• const EmberVersion emberVersion
• #define EMBER_RELEASE_TYPE_TO_STRING_STRUCT_DATA
• #define EUI64_SIZE
• #define EXTENDED_PAN_ID_SIZE
• #define EMBER_ENCRYPTION_KEY_SIZE
• #define EMBER_CERTIFICATE_SIZE
• #define EMBER_PUBLIC_KEY_SIZE
• #define EMBER_PRIVATE_KEY_SIZE
• #define EMBER_SMAC_SIZE
• #define EMBER_SIGNATURE_SIZE
• #define EMBER_AES_HASH_BLOCK_SIZE
• #define EMBER_CERTIFICATE_283K1_SIZE
• #define EMBER_PUBLIC_KEY_283K1_SIZE
• #define EMBER_PRIVATE_KEY_283K1_SIZE
• #define EMBER_SIGNATURE_283K1_SIZE
• #define __EMBERSTATUS_TYPE__
• #define EMBER_MAX_802_15_4_CHANNEL_NUMBER
• #define EMBER_MIN_802_15_4_CHANNEL_NUMBER
• #define EMBER_NUM_802_15_4_CHANNELS
• #define EMBER_ALL_802_15_4_CHANNELS_MASK
• #define EMBER_ZIGBEE_COORDINATOR_ADDRESS
• #define EMBER_NULL_NODE_ID
• #define EMBER_NULL_BINDING
• #define EMBER_TABLE_ENTRY_UNUSED_NODE_ID
• #define EMBER_MULTICAST_NODE_ID
• #define EMBER_UNKNOWN_NODE_ID
• #define EMBER_DISCOVERY_ACTIVE_NODE_ID
• #define EMBER_NULL_ADDRESS_TABLE_INDEX
• #define EMBER_ZDO_ENDPOINT
• #define EMBER_BROADCAST_ENDPOINT
• #define EMBER_ZDO_PROFILE_ID
• #define EMBER_WILDCARD_PROFILE_ID
• #define EMBER_MAXIMUM_STANDARD_PROFILE_ID
• #define EMBER_BROADCAST_TABLE_TIMEOUT_QS
• #define EMBER_MANUFACTURER_ID

```

ZigBee Broadcast Addresses

ZigBee specifies three different broadcast addresses that reach different collections of nodes. Broadcasts are normally sent only to routers. Broadcasts can also be forwarded to end devices, either all of them or only those that do not sleep. Broadcasting to end devices is both significantly more resource-intensive and significantly less reliable than broadcasting to routers.

- #define `EMBER_BROADCAST_ADDRESS`
- #define `EMBER_RX_ON_WHEN_IDLE_BROADCAST_ADDRESS`
- #define `EMBER_SLEEPY_BROADCAST_ADDRESS`

Ember Concentrator Types

- #define `EMBER_LOW_RAM_CONCENTRATOR`
- #define `EMBER_HIGH_RAM_CONCENTRATOR`

txPowerModes for emberSetTxPowerMode and mfglibSetPower

- #define `EMBER_TX_POWER_MODE_DEFAULT`
- #define `EMBER_TX_POWER_MODE_BOOST`
- #define `EMBER_TX_POWER_MODE_ALTERNATE`
- #define `EMBER_TX_POWER_MODE_BOOST_AND_ALTERNATE`

Alarm Message and Counters Request Definitions

- #define `EMBER_PRIVATE_PROFILE_ID`
- #define `EMBER_PRIVATE_PROFILE_ID_START`
- #define `EMBER_PRIVATE_PROFILE_ID_END`
- #define `EMBER_BROADCAST_ALARM_CLUSTER`
- #define `EMBER_UNICAST_ALARM_CLUSTER`
- #define `EMBER_CACHED_UNICAST_ALARM_CLUSTER`
- #define `EMBER_REPORT_COUNTERS_REQUEST`
- #define `EMBER_REPORT_COUNTERS_RESPONSE`
- #define `EMBER_REPORT_AND_CLEAR_COUNTERS_REQUEST`
- #define `EMBER_REPORT_AND_CLEAR_COUNTERS_RESPONSE`
- #define `EMBER_OTA_CERTIFICATE_UPGRADE_CLUSTER`

ZDO response status.

Most responses to ZDO commands contain a status byte. The meaning of this byte is defined by the ZigBee Device Profile.

- enum `EmberZdoStatus` {
`EMBER_ZDP_SUCCESS`, `EMBER_ZDP_INVALID_REQUEST_TYPE`, `EMBER_ZDP_DEVICE_NOT_FOUND`, `EMBER_ZDP_INVALID_ENDPOINT`,
`EMBER_ZDP_NOT_ACTIVE`, `EMBER_ZDP_NOT_SUPPORTED`, `EMBER_ZDP_TIMEOUT`, `EMBER_ZDP_NO_MATCH`,
`EMBER_ZDP_NO_ENTRY`, `EMBER_ZDP_NO_DESCRIPTOR`, `EMBER_ZDP_`

```

INSUFFICIENT_SPACE, EMBER_ZDP_NOT_PERMITTED,
EMBER_ZDP_TABLE_FULL, EMBER_ZDP_NOT_AUTHORIZED, EMBER_N-
WK_ALREADY_PRESENT, EMBER_NWK_TABLE_FULL,
EMBER_NWK_UNKNOWN_DEVICE }

```

Network and IEEE Address Request/Response

Defines for ZigBee device profile cluster IDs follow. These include descriptions of the formats of the messages.

Note that each message starts with a 1-byte transaction sequence number. This sequence number is used to match a response command frame to the request frame that it is replying to. The application shall maintain a 1-byte counter that is copied into this field and incremented by one for each command sent. When a value of 0xff is reached, the next command shall re-start the counter with a value of 0x00

```

Network request: <transaction sequence number: 1>
                  <EUI64:8> <type:1> <start index:1>
IEEE request:    <transaction sequence number: 1>
                  <node ID:2> <type:1> <start index:1>
                  <type> = 0x00 single address response, ignore the start index
                  = 0x01 extended response -> sends kid's IDs as well
Response: <transaction sequence number: 1>
          <status:1> <EUI64:8> <node ID:2>
          <ID count:1> <start index:1> <child ID:2>*

```

- #define [NETWORK_ADDRESS_REQUEST](#)
- #define [NETWORK_ADDRESS_RESPONSE](#)
- #define [IEEE_ADDRESS_REQUEST](#)
- #define [IEEE_ADDRESS_RESPONSE](#)

Node Descriptor Request/Response

@code

```

Request: <transaction sequence number: 1> <node ID:2>
Response: <transaction sequence number: 1> <status:1> <node ID:2>

```

// <node descriptor: 13> // // Node Descriptor field is divided into subfields of bitmasks as follows: // (Note: All lengths below are given in bits rather than bytes.) // Logical Type: 3 // Complex Descriptor Available: 1 // User Descriptor Available: 1 // (reserved/unused): 3 // APS Flags: 3 // Frequency Band: 5 // MAC capability flags: 8 // Manufacturer Code: 16 // Maximum buffer size: 8 // Maximum incoming transfer size: 16 // Server mask: 16 // Maximum outgoing transfer size: 16 // Descriptor Capability Flags: 8 // See ZigBee document 053474, Section 2.3.2.3 for more details.

- #define [NODE_DESCRIPTOR_REQUEST](#)
- #define [NODE_DESCRIPTOR_RESPONSE](#)

Power Descriptor Request / Response

@code

```
Request: <transaction sequence number: 1> <node ID:2>
Response: <transaction sequence number: 1> <status:1> <node ID:2>
         <current power mode, available power sources:1>
         <current power source, current power source level:1>
```

// See ZigBee document 053474, Section 2.3.2.4 for more details.

- #define [POWER_DESCRIPTOR_REQUEST](#)
- #define [POWER_DESCRIPTOR_RESPONSE](#)

Simple Descriptor Request / Response

```
Request: <transaction sequence number: 1>
         <node ID:2> <endpoint:1>
Response: <transaction sequence number: 1>
         <status:1> <node ID:2> <length:1> <endpoint:1>
         <app profile ID:2> <app device ID:2>
         <app device version, app flags:1>
         <input cluster count:1> <input cluster:2>*
         <output cluster count:1> <output cluster:2>*
```

- #define [SIMPLE_DESCRIPTOR_REQUEST](#)
- #define [SIMPLE_DESCRIPTOR_RESPONSE](#)

Active Endpoints Request / Response

```
Request: <transaction sequence number: 1> <node ID:2>
Response: <transaction sequence number: 1>
         <status:1> <node ID:2> <endpoint count:1> <endpoint:1>*
```

- #define [ACTIVE_ENDPOINTS_REQUEST](#)
- #define [ACTIVE_ENDPOINTS_RESPONSE](#)

Match Descriptors Request / Response

```
Request: <transaction sequence number: 1>
         <node ID:2> <app profile ID:2>
         <input cluster count:1> <input cluster:2>*
         <output cluster count:1> <output cluster:2>*
Response: <transaction sequence number: 1>
         <status:1> <node ID:2> <endpoint count:1> <endpoint:1>*
```

- #define [MATCH_DESCRIPTOR_REQUEST](#)
- #define [MATCH_DESCRIPTOR_RESPONSE](#)

Discovery Cache Request / Response

```
Request: <transaction sequence number: 1>
         <source node ID:2> <source EUI64:8>
Response: <transaction sequence number: 1>
         <status (== EMBER_ZDP_SUCCESS):1>
```

- #define [DISCOVERY_CACHE_REQUEST](#)
- #define [DISCOVERY_CACHE_RESPONSE](#)

End Device Announce and End Device Announce Response

Request: <transaction sequence number: 1>
 <node ID:2> <EUI64:8> <capabilities:1>
 No response is sent.

- #define [END_DEVICE_ANNOUNCE](#)
- #define [END_DEVICE_ANNOUNCE_RESPONSE](#)

System Server Discovery Request / Response

This is broadcast and only servers which have matching services respond. The response contains the request services that the recipient provides.

Request: <transaction sequence number: 1> <server mask:2>
 Response: <transaction sequence number: 1>
 <status (== [EMBER_ZDP_SUCCESS](#)):1> <server mask:2>

- #define [SYSTEM_SERVER_DISCOVERY_REQUEST](#)
- #define [SYSTEM_SERVER_DISCOVERY_RESPONSE](#)

Parent Announce and Parent Announce Response

This is broadcast and only servers which have matching children respond. The response contains the list of children that the recipient now holds.

Request: <transaction sequence number: 1>
 <number of children:1> <child EUI64:8> <child Age:4>*
 Response: <transaction sequence number: 1>
 <number of children:1> <child EUI64:8> <child Age:4>*

- #define [PARENT_ANNOUNCE](#)
- #define [PARENT_ANNOUNCE_RESPONSE](#)

ZDO server mask bits

These are used in server discovery requests and responses.

- enum [EmberZdoServerMask](#) {
 [EMBER_ZDP_PRIMARY_TRUST_CENTER](#), [EMBER_ZDP_SECONDARY_TRUST_CENTER](#), [EMBER_ZDP_PRIMARY_BINDING_TABLE_CACHE](#), [EMBER_ZDP_SECONDARY_BINDING_TABLE_CACHE](#),
 [EMBER_ZDP_PRIMARY_DISCOVERY_CACHE](#), [EMBER_ZDP_SECONDARY_DISCOVERY_CACHE](#), [EMBER_ZDP_NETWORK_MANAGER](#) }

Find Node Cache Request / Response

This is broadcast and only discovery servers which have the information for the device of interest, or the device of interest itself, respond. The requesting device can then direct any service discovery requests to the responder.


```
Request: <transaction sequence number: 1>
         <device of interest ID:2> <d-of-i EUI64:8>
Response: <transaction sequence number: 1>
         <responder ID:2> <device of interest ID:2> <d-of-i EUI64:8>
```

- #define [FIND_NODE_CACHE_REQUEST](#)
- #define [FIND_NODE_CACHE_RESPONSE](#)

End Device Bind Request / Response

```
Request: <transaction sequence number: 1>
         <node ID:2> <EUI64:8> <endpoint:1> <app profile ID:2>
         <input cluster count:1> <input cluster:2>*
         <output cluster count:1> <output cluster:2>*
Response: <transaction sequence number: 1> <status:1>
```

- #define [END_DEVICE_BIND_REQUEST](#)
- #define [END_DEVICE_BIND_RESPONSE](#)

Binding types and Request / Response

Bind and unbind have the same formats. There are two possible formats, depending on whether the destination is a group address or a device address. Device addresses include an endpoint, groups don't.

```
Request: <transaction sequence number: 1>
         <source EUI64:8> <source endpoint:1>
         <cluster ID:2> <destination address:3 or 10>
Destination address:
         <0x01:1> <destination group:2>
Or:
         <0x03:1> <destination EUI64:8> <destination endpoint:1>
Response: <transaction sequence number: 1> <status:1>
```

- #define [UNICAST_BINDING](#)
- #define [UNICAST_MANY_TO_ONE_BINDING](#)
- #define [MULTICAST_BINDING](#)
- #define [BIND_REQUEST](#)
- #define [BIND_RESPONSE](#)
- #define [UNBIND_REQUEST](#)
- #define [UNBIND_RESPONSE](#)

LQI Table Request / Response

```
Request: <transaction sequence number: 1> <start index:1>
Response: <transaction sequence number: 1> <status:1>
         <neighbor table entries:1> <start index:1>
         <entry count:1> <entry:22>*
         <entry> = <extended PAN ID:8> <EUI64:8> <node ID:2>
                 <device type, rx on when idle, relationship:1>
                 <permit joining:1> <depth:1> <LQI:1>
```

The device-type byte has the following fields:

Name	Mask	Values
device type	0x03	0x00 coordinator 0x01 router

		0x02 end device
		0x03 unknown
rx mode	0x0C	0x00 off when idle 0x04 on when idle 0x08 unknown
relationship	0x70	0x00 parent 0x10 child 0x20 sibling 0x30 other 0x40 previous child
reserved	0x10	

The permit-joining byte has the following fields

Name	Mask	Values
permit joining	0x03	0x00 not accepting join requests 0x01 accepting join requests 0x02 unknown
reserved	0xFC	

- `#define LQI_TABLE_REQUEST`
- `#define LQI_TABLE_RESPONSE`

Routing Table Request / Response

```
Request: <transaction sequence number: 1> <start index:1>
Response: <transaction sequence number: 1> <status:1>
          <routing table entries:1> <start index:1>
          <entry count:1> <entry:5>*
          <entry> = <destination address:2>
                   <status:1>
                   <next hop:2>
```

The status byte has the following fields:

Name	Mask	Values
status	0x07	0x00 active 0x01 discovery underway 0x02 discovery failed 0x03 inactive 0x04 validation underway
flags	0x38	0x08 memory constrained 0x10 many-to-one 0x20 route record required
reserved	0xC0	

- `#define ROUTING_TABLE_REQUEST`
- `#define ROUTING_TABLE_RESPONSE`

Binding Table Request / Response

```
Request: <transaction sequence number: 1> <start index:1>
Response: <transaction sequence number: 1>
          <status:1> <binding table entries:1> <start index:1>
          <entry count:1> <entry:14/21>*
          <entry> = <source EUI64:8> <source endpoint:1> <cluster ID:2>
                   <dest addr mode:1> <dest:2/8> <dest endpoint:0/1>
```

Note

If Dest. Address Mode = 0x03, then the Long Dest. Address will be used and Dest. endpoint will be included. If Dest. Address Mode = 0x01, then the Short Dest. Address will be used and there will be no Dest. endpoint.

- #define [BINDING_TABLE_REQUEST](#)
- #define [BINDING_TABLE_RESPONSE](#)

Leave Request / Response

Request: <transaction sequence number: 1> <EUI64:8> <flags:1>
 The flag bits are:
 0x40 [remove](#) children
 0x80 [rejoin](#)
 Response: <transaction sequence number: 1> <status:1>

- #define [LEAVE_REQUEST](#)
- #define [LEAVE_RESPONSE](#)
- #define [LEAVE_REQUEST_REMOVE_CHILDREN_FLAG](#)
- #define [LEAVE_REQUEST_REJOIN_FLAG](#)

Permit Joining Request / Response

Request: <transaction sequence number: 1>
 <duration:1> <permit authentication:1>
 Response: <transaction sequence number: 1> <status:1>

- #define [PERMIT_JOINING_REQUEST](#)
- #define [PERMIT_JOINING_RESPONSE](#)

Network Update Request / Response

Request: <transaction sequence number: 1>
 <scan channels:4> <duration:1> <count:0/1> <manager:0/2>

If the duration is in 0x00 ... 0x05, then 'count' is present but not 'manager'. Perform 'count' scans of the given duration on the given channels.

If duration is 0xFE, then 'channels' should have a single channel and 'count' and 'manager' are not present. Switch to the indicated channel.

If duration is 0xFF, then 'count' is not present. Set the active channels and the network manager ID to the values given.

Unicast requests always [get](#) a response, which is [INVALID_REQUEST](#) if the duration is not a legal value.

Response: <transaction sequence number: 1> <status:1>
 <scanned channels:4> <transmissions:2> <failures:2>
 <energy count:1> <energy:1>*

- #define [NWK_UPDATE_REQUEST](#)
- #define [NWK_UPDATE_RESPONSE](#)

Unsupported

Not mandatory and not supported.

- `#define COMPLEX_DESCRIPTOR_REQUEST`
- `#define COMPLEX_DESCRIPTOR_RESPONSE`
- `#define USER_DESCRIPTOR_REQUEST`
- `#define USER_DESCRIPTOR_RESPONSE`
- `#define DISCOVERY_REGISTER_REQUEST`
- `#define DISCOVERY_REGISTER_RESPONSE`
- `#define USER_DESCRIPTOR_SET`
- `#define USER_DESCRIPTOR_CONFIRM`
- `#define NETWORK_DISCOVERY_REQUEST`
- `#define NETWORK_DISCOVERY_RESPONSE`
- `#define DIRECT_JOIN_REQUEST`
- `#define DIRECT_JOIN_RESPONSE`
- `#define CLUSTER_ID_RESPONSE_MINIMUM`

ZDO configuration flags.

For controlling which ZDO requests are passed to the application. These are normally controlled via the following configuration definitions:

`EMBER_APPLICATION_RECEIVES_SUPPORTED_ZDO_REQUESTS` `EMBER_APPLICATION_HANDLES_UNSUPPORTED_ZDO_REQUESTS` `EMBER_APPLICATION_HANDLES_ENDPOINT_ZDO_REQUESTS` `EMBER_APPLICATION_HANDLES_BINDING_ZDO_REQUESTS`

See `ember-configuration.h` for more information.

- `enum EmberZdoConfigurationFlags { EMBER_APP_RECEIVES_SUPPORTED_ZDO_REQUESTS, EMBER_APP_HANDLES_UNSUPPORTED_ZDO_REQUESTS, EMBER_APP_HANDLES_ZDO_ENDPOINT_REQUESTS, EMBER_APP_HANDLES_ZDO_BINDING_REQUESTS }`

6.2.1 Detailed Description

See [ember-types.h](#) for source code.

6.2.2 Macro Definition Documentation

6.2.2.1 `#define EMBER_RELEASE_TYPE_TO_STRING_STRUCT_DATA`

[EmberReleaseTypeStruct](#) Data that relates release type to the correct string.

Definition at line 75 of file [ember-types.h](#).

6.2.2.2 `#define EUI64_SIZE`

Size of EUI64 (an IEEE address) in bytes (8).

Definition at line 107 of file [ember-types.h](#).

6.2.2.3 #define EXTENDED_PAN_ID_SIZE

Size of an extended PAN identifier in bytes (8).

Definition at line 112 of file [ember-types.h](#).

6.2.2.4 #define EMBER_ENCRYPTION_KEY_SIZE

Size of an encryption key in bytes (16).

Definition at line 117 of file [ember-types.h](#).

6.2.2.5 #define EMBER_CERTIFICATE_SIZE

Size of Implicit Certificates used for Certificate Based Key Exchange.

Definition at line 123 of file [ember-types.h](#).

6.2.2.6 #define EMBER_PUBLIC_KEY_SIZE

Size of Public Keys used in Elliptical Cryptography ECMQV algorithms.

Definition at line 128 of file [ember-types.h](#).

6.2.2.7 #define EMBER_PRIVATE_KEY_SIZE

Size of Private Keys used in Elliptical Cryptography ECMQV algorithms.

Definition at line 133 of file [ember-types.h](#).

6.2.2.8 #define EMBER_SMAC_SIZE

Size of the SMAC used in Elliptical Cryptography ECMQV algorithms.

Definition at line 138 of file [ember-types.h](#).

6.2.2.9 #define EMBER_SIGNATURE_SIZE

Size of the DSA signature used in Elliptical Cryptography Digital Signature Algorithms.

Definition at line 144 of file [ember-types.h](#).

6.2.2.10 #define EMBER_AES_HASH_BLOCK_SIZE

The size of AES-128 MMO hash is 16-bytes. This is defined in the core. ZigBee specification.

Definition at line 149 of file [ember-types.h](#).

6.2.2.11 #define EMBER_CERTIFICATE_283K1_SIZE

Size of Implicit Certificates used for Certificate Based Key Exchange using the ECC283K1 curve in bytes.

Definition at line 155 of file [ember-types.h](#).

6.2.2.12 #define EMBER_PUBLIC_KEY_283K1_SIZE

Size of Public Keys used in SECT283k1 Elliptical Cryptography ECMQV algorithms.

Definition at line 160 of file [ember-types.h](#).

6.2.2.13 #define EMBER_PRIVATE_KEY_283K1_SIZE

Size of Private Keys used SECT283k1 in Elliptical Cryptography ECMQV algorithms.

Definition at line 165 of file [ember-types.h](#).

6.2.2.14 #define EMBER_SIGNATURE_283K1_SIZE

Size of the DSA signature used in SECT283k1 Elliptical Cryptography Digital Signature Algorithms.

Definition at line 171 of file [ember-types.h](#).

6.2.2.15 #define __EMBERSTATUS_TYPE__

Return type for Ember functions.

Definition at line 177 of file [ember-types.h](#).

6.2.2.16 #define EMBER_MAX_802_15_4_CHANNEL_NUMBER

The maximum 802.15.4 channel number is 26.

Definition at line 215 of file [ember-types.h](#).

6.2.2.17 #define EMBER_MIN_802_15_4_CHANNEL_NUMBER

The minimum 802.15.4 channel number is 11.

Definition at line 220 of file [ember-types.h](#).

6.2.2.18 #define EMBER_NUM_802_15_4_CHANNELS

There are sixteen 802.15.4 channels.

Definition at line 225 of file [ember-types.h](#).

6.2.2.19 #define EMBER_ALL_802_15_4_CHANNELS_MASK

Bitmask to scan all 802.15.4 channels.

Definition at line 231 of file [ember-types.h](#).

6.2.2.20 #define EMBER_ZIGBEE_COORDINATOR_ADDRESS

The network ID of the coordinator in a ZigBee network is 0x0000.

Definition at line 236 of file [ember-types.h](#).

6.2.2.21 #define EMBER_NULL_NODE_ID

A distinguished network ID that will never be assigned to any node. Used to indicate the absence of a node ID.

Definition at line 242 of file [ember-types.h](#).

6.2.2.22 #define EMBER_NULL_BINDING

A distinguished binding index used to indicate the absence of a binding.

Definition at line 248 of file [ember-types.h](#).

6.2.2.23 #define EMBER_TABLE_ENTRY_UNUSED_NODE_ID

A distinguished network ID that will never be assigned to any node.

This value is used when setting or getting the remote node ID in the address table or getting the remote node ID from the binding table. It indicates that address or binding table entry is not in use.

Definition at line 259 of file [ember-types.h](#).

6.2.2.24 #define EMBER_MULTICAST_NODE_ID

A distinguished network ID that will never be assigned to any node. This value is returned when getting the remote node ID from the binding table and the given binding table index refers to a multicast binding entry.

Definition at line 267 of file [ember-types.h](#).

6.2.2.25 #define EMBER_UNKNOWN_NODE_ID

A distinguished network ID that will never be assigned to any node. This value is used when getting the remote node ID from the address or binding tables. It indicates that the address or binding table entry is currently in use but the node ID corresponding to the EUI64 in the table is currently unknown.

Definition at line 276 of file [ember-types.h](#).

6.2.2.26 #define EMBER_DISCOVERY_ACTIVE_NODE_ID

A distinguished network ID that will never be assigned to any node. This value is used when getting the remote node ID from the address or binding tables. It indicates that the address or binding table entry is currently in use and network address discovery is underway.

Definition at line 285 of file [ember-types.h](#).

6.2.2.27 #define EMBER_NULL_ADDRESS_TABLE_INDEX

A distinguished address table index used to indicate the absence of an address table entry.

Definition at line 291 of file [ember-types.h](#).

6.2.2.28 #define EMBER_ZDO_ENDPOINT

The endpoint where the ZigBee Device Object (ZDO) resides.

Definition at line 296 of file [ember-types.h](#).

6.2.2.29 #define EMBER_BROADCAST_ENDPOINT

The broadcast endpoint, as defined in the ZigBee spec.

Definition at line 301 of file [ember-types.h](#).

6.2.2.30 #define EMBER_ZDO_PROFILE_ID

The profile ID used by the ZigBee Device Object (ZDO).

Definition at line 306 of file [ember-types.h](#).

6.2.2.31 #define EMBER_WILDCARD_PROFILE_ID

The profile ID used to address all the public profiles.

Definition at line 311 of file [ember-types.h](#).

6.2.2.32 #define EMBER_MAXIMUM_STANDARD_PROFILE_ID

The maximum value for a profile ID in the standard profile range.

Definition at line 316 of file [ember-types.h](#).

6.2.2.33 #define EMBER_BROADCAST_TABLE_TIMEOUT_QS

The broadcast table timeout. How long a broadcast entry persists in the local device's broadcast table. This is the maximum length it will persist, in quarter seconds.

Definition at line 323 of file [ember-types.h](#).

6.2.2.34 #define EMBER_MANUFACTURER_ID

Ember's Manufacturer ID.

Definition at line 329 of file [ember-types.h](#).

6.2.2.35 #define EMBER_BROADCAST_ADDRESS

Broadcast to all routers.

Definition at line 378 of file [ember-types.h](#).

6.2.2.36 #define EMBER_RX_ON_WHEN_IDLE_BROADCAST_ADDRESS

Broadcast to all non-sleepy devices.

Definition at line 380 of file [ember-types.h](#).

6.2.2.37 #define EMBER_SLEEPY_BROADCAST_ADDRESS

Broadcast to all devices, including sleepy end devices.

Definition at line 382 of file [ember-types.h](#).

6.2.2.38 #define EMBER_MIN_BROADCAST_ADDRESS

Definition at line 387 of file [ember-types.h](#).

6.2.2.39 #define emberIsZigbeeBroadcastAddress(address)

Definition at line 389 of file [ember-types.h](#).

6.2.2.40 #define EMBER_LOW_RAM_CONCENTRATOR

A concentrator with insufficient memory to store source routes for the entire network. Route records are sent to the concentrator prior to every inbound APS unicast.

Definition at line 711 of file [ember-types.h](#).

6.2.2.41 #define EMBER_HIGH_RAM_CONCENTRATOR

A concentrator with sufficient memory to store source routes for the entire network. Remote nodes stop sending route records once the concentrator has successfully received one.

Definition at line 716 of file [ember-types.h](#).

6.2.2.42 #define EMBER_JOIN_DECISION_STRINGS

@ brief Defines the CLI enumerations for the [EmberJoinDecision](#) enum

Definition at line 744 of file [ember-types.h](#).

6.2.2.43 **#define EMBER_DEVICE_UPDATE_STRINGS**

@ brief Defines the CLI enumerations for the [EmberDeviceUpdate](#) enum.

Definition at line 779 of file [ember-types.h](#).

6.2.2.44 **#define emberInitializeNetworkParameters(*parameters*)**

Definition at line 951 of file [ember-types.h](#).

6.2.2.45 **#define EMBER_COUNTER_STRINGS**

@ brief Defines the CLI enumerations for the [EmberCounterType](#) enum.

Definition at line 1226 of file [ember-types.h](#).

6.2.2.46 **#define EMBER_TX_POWER_MODE_DEFAULT**

The application should call `::emberSetTxPowerMode()` with the `txPowerMode` parameter set to this value to disable all power mode options, resulting in normal power mode and bi-directional RF transmitter output.

Definition at line 1319 of file [ember-types.h](#).

6.2.2.47 **#define EMBER_TX_POWER_MODE_BOOST**

The application should call `::emberSetTxPowerMode()` with the `txPowerMode` parameter set to this value to enable boost power mode.

Definition at line 1323 of file [ember-types.h](#).

6.2.2.48 **#define EMBER_TX_POWER_MODE_ALTERNATE**

The application should call `::emberSetTxPowerMode()` with the `txPowerMode` parameter set to this value to enable the alternate transmitter output.

Definition at line 1328 of file [ember-types.h](#).

6.2.2.49 **#define EMBER_TX_POWER_MODE_BOOST_AND_ALTERNATE**

The application should call `::emberSetTxPowerMode()` with the `txPowerMode` parameter set to this value to enable both boost mode and the alternate transmitter output.

Definition at line 1333 of file [ember-types.h](#).

6.2.2.50 **#define EMBER_PRIVATE_PROFILE_ID**

This is a ZigBee application profile ID that has been assigned to Ember Corporation.

It is used to send for sending messages that have a specific, non-standard interaction with the Ember stack. Its only current use is for alarm messages and stack counters requests.

Definition at line 1357 of file [ember-types.h](#).

6.2.2.51 `#define EMBER_PRIVATE_PROFILE_ID_START`

Ember's first private profile ID.

Definition at line 1362 of file [ember-types.h](#).

6.2.2.52 `#define EMBER_PRIVATE_PROFILE_ID_END`

Ember's last private profile ID.

Definition at line 1367 of file [ember-types.h](#).

6.2.2.53 `#define EMBER_BROADCAST_ALARM_CLUSTER`

Alarm messages provide a reliable means for communicating with sleeping end devices.

A messages sent to a sleeping device is normally buffered on the device's parent for a short time (the precise time can be specified using the configuration parameter [EMBER_INDIRECT_TRANSMISSION_TIMEOUT](#)). If the child does not poll its parent within that time the message is discarded.

In contrast, alarm messages are buffered by the parent indefinitely. Because of the limited RAM available, alarm messages are necessarily brief. In particular, the parent only stores alarm payloads. The header information in alarm messages is not stored on the parent.

The memory used for buffering alarm messages is allocated statically. The amount of memory set aside for alarms is controlled by two configuration parameters:

- [EMBER_BROADCAST_ALARM_DATA_SIZE](#)
- [EMBER_UNICAST_ALARM_DATA_SIZE](#)

Alarm messages must use the [EMBER_PRIVATE_PROFILE_ID](#) as the application profile ID. The source and destination endpoints are ignored.

Broadcast alarms must use [EMBER_BROADCAST_ALARM_CLUSTER](#) as the cluster id and messages with this cluster ID must be sent to [EMBER_RX_ON_WHEN_IDLE_BROADCAST_ADDRESS](#). A broadcast alarm may not contain more than [EMBER_BROADCAST_ALARM_DATA_SIZE](#) bytes of payload.

Broadcast alarm messages arriving at a node are passed to the application via `::ember-IncomingMessageHandler()`. If the receiving node has sleepy end device children, the payload of the alarm is saved and then forwarded to those children when they poll for data. When a sleepy child polls its parent, it receives only the most recently arrived broadcast alarm. If the child has already received the most recent broadcast alarm it is not forwarded again.

Definition at line 1407 of file [ember-types.h](#).

6.2.2.54 `#define EMBER_UNICAST_ALARM_CLUSTER`

Unicast alarms must use [EMBER_UNICAST_ALARM_CLUSTER](#) as the cluster id and messages with this cluster ID must be unicast.

The payload of a unicast alarm consists of three one-byte length fields followed by three variable length fields.

1. flags length
2. priority length (must be 0 or 1)
3. data length
4. flags
5. priority
6. payload

The three lengths must total [EMBER_UNICAST_ALARM_DATA_SIZE](#) or less.

When a unicast alarm message arrives at its destination it is passed to the application via `::emberIncomingMessageHandler()`. When a node receives a unicast alarm message whose destination is a sleepy end device child of that node, the payload of the message is saved until the child polls for data. To conserve memory, the values of the length fields are not saved. The alarm will be forwarded to the child using the [EMBER_CACHED_UNICAST_ALARM_CLUSTER](#) cluster ID.

If a unicast alarm arrives when a previous one is still pending, the two payloads are combined. This combining is controlled by the length fields in the arriving message. The incoming flag bytes are or'ed with those of the pending message. If the priority field is not present, or if it is present and the incoming priority value is equal or greater than the pending priority value, the pending data is replaced by the incoming data.

Because the length fields are not saved, the application designer must fix on a set of field lengths that will be used for all unicast alarm message sent to a particular device.

Definition at line [1445](#) of file [ember-types.h](#).

6.2.2.55 `#define EMBER_CACHED_UNICAST_ALARM_CLUSTER`

A unicast alarm that has been cached on the parent of a sleepy end device is delivered to that device using the [EMBER_CACHED_UNICAST_ALARM_CLUSTER](#) cluster ID. The payload consists of three variable length fields.

1. flags
2. priority
3. payload

The parent will pad the payload out to [EMBER_UNICAST_ALARM_DATA_SIZE](#) bytes.

The lengths of the these fields must be fixed by the application designer and must be the same for all unicast alarms sent to a particular device.

Definition at line [1462](#) of file [ember-types.h](#).

6.2.2.56 `#define EMBER_REPORT_COUNTERS_REQUEST`

The cluster id used to request that a node respond with a report of its Ember stack counters. See `app/util/counters/counters-ota.h`.

Definition at line [1467](#) of file [ember-types.h](#).

6.2.2.57 #define EMBER_REPORT_COUNTERS_RESPONSE

The cluster id used to respond to an EMBER_REPORT_COUNTERS_REQUEST.

Definition at line 1470 of file [ember-types.h](#).

6.2.2.58 #define EMBER_REPORT_AND_CLEAR_COUNTERS_REQUEST

The cluster id used to request that a node respond with a report of its Ember stack counters. The node will also reset its clusters to zero after a successful response. See [app/util/counters/counters-ota.h](#).

Definition at line 1476 of file [ember-types.h](#).

6.2.2.59 #define EMBER_REPORT_AND_CLEAR_COUNTERS_RESPONSE

The cluster id used to respond to an EMBER_REPORT_AND_CLEAR_COUNTERS_REQUEST.

Definition at line 1479 of file [ember-types.h](#).

6.2.2.60 #define EMBER_OTA_CERTIFICATE_UPGRADE_CLUSTER

The cluster id used to send and receive Over-the-air certificate messages. This is used to field upgrade devices with Smart Energy Certificates and other security data.

Definition at line 1485 of file [ember-types.h](#).

6.2.2.61 #define EMBER_STANDARD_SECURITY_MODE

This is an [EmberInitialSecurityBitmask](#) value but it does not actually set anything. It is the default mode used by the ZigBee Pro stack. It is defined here so that no legacy code is broken by referencing it.

Definition at line 1574 of file [ember-types.h](#).

6.2.2.62 #define EMBER_TRUST_CENTER_NODE_ID

This is the short address of the trust center. It never changes from this value throughout the life of the network.

Definition at line 1579 of file [ember-types.h](#).

6.2.2.63 #define EMBER_NO_TRUST_CENTER_MODE

This is the legacy name for the Distributed Trust Center Mode.

Definition at line 1730 of file [ember-types.h](#).

6.2.2.64 #define EMBER_GLOBAL_LINK_KEY

This is the legacy name for the Trust Center Global Link Key.

Definition at line [1734](#) of file [ember-types.h](#).

6.2.2.65 #define EMBER_MFG_SECURITY_CONFIG_MAGIC_NUMBER

This magic number prevents accidentally changing the key settings. The ::emberSetMfg-SecurityConfig() API will return EMBER_INVALID_CALL unless it is passed in.

Definition at line [2138](#) of file [ember-types.h](#).

6.2.2.66 #define EMBER_MAC_FILTER_MATCH_ENABLED_MASK

Definition at line [2178](#) of file [ember-types.h](#).

6.2.2.67 #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_MASK

Definition at line [2179](#) of file [ember-types.h](#).

6.2.2.68 #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_MASK

Definition at line [2180](#) of file [ember-types.h](#).

6.2.2.69 #define EMBER_MAC_FILTER_MATCH_ON_DEST_MASK

Definition at line [2181](#) of file [ember-types.h](#).

6.2.2.70 #define EMBER_MAC_FILTER_MATCH_ON_SOURCE_MASK

Definition at line [2182](#) of file [ember-types.h](#).

6.2.2.71 #define EMBER_MAC_FILTER_MATCH_ENABLED

Definition at line [2185](#) of file [ember-types.h](#).

6.2.2.72 #define EMBER_MAC_FILTER_MATCH_DISABLED

Definition at line [2186](#) of file [ember-types.h](#).

6.2.2.73 #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_NONE

Definition at line [2189](#) of file [ember-types.h](#).

6.2.2.74 #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_LOCAL

Definition at line [2190](#) of file [ember-types.h](#).

6.2.2.75 #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_BROADCAST

Definition at line 2191 of file [ember-types.h](#).

6.2.2.76 #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_NONE

Definition at line 2194 of file [ember-types.h](#).

6.2.2.77 #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_NON_LOCAL

Definition at line 2195 of file [ember-types.h](#).

6.2.2.78 #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_LOCAL

Definition at line 2196 of file [ember-types.h](#).

6.2.2.79 #define EMBER_MAC_FILTER_MATCH_ON_DEST_BROADCAST_SHORT

Definition at line 2199 of file [ember-types.h](#).

6.2.2.80 #define EMBER_MAC_FILTER_MATCH_ON_DEST_UNICAST_SHORT

Definition at line 2200 of file [ember-types.h](#).

6.2.2.81 #define EMBER_MAC_FILTER_MATCH_ON_DEST_UNICAST_LONG

Definition at line 2201 of file [ember-types.h](#).

6.2.2.82 #define EMBER_MAC_FILTER_MATCH_ON_SOURCE_LONG

Definition at line 2204 of file [ember-types.h](#).

6.2.2.83 #define EMBER_MAC_FILTER_MATCH_ON_SOURCE_SHORT

Definition at line 2205 of file [ember-types.h](#).

6.2.2.84 #define EMBER_MAC_FILTER_MATCH_ON_SOURCE_NONE

Definition at line 2206 of file [ember-types.h](#).

6.2.2.85 #define EMBER_MAC_FILTER_MATCH_END

Definition at line 2209 of file [ember-types.h](#).

6.2.2.86 #define NETWORK_ADDRESS_REQUEST

Definition at line [2293](#) of file [ember-types.h](#).

6.2.2.87 #define NETWORK_ADDRESS_RESPONSE

Definition at line [2294](#) of file [ember-types.h](#).

6.2.2.88 #define IEEE_ADDRESS_REQUEST

Definition at line [2295](#) of file [ember-types.h](#).

6.2.2.89 #define IEEE_ADDRESS_RESPONSE

Definition at line [2296](#) of file [ember-types.h](#).

6.2.2.90 #define NODE_DESCRIPTOR_REQUEST

Definition at line [2324](#) of file [ember-types.h](#).

6.2.2.91 #define NODE_DESCRIPTOR_RESPONSE

Definition at line [2325](#) of file [ember-types.h](#).

6.2.2.92 #define POWER_DESCRIPTOR_REQUEST

Definition at line [2338](#) of file [ember-types.h](#).

6.2.2.93 #define POWER_DESCRIPTOR_RESPONSE

Definition at line [2339](#) of file [ember-types.h](#).

6.2.2.94 #define SIMPLE_DESCRIPTOR_REQUEST

Definition at line [2355](#) of file [ember-types.h](#).

6.2.2.95 #define SIMPLE_DESCRIPTOR_RESPONSE

Definition at line [2356](#) of file [ember-types.h](#).

6.2.2.96 #define ACTIVE_ENDPOINTS_REQUEST

Definition at line [2367](#) of file [ember-types.h](#).

6.2.2.97 #define ACTIVE_ENDPOINTS_RESPONSE

Definition at line 2368 of file [ember-types.h](#).

6.2.2.98 #define MATCH_DESCRIPTOR_REQUEST

Definition at line 2382 of file [ember-types.h](#).

6.2.2.99 #define MATCH_DESCRIPTOR_RESPONSE

Definition at line 2383 of file [ember-types.h](#).

6.2.2.100 #define DISCOVERY_CACHE_REQUEST

Definition at line 2395 of file [ember-types.h](#).

6.2.2.101 #define DISCOVERY_CACHE_RESPONSE

Definition at line 2396 of file [ember-types.h](#).

6.2.2.102 #define END_DEVICE_ANNOUNCE

Definition at line 2407 of file [ember-types.h](#).

6.2.2.103 #define END_DEVICE_ANNOUNCE_RESPONSE

Definition at line 2408 of file [ember-types.h](#).

6.2.2.104 #define SYSTEM_SERVER_DISCOVERY_REQUEST

Definition at line 2422 of file [ember-types.h](#).

6.2.2.105 #define SYSTEM_SERVER_DISCOVERY_RESPONSE

Definition at line 2423 of file [ember-types.h](#).

6.2.2.106 #define PARENT_ANNOUNCE

Definition at line 2438 of file [ember-types.h](#).

6.2.2.107 #define PARENT_ANNOUNCE_RESPONSE

Definition at line 2439 of file [ember-types.h](#).

6.2.2.108 #define FIND_NODE_CACHE_REQUEST

Definition at line [2476](#) of file [ember-types.h](#).

6.2.2.109 #define FIND_NODE_CACHE_RESPONSE

Definition at line [2477](#) of file [ember-types.h](#).

6.2.2.110 #define END_DEVICE_BIND_REQUEST

Definition at line [2490](#) of file [ember-types.h](#).

6.2.2.111 #define END_DEVICE_BIND_RESPONSE

Definition at line [2491](#) of file [ember-types.h](#).

6.2.2.112 #define UNICAST_BINDING

Definition at line [2511](#) of file [ember-types.h](#).

6.2.2.113 #define UNICAST_MANY_TO_ONE_BINDING

Definition at line [2512](#) of file [ember-types.h](#).

6.2.2.114 #define MULTICAST_BINDING

Definition at line [2513](#) of file [ember-types.h](#).

6.2.2.115 #define BIND_REQUEST

Definition at line [2515](#) of file [ember-types.h](#).

6.2.2.116 #define BIND_RESPONSE

Definition at line [2516](#) of file [ember-types.h](#).

6.2.2.117 #define UNBIND_REQUEST

Definition at line [2517](#) of file [ember-types.h](#).

6.2.2.118 #define UNBIND_RESPONSE

Definition at line [2518](#) of file [ember-types.h](#).

6.2.2.119 #define LQI_TABLE_REQUEST

Definition at line [2568](#) of file [ember-types.h](#).

6.2.2.120 #define LQI_TABLE_RESPONSE

Definition at line [2569](#) of file [ember-types.h](#).

6.2.2.121 #define ROUTING_TABLE_REQUEST

Definition at line [2604](#) of file [ember-types.h](#).

6.2.2.122 #define ROUTING_TABLE_RESPONSE

Definition at line [2605](#) of file [ember-types.h](#).

6.2.2.123 #define BINDING_TABLE_REQUEST

Definition at line [2626](#) of file [ember-types.h](#).

6.2.2.124 #define BINDING_TABLE_RESPONSE

Definition at line [2627](#) of file [ember-types.h](#).

6.2.2.125 #define LEAVE_REQUEST

Definition at line [2640](#) of file [ember-types.h](#).

6.2.2.126 #define LEAVE_RESPONSE

Definition at line [2641](#) of file [ember-types.h](#).

6.2.2.127 #define LEAVE_REQUEST_REMOVE_CHILDREN_FLAG

Definition at line [2643](#) of file [ember-types.h](#).

6.2.2.128 #define LEAVE_REQUEST_REJOIN_FLAG

Definition at line [2644](#) of file [ember-types.h](#).

6.2.2.129 #define PERMIT_JOINING_REQUEST

Definition at line [2655](#) of file [ember-types.h](#).

6.2.2.130 #define PERMIT_JOINING_RESPONSE

Definition at line [2656](#) of file [ember-types.h](#).

6.2.2.131 #define NWK_UPDATE_REQUEST

Definition at line [2684](#) of file [ember-types.h](#).

6.2.2.132 #define NWK_UPDATE_RESPONSE

Definition at line [2685](#) of file [ember-types.h](#).

6.2.2.133 #define COMPLEX_DESCRIPTOR_REQUEST

Definition at line [2691](#) of file [ember-types.h](#).

6.2.2.134 #define COMPLEX_DESCRIPTOR_RESPONSE

Definition at line [2692](#) of file [ember-types.h](#).

6.2.2.135 #define USER_DESCRIPTOR_REQUEST

Definition at line [2693](#) of file [ember-types.h](#).

6.2.2.136 #define USER_DESCRIPTOR_RESPONSE

Definition at line [2694](#) of file [ember-types.h](#).

6.2.2.137 #define DISCOVERY_REGISTER_REQUEST

Definition at line [2695](#) of file [ember-types.h](#).

6.2.2.138 #define DISCOVERY_REGISTER_RESPONSE

Definition at line [2696](#) of file [ember-types.h](#).

6.2.2.139 #define USER_DESCRIPTOR_SET

Definition at line [2697](#) of file [ember-types.h](#).

6.2.2.140 #define USER_DESCRIPTOR_CONFIRM

Definition at line [2698](#) of file [ember-types.h](#).

6.2.2.141 #define NETWORK_DISCOVERY_REQUEST

Definition at line 2699 of file [ember-types.h](#).

6.2.2.142 #define NETWORK_DISCOVERY_RESPONSE

Definition at line 2700 of file [ember-types.h](#).

6.2.2.143 #define DIRECT_JOIN_REQUEST

Definition at line 2701 of file [ember-types.h](#).

6.2.2.144 #define DIRECT_JOIN_RESPONSE

Definition at line 2702 of file [ember-types.h](#).

6.2.2.145 #define CLUSTER_ID_RESPONSE_MINIMUM

Definition at line 2705 of file [ember-types.h](#).

6.2.2.146 #define WEAK_TEST

Definition at line 2739 of file [ember-types.h](#).

6.2.3 Typedef Documentation**6.2.3.1 typedef uint8_t EmberStatus**

[EmberReleaseTypeStruct](#) Data that relates release type to the correct string.

Definition at line 178 of file [ember-types.h](#).

6.2.3.2 typedef uint8_t EmberEUI64[EUI64_SIZE]

EUI 64-bit ID (an IEEE address).

Definition at line 186 of file [ember-types.h](#).

6.2.3.3 typedef uint8_t EmberMessageBuffer

Incoming and outgoing messages are stored in buffers. These buffers are allocated and freed as needed.

Buffers are 32 bytes in length and can be linked together to hold longer messages.

See packet-buffer.h for APIs related to stack and linked buffers.

Definition at line 197 of file [ember-types.h](#).

6.2.3.4 `typedef uint16_t EmberNodeId`

16-bit ZigBee network address.

Definition at line 202 of file [ember-types.h](#).

6.2.3.5 `typedef uint16_t EmberMulticastId`

16-bit ZigBee multicast group identifier.

Definition at line 205 of file [ember-types.h](#).

6.2.3.6 `typedef uint16_t EmberPanId`

802.15.4 PAN ID.

Definition at line 210 of file [ember-types.h](#).

6.2.3.7 `typedef uint8_t EmberTaskId`

brief An identifier for a task

Definition at line 1264 of file [ember-types.h](#).

6.2.3.8 `typedef PGM_struct EmberEventData_S EmberEventData`

6.2.3.9 `typedef uint16_t EmberMacFilterMatchData`

This is a bitmask describing a filter for MAC data messages that the stack should accept and passthrough to the application.

Definition at line 2176 of file [ember-types.h](#).

6.2.3.10 `typedef uint8_t EmberLibraryStatus`

This indicates the presence, absence, or status of an Ember stack library.

Definition at line 2224 of file [ember-types.h](#).

6.2.4 Enumeration Type Documentation

6.2.4.1 `enum EmberVersionType`

Type of Ember software version.

Enumerator:

EMBER_VERSION_TYPE_PRE_RELEASE

EMBER_VERSION_TYPE_ALPHA_1

EMBER_VERSION_TYPE_ALPHA_2

EMBER_VERSION_TYPE_ALPHA_3

EMBER_VERSION_TYPE_BETA_1
EMBER_VERSION_TYPE_BETA_2
EMBER_VERSION_TYPE_BETA_3
EMBER_VERSION_TYPE_GA

Definition at line 37 of file [ember-types.h](#).

6.2.4.2 enum EmberLeaveRequestFlags

[EmberReleaseTypeStruct](#) Data that relates release type to the correct string.

Enumerator:

EMBER_ZIGBEE_LEAVE_AND_REJOIN Leave and rejoin
EMBER_ZIGBEE_LEAVE_AND_REMOVE_CHILDREN Send all children leave command

Definition at line 333 of file [ember-types.h](#).

6.2.4.3 enum EmberLeaveReason

[EmberReleaseTypeStruct](#) Data that relates release type to the correct string.

Enumerator:

EMBER_LEAVE_REASON_NONE
EMBER_LEAVE_DUE_TO_NWK_LEAVE_MESSAGE
EMBER_LEAVE_DUE_TO_APS_REMOVE_MESSAGE
EMBER_LEAVE_DUE_TO_ZDO_LEAVE_MESSAGE
EMBER_LEAVE_DUE_TO_ZLL_TOUCHLINK
EMBER_LEAVE_DUE_TO_APP_EVENT_1

Definition at line 347 of file [ember-types.h](#).

6.2.4.4 enum EmberNodeType

Defines the possible types of nodes and the roles that a node might play in a network.

Enumerator:

EMBER_UNKNOWN_DEVICE Device is not joined
EMBER_COORDINATOR Will relay messages and can act as a parent to other nodes.
EMBER_ROUTER Will relay messages and can act as a parent to other nodes.
EMBER_END_DEVICE Communicates only with its parent and will not relay messages.
EMBER_SLEEPY_END_DEVICE An end device whose radio can be turned off to save power. The application must call `::emberPollForData()` to receive messages.

EMBER_MOBILE_END_DEVICE A sleepy end device that can move through the network.

EMBER_RF4CE_TARGET RF4CE target node.

EMBER_RF4CE_CONTROLLER RF4CE controller node.

Definition at line 398 of file [ember-types.h](#).

6.2.4.5 enum EmberEndDeviceConfiguration

The configuration advertised by the end device to the parent when joining/rejoining.

Enumerator:

EMBER_END_DEVICE_CONFIG_NONE

EMBER_END_DEVICE_CONFIG_PERSIST_DATA_ON_PARENT

Definition at line 428 of file [ember-types.h](#).

6.2.4.6 enum EmberNetworkInitBitmask

Defines the options that should be used when initializing the node's network configuration.

Enumerator:

EMBER_NETWORK_INIT_NO_OPTIONS

EMBER_NETWORK_INIT_PARENT_INFO_IN_TOKEN The Parent Node ID and EUI64 are stored in a token. This prevents the need to perform an Orphan scan on startup.

Definition at line 456 of file [ember-types.h](#).

6.2.4.7 enum EmberApsOption

Options to use when sending a message.

The discover route, APS retry, and APS indirect options may be used together. Poll response cannot be combined with any other options.

Enumerator:

EMBER_APS_OPTION_NONE No options.

EMBER_APS_OPTION_DSA_SIGN This signs the application layer message body (APS Frame not included) and appends the ECDSA signature to the end of the message. Needed by Smart Energy applications. This requires the CBKE and ECC libraries. The [emberDsaSignHandler\(\)](#) function is called after DSA signing is complete but before the message has been sent by the APS layer. Note that when passing a buffer to the stack for DSA signing, the final byte in the buffer has special significance as an indicator of how many leading bytes should be ignored for signature purposes. Refer to API documentation of [emberDsaSign\(\)](#) or the dsaSign EZSP command for further details about this requirement.

EMBER_APS_OPTION_ENCRYPTION Send the message using APS Encryption, using the Link Key shared with the destination node to encrypt the data at the APS Level.

EMBER_APS_OPTION_RETRY Resend the message using the APS retry mechanism. In the mesh stack, this option and the enable route discovery option must be enabled for an existing route to be repaired automatically.

EMBER_APS_OPTION_ENABLE_ROUTE_DISCOVERY Send the message with the NWK 'enable route discovery' flag, which causes a route discovery to be initiated if no route to the destination is known. Note that in the mesh stack, this option and the APS retry option must be enabled an existing route to be repaired automatically.

EMBER_APS_OPTION_FORCE_ROUTE_DISCOVERY Send the message with the NWK 'force route discovery' flag, which causes a route discovery to be initiated even if one is known.

EMBER_APS_OPTION_SOURCE_EUI64 Include the source EUI64 in the network frame.

EMBER_APS_OPTION_DESTINATION_EUI64 Include the destination EUI64 in the network frame.

EMBER_APS_OPTION_ENABLE_ADDRESS_DISCOVERY Send a ZDO request to discover the node ID of the destination, if it is not already know.

EMBER_APS_OPTION_POLL_RESPONSE This message is being sent in response to a call to ::emberPollHandler(). It causes the message to be sent immediately instead of being queued up until the next poll from the (end device) destination.

EMBER_APS_OPTION_ZDO_RESPONSE_REQUIRED This incoming message is a valid ZDO request and the application is responsible for sending a ZDO response. This flag is used only within emberIncomingMessageHandler() when EMBER_APPLICATION_RECEIVES_UNSUPPORTED_ZDO_REQUESTS is defined.

EMBER_APS_OPTION_FRAGMENT This message is part of a fragmented message. This option may only be set for unicasts. The groupId field gives the index of this fragment in the low-order byte. If the low-order byte is zero this is the first fragment and the high-order byte contains the number of fragments in the message.

Definition at line 486 of file [ember-types.h](#).

6.2.4.8 enum EmberIncomingMessageType

Defines the possible incoming message types.

Enumerator:

EMBER_INCOMING_UNICAST Unicast.

EMBER_INCOMING_UNICAST_REPLY Unicast reply.

EMBER_INCOMING_MULTICAST Multicast.

EMBER_INCOMING_MULTICAST_LOOPBACK Multicast sent by the local device.

EMBER_INCOMING_BROADCAST Broadcast.

EMBER_INCOMING_BROADCAST_LOOPBACK Broadcast sent by the local device.

Definition at line 559 of file [ember-types.h](#).

6.2.4.9 enum EmberOutgoingMessageType

Defines the possible outgoing message types.

Enumerator:

EMBER_OUTGOING_DIRECT Unicast sent directly to an EmberNodeId.

EMBER_OUTGOING_VIA_ADDRESS_TABLE Unicast sent using an entry in the address table.

EMBER_OUTGOING_VIA_BINDING Unicast sent using an entry in the binding table.

EMBER_OUTGOING_MULTICAST Multicast message. This value is passed to emberMessageSentHandler() only. It may not be passed to emberSendUnicast().

EMBER_OUTGOING_MULTICAST_WITH_ALIAS aliased multicast message. This value is passed to emberMessageSentHandler() only. It may not be passed to emberSendUnicast().

EMBER_OUTGOING_BROADCAST_WITH_ALIAS aliased Broadcast message. This value is passed to emberMessageSentHandler() only. It may not be passed to emberSendUnicast().

EMBER_OUTGOING_BROADCAST Broadcast message. This value is passed to emberMessageSentHandler() only. It may not be passed to emberSendUnicast().

Definition at line 584 of file [ember-types.h](#).

6.2.4.10 enum EmberZigbeeCommandType

A type of command received by the stack.

This enum provides a way to indicate which protocol layer in the Ember stack an incoming command was meant for.

Enumerator:

EMBER_ZIGBEE_COMMAND_TYPE_MAC Describes an 802.15.4 MAC layer command.

EMBER_ZIGBEE_COMMAND_TYPE_NWK Describes a ZigBee Network layer command.

EMBER_ZIGBEE_COMMAND_TYPE_APS Describes a ZigBee Application Support layer command.

EMBER_ZIGBEE_COMMAND_TYPE_ZDO Describes a ZigBee Device Object command.

EMBER_ZIGBEE_COMMAND_TYPE_ZCL Describes a ZigBee Cluster Library command.

EMBER_ZIGBEE_COMMAND_TYPE_BEACON Although a beacon is not a MAC command, we have it here for simplicity.

Definition at line 616 of file [ember-types.h](#).

6.2.4.11 enum EmberNetworkStatus

Defines the possible join states for a node.

Enumerator:

EMBER_NO_NETWORK The node is not associated with a network in any way.

EMBER_JOINING_NETWORK The node is currently attempting to join a network.

EMBER_JOINED_NETWORK The node is joined to a network.

EMBER_JOINED_NETWORK_NO_PARENT The node is an end device joined to a network but its parent is not responding.

EMBER_LEAVING_NETWORK The node is in the process of leaving its current network.

Definition at line 641 of file [ember-types.h](#).

6.2.4.12 enum EmberNetworkScanType

Type for a network scan.

Enumerator:

EMBER_ENERGY_SCAN An energy scan scans each channel for its RSSI value.

EMBER_ACTIVE_SCAN An active scan scans each channel for available networks.

Definition at line 665 of file [ember-types.h](#).

6.2.4.13 enum EmberBindingType

Defines binding types.

Enumerator:

EMBER_UNUSED_BINDING A binding that is currently not in use.

EMBER_UNICAST_BINDING A unicast binding whose 64-bit identifier is the destination EUI64.

EMBER_MANY_TO_ONE_BINDING A unicast binding whose 64-bit identifier is the many-to-one destination EUI64. Route discovery should be disabled when sending unicasts via many-to-one bindings.

EMBER_MULTICAST_BINDING A multicast binding whose 64-bit identifier is the group address. A multicast binding can be used to send messages to the group and to receive messages sent to the group.

Definition at line 682 of file [ember-types.h](#).

6.2.4.14 enum EmberJoinDecision

Decision made by the Trust Center when a node attempts to join.

Enumerator:

EMBER_USE_PRECONFIGURED_KEY Allow the node to join. The node has the key.
EMBER_SEND_KEY_IN_THE_CLEAR Allow the node to join. Send the key to the node.
EMBER_DENY_JOIN Deny join.
EMBER_NO_ACTION Take no action.

Definition at line 725 of file [ember-types.h](#).

6.2.4.15 enum EmberDeviceUpdate

The Status of the Update Device message sent to the Trust Center. The device may have joined or rejoined insecurely, rejoined securely, or left. MAC Security has been deprecated and therefore there is no secure join.

Enumerator:

EMBER_STANDARD_SECURITY_SECURED_REJOIN
EMBER_STANDARD_SECURITY_UNSECURED_JOIN
EMBER_DEVICE_LEFT
EMBER_STANDARD_SECURITY_UNSECURED_REJOIN
EMBER_HIGH_SECURITY_SECURED_REJOIN
EMBER_HIGH_SECURITY_UNSECURED_JOIN
EMBER_HIGH_SECURITY_UNSECURED_REJOIN

Definition at line 759 of file [ember-types.h](#).

6.2.4.16 enum EmberRejoinReason

Notes the last rejoin reason.

Enumerator:

EMBER_REJOIN_REASON_NONE
EMBER_REJOIN_DUE_TO_NWK_KEY_UPDATE
EMBER_REJOIN_DUE_TO_LEAVE_MESSAGE
EMBER_REJOIN_DUE_TO_NO_PARENT
EMBER_REJOIN_DUE_TO_ZLL_TOUCHLINK
EMBER_REJOIN_DUE_TO_APP_EVENT_5
EMBER_REJOIN_DUE_TO_APP_EVENT_4
EMBER_REJOIN_DUE_TO_APP_EVENT_3
EMBER_REJOIN_DUE_TO_APP_EVENT_2
EMBER_REJOIN_DUE_TO_APP_EVENT_1

Definition at line 793 of file [ember-types.h](#).

6.2.4.17 enum EmberClusterListId

Defines the lists of clusters that must be provided for each endpoint.

Enumerator:

EMBER_INPUT_CLUSTER_LIST Input clusters the endpoint will accept.

EMBER_OUTPUT_CLUSTER_LIST Output clusters the endpoint can send.

Definition at line 823 of file [ember-types.h](#).

6.2.4.18 enum EmberEventUnits

Either marks an event as inactive or specifies the units for the event execution time.

Enumerator:

EMBER_EVENT_INACTIVE The event is not scheduled to run.

EMBER_EVENT_MS_TIME The execution time is in approximate milliseconds.

EMBER_EVENT_QS_TIME The execution time is in 'binary' quarter seconds (256 approximate milliseconds each).

EMBER_EVENT_MINUTE_TIME The execution time is in 'binary' minutes (65536 approximate milliseconds each).

EMBER_EVENT_ZERO_DELAY The event is scheduled to run at the earliest opportunity.

Definition at line 841 of file [ember-types.h](#).

6.2.4.19 enum EmberJoinMethod

The type of method used for joining.

Enumerator:

EMBER_USE_MAC_ASSOCIATION Normally devices use MAC Association to join a network, which respects the "permit joining" flag in the MAC Beacon. For mobile nodes this value causes the device to use an Ember Mobile Node Join, which is functionally equivalent to a MAC association. This value should be used by default.

EMBER_USE_NWK_REJOIN For those networks where the "permit joining" flag is never turned on, they will need to use a ZigBee NWK Rejoin. This value causes the rejoin to be sent withOUT NWK security and the Trust Center will be asked to send the NWK key to the device. The NWK key sent to the device can be encrypted with the device's corresponding Trust Center link key. That is determined by the [EmberJoinDecision](#) on the Trust Center returned by the `::emberTrustCenterJoinHandler()`. For a mobile node this value will cause it to use an Ember Mobile node rejoin, which is functionally equivalent.

EMBER_USE_NWK_REJOIN_HAVE_NWK_KEY

EMBER_USE_NWK_COMMISSIONING For those networks where all network and security information is known ahead of time, a router device may be commissioned such that it does not need to send any messages to begin communicating on the network.

Definition at line 866 of file [ember-types.h](#).

6.2.4.20 enum EmberCounterType

Defines the events reported to the application by the `::emberCounterHandler()`.

Enumerator:

EMBER_COUNTER_MAC_RX_BROADCAST The MAC received a broadcast.

EMBER_COUNTER_MAC_TX_BROADCAST The MAC transmitted a broadcast.

EMBER_COUNTER_MAC_RX_UNICAST The MAC received a unicast.

EMBER_COUNTER_MAC_TX_UNICAST_SUCCESS The MAC successfully transmitted a unicast.

EMBER_COUNTER_MAC_TX_UNICAST_RETRY The MAC retried a unicast. This is a placeholder and is not used by the `::emberCounterHandler()` callback. Instead the number of MAC retries are returned in the data parameter of the callback for the [EMBER_COUNTER_MAC_TX_UNICAST_SUCCESS](#) and [EMBER_COUNTER_MAC_TX_UNICAST_FAILED](#) types.

EMBER_COUNTER_MAC_TX_UNICAST_FAILED The MAC unsuccessfully transmitted a unicast.

EMBER_COUNTER_APS_DATA_RX_BROADCAST The APS layer received a data broadcast.

EMBER_COUNTER_APS_DATA_TX_BROADCAST The APS layer transmitted a data broadcast.

EMBER_COUNTER_APS_DATA_RX_UNICAST The APS layer received a data unicast.

EMBER_COUNTER_APS_DATA_TX_UNICAST_SUCCESS The APS layer successfully transmitted a data unicast.

EMBER_COUNTER_APS_DATA_TX_UNICAST_RETRY The APS layer retried a data unicast. This is a placeholder and is not used by the `::emberCounterHandler()` callback. Instead the number of APS retries are returned in the data parameter of the callback for the [EMBER_COUNTER_APS_DATA_TX_UNICAST_SUCCESS](#) and [EMBER_COUNTER_APS_DATA_TX_UNICAST_FAILED](#) types.

EMBER_COUNTER_APS_DATA_TX_UNICAST_FAILED The APS layer unsuccessfully transmitted a data unicast.

EMBER_COUNTER_ROUTE_DISCOVERY_INITIATED The network layer successfully submitted a new route discovery to the MAC.

EMBER_COUNTER_NEIGHBOR_ADDED An entry was added to the neighbor table.

EMBER_COUNTER_NEIGHBOR_REMOVED An entry was removed from the neighbor table.

EMBER_COUNTER_NEIGHBOR_STALE A neighbor table entry became stale because it had not been heard from.

EMBER_COUNTER_JOIN_INDICATION A node joined or rejoined to the network via this node.

EMBER_COUNTER_CHILD_REMOVED An entry was removed from the child table.

- EMBER_COUNTER_ASH_OVERFLOW_ERROR*** EZSP-UART only. An overflow error occurred in the UART.
- EMBER_COUNTER_ASH_FRAMING_ERROR*** EZSP-UART only. A framing error occurred in the UART.
- EMBER_COUNTER_ASH_OVERRUN_ERROR*** EZSP-UART only. An overrun error occurred in the UART.
- EMBER_COUNTER_NWK_FRAME_COUNTER_FAILURE*** A message was dropped at the Network layer because the NWK frame counter was not higher than the last message seen from that source.
- EMBER_COUNTER_APS_FRAME_COUNTER_FAILURE*** A message was dropped at the APS layer because the APS frame counter was not higher than the last message seen from that source.
- EMBER_COUNTER_ASH_XOFF*** EZSP-UART only. An XOFF was transmitted by the UART.
- EMBER_COUNTER_APS_LINK_KEY_NOT_AUTHORIZED*** A message was dropped at the APS layer because it had APS encryption but the key associated with the sender has not been authenticated, and thus the key is not authorized for use in APS data messages.
- EMBER_COUNTER_NWK_DECRYPTION_FAILURE*** A NWK encrypted message was received but dropped because decryption failed.
- EMBER_COUNTER_APS_DECRYPTION_FAILURE*** An APS encrypted message was received but dropped because decryption failed.
- EMBER_COUNTER_ALLOCATE_PACKET_BUFFER_FAILURE*** The number of times we failed to allocate a set of linked packet buffers. This doesn't necessarily mean that the packet buffer count was 0 at the time, but that the number requested was greater than the number free.
- EMBER_COUNTER_RELAYED_UNICAST*** The number of relayed unicast packets.
- EMBER_COUNTER_PHY_TO_MAC_QUEUE_LIMIT_REACHED*** The number of times we dropped a packet due to reaching the preset PHY to MAC queue limit (`emMaxPhyToMacQueueLength`). The limit will determine how many messages are accepted by the PHY between calls to `emberTick()`. After that limit is hit, packets will be dropped. The number of dropped packets will be recorded in this counter.
- NOTE: For each call to `emberCounterHandler()` there may be more than 1 packet that was dropped due to the limit reached. The actual number of packets dropped will be returned in the 'data' parameter passed to that function.
- EMBER_COUNTER_PACKET_VALIDATE_LIBRARY_DROPPED_COUNT*** The number of times we dropped a packet due to the packet-validate library checking a packet and rejecting it due to length or other formatting problems.
- EMBER_COUNTER_TYPE_NWK_RETRY_OVERFLOW*** The number of times the NWK retry queue is full and a new message failed to be added.
- EMBER_COUNTER_PHY_CCA_FAIL_COUNT*** The number of times the PHY layer was unable to transmit due to a failed CCA
- EMBER_COUNTER_BROADCAST_TABLE_FULL*** The number of times a NWK broadcast was dropped because the broadcast table was full.
- EMBER_COUNTER_TYPE_COUNT*** A placeholder giving the number of Ember counter types.

Definition at line 1089 of file [ember-types.h](#).

6.2.4.21 enum EmberInitialSecurityBitmask

This is the Initial Security Bitmask that controls the use of various security features.

Enumerator:

- EMBER_DISTRIBUTED_TRUST_CENTER_MODE** This enables Distributed Trust Center Mode for the device forming the network. (Previously known as [EMBE-R_NO_TRUST_CENTER_MODE](#))
- EMBER_TRUST_CENTER_GLOBAL_LINK_KEY** This enables a Global Link Key for the Trust Center. All nodes will share the same Trust Center Link Key.
- EMBER_PRECONFIGURED_NETWORK_KEY_MODE** This enables devices that perform MAC Association with a pre-configured Network Key to join the network. It is only set on the Trust Center.
- EMBER_HAVE_TRUST_CENTER_EUI64** This denotes that the [EmberInitialSecurityState::preconfiguredTrustCenterEui64](#) has a value in it containing the trust center EUI64. The device will only join a network and accept commands from a trust center with that EUI64. Normally this bit is NOT set, and the EUI64 of the trust center is learned during the join process. When commissioning a device to join onto an existing network that is using a trust center, and without sending any messages, this bit must be set and the field [EmberInitialSecurityState::preconfiguredTrustCenterEui64](#) must be populated with the appropriate EUI64.
- EMBER_TRUST_CENTER_USES_HASHED_LINK_KEY** This denotes that the [EmberInitialSecurityState::preconfiguredKey](#) is not the actual Link Key but a Root Key known only to the Trust Center. It is hashed with the IEEE Address of the destination device in order to create the actual Link Key used in encryption. This bit is only used by the Trust Center. The joining device need not set this.
- EMBER_HAVE_PRECONFIGURED_KEY** This denotes that the [EmberInitialSecurityState::preconfiguredKey](#) element has valid data that should be used to configure the initial security state.
- EMBER_HAVE_NETWORK_KEY** This denotes that the [EmberInitialSecurityState::networkKey](#) element has valid data that should be used to configure the initial security state.
- EMBER_GET_LINK_KEY_WHEN_JOINING** This denotes to a joining node that it should attempt to acquire a Trust Center Link Key during joining. This is necessary if the device does not have a pre-configured key, or wants to obtain a new one (since it may be using a well-known key during joining).
- EMBER_REQUIRE_ENCRYPTED_KEY** This denotes that a joining device should only accept an encrypted network key from the Trust Center (using its pre-configured key). A key sent in-the-clear by the Trust Center will be rejected and the join will fail. This option is only valid when utilizing a pre-configured key.
- EMBER_NO_FRAME_COUNTER_RESET** This denotes whether the device should NOT reset its outgoing frame counters (both NWK and APS) when `::emberSetInitialSecurityState()` is called. Normally it is advised to reset the frame counter before joining a new network. However in cases where a device is joining to the same network again (but not using `::emberRejoinNetwork()`) it should keep the NWK and APS frame counters stored in its tokens.
- NOTE: The application is allowed to dynamically change the behavior via `EMBER_EXT_NO_FRAME_COUNTER_RESET` field.
- EMBER_GET_PRECONFIGURED_KEY_FROM_INSTALL_CODE** This denotes that the device should obtain its preconfigured key from an installation code

stored in the manufacturing token. The token contains a value that will be hashed to obtain the actual preconfigured key. If that token is not valid than the call to `::emberSetInitialSecurityState()` will fail.

Definition at line 1586 of file [ember-types.h](#).

6.2.4.22 enum EmberExtendedSecurityBitmask

This is the Extended Security Bitmask that controls the use of various extended security features.

Enumerator:

EMBER_JOINER_GLOBAL_LINK_KEY This denotes whether a joiner node (router or end-device) uses a Global Link Key or a Unique Link Key.

EMBER_EXT_NO_FRAME_COUNTER_RESET This denotes whether the device's outgoing frame counter is allowed to be reset during forming or joining. If flag is set, the outgoing frame counter is not allowed to be reset. If flag is not set, the frame counter is allowed to be reset.

EMBER_NWK_LEAVE_REQUEST_NOT_ALLOWED This denotes whether a router node should discard or accept network Leave Commands.

Definition at line 1683 of file [ember-types.h](#).

6.2.4.23 enum EmberCurrentSecurityBitmask

This is the Current Security Bitmask that details the use of various security features.

Enumerator:

EMBER_STANDARD_SECURITY_MODE_ This denotes that the device is running in a network with ZigBee Standard Security.

EMBER_DISTRIBUTED_TRUST_CENTER_MODE_ This denotes that the device is running in a network without a centralized Trust Center.

EMBER_TRUST_CENTER_GLOBAL_LINK_KEY_ This denotes that the device has a Global Link Key. The Trust Center Link Key is the same across multiple nodes.

EMBER_HAVE_TRUST_CENTER_LINK_KEY This denotes that the node has a Trust Center Link Key.

EMBER_TRUST_CENTER_USES_HASHED_LINK_KEY_ This denotes that the Trust Center is using a Hashed Link Key.

Definition at line 1791 of file [ember-types.h](#).

6.2.4.24 enum EmberKeyStructBitmask

This bitmask describes the presence of fields within the [EmberKeyStruct](#).

Enumerator:

EMBER_KEY_HAS_SEQUENCE_NUMBER This indicates that the key has a sequence number associated with it. (i.e. a Network Key).

EMBER_KEY_HAS_OUTGOING_FRAME_COUNTER This indicates that the key has an outgoing frame counter and the corresponding value within the [EmberKeyStruct](#) has been populated with the data.

EMBER_KEY_HAS_INCOMING_FRAME_COUNTER This indicates that the key has an incoming frame counter and the corresponding value within the [EmberKeyStruct](#) has been populated with the data.

EMBER_KEY_HAS_PARTNER_EUI64 This indicates that the key has an associated Partner EUI64 address and the corresponding value within the [EmberKeyStruct](#) has been populated with the data.

EMBER_KEY_IS_AUTHORIZED This indicates the key is authorized for use in APS data messages. If the key is not authorized for use in APS data messages it has not yet gone through a key agreement protocol, such as CBKE (i.e. ECC)

EMBER_KEY_PARTNER_IS_SLEEPY This indicates that the partner associated with the link is a sleepy end device. This bit is set automatically if the local device hears a device announce from the partner indicating it is not an 'RX on when idle' device.

Definition at line 1843 of file [ember-types.h](#).

6.2.4.25 enum EmberKeyType

This denotes the type of security key.

Enumerator:

EMBER_TRUST_CENTER_LINK_KEY This denotes that the key is a Trust Center Link Key.

EMBER_TRUST_CENTER_MASTER_KEY This denotes that the key is a Trust Center Master Key.

EMBER_CURRENT_NETWORK_KEY This denotes that the key is the Current Network Key.

EMBER_NEXT_NETWORK_KEY This denotes that the key is the Next Network Key.

EMBER_APPLICATION_LINK_KEY This denotes that the key is an Application Link Key

EMBER_APPLICATION_MASTER_KEY This denotes that the key is an Application Master Key

Definition at line 1878 of file [ember-types.h](#).

6.2.4.26 enum EmberKeyStatus

This denotes the status of an attempt to establish a key with another device.

Enumerator:

EMBER_KEY_STATUS_NONE

EMBER_APP_LINK_KEY_ESTABLISHED

EMBER_APP_MASTER_KEY_ESTABLISHED

EMBER_TRUST_CENTER_LINK_KEY_ESTABLISHED
EMBER_KEY_ESTABLISHMENT_TIMEOUT
EMBER_KEY_TABLE_FULL
EMBER_TC_RESPONDED_TO_KEY_REQUEST
EMBER_TC_APP_KEY_SENT_TO_REQUESTER
EMBER_TC_RESPONSE_TO_KEY_REQUEST_FAILED
EMBER_TC_REQUEST_KEY_TYPE_NOT_SUPPORTED
EMBER_TC_NO_LINK_KEY_FOR_REQUESTER
EMBER_TC_REQUESTER_EUI64_UNKNOWN
EMBER_TC_RECEIVED_FIRST_APP_KEY_REQUEST
EMBER_TC_TIMEOUT_WAITING_FOR_SECOND_APP_KEY_REQUEST
EMBER_TC_NON_MATCHING_APP_KEY_REQUEST_RECEIVED
EMBER_TC_FAILED_TO_SEND_APP_KEYS
EMBER_TC_FAILED_TO_STORE_APP_KEY_REQUEST
EMBER_TC_REJECTED_APP_KEY_REQUEST
EMBER_TC_FAILED_TO_GENERATE_NEW_KEY
EMBER_TC_FAILED_TO_SEND_TC_KEY
EMBER_TRUST_CENTER_IS_PRE_R21
EMBER_TC_REQUESTER_VERIFY_KEY_TIMEOUT
EMBER_TC_REQUESTER_VERIFY_KEY_FAILURE
EMBER_TC_REQUESTER_VERIFY_KEY_SUCCESS
EMBER_VERIFY_LINK_KEY_FAILURE
EMBER_VERIFY_LINK_KEY_SUCCESS

Definition at line 1929 of file [ember-types.h](#).

6.2.4.27 enum EmberLinkKeyRequestPolicy

This enumeration determines whether or not a Trust Center answers link key requests.

Enumerator:

EMBER_DENY_KEY_REQUESTS
EMBER_ALLOW_KEY_REQUESTS
EMBER_GENERATE_NEW_TC_LINK_KEY

Definition at line 1982 of file [ember-types.h](#).

6.2.4.28 enum EmberKeySettings

Enumerator:

EMBER_KEY_PERMISSIONS_NONE
EMBER_KEY_PERMISSIONS_READING_ALLOWED
EMBER_KEY_PERMISSIONS_HASHING_ALLOWED

Definition at line 2114 of file [ember-types.h](#).

6.2.4.29 enum EmberMacPassthroughType

The types of MAC passthrough messages that an application may receive. This is a bit-mask.

Enumerator:

EMBER_MAC_PASSTHROUGH_NONE No MAC passthrough messages
EMBER_MAC_PASSTHROUGH_SE_INTERPAN SE InterPAN messages
EMBER_MAC_PASSTHROUGH_EMBERNET EmberNet and first generation (v1) standalone bootloader messages
EMBER_MAC_PASSTHROUGH_EMBERNET_SOURCE EmberNet messages filtered by their source address.
EMBER_MAC_PASSTHROUGH_APPLICATION Application-specific passthrough messages.
EMBER_MAC_PASSTHROUGH_CUSTOM Custom inter-pan filter

Definition at line [2146](#) of file [ember-types.h](#).

6.2.4.30 enum EmberZdoStatus

Enumerator:

EMBER_ZDP_SUCCESS
EMBER_ZDP_INVALID_REQUEST_TYPE
EMBER_ZDP_DEVICE_NOT_FOUND
EMBER_ZDP_INVALID_ENDPOINT
EMBER_ZDP_NOT_ACTIVE
EMBER_ZDP_NOT_SUPPORTED
EMBER_ZDP_TIMEOUT
EMBER_ZDP_NO_MATCH
EMBER_ZDP_NO_ENTRY
EMBER_ZDP_NO_DESCRIPTOR
EMBER_ZDP_INSUFFICIENT_SPACE
EMBER_ZDP_NOT_PERMITTED
EMBER_ZDP_TABLE_FULL
EMBER_ZDP_NOT_AUTHORIZED
EMBER_NWK_ALREADY_PRESENT
EMBER_NWK_TABLE_FULL
EMBER_NWK_UNKNOWN_DEVICE

Definition at line [2237](#) of file [ember-types.h](#).

6.2.4.31 enum EmberZdoServerMask

Enumerator:

EMBER_ZDP_PRIMARY_TRUST_CENTER
EMBER_ZDP_SECONDARY_TRUST_CENTER
EMBER_ZDP_PRIMARY_BINDING_TABLE_CACHE
EMBER_ZDP_SECONDARY_BINDING_TABLE_CACHE
EMBER_ZDP_PRIMARY_DISCOVERY_CACHE
EMBER_ZDP_SECONDARY_DISCOVERY_CACHE
EMBER_ZDP_NETWORK_MANAGER

Definition at line 2447 of file [ember-types.h](#).

6.2.4.32 enum EmberZdoConfigurationFlags

Enumerator:

EMBER_APP_RECEIVES_SUPPORTED_ZDO_REQUESTS
EMBER_APP_HANDLES_UNSUPPORTED_ZDO_REQUESTS
EMBER_APP_HANDLES_ZDO_ENDPOINT_REQUESTS
EMBER_APP_HANDLES_ZDO_BINDING_REQUESTS

Definition at line 2721 of file [ember-types.h](#).

6.2.5 Function Documentation

6.2.5.1 uint8_t* emberKeyContents (EmberKeyData * key)

This function allows the programmer to gain access to the actual key data bytes of the [EmberKeyData](#) struct.

Parameters

<i>key</i>	A Pointer to an EmberKeyData structure.
------------	---

Returns

uint8_t* Returns a pointer to the first byte of the Key data.

6.2.5.2 uint8_t* emberCertificateContents (EmberCertificateData * cert)

This function allows the programmer to gain access to the actual certificate data bytes of the [EmberCertificateData](#) struct.

Parameters

<i>cert</i>	A Pointer to an EmberCertificateData structure.
-------------	---

Returns

uint8_t* Returns a pointer to the first byte of the certificate data.

6.2.5.3 uint8_t* emberPublicKeyContents (EmberPublicKeyData * key)

This function allows the programmer to gain access to the actual public key data bytes of the [EmberPublicKeyData](#) struct.

Parameters

<i>key</i>	A Pointer to an EmberPublicKeyData structure.
------------	---

Returns

uint8_t* Returns a pointer to the first byte of the public key data.

6.2.5.4 uint8_t* emberPrivateKeyContents (EmberPrivateKeyData * key)

This function allows the programmer to gain access to the actual private key data bytes of the [EmberPrivateKeyData](#) struct.

Parameters

<i>key</i>	A Pointer to an EmberPrivateKeyData structure.
------------	--

Returns

uint8_t* Returns a pointer to the first byte of the private key data.

6.2.5.5 uint8_t* emberSmacContents (EmberSmacData * key)

This function allows the programmer to gain access to the actual SMAC (Secured Message Authentication Code) data of the [EmberSmacData](#) struct.

6.2.5.6 uint8_t* emberSignatureContents (EmberSignatureData * sig)

This function allows the programmer to gain access to the actual ECDSA signature data of the [EmberSignatureData](#) struct.

6.2.5.7 uint8_t* emberCertificate283k1Contents (EmberCertificate283k1Data * cert)

This function allows the programmer to gain access to the actual certificate data bytes of the [Ember283k1CertificateData](#) struct.

Parameters

<i>cert</i>	A Pointer to an Ember283k1CertificateData structure.
-------------	--

Returns

uint8_t* Returns a pointer to the first byte of the certificate data.

6.2.5.8 uint8_t* emberPublicKey283k1Contents (EmberPublicKey283k1Data * key)

This function allows the programmer to gain access to the actual public key data bytes of the Ember283k1PublicKeyData struct.

Parameters

<i>key</i>	A Pointer to an Ember283k1PublicKeyData structure.
------------	--

Returns

uint8_t* Returns a pointer to the first byte of the public key data.

6.2.5.9 uint8_t* emberPrivateKey283k1Contents (EmberPrivateKey283k1Data * key)

This function allows the programmer to gain access to the actual private key data bytes of the Ember283k1PrivateKeyData struct.

Parameters

<i>key</i>	A Pointer to an Ember283k1PrivateKeyData structure.
------------	---

Returns

uint8_t* Returns a pointer to the first byte of the private key data.

6.2.5.10 uint8_t* ember283k1SignatureContents (Ember283k1SignatureData * sig)

This function allows the programmer to gain access to the actual ECDSA signature data of the Ember283k1SignatureData struct.

6.2.6 Variable Documentation**6.2.6.1 const EmberVersion emberVersion**

Struct containing the version info.

6.3 Sending and Receiving Messages

Data Structures

- struct [InterPanHeader](#)
A struct for keeping track of all of the header info.

Macros

- #define [INTER_PAN_UNICAST](#)
- #define [INTER_PAN_BROADCAST](#)
- #define [INTER_PAN_MULTICAST](#)
- #define [MAX_INTER_PAN_MAC_SIZE](#)
- #define [STUB_NWK_SIZE](#)
- #define [STUB_NWK_FRAME_CONTROL](#)
- #define [MAX_STUB_APS_SIZE](#)
- #define [MAX_INTER_PAN_HEADER_SIZE](#)
- #define [INTER_PAN_UNICAST](#)
- #define [INTER_PAN_BROADCAST](#)
- #define [INTER_PAN_MULTICAST](#)
- #define [MAX_INTER_PAN_MAC_SIZE](#)
- #define [STUB_NWK_SIZE](#)
- #define [STUB_NWK_FRAME_CONTROL](#)
- #define [MAX_STUB_APS_SIZE](#)
- #define [MAX_INTER_PAN_HEADER_SIZE](#)

Functions

- [EmberMessageBuffer](#) [makeInterPanMessage](#) ([InterPanHeader](#) *headerData, [EmberMessageBuffer](#) payload)
- uint8_t [parseInterPanMessage](#) ([EmberMessageBuffer](#) message, uint8_t startOffset, [InterPanHeader](#) *headerData)
- uint8_t [makeInterPanMessage](#) ([InterPanHeader](#) *headerData, uint8_t *message, uint8_t maxLength, uint8_t *payload, uint8_t payloadLength)
- uint8_t [parseInterPanMessage](#) (uint8_t *message, uint8_t messageLength, [InterPanHeader](#) *headerData)

6.3.1 Detailed Description

See also [ami-inter-pan.h](#) for source code.

See also [ami-inter-pan-host.h](#) for source code.

6.3.2 Macro Definition Documentation

6.3.2.1 #define INTER_PAN_UNICAST

Definition at line 29 of file [ami-inter-pan.h](#).

6.3.2.2 #define INTER_PAN_BROADCAST

Definition at line 30 of file [ami-inter-pan.h](#).

6.3.2.3 #define INTER_PAN_MULTICAST

Definition at line 31 of file [ami-inter-pan.h](#).

6.3.2.4 #define MAX_INTER_PAN_MAC_SIZE

Definition at line 34 of file [ami-inter-pan.h](#).

6.3.2.5 #define STUB_NWK_SIZE

Definition at line 38 of file [ami-inter-pan.h](#).

6.3.2.6 #define STUB_NWK_FRAME_CONTROL

Definition at line 39 of file [ami-inter-pan.h](#).

6.3.2.7 #define MAX_STUB_APS_SIZE

Definition at line 42 of file [ami-inter-pan.h](#).

6.3.2.8 #define MAX_INTER_PAN_HEADER_SIZE

Definition at line 45 of file [ami-inter-pan.h](#).

6.3.2.9 #define INTER_PAN_UNICAST

The three types of inter-PAN messages. The values are actually the corresponding APS frame controls. 0x03 is the special interPAN message type. Unicast mode is 0x00, broadcast mode is 0x08, and multicast mode is 0x0C.

Definition at line 28 of file [ami-inter-pan-host.h](#).

6.3.2.10 #define INTER_PAN_BROADCAST

Definition at line 29 of file [ami-inter-pan-host.h](#).

6.3.2.11 #define INTER_PAN_MULTICAST

Definition at line 30 of file [ami-inter-pan-host.h](#).

6.3.2.12 #define MAX_INTER_PAN_MAC_SIZE

Definition at line 34 of file [ami-inter-pan-host.h](#).

6.3.2.13 #define STUB_NWK_SIZE

Definition at line 38 of file [ami-inter-pan-host.h](#).

6.3.2.14 #define STUB_NWK_FRAME_CONTROL

Definition at line 39 of file [ami-inter-pan-host.h](#).

6.3.2.15 #define MAX_STUB_APS_SIZE

Definition at line 42 of file [ami-inter-pan-host.h](#).

6.3.2.16 #define MAX_INTER_PAN_HEADER_SIZE

Definition at line 45 of file [ami-inter-pan-host.h](#).

6.3.3 Function Documentation

6.3.3.1 EmberMessageBuffer makeInterPanMessage (InterPanHeader * *headerData*, EmberMessageBuffer *payload*)

Creates an interpan message suitable for passing to emberSendRawMessage().

6.3.3.2 uint8_t parseInterPanMessage (EmberMessageBuffer *message*, uint8_t *startOffset*, InterPanHeader * *headerData*)

This is meant to be called on the message and offset values passed to emberMacPassthroughMessageHandler(...). The header is parsed and the various fields are written to the [InterPanHeader](#). The returned value is the offset of the payload in the message, or 0 if the message is not a correctly formed AMI interPAN message.

6.3.3.3 uint8_t makeInterPanMessage (InterPanHeader * *headerData*, uint8_t * *message*, uint8_t *maxLength*, uint8_t * *payload*, uint8_t *payloadLength*)

Create an interpan message. *message* needs to have enough space for the message contents. Upon return, the return value will be the length of the message, or 0 in case of error.

6.3.3.4 uint8_t parseInterPanMessage (uint8_t * *message*, uint8_t *messageLength*, InterPanHeader * *headerData*)

This is meant to be called on the message passed to emberMacPassthroughMessageHandler(...). The header is parsed and the various fields are written to the [InterPanHeader](#). The returned value is the offset of the payload in the message, or 0 if the message is not a correctly formed AMI interPAN message.

6.4 Ember Status Codes

Macros

- #define [DEFINE_ERROR](#)(symbol, value)

Enumerations

- enum { [EMBER_ERROR_CODE_COUNT](#) }

Generic Messages

These messages are system wide.

- #define [EMBER_SUCCESS](#)(x00)
- #define [EMBER_ERR_FATAL](#)(x01)
- #define [EMBER_BAD_ARGUMENT](#)(x02)
- #define [EMBER_NOT_FOUND](#)(x03)
- #define [EMBER_EEPROM_MFG_STACK_VERSION_MISMATCH](#)(x04)
- #define [EMBER_INCOMPATIBLE_STATIC_MEMORY_DEFINITIONS](#)(x05)
- #define [EMBER_EEPROM_MFG_VERSION_MISMATCH](#)(x06)
- #define [EMBER_EEPROM_STACK_VERSION_MISMATCH](#)(x07)

Packet Buffer Module Errors

- #define [EMBER_NO_BUFFERS](#)(x18)

Serial Manager Errors

- #define [EMBER_SERIAL_INVALID_BAUD_RATE](#)(x20)
- #define [EMBER_SERIAL_INVALID_PORT](#)(x21)
- #define [EMBER_SERIAL_TX_OVERFLOW](#)(x22)
- #define [EMBER_SERIAL_RX_OVERFLOW](#)(x23)
- #define [EMBER_SERIAL_RX_FRAME_ERROR](#)(x24)
- #define [EMBER_SERIAL_RX_PARITY_ERROR](#)(x25)
- #define [EMBER_SERIAL_RX_EMPTY](#)(x26)
- #define [EMBER_SERIAL_RX_OVERRUN_ERROR](#)(x27)

MAC Errors

- #define [EMBER_MAC_TRANSMIT_QUEUE_FULL](#)(x39)
- #define [EMBER_MAC_UNKNOWN_HEADER_TYPE](#)(x3A)
- #define [EMBER_MAC_ACK_HEADER_TYPE](#)(x3B)
- #define [EMBER_MAC_SCANNING](#)(x3D)
- #define [EMBER_MAC_NO_DATA](#)(x31)
- #define [EMBER_MAC_JOINED_NETWORK](#)(x32)
- #define [EMBER_MAC_BAD_SCAN_DURATION](#)(x33)
- #define [EMBER_MAC_INCORRECT_SCAN_TYPE](#)(x34)

- #define [EMBER_MAC_INVALID_CHANNEL_MASK](#)(x35)
- #define [EMBER_MAC_COMMAND_TRANSMIT_FAILURE](#)(x36)
- #define [EMBER_MAC_NO_ACK_RECEIVED](#)(x40)
- #define [EMBER_MAC_RADIO_NETWORK_SWITCH_FAILED](#)(x41)
- #define [EMBER_MAC_INDIRECT_TIMEOUT](#)(x42)

Simulated EEPROM Errors

- #define [EMBER_SIM_EEPROM_ERASE_PAGE_GREEN](#)(x43)
- #define [EMBER_SIM_EEPROM_ERASE_PAGE_RED](#)(x44)
- #define [EMBER_SIM_EEPROM_FULL](#)(x45)
- #define [EMBER_SIM_EEPROM_INIT_1_FAILED](#)(x48)
- #define [EMBER_SIM_EEPROM_INIT_2_FAILED](#)(x49)
- #define [EMBER_SIM_EEPROM_INIT_3_FAILED](#)(x4A)
- #define [EMBER_SIM_EEPROM_REPAIRING](#)(x4D)

Flash Errors

- #define [EMBER_ERR_FLASH_WRITE_INHIBITED](#)(x46)
- #define [EMBER_ERR_FLASH_VERIFY_FAILED](#)(x47)
- #define [EMBER_ERR_FLASH_PROG_FAIL](#)(x4B)
- #define [EMBER_ERR_FLASH_ERASE_FAIL](#)(x4C)

Bootloader Errors

- #define [EMBER_ERR_BOOTLOADER_TRAP_TABLE_BAD](#)(x58)
- #define [EMBER_ERR_BOOTLOADER_TRAP_UNKNOWN](#)(x59)
- #define [EMBER_ERR_BOOTLOADER_NO_IMAGE](#)(x05A)

Transport Errors

- #define [EMBER_DELIVERY_FAILED](#)(x66)
- #define [EMBER_BINDING_INDEX_OUT_OF_RANGE](#)(x69)
- #define [EMBER_ADDRESS_TABLE_INDEX_OUT_OF_RANGE](#)(x6A)
- #define [EMBER_INVALID_BINDING_INDEX](#)(x6C)
- #define [EMBER_INVALID_CALL](#)(x70)
- #define [EMBER_COST_NOT_KNOWN](#)(x71)
- #define [EMBER_MAX_MESSAGE_LIMIT_REACHED](#)(x72)
- #define [EMBER_MESSAGE_TOO_LONG](#)(x74)
- #define [EMBER_BINDING_IS_ACTIVE](#)(x75)
- #define [EMBER_ADDRESS_TABLE_ENTRY_IS_ACTIVE](#)(x76)

Green Power status codes

- #define [EMBER_MATCH](#)(x78)
- #define [EMBER_DROP_FRAME](#)(x79)
- #define [EMBER_PASS_UNPROCESSED](#)(x7A)
- #define [EMBER_TX_THEN_DROP](#)(x7B)
- #define [EMBER_NO_SECURITY](#)(x7C)
- #define [EMBER_COUNTER_FAILURE](#)(x7D)
- #define [EMBER_AUTH_FAILURE](#)(x7E)
- #define [EMBER_UNPROCESSED](#)(x7F)

HAL Module Errors

- #define [EMBER_ADC_CONVERSION_DONE](#)(x80)
- #define [EMBER_ADC_CONVERSION_BUSY](#)(x81)
- #define [EMBER_ADC_CONVERSION_DEFERRED](#)(x82)
- #define [EMBER_ADC_NO_CONVERSION_PENDING](#)(x84)
- #define [EMBER_SLEEP_INTERRUPTED](#)(x85)

PHY Errors

- #define [EMBER_PHY_TX_UNDERFLOW](#)(x88)
- #define [EMBER_PHY_TX_INCOMPLETE](#)(x89)
- #define [EMBER_PHY_INVALID_CHANNEL](#)(x8A)
- #define [EMBER_PHY_INVALID_POWER](#)(x8B)
- #define [EMBER_PHY_TX_BUSY](#)(x8C)
- #define [EMBER_PHY_TX_CCA_FAIL](#)(x8D)
- #define [EMBER_PHY_OSCILLATOR_CHECK_FAILED](#)(x8E)
- #define [EMBER_PHY_ACK_RECEIVED](#)(x8F)

Return Codes Passed to emberStackStatusHandler()

See also ::emberStackStatusHandler().

- #define [EMBER_NETWORK_UP](#)(x90)
- #define [EMBER_NETWORK_DOWN](#)(x91)
- #define [EMBER_JOIN_FAILED](#)(x94)
- #define [EMBER_MOVE_FAILED](#)(x96)
- #define [EMBER_CANNOT_JOIN_AS_ROUTER](#)(x98)
- #define [EMBER_NODE_ID_CHANGED](#)(x99)
- #define [EMBER_PAN_ID_CHANGED](#)(x9A)
- #define [EMBER_CHANNEL_CHANGED](#)(x9B)
- #define [EMBER_NO_BEACONS](#)(xAB)
- #define [EMBER_RECEIVED_KEY_IN_THE_CLEAR](#)(xAC)
- #define [EMBER_NO_NETWORK_KEY_RECEIVED](#)(xAD)
- #define [EMBER_NO_LINK_KEY_RECEIVED](#)(xAE)
- #define [EMBER_PRECONFIGURED_KEY_REQUIRED](#)(xAF)

Security Errors

- #define [EMBER_KEY_INVALID](#)(xB2)
- #define [EMBER_INVALID_SECURITY_LEVEL](#)(x95)
- #define [EMBER_APS_ENCRYPTION_ERROR](#)(xA6)
- #define [EMBER_TRUST_CENTER_MASTER_KEY_NOT_SET](#)(xA7)
- #define [EMBER_SECURITY_STATE_NOT_SET](#)(xA8)
- #define [EMBER_KEY_TABLE_INVALID_ADDRESS](#)(xB3)
- #define [EMBER_SECURITY_CONFIGURATION_INVALID](#)(xB7)
- #define [EMBER_TOO_SOON_FOR_SWITCH_KEY](#)(xB8)
- #define [EMBER_SIGNATURE_VERIFY_FAILURE](#)(xB9)
- #define [EMBER_KEY_NOT_AUTHORIZED](#)(xBB)
- #define [EMBER_SECURITY_DATA_INVALID](#)(xBD)

Miscellaneous Network Errors

- #define [EMBER_NOT_JOINED](#)(x93)
- #define [EMBER_NETWORK_BUSY](#)(xA1)
- #define [EMBER_INVALID_ENDPOINT](#)(xA3)
- #define [EMBER_BINDING_HAS_CHANGED](#)(xA4)
- #define [EMBER_INSUFFICIENT_RANDOM_DATA](#)(xA5)
- #define [EMBER_SOURCE_ROUTE_FAILURE](#)(xA9)
- #define [EMBER_MANY_TO_ONE_ROUTE_FAILURE](#)(xAA)

Miscellaneous Utility Errors

- #define [EMBER_STACK_AND_HARDWARE_MISMATCH](#)(xB0)
- #define [EMBER_INDEX_OUT_OF_RANGE](#)(xB1)
- #define [EMBER_TABLE_FULL](#)(xB4)
- #define [EMBER_TABLE_ENTRY_ERASED](#)(xB6)
- #define [EMBER_LIBRARY_NOT_PRESENT](#)(xB5)
- #define [EMBER_OPERATION_IN_PROGRESS](#)(xBA)
- #define [EMBER_TRUST_CENTER_EUI_HAS_CHANGED](#)(xBC)

ZigBee RF4CE specific errors.

- #define [EMBER_NO_RESPONSE](#)(xC0)
- #define [EMBER_DUPLICATE_ENTRY](#)(xC1)
- #define [EMBER_NOT_PERMITTED](#)(xC2)
- #define [EMBER_DISCOVERY_TIMEOUT](#)(xC3)
- #define [EMBER_DISCOVERY_ERROR](#)(xC4)
- #define [EMBER_SECURITY_TIMEOUT](#)(xC5)
- #define [EMBER_SECURITY_FAILURE](#)(xC6)

Application Errors

These error codes are available for application use.

- #define [EMBER_APPLICATION_ERROR_0](#)(xF0)
- #define [EMBER_APPLICATION_ERROR_1](#)(xF1)
- #define [EMBER_APPLICATION_ERROR_2](#)(xF2)
- #define [EMBER_APPLICATION_ERROR_3](#)(xF3)
- #define [EMBER_APPLICATION_ERROR_4](#)(xF4)
- #define [EMBER_APPLICATION_ERROR_5](#)(xF5)
- #define [EMBER_APPLICATION_ERROR_6](#)(xF6)
- #define [EMBER_APPLICATION_ERROR_7](#)(xF7)
- #define [EMBER_APPLICATION_ERROR_8](#)(xF8)
- #define [EMBER_APPLICATION_ERROR_9](#)(xF9)
- #define [EMBER_APPLICATION_ERROR_10](#)(xFA)
- #define [EMBER_APPLICATION_ERROR_11](#)(xFB)
- #define [EMBER_APPLICATION_ERROR_12](#)(xFC)
- #define [EMBER_APPLICATION_ERROR_13](#)(xFD)
- #define [EMBER_APPLICATION_ERROR_14](#)(xFE)
- #define [EMBER_APPLICATION_ERROR_15](#)(xFF)

6.4.1 Detailed Description

Many EmberZNet API functions return an [EmberStatus](#) value to indicate the success or failure of the call. Return codes are one byte long. This page documents the possible status codes and their meanings.

See [error-def.h](#) for source code.

See also [error.h](#) for information on how the values for the return codes are built up from these definitions. The file [error-def.h](#) is separated from [error.h](#) because utilities will use this file to parse the return codes.

Note

Do not include [error-def.h](#) directly. It is included by [error.h](#) inside an enum typedef, which is in turn included by [ember.h](#).

6.4.2 Macro Definition Documentation

6.4.2.1 #define DEFINE_ERROR(*symbol*, *value*)

Macro used by [error-def.h](#) to define all of the return codes.

Parameters

<i>symbol</i>	The name of the constant being defined. All Ember returns begin with EMBER_. For example, ::EMBER_CONNECTION_OPEN.
<i>value</i>	The value of the return code. For example, 0x61.

Definition at line 35 of file [error.h](#).

6.4.2.2 #define EMBER_SUCCESS(x00)

The generic "no error" message.

Definition at line 43 of file [error-def.h](#).

6.4.2.3 #define EMBER_ERR_FATAL(x01)

The generic "fatal error" message.

Definition at line 53 of file [error-def.h](#).

6.4.2.4 #define EMBER_BAD_ARGUMENT(x02)

An invalid value was passed as an argument to a function.

Definition at line 63 of file [error-def.h](#).

6.4.2.5 #define EMBER_NOT_FOUND(x03)

The requested information was not found.

Definition at line 73 of file [error-def.h](#).

6.4.2.6 #define EMBER_EEPROM_MFG_STACK_VERSION_MISMATCH(x04)

The manufacturing and stack token format in non-volatile memory is different than what the stack expects (returned at initialization).

Definition at line 84 of file [error-def.h](#).

6.4.2.7 #define EMBER_INCOMPATIBLE_STATIC_MEMORY_DEFINITIONS(x05)

The static memory definitions in ember-static-memory.h are incompatible with this stack version.

Definition at line 95 of file [error-def.h](#).

6.4.2.8 #define EMBER_EEPROM_MFG_VERSION_MISMATCH(x06)

The manufacturing token format in non-volatile memory is different than what the stack expects (returned at initialization).

Definition at line 106 of file [error-def.h](#).

6.4.2.9 #define EMBER_EEPROM_STACK_VERSION_MISMATCH(x07)

The stack token format in non-volatile memory is different than what the stack expects (returned at initialization).

Definition at line 117 of file [error-def.h](#).

6.4.2.10 #define EMBER_NO_BUFFERS(x18)

There are no more buffers.

Definition at line 134 of file [error-def.h](#).

6.4.2.11 #define EMBER_SERIAL_INVALID_BAUD_RATE(x20)

Specified an invalid baud rate.

Definition at line 150 of file [error-def.h](#).

6.4.2.12 #define EMBER_SERIAL_INVALID_PORT(x21)

Specified an invalid serial port.

Definition at line 160 of file [error-def.h](#).

6.4.2.13 #define EMBER_SERIAL_TX_OVERFLOW(x22)

Tried to send too much data.

Definition at line 170 of file [error-def.h](#).

6.4.2.14 #define EMBER_SERIAL_RX_OVERFLOW(x23)

There was not enough space to store a received character and the character was dropped.

Definition at line 181 of file [error-def.h](#).

6.4.2.15 #define EMBER_SERIAL_RX_FRAME_ERROR(x24)

Detected a UART framing error.

Definition at line 191 of file [error-def.h](#).

6.4.2.16 #define EMBER_SERIAL_RX_PARITY_ERROR(x25)

Detected a UART parity error.

Definition at line 201 of file [error-def.h](#).

6.4.2.17 #define EMBER_SERIAL_RX_EMPTY(x26)

There is no received data to process.

Definition at line 211 of file [error-def.h](#).

6.4.2.18 #define EMBER_SERIAL_RX_OVERRUN_ERROR(x27)

The receive interrupt was not handled in time, and a character was dropped.

Definition at line 222 of file [error-def.h](#).

6.4.2.19 #define EMBER_MAC_TRANSMIT_QUEUE_FULL(x39)

The MAC transmit queue is full.

Definition at line 238 of file [error-def.h](#).

6.4.2.20 #define EMBER_MAC_UNKNOWN_HEADER_TYPE(x3A)

MAC header FCF error on receive.

Definition at line 249 of file [error-def.h](#).

6.4.2.21 #define EMBER_MAC_ACK_HEADER_TYPE(x3B)

MAC ACK header received.

Definition at line 258 of file [error-def.h](#).

6.4.2.22 #define EMBER_MAC_SCANNING(x3D)

The MAC can't complete this task because it is scanning.

Definition at line 269 of file [error-def.h](#).

6.4.2.23 #define EMBER_MAC_NO_DATA(x31)

No pending data exists for device doing a data poll.

Definition at line 279 of file [error-def.h](#).

6.4.2.24 #define EMBER_MAC_JOINED_NETWORK(x32)

Attempt to scan when we are joined to a network.

Definition at line 289 of file [error-def.h](#).

6.4.2.25 #define EMBER_MAC_BAD_SCAN_DURATION(x33)

Scan duration must be 0 to 14 inclusive. Attempt was made to scan with an incorrect duration value.

Definition at line 300 of file [error-def.h](#).

6.4.2.26 #define EMBER_MAC_INCORRECT_SCAN_TYPE(x34)

emberStartScan was called with an incorrect scan type.

Definition at line 310 of file [error-def.h](#).

6.4.2.27 #define EMBER_MAC_INVALID_CHANNEL_MASK(x35)

emberStartScan was called with an invalid channel mask.

Definition at line 320 of file [error-def.h](#).

6.4.2.28 **#define EMBER_MAC_COMMAND_TRANSMIT_FAILURE(x36)**

Failed to scan current channel because we were unable to transmit the relevant MAC command.

Definition at line 331 of file [error-def.h](#).

6.4.2.29 **#define EMBER_MAC_NO_ACK_RECEIVED(x40)**

We expected to receive an ACK following the transmission, but the MAC level ACK was never received.

Definition at line 342 of file [error-def.h](#).

6.4.2.30 **#define EMBER_MAC_RADIO_NETWORK_SWITCH_FAILED(x41)**

MAC failed to transmit a message because could not successfully perform a radio network switch.

Definition at line 353 of file [error-def.h](#).

6.4.2.31 **#define EMBER_MAC_INDIRECT_TIMEOUT(x42)**

Indirect data message timed out before polled.

Definition at line 363 of file [error-def.h](#).

6.4.2.32 **#define EMBER_SIM_EEPROM_ERASE_PAGE_GREEN(x43)**

The Simulated EEPROM is telling the application that there is at least one flash page to be erased. The GREEN status means the current page has not filled above the ::ERASE_CRITICAL_THRESHOLD.

The application should call the function ::halSimEepromErasePage() when it can to erase a page.

Definition at line 386 of file [error-def.h](#).

6.4.2.33 **#define EMBER_SIM_EEPROM_ERASE_PAGE_RED(x44)**

The Simulated EEPROM is telling the application that there is at least one flash page to be erased. The RED status means the current page has filled above the ::ERASE_CRITICAL_THRESHOLD.

Due to the shrinking availability of write space, there is a danger of data loss. The application must call the function ::halSimEepromErasePage() as soon as possible to erase a page.

Definition at line 402 of file [error-def.h](#).

6.4.2.34 **#define EMBER_SIM_EEPROM_FULL(x45)**

The Simulated EEPROM has run out of room to write any new data and the data trying to be set has been lost. This error code is the result of ignoring the ::SIM_EEPROM_ERASE_PAGE_RED error code.

The application must call the function ::halSimEepromErasePage() to make room for any further calls to set a token.

Definition at line 417 of file [error-def.h](#).

6.4.2.35 **#define EMBER_SIM_EEPROM_INIT_1_FAILED(x48)**

Attempt 1 to initialize the Simulated EEPROM has failed.

This failure means the information already stored in Flash (or a lack thereof), is fatally incompatible with the token information compiled into the code image being run.

Definition at line 435 of file [error-def.h](#).

6.4.2.36 **#define EMBER_SIM_EEPROM_INIT_2_FAILED(x49)**

Attempt 2 to initialize the Simulated EEPROM has failed.

This failure means Attempt 1 failed, and the token system failed to properly reload default tokens and reset the Simulated EEPROM.

Definition at line 448 of file [error-def.h](#).

6.4.2.37 **#define EMBER_SIM_EEPROM_INIT_3_FAILED(x4A)**

Attempt 3 to initialize the Simulated EEPROM has failed.

This failure means one or both of the tokens ::TOKEN_MFG_NVDATA_VERSION or ::TOKEN_STACK_NVDATA_VERSION were incorrect and the token system failed to properly reload default tokens and reset the Simulated EEPROM.

Definition at line 462 of file [error-def.h](#).

6.4.2.38 **#define EMBER_SIM_EEPROM_REPAIRING(x4D)**

The Simulated EEPROM is repairing itself.

While there's nothing for an app to do when the SimEE is going to repair itself (SimEE has to be fully functional for the rest of the system to work), alert the application to the fact that repairing is occurring. There are debugging scenarios where an app might want to know that repairing is happening; such as monitoring frequency.

Note

Common situations will trigger an expected repair, such as using an erased chip or changing token definitions.

Definition at line 480 of file [error-def.h](#).

6.4.2.39 `#define EMBER_ERR_FLASH_WRITE_INHIBITED(x46)`

A fatal error has occurred while trying to write data to the Flash. The target memory attempting to be programmed is already programmed. The flash write routines were asked to flip a bit from a 0 to 1, which is physically impossible and the write was therefore inhibited. The data in the flash cannot be trusted after this error.

Definition at line 501 of file [error-def.h](#).

6.4.2.40 `#define EMBER_ERR_FLASH_VERIFY_FAILED(x47)`

A fatal error has occurred while trying to write data to the Flash and the write verification has failed. The data in the flash cannot be trusted after this error, and it is possible this error is the result of exceeding the life cycles of the flash.

Definition at line 514 of file [error-def.h](#).

6.4.2.41 `#define EMBER_ERR_FLASH_PROG_FAIL(x4B)`

Description:

A fatal error has occurred while trying to write data to the flash, possibly due to write protection or an invalid address. The data in the flash cannot be trusted after this error, and it is possible this error is the result of exceeding the life cycles of the flash.

Definition at line 527 of file [error-def.h](#).

6.4.2.42 `#define EMBER_ERR_FLASH_ERASE_FAIL(x4C)`

Description:

A fatal error has occurred while trying to erase flash, possibly due to write protection. The data in the flash cannot be trusted after this error, and it is possible this error is the result of exceeding the life cycles of the flash.

Definition at line 540 of file [error-def.h](#).

6.4.2.43 `#define EMBER_ERR_BOOTLOADER_TRAP_TABLE_BAD(x58)`

The bootloader received an invalid message (failed attempt to go into bootloader).

Definition at line 559 of file [error-def.h](#).

6.4.2.44 `#define EMBER_ERR_BOOTLOADER_TRAP_UNKNOWN(x59)`

Bootloader received an invalid message (failed attempt to go into bootloader).

Definition at line 570 of file [error-def.h](#).

6.4.2.45 `#define EMBER_ERR_BOOTLOADER_NO_IMAGE(x05A)`

The bootloader cannot complete the bootload operation because either an image was not found or the image exceeded memory bounds.

Definition at line 581 of file [error-def.h](#).

6.4.2.46 **#define EMBER_DELIVERY_FAILED(x66)**

The APS layer attempted to send or deliver a message, but it failed.

Definition at line 599 of file [error-def.h](#).

6.4.2.47 **#define EMBER_BINDING_INDEX_OUT_OF_RANGE(x69)**

This binding index is out of range for the current binding table.

Definition at line 609 of file [error-def.h](#).

6.4.2.48 **#define EMBER_ADDRESS_TABLE_INDEX_OUT_OF_RANGE(x6A)**

This address table index is out of range for the current address table.

Definition at line 620 of file [error-def.h](#).

6.4.2.49 **#define EMBER_INVALID_BINDING_INDEX(x6C)**

An invalid binding table index was given to a function.

Definition at line 630 of file [error-def.h](#).

6.4.2.50 **#define EMBER_INVALID_CALL(x70)**

The API call is not allowed given the current state of the stack.

Definition at line 641 of file [error-def.h](#).

6.4.2.51 **#define EMBER_COST_NOT_KNOWN(x71)**

The link cost to a node is not known.

Definition at line 651 of file [error-def.h](#).

6.4.2.52 **#define EMBER_MAX_MESSAGE_LIMIT_REACHED(x72)**

The maximum number of in-flight messages (i.e. [EMBER_APS_UNICAST_MESSAGE_COUNT](#)) has been reached.

Definition at line 662 of file [error-def.h](#).

6.4.2.53 **#define EMBER_MESSAGE_TOO_LONG(x74)**

The message to be transmitted is too big to fit into a single over-the-air packet.

Definition at line 672 of file [error-def.h](#).

6.4.2.54 #define EMBER_BINDING_IS_ACTIVE(x75)

The application is trying to delete or overwrite a binding that is in use.

Definition at line 683 of file [error-def.h](#).

6.4.2.55 #define EMBER_ADDRESS_TABLE_ENTRY_IS_ACTIVE(x76)

The application is trying to overwrite an address table entry that is in use.

Definition at line 693 of file [error-def.h](#).

6.4.2.56 #define EMBER_MATCH(x78)

security match

Definition at line 710 of file [error-def.h](#).

6.4.2.57 #define EMBER_DROP_FRAME(x79)

drop frame

Definition at line 718 of file [error-def.h](#).

6.4.2.58 #define EMBER_PASS_UNPROCESSED(x7A)

security match

Definition at line 726 of file [error-def.h](#).

6.4.2.59 #define EMBER_TX_THEN_DROP(x7B)

security match

Definition at line 734 of file [error-def.h](#).

6.4.2.60 #define EMBER_NO_SECURITY(x7C)

security match

Definition at line 742 of file [error-def.h](#).

6.4.2.61 #define EMBER_COUNTER_FAILURE(x7D)

security match

Definition at line 750 of file [error-def.h](#).

6.4.2.62 #define EMBER_AUTH_FAILURE(x7E)

security match

Definition at line 758 of file [error-def.h](#).

6.4.2.63 #define EMBER_UNPROCESSED(x7F)

security match

Definition at line 766 of file [error-def.h](#).

6.4.2.64 #define EMBER_ADC_CONVERSION_DONE(x80)

Conversion is complete.

Definition at line 784 of file [error-def.h](#).

6.4.2.65 #define EMBER_ADC_CONVERSION_BUSY(x81)

Conversion cannot be done because a request is being processed.

Definition at line 795 of file [error-def.h](#).

6.4.2.66 #define EMBER_ADC_CONVERSION_DEFERRED(x82)

Conversion is deferred until the current request has been processed.

Definition at line 806 of file [error-def.h](#).

6.4.2.67 #define EMBER_ADC_NO_CONVERSION_PENDING(x84)

No results are pending.

Definition at line 816 of file [error-def.h](#).

6.4.2.68 #define EMBER_SLEEP_INTERRUPTED(x85)

Sleeping (for a duration) has been abnormally interrupted and exited prematurely.

Definition at line 827 of file [error-def.h](#).

6.4.2.69 #define EMBER_PHY_TX_UNDERFLOW(x88)

The transmit hardware buffer underflowed.

Definition at line 844 of file [error-def.h](#).

6.4.2.70 #define EMBER_PHY_TX_INCOMPLETE(x89)

The transmit hardware did not finish transmitting a packet.

Definition at line 854 of file [error-def.h](#).

6.4.2.71 #define EMBER_PHY_INVALID_CHANNEL(x8A)

An unsupported channel setting was specified.

Definition at line 864 of file [error-def.h](#).

6.4.2.72 #define EMBER_PHY_INVALID_POWER(x8B)

An unsupported power setting was specified.

Definition at line 874 of file [error-def.h](#).

6.4.2.73 #define EMBER_PHY_TX_BUSY(x8C)

The requested operation cannot be completed because the radio is currently busy, either transmitting a packet or performing calibration.

Definition at line 885 of file [error-def.h](#).

6.4.2.74 #define EMBER_PHY_TX_CCA_FAIL(x8D)

The transmit attempt failed because all CCA attempts indicated that the channel was busy.

Definition at line 896 of file [error-def.h](#).

6.4.2.75 #define EMBER_PHY_OSCILLATOR_CHECK_FAILED(x8E)

The software installed on the hardware doesn't recognize the hardware radio type.

Definition at line 907 of file [error-def.h](#).

6.4.2.76 #define EMBER_PHY_ACK_RECEIVED(x8F)

The expected ACK was received after the last transmission.

Definition at line 917 of file [error-def.h](#).

6.4.2.77 #define EMBER_NETWORK_UP(x90)

The stack software has completed initialization and is ready to send and receive packets over the air.

Definition at line 936 of file [error-def.h](#).

6.4.2.78 #define EMBER_NETWORK_DOWN(x91)

The network is not operating.

Definition at line 946 of file [error-def.h](#).

6.4.2.79 #define EMBER_JOIN_FAILED(x94)

An attempt to join a network failed.

Definition at line 956 of file [error-def.h](#).

6.4.2.80 #define EMBER_MOVE_FAILED(x96)

After moving, a mobile node's attempt to re-establish contact with the network failed.

Definition at line 967 of file [error-def.h](#).

6.4.2.81 #define EMBER_CANNOT_JOIN_AS_ROUTER(x98)

An attempt to join as a router failed due to a ZigBee versus ZigBee Pro incompatibility. ZigBee devices joining ZigBee Pro networks (or vice versa) must join as End Devices, not Routers.

Definition at line 979 of file [error-def.h](#).

6.4.2.82 #define EMBER_NODE_ID_CHANGED(x99)

The local node ID has changed. The application can obtain the new node ID by calling `::emberGetNodeId()`.

Definition at line 989 of file [error-def.h](#).

6.4.2.83 #define EMBER_PAN_ID_CHANGED(x9A)

The local PAN ID has changed. The application can obtain the new PAN ID by calling `::emberGetPanId()`.

Definition at line 999 of file [error-def.h](#).

6.4.2.84 #define EMBER_CHANNEL_CHANGED(x9B)

The channel has changed.

Definition at line 1007 of file [error-def.h](#).

6.4.2.85 #define EMBER_NO_BEACONS(xAB)

An attempt to join or rejoin the network failed because no router beacons could be heard by the joining node.

Definition at line 1016 of file [error-def.h](#).

6.4.2.86 #define EMBER_RECEIVED_KEY_IN_THE_CLEAR(xAC)

An attempt was made to join a Secured Network using a pre-configured key, but the Trust Center sent back a Network Key in-the-clear when an encrypted Network Key was required. ([EMBER_REQUIRE_ENCRYPTED_KEY](#)).

Definition at line 1027 of file [error-def.h](#).

6.4.2.87 #define EMBER_NO_NETWORK_KEY_RECEIVED(xAD)

An attempt was made to join a Secured Network, but the device did not receive a Network Key.

Definition at line 1037 of file [error-def.h](#).

6.4.2.88 **#define EMBER_NO_LINK_KEY_RECEIVED(xAE)**

After a device joined a Secured Network, a Link Key was requested ([EMBER_GET_LINK_KEY_WHEN_JOINING](#)) but no response was ever received.

Definition at line 1047 of file [error-def.h](#).

6.4.2.89 **#define EMBER_PRECONFIGURED_KEY_REQUIRED(xAF)**

An attempt was made to join a Secured Network without a pre-configured key, but the Trust Center sent encrypted data using a pre-configured key.

Definition at line 1058 of file [error-def.h](#).

6.4.2.90 **#define EMBER_KEY_INVALID(xB2)**

The passed key data is not valid. A key of all zeros or all F's are reserved values and cannot be used.

Definition at line 1074 of file [error-def.h](#).

6.4.2.91 **#define EMBER_INVALID_SECURITY_LEVEL(x95)**

The chosen security level (the value of [EMBER_SECURITY_LEVEL](#)) is not supported by the stack.

Definition at line 1084 of file [error-def.h](#).

6.4.2.92 **#define EMBER_APS_ENCRYPTION_ERROR(xA6)**

There was an error in trying to encrypt at the APS Level.

This could result from either an inability to determine the long address of the recipient from the short address (no entry in the binding table) or there is no link key entry in the table associated with the destination, or there was a failure to load the correct key into the encryption core.

Definition at line 1098 of file [error-def.h](#).

6.4.2.93 **#define EMBER_TRUST_CENTER_MASTER_KEY_NOT_SET(xA7)**

There was an attempt to form a network using High security without setting the Trust Center master key first.

Definition at line 1107 of file [error-def.h](#).

6.4.2.94 **#define EMBER_SECURITY_STATE_NOT_SET(xA8)**

There was an attempt to form or join a network with security without calling `::emberSetInitialSecurityState()` first.

Definition at line 1116 of file [error-def.h](#).

6.4.2.95 **#define EMBER_KEY_TABLE_INVALID_ADDRESS(xB3)**

There was an attempt to set an entry in the key table using an invalid long address. An entry cannot be set using either the local device's or Trust Center's IEEE address. Or an entry already exists in the table with the same IEEE address. An Address of all zeros or all F's are not valid addresses in 802.15.4.

Definition at line 1129 of file [error-def.h](#).

6.4.2.96 **#define EMBER_SECURITY_CONFIGURATION_INVALID(xB7)**

There was an attempt to set a security configuration that is not valid given the other security settings.

Definition at line 1138 of file [error-def.h](#).

6.4.2.97 **#define EMBER_TOO_SOON_FOR_SWITCH_KEY(xB8)**

There was an attempt to broadcast a key switch too quickly after broadcasting the next network key. The Trust Center must wait at least a period equal to the broadcast timeout so that all routers have a chance to receive the broadcast of the new network key.

Definition at line 1149 of file [error-def.h](#).

6.4.2.98 **#define EMBER_SIGNATURE_VERIFY_FAILURE(xB9)**

The received signature corresponding to the message that was passed to the CBKE Library failed verification, it is not valid.

Definition at line 1158 of file [error-def.h](#).

6.4.2.99 **#define EMBER_KEY_NOT_AUTHORIZED(xBB)**

The message could not be sent because the link key corresponding to the destination is not authorized for use in APS data messages. APS Commands (sent by the stack) are allowed. To use it for encryption of APS data messages it must be authorized using a key agreement protocol (such as CBKE).

Definition at line 1170 of file [error-def.h](#).

6.4.2.100 **#define EMBER_SECURITY_DATA_INVALID(xBD)**

The security data provided was not valid, or an integrity check failed.

Definition at line 1180 of file [error-def.h](#).

6.4.2.101 **#define EMBER_NOT_JOINED(x93)**

The node has not joined a network.

Definition at line 1198 of file [error-def.h](#).

6.4.2.102 #define EMBER_NETWORK_BUSY(xA1)

A message cannot be sent because the network is currently overloaded.

Definition at line 1208 of file [error-def.h](#).

6.4.2.103 #define EMBER_INVALID_ENDPOINT(xA3)

The application tried to send a message using an endpoint that it has not defined.

Definition at line 1219 of file [error-def.h](#).

6.4.2.104 #define EMBER_BINDING_HAS_CHANGED(xA4)

The application tried to use a binding that has been remotely modified and the change has not yet been reported to the application.

Definition at line 1230 of file [error-def.h](#).

6.4.2.105 #define EMBER_INSUFFICIENT_RANDOM_DATA(xA5)

An attempt to generate random bytes failed because of insufficient random data from the radio.

Definition at line 1240 of file [error-def.h](#).

6.4.2.106 #define EMBER_SOURCE_ROUTE_FAILURE(xA9)

A ZigBee route error command frame was received indicating that a source routed message from this node failed en route.

Definition at line 1250 of file [error-def.h](#).

6.4.2.107 #define EMBER_MANY_TO_ONE_ROUTE_FAILURE(xA4)

A ZigBee route error command frame was received indicating that a message sent to this node along a many-to-one route failed en route. The route error frame was delivered by an ad-hoc search for a functioning route.

Definition at line 1261 of file [error-def.h](#).

6.4.2.108 #define EMBER_STACK_AND_HARDWARE_MISMATCH(xB0)

A critical and fatal error indicating that the version of the stack trying to run does not match with the chip it is running on. The software (stack) on the chip must be replaced with software that is compatible with the chip.

Definition at line 1282 of file [error-def.h](#).

6.4.2.109 #define EMBER_INDEX_OUT_OF_RANGE(xB1)

An index was passed into the function that was larger than the valid range.

Definition at line 1293 of file [error-def.h](#).

6.4.2.110 **#define EMBER_TABLE_FULL(xB4)**

There are no empty entries left in the table.

Definition at line 1302 of file [error-def.h](#).

6.4.2.111 **#define EMBER_TABLE_ENTRY_ERASED(xB6)**

The requested table entry has been erased and contains no valid data.

Definition at line 1312 of file [error-def.h](#).

6.4.2.112 **#define EMBER_LIBRARY_NOT_PRESENT(xB5)**

The requested function cannot be executed because the library that contains the necessary functionality is not present.

Definition at line 1322 of file [error-def.h](#).

6.4.2.113 **#define EMBER_OPERATION_IN_PROGRESS(xBA)**

The stack accepted the command and is currently processing the request. The results will be returned via an appropriate handler.

Definition at line 1332 of file [error-def.h](#).

6.4.2.114 **#define EMBER_TRUST_CENTER_EUI_HAS_CHANGED(xBC)**

The EUI of the Trust center has changed due to a successful rejoin. The device may need to perform other authentication to verify the new TC is authorized to take over.

Definition at line 1343 of file [error-def.h](#).

6.4.2.115 **#define EMBER_NO_RESPONSE(xC0)**

The ZigBee RF4CE stack has not received the response it was waiting for.

Definition at line 1360 of file [error-def.h](#).

6.4.2.116 **#define EMBER_DUPLICATE_ENTRY(xC1)**

The ZigBee RF4CE stack has detected a duplicate entry in the pairing table.

Definition at line 1370 of file [error-def.h](#).

6.4.2.117 **#define EMBER_NOT_PERMITTED(xC2)**

A pairing request was denied by the recipient node or an attempt to update a security link key was not possible due to one or more nodes not supporting security.

Definition at line 1381 of file [error-def.h](#).

6.4.2.118 **#define EMBER_DISCOVERY_TIMEOUT(xC3)**

The node has timed out during auto discovery response mode.

Definition at line 1390 of file [error-def.h](#).

6.4.2.119 **#define EMBER_DISCOVERY_ERROR(xC4)**

The node has received two matching discovery request command frames from two different nodes while in auto discovery response mode.

Definition at line 1401 of file [error-def.h](#).

6.4.2.120 **#define EMBER_SECURITY_TIMEOUT(xC5)**

The node has timed while transferring the (n+1) key seed messages to the pairing originator.

Definition at line 1412 of file [error-def.h](#).

6.4.2.121 **#define EMBER_SECURITY_FAILURE(xC6)**

Generic error code indicating a security failure.

Definition at line 1422 of file [error-def.h](#).

6.4.2.122 **#define EMBER_APPLICATION_ERROR_0(xF0)**

This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.

Definition at line 1440 of file [error-def.h](#).

6.4.2.123 **#define EMBER_APPLICATION_ERROR_1(xF1)**

This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.

Definition at line 1441 of file [error-def.h](#).

6.4.2.124 **#define EMBER_APPLICATION_ERROR_2(xF2)**

This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.

Definition at line 1442 of file [error-def.h](#).

6.4.2.125 **#define EMBER_APPLICATION_ERROR_3(xF3)**

This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.

Definition at line 1443 of file [error-def.h](#).

6.4.2.126 **#define EMBER_APPLICATION_ERROR_4(xF4)**

This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.

Definition at line 1444 of file [error-def.h](#).

6.4.2.127 **#define EMBER_APPLICATION_ERROR_5(xF5)**

This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.

Definition at line 1445 of file [error-def.h](#).

6.4.2.128 **#define EMBER_APPLICATION_ERROR_6(xF6)**

This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.

Definition at line 1446 of file [error-def.h](#).

6.4.2.129 **#define EMBER_APPLICATION_ERROR_7(xF7)**

This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.

Definition at line 1447 of file [error-def.h](#).

6.4.2.130 **#define EMBER_APPLICATION_ERROR_8(xF8)**

This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.

Definition at line 1448 of file [error-def.h](#).

6.4.2.131 **#define EMBER_APPLICATION_ERROR_9(xF9)**

This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.

Definition at line 1449 of file [error-def.h](#).

6.4.2.132 **#define EMBER_APPLICATION_ERROR_10(xFA)**

This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.

Definition at line 1450 of file [error-def.h](#).

6.4.2.133 `#define EMBER_APPLICATION_ERROR_11(xFB)`

This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.

Definition at line 1451 of file [error-def.h](#).

6.4.2.134 `#define EMBER_APPLICATION_ERROR_12(xFC)`

This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.

Definition at line 1452 of file [error-def.h](#).

6.4.2.135 `#define EMBER_APPLICATION_ERROR_13(xFD)`

This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.

Definition at line 1453 of file [error-def.h](#).

6.4.2.136 `#define EMBER_APPLICATION_ERROR_14(xFE)`

This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.

Definition at line 1454 of file [error-def.h](#).

6.4.2.137 `#define EMBER_APPLICATION_ERROR_15(xFF)`

This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.

Definition at line 1455 of file [error-def.h](#).

6.4.3 Enumeration Type Documentation

6.4.3.1 anonymous enum

Enumerator:

EMBER_ERROR_CODE_COUNT Gets defined as a count of all the possible return codes in the EmberZNet stack API.

Definition at line 39 of file [error.h](#).

6.5 Smart Energy Security

Functions

- `EmberStatus emberGetCertificate` (`EmberCertificateData *result`)
- `EmberStatus emberGenerateCbkeKeys` (`void`)
- `EmberStatus emberCalculateSmacs` (`bool amInitiator`, `EmberCertificateData *partnerCert`, `EmberPublicKeyData *partnerEphemeralPublicKey`)
- `EmberStatus emberClearTemporaryDataMaybeStoreLinkKey` (`bool storeLinkKey`)
- `EmberStatus emberDsaSign` (`EmberMessageBuffer messageToSign`)
- `void emberGenerateCbkeKeysHandler` (`EmberStatus status`, `EmberPublicKeyData *ephemeralPublicKey`)
- `void emberCalculateSmacsHandler` (`EmberStatus status`, `EmberSmacData *initiatorSmac`, `EmberSmacData *responderSmac`)
- `void emberDsaSignHandler` (`EmberStatus status`, `EmberMessageBuffer signedMessage`)
- `EmberStatus emberSetPreinstalledCbkeData` (`EmberPublicKeyData *caPublic`, `EmberCertificateData *myCert`, `EmberPrivateKeyData *myKey`)
- `bool emberGetStackCertificateEui64` (`EmberEUI64 certEui64`)
- `EmberStatus emberDsaVerify` (`EmberMessageDigest *digest`, `EmberCertificateData *signerCertificate`, `EmberSignatureData *receivedSig`)
- `void emberDsaVerifyHandler` (`EmberStatus status`)

6.5.1 Detailed Description

This file describes functionality for Certificate Based Key Exchange (CBKE). This is used by Smart Energy devices to generate and store ephemeral ECC keys, derive the SMACs for the Key establishment protocol, and sign messages using their private key for the Demand Response Load Control client cluster.

See [cbke-crypto-engine.h](#) for source code.

6.5.2 Function Documentation

6.5.2.1 `EmberStatus emberGetCertificate (EmberCertificateData * result)`

Retrieves the implicit certificate stored in the MFG tokens of the device.

Parameters

<i>result</i>	A pointer to an EmberCertificateData structure where the retrieved certificate will be stored.
---------------	--

Returns

[EMBER_SUCCESS](#) if the certificate was successfully retrieved. [EMBER_ERR_FATAL](#) if the token contains uninitialized data.

6.5.2.2 `EmberStatus emberGenerateCbkeKeys (void)`

This function begins the process of generating an ephemeral public/private ECC key pair.

If no other ECC operation is going on, it will immediately return with [EMBER_OPERATION_IN_PROGRESS](#). It will delay a period of time to let APS retries take place, but then it will shutdown the radio and consume the CPU processing until the key generation is complete. This may take up to 1 second.

The generated results of the key generation is returned via [emberGenerateCbkeKeysHandler\(\)](#).

Returns

[EMBER_OPERATION_IN_PROGRESS](#) if the stack has queued up the operation for execution.

6.5.2.3 EmberStatus emberCalculateSmacs (bool *amInitiator*, EmberCertificateData * *partnerCert*, EmberPublicKeyData * *partnerEphemeralPublicKey*)

This function will begin the process of generating the shared secret, the new link key, and the Secured Message Authentication Code (SMAC).

If no other ECC operation is going on, it will immediately return with [EMBER_OPERATION_IN_PROGRESS](#). It will delay a period of time to let APS retries take place, but then it will shutdown the radio and consume the CPU processing until SMACs calculations are complete. This may take up to 3.5 seconds.

The calculated SMACS are returned via [emberCalculateSmacsHandler\(\)](#).

Parameters

<i>amInitiator</i>	This bool indicates whether or not the device is the one that initiated the CBKE with the remote device, or whether it was the responder to the exchange.
<i>partnerCert</i>	A pointer to an EmberCertificateData structure that contains the CBKE partner's implicit certificate.
<i>partner-Ephemeral-PublicKey</i>	A pointer to an EmberPublicKeyData structure that contains the CBKE partner's ephemeral public key.

Returns

[EMBER_OPERATION_IN_PROGRESS](#) if the stack has queued up the operation for execution.

6.5.2.4 EmberStatus emberClearTemporaryDataMaybeStoreLinkKey (bool *storeLinkKey*)

This function should be called when all CBKE operations are done. Any temporary data created during calls to [emberGenerateCbkeKeys\(\)](#) or [emberCalculateSmacs\(\)](#) is wiped out. If the local device has validated that the partner device has generated the same SMACS as itself, it should set 'storeLinkKey' to true. Otherwise it should pass in false.

Parameters

<i>storeLinkKey</i>	This tells the stack whether to store the newly generated link key, or discard it.
---------------------	--

Returns

If storeLinkkey is false, this function returns `EMBER_ERR_FATAL` always. If storeLinkKey is true, then this function returns the results of whether or not the link key was stored. `EMBER_SUCCESS` is returned when key was stored successfully.

- 6.5.2.5 `EmberStatus emberDsaSign (EmberMessageBuffer messageToSign)`
- 6.5.2.6 `void emberGenerateCbkeKeysHandler (EmberStatus status,
EmberPublicKeyData * ephemeralPublicKey)`
- 6.5.2.7 `void emberCalculateSmacsHandler (EmberStatus status, EmberSmacData *
initiatorSmac, EmberSmacData * responderSmac)`
- 6.5.2.8 `void emberDsaSignHandler (EmberStatus status, EmberMessageBuffer
signedMessage)`
- 6.5.2.9 `EmberStatus emberSetPreinstalledCbkeData (EmberPublicKeyData *
caPublic, EmberCertificateData * myCert, EmberPrivateKeyData * myKey)`
- 6.5.2.10 `bool emberGetStackCertificateEui64 (EmberEUI64 certEui64)`
- 6.5.2.11 `EmberStatus emberDsaVerify (EmberMessageDigest * digest,
EmberCertificateData * signerCertificate, EmberSignatureData *
receivedSig)`
- 6.5.2.12 `void emberDsaVerifyHandler (EmberStatus status)`

6.6 Configuration

Macros

- #define [EMBER_API_MAJOR_VERSION](#)
- #define [EMBER_API_MINOR_VERSION](#)
- #define [EMBER_STACK_PROFILE](#)
- #define [EMBER_MAX_END_DEVICE_CHILDREN](#)
- #define [EMBER_SECURITY_LEVEL](#)
- #define [EMBER_CHILD_TABLE_SIZE](#)
- #define [EMBER_KEY_TABLE_SIZE](#)
- #define [EMBER_CERTIFICATE_TABLE_SIZE](#)
- #define [EMBER_MAX_DEPTH](#)
- #define [EMBER_MAX_HOPS](#)
- #define [EMBER_PACKET_BUFFER_COUNT](#)
- #define [EMBER_MAX_NEIGHBOR_TABLE_SIZE](#)
- #define [EMBER_NEIGHBOR_TABLE_SIZE](#)
- #define [EMBER_INDIRECT_TRANSMISSION_TIMEOUT](#)
- #define [EMBER_MAX_INDIRECT_TRANSMISSION_TIMEOUT](#)
- #define [EMBER_SEND_MULTICASTS_TO_SLEEPY_ADDRESS](#)
- #define [EMBER_END_DEVICE_POLL_TIMEOUT](#)
- #define [EMBER_END_DEVICE_POLL_TIMEOUT_SHIFT](#)
- #define [EMBER_MOBILE_NODE_POLL_TIMEOUT](#)
- #define [EMBER_APS_UNICAST_MESSAGE_COUNT](#)
- #define [EMBER_BINDING_TABLE_SIZE](#)
- #define [EMBER_ADDRESS_TABLE_SIZE](#)
- #define [EMBER_RESERVED_MOBILE_CHILD_ENTRIES](#)
- #define [EMBER_ROUTE_TABLE_SIZE](#)
- #define [EMBER_DISCOVERY_TABLE_SIZE](#)
- #define [EMBER_MULTICAST_TABLE_SIZE](#)
- #define [EMBER_SOURCE_ROUTE_TABLE_SIZE](#)
- #define [EMBER_DEFAULT_BROADCAST_TABLE_SIZE](#)
- #define [EMBER_BROADCAST_TABLE_SIZE](#)
- #define [EMBER_RETRY_QUEUE_SIZE](#)
- #define [EMBER_ASSERT_SERIAL_PORT](#)
- #define [EMBER_MAXIMUM_ALARM_DATA_SIZE](#)
- #define [EMBER_BROADCAST_ALARM_DATA_SIZE](#)
- #define [EMBER_UNICAST_ALARM_DATA_SIZE](#)
- #define [EMBER_FRAGMENT_DELAY_MS](#)
- #define [EMBER_FRAGMENT_MAX_WINDOW_SIZE](#)
- #define [EMBER_FRAGMENT_WINDOW_SIZE](#)
- #define [EMBER_BINDING_TABLE_TOKEN_SIZE](#)
- #define [EMBER_CHILD_TABLE_TOKEN_SIZE](#)
- #define [EMBER_KEY_TABLE_TOKEN_SIZE](#)
- #define [EMBER_REQUEST_KEY_TIMEOUT](#)
- #define [EMBER_TRANSIENT_KEY_TIMEOUT_S](#)
- #define [EMBER_END_DEVICE_BIND_TIMEOUT](#)
- #define [EMBER_PAN_ID_CONFLICT_REPORT_THRESHOLD](#)
- #define [EMBER_TASK_COUNT](#)
- #define [EMBER_MAX_SUPPORTED_NETWORKS](#)

- `#define EMBER_SUPPORTED_NETWORKS`
- `#define EMBER_ZLL_GROUP_ADDRESSES`
- `#define EMBER_ZLL_RSSI_THRESHOLD`
- `#define EMBER_RF4CE_PAIRING_TABLE_SIZE`
- `#define EMBER_RF4CE_PAIRING_TABLE_TOKEN_SIZE`
- `#define EMBER_RF4CE_PENDING_OUTGOING_PACKET_TABLE_SIZE`
- `#define EMBER_GP_PROXY_TABLE_SIZE`
- `#define EMBER_GP_PROXY_TABLE_TOKEN_SIZE`
- `#define EMBER_GP_SINK_TABLE_SIZE`
- `#define EMBER_GP_SINK_TABLE_TOKEN_SIZE`
- `#define EZSP_HOST_SOURCE_ROUTE_TABLE_SIZE`
- `#define EZSP_HOST_RX_POOL_SIZE`
- `#define EZSP_HOST_FORM_AND_JOIN_BUFFER_SIZE`

6.6.1 Detailed Description

All configurations have defaults, therefore many applications may not need to do anything special. However, you can override these defaults by creating a `CONFIGURATION_HEADER` and within this header, defining the appropriate macro to a different size. For example, to reduce the number of allocated packet buffers from 24 (the default) to 8:

```
#define EMBER_PACKET_BUFFER_COUNT 8
```

The convenience stubs provided in `hal/ember-configuration.c` can be overridden by defining the appropriate macro and providing the corresponding callback function. For example, an application with custom debug channel input must implement `emberDebugHandler()` to process it. Along with the function definition, the application should provide the following line in its `CONFIGURATION_HEADER`:

```
#define EMBER_APPLICATION_HAS_DEBUG_HANDLER
```

See [ember-configuration-defaults.h](#) for source code.

See [ezsp-host-configuration-defaults.h](#) for source code.

6.6.2 Macro Definition Documentation

6.6.2.1 `#define EMBER_API_MAJOR_VERSION`

The major version number of the Ember stack release that the application is built against.

Definition at line 58 of file [ember-configuration-defaults.h](#).

6.6.2.2 `#define EMBER_API_MINOR_VERSION`

The minor version number of the Ember stack release that the application is built against.

Definition at line 65 of file [ember-configuration-defaults.h](#).

6.6.2.3 #define EMBER_STACK_PROFILE

Specifies the stack profile. The default is Profile 0.

You can set this to Profile 1 (ZigBee) or Profile 2 (ZigBee Pro) in your application's configuration header (.h) file using:

```
#define EMBER_STACK_PROFILE 1
```

or

```
#define EMBER_STACK_PROFILE 2
```

Definition at line 81 of file [ember-configuration-defaults.h](#).

6.6.2.4 #define EMBER_MAX_END_DEVICE_CHILDREN

The maximum number of end device children that a router will support. For profile 0 the default value is 6, for profile 1 the value is 14.

Definition at line 98 of file [ember-configuration-defaults.h](#).

6.6.2.5 #define EMBER_SECURITY_LEVEL

The security level used for security at the MAC and network layers. The supported values are 0 (no security) and 5 (payload is encrypted and a four-byte MIC is used for authentication).

Definition at line 123 of file [ember-configuration-defaults.h](#).

6.6.2.6 #define EMBER_CHILD_TABLE_SIZE

The maximum number of children that a node may have.

For the tree stack this values defaults to the sum of [EMBER_MAX_END_DEVICE_CHILDREN](#) and `::EMBER_MAX_ROUTER_CHILDREN`. For the mesh stack this defaults to the value of [EMBER_MAX_END_DEVICE_CHILDREN](#). In the mesh stack router children are not stored in the child table.

Each child table entry requires 4 bytes of RAM and a 10 byte token.

Application definitions for [EMBER_CHILD_TABLE_SIZE](#) that are larger than the default value are ignored and the default value used instead.

Definition at line 152 of file [ember-configuration-defaults.h](#).

6.6.2.7 #define EMBER_KEY_TABLE_SIZE

The maximum number of link and master keys that a node can store, **not** including the Trust Center Link Key. The stack maintains special storage for the Trust Center Link Key.

For the Trust Center, this controls how many totally unique Trust Center Link Keys may be stored. The rest of the devices in the network will use a global or hashed link key.

For normal nodes, this controls the number of Application Link Keys it can store. The Trust Center Link Key is stored separately from this table.

Definition at line 169 of file [ember-configuration-defaults.h](#).

6.6.2.8 #define EMBER_CERTIFICATE_TABLE_SIZE

The number of entries for the field upgradeable certificate table. Normally certificates (such as SE certs) are stored in the runtime-unmodifiable MFG area. However for those devices wishing to add new certificates after manufacturing, they will have to use the normal token space. This defines the size of that table. For most devices 0 is appropriate since there is no need to change certificates in the field. For those wishing to field upgrade devices with new certificates, 1 is the correct size. Anything more is simply wasting SimEEPROM.

Definition at line 182 of file [ember-configuration-defaults.h](#).

6.6.2.9 #define EMBER_MAX_DEPTH

The maximum depth of the tree in ZigBee 2006. This implicitly determines the maximum diameter of the network ([EMBER_MAX_HOPS](#)) if that value is not overridden.

Definition at line 195 of file [ember-configuration-defaults.h](#).

6.6.2.10 #define EMBER_MAX_HOPS

The maximum number of hops for a message.

When the radius is not supplied by the Application (i.e. 0) or the stack is sending a message, then the default is two times the max depth ([EMBER_MAX_DEPTH](#)).

Definition at line 208 of file [ember-configuration-defaults.h](#).

6.6.2.11 #define EMBER_PACKET_BUFFER_COUNT

The number of Packet Buffers available to the Stack. The default is 75.

Each buffer requires 36 bytes of RAM (32 for the buffer itself plus 4 bytes of overhead).

Definition at line 218 of file [ember-configuration-defaults.h](#).

6.6.2.12 #define EMBER_MAX_NEIGHBOR_TABLE_SIZE

The maximum number of router neighbors the stack can keep track of.

A neighbor is a node within radio range. The maximum allowed value is 16. End device children are kept track of in the child table, not the neighbor table. The default is 16. Setting this value lower than 8 is not recommended.

Each neighbor table entry consumes 18 bytes of RAM (6 for the table itself and 12 bytes of security data).

Definition at line 232 of file [ember-configuration-defaults.h](#).

6.6.2.13 #define EMBER_NEIGHBOR_TABLE_SIZE

Definition at line 234 of file [ember-configuration-defaults.h](#).

6.6.2.14 #define EMBER_INDIRECT_TRANSMISSION_TIMEOUT

The maximum amount of time (in milliseconds) that the MAC will hold a message for indirect transmission to a child.

The default is 3000 milliseconds (3 sec). The maximum value is 30 seconds (30000 milliseconds). Larger values will cause rollover confusion.

Definition at line 244 of file [ember-configuration-defaults.h](#).

6.6.2.15 #define EMBER_MAX_INDIRECT_TRANSMISSION_TIMEOUT

Definition at line 246 of file [ember-configuration-defaults.h](#).

6.6.2.16 #define EMBER_SEND_MULTICASTS_TO_SLEEPY_ADDRESS

This defines the behavior for what address multicasts are sent to. The normal address is RxOnWhenIdle=true (0xFFFD). However setting this to true can change locally generated multicasts to be sent to the sleepy broadcast address (0xFFFF). Changing the default is NOT ZigBee Pro compliant and may not be interoperable.

Definition at line 259 of file [ember-configuration-defaults.h](#).

6.6.2.17 #define EMBER_END_DEVICE_POLL_TIMEOUT

The maximum amount of time, in units determined by [EMBER_END_DEVICE_POLL_TIMEOUT_SHIFT](#), that an [EMBER_END_DEVICE](#) or [EMBER_SLEEPY_END_DEVICE](#) can wait between polls. The timeout value in seconds is $\text{EMBER_END_DEVICE_POLL_TIMEOUT} \ll \text{EMBER_END_DEVICE_POLL_TIMEOUT_SHIFT}$. If no poll is heard within this time, then the parent removes the end device from its tables. Note: there is a separate [EMBER_MOBILE_NODE_POLL_TIMEOUT](#) for mobile end devices.

Using the default values of both [EMBER_END_DEVICE_POLL_TIMEOUT](#) and [EMBER_END_DEVICE_POLL_TIMEOUT_SHIFT](#) results in a timeout of 320 seconds, or just over five minutes. The maximum value for [EMBER_END_DEVICE_POLL_TIMEOUT](#) is 255.

Definition at line 278 of file [ember-configuration-defaults.h](#).

6.6.2.18 #define EMBER_END_DEVICE_POLL_TIMEOUT_SHIFT

The units used for timing out end devices on their parents. See [EMBER_END_DEVICE_POLL_TIMEOUT](#) for an explanation of how this value is used.

The default value of 6 means gives [EMBER_END_DEVICE_POLL_TIMEOUT](#) a default unit of 64 seconds, or approximately one minute. The maximum value for [EMBER_END_DEVICE_POLL_TIMEOUT_SHIFT](#) is 14.

Definition at line 289 of file [ember-configuration-defaults.h](#).

6.6.2.19 #define EMBER_MOBILE_NODE_POLL_TIMEOUT

The maximum amount of time (in quarter-seconds) that a mobile node can wait between polls. If no poll is heard within this timeout, then the parent removes the mobile node from

its tables. The default is 20 quarter seconds (5 seconds). The maximum is 255 quarter seconds.

Definition at line 299 of file [ember-configuration-defaults.h](#).

6.6.2.20 #define EMBER_APS_UNICAST_MESSAGE_COUNT

The maximum number of APS retried messages that the stack can be transmitting at any time. Here, "transmitting" means the time between the call to ::emberSendUnicast() and the subsequent callback to ::emberMessageSentHandler().

Note

A message will typically use one packet buffer for the message header and one or more packet buffers for the payload. The default is 10 messages.

Each APS retried message consumes 6 bytes of RAM, in addition to two or more packet buffers.

Definition at line 315 of file [ember-configuration-defaults.h](#).

6.6.2.21 #define EMBER_BINDING_TABLE_SIZE

The maximum number of bindings supported by the stack. The default is 0 bindings. Each binding consumes 2 bytes of RAM.

Definition at line 321 of file [ember-configuration-defaults.h](#).

6.6.2.22 #define EMBER_ADDRESS_TABLE_SIZE

The maximum number of EUI64<->network address associations that the stack can maintain. The default value is 8.

Address table entries are 10 bytes in size.

Definition at line 329 of file [ember-configuration-defaults.h](#).

6.6.2.23 #define EMBER_RESERVED_MOBILE_CHILD_ENTRIES

The number of child table entries reserved for use only by mobile nodes. The default value is 0.

The maximum number of non-mobile children for a parent is [EMBER_CHILD_TABLE_SIZE](#) - [EMBER_RESERVED_MOBILE_CHILD_ENTRIES](#).

Definition at line 339 of file [ember-configuration-defaults.h](#).

6.6.2.24 #define EMBER_ROUTE_TABLE_SIZE

The maximum number of destinations to which a node can route messages. This include both messages originating at this node and those relayed for others. The default value is 16.

Route table entries are 6 bytes in size.

Definition at line 352 of file [ember-configuration-defaults.h](#).

6.6.2.25 #define EMBER_DISCOVERY_TABLE_SIZE

The number of simultaneous route discoveries that a node will support.

Discovery table entries are 9 bytes in size.

Definition at line 368 of file [ember-configuration-defaults.h](#).

6.6.2.26 #define EMBER_MULTICAST_TABLE_SIZE

The maximum number of multicast groups that the device may be a member of. The default value is 8.

Multicast table entries are 3 bytes in size.

Definition at line 381 of file [ember-configuration-defaults.h](#).

6.6.2.27 #define EMBER_SOURCE_ROUTE_TABLE_SIZE

The maximum number of source route table entries supported by the utility code in `app/util/source-route.-c`. The maximum source route table size is 255 entries, since a one-byte index is used, and the index 0xFF is reserved. The default value is 32.

Source route table entries are 4 bytes in size.

Definition at line 391 of file [ember-configuration-defaults.h](#).

6.6.2.28 #define EMBER_DEFAULT_BROADCAST_TABLE_SIZE

The maximum number broadcasts during a single broadcast timeout period. The minimum and default value is 15 and can only be changed only on compatible Ember stacks. Be very careful when changing the broadcast table size as it effects timing of the broadcasts as well as number of possible broadcasts. Additionally, this value must be universal for all devices in the network otherwise a single router can overwhelm all its neighbors with more broadcasts than they can support. In general, this value should be left alone.

Broadcast table entries are 5 bytes in size.

Definition at line 414 of file [ember-configuration-defaults.h](#).

6.6.2.29 #define EMBER_BROADCAST_TABLE_SIZE

Definition at line 417 of file [ember-configuration-defaults.h](#).

6.6.2.30 #define EMBER_RETRY_QUEUE_SIZE

Definition at line 426 of file [ember-configuration-defaults.h](#).

6.6.2.31 #define EMBER_ASSERT_SERIAL_PORT

Settings to control if and where assert information will be printed.

The output can be suppressed by defining `EMBER_ASSERT_OUTPUT_DISABLED`. The serial port to which the output is sent can be changed by defining `EMBER_ASSERT_SERIAL_PORT` as the desired port.

The default is to have assert output on and sent to serial port 1.

Definition at line 442 of file `ember-configuration-defaults.h`.

6.6.2.32 `#define EMBER_MAXIMUM_ALARM_DATA_SIZE`

The absolute maximum number of payload bytes in an alarm message.

The three length bytes in `EMBER_UNICAST_ALARM_CLUSTER` messages do not count towards this limit.

`EMBER_MAXIMUM_ALARM_DATA_SIZE` is defined to be 16.

The maximum payload on any particular device is determined by the configuration parameters, `EMBER_BROADCAST_ALARM_DATA_SIZE` and `EMBER_UNICAST_ALARM_DATA_SIZE`, neither of which may be greater than `EMBER_MAXIMUM_ALARM_DATA_SIZE`.

Definition at line 458 of file `ember-configuration-defaults.h`.

6.6.2.33 `#define EMBER_BROADCAST_ALARM_DATA_SIZE`

The sizes of the broadcast and unicast alarm buffers in bytes.

Devices have a single broadcast alarm buffer. Routers have one unicast alarm buffer for each child table entry. The total RAM used for alarms is

```
EMBER_BROADCAST_ALARM_DATA_SIZE
+ (EMBER_UNICAST_ALARM_DATA_SIZE *
  EMBER_CHILD_TABLE_SIZE)
```

`EMBER_BROADCAST_ALARM_DATA_SIZE` is the size of the alarm broadcast buffer. Broadcast alarms whose length is larger will not be buffered or forwarded to sleepy end device children. This parameter must be in the inclusive range 0 ... `EMBER_MAXIMUM_ALARM_DATA_SIZE`. The default value is 0.

Definition at line 478 of file `ember-configuration-defaults.h`.

6.6.2.34 `#define EMBER_UNICAST_ALARM_DATA_SIZE`

The size of the unicast alarm buffers allocated for end device children.

Unicast alarms whose length is larger will not be buffered or forwarded to sleepy end device children. This parameter must be in the inclusive range 0 ... `EMBER_MAXIMUM_ALARM_DATA_SIZE`. The default value is 0.

Definition at line 492 of file `ember-configuration-defaults.h`.

6.6.2.35 `#define EMBER_FRAGMENT_DELAY_MS`

The time the stack will wait (in milliseconds) between sending blocks of a fragmented message. The default value is 0.

Definition at line 501 of file `ember-configuration-defaults.h`.

6.6.2.36 #define EMBER_FRAGMENT_MAX_WINDOW_SIZE

The maximum number of blocks of a fragmented message that can be sent in a single window is defined to be 8.

Definition at line 507 of file [ember-configuration-defaults.h](#).

6.6.2.37 #define EMBER_FRAGMENT_WINDOW_SIZE

The number of blocks of a fragmented message that can be sent in a single window. The maximum is [EMBER_FRAGMENT_MAX_WINDOW_SIZE](#). The default value is 1.

Definition at line 514 of file [ember-configuration-defaults.h](#).

6.6.2.38 #define EMBER_BINDING_TABLE_TOKEN_SIZE

Definition at line 520 of file [ember-configuration-defaults.h](#).

6.6.2.39 #define EMBER_CHILD_TABLE_TOKEN_SIZE

Definition at line 523 of file [ember-configuration-defaults.h](#).

6.6.2.40 #define EMBER_KEY_TABLE_TOKEN_SIZE

Definition at line 526 of file [ember-configuration-defaults.h](#).

6.6.2.41 #define EMBER_REQUEST_KEY_TIMEOUT

The length of time that the device will wait for an answer to its Application Key Request. For the Trust Center this is the time it will hold the first request and wait for a second matching request. If both arrive within this time period, the Trust Center will reply to both with the new key. If both requests are not received then the Trust Center will discard the request. The time is in minutes. The maximum time is 10 minutes. A value of 0 minutes indicates that the Trust Center will not buffer the request but instead respond immediately. Only 1 outstanding request is supported at a time.

The Zigbee Pro Compliant value is 0.

Definition at line 542 of file [ember-configuration-defaults.h](#).

6.6.2.42 #define EMBER_TRANSIENT_KEY_TIMEOUT_S

The length of time, in seconds, that a trust center will store a transient link key that a device can use to join its network. A transient key is added with a call to `emberAddTransientLinkKey`. After the transient key is added, it will be removed once this amount of time has passed. A joining device will not be able to use that key to join until it is added again on the trust center. The default value is 300 seconds, i.e., 5 minutes.

Definition at line 556 of file [ember-configuration-defaults.h](#).

6.6.2.43 #define EMBER_END_DEVICE_BIND_TIMEOUT

The time the coordinator will wait (in seconds) for a second end device bind request to arrive. The default value is 60.

Definition at line 563 of file [ember-configuration-defaults.h](#).

6.6.2.44 #define EMBER_PAN_ID_CONFLICT_REPORT_THRESHOLD

The number of PAN id conflict reports that must be received by the network manager within one minute to trigger a PAN id change. Very rarely, a corrupt beacon can pass the CRC check and trigger a false PAN id conflict. This is more likely to happen in very large dense networks. Setting this value to 2 or 3 dramatically reduces the chances of a spurious PAN id change. The maximum value is 63. The default value is 1.

Definition at line 575 of file [ember-configuration-defaults.h](#).

6.6.2.45 #define EMBER_TASK_COUNT

The number of event tasks that can be tracked for the purpose of processor idling. The Ember Zigbee Pro and Zigbee RF4CE stacks require 1 task each, an application and associated libraries may use additional tasks, though typically no more than 4 are needed for most applications.

Definition at line 584 of file [ember-configuration-defaults.h](#).

6.6.2.46 #define EMBER_MAX_SUPPORTED_NETWORKS

The number of networks supported by the stack.

Definition at line 589 of file [ember-configuration-defaults.h](#).

6.6.2.47 #define EMBER_SUPPORTED_NETWORKS

Definition at line 594 of file [ember-configuration-defaults.h](#).

6.6.2.48 #define EMBER_ZLL_GROUP_ADDRESSES

The number of unique group identifiers that this device requires.

Definition at line 601 of file [ember-configuration-defaults.h](#).

6.6.2.49 #define EMBER_ZLL_RSSI_THRESHOLD

The RSSI threshold applied to incoming scan requests.

Definition at line 607 of file [ember-configuration-defaults.h](#).

6.6.2.50 #define EMBER_RF4CE_PAIRING_TABLE_SIZE

The maximum number of pairings supported by the stack.

Definition at line 613 of file [ember-configuration-defaults.h](#).

6.6.2.51 #define EMBER_RF4CE_PAIRING_TABLE_TOKEN_SIZE

The maximum number of pairings stored in non-volatile memory.

Definition at line 619 of file [ember-configuration-defaults.h](#).

6.6.2.52 #define EMBER_RF4CE_PENDING_OUTGOING_PACKET_TABLE_SIZE

The maximum number of outgoing RF4CE packets supported by the stack.

Definition at line 625 of file [ember-configuration-defaults.h](#).

6.6.2.53 #define EMBER_GP_PROXY_TABLE_SIZE

The number of proxy table entries supported.

Definition at line 631 of file [ember-configuration-defaults.h](#).

6.6.2.54 #define EMBER_GP_PROXY_TABLE_TOKEN_SIZE

The maximum number of pairings stored in non-volatile memory.

Definition at line 638 of file [ember-configuration-defaults.h](#).

6.6.2.55 #define EMBER_GP_SINK_TABLE_SIZE

The number of sink table entries supported.

Definition at line 644 of file [ember-configuration-defaults.h](#).

6.6.2.56 #define EMBER_GP_SINK_TABLE_TOKEN_SIZE

The maximum number of pairings stored in non-volatile memory.

Definition at line 650 of file [ember-configuration-defaults.h](#).

6.6.2.57 #define EZSP_HOST_SOURCE_ROUTE_TABLE_SIZE

The size of the source route table on the EZSP host.

Note

This configuration value sets the size of the source route table on the host, not on the node. [EMBER_SOURCE_ROUTE_TABLE_SIZE](#) sets `::EZSP_CONFIG_SOURCE_ROUTE_TABLE_SIZE` if `ezsp-utils.c` is used, which sets the size of the source route table on the NCP.

Definition at line 32 of file [ezsp-host-configuration-defaults.h](#).

6.6.2.58 #define EZSP_HOST_RX_POOL_SIZE

Define the size of the receive buffer pool on the EZSP host.

The number of receive buffers does not need to be greater than the number of packet buffers available on the ncp, because this in turn is the maximum number of callbacks that could be received between commands. In reality a value of 20 is a generous allocation.

Definition at line 43 of file [ezsp-host-configuration-defaults.h](#).

6.6.2.59 #define EZSP_HOST_FORM_AND_JOIN_BUFFER_SIZE

The size of the buffer for caching data during scans.

The form and join host library uses a flat buffer to store channel energy, pan ids, and matching networks. The underlying data structure is an `uint16_t[]`, so the true storage size is twice this value. The library requires the buffer be at least 32 bytes, so the minimum size here is 16. A matching network requires 16 to 20 bytes, depending on struct padding.

Definition at line 55 of file [ezsp-host-configuration-defaults.h](#).

6.7 Hardware Abstraction Layer (HAL) API Reference

Modules

- [HAL Configuration](#)
- [Microcontroller General Functionality](#)
- [SPI Protocol](#)
- [System Timer](#)
- [Sample APIs for Peripheral Access](#)
- [HAL Utilities](#)

6.7.1 Detailed Description

STM32F103RET Host Microcontroller

HAL function names have the following prefix conventions:

halCommon: API that is used by the EmberZNet stack and can also be called from an application. This API must be implemented. Custom applications can change the implementation of the API but its functionality must remain the same.

hal: API that is used by sample applications. Custom applications can remove this API or change its implementation as they see fit.

halStack: API used only by the EmberZNet stack. This API must be implemented and should not be directly called from any application. Custom applications can change the implementation of the API, but its functionality must remain the same.

halInternal: API that is internal to the HAL. The EmberZNet stack and applications must never call this API directly. Custom applications can change this API as they see fit. However, be careful not to impact the functionality of any halStack or halCommon APIs.

See also [hal.h](#).

6.8 HAL Configuration

Modules

- [Common PLATFORM_HEADER Configuration](#)

6.8.1 Detailed Description

Configuration information that affects the entire HAL.

6.9 Common PLATFORM_HEADER Configuration

Modules

- [STM32F103RET IAR Specific PLATFORM_HEADER Configuration](#)

Macros

- #define [MEMSET](#)(d, v, l)
- #define [MEMCOPY](#)(d, s, l)
- #define [MEMMOVE](#)(d, s, l)
- #define [MEMCOMPARE](#)(s0, s1, l)
- #define [MEMPGMCOMPARE](#)(s0, s1, l)

Generic Types

- #define [TRUE](#)
- #define [FALSE](#)
- #define [NULL](#)

Bit Manipulation Macros

- #define [BIT](#)(x)
- #define [BIT32](#)(x)
- #define [SETBIT](#)(reg, bit)
- #define [SETBITS](#)(reg, bits)
- #define [CLEARBIT](#)(reg, bit)
- #define [CLEARBITS](#)(reg, bits)
- #define [READBIT](#)(reg, bit)
- #define [READBITS](#)(reg, bits)

Byte Manipulation Macros

- #define [LOW_BYTE](#)(n)
- #define [HIGH_BYTE](#)(n)
- #define [HIGH_LOW_TO_INT](#)(high, low)
- #define [BYTE_0](#)(n)
- #define [BYTE_1](#)(n)
- #define [BYTE_2](#)(n)
- #define [BYTE_3](#)(n)
- #define [COUNTOF](#)(a)

Time Manipulation Macros

- `#define elapsedTimeInt8u(oldTime, newTime)`
- `#define elapsedTimeInt16u(oldTime, newTime)`
- `#define elapsedTimeInt32u(oldTime, newTime)`
- `#define MAX_INT8U_VALUE`
- `#define HALF_MAX_INT8U_VALUE`
- `#define timeGTorEqualInt8u(t1, t2)`
- `#define MAX_INT16U_VALUE`
- `#define HALF_MAX_INT16U_VALUE`
- `#define timeGTorEqualInt16u(t1, t2)`
- `#define MAX_INT32U_VALUE`
- `#define HALF_MAX_INT32U_VALUE`
- `#define timeGTorEqualInt32u(t1, t2)`

Miscellaneous Macros

- `#define UNUSED_VAR(x)`
- `#define DEBUG_LEVEL`

6.9.1 Detailed Description

Compiler and Platform specific definitions and typedefs common to all platforms. Some definitions can be overridden by the specific PLATFORM_HEADER for your platform. [platform-common.h](#) provides PLATFORM_HEADER defaults and common definitions. This head should never be included directly, it should only be included by the specific PLATFORM_HEADER used by your platform.

See [platform-common.h](#) for source code.

6.9.2 Macro Definition Documentation

6.9.2.1 `#define MEMSET(d, v, l)`

Friendly convenience macro pointing to the C Stdlib functions.

Definition at line 165 of file [platform-common.h](#).

6.9.2.2 `#define MEMCOPY(d, s, l)`

Definition at line 166 of file [platform-common.h](#).

6.9.2.3 `#define MEMMOVE(d, s, l)`

Definition at line 167 of file [platform-common.h](#).

6.9.2.4 `#define MEMCOMPARE(s0, s1, l)`

Definition at line 168 of file [platform-common.h](#).

6.9.2.5 **#define MEMPGMCOMPARE(*s0*, *s1*, *l*)**

Definition at line 169 of file [platform-common.h](#).

6.9.2.6 **#define TRUE**

An alias for one, used for clarity.

Definition at line 193 of file [platform-common.h](#).

6.9.2.7 **#define FALSE**

An alias for zero, used for clarity.

Definition at line 198 of file [platform-common.h](#).

6.9.2.8 **#define NULL**

The null pointer.

Definition at line 204 of file [platform-common.h](#).

6.9.2.9 **#define BIT(*x*)**

Useful to reference a single bit of a byte.

Definition at line 218 of file [platform-common.h](#).

6.9.2.10 **#define BIT32(*x*)**

Useful to reference a single bit of an uint32_t type.

Definition at line 223 of file [platform-common.h](#).

6.9.2.11 **#define SETBIT(*reg*, *bit*)**

Sets *bit* in the *reg* register or byte.

Note

Assuming *reg* is an IO register, some platforms (such as the AVR) can implement this in a single atomic operation.

Definition at line 230 of file [platform-common.h](#).

6.9.2.12 **#define SETBITS(*reg*, *bits*)**

Sets the bits in the *reg* register or the byte as specified in the bitmask *bits*.

Note

This is never a single atomic operation.

Definition at line 237 of file [platform-common.h](#).

6.9.2.13 #define CLEARBIT(*reg*, *bit*)

Clears a bit in the *reg* register or byte.

Note

Assuming *reg* is an IO register, some platforms (such as the AVR) can implement this in a single atomic operation.

Definition at line 244 of file [platform-common.h](#).

6.9.2.14 #define CLEARBITS(*reg*, *bits*)

Clears the bits in the *reg* register or byte as specified in the bitmask *bits*.

Note

This is never a single atomic operation.

Definition at line 251 of file [platform-common.h](#).

6.9.2.15 #define READBIT(*reg*, *bit*)

Returns the value of *bit* within the register or byte *reg*.

Definition at line 256 of file [platform-common.h](#).

6.9.2.16 #define READBITS(*reg*, *bits*)

Returns the value of the bitmask *bits* within the register or byte *reg*.

Definition at line 262 of file [platform-common.h](#).

6.9.2.17 #define LOW_BYTE(*n*)

Returns the low byte of the 16-bit value *n* as an `uint8_t`.

Definition at line 276 of file [platform-common.h](#).

6.9.2.18 #define HIGH_BYTE(*n*)

Returns the high byte of the 16-bit value *n* as an `uint8_t`.

Definition at line 281 of file [platform-common.h](#).

6.9.2.19 #define HIGH_LOW_TO_INT(*high*, *low*)

Returns the value built from the two `uint8_t` values `high` and `low`.

Definition at line 287 of file [platform-common.h](#).

6.9.2.20 #define BYTE_0(*n*)

Returns the low byte of the 32-bit value `n` as an `uint8_t`.

Definition at line 295 of file [platform-common.h](#).

6.9.2.21 #define BYTE_1(*n*)

Returns the second byte of the 32-bit value `n` as an `uint8_t`.

Definition at line 300 of file [platform-common.h](#).

6.9.2.22 #define BYTE_2(*n*)

Returns the third byte of the 32-bit value `n` as an `uint8_t`.

Definition at line 305 of file [platform-common.h](#).

6.9.2.23 #define BYTE_3(*n*)

Returns the high byte of the 32-bit value `n` as an `uint8_t`.

Definition at line 310 of file [platform-common.h](#).

6.9.2.24 #define COUNTOF(*a*)

Returns the number of entries in an array.

Definition at line 315 of file [platform-common.h](#).

6.9.2.25 #define elapsedTimeInt8u(*oldTime*, *newTime*)

Returns the elapsed time between two 8 bit values. Result may not be valid if the time samples differ by more than 127.

Definition at line 330 of file [platform-common.h](#).

6.9.2.26 #define elapsedTimeInt16u(*oldTime*, *newTime*)

Returns the elapsed time between two 16 bit values. Result may not be valid if the time samples differ by more than 32767.

Definition at line 337 of file [platform-common.h](#).

6.9.2.27 #define elapsedTimeInt32u(*oldTime*, *newTime*)

Returns the elapsed time between two 32 bit values. Result may not be valid if the time samples differ by more than 2147483647.

Definition at line 344 of file [platform-common.h](#).

6.9.2.28 #define MAX_INT8U_VALUE

Returns true if t1 is greater than t2. Can only account for 1 wrap around of the variable before it is wrong.

Definition at line 351 of file [platform-common.h](#).

6.9.2.29 #define HALF_MAX_INT8U_VALUE

Returns the elapsed time between two 8 bit values. Result may not be valid if the time samples differ by more than 127.

Definition at line 352 of file [platform-common.h](#).

6.9.2.30 #define timeGTorEqualInt8u(*t1*, *t2*)

Returns the elapsed time between two 8 bit values. Result may not be valid if the time samples differ by more than 127.

Definition at line 353 of file [platform-common.h](#).

6.9.2.31 #define MAX_INT16U_VALUE

Returns true if t1 is greater than t2. Can only account for 1 wrap around of the variable before it is wrong.

Definition at line 360 of file [platform-common.h](#).

6.9.2.32 #define HALF_MAX_INT16U_VALUE

Returns the elapsed time between two 8 bit values. Result may not be valid if the time samples differ by more than 127.

Definition at line 361 of file [platform-common.h](#).

6.9.2.33 #define timeGTorEqualInt16u(*t1*, *t2*)

Returns the elapsed time between two 8 bit values. Result may not be valid if the time samples differ by more than 127.

Definition at line 362 of file [platform-common.h](#).

6.9.2.34 #define MAX_INT32U_VALUE

Returns true if t1 is greater than t2. Can only account for 1 wrap around of the variable before it is wrong.

Definition at line 369 of file [platform-common.h](#).

6.9.2.35 **#define HALF_MAX_INT32U_VALUE**

Returns the elapsed time between two 8 bit values. Result may not be valid if the time samples differ by more than 127.

Definition at line 370 of file [platform-common.h](#).

6.9.2.36 **#define timeGTorEqualInt32u(t1, t2)**

Returns the elapsed time between two 8 bit values. Result may not be valid if the time samples differ by more than 127.

Definition at line 371 of file [platform-common.h](#).

6.9.2.37 **#define UNUSED_VAR(x)**

Description:

Useful macro for avoiding compiler warnings related to unused function arguments or unused variables.

Definition at line 388 of file [platform-common.h](#).

6.9.2.38 **#define DEBUG_LEVEL**

Set debug level based on whether DEBUG or DEBUG_OFF are defined.

Definition at line 402 of file [platform-common.h](#).

6.10 STM32F103RET IAR Specific PLATFORM_HEADER Configuration

Macros

- #define [halResetWatchdog\(\)](#)
- #define [SIGNED_ENUM](#)
- #define [_HAL_USE_COMMON_DIVMOD_](#)
- #define [_HAL_USE_COMMON_PGM_](#)
- #define [PLATCOMMONOKTOINCLUDE](#)

Functions

- void [halInternalResetWatchDog](#) (void)

Master Variable Types

These are a set of typedefs to make the size of all variable declarations explicitly known. Since the IAR host code links against the ST Standard peripheral library, we need to map Ember's variable types to ST's variable types.

Note

ST uses IAR's variable types, found in stdint.h.

- typedef bool [boolean](#)
- typedef uint8_t [int8u](#)
- typedef int8_t [int8s](#)
- typedef uint16_t [int16u](#)
- typedef int16_t [int16s](#)
- typedef uint32_t [int32u](#)
- typedef int32_t [int32s](#)
- typedef uint32_t [PointerType](#)

Miscellaneous Macros

- void [halInternalAssertFailed](#) (const char *filename, int linenumber)
- #define [simulatedSerialTimePasses\(\)](#)
- #define [simulatedSerialTimePasses\(\)](#)
- #define [BIGENDIAN_CPU](#)
- #define [MAIN_FUNCTION_PARAMETERS](#)
- #define [MAIN_FUNCTION_ARGUMENTS](#)
- #define [__SOURCEFILE__](#)
- #define [assert](#)(condition)
- #define [simulatedTimePasses\(\)](#)
- #define [simulatedTimePassesMs](#)(x)
- #define [STRINGIZE](#)(X)
- #define [ALIGNMENT](#)(X)

Global Interrupt Manipulation Macros

- `#define DISABLE_INTERRUPTS\(\)`
- `#define RESTORE_INTERRUPTS\(\)`
- `#define INTERRUPTS_ON\(\)`
- `#define INTERRUPTS_OFF\(\)`
- `#define INTERRUPTS_ARE_OFF\(\)`
- `#define INTERRUPTS_WERE_ON\(\)`
- `#define ATOMIC\(blah\)`
- `#define HANDLE_PENDING_INTERRUPTS\(\)`

Generic Types

- `#define NULL`

6.10.1 Detailed Description

Compiler and Platform specific definitions and typedefs for the STM32F103RET Host built with the IAR ARM C compiler.

Note

[iar-st.h](#) should be included first in all source files by setting the preprocessor macro `PLATFORM_HEADER` to point to it. [iar-st.h](#) automatically includes [platform-common.h](#).

See [Common PLATFORM_HEADER Configuration](#) for common documentation.

See [iar-st.h](#) for source code.

6.10.2 Macro Definition Documentation

6.10.2.1 `#define halResetWatchdog()`

Macro to reset the watchdog timer.

Note

Be very very careful when using this as you can easily get into an infinite loop if you are not careful.

Definition at line [78](#) of file [iar-st.h](#).

6.10.2.2 `#define SIGNED_ENUM`

Some platforms need to cast enum values that have the high bit set.

Definition at line [84](#) of file [iar-st.h](#).

6.10.2.3 #define simulatedSerialTimePasses()

Stub for code not running in simulation.

Definition at line 169 of file [iar-st.h](#).

6.10.2.4 #define simulatedSerialTimePasses()

Stub for code not running in simulation.

Definition at line 169 of file [iar-st.h](#).

6.10.2.5 #define _HAL_USE_COMMON_DIVMOD_

Use the Divide and Modulus Operations from [platform-common.h](#).

Definition at line 95 of file [iar-st.h](#).

6.10.2.6 #define _HAL_USE_COMMON_PGM_

Use the Master Program Memory Declarations from [platform-common.h](#).

Definition at line 101 of file [iar-st.h](#).

6.10.2.7 #define BIGENDIAN_CPU

A convenient method for code to know what endiannes processor it is running on. For the Cortex-M3, we are little endian.

Definition at line 114 of file [iar-st.h](#).

6.10.2.8 #define MAIN_FUNCTION_PARAMETERS

Define the parameters to main(), and for those functions that are passed the arguments from main().

Definition at line 120 of file [iar-st.h](#).

6.10.2.9 #define MAIN_FUNCTION_ARGUMENTS

Stub for code not running in simulation.

Definition at line 121 of file [iar-st.h](#).

6.10.2.10 #define __SOURCEFILE__

The **SOURCEFILE** macro is used by asserts to list the filename if it isn't otherwise defined, set it to the compiler intrinsic which specifies the whole filename and path of the sourcefile.

Definition at line 130 of file [iar-st.h](#).

6.10.2.11 #define assert(*condition*)

A custom implementation of the C language assert macro. This macro implements the conditional evaluation and calls the function [halInternalAssertFailed\(\)](#).

Definition at line [146](#) of file [iar-st.h](#).

6.10.2.12 #define simulatedTimePasses()

Stub for code not running in simulation.

Definition at line [161](#) of file [iar-st.h](#).

6.10.2.13 #define simulatedTimePassesMs(*x*)

Stub for code not running in simulation.

Definition at line [165](#) of file [iar-st.h](#).

6.10.2.14 #define STRINGIZE(*X*)

Convenience macro for turning a token into a string.

Definition at line [174](#) of file [iar-st.h](#).

6.10.2.15 #define ALIGNMENT(*X*)

Provide a portable way to align data.

Definition at line [179](#) of file [iar-st.h](#).

6.10.2.16 #define DISABLE_INTERRUPTS()

Disable interrupts, saving the previous state so it can be later restored with [RESTORE_INTERRUPTS\(\)](#).

Note

Do not fail to call [RESTORE_INTERRUPTS\(\)](#).
It is safe to nest this call.

Definition at line [217](#) of file [iar-st.h](#).

6.10.2.17 #define RESTORE_INTERRUPTS()

Restore the global interrupt state previously saved by [DISABLE_INTERRUPTS\(\)](#)

Note

Do not call without having first called [DISABLE_INTERRUPTS\(\)](#) to have saved the state.
It is safe to nest this call.

Definition at line [231](#) of file [iar-st.h](#).

6.10.2.18 #define INTERRUPTS_ON()

Enable global interrupts without regard to the current or previous state.

Definition at line 241 of file [iar-st.h](#).

6.10.2.19 #define INTERRUPTS_OFF()

Disable global interrupts without regard to the current or previous state.

Definition at line 251 of file [iar-st.h](#).

6.10.2.20 #define INTERRUPTS_ARE_OFF()**Returns**

true if global interrupts are disabled.

Definition at line 260 of file [iar-st.h](#).

6.10.2.21 #define INTERRUPTS_WERE_ON()**Returns**

true if global interrupt flag was enabled when [DISABLE_INTERRUPTS\(\)](#) was called.

Definition at line 267 of file [iar-st.h](#).

6.10.2.22 #define ATOMIC(blah)

A block of code may be made atomic by wrapping it with this macro. Something which is atomic cannot be interrupted by interrupts.

Definition at line 274 of file [iar-st.h](#).

6.10.2.23 #define HANDLE_PENDING_INTERRUPTS()

Allows any pending interrupts to be executed. Usually this would be called at a safe point while interrupts are disabled (such as within an ISR).

Takes no action if interrupts are already enabled.

Definition at line 290 of file [iar-st.h](#).

6.10.2.24 #define NULL

The null pointer.

Definition at line 313 of file [iar-st.h](#).

6.10.2.25 `#define PLATCOMMONOKTOINCLUDE`

Include [platform-common.h](#) last to pick up defaults and common definitions.

Include `string.h` for the C Standard Library memory routines used in `platform-common`.

Definition at line 329 of file [iar-st.h](#).

6.10.3 Typedef Documentation

6.10.3.1 `typedef bool boolean`

A typedef to make the size of the variable explicitly known.

Definition at line 55 of file [iar-st.h](#).

6.10.3.2 `typedef uint8_t int8u`

A typedef to make the size of the variable explicitly known.

Definition at line 56 of file [iar-st.h](#).

6.10.3.3 `typedef int8_t int8s`

A typedef to make the size of the variable explicitly known.

Definition at line 57 of file [iar-st.h](#).

6.10.3.4 `typedef uint16_t int16u`

A typedef to make the size of the variable explicitly known.

Definition at line 58 of file [iar-st.h](#).

6.10.3.5 `typedef int16_t int16s`

A typedef to make the size of the variable explicitly known.

Definition at line 59 of file [iar-st.h](#).

6.10.3.6 `typedef uint32_t int32u`

A typedef to make the size of the variable explicitly known.

Definition at line 60 of file [iar-st.h](#).

6.10.3.7 `typedef int32_t int32s`

A typedef to make the size of the variable explicitly known.

Definition at line 61 of file [iar-st.h](#).

6.10.3.8 typedef uint32_t PointerType

A typedef to make the size of the variable explicitly known.

Definition at line 62 of file [iar-st.h](#).

6.10.4 Function Documentation

6.10.4.1 void halInternalResetWatchDog (void)

Internal function to reset the watchdog timer.

Note

Be very very careful when using this as you can easily get into an infinite loop if you are not careful.

6.10.4.2 void halInternalAssertFailed (const char * *filename*, int *linenumber*)

A prototype definition for use by the assert macro.

6.11 Microcontroller General Functionality

Modules

- [STM32F103RET General Functionality](#)
- [ST Microcontroller Standard Peripherals Library Inclusions and Definitions](#)

Macros

- `#define` [MICRO_DISABLE_WATCH_DOG_KEY](#)
- `#define` [MILLISECOND_TICKS_PER_SECOND](#)
- `#define` [MILLISECOND_TICKS_PER_DECISECOND](#)
- `#define` [MILLISECOND_TICKS_PER_QUARTERSECOND](#)
- `#define` [MILLISECOND_TICKS_PER_MINUTE](#)
- `#define` [MILLISECOND_TICKS_PER_HOUR](#)
- `#define` [MILLISECOND_TICKS_PER_DAY](#)

Enumerations

- `enum` [SleepModes](#) {
[SLEEPMODE_RUNNING](#), [SLEEPMODE_IDLE](#), [SLEEPMODE_WAKETIMER](#),
[SLEEPMODE_MAINTAINTIMER](#),
[SLEEPMODE_NOTIMER](#), [SLEEPMODE_RESERVED](#), [SLEEPMODE_POWERDOWN](#), [SLEEPMODE_POWERSAVE](#) }

Functions

- `void` [halInit](#) (void)
- `void` [halReboot](#) (void)
- `void` [halPowerUp](#) (void)
- `void` [halPowerDown](#) (void)
- `void` [halInternalEnableWatchDog](#) (void)
- `void` [halInternalDisableWatchDog](#) (uint8_t magicKey)
- `void` [halCommonDelayMicroseconds](#) (uint16_t us)
- `void` [halCommonDelayMilliseconds](#) (uint16_t ms)
- `uint8_t` [halGetResetInfo](#) (void)
- `PGM_P` [halGetResetString](#) (void)
- `void` [halStackSeedRandom](#) (uint32_t seed)
- `uint16_t` [halCommonGetRandom](#) (void)
- `void` [halSleep](#) ([SleepModes](#) sleepMode)

6.11.1 Detailed Description

HAL functions common across all microcontroller-specific files.

Note

The micro specific definitions, [STM32F103RET General Functionality](#), is chosen by the build include path pointing at the appropriate directory.

See [micro-common.h](#) for source code.

6.11.2 Macro Definition Documentation

6.11.2.1 `#define MICRO_DISABLE_WATCH_DOG_KEY`

The value that must be passed as the single parameter to [halInternalDisableWatchDog\(\)](#) in order to successfully disable the watchdog timer.

Definition at line 41 of file [micro-common.h](#).

6.11.2.2 `#define MILLISECOND_TICKS_PER_SECOND`

Definition at line 183 of file [micro-common.h](#).

6.11.2.3 `#define MILLISECOND_TICKS_PER_DECISECOND`

Definition at line 189 of file [micro-common.h](#).

6.11.2.4 `#define MILLISECOND_TICKS_PER_QUARTERSECOND`

Definition at line 193 of file [micro-common.h](#).

6.11.2.5 `#define MILLISECOND_TICKS_PER_MINUTE`

Definition at line 197 of file [micro-common.h](#).

6.11.2.6 `#define MILLISECOND_TICKS_PER_HOUR`

Definition at line 201 of file [micro-common.h](#).

6.11.2.7 `#define MILLISECOND_TICKS_PER_DAY`

Definition at line 205 of file [micro-common.h](#).

6.11.3 Enumeration Type Documentation

6.11.3.1 `enum SleepModes`

Enumerations for the possible microcontroller sleep modes.

NOTE: Refer to a specific micro's implementation of [halSleep\(\)](#) to see what modes are actually supported.

- `SLEEPMODE_RUNNING` Everything is active and running. In practice this mode is not used, but it is defined for completeness of information.
- `SLEEPMODE_IDLE` Only the CPU is idled. The rest of the chip continues running normally. The chip will wake from any interrupt.

- **SLEEPMODE_WAKETIMER** The sleep timer clock sources remain running. The RC is always running and the 32kHz XTAL depends on system timer config. Wakeup is possible from both GPIO and the sleep timer. System time is maintained. The sleep timer is assumed to be configured properly for wake events.
- **SLEEPMODE_MAINTAINTIMER** The sleep timer clock sources remain running. The RC is always running and the 32kHz XTAL depends on the board header. Wakeup is possible from only GPIO. System time is maintained.
- **SLEEPMODE_NOTIMER** The sleep timer clock sources (both RC and XTAL) are turned off. Wakeup is possible from only GPIO. System time is lost.
- **SLEEPMODE_RESERVED** Reserved/Unused
- **SLEEPMODE_POWERDOWN** Deprecated
- **SLEEPMODE_POWERSAVE** Deprecated

Enumerator:

SLEEPMODE_RUNNING
SLEEPMODE_IDLE
SLEEPMODE_WAKETIMER
SLEEPMODE_MAINTAINTIMER
SLEEPMODE_NOTIMER
SLEEPMODE_RESERVED
SLEEPMODE_POWERDOWN
SLEEPMODE_POWERSAVE

Definition at line 145 of file [micro-common.h](#).

6.11.4 Function Documentation

6.11.4.1 void halInit (void)

Initializes microcontroller-specific peripherals.

6.11.4.2 void halReboot (void)

Restarts the microcontroller.

6.11.4.3 void halPowerUp (void)

Powers up microcontroller peripherals.

6.11.4.4 void halPowerDown (void)

Powers down microcontroller peripherals.

6.11.4.5 void halInternalEnableWatchDog (void)

Enables the watchdog timer, if there is one and it is reasonable to be enabled.

6.11.4.6 void halInternalDisableWatchDog (uint8_t *magicKey*)

Disables the watchdog timer, if there is one and it can be disabled.

Note

To prevent the watchdog from being disabled accidentally, a magic key must be provided.

Parameters

<i>magicKey</i>	A value (MICRO_DISABLE_WATCH_DOG_KEY) that enables the function.
-----------------	--

6.11.4.7 void halCommonDelayMicroseconds (uint16_t *us*)

Blocks the current thread of execution for the specified amount of time, in microseconds.

The function is implemented with either cycle-counted busy loops or a convenient timer. It is intended to create the short blocking delays such as when interfacing with hardware peripherals.

The accuracy of the timing provided by this function is not specified, but a best faith effort is obtain an accurate delay. The implementation may be changed, but this function should be reasonably accurate.

Parameters

<i>us</i>	The specified time, in microseconds. Values should be between 1 and 65535 microseconds.
-----------	---

6.11.4.8 void halCommonDelayMilliseconds (uint16_t *ms*)

Blocks the current thread of execution for the specified amount of time, in milliseconds..

This function depends on [halCommonDelayMicroseconds\(\)](#).

Parameters

<i>ms</i>	The specified time, in milliseconds.
-----------	--------------------------------------

6.11.4.9 uint8_t halGetResetInfo (void)

Gets information about what caused the microcontroller to reset.

Returns

A code identifying the cause of the reset.

6.11.4.10 PGM_P halGetResetString (void)

Calls [halGetResetInfo\(\)](#) and supplies a string describing it.

Returns

A pointer to a program space string.

6.11.4.11 void halStackSeedRandom (uint32_t seed)

Seeds the [halCommonGetRandom\(\)](#) pseudorandom number generator.

Parameters

<i>seed</i>	A seed for the pseudorandom number generator.
-------------	---

6.11.4.12 uint16_t halCommonGetRandom (void)

Generate pseudorandom numbers. Implementation is host specific.

6.11.4.13 void halSleep (SleepModes sleepMode)

Puts the microcontroller to sleep in a specified mode.

Note

This routine always enables interrupts.

Parameters

<i>sleepMode</i>	A microcontroller sleep mode
------------------	------------------------------

6.12 STM32F103RET General Functionality

Macros

- `#define` [MILLISECOND_TICKS_PER_SECOND](#)

Functions

- `void` [halInternalInitSysTick](#) (`void`)
- `#define` [RESET_UNKNOWN](#)
- `#define` [RESET_LOW_POWER](#)
- `#define` [RESET_WINDOW_WATCHDOG](#)
- `#define` [RESET_INDEPENDENT_WATCHDOG](#)
- `#define` [RESET_SOFTWARE](#)
- `#define` [RESET_POR_PDR](#)
- `#define` [RESET_PIN](#)
- `#define` [RESET_UNSET](#)

6.12.1 Detailed Description

HAL functions specific to this micro. See [Microcontroller General Functionality](#) for common documentation.

The definitions in the micro specific header provide the necessary pieces to link the common functionality to a specific micro.

See [micro-specific.h](#) for source code.

6.12.2 Macro Definition Documentation

6.12.2.1 `#define` MILLISECOND_TICKS_PER_SECOND

The number of ticks specific to this host (as returned from [halCommonGetInt32uMillisecond-Tick](#)) that represent an actual second.

Definition at line 27 of file [micro-specific.h](#).

6.12.2.2 `#define` RESET_UNKNOWN

A name given to a reset event. The name is derived from the datasheet and the value is the index into the `resetString` structure.

Definition at line 35 of file [micro-specific.h](#).

6.12.2.3 `#define` RESET_LOW_POWER

A name given to a reset event. The name is derived from the datasheet and the value is the index into the `resetString` structure.

Definition at line 36 of file [micro-specific.h](#).

6.12.2.4 **#define RESET_WINDOW_WATCHDOG**

A name given to a reset event. The name is derived from the datasheet and the value is the index into the resetString structure.

Definition at line 37 of file [micro-specific.h](#).

6.12.2.5 **#define RESET_INDEPENDENT_WATCHDOG**

A name given to a reset event. The name is derived from the datasheet and the value is the index into the resetString structure.

Definition at line 38 of file [micro-specific.h](#).

6.12.2.6 **#define RESET_SOFTWARE**

A name given to a reset event. The name is derived from the datasheet and the value is the index into the resetString structure.

Definition at line 39 of file [micro-specific.h](#).

6.12.2.7 **#define RESET_POR_PDR**

A name given to a reset event. The name is derived from the datasheet and the value is the index into the resetString structure.

Definition at line 40 of file [micro-specific.h](#).

6.12.2.8 **#define RESET_PIN**

A name given to a reset event. The name is derived from the datasheet and the value is the index into the resetString structure.

Definition at line 41 of file [micro-specific.h](#).

6.12.2.9 **#define RESET_UNSET**

A name given to a reset event. The name is derived from the datasheet and the value is the index into the resetString structure.

Definition at line 42 of file [micro-specific.h](#).

6.12.3 **Function Documentation**

6.12.3.1 **void halInternalInitSysTick (void)**

Initialize the SysTick timer to provide a microsecond time base for use by [halCommon-DelayMicroseconds\(\)](#).

6.13 ST Microcontroller Standard Peripherals Library Inclusions and Definitions

Macros

- #define [assert_param](#)(condition)

Functions

- void [halInternalAssertFailed](#) (const char *filename, int linenumber)

6.13.1 Detailed Description

ST Microcontroller's Standard Peripherals Library inclusions and definitions. This file is included from ST's Standard Peripherals Library and includes the headers for the peripherals found in ST's Library. It also defines the assert macro used by ST's Library. The actual documentation for ST's Standard Peripherals Library is beyond the scope of Ember's documentation.

Note

While this file's name, [stm32f10x_conf.h](#), does not conform to Ember's file naming convention, this file is included from ST's Standard Peripherals Library. Not renaming this file means the library does not have to be modified.

See [Microcontroller General Functionality](#) for common documentation.

See [stm32f10x_conf.h](#) for source code.

6.13.2 Macro Definition Documentation

6.13.2.1 #define assert_param(*condition*)

The assert_param macro is used by ST's Library to check a function's parameters. This macro redirects to Ember's assert function. This macro redirect is the same definition of assert as used in the the PLATFORM_HEADER.

Definition at line [75](#) of file [stm32f10x_conf.h](#).

6.13.3 Function Documentation

6.13.3.1 void halInternalAssertFailed (const char * *filename*, int *linenumber*)

A prototype definition of the Ember assert function for use by the assert_param macro.

6.14 SPI Protocol

Modules

- [STM32F103RET Specific SPI Protocol](#)

Functions

- void [halNcpSerialInit](#) (void)
- void [halNcpSerialPowerup](#) (void)
- void [halNcpSerialPowerdown](#) (void)
- EzspStatus [halNcpHardReset](#) (void)
- EzspStatus [halNcpHardResetReqBootload](#) (bool requestBootload)
- void [halNcpWakeUp](#) (void)
- void [halNcpSendCommand](#) (void)
- void [halNcpSendRawCommand](#) (void)
- EzspStatus [halNcpPollForResponse](#) (void)
- void [halNcpIsAwakeIsr](#) (bool isAwake)
- bool [halNcpHasData](#) (void)
- bool [halNcpVerifySpiProtocolVersion](#) (void)
- bool [halNcpVerifySpiProtocolActive](#) (void)

Variables

- uint8_t * [halNcpFrame](#)
- uint8_t [halNcpSpiErrorByte](#)

6.14.1 Detailed Description

Example host common SPI Protocol implementation for interfacing with a NCP. For complete documentation of the SPI Protocol, refer to the NCP docs.

Note

The micro specific definitions, [STM32F103RET Specific SPI Protocol](#), is chosen by the build include path pointing at the appropriate directory.

See [spi-protocol-common.h](#) for source code.

6.14.2 Function Documentation

6.14.2.1 void [halNcpSerialInit](#) (void)

Initializes the SPI Protocol.

6.14.2.2 void [halNcpSerialPowerup](#) (void)

Reinitializes the SPI Protocol when coming out of sleep (powerdown).

6.14.2.3 void halNcpSerialPowerdown (void)

Shuts down the SPI Protocol when entering sleep (powerdown).

6.14.2.4 EzspStatus halNcpHardReset (void)

Forcefully resets the NCP by pulling on the nRESET line; waits for the NCP to boot; verifies that it has booted; verifies the NCP is active; verifies the SPI Protocol version. When this function returns, the NCP is ready to accept all commands.

This function is the same as [halNcpHardResetReqBootload\(\)](#), except that the NCP cannot be told to enter bootload mode through the nWAKE signal.

Returns

A EzspStatus value indicating the success or failure of the command.

6.14.2.5 EzspStatus halNcpHardResetReqBootload (bool requestBootload)

Forcefully resets the NCP by pulling on the nRESET line; sets the nWAKE signal based upon the state of the requestBootload bool; waits for the NCP to boot; verifies that it has booted; verifies the NCP is active; verifies the SPI Protocol version. When this function returns, the NCP is ready to accept all commands.

This function is the same as [halNcpHardReset\(\)](#), except that the ability to request the NCP enter bootload mode through the nWAKE signal is made available.

Returns

A EzspStatus value indicating the success or failure of the command.

6.14.2.6 void halNcpWakeUp (void)

If the Host thinks that the NCP is sleeping and wants to wake it up, the EZSP calls [halNcpWakeUp\(\)](#).

Waking up can take some time (milliseconds) so [halNcpWakeUp\(\)](#) returns immediately and the SPI Protocol calls [halNcpIsAwakeIsr\(\)](#) once the wakeup handshaking is complete and the NCP is ready to accept commands.

6.14.2.7 void halNcpSendCommand (void)

The EZSP writes a command into the command buffer and then calls [halNcpSendCommand\(\)](#).

This function assumes the command being sent is an EZSP frame and therefore sets the SPI Byte for an EZSP Frame. If sending a command other than EZSP, use [halNcpSendRawCommand\(\)](#). This function returns immediately after transmission of the Command has completed and the transaction has entered the Wait section. The EZSP must now call [halNcpPollForResponse\(\)](#) until the Response is received.

6.14.2.8 void halNcpSendRawCommand (void)

The upper layer writes a command into the command buffer and then calls [halNcpSendRawCommand\(\)](#).

This function makes no assumption about the data in the SpiBuffer, it will just faithfully try to perform the transaction. This function returns immediately after transmission of the Command has completed and the transaction has entered the Wait section. The upper layer must now call [halNcpPollForResponse\(\)](#) until the the Response is received.

6.14.2.9 EzspStatus halNcpPollForResponse (void)

After sending a Command with [halNcpSendCommand\(\)](#), the upper layer repeatedly calls this function until the SPI Protocol has finished reception of a Response.

Returns

A EzspStatus value indicating the success or failure of the command.

6.14.2.10 void halNcpIsAwakeIsr (bool *isAwake*)

The SPI Protocol calls [halNcpIsAwakeIsr\(\)](#) once the wakeup handshaking is complete and the NCP is ready to accept a command.

Parameters

<i>isAwake</i>	true if the wake handshake completed and the NCP is awake. false is the wake handshake failed and the NCP is unresponsive.
----------------	--

6.14.2.11 bool halNcpHasData (void)

If the Host wants to find out whether the NCP has a pending callback, the EZSP calls [halNcpHasData\(\)](#). If this function returns true then the EZSP will send a callback command.

6.14.2.12 bool halNcpVerifySpiProtocolVersion (void)

Transmits the SPI Protocol Version Command and checks the response against a literal value to verify the SPI Protocol version.

Returns

true if the SPI Protocol Version used in this function matches the version returned by the NCP. false is the versions do not match.

6.14.2.13 bool halNcpVerifySpiProtocolActive (void)

Transmits the SPI Status Command and checks the response against a literal value to verify the SPI Protocol is active.

Returns

true if the SPI Protocol is active. false if the SPI Protocol is not active.

6.14.3 Variable Documentation**6.14.3.1 uint8_t* halNcpFrame**

A pointer to the length byte at the start of the Payload. Upper layers will write the command to this location before starting a transaction. The upper layer will read the response from this location after a transaction completes. This pointer is the upper layers' primary access into the command/response buffer.

6.14.3.2 uint8_t halNcpSpipErrorByte

This error byte is the third byte found in a special SPI Protocol error case. It provides more detail concerning the error. Refer to the NCP docs for a more detailed description of this byte. The application does not need to work with this byte, but it can be useful information when developing.

6.15 STM32F103RET Specific SPI Protocol

SPI Protocol Interface

- `#define SPIP_nSSEL_PORT`
- `#define SPIP_nSSEL_PIN`
- `#define SPIP_MOSI_PORT`
- `#define SPIP_MOSI_PIN`
- `#define SPIP_MISO_PORT`
- `#define SPIP_MISO_PIN`
- `#define SPIP_SCLK_PORT`
- `#define SPIP_SCLK_PIN`
- `#define SPIP_nHOST_INT_PORT`
- `#define SPIP_nHOST_INT_PIN`
- `#define SPIP_nWAKE_PORT`
- `#define SPIP_nWAKE_PIN`
- `#define SPIP_nRESET_PORT`
- `#define SPIP_nRESET_PIN`

SPI Protocol timing parameters.

Note

Remember: TIM2 is configured to produce a 125us tick.

- `#define WAIT_SECTION_TIMEOUT`
- `#define WAKE_HANDSHAKE_TIMEOUT`
- `#define STARTUP_TIMEOUT`
- `#define INTER_COMMAND_SPACING`
- `#define NCP_RESET_DELAY`

6.15.1 Detailed Description

Example host specific SPI Protocol implementation for interfacing with a NCP. For complete documentation of the SPI Protocol, refer to the NCP docs.

See [SPI Protocol](#) for common documentation.

The definitions in the micro specific header provide the necessary pieces to link the common functionality to a specific micro.

See [spi-protocol-specific.h](#) for source code.

6.15.2 Macro Definition Documentation

6.15.2.1 `#define SPIP_nSSEL_PORT`

The actual port that nSSEL is connected to, PA4, which is configured as a general purpose output.

Definition at line 33 of file [spi-protocol-specific.h](#).

6.15.2.2 **#define SPIP_nSSEL_PIN**

The actual pin that nSSEL is connected to, PA4, which is configured as a general purpose output.

Definition at line 37 of file [spi-protocol-specific.h](#).

6.15.2.3 **#define SPIP_MOSI_PORT**

The actual port that MOSI is connected to, PA7, which is configured as alternate function push-pull.

Definition at line 43 of file [spi-protocol-specific.h](#).

6.15.2.4 **#define SPIP_MOSI_PIN**

The actual pin that MOSI is connected to, PA7, which is configured as alternate function push-pull.

Definition at line 47 of file [spi-protocol-specific.h](#).

6.15.2.5 **#define SPIP_MISO_PORT**

The actual port that MISO is connected to, PA6, which is configured as input with pull-up.

Definition at line 52 of file [spi-protocol-specific.h](#).

6.15.2.6 **#define SPIP_MISO_PIN**

The actual pin that MISO is connected to, PA6, which is configured as input with pull-up.

Definition at line 56 of file [spi-protocol-specific.h](#).

6.15.2.7 **#define SPIP_SCLK_PORT**

The actual port that SCLK is connected to, PA5, which is configured as alternate function push-pull.

Definition at line 61 of file [spi-protocol-specific.h](#).

6.15.2.8 **#define SPIP_SCLK_PIN**

The actual pin that SCLK is connected to, PA5, which is configured as alternate function push-pull.

Definition at line 65 of file [spi-protocol-specific.h](#).

6.15.2.9 **#define SPIP_nHOST_INT_PORT**

The actual port that nHOST_INT is connected to, PC4, which is configured as input with pull-up; EXTI4 interrupt, falling edge.

Definition at line 70 of file [spi-protocol-specific.h](#).

6.15.2.10 #define SPIP_nHOST_INT_PIN

The actual pin that nHOST_INT is connected to, PC4, which is configured as input with pull-up; EXTI4 interrupt, falling edge.

Definition at line 74 of file [spi-protocol-specific.h](#).

6.15.2.11 #define SPIP_nWAKE_PORT

The actual port that nWAKE is connected to, PC5, which is configured as general purpose output.

Definition at line 79 of file [spi-protocol-specific.h](#).

6.15.2.12 #define SPIP_nWAKE_PIN

The actual pin that nWAKE is connected to, PC5, which is configured as general purpose output.

Definition at line 83 of file [spi-protocol-specific.h](#).

6.15.2.13 #define SPIP_nRESET_PORT

The actual port that nRESET is connected to, PB0, which is configured as general purpose output.

Definition at line 88 of file [spi-protocol-specific.h](#).

6.15.2.14 #define SPIP_nRESET_PIN

The actual pin that nRESET is connected to, PB0, which is configured as general purpose output.

Definition at line 92 of file [spi-protocol-specific.h](#).

6.15.2.15 #define WAIT_SECTION_TIMEOUT

Wait section timeout is 300ms.

Definition at line 106 of file [spi-protocol-specific.h](#).

6.15.2.16 #define WAKE_HANDSHAKE_TIMEOUT

Wait handshake timeout is 300ms.

Definition at line 110 of file [spi-protocol-specific.h](#).

6.15.2.17 #define STARTUP_TIMEOUT

Startup timeout is 7500ms.

Definition at line 114 of file [spi-protocol-specific.h](#).

6.15.2.18 #define INTER_COMMAND_SPACING

Intercommand spacing is 1ms.

Definition at line 118 of file [spi-protocol-specific.h](#).

6.15.2.19 #define NCP_RESET_DELAY

The time to assert nRESET is 26 microseconds.

Definition at line 122 of file [spi-protocol-specific.h](#).

6.16 System Timer

Functions

- `uint16_t halInternalStartSystemTimer (void)`
- `uint16_t halCommonGetInt16uMillisecondTick (void)`
- `uint32_t halCommonGetInt32uMillisecondTick (void)`
- `uint16_t halCommonGetInt16uQuarterSecondTick (void)`
- `void halCommonSetSystemTime (uint32_t time)`

6.16.1 Detailed Description

Functions that provide access to the system timer. A single system tick (as returned by `halCommonGetInt16uMillisecondTick()` and `halCommonGetInt32uMillisecondTick()`) is approximately 1 millisecond.

Note

The actual time of a tick is specific to each micro.

A single quarter-second tick (as returned by `halCommonGetInt16uQuarterSecondTick()`) is approximately 0.25 seconds.

The values used by the time support functions will wrap after an interval. The length of the interval depends on the length of the tick and the number of bits in the value. However, there is no issue when comparing time deltas of less than half this interval with a subtraction, if all data types are the same.

See [system-timer.h](#) for source code.

6.16.2 Function Documentation

6.16.2.1 `uint16_t halInternalStartSystemTimer (void)`

Initializes the system tick.

Returns

Time to update the async registers after timer is started (units of 100 microseconds).

6.16.2.2 `uint16_t halCommonGetInt16uMillisecondTick (void)`

Returns the current system time in system ticks, as a 16-bit value.

Returns

The least significant 16 bits of the current system time, in system ticks.

6.16.2.3 `uint32_t halCommonGetInt32uMillisecondTick (void)`

Returns the current system time in system ticks, as a 32-bit value.

Returns

The least significant 32 bits of the current system time, in system ticks.

6.16.2.4 `uint16_t halCommonGetInt16uQuarterSecondTick (void)`

Returns the current system time in quarter second ticks, as a 16-bit value.

Returns

The least significant 16 bits of the current system time, in system ticks multiplied by 256.

6.16.2.5 `void halCommonSetSystemTime (uint32_t time)`

Set the current system time.

Parameters

<i>time</i>	A 32 bit value, expressed in milliseconds, that will become the current system time.
-------------	--

6.17 Sample APIs for Peripheral Access

Modules

- [Serial UART Communication](#)
- [ADC Control](#)
- [Button Control](#)
- [Buzzer Control](#)
- [LED Control](#)
- [Bootloader EEPROM Control](#)

6.17.1 Detailed Description

These are sample API for accessing peripherals and can be modified as needed for your applications.

6.18 Serial UART Communication

Modules

- [STM32F103RET Specific UART](#)

Enumerations

- enum [SerialBaudRate](#) {
[DEFINE_BAUD](#), [DEFINE_BAUD](#), [DEFINE_BAUD](#), [DEFINE_BAUD](#),
[DEFINE_BAUD](#), [DEFINE_BAUD](#), [DEFINE_BAUD](#), [DEFINE_BAUD](#),
[DEFINE_BAUD](#), [DEFINE_BAUD](#), [DEFINE_BAUD](#), [DEFINE_BAUD](#),
[DEFINE_BAUD](#), [DEFINE_BAUD](#), [DEFINE_BAUD](#), [DEFINE_BAUD](#),
[DEFINE_BAUD](#), [DEFINE_BAUD](#), [DEFINE_BAUD](#) }
- enum [NameOfType](#) { [DEFINE_PARITY](#), [DEFINE_PARITY](#), [DEFINE_PARITY](#) }

Serial HAL APIs

These functions must be implemented by the HAL in order for the serial code to operate. Only the higher-level serial code uses these functions, so they should not be called directly. The HAL should also implement the appropriate interrupt handlers to drain the TX queues and fill the RX FIFO queue, as necessary.

- [EmberStatus](#) [halInternalUartInit](#) (uint8_t port, [SerialBaudRate](#) rate, SerialParity parity, uint8_t stopBits)
- uint16_t [halInternalPrintfWriteAvailable](#) (void)
- uint16_t [halInternalPrintfReadAvailable](#) (void)
- void [halInternalForcePrintf](#) (bool onOff)

6.18.1 Detailed Description

This API contains the common HAL interfaces that hosts must implement for the high-level serial code. This header describes the interface between the high-level serial APIs in [app/util/serial/serial.h](#) and the low level UART implementation.

Some functions in this file return an [EmberStatus](#) value. See [error-def.h](#) for definitions of all [EmberStatus](#) return values.

See [serial.h](#) for source code.

6.18.2 Enumeration Type Documentation

6.18.2.1 enum SerialBaudRate

Assign numerical values for variables that hold Baud Rate parameters.

Enumerator:

DEFINE_BAUD
DEFINE_BAUD

```
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
```

Definition at line 33 of file [hal/host/serial.h](#).

6.18.2.2 enum NameOfType

Assign numerical values for the types of parity. Use for variables that hold Parity parameters.

Enumerator:

```
DEFINE_PARITY
DEFINE_PARITY
DEFINE_PARITY
```

Definition at line 69 of file [hal/host/serial.h](#).

6.18.3 Function Documentation

6.18.3.1 EmberStatus halInternalUartInit (uint8_t port, SerialBaudRate rate, SerialParity parity, uint8_t stopBits)

Initializes the UART to the given settings (same parameters as [emberSerialInit\(\)](#)).

Parameters

<i>port</i>	Serial port number (0 or 1).
<i>rate</i>	Baud rate (see SerialBaudRate).
<i>parity</i>	Parity value (see SerialParity).
<i>stopBits</i>	Number of stop bits.

Returns

An error code if initialization failed (such as invalid baud rate), otherwise EMBER_SUCCESS.

6.18.3.2 uint16_t halInternalPrintfWriteAvailable (void)

Returns the number bytes available in the transmit queue when using the EMBER_SERIAL_USE_STDIO variant of the Ember serial library.

Returns

Number of bytes available in the transmit queue.

6.18.3.3 uint16_t halInternalPrintfReadAvailable (void)

Returns the number bytes available in the receive queue when using the EMBER_SERIAL_USE_STDIO variant of the Ember serial library.

Returns

Number of bytes available in the receive queue.

6.18.3.4 void halInternalForcePrintf (bool *onOff*)

This function enables/disables EMBER_SERIAL_USE_STDIO printing behavior that is compatible with [emberSerialGuaranteedPrintf\(\)](#) and a replacement for halInternalForceWriteUartData(). (blocking, bypass queue, and polling)

6.19 STM32F103RET Specific UART

Macros

- `#define stdout`

Functions

- `size_t fflush (int handle)`

6.19.1 Detailed Description

STM32F102RET host uart driver operating on top of ST's Standard Peripheral Library; supporting IAR's standard library IO routines. See [Serial UART Communication](#) for common documentation.

See [uart.h](#) for source code.

6.19.2 Macro Definition Documentation

6.19.2.1 `#define stdout`

Define the stdout stream. Since we compile with `DLib_Config_Normal.h` it does not define 'stdout'. There is a low-level IO define '`_LLIO_STDOUT`' which is equivalent to stdout. Therefore, we define 'stdout' to be '`_LLIO_STDOUT`'.

Definition at line 41 of file [uart.h](#).

6.19.3 Function Documentation

6.19.3.1 `size_t fflush (int handle)`

Flush the output stream. `DLib_Config_Full.h` defines [fflush\(\)](#), but this library includes too much code so we compile with `DLib_Config_Normal.h` instead which does not define [fflush\(\)](#). Therefore, we manually define [fflush\(\)](#) in the low level UART driver. This function simply redirects to the `__write()` function with a NULL buffer, triggering a flush.

Parameters

<i>handle</i>	The output stream. Should be set to 'stdout' like normal.
---------------	---

Returns

Zero, indicating success.

6.20 ADC Control

Modules

- [STM32F103RET Specific ADC](#)

6.20.1 Detailed Description

There is no common ADC functionality, only micro specific functionality.

6.21 STM32F103RET Specific ADC

Macros

- `#define TEMP_SENSOR_PIN`
- `#define TEMP_SENSOR_PORT`
- `#define TEMP_SENSOR_ADC`
- `#define TEMP_SENSOR_ADC_CHAN`
- `#define TEMP_ENABLE_PIN`
- `#define TEMP_ENABLE_PORT`

Functions

- `void halInternalInitAdc (void)`
- `uint16_t halSampleAdc (void)`
- `int32_t halConvertValueToVolts (uint16_t value)`

6.21.1 Detailed Description

Example API functions for operating an ADC.

Note

On the STM32F103RET example host, this driver is written specifically to interact with the breakout board temp sensor.

See [adc.h](#) for source code.

6.21.2 Macro Definition Documentation

6.21.2.1 `#define TEMP_SENSOR_PIN`

The actual pin that the temp sensor is connected to.

Definition at line 22 of file [adc.h](#).

6.21.2.2 `#define TEMP_SENSOR_PORT`

The actual port that the temp sensor is connected to.

Definition at line 25 of file [adc.h](#).

6.21.2.3 `#define TEMP_SENSOR_ADC`

The actual ADC that the temp sensor is connected to.

Definition at line 28 of file [adc.h](#).

6.21.2.4 #define TEMP_SENSOR_ADC_CHAN

The actual ADC channel that the temp sensor is connected to.
Definition at line 31 of file [adc.h](#).

6.21.2.5 #define TEMP_ENABLE_PIN

The actual pin that the temp sensor enable is connected to.
Definition at line 35 of file [adc.h](#).

6.21.2.6 #define TEMP_ENABLE_PORT

The actual port that the temp sensor enable is connected to.
Definition at line 38 of file [adc.h](#).

6.21.3 Function Documentation

6.21.3.1 void halInternalInitAdc (void)

Initialize the ADC.

6.21.3.2 uint16_t halSampleAdc (void)

Take a raw reading of the ADC.

Note

This function is blocking.

Returns

The raw value read from the ADC.

6.21.3.3 int32_t halConvertValueToVolts (uint16_t value)

Convert the raw register value (the unaltered value taken directly from the ADC's data register) into a signed fixed point value with units 10^{-4} Volts.

Parameters

<i>value</i>	An uint16_t to be converted.
--------------	------------------------------

Returns

Volts as signed fixed point with units 10^{-4} Volts.

6.22 Button Control

Modules

- [STM32F103RET Specific Button](#)

Functions

- void [halInternalInitButton](#) (void)
- uint8_t [halButtonState](#) (uint8_t button)
- uint8_t [halButtonPinState](#) (uint8_t button)
- void [halButtonIsr](#) (uint8_t button, uint8_t state)

Button State Definitions

A set of numerical definitions for use with the button APIs indicating the state of a button.

- #define [BUTTON_PRESSED](#)
- #define [BUTTON_RELEASED](#)

6.22.1 Detailed Description

Sample generic API functions for using push-buttons.

Note

The micro specific definitions, [STM32F103RET Specific Button](#), is chosen by the build include path pointing at the appropriate directory.

See [button-common.h](#) for source code.

6.22.2 Macro Definition Documentation

6.22.2.1 #define BUTTON_PRESSED

Button state is pressed.

Definition at line 29 of file [button-common.h](#).

6.22.2.2 #define BUTTON_RELEASED

Button state is released.

Definition at line 33 of file [button-common.h](#).

6.22.3 Function Documentation

6.22.3.1 void halInternalInitButton (void)

Initializes the buttons. Must be called before the buttons can be used.

6.22.3.2 `uint8_t halButtonState (uint8_t button)`

Returns the current state (pressed or released) of a button.

Note

This function is correlated with `halButtonIsr()` and so returns the shadow state rather than reading the actual state of the pin.

Parameters

<i>button</i>	The button being queried, either BUTTON0 or BUTTON1 as defined in button-specific.h .
---------------	---

Returns

`BUTTON_PRESSED` if the button is pressed or `BUTTON_RELEASED` if the button is not pressed.

6.22.3.3 `uint8_t halButtonPinState (uint8_t button)`

Returns the current state (pressed or released) of the pin associated with a button.

This reads the actual state of the pin and can be used on startup to determine the initial position of the buttons.

Parameters

<i>button</i>	The button being queried, either BUTTON0 or BUTTON1 as defined in button-specific.h .
---------------	---

Returns

`BUTTON_PRESSED` if the button is pressed or `BUTTON_RELEASED` if the button is not pressed.

6.22.3.4 `void halButtonIsr (uint8_t button, uint8_t state)`

A callback called in interrupt context whenever a button changes its state.

Application Usage:

Must be implemented by the application. This function should contain the functionality to be executed in response to changes of state in each of the buttons, or callbacks to the appropriate functionality.

Parameters

<i>button</i>	The button which has changed state, either BUTTON0 or BUTTON1 as defined in button-specific.h .
<i>state</i>	The new state of the button referenced by the button parameter, either <code>BUTTON_PRESSED</code> if the button has been pressed or <code>BUTTON_RELEASED</code> if the button has been released.

6.23 STM32F103RET Specific Button

Macros

- `#define BUTTON0`
- `#define BUTTON0_PIN`
- `#define BUTTON0_PORT`
- `#define BUTTON0_EXTI_SOURCE_PORT`
- `#define BUTTON0_EXTI_SOURCE_PIN`
- `#define BUTTON0_IRQ`
- `#define BUTTON1`
- `#define BUTTON1_PIN`
- `#define BUTTON1_PORT`
- `#define BUTTON1_EXTI_SOURCE_PORT`
- `#define BUTTON1_EXTI_SOURCE_PIN`
- `#define BUTTON1_IRQ`
- `#define BUTTON01_ISR`

6.23.1 Detailed Description

Sample micro specific API funtions and defines for using push-buttons. See [Button Control](#) for common documentation.

The definitions in the micro specific header provide the necessary pieces to link the common functionality to a specific micro.

See [button-specific.h](#) for source code.

6.23.2 Macro Definition Documentation

6.23.2.1 `#define BUTTON0`

Simple numerical definition of BUTTON0.

Definition at line 26 of file [button-specific.h](#).

6.23.2.2 `#define BUTTON0_PIN`

The actual pin that BUTTON0 is connected to.

Definition at line 30 of file [button-specific.h](#).

6.23.2.3 `#define BUTTON0_PORT`

The actual port that BUTTON0 is connected to.

Definition at line 34 of file [button-specific.h](#).

6.23.2.4 `#define BUTTON0_EXTI_SOURCE_PORT`

The actual source port that BUTTON0 is connected to for external interrupts.

Definition at line 39 of file [button-specific.h](#).

6.23.2.5 #define BUTTON0_EXTI_SOURCE_PIN

The actual source pin that BUTTON0 is connected to for external interrupts.

Definition at line 44 of file [button-specific.h](#).

6.23.2.6 #define BUTTON0_IRQ

The actual external interrupt IRQ number for BUTTON0.

Definition at line 48 of file [button-specific.h](#).

6.23.2.7 #define BUTTON1

Simple numerical definition of BUTTON1.

Definition at line 53 of file [button-specific.h](#).

6.23.2.8 #define BUTTON1_PIN

The actual pin that BUTTON1 is connected to.

Definition at line 57 of file [button-specific.h](#).

6.23.2.9 #define BUTTON1_PORT

The actual port that BUTTON1 is connected to.

Definition at line 61 of file [button-specific.h](#).

6.23.2.10 #define BUTTON1_EXTI_SOURCE_PORT

The actual source port that BUTTON1 is connected to for external interrupts.

Definition at line 66 of file [button-specific.h](#).

6.23.2.11 #define BUTTON1_EXTI_SOURCE_PIN

The actual source pin that BUTTON1 is connected to for external interrupts.

Definition at line 71 of file [button-specific.h](#).

6.23.2.12 #define BUTTON1_IRQ

The actual external interrupt IRQ number for BUTTON1.

Definition at line 75 of file [button-specific.h](#).

6.23.2.13 #define BUTTON01_ISR

The actual external interrupt ISR handler. Due to the choice of GPIO, BUTTON0 and BUTTON1 share the same ISR handler.

Definition at line 81 of file [button-specific.h](#).

6.24 Buzzer Control

Modules

- [STM32F103RET Specific Buzzer](#)

6.24.1 Detailed Description

There is no common buzzer functionality, only micro specific functionality.

6.25 STM32F103RET Specific Buzzer

Functions

- void [halPlayTune_P](#) (uint8_t PGM *tune, bool bkg)
- void [halStartBuzzerTone](#) (uint16_t frequency)
- void [halStopBuzzerTone](#) (void)

Variables

- uint8_t PGM [hereIamTune](#) []

Note Definitions

Flats are used instead of sharps because # is a special character.

- #define [NOTE_C3](#)
- #define [NOTE_Db3](#)
- #define [NOTE_D3](#)
- #define [NOTE_Eb3](#)
- #define [NOTE_E3](#)
- #define [NOTE_F3](#)
- #define [NOTE_Gb3](#)
- #define [NOTE_G3](#)
- #define [NOTE_Ab3](#)
- #define [NOTE_A3](#)
- #define [NOTE_Bb3](#)
- #define [NOTE_B3](#)
- #define [NOTE_C4](#)
- #define [NOTE_Db4](#)
- #define [NOTE_D4](#)
- #define [NOTE_Eb4](#)
- #define [NOTE_E4](#)
- #define [NOTE_F4](#)
- #define [NOTE_Gb4](#)
- #define [NOTE_G4](#)
- #define [NOTE_Ab4](#)
- #define [NOTE_A4](#)
- #define [NOTE_Bb4](#)
- #define [NOTE_B4](#)
- #define [NOTE_C5](#)
- #define [NOTE_Db5](#)
- #define [NOTE_D5](#)
- #define [NOTE_Eb5](#)
- #define [NOTE_E5](#)
- #define [NOTE_F5](#)
- #define [NOTE_Gb5](#)
- #define [NOTE_G5](#)
- #define [NOTE_Ab5](#)
- #define [NOTE_A5](#)
- #define [NOTE_Bb5](#)
- #define [NOTE_B5](#)

6.25.1 Detailed Description

Example API functions for operating a piezo buzzer.

Note

On the STM32F103RET example host, the buzzer is tied to GPIO PC6 using TIM3 Channel 1.

See [buzzer.h](#) for source code.

6.25.2 Macro Definition Documentation

6.25.2.1 #define NOTE_C3

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 32 of file [buzzer.h](#).

6.25.2.2 #define NOTE_Db3

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 33 of file [buzzer.h](#).

6.25.2.3 #define NOTE_D3

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 34 of file [buzzer.h](#).

6.25.2.4 #define NOTE_Eb3

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 35 of file [buzzer.h](#).

6.25.2.5 #define NOTE_E3

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 36 of file [buzzer.h](#).

6.25.2.6 **#define NOTE_F3**

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 37 of file [buzzer.h](#).

6.25.2.7 **#define NOTE_Gb3**

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 38 of file [buzzer.h](#).

6.25.2.8 **#define NOTE_G3**

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 39 of file [buzzer.h](#).

6.25.2.9 **#define NOTE_Ab3**

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 40 of file [buzzer.h](#).

6.25.2.10 **#define NOTE_A3**

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 41 of file [buzzer.h](#).

6.25.2.11 **#define NOTE_Bb3**

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 42 of file [buzzer.h](#).

6.25.2.12 **#define NOTE_B3**

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 43 of file [buzzer.h](#).

6.25.2.13 **#define NOTE_C4**

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 44 of file [buzzer.h](#).

6.25.2.14 **#define NOTE_Db4**

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 45 of file [buzzer.h](#).

6.25.2.15 **#define NOTE_D4**

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 46 of file [buzzer.h](#).

6.25.2.16 **#define NOTE_Eb4**

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 47 of file [buzzer.h](#).

6.25.2.17 **#define NOTE_E4**

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 48 of file [buzzer.h](#).

6.25.2.18 **#define NOTE_F4**

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 49 of file [buzzer.h](#).

6.25.2.19 **#define NOTE_Gb4**

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 50 of file [buzzer.h](#).

6.25.2.20 **#define NOTE_G4**

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 51 of file [buzzer.h](#).

6.25.2.21 **#define NOTE_Ab4**

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 52 of file [buzzer.h](#).

6.25.2.22 **#define NOTE_A4**

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 53 of file [buzzer.h](#).

6.25.2.23 **#define NOTE_Bb4**

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 54 of file [buzzer.h](#).

6.25.2.24 **#define NOTE_B4**

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 55 of file [buzzer.h](#).

6.25.2.25 **#define NOTE_C5**

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 56 of file [buzzer.h](#).

6.25.2.26 #define NOTE_Db5

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 57 of file [buzzer.h](#).

6.25.2.27 #define NOTE_D5

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 58 of file [buzzer.h](#).

6.25.2.28 #define NOTE_Eb5

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 59 of file [buzzer.h](#).

6.25.2.29 #define NOTE_E5

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 60 of file [buzzer.h](#).

6.25.2.30 #define NOTE_F5

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 61 of file [buzzer.h](#).

6.25.2.31 #define NOTE_Gb5

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 62 of file [buzzer.h](#).

6.25.2.32 #define NOTE_G5

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 63 of file [buzzer.h](#).

6.25.2.33 #define NOTE_Ab5

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 64 of file [buzzer.h](#).

6.25.2.34 #define NOTE_A5

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 65 of file [buzzer.h](#).

6.25.2.35 #define NOTE_Bb5

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 66 of file [buzzer.h](#).

6.25.2.36 #define NOTE_B5

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line 67 of file [buzzer.h](#).

6.25.3 Function Documentation

6.25.3.1 void halPlayTune_P (uint8_t PGM * *tune*, bool *bkg*)

Plays a tune on the piezo buzzer.

The tune is played in the background if *bkg* is true. Otherwise, the API blocks until the playback of the tune is complete.

Parameters

<i>tune</i>	A pointer to tune to play.
<i>bkg</i>	Determines whether the tune plays in the background. If true, tune plays in background; if false, tune plays in foreground.

A tune is implemented as follows:

```
uint8_t PGM hereIamTune[] = { //All tunes are stored in flash.
    NOTE_B4, 1, //Plays the note B4 for 100 milliseconds.
    0, 1, //Pause for 100 milliseconds.
    NOTE_B4, 1, //Plays the note B4 for 100 milliseconds.
    0, 1, //Pause for 100 milliseconds.
    NOTE_B4, 1, //Plays the note B4 for 100 milliseconds.
    0, 1, //Pause for 100 milliseconds.
    NOTE_B5, 5, //Plays the note B5 for 500 milliseconds.
    0, 0 //NULL terminates the tune.
};
```

6.25.3.2 void halStartBuzzerTone (uint16_t frequency)

Plays a tone on the piezo buzzer. The tone will play continuously until [halStopBuzzerTone\(\)](#) is called.

Parameters

<i>frequency</i>	The frequency of the tone to play.
------------------	------------------------------------

6.25.3.3 void halStopBuzzerTone (void)

Stops playing a tone that was started by [halStartBuzzerTone\(\)](#).

6.25.4 Variable Documentation

6.25.4.1 uint8_t PGM hereIamTune[]

Extern definition of Ember’s traditional little "here I am" announcement tune, which lives in the buzzer module.

6.26 LED Control

Modules

- [STM32F103RET Specific LED](#)

Typedefs

- typedef enum [HalBoardLedPins](#) [HalBoardLed](#)

Functions

- void [halInternalInitLed](#) (void)
- void [halToggleLed](#) ([HalBoardLed](#) led)
- void [halSetLed](#) ([HalBoardLed](#) led)
- void [halClearLed](#) ([HalBoardLed](#) led)

6.26.1 Detailed Description

Sample generic API funtions for controlling LEDs. When specifying an LED to use, always use the BOARDLEDx definitions that are defined in the HalBoardLedPins enum in the micro specific led header.

Note

The micro specific definitions, [STM32F103RET Specific LED](#), is chosen by the build include path pointing at the appropriate directoy.

See [led-common.h](#) for source code.

6.26.2 Typedef Documentation

6.26.2.1 typedef enum HalBoardLedPins HalBoardLed

Ensures that the definitions for the LEDs are always used as parameters to the LED functions.

Note

Even though many compilers will use 16 bits for an enum instead of 8, we choose to use an enum here. The possible compiler inefficiency does not affect stack-based parameters and local variables, which is the general case for led paramters.

Definition at line [37](#) of file [led-common.h](#).

6.26.3 Function Documentation

6.26.3.1 void halInternalInitLed (void)

Configures GPIOs pertaining to the control of LEDs.

6.26.3.2 void halToggleLed (HalBoardLed led)

Atomically wraps an XOR or similar operation for a single GPIO pin attached to an LED.

Parameters

<i>led</i>	Identifier for the LED to be toggled.
------------	---------------------------------------

6.26.3.3 void halSetLed (HalBoardLed led)

Turns on (sets) a GPIO pin connected to an LED so that the LED turns on.

Parameters

<i>led</i>	Identifier for the LED to turn on.
------------	------------------------------------

6.26.3.4 void halClearLed (HalBoardLed led)

Turns off (clears) a GPIO pin connected to an LED, which turns off the LED.

Parameters

<i>led</i>	Identifier for the LED to turn off.
------------	-------------------------------------

6.27 STM32F103RET Specific LED

Macros

- `#define BOARDLED0_PIN`
- `#define BOARDLED0_PORT`
- `#define BOARDLED1_PIN`
- `#define BOARDLED1_PORT`

Enumerations

- `enum HalBoardLedPins { BOARDLED0, BOARDLED1, BOARD_ACTIVITY_LED, BOARD_HEARTBEAT_LED }`

6.27.1 Detailed Description

Sample micro specific API functions and defines for controlling LEDs. See [LED Control](#) for common documentation.

The definitions in the micro specific header provide the necessary pieces to link the common functionality to a specific micro.

See [led-specific.h](#) for source code.

6.27.2 Macro Definition Documentation

6.27.2.1 `#define BOARDLED0_PIN`

The actual pin that BOARDLED0 is connected to.

Definition at line 39 of file [led-specific.h](#).

6.27.2.2 `#define BOARDLED0_PORT`

The actual port that BOARDLED0 is connected to.

Definition at line 44 of file [led-specific.h](#).

6.27.2.3 `#define BOARDLED1_PIN`

The actual pin that BOARDLED1 is connected to.

Definition at line 50 of file [led-specific.h](#).

6.27.2.4 `#define BOARDLED1_PORT`

The actual port that BOARDLED1 is connected to.

Definition at line 55 of file [led-specific.h](#).

6.27.3 Enumeration Type Documentation

6.27.3.1 enum HalBoardLedPins

Assign each LED to a convenient name that is a simple identifier. BOARD_ACTIVITY_LED and BOARD_HEARTBEAT_LED provide a further layer of abstraction on top of the LEDs for verbose coding.

Enumerator:

BOARDLED0

BOARDLED1

BOARD_ACTIVITY_LED

BOARD_HEARTBEAT_LED

Definition at line 28 of file [led-specific.h](#).

6.28 Bootloader EEPROM Control

Data Structures

- struct [HalEepromInformationType](#)

This structure defines a variety of information about the attached external EEPROM device.

Macros

- #define [EEPROM_PAGE_SIZE](#)
- #define [EEPROM_FIRST_PAGE](#)
- #define [EEPROM_IMAGE_START](#)
- #define [EEPROM_SUCCESS](#)
- #define [EEPROM_ERR](#)
- #define [EEPROM_ERR_MASK](#)
- #define [EEPROM_ERR_PG_BOUNDARY](#)
- #define [EEPROM_ERR_PG_SZ](#)
- #define [EEPROM_ERR_WRT_DATA](#)
- #define [EEPROM_ERR_IMG_SZ](#)
- #define [EEPROM_ERR_ADDR](#)
- #define [EEPROM_ERR_INVALID_CHIP](#)
- #define [EEPROM_ERR_ERASE_REQUIRED](#)
- #define [EEPROM_ERR_NO_ERASE_SUPPORT](#)

EEPROM interaction functions.

- uint8_t [halEepromInit](#) (void)
- void [halEepromShutdown](#) (void)
- const [HalEepromInformationType](#) * [halEepromInfo](#) (void)
- bool [halEepromBusy](#) (void)
- uint8_t [halEepromRead](#) (uint32_t address, uint8_t *data, uint16_t len)
- uint8_t [halEepromWrite](#) (uint32_t address, const uint8_t *data, uint16_t len)
- uint8_t [halEepromErase](#) (uint32_t address, uint32_t totalLength)
- #define [EEPROM_INFO_VERSION](#)
- #define [EEPROM_INFO_MAJOR_VERSION](#)
- #define [EEPROM_INFO_MAJOR_VERSION_MASK](#)
- #define [EEPROM_INFO_MIN_VERSION_WITH_WORD_SIZE_SUPPORT](#)
- #define [EEPROM_CAPABILITIES_ERASE_SUPPORTED](#)
- #define [EEPROM_CAPABILITIES_PAGE_ERASE_REQD](#)

6.28.1 Detailed Description

Functions and definitions for generic EEPROM operation. Changing EEPROM size will change the size of the application image space without changing the size or relative location of the recovery and reserved sections. See `eeprom.c` for more information on modifying EEPROM functionality.

See [bootloader-eeprom.h](#) for source code.

6.28.2 Macro Definition Documentation

6.28.2.1 #define EEPROM_PAGE_SIZE

Definition of an EEPROM page size, in bytes. This definition is deprecated, and should no longer be used.

Definition at line 23 of file [bootloader-eprom.h](#).

6.28.2.2 #define EEPROM_FIRST_PAGE

Define the location of the first page in EEPROM. This definition is deprecated, and should no longer be used.

Definition at line 28 of file [bootloader-eprom.h](#).

6.28.2.3 #define EEPROM_IMAGE_START

Define the location of the image start in EEPROM as a function of the [EEPROM_FIRST_PAGE](#) and [EEPROM_PAGE_SIZE](#). This definition is deprecated, and should no longer be used.

Definition at line 34 of file [bootloader-eprom.h](#).

6.28.2.4 #define EEPROM_SUCCESS

Define EEPROM success status.

Definition at line 38 of file [bootloader-eprom.h](#).

6.28.2.5 #define EEPROM_ERR

Define EEPROM error status.

Definition at line 42 of file [bootloader-eprom.h](#).

6.28.2.6 #define EEPROM_ERR_MASK

Define EEPROM error mask.

Definition at line 46 of file [bootloader-eprom.h](#).

6.28.2.7 #define EEPROM_ERR_PG_BOUNDARY

Define EEPROM page boundary error.

Definition at line 50 of file [bootloader-eprom.h](#).

6.28.2.8 #define EEPROM_ERR_PG_SZ

Define EEPROM page size error.

Definition at line 54 of file [bootloader-eprom.h](#).

6.28.2.9 #define EEPROM_ERR_WRT_DATA

Define EEPROM write data error.

Definition at line 58 of file [bootloader-eprom.h](#).

6.28.2.10 #define EEPROM_ERR_IMG_SZ

Define EEPROM image too large error.

Definition at line 62 of file [bootloader-eprom.h](#).

6.28.2.11 #define EEPROM_ERR_ADDR

Define EEPROM invalid address error.

Definition at line 66 of file [bootloader-eprom.h](#).

6.28.2.12 #define EEPROM_ERR_INVALID_CHIP

Define EEPROM chip initialization error.

Definition at line 70 of file [bootloader-eprom.h](#).

6.28.2.13 #define EEPROM_ERR_ERASE_REQUIRED

Define EEPROM erase required error.

Definition at line 74 of file [bootloader-eprom.h](#).

6.28.2.14 #define EEPROM_ERR_NO_ERASE_SUPPORT

Define EEPROM error for no erase support.

Definition at line 78 of file [bootloader-eprom.h](#).

6.28.2.15 #define EEPROM_INFO_VERSION

The current version of the [HalEepromInformationType](#) data structure.

Definition at line 123 of file [bootloader-eprom.h](#).

6.28.2.16 #define EEPROM_INFO_MAJOR_VERSION

The current version of the [HalEepromInformationType](#) data structure.

Definition at line 124 of file [bootloader-eprom.h](#).

6.28.2.17 #define EEPROM_INFO_MAJOR_VERSION_MASK

The current version of the [HalEepromInformationType](#) data structure.

Definition at line 125 of file [bootloader-eprom.h](#).

6.28.2.18 `#define EEPROM_INFO_MIN_VERSION_WITH_WORD_SIZE_SUPPORT`

The current version of the [HalEepromInformationType](#) data structure.

Definition at line 131 of file [bootloader-eprom.h](#).

6.28.2.19 `#define EEPROM_CAPABILITIES_ERASE_SUPPORTED`

Eeprom capabilities mask that indicates the erase API is supported.

Definition at line 135 of file [bootloader-eprom.h](#).

6.28.2.20 `#define EEPROM_CAPABILITIES_PAGE_ERASE_REQD`

Eeprom capabilities mask that indicates page erasing is required before new data can be written to a device.

Definition at line 140 of file [bootloader-eprom.h](#).

6.28.3 Function Documentation

6.28.3.1 `uint8_t halEepromInit (void)`

Initialize EEPROM. Note: some earlier drivers may assert instead of returning an error if initialization fails.

Returns

[EEPROM_SUCCESS](#) or [EEPROM_ERR_INVALID_CHIP](#)

6.28.3.2 `void halEepromShutdown (void)`

Shutdown the EEPROM to conserve power.

6.28.3.3 `const HalEepromInformationType* halEepromInfo (void)`

Call this function to get information about the external EEPROM and its capabilities.

The format of this call must not be altered. However, the content can be changed to work with a different device.

Returns

A pointer to a [HalEepromInformationType](#) data structure, or NULL if the driver does not support this API

6.28.3.4 bool halEepromBusy (void)

Determine if the external EEPROM is still busy performing the last operation, such as a write or an erase.

The format of this call must not be altered. However, the content can be changed to work with a different device.

Returns

true if still busy or false if not.

6.28.3.5 uint8_t halEepromRead (uint32_t address, uint8_t * data, uint16_t len)

Read from the external EEPROM.

This is the standard external EEPROM read function. The format of this call must not be altered. However, the content can be changed to work with a different device. Note: Not all storage implementations support accesses that are not page aligned, refer to the [HalEepromInformationType](#) structure for more information.

Parameters

<i>address</i>	The address to start reading from.
<i>data</i>	A pointer to where read data is stored.
<i>len</i>	The length of data to read.

Returns

[EEPROM_SUCCESS](#) or [EEPROM_ERR](#)

6.28.3.6 uint8_t halEepromWrite (uint32_t address, const uint8_t * data, uint16_t len)

Write to the external EEPROM.

This is the standard external EEPROM write function. The format of this call must not be altered. However, the content can be changed to work with a different device. Note: Not all storage implementations support accesses that are not page aligned, refer to the [HalEepromInformationType](#) structure for more information. Note: Some storage devices require contents to be erased before new data can be written, and will return an [EEPROM_ERR_ERASE_REQUIRED](#) error if write is called on a location that is not already erased. Refer to the [HalEepromInformationType](#) structure to see if the attached storage device requires erasing.

Parameters

<i>address</i>	The address to start writing to.
<i>data</i>	A pointer to the data to write.
<i>len</i>	The length of data to write.

Returns

EEPROM_SUCCESS or [EEPROM_ERR](#)

6.28.3.7 uint8_t halEepromErase (uint32_t address, uint32_t totalLength)

Erases the specified region of the external EEPROM.

The format of this call must not be altered. However, the content can be changed to work with a different device.
Note: Most devices require the specified region to be page aligned, and will return an error if an unaligned region is specified.
Note: Many devices take an extremely long time to perform an erase operation. When erasing a large region, it may be preferable to make multiple calls to this API so that other application functionality can be performed while the erase is in progress.
The ::halEepromBusy() API may be used to determine when the last erase operation has completed. Erase timing information can be found in the HalEepromInformationType structure.

Parameters

<i>address</i>	Address to start erasing
<i>len</i>	Length of the region to be erased

Returns

[EEPROM_SUCCESS](#) or [EEPROM_ERR](#).

6.29 HAL Utilities

Modules

- [Cyclic Redundancy Code \(CRC\)](#)

6.29.1 Detailed Description

6.30 Cyclic Redundancy Code (CRC)

Macros

- #define [INITIAL_CRC](#)
- #define [CRC32_START](#)
- #define [CRC32_END](#)

Functions

- uint16_t [halCommonCrc16](#) (uint8_t newByte, uint16_t prevResult)
- uint32_t [halCommonCrc32](#) (uint8_t newByte, uint32_t prevResult)

6.30.1 Detailed Description

Functions that provide access to cyclic redundancy code (CRC) calculation. See [crc.h](#) for source code.

6.30.2 Macro Definition Documentation

6.30.2.1 #define INITIAL_CRC

Commonly used initial CRC32 value.

Definition at line [51](#) of file [crc.h](#).

6.30.2.2 #define CRC32_START

Commonly used initial CRC32 value.

Definition at line [56](#) of file [crc.h](#).

6.30.2.3 #define CRC32_END

Commonly used end CRC32 value for polynomial run LSB-MSB.

Definition at line [61](#) of file [crc.h](#).

6.30.3 Function Documentation

6.30.3.1 uint16_t halCommonCrc16 (uint8_t newByte, uint16_t prevResult)

Calculates 16-bit cyclic redundancy code (CITT CRC 16).

Applies the standard CITT CRC 16 polynomial to a single byte. It should support being called first with an initial value, then repeatedly until all data is processed.

Parameters

<i>newByte</i>	The new byte to be run through CRC.
<i>prevResult</i>	The previous CRC result.

Returns

The new CRC result.

6.30.3.2 `uint32_t halCommonCrc32 (uint8_t newByte, uint32_t prevResult)`

Calculates 32-bit cyclic redundancy code.

Note

On some radios or micros, the CRC for error detection on packet data is calculated in hardware.

Applies a CRC32 polynomial to a single byte. It should support being called first with an initial value, then repeatedly until all data is processed.

Parameters

<i>newByte</i>	The new byte to be run through CRC.
<i>prevResult</i>	The previous CRC result.

Returns

The new CRC result.

6.31 Application Utilities API Reference

Modules

- [Forming and Joining Networks](#)
- [Bootloading](#)
- [Command Interpreters](#)
- [ZigBee Device Object \(ZDO\) Information](#)
- [Message Fragmentation](#)
- [Network Manager](#)
- [Serial Communication](#)

6.31.1 Detailed Description

6.32 Forming and Joining Networks

Macros

- #define `NETWORK_STORAGE_SIZE`
- #define `NETWORK_STORAGE_SIZE_SHIFT`
- #define `FORM_AND_JOIN_MAX_NETWORKS`

Functions

- `EmberStatus emberScanForUnusedPanId` (uint32_t channelMask, uint8_t duration)
- `EmberStatus emberScanForJoinableNetwork` (uint32_t channelMask, uint8_t *extended-PanId)
- `EmberStatus emberScanForNextJoinableNetwork` (void)
- bool `emberFormAndJoinIsScanning` (void)
- bool `emberFormAndJoinCanContinueJoinableNetworkScan` (void)
- void `emberUnusedPanIdFoundHandler` (EmberPanId panId, uint8_t channel)
- void `emberJoinableNetworkFoundHandler` (EmberZigbeeNetwork *networkFound, uint8_t lqi, int8_t rssi)
- void `emberScanErrorHandler` (EmberStatus status)
- bool `emberFormAndJoinScanCompleteHandler` (uint8_t channel, EmberStatus status)
- bool `emberFormAndJoinNetworkFoundHandler` (EmberZigbeeNetwork *networkFound, uint8_t lqi, int8_t rssi)
- bool `emberFormAndJoinEnergyScanResultHandler` (uint8_t channel, int8_t maxRssiValue)
- void `emberFormAndJoinTick` (void)
- void `emberFormAndJoinTaskInit` (void)
- void `emberFormAndJoinRunTask` (void)
- void `emberFormAndJoinCleanup` (EmberStatus status)

Variables

- bool `emberEnableDualChannelScan`

6.32.1 Detailed Description

Functions for finding an existing network to join and for finding an unused PAN id with which to form a network.

Summary of application requirements:

For the SOC:

- Define `::EMBER_APPLICATION_HAS_ENERGY_SCAN_RESULT_HANDLER` in the configuration header.
- Call `emberFormAndJoinTick()` regularly in the main loop.
- Include `form-and-join.c` and `form-and-join-node-adapter.c` in the build.
- Optionally include `form-and-join-node-callbacks.c` in the build.

- If processor idling is desired: – Call `emberFormAndJoinTaskInit()` to initialize the form and join task – Call `emberFormAndJoinRunTask()` regularly in the main loop instead of `emberFormAndJoinTick()`

For an EZSP Host:

- Define `::EZSP_APPLICATION_HAS_ENERGY_SCAN_RESULT_HANDLER` in the configuration header.
- Include `form-and-join.c` and `form-and-join-host-adapter.c` in the build.
- Optionally include `form-and-join-host-callbacks.c` in the build.

For either platform, the application can omit the `form-and-join-*-callback.c` file from the build and implement the callbacks itself if necessary. In this case the appropriate form-and-join callback function must be called from within each callback, as is done within the `form-and-join-*-callback.c` files.

On either platform, `FORM_AND_JOIN_MAX_NETWORKS` can be explicitly defined to limit (or expand) the number of joinable networks that the library will save for consideration during the scan process.

The library is able to resume scanning for joinable networks from where it left off, via a call to `emberScanForNextJoinableNetwork()`. Thus if the first joinable network found is not the correct one, the application can continue scanning without starting from the beginning and without finding the same network that it has already rejected. The library can also be used on the host processor.

6.32.2 Macro Definition Documentation

6.32.2.1 `#define NETWORK_STORAGE_SIZE`

Number of bytes required to store relevant info for a saved network.

This constant represents the minimum number of bytes required to store all members of the `NetworkInfo` struct used in the adapter code. Its value should not be changed unless the underlying adapter code is updated accordingly. Note that this constant's value may be different than `sizeof(NetworkInfo)` because some compilers pad the structs to align on word boundaries. Thus, the adapter code stores/retrieves these pieces of data individually (to be platform-agnostic) rather than as a struct.

For efficiency's sake, this number should be kept to a power of 2 and not and not exceed 32 (`PACKET_BUFFER_SIZE`).

Definition at line 68 of file [form-and-join.h](#).

6.32.2.2 `#define NETWORK_STORAGE_SIZE_SHIFT`

Log_base2 of `NETWORK_STORAGE_SIZE`.

Definition at line 72 of file [form-and-join.h](#).

6.32.2.3 `#define FORM_AND_JOIN_MAX_NETWORKS`

Number of joinable networks that can be remembered during the scan process.

Note for SoC Platforms: This is currently limited to a maximum of 15 due to the size of each network entry (16 bytes) and the EmberMessageBuffer API's requirement that total buffer storage length be kept to an 8-bit quantity (less than 256).

Note for EZSP Host Platforms: In the host implementation of this library, the storage size for the detected networks buffer is controlled by [EZSP_HOST_FORM_AND_JOIN_BUFFER_SIZE](#), so that limits the highest value that the host can set for FORM_AND_JOIN_MAX_NETWORKS.

Definition at line 94 of file [form-and-join.h](#).

6.32.3 Function Documentation

6.32.3.1 EmberStatus emberScanForUnusedPanId (uint32_t *channelMask*, uint8_t *duration*)

Find an unused PAN id.

Does an energy scan on the indicated channels and randomly chooses one from amongst those with the least average energy. Then picks a short PAN id that does not appear during an active scan on the chosen channel. The chosen PAN id and channel are returned via the [emberUnusedPanIdFoundHandler\(\)](#) callback. If an error occurs, the application is informed via the [emberScanErrorHandler\(\)](#).

Parameters

<i>channelMask</i>	
<i>duration</i>	The duration of the energy scan. See the documentation for ::emberStartScan() in stack/include/network-formation.h for information on duration values.

Returns

EMBER_LIBRARY_NOT_PRESENT if the form and join library is not available.

6.32.3.2 EmberStatus emberScanForJoinableNetwork (uint32_t *channelMask*, uint8_t * *extendedPanId*)

Finds a joinable network.

Performs an active scan on the specified channels looking for networks that:

1. currently permit joining,
2. match the stack profile of the application,
3. match the extended PAN id argument if it is not NULL.

Upon finding a matching network, the application is notified via the [emberJoinableNetworkFoundHandler\(\)](#) callback, and scanning stops. If an error occurs during the scanning process, the application is informed via the [emberScanErrorHandler\(\)](#), and scanning stops.

If the application determines that the discovered network is not the correct one, it may call [emberScanForNextJoinableNetwork\(\)](#) to continue the scanning process where it was left off and find a different joinable network. If the next network is not the correct one,

the application can continue to call [emberScanForNextJoinableNetwork\(\)](#). Each call must occur within 30 seconds of the previous one, otherwise the state of the scan process is deleted to free up memory. Calling [emberScanForJoinableNetwork\(\)](#) causes any old state to be forgotten and starts scanning from the beginning.

Parameters

<i>channelMask</i>	
<i>extendedPan-Id</i>	

Returns

EMBER_LIBRARY_NOT_PRESENT if the form and join library is not available.

6.32.3.3 EmberStatus emberScanForNextJoinableNetwork (void)

See [emberScanForJoinableNetwork\(\)](#).

6.32.3.4 bool emberFormAndJoinIsScanning (void)

Returns true if and only if the form and join library is in the process of scanning and is therefore expecting scan results to be passed to it from the application.

6.32.3.5 bool emberFormAndJoinCanContinueJoinableNetworkScan (void)

Returns true if and only if the application can continue a joinable network scan by calling [emberScanForNextJoinableNetwork\(\)](#). See [emberScanForJoinableNetwork\(\)](#).

6.32.3.6 void emberUnusedPanIdFoundHandler (EmberPanId panId, uint8_t channel)

A callback the application needs to implement.

Notifies the application of the PAN id and channel found following a call to [emberScanForUnusedPanId\(\)](#).

Parameters

<i>panId</i>	
<i>channel</i>	

6.32.3.7 void emberJoinableNetworkFoundHandler (EmberZigbeeNetwork * networkFound, uint8_t lqi, int8_t rssi)

A callback the application needs to implement.

Notifies the application of the network found after a call to [emberScanForJoinableNetwork\(\)](#) or [emberScanForNextJoinableNetwork\(\)](#).

Parameters

<i>network-Found</i>	
<i>lqi</i>	The lqi value of the received beacon.
<i>rsssi</i>	The rssi value of the received beacon.

6.32.3.8 void emberScanErrorHandler (EmberStatus *status*)

A callback the application needs to implement.

If an error occurs while scanning, this function is called and the scan effort is aborted.

Possible return status values are:

- EMBER_INVALID_CALL: if `emberScanForNextJoinableNetwork()` is called more than 30 seconds after a previous call to `emberScanForJoinableNetwork()` or `emberScanForNextJoinableNetwork()`.
- EMBER_NO_BUFFERS: if there is not enough memory to start a scan.
- EMBER_NO_BEACONS: if no joinable beacons are found.
- EMBER_MAC_SCANNING: if a scan is already in progress.

Parameters

<i>status</i>	
---------------	--

6.32.3.9 bool emberFormAndJoinScanCompleteHandler (uint8_t *channel*, EmberStatus *status*)

The application must call this function from within its `emberScanCompleteHandler()` (on the node) or `ezspScanCompleteHandler()` (on an EZSP host). Default callback implementations are provided in the `form-and-join-*-callbacks.c` files.

Returns

true iff the library made use of the call.

6.32.3.10 bool emberFormAndJoinNetworkFoundHandler (EmberZigbeeNetwork * *networkFound*, uint8_t *lqi*, int8_t *rsssi*)

The application must call this function from within its `emberNetworkFoundHandler()` (on the node) or `ezspNetworkFoundHandler()` (on an EZSP host). Default callback implementations are provided in the `form-and-join-*-callbacks.c` files.

Returns

true iff the library made use of the call.

6.32.3.11 **bool emberFormAndJoinEnergyScanResultHandler (uint8_t *channel*, int8_t *maxRssiValue*)**

The application must call this function from within its `emberEnergyScanResultHandler()` (on the node) or `ezspEnergyScanResultHandler()` (on an EZSP host). Default callback implementations are provided in the `form-and-join-*-callbacks.c` files.

Returns

true iff the library made use of the call.

6.32.3.12 **void emberFormAndJoinTick (void)**

Used by the form and join code on the node to time out a joinable scan after 30 seconds of inactivity. The application must call `emberFormAndJoinTick()` regularly. This function does not exist for the EZSP host library.

6.32.3.13 **void emberFormAndJoinTaskInit (void)**

When processor idling is desired on the SOC, this must be called to properly initialize the form and join library.

6.32.3.14 **void emberFormAndJoinRunTask (void)**

When processor idling is desired on the SOC, this should be called regularly instead of `emberFormAndJoinTick()`

6.32.3.15 **void emberFormAndJoinCleanup (EmberStatus *status*)**

When form-and-join state is no longer needed, the application can call this routine to cleanup and free resources. On the SOC platforms this will free the allocated message buffer.

6.32.4 Variable Documentation

6.32.4.1 **bool emberEnableDualChannelScan**

With some board layouts, the EM250 and EM260 are susceptible to a dual channel issue in which packets from 12 channels above or below can sometimes be heard faintly. This affects channels 11 - 14 and 23 - 26. Hardware reference designs EM250_REF_DES_LAT, version C0 and EM250_REF_DES_CER, version B0 solve the problem.

Setting the `emberEnableDualChannelScan` variable to true enables a software workaround to the dual channel issue which can be used with vulnerable boards. After `emberScanForJoinableNetwork()` discovers a network on one of the susceptible channels, the channel number that differs by 12 is also scanned. If the same network can be heard there, the true channel is determined by comparing the link quality of the received beacons. The default value of `emberEnableDualChannelScan` is true for the EM250 and EM260. It is not used on other platforms.

6.33 Bootloading

Modules

- [Stand-Alone Bootloader for EZSP](#)
- [Stand-Alone Bootloader Library](#)

6.33.1 Detailed Description

For a thorough discussion of bootloading, see the Bootloading chapter of the *EmberZNet Application Developers Guide*. There are three forms of the bootloading API.

6.34 Stand-Alone Bootloader for EZSP

Macros

- `#define TICKS_PER_QUARTER_SECOND`

Functions

- `bool hostBootloadUtilLaunchRequestHandler (uint8_t lqi, int8_t rssi, uint16_t manufacturerId, uint8_t *hardwareTag, EmberEUI64 sourceEui)`
- `void hostBootloadUtilQueryResponseHandler (uint8_t lqi, int8_t rssi, bool bootloaderActive, uint16_t manufacturerId, uint8_t *hardwareTag, EmberEUI64 targetEui, uint8_t bootloaderCapabilities, uint8_t platform, uint8_t micro, uint8_t phy, uint16_t blVersion)`
- `void hostBootloadReinitHandler (void)`
- `bool isTheSameEui64 (EmberEUI64 sourceEui, EmberEUI64 targetEui)`
- `void printLittleEndianEui64 (uint8_t port, EmberEUI64 eui64)`
- `void printBigEndianEui64 (uint8_t port, EmberEUI64 eui64)`
- `EmberStatus debugPrintf (uint8_t port, PGM_P formatString,...)`

Variables

- `uint16_t nodeBIVersion`
- `uint8_t nodePlat`
- `uint8_t nodeMicro`
- `uint8_t nodePhy`
- `EzspStatus bootloadEzspLastError`
- `EzspStatus ignoreNextEzspError`

6.34.1 Detailed Description

All functions and variables defined here can be used by applications. See [bootload-ezsp-utils.h](#) for source code.

6.34.2 Macro Definition Documentation

6.34.2.1 `#define TICKS_PER_QUARTER_SECOND`

Definition at line 23 of file [bootload-ezsp-utils.h](#).

6.34.3 Function Documentation

6.34.3.1 `bool hostBootloadUtilLaunchRequestHandler (uint8_t lqi, int8_t rssi, uint16_t manufacturerId, uint8_t * hardwareTag, EmberEUI64 sourceEui)`

A callback function invoked by `bootload-ezsp-utils` when a bootload launch request message is received.

The application may choose whether or not to enter the bootloader by checking the manufacturer-Id, hardwareTag, and sourceEui. If the application chooses to launch the bootloader, the bootloader will launch after successful completion of the bootloader launch authentication protocol.

Parameters

<i>lqi</i>	The link quality from the node that generated this bootload launch request.
<i>rssi</i>	The energy level (in units of dBm) observed during the reception.
<i>manufacturer-Id</i>	The manufacturer specification (vendor specific) of the sending node.
<i>hardwareTag</i>	The hardware specification (vendor specific) of the sending node.
<i>sourceEui</i>	The EUI64 of the sending node.

Returns

true if the application wishes to launch the bootloader, false if the application does not wish to launch the bootloader.

6.34.3.2 void hostBootloadUtilQueryResponseHandler (uint8_t *lqi*, int8_t *rssi*, bool *bootloaderActive*, uint16_t *manufacturerId*, uint8_t * *hardwareTag*, EmberEUI64 *targetEui*, uint8_t *bootloaderCapabilities*, uint8_t *platform*, uint8_t *micro*, uint8_t *phy*, uint16_t *blVersion*)

A callback function invoked by bootload-ezsp-utils when a bootload query response message is received.

This is particularly useful when the application needs to decide which node to bootload. Several attributes of the responding node are provided to the application. The application can use these attributes to decide whether to bootload or how to bootload a given node.

Parameters

<i>lqi</i>	The link quality from the node that generated this bootload query response.
<i>rssi</i>	The energy level (in units of dBm) observed during the reception.
<i>bootloader-Active</i>	true if the responding node is running the bootloader, false if not.
<i>manufacturer-Id</i>	The manufacturer specification (vendor specific) of the responding node.
<i>hardwareTag</i>	The hardware specification (vendor specific) of the responding node.
<i>targetEui</i>	The EUI64 of the responding node.
<i>bootloader-Capabilities</i>	If the lsb is 1, the bootloader on the responding node supports encrypted bootloader message payloads.
<i>platform</i>	The type of platform of the responding node. 1 is avr-atmega, 2 is xap2b.
<i>micro</i>	The type of microcontroller on the responding node. Value depends on platform. 1 is the avr-atmega 64, 2 is the avr-atmega 128, 1 is the xap2b em250.
<i>phy</i>	The type of phy of the responding node. 1 is em2420, 2 is em250.
<i>blVersion</i>	The version of standalone bootloader of the responding node. This is a 2 byte field. The high byte is the version and the low byte is the build. A value of 0xFFFF means unknown. For example, a version field of 0x1234 is version 1.2, build 34.

6.34.3.3 void hostBootloadReinitHandler (void)

A callback function invoked by bootload-ezsp-utils when a NCP has finished being bootloaded.

The application can handle this as simply as calling on [halReboot\(\)](#) or as complex as needed.

6.34.3.4 bool isTheSameEui64 (EmberEUI64 *sourceEui*, EmberEUI64 *targetEui*)

A function to compare EUI64s.

Compare two EUI64s.

Parameters

<i>sourceEui</i>	The EUI64 of the sending node.
<i>targetEui</i>	The EUI64 of the responding node.

Returns

true if the EUI64s are the same. false if the EUI64s are different.

6.34.3.5 void printLittleEndianEui64 (uint8_t *port*, EmberEUI64 *eui64*)

A function to display an EUI64.

Display an EUI64 in little endian format.

Parameters

<i>port</i>	The serial port to use. 0 for Mega128 port. 0 or 1 for Linux ports.
<i>eui64</i>	The EUI64 to display.

6.34.3.6 void printBigEndianEui64 (uint8_t *port*, EmberEUI64 *eui64*)

A function to display an EUI64.

Display an EUI64 in big endian format.

Parameters

<i>port</i>	The serial port to use. 0 for Mega128 port. 0 or 1 for Linux ports.
<i>eui64</i>	The EUI64 to display.

6.34.3.7 EmberStatus debugPrintf (uint8_t *port*, PGM_P *formatString*, ...)

A function to similar to [emberSerialPrintf\(\)](#).

Output to local ports.

Parameters

<i>port</i>	The serial port to use. 0 for Mega128 port. 0 or 1 for Linux ports.
<i>formatString</i>	The string to print.
...	Format specifiers.

Returns

One of the following (see the Main Page):

- EMBER_SERIAL_TX_OVERFLOW indicates that data was dropped.
- EMBER_NO_BUFFERS indicates that there was an insufficient number of available stack buffers.
- EMBER_SUCCESS.

6.34.4 Variable Documentation

6.34.4.1 uint16_t nodeBIVersion

6.34.4.2 uint8_t nodePlat

6.34.4.3 uint8_t nodeMicro

6.34.4.4 uint8_t nodePhy

6.34.4.5 EzspStatus bootloadEzspLastError

6.34.4.6 EzspStatus ignoreNextEzspError

6.35 Stand-Alone Bootloader Library

Macros

- #define `BOOTLOAD_HARDWARE_TAG_SIZE`

Enumerations

- enum `bootloadMode` { `BOOTLOAD_MODE_NONE`, `BOOTLOAD_MODE_PASSTHRU` }
- enum `bootloadState` { `BOOTLOAD_STATE_NORMAL`, `BOOTLOAD_STATE_QUERY`, `BOOTLOAD_STATE_WAIT_FOR_AUTH_CHALLENGE`, `BOOTLOAD_STATE_WAIT_FOR_AUTH_RESPONSE`, `BOOTLOAD_STATE_DELAY_BEFORE_START`, `BOOTLOAD_STATE_START_UNICAST_BOOTLOAD`, `BOOTLOAD_STATE_START_BROADCAST_BOOTLOAD`, `BOOTLOAD_STATE_START_SENDING_IMAGE`, `BOOTLOAD_STATE_SENDING_IMAGE`, `BOOTLOAD_STATE_WAIT_FOR_IMAGE_ACK`, `BOOTLOAD_STATE_WAIT_FOR_COMPLETE_ACK`, `BOOTLOAD_STATE_DONE` }

Functions

- void `bootloadUtilInit` (uint8_t appPort, uint8_t bootloadPort)
- `EmberStatus` `bootloadUtilSendRequest` (`EmberEUI64` targetEui, uint16_t mfgId, uint8_t hardwareTag[`BOOTLOAD_HARDWARE_TAG_SIZE`], uint8_t encryptKey[`BOOTLOAD_AUTH_COMMON_SIZE`], `bootloadMode` mode)
- void `bootloadUtilSendQuery` (`EmberEUI64` target)
- void `bootloadUtilStartBootload` (`EmberEUI64` target, `bootloadMode` mode)
- void `bootloadUtilTick` (void)
- bool `bootloadUtilLaunchRequestHandler` (uint16_t manufacturerId, uint8_t hardwareTag[`BOOTLOAD_HARDWARE_TAG_SIZE`], `EmberEUI64` sourceEui)
- void `bootloadUtilQueryResponseHandler` (bool bootloaderActive, uint16_t manufacturerId, uint8_t hardwareTag[`BOOTLOAD_HARDWARE_TAG_SIZE`], `EmberEUI64` targetEui, uint8_t bootloaderCapabilities, uint8_t platform, uint8_t micro, uint8_t phy, uint16_t blVersion)
- void `bootloadUtilSendAuthResponse` (`EmberEUI64` target)

Authentication Challenge and Response

The authentication challenge and response must be the same size. The size is chosen to be evenly divisible by the size of a 128-bit AES block.

- #define `BOOTLOAD_AUTH_COMMON_SIZE`
- #define `BOOTLOAD_AUTH_CHALLENGE_SIZE`
- #define `BOOTLOAD_AUTH_RESPONSE_SIZE`

Bootload State Variables

Used to check whether a bootloading process is currently happening.

- [bootloadState blState](#)
- [#define IS_BOOTLOADING](#)

6.35.1 Detailed Description

All functions and variables defined here can be used by applications. See [bootload-utils.h](#) for source code.

Applications can use this stand-alone bootload library to:

1. Load a new (application) image on itself via serial bootload through uart port 1 using the xmodem protocol.
2. Load a new image on a remote node over-the-air (OTA) from a host (PC), also known as a passthru bootload.
3. Recover a node that failed during the bootloading process, also known as a recovery bootload.

Note from the diagrams below that with over-the-air bootloading the source node (node transmitting bootload packets) and the target node (node being loaded with a new image) need to be one hop away because bootload packets are IEEE 802.15.4 packets.

In case of recovery, the source (recovery) node does not need to be part of the network since all recovery packets are 802.15.4 packets.

A diagram for typical serial bootloading:

[host pc] –(RS232 or Ethernet/IP network)– {uart1 or port 4901}[node]

A diagram for typical passthru bootloading:

[host pc] –(RS232 or Ethernet)– [source node]–(OTA)–[target node]

A diagram for typical recovery bootloading:

[source node] –(OTA)–[target node]

Note

Applications that use the bootload utilities need to `#define EMBER_APPLICATION_HAS_BOOTLOAD_HANDLERS` within their `CONFIGURATION_HEADER`.

6.35.2 Macro Definition Documentation

6.35.2.1 `#define BOOTLOAD_AUTH_COMMON_SIZE`

Definition at line 66 of file [bootload-utils.h](#).

6.35.2.2 `#define BOOTLOAD_AUTH_CHALLENGE_SIZE`

Definition at line 67 of file [bootload-utils.h](#).

6.35.2.3 `#define BOOTLOAD_AUTH_RESPONSE_SIZE`

Definition at line 68 of file [bootload-utils.h](#).

6.35.2.4 `#define BOOTLOAD_HARDWARE_TAG_SIZE`

Size of hardware tag which is an array of `uint8_t`.

// End set of defines

Definition at line 76 of file [bootload-utils.h](#).

6.35.2.5 `#define IS_BOOTLOADING`

Definition at line 300 of file [bootload-utils.h](#).

6.35.3 Enumeration Type Documentation

6.35.3.1 `enum bootloadMode`

Bootload modes supported by the bootload utility library.

Enumerator:

BOOTLOAD_MODE_NONE Used when we are not currently doing any bootloading.

BOOTLOAD_MODE_PASSTHRU Used when doing normal and recovery passthru bootload.

Definition at line 82 of file [bootload-utils.h](#).

6.35.3.2 `enum bootloadState`

A bootload state is a value that an application can check to see if bootloading is in progress.

This is necessary because we want the application to be aware that bootloading is going on and it needs to limit its activities. For example, when passthru bootloading is going on, do not print anything to a serial port because it may violate the XModem protocol. Also, try to limit radio activities to a minimum to avoid any interruptions to bootload progress. Used in a bootload state machine.

Enumerator:

BOOTLOAD_STATE_NORMAL Start state

BOOTLOAD_STATE_QUERY After send query message

BOOTLOAD_STATE_WAIT_FOR_AUTH_CHALLENGE Wait for authentication challenge

BOOTLOAD_STATE_WAIT_FOR_AUTH_RESPONSE Wait for authentication response

BOOTLOAD_STATE_DELAY_BEFORE_START Delay state before start new action

BOOTLOAD_STATE_START_UNICAST_BOOTLOAD After start unicast bootloading

BOOTLOAD_STATE_START_BROADCAST_BOOTLOAD After start broadcast bootloading

BOOTLOAD_STATE_START_SENDING_IMAGE Need to start XMODEM code

BOOTLOAD_STATE_SENDING_IMAGE During sending OTA data messages

BOOTLOAD_STATE_WAIT_FOR_IMAGE_ACK Wait for OTA data ack

BOOTLOAD_STATE_WAIT_FOR_COMPLETE_ACK Wait for OTA end transmission ack

BOOTLOAD_STATE_DONE Finish bootloading

Definition at line 106 of file [bootload-utils.h](#).

6.35.4 Function Documentation

6.35.4.1 void bootloadUtilInit (uint8_t appPort, uint8_t bootloadPort)

Bootload library initialization.

The application needs to define the ports to be used for printing information and for a (passthru) bootload.

Note

Generally it's a good idea to use different ports for the application and for bootloading because when doing passthru bootloading, we do not want to print any additional data that can cause an XModem transaction to fail.

Parameters

<i>appPort</i>	Port used for printing information.
<i>bootloadPort</i>	Port used for passthru bootloading.

6.35.4.2 EmberStatus bootloadUtilSendRequest (EmberEUI64 targetEui, uint16_t mfgId, uint8_t hardwareTag[BOOTLOAD_HARDWARE_TAG_SIZE], uint8_t encryptKey[BOOTLOAD_AUTH_COMMON_SIZE], bootloadMode mode)

Start the bootload process on a remote node that is currently running stack/application.

The source node sends a bootload request message to initiate the bootload authentication process. The source node then enters a state waiting for the target node to send an authentication challenge, which it will encrypt and send back as a response. MfgId and hardwareTag information is sent over the air to the target node to verify whether to go into bootload mode. The encryption key is saved on the source node for later authentication. The mode indicates the bootload mode that the source will be using.

Parameters

<i>targetEui</i>	Node to be bootloaded.
<i>mfgId</i>	Manufacturer ID (vendor specific).

<i>hardwareTag</i>	Hardware ID, such as a board (vendor specific).
<i>encryptKey</i>	Key used in the authentication process.
<i>mode</i>	Bootload mode to be used is passthru (0x01).

Returns

EMBER_SUCCESS if successful, or EMBER_NO_BUFFERS, or EMBER_ERR_FATAL if the function was called too soon after a previous call to it.

6.35.4.3 void bootloadUtilSendQuery (EmberEUI64 *target*)

A function to send query message to gather basic information about the node(s).

There are two types of query messages: broadcast and unicast. Broadcast query is generally used to gather information regarding a neighboring node, especially the eui64 of the node. Unicast query is used when we already know the eui64 of the target node that we need information from.

Parameters

<i>target</i>	The node we want to gather information from. If the value is NULL, that means we want to do a broadcast query.
---------------	--

6.35.4.4 void bootloadUtilStartBootload (EmberEUI64 *target*, bootloadMode *mode*)

Start the bootload process on a remote node that is already running in bootload mode.

This is generally to recover a node that failed during bootload. The failure can be caused by the source node resetting, the network being too busy, a software reset, and so on. However, the failure is not caused by a target node losing power. After the failure, the node stays in bootload mode on the same (current) channel.

Parameters

<i>target</i>	remote node to be bootloaded. If the value is NULL, that means we do not know the eui64 of the target node. A broadcast (start bootload) packet is sent and the first node that replies will be bootloaded.
<i>mode</i>	bootload mode to be used, such as passthru (0x01).

6.35.4.5 void bootloadUtilTick (void)

A function in the application's heartbeat or tick function that contains basic bootloading state machine and also manages the bootload timer.

6.35.4.6 bool bootloadUtilLaunchRequestHandler (uint16_t *manufacturerId*, uint8_t *hardwareTag*[BOOTLOAD_HARDWARE_TAG_SIZE], EmberEUI64 *sourceEui*)

A callback function invoked by bootload-utils when a bootload request message is received.

The application may choose whether or not to enter the bootloader by checking the manufacturer-Id, hardwareTag, and sourceEui. If the application chooses to launch the bootloader, the bootloader will launch after successful completion of the bootloader launch authentication protocol.

Parameters

<i>manufacturer-Id</i>	The manufacturer specification (vendor specific) of the sending node.
<i>hardwareTag</i>	The hardware specification (vendor specific) of the sending node.
<i>sourceEui</i>	The EUI64 of the sending node.

Returns

true if the application wishes to launch the bootloader, false if the application does not wish to launch the bootloader.

6.35.4.7 void `bootloadUtilQueryResponseHandler (bool bootloaderActive, uint16_t manufacturerId, uint8_t hardwareTag[BOOTLOAD_HARDWARE_TAG_SIZE], EmberEUI64 targetEui, uint8_t bootloaderCapabilities, uint8_t platform, uint8_t micro, uint8_t phy, uint16_t blVersion)`

A callback function invoked by bootload-utils when a bootload query response message is received.

This is particularly useful when the application needs to decide which node to bootload. Several attributes of the responding node are provided to the application. The application can use these attributes to decide whether to bootload or how to bootload a given node.

Parameters

<i>bootloader-Active</i>	true if the responding node is running the bootloader, false if not.
<i>manufacturer-Id</i>	The manufacturer specification (vendor specific) of the responding node.
<i>hardwareTag</i>	The hardware specification (vendor specific) of the responding node.
<i>targetEui</i>	The EUI64 of the responding node.
<i>bootloader-Capabilities</i>	If the lsb is 1, the bootloader on the responding node supports encrypted bootloader message payloads.
<i>platform</i>	The type of platform of the responding node. 1 is avr-atmega, 2 is xap2b.
<i>micro</i>	The type of microcontroller on the responding node. Value depends on platform. 1 is the avr-atmega 64, 2 is the avr-atmega 128, 1 is the xap2b em250.
<i>phy</i>	The type of phy of the responding node. 1 is em2420, 2 is em250.
<i>blVersion</i>	The version of standalone bootloader of the responding node. This is a 2 byte field. The high byte is the version and the low byte is the build. A value of 0xFFFF means unknown. For example, a version field of 0x1234 is version 1.2, build 34.

6.35.4.8 void bootloaderUtilSendAuthResponse (EmberEUI64 *target*)

A function called by a parent node to send an authentication response message to the sleepy or mobile end-device target node.

The message is sent as a Just-In-Time (JIT) message, hence, the end-device target needs to poll for the message.

The bootloader utility library will call this function automatically if bootloading the router node.

Parameters

<i>target</i>	The end-device target node being bootloaded.
---------------	--

6.35.5 Variable Documentation

6.35.5.1 bootloaderState blState

6.36 Command Interpreters

Modules

- [Command Interpreter 2](#)

6.36.1 Detailed Description

6.37 Command Interpreter 2

Data Structures

- struct `EmberCommandEntry`
Command entry for a command table.

Macros

- #define `MAX_TOKEN_COUNT`
- #define `emberCommandEntryAction`(name, action, argumentTypes, description)
- #define `emberCommandEntryActionWithDetails`(name, action, argumentTypes, description, argument-DescriptionArray)
- #define `emberCommandEntrySubMenu`(name, subMenu, description)
- #define `emberCommandEntryTerminator`()
- #define `EMBER_COMMAND_INTERPRETER_CONFIGURATION_ECHO`
- #define `emberProcessCommandInput`(port)
- #define `emberCommandInterpreterEchoOn`()
- #define `emberCommandInterpreterEchoOff`()
- #define `emberCommandInterpreterIsEchoOn`()

Typedefs

- typedef void(* `CommandAction`)(void)

Enumerations

- enum `EmberCommandStatus` {
`EMBER_CMD_SUCCESS`, `EMBER_CMD_ERR_PORT_PROBLEM`, `EMBER_CMD_ERR_NO_SUCH_COMMAND`, `EMBER_CMD_ERR_WRONG_NUMBER_OF_ARGUMENTS`,
`EMBER_CMD_ERR_ARGUMENT_OUT_OF_RANGE`, `EMBER_CMD_ERR_ARGUMENT_SYNTAX_ERROR`, `EMBER_CMD_ERR_STRING_TOO_LONG`, `EMBER_CMD_ERR_INVALID_ARGUMENT_TYPE` }

Functions

- void `emberCommandReaderSetDefaultBase` (uint8_t base)
- void `emberCommandActionHandler` (const `CommandAction` action)
- void `emberCommandErrorHandler` (`EmberCommandStatus` status)
- void `emberPrintCommandUsage` (`EmberCommandEntry` *entry)
- void `emberPrintCommandUsageNotes` (void)
- void `emberPrintCommandTable` (void)
- void `emberCommandClearBuffer` (void)
- void `emberCommandReaderInit` (void)
- bool `emberProcessCommandString` (uint8_t *input, uint8_t sizeOrPort)

Variables

- [EmberCommandEntry](#) * emberCurrentCommand
- [EmberCommandEntry](#) emberCommandTable []
- [uint8_t](#) emberCommandInterpreter2Configuration

Command Table Settings

- [#define](#) EMBER_MAX_COMMAND_ARGUMENTS
- [#define](#) EMBER_COMMAND_BUFFER_LENGTH
- [#define](#) EMBER_COMMAND_INTEPRETER_HAS_DESCRIPTION_FIELD

Functions to Retrieve Arguments

Use the following functions in your functions that process commands to retrieve arguments from the command interpreter. These functions pull out unsigned integers, signed integers, and strings, and hex strings. Index 0 is the first command argument.

- [uint8_t](#) emberCommandArgumentCount (void)
- [uint32_t](#) emberUnsignedCommandArgument (uint8_t argNum)
- [int32_t](#) emberSignedCommandArgument (uint8_t argNum)
- [bool](#) emberStringToHostOrderIpv4Address (const [uint8_t](#) *string, [uint32_t](#) *host-OrderIpv4Address)
- [bool](#) emberStringArgumentToHostOrderIpv4Address (uint8_t argNum, [uint32_t](#) *host-OrderIpv4Address)
- [uint8_t](#) * emberStringCommandArgument (int8_t argNum, [uint8_t](#) *length)
- [const char](#) * emberCommandName (void)
- [uint8_t](#) emberCopyStringArgument (int8_t argNum, [uint8_t](#) *destination, [uint8_t](#) maxLength, [bool](#) leftPad)
- [uint8_t](#) emberCopyBigEndianEui64Argument (int8_t index, [EmberEUI64](#) destination)
- [#define](#) emberCopyKeyArgument(index, keyDataPointer)
- [#define](#) emberCopyEui64Argument(index, eui64)
- [#define](#) emberGetEui64Argument(index, eui64)

6.37.1 Detailed Description

Interpret serial port commands. See command-interpreter2.c for source code.

See the following application usage example followed by a brief explanation.

```
// Usage: network form 22 0xAB12 -3 { 00 01 02 A3 A4 A5 A6 A7 }
void formCommand(void)
{
    uint8_t channel = emberUnsignedCommandArgument(0)
    ;
    uint16_t panId = emberUnsignedCommandArgument(1)
    ;
    int8_t power = emberSignedCommandArgument(2);
    uint8_t length;
    uint8_t *eui64 = emberStringCommandArgument(3, &
        length);
    ...
    ... call emberFormNetwork() etc
    ...
}
```

```

}

// The main command table.
EmberCommandEntry emberCommandTable[] = {
    emberCommandEntrySubMenu("network",    networkCommands,
        "Network form/join commands"),
    emberCommandEntryAction("status",      statusCommand,
        "Prints application status"),
    ...
    emberCommandEntryTerminator()
};

// The table of network commands.
EmberCommandEntry networkCommands[] = {
    emberCommandEntryAction("form", formCommand, "uvsh", "Form a network"),
    emberCommandEntryAction("join", joinCommand, "uvsh", "Join a network"),
    ...
    emberCommandEntryTerminator()
};

void main(void)
{
    emberCommandReaderInit();
    while(0) {
        ...
        // Process input and print prompt if it returns true.
        if (emberProcessCommandInput(serialPort)) {
            emberSerialPrintf(1, "%p>", PROMPT);
        }
        ...
    }
}

```

1. Applications specify the commands that can be interpreted by defining the `emberCommandTable` array of type `EmberCommandEntry`. The table includes the following information for each command:
 - (a) The full command name.
 - (b) Your application's function name that implements the command.
 - (c) An `EmberCommandEntry::argumentTypes` string specifies the number and types of arguments the command accepts. See `::argumentTypes` for details.
 - (d) A description string explains the command.
2. A default error handler `emberCommandErrorHandler()` is provided to deal with incorrect command input. Applications may override it.
3. The application calls `emberCommandReaderInit()` to initialize, and `emberProcessCommandInput()` in its main loop.
4. Within the application's command functions, use `emberXXXCommandArgument()` functions to retrieve command arguments.

The command interpreter does extensive processing and validation of the command input before calling the function that implements the command. It checks that the number, type, syntax, and range of all arguments are correct. It performs any conversions necessary (for example, converting integers and strings input in hexadecimal notation into the corresponding bytes), so that no additional parsing is necessary within command functions. If there is an error in the command input, `emberCommandErrorHandler()` is called rather than a command function.

The command interpreter allows inexact matches of command names. The input command may be either shorter or longer than the actual command. However, if more than one inexact match is found and there is no exact match, an error of type `EMBER_CMD_ERR_NO_SUCH_COMMAND` will be generated. To disable this feature, define `EMBER_REQUIRE_EXACT_COMMAND_NAME` in the application configuration header.

6.37.2 Macro Definition Documentation

6.37.2.1 #define EMBER_MAX_COMMAND_ARGUMENTS

The maximum number of arguments a command can have. A nested command counts as an argument.

Definition at line 104 of file [command-interpreter2.h](#).

6.37.2.2 #define EMBER_COMMAND_BUFFER_LENGTH

The maximum number of arguments a command can have. A nested command counts as an argument.

Definition at line 108 of file [command-interpreter2.h](#).

6.37.2.3 #define EMBER_COMMAND_INTEPRETER_HAS_DESCRIPTION_FIELD

Whether or not the command entry structure will include descriptions for the commands. This consumes additional CONST space, which is expensive on the XAP. By default descriptions are not included.

Definition at line 116 of file [command-interpreter2.h](#).

6.37.2.4 #define MAX_TOKEN_COUNT

Definition at line 122 of file [command-interpreter2.h](#).

6.37.2.5 #define emberCommandEntryAction(*name*, *action*, *argumentTypes*, *description*)

Definition at line 187 of file [command-interpreter2.h](#).

6.37.2.6 #define emberCommandEntryActionWithDetails(*name*, *action*, *argumentTypes*, *description*, *argumentDescriptionArray*)

Definition at line 190 of file [command-interpreter2.h](#).

6.37.2.7 #define emberCommandEntrySubMenu(*name*, *subMenu*, *description*)

Definition at line 198 of file [command-interpreter2.h](#).

6.37.2.8 #define emberCommandEntryTerminator()

Definition at line 202 of file [command-interpreter2.h](#).

6.37.2.9 #define EMBER_COMMAND_INTERPRETER_CONFIGURATION_ECHO

Definition at line 243 of file [command-interpreter2.h](#).

6.37.2.10 #define emberCopyKeyArgument(*index*, *keyDataPointer*)

A convenience macro for copying security key arguments to an [EmberKeyData](#) pointer.

Definition at line 329 of file [command-interpreter2.h](#).

6.37.2.11 #define emberCopyEui64Argument(*index*, *eui64*)

A convenience macro for copying eui64 arguments to an EmberEUI64.

Definition at line 336 of file [command-interpreter2.h](#).

6.37.2.12 #define emberGetEui64Argument(*index*, *eui64*)

A convenience macro for copying security key arguments to an [EmberKeyData](#) pointer.

Definition at line 338 of file [command-interpreter2.h](#).

6.37.2.13 #define emberProcessCommandInput(*port*)

Process input coming in on the given serial port.

Returns

true if an end of line character was read. If the application uses a command line prompt, this indicates it is time to print the prompt.

```
void emberProcessCommandInput(uint8_t port);
```

Definition at line 384 of file [command-interpreter2.h](#).

6.37.2.14 #define emberCommandInterpreterEchoOn()

Turn echo of command line on.

Definition at line 389 of file [command-interpreter2.h](#).

6.37.2.15 #define emberCommandInterpreterEchoOff()

Turn echo of command line off.

Definition at line 395 of file [command-interpreter2.h](#).

6.37.2.16 #define emberCommandInterpreterIsEchoOn()

Returns true if echo is on, false otherwise.

Definition at line 401 of file [command-interpreter2.h](#).

6.37.3 Typedef Documentation**6.37.3.1 typedef void(* CommandAction)(void)**

Definition at line 124 of file [command-interpreter2.h](#).

6.37.4 Enumeration Type Documentation

6.37.4.1 enum EmberCommandStatus

Command error states.

If you change this list, ensure you also change the strings that describe these errors in the array `emberCommandErrorNames[]` in `command-interpreter.c`.

Enumerator:

EMBER_CMD_SUCCESS
EMBER_CMD_ERR_PORT_PROBLEM
EMBER_CMD_ERR_NO_SUCH_COMMAND
EMBER_CMD_ERR_WRONG_NUMBER_OF_ARGUMENTS
EMBER_CMD_ERR_ARGUMENT_OUT_OF_RANGE
EMBER_CMD_ERR_ARGUMENT_SYNTAX_ERROR
EMBER_CMD_ERR_STRING_TOO_LONG
EMBER_CMD_ERR_INVALID_ARGUMENT_TYPE

Definition at line 251 of file [command-interpreter2.h](#).

6.37.5 Function Documentation

6.37.5.1 uint8_t emberCommandArgumentCount (void)

Returns the number of arguments for the current command.

6.37.5.2 uint32_t emberUnsignedCommandArgument (uint8_t argNum)

Retrieves unsigned integer arguments.

6.37.5.3 int32_t emberSignedCommandArgument (uint8_t argNum)

Retrieves signed integer arguments.

6.37.5.4 bool emberStringToHostOrderIpv4Address (const uint8_t * string, uint32_t * hostOrderIpv4Address)

Parses an IPv4 address string and returns a host order uint32_t. Returns true if address is valid dotted quad notation (A.B.C.D), false otherwise.

6.37.5.5 bool emberStringArgumentToHostOrderIpv4Address (uint8_t argNum, uint32_t * hostOrderIpv4Address)

Parses an IPv4 address string from a command argument and returns host order uint32_t. Returns true if address is valid dotted quad notation (A.B.C.D), false otherwise.

6.37.5.6 `uint8_t* emberStringCommandArgument (int8_t argNum, uint8_t * length)`

Retrieve quoted string or hex string arguments. Hex strings have already been converted into binary. To retrieve the name of the command itself, use an `argNum` of -1. For example, to retrieve the first character of the command, do: `uint8_t firstChar = emberStringCommandArgument(-1, NULL)[0]`. If the command is nested, an index of -2, -3, etc will work to retrieve the higher level command names. Note that [-1] only returns the text entered. If an abbreviated command name is entered only the text entered will be returned with [-1].

6.37.5.7 `const char* emberCommandName (void)`

A convenience macro for copying security key arguments to an [EmberKeyData](#) pointer.

6.37.5.8 `uint8_t emberCopyStringArgument (int8_t argNum, uint8_t * destination, uint8_t maxLength, bool leftPad)`

Copies the string argument to the given destination up to `maxLength`. If the argument length is nonzero but less than `maxLength` and `leftPad` is true, leading zeroes are prepended to bring the total length of the target up to `maxLength`. If the argument is longer than the `maxLength`, it is truncated to `maxLength`. Returns the minimum of the argument length and `maxLength`.

This function is commonly used for reading in hex strings such as EUI64 or key data and left padding them with zeroes. See [emberCopyKeyArgument](#) and [emberCopyEui64Argument](#) for convenience macros for this purpose.

6.37.5.9 `uint8_t emberCopyBigEndianEui64Argument (int8_t index, EmberEUI64 destination)`

Copies eui64 arguments in big-endian format to an `EmberEUI64`. This is useful because eui64s are often presented to users in big-endian format even though they are used in software in little-endian format.

6.37.5.10 `void emberCommandReaderSetDefaultBase (uint8_t base)`

6.37.5.11 `void emberCommandActionHandler (const CommandAction action)`

The application may implement this handler. To override the default handler, define `EMBER_APPLICATION_HAS_COMMAND_ACTION_HANDLER` in the `CONFIGURATION_HEADER`.

6.37.5.12 `void emberCommandErrorHandler (EmberCommandStatus status)`

The application may implement this handler. To override the default handler, define `EMBER_APPLICATION_HAS_COMMAND_ERROR_HANDLER` in the `CONFIGURATION_HEADER`. Defining this will also remove the help functions [emberPrintCommandUsage\(\)](#), [emberPrintCommandUsageNotes\(\)](#), and [emberPrintCommandTable\(\)](#).

6.37.5.13 void emberPrintCommandUsage (EmberCommandEntry * *entry*)

6.37.5.14 void emberPrintCommandUsageNotes (void)

6.37.5.15 void emberPrintCommandTable (void)

6.37.5.16 void emberCommandClearBuffer (void)

6.37.5.17 void emberCommandReaderInit (void)

Initialize the command interpreter.

6.37.5.18 bool emberProcessCommandString (uint8_t * *input*, uint8_t *sizeOrPort*)

Process the given string as a command.

6.37.6 Variable Documentation

6.37.6.1 EmberCommandEntry* emberCurrentCommand

A pointer to the currently matching command entry. Only valid from within a command function. If the original command was nested, points to the final (non-nested) command entry.

6.37.6.2 EmberCommandEntry emberCommandTable[]

6.37.6.3 uint8_t emberCommandInterpreter2Configuration

Configuration byte.

6.38 ZigBee Device Object (ZDO) Information

Macros

- #define [ZDO_MESSAGE_OVERHEAD](#)

Device Discovery Functions

- [EmberStatus emberNetworkAddressRequest](#) ([EmberEUI64](#) target, bool reportKids, uint8_t childStartIndex)
- [EmberStatus emberIeeeAddressRequest](#) ([EmberNodeId](#) target, bool reportKids, uint8_t childStartIndex, [EmberApsOption](#) options)

Service Discovery Functions

- [EmberStatus ezspMatchDescriptorsRequest](#) ([EmberNodeId](#) target, uint16_t profile, uint8_t inCount, uint8_t outCount, uint16_t *inClusters, uint16_t *outClusters, [EmberApsOption](#) options)

Binding Manager Functions

- [EmberStatus ezspEndDeviceBindRequest](#) ([EmberNodeId](#) localNodeId, [EmberEUI64](#) localEui64, uint8_t endpoint, uint16_t profile, uint8_t inCount, uint8_t outCount, uint16_t *inClusters, uint16_t *outClusters, [EmberApsOption](#) options)

Function to Decode Address Response Messages

- [EmberNodeId ezspDecodeAddressResponse](#) (uint8_t *response, [EmberEUI64](#) eui64-Return)

Service Discovery Functions

- [EmberStatus emberNodeDescriptorRequest](#) ([EmberNodeId](#) target, [EmberApsOption](#) options)
- [EmberStatus emberPowerDescriptorRequest](#) ([EmberNodeId](#) target, [EmberApsOption](#) options)
- [EmberStatus emberSimpleDescriptorRequest](#) ([EmberNodeId](#) target, uint8_t targetEndpoint, [EmberApsOption](#) options)
- [EmberStatus emberActiveEndpointsRequest](#) ([EmberNodeId](#) target, [EmberApsOption](#) options)

Binding Manager Functions

- [EmberStatus emberBindRequest](#) ([EmberNodeId](#) target, [EmberEUI64](#) source, uint8_t sourceEndpoint, uint16_t clusterId, uint8_t type, [EmberEUI64](#) destination, [EmberMulticastId](#) groupAddress, uint8_t destinationEndpoint, [EmberApsOption](#) options)
- [EmberStatus emberUnbindRequest](#) ([EmberNodeId](#) target, [EmberEUI64](#) source, uint8_t sourceEndpoint, uint16_t clusterId, uint8_t type, [EmberEUI64](#) destination, [EmberMulticastId](#) groupAddress, uint8_t destinationEndpoint, [EmberApsOption](#) options)

Node Manager Functions

- [EmberStatus emberLqiTableRequest](#) ([EmberNodeId](#) target, [uint8_t](#) startIndex, [EmberApsOption](#) options)
- [EmberStatus emberRoutingTableRequest](#) ([EmberNodeId](#) target, [uint8_t](#) startIndex, [EmberApsOption](#) options)
- [EmberStatus emberBindingTableRequest](#) ([EmberNodeId](#) target, [uint8_t](#) startIndex, [EmberApsOption](#) options)
- [EmberStatus emberLeaveRequest](#) ([EmberNodeId](#) target, [EmberEUI64](#) deviceAddress, [uint8_t](#) leaveRequestFlags, [EmberApsOption](#) options)
- [EmberStatus emberPermitJoiningRequest](#) ([EmberNodeId](#) target, [uint8_t](#) duration, [uint8_t](#) authentication, [EmberApsOption](#) options)
- [void emberSetZigDevRequestRadius](#) ([uint8_t](#) radius)
- [uint8_t emberGetZigDevRequestRadius](#) ([void](#))
- [uint8_t emberGetLastZigDevRequestSequence](#) ([void](#))
- [uint8_t emberGetLastAppZigDevRequestSequence](#) ([void](#))

6.38.1 Detailed Description

For getting information about nodes of a ZigBee network via a ZigBee Device Object (ZDO). See [zigbee-device-host.h](#) and [zigbee-device-common.h](#) for source code.

The ZDO library provides functions that construct and send several common ZDO requests. It also provides a function for extracting the two addresses from a ZDO address response. The format of all the ZDO requests and responses that the stack supports is described in [stack/include/zigbee-device-stack.h](#). Since the library doesn't handle all of these requests and responses, the application must construct any other requests it wishes to send and decode any other responses it wishes to receive.

The request sending functions do the following:

1. Construct a correctly formatted payload buffer.
2. Fill in the APS frame with the correct values.
3. Send the message by calling either `::ezspSendBroadcast()` or `::ezspSendUnicast()`.

The result of the send is reported to the application as normal via `::ezspMessageSentHandler()`.

The following code shows an example of an application's use of [emberSimpleDescriptorRequest\(\)](#). The command interpreter would call this function and supply the arguments.

```
void sendSimpleDescriptorRequest(EmberCommandState *state)
{
    EmberNodeId target = emberUnsignedCommandArgument
        (state, 0);
    uint8_t targetEndpoint = emberUnsignedCommandArgument
        (state, 1);
    if (emberSimpleDescriptorRequest(target,
                                    targetEndpoint,
                                    EMBER_APS_OPTION_NONE)
        != EMBER_SUCCESS) {
        emberSerialPrintf(SERIAL_PORT, "
        emberSimpleDescriptorRequest failed\r\n");
    }
}
```

The following code shows an example of an application's use of [ezspDecodeAddressResponse\(\)](#).

```

void ezspIncomingMessageHandler(EmberIncomingMessageType
    type,
                                EmberApsFrame *apsFrame,
                                uint8_t lastHopLqi,
                                int8_t lastHopRssi,
                                EmberNodeId sender,
                                uint8_t bindingIndex,
                                uint8_t addressIndex,
                                uint8_t messageLength,
                                uint8_t *messageContents)
{
    if (apsFrame->profileId == EMBER_ZDO_PROFILE_ID)
    {
        switch (apsFrame->clusterId) {
            case NETWORK_ADDRESS_RESPONSE:
            case IEEE_ADDRESS_RESPONSE:
            {
                EmberEUI64 eui64;
                EmberNodeId nodeId = ezspDecodeAddressResponse
                    (messageContents,
                     eui64);

                // Use nodeId and eui64 here.
                break;
            }
            default:
                // Handle other incoming ZDO responses here.
        }
    }
    else {
        // Handle incoming application messages here.
    }
}

```

6.38.2 Macro Definition Documentation

6.38.2.1 #define ZDO_MESSAGE_OVERHEAD

ZDO messages start with a sequence number.

Definition at line 16 of file [zigbee-device-common.h](#).

6.38.3 Function Documentation

6.38.3.1 EmberStatus emberNetworkAddressRequest (EmberEUI64 target, bool reportKids, uint8_t childStartIndex)

Request the 16 bit network address of a node whose EUI64 is known.

Parameters

<i>target</i>	The EUI64 of the node.
<i>reportKids</i>	true to request that the target list their children in the response.
<i>childStart-Index</i>	The index of the first child to list in the response. Ignored if <i>reportKids</i> is false.

Returns

An [EmberStatus](#) value.

- [EMBER_SUCCESS](#) - The request was transmitted successfully.
- [EMBER_NO_BUFFERS](#) - Insufficient message buffers were available to construct the request.
- [EMBER_NETWORK_DOWN](#) - The node is not part of a network.
- [EMBER_NETWORK_BUSY](#) - Transmission of the request failed.

6.38.3.2 EmberStatus emberleeeeAddressRequest (EmberNodeId *target*, bool *reportKids*, uint8_t *childStartIndex*, EmberApsOption *options*)

Request the EUI64 of a node whose 16 bit network address is known.

Parameters

<i>target</i>	The network address of the node.
<i>reportKids</i>	true to request that the target list their children in the response.
<i>childStartIndex</i>	The index of the first child to list in the response. Ignored if <i>reportKids</i> is false.
<i>options</i>	The options to use when sending the request. See ::emberSendUnicast() for a description.

Returns

An [EmberStatus](#) value.

- [EMBER_SUCCESS](#)
- [EMBER_NO_BUFFERS](#)
- [EMBER_NETWORK_DOWN](#)
- [EMBER_NETWORK_BUSY](#)

6.38.3.3 EmberStatus ezspMatchDescriptorsRequest (EmberNodeId *target*, uint16_t *profile*, uint8_t *inCount*, uint8_t *outCount*, uint16_t * *inClusters*, uint16_t * *outClusters*, EmberApsOption *options*)

Request the specified node to send a list of its endpoints that match the specified application profile and, optionally, lists of input and/or output clusters.

Parameters

<i>target</i>	The node whose matching endpoints are desired. The request can be sent unicast or broadcast ONLY to the "RX-on-when-idle-address" (0xFFFD) If sent as a broadcast, any node that has matching endpoints will send a response.
<i>profile</i>	The application profile to match.
<i>inCount</i>	The number of input clusters. To not match any input clusters, set this value to 0.
<i>outCount</i>	The number of output clusters. To not match any output clusters, set this value to 0.
<i>inClusters</i>	The list of input clusters.
<i>outClusters</i>	The list of output clusters.
<i>options</i>	The options to use when sending the unicast request. See emberSendUnicast() for a description. This parameter is ignored if the target is a broadcast address.

Returns

An [EmberStatus](#) value. [EMBER_SUCCESS](#), [EMBER_NO_BUFFERS](#), [EMBER_NETWORK_DOWN](#) or [EMBER_NETWORK_BUSY](#).

6.38.3.4 `EmberStatus ezspEndDeviceBindRequest (EmberNodeId localNodeId,
EmberEUI64 localEui64, uint8_t endpoint, uint16_t profile, uint8_t
inCount, uint8_t outCount, uint16_t * inClusters, uint16_t * outClusters,
EmberApsOption options)`

An end device bind request to the coordinator. If the coordinator receives a second end device bind request then a binding is created for every matching cluster.

Parameters

<i>localNodeId</i>	The node ID of the local device.
<i>localEui64</i>	The EUI64 of the local device.
<i>endpoint</i>	The endpoint to be bound.
<i>profile</i>	The application profile of the endpoint.
<i>inCount</i>	The number of input clusters.
<i>outCount</i>	The number of output clusters.
<i>inClusters</i>	The list of input clusters.
<i>outClusters</i>	The list of output clusters.
<i>options</i>	The options to use when sending the request. See emberSendUnicast() for a description.

Returns

An EmberStatus value. EMBER_SUCCESS, EMBER_NO_BUFFERS, EMBER_NETWORK_DOWN or EMBER_NETWORK_BUSY.

6.38.3.5 `EmberNodeId ezspDecodeAddressResponse (uint8_t * response,
EmberEUI64 eui64Return)`

Extracts the EUI64 and the node ID from an address response message.

Parameters

<i>response</i>	The received ZDO message with cluster ID NETWORK_ADDRESS_RESPONSE or IEEE_ADDRESS_RESPONSE.
<i>eui64Return</i>	The EUI64 from the response is copied here.

Returns

Returns the node ID from the response if the response status was EMBER_ZDP_SUCCESS. Otherwise, returns EMBER_NULL_NODE_ID.

6.38.3.6 `EmberStatus emberNodeDescriptorRequest (EmberNodeId target,
EmberApsOption options)`

Request the specified node to send its node descriptor. The node descriptor contains information about the capabilities of the ZigBee node. It describes logical type, APS flags, frequency band, MAC capabilities flags, manufacturer code and maximum buffer size. It is defined in the ZigBee Application Framework Specification.

Parameters

<i>target</i>	The node whose node descriptor is desired.
<i>options</i>	The options to use when sending the request. See emberSendUnicast() for a description.

Returns

An [EmberStatus](#) value. [EMBER_SUCCESS](#), [EMBER_NO_BUFFERS](#), [EMBER_N-ETWORK_DOWN](#) or [EMBER_NETWORK_BUSY](#).

6.38.3.7 [EmberStatus emberPowerDescriptorRequest](#) ([EmberNodeId](#) *target*, [EmberApsOption](#) *options*)

Request the specified node to send its power descriptor. The power descriptor gives a dynamic indication of the power status of the node. It describes current power mode, available power sources, current power source and current power source level. It is defined in the ZigBee Application Framework Specification.

Parameters

<i>target</i>	The node whose power descriptor is desired.
<i>options</i>	The options to use when sending the request. See emberSendUnicast() for a description.

Returns

An [EmberStatus](#) value. [EMBER_SUCCESS](#), [EMBER_NO_BUFFERS](#), [EMBER_N-ETWORK_DOWN](#) or [EMBER_NETWORK_BUSY](#).

6.38.3.8 [EmberStatus emberSimpleDescriptorRequest](#) ([EmberNodeId](#) *target*, [uint8_t](#) *targetEndpoint*, [EmberApsOption](#) *options*)

Request the specified node to send the simple descriptor for the specified endpoint. The simple descriptor contains information specific to a single endpoint. It describes the application profile identifier, application device identifier, application device version, application flags, application input clusters and application output clusters. It is defined in the ZigBee Application Framework Specification.

Parameters

<i>target</i>	The node of interest.
<i>target-Endpoint</i>	The endpoint on the target node whose simple descriptor is desired.
<i>options</i>	The options to use when sending the request. See emberSendUnicast() for a description.

Returns

An [EmberStatus](#) value. [EMBER_SUCCESS](#), [EMBER_NO_BUFFERS](#), [EMBER_N-ETWORK_DOWN](#) or [EMBER_NETWORK_BUSY](#).

6.38.3.9 EmberStatus emberActiveEndpointsRequest (EmberNodeId *target*, EmberApsOption *options*)

Request the specified node to send a list of its active endpoints. An active endpoint is one for which a simple descriptor is available.

Parameters

<i>target</i>	The node whose active endpoints are desired.
<i>options</i>	The options to use when sending the request. See emberSendUnicast() for a description.

Returns

An EmberStatus value. [EMBER_SUCCESS](#), [EMBER_NO_BUFFERS](#), [EMBER_N-ETWORK_DOWN](#) or [EMBER_NETWORK_BUSY](#).

6.38.3.10 EmberStatus emberBindRequest (EmberNodeId *target*, EmberEUI64 *source*, uint8_t *sourceEndpoint*, uint16_t *clusterId*, uint8_t *type*, EmberEUI64 *destination*, EmberMulticastId *groupAddress*, uint8_t *destinationEndpoint*, EmberApsOption *options*)

Send a request to create a binding entry with the specified contents on the specified node.

Parameters

<i>target</i>	The node on which the binding will be created.
<i>source</i>	The source EUI64 in the binding entry.
<i>source-Endpoint</i>	The source endpoint in the binding entry.
<i>clusterId</i>	The cluster ID in the binding entry.
<i>type</i>	The type of binding, either UNICAST_BINDING , MULTICAST_BINDING , or UNICAST_MANY_TO_ONE_BINDING . UNICAST_MANY_TO_ONE_BINDING is an Ember-specific extension and should be used only when the target is an Ember device.
<i>destination</i>	The destination EUI64 in the binding entry for UNICAST_BINDING or UNICAST_MANY_TO_ONE_BINDING .
<i>group-Address</i>	The group address for the MULTICAST_BINDING .
<i>destination-Endpoint</i>	The destination endpoint in the binding entry for the UNICAST_BINDING or UNICAST_MANY_TO_ONE_BINDING .
<i>options</i>	The options to use when sending the request. See emberSendUnicast() for a description.

Returns

An EmberStatus value. [EMBER_SUCCESS](#), [EMBER_NO_BUFFERS](#), [EMBER_N-ETWORK_DOWN](#) or [EMBER_NETWORK_BUSY](#).

6.38.3.11 `EmberStatus emberUnbindRequest (EmberNodeId target, EmberEUI64 source, uint8_t sourceEndpoint, uint16_t clusterId, uint8_t type, EmberEUI64 destination, EmberMulticastId groupAddress, uint8_t destinationEndpoint, EmberApsOption options)`

Send a request to remove a binding entry with the specified contents from the specified node.

Parameters

<i>target</i>	The node on which the binding will be removed.
<i>source</i>	The source EUI64 in the binding entry.
<i>source-Endpoint</i>	The source endpoint in the binding entry.
<i>clusterId</i>	The cluster ID in the binding entry.
<i>type</i>	The type of binding, either UNICAST_BINDING , MULTICAST_BINDING , or UNICAST_MANY_TO_ONE_BINDING . UNICAST_MANY_TO_ONE_BINDING is an Ember-specific extension and should be used only when the target is an Ember device.
<i>destination</i>	The destination EUI64 in the binding entry for the UNICAST_BINDING or UNICAST_MANY_TO_ONE_BINDING .
<i>group-Address</i>	The group address for the MULTICAST_BINDING .
<i>destination-Endpoint</i>	The destination endpoint in the binding entry for the UNICAST_BINDING or UNICAST_MANY_TO_ONE_BINDING .
<i>options</i>	The options to use when sending the request. See emberSendUnicast() for a description.

Returns

An [EmberStatus](#) value.

- [EMBER_SUCCESS](#)
- [EMBER_NO_BUFFERS _ EMBER_NETWORK_DOWN](#)
- [EMBER_NETWORK_BUSY](#)

6.38.3.12 `EmberStatus emberLqiTableRequest (EmberNodeId target, uint8_t startIndex, EmberApsOption options)`

Request the specified node to send its LQI (neighbor) table. The response gives PAN ID, EUI64, node ID and cost for each neighbor. The EUI64 is only available if security is enabled. The other fields in the response are set to zero. The response format is defined in the ZigBee Device Profile Specification.

Parameters

<i>target</i>	The node whose LQI table is desired.
<i>startIndex</i>	The index of the first neighbor to include in the response.
<i>options</i>	The options to use when sending the request. See emberSendUnicast() for a description.

Returns

An EmberStatus value. [EMBER_SUCCESS](#), [EMBER_NO_BUFFERS](#), [EMBER_NETWORK_DOWN](#) or [EMBER_NETWORK_BUSY](#).

6.38.3.13 EmberStatus emberRoutingTableRequest (EmberNodeId *target*, uint8_t *startIndex*, EmberApsOption *options*)

Request the specified node to send its routing table. The response gives destination node ID, status and many-to-one flags, and the next hop node ID. The response format is defined in the ZigBee Device Profile Specification.

Parameters

<i>target</i>	The node whose routing table is desired.
<i>startIndex</i>	The index of the first route entry to include in the response.
<i>options</i>	The options to use when sending the request. See emberSendUnicast() for a description.

Returns

An EmberStatus value. [EMBER_SUCCESS](#), [EMBER_NO_BUFFERS](#), [EMBER_NETWORK_DOWN](#) or [EMBER_NETWORK_BUSY](#).

6.38.3.14 EmberStatus emberBindingTableRequest (EmberNodeId *target*, uint8_t *startIndex*, EmberApsOption *options*)

Request the specified node to send its nonvolatile bindings. The response gives source address, source endpoint, cluster ID, destination address and destination endpoint for each binding entry. The response format is defined in the ZigBee Device Profile Specification. Note that bindings that have the Ember-specific [UNICAST_MANY_TO_ONE_BINDING](#) type are reported as having the standard [UNICAST_BINDING](#) type.

Parameters

<i>target</i>	The node whose binding table is desired.
<i>startIndex</i>	The index of the first binding entry to include in the response.
<i>options</i>	The options to use when sending the request. See emberSendUnicast() for a description.

Returns

An EmberStatus value. [EMBER_SUCCESS](#), [EMBER_NO_BUFFERS](#), [EMBER_NETWORK_DOWN](#) or [EMBER_NETWORK_BUSY](#).

6.38.3.15 EmberStatus emberLeaveRequest (EmberNodeId *target*, EmberEUI64 *deviceAddress*, uint8_t *leaveRequestFlags*, EmberApsOption *options*)

Request the specified node to remove the specified device from the network. The device to be removed must be the node to which the request is sent or one of its children.

Parameters

<i>target</i>	The node which will remove the device.
<i>device-Address</i>	All zeros if the target is to remove itself from the network or the EUI64 of a child of the target device to remove that child.
<i>leave-Request-Flags</i>	A bitmask of leave options. Include LEAVE_REQUEST_REMOVE_CHILDREN_FLAG if the target is to remove their children and/or LEAVE_REQUEST_REJOIN_FLAG if the target is to rejoin the network immediately after leaving.
<i>options</i>	The options to use when sending the request. See emberSendUnicast() for a description.

Returns

An EmberStatus value. [EMBER_SUCCESS](#), [EMBER_NO_BUFFERS](#), [EMBER_NETWORK_DOWN](#) or [EMBER_NETWORK_BUSY](#).

6.38.3.16 EmberStatus emberPermitJoiningRequest (EmberNodeId *target*, uint8_t *duration*, uint8_t *authentication*, EmberApsOption *options*)

Request the specified node to allow or disallow association.

Parameters

<i>target</i>	The node which will allow or disallow association. The request can be broadcast by using a broadcast address (0xFFFC/0xFFFD/0xFFFF). No response is sent if the request is broadcast.
<i>duration</i>	A value of 0x00 disables joining. A value of 0xFF enables joining. Any other value enables joining for that number of seconds.
<i>authentication</i>	Controls Trust Center authentication behavior.
<i>options</i>	The options to use when sending the request. See emberSendUnicast() for a description. This parameter is ignored if the target is a broadcast address.

Returns

An EmberStatus value. [EMBER_SUCCESS](#), [EMBER_NO_BUFFERS](#), [EMBER_NETWORK_DOWN](#) or [EMBER_NETWORK_BUSY](#).

6.38.3.17 void emberSetZigDevRequestRadius (uint8_t *radius*)

Change the default radius for broadcast ZDO requests.

Parameters

<i>radius</i>	The radius to be used for future ZDO request broadcasts.
---------------	--

6.38.3.18 uint8_t emberGetZigDevRequestRadius (void)

Retrieve the default radius for broadcast ZDO requests.

Returns

The radius to be used for future ZDO request broadcasts.

6.38.3.19 uint8_t emberGetLastZigDevRequestSequence (void)

Provide access to the application ZDO transaction sequence number for last request. This function has been deprecated and replaced by [emberGetLastAppZigDevRequestSequence\(\)](#).

Returns

Last application ZDO transaction sequence number used

6.38.3.20 uint8_t emberGetLastAppZigDevRequestSequence (void)

Provide access to the application ZDO transaction sequence number for last request.

Returns

Last application ZDO transaction sequence number used

6.39 Message Fragmentation

Initialization

- void `ezspFragmentInit` (uint16_t receiveBufferLength, uint8_t *receiveBuffer)

Transmitting

- `EmberStatus ezspFragmentSendUnicast` (`EmberOutgoingMessageType` type, uint16_t indexOrDestination, `EmberApsFrame` *apsFrame, uint8_t maxFragmentSize, uint16_t messageLength, uint8_t *messageContents)
- `EmberStatus ezspFragmentSourceRouteHandler` (void)
- bool `ezspFragmentMessageSent` (`EmberApsFrame` *apsFrame, `EmberStatus` status)
- void `ezspFragmentMessageSentHandler` (`EmberStatus` status)

Receiving

- bool `ezspFragmentIncomingMessage` (`EmberApsFrame` *apsFrame, `EmberNodeId` sender, uint16_t *messageLength, uint8_t **messageContents)
- void `ezspFragmentTick` (void)

6.39.1 Detailed Description

Fragmented message support for EZSP Hosts. Splits long messages into smaller blocks for transmission and reassembles received blocks. See `fragment-host.c` for source code.

::EZSP_CONFIG_FRAGMENT_WINDOW_SIZE controls how many blocks are sent at a time. ::EZSP_CONFIG_FRAGMENT_DELAY_MS controls the spacing between blocks.

Before calling any of the other functions listed here, the application must call `ezspFragmentInit()`.

To send a long message, the application calls `ezspFragmentSendUnicast()`. The application must add a call to `ezspFragmentMessageSent()` at the start of its `ezspMessageSentHandler()`. If `ezspFragmentMessageSent()` returns true, the fragmentation code has handled the event and the application must not process it further. The fragmentation code calls the application-defined `ezspFragmentMessageSentHandler()` when it has finished sending the long message.

To receive a long message, the application must add a call to `ezspFragmentIncomingMessage()` at the start of its `ezspIncomingMessageHandler()`. If `ezspFragmentIncomingMessage()` returns true, the fragmentation code has handled the message and the application must not process it further. The application must also call `ezspFragmentTick()` regularly.

6.39.2 Function Documentation

6.39.2.1 void ezspFragmentInit (uint16_t receiveBufferLength, uint8_t * receiveBuffer)

Initialize variables and buffers used for sending and receiving long messages. This function reads the values of ::EZSP_CONFIG_MAX_HOPS and ::EZSP_CONFIG_FRAGMENT_WINDOW_SIZE. The application must set these values before calling this function.

Parameters

<i>receive-BufferLength</i>	The length of receiveBuffer. Incoming messages longer than this will be dropped.
<i>receiveBuffer</i>	The buffer used to reassemble incoming long messages. Once the message is complete, this buffer will be passed back to the application by ezspFragmentIncomingMessage() .

6.39.2.2 EmberStatus ezspFragmentSendUnicast (EmberOutgoingMessageType type, uint16_t indexOrDestination, EmberApsFrame * apsFrame, uint8_t maxFragmentSize, uint16_t messageLength, uint8_t * messageContents)

Sends a long message by splitting it into blocks. Only one long message can be sent at a time. Calling this function a second time aborts the first message.

Parameters

<i>type</i>	Specifies the outgoing message type. Must be one of EMBER_OUTGOING_DIRECT , EMBER_OUTGOING_VIA_ADDRESS_TABLE , or EMBER_OUTGOING_VIA_BINDING .
<i>indexOrDestination</i>	Depending on the type of addressing used, this is either the EmberNodeId of the destination, an index into the address table, or an index into the binding table.
<i>apsFrame</i>	The APS frame for the message.
<i>max-Fragment-Size</i>	The message will be broken into blocks no larger than this.
<i>message-Length</i>	The length of the messageContents parameter in bytes.
<i>message-Contents</i>	The long message to be sent.

Returns

An EmberStatus value.

- [EMBER_SUCCESS](#)
- [EMBER_MESSAGE_TOO_LONG](#)
- [EMBER_NETWORK_DOWN](#)
- [EMBER_NETWORK_BUSY](#)
- [EMBER_INVALID_CALL](#) is returned if messageLength is zero or if the window size (::EZSP_CONFIG_FRAGMENT_WINDOW_SIZE) is zero.

6.39.2.3 EmberStatus ezspFragmentSourceRouteHandler (void)

A callback invoked just before each block of the current long message is sent. If the message is to be source routed, the application must define this callback and call ezspSetSourceRoute() in it.

The application must define EZSP_APPLICATION_HAS_FRAGMENT_SOURCE_ROUTE_HANDLER in its configuration header if it defines this callback.

Returns

[EMBER_SUCCESS](#) if the source route has been set. Any other value will abort transmission of the current long message.

6.39.2.4 **bool ezspFragmentMessageSent (EmberApsFrame * *apsFrame*, EmberStatus *status*)**

The application must call this function at the start of its `ezspMessageSentHandler()`. If it returns true, the fragmentation code has handled the event and the application must not process it further.

Parameters

<i>apsFrame</i>	The APS frame passed to <code>ezspMessageSentHandler()</code> .
<i>status</i>	The status passed to <code>ezspMessageSentHandler()</code> .

Returns

true if the sent message was a block of a long message. The fragmentation code has handled the event so the application must return immediately from its `ezspMessageSentHandler()`. Returns false otherwise. The fragmentation code has not handled the event so the application must continue to process it.

6.39.2.5 **void ezspFragmentMessageSentHandler (EmberStatus *status*)**

The fragmentation code calls this application-defined handler when it finishes sending a long message.

Parameters

<i>status</i>	EMBER_SUCCESS if all the blocks of the long message were delivered to the destination, otherwise EMBER_DELIVERY_FAILED , EMBER_NETWORK_DOWN or EMBER_NETWORK_BUSY .
---------------	---

6.39.2.6 **bool ezspFragmentIncomingMessage (EmberApsFrame * *apsFrame*, EmberNodeId *sender*, uint16_t * *messageLength*, uint8_t ** *messageContents*)**

The application must call this function at the start of its `ezspIncomingMessageHandler()`. If it returns true, the fragmentation code has handled the message and the application must not process it further. When the final block of a long message is received, this function replaces the message with the reassembled long message and returns false so that the application processes it.

Parameters

<i>apsFrame</i>	The APS frame passed to <code>ezspIncomingMessageHandler()</code> .
<i>sender</i>	The sender passed to <code>ezspIncomingMessageHandler()</code> .
<i>messageLength</i>	A pointer to the message length passed to <code>ezspIncomingMessageHandler()</code> .

<i>message-Contents</i>	A pointer to the message contents passed to ezspIncomingMessageHandler().
-------------------------	---

Returns

true if the incoming message was a block of an incomplete long message. The fragmentation code has handled the message so the application must return immediately from its ezspIncomingMessageHandler(). Returns false if the incoming message was not part of a long message. The fragmentation code has not handled the message so the application must continue to process it. Returns false if the incoming message was a block that completed a long message. The fragmentation code replaces the message with the reassembled long message so the application must continue to process it.

6.39.2.7 void ezspFragmentTick (void)

Used by the fragmentation code to time incoming blocks. The application must call this function regularly.

6.40 Network Manager

Macros

- `#define NM_WARNING_LIMIT`
- `#define NM_WINDOW_SIZE`
- `#define NM_CHANNEL_MASK`
- `#define NM_WATCHLIST_SIZE`

Functions

- `void nmUtilWarningHandler (void)`
- `bool nmUtilProcessIncoming (EmberApsFrame *apsFrame, uint8_t messageLength, uint8_t *message)`
- `EmberStatus nmUtilChangeChannelRequest (void)`

6.40.1 Detailed Description

The network manager is an optional function of one device in the ZigBee network. Devices on the network send unsolicited ZDO energy scan reports to the network manager when more than 25% of unicasts fail within a rolling window, but no more than once every 15 minutes.

See [network-manager.h](#) for source code.

The network manager is the coordinator by default but can be changed via `emberSetNetworkManagerRequest()`. It processes the energy scan reports from the devices on the network, and is responsible for determining if the network should change channels in an attempt to resolve reliability problems that might be caused by RF interference.

Note that EmberZNet networks are quite robust to many interferers such as 802.11 (WiFi), and the presence of interferers does not necessarily degrade application performance or require a channel change. Because changing channels is disruptive to network operation, channel changes should not be done solely because of observed higher noise levels, as the noise may not be causing any problem.

Also note that receipt of unsolicited scan reports is only an indication of unicast failures in the network. These might be caused by RF interference, or for some other reason such as a device failure. In addition, only the application can tell whether the delivery failures caused an actual problem for the application. In general, it is difficult to automatically determine with certainty that network problems are caused by RF interference. Channel changes should therefore be done sparingly and with careful application design.

The stack provides three APIs in `include/zigbee-device-stack.h`:

- `emberEnergyScanRequest`
- `emberSetNetworkManagerRequest`
- `emberChannelChangeRequest`

This library provides some additional functions:

- `nmUtilProcessIncomingMessage`

- nmUtilWarningHandler
- nmUtilChangeChannelRequest

An application implementing network manager functionality using this library should pass all incoming messages to nmUtilProcessIncomingMessage, which will return true if the message was processed as a ZDO energy scan report. The application should not make any calls to emberEnergyScanRequest(), as the library assumes all incoming scan reports are unsolicited and indicate unicast failures.

When NM_WARNING_LIMIT reports have been processed within NM_WINDOW_SIZE minutes, the nmUtilWarningHandler callback, which must be implemented by the application, is invoked. The default values for these parameters are set in [network-manager.h](#) and may be modified using #defines within the application configuration header.

The application may use the nmUtilWarningHandler callback, along with other application-specific information, to decide if and when to change the channel by calling nmUtilChangeChannelRequest. This function chooses a new channel from the NM_CHANNEL_MASK parameter using information gathered over time.

In the event of a network-wide channel change, it is possible that some devices, especially sleepy end devices, do not receive the broadcast and remain on the old channel. Devices should use the API emberFindAndRejoinNetwork to get back to the right channel.

Two implementations of this library are provided: network-manager.c, and network-manager-lite.c. The former keeps track of the mean and deviation of the energy on each channel and uses these stats to choose the channel to change to. This consumes a fair amount of RAM. The latter takes the simpler (and possibly more effective) approach of just avoiding past bad channels. Application developers are encouraged to use and modify either of these solutions to take into account their own application-specific needs.

6.40.2 Macro Definition Documentation

6.40.2.1 #define NM_WARNING_LIMIT

Definition at line 97 of file [network-manager.h](#).

6.40.2.2 #define NM_WINDOW_SIZE

Definition at line 101 of file [network-manager.h](#).

6.40.2.3 #define NM_CHANNEL_MASK

Definition at line 107 of file [network-manager.h](#).

6.40.2.4 #define NM_WATCHLIST_SIZE

Definition at line 113 of file [network-manager.h](#).

6.40.3 Function Documentation

6.40.3.1 void nmUtilWarningHandler (void)

callback called when unsolicited scan reports hit limit. This callback must be implemented by the application. It is called when the number of unsolicited scan reports received within NM_WINDOW_LIMIT minutes reaches NM_WARNING_LIMIT.

6.40.3.2 bool nmUtilProcessIncoming (EmberApsFrame * *apsFrame*, uint8_t *messageLength*, uint8_t * *message*)

Called from the app in emberIncomingMessageHandler. Returns true if and only if the library processed the message.

Parameters

<i>apsFrame</i>	
<i>messageLength</i>	
<i>message</i>	

6.40.3.3 EmberStatus nmUtilChangeChannelRequest (void)

Chooses a new channel and broadcasts a ZDO channel change request.

6.41 Serial Communication

Functions

- [EmberStatus emberSerialInit](#) (uint8_t port, [SerialBaudRate](#) rate, SerialParity parity, uint8_t stopBits)
- uint16_t [emberSerialReadAvailable](#) (uint8_t port)
- [EmberStatus emberSerialReadByte](#) (uint8_t port, uint8_t *dataByte)
- [EmberStatus emberSerialReadData](#) (uint8_t port, uint8_t *data, uint16_t length, uint16_t *bytesRead)
- [EmberStatus emberSerialReadDataTimeout](#) (uint8_t port, uint8_t *data, uint16_t length, uint16_t *bytesRead, uint16_t firstByteTimeout, uint16_t subsequentByteTimeout)
- [EmberStatus emberSerialReadLine](#) (uint8_t port, char *data, uint8_t max)
- [EmberStatus emberSerialReadPartialLine](#) (uint8_t port, char *data, uint8_t max, uint8_t *index)
- uint16_t [emberSerialWriteAvailable](#) (uint8_t port)
- uint16_t [emberSerialWriteUsed](#) (uint8_t port)
- [EmberStatus emberSerialWriteByte](#) (uint8_t port, uint8_t dataByte)
- [EmberStatus emberSerialWriteHex](#) (uint8_t port, uint8_t dataByte)
- [EmberStatus emberSerialWriteString](#) (uint8_t port, PGM_P string)
- XAP2B_PAGEZERO_ON [EmberStatus emberSerialPrintf](#) (uint8_t port, PGM_P formatString,...)
- XAP2B_PAGEZERO_OFF
XAP2B_PAGEZERO_ON [EmberStatus emberSerialPrintfLine](#) (uint8_t port, PGM_P formatString,...)
- XAP2B_PAGEZERO_OFF
XAP2B_PAGEZERO_ON [EmberStatus emberSerialPrintCarriageReturn](#) (uint8_t port)
- XAP2B_PAGEZERO_OFF [EmberStatus emberSerialPrintfVarArg](#) (uint8_t port, PGM_P formatString, va_list ap)
- [EmberStatus emberSerialWriteData](#) (uint8_t port, uint8_t *data, uint8_t length)
- XAP2B_PAGEZERO_ON [EmberStatus emberSerialWaitSend](#) (uint8_t port)
- XAP2B_PAGEZERO_OFF [EmberStatus emberSerialGuaranteedPrintf](#) (uint8_t port, PGM_P formatString,...)
- void [emberSerialBufferTick](#) (void)
- void [emberSerialFlushRx](#) (uint8_t port)
- bool [emberSerialUnused](#) (uint8_t port)

6.41.1 Detailed Description

Unless otherwise noted, the EmberNet stack does not use these functions, and therefore the HAL is not required to implement them. However, many of the supplied example applications do use them. On some platforms, they are also required by DEBUG builds of the stack

Many of these functions return an [EmberStatus](#) value. See `stack/include/error-defs.h` for definitions of all [EmberStatus](#) return values. See `app/util/serial/serial.h` for source code. To use these serial routines, they must be properly configured.

If the Ember serial library is built using `EMBER_SERIAL_USE_STDIO`, then the Ember serial code will redirect to `stdio.h`. `EMBER_SERIAL_USE_STDIO` will not consume any

of the usual Ember serial library buffers and does not require use of any of the other EMBER_SERIALx definitions described here. In this mode, the only required lower layers are:

- putchar()
- getchar()
- fflush(stdout)
- [halInternalUartInit\(\)](#)
- [halInternalPrintfWriteAvailable\(\)](#)
- [halInternalPrintfReadAvailable\(\)](#)
- [halInternalForcePrintf\(\)](#)

The functions can work in two ways, depending on how messages waiting for transmission are stored:

- Buffered mode: Uses stack linked buffers. This method can be more efficient if many messages received over the air also need to be transmitted over the serial interface.
- FIFO mode: Uses a statically allocated queue of bytes, and data to be transmitted is copied into the queue.

(These modes deal only with data transmission. Data **reception** always occurs in a FIFO mode.)

The current version of these sources provides support for as many as two serial ports, but it can be easily extended. The ports are numbered 0 and 1 and should be accessed using those numbers. The ports can be set up independently of each other.

To enable a port, a Use mode (buffered or FIFO) and a Queue Size must be declared on the port. In FIFO mode, the Queue Size is the size of the FIFO and represents the number of bytes that can be waiting for transmission at any given time. In buffered mode, the Queue Size represents the number of whole messages that can be waiting for transmission at any given time. A single message is created for each call to any of the serial APIs.

To specify a Use mode and Queue Size, place declarations in the compiler preprocessor options when building your application:

- **Use Mode:**
 - ::EMBER_SERIAL0_MODE>::EMBER_SERIAL_BUFFER or ::EMBER_SERIAL_FIFO
 - ::EMBER_SERIAL1_MODE>::EMBER_SERIAL_BUFFER or ::EMBER_SERIAL_FIFO
- **Queue Size:**
 - ::EMBER_SERIAL0_TX_QUEUE_SIZE=2
 - ::EMBER_SERIAL0_RX_QUEUE_SIZE=4
 - ::EMBER_SERIAL1_TX_QUEUE_SIZE=8
 - ::EMBER_SERIAL1_RX_QUEUE_SIZE=16

Note the following:

- If buffered mode is declared, [emberSerialBufferTick\(\)](#) should be called in the application's main event loop.
- If buffered mode is declared, the Tx queue size **MUST** be ≤ 255
- On the AVR platform, Rx & Tx queue sizes are limited to powers of 2 ≤ 128
- By default, both ports are unused.

You can also use declarations to specify what should be done if an attempt is made to send more data than the queue can accommodate:

- `::EMBER_SERIAL0_BLOCKING`
- `::EMBER_SERIAL1_BLOCKING`

Be aware that since blocking spins in a loop, doing nothing until space is available, it can adversely affect any code that has tight timing requirements.

If `::EMBER_SERIAL0_BLOCKING` or `::EMBER_SERIAL1_BLOCKING` is defined, then the call to the port will block until space is available, guaranteeing that the entire message is sent. Note that in buffered mode, even if blocking mode is in effect entire messages may be dropped if insufficient stack buffers are available to hold them. When this happens, [EMBER_NO_BUFFERS](#) is returned.

If no blocking mode is defined, the serial code defaults to non-blocking mode. In this event, when the queue is too short, the data that don't fit are dropped. In FIFO mode, this may result bytes being dropped, starting in the middle of message. In buffered mode, the entire message is dropped. When data is dropped, `::EMBER_SERIALTX_OVERFLOW` is returned.

To minimize code size, very little error checking is done on the given parameters. Specifying an invalid or unused serial port may result in unexplained behavior. In some cases [EMBER_ERR_FATAL](#) may be returned.

6.41.2 Function Documentation

6.41.2.1 `EmberStatus emberSerialInit (uint8_t port, SerialBaudRate rate, SerialParity parity, uint8_t stopBits)`

Initializes a serial port to a specific baud rate, parity, and number of stop bits. Eight data bits are always used.

Parameters

<i>port</i>	A serial port number (0 or 1).
<i>rate</i>	The baud rate (see <code>SerialBaudRate</code>).
<i>parity</i>	The parity value (see <code>SerialParity</code>).
<i>stopBits</i>	The number of stop bits.

Returns

An error code if initialization failed (such as invalid baudrate), or [EMBER_SUCCESS](#).

6.41.2.2 `uint16_t emberSerialReadAvailable (uint8_t port)`

Returns the number of bytes currently available for reading in the specified RX queue.

Parameters

<i>port</i>	A serial port number (0 or 1).
-------------	--------------------------------

Returns

The number of bytes available.

6.41.2.3 `EmberStatus emberSerialReadByte (uint8_t port, uint8_t * dataByte)`

Reads a byte from the specified RX queue. If an error is returned, the dataByte should be ignored. For errors other than `EMBER_SERIAL_RX_EMPTY` multiple bytes of data may have been lost and serial protocols should attempt to resynchronize.

Parameters

<i>port</i>	A serial port number (0 or 1).
<i>dataByte</i>	A pointer to storage location for the byte.

Returns

One of the following (see the Main Page):

- `EMBER_SERIAL_RX_EMPTY` if no data is available
- `EMBER_SERIAL_RX_OVERFLOW` if the serial receive fifo was out of space
- `EMBER_SERIAL_RX_FRAME_ERROR` if a framing error was received
- `EMBER_SERIAL_RX_PARITY_ERROR` if a parity error was received
- `EMBER_SERIAL_RX_OVERRUN_ERROR` if the hardware fifo was out of space
- `EMBER_SUCCESS` if a data byte is returned

6.41.2.4 `EmberStatus emberSerialReadData (uint8_t port, uint8_t * data, uint16_t length, uint16_t * bytesRead)`

Reads bytes from the specified RX queue. Blocks until the full length has been read or an error occurs. In the event of an error, some valid data may have already been read before the error occurred, in which case that data will be in the buffer pointed to by data and the number of bytes successfully read will be placed in bytesRead.

Parameters

<i>port</i>	A serial port number (0 or 1).
<i>data</i>	A pointer to storage location for the data. It must be at least length in size.
<i>length</i>	The number of bytes to read.
<i>bytesRead</i>	A pointer to a location that will receive the number of bytes read. If the function returns early due to an error, this value may be less than length. This parameter may be NULL, in which case it is ignored.

Returns

One of the following (see the Main Page):

- [EMBER_SERIAL_RX_OVERFLOW](#) if the serial receive fifo was out of space
- [EMBER_SERIAL_RX_FRAME_ERROR](#) if a framing error was received
- [EMBER_SERIAL_RX_PARITY_ERROR](#) if a parity error was received
- [EMBER_SERIAL_RX_OVERRUN_ERROR](#) if the hardware fifo was out of space
- [EMBER_SUCCESS](#) if all the data requested is returned

6.41.2.5 `EmberStatus emberSerialReadDataTimeout (uint8_t port, uint8_t * data, uint16_t length, uint16_t * bytesRead, uint16_t firstByteTimeout, uint16_t subsequentByteTimeout)`

Reads bytes from the specified RX queue, up to a maximum of `length` bytes. The function may return before `length` bytes is read if a timeout is reached or an error occurs. Returns [EMBER_SERIAL_RX_EMPTY](#) if a timeout occurs.

Parameters

<i>port</i>	A serial port number (0 or 1).
<i>data</i>	A pointer to storage location for the data. It must be at least <code>length</code> in size.
<i>length</i>	The maximum number of bytes to read.
<i>bytesRead</i>	A pointer to a location that will receive the number of bytes read. If the function returns early due to an error or timeout, this value may be less than <code>length</code> . This parameter may be NULL, in which case it is ignored.
<i>firstByteTimeout</i>	The amount of time, in milliseconds, to wait for the first byte to arrive (if the queue is empty when the function is called). This value must be a minimum of 2 due to the timer resolution.
<i>subsequentByteTimeout</i>	The amount of time, in milliseconds, to wait after the previous byte was received for the next byte to arrive. This value must be a minimum of 2 due to the timer resolution.

Returns

One of the following (see the Main Page):

- [EMBER_SERIAL_RX_EMPTY](#) if the timeout was exceeded before the requested amount of data was read
- [EMBER_SERIAL_RX_OVERFLOW](#) if the serial receive fifo was out of space
- [EMBER_SERIAL_RX_FRAME_ERROR](#) if a framing error was received
- [EMBER_SERIAL_RX_PARITY_ERROR](#) if a parity error was received
- [EMBER_SERIAL_RX_OVERRUN_ERROR](#) if the hardware fifo was out of space
- [EMBER_SUCCESS](#) if all the data requested is returned

6.41.2.6 EmberStatus emberSerialReadLine (uint8_t *port*, char * *data*, uint8_t *max*)

Simulates a terminal interface, reading a line of characters at a time. Supports backspace. Always converts to uppercase. Blocks until a line has been read or max has been exceeded. Calls on [halResetWatchdog\(\)](#).

Parameters

<i>port</i>	A serial port number (0 or 1).
<i>data</i>	A pointer to storage location for the read line. There must be <code>max</code> contiguous bytes available at this location.
<i>max</i>	The maximum number of bytes to read.

Returns

[EMBER_SUCCESS](#)

6.41.2.7 EmberStatus emberSerialReadPartialLine (uint8_t *port*, char * *data*, uint8_t *max*, uint8_t * *index*)

Simulates a partial terminal interface, reading a line of characters at a time. Supports backspace. Always converts to uppercase. returns [EMBER_SUCCESS](#) when a line has been read or max has been exceeded. Must initialize the index variable to 0 to start a line.

Parameters

<i>port</i>	A serial port number (0 or 1).
<i>data</i>	A pointer to storage location for the read line. There must be <code>max</code> contiguous bytes available at this location.
<i>max</i>	The maximum number of bytes to read.
<i>index</i>	The address of a variable that holds the place in the <code>data</code> to continue. Set to 0 to start a line read.

Returns

One of the following (see the Main Page):

- [EMBER_SERIAL_RX_EMPTY](#) if a partial line is in progress.
- [EMBER_SERIAL_RX_OVERFLOW](#) if the serial receive fifo was out of space.
- [EMBER_SERIAL_RX_FRAME_ERROR](#) if a framing error was received.
- [EMBER_SERIAL_RX_PARITY_ERROR](#) if a parity error was received.
- [EMBER_SERIAL_RX_OVERRUN_ERROR](#) if the hardware fifo was out of space.
- [EMBER_SUCCESS](#) if a full line is ready.

6.41.2.8 uint16_t emberSerialWriteAvailable (uint8_t *port*)

Returns the number of bytes (in FIFO mode) or messages (in buffered mode) that can currently be queued to send without blocking or dropping.

Parameters

<i>port</i>	A serial port number (0 or 1).
-------------	--------------------------------

Returns

The number of bytes or messages available for queueing.

6.41.2.9 uint16_t emberSerialWriteUsed (uint8_t port)

Returns the number of bytes (in FIFO mode) or messages (in buffered mode) that are currently queued and still being sent.

Parameters

<i>port</i>	A serial port number (0 or 1).
-------------	--------------------------------

Returns

The number of bytes or messages available for queueing.

6.41.2.10 EmberStatus emberSerialWriteByte (uint8_t port, uint8_t dataByte)

Queues a single byte of data for transmission on the specified port.

Parameters

<i>port</i>	A serial port number (0 or 1).
<i>dataByte</i>	The byte to be queued.

Returns

One of the following (see the Main Page):

- [EMBER_SERIAL_TX_OVERFLOW](#) indicates that data was dropped.
- [EMBER_NO_BUFFERS](#) indicates that there was an insufficient number of available stack buffers.
- [EMBER_SUCCESS](#).

6.41.2.11 EmberStatus emberSerialWriteHex (uint8_t port, uint8_t dataByte)

Converts a given byte of data to its two-character ASCII hex representation and queues it for transmission on the specified port. Values less than 0xF are always zero padded and queued as "0F".

Parameters

<i>port</i>	A serial port number (0 or 1).
<i>dataByte</i>	The byte to be converted.

Returns

One of the following (see the Main Page):

- [EMBER_SERIAL_TX_OVERFLOW](#) indicates that data was dropped.
- [EMBER_NO_BUFFERS](#) indicates that there was an insufficient number of available stack buffers.
- [EMBER_SUCCESS](#).

6.41.2.12 EmberStatus emberSerialWriteString (uint8_t *port*, PGM_P *string*)

Queues a string for transmission on the specified port.

Parameters

<i>port</i>	A serial port number (0 or 1).
<i>string</i>	The string to be queued.

Returns

One of the following (see the Main Page):

- [EMBER_SERIAL_TX_OVERFLOW](#) indicates that data was dropped.
- [EMBER_NO_BUFFERS](#) indicates that there was an insufficient number of available stack buffers.
- [EMBER_SUCCESS](#).

6.41.2.13 XAP2B_PAGEZERO_ON EmberStatus emberSerialPrintf (uint8_t *port*, PGM_P *formatString*, ...)

Printf for printing on a specified port. Supports the following format specifiers:

- %% percent sign
- c single-byte character
- s RAM string
- p flash string (nonstandard specifier)
- u 2-byte unsigned decimal
- d 2-byte signed decimal
- l 4-byte signed decimal
- x %2x %4x 1-, 2-, 4-byte hex value (always 0 padded) (nonstandard specifier)

Parameters

<i>port</i>	A serial port number (0 or 1).
<i>formatString</i>	The string to print.
...	Format specifiers.

Returns

One of the following (see the Main Page):

- [EMBER_SERIAL_TX_OVERFLOW](#) indicates that data was dropped.
- [EMBER_NO_BUFFERS](#) indicates that there was an insufficient number of available stack buffers.
- [EMBER_SUCCESS](#).

6.41.2.14 XAP2B_PAGEZERO_OFF XAP2B_PAGEZERO_ON EmberStatus emberSerialPrintfLine (uint8_t *port*, PGM_P *formatString*, ...)

Printf for printing on a specified port. Same as [emberSerialPrintf\(\)](#) except it prints a carriage return at the the end of the text.

Parameters

<i>port</i>	A serial port number (0 or 1).
<i>formatString</i>	The string to print.
...	Format specifiers.

Returns

One of the following (see the Main Page):

- [EMBER_SERIAL_TX_OVERFLOW](#) indicates that data was dropped.
- [EMBER_NO_BUFFERS](#) indicates that there was an insufficient number of available stack buffers.
- [EMBER_SUCCESS](#).

6.41.2.15 XAP2B_PAGEZERO_OFF XAP2B_PAGEZERO_ON EmberStatus emberSerialPrintCarriageReturn (uint8_t *port*)

Prints "\r\n" to the specified serial port.

Parameters

<i>port</i>	A serial port number (0 or 1).
-------------	--------------------------------

Returns

One of the following (see the Main Page):

- [EMBER_SERIAL_TX_OVERFLOW](#) indicates that data was dropped.
- [EMBER_NO_BUFFERS](#) indicates that there was an insufficient number of available stack buffers.
- [EMBER_SUCCESS](#).

6.41.2.16 XAP2B_PAGEZERO_OFF EmberStatus emberSerialPrintfVarArg (uint8_t *port*, PGM_P *formatString*, va_list *ap*)

Prints a format string with a variable argument list.

Parameters

<i>port</i>	A serial port number (0 or 1).
<i>formatString</i>	A printf style format string.
<i>ap</i>	A variable argument list.

Returns

One of the following (see the Main Page):

- [EMBER_SERIAL_TX_OVERFLOW](#) indicates that data was dropped.
- [EMBER_NO_BUFFERS](#) indicates that there was an insufficient number of available stack buffers.
- [EMBER_SUCCESS](#).

6.41.2.17 EmberStatus emberSerialWriteData (uint8_t *port*, uint8_t * *data*, uint8_t *length*)

Queues an arbitrary chunk of data for transmission on a specified port.

Parameters

<i>port</i>	A serial port number (0 or 1).
<i>data</i>	A pointer to data.
<i>length</i>	The number of bytes to queue.

Returns

One of the following (see the Main Page):

- [EMBER_SERIAL_TX_OVERFLOW](#) indicates that data was dropped.
- [EMBER_NO_BUFFERS](#) indicates that there was an insufficient number of available stack buffers.
- [EMBER_SUCCESS](#).

6.41.2.18 XAP2B_PAGEZERO_ON EmberStatus emberSerialWaitSend (uint8_t *port*)

Waits for all data currently queued on the specified port to be transmitted before returning. **Note:** Call this function before serial reinitialization to ensure that transmission is complete.

Parameters

<i>port</i>	A serial port number (0 or 1).
-------------	--------------------------------

Returns

One of the following (see the Main Page):

- [EMBER_SERIAL_TX_OVERFLOW](#) indicates that data was dropped.
- [EMBER_NO_BUFFERS](#) indicates that there was an insufficient number of available stack buffers.
- [EMBER_SUCCESS](#).

6.41.2.19 XAP2B_PAGEZERO_OFF EmberStatus emberSerialGuaranteedPrintf (uint8_t port, PGM_P formatString, ...)

A printf routine that takes over the specified serial port and immediately transmits the given data regardless of what is currently queued. Does not return until the transmission is complete.

Application Usage:

Useful for fatal situations (such as asserts) where the node will be reset, but information on the cause for the reset needs to be transmitted first.

Parameters

<i>port</i>	A serial port number (0 or 1).
<i>formatString</i>	The string to print.
<i>...</i>	Formatting specifiers. See emberSerialPrintf() for arguments.

Returns

One of the following (see the Main Page):

- [EMBER_SERIAL_TX_OVERFLOW](#) indicates that data was dropped.
- [EMBER_NO_BUFFERS](#) indicates that there was an insufficient number of available stack buffers.
- [EMBER_SUCCESS](#).

6.41.2.20 void emberSerialBufferTick (void)

When a serial port is used in buffered mode, this must be called in an application's main event loop, similar to `emberTick()`. It frees buffers that are used to queue messages. **Note:** This function has no effect if FIFO mode is being used.

6.41.2.21 void emberSerialFlushRx (uint8_t port)

Flushes the receive buffer in case none of the incoming serial data is wanted.

Parameters

<i>port</i>	A serial port number (0 or 1).
-------------	--------------------------------

6.41.2.22 bool emberSerialUnused (uint8_t *port*)

Indicates whether the port is unused or invalid.

Parameters

<i>port</i>	A serial port number.
-------------	-----------------------

Returns

true if the port is unused or invalid.

6.42 Deprecated Files

Chapter 7

Data Structure Documentation

7.1 EmberAesMmoHashContext Struct Reference

```
#include <ember-types.h>
```

Data Fields

- `uint8_t result` [[EMBER_AES_HASH_BLOCK_SIZE](#)]
- `uint32_t length`

7.1.1 Detailed Description

This data structure contains the context data when calculating an AES MMO hash (message digest).

Definition at line [1537](#) of file [ember-types.h](#).

7.1.2 Field Documentation

7.1.2.1 `uint8_t EmberAesMmoHashContext::result[EMBER_AES_HASH_BLOCK_SIZE]`

Definition at line [1538](#) of file [ember-types.h](#).

7.1.2.2 `uint32_t EmberAesMmoHashContext::length`

Definition at line [1539](#) of file [ember-types.h](#).

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

- [ember-types.h](#)

7.2 EmberApsFrame Struct Reference

```
#include <ember-types.h>
```

Data Fields

- `uint16_t` [profileId](#)
- `uint16_t` [clusterId](#)
- `uint8_t` [sourceEndpoint](#)
- `uint8_t` [destinationEndpoint](#)
- [EmberApsOption](#) [options](#)
- `uint16_t` [groupId](#)
- `uint8_t` [sequence](#)

7.2.1 Detailed Description

An in-memory representation of a ZigBee APS frame of an incoming or outgoing message.

Definition at line 960 of file [ember-types.h](#).

7.2.2 Field Documentation

7.2.2.1 `uint16_t` [EmberApsFrame::profileId](#)

The application profile ID that describes the format of the message.

Definition at line 962 of file [ember-types.h](#).

7.2.2.2 `uint16_t` [EmberApsFrame::clusterId](#)

The cluster ID for this message.

Definition at line 964 of file [ember-types.h](#).

7.2.2.3 `uint8_t` [EmberApsFrame::sourceEndpoint](#)

The source endpoint.

Definition at line 966 of file [ember-types.h](#).

7.2.2.4 `uint8_t` [EmberApsFrame::destinationEndpoint](#)

The destination endpoint.

Definition at line 968 of file [ember-types.h](#).

7.2.2.5 [EmberApsOption](#) [EmberApsFrame::options](#)

A bitmask of options from the enumeration above.

Definition at line 970 of file [ember-types.h](#).

7.2.2.6 `uint16_t EmberApsFrame::groupId`

The group ID for this message, if it is multicast mode.

Definition at line 972 of file [ember-types.h](#).

7.2.2.7 `uint8_t EmberApsFrame::sequence`

The sequence number.

Definition at line 974 of file [ember-types.h](#).

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

- [ember-types.h](#)

7.3 EmberBindingTableEntry Struct Reference

```
#include <ember-types.h>
```

Data Fields

- [EmberBindingType](#) type
- `uint8_t` local
- `uint16_t` clusterId
- `uint8_t` remote
- [EmberEUI64](#) identifier
- `uint8_t` networkIndex

7.3.1 Detailed Description

Defines an entry in the binding table.

A binding entry specifies a local endpoint, a remote endpoint, a cluster ID and either the destination EUI64 (for unicast bindings) or the 64-bit group address (for multicast bindings).

Definition at line 984 of file [ember-types.h](#).

7.3.2 Field Documentation

7.3.2.1 `EmberBindingType EmberBindingTableEntry::type`

The type of binding.

Definition at line 986 of file [ember-types.h](#).

7.3.2.2 `uint8_t EmberBindingTableEntry::local`

The endpoint on the local node.

Definition at line 988 of file [ember-types.h](#).

7.3.2.3 `uint16_t EmberBindingTableEntry::clusterId`

A cluster ID that matches one from the local endpoint's simple descriptor. This cluster ID is set by the provisioning application to indicate which part an endpoint's functionality is bound to this particular remote node and is used to distinguish between unicast and multicast bindings. Note that a binding can be used to send messages with any cluster ID, not just that listed in the binding.

Definition at line 996 of file [ember-types.h](#).

7.3.2.4 `uint8_t EmberBindingTableEntry::remote`

The endpoint on the remote node (specified by `identifier`).

Definition at line 998 of file [ember-types.h](#).

7.3.2.5 `EmberEUI64 EmberBindingTableEntry::identifier`

A 64-bit identifier. This is either:

- The destination EUI64, for unicasts
- A 16-bit multicast group address, for multicasts

Definition at line 1003 of file [ember-types.h](#).

7.3.2.6 `uint8_t EmberBindingTableEntry::networkIndex`

The index of the network the binding belongs to.

Definition at line 1005 of file [ember-types.h](#).

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

- [ember-types.h](#)

7.4 EmberCertificate283k1Data Struct Reference

```
#include <ember-types.h>
```

Data Fields

- `uint8_t contents` [[EMBER_CERTIFICATE_283K1_SIZE](#)]

7.4.1 Detailed Description

This data structure contains the certificate data that is used for Certificate Based Key Exchange (CBKE) in SECT283k1 Elliptical Cryptography.

Definition at line 1544 of file [ember-types.h](#).

7.4.2 Field Documentation

7.4.2.1 uint8_t EmberCertificate283k1Data::contents[EMBER_CERTIFICATE_283K1_SIZE]

Definition at line 1546 of file [ember-types.h](#).

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

- [ember-types.h](#)

7.5 EmberCertificateData Struct Reference

```
#include <ember-types.h>
```

Data Fields

- uint8_t [contents](#) [EMBER_CERTIFICATE_SIZE]

7.5.1 Detailed Description

This data structure contains the certificate data that is used for Certificate Based Key Exchange (CBKE).

Definition at line 1499 of file [ember-types.h](#).

7.5.2 Field Documentation

7.5.2.1 uint8_t EmberCertificateData::contents[EMBER_CERTIFICATE_SIZE]

Definition at line 1500 of file [ember-types.h](#).

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

- [ember-types.h](#)

7.6 EmberCommandEntry Struct Reference

```
#include <command-interpret2.h>
```

Data Fields

- PGM_P [name](#)
- [CommandAction](#) [action](#)
- PGM_P [argumentTypes](#)
- PGM_P [description](#)
- PGM_P const * [argumentDescriptions](#)

7.6.1 Detailed Description

Command entry for a command table.

Definition at line 129 of file [command-interpreter2.h](#).

7.6.2 Field Documentation

7.6.2.1 PGM_P EmberCommandEntry::name

Use letters, digits, and underscores, '_', for the command name. Command names are case-sensitive.

Definition at line 136 of file [command-interpreter2.h](#).

7.6.2.2 CommandAction EmberCommandEntry::action

A reference to a function in the application that implements the command. If this entry refers to a nested command, then action field has to be set to NULL.

Definition at line 142 of file [command-interpreter2.h](#).

7.6.2.3 PGM_P EmberCommandEntry::argumentTypes

In case of normal (non-nested) commands, argumentTypes is a string that specifies the number and types of arguments the command accepts. The argument specifiers are:

- u: one-byte unsigned integer.
- v: two-byte unsigned integer
- w: four-byte unsigned integer
- s: one-byte signed integer
- r: two-byte signed integer
- q: four-byte signed integer
- b: string. The argument can be entered in ascii by using quotes, for example: "foo". Or it may be entered in hex by using curly braces, for example: { 08 A1 f2 }. There must be an even number of hex digits, and spaces are ignored.
- *: zero or more of the previous type. If used, this must be the last specifier.
- ?: Unknown number of arguments. If used this must be the only character. This means, that command interpreter will not perform any validation of arguments, and will call the action directly, trusting it that it will handle with whatever arguments are passed in. Integer arguments can be either decimal or hexadecimal. A 0x prefix indicates a hexadecimal integer. Example: 0x3ed.

In case of a nested command (action is NULL), then this field contains a pointer to the nested [EmberCommandEntry](#) array.

Definition at line 171 of file [command-interpreter2.h](#).

7.6.2.4 PGM_P EmberCommandEntry::description

A description of the command.

Definition at line 176 of file [command-interpreter2.h](#).

7.6.2.5 PGM_P const* EmberCommandEntry::argumentDescriptions

An array of strings, with a NULL terminator, indicating what each argument is.

Definition at line 180 of file [command-interpreter2.h](#).

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

- [command-interpreter2.h](#)

7.7 EmberCurrentSecurityState Struct Reference

```
#include <ember-types.h>
```

Data Fields

- [EmberCurrentSecurityBitmask](#) bitmask
- [EmberEUI64](#) trustCenterLongAddress

7.7.1 Detailed Description

This describes the security features used by the stack for a joined device.

Definition at line 1828 of file [ember-types.h](#).

7.7.2 Field Documentation

7.7.2.1 EmberCurrentSecurityBitmask EmberCurrentSecurityState::bitmask

This bitmask indicates the security features currently in use on this node.

Definition at line 1831 of file [ember-types.h](#).

7.7.2.2 EmberEUI64 EmberCurrentSecurityState::trustCenterLongAddress

This indicates the EUI64 of the Trust Center. It will be all zeroes if the Trust Center Address is not known (i.e. the device is in a Distributed Trust Center network).

Definition at line 1835 of file [ember-types.h](#).

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

- [ember-types.h](#)

7.8 EmberEventControl Struct Reference

```
#include <ember-types.h>
```

Data Fields

- [EmberEventUnits](#) status
- [EmberTaskId](#) taskid
- [uint32_t](#) timeToExecute

7.8.1 Detailed Description

Control structure for events.

This structure should not be accessed directly. This holds the event status (one of the *EM-BER_EVENT_* values) and the time left before the event fires.

Definition at line 1272 of file [ember-types.h](#).

7.8.2 Field Documentation

7.8.2.1 EmberEventUnits EmberEventControl::status

The event's status, either inactive or the units for timeToExecute.

Definition at line 1274 of file [ember-types.h](#).

7.8.2.2 EmberTaskId EmberEventControl::taskid

The id of the task this event belongs to.

Definition at line 1276 of file [ember-types.h](#).

7.8.2.3 uint32_t EmberEventControl::timeToExecute

How long before the event fires. Units are always in milliseconds

Definition at line 1280 of file [ember-types.h](#).

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

- [ember-types.h](#)

7.9 EmberEventData_S Struct Reference

```
#include <ember-types.h>
```

Data Fields

- [EmberEventControl](#) * control
- void(* [handler](#))(void)

7.9.1 Detailed Description

Complete events with a control and a handler procedure.

An application typically creates an array of events along with their handlers. The main loop passes the array to `::emberRunEvents()` in order to call the handlers of any events whose time has arrived.

Definition at line 1290 of file [ember-types.h](#).

7.9.2 Field Documentation

7.9.2.1 `EmberEventControl* EmberEventData_S::control`

The control structure for the event.

Definition at line 1292 of file [ember-types.h](#).

7.9.2.2 `void(* EmberEventData_S::handler)(void)`

The procedure to call when the event fires.

Definition at line 1294 of file [ember-types.h](#).

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

- [ember-types.h](#)

7.10 EmberInitialSecurityState Struct Reference

```
#include <ember-types.h>
```

Data Fields

- `uint16_t` [bitmask](#)
- `EmberKeyData` [preconfiguredKey](#)
- `EmberKeyData` [networkKey](#)
- `uint8_t` [networkKeySequenceNumber](#)
- `EmberEUI64` [preconfiguredTrustCenterEui64](#)

7.10.1 Detailed Description

This describes the Initial Security features and requirements that will be used when forming or joining the network.

Definition at line 1748 of file [ember-types.h](#).

7.10.2 Field Documentation

7.10.2.1 `uint16_t EmberInitialSecurityState::bitmask`

This bitmask enumerates which security features should be used, as well as the presence of valid data within other elements of the [EmberInitialSecurityState](#) data structure. For more details see the [EmberInitialSecurityBitmask](#).

Definition at line 1753 of file [ember-types.h](#).

7.10.2.2 `EmberKeyData EmberInitialSecurityState::preconfiguredKey`

This is the pre-configured key that can be used by devices when joining the network if the Trust Center does not send the initial security data in-the-clear. For the Trust Center, it will be the global link key and **must** be set regardless of whether joining devices are expected to have a pre-configured Link Key. This parameter will only be used if the [EmberInitialSecurityState::bitmask](#) sets the bit indicating [EMBER_HAVE_PRECONFIGURED_KEY](#).

Definition at line 1762 of file [ember-types.h](#).

7.10.2.3 `EmberKeyData EmberInitialSecurityState::networkKey`

This is the Network Key used when initially forming the network. This must be set on the Trust Center. It is not needed for devices joining the network. This parameter will only be used if the [EmberInitialSecurityState::bitmask](#) sets the bit indicating [EMBER_HAVE_NETWORK_KEY](#).

Definition at line 1768 of file [ember-types.h](#).

7.10.2.4 `uint8_t EmberInitialSecurityState::networkKeySequenceNumber`

This is the sequence number associated with the network key. It must be set if the Network Key is set. It is used to indicate a particular of the network key for updating and switching. This parameter will only be used if the [EMBER_HAVE_NETWORK_KEY](#) is set. Generally it should be set to 0 when forming the network; joining devices can ignore this value.

Definition at line 1775 of file [ember-types.h](#).

7.10.2.5 `EmberEUI64 EmberInitialSecurityState::preconfiguredTrustCenterEui64`

This is the long address of the trust center on the network that will be joined. It is usually NOT set prior to joining the network and instead it is learned during the joining message exchange. This field is only examined if [EMBER_HAVE_TRUST_CENTER_EUI64](#) is set in the [EmberInitialSecurityState::bitmask](#). Most devices should clear that bit and leave this field alone. This field must be set when using commissioning mode. It is required to be in little-endian format.

Definition at line 1783 of file [ember-types.h](#).

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

- [ember-types.h](#)

7.11 EmberKeyData Struct Reference

```
#include <ember-types.h>
```

Data Fields

- `uint8_t contents` [[EMBER_ENCRYPTION_KEY_SIZE](#)]

7.11.1 Detailed Description

This data structure contains the key data that is passed into various other functions.

Definition at line [1492](#) of file [ember-types.h](#).

7.11.2 Field Documentation

7.11.2.1 `uint8_t EmberKeyData::contents`[[EMBER_ENCRYPTION_KEY_SIZE](#)]

This is the key byte data.

Definition at line [1494](#) of file [ember-types.h](#).

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

- [ember-types.h](#)

7.12 EmberKeyStruct Struct Reference

```
#include <ember-types.h>
```

Data Fields

- [EmberKeyStructBitmask](#) bitmask
- [EmberKeyType](#) type
- [EmberKeyData](#) key
- `uint32_t outgoingFrameCounter`
- `uint32_t incomingFrameCounter`
- `uint8_t sequenceNumber`
- [EmberEUI64](#) partnerEUI64

7.12.1 Detailed Description

This describes a one of several different types of keys and its associated data.

Definition at line [1901](#) of file [ember-types.h](#).

7.12.2 Field Documentation

7.12.2.1 EmberKeyStructBitmask EmberKeyStruct::bitmask

This bitmask indicates whether various fields in the structure contain valid data. It also contains the index of the network the key belongs to.

Definition at line 1905 of file [ember-types.h](#).

7.12.2.2 EmberKeyType EmberKeyStruct::type

This indicates the type of the security key.

Definition at line 1907 of file [ember-types.h](#).

7.12.2.3 EmberKeyData EmberKeyStruct::key

This is the actual key data.

Definition at line 1909 of file [ember-types.h](#).

7.12.2.4 uint32_t EmberKeyStruct::outgoingFrameCounter

This is the outgoing frame counter associated with the key. It will contain valid data based on the [EmberKeyStructBitmask](#).

Definition at line 1912 of file [ember-types.h](#).

7.12.2.5 uint32_t EmberKeyStruct::incomingFrameCounter

This is the incoming frame counter associated with the key. It will contain valid data based on the [EmberKeyStructBitmask](#).

Definition at line 1915 of file [ember-types.h](#).

7.12.2.6 uint8_t EmberKeyStruct::sequenceNumber

This is the sequence number associated with the key. It will contain valid data based on the [EmberKeyStructBitmask](#).

Definition at line 1918 of file [ember-types.h](#).

7.12.2.7 EmberEUI64 EmberKeyStruct::partnerEUI64

This is the Partner EUI64 associated with the key. It will contain valid data based on the [EmberKeyStructBitmask](#).

Definition at line 1921 of file [ember-types.h](#).

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

- [ember-types.h](#)

7.13 EmberMacFilterMatchStruct Struct Reference

```
#include <ember-types.h>
```

Data Fields

- `uint8_t` [filterIndexMatch](#)
- `EmberMacPassthroughType` [legacyPassthroughType](#)
- `EmberMessageBuffer` [message](#)

7.13.1 Detailed Description

This structure indicates a matching raw MAC message has been received by the application configured MAC filters.

Definition at line [2214](#) of file [ember-types.h](#).

7.13.2 Field Documentation

7.13.2.1 `uint8_t` `EmberMacFilterMatchStruct::filterIndexMatch`

Definition at line [2215](#) of file [ember-types.h](#).

7.13.2.2 `EmberMacPassthroughType` `EmberMacFilterMatchStruct::legacyPassthroughType`

Definition at line [2216](#) of file [ember-types.h](#).

7.13.2.3 `EmberMessageBuffer` `EmberMacFilterMatchStruct::message`

Definition at line [2217](#) of file [ember-types.h](#).

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

- [ember-types.h](#)

7.14 EmberMessageDigest Struct Reference

```
#include <ember-types.h>
```

Data Fields

- `uint8_t` [contents](#) [[EMBER_AES_HASH_BLOCK_SIZE](#)]

7.14.1 Detailed Description

This data structure contains an AES-MMO Hash (the message digest).

Definition at line 1530 of file [ember-types.h](#).

7.14.2 Field Documentation

7.14.2.1 `uint8_t EmberMessageDigest::contents[EMBER_AES_HASH_BLOCK_SIZE]`

Definition at line 1531 of file [ember-types.h](#).

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

- [ember-types.h](#)

7.15 EmberMfgSecurityStruct Struct Reference

```
#include <ember-types.h>
```

Data Fields

- [EmberKeySettings](#) `keySettings`

7.15.1 Detailed Description

This structure is used to get/set the security config that is stored in manufacturing tokens.

Definition at line 2129 of file [ember-types.h](#).

7.15.2 Field Documentation

7.15.2.1 `EmberKeySettings EmberMfgSecurityStruct::keySettings`

Definition at line 2130 of file [ember-types.h](#).

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

- [ember-types.h](#)

7.16 EmberMulticastTableEntry Struct Reference

```
#include <ember-types.h>
```

Data Fields

- [EmberMulticastId](#) `multicastId`
- `uint8_t` `endpoint`
- `uint8_t` `networkIndex`

7.16.1 Detailed Description

Defines an entry in the multicast table.

A multicast table entry indicates that a particular endpoint is a member of a particular multicast group. Only devices with an endpoint in a multicast group will receive messages sent to that multicast group.

Definition at line 1073 of file [ember-types.h](#).

7.16.2 Field Documentation

7.16.2.1 EmberMulticastId EmberMulticastTableEntry::multicastId

The multicast group ID.

Definition at line 1075 of file [ember-types.h](#).

7.16.2.2 uint8_t EmberMulticastTableEntry::endpoint

The endpoint that is a member, or 0 if this entry is not in use (the ZDO is not a member of any multicast groups).

Definition at line 1079 of file [ember-types.h](#).

7.16.2.3 uint8_t EmberMulticastTableEntry::networkIndex

The network index of the network the entry is related to.

Definition at line 1081 of file [ember-types.h](#).

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

- [ember-types.h](#)

7.17 EmberNeighborTableEntry Struct Reference

```
#include <ember-types.h>
```

Data Fields

- uint16_t [shortId](#)
- uint8_t [averageLqi](#)
- uint8_t [inCost](#)
- uint8_t [outCost](#)
- uint8_t [age](#)
- [EmberEUI64](#) [longId](#)

7.17.1 Detailed Description

Defines an entry in the neighbor table.

A neighbor table entry stores information about the reliability of RF links to and from neighboring nodes.

Definition at line 1014 of file [ember-types.h](#).

7.17.2 Field Documentation

7.17.2.1 `uint16_t EmberNeighborTableEntry::shortId`

The neighbor's two byte network id.

Definition at line 1016 of file [ember-types.h](#).

7.17.2.2 `uint8_t EmberNeighborTableEntry::averageLqi`

An exponentially weighted moving average of the link quality values of incoming packets from this neighbor as reported by the PHY.

Definition at line 1019 of file [ember-types.h](#).

7.17.2.3 `uint8_t EmberNeighborTableEntry::inCost`

The incoming cost for this neighbor, computed from the average LQI. Values range from 1 for a good link to 7 for a bad link.

Definition at line 1022 of file [ember-types.h](#).

7.17.2.4 `uint8_t EmberNeighborTableEntry::outCost`

The outgoing cost for this neighbor, obtained from the most recently received neighbor exchange message from the neighbor. A value of zero means that a neighbor exchange message from the neighbor has not been received recently enough, or that our id was not present in the most recently received one. EmberZNet Pro only.

Definition at line 1029 of file [ember-types.h](#).

7.17.2.5 `uint8_t EmberNeighborTableEntry::age`

In EmberZNet Pro, the number of aging periods elapsed since a neighbor exchange message was last received from this neighbor. In stack profile 1, the number of aging periods since any packet was received. An entry with an age greater than 3 is considered stale and may be reclaimed. The aging period is 16 seconds.

Definition at line 1035 of file [ember-types.h](#).

7.17.2.6 `EmberEUI64 EmberNeighborTableEntry::longId`

The 8 byte EUI64 of the neighbor.

Definition at line [1037](#) of file [ember-types.h](#).

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

- [ember-types.h](#)

7.18 EmberNetworkInitStruct Struct Reference

```
#include <ember-types.h>
```

Data Fields

- [EmberNetworkInitBitmask](#) bitmask

7.18.1 Detailed Description

Defines the network initialization configuration that should be used when `::emberNetworkInitExtended()` is called by the application.

Definition at line [474](#) of file [ember-types.h](#).

7.18.2 Field Documentation

7.18.2.1 [EmberNetworkInitBitmask](#) [EmberNetworkInitStruct::bitmask](#)

Definition at line [475](#) of file [ember-types.h](#).

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

- [ember-types.h](#)

7.19 EmberNetworkParameters Struct Reference

```
#include <ember-types.h>
```

Data Fields

- `uint8_t` [extendedPanId](#) [8]
- `uint16_t` [panId](#)
- `int8_t` [radioTxPower](#)
- `uint8_t` [radioChannel](#)
- [EmberJoinMethod](#) [joinMethod](#)
- [EmberNodeId](#) [nwkManagerId](#)
- `uint8_t` [nwkUpdateId](#)
- `uint32_t` [channels](#)

7.19.1 Detailed Description

Holds network parameters.

For information about power settings and radio channels, see the technical specification for the RF communication module in your Developer Kit.

Definition at line 915 of file [ember-types.h](#).

7.19.2 Field Documentation

7.19.2.1 `uint8_t EmberNetworkParameters::extendedPanId[8]`

The network's extended PAN identifier.

Definition at line 917 of file [ember-types.h](#).

7.19.2.2 `uint16_t EmberNetworkParameters::panId`

The network's PAN identifier.

Definition at line 919 of file [ember-types.h](#).

7.19.2.3 `int8_t EmberNetworkParameters::radioTxPower`

A power setting, in dBm.

Definition at line 921 of file [ember-types.h](#).

7.19.2.4 `uint8_t EmberNetworkParameters::radioChannel`

A radio channel. Be sure to specify a channel supported by the radio.

Definition at line 923 of file [ember-types.h](#).

7.19.2.5 `EmberJoinMethod EmberNetworkParameters::joinMethod`

Join method: The protocol messages used to establish an initial parent. It is ignored when forming a ZigBee network, or when querying the stack for its network parameters.

Definition at line 928 of file [ember-types.h](#).

7.19.2.6 `EmberNodeId EmberNetworkParameters::nwkManagerId`

NWK Manager ID. The ID of the network manager in the current network. This may only be set at joining when using `EMBER_USE_NWK_COMMISSIONING` as the join method.

Definition at line 934 of file [ember-types.h](#).

7.19.2.7 `uint8_t EmberNetworkParameters::nwkUpdateId`

NWK Update ID. The value of the ZigBee `nwkUpdateId` known by the stack. This is used to determine the newest instance of the network after a PAN ID or channel change. This may only be set at joining when using `EMBER_USE_NWK_COMMISSIONING` as the join method.

Definition at line 940 of file [ember-types.h](#).

7.19.2.8 `uint32_t EmberNetworkParameters::channels`

NWK channel mask. The list of preferred channels that the NWK manager has told this device to use when searching for the network. This may only be set at joining when using `EMBER_USE_NWK_COMMISSIONING` as the join method.

Definition at line 946 of file [ember-types.h](#).

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

- [ember-types.h](#)

7.20 EmberPrivateKey283k1Data Struct Reference

```
#include <ember-types.h>
```

Data Fields

- `uint8_t contents` [[EMBER_PRIVATE_KEY_283K1_SIZE](#)]

7.20.1 Detailed Description

This data structure contains the private key data that is used for Certificate Based Key Exchange (CBKE) in SECT283k1 Elliptical Cryptography.

Definition at line 1557 of file [ember-types.h](#).

7.20.2 Field Documentation

7.20.2.1 `uint8_t EmberPrivateKey283k1Data::contents`[[EMBER_PRIVATE_KEY_283K1_SIZE](#)]

Definition at line 1558 of file [ember-types.h](#).

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

- [ember-types.h](#)

7.21 EmberPrivateKeyData Struct Reference

```
#include <ember-types.h>
```

Data Fields

- `uint8_t contents` [[EMBER_PRIVATE_KEY_SIZE](#)]

7.21.1 Detailed Description

This data structure contains the private key data that is used for Certificate Based Key Exchange (CBKE).

Definition at line [1511](#) of file [ember-types.h](#).

7.21.2 Field Documentation

7.21.2.1 `uint8_t EmberPrivateKeyData::contents`[[EMBER_PRIVATE_KEY_SIZE](#)]

Definition at line [1512](#) of file [ember-types.h](#).

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

- [ember-types.h](#)

7.22 EmberPublicKey283k1Data Struct Reference

```
#include <ember-types.h>
```

Data Fields

- `uint8_t contents` [[EMBER_PUBLIC_KEY_283K1_SIZE](#)]

7.22.1 Detailed Description

This data structure contains the public key data that is used for Certificate Based Key Exchange (CBKE) in SECT283k1 Elliptical Cryptography.

Definition at line [1551](#) of file [ember-types.h](#).

7.22.2 Field Documentation

7.22.2.1 `uint8_t EmberPublicKey283k1Data::contents`[[EMBER_PUBLIC_KEY_283K1_SIZE](#)]

Definition at line [1552](#) of file [ember-types.h](#).

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

- [ember-types.h](#)

7.23 EmberPublicKeyData Struct Reference

```
#include <ember-types.h>
```

Data Fields

- `uint8_t contents` [[EMBER_PUBLIC_KEY_SIZE](#)]

7.23.1 Detailed Description

This data structure contains the public key data that is used for Certificate Based Key Exchange (CBKE).

Definition at line [1505](#) of file [ember-types.h](#).

7.23.2 Field Documentation

7.23.2.1 `uint8_t EmberPublicKeyData::contents[EMBER_PUBLIC_KEY_SIZE]`

Definition at line [1506](#) of file [ember-types.h](#).

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

- [ember-types.h](#)

7.24 EmberReleaseTypeStruct Struct Reference

```
#include <ember-types.h>
```

Data Fields

- `EmberVersionType typeNum`
- `PGM_P typeString`

7.24.1 Detailed Description

A structure relating version types to human readable strings.

Definition at line [67](#) of file [ember-types.h](#).

7.24.2 Field Documentation

7.24.2.1 `EmberVersionType EmberReleaseTypeStruct::typeNum`

Definition at line [68](#) of file [ember-types.h](#).

7.24.2.2 PGM_P EmberReleaseTypeStruct::typeString

Definition at line 69 of file [ember-types.h](#).

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

- [ember-types.h](#)

7.25 EmberRouteTableEntry Struct Reference

```
#include <ember-types.h>
```

Data Fields

- uint16_t [destination](#)
- uint16_t [nextHop](#)
- uint8_t [status](#)
- uint8_t [age](#)
- uint8_t [concentratorType](#)
- uint8_t [routeRecordState](#)

7.25.1 Detailed Description

Defines an entry in the route table.

A route table entry stores information about the next hop along the route to the destination.

Definition at line 1045 of file [ember-types.h](#).

7.25.2 Field Documentation

7.25.2.1 uint16_t EmberRouteTableEntry::destination

The short id of the destination.

Definition at line 1047 of file [ember-types.h](#).

7.25.2.2 uint16_t EmberRouteTableEntry::nextHop

The short id of the next hop to this destination.

Definition at line 1049 of file [ember-types.h](#).

7.25.2.3 uint8_t EmberRouteTableEntry::status

Indicates whether this entry is active (0), being discovered (1), or unused (3).

Definition at line 1052 of file [ember-types.h](#).

7.25.2.4 `uint8_t EmberRouteTableEntry::age`

The number of seconds since this route entry was last used to send a packet.

Definition at line 1055 of file [ember-types.h](#).

7.25.2.5 `uint8_t EmberRouteTableEntry::concentratorType`

Indicates whether this destination is a High RAM Concentrator (2), a Low RAM Concentrator (1), or not a concentrator (0).

Definition at line 1058 of file [ember-types.h](#).

7.25.2.6 `uint8_t EmberRouteTableEntry::routeRecordState`

For a High RAM Concentrator, indicates whether a route record is needed (2), has been sent (1), or is no long needed (0) because a source routed message from the concentrator has been received.

Definition at line 1063 of file [ember-types.h](#).

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

- [ember-types.h](#)

7.26 EmberSignature283k1Data Struct Reference

```
#include <ember-types.h>
```

Data Fields

- `uint8_t contents` [[EMBER_SIGNATURE_283K1_SIZE](#)]

7.26.1 Detailed Description

This data structure contains a DSA signature used in SECT283k1 Elliptical Cryptography. It is the bit concatenation of the 'r' and 's' components of the signature.

Definition at line 1565 of file [ember-types.h](#).

7.26.2 Field Documentation

7.26.2.1 `uint8_t EmberSignature283k1Data::contents`[[EMBER_SIGNATURE_283K1_SIZE](#)]

Definition at line 1566 of file [ember-types.h](#).

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

- [ember-types.h](#)

7.27 EmberSignatureData Struct Reference

```
#include <ember-types.h>
```

Data Fields

- `uint8_t contents` [[EMBER_SIGNATURE_SIZE](#)]

7.27.1 Detailed Description

This data structure contains a DSA signature. It is the bit concatenation of the 'r' and 's' components of the signature.

Definition at line [1524](#) of file [ember-types.h](#).

7.27.2 Field Documentation

7.27.2.1 `uint8_t EmberSignatureData::contents[EMBER_SIGNATURE_SIZE]`

Definition at line [1525](#) of file [ember-types.h](#).

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

- [ember-types.h](#)

7.28 EmberSmacData Struct Reference

```
#include <ember-types.h>
```

Data Fields

- `uint8_t contents` [[EMBER_SMAC_SIZE](#)]

7.28.1 Detailed Description

This data structure contains the Shared Message Authentication Code (SMAC) data that is used for Certificate Based Key Exchange (CBKE).

Definition at line [1517](#) of file [ember-types.h](#).

7.28.2 Field Documentation

7.28.2.1 `uint8_t EmberSmacData::contents[EMBER_SMAC_SIZE]`

Definition at line [1518](#) of file [ember-types.h](#).

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

- [ember-types.h](#)

7.29 EmberTaskControl Struct Reference

```
#include <ember-types.h>
```

Data Fields

- uint32_t [nextEventTime](#)
- [EmberEventData](#) * [events](#)
- bool [busy](#)

7.29.1 Detailed Description

Control structure for tasks.

This structure should not be accessed directly.

Definition at line [1301](#) of file [ember-types.h](#).

7.29.2 Field Documentation

7.29.2.1 uint32_t EmberTaskControl::nextEventTime

Definition at line [1303](#) of file [ember-types.h](#).

7.29.2.2 EmberEventData* EmberTaskControl::events

Definition at line [1305](#) of file [ember-types.h](#).

7.29.2.3 bool EmberTaskControl::busy

Definition at line [1307](#) of file [ember-types.h](#).

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

- [ember-types.h](#)

7.30 EmberVersion Struct Reference

```
#include <ember-types.h>
```

Data Fields

- uint16_t [build](#)
- uint8_t [major](#)
- uint8_t [minor](#)
- uint8_t [patch](#)
- uint8_t [special](#)
- [EmberVersionType](#) type

7.30.1 Detailed Description

Version struct containing all version information.

Definition at line 90 of file [ember-types.h](#).

7.30.2 Field Documentation

7.30.2.1 uint16_t EmberVersion::build

Definition at line 91 of file [ember-types.h](#).

7.30.2.2 uint8_t EmberVersion::major

Definition at line 92 of file [ember-types.h](#).

7.30.2.3 uint8_t EmberVersion::minor

Definition at line 93 of file [ember-types.h](#).

7.30.2.4 uint8_t EmberVersion::patch

Definition at line 94 of file [ember-types.h](#).

7.30.2.5 uint8_t EmberVersion::special

Definition at line 95 of file [ember-types.h](#).

7.30.2.6 EmberVersionType EmberVersion::type

Definition at line 96 of file [ember-types.h](#).

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

- [ember-types.h](#)

7.31 EmberZigbeeNetwork Struct Reference

```
#include <ember-types.h>
```

Data Fields

- uint16_t [panId](#)
- uint8_t [channel](#)
- bool [allowingJoin](#)
- uint8_t [extendedPanId](#) [8]
- uint8_t [stackProfile](#)
- uint8_t [nwUpdateId](#)

7.31.1 Detailed Description

Defines a ZigBee network and the associated parameters.

Definition at line 441 of file [ember-types.h](#).

7.31.2 Field Documentation

7.31.2.1 `uint16_t EmberZigbeeNetwork::panId`

Definition at line 442 of file [ember-types.h](#).

7.31.2.2 `uint8_t EmberZigbeeNetwork::channel`

Definition at line 443 of file [ember-types.h](#).

7.31.2.3 `bool EmberZigbeeNetwork::allowingJoin`

Definition at line 444 of file [ember-types.h](#).

7.31.2.4 `uint8_t EmberZigbeeNetwork::extendedPanId[8]`

Definition at line 445 of file [ember-types.h](#).

7.31.2.5 `uint8_t EmberZigbeeNetwork::stackProfile`

Definition at line 446 of file [ember-types.h](#).

7.31.2.6 `uint8_t EmberZigbeeNetwork::nwkUpdateId`

Definition at line 447 of file [ember-types.h](#).

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

- [ember-types.h](#)

7.32 HalEepromInformationType Struct Reference

```
#include <bootloader-EEPROM.h>
```

Data Fields

- `uint16_t` [version](#)
- `uint16_t` [capabilitiesMask](#)
- `uint16_t` [pageEraseMs](#)
- `uint16_t` [partEraseMs](#)
- `uint32_t` [pageSize](#)

- uint32_t [partSize](#)
- const char *const [partDescription](#)
- uint8_t [wordSizeBytes](#)

7.32.1 Detailed Description

This structure defines a variety of information about the attached external EEPROM device.

Definition at line 99 of file [bootloader-eeprom.h](#).

7.32.2 Field Documentation

7.32.2.1 uint16_t HalEepromInformationType::version

The version of this data structure

Definition at line 101 of file [bootloader-eeprom.h](#).

7.32.2.2 uint16_t HalEepromInformationType::capabilitiesMask

A bitmask describing the capabilities of this particular external EEPROM

Definition at line 103 of file [bootloader-eeprom.h](#).

7.32.2.3 uint16_t HalEepromInformationType::pageEraseMs

Maximum time it takes to erase a page. (in 1025Hz Milliseconds)

Definition at line 105 of file [bootloader-eeprom.h](#).

7.32.2.4 uint16_t HalEepromInformationType::partEraseMs

Maximum time it takes to erase the entire part. (in 1024Hz Milliseconds)

Definition at line 107 of file [bootloader-eeprom.h](#).

7.32.2.5 uint32_t HalEepromInformationType::pageSize

The size of a single erasable page in bytes

Definition at line 109 of file [bootloader-eeprom.h](#).

7.32.2.6 uint32_t HalEepromInformationType::partSize

The total size of the external EEPROM in bytes

Definition at line 111 of file [bootloader-eeprom.h](#).

7.32.2.7 `const char* const HalEepromInformationType::partDescription`

Pointer to a string describing the attached external EEPROM

Definition at line 113 of file [bootloader-eeprom.h](#).

7.32.2.8 `uint8_t HalEepromInformationType::wordSizeBytes`

The number of bytes in a word for the external EEPROM

Definition at line 115 of file [bootloader-eeprom.h](#).

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

- [bootloader-eeprom.h](#)

7.33 InterPanHeader Struct Reference

```
#include <ami-inter-pan.h>
```

Data Fields

- `uint8_t` [messageType](#)
- `uint16_t` [panId](#)
- `bool` [hasLongAddress](#)
- [EmberNodeId](#) [shortAddress](#)
- [EmberEUI64](#) [longAddress](#)
- `uint16_t` [profileId](#)
- `uint16_t` [clusterId](#)
- `uint16_t` [groupId](#)

7.33.1 Detailed Description

A struct for keeping track of all of the header info.

A struct for keeping track of all of the interpan header info.

Definition at line 51 of file [ami-inter-pan.h](#).

7.33.2 Field Documentation

7.33.2.1 `uint8_t InterPanHeader::messageType`

Definition at line 52 of file [ami-inter-pan.h](#).

7.33.2.2 `uint16_t InterPanHeader::panId`

Definition at line 57 of file [ami-inter-pan.h](#).

7.33.2.3 `bool InterPanHeader::hasLongAddress`

Definition at line 58 of file [ami-inter-pan.h](#).

7.33.2.4 `EmberNodeId InterPanHeader::shortAddress`

Definition at line 59 of file [ami-inter-pan.h](#).

7.33.2.5 `EmberEUI64 InterPanHeader::longAddress`

Definition at line 60 of file [ami-inter-pan.h](#).

7.33.2.6 `uint16_t InterPanHeader::profileId`

Definition at line 63 of file [ami-inter-pan.h](#).

7.33.2.7 `uint16_t InterPanHeader::clusterId`

Definition at line 64 of file [ami-inter-pan.h](#).

7.33.2.8 `uint16_t InterPanHeader::groupId`

Definition at line 65 of file [ami-inter-pan.h](#).

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

- [ami-inter-pan.h](#)
- [ami-inter-pan-host.h](#)

Chapter 8

File Documentation

8.1 `_STM32F103RET_Host_API.top` File Reference

8.1.1 Detailed Description

Starting page for the Ember API documentation for the STM32F103RET Host, exclusively for building documentation. This file is used by Doxygen to generate the main page for the Ember API documentation, STM32F103RET Host.

Definition in file [_STM32F103RET_Host_API.top](#).

8.2 `_STM32F103RET_Host_API.top`

00001

8.3 `adc.h` File Reference

Macros

- #define [TEMP_SENSOR_PIN](#)
- #define [TEMP_SENSOR_PORT](#)
- #define [TEMP_SENSOR_ADC](#)
- #define [TEMP_SENSOR_ADC_CHAN](#)
- #define [TEMP_ENABLE_PIN](#)
- #define [TEMP_ENABLE_PORT](#)

Functions

- void [halInternalInitAdc](#) (void)
- uint16_t [halSampleAdc](#) (void)
- int32_t [halConvertValueToVolts](#) (uint16_t value)

8.3.1 Detailed Description

See [STM32F103RET Specific ADC](#) for documentation.

Definition in file [adc.h](#).

8.4 adc.h

```

00001
00017 #ifndef __ADC_H__
00018 #define __ADC_H__
00019
00022 #define TEMP_SENSOR_PIN      GPIO_Pin_0
00023
00025 #define TEMP_SENSOR_PORT     GPIOC
00026
00028 #define TEMP_SENSOR_ADC      ADC1
00029
00031 #define TEMP_SENSOR_ADC_CHAN 10
00032
00035 #define TEMP_ENABLE_PIN      GPIO_Pin_8
00036
00038 #define TEMP_ENABLE_PORT     GPIOA
00039
00040
00043 void halInternalInitAdc(void);
00044
00051 uint16_t halSampleAdc(void);
00052
00061 int32_t halConvertValueToVolts(uint16_t value);
00062
00063 #endif //__ADC_H__
00064

```

8.5 ami-inter-pan-host.h File Reference

Data Structures

- struct [InterPanHeader](#)
A struct for keeping track of all of the header info.

Macros

- #define [INTER_PAN_UNICAST](#)
- #define [INTER_PAN_BROADCAST](#)
- #define [INTER_PAN_MULTICAST](#)
- #define [MAX_INTER_PAN_MAC_SIZE](#)
- #define [STUB_NWK_SIZE](#)
- #define [STUB_NWK_FRAME_CONTROL](#)
- #define [MAX_STUB_APS_SIZE](#)
- #define [MAX_INTER_PAN_HEADER_SIZE](#)

Functions

- uint8_t [makeInterPanMessage](#) ([InterPanHeader](#) *headerData, uint8_t *message, uint8_t maxLength, uint8_t *payload, uint8_t payloadLength)
- uint8_t [parseInterPanMessage](#) (uint8_t *message, uint8_t messageLength, [InterPanHeader](#) *headerData)

8.5.1 Detailed Description

Utilities for sending and receiving ZigBee AMI InterPAN messages. See [Sending and Receiving Messages](#) for documentation.

Deprecated The ami-inter-pan library is deprecated and will be removed in a future release. Similar functionality is available in the Inter-PAN plugin in Application Framework.

Definition in file [ami-inter-pan-host.h](#).

8.6 ami-inter-pan-host.h

```

00001
00019 #ifndef AMI_INTER_PAN_HOST_H
00020 #define AMI_INTER_PAN_HOST_H
00021
00028 #define INTER_PAN_UNICAST 0x03
00029 #define INTER_PAN_BROADCAST 0x0B
00030 #define INTER_PAN_MULTICAST 0x0F
00031
00032
00033 // Frame control, sequence, dest PAN ID, dest, source PAN ID, source.
00034 #define MAX_INTER_PAN_MAC_SIZE (2 + 1 + 2 + 8 + 2 + 8)
00035 //Short form has a short destination.
00036
00037 // NWK stub frame has two control bytes.
00038 #define STUB_NWK_SIZE 2
00039 #define STUB_NWK_FRAME_CONTROL 0x000B
00040
00041 // APS frame control, group ID, cluster ID, profile ID
00042 #define MAX_STUB_APS_SIZE (1 + 2 + 2 + 2)
00043
00044 // Short form has no group ID.
00045 #define MAX_INTER_PAN_HEADER_SIZE \
00046 (MAX_INTER_PAN_MAC_SIZE + STUB_NWK_SIZE + MAX_STUB_APS_SIZE)
00047
00052 typedef struct {
00053     uint8_t messageType;           // one of the INTER_PAN...CAST values
00054
00055     // MAC addressing
00056     // For outgoing messages this is the destination. For incoming messages
00057     // it is the source, which always has a long address.
00058     uint16_t panId;
00059     bool hasLongAddress;           // always true for incoming messages
00060     EmberNodeId shortAddress;
00061     EmberEUI64 longAddress;
00062
00063     // APS data
00064     uint16_t profileId;
00065     uint16_t clusterId;
00066     uint16_t groupId;             // only used for INTER_PAN_MULTICAST
00067 } InterPanHeader;
00068
00075 uint8_t makeInterPanMessage(InterPanHeader *
    headerData,
00076                             uint8_t *message,
00077                             uint8_t maxLength,
00078                             uint8_t *payload,
00079                             uint8_t payloadLength);
00080
00088 uint8_t parseInterPanMessage(uint8_t *message,
00089                             uint8_t messageLength,
00090                             InterPanHeader *headerData);
00091
00092 #endif // AMI_INTER_PAN_HOST_H
00093

```

8.7 ami-inter-pan.h File Reference

Data Structures

- struct [InterPanHeader](#)
A struct for keeping track of all of the header info.

Macros

- #define [INTER_PAN_UNICAST](#)
- #define [INTER_PAN_BROADCAST](#)
- #define [INTER_PAN_MULTICAST](#)
- #define [MAX_INTER_PAN_MAC_SIZE](#)
- #define [STUB_NWK_SIZE](#)
- #define [STUB_NWK_FRAME_CONTROL](#)
- #define [MAX_STUB_APS_SIZE](#)
- #define [MAX_INTER_PAN_HEADER_SIZE](#)

Functions

- [EmberMessageBuffer makeInterPanMessage](#) ([InterPanHeader](#) *headerData, [EmberMessageBuffer](#) payload)
- [uint8_t parseInterPanMessage](#) ([EmberMessageBuffer](#) message, [uint8_t](#) startOffset, [InterPanHeader](#) *headerData)

8.7.1 Detailed Description

Utilities for sending and receiving ZigBee AMI InterPAN messages. See [Sending and Receiving Messages](#) for documentation.

Deprecated The ami-inter-pan library is deprecated and will be removed in a future release. Similar functionality is available in the Inter-PAN plugin in Application Framework.

Definition in file [ami-inter-pan.h](#).

8.8 ami-inter-pan.h

```
00001
00019 #ifndef AMI_INTER_PAN_H
00020 #define AMI_INTER_PAN_H
00021
00022 // The three types of inter-PAN messages. The values are actually the
00023 // corresponding APS frame controls.
00024 //
00025 // 0x03 is the special interPAN message type. Unicast mode is 0x00,
00026 // broadcast mode is 0x08, and multicast mode is 0x0C.
00027 //
00028
00029 #define INTER_PAN_UNICAST 0x03
00030 #define INTER_PAN_BROADCAST 0x0B
00031 #define INTER_PAN_MULTICAST 0x0F
00032
00033 // Frame control, sequence, dest PAN ID, dest, source PAN ID, source.
```

```

00034 #define MAX_INTER_PAN_MAC_SIZE (2 + 1 + 2 + 8 + 2 + 8)
00035 // Short form has a short destination.
00036
00037 // NWK stub frame has two control bytes.
00038 #define STUB_NWK_SIZE 2
00039 #define STUB_NWK_FRAME_CONTROL 0x000B
00040
00041 // APS frame control, group ID, cluster ID, profile ID
00042 #define MAX_STUB_APS_SIZE (1 + 2 + 2 + 2)
00043 // Short form has no group ID.
00044
00045 #define MAX_INTER_PAN_HEADER_SIZE \
00046 (MAX_INTER_PAN_MAC_SIZE + STUB_NWK_SIZE + MAX_STUB_APS_SIZE)
00047
00051 typedef struct {
00052     uint8_t messageType;           // one of the INTER_PAN_...CAST
00053     values
00054     // MAC addressing
00055     // For outgoing messages this is the destination. For incoming messages
00056     // it is the source, which always has a long address.
00057     uint16_t panId;
00058     bool hasLongAddress;           // always true for incoming messages
00059     EmberNodeId shortAddress;
00060     EmberEUI64 longAddress;
00061
00062     // APS data
00063     uint16_t profileId;
00064     uint16_t clusterId;
00065     uint16_t groupId;             // only used for INTER_PAN_MULTICAST
00066 } InterPanHeader;
00067
00068
00072 EmberMessageBuffer makeInterPanMessage(
00073     InterPanHeader *headerData,
00074     EmberMessageBuffer
00075     payload);
00076
00077
00082 uint8_t parseInterPanMessage(EmberMessageBuffer
00083     message,
00084     uint8_t startOffset,
00085     InterPanHeader *headerData);
00086 #endif // AMI_INTER_PAN_H
00087

```

8.9 bootload-ezsp-utils.h File Reference

Macros

- #define [TICKS_PER_QUARTER_SECOND](#)

Functions

- bool [hostBootloadUtilLaunchRequestHandler](#) (uint8_t lqi, int8_t rssi, uint16_t manufacturerId, uint8_t *hardwareTag, [EmberEUI64](#) sourceEui)
- void [hostBootloadUtilQueryResponseHandler](#) (uint8_t lqi, int8_t rssi, bool bootloadActive, uint16_t manufacturerId, uint8_t *hardwareTag, [EmberEUI64](#) targetEui, uint8_t bootloaderCapabilities, uint8_t platform, uint8_t micro, uint8_t phy, uint16_t blVersion)
- void [hostBootloadReinitHandler](#) (void)
- bool [isTheSameEui64](#) ([EmberEUI64](#) sourceEui, [EmberEUI64](#) targetEui)
- void [printLittleEndianEui64](#) (uint8_t port, [EmberEUI64](#) eui64)
- void [printBigEndianEui64](#) (uint8_t port, [EmberEUI64](#) eui64)
- [EmberStatus](#) [debugPrintf](#) (uint8_t port, PGM_P formatString,...)

Variables

- uint16_t [nodeBlVersion](#)
- uint8_t [nodePlat](#)
- uint8_t [nodeMicro](#)
- uint8_t [nodePhy](#)
- EzspStatus [bootloadEzspLastError](#)
- EzspStatus [ignoreNextEzspError](#)

8.9.1 Detailed Description

Utilities used for performing stand-alone bootloading over EZSP. See [Bootloading](#) for documentation.

Definition in file [bootload-ezsp-utils.h](#).

8.10 bootload-ezsp-utils.h

```

00001
00016 // application timers are based on quarter second intervals, each
00017 // quarter second is measured in millisecond ticks. This value defines
00018 // the approximate number of millisecond ticks in a quarter second.
00019 // Account for variations in system timers.
00020 #ifdef AVR_ATMEGA_32
00021 #define TICKS_PER_QUARTER_SECOND 225
00022 #else
00023 #define TICKS_PER_QUARTER_SECOND 250
00024 #endif
00025
00026 // Node build info
00027 extern uint16_t nodeBlVersion;
00028 extern uint8_t nodePlat;
00029 extern uint8_t nodeMicro;
00030 extern uint8_t nodePhy;
00031
00032
00033 // Both of these need to be correctly handled in the applications's
00034 // ezspErrorHandler().
00035 // ezsp error info
00036 extern EzspStatus bootloadEzspLastError;
00037 // If this is not EZSP_SUCCESS, the next call to ezspErrorHandler()
00038 // will ignore this error.
00039 extern EzspStatus ignoreNextEzspError;
00040
00041
00042 // *****
00043 // Callback functions used by the bootload library.
00044
00072 bool hostBootloadUtilLaunchRequestHandler(
    uint8_t lqi,
                                int8_t rssi,
                                uint16_t manufacturerId,
                                uint8_t *hardwareTag,
                                EmberEUI64 sourceEui);
00073
00121 void hostBootloadUtilQueryResponseHandler(
    uint8_t lqi,
                                int8_t rssi,
                                bool bootloaderActive,
                                uint16_t manufacturerId,
                                uint8_t *hardwareTag,
                                EmberEUI64 targetEui,
                                uint8_t bootloaderCapabilities,
                                uint8_t platform,
                                uint8_t micro,
                                uint8_t phy,
                                uint16_t blVersion);
00132

```

```

00140 void hostBootloadReinitHandler(void);
00141
00142
00143 // Support routines in the bootloader utils library
00144
00156 bool isTheSameEui64(EmberEUI64 sourceEui, EmberEUI64
    targetEui);
00157
00168 void printLittleEndianEui64(uint8_t port, EmberEUI64
    eui64);
00169
00180 void printBigEndianEui64(uint8_t port, EmberEUI64
    eui64);
00181
00199 EmberStatus debugPrintf(uint8_t port, PGM_P formatString,
    ...);
00200

```

8.11 bootload-utils.h File Reference

Macros

- `#define BOOTLOAD_HARDWARE_TAG_SIZE`

Enumerations

- `enum bootloadMode { BOOTLOAD_MODE_NONE, BOOTLOAD_MODE_PASSTHRU }`
- `enum bootloadState { BOOTLOAD_STATE_NORMAL, BOOTLOAD_STATE_QUERY, BOOTLOAD_STATE_WAIT_FOR_AUTH_CHALLENGE, BOOTLOAD_STATE_WAIT_FOR_AUTH_RESPONSE, BOOTLOAD_STATE_DELAY_BEFORE_START, BOOTLOAD_STATE_START_UNICAST_BOOTLOAD, BOOTLOAD_STATE_START_BROADCAST_BOOTLOAD, BOOTLOAD_STATE_START_SENDING_IMAGE, BOOTLOAD_STATE_SENDING_IMAGE, BOOTLOAD_STATE_WAIT_FOR_IMAGE_ACK, BOOTLOAD_STATE_WAIT_FOR_COMPLETE_ACK, BOOTLOAD_STATE_DONE }`

Functions

- `void bootloadUtilInit (uint8_t appPort, uint8_t bootloadPort)`
- `EmberStatus bootloadUtilSendRequest (EmberEUI64 targetEui, uint16_t mfgId, uint8_t hardwareTag[BOOTLOAD_HARDWARE_TAG_SIZE], uint8_t encryptKey[BOOTLOAD_AUTH_COMMON_SIZE], bootloadMode mode)`
- `void bootloadUtilSendQuery (EmberEUI64 target)`
- `void bootloadUtilStartBootload (EmberEUI64 target, bootloadMode mode)`
- `void bootloadUtilTick (void)`
- `bool bootloadUtilLaunchRequestHandler (uint16_t manufacturerId, uint8_t hardwareTag[BOOTLOAD_HARDWARE_TAG_SIZE], EmberEUI64 sourceEui)`
- `void bootloadUtilQueryResponseHandler (bool bootloaderActive, uint16_t manufacturerId, uint8_t hardwareTag[BOOTLOAD_HARDWARE_TAG_SIZE], EmberEUI64 targetEui, uint8_t bootloaderCapabilities, uint8_t platform, uint8_t micro, uint8_t phy, uint16_t blVersion)`
- `void bootloadUtilSendAuthResponse (EmberEUI64 target)`

Authentication Challenge and Response

The authentication challenge and response must be the same size. The size is chosen to be evenly divisible by the size of a 128-bit AES block.

- #define [BOOTLOAD_AUTH_COMMON_SIZE](#)
- #define [BOOTLOAD_AUTH_CHALLENGE_SIZE](#)
- #define [BOOTLOAD_AUTH_RESPONSE_SIZE](#)

Bootload State Variables

Used to check whether a bootloading process is currently happening.

- #define [IS_BOOTLOADING](#)
- [bootloadState](#) blState

8.11.1 Detailed Description

Utilities used for performing stand-alone bootloading. See [Bootloading](#) for documentation.

Definition in file [bootload-utils.h](#).

8.12 bootload-utils.h

```

00001
00058 // *****
00059 // Literals that are needed by the application.
00060
00066 #define BOOTLOAD_AUTH_COMMON_SIZE 16
00067 #define BOOTLOAD_AUTH_CHALLENGE_SIZE BOOTLOAD_AUTH_COMMON_SIZE
00068 #define BOOTLOAD_AUTH_RESPONSE_SIZE BOOTLOAD_AUTH_COMMON_SIZE
00069
00076 #define BOOTLOAD_HARDWARE_TAG_SIZE 16
00077
00078 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00079
00082 enum bootloadMode
00083 #else
00084 typedef uint8_t bootloadMode;
00085 enum
00086 #endif
00087 {
00089     BOOTLOAD_MODE_NONE,
00091     BOOTLOAD_MODE_PASSTHRU,
00092 };
00093
00094 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00095
00106 enum bootloadState
00107 #else
00108 typedef uint8_t bootloadState;
00109 enum
00110 #endif
00111 {
00112     BOOTLOAD_STATE_NORMAL,
00113     BOOTLOAD_STATE_QUERY,
00114     BOOTLOAD_STATE_WAIT_FOR_AUTH_CHALLENGE,
00115     BOOTLOAD_STATE_WAIT_FOR_AUTH_RESPONSE,
00116     BOOTLOAD_STATE_DELAY_BEFORE_START,
00117     BOOTLOAD_STATE_START_UNICAST_BOOTLOAD,
00118     BOOTLOAD_STATE_START_BROADCAST_BOOTLOAD,
00119     BOOTLOAD_STATE_START_SENDING_IMAGE,

```

```

00120 BOOTLOAD_STATE_SENDING_IMAGE,
00121 BOOTLOAD_STATE_WAIT_FOR_IMAGE_ACK,
00122 BOOTLOAD_STATE_WAIT_FOR_COMPLETE_ACK,
00123 BOOTLOAD_STATE_DONE
00124 };
00125
00126
00127 // *****
00128 // Public functions that are called by the application.
00129
00143 void bootloadUtilInit(uint8_t appPort, uint8_t bootloadPort);
00144
00166 EmberStatus bootloadUtilSendRequest(
00167     EmberEUI64 targetEui,
00168     uint16_t mfgId,
00169     uint8_t hardwareTag[
00170         BOOTLOAD_HARDWARE_TAG_SIZE],
00171     uint8_t encryptKey[BOOTLOAD_AUTH_COMMON_SIZE],
00172     bootloadMode mode);
00173
00184 void bootloadUtilSendQuery(EmberEUI64 target);
00185
00203 void bootloadUtilStartBootload(EmberEUI64
00204     target, bootloadMode mode);
00205
00210 void bootloadUtilTick(void);
00211
00212
00213 // *****
00214 // Callback functions used by the bootload library.
00215
00234 bool bootloadUtilLaunchRequestHandler(uint16_t
00235     manufacturerId,
00236     uint8_t hardwareTag[
00237         BOOTLOAD_HARDWARE_TAG_SIZE],
00238     EmberEUI64 sourceEui);
00239
00267 void bootloadUtilQueryResponseHandler(bool
00268     bootloaderActive,
00269     uint16_t manufacturerId,
00270     uint8_t hardwareTag[
00271         BOOTLOAD_HARDWARE_TAG_SIZE],
00272     EmberEUI64 targetEui,
00273     uint8_t bootloaderCapabilities,
00274     uint8_t platform,
00275     uint8_t micro,
00276     uint8_t phy,
00277     uint16_t blVersion);
00278
00289 void bootloadUtilSendAuthResponse(EmberEUI64
00290     target);
00291
00292 // *****
00293 // Bootload state variables
00294
00299 extern bootloadState blState;
00300 #define IS_BOOTLOADING ((blState != BOOTLOAD_STATE_NORMAL) && \
00301     (blState != BOOTLOAD_STATE_DONE))
00302

```

8.13 bootloader-eeeprom.h File Reference

Data Structures

- struct [HalEepromInformationType](#)

This structure defines a variety of information about the attached external EEPROM device.

Macros

- #define [EEPROM_PAGE_SIZE](#)
- #define [EEPROM_FIRST_PAGE](#)
- #define [EEPROM_IMAGE_START](#)
- #define [EEPROM_SUCCESS](#)
- #define [EEPROM_ERR](#)
- #define [EEPROM_ERR_MASK](#)
- #define [EEPROM_ERR_PG_BOUNDARY](#)
- #define [EEPROM_ERR_PG_SZ](#)
- #define [EEPROM_ERR_WRT_DATA](#)
- #define [EEPROM_ERR_IMG_SZ](#)
- #define [EEPROM_ERR_ADDR](#)
- #define [EEPROM_ERR_INVALID_CHIP](#)
- #define [EEPROM_ERR_ERASE_REQUIRED](#)
- #define [EEPROM_ERR_NO_ERASE_SUPPORT](#)

EEPROM interaction functions.

- #define [EEPROM_INFO_VERSION](#)
- #define [EEPROM_INFO_MAJOR_VERSION](#)
- #define [EEPROM_INFO_MAJOR_VERSION_MASK](#)
- #define [EEPROM_INFO_MIN_VERSION_WITH_WORD_SIZE_SUPPORT](#)
- #define [EEPROM_CAPABILITIES_ERASE_SUPPORTED](#)
- #define [EEPROM_CAPABILITIES_PAGE_ERASE_REQD](#)
- uint8_t [halEepromInit](#) (void)
- void [halEepromShutdown](#) (void)
- const [HalEepromInformationType](#) * [halEepromInfo](#) (void)
- bool [halEepromBusy](#) (void)
- uint8_t [halEepromRead](#) (uint32_t address, uint8_t *data, uint16_t len)
- uint8_t [halEepromWrite](#) (uint32_t address, const uint8_t *data, uint16_t len)
- uint8_t [halEepromErase](#) (uint32_t address, uint32_t totalLength)

8.14 bootloader-eeeprom.h

```

00001
00017 #ifndef __BOOTLOADER_EEPROM_H__
00018 #define __BOOTLOADER_EEPROM_H__
00019
00023 #define EEPROM_PAGE_SIZE      (128u1)
00024
00028 #define EEPROM_FIRST_PAGE    (0)
00029
00034 #define EEPROM_IMAGE_START    (EEPROM_FIRST_PAGE*EEPROM_PAGE_SIZE)
00035
00038 #define EEPROM_SUCCESS 0
00039
00042 #define EEPROM_ERR 1
00043
00046 #define EEPROM_ERR_MASK 0x80
00047
00050 #define EEPROM_ERR_PG_BOUNDARY 0x81
00051
00054 #define EEPROM_ERR_PG_SZ 0x82
00055
00058 #define EEPROM_ERR_WRT_DATA 0x83
00059

```



```

00062 #define EEPROM_ERR_IMG_SZ 0x84
00063
00066 #define EEPROM_ERR_ADDR 0x85
00067
00070 #define EEPROM_ERR_INVALID_CHIP 0x86
00071
00074 #define EEPROM_ERR_ERASE_REQUIRED 0x87
00075
00078 #define EEPROM_ERR_NO_ERASE_SUPPORT 0x88
00079
00090 uint8_t halEepromInit(void);
00091
00094 void halEepromShutdown(void);
00095
00099 typedef struct {
00101     uint16_t version;
00103     uint16_t capabilitiesMask;
00105     uint16_t pageEraseMs;
00107     uint16_t partEraseMs;
00109     uint32_t pageSize;
00111     uint32_t partSize;
00113     const char * const partDescription;
00115     uint8_t wordSizeBytes;
00116 } HalEepromInformationType;
00117
00120 // NOTE: We explicitly did NOT update the version number to 0x0102
00121 // because we did not update the associated drivers to include word size
00122 // support.
00123 // The existing host drivers have only 1-byte word sizes.
00123 #define EEPROM_INFO_VERSION (0x0101)
00124 #define EEPROM_INFO_MAJOR_VERSION (0x0100)
00125 #define EEPROM_INFO_MAJOR_VERSION_MASK (0xFF00)
00126 // *** Eeprom info version history: ***
00127 // 0x0102 - Added a word size field to specify the number of bytes per flash
00128 // word in the EEPROM. Writes should always be aligned to the word
00129 // size and have a length that is a multiple of the word size.
00130 // 0x0101 - Initial version
00131 #define EEPROM_INFO_MIN_VERSION_WITH_WORD_SIZE_SUPPORT 0x0102
00132
00135 #define EEPROM_CAPABILITIES_ERASE_SUPPORTED (0x0001)
00136
00140 #define EEPROM_CAPABILITIES_PAGE_ERASE_REQD (0x0002)
00141
00151 const HalEepromInformationType *halEepromInfo
(void);
00152
00161 bool halEepromBusy(void);
00162
00178 uint8_t halEepromRead(uint32_t address, uint8_t *data, uint16_t
len);
00179
00201 uint8_t halEepromWrite(uint32_t address, const uint8_t *data,
uint16_t len);
00202
00223 uint8_t halEepromErase(uint32_t address, uint32_t totalLength);
00224
00225
00228 #endif // __BOOTLOADER_EEPROM_H__
00229

```

8.15 button-common.h File Reference

```
#include "button-specific.h"
```

Functions

- void [halInternalInitButton](#) (void)
- uint8_t [halButtonState](#) (uint8_t button)
- uint8_t [halButtonPinState](#) (uint8_t button)
- void [halButtonIsr](#) (uint8_t button, uint8_t state)

Button State Definitions

A set of numerical definitions for use with the button APIs indicating the state of a button.

- `#define` [BUTTON_PRESSED](#)
- `#define` [BUTTON_RELEASED](#)

8.15.1 Detailed Description

See [Button Control](#) and micro specific modules for documentation.

Definition in file [button-common.h](#).

8.16 button-common.h

```

00001
00017 #ifndef __BUTTON_COMMON_H__
00018 #define __BUTTON_COMMON_H__
00019
00020
00029 #define BUTTON_PRESSED 1
00030
00033 #define BUTTON_RELEASED 0
00034
00040 void halInternalInitButton(void);
00041
00053 uint8_t halButtonState(uint8_t button);
00054
00067 uint8_t halButtonPinState(uint8_t button);
00068
00083 void halButtonIsr(uint8_t button, uint8_t state);
00084
00085
00086 //Pull in the micro specific button definitions. The specific header is chosen
00087 //by the build include path pointing at the appropriate directory.
00088 #include "button-specific.h"
00089
00090
00091 #endif //__BUTTON_COMMON_H__
00092

```

8.17 button-specific.h File Reference

Macros

- `#define` [BUTTON0](#)
- `#define` [BUTTON0_PIN](#)
- `#define` [BUTTON0_PORT](#)
- `#define` [BUTTON0_EXTL_SOURCE_PORT](#)
- `#define` [BUTTON0_EXTL_SOURCE_PIN](#)
- `#define` [BUTTON0_IRQ](#)
- `#define` [BUTTON1](#)
- `#define` [BUTTON1_PIN](#)
- `#define` [BUTTON1_PORT](#)
- `#define` [BUTTON1_EXTL_SOURCE_PORT](#)
- `#define` [BUTTON1_EXTL_SOURCE_PIN](#)
- `#define` [BUTTON1_IRQ](#)
- `#define` [BUTTON01_ISR](#)

8.17.1 Detailed Description

See [Button Control](#) and [STM32F103RET Specific Button](#) for documentation.

Definition in file [button-specific.h](#).

8.18 button-specific.h

```

00001
00020 #ifndef __BUTTON_SPECIFIC_H__
00021 #define __BUTTON_SPECIFIC_H__
00022
00023
00026 #define BUTTON0                0 //Just a simple identifier for comparisons
00027
00030 #define BUTTON0_PIN            GPIO_Pin_10
00031
00034 #define BUTTON0_PORT          GPIOB
00035
00039 #define BUTTON0_EXTI_SOURCE_PORT GPIO_PortSourceGPIOB
00040
00044 #define BUTTON0_EXTI_SOURCE_PIN GPIO_PinSource10
00045
00048 #define BUTTON0_IRQ            EXTI15_10_IRQn
00049
00050
00053 #define BUTTON1                1 //Just a simple identifier for comparisons
00054
00057 #define BUTTON1_PIN            GPIO_Pin_11
00058
00061 #define BUTTON1_PORT          GPIOB
00062
00066 #define BUTTON1_EXTI_SOURCE_PORT GPIO_PortSourceGPIOB
00067
00071 #define BUTTON1_EXTI_SOURCE_PIN GPIO_PinSource11
00072
00075 #define BUTTON1_IRQ            EXTI15_10_IRQn
00076
00077
00081 #define BUTTON01_ISR           EXTI15_10_IRQHandler
00082
00083
00084 #endif //__BUTTON_SPECIFIC_H__
00085

```

8.19 buzzer.h File Reference

Functions

- void [halPlayTune_P](#) (uint8_t PGM *tune, bool bkg)
- void [halStartBuzzerTone](#) (uint16_t frequency)
- void [halStopBuzzerTone](#) (void)

Variables

- uint8_t PGM [hereIamTune](#) []

Note Definitions

Flats are used instead of sharps because # is a special character.

- #define [NOTE_C3](#)
- #define [NOTE_Db3](#)
- #define [NOTE_D3](#)
- #define [NOTE_Eb3](#)
- #define [NOTE_E3](#)
- #define [NOTE_F3](#)
- #define [NOTE_Gb3](#)
- #define [NOTE_G3](#)
- #define [NOTE_Ab3](#)
- #define [NOTE_A3](#)
- #define [NOTE_Bb3](#)
- #define [NOTE_B3](#)
- #define [NOTE_C4](#)
- #define [NOTE_Db4](#)
- #define [NOTE_D4](#)
- #define [NOTE_Eb4](#)
- #define [NOTE_E4](#)
- #define [NOTE_F4](#)
- #define [NOTE_Gb4](#)
- #define [NOTE_G4](#)
- #define [NOTE_Ab4](#)
- #define [NOTE_A4](#)
- #define [NOTE_Bb4](#)
- #define [NOTE_B4](#)
- #define [NOTE_C5](#)
- #define [NOTE_Db5](#)
- #define [NOTE_D5](#)
- #define [NOTE_Eb5](#)
- #define [NOTE_E5](#)
- #define [NOTE_F5](#)
- #define [NOTE_Gb5](#)
- #define [NOTE_G5](#)
- #define [NOTE_Ab5](#)
- #define [NOTE_A5](#)
- #define [NOTE_Bb5](#)
- #define [NOTE_B5](#)

8.19.1 Detailed Description

See [STM32F103RET Specific Buzzer](#) for documentation.

Definition in file [buzzer.h](#).

8.20 buzzer.h

```

00001
00017 #ifndef __BUZZER_H__
00018 #define __BUZZER_H__
00019
00020
00032 #define NOTE_C3 (130/4)
00033 #define NOTE_Db3 (138/4)
00034 #define NOTE_D3 (146/4)
00035 #define NOTE_Eb3 (155/4)
00036 #define NOTE_E3 (164/4)
00037 #define NOTE_F3 (174/4)
00038 #define NOTE_Gb3 (185/4)
00039 #define NOTE_G3 (196/4)
00040 #define NOTE_Ab3 (207/4)
00041 #define NOTE_A3 (220/4)
00042 #define NOTE_Bb3 (233/4)
00043 #define NOTE_B3 (246/4)
00044 #define NOTE_C4 (261/4)
00045 #define NOTE_Db4 (277/4)
00046 #define NOTE_D4 (293/4)
00047 #define NOTE_Eb4 (311/4)
00048 #define NOTE_E4 (329/4)
00049 #define NOTE_F4 (349/4)
00050 #define NOTE_Gb4 (369/4)
00051 #define NOTE_G4 (392/4)
00052 #define NOTE_Ab4 (415/4)
00053 #define NOTE_A4 (440/4)
00054 #define NOTE_Bb4 (466/4)
00055 #define NOTE_B4 (493/4)
00056 #define NOTE_C5 (523/4)
00057 #define NOTE_Db5 (554/4)
00058 #define NOTE_D5 (587/4)
00059 #define NOTE_Eb5 (622/4)
00060 #define NOTE_E5 (659/4)
00061 #define NOTE_F5 (698/4)
00062 #define NOTE_Gb5 (739/4)
00063 #define NOTE_G5 (783/4)
00064 #define NOTE_Ab5 (830/4)
00065 #define NOTE_A5 (880/4)
00066 #define NOTE_Bb5 (932/4)
00067 #define NOTE_B5 (987/4)
00068
00096 void halPlayTune_P(uint8_t PGM *tune, bool bkg);
00097
00098
00104 void halStartBuzzerTone(uint16_t frequency);
00105
00106
00109 void halStopBuzzerTone(void);
00110
00111
00115 extern uint8_t PGM hereIamTune[];
00116
00117 #endif //__BUZZER_H__
00118

```

8.21 cbke-crypto-engine.h File Reference

Functions

- [EmberStatus emberGetCertificate](#) ([EmberCertificateData](#) *result)
- [EmberStatus emberGenerateCbkeKeys](#) (void)
- [EmberStatus emberCalculateSmacs](#) (bool amInitiator, [EmberCertificateData](#) *partnerCert, [EmberPublicKeyData](#) *partnerEphemeralPublicKey)
- [EmberStatus emberClearTemporaryDataMaybeStoreLinkKey](#) (bool storeLinkKey)
- [EmberStatus emberDsaSign](#) ([EmberMessageBuffer](#) messageToSign)
- void [emberGenerateCbkeKeysHandler](#) ([EmberStatus](#) status, [EmberPublicKeyData](#) *ephemeralPublicKey)

- void `emberCalculateSmacsHandler` (`EmberStatus` status, `EmberSmacData` *initiatorSmac, `EmberSmacData` *responderSmac)
- void `emberDsaSignHandler` (`EmberStatus` status, `EmberMessageBuffer` signedMessage)
- `EmberStatus` `emberSetPreinstalledCbkeData` (`EmberPublicKeyData` *caPublic, `EmberCertificateData` *myCert, `EmberPrivateKeyData` *myKey)
- bool `emberGetStackCertificateEui64` (`EmberEUI64` certEui64)
- `EmberStatus` `emberDsaVerify` (`EmberMessageDigest` *digest, `EmberCertificateData` *signerCertificate, `EmberSignatureData` *receivedSig)
- void `emberDsaVerifyHandler` (`EmberStatus` status)
- `EmberStatus` `emberGetCertificate283k1` (`EmberCertificate283k1Data` *result)
- `EmberStatus` `emberGenerateCbkeKeys283k1` (void)
- `EmberStatus` `emberCalculateSmacs283k1` (bool amInitiator, `EmberCertificate283k1Data` *partnerCert, `EmberPublicKey283k1Data` *partnerEphemeralPublicKey)
- `EmberStatus` `emberClearTemporaryDataMaybeStoreLinkKey283k1` (bool storeLinkKey)
- void `emberGenerateCbkeKeysHandler283k1` (`EmberStatus` status, `EmberPublicKey283k1Data` *ephemeralPublicKey)
- void `emberCalculateSmacsHandler283k1` (`EmberStatus` status, `EmberSmacData` *initiatorSmac, `EmberSmacData` *responderSmac)
- `EmberStatus` `emberSetPreinstalledCbkeData283k1` (`EmberPublicKey283k1Data` *caPublic, `EmberCertificate283k1Data` *myCert, `EmberPrivateKey283k1Data` *myKey)
- `EmberStatus` `emberDsaVerify283k1` (`EmberMessageDigest` *digest, `EmberCertificate283k1Data` *signerCertificate, `EmberSignature283k1Data` *receivedSig)

8.21.1 Detailed Description

EmberZNet Smart Energy security API. See [Smart Energy Security](#) for documentation.

Definition in file [cbke-crypto-engine.h](#).

8.21.2 Function Documentation

8.21.2.1 `EmberStatus emberGetCertificate283k1 (EmberCertificate283k1Data * result)`

Retrieves the implicit certificate stored in the MFG tokens of the device.

Parameters

<i>result</i>	A pointer to an EmberCertificate283k1Data structure where the retrieved certificate will be stored.
---------------	---

Returns

[EMBER_SUCCESS](#) if the certificate was successfully retrieved. [EMBER_ERR_FATAL](#) if the token contains uninitialized data.

8.21.2.2 `EmberStatus emberGenerateCbkeKeys283k1 (void)`

This function begins the process of generating an ephemeral public/private ECC key pair.

If no other ECC operation is going on, it will immediately return with [EMBER_OPERATION_IN_PROGRESS](#). It will delay a period of time to let APS retries take place, but then it will shutdown the radio and consume the CPU processing until the key generation is complete. This may take up to 1 second.

The generated results of the key generation is returned via [emberGenerateCbkeKeysHandler283k1\(\)](#).

Returns

[EMBER_OPERATION_IN_PROGRESS](#) if the stack has queued up the operation for execution.

8.21.2.3 `EmberStatus emberCalculateSmacs283k1 (bool amInitiator,
EmberCertificate283k1Data * partnerCert, EmberPublicKey283k1Data *
partnerEphemeralPublicKey)`

This function will begin the process of generating the shared secret, the new link key, and the Secured Message Authentication Code (SMAC).

If no other ECC operation is going on, it will immediately return with [EMBER_OPERATION_IN_PROGRESS](#). It will delay a period of time to let APS retries take place, but then it will shutdown the radio and consume the CPU processing until SMACs calculations are complete. This may take up to 3.5 seconds.

The calculated SMACS are returned via [emberCalculateSmacsHandler\(\)](#).

Parameters

<i>amInitiator</i>	This bool indicates whether or not the device is the one that initiated the CBKE with the remote device, or whether it was the responder to the exchange.
<i>partnerCert</i>	A pointer to an EmberCertificate283k1Data structure that contains the CBKE partner's implicit certificate.
<i>partner-Ephemeral-PublicKey</i>	A pointer to an EmberPublicKey283k1Data structure that contains the CBKE partner's ephemeral public key.

Returns

[EMBER_OPERATION_IN_PROGRESS](#) if the stack has queued up the operation for execution.

8.21.2.4 `EmberStatus emberClearTemporaryDataMaybeStoreLinkKey283k1 (bool
storeLinkKey)`

This function should be called when all CBKE 283k1 operations are done. Any temporary data created during calls to [emberGenerateCbkeKeys283k1\(\)](#) or [emberCalculateSmacs283k1\(\)](#) is wiped out. If the local device has validated that the partner device has generated the same SMACS as itself, it should set 'storeLinkKey' to true. Otherwise it should pass in false.

Parameters

<i>storeLinkKey</i>	This tells the stack whether to store the newly generated link key, or discard it.
---------------------	--

Returns

If *storeLinkKey* is false, this function returns [EMBER_ERR_FATAL](#) always. If *storeLinkKey* is true, then this function returns the results of whether or not the link key was stored. [EMBER_SUCCESS](#) is returned when key was stored successfully.

- 8.21.2.5** `void emberGenerateCbkeKeysHandler283k1 (EmberStatus status,
EmberPublicKey283k1Data * ephemeralPublicKey)`
- 8.21.2.6** `void emberCalculateSmacsHandler283k1 (EmberStatus status,
EmberSmacData * initiatorSmac, EmberSmacData * responderSmac)`
- 8.21.2.7** `EmberStatus emberSetPreinstalledCbkeData283k1 (EmberPublic-
Key283k1Data * caPublic, EmberCertificate283k1Data * myCert,
EmberPrivateKey283k1Data * myKey)`
- 8.21.2.8** `EmberStatus emberDsaVerify283k1 (EmberMessageDigest * digest,
EmberCertificate283k1Data * signerCertificate, EmberSignature283k1Data
* receivedSig)`

8.22 cbke-crypto-engine.h

```

00001
00030 EmberStatus emberGetCertificate(
    EmberCertificateData* result);
00031
00047 EmberStatus emberGenerateCbkeKeys(void);
00048
00071 EmberStatus emberCalculateSmacs(bool amInitiator,
00072                                EmberCertificateData*
    partnerCert,
00073                                EmberPublicKeyData*
    partnerEphemeralPublicKey);
00074
00089 EmberStatus emberClearTemporaryDataMaybeStoreLinkKey
    (bool storeLinkKey);
00090
00091 /* @brief LEGACY FUNCTION: This functionality has been replaced by a single
00092  * bit in the ::EmberApsFrame, ::EMBER_APS_OPTION_DSA_SIGN. Devices wishing
00093  * to send signed messages should use that as it requires fewer function calls
00094  * and message buffering. emberDsaSignHandler() is still called when using
00095  * ::EMBER_APS_OPTION_DSA_SIGN. However, this function is still supported.
00096  *
00097  * This function begins the process of signing the passed message
00098  * contained within the buffer. If no other ECC operation is going on,
00099  * it will immediately return with ::EMBER_OPERATION_IN_PROGRESS.
00100  * It will delay a period of time to let APS retries take place, but then it
00101  * will shutdown the radio and consume the CPU processing until the signing
00102  * is complete. This may take up to 1 second.
00103  *
00104  * The signed message will be returned in ::emberDsaSignHandler().
00105  *
00106  * Note that the last byte of the buffer contents passed to this function has
00107  * special significance. As the typical use case for DSA signing is to sign
    the
00108  * ZCL payload of a DRLC Report Event Status message in SE 1.0, there is often
00109  * both a signed portion (ZCL payload) and an unsigned portion (ZCL header).
00110  * The last byte in the content of messageToSign is therefore used as a
00111  * special indicator to signify how many bytes of leading data in the buffer

```



```

00112 * should be excluded from consideration during the signing process. If the
00113 * signature needs to cover the entire buffer (all bytes except last one),
00114 * the caller should ensure that the last byte of the buffer contents is 0.
00115 * When the signature operation is complete, this final byte will be replaced
00116 * by the signature type indicator (0x01 for ECDSA signatures), and the
00117 * actual signature will be appended to the buffer after this byte.
00118 *
00119 * @param messageToSign The message buffer containing the complete message,
00120 * both the to-be-signed portion as well as any leading data excluded from
00121 * the signing operation. See note above regarding special requirements
00122 * for this buffer.
00123 *
00124 * @return ::EMBER_OPERATION_IN_PROGRESS if the stack has queued up the
00125 * operation for execution. ::EMBER_INVALID_CALL if the operation can't be
00126 * performed in this context (possibly because another ECC operation is
00127 * pending.)
00128 */
00129 EmberStatus emberDsaSign(EmberMessageBuffer
    messageToSign);
00130
00131
00132 /* @brief This function is an application callback that must be defined
00133 * when using CBKE. It is called when the ephemeral key generation operation
00134 * is complete. The newly generated public key is passed back to the
00135 * application to be sent to the CBKE partner over-the-air. Internally
00136 * the stack saves the public and private key pair until it the function
00137 * ::emberClearTemporaryDataMaybeStoreLinkKey() is called by the application.
00138 *
00139 * @param status This is the ::EmberStatus value indicating the success or
00140 * failure of the operation.
00141 * @param ephemeralPublicKey A pointer to an ::EmberPublicKeyData structure
00142 * containing the newly generated public key.
00143 */
00144 void emberGenerateCbkeKeysHandler(EmberStatus
    status,
00145
    EmberPublicKeyData*
    ephemeralPublicKey);
00146
00147 /* @brief This function is an application callback that must be defined
00148 * when using CBKE. It is called when the shared secret generation is
00149 * complete and the link key and SMACs have been derived. The link key is
00150 * stored in a temporary location until the application decides to
00151 * store or discard the key by calling
00152 * ::emberClearTemporaryDataMaybeStoreLinkKey().
00153 *
00154 * @param status This is the ::EmberStatus value indicating the success or
00155 * failure of the operation.
00156 * @param initiatorSmac This is a pointer to the ::EmberSmacData structure
00157 * to the initiator's version of the SMAC.
00158 * @param responderSmac This is a pointer to the ::EmberSmacData structure
00159 * to the responder's version of the SMAC.
00160 */
00161 void emberCalculateSmacsHandler(EmberStatus
    status,
00162
    EmberSmacData* initiatorSmac,
00163
    EmberSmacData* responderSmac);
00164
00165 /* @brief This function is an application callback that must be defined
00166 * when using CBKE. This callback is provided to the application to let
00167 * it know that the ECC operations have completed and the radio has been turned
00168 * back on. When using the sign-and-send option of the ::EmberApsFrame,
00169 * ::EMBER_APS_OPTION_DSA_SIGN, the handler will NOT return the complete
00170 * signed message. This callback is merely informative. If ::emberDsaSign()
00171 * has been called, the message plus signature will be returned to the caller
00172 * and it must be sent separately by one of the message send primitives
00173 * (such as ::emberSendUnicast()).
00174 *
00175 * @param status This is the ::EmberStatus value indicating the success or
00176 * failure of the operation.
00177 * @param signedMessage This is the ::EmberMessageBuffer indicating the newly
00178 * signed message, if ::emberDsaSign() was called. If message was signed
00179 * using ::EMBER_APS_OPTION_DSA_SIGN then this will be
00180 * ::EMBER_NULL_MESSAGE_BUFFER.
00181 */
00182 void emberDsaSignHandler(EmberStatus status,
    EmberMessageBuffer signedMessage);
00183
00184
00185
00186 /* @brief This function is used to update the Smart Energy certificate,
00187 * CA public key, and local private key that the device uses for CBKE.

```

```

00188 * The preferred method for adding certificates is to pre-install them
00189 * in MFG tokens when the chip is manufactured. However this function
00190 * allows the certificate to be updated at runtime after the device has
00191 * been deployed.
00192 * The behavior of this function differs based on the hardware platform.
00193 *
00194 * For the 2xx:
00195 * To use this functionality the application must also set
00196 * the stack configuration value ::EMBER_CERTIFICATE_TABLE_SIZE to 1.
00197 * Attempts to call this function with ::EMBER_CERTIFICATE_TABLE_SIZE of 0
00198 * will return ::EMBER_SECURITY_CONFIGURATION_INVALID.
00199 * The passed security data will be persistently stored in stack tokens.
00200 * The certificate contains the EUI64 it is associated with. If that
00201 * EUI64 matches the EUI64 currently in use by the device, this
00202 * function may be called at any time, even while running in a network.
00203 * If the EUI64 does not match, this function may only be called when the
00204 * network is in a state of ::EMBER_NO_NETWORK. Attempts to do otherwise
00205 * will result in a return value of ::EMBER_INVALID_CALL.
00206 *
00207 * For the 3xx:
00208 * This function allows a one-time write of the MFG token if it has not
00209 * already been set. It does NOT utilize the ::EMBER_CERTIFICATE_TABLE_SIZE
00210 * so that should remain set at 0. Attempts to write the certificate that
00211 * has already been written will return a result of
00212 * ::EMBER_ERR_FLASH_WRITE_INHIBITED. If the EUI64 in the certificate is
00213 * the same as the current EUI of the device then this function may be called
00214 * while the stack is up. If the EUI in the certificate is different than
00215 * the current value, this function may only be called when the network is in
00216 * a state of ::EMBER_NO_NETWORK. Attempts to do otherwise will result in a
00217 * return value of ::EMBER_INVALID_CALL. If the EUI in the certificate is
00218 * different than the current value this function will also write the
00219 * Custom EUI64 MFG token. If that token has already been written the
    operation
00220 * will fail and return a result of ::EMBER_BAD_ARGUMENT.
00221 * If all the above criteria is met the token will be written and
00222 * ::EMBER_SUCCESS will be returned.
00223 *
00224 * @note The device will immediately and persistently <b>change its EUI64
00225 * to match the value in the certificate</b>.
00226 *
00227 * @param caPublic A pointer to the CA public key data that will be stored
00228 * in stack tokens.
00229 * @param myCert A pointer to the certificate data that will be stored in
00230 * stack tokens.
00231 * @param mykey A pointer to the private key data that will be stored in
00232 * stack tokens.
00233 * @return The ::EmberStatus value indicating success or failure of the
00234 * operation.
00235 */
00236 EmberStatus emberSetPreinstalledCbkeData
    (EmberPublicKeyData* caPublic,
00237
00238     EmberCertificateData
00239     * myCert,
00240     EmberPrivateKeyData
00241     * myKey);
00242
00243 /* @brief This function retrieves the EUI64 from the stack token
00244 * Smart Energy Certificate (it does not examine the MFG token certificate)
00245 * and returns the value in the "Subject" field (the EUI64) to the caller.
00246 * If no stack token is set, the ::EMBER_CERTIFICATE_TABLE_SIZE is zero
00247 * or if the CBKE library is not present, this function returns false
00248 * and the EUI64 for the return value is not set.
00249 *
00250 * @param certEui64 The location of the return value for the EUI64.
00251 * @return true if the stack token certificate is set and the EUI64
00252 * return value is valid. false otherwise.
00253 */
00254 bool emberGetStackCertificateEui64(EmberEUI64
    certEui64);
00255
00256 /* @brief This function verifies the ECDSA signature of the
00257 * calculated digest and the associated received signature, using
00258 * the signerCertificate passed in. It is expected that the application
00259 * obtains the signerCertificate and performs the message digest calculation
00260 * on its own.
00261 */
00262 EmberStatus emberDsaVerify(EmberMessageDigest
    * digest,
00263
00264     EmberCertificateData*
    signerCertificate,

```

```

00261                                     EmberSignatureData* receivedSig);
00262
00263 /* @brief This callback is executed by the stack when the DSA verification
00264  * has completed and has a result. If the result is EMBER_SUCCESS, the
00265  * signature is valid. If the result is EMBER_SIGNATURE_VERIFY_FAILURE
00266  * then the signature is invalid. If the result is anything else then the
00267  * signature verify operation failed and the validity is unknown.
00268  */
00269 void emberDsaVerifyHandler(EmberStatus status);
00270
00271 // Start of 283k1 Curve Definitions
00272
00273 EmberStatus emberGetCertificate283k1(
00274     EmberCertificate283k1Data* result);
00275
00276 EmberStatus emberGenerateCbkeKeys283k1(
00277     void);
00278
00279 EmberStatus emberCalculateSmacs283k1(bool
00280     amInitiator,
00281     EmberCertificate283k1Data
00282     * partnerCert,
00283     EmberPublicKey283k1Data
00284     * partnerEphemeralPublicKey);
00285
00286 EmberStatus emberClearTemporaryDataMaybeStoreLinkKey283k1(
00287     bool storeLinkKey);
00288
00289 /* @brief This function is an application callback that must be defined
00290  * when using the CBKE 283k1 Library. It is called when the ephemeral key
00291  * generation operation is complete. The newly generated public key is passed
00292  * back to the application to be sent to the CBKE partner over-the-air.
00293  * Internally
00294  * the stack saves the public and private key pair until it the function
00295  * ::emberClearTemporaryDataMaybeStoreLinkKey283k1() is called by the
00296  * application.
00297  *
00298  * @param status This is the ::EmberStatus value indicating the success or
00299  * failure of the operation.
00300  * @param ephemeralPublicKey A pointer to an ::EmberPublicKeyData structure
00301  * containing the newly generated public key.
00302  */
00303 void emberGenerateCbkeKeysHandler283k1(
00304     EmberStatus status,
00305     EmberPublicKey283k1Data
00306     * ephemeralPublicKey);
00307
00308 /* @brief This function is an application callback that must be defined
00309  * when using CBKE. It is called when the shared secret generation is
00310  * complete and the link key and SMACs have been derived. The link key is
00311  * stored in a temporary location until the application decides to
00312  * store or discard the key by calling
00313  * ::emberClearTemporaryDataMaybeStoreLinkKey283k1().
00314  *
00315  * @param status This is the ::EmberStatus value indicating the success or
00316  * failure of the operation.
00317  * @param initiatorSmac This is a pointer to the ::EmberSmacData structure
00318  * to the initiator's version of the SMAC.
00319  * @param responderSmac This is a pointer to the ::EmberSmacData structure
00320  * to the responder's version of the SMAC.
00321  */
00322 void emberCalculateSmacsHandler283k1(EmberStatus
00323     status,
00324     EmberSmacData* initiatorSmac,
00325     EmberSmacData* responderSmac)
00326 ;
00327
00328 /* This function allows a one-time write of the MFG token if it has not
00329  * already been set. It does NOT utilize the ::EMBER_CERTIFICATE_TABLE_SIZE
00330  * so that should remain set at 0. Attempts to write the certificate that
00331  * has already been written will return a result of
00332  * ::EMBER_ERR_FLASH_WRITE_INHIBITED. If the EUI64 in the certificate is
00333  * the same as the current EUI of the device then this function may be called
00334  * while the stack is up. If the EUI in the certificate is different than
00335  * the current value, this function may only be called when the network is in
00336  * a state of ::EMBER_NO_NETWORK. Attempts to do otherwise will result in a
00337  * return value of ::EMBER_INVALID_CALL. If the EUI in the certificate is
00338  * different than the current value this function will also write the

```

```

00391 * Custom EUI64 MFG token. If that token has already been written the
00392 * operation
00393 * will fail and return a result of ::EMBER_BAD_ARGUMENT.
00394 * If all the above criteria is met the token will be written and
00395 * ::EMBER_SUCCESS will be returned.
00396 *
00397 * @note The device will immediately and persistently <b>change its EUI64
00398 * to match the value in the certificate</b>.
00399 *
00400 * @param caPublic A pointer to the 283k1 CA public key data that will be
00401 * stored
00402 * in stack tokens.
00403 * @param myCert A pointer to the 283k1 certificate data that will be stored in
00404 * stack tokens.
00405 * @param mykey A pointer to the 283k1 private key data that will be stored in
00406 * stack tokens.
00407 * @return The ::EmberStatus value indicating success or failure of the
00408 * operation.
00409 */
00410 EmberStatus emberSetPreinstalledCbkeData283k1(
00411     EmberPublicKey283k1Data* caPublic,
00412     EmberCertificate283k1Data
00413     * myCert,
00414     EmberPrivateKey283k1Data
00415     * myKey);
00416
00417 /* @brief This function verifies the ECDSA signature of the
00418 * calculated digest and the associated received signature, using
00419 * the signerCertificate passed in. It is expected that the application
00420 * obtains the signerCertificate and performs the message digest calculation
00421 * on its own.
00422 */
00423 EmberStatus emberDsaVerify283k1(
00424     EmberMessageDigest* digest,
00425     EmberCertificate283k1Data
00426     * signerCertificate,
00427     EmberSignature283k1Data*
00428     receivedSig);
00429
00430
00431

```

8.23 command-interpreter2.h File Reference

Data Structures

- struct [EmberCommandEntry](#)
Command entry for a command table.

Macros

- #define [MAX_TOKEN_COUNT](#)
- #define [emberCommandEntryAction](#)(name, action, argumentTypes, description)
- #define [emberCommandEntryActionWithDetails](#)(name, action, argumentTypes, description, argument-DescriptionArray)
- #define [emberCommandEntrySubMenu](#)(name, subMenu, description)
- #define [emberCommandEntryTerminator](#)()
- #define [EMBER_COMMAND_INTERPRETER_CONFIGURATION_ECHO](#)
- #define [emberProcessCommandInput](#)(port)
- #define [emberCommandInterpreterEchoOn](#)()
- #define [emberCommandInterpreterEchoOff](#)()
- #define [emberCommandInterpreterIsEchoOn](#)()

Typedefs

- typedef void(* [CommandAction](#))(void)

Enumerations

- enum [EmberCommandStatus](#) {
[EMBER_CMD_SUCCESS](#), [EMBER_CMD_ERR_PORT_PROBLEM](#), [EMBER_CMD_ERR_NO_SUCH_COMMAND](#), [EMBER_CMD_ERR_WRONG_NUMBER_OF_ARGUMENTS](#),
[EMBER_CMD_ERR_ARGUMENT_OUT_OF_RANGE](#), [EMBER_CMD_ERR_ARGUMENT_SYNTAX_ERROR](#), [EMBER_CMD_ERR_STRING_TOO_LONG](#), [EMBER_CMD_ERR_INVALID_ARGUMENT_TYPE](#) }

Functions

- void [emberCommandReaderSetDefaultBase](#) (uint8_t base)
- void [emberCommandActionHandler](#) (const [CommandAction](#) action)
- void [emberCommandErrorHandler](#) ([EmberCommandStatus](#) status)
- void [emberPrintCommandUsage](#) ([EmberCommandEntry](#) *entry)
- void [emberPrintCommandUsageNotes](#) (void)
- void [emberPrintCommandTable](#) (void)
- void [emberCommandClearBuffer](#) (void)
- void [emberCommandReaderInit](#) (void)
- bool [emberProcessCommandString](#) (uint8_t *input, uint8_t sizeOrPort)

Variables

- [EmberCommandEntry](#) * [emberCurrentCommand](#)
- [EmberCommandEntry](#) [emberCommandTable](#) []
- uint8_t [emberCommandInterpreter2Configuration](#)

Command Table Settings

- #define [EMBER_MAX_COMMAND_ARGUMENTS](#)
- #define [EMBER_COMMAND_BUFFER_LENGTH](#)
- #define [EMBER_COMMAND_INTEPRETER_HAS_DESCRIPTION_FIELD](#)

Functions to Retrieve Arguments

Use the following functions in your functions that process commands to retrieve arguments from the command interpreter. These functions pull out unsigned integers, signed integers, and strings, and hex strings. Index 0 is the first command argument.

- #define [emberCopyKeyArgument](#)(index, keyDataPointer)
- #define [emberCopyEui64Argument](#)(index, eui64)
- #define [emberGetEui64Argument](#)(index, eui64)
- uint8_t [emberCommandArgumentCount](#) (void)
- uint32_t [emberUnsignedCommandArgument](#) (uint8_t argNum)

- `int32_t emberSignedCommandArgument` (`uint8_t argNum`)
- `bool emberStringToHostOrderIpv4Address` (`const uint8_t *string`, `uint32_t *hostOrderIpv4Address`)
- `bool emberStringArgumentToHostOrderIpv4Address` (`uint8_t argNum`, `uint32_t *hostOrderIpv4Address`)
- `uint8_t * emberStringCommandArgument` (`int8_t argNum`, `uint8_t *length`)
- `const char * emberCommandName` (`void`)
- `uint8_t emberCopyStringArgument` (`int8_t argNum`, `uint8_t *destination`, `uint8_t maxLength`, `bool leftPad`)
- `uint8_t emberCopyBigEndianEui64Argument` (`int8_t index`, `EmberEUI64 destination`)

8.23.1 Detailed Description

Processes commands coming from the serial port. See [Command Interpreter 2](#) for documentation.

Definition in file [command-interpreter2.h](#).

8.24 command-interpreter2.h

```

00001
00010 #ifndef __COMMAND_INTERPRETER2_H__
00011 #define __COMMAND_INTERPRETER2_H__
00012
00100 #ifndef EMBER_MAX_COMMAND_ARGUMENTS
00101
00104 #define EMBER_MAX_COMMAND_ARGUMENTS 16
00105 #endif
00106
00107 #ifndef EMBER_COMMAND_BUFFER_LENGTH
00108 #define EMBER_COMMAND_BUFFER_LENGTH 100
00109 #endif
00110
00115 #if defined(DOXYGEN_SHOULD_SKIP_THIS)
00116 #define EMBER_COMMAND_INTEPRETER_HAS_DESCRIPTION_FIELD
00117 #endif
00118
00122 // The (+ 1) takes into account the leading command.
00123 #define MAX_TOKEN_COUNT (EMBER_MAX_COMMAND_ARGUMENTS + 1)
00124
00125 typedef void (*CommandAction) (void);
00126
00127 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00128
00130 typedef struct {
00131 #else
00132 typedef PGM struct {
00133 #endif
00134
00137     PGM_P name;
00143     CommandAction action;
00172     PGM_P argumentTypes;
00176 #if defined(EMBER_COMMAND_INTEPRETER_HAS_DESCRIPTION_FIELD)
00177     PGM_P description;
00178
00181     PGM_P const * argumentDescriptions;
00182 #endif
00183 } EmberCommandEntry;
00184
00185
00186 #if defined(EMBER_COMMAND_INTEPRETER_HAS_DESCRIPTION_FIELD)
00187 /* @brief Macro to define a CLI action */
00188 #define emberCommandEntryAction(name, action, argumentTypes, description) \
00189     { (name), (action), (argumentTypes), (description), NULL }
00190
00191 #define emberCommandEntryActionWithDetails(name, \

```

```

00192                                     action,      \
00193                                     argumentTypes, \
00194                                     description,    \
00195                                     argumentDescriptionArray) \
00196     { (name), (action), (argumentTypes), (description),
      (argumentDescriptionArray) }

00197
00198 /* @brief Macro to define a CLI sub-menu (nested command) */
00199 #define emberCommandEntrySubMenu(name, subMenu, description) \
00200     { (name), NULL, (PGM_P)(subMenu), (description), NULL }
00201
00202 /* @brief Macro to define a command entry array terminator.*/
00203 #define emberCommandEntryTerminator() \
00204     { NULL, NULL, NULL, NULL, NULL }
00205
00206 #else // Don't include description data in struct
00207
00208 /* @brief Macro to define a CLI action */
00209 #define emberCommandEntryAction(name, action, argumentTypes, description) \
00210     { (name), (action), (argumentTypes) }
00211
00212 #define emberCommandEntryActionWithDetails(name, \
00213     action, \
00214     argumentTypes, \
00215     description, \
00216     argumentDescriptionArray) \
00217     { (name), (action), (argumentTypes) }
00218
00219 /* @brief Macro to define a CLI sub-menu (nested command) */
00220 #define emberCommandEntrySubMenu(name, subMenu, description) \
00221     { (name), NULL, (PGM_P)(subMenu) }
00222
00223 /* @brief Macro to define a command entry array terminator.*/
00224 #define emberCommandEntryTerminator() \
00225     { NULL, NULL, NULL }
00226
00227 #endif
00228
00235 extern EmberCommandEntry *emberCurrentCommand
00236 ;
00237 extern EmberCommandEntry emberCommandTable[];
00238
00242 extern uint8_t emberCommandInterpreter2Configuration
00243 ;
00244 #define EMBER_COMMAND_INTERPRETER_CONFIGURATION_ECHO (0x01)
00245
00246 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00247
00252 enum EmberCommandStatus
00253 #else
00254 typedef uint8_t EmberCommandStatus;
00255 enum
00256 #endif
00257 {
00258     EMBER_CMD_SUCCESS,
00259     EMBER_CMD_ERR_PORT_PROBLEM,
00260     EMBER_CMD_ERR_NO_SUCH_COMMAND,
00261     EMBER_CMD_ERR_WRONG_NUMBER_OF_ARGUMENTS
00262 ,
00263     EMBER_CMD_ERR_ARGUMENT_OUT_OF_RANGE,
00264     EMBER_CMD_ERR_ARGUMENT_SYNTAX_ERROR,
00265     EMBER_CMD_ERR_STRING_TOO_LONG,
00266     EMBER_CMD_ERR_INVALID_ARGUMENT_TYPE
00267 };
00277 uint8_t emberCommandArgumentCount(void);
00278
00280 uint32_t emberUnsignedCommandArgument(uint8_t
      argNum);
00281
00283 int32_t emberSignedCommandArgument(uint8_t argNum);
00284
00288 bool emberStringToHostOrderIpv4Address(const
      uint8_t* string, uint32_t* hostOrderIpv4Address);
00289
00293 bool emberStringArgumentToHostOrderIpv4Address

```

```

        (uint8_t argNum, uint32_t* hostOrderIpv4Address);
00294
00295
00306 uint8_t *emberStringCommandArgument(int8_t argNum,
uint8_t *length);
00307
00308 const char *emberCommandName(void);
00309
00322 uint8_t emberCopyStringArgument(int8_t argNum,
00323                                uint8_t *destination,
00324                                uint8_t maxLength,
00325                                bool leftPad);
00326
00330 #define emberCopyKeyArgument(index, keyDataPointer) \
00331     (emberCopyStringArgument((index), \
00332                             emberKeyContents((keyDataPointer)), \
00333                             EMBER_ENCRYPTION_KEY_SIZE, \
00334                             true))
00335
00337 #define emberCopyEui64Argument(index, eui64) \
00338     (emberCopyStringArgument((index), (eui64), EUI64_SIZE, true))
00339 #define emberGetEui64Argument(index, eui64) \
00340     (emberCopyStringArgument((index), (eui64), EUI64_SIZE, true))
00341
00346 uint8_t emberCopyBigEndianEui64Argument(int8_t
index, EmberEUI64 destination);
00347
00351 void emberCommandReaderSetDefaultBase(uint8_t
base);
00352
00357 void emberCommandActionHandler(const CommandAction
action);
00364 void emberCommandErrorHandler(EmberCommandStatus status
);
00365 void emberPrintCommandUsage(EmberCommandEntry
*entry);
00366 void emberPrintCommandUsageNotes(void);
00367 void emberPrintCommandTable(void);
00368 void emberCommandClearBuffer(void);
00369
00372 void emberCommandReaderInit(void);
00373
00376 bool emberProcessCommandString(uint8_t *input, uint8_t
sizeOrPort);
00377
00386 #define emberProcessCommandInput(port) \
00387     emberProcessCommandString(NULL, port)
00388
00391 #define emberCommandInterpreterEchoOn() \
00392     (emberCommandInterpreter2Configuration \
00393      |= EMBER_COMMAND_INTERPRETER_CONFIGURATION_ECHO)
00394
00397 #define emberCommandInterpreterEchoOff() \
00398     (emberCommandInterpreter2Configuration \
00399      &= (~EMBER_COMMAND_INTERPRETER_CONFIGURATION_ECHO))
00400
00403 #define emberCommandInterpreterIsEchoOn() \
00404     (emberCommandInterpreter2Configuration \
00405      & EMBER_COMMAND_INTERPRETER_CONFIGURATION_ECHO)
00406
00409 #endif // __COMMAND_INTERPRETER2_H__

```

8.25 crc.h File Reference

Macros

- #define [INITIAL_CRC](#)
- #define [CRC32_START](#)
- #define [CRC32_END](#)

Functions

- uint16_t [halCommonCrc16](#) (uint8_t newByte, uint16_t prevResult)
- uint32_t [halCommonCrc32](#) (uint8_t newByte, uint32_t prevResult)

8.25.1 Detailed Description

See [Cyclic Redundancy Code \(CRC\)](#) for detailed documentation.

Definition in file [crc.h](#).

8.26 [crc.h](#)

```

00001
00007 #ifndef __CRC_H__
00008 #define __CRC_H__
00009
00028 uint16_t halCommonCrc16(uint8_t newByte, uint16_t prevResult);
00029
00030
00046 uint32_t halCommonCrc32(uint8_t newByte, uint32_t prevResult);
00047
00048
00051 #define INITIAL_CRC          0xFFFFFFFFL
00052
00053
00056 #define CRC32_START          INITIAL_CRC
00057
00058
00061 #define CRC32_END            0xDEBB20E3L
00062
00063
00067 #endif //__CRC_H__
00068

```

8.27 [ember-configuration-defaults.h](#) File Reference

Macros

- #define [EMBER_API_MAJOR_VERSION](#)
- #define [EMBER_API_MINOR_VERSION](#)
- #define [EMBER_STACK_PROFILE](#)
- #define [EMBER_MAX_END_DEVICE_CHILDREN](#)
- #define [EMBER_SECURITY_LEVEL](#)
- #define [EMBER_CHILD_TABLE_SIZE](#)
- #define [EMBER_KEY_TABLE_SIZE](#)
- #define [EMBER_CERTIFICATE_TABLE_SIZE](#)
- #define [EMBER_MAX_DEPTH](#)
- #define [EMBER_MAX_HOPS](#)
- #define [EMBER_PACKET_BUFFER_COUNT](#)
- #define [EMBER_MAX_NEIGHBOR_TABLE_SIZE](#)
- #define [EMBER_NEIGHBOR_TABLE_SIZE](#)
- #define [EMBER_INDIRECT_TRANSMISSION_TIMEOUT](#)
- #define [EMBER_MAX_INDIRECT_TRANSMISSION_TIMEOUT](#)
- #define [EMBER_SEND_MULTICASTS_TO_SLEEPY_ADDRESS](#)
- #define [EMBER_END_DEVICE_POLL_TIMEOUT](#)

- #define [EMBER_END_DEVICE_POLL_TIMEOUT_SHIFT](#)
- #define [EMBER_MOBILE_NODE_POLL_TIMEOUT](#)
- #define [EMBER_APS_UNICAST_MESSAGE_COUNT](#)
- #define [EMBER_BINDING_TABLE_SIZE](#)
- #define [EMBER_ADDRESS_TABLE_SIZE](#)
- #define [EMBER_RESERVED_MOBILE_CHILD_ENTRIES](#)
- #define [EMBER_ROUTE_TABLE_SIZE](#)
- #define [EMBER_DISCOVERY_TABLE_SIZE](#)
- #define [EMBER_MULTICAST_TABLE_SIZE](#)
- #define [EMBER_SOURCE_ROUTE_TABLE_SIZE](#)
- #define [EMBER_DEFAULT_BROADCAST_TABLE_SIZE](#)
- #define [EMBER_BROADCAST_TABLE_SIZE](#)
- #define [EMBER_RETRY_QUEUE_SIZE](#)
- #define [EMBER_ASSERT_SERIAL_PORT](#)
- #define [EMBER_MAXIMUM_ALARM_DATA_SIZE](#)
- #define [EMBER_BROADCAST_ALARM_DATA_SIZE](#)
- #define [EMBER_UNICAST_ALARM_DATA_SIZE](#)
- #define [EMBER_FRAGMENT_DELAY_MS](#)
- #define [EMBER_FRAGMENT_MAX_WINDOW_SIZE](#)
- #define [EMBER_FRAGMENT_WINDOW_SIZE](#)
- #define [EMBER_BINDING_TABLE_TOKEN_SIZE](#)
- #define [EMBER_CHILD_TABLE_TOKEN_SIZE](#)
- #define [EMBER_KEY_TABLE_TOKEN_SIZE](#)
- #define [EMBER_REQUEST_KEY_TIMEOUT](#)
- #define [EMBER_TRANSIENT_KEY_TIMEOUT_S](#)
- #define [EMBER_END_DEVICE_BIND_TIMEOUT](#)
- #define [EMBER_PAN_ID_CONFLICT_REPORT_THRESHOLD](#)
- #define [EMBER_TASK_COUNT](#)
- #define [EMBER_MAX_SUPPORTED_NETWORKS](#)
- #define [EMBER_SUPPORTED_NETWORKS](#)
- #define [EMBER_ZLL_GROUP_ADDRESSES](#)
- #define [EMBER_ZLL_RSSI_THRESHOLD](#)
- #define [EMBER_RF4CE_PAIRING_TABLE_SIZE](#)
- #define [EMBER_RF4CE_PAIRING_TABLE_TOKEN_SIZE](#)
- #define [EMBER_RF4CE_PENDING_OUTGOING_PACKET_TABLE_SIZE](#)
- #define [EMBER_GP_PROXY_TABLE_SIZE](#)
- #define [EMBER_GP_PROXY_TABLE_TOKEN_SIZE](#)
- #define [EMBER_GP_SINK_TABLE_SIZE](#)
- #define [EMBER_GP_SINK_TABLE_TOKEN_SIZE](#)

8.27.1 Detailed Description

User-configurable stack memory allocation defaults.

Note

Application developers should **not** modify any portion of this file. Doing so may cause mysterious bugs. Allocations should be adjusted only by defining the appropriate macros in the application's `CONFIGURATION_HEADER`.

See [Configuration](#) for documentation.

Definition in file [ember-configuration-defaults.h](#).

8.28 ember-configuration-defaults.h

```

00001
00014 //  Todo:
00015 //  -  explain how to use a configuration header
00016 //  -  the documentation of the custom handlers should
00017 //      go in hal/ember-configuration.c, not here
00018 //  -  the stack profile documentation is out of date
00019
00047 #ifndef __EMBER_CONFIGURATION_DEFAULTS_H__
00048 #define __EMBER_CONFIGURATION_DEFAULTS_H__
00049
00050 #ifdef CONFIGURATION_HEADER
00051 #include CONFIGURATION_HEADER
00052 #endif
00053
00054 #ifndef EMBER_API_MAJOR_VERSION
00055
00058     #define EMBER_API_MAJOR_VERSION 2
00059 #endif
00060
00061 #ifndef EMBER_API_MINOR_VERSION
00062
00065     #define EMBER_API_MINOR_VERSION 0
00066 #endif
00067
00080 #ifndef EMBER_STACK_PROFILE
00081     #define EMBER_STACK_PROFILE 0
00082 #endif
00083
00084 #if (EMBER_STACK_PROFILE == 2)
00085 #define EMBER_MAX_DEPTH          15
00086 #define EMBER_SECURITY_LEVEL     5
00087 #define EMBER_MIN_ROUTE_TABLE_SIZE 10
00088 #define EMBER_MIN_DISCOVERY_TABLE_SIZE 4
00089 #define EMBER_INDIRECT_TRANSMISSION_TIMEOUT 7680
00090 #define EMBER_SEND_MULTICASTS_TO_SLEEPY_ADDRESS false
00091 #endif
00092
00093 #ifndef EMBER_MAX_END_DEVICE_CHILDREN
00094
00098     #define EMBER_MAX_END_DEVICE_CHILDREN 6
00099 #endif
00100
00101 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00102 /* Need to put in a compile time check to make sure that we aren't specifying
00103  * too many child devices. The NCP may or may not support 64 end devices. But
00104  * the host code doesn't matter.
00105  */
00106 #if defined(HAL_HAS_INT64) || defined(EZSP_HOST)
00107     #if EMBER_MAX_END_DEVICE_CHILDREN > 64
00108         #error "EMBER_MAX_END_DEVICE_CHILDREN can not exceed 64."
00109     #endif
00110 #else
00111     #if EMBER_MAX_END_DEVICE_CHILDREN > 32
00112         #error "EMBER_MAX_END_DEVICE_CHILDREN can not exceed 32."
00113     #endif
00114 #endif
00115
00116 #endif // DOXYGEN_SHOULD_SKIP_THIS
00117
00118 #ifndef EMBER_SECURITY_LEVEL
00119
00123     #define EMBER_SECURITY_LEVEL 5
00124 #endif
00125
00126 #if ! (EMBER_SECURITY_LEVEL == 0          \
00127        || EMBER_SECURITY_LEVEL == 5)
00128     #error "Unsupported security level"
00129 #endif
00130
00131 #ifdef EMBER_CHILD_TABLE_SIZE
00132     #if (EMBER_MAX_END_DEVICE_CHILDREN < EMBER_CHILD_TABLE_SIZE)
00133         #undef EMBER_CHILD_TABLE_SIZE
00134     #endif
00135 #endif
00136
00137 #ifndef EMBER_CHILD_TABLE_SIZE
00138

```

```

00152 #define EMBER_CHILD_TABLE_SIZE EMBER_MAX_END_DEVICE_CHILDREN
00153 #endif
00154
00168 #ifndef EMBER_KEY_TABLE_SIZE
00169 #define EMBER_KEY_TABLE_SIZE 0
00170 #endif
00171
00181 #ifndef EMBER_CERTIFICATE_TABLE_SIZE
00182 #define EMBER_CERTIFICATE_TABLE_SIZE 0
00183 #else
00184 #if EMBER_CERTIFICATE_TABLE_SIZE > 1
00185 #error "EMBER_CERTIFICATE_TABLE_SIZE > 1 is not supported!"
00186 #endif
00187 #endif
00188
00194 #ifndef EMBER_MAX_DEPTH
00195 #define EMBER_MAX_DEPTH 15
00196 #elif (EMBER_MAX_DEPTH > 15)
00197 // Depth is a 4-bit field
00198 #error "EMBER_MAX_DEPTH cannot be greater than 15"
00199 #endif
00200
00207 #ifndef EMBER_MAX_HOPS
00208 #define EMBER_MAX_HOPS (2 * EMBER_MAX_DEPTH)
00209 #endif
00210
00217 #ifndef EMBER_PACKET_BUFFER_COUNT
00218 #define EMBER_PACKET_BUFFER_COUNT 75
00219 #endif
00220
00232 #define EMBER_MAX_NEIGHBOR_TABLE_SIZE 16
00233 #ifndef EMBER_NEIGHBOR_TABLE_SIZE
00234 #define EMBER_NEIGHBOR_TABLE_SIZE 16
00235 #endif
00236
00243 #ifndef EMBER_INDIRECT_TRANSMISSION_TIMEOUT
00244 #define EMBER_INDIRECT_TRANSMISSION_TIMEOUT 3000
00245 #endif
00246 #define EMBER_MAX_INDIRECT_TRANSMISSION_TIMEOUT 30000
00247 #if (EMBER_INDIRECT_TRANSMISSION_TIMEOUT
\
00248 > EMBER_MAX_INDIRECT_TRANSMISSION_TIMEOUT)
00249 #error "Indirect transmission timeout too large."
00250 #endif
00251
00258 #ifndef EMBER_SEND_MULTICASTS_TO_SLEEPY_ADDRESS
00259 #define EMBER_SEND_MULTICASTS_TO_SLEEPY_ADDRESS false
00260 #endif
00261
00262
00277 #ifndef EMBER_END_DEVICE_POLL_TIMEOUT
00278 #define EMBER_END_DEVICE_POLL_TIMEOUT 5
00279 #endif
00280
00288 #ifndef EMBER_END_DEVICE_POLL_TIMEOUT_SHIFT
00289 #define EMBER_END_DEVICE_POLL_TIMEOUT_SHIFT 6
00290 #endif
00291
00298 #ifndef EMBER_MOBILE_NODE_POLL_TIMEOUT
00299 #define EMBER_MOBILE_NODE_POLL_TIMEOUT 20
00300 #endif
00301
00314 #ifndef EMBER_APS_UNICAST_MESSAGE_COUNT
00315 #define EMBER_APS_UNICAST_MESSAGE_COUNT 10
00316 #endif
00317
00320 #ifndef EMBER_BINDING_TABLE_SIZE
00321 #define EMBER_BINDING_TABLE_SIZE 0
00322 #endif
00323
00328 #ifndef EMBER_ADDRESS_TABLE_SIZE
00329 #define EMBER_ADDRESS_TABLE_SIZE 8
00330 #endif
00331
00338 #ifndef EMBER_RESERVED_MOBILE_CHILD_ENTRIES
00339 #define EMBER_RESERVED_MOBILE_CHILD_ENTRIES 0
00340 #endif
00341
00348 #ifndef EMBER_ROUTE_TABLE_SIZE
00349 #define EMBER_MIN_ROUTE_TABLE_SIZE

```

```

00350     #define EMBER_ROUTE_TABLE_SIZE EMBER_MIN_ROUTE_TABLE_SIZE
00351     #else
00352     #define EMBER_ROUTE_TABLE_SIZE 16
00353     #endif
00354 #elif defined(EMBER_MIN_ROUTE_TABLE_SIZE) \
00355     && EMBER_ROUTE_TABLE_SIZE < EMBER_MIN_ROUTE_TABLE_SIZE
00356     #error "EMBER_ROUTE_TABLE_SIZE is less than required by stack profile."
00357 #endif
00358
00364 #ifndef EMBER_DISCOVERY_TABLE_SIZE
00365     #ifdef EMBER_MIN_DISCOVERY_TABLE_SIZE
00366         #define EMBER_DISCOVERY_TABLE_SIZE EMBER_MIN_DISCOVERY_TABLE_SIZE
00367     #else
00368     #define EMBER_DISCOVERY_TABLE_SIZE 8
00369     #endif
00370 #elif defined(EMBER_MIN_DISCOVERY_TABLE_SIZE) \
00371     && EMBER_DISCOVERY_TABLE_SIZE < EMBER_MIN_DISCOVERY_TABLE_SIZE
00372     #error "EMBER_DISCOVERY_TABLE_SIZE is less than required by stack profile."
00373 #endif
00374
00380 #ifndef EMBER_MULTICAST_TABLE_SIZE
00381     #define EMBER_MULTICAST_TABLE_SIZE 8
00382 #endif
00383
00390 #ifndef EMBER_SOURCE_ROUTE_TABLE_SIZE
00391     #define EMBER_SOURCE_ROUTE_TABLE_SIZE 32
00392 #endif
00393
00406 // TODO: we don't have one stack profile anymore, we might have two networks
00407 // the first one is Zigbee Pro, the second one is RF4CE.
00408 #if (EMBER_STACK_PROFILE == 2) && !defined(EMBER_TEST)
00409     #if defined(EMBER_BROADCAST_TABLE_SIZE)
00410         #error "Cannot override broadcast table size unless (EMBER_STACK_PROFILE !=
00411             2)"
00412     #endif
00413 #endif
00414 #define EMBER_DEFAULT_BROADCAST_TABLE_SIZE 15
00415
00416 #ifndef EMBER_BROADCAST_TABLE_SIZE
00417     #define EMBER_BROADCAST_TABLE_SIZE EMBER_DEFAULT_BROADCAST_TABLE_SIZE
00418 #elif EMBER_BROADCAST_TABLE_SIZE < EMBER_DEFAULT_BROADCAST_TABLE_SIZE
00419     #error "EMBER_BROADCAST_TABLE_SIZE is less than the minimum value of 15."
00420 #elif 254 < EMBER_BROADCAST_TABLE_SIZE
00421     #error "EMBER_BROADCAST_TABLE_SIZE is larger than the maximum value of 254."
00422 #endif
00423
00424
00425 #ifndef EMBER_RETRY_QUEUE_SIZE
00426     #define EMBER_RETRY_QUEUE_SIZE 8
00427 #endif
00428
00429
00430
00440 #if !defined(EMBER_ASSERT_OUTPUT_DISABLED) \
00441     && !defined(EMBER_ASSERT_SERIAL_PORT)
00442     #define EMBER_ASSERT_SERIAL_PORT 1
00443 #endif
00444
00458 #define EMBER_MAXIMUM_ALARM_DATA_SIZE 16
00459
00477 #ifndef EMBER_BROADCAST_ALARM_DATA_SIZE
00478     #define EMBER_BROADCAST_ALARM_DATA_SIZE 0
00479 #elif EMBER_MAXIMUM_ALARM_DATA_SIZE < EMBER_BROADCAST_ALARM_DATA_SIZE
00480     #error "EMBER_BROADCAST_ALARM_DATA_SIZE is too large."
00481 #endif
00482
00491 #ifndef EMBER_UNICAST_ALARM_DATA_SIZE
00492     #define EMBER_UNICAST_ALARM_DATA_SIZE 0
00493 #elif EMBER_MAXIMUM_ALARM_DATA_SIZE < EMBER_UNICAST_ALARM_DATA_SIZE
00494     #error "EMBER_UNICAST_ALARM_DATA_SIZE is too large."
00495 #endif
00496
00500 #ifndef EMBER_FRAGMENT_DELAY_MS
00501     #define EMBER_FRAGMENT_DELAY_MS 0
00502 #endif
00503
00507 #define EMBER_FRAGMENT_MAX_WINDOW_SIZE 8
00508
00513 #ifndef EMBER_FRAGMENT_WINDOW_SIZE

```

```

00514 #define EMBER_FRAGMENT_WINDOW_SIZE 1
00515 #elif EMBER_FRAGMENT_MAX_WINDOW_SIZE < EMBER_FRAGMENT_WINDOW_SIZE
00516 #error "EMBER_FRAGMENT_WINDOW_SIZE is too large."
00517 #endif
00518
00519 #ifndef EMBER_BINDING_TABLE_TOKEN_SIZE
00520 #define EMBER_BINDING_TABLE_TOKEN_SIZE EMBER_BINDING_TABLE_SIZE
00521 #endif
00522 #ifndef EMBER_CHILD_TABLE_TOKEN_SIZE
00523 #define EMBER_CHILD_TABLE_TOKEN_SIZE EMBER_CHILD_TABLE_SIZE
00524 #endif
00525 #ifndef EMBER_KEY_TABLE_TOKEN_SIZE
00526 #define EMBER_KEY_TABLE_TOKEN_SIZE EMBER_KEY_TABLE_SIZE
00527 #endif
00528
00541 #ifndef EMBER_REQUEST_KEY_TIMEOUT
00542 #define EMBER_REQUEST_KEY_TIMEOUT 0
00543 #elif EMBER_REQUEST_KEY_TIMEOUT > 10
00544 #error "EMBER_REQUEST_KEY_TIMEOUT is too large."
00545 #endif
00546
00555 #ifndef EMBER_TRANSIENT_KEY_TIMEOUT_S
00556 #define EMBER_TRANSIENT_KEY_TIMEOUT_S (300)
00557 #endif
00558
00562 #ifndef EMBER_END_DEVICE_BIND_TIMEOUT
00563 #define EMBER_END_DEVICE_BIND_TIMEOUT 60
00564 #endif
00565
00574 #ifndef EMBER_PAN_ID_CONFLICT_REPORT_THRESHOLD
00575 #define EMBER_PAN_ID_CONFLICT_REPORT_THRESHOLD 1
00576 #endif
00577
00583 #ifndef EMBER_TASK_COUNT
00584 #define EMBER_TASK_COUNT (4)
00585 #endif
00586
00589 #define EMBER_MAX_SUPPORTED_NETWORKS 2
00590 #ifndef EMBER_SUPPORTED_NETWORKS
00591 #ifdef EMBER_TEST
00592 #define EMBER_SUPPORTED_NETWORKS 2
00593 #else
00594 #define EMBER_SUPPORTED_NETWORKS 1
00595 #endif
00596 #endif
00597
00598 #ifndef EMBER_ZLL_GROUP_ADDRESSES
00599
00601 #define EMBER_ZLL_GROUP_ADDRESSES 1
00602 #endif
00603
00604 #ifndef EMBER_ZLL_RSSI_THRESHOLD
00605
00607 #define EMBER_ZLL_RSSI_THRESHOLD -128
00608 #endif
00609
00610 #ifndef EMBER_RF4CE_PAIRING_TABLE_SIZE
00611
00613 #define EMBER_RF4CE_PAIRING_TABLE_SIZE 0
00614 #endif
00615
00616 #ifndef EMBER_RF4CE_PAIRING_TABLE_TOKEN_SIZE
00617
00619 #define EMBER_RF4CE_PAIRING_TABLE_TOKEN_SIZE EMBER_RF4CE_PAIRING_TABLE_SIZE
00620 #endif
00621
00622 #ifndef EMBER_RF4CE_PENDING_OUTGOING_PACKET_TABLE_SIZE
00623
00625 #define EMBER_RF4CE_PENDING_OUTGOING_PACKET_TABLE_SIZE 0
00626 #endif
00627
00628 #ifndef EMBER_GP_PROXY_TABLE_SIZE
00629
00631 #define EMBER_GP_PROXY_TABLE_SIZE 5
00632 //XXXEZSP
00633 #endif
00634
00635 #ifndef EMBER_GP_PROXY_TABLE_TOKEN_SIZE
00636
00638 #define EMBER_GP_PROXY_TABLE_TOKEN_SIZE EMBER_GP_PROXY_TABLE_SIZE

```

```

00639 #endif
00640
00641 #ifndef EMBER_GP_SINK_TABLE_SIZE
00642
00644     #define EMBER_GP_SINK_TABLE_SIZE 0
00645 #endif
00646
00647 #ifndef EMBER_GP_SINK_TABLE_TOKEN_SIZE
00648
00650     #define EMBER_GP_SINK_TABLE_TOKEN_SIZE EMBER_GP_SINK_TABLE_SIZE
00651 #endif
00652
00653
00656 #endif //__EMBER_CONFIGURATION_DEFAULTS_H__

```

8.29 ember-types.h File Reference

```

#include "stack/include/error.h"
#include "stack/include/zll-types.h"
#include "stack/include/rf4ce-types.h"
#include "stack/include/gp-types.h"

```

Data Structures

- struct [EmberReleaseTypeStruct](#)
A structure relating version types to human readable strings.
- struct [EmberVersion](#)
Version struct containing all version information.
- struct [EmberZigbeeNetwork](#)
Defines a ZigBee network and the associated parameters.
- struct [EmberNetworkInitStruct](#)
Defines the network initialization configuration that should be used when ::emberNetwork-InitExtended() is called by the application.
- struct [EmberNetworkParameters](#)
Holds network parameters.
- struct [EmberApsFrame](#)
An in-memory representation of a ZigBee APS frame of an incoming or outgoing message.
- struct [EmberBindingTableEntry](#)
Defines an entry in the binding table.
- struct [EmberNeighborTableEntry](#)
Defines an entry in the neighbor table.
- struct [EmberRouteTableEntry](#)
Defines an entry in the route table.
- struct [EmberMulticastTableEntry](#)
Defines an entry in the multicast table.
- struct [EmberEventControl](#)
Control structure for events.
- struct [EmberEventData_S](#)
Complete events with a control and a handler procedure.
- struct [EmberTaskControl](#)
Control structure for tasks.

- struct [EmberKeyData](#)
This data structure contains the key data that is passed into various other functions.
- struct [EmberCertificateData](#)
This data structure contains the certificate data that is used for Certificate Based Key Exchange (CBKE).
- struct [EmberPublicKeyData](#)
This data structure contains the public key data that is used for Certificate Based Key Exchange (CBKE).
- struct [EmberPrivateKeyData](#)
This data structure contains the private key data that is used for Certificate Based Key Exchange (CBKE).
- struct [EmberSmacData](#)
This data structure contains the Shared Message Authentication Code (SMAC) data that is used for Certificate Based Key Exchange (CBKE).
- struct [EmberSignatureData](#)
This data structure contains a DSA signature. It is the bit concatenation of the 'r' and 's' components of the signature.
- struct [EmberMessageDigest](#)
This data structure contains an AES-MMO Hash (the message digest).
- struct [EmberAesMmoHashContext](#)
This data structure contains the context data when calculating an AES MMO hash (message digest).
- struct [EmberCertificate283k1Data](#)
This data structure contains the certificate data that is used for Certificate Based Key Exchange (CBKE) in SECT283k1 Elliptical Cryptography.
- struct [EmberPublicKey283k1Data](#)
This data structure contains the public key data that is used for Certificate Based Key Exchange (CBKE) in SECT283k1 Elliptical Cryptography.
- struct [EmberPrivateKey283k1Data](#)
This data structure contains the private key data that is used for Certificate Based Key Exchange (CBKE) in SECT283k1 Elliptical Cryptography.
- struct [EmberSignature283k1Data](#)
This data structure contains a DSA signature used in SECT283k1 Elliptical Cryptography. It is the bit concatenation of the 'r' and 's' components of the signature.
- struct [EmberInitialSecurityState](#)
This describes the Initial Security features and requirements that will be used when forming or joining the network.
- struct [EmberCurrentSecurityState](#)
This describes the security features used by the stack for a joined device.
- struct [EmberKeyStruct](#)
This describes a one of several different types of keys and its associated data.
- struct [EmberMfgSecurityStruct](#)
This structure is used to get/set the security config that is stored in manufacturing tokens.
- struct [EmberMacFilterMatchStruct](#)
This structure indicates a matching raw MAC message has been received by the application configured MAC filters.

Macros

- #define [EMBER_MIN_BROADCAST_ADDRESS](#)
- #define [emberIsZigbeeBroadcastAddress\(address\)](#)
- #define [EMBER_JOIN_DECISION_STRINGS](#)
- #define [EMBER_DEVICE_UPDATE_STRINGS](#)
- #define [emberInitializeNetworkParameters\(parameters\)](#)
- #define [EMBER_COUNTER_STRINGS](#)
- #define [EMBER_STANDARD_SECURITY_MODE](#)
- #define [EMBER_TRUST_CENTER_NODE_ID](#)
- #define [EMBER_NO_TRUST_CENTER_MODE](#)
- #define [EMBER_GLOBAL_LINK_KEY](#)
- #define [EMBER_MFG_SECURITY_CONFIG_MAGIC_NUMBER](#)
- #define [EMBER_MAC_FILTER_MATCH_ENABLED_MASK](#)
- #define [EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_MASK](#)
- #define [EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_MASK](#)
- #define [EMBER_MAC_FILTER_MATCH_ON_DEST_MASK](#)
- #define [EMBER_MAC_FILTER_MATCH_ON_SOURCE_MASK](#)
- #define [EMBER_MAC_FILTER_MATCH_ENABLED](#)
- #define [EMBER_MAC_FILTER_MATCH_DISABLED](#)
- #define [EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_NONE](#)
- #define [EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_LOCAL](#)
- #define [EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_BROADCAST](#)
- #define [EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_NONE](#)
- #define [EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_NON_LOCAL](#)
- #define [EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_LOCAL](#)
- #define [EMBER_MAC_FILTER_MATCH_ON_DEST_BROADCAST_SHORT](#)
- #define [EMBER_MAC_FILTER_MATCH_ON_DEST_UNICAST_SHORT](#)
- #define [EMBER_MAC_FILTER_MATCH_ON_DEST_UNICAST_LONG](#)
- #define [EMBER_MAC_FILTER_MATCH_ON_SOURCE_LONG](#)
- #define [EMBER_MAC_FILTER_MATCH_ON_SOURCE_SHORT](#)
- #define [EMBER_MAC_FILTER_MATCH_ON_SOURCE_NONE](#)
- #define [EMBER_MAC_FILTER_MATCH_END](#)
- #define [WEAK_TEST](#)

Typedefs

- typedef uint8_t [EmberTaskId](#)
- typedef PGM struct [EmberEventData_S](#) [EmberEventData](#)
- typedef uint16_t [EmberMacFilterMatchData](#)
- typedef uint8_t [EmberLibraryStatus](#)

Enumerations

- enum [EmberNodeType](#) {
[EMBER_UNKNOWN_DEVICE](#), [EMBER_COORDINATOR](#), [EMBER_ROUTER](#),
[EMBER_END_DEVICE](#),
[EMBER_SLEEPY_END_DEVICE](#), [EMBER_MOBILE_END_DEVICE](#), [EMBER-
_RF4CE_TARGET](#), [EMBER_RF4CE_CONTROLLER](#) }

- enum EmberEndDeviceConfiguration { EMBER_END_DEVICE_CONFIG_NONE, EMBER_END_DEVICE_CONFIG_PERSIST_DATA_ON_PARENT }
- enum EmberNetworkInitBitmask { EMBER_NETWORK_INIT_NO_OPTIONS, EMBER_NETWORK_INIT_PARENT_INFO_IN_TOKEN }
- enum EmberApsOption { EMBER_APS_OPTION_NONE, EMBER_APS_OPTION_DSA_SIGN, EMBER_APS_OPTION_ENCRYPTION, EMBER_APS_OPTION_RETRY, EMBER_APS_OPTION_ENABLE_ROUTE_DISCOVERY, EMBER_APS_OPTION_FORCE_ROUTE_DISCOVERY, EMBER_APS_OPTION_SOURCE_EUI64, EMBER_APS_OPTION_DESTINATION_EUI64, EMBER_APS_OPTION_ENABLE_ADDRESS_DISCOVERY, EMBER_APS_OPTION_POLL_RESPONSE, EMBER_APS_OPTION_ZDO_RESPONSE_REQUIRED, EMBER_APS_OPTION_FRAGMENT }
- enum EmberIncomingMessageType { EMBER_INCOMING_UNICAST, EMBER_INCOMING_UNICAST_REPLY, EMBER_INCOMING_MULTICAST, EMBER_INCOMING_MULTICAST_LOOPBACK, EMBER_INCOMING_BROADCAST, EMBER_INCOMING_BROADCAST_LOOPBACK }
- enum EmberOutgoingMessageType { EMBER_OUTGOING_DIRECT, EMBER_OUTGOING_VIA_ADDRESS_TABLE, EMBER_OUTGOING_VIA_BINDING, EMBER_OUTGOING_MULTICAST, EMBER_OUTGOING_MULTICAST_WITH_ALIAS, EMBER_OUTGOING_BROADCAST_WITH_ALIAS, EMBER_OUTGOING_BROADCAST }
- enum EmberZigbeeCommandType { EMBER_ZIGBEE_COMMAND_TYPE_MAC, EMBER_ZIGBEE_COMMAND_TYPE_NWK, EMBER_ZIGBEE_COMMAND_TYPE_APS, EMBER_ZIGBEE_COMMAND_TYPE_ZDO, EMBER_ZIGBEE_COMMAND_TYPE_ZCL, EMBER_ZIGBEE_COMMAND_TYPE_BEACON }
- enum EmberNetworkStatus { EMBER_NO_NETWORK, EMBER_JOINING_NETWORK, EMBER_JOINED_NETWORK, EMBER_JOINED_NETWORK_NO_PARENT, EMBER_LEAVING_NETWORK }
- enum EmberNetworkScanType { EMBER_ENERGY_SCAN, EMBER_ACTIVE_SCAN }
- enum EmberBindingType { EMBER_UNUSED_BINDING, EMBER_UNICAST_BINDING, EMBER_MANY_TO_ONE_BINDING, EMBER_MULTICAST_BINDING }
- enum EmberJoinDecision { EMBER_USE_PRECONFIGURED_KEY, EMBER_SEND_KEY_IN_THE_CLEAR, EMBER_DENY_JOIN, EMBER_NO_ACTION }
- enum EmberDeviceUpdate { EMBER_STANDARD_SECURITY_SECURED_REJOIN, EMBER_STANDARD_SECURITY_UNSECURED_JOIN, EMBER_DEVICE_LEFT, EMBER_STANDARD_SECURITY_UNSECURED_REJOIN, EMBER_HIGH_SECURITY_SECURED_REJOIN, EMBER_HIGH_SECURITY_UNSECURED_JOIN, EMBER_HIGH_SECURITY_UNSECURED_REJOIN }
- enum EmberRejoinReason { EMBER_REJOIN_REASON_NONE, EMBER_REJOIN_DUE_TO_NWK_KEY_UPDATE, EMBER_REJOIN_DUE_TO_LEAVE_MESSAGE, EMBER_REJOIN_DUE_TO_NO_PARENT, EMBER_REJOIN_DUE_TO_ZLL_TOUCHLINK, EMBER_REJOIN_DUE_TO_APP_EVENT_5, EMBER_REJOIN_DUE_TO_APP_EVENT_4, EMBER_REJOIN_DUE_TO_APP_EVENT_3, EMBER_REJOIN_DUE_TO_APP_EVENT_2, EMBER_REJOIN_DUE_TO_APP_EVENT_1 }

- N_DUE_TO_APP_EVENT_3,
EMBER_REJOIN_DUE_TO_APP_EVENT_2, EMBER_REJOIN_DUE_TO_APP_EVENT_1 }
- enum EmberClusterListId { EMBER_INPUT_CLUSTER_LIST, EMBER_OUTPUT_CLUSTER_LIST }
- enum EmberEventUnits {
EMBER_EVENT_INACTIVE, EMBER_EVENT_MS_TIME, EMBER_EVENT_QS_TIME, EMBER_EVENT_MINUTE_TIME,
EMBER_EVENT_ZERO_DELAY }
- enum EmberJoinMethod { EMBER_USE_MAC_ASSOCIATION, EMBER_USE_NWK_REJOIN, EMBER_USE_NWK_REJOIN_HAVE_NWK_KEY, EMBER_USE_NWK_COMMISSIONING }
- enum EmberCounterType {
EMBER_COUNTER_MAC_RX_BROADCAST, EMBER_COUNTER_MAC_TX_BROADCAST, EMBER_COUNTER_MAC_RX_UNICAST, EMBER_COUNTER_MAC_TX_UNICAST_SUCCESS,
EMBER_COUNTER_MAC_TX_UNICAST_RETRY, EMBER_COUNTER_MAC_TX_UNICAST_FAILED, EMBER_COUNTER_APS_DATA_RX_BROADCAST, EMBER_COUNTER_APS_DATA_TX_BROADCAST,
EMBER_COUNTER_APS_DATA_RX_UNICAST, EMBER_COUNTER_APS_DATA_TX_UNICAST_SUCCESS, EMBER_COUNTER_APS_DATA_TX_UNICAST_RETRY, EMBER_COUNTER_APS_DATA_TX_UNICAST_FAILED,
EMBER_COUNTER_ROUTE_DISCOVERY_INITIATED, EMBER_COUNTER_NEIGHBOR_ADDED, EMBER_COUNTER_NEIGHBOR_REMOVED, EMBER_COUNTER_NEIGHBOR_STALE,
EMBER_COUNTER_JOIN_INDICATION, EMBER_COUNTER_CHILD_REMOVED, EMBER_COUNTER_ASH_OVERFLOW_ERROR, EMBER_COUNTER_ASH_FRAMING_ERROR,
EMBER_COUNTER_ASH_OVERRUN_ERROR, EMBER_COUNTER_NWK_FRAME_COUNTER_FAILURE, EMBER_COUNTER_APS_FRAME_COUNTER_FAILURE, EMBER_COUNTER_ASH_XOFF,
EMBER_COUNTER_APS_LINK_KEY_NOT_AUTHORIZED, EMBER_COUNTER_NWK_DECRYPTION_FAILURE, EMBER_COUNTER_APS_DECRYPTION_FAILURE, EMBER_COUNTER_ALLOCATE_PACKET_BUFFER_FAILURE,
EMBER_COUNTER_RELAYED_UNICAST, EMBER_COUNTER_PHY_TO_MAC_QUEUE_LIMIT_REACHED, EMBER_COUNTER_PACKET_VALIDATE_LIBRARY_DROPPED_COUNT, EMBER_COUNTER_TYPE_NWK_RETRY_OVERFLOW,
EMBER_COUNTER_PHY_CCA_FAIL_COUNT, EMBER_COUNTER_BROADCAST_TABLE_FULL, EMBER_COUNTER_TYPE_COUNT }
- enum EmberInitialSecurityBitmask {
EMBER_DISTRIBUTED_TRUST_CENTER_MODE, EMBER_TRUST_CENTER_GLOBAL_LINK_KEY, EMBER_PRECONFIGURED_NETWORK_KEY_MODE, EMBER_HAVE_TRUST_CENTER_EUI64,
EMBER_TRUST_CENTER_USES_HASHED_LINK_KEY, EMBER_HAVE_PRECONFIGURED_KEY, EMBER_HAVE_NETWORK_KEY, EMBER_GET_LINK_KEY_WHEN_JOINING,
EMBER_REQUIRE_ENCRYPTED_KEY, EMBER_NO_FRAME_COUNTER_RESET, EMBER_GET_PRECONFIGURED_KEY_FROM_INSTALL_CODE }
- enum EmberExtendedSecurityBitmask { EMBER_JOINER_GLOBAL_LINK_KEY, EMBER_EXT_NO_FRAME_COUNTER_RESET, EMBER_NWK_LEAVE_REQUEST_NOT_ALLOWED }

- enum EmberCurrentSecurityBitmask {
EMBER_STANDARD_SECURITY_MODE_, EMBER_DISTRIBUTED_TRUST_CENTER_MODE_, EMBER_TRUST_CENTER_GLOBAL_LINK_KEY_, EMBER_HAVE_TRUST_CENTER_LINK_KEY_,
EMBER_TRUST_CENTER_USES_HASHED_LINK_KEY_ }
- enum EmberKeyStructBitmask {
EMBER_KEY_HAS_SEQUENCE_NUMBER, EMBER_KEY_HAS_OUTGOING_FRAME_COUNTER, EMBER_KEY_HAS_INCOMING_FRAME_COUNTER, EMBER_KEY_HAS_PARTNER_EUI64,
EMBER_KEY_IS_AUTHORIZED, EMBER_KEY_PARTNER_IS_SLEEPY }
- enum EmberKeyType {
EMBER_TRUST_CENTER_LINK_KEY, EMBER_TRUST_CENTER_MASTER_KEY, EMBER_CURRENT_NETWORK_KEY, EMBER_NEXT_NETWORK_KEY,
EMBER_APPLICATION_LINK_KEY, EMBER_APPLICATION_MASTER_KEY }
- enum EmberKeyStatus {
EMBER_KEY_STATUS_NONE, EMBER_APP_LINK_KEY_ESTABLISHED, EMBER_APP_MASTER_KEY_ESTABLISHED, EMBER_TRUST_CENTER_LINK_KEY_ESTABLISHED,
EMBER_KEY_ESTABLISHMENT_TIMEOUT, EMBER_KEY_TABLE_FULL, EMBER_TC_RESPONDED_TO_KEY_REQUEST, EMBER_TC_APP_KEY_SENT_TO_REQUESTER,
EMBER_TC_RESPONSE_TO_KEY_REQUEST_FAILED, EMBER_TC_REQUEST_KEY_TYPE_NOT_SUPPORTED, EMBER_TC_NO_LINK_KEY_FOR_REQUESTER, EMBER_TC_REQUESTER_EUI64_UNKNOWN,
EMBER_TC_RECEIVED_FIRST_APP_KEY_REQUEST, EMBER_TC_TIMEOUT_WAITING_FOR_SECOND_APP_KEY_REQUEST, EMBER_TC_NON_MATCHING_APP_KEY_REQUEST_RECEIVED, EMBER_TC_FAILED_TO_SEND_APP_KEYS,
EMBER_TC_FAILED_TO_STORE_APP_KEY_REQUEST, EMBER_TC_REJECTED_APP_KEY_REQUEST, EMBER_TC_FAILED_TO_GENERATE_NEW_KEY, EMBER_TC_FAILED_TO_SEND_TC_KEY,
EMBER_TRUST_CENTER_IS_PRE_R21, EMBER_TC_REQUESTER_VERIFY_KEY_TIMEOUT, EMBER_TC_REQUESTER_VERIFY_KEY_FAILURE, EMBER_TC_REQUESTER_VERIFY_KEY_SUCCESS,
EMBER_VERIFY_LINK_KEY_FAILURE, EMBER_VERIFY_LINK_KEY_SUCCESS }
- enum EmberLinkKeyRequestPolicy { EMBER_DENY_KEY_REQUESTS, EMBER_ALLOW_KEY_REQUESTS, EMBER_GENERATE_NEW_TC_LINK_KEY }
- enum EmberKeySettings { EMBER_KEY_PERMISSIONS_NONE, EMBER_KEY_PERMISSIONS_READING_ALLOWED, EMBER_KEY_PERMISSIONS_HASHING_ALLOWED }
- enum EmberMacPassthroughType {
EMBER_MAC_PASSTHROUGH_NONE, EMBER_MAC_PASSTHROUGH_SE_INTERPAN, EMBER_MAC_PASSTHROUGH_EMBERNET, EMBER_MAC_PASSTHROUGH_EMBERNET_SOURCE,
EMBER_MAC_PASSTHROUGH_APPLICATION, EMBER_MAC_PASSTHROUGH_CUSTOM }

Functions

- uint8_t * emberKeyContents (EmberKeyData *key)

- uint8_t * emberCertificateContents (EmberCertificateData *cert)
- uint8_t * emberPublicKeyContents (EmberPublicKeyData *key)
- uint8_t * emberPrivateKeyContents (EmberPrivateKeyData *key)
- uint8_t * emberSmacContents (EmberSmacData *key)
- uint8_t * emberSignatureContents (EmberSignatureData *sig)
- uint8_t * emberCertificate283k1Contents (EmberCertificate283k1Data *cert)
- uint8_t * emberPublicKey283k1Contents (EmberPublicKey283k1Data *key)
- uint8_t * emberPrivateKey283k1Contents (EmberPrivateKey283k1Data *key)
- uint8_t * ember283k1SignatureContents (Ember283k1SignatureData *sig)

Miscellaneous Ember Types

- #define EMBER_RELEASE_TYPE_TO_STRING_STRUCT_DATA
- #define EUI64_SIZE
- #define EXTENDED_PAN_ID_SIZE
- #define EMBER_ENCRYPTION_KEY_SIZE
- #define EMBER_CERTIFICATE_SIZE
- #define EMBER_PUBLIC_KEY_SIZE
- #define EMBER_PRIVATE_KEY_SIZE
- #define EMBER_SMAC_SIZE
- #define EMBER_SIGNATURE_SIZE
- #define EMBER_AES_HASH_BLOCK_SIZE
- #define EMBER_CERTIFICATE_283K1_SIZE
- #define EMBER_PUBLIC_KEY_283K1_SIZE
- #define EMBER_PRIVATE_KEY_283K1_SIZE
- #define EMBER_SIGNATURE_283K1_SIZE
- #define __EMBERSTATUS_TYPE__
- #define EMBER_MAX_802_15_4_CHANNEL_NUMBER
- #define EMBER_MIN_802_15_4_CHANNEL_NUMBER
- #define EMBER_NUM_802_15_4_CHANNELS
- #define EMBER_ALL_802_15_4_CHANNELS_MASK
- #define EMBER_ZIGBEE_COORDINATOR_ADDRESS
- #define EMBER_NULL_NODE_ID
- #define EMBER_NULL_BINDING
- #define EMBER_TABLE_ENTRY_UNUSED_NODE_ID
- #define EMBER_MULTICAST_NODE_ID
- #define EMBER_UNKNOWN_NODE_ID
- #define EMBER_DISCOVERY_ACTIVE_NODE_ID
- #define EMBER_NULL_ADDRESS_TABLE_INDEX
- #define EMBER_ZDO_ENDPOINT
- #define EMBER_BROADCAST_ENDPOINT
- #define EMBER_ZDO_PROFILE_ID
- #define EMBER_WILDCARD_PROFILE_ID
- #define EMBER_MAXIMUM_STANDARD_PROFILE_ID
- #define EMBER_BROADCAST_TABLE_TIMEOUT_QS
- #define EMBER_MANUFACTURER_ID

- enum `EmberVersionType` {
`EMBER_VERSION_TYPE_PRE_RELEASE`, `EMBER_VERSION_TYPE_ALPHA_1`, `EMBER_VERSION_TYPE_ALPHA_2`, `EMBER_VERSION_TYPE_ALPHA_3`,
`EMBER_VERSION_TYPE_BETA_1`, `EMBER_VERSION_TYPE_BETA_2`, `EMBER_VERSION_TYPE_BETA_3`, `EMBER_VERSION_TYPE_GA` }
- enum `EmberLeaveRequestFlags` { `EMBER_ZIGBEE_LEAVE_AND_REJOIN`, `EMBER_ZIGBEE_LEAVE_AND_REMOVE_CHILDREN` }
- enum `EmberLeaveReason` {
`EMBER_LEAVE_REASON_NONE`, `EMBER_LEAVE_DUE_TO_NWK_LEAVE_MESSAGE`, `EMBER_LEAVE_DUE_TO_APS_REMOVE_MESSAGE`, `EMBER_LEAVE_DUE_TO_ZDO_LEAVE_MESSAGE`,
`EMBER_LEAVE_DUE_TO_ZLL_TOUCHLINK`, `EMBER_LEAVE_DUE_TO_APP_EVENT_1` }
- typedef `uint8_t` `EmberStatus`
- typedef `uint8_t` `EmberEUI64` [`EUI64_SIZE`]
- typedef `uint8_t` `EmberMessageBuffer`
- typedef `uint16_t` `EmberNodeId`
- typedef `uint16_t` `EmberMulticastId`
- typedef `uint16_t` `EmberPanId`
- const `EmberVersion` `emberVersion`

ZigBee Broadcast Addresses

ZigBee specifies three different broadcast addresses that reach different collections of nodes. Broadcasts are normally sent only to routers. Broadcasts can also be forwarded to end devices, either all of them or only those that do not sleep. Broadcasting to end devices is both significantly more resource-intensive and significantly less reliable than broadcasting to routers.

- #define `EMBER_BROADCAST_ADDRESS`
- #define `EMBER_RX_ON_WHEN_IDLE_BROADCAST_ADDRESS`
- #define `EMBER_SLEEPY_BROADCAST_ADDRESS`

Ember Concentrator Types

- #define `EMBER_LOW_RAM_CONCENTRATOR`
- #define `EMBER_HIGH_RAM_CONCENTRATOR`

txPowerModes for `emberSetTxPowerMode` and `mfglibSetPower`

- #define `EMBER_TX_POWER_MODE_DEFAULT`
- #define `EMBER_TX_POWER_MODE_BOOST`
- #define `EMBER_TX_POWER_MODE_ALTERNATE`
- #define `EMBER_TX_POWER_MODE_BOOST_AND_ALTERNATE`

Alarm Message and Counters Request Definitions

- #define [EMBER_PRIVATE_PROFILE_ID](#)
- #define [EMBER_PRIVATE_PROFILE_ID_START](#)
- #define [EMBER_PRIVATE_PROFILE_ID_END](#)
- #define [EMBER_BROADCAST_ALARM_CLUSTER](#)
- #define [EMBER_UNICAST_ALARM_CLUSTER](#)
- #define [EMBER_CACHED_UNICAST_ALARM_CLUSTER](#)
- #define [EMBER_REPORT_COUNTERS_REQUEST](#)
- #define [EMBER_REPORT_COUNTERS_RESPONSE](#)
- #define [EMBER_REPORT_AND_CLEAR_COUNTERS_REQUEST](#)
- #define [EMBER_REPORT_AND_CLEAR_COUNTERS_RESPONSE](#)
- #define [EMBER_OTA_CERTIFICATE_UPGRADE_CLUSTER](#)

ZDO response status.

Most responses to ZDO commands contain a status byte. The meaning of this byte is defined by the ZigBee Device Profile.

- enum [EmberZdoStatus](#) {
[EMBER_ZDP_SUCCESS](#), [EMBER_ZDP_INVALID_REQUEST_TYPE](#), [EMBER_ZDP_DEVICE_NOT_FOUND](#), [EMBER_ZDP_INVALID_ENDPOINT](#),
[EMBER_ZDP_NOT_ACTIVE](#), [EMBER_ZDP_NOT_SUPPORTED](#), [EMBER_ZDP_TIMEOUT](#), [EMBER_ZDP_NO_MATCH](#),
[EMBER_ZDP_NO_ENTRY](#), [EMBER_ZDP_NO_DESCRIPTOR](#), [EMBER_ZDP_INSUFFICIENT_SPACE](#), [EMBER_ZDP_NOT_PERMITTED](#),
[EMBER_ZDP_TABLE_FULL](#), [EMBER_ZDP_NOT_AUTHORIZED](#), [EMBER_NWK_ALREADY_PRESENT](#), [EMBER_NWK_TABLE_FULL](#),
[EMBER_NWK_UNKNOWN_DEVICE](#) }

Network and IEEE Address Request/Response

Defines for ZigBee device profile cluster IDs follow. These include descriptions of the formats of the messages.

Note that each message starts with a 1-byte transaction sequence number. This sequence number is used to match a response command frame to the request frame that it is replying to. The application shall maintain a 1-byte counter that is copied into this field and incremented by one for each command sent. When a value of 0xff is reached, the next command shall re-start the counter with a value of 0x00

```
Network request: <transaction sequence number: 1>
                 <EUI64:8> <type:1> <start index:1>
IEEE request:   <transaction sequence number: 1>
                 <node ID:2> <type:1> <start index:1>
                 <type> = 0x00 single address response, ignore the start index
                   = 0x01 extended response -> sends kid's IDs as well
Response: <transaction sequence number: 1>
          <status:1> <EUI64:8> <node ID:2>
          <ID count:1> <start index:1> <child ID:2>*
```

- #define [NETWORK_ADDRESS_REQUEST](#)
- #define [NETWORK_ADDRESS_RESPONSE](#)
- #define [IEEE_ADDRESS_REQUEST](#)
- #define [IEEE_ADDRESS_RESPONSE](#)

Node Descriptor Request/Response

@code

Request: <transaction sequence number: 1> <node ID:2>

Response: <transaction sequence number: 1> <status:1> <node ID:2>

// <node descriptor: 13> // // Node Descriptor field is divided into subfields of bitmasks as follows: // (Note: All lengths below are given in bits rather than bytes.) // Logical Type: 3 // Complex Descriptor Available: 1 // User Descriptor Available: 1 // (reserved/unused): 3 // APS Flags: 3 // Frequency Band: 5 // MAC capability flags: 8 // Manufacturer Code: 16 // Maximum buffer size: 8 // Maximum incoming transfer size: 16 // Server mask: 16 // Maximum outgoing transfer size: 16 // Descriptor Capability Flags: 8 // See ZigBee document 053474, Section 2.3.2.3 for more details.

- #define [NODE_DESCRIPTOR_REQUEST](#)
- #define [NODE_DESCRIPTOR_RESPONSE](#)

Power Descriptor Request / Response

@code

Request: <transaction sequence number: 1> <node ID:2>

Response: <transaction sequence number: 1> <status:1> <node ID:2>
 <current power mode, available power sources:1>
 <current power source, current power source level:1>

// See ZigBee document 053474, Section 2.3.2.4 for more details.

- #define [POWER_DESCRIPTOR_REQUEST](#)
- #define [POWER_DESCRIPTOR_RESPONSE](#)

Simple Descriptor Request / Response

Request: <transaction sequence number: 1>

 <node ID:2> <endpoint:1>

Response: <transaction sequence number: 1>

 <status:1> <node ID:2> <length:1> <endpoint:1>

 <app profile ID:2> <app device ID:2>

 <app device version, app flags:1>

 <input cluster count:1> <input cluster:2>*

 <output cluster count:1> <output cluster:2>*

- #define [SIMPLE_DESCRIPTOR_REQUEST](#)
- #define [SIMPLE_DESCRIPTOR_RESPONSE](#)

Active Endpoints Request / Response

Request: <transaction sequence number: 1> <node ID:2>

Response: <transaction sequence number: 1>
 <status:1> <node ID:2> <endpoint count:1> <endpoint:1>*

- #define [ACTIVE_ENDPOINTS_REQUEST](#)
- #define [ACTIVE_ENDPOINTS_RESPONSE](#)

Match Descriptors Request / Response

```
Request: <transaction sequence number: 1>
        <node ID:2> <app profile ID:2>
        <input cluster count:1> <input cluster:2>*
        <output cluster count:1> <output cluster:2>*
Response: <transaction sequence number: 1>
         <status:1> <node ID:2> <endpoint count:1> <endpoint:1>*
```

- #define [MATCH_DESCRIPTOR_REQUEST](#)
- #define [MATCH_DESCRIPTOR_RESPONSE](#)

Discovery Cache Request / Response

```
Request: <transaction sequence number: 1>
        <source node ID:2> <source EUI64:8>
Response: <transaction sequence number: 1>
         <status (== EMBER_ZDP_SUCCESS):1>
```

- #define [DISCOVERY_CACHE_REQUEST](#)
- #define [DISCOVERY_CACHE_RESPONSE](#)

End Device Announce and End Device Announce Response

```
Request: <transaction sequence number: 1>
        <node ID:2> <EUI64:8> <capabilities:1>
No response is sent.
```

- #define [END_DEVICE_ANNOUNCE](#)
- #define [END_DEVICE_ANNOUNCE_RESPONSE](#)

System Server Discovery Request / Response

This is broadcast and only servers which have matching services respond. The response contains the request services that the recipient provides.

```
Request: <transaction sequence number: 1> <server mask:2>
Response: <transaction sequence number: 1>
         <status (== EMBER_ZDP_SUCCESS):1> <server mask:2>
```

- #define [SYSTEM_SERVER_DISCOVERY_REQUEST](#)
- #define [SYSTEM_SERVER_DISCOVERY_RESPONSE](#)

Parent Announce and Parent Announce Response

This is broadcast and only servers which have matching children respond. The response contains the list of children that the recipient now holds.

```
Request: <transaction sequence number: 1>
        <number of children:1> <child EUI64:8> <child Age:4>*
Response: <transaction sequence number: 1>
         <number of children:1> <child EUI64:8> <child Age:4>*
```

- #define [PARENT_ANNOUNCE](#)
- #define [PARENT_ANNOUNCE_RESPONSE](#)

ZDO server mask bits

These are used in server discovery requests and responses.

- enum `EmberZdoServerMask` {
`EMBER_ZDP_PRIMARY_TRUST_CENTER`, `EMBER_ZDP_SECONDARY_TRUST_CENTER`, `EMBER_ZDP_PRIMARY_BINDING_TABLE_CACHE`, `EMBER_ZDP_SECONDARY_BINDING_TABLE_CACHE`,
`EMBER_ZDP_PRIMARY_DISCOVERY_CACHE`, `EMBER_ZDP_SECONDARY_DISCOVERY_CACHE`, `EMBER_ZDP_NETWORK_MANAGER` }

Find Node Cache Request / Response

This is broadcast and only discovery servers which have the information for the device of interest, or the device of interest itself, respond. The requesting device can then direct any service discovery requests to the responder.

```
Request: <transaction sequence number: 1>
         <device of interest ID:2> <d-of-i EUI64:8>
Response: <transaction sequence number: 1>
          <responder ID:2> <device of interest ID:2> <d-of-i EUI64:8>
```

- #define `FIND_NODE_CACHE_REQUEST`
- #define `FIND_NODE_CACHE_RESPONSE`

End Device Bind Request / Response

```
Request: <transaction sequence number: 1>
         <node ID:2> <EUI64:8> <endpoint:1> <app profile ID:2>
         <input cluster count:1> <input cluster:2>*
         <output cluster count:1> <output cluster:2>*
Response: <transaction sequence number: 1> <status:1>
```

- #define `END_DEVICE_BIND_REQUEST`
- #define `END_DEVICE_BIND_RESPONSE`

Binding types and Request / Response

Bind and unbind have the same formats. There are two possible formats, depending on whether the destination is a group address or a device address. Device addresses include an endpoint, groups don't.

```
Request: <transaction sequence number: 1>
         <source EUI64:8> <source endpoint:1>
         <cluster ID:2> <destination address:3 or 10>
Destination address:
         <0x01:1> <destination group:2>
Or:
         <0x03:1> <destination EUI64:8> <destination endpoint:1>
Response: <transaction sequence number: 1> <status:1>
```

- #define `UNICAST_BINDING`
- #define `UNICAST_MANY_TO_ONE_BINDING`
- #define `MULTICAST_BINDING`
- #define `BIND_REQUEST`

- #define [BIND_RESPONSE](#)
- #define [UNBIND_REQUEST](#)
- #define [UNBIND_RESPONSE](#)

LQI Table Request / Response

```
Request: <transaction sequence number: 1> <start index:1>
Response: <transaction sequence number: 1> <status:1>
         <neighbor table entries:1> <start index:1>
         <entry count:1> <entry:22>*
         <entry> = <extended PAN ID:8> <EUI64:8> <node ID:2>
                   <device type, rx on when idle, relationship:1>
                   <permit joining:1> <depth:1> <LQI:1>
```

The device-type byte has the following fields:

Name	Mask	Values
device type	0x03	0x00 coordinator 0x01 router 0x02 end device 0x03 unknown
rx mode	0x0C	0x00 off when idle 0x04 on when idle 0x08 unknown
relationship	0x70	0x00 parent 0x10 child 0x20 sibling 0x30 other 0x40 previous child
reserved	0x10	

The permit-joining byte has the following fields

Name	Mask	Values
permit joining	0x03	0x00 not accepting join requests 0x01 accepting join requests 0x02 unknown
reserved	0xFC	

- #define [LQI_TABLE_REQUEST](#)
- #define [LQI_TABLE_RESPONSE](#)

Routing Table Request / Response

```
Request: <transaction sequence number: 1> <start index:1>
Response: <transaction sequence number: 1> <status:1>
         <routing table entries:1> <start index:1>
         <entry count:1> <entry:5>*
         <entry> = <destination address:2>
                   <status:1>
                   <next hop:2>
```

The status byte has the following fields:

Name	Mask	Values
status	0x07	0x00 active 0x01 discovery underway 0x02 discovery failed 0x03 inactive 0x04 validation underway

```

flags          0x38
                0x08 memory constrained
                0x10 many-to-one
                0x20 route record required

reserved       0xC0

```

- #define ROUTING_TABLE_REQUEST
- #define ROUTING_TABLE_RESPONSE

Binding Table Request / Response

```

Request: <transaction sequence number: 1> <start index:1>
Response: <transaction sequence number: 1>
          <status:1> <binding table entries:1> <start index:1>
          <entry count:1> <entry:14/21>*
          <entry> = <source EUI64:8> <source endpoint:1> <cluster ID:2>
                   <dest addr mode:1> <dest:2/8> <dest endpoint:0/1>

```

Note

If Dest. Address Mode = 0x03, then the Long Dest. Address will be used and Dest. endpoint will be included. If Dest. Address Mode = 0x01, then the Short Dest. Address will be used and there will be no Dest. endpoint.

- #define BINDING_TABLE_REQUEST
- #define BINDING_TABLE_RESPONSE

Leave Request / Response

```

Request: <transaction sequence number: 1> <EUI64:8> <flags:1>
        The flag bits are:
        0x40 remove children
        0x80 rejoin
Response: <transaction sequence number: 1> <status:1>

```

- #define LEAVE_REQUEST
- #define LEAVE_RESPONSE
- #define LEAVE_REQUEST_REMOVE_CHILDREN_FLAG
- #define LEAVE_REQUEST_REJOIN_FLAG

Permit Joining Request / Response

```

Request: <transaction sequence number: 1>
          <duration:1> <permit authentication:1>
Response: <transaction sequence number: 1> <status:1>

```

- #define PERMIT_JOINING_REQUEST
- #define PERMIT_JOINING_RESPONSE

Network Update Request / Response

```

Request: <transaction sequence number: 1>
          <scan channels:4> <duration:1> <count:0/1> <manager:0/2>

```

If the duration is in 0x00 ... 0x05, then 'count' is present but not 'manager'. Perform 'count' scans of the given duration on the given channels.

If duration is 0xFE, then 'channels' should have a single channel and 'count' and 'manager' are not present. Switch to the indicated channel.

If duration is 0xFF, then 'count' is not present. Set the active channels and the network manager ID to the values given.

Unicast requests always get a response, which is INVALID_REQUEST if the duration is not a legal value.

```
Response: <transaction sequence number: 1> <status:1>
<scanned channels:4> <transmissions:2> <failures:2>
<energy count:1> <energy:1>*
```

- #define [NWK_UPDATE_REQUEST](#)
- #define [NWK_UPDATE_RESPONSE](#)

Unsupported

Not mandatory and not supported.

- #define [COMPLEX_DESCRIPTOR_REQUEST](#)
- #define [COMPLEX_DESCRIPTOR_RESPONSE](#)
- #define [USER_DESCRIPTOR_REQUEST](#)
- #define [USER_DESCRIPTOR_RESPONSE](#)
- #define [DISCOVERY_REGISTER_REQUEST](#)
- #define [DISCOVERY_REGISTER_RESPONSE](#)
- #define [USER_DESCRIPTOR_SET](#)
- #define [USER_DESCRIPTOR_CONFIRM](#)
- #define [NETWORK_DISCOVERY_REQUEST](#)
- #define [NETWORK_DISCOVERY_RESPONSE](#)
- #define [DIRECT_JOIN_REQUEST](#)
- #define [DIRECT_JOIN_RESPONSE](#)
- #define [CLUSTER_ID_RESPONSE_MINIMUM](#)

ZDO configuration flags.

For controlling which ZDO requests are passed to the application. These are normally controlled via the following configuration definitions:

```
EMBER_APPLICATION_RECEIVES_SUPPORTED_ZDO_REQUESTS EMBER_APPLICATION_HANDLES_UNSUPPORTED_ZDO_REQUESTS EMBER_APPLICATION_HANDLES_ENDPOINT_ZDO_REQUESTS EMBER_APPLICATION_HANDLES_BINDING_ZDO_REQUESTS
```

See ember-configuration.h for more information.

- enum [EmberZdoConfigurationFlags](#) { [EMBER_APP_RECEIVES_SUPPORTED_ZDO_REQUESTS](#), [EMBER_APP_HANDLES_UNSUPPORTED_ZDO_REQUESTS](#), [EMBER_APP_HANDLES_ZDO_ENDPOINT_REQUESTS](#), [EMBER_APP_HANDLES_ZDO_BINDING_REQUESTS](#) }

8.29.1 Detailed Description

Ember data type definitions. See [Ember Common Data Types](#) for details.

Definition in file [ember-types.h](#).

8.30 ember-types.h

```

00001
00020 #ifndef EMBER_TYPES_H
00021 #define EMBER_TYPES_H
00022
00023 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00024 #include "stack/config/ember-configuration-defaults.h"
00025 #endif
00026 #include "stack/include/ember-static-struct.h"
00027 #endif //DOXYGEN_SHOULD_SKIP_THIS
00028
00032
00036 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00037 enum EmberVersionType
00038 #else
00039 typedef uint8_t EmberVersionType;
00040 enum
00041 #endif
00042 {
00043     EMBER_VERSION_TYPE_PRE_RELEASE = 0x00,
00044
00045     //Alpha, should be used rarely
00046     EMBER_VERSION_TYPE_ALPHA_1     = 0x11,
00047     EMBER_VERSION_TYPE_ALPHA_2     = 0x12,
00048     EMBER_VERSION_TYPE_ALPHA_3     = 0x13,
00049     // Leave space in case we decide to add other types in the future.
00050     EMBER_VERSION_TYPE_BETA_1      = 0x21,
00051     EMBER_VERSION_TYPE_BETA_2      = 0x22,
00052     EMBER_VERSION_TYPE_BETA_3      = 0x23,
00053
00054
00055
00056     // Anything other than 0xAA is considered pre-release
00057     // We may define other types in the future (e.g. beta, alpha)
00058     // We chose an arbitrary number (0xAA) to allow for expansion, but
00059     // to prevent ambiguity in case 0x00 or 0xFF is accidentally retrieved
00060     // as the version type.
00061     EMBER_VERSION_TYPE_GA = 0xAA,
00062 };
00063
00067 typedef struct {
00068     EmberVersionType typeNum;
00069     PGM_P typeString;
00070 } EmberReleaseTypeStruct;
00071
00075 #define EMBER_RELEASE_TYPE_TO_STRING_STRUCT_DATA
00076 { EMBER_VERSION_TYPE_PRE_RELEASE, "Pre-Release" }, \
00077 { EMBER_VERSION_TYPE_ALPHA_1, "Alpha 1" }, \
00078 { EMBER_VERSION_TYPE_ALPHA_2, "Alpha 2" }, \
00079 { EMBER_VERSION_TYPE_ALPHA_3, "Alpha 3" }, \
00080 { EMBER_VERSION_TYPE_BETA_1, "Beta 1" }, \
00081 { EMBER_VERSION_TYPE_BETA_2, "Beta 2" }, \
00082 { EMBER_VERSION_TYPE_BETA_3, "Beta 3" }, \
00083 { EMBER_VERSION_TYPE_GA, "GA" }, \
00084 { 0xFF, NULL },
00085
00086
00090 typedef struct {
00091     uint16_t build;
00092     uint8_t major;
00093     uint8_t minor;
00094     uint8_t patch;
00095     uint8_t special;
00096     EmberVersionType type;
00097 } EmberVersion;
00098
00102 extern const EmberVersion emberVersion;
00103

```

```

00107 #define EUI64_SIZE 8
00108
00112 #define EXTENDED_PAN_ID_SIZE 8
00113
00117 #define EMBER_ENCRYPTION_KEY_SIZE 16
00118
00123 #define EMBER_CERTIFICATE_SIZE 48
00124
00128 #define EMBER_PUBLIC_KEY_SIZE 22
00129
00133 #define EMBER_PRIVATE_KEY_SIZE 21
00134
00138 #define EMBER_SMAC_SIZE 16
00139
00144 #define EMBER_SIGNATURE_SIZE 42
00145
00149 #define EMBER_AES_HASH_BLOCK_SIZE 16
00150
00155 #define EMBER_CERTIFICATE_283K1_SIZE 74
00156
00160 #define EMBER_PUBLIC_KEY_283K1_SIZE 37
00161
00165 #define EMBER_PRIVATE_KEY_283K1_SIZE 36
00166
00171 #define EMBER_SIGNATURE_283K1_SIZE 72
00172
00176 #ifndef __EMBERSTATUS_TYPE__
00177 #define __EMBERSTATUS_TYPE__
00178     typedef uint8_t EmberStatus;
00179 #endif //__EMBERSTATUS_TYPE__
00180
00181 #include "stack/include/error.h"
00182
00186 typedef uint8_t EmberEUI64[EUI64_SIZE];
00187
00197 typedef uint8_t EmberMessageBuffer;
00198
00202 typedef uint16_t EmberNodeId;
00203
00205 typedef uint16_t EmberMulticastId;
00206
00210 typedef uint16_t EmberPanId;
00211
00215 #define EMBER_MAX_802_15_4_CHANNEL_NUMBER 26
00216
00220 #define EMBER_MIN_802_15_4_CHANNEL_NUMBER 11
00221
00225 #define EMBER_NUM_802_15_4_CHANNELS \
00226     (EMBER_MAX_802_15_4_CHANNEL_NUMBER - EMBER_MIN_802_15_4_CHANNEL_NUMBER + 1)
00227
00231 #define EMBER_ALL_802_15_4_CHANNELS_MASK 0x07FFF800UL
00232
00236 #define EMBER_ZIGBEE_COORDINATOR_ADDRESS 0x0000
00237
00242 #define EMBER_NULL_NODE_ID 0xFFFF
00243
00248 #define EMBER_NULL_BINDING 0xFF
00249
00259 #define EMBER_TABLE_ENTRY_UNUSED_NODE_ID 0xFFFF
00260
00267 #define EMBER_MULTICAST_NODE_ID 0xFFFE
00268
00276 #define EMBER_UNKNOWN_NODE_ID 0xFFFD
00277
00285 #define EMBER_DISCOVERY_ACTIVE_NODE_ID 0xFFFC
00286
00291 #define EMBER_NULL_ADDRESS_TABLE_INDEX 0xFF
00292
00296 #define EMBER_ZDO_ENDPOINT 0
00297
00301 #define EMBER_BROADCAST_ENDPOINT 0xFF
00302
00306 #define EMBER_ZDO_PROFILE_ID 0x0000
00307
00311 #define EMBER_WILDCARD_PROFILE_ID 0xFFFF
00312
00316 #define EMBER_MAXIMUM_STANDARD_PROFILE_ID 0x7FFF
00317
00323 #define EMBER_BROADCAST_TABLE_TIMEOUT_QS (20 * 4)
00324

```

```

00325
00329 #define EMBER_MANUFACTURER_ID 0x1002
00330
00331
00332 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00333 enum EmberLeaveRequestFlags
00334 #else
00335 typedef uint8_t EmberLeaveRequestFlags;
00336 enum
00337 #endif
00338 {
00340     EMBER_ZIGBEE_LEAVE_AND_REJOIN          = 0x80,
00341
00343     EMBER_ZIGBEE_LEAVE_AND_REMOVE_CHILDREN
    = 0x40,
00344 };
00345
00346 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00347 enum EmberLeaveReason
00348 #else
00349 typedef uint8_t EmberLeaveReason;
00350 enum
00351 #endif
00352 {
00353     EMBER_LEAVE_REASON_NONE                = 0,
00354     EMBER_LEAVE_DUE_TO_NWK_LEAVE_MESSAGE  = 1
00355     ,
00356     EMBER_LEAVE_DUE_TO_APS_REMOVE_MESSAGE =
    2,
00357     // Currently, the stack does not process the ZDO leave message since it is
    optional
00358     EMBER_LEAVE_DUE_TO_ZDO_LEAVE_MESSAGE  = 3
00359     ,
00360     EMBER_LEAVE_DUE_TO_ZLL_TOUCHLINK      = 4,
00361     EMBER_LEAVE_DUE_TO_APP_EVENT_1        = 0xFF,
00362 };
00363
00364
00365
00378 #define EMBER_BROADCAST_ADDRESS 0xFFFFC
00379
00380 #define EMBER_RX_ON_WHEN_IDLE_BROADCAST_ADDRESS 0xFFFFD
00381
00382 #define EMBER_SLEEPY_BROADCAST_ADDRESS 0xFFFFF
00383
00386 // From table 3.51 of 053474r14
00387 #define EMBER_MIN_BROADCAST_ADDRESS 0xFFFF8
00388
00389 #define emberIsZigbeeBroadcastAddress(address) \
00390     (EMBER_MIN_BROADCAST_ADDRESS <= ((uint16_t) (address)))
00391
00392
00397 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00398 enum EmberNodeType
00399 #else
00400 typedef uint8_t EmberNodeType;
00401 enum
00402 #endif
00403 {
00405     EMBER_UNKNOWN_DEVICE = 0,
00407     EMBER_COORDINATOR = 1,
00409     EMBER_ROUTER = 2,
00411     EMBER_END_DEVICE = 3,
00415     EMBER_SLEEPY_END_DEVICE = 4,
00417     EMBER_MOBILE_END_DEVICE = 5,
00419     EMBER_RF4CE_TARGET = 6,
00421     EMBER_RF4CE_CONTROLLER = 7,
00422 };
00423
00427 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00428 enum EmberEndDeviceConfiguration
00429 #else
00430 typedef uint8_t EmberEndDeviceConfiguration;
00431 enum
00432 #endif
00433 {
00434     EMBER_END_DEVICE_CONFIG_NONE          =
    0x00,
00435     EMBER_END_DEVICE_CONFIG_PERSIST_DATA_ON_PARENT

```



```

    = 0x01,
00436 };
00437
00441 typedef struct {
00442     uint16_t panId;
00443     uint8_t channel;
00444     bool allowingJoin;
00445     uint8_t extendedPanId[8];
00446     uint8_t stackProfile;
00447     uint8_t nwkUpdateId;
00448 } EmberZigbeeNetwork;
00449
00450
00455 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00456 enum EmberNetworkInitBitmask
00457 #else
00458 typedef uint16_t EmberNetworkInitBitmask;
00459 enum
00460 #endif
00461 {
00462     EMBER_NETWORK_INIT_NO_OPTIONS = 0x0000
00463 ,
00464     EMBER_NETWORK_INIT_PARENT_INFO_IN_TOKEN
    = 0x0001,
00465 };
00466
00474 typedef struct {
00475     EmberNetworkInitBitmask bitmask;
00476 } EmberNetworkInitStruct;
00477
00478
00485 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00486 enum EmberApsOption
00487 #else
00488 typedef uint16_t EmberApsOption;
00489 enum
00490 #endif
00491 {
00493     EMBER_APS_OPTION_NONE = 0x0000,
00494
00495 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00496     EMBER_APS_OPTION_ENCRYPT_WITH_TRANSIENT_KEY = 0x0001,
00497     EMBER_APS_OPTION_USE_ALIAS_SEQUENCE_NUMBER = 0x0002,
00498 #endif
00499
00511     EMBER_APS_OPTION_DSA_SIGN = 0x0010,
00514     EMBER_APS_OPTION_ENCRYPTION = 0x0020
00515 ,
00518     EMBER_APS_OPTION_RETRY = 0x0040,
00524     EMBER_APS_OPTION_ENABLE_ROUTE_DISCOVERY
    = 0x0100,
00527     EMBER_APS_OPTION_FORCE_ROUTE_DISCOVERY
    = 0x0200,
00529     EMBER_APS_OPTION_SOURCE_EUI64 =
0x0400,
00531     EMBER_APS_OPTION_DESTINATION_EUI64 =
0x0800,
00534     EMBER_APS_OPTION_ENABLE_ADDRESS_DISCOVERY
    = 0x1000,
00539     EMBER_APS_OPTION_POLL_RESPONSE =
0x2000,
00544     EMBER_APS_OPTION_ZDO_RESPONSE_REQUIRED
    = 0x4000,
00550     EMBER_APS_OPTION_FRAGMENT =
SIGNED_ENUM 0x8000
00551 };
00552
00553
00554
00558 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00559 enum EmberIncomingMessageType
00560 #else
00561 typedef uint8_t EmberIncomingMessageType;
00562 enum
00563 #endif
00564 {
00566     EMBER_INCOMING_UNICAST,
00568     EMBER_INCOMING_UNICAST_REPLY,
00570     EMBER_INCOMING_MULTICAST,

```

```

00572     EMBER_INCOMING_MULTICAST_LOOPBACK,
00574     EMBER_INCOMING_BROADCAST,
00576     EMBER_INCOMING_BROADCAST_LOOPBACK
00577 };
00578
00579
00583 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00584 enum EmberOutgoingMessageType
00585 #else
00586 typedef uint8_t EmberOutgoingMessageType;
00587 enum
00588 #endif
00589 {
00591     EMBER_OUTGOING_DIRECT,
00593     EMBER_OUTGOING_VIA_ADDRESS_TABLE,
00595     EMBER_OUTGOING_VIA_BINDING,
00598     EMBER_OUTGOING_MULTICAST,
00601     EMBER_OUTGOING_MULTICAST_WITH_ALIAS,
00604     EMBER_OUTGOING_BROADCAST_WITH_ALIAS,
00607     EMBER_OUTGOING_BROADCAST
00608 };
00609
00615 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00616 enum EmberZigbeeCommandType
00617 #else
00618 typedef uint8_t EmberZigbeeCommandType;
00619 enum
00620 #endif
00621 {
00623     EMBER_ZIGBEE_COMMAND_TYPE_MAC,
00625     EMBER_ZIGBEE_COMMAND_TYPE_NWK,
00627     EMBER_ZIGBEE_COMMAND_TYPE_APS,
00629     EMBER_ZIGBEE_COMMAND_TYPE_ZDO,
00631     EMBER_ZIGBEE_COMMAND_TYPE_ZCL,
00632
00634     EMBER_ZIGBEE_COMMAND_TYPE_BEACON,
00635 };
00636
00640 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00641 enum EmberNetworkStatus
00642 #else
00643 typedef uint8_t EmberNetworkStatus;
00644 enum
00645 #endif
00646 {
00648     EMBER_NO_NETWORK,
00650     EMBER_JOINING_NETWORK,
00652     EMBER_JOINED_NETWORK,
00655     EMBER_JOINED_NETWORK_NO_PARENT,
00657     EMBER_LEAVING_NETWORK
00658 };
00659
00660
00664 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00665 enum EmberNetworkScanType
00666 #else
00667 typedef uint8_t EmberNetworkScanType;
00668 enum
00669 #endif
00670 {
00672     EMBER_ENERGY_SCAN,
00674     EMBER_ACTIVE_SCAN
00675 };
00676
00677
00681 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00682 enum EmberBindingType
00683 #else
00684 typedef uint8_t EmberBindingType;
00685 enum
00686 #endif
00687 {
00689     EMBER_UNUSED_BINDING           = 0,
00691     EMBER_UNICAST_BINDING           = 1,
00695     EMBER_MANY_TO_ONE_BINDING      = 2,
00699     EMBER_MULTICAST_BINDING        = 3,
00700 };
00701
00702
00711 #define EMBER_LOW_RAM_CONCENTRATOR 0xFFFF8

```

```

00712
00716 #define EMBER_HIGH_RAM_CONCENTRATOR 0xFFFF9
00717
00719
00720
00724 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00725 enum EmberJoinDecision
00726 #else
00727 typedef uint8_t EmberJoinDecision;
00728 enum
00729 #endif
00730 {
00732     EMBER_USE_PRECONFIGURED_KEY = 0,
00734     EMBER_SEND_KEY_IN_THE_CLEAR,
00736     EMBER_DENY_JOIN,
00738     EMBER_NO_ACTION
00739 };
00740
00744 #define EMBER_JOIN_DECISION_STRINGS \
00745     "use preconfigured key", \
00746     "send key in the clear", \
00747     "deny join", \
00748     "no action",
00749
00750
00756 // These map to the actual values within the APS Command frame so they cannot
00757 // be arbitrarily changed.
00758 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00759 enum EmberDeviceUpdate
00760 #else
00761 typedef uint8_t EmberDeviceUpdate;
00762 enum
00763 #endif
00764 {
00765     EMBER_STANDARD_SECURITY_SECURED_REJOIN
        = 0,
00766     EMBER_STANDARD_SECURITY_UNSECURED_JOIN
        = 1,
00767     EMBER_DEVICE_LEFT = 2,
00768     EMBER_STANDARD_SECURITY_UNSECURED_REJOIN
        = 3,
00769     EMBER_HIGH_SECURITY_SECURED_REJOIN =
        4,
00770     EMBER_HIGH_SECURITY_UNSECURED_JOIN =
        5,
00771     /* 6 Reserved */
00772     EMBER_HIGH_SECURITY_UNSECURED_REJOIN
        = 7,
00773     /* 8 - 15 Reserved */
00774 };
00775
00779 #define EMBER_DEVICE_UPDATE_STRINGS \
00780     "secured rejoin", \
00781     "UNsecured join", \
00782     "device left", \
00783     "UNsecured rejoin", \
00784     "high secured rejoin", \
00785     "high UNsecured join", \
00786     "RESERVED", \
00787     "high UNsecured rejoin", \
00788
00792 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00793 enum EmberRejoinReason
00794 #else
00795 typedef uint8_t EmberRejoinReason;
00796 enum
00797 #endif
00798 {
00799     EMBER_REJOIN_REASON_NONE = 0,
00800     EMBER_REJOIN_DUE_TO_NWK_KEY_UPDATE = 1,
00801     EMBER_REJOIN_DUE_TO_LEAVE_MESSAGE = 2,
00802     EMBER_REJOIN_DUE_TO_NO_PARENT = 3,
00803     EMBER_REJOIN_DUE_TO_ZLL_TOUCHLINK = 4,
00804
00805     // App. Framework events
00806     // 0xA0 - 0xE0
00807
00808     // Customer Defined Events
00809     // I numbered these backwards in case there is ever request
00810     // for more application events. We can expand them

```

```

00811 // without renumbering the previous ones.
00812 EMBER_REJOIN_DUE_TO_APP_EVENT_5 = 0xFB,
00813 EMBER_REJOIN_DUE_TO_APP_EVENT_4 = 0xFC,
00814 EMBER_REJOIN_DUE_TO_APP_EVENT_3 = 0xFD,
00815 EMBER_REJOIN_DUE_TO_APP_EVENT_2 = 0xFE,
00816 EMBER_REJOIN_DUE_TO_APP_EVENT_1 = 0xFF,
00817 };
00818
00822 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00823 enum EmberClusterListId
00824 #else
00825 typedef uint8_t EmberClusterListId;
00826 enum
00827 #endif
00828 {
00830     EMBER_INPUT_CLUSTER_LIST = 0,
00832     EMBER_OUTPUT_CLUSTER_LIST = 1
00833 };
00834
00835
00840 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00841 enum EmberEventUnits
00842 #else
00843 typedef uint8_t EmberEventUnits;
00844 enum
00845 #endif
00846 {
00848     EMBER_EVENT_INACTIVE = 0,
00850     EMBER_EVENT_MS_TIME,
00853     EMBER_EVENT_QS_TIME,
00856     EMBER_EVENT_MINUTE_TIME,
00858     EMBER_EVENT_ZERO_DELAY
00859 };
00860
00861
00865 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00866 enum EmberJoinMethod
00867 #else
00868 typedef uint8_t EmberJoinMethod;
00869 enum
00870 #endif
00871 {
00877     EMBER_USE_MAC_ASSOCIATION = 0,
00878
00889     EMBER_USE_NWK_REJOIN = 1,
00890
00891
00892     /* For those networks where the "permit joining" flag is never turned
00893      * on, they will need to use a NWK Rejoin. If those devices have been
00894      * preconfigured with the NWK key (including sequence number) they can use
00895      * a secured rejoin. This is only necessary for end devices since they need
00896      * a parent. Routers can simply use the ::EMBER_USE_NWK_COMMISSIONING
00897      * join method below.
00898      */
00899     EMBER_USE_NWK_REJOIN_HAVE_NWK_KEY = 2,
00900
00905     EMBER_USE_NWK_COMMISSIONING = 3,
00906 };
00907
00908
00915 typedef struct {
00917     uint8_t extendedPanId[8];
00919     uint16_t panId;
00921     int8_t radioTxPower;
00923     uint8_t radioChannel;
00928     EmberJoinMethod joinMethod;
00929
00934     EmberNodeId nwkJoinManagerId;
00940     uint8_t nwkJoinUpdateId;
00946     uint32_t channels;
00947 } EmberNetworkParameters;
00948
00949
00950 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00951 #define emberInitializeNetworkParameters(parameters) \
00952     (MEMSET(parameters, 0, sizeof(EmberNetworkParameters)))
00953 #else
00954 void emberInitializeNetworkParameters(
00955     EmberNetworkParameters* parameters);
00956 #endif

```

```

00956
00960 typedef struct {
00962     uint16_t profileId;
00964     uint16_t clusterId;
00966     uint8_t sourceEndpoint;
00968     uint8_t destinationEndpoint;
00970     EmberApsOption options;
00972     uint16_t groupId;
00974     uint8_t sequence;
00975 } EmberApsFrame;
00976
00977
00984 typedef struct {
00986     EmberBindingType type;
00988     uint8_t local;
00996     uint16_t clusterId;
00998     uint8_t remote;
01003     EmberEUI64 identifier;
01005     uint8_t networkIndex;
01006 } EmberBindingTableEntry;
01007
01008
01014 typedef struct {
01016     uint16_t shortId;
01019     uint8_t averageLqi;
01022     uint8_t inCost;
01029     uint8_t outCost;
01035     uint8_t age;
01037     EmberEUI64 longId;
01038 } EmberNeighborTableEntry;
01039
01045 typedef struct {
01047     uint16_t destination;
01049     uint16_t nextHop;
01052     uint8_t status;
01055     uint8_t age;
01058     uint8_t concentratorType;
01063     uint8_t routeRecordState;
01064 } EmberRouteTableEntry;
01065
01073 typedef struct {
01075     EmberMulticastId multicastId;
01079     uint8_t endpoint;
01081     uint8_t networkIndex;
01082 } EmberMulticastTableEntry;
01083
01088 #ifndef DOXYGEN_SHOULD_SKIP_THIS
01089 enum EmberCounterType
01090 #else
01091 typedef uint8_t EmberCounterType;
01092 enum
01093 #endif
01094 {
01096     EMBER_COUNTER_MAC_RX_BROADCAST = 0,
01098     EMBER_COUNTER_MAC_TX_BROADCAST = 1,
01100     EMBER_COUNTER_MAC_RX_UNICAST = 2,
01102     EMBER_COUNTER_MAC_TX_UNICAST_SUCCESS = 3,
01108     EMBER_COUNTER_MAC_TX_UNICAST_RETRY = 4,
01110     EMBER_COUNTER_MAC_TX_UNICAST_FAILED = 5,
01111
01113     EMBER_COUNTER_APS_DATA_RX_BROADCAST = 6,
01115     EMBER_COUNTER_APS_DATA_TX_BROADCAST = 7,
01117     EMBER_COUNTER_APS_DATA_RX_UNICAST = 8,
01119     EMBER_COUNTER_APS_DATA_TX_UNICAST_SUCCESS
= 9,
01125     EMBER_COUNTER_APS_DATA_TX_UNICAST_RETRY
= 10,
01127     EMBER_COUNTER_APS_DATA_TX_UNICAST_FAILED
= 11,
01128
01131     EMBER_COUNTER_ROUTE_DISCOVERY_INITIATED
= 12,
01132
01134     EMBER_COUNTER_NEIGHBOR_ADDED = 13,
01136     EMBER_COUNTER_NEIGHBOR_REMOVED = 14,
01138     EMBER_COUNTER_NEIGHBOR_STALE = 15,
01139
01141     EMBER_COUNTER_JOIN_INDICATION = 16,
01143     EMBER_COUNTER_CHILD_REMOVED = 17,
01144

```

```

01146 EMBER_COUNTER_ASH_OVERFLOW_ERROR = 18,
01148 EMBER_COUNTER_ASH_FRAMING_ERROR = 19,
01150 EMBER_COUNTER_ASH_OVERRUN_ERROR = 20,
01151
01154 EMBER_COUNTER_NWK_FRAME_COUNTER_FAILURE
= 21,
01155
01158 EMBER_COUNTER_APS_FRAME_COUNTER_FAILURE
= 22,
01159
01161 EMBER_COUNTER_ASH_XOFF = 23,
01162
01166 EMBER_COUNTER_APS_LINK_KEY_NOT_AUTHORIZED
= 24,
01167
01170 EMBER_COUNTER_NWK_DECRYPTION_FAILURE = 25
,
01171
01174 EMBER_COUNTER_APS_DECRYPTION_FAILURE = 26
,
01175
01180 EMBER_COUNTER_ALLOCATE_PACKET_BUFFER_FAILURE
= 27,
01181
01183 EMBER_COUNTER_RELAYED_UNICAST = 28,
01184
01196 EMBER_COUNTER_PHY_TO_MAC_QUEUE_LIMIT_REACHED
= 29,
01197
01202 EMBER_COUNTER_PACKET_VALIDATE_LIBRARY_DROPPED_COUNT
= 30,
01203
01207 EMBER_COUNTER_TYPE_NWK_RETRY_OVERFLOW =
31,
01208
01212 EMBER_COUNTER_PHY_CCA_FAIL_COUNT = 32,
01213
01217 EMBER_COUNTER_BROADCAST_TABLE_FULL = 33,
01218
01220 EMBER_COUNTER_TYPE_COUNT = 34,
01221 };
01222
01226 #define EMBER_COUNTER_STRINGS
01227     "Mac Rx Bcast",
01228     "Mac Tx Bcast",
01229     "Mac Rx Ucast",
01230     "Mac Tx Ucast",
01231     "Mac Tx Ucast Retry",
01232     "Mac Tx Ucast Fail",
01233     "APS Rx Bcast",
01234     "APS Tx Bcast",
01235     "APS Rx Ucast",
01236     "APS Tx Ucast Success",
01237     "APS Tx Ucast Retry",
01238     "APS Tx Ucast Fail",
01239     "Route Disc Initiated",
01240     "Neighbor Added",
01241     "Neighbor Removed",
01242     "Neighbor Stale",
01243     "Join Indication",
01244     "Child Moved",
01245     "ASH Overflow",
01246     "ASH Frame Error",
01247     "ASH Overrun Error",
01248     "NWK FC Failure",
01249     "APS FC Failure",
01250     "ASH XOff",
01251     "APS Unauthorized Key",
01252     "NWK Decrypt Failures",
01253     "APS Decrypt Failures",
01254     "Packet Buffer Allocate Failures",
01255     "Relayed Ucast",
01256     "Phy to MAC queue limit reached",
01257     "Packet Validate drop count",
01258     "NWK retry overflow",
01259     "CCA Failures",
01260     "Broadcast table full",
01261     NULL
01262
01264 typedef uint8_t EmberTaskId;

```

```

01265
01272 typedef struct {
01274     EmberEventUnits status;
01276     EmberTaskId taskId;
01280     uint32_t timeToExecute;
01281 } EmberEventControl;
01282
01290 typedef PGM_struct EmberEventData_S {
01292     EmberEventControl *control;
01294     void (*handler)(void);
01295 } EmberEventData;
01296
01301 typedef struct {
01302     // The time when the next event associated with this task will fire
01303     uint32_t nextEventTime;
01304     // The list of events associated with this task
01305     EmberEventData *events;
01306     // A flag that indicates the task has something to do other than events
01307     bool busy;
01308 } EmberTaskControl;
01309
01314
01319 #define EMBER_TX_POWER_MODE_DEFAULT          0x0000
01320
01323 #define EMBER_TX_POWER_MODE_BOOST            0x0001
01324
01328 #define EMBER_TX_POWER_MODE_ALTERNATE        0x0002
01329
01333 #define EMBER_TX_POWER_MODE_BOOST_AND_ALTERNATE (EMBER_TX_POWER_MODE_BOOST
\
| EMBER_TX_POWER_MODE_ALTERNATE)
01334
01335 #ifndef DOXYGEN_SHOULD_SKIP_THIS
01336 // The application does not ever need to call emberSetTxPowerMode() with the
01337 // txPowerMode parameter set to this value. This value is used internally by
01338 // the stack to indicate that the default token configuration has not been
01339 // overridden by a prior call to emberSetTxPowerMode().
01340 #define EMBER_TX_POWER_MODE_USE_TOKEN        0x8000
01341 #endif//DOXYGEN_SHOULD_SKIP_THIS
01342
01344
01349
01357 #define EMBER_PRIVATE_PROFILE_ID 0xC00E
01358
01362 #define EMBER_PRIVATE_PROFILE_ID_START 0xC00D
01363
01367 #define EMBER_PRIVATE_PROFILE_ID_END 0xC016
01368
01407 #define EMBER_BROADCAST_ALARM_CLUSTER 0x0000
01408
01445 #define EMBER_UNICAST_ALARM_CLUSTER 0x0001
01446
01462 #define EMBER_CACHED_UNICAST_ALARM_CLUSTER 0x0002
01463
01467 #define EMBER_REPORT_COUNTERS_REQUEST 0x0003
01468
01470 #define EMBER_REPORT_COUNTERS_RESPONSE 0x8003
01471
01476 #define EMBER_REPORT_AND_CLEAR_COUNTERS_REQUEST 0x0004
01477
01479 #define EMBER_REPORT_AND_CLEAR_COUNTERS_RESPONSE 0x8004
01480
01485 #define EMBER_OTA_CERTIFICATE_UPGRADE_CLUSTER 0x0005
01486
01488
01489
01492 typedef struct {
01494     uint8_t contents[EMBER_ENCRYPTION_KEY_SIZE];
01495 } EmberKeyData;
01496
01499 typedef struct {
01500     uint8_t contents[EMBER_CERTIFICATE_SIZE];
01501 } EmberCertificateData;
01502
01505 typedef struct {
01506     uint8_t contents[EMBER_PUBLIC_KEY_SIZE];
01507 } EmberPublicKeyData;
01508
01511 typedef struct {
01512     uint8_t contents[EMBER_PRIVATE_KEY_SIZE];
01513 } EmberPrivateKeyData;

```

```

01514
01517 typedef struct {
01518     uint8_t contents[EMBER_SMAC_SIZE];
01519 } EmberSmacData;
01520
01524 typedef struct {
01525     uint8_t contents[EMBER_SIGNATURE_SIZE];
01526 } EmberSignatureData;
01527
01530 typedef struct {
01531     uint8_t contents[EMBER_AES_HASH_BLOCK_SIZE];
01532 } EmberMessageDigest;
01533
01537 typedef struct {
01538     uint8_t result[EMBER_AES_HASH_BLOCK_SIZE];
01539     uint32_t length;
01540 } EmberAesMmoHashContext;
01541
01544 typedef struct {
01545     /* This is the certificate byte data. */
01546     uint8_t contents[EMBER_CERTIFICATE_283K1_SIZE];
01547 } EmberCertificate283k1Data;
01548
01551 typedef struct {
01552     uint8_t contents[EMBER_PUBLIC_KEY_283K1_SIZE];
01553 } EmberPublicKey283k1Data;
01554
01557 typedef struct {
01558     uint8_t contents[EMBER_PRIVATE_KEY_283K1_SIZE];
01559 } EmberPrivateKey283k1Data;
01560
01565 typedef struct {
01566     uint8_t contents[EMBER_SIGNATURE_283K1_SIZE];
01567 } EmberSignature283k1Data;
01568
01574 #define EMBER_STANDARD_SECURITY_MODE 0x0000
01575
01579 #define EMBER_TRUST_CENTER_NODE_ID 0x0000
01580
01581
01585 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01586 enum EmberInitialSecurityBitmask
01587 #else
01588 typedef uint16_t EmberInitialSecurityBitmask;
01589 enum
01590 #endif
01591 {
01594     EMBER_DISTRIBUTED_TRUST_CENTER_MODE
01595     = 0x0002,
01597     EMBER_TRUST_CENTER_GLOBAL_LINK_KEY
01598     = 0x0004,
01600     EMBER_PRECONFIGURED_NETWORK_KEY_MODE
01601     = 0x0008,
01602
01603     #if !defined DOXYGEN_SHOULD_SKIP_THIS
01604         // Hidden fields used internally.
01604         EMBER_HAVE_TRUST_CENTER_UNKNOWN_KEY_TOKEN = 0x0010,
01605         EMBER_HAVE_TRUST_CENTER_LINK_KEY_TOKEN   = 0x0020,
01606         EMBER_HAVE_TRUST_CENTER_MASTER_KEY_TOKEN = 0x0030,
01607     #endif
01608
01618     EMBER_HAVE_TRUST_CENTER_EUI64
01619     = 0x0040,
01619
01626     EMBER_TRUST_CENTER_USES_HASHED_LINK_KEY
01627     = 0x0084,
01627
01631     EMBER_HAVE_PRECONFIGURED_KEY
01632     = 0x0100,
01635     EMBER_HAVE_NETWORK_KEY
01636     = 0x0200,
01641     EMBER_GET_LINK_KEY_WHEN_JOINING
01642     = 0x0400,
01647     EMBER_REQUIRE_ENCRYPTED_KEY
01648     = 0x0800
01649     ,
01658     EMBER_NO_FRAME_COUNTER_RESET
01659     = 0x1000,
01664     EMBER_GET_PRECONFIGURED_KEY_FROM_INSTALL_CODE
01665     = 0x2000,
01665
01666     #if !defined DOXYGEN_SHOULD_SKIP_THIS

```



```

01667 // Internal data
01668 EM_SAVED_IN_TOKEN = 0x4000,
01669 #define EM_SECURITY_INITIALIZED 0x00008000L
01670
01671 // This is only used internally. High security is not released or supported
01672 // except for golden unit compliance.
01673 #define EMBER_HIGH_SECURITY_MODE 0x0001
01674 #else
01675 /* All other bits are reserved and must be zero. */
01676 #endif
01677 };
01678
01682 #ifndef DOXYGEN_SHOULD_SKIP_THIS
01683 enum EmberExtendedSecurityBitmask
01684 #else
01685 typedef uint16_t EmberExtendedSecurityBitmask;
01686 enum
01687 #endif
01688 {
01689 #ifndef DOXYGEN_SHOULD_SKIP_THIS
01690 // If this bit is set, we set the 'key token data' field in the Initial
01691 // Security Bitmask to 0 (No Preconfig Key token), otherwise we leave the
01692 // field as it is.
01693 EMBER_PRECONFIG_KEY_NOT_VALID = 0x0001,
01694 #endif
01695
01696 // bits 1-3 are unused.
01697
01700 EMBER_JOINER_GLOBAL_LINK_KEY = 0x0010,
01701
01707 EMBER_EXT_NO_FRAME_COUNTER_RESET = 0x0020,
01708
01709 // bit 6-7 reserved for future use (stored in TOKEN).
01710
01713 EMBER_NWK_LEAVE_REQUEST_NOT_ALLOWED =
01714 0x0100,
01715 #ifndef DOXYGEN_SHOULD_SKIP_THIS
01716
01720 EMBER_R18_STACK_BEHAVIOR = 0x0200,
01721 #endif
01722
01723 // bit 10 and 11 are stored in RAM only.
01724 // bit 11 is reserved for future use.
01725 // bits 12-15 are unused.
01726 };
01727
01730 #define EMBER_NO_TRUST_CENTER_MODE EMBER_DISTRIBUTED_TRUST_CENTER_MODE
01731
01734 #define EMBER_GLOBAL_LINK_KEY EMBER_TRUST_CENTER_GLOBAL_LINK_KEY
01735
01736
01737 #if !defined DOXYGEN_SHOULD_SKIP_THIS
01738 #define NO_TRUST_CENTER_KEY_TOKEN 0x0000
01739 #define TRUST_CENTER_KEY_TOKEN_MASK 0x0030
01740 #define SECURITY_BIT_TOKEN_MASK 0x71FF
01741
01742 #define SECURITY_LOWER_BIT_MASK 0x000000FF // ""
01743 #define SECURITY_UPPER_BIT_MASK 0x00FF0000L // ""
01744 #endif
01745
01748 typedef struct {
01753 uint16_t bitmask;
01762 EmberKeyData preconfiguredKey;
01768 EmberKeyData networkKey;
01775 uint8_t networkKeySequenceNumber;
01783 EmberEUI64 preconfiguredTrustCenterEui64
01784 } EmberInitialSecurityState;
01785
01786
01790 #ifndef DOXYGEN_SHOULD_SKIP_THIS
01791 enum EmberCurrentSecurityBitmask
01792 #else
01793 typedef uint16_t EmberCurrentSecurityBitmask;
01794 enum
01795 #endif
01796 {
01797 #if defined DOXYGEN_SHOULD_SKIP_THIS
01798 // These options are the same for Initial and Current Security state

```

```

01799
01802     EMBER_STANDARD_SECURITY_MODE_           =
01803     0x0000,
01805     EMBER_DISTRIBUTED_TRUST_CENTER_MODE_
01806     = 0x0002,
01808     EMBER_TRUST_CENTER_GLOBAL_LINK_KEY_
01809     = 0x0004,
01809 #else
01810     // Bit 3 reserved
01811 #endif
01812
01813     EMBER_HAVE_TRUST_CENTER_LINK_KEY           =
01814     0x0010,
01814
01816     EMBER_TRUST_CENTER_USES_HASHED_LINK_KEY_
01817     = 0x0084,
01817
01818     // Bits 1,5,6, 8-15 reserved
01819 };
01820
01821 #if !defined DOXYGEN_SHOULD_SKIP_THIS
01822     #define INITIAL_AND_CURRENT_BITMASK        0x00FF
01823 #endif
01824
01825
01828 typedef struct {
01831     EmberCurrentSecurityBitmask bitmask;
01835     EmberEUI64 trustCenterLongAddress;
01836 } EmberCurrentSecurityState;
01837
01838
01842 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01843 enum EmberKeyStructBitmask
01844 #else
01845 typedef uint16_t EmberKeyStructBitmask;
01846 enum
01847 #endif
01848 {
01851     EMBER_KEY_HAS_SEQUENCE_NUMBER           = 0x0001,
01855     EMBER_KEY_HAS_OUTGOING_FRAME_COUNTER =
01856     0x0002,
01859     EMBER_KEY_HAS_INCOMING_FRAME_COUNTER =
01860     0x0004,
01863     EMBER_KEY_HAS_PARTNER_EUI64             = 0x0008,
01867     EMBER_KEY_IS_AUTHORIZED                 = 0x0010,
01872     EMBER_KEY_PARTNER_IS_SLEEPY            = 0x0020,
01873
01874 };
01875
01877 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01878 enum EmberKeyType
01879 #else
01880 typedef uint8_t EmberKeyType;
01881 enum
01882 #endif
01883 {
01885     EMBER_TRUST_CENTER_LINK_KEY             = 1,
01887     EMBER_TRUST_CENTER_MASTER_KEY          = 2,
01889     EMBER_CURRENT_NETWORK_KEY              = 3,
01891     EMBER_NEXT_NETWORK_KEY                 = 4,
01893     EMBER_APPLICATION_LINK_KEY             = 5,
01895     EMBER_APPLICATION_MASTER_KEY           = 6,
01896 };
01897
01901 typedef struct {
01905     EmberKeyStructBitmask bitmask;
01907     EmberKeyType type;
01909     EmberKeyData key;
01912     uint32_t outgoingFrameCounter;
01915     uint32_t incomingFrameCounter;
01918     uint8_t sequenceNumber;
01921     EmberEUI64 partnerEUI64;
01922 } EmberKeyStruct;
01923
01924
01928 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01929 enum EmberKeyStatus
01930 #else
01931 typedef uint8_t EmberKeyStatus;
01932 enum

```

```

01933 #endif
01934 {
01935     EMBER_KEY_STATUS_NONE                = 0,
01936     EMBER_APP_LINK_KEY_ESTABLISHED       = 1,
01937     EMBER_APP_MASTER_KEY_ESTABLISHED     = 2,
01938     EMBER_TRUST_CENTER_LINK_KEY_ESTABLISHED
01939     = 3,
01940     EMBER_KEY_ESTABLISHMENT_TIMEOUT      = 4,
01941     EMBER_KEY_TABLE_FULL                 = 5,
01942
01943     // These are success status values applying only to the
01944     // Trust Center answering key requests
01945     EMBER_TC_RESPONDED_TO_KEY_REQUEST     = 6
01946 ,
01947     EMBER_TC_APP_KEY_SENT_TO_REQUESTER    =
01948     7,
01949     // These are failure status values applying only to the
01950     // Trust Center answering key requests
01951     EMBER_TC_RESPONSE_TO_KEY_REQUEST_FAILED
01952     = 8,
01953     EMBER_TC_REQUEST_KEY_TYPE_NOT_SUPPORTED
01954     = 9,
01955     EMBER_TC_NO_LINK_KEY_FOR_REQUESTER    =
01956     10,
01957     EMBER_TC_REQUESTER_EUI64_UNKNOWN      = 11
01958 ,
01959     EMBER_TC_RECEIVED_FIRST_APP_KEY_REQUEST
01960     = 12,
01961     EMBER_TC_TIMEOUT_WAITING_FOR_SECOND_APP_KEY_REQUEST
01962     = 13,
01963     EMBER_TC_NON_MATCHING_APP_KEY_REQUEST_RECEIVED
01964     = 14,
01965     EMBER_TC_FAILED_TO_SEND_APP_KEYS      = 15
01966 ,
01967     EMBER_TC_FAILED_TO_STORE_APP_KEY_REQUEST
01968     = 16,
01969     EMBER_TC_REJECTED_APP_KEY_REQUEST     =
01970     17,
01971     EMBER_TC_FAILED_TO_GENERATE_NEW_KEY    =
01972     18,
01973     EMBER_TC_FAILED_TO_SEND_TC_KEY        = 19,
01974
01975     // These are generic status values for a key requester.
01976     EMBER_TRUST_CENTER_IS_PRE_R21         = 30,
01977
01978     // These are status values applying only to the Trust Center
01979     // verifying link keys.
01980     EMBER_TC_REQUESTER_VERIFY_KEY_TIMEOUT
01981     = 50,
01982     EMBER_TC_REQUESTER_VERIFY_KEY_FAILURE
01983     = 51,
01984     EMBER_TC_REQUESTER_VERIFY_KEY_SUCCESS
01985     = 52,
01986
01987     // These are status values applying only to the key requester
01988     // verifying link keys.
01989     EMBER_VERIFY_LINK_KEY_FAILURE         = 100,
01990     EMBER_VERIFY_LINK_KEY_SUCCESS        = 101,
01991 };
01992
01993 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01994 enum EmberLinkKeyRequestPolicy
01995 #else
01996 typedef uint8_t EmberLinkKeyRequestPolicy;
01997 #endif
01998 #endif
01999 {
02000     EMBER_DENY_KEY_REQUESTS              = 0x00,
02001     EMBER_ALLOW_KEY_REQUESTS              = 0x01,
02002     EMBER_GENERATE_NEW_TC_LINK_KEY        = 0x02,
02003 };
02004
02005 #if defined DOXYGEN_SHOULD_SKIP_THIS
02006 uint8_t* emberKeyContents(EmberKeyData* key);
02007 #else
02008 #define emberKeyContents(key) ((key)->contents)
02009 #endif

```

```

02006
02014 #if defined DOXYGEN_SHOULD_SKIP_THIS
02015 uint8_t* emberCertificateContents(EmberCertificateData
    * cert);
02016 #else
02017 #define emberCertificateContents(cert) ((cert)->contents)
02018 #endif
02019
02027 #if defined DOXYGEN_SHOULD_SKIP_THIS
02028 uint8_t* emberPublicKeyContents(EmberPublicKeyData
    * key);
02029 #else
02030 #define emberPublicKeyContents(key) ((key)->contents)
02031 #endif
02032
02040 #if defined DOXYGEN_SHOULD_SKIP_THIS
02041 uint8_t* emberPrivateKeyContents(EmberPrivateKeyData
    * key);
02042 #else
02043 #define emberPrivateKeyContents(key) ((key)->contents)
02044 #endif
02045
02050 #if defined DOXYGEN_SHOULD_SKIP_THIS
02051 uint8_t* emberSmacContents(EmberSmacData* key);
02052 #else
02053 #define emberSmacContents(key) ((key)->contents)
02054 #endif
02055
02059 #if defined DOXYGEN_SHOULD_SKIP_THIS
02060 uint8_t* emberSignatureContents(EmberSignatureData
    * sig);
02061 #else
02062 #define emberSignatureContents(sig) ((sig)->contents)
02063 #endif
02064
02072 #if defined DOXYGEN_SHOULD_SKIP_THIS
02073 uint8_t* emberCertificate283k1Contents(
    EmberCertificate283k1Data* cert);
02074 #else
02075 #define emberCertificate283k1Contents(cert) ((cert)->contents)
02076 #endif
02077
02085 #if defined DOXYGEN_SHOULD_SKIP_THIS
02086 uint8_t* emberPublicKey283k1Contents(
    EmberPublicKey283k1Data* key);
02087 #else
02088 #define emberPublicKey283k1Contents(key) ((key)->contents)
02089 #endif
02090
02098 #if defined DOXYGEN_SHOULD_SKIP_THIS
02099 uint8_t* emberPrivateKey283k1Contents(
    EmberPrivateKey283k1Data* key);
02100 #else
02101 #define emberPrivateKey283k1Contents(key) ((key)->contents)
02102 #endif
02103
02107 #if defined DOXYGEN_SHOULD_SKIP_THIS
02108 uint8_t* ember283k1SignatureContents(
    Ember283k1SignatureData* sig);
02109 #else
02110 #define ember283k1SignatureContents(sig) ((sig)->contents)
02111 #endif
02112
02113 #ifndef DOXYGEN_SHOULD_SKIP_THIS
02114 enum EmberKeySettings
02115 #else
02116 typedef uint16_t EmberKeySettings;
02117 enum
02118 #endif
02119 {
02120     EMBER_KEY_PERMISSIONS_NONE = 0x0000,
02121     EMBER_KEY_PERMISSIONS_READING_ALLOWED =
02122         0x0001,
02122     EMBER_KEY_PERMISSIONS_HASHING_ALLOWED =
02123         0x0002,
02123 };
02124
02125
02129 typedef struct {
02130     EmberKeySettings keySettings;

```

```

02131 } EmberMfgSecurityStruct;
02132
02133
02138 #define EMBER_MFG_SECURITY_CONFIG_MAGIC_NUMBER 0xCABAD11FUL
02139
02140
02145 #ifdef DOXYGEN_SHOULD_SKIP_THIS
02146 enum EmberMacPassthroughType
02147 #else
02148 typedef uint8_t EmberMacPassthroughType;
02149 enum
02150 #endif
02151 {
02153     EMBER_MAC_PASSTHROUGH_NONE = 0x00,
02155     EMBER_MAC_PASSTHROUGH_SE_INTERPAN =
02156         0x01,
02157     EMBER_MAC_PASSTHROUGH_EMBERNET = 0x02,
02159     EMBER_MAC_PASSTHROUGH_EMBERNET_SOURCE =
02160         0x04,
02161     EMBER_MAC_PASSTHROUGH_APPLICATION =
02162         0x08,
02163     EMBER_MAC_PASSTHROUGH_CUSTOM = 0x10,
02164
02165     #if !defined DOXYGEN_SHOULD_SKIP_THIS
02166         EM_MAC_PASSTHROUGH_INTERNAL_ZLL = 0x80,
02168         EM_MAC_PASSTHROUGH_INTERNAL_GP = 0x40
02169     #endif
02170 };
02171
02176 typedef uint16_t EmberMacFilterMatchData;
02177
02178 #define EMBER_MAC_FILTER_MATCH_ENABLED_MASK 0x0001
02179 #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_MASK 0x0003
02180 #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_MASK 0x000C
02181 #define EMBER_MAC_FILTER_MATCH_ON_DEST_MASK 0x0030
02182 #define EMBER_MAC_FILTER_MATCH_ON_SOURCE_MASK 0x0080
02183
02184 // Globally turn on/off this filter
02185 #define EMBER_MAC_FILTER_MATCH_ENABLED 0x0000
02186 #define EMBER_MAC_FILTER_MATCH_DISABLED 0x0001
02187
02188 // Pick either one of these
02189 #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_NONE 0x0000
02190 #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_LOCAL 0x0001
02191 #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_BROADCAST 0x0002
02192
02193 // and one of these
02194 #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_NONE 0x0000
02195 #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_NON_LOCAL 0x0004
02196 #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_LOCAL 0x0008
02197
02198 // and one of these
02199 #define EMBER_MAC_FILTER_MATCH_ON_DEST_BROADCAST_SHORT 0x0000
02200 #define EMBER_MAC_FILTER_MATCH_ON_DEST_UNICAST_SHORT 0x0010
02201 #define EMBER_MAC_FILTER_MATCH_ON_DEST_UNICAST_LONG 0x0020
02202
02203 // and one of these
02204 #define EMBER_MAC_FILTER_MATCH_ON_SOURCE_LONG 0x0000
02205 #define EMBER_MAC_FILTER_MATCH_ON_SOURCE_SHORT 0x0080
02206 #define EMBER_MAC_FILTER_MATCH_ON_SOURCE_NONE 0x0100
02207
02208 // Last entry should set this and nothing else. No other bits will be
    examined.
02209 #define EMBER_MAC_FILTER_MATCH_END 0x8000
02210
02214 typedef struct {
02215     uint8_t filterIndexMatch;
02216     EmberMacPassthroughType legacyPassthroughType
02217     ;
02217     EmberMessageBuffer message;
02218 } EmberMacFilterMatchStruct;
02219
02220
02224 typedef uint8_t EmberLibraryStatus;
02225
02230
02236 #ifdef DOXYGEN_SHOULD_SKIP_THIS
02237 enum EmberZdoStatus
02238 #else

```

```

02239 typedef uint8_t EmberZdoStatus;
02240 enum
02241 #endif
02242 {
02243     // These values are taken from Table 48 of ZDP Errata 043238r003 and Table 2
02244     // of NWK 02130r10.
02245     EMBER_ZDP_SUCCESS = 0x00,
02246     // 0x01 to 0x7F are reserved
02247     EMBER_ZDP_INVALID_REQUEST_TYPE = 0x80,
02248     EMBER_ZDP_DEVICE_NOT_FOUND = 0x81,
02249     EMBER_ZDP_INVALID_ENDPOINT = 0x82,
02250     EMBER_ZDP_NOT_ACTIVE = 0x83,
02251     EMBER_ZDP_NOT_SUPPORTED = 0x84,
02252     EMBER_ZDP_TIMEOUT = 0x85,
02253     EMBER_ZDP_NO_MATCH = 0x86,
02254     // 0x87 is reserved
02255     EMBER_ZDP_NO_ENTRY = 0x88,
02256     EMBER_ZDP_NO_DESCRIPTOR = 0x89,
02257     EMBER_ZDP_INSUFFICIENT_SPACE = 0x8a,
02258     EMBER_ZDP_NOT_PERMITTED = 0x8b,
02259     EMBER_ZDP_TABLE_FULL = 0x8c,
02260     EMBER_ZDP_NOT_AUTHORIZED = 0x8d,
02261
02262     EMBER_NWK_ALREADY_PRESENT = 0xc5,
02263     EMBER_NWK_TABLE_FULL = 0xc7,
02264     EMBER_NWK_UNKNOWN_DEVICE = 0xc8
02265 };
02266
02279
02280
02281
02282
02283
02284
02285
02286
02287
02288
02289
02290
02291
02292
02293 #define NETWORK_ADDRESS_REQUEST 0x0000
02294 #define NETWORK_ADDRESS_RESPONSE 0x8000
02295 #define IEEE_ADDRESS_REQUEST 0x0001
02296 #define IEEE_ADDRESS_RESPONSE 0x8001
02297
02298
02305 // <node descriptor: 13>
02306 //
02307 // Node Descriptor field is divided into subfields of bitmasks as follows:
02308 // (Note: All lengths below are given in bits rather than bytes.)
02309 // Logical Type: 3
02310 // Complex Descriptor Available: 1
02311 // User Descriptor Available: 1
02312 // (reserved/unused): 3
02313 // APS Flags: 3
02314 // Frequency Band: 5
02315 // MAC capability flags: 8
02316 // Manufacturer Code: 16
02317 // Maximum buffer size: 8
02318 // Maximum incoming transfer size: 16
02319 // Server mask: 16
02320 // Maximum outgoing transfer size: 16
02321 // Descriptor Capability Flags: 8
02322 // See ZigBee document 053474, Section 2.3.2.3 for more details.
02324 #define NODE_DESCRIPTOR_REQUEST 0x0002
02325 #define NODE_DESCRIPTOR_RESPONSE 0x8002
02326
02327
02336 // See ZigBee document 053474, Section 2.3.2.4 for more details.
02338 #define POWER_DESCRIPTOR_REQUEST 0x0003
02339 #define POWER_DESCRIPTOR_RESPONSE 0x8003
02340
02341
02355 #define SIMPLE_DESCRIPTOR_REQUEST 0x0004
02356 #define SIMPLE_DESCRIPTOR_RESPONSE 0x8004
02357
02358
02367 #define ACTIVE_ENDPOINTS_REQUEST 0x0005

```

```

02368 #define ACTIVE_ENDPOINTS_RESPONSE    0x8005
02369
02370
02382 #define MATCH_DESCRIPTOR_REQUEST    0x0006
02383 #define MATCH_DESCRIPTOR_RESPONSE    0x8006
02384
02385
02395 #define DISCOVERY_CACHE_REQUEST    0x0012
02396 #define DISCOVERY_CACHE_RESPONSE    0x8012
02397
02398
02407 #define END_DEVICE_ANNOUNCE    0x0013
02408 #define END_DEVICE_ANNOUNCE_RESPONSE    0x8013
02409
02410
02422 #define SYSTEM_SERVER_DISCOVERY_REQUEST    0x0015
02423 #define SYSTEM_SERVER_DISCOVERY_RESPONSE    0x8015
02424
02425
02438 #define PARENT_ANNOUNCE    0x001F
02439 #define PARENT_ANNOUNCE_RESPONSE    0x801F
02440
02441
02446 #ifdef DOXYGEN_SHOULD_SKIP_THIS
02447 enum EmberZdoServerMask
02448 #else
02449 typedef uint16_t EmberZdoServerMask;
02450 enum
02451 #endif
02452 {
02453     EMBER_ZDP_PRIMARY_TRUST_CENTER    =
02454     0x0001,
02455     EMBER_ZDP_SECONDARY_TRUST_CENTER    =
02456     0x0002,
02457     EMBER_ZDP_PRIMARY_BINDING_TABLE_CACHE
02458     = 0x0004,
02459     EMBER_ZDP_SECONDARY_BINDING_TABLE_CACHE
02460     = 0x0008,
02461     EMBER_ZDP_PRIMARY_DISCOVERY_CACHE    =
02462     0x0010,
02463     EMBER_ZDP_SECONDARY_DISCOVERY_CACHE    =
02464     0x0020,
02465     EMBER_ZDP_NETWORK_MANAGER    = 0x0040,
02466     // Bits 0x0080 to 0x8000 are reserved.
02467 };
02468
02476 #define FIND_NODE_CACHE_REQUEST    0x001C
02477 #define FIND_NODE_CACHE_RESPONSE    0x801C
02478
02479
02490 #define END_DEVICE_BIND_REQUEST    0x0020
02491 #define END_DEVICE_BIND_RESPONSE    0x8020
02492
02493
02511 #define UNICAST_BINDING    0x03
02512 #define UNICAST_MANY_TO_ONE_BINDING    0x83
02513 #define MULTICAST_BINDING    0x01
02514
02515 #define BIND_REQUEST    0x0021
02516 #define BIND_RESPONSE    0x8021
02517 #define UNBIND_REQUEST    0x0022
02518 #define UNBIND_RESPONSE    0x8022
02519
02520
02568 #define LQI_TABLE_REQUEST    0x0031
02569 #define LQI_TABLE_RESPONSE    0x8031
02570
02571
02604 #define ROUTING_TABLE_REQUEST    0x0032
02605 #define ROUTING_TABLE_RESPONSE    0x8032
02606
02607
02626 #define BINDING_TABLE_REQUEST    0x0033
02627 #define BINDING_TABLE_RESPONSE    0x8033
02628
02629
02640 #define LEAVE_REQUEST    0x0034
02641 #define LEAVE_RESPONSE    0x8034
02642
02643 #define LEAVE_REQUEST_REMOVE_CHILDREN_FLAG    0x40

```

```

02644 #define LEAVE_REQUEST_REJOIN_FLAG          0x80
02645
02646
02655 #define PERMIT_JOINING_REQUEST              0x0036
02656 #define PERMIT_JOINING_RESPONSE            0x8036
02657
02658
02684 #define NWK_UPDATE_REQUEST                  0x0038
02685 #define NWK_UPDATE_RESPONSE                0x8038
02686
02687
02691 #define COMPLEX_DESCRIPTOR_REQUEST          0x0010
02692 #define COMPLEX_DESCRIPTOR_RESPONSE        0x8010
02693 #define USER_DESCRIPTOR_REQUEST            0x0011
02694 #define USER_DESCRIPTOR_RESPONSE          0x8011
02695 #define DISCOVERY_REGISTER_REQUEST         0x0012
02696 #define DISCOVERY_REGISTER_RESPONSE       0x8012
02697 #define USER_DESCRIPTOR_SET                0x0014
02698 #define USER_DESCRIPTOR_CONFIRM            0x8014
02699 #define NETWORK_DISCOVERY_REQUEST         0x0030
02700 #define NETWORK_DISCOVERY_RESPONSE        0x8030
02701 #define DIRECT_JOIN_REQUEST               0x0035
02702 #define DIRECT_JOIN_RESPONSE              0x8035
02703
02704
02705 #define CLUSTER_ID_RESPONSE_MINIMUM      0x8000
02706
02707
02720 #ifdef DOXYGEN_SHOULD_SKIP_THIS
02721 enum EmberZdoConfigurationFlags
02722 #else
02723 typedef uint8_t EmberZdoConfigurationFlags;
02724 enum
02725 #endif
02726 {
02727 {
02728     EMBER_APP_RECEIVES_SUPPORTED_ZDO_REQUESTS
02729     = 0x01,
02729     EMBER_APP_HANDLES_UNSUPPORTED_ZDO_REQUESTS
02730     = 0x02,
02730     EMBER_APP_HANDLES_ZDO_ENDPOINT_REQUESTS
02731     = 0x04,
02731     EMBER_APP_HANDLES_ZDO_BINDING_REQUESTS
02732     = 0x08
02732 };
02733
02734
02736 #if defined(EMBER_TEST)
02737 #define WEAK_TEST WEAK()//__attribute__((weak))
02738 #else
02739 #define WEAK_TEST
02740 #endif
02741
02742
02743
02744 #endif // EMBER_TYPES_H
02745
02749 #include "stack/include/zll-types.h"
02750 #include "stack/include/rf4ce-types.h"
02751 #include "stack/include/gp-types.h"
02752

```

8.31 error-def.h File Reference

Generic Messages

These messages are system wide.

- #define [EMBER_SUCCESS](#)(x00)
- #define [EMBER_ERR_FATAL](#)(x01)
- #define [EMBER_BAD_ARGUMENT](#)(x02)
- #define [EMBER_NOT_FOUND](#)(x03)

- #define [EMBER_EEPROM_MFG_STACK_VERSION_MISMATCH\(x04\)](#)
- #define [EMBER_INCOMPATIBLE_STATIC_MEMORY_DEFINITIONS\(x05\)](#)
- #define [EMBER_EEPROM_MFG_VERSION_MISMATCH\(x06\)](#)
- #define [EMBER_EEPROM_STACK_VERSION_MISMATCH\(x07\)](#)

Packet Buffer Module Errors

- #define [EMBER_NO_BUFFERS\(x18\)](#)

Serial Manager Errors

- #define [EMBER_SERIAL_INVALID_BAUD_RATE\(x20\)](#)
- #define [EMBER_SERIAL_INVALID_PORT\(x21\)](#)
- #define [EMBER_SERIAL_TX_OVERFLOW\(x22\)](#)
- #define [EMBER_SERIAL_RX_OVERFLOW\(x23\)](#)
- #define [EMBER_SERIAL_RX_FRAME_ERROR\(x24\)](#)
- #define [EMBER_SERIAL_RX_PARITY_ERROR\(x25\)](#)
- #define [EMBER_SERIAL_RX_EMPTY\(x26\)](#)
- #define [EMBER_SERIAL_RX_OVERRUN_ERROR\(x27\)](#)

MAC Errors

- #define [EMBER_MAC_TRANSMIT_QUEUE_FULL\(x39\)](#)
- #define [EMBER_MAC_UNKNOWN_HEADER_TYPE\(x3A\)](#)
- #define [EMBER_MAC_ACK_HEADER_TYPE\(x3B\)](#)
- #define [EMBER_MAC_SCANNING\(x3D\)](#)
- #define [EMBER_MAC_NO_DATA\(x31\)](#)
- #define [EMBER_MAC_JOINED_NETWORK\(x32\)](#)
- #define [EMBER_MAC_BAD_SCAN_DURATION\(x33\)](#)
- #define [EMBER_MAC_INCORRECT_SCAN_TYPE\(x34\)](#)
- #define [EMBER_MAC_INVALID_CHANNEL_MASK\(x35\)](#)
- #define [EMBER_MAC_COMMAND_TRANSMIT_FAILURE\(x36\)](#)
- #define [EMBER_MAC_NO_ACK_RECEIVED\(x40\)](#)
- #define [EMBER_MAC_RADIO_NETWORK_SWITCH_FAILED\(x41\)](#)
- #define [EMBER_MAC_INDIRECT_TIMEOUT\(x42\)](#)

Simulated EEPROM Errors

- #define [EMBER_SIM_EEPROM_ERASE_PAGE_GREEN\(x43\)](#)
- #define [EMBER_SIM_EEPROM_ERASE_PAGE_RED\(x44\)](#)
- #define [EMBER_SIM_EEPROM_FULL\(x45\)](#)
- #define [EMBER_SIM_EEPROM_INIT_1_FAILED\(x48\)](#)
- #define [EMBER_SIM_EEPROM_INIT_2_FAILED\(x49\)](#)
- #define [EMBER_SIM_EEPROM_INIT_3_FAILED\(x4A\)](#)
- #define [EMBER_SIM_EEPROM_REPAIRING\(x4D\)](#)

Flash Errors

- #define [EMBER_ERR_FLASH_WRITE_INHIBITED](#)(x46)
- #define [EMBER_ERR_FLASH_VERIFY_FAILED](#)(x47)
- #define [EMBER_ERR_FLASH_PROG_FAIL](#)(x4B)
- #define [EMBER_ERR_FLASH_ERASE_FAIL](#)(x4C)

Bootloader Errors

- #define [EMBER_ERR_BOOTLOADER_TRAP_TABLE_BAD](#)(x58)
- #define [EMBER_ERR_BOOTLOADER_TRAP_UNKNOWN](#)(x59)
- #define [EMBER_ERR_BOOTLOADER_NO_IMAGE](#)(x05A)

Transport Errors

- #define [EMBER_DELIVERY_FAILED](#)(x66)
- #define [EMBER_BINDING_INDEX_OUT_OF_RANGE](#)(x69)
- #define [EMBER_ADDRESS_TABLE_INDEX_OUT_OF_RANGE](#)(x6A)
- #define [EMBER_INVALID_BINDING_INDEX](#)(x6C)
- #define [EMBER_INVALID_CALL](#)(x70)
- #define [EMBER_COST_NOT_KNOWN](#)(x71)
- #define [EMBER_MAX_MESSAGE_LIMIT_REACHED](#)(x72)
- #define [EMBER_MESSAGE_TOO_LONG](#)(x74)
- #define [EMBER_BINDING_IS_ACTIVE](#)(x75)
- #define [EMBER_ADDRESS_TABLE_ENTRY_IS_ACTIVE](#)(x76)

Green Power status codes

- #define [EMBER_MATCH](#)(x78)
- #define [EMBER_DROP_FRAME](#)(x79)
- #define [EMBER_PASS_UNPROCESSED](#)(x7A)
- #define [EMBER_TX_THEN_DROP](#)(x7B)
- #define [EMBER_NO_SECURITY](#)(x7C)
- #define [EMBER_COUNTER_FAILURE](#)(x7D)
- #define [EMBER_AUTH_FAILURE](#)(x7E)
- #define [EMBER_UNPROCESSED](#)(x7F)

HAL Module Errors

- #define [EMBER_ADC_CONVERSION_DONE](#)(x80)
- #define [EMBER_ADC_CONVERSION_BUSY](#)(x81)
- #define [EMBER_ADC_CONVERSION_DEFERRED](#)(x82)
- #define [EMBER_ADC_NO_CONVERSION_PENDING](#)(x84)
- #define [EMBER_SLEEP_INTERRUPTED](#)(x85)

PHY Errors

- #define [EMBER_PHY_TX_UNDERFLOW](#)(x88)
- #define [EMBER_PHY_TX_INCOMPLETE](#)(x89)
- #define [EMBER_PHY_INVALID_CHANNEL](#)(x8A)
- #define [EMBER_PHY_INVALID_POWER](#)(x8B)
- #define [EMBER_PHY_TX_BUSY](#)(x8C)
- #define [EMBER_PHY_TX_CCA_FAIL](#)(x8D)
- #define [EMBER_PHY_OSCILLATOR_CHECK_FAILED](#)(x8E)
- #define [EMBER_PHY_ACK_RECEIVED](#)(x8F)

Return Codes Passed to emberStackStatusHandler()

See also ::emberStackStatusHandler().

- #define [EMBER_NETWORK_UP](#)(x90)
- #define [EMBER_NETWORK_DOWN](#)(x91)
- #define [EMBER_JOIN_FAILED](#)(x94)
- #define [EMBER_MOVE_FAILED](#)(x96)
- #define [EMBER_CANNOT_JOIN_AS_ROUTER](#)(x98)
- #define [EMBER_NODE_ID_CHANGED](#)(x99)
- #define [EMBER_PAN_ID_CHANGED](#)(x9A)
- #define [EMBER_CHANNEL_CHANGED](#)(x9B)
- #define [EMBER_NO_BEACONS](#)(xAB)
- #define [EMBER_RECEIVED_KEY_IN_THE_CLEAR](#)(xAC)
- #define [EMBER_NO_NETWORK_KEY_RECEIVED](#)(xAD)
- #define [EMBER_NO_LINK_KEY_RECEIVED](#)(xAE)
- #define [EMBER_PRECONFIGURED_KEY_REQUIRED](#)(xAF)

Security Errors

- #define [EMBER_KEY_INVALID](#)(xB2)
- #define [EMBER_INVALID_SECURITY_LEVEL](#)(x95)
- #define [EMBER_APS_ENCRYPTION_ERROR](#)(xA6)
- #define [EMBER_TRUST_CENTER_MASTER_KEY_NOT_SET](#)(xA7)
- #define [EMBER_SECURITY_STATE_NOT_SET](#)(xA8)
- #define [EMBER_KEY_TABLE_INVALID_ADDRESS](#)(xB3)
- #define [EMBER_SECURITY_CONFIGURATION_INVALID](#)(xB7)
- #define [EMBER_TOO_SOON_FOR_SWITCH_KEY](#)(xB8)
- #define [EMBER_SIGNATURE_VERIFY_FAILURE](#)(xB9)
- #define [EMBER_KEY_NOT_AUTHORIZED](#)(xBB)
- #define [EMBER_SECURITY_DATA_INVALID](#)(xBD)

Miscellaneous Network Errors

- #define [EMBER_NOT_JOINED](#)(x93)
- #define [EMBER_NETWORK_BUSY](#)(xA1)
- #define [EMBER_INVALID_ENDPOINT](#)(xA3)
- #define [EMBER_BINDING_HAS_CHANGED](#)(xA4)
- #define [EMBER_INSUFFICIENT_RANDOM_DATA](#)(xA5)
- #define [EMBER_SOURCE_ROUTE_FAILURE](#)(xA9)
- #define [EMBER_MANY_TO_ONE_ROUTE_FAILURE](#)(xAA)

Miscellaneous Utility Errors

- #define [EMBER_STACK_AND_HARDWARE_MISMATCH](#)(xB0)
- #define [EMBER_INDEX_OUT_OF_RANGE](#)(xB1)
- #define [EMBER_TABLE_FULL](#)(xB4)
- #define [EMBER_TABLE_ENTRY_ERASED](#)(xB6)
- #define [EMBER_LIBRARY_NOT_PRESENT](#)(xB5)
- #define [EMBER_OPERATION_IN_PROGRESS](#)(xBA)
- #define [EMBER_TRUST_CENTER_EUI_HAS_CHANGED](#)(xBC)

ZigBee RF4CE specific errors.

- #define [EMBER_NO_RESPONSE](#)(xC0)
- #define [EMBER_DUPLICATE_ENTRY](#)(xC1)
- #define [EMBER_NOT_PERMITTED](#)(xC2)
- #define [EMBER_DISCOVERY_TIMEOUT](#)(xC3)
- #define [EMBER_DISCOVERY_ERROR](#)(xC4)
- #define [EMBER_SECURITY_TIMEOUT](#)(xC5)
- #define [EMBER_SECURITY_FAILURE](#)(xC6)

Application Errors

These error codes are available for application use.

- #define [EMBER_APPLICATION_ERROR_0](#)(xF0)
- #define [EMBER_APPLICATION_ERROR_1](#)(xF1)
- #define [EMBER_APPLICATION_ERROR_2](#)(xF2)
- #define [EMBER_APPLICATION_ERROR_3](#)(xF3)
- #define [EMBER_APPLICATION_ERROR_4](#)(xF4)
- #define [EMBER_APPLICATION_ERROR_5](#)(xF5)
- #define [EMBER_APPLICATION_ERROR_6](#)(xF6)
- #define [EMBER_APPLICATION_ERROR_7](#)(xF7)
- #define [EMBER_APPLICATION_ERROR_8](#)(xF8)
- #define [EMBER_APPLICATION_ERROR_9](#)(xF9)
- #define [EMBER_APPLICATION_ERROR_10](#)(xFA)
- #define [EMBER_APPLICATION_ERROR_11](#)(xFB)
- #define [EMBER_APPLICATION_ERROR_12](#)(xFC)
- #define [EMBER_APPLICATION_ERROR_13](#)(xFD)
- #define [EMBER_APPLICATION_ERROR_14](#)(xFE)
- #define [EMBER_APPLICATION_ERROR_15](#)(xFF)

8.31.1 Detailed Description

Return-code definitions for EmberZNet stack API functions. See [Ember Status Codes](#) for documentation.

Definition in file [error-def.h](#).

8.32 error-def.h

```

00001
00038
00039 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00040
00043 #define EMBER_SUCCESS(0x00)
00044 #else
00045 DEFINE_ERROR(SUCCESS, 0)
00046 #endif //DOXYGEN_SHOULD_SKIP_THIS
00047
00048
00049 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00050
00053 #define EMBER_ERR_FATAL(0x01)
00054 #else
00055 DEFINE_ERROR(ERR_FATAL, 0x01)
00056 #endif //DOXYGEN_SHOULD_SKIP_THIS
00057
00058
00059 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00060
00063 #define EMBER_BAD_ARGUMENT(0x02)
00064 #else
00065 DEFINE_ERROR(BAD_ARGUMENT, 0x02)
00066 #endif //DOXYGEN_SHOULD_SKIP_THIS
00067
00068
00069 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00070
00073 #define EMBER_NOT_FOUND(0x03)
00074 #else
00075 DEFINE_ERROR(NOT_FOUND, 0x03)
00076 #endif //DOXYGEN_SHOULD_SKIP_THIS
00077
00078
00079 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00080
00084 #define EMBER_EEPROM_MFG_STACK_VERSION_MISMATCH(0x04)
00085 #else
00086 DEFINE_ERROR(EEPROM_MFG_STACK_VERSION_MISMATCH, 0x04)
00087 #endif //DOXYGEN_SHOULD_SKIP_THIS
00088
00089
00090 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00091
00095 #define EMBER_INCOMPATIBLE_STATIC_MEMORY_DEFINITIONS(0x05)
00096 #else
00097 DEFINE_ERROR(INCOMPATIBLE_STATIC_MEMORY_DEFINITIONS, 0x05)
00098 #endif //DOXYGEN_SHOULD_SKIP_THIS
00099
00100
00101 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00102
00106 #define EMBER_EEPROM_MFG_VERSION_MISMATCH(0x06)
00107 #else
00108 DEFINE_ERROR(EEPROM_MFG_VERSION_MISMATCH, 0x06)
00109 #endif //DOXYGEN_SHOULD_SKIP_THIS
00110
00111
00112 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00113
00117 #define EMBER_EEPROM_STACK_VERSION_MISMATCH(0x07)
00118 #else
00119 DEFINE_ERROR(EEPROM_STACK_VERSION_MISMATCH, 0x07)
00120 #endif //DOXYGEN_SHOULD_SKIP_THIS
00121
00122
00123
00124
00129
00130 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00131
00134 #define EMBER_NO_BUFFERS(0x18)
00135 #else
00136 DEFINE_ERROR(NO_BUFFERS, 0x18)
00137 #endif //DOXYGEN_SHOULD_SKIP_THIS
00138
00140
00145

```

```

00146 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00147
00150 #define EMBER_SERIAL_INVALID_BAUD_RATE(0x20)
00151 #else
00152 DEFINE_ERROR(SERIAL_INVALID_BAUD_RATE, 0x20)
00153 #endif //DOXYGEN_SHOULD_SKIP_THIS
00154
00155
00156 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00157
00160 #define EMBER_SERIAL_INVALID_PORT(0x21)
00161 #else
00162 DEFINE_ERROR(SERIAL_INVALID_PORT, 0x21)
00163 #endif //DOXYGEN_SHOULD_SKIP_THIS
00164
00165
00166 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00167
00170 #define EMBER_SERIAL_TX_OVERFLOW(0x22)
00171 #else
00172 DEFINE_ERROR(SERIAL_TX_OVERFLOW, 0x22)
00173 #endif //DOXYGEN_SHOULD_SKIP_THIS
00174
00175
00176 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00177
00181 #define EMBER_SERIAL_RX_OVERFLOW(0x23)
00182 #else
00183 DEFINE_ERROR(SERIAL_RX_OVERFLOW, 0x23)
00184 #endif //DOXYGEN_SHOULD_SKIP_THIS
00185
00186
00187 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00188
00191 #define EMBER_SERIAL_RX_FRAME_ERROR(0x24)
00192 #else
00193 DEFINE_ERROR(SERIAL_RX_FRAME_ERROR, 0x24)
00194 #endif //DOXYGEN_SHOULD_SKIP_THIS
00195
00196
00197 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00198
00201 #define EMBER_SERIAL_RX_PARITY_ERROR(0x25)
00202 #else
00203 DEFINE_ERROR(SERIAL_RX_PARITY_ERROR, 0x25)
00204 #endif //DOXYGEN_SHOULD_SKIP_THIS
00205
00206
00207 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00208
00211 #define EMBER_SERIAL_RX_EMPTY(0x26)
00212 #else
00213 DEFINE_ERROR(SERIAL_RX_EMPTY, 0x26)
00214 #endif //DOXYGEN_SHOULD_SKIP_THIS
00215
00216
00217 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00218
00222 #define EMBER_SERIAL_RX_OVERRUN_ERROR(0x27)
00223 #else
00224 DEFINE_ERROR(SERIAL_RX_OVERRUN_ERROR, 0x27)
00225 #endif //DOXYGEN_SHOULD_SKIP_THIS
00226
00228
00233
00234 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00235
00238 #define EMBER_MAC_TRANSMIT_QUEUE_FULL(0x39)
00239 #else
00240 // Internal
00241 DEFINE_ERROR(MAC_TRANSMIT_QUEUE_FULL, 0x39)
00242 #endif //DOXYGEN_SHOULD_SKIP_THIS
00243
00244
00245 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00246
00249 #define EMBER_MAC_UNKNOWN_HEADER_TYPE(0x3A)
00250 #else
00251 DEFINE_ERROR(MAC_UNKNOWN_HEADER_TYPE, 0x3A)
00252 #endif //DOXYGEN_SHOULD_SKIP_THIS

```

```

00253
00254 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00255
00258 #define EMBER_MAC_ACK_HEADER_TYPE(0x3B)
00259 #else
00260 DEFINE_ERROR(MAC_ACK_HEADER_TYPE, 0x3B)
00261 #endif //DOXYGEN_SHOULD_SKIP_THIS
00262
00263
00264
00265 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00266
00269 #define EMBER_MAC_SCANNING(0x3D)
00270 #else
00271 DEFINE_ERROR(MAC_SCANNING, 0x3D)
00272 #endif //DOXYGEN_SHOULD_SKIP_THIS
00273
00274
00275 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00276
00279 #define EMBER_MAC_NO_DATA(0x31)
00280 #else
00281 DEFINE_ERROR(MAC_NO_DATA, 0x31)
00282 #endif //DOXYGEN_SHOULD_SKIP_THIS
00283
00284
00285 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00286
00289 #define EMBER_MAC_JOINED_NETWORK(0x32)
00290 #else
00291 DEFINE_ERROR(MAC_JOINED_NETWORK, 0x32)
00292 #endif //DOXYGEN_SHOULD_SKIP_THIS
00293
00294
00295 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00296
00300 #define EMBER_MAC_BAD_SCAN_DURATION(0x33)
00301 #else
00302 DEFINE_ERROR(MAC_BAD_SCAN_DURATION, 0x33)
00303 #endif //DOXYGEN_SHOULD_SKIP_THIS
00304
00305
00306 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00307
00310 #define EMBER_MAC_INCORRECT_SCAN_TYPE(0x34)
00311 #else
00312 DEFINE_ERROR(MAC_INCORRECT_SCAN_TYPE, 0x34)
00313 #endif //DOXYGEN_SHOULD_SKIP_THIS
00314
00315
00316 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00317
00320 #define EMBER_MAC_INVALID_CHANNEL_MASK(0x35)
00321 #else
00322 DEFINE_ERROR(MAC_INVALID_CHANNEL_MASK, 0x35)
00323 #endif //DOXYGEN_SHOULD_SKIP_THIS
00324
00325
00326 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00327
00331 #define EMBER_MAC_COMMAND_TRANSMIT_FAILURE(0x36)
00332 #else
00333 DEFINE_ERROR(MAC_COMMAND_TRANSMIT_FAILURE, 0x36)
00334 #endif //DOXYGEN_SHOULD_SKIP_THIS
00335
00336
00337 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00338
00342 #define EMBER_MAC_NO_ACK_RECEIVED(0x40)
00343 #else
00344 DEFINE_ERROR(MAC_NO_ACK_RECEIVED, 0x40)
00345 #endif //DOXYGEN_SHOULD_SKIP_THIS
00346
00347
00348 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00349
00353 #define EMBER_MAC_RADIO_NETWORK_SWITCH_FAILED(0x41)
00354 #else
00355 DEFINE_ERROR(MAC_RADIO_NETWORK_SWITCH_FAILED, 0x41)
00356 #endif //DOXYGEN_SHOULD_SKIP_THIS

```

```

00357
00358
00359 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00360
00363 #define EMBER_MAC_INDIRECT_TIMEOUT(0x42)
00364 #else
00365 DEFINE_ERROR(MAC_INDIRECT_TIMEOUT, 0x42)
00366 #endif //DOXYGEN_SHOULD_SKIP_THIS
00367
00369
00370
00375
00376
00377 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00378
00386 #define EMBER_SIM_EEPROM_ERASE_PAGE_GREEN(0x43)
00387 #else
00388 DEFINE_ERROR(SIM_EEPROM_ERASE_PAGE_GREEN, 0x43)
00389 #endif //DOXYGEN_SHOULD_SKIP_THIS
00390
00391
00392 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00393
00402 #define EMBER_SIM_EEPROM_ERASE_PAGE_RED(0x44)
00403 #else
00404 DEFINE_ERROR(SIM_EEPROM_ERASE_PAGE_RED, 0x44)
00405 #endif //DOXYGEN_SHOULD_SKIP_THIS
00406
00407
00408 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00409
00417 #define EMBER_SIM_EEPROM_FULL(0x45)
00418 #else
00419 DEFINE_ERROR(SIM_EEPROM_FULL, 0x45)
00420 #endif //DOXYGEN_SHOULD_SKIP_THIS
00421
00422
00423 // Errors 46 and 47 are now defined below in the
00424 // flash error block (was attempting to prevent renumbering)
00425
00426
00427 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00428
00435 #define EMBER_SIM_EEPROM_INIT_1_FAILED(0x48)
00436 #else
00437 DEFINE_ERROR(SIM_EEPROM_INIT_1_FAILED, 0x48)
00438 #endif //DOXYGEN_SHOULD_SKIP_THIS
00439
00440
00441 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00442
00448 #define EMBER_SIM_EEPROM_INIT_2_FAILED(0x49)
00449 #else
00450 DEFINE_ERROR(SIM_EEPROM_INIT_2_FAILED, 0x49)
00451 #endif //DOXYGEN_SHOULD_SKIP_THIS
00452
00453
00454 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00455
00462 #define EMBER_SIM_EEPROM_INIT_3_FAILED(0x4A)
00463 #else
00464 DEFINE_ERROR(SIM_EEPROM_INIT_3_FAILED, 0x4A)
00465 #endif //DOXYGEN_SHOULD_SKIP_THIS
00466
00467
00468 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00469
00480 #define EMBER_SIM_EEPROM_REPAIRING(0x4D)
00481 #else
00482 DEFINE_ERROR(SIM_EEPROM_REPAIRING, 0x4D)
00483 #endif //DOXYGEN_SHOULD_SKIP_THIS
00484
00486
00487
00492
00493 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00494
00501 #define EMBER_ERR_FLASH_WRITE_INHIBITED(0x46)
00502 #else
00503 DEFINE_ERROR(ERR_FLASH_WRITE_INHIBITED, 0x46)

```



```

00504 #endif //DOXYGEN_SHOULD_SKIP_THIS
00505
00506
00507 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00508
00514 #define EMBER_ERR_FLASH_VERIFY_FAILED(0x47)
00515 #else
00516 DEFINE_ERROR(ERR_FLASH_VERIFY_FAILED, 0x47)
00517 #endif //DOXYGEN_SHOULD_SKIP_THIS
00518
00519
00520 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00521
00527 #define EMBER_ERR_FLASH_PROG_FAIL(0x4B)
00528 #else
00529 DEFINE_ERROR(ERR_FLASH_PROG_FAIL, 0x4B)
00530 #endif //DOXYGEN_SHOULD_SKIP_THIS
00531
00532
00533 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00534
00540 #define EMBER_ERR_FLASH_ERASE_FAIL(0x4C)
00541 #else
00542 DEFINE_ERROR(ERR_FLASH_ERASE_FAIL, 0x4C)
00543 #endif //DOXYGEN_SHOULD_SKIP_THIS
00544
00546
00547
00552
00553
00554 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00555
00559 #define EMBER_ERR_BOOTLOADER_TRAP_TABLE_BAD(0x58)
00560 #else
00561 DEFINE_ERROR(ERR_BOOTLOADER_TRAP_TABLE_BAD, 0x58)
00562 #endif //DOXYGEN_SHOULD_SKIP_THIS
00563
00564
00565 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00566
00570 #define EMBER_ERR_BOOTLOADER_TRAP_UNKNOWN(0x59)
00571 #else
00572 DEFINE_ERROR(ERR_BOOTLOADER_TRAP_UNKNOWN, 0x59)
00573 #endif //DOXYGEN_SHOULD_SKIP_THIS
00574
00575
00576 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00577
00581 #define EMBER_ERR_BOOTLOADER_NO_IMAGE(0x05A)
00582 #else
00583 DEFINE_ERROR(ERR_BOOTLOADER_NO_IMAGE, 0x05A)
00584 #endif //DOXYGEN_SHOULD_SKIP_THIS
00585
00587
00588
00593
00594 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00595
00599 #define EMBER_DELIVERY_FAILED(0x66)
00600 #else
00601 DEFINE_ERROR(DELIVERY_FAILED, 0x66)
00602 #endif //DOXYGEN_SHOULD_SKIP_THIS
00603
00604
00605 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00606
00609 #define EMBER_BINDING_INDEX_OUT_OF_RANGE(0x69)
00610 #else
00611 DEFINE_ERROR(BINDING_INDEX_OUT_OF_RANGE, 0x69)
00612 #endif //DOXYGEN_SHOULD_SKIP_THIS
00613
00614
00615 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00616
00620 #define EMBER_ADDRESS_TABLE_INDEX_OUT_OF_RANGE(0x6A)
00621 #else
00622 DEFINE_ERROR(ADDRESS_TABLE_INDEX_OUT_OF_RANGE, 0x6A)
00623 #endif //DOXYGEN_SHOULD_SKIP_THIS
00624
00625

```

```

00626 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00627
00630 #define EMBER_INVALID_BINDING_INDEX(0x6C)
00631 #else
00632 DEFINE_ERROR(INVALID_BINDING_INDEX, 0x6C)
00633 #endif //DOXYGEN_SHOULD_SKIP_THIS
00634
00635
00636 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00637
00641 #define EMBER_INVALID_CALL(0x70)
00642 #else
00643 DEFINE_ERROR(INVALID_CALL, 0x70)
00644 #endif //DOXYGEN_SHOULD_SKIP_THIS
00645
00646
00647 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00648
00651 #define EMBER_COST_NOT_KNOWN(0x71)
00652 #else
00653 DEFINE_ERROR(COST_NOT_KNOWN, 0x71)
00654 #endif //DOXYGEN_SHOULD_SKIP_THIS
00655
00656
00657 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00658
00662 #define EMBER_MAX_MESSAGE_LIMIT_REACHED(0x72)
00663 #else
00664 DEFINE_ERROR(MAX_MESSAGE_LIMIT_REACHED, 0x72)
00665 #endif //DOXYGEN_SHOULD_SKIP_THIS
00666
00667 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00668
00672 #define EMBER_MESSAGE_TOO_LONG(0x74)
00673 #else
00674 DEFINE_ERROR(MESSAGE_TOO_LONG, 0x74)
00675 #endif //DOXYGEN_SHOULD_SKIP_THIS
00676
00677
00678 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00679
00683 #define EMBER_BINDING_IS_ACTIVE(0x75)
00684 #else
00685 DEFINE_ERROR(BINDING_IS_ACTIVE, 0x75)
00686 #endif //DOXYGEN_SHOULD_SKIP_THIS
00687
00688 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00689
00693 #define EMBER_ADDRESS_TABLE_ENTRY_IS_ACTIVE(0x76)
00694 #else
00695 DEFINE_ERROR(ADDRESS_TABLE_ENTRY_IS_ACTIVE, 0x76)
00696 #endif //DOXYGEN_SHOULD_SKIP_THIS
00697
00699 //
00700
00705
00706 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00707
00710 #define EMBER_MATCH(0x78)
00711 #else
00712 DEFINE_ERROR(MATCH, 0x78)
00713 #endif //DOXYGEN_SHOULD_SKIP_THIS
00714 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00715
00718 #define EMBER_DROP_FRAME(0x79)
00719 #else
00720 DEFINE_ERROR(DROP_FRAME, 0x79)
00721 #endif //DOXYGEN_SHOULD_SKIP_THIS
00722
00725 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00726 #define EMBER_PASS_UNPROCESSED(0x7A)
00727 #else
00728 DEFINE_ERROR(PASS_UNPROCESSED, 0x7A)
00729 #endif //DOXYGEN_SHOULD_SKIP_THIS
00730
00733 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00734 #define EMBER_TX_THEN_DROP(0x7B)
00735 #else
00736 DEFINE_ERROR(TX_THEN_DROP, 0x7B)
00737 #endif //DOXYGEN_SHOULD_SKIP_THIS

```

```

00738
00741 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00742 #define EMBER_NO_SECURITY(0x7C)
00743 #else
00744 DEFINE_ERROR(NO_SECURITY, 0x7C)
00745 #endif //DOXYGEN_SHOULD_SKIP_THIS
00746
00749 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00750 #define EMBER_COUNTER_FAILURE(0x7D)
00751 #else
00752 DEFINE_ERROR(COUNTER_FAILURE, 0x7D)
00753 #endif //DOXYGEN_SHOULD_SKIP_THIS
00754
00757 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00758 #define EMBER_AUTH_FAILURE(0x7E)
00759 #else
00760 DEFINE_ERROR(AUTH_FAILURE, 0x7E)
00761 #endif //DOXYGEN_SHOULD_SKIP_THIS
00762
00765 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00766 #define EMBER_UNPROCESSED(0x7F)
00767 #else
00768 DEFINE_ERROR(UNPROCESSED, 0x7F)
00769 #endif //DOXYGEN_SHOULD_SKIP_THIS
00770
00772 //
00773
00778
00779
00780 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00781
00784 #define EMBER_ADC_CONVERSION_DONE(0x80)
00785 #else
00786 DEFINE_ERROR(ADC_CONVERSION_DONE, 0x80)
00787 #endif //DOXYGEN_SHOULD_SKIP_THIS
00788
00789
00790 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00791
00795 #define EMBER_ADC_CONVERSION_BUSY(0x81)
00796 #else
00797 DEFINE_ERROR(ADC_CONVERSION_BUSY, 0x81)
00798 #endif //DOXYGEN_SHOULD_SKIP_THIS
00799
00800
00801 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00802
00806 #define EMBER_ADC_CONVERSION_DEFERRED(0x82)
00807 #else
00808 DEFINE_ERROR(ADC_CONVERSION_DEFERRED, 0x82)
00809 #endif //DOXYGEN_SHOULD_SKIP_THIS
00810
00811
00812 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00813
00816 #define EMBER_ADC_NO_CONVERSION_PENDING(0x84)
00817 #else
00818 DEFINE_ERROR(ADC_NO_CONVERSION_PENDING, 0x84)
00819 #endif //DOXYGEN_SHOULD_SKIP_THIS
00820
00821
00822 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00823
00827 #define EMBER_SLEEP_INTERRUPTED(0x85)
00828 #else
00829 DEFINE_ERROR(SLEEP_INTERRUPTED, 0x85)
00830 #endif //DOXYGEN_SHOULD_SKIP_THIS
00831
00833
00838
00839
00840 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00841
00844 #define EMBER_PHY_TX_UNDERFLOW(0x88)
00845 #else
00846 DEFINE_ERROR(PHY_TX_UNDERFLOW, 0x88)
00847 #endif //DOXYGEN_SHOULD_SKIP_THIS
00848
00849
00850 #ifdef DOXYGEN_SHOULD_SKIP_THIS

```

```

00851
00854 #define EMBER_PHY_TX_INCOMPLETE(0x89)
00855 #else
00856 DEFINE_ERROR(PHY_TX_INCOMPLETE, 0x89)
00857 #endif //DOXYGEN_SHOULD_SKIP_THIS
00858
00859
00860 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00861
00864 #define EMBER_PHY_INVALID_CHANNEL(0x8A)
00865 #else
00866 DEFINE_ERROR(PHY_INVALID_CHANNEL, 0x8A)
00867 #endif //DOXYGEN_SHOULD_SKIP_THIS
00868
00869
00870 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00871
00874 #define EMBER_PHY_INVALID_POWER(0x8B)
00875 #else
00876 DEFINE_ERROR(PHY_INVALID_POWER, 0x8B)
00877 #endif //DOXYGEN_SHOULD_SKIP_THIS
00878
00879
00880 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00881
00885 #define EMBER_PHY_TX_BUSY(0x8C)
00886 #else
00887 DEFINE_ERROR(PHY_TX_BUSY, 0x8C)
00888 #endif //DOXYGEN_SHOULD_SKIP_THIS
00889
00890
00891 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00892
00896 #define EMBER_PHY_TX_CCA_FAIL(0x8D)
00897 #else
00898 DEFINE_ERROR(PHY_TX_CCA_FAIL, 0x8D)
00899 #endif //DOXYGEN_SHOULD_SKIP_THIS
00900
00901
00902 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00903
00907 #define EMBER_PHY_OSCILLATOR_CHECK_FAILED(0x8E)
00908 #else
00909 DEFINE_ERROR(PHY_OSCILLATOR_CHECK_FAILED, 0x8E)
00910 #endif //DOXYGEN_SHOULD_SKIP_THIS
00911
00912
00913 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00914
00917 #define EMBER_PHY_ACK_RECEIVED(0x8F)
00918 #else
00919 DEFINE_ERROR(PHY_ACK_RECEIVED, 0x8F)
00920 #endif //DOXYGEN_SHOULD_SKIP_THIS
00921
00923
00929
00930
00931 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00932
00936 #define EMBER_NETWORK_UP(0x90)
00937 #else
00938 DEFINE_ERROR(NETWORK_UP, 0x90)
00939 #endif //DOXYGEN_SHOULD_SKIP_THIS
00940
00941
00942 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00943
00946 #define EMBER_NETWORK_DOWN(0x91)
00947 #else
00948 DEFINE_ERROR(NETWORK_DOWN, 0x91)
00949 #endif //DOXYGEN_SHOULD_SKIP_THIS
00950
00951
00952 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00953
00956 #define EMBER_JOIN_FAILED(0x94)
00957 #else
00958 DEFINE_ERROR(JOIN_FAILED, 0x94)
00959 #endif //DOXYGEN_SHOULD_SKIP_THIS
00960

```

```

00961
00962 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00963
00967 #define EMBER_MOVE_FAILED(0x96)
00968 #else
00969 DEFINE_ERROR(MOVE_FAILED, 0x96)
00970 #endif //DOXYGEN_SHOULD_SKIP_THIS
00971
00972
00973 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00974
00979 #define EMBER_CANNOT_JOIN_AS_ROUTER(0x98)
00980 #else
00981 DEFINE_ERROR(CANNOT_JOIN_AS_ROUTER, 0x98)
00982 #endif //DOXYGEN_SHOULD_SKIP_THIS
00983
00984
00985 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00986
00989 #define EMBER_NODE_ID_CHANGED(0x99)
00990 #else
00991 DEFINE_ERROR(NODE_ID_CHANGED, 0x99)
00992 #endif
00993
00994
00995 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00996
00999 #define EMBER_PAN_ID_CHANGED(0x9A)
01000 #else
01001 DEFINE_ERROR(PAN_ID_CHANGED, 0x9A)
01002 #endif
01003
01004 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01005
01007 #define EMBER_CHANNEL_CHANGED(0x9B)
01008 #else
01009 DEFINE_ERROR(CHANNEL_CHANGED, 0x9B)
01010 #endif
01011
01012 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01013
01016 #define EMBER_NO_BEACONS(0xAB)
01017 #else
01018 DEFINE_ERROR(NO_BEACONS, 0xAB)
01019 #endif
01020
01021
01022 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01023
01027 #define EMBER_RECEIVED_KEY_IN_THE_CLEAR(0xAC)
01028 #else
01029 DEFINE_ERROR(RECEIVED_KEY_IN_THE_CLEAR, 0xAC)
01030 #endif
01031
01032
01033 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01034
01037 #define EMBER_NO_NETWORK_KEY_RECEIVED(0xAD)
01038 #else
01039 DEFINE_ERROR(NO_NETWORK_KEY_RECEIVED, 0xAD)
01040 #endif
01041
01042
01043 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01044
01047 #define EMBER_NO_LINK_KEY_RECEIVED(0xAE)
01048 #else
01049 DEFINE_ERROR(NO_LINK_KEY_RECEIVED, 0xAE)
01050 #endif
01051
01052
01053 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01054
01058 #define EMBER_PRECONFIGURED_KEY_REQUIRED(0xAF)
01059 #else
01060 DEFINE_ERROR(PRECONFIGURED_KEY_REQUIRED, 0xAF)
01061 #endif
01062
01063
01065

```

```

01069 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01070
01074 #define EMBER_KEY_INVALID(0xB2)
01075 #else
01076 DEFINE_ERROR(KEY_INVALID, 0xB2)
01077 #endif // DOXYGEN_SHOULD_SKIP_THIS
01078
01079 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01080
01084 #define EMBER_INVALID_SECURITY_LEVEL(0x95)
01085 #else
01086 DEFINE_ERROR(INVALID_SECURITY_LEVEL, 0x95)
01087 #endif //DOXYGEN_SHOULD_SKIP_THIS
01088
01089 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01090
01098 #define EMBER_APS_ENCRYPTION_ERROR(0xA6)
01099 #else
01100     DEFINE_ERROR(APS_ENCRYPTION_ERROR, 0xA6)
01101 #endif //DOXYGEN_SHOULD_SKIP_THIS
01102
01103 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01104
01107 #define EMBER_TRUST_CENTER_MASTER_KEY_NOT_SET(0xA7)
01108 #else
01109     DEFINE_ERROR(TRUST_CENTER_MASTER_KEY_NOT_SET, 0xA7)
01110 #endif //DOXYGEN_SHOULD_SKIP_THIS
01111
01112 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01113
01116 #define EMBER_SECURITY_STATE_NOT_SET(0xA8)
01117 #else
01118     DEFINE_ERROR(SEcurity_STATE_NOT_SET, 0xA8)
01119 #endif //DOXYGEN_SHOULD_SKIP_THIS
01120
01121 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01122
01129 #define EMBER_KEY_TABLE_INVALID_ADDRESS(0xB3)
01130 #else
01131 DEFINE_ERROR(KEY_TABLE_INVALID_ADDRESS, 0xB3)
01132 #endif //DOXYGEN_SHOULD_SKIP_THIS
01133
01134 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01135
01138 #define EMBER_SECURITY_CONFIGURATION_INVALID(0xB7)
01139 #else
01140 DEFINE_ERROR(SEcurity_CONFIGURATION_INVALID, 0xB7)
01141 #endif //DOXYGEN_SHOULD_SKIP_THIS
01142
01143 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01144
01149 #define EMBER_TOO_SOON_FOR_SWITCH_KEY(0xB8)
01150 #else
01151     DEFINE_ERROR(TOO_SOON_FOR_SWITCH_KEY, 0xB8)
01152 #endif
01153
01154 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01155
01158 #define EMBER_SIGNATURE_VERIFY_FAILURE(0xB9)
01159 #else
01160     DEFINE_ERROR(SIGNATURE_VERIFY_FAILURE, 0xB9)
01161 #endif
01162
01163 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01164
01170 #define EMBER_KEY_NOT_AUTHORIZED(0xBB)
01171 #else
01172     DEFINE_ERROR(KEY_NOT_AUTHORIZED, 0xBB)
01173 #endif
01174
01175
01176 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01177
01180 #define EMBER_SECURITY_DATA_INVALID(0xBD)
01181 #else
01182     DEFINE_ERROR(SEcurity_DATA_INVALID, 0xBD)
01183 #endif
01184
01186
01187

```

```

01192
01193
01194 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01195
01198 #define EMBER_NOT_JOINED(0x93)
01199 #else
01200 DEFINE_ERROR(NOT_JOINED, 0x93)
01201 #endif //DOXYGEN_SHOULD_SKIP_THIS
01202
01203 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01204
01208 #define EMBER_NETWORK_BUSY(0xA1)
01209 #else
01210 DEFINE_ERROR(NETWORK_BUSY, 0xA1)
01211 #endif //DOXYGEN_SHOULD_SKIP_THIS
01212
01213
01214 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01215
01219 #define EMBER_INVALID_ENDPOINT(0xA3)
01220 #else
01221 DEFINE_ERROR(INVALID_ENDPOINT, 0xA3)
01222 #endif //DOXYGEN_SHOULD_SKIP_THIS
01223
01224
01225 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01226
01230 #define EMBER_BINDING_HAS_CHANGED(0xA4)
01231 #else
01232 DEFINE_ERROR(BINDING_HAS_CHANGED, 0xA4)
01233 #endif //DOXYGEN_SHOULD_SKIP_THIS
01234
01235 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01236
01240 #define EMBER_INSUFFICIENT_RANDOM_DATA(0xA5)
01241 #else
01242     DEFINE_ERROR(INSUFFICIENT_RANDOM_DATA, 0xA5)
01243 #endif //DOXYGEN_SHOULD_SKIP_THIS
01244
01245
01246 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01247
01250 #define EMBER_SOURCE_ROUTE_FAILURE(0xA9)
01251 #else
01252     DEFINE_ERROR(SOURCE_ROUTE_FAILURE, 0xA9)
01253 #endif
01254
01255 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01256
01261 #define EMBER_MANY_TO_ONE_ROUTE_FAILURE(0xAA)
01262 #else
01263     DEFINE_ERROR(MANY_TO_ONE_ROUTE_FAILURE, 0xAA)
01264 #endif
01265
01266
01268
01273
01274
01275 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01276
01282 #define EMBER_STACK_AND_HARDWARE_MISMATCH(0xB0)
01283 #else
01284 DEFINE_ERROR(STACK_AND_HARDWARE_MISMATCH, 0xB0)
01285 #endif //DOXYGEN_SHOULD_SKIP_THIS
01286
01287
01288 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01289
01293 #define EMBER_INDEX_OUT_OF_RANGE(0xB1)
01294 #else
01295 DEFINE_ERROR(INDEX_OUT_OF_RANGE, 0xB1)
01296 #endif
01297
01298 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01299
01302 #define EMBER_TABLE_FULL(0xB4)
01303 #else
01304 DEFINE_ERROR(TABLE_FULL, 0xB4)
01305 #endif //DOXYGEN_SHOULD_SKIP_THIS
01306

```

```

01307 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01308
01312 #define EMBER_TABLE_ENTRY_ERASED(0xB6)
01313 #else
01314 DEFINE_ERROR(TABLE_ENTRY_ERASED, 0xB6)
01315 #endif
01316
01317 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01318
01322 #define EMBER_LIBRARY_NOT_PRESENT(0xB5)
01323 #else
01324 DEFINE_ERROR(LIBRARY_NOT_PRESENT, 0xB5)
01325 #endif
01326
01327 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01328
01332 #define EMBER_OPERATION_IN_PROGRESS(0xBA)
01333 #else
01334 DEFINE_ERROR(OPERATION_IN_PROGRESS, 0xBA)
01335 #endif
01336
01337 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01338
01343 #define EMBER_TRUST_CENTER_EUI_HAS_CHANGED(0xBC)
01344 #else
01345     DEFINE_ERROR(TRUST_CENTER_EUI_HAS_CHANGED, 0xBC)
01346 #endif
01347
01349
01354
01355 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01356
01360 #define EMBER_NO_RESPONSE(0xC0)
01361 #else
01362     DEFINE_ERROR(NO_RESPONSE, 0xC0)
01363 #endif
01364
01365 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01366
01370 #define EMBER_DUPLICATE_ENTRY(0xC1)
01371 #else
01372     DEFINE_ERROR(DUPLICATE_ENTRY, 0xC1)
01373 #endif
01374
01375 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01376
01381 #define EMBER_NOT_PERMITTED(0xC2)
01382 #else
01383     DEFINE_ERROR(NOT_PERMITTED, 0xC2)
01384 #endif
01385
01386 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01387
01390 #define EMBER_DISCOVERY_TIMEOUT(0xC3)
01391 #else
01392     DEFINE_ERROR(DISCOVERY_TIMEOUT, 0xC3)
01393 #endif
01394
01395
01396 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01397
01401 #define EMBER_DISCOVERY_ERROR(0xC4)
01402 #else
01403     DEFINE_ERROR(DISCOVERY_ERROR, 0xC4)
01404 #endif
01405
01406
01407 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01408
01412 #define EMBER_SECURITY_TIMEOUT(0xC5)
01413 #else
01414     DEFINE_ERROR(SEcurity_TIMEOUT, 0xC5)
01415 #endif
01416
01417
01418 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01419
01422 #define EMBER_SECURITY_FAILURE(0xC6)
01423 #else
01424     DEFINE_ERROR(SEcurity_FAILURE, 0xC6)

```



```

01425 #endif
01426
01428
01434
01435 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01436
01440 #define EMBER_APPLICATION_ERROR_0 (0xF0)
01441 #define EMBER_APPLICATION_ERROR_1 (0xF1)
01442 #define EMBER_APPLICATION_ERROR_2 (0xF2)
01443 #define EMBER_APPLICATION_ERROR_3 (0xF3)
01444 #define EMBER_APPLICATION_ERROR_4 (0xF4)
01445 #define EMBER_APPLICATION_ERROR_5 (0xF5)
01446 #define EMBER_APPLICATION_ERROR_6 (0xF6)
01447 #define EMBER_APPLICATION_ERROR_7 (0xF7)
01448 #define EMBER_APPLICATION_ERROR_8 (0xF8)
01449 #define EMBER_APPLICATION_ERROR_9 (0xF9)
01450 #define EMBER_APPLICATION_ERROR_10 (0xFA)
01451 #define EMBER_APPLICATION_ERROR_11 (0xFB)
01452 #define EMBER_APPLICATION_ERROR_12 (0xFC)
01453 #define EMBER_APPLICATION_ERROR_13 (0xFD)
01454 #define EMBER_APPLICATION_ERROR_14 (0xFE)
01455 #define EMBER_APPLICATION_ERROR_15 (0xFF)
01456 #else
01457 #define DEFINE_ERROR( APPLICATION_ERROR_0, 0xF0)
01458 #define DEFINE_ERROR( APPLICATION_ERROR_1, 0xF1)
01459 #define DEFINE_ERROR( APPLICATION_ERROR_2, 0xF2)
01460 #define DEFINE_ERROR( APPLICATION_ERROR_3, 0xF3)
01461 #define DEFINE_ERROR( APPLICATION_ERROR_4, 0xF4)
01462 #define DEFINE_ERROR( APPLICATION_ERROR_5, 0xF5)
01463 #define DEFINE_ERROR( APPLICATION_ERROR_6, 0xF6)
01464 #define DEFINE_ERROR( APPLICATION_ERROR_7, 0xF7)
01465 #define DEFINE_ERROR( APPLICATION_ERROR_8, 0xF8)
01466 #define DEFINE_ERROR( APPLICATION_ERROR_9, 0xF9)
01467 #define DEFINE_ERROR( APPLICATION_ERROR_10, 0xFA)
01468 #define DEFINE_ERROR( APPLICATION_ERROR_11, 0xFB)
01469 #define DEFINE_ERROR( APPLICATION_ERROR_12, 0xFC)
01470 #define DEFINE_ERROR( APPLICATION_ERROR_13, 0xFD)
01471 #define DEFINE_ERROR( APPLICATION_ERROR_14, 0xFE)
01472 #define DEFINE_ERROR( APPLICATION_ERROR_15, 0xFF)
01473 #endif //DOXYGEN_SHOULD_SKIP_THIS
01474
01476

```

8.33 error.h File Reference

Macros

- `#define` [__EMBERSTATUS_TYPE__](#)
- `#define` [DEFINE_ERROR](#)(symbol, value)

Typedefs

- `typedef uint8_t` [EmberStatus](#)

Enumerations

- `enum` { [EMBER_ERROR_CODE_COUNT](#) }

8.33.1 Detailed Description

Return codes for Ember API functions and module definitions. See [Ember Status Codes](#) for documentation.

Definition in file [error.h](#).

8.33.2 Macro Definition Documentation

8.33.2.1 #define __EMBERSTATUS_TYPE__

Return type for Ember functions.

Definition at line 18 of file [error.h](#).

8.33.3 Typedef Documentation

8.33.3.1 typedef uint8_t EmberStatus

Definition at line 19 of file [error.h](#).

8.34 error.h

```

00001
00011 #ifndef __ERRORS_H__
00012 #define __ERRORS_H__
00013
00017 #ifndef __EMBERSTATUS_TYPE__
00018 #define __EMBERSTATUS_TYPE__
00019     typedef uint8_t EmberStatus;
00020 #endif //__EMBERSTATUS_TYPE__
00021
00035 #define DEFINE_ERROR(symbol, value) \
00036     EMBER_ ## symbol = value,
00037
00038
00039 enum {
00040     #ifndef DOXYGEN_SHOULD_SKIP_THIS
00041     #include "include/error-def.h"
00042     #endif //DOXYGEN_SHOULD_SKIP_THIS
00043     EMBER_ERROR_CODE_COUNT
00046
00047
00048 };
00049
00050 #undef DEFINE_ERROR
00051
00052 #endif // __ERRORS_H__
00053

```

8.35 ezsp-host-configuration-defaults.h File Reference

Macros

- #define [EZSP_HOST_SOURCE_ROUTE_TABLE_SIZE](#)
- #define [EZSP_HOST_RX_POOL_SIZE](#)
- #define [EZSP_HOST_FORM_AND_JOIN_BUFFER_SIZE](#)

8.35.1 Detailed Description

User-configurable parameters for host applications. The default values set in this file can be overridden by putting #defines into the host application's CONFIGURATION_HEADER.

See [Configuration](#) for documentation.

Definition in file [ezsp-host-configuration-defaults.h](#).

8.36 ezsp-host-configuration-defaults.h

```

00001
00019 #ifndef CONFIGURATION_HEADER
00020 #include CONFIGURATION_HEADER
00021 #endif
00022
00023 #ifndef EZSP_HOST_SOURCE_ROUTE_TABLE_SIZE
00024
00032 #define EZSP_HOST_SOURCE_ROUTE_TABLE_SIZE 32
00033 #endif
00034
00035 #ifndef EZSP_HOST_RX_POOL_SIZE
00036
00043 #define EZSP_HOST_RX_POOL_SIZE 20
00044 #endif
00045
00046 #ifndef EZSP_HOST_FORM_AND_JOIN_BUFFER_SIZE
00047
00055 #define EZSP_HOST_FORM_AND_JOIN_BUFFER_SIZE 40
00056 #endif
00057

```

8.37 form-and-join.h File Reference

Macros

- #define [NETWORK_STORAGE_SIZE](#)
- #define [NETWORK_STORAGE_SIZE_SHIFT](#)
- #define [FORM_AND_JOIN_MAX_NETWORKS](#)

Functions

- [EmberStatus emberScanForUnusedPanId](#) (uint32_t channelMask, uint8_t duration)
- [EmberStatus emberScanForJoinableNetwork](#) (uint32_t channelMask, uint8_t *extended-PanId)
- [EmberStatus emberScanForNextJoinableNetwork](#) (void)
- bool [emberFormAndJoinIsScanning](#) (void)
- bool [emberFormAndJoinCanContinueJoinableNetworkScan](#) (void)
- void [emberUnusedPanIdFoundHandler](#) (EmberPanId panId, uint8_t channel)
- void [emberJoinableNetworkFoundHandler](#) (EmberZigbeeNetwork *networkFound, uint8_t lqi, int8_t rssi)
- void [emberScanErrorHandler](#) (EmberStatus status)
- bool [emberFormAndJoinScanCompleteHandler](#) (uint8_t channel, EmberStatus status)
- bool [emberFormAndJoinNetworkFoundHandler](#) (EmberZigbeeNetwork *networkFound, uint8_t lqi, int8_t rssi)
- bool [emberFormAndJoinEnergyScanResultHandler](#) (uint8_t channel, int8_t maxRssi-Value)
- void [emberFormAndJoinTick](#) (void)
- void [emberFormAndJoinTaskInit](#) (void)
- void [emberFormAndJoinRunTask](#) (void)
- void [emberFormAndJoinCleanup](#) (EmberStatus status)

Variables

- bool [emberEnableDualChannelScan](#)

8.37.1 Detailed Description

Utilities for forming and joining networks. See [Forming and Joining Networks](#) for documentation.

Definition in file [form-and-join.h](#).

8.38 form-and-join.h

```

00001
00068 #define NETWORK_STORAGE_SIZE 16
00069
00072 #define NETWORK_STORAGE_SIZE_SHIFT 4
00073
00087 #ifndef FORM_AND_JOIN_MAX_NETWORKS
00088     #ifndef EZSP_HOST
00089         // the host's buffer is 16-bit array, so translate to bytes for comparison
00090         #define FORM_AND_JOIN_MAX_NETWORKS \
00091             (EZSP_HOST_FORM_AND_JOIN_BUFFER_SIZE * 2 / NETWORK_STORAGE_SIZE)
00092     #else
00093         // use highest value that won't exceed max EmberMessageBuffer length
00094         #define FORM_AND_JOIN_MAX_NETWORKS 15
00095     #endif
00096 #endif
00097
00098 // Check that this value isn't too large for the SoC implementation to handle
00099 #ifndef EZSP_HOST
00100     #if (FORM_AND_JOIN_MAX_NETWORKS > 15)
00101         #error "FORM_AND_JOIN_MAX_NETWORKS can't exceed 15 on SoC platform"
00102     #endif
00103 #endif
00104
00121 EmberStatus emberScanForUnusedPanId(uint32_t
channelMask, uint8_t duration);
00122
00149 EmberStatus emberScanForJoinableNetwork(
uint32_t channelMask, uint8_t* extendedPanId);
00150
00152 EmberStatus emberScanForNextJoinableNetwork
(void);
00153
00169 extern bool emberEnableDualChannelScan;
00170
00175 bool emberFormAndJoinIsScanning(void);
00176
00181 bool emberFormAndJoinCanContinueJoinableNetworkScan
(void);
00182
00183 //
-----
00184 // Callbacks the application needs to implement.
00185
00194 void emberUnusedPanIdFoundHandler(EmberPanId
panId, uint8_t channel);
00195
00206 void emberJoinableNetworkFoundHandler(
EmberZigbeeNetwork *networkFound,
00207                                     uint8_t lqi,
00208                                     int8_t rssi);
00209
00227 void emberScanErrorHandler(EmberStatus status);
00228
00229 //
-----
00230 // Library functions the application must call from within the
00231 // corresponding EmberZNet or EZSP callback.
00232
00240 bool emberFormAndJoinScanCompleteHandler(
uint8_t channel, EmberStatus status);
00241
00249 bool emberFormAndJoinNetworkFoundHandler(
EmberZigbeeNetwork *networkFound,
00250                                     uint8_t lqi,
00251                                     int8_t rssi);

```

```

00252
00260 bool emberFormAndJoinEnergyScanResultHandler
      (uint8_t channel, int8_t maxRssiValue);
00261
00266 void emberFormAndJoinTick(void);
00267
00271 void emberFormAndJoinTaskInit(void);
00272
00276 void emberFormAndJoinRunTask(void);
00277
00282 void emberFormAndJoinCleanup(EmberStatus
      status);
00283
00284
00285

```

8.39 fragment-host.h File Reference

Initialization

- void [ezspFragmentInit](#) (uint16_t receiveBufferLength, uint8_t *receiveBuffer)

Transmitting

- [EmberStatus ezspFragmentSendUnicast](#) ([EmberOutgoingMessageType](#) type, uint16_t indexOrDestination, [EmberApsFrame](#) *apsFrame, uint8_t maxFragmentSize, uint16_t messageLength, uint8_t *messageContents)
- [EmberStatus ezspFragmentSourceRouteHandler](#) (void)
- bool [ezspFragmentMessageSent](#) ([EmberApsFrame](#) *apsFrame, [EmberStatus](#) status)
- void [ezspFragmentMessageSentHandler](#) ([EmberStatus](#) status)

Receiving

- bool [ezspFragmentIncomingMessage](#) ([EmberApsFrame](#) *apsFrame, [EmberNodeId](#) sender, uint16_t *messageLength, uint8_t **messageContents)
- void [ezspFragmentTick](#) (void)

8.39.1 Detailed Description

Fragmented message support for EZSP Hosts. Splits long messages into smaller blocks for transmission and reassembles received blocks. See [Message Fragmentation](#) for documentation.

Deprecated The fragment library is deprecated and will be removed in a future release. Similar functionality is available in the Fragmentation plugin in Application Framework.

Definition in file [fragment-host.h](#).

8.40 fragment-host.h

```

00001
00059 void ezspFragmentInit(uint16_t receiveBufferLength, uint8_t *

```

```

        receiveBuffer);
00060
00095 EmberStatus ezspFragmentSendUnicast (
    EmberOutgoingMessageType type,
00096                                     uint16_t indexOrDestination,
00097                                     EmberApsFrame *apsFrame,
00098                                     uint8_t maxFragmentSize,
00099                                     uint16_t messageLength,
00100                                     uint8_t *messageContents);
00101
00114 EmberStatus ezspFragmentSourceRouteHandler
    (void);
00115
00130 bool ezspFragmentMessageSent (EmberApsFrame
    *apsFrame, EmberStatus status);
00131
00140 void ezspFragmentMessageSentHandler (EmberStatus
    status);
00141
00173 bool ezspFragmentIncomingMessage (EmberApsFrame
    *apsFrame,
00174                                     EmberNodeId sender,
00175                                     uint16_t *messageLength,
00176                                     uint8_t **messageContents);
00177
00182 void ezspFragmentTick(void);
00183

```

8.41 hal.h File Reference

```

#include "host/button-common.h"
#include "host/crc.h"
#include "host/led-common.h"
#include "host/micro-common.h"
#include "host/serial.h"
#include "host/system-timer.h"
#include "host/bootloader-EEPROM.h"
#include "adc.h"
#include "buzzer.h"

```

Macros

- #define `emAmHost()`

8.41.1 Detailed Description

Generic set of HAL includes for all platforms. See also [Hardware Abstraction Layer \(HAL\) API Reference](#) for more documentation.

Some HAL includes are not used or present in builds intended for the Host processor connected to the Ember Network Coprocessor.

Definition in file [hal.h](#).

8.41.2 Macro Definition Documentation

8.41.2.1 #define emAmHost()

Definition at line 250 of file [hal.h](#).

8.42 hal.h

```

00001
00051 #ifndef __HAL_H__
00052 #define __HAL_H__
00053
00054 #ifdef HAL_HOST
00055
00056 #include "host/button-common.h"
00057 #include "host/crc.h"
00058 #include "host/led-common.h"
00059 #include "host/micro-common.h"
00060 #include "host/serial.h"
00061 #include "host/system-timer.h"
00062 #include "host/bootloader-eeprom.h"
00063 //Pull in the micro specific ADC, buzzer, and clocks headers. The
00064 //specific header is chosen by the build include path pointing at
00065 //the appropriate directory.
00066 #include "adc.h"
00067 #include "buzzer.h"
00068
00069 #else //HAL_MICRO
00070
00071 // Keep micro and board first for specifics used by other headers
00072 #include "micro/micro.h"
00073 #include "micro/antenna.h"
00074 #if !defined(STACK) && defined(BOARD_HEADER)
00075 #include BOARD_HEADER
00076 #endif
00077
00078 #if (defined(EMBER_STACK_CONNECT))
00079 #if (defined(UNIX_HOST) && !defined(EMBER_TEST))
00080 #include "micro/adc.h"
00081 #include "micro/button.h"
00082 #include "micro/buzzer.h"
00083 #include "micro/crc.h"
00084 #include "micro/endian.h"
00085 #include "micro/led.h"
00086 #include "micro/random.h"
00087 #include "micro/serial.h"
00088 #include "micro/spi.h"
00089 #include "micro/system-timer.h"
00090 #else
00091 // TODO: here we include only the functionalities that we will have on
mustang
00092 #if (defined(CORTEXM3))
00093 #include "micro/adc.h"
00094 #include "micro/bootloader-eeprom.h"
00095 #if ((defined _EFR_DEVICE) || (defined CORTEXM3_EMBER_MICRO))
00096 #include "micro/bootloader-interface.h"
00097 #endif
00098 #include "micro/button.h"
00099 #include "micro/led.h"
00100 #include "micro/buzzer.h"
00101 #include "micro/crc.h"
00102 #include "micro/diagnostic.h"
00103 #include "micro/endian.h"
00104 #endif //CORTEXM3
00105 #include "micro/flash.h"
00106 #include "micro/sim-eeprom.h"
00107 #include "micro/system-timer.h"
00108 #include "micro/symbol-timer.h"
00109 #include "micro/spi.h"
00110 #if (defined(CORTEXM3) || defined(EMBER_TEST))
00111 #include "micro/serial.h"
00112 #else
00113 #include "micro/serial-minimal.h"
00114 #endif
00115 #include "micro/random.h"
00116 #include "micro/token.h"
00117 #ifdef EMBER_TEST

```

```

00118     #include "micro/adc.h"
00119     #include "micro/bootloader-interface.h"
00120     #include "micro/button.h"
00121     #include "micro/led.h"
00122     #endif
00123 #endif // UNIX_HOST && !EMBER_TEST
00124 #elif (defined(EMBER_STACK_OWL_RX))
00125 // TODO: here we include only the functionalities that we will have on OWL-RX
00126 #include "micro/button.h"
00127 #include "micro/flash.h"
00128 #include "micro/led.h"
00129 #include "micro/dog_glcd.h"
00130 #include "micro/system-timer.h"
00131 #include "micro/symbol-timer.h"
00132 #include "micro/spi.h"
00133 #include "micro/serial-minimal.h"
00134 #include "micro/endian.h"
00135 // #include "micro/random.h"
00136 // #include "micro/token.h"
00137 // #ifdef EMBER_TEST
00138 //     #include "micro/adc.h"
00139 //     #include "micro/bootloader-interface.h"
00140 //     #include "micro/button.h"
00141 //     #include "micro/led.h"
00142 // #endif
00143 #elif (defined(EMBER_STACK_OWL_TX))
00144 // TODO: here we include only the functionalities that we will have on OWL-TX
00145 // #include "micro/button.h"
00146 // #include "micro/flash.h"
00147 // #include "micro/led.h"
00148 // #include "micro/dog_glcd.h"
00149 // #include "micro/system-timer.h"
00150 // #include "micro/symbol-timer.h"
00151 // #include "micro/spi.h"
00152 // #include "micro/serial-minimal.h"
00153 // #include "micro/random.h"
00154 // #include "micro/token.h"
00155 // #ifdef EMBER_TEST
00156 //     #include "micro/adc.h"
00157 //     #include "micro/bootloader-interface.h"
00158 //     #include "micro/button.h"
00159 //     #include "micro/led.h"
00160 // #endif
00161 #elif (defined(EMBER_STACK_WASP))
00162 // TODO: here we include only the functionalities that we will have on
// mustang
00163 #if (defined(CORTEXM3))
00164 //     #include "micro/adc.h"
00165 //     #include "micro/bootloader-eeprom.h"
00166 //     #include "micro/button.h"
00167 //     #include "micro/buzzer.h"
00168 //     #include "micro/led.h"
00169 //     #include "micro/diagnostic.h"
00170 #endif //CORTEXM3
00171 #include "micro/flash.h"
00172 #include "micro/system-timer.h"
00173 #include "micro/symbol-timer.h"
00174 #include "micro/spi.h"
00175 #if (defined(CORTEXM3))
00176     #include "micro/serial.h"
00177 #else
00178     #include "micro/serial-minimal.h"
00179 #endif
00180 #include "micro/random.h"
00181 #include "micro/token.h"
00182 #ifdef EMBER_TEST
00183     #include "micro/adc.h"
00184     #include "micro/bootloader-interface.h"
00185     #include "micro/button.h"
00186     #include "micro/led.h"
00187 #endif
00188 #elif (! defined(EMBER_STACK_IP))
00189 // Pro Stack
00190 #include "micro/adc.h"
00191 #include "micro/button.h"
00192 #include "micro/buzzer.h"
00193 #include "micro/crc.h"
00194 #include "micro/endian.h"
00195 #include "micro/led.h"
00196 #include "micro/random.h"

```



```

00197 #include "micro/serial.h"
00198 #include "micro/spi.h"
00199 #include "micro/system-timer.h"
00200 #include "micro/bootloader-eeeprom.h"
00201
00202 //Host processors do not use the following modules, therefore the header
00203 //files should be ignored.
00204 #ifndef EZSP_HOST
00205     #include "micro/bootloader-interface.h"
00206     #include "micro/diagnostic.h"
00207     #include "micro/token.h"
00208     //No public HAL code in release 4.0 uses the symbol timer,
00209     //therefore it should not be in doxygen.
00210     #ifndef DOXYGEN_SHOULD_SKIP_THIS
00211         #include "micro/symbol-timer.h"
00212     #endif // DOXYGEN_SHOULD_SKIP_THIS
00213 #endif //EZSP_HOST
00214
00215 #else
00216     // IP Stack
00217     #include "micro/adc.h"
00218     #include "micro/button.h"
00219     #include "micro/buzzer.h"
00220     #include "micro/crc.h"
00221     #include "micro/endian.h"
00222     #include "micro/led.h"
00223     #include "micro/random.h"
00224     #include "micro/serial.h"
00225     #include "micro/spi.h"
00226     #include "micro/system-timer.h"
00227     //Host processors do not use the following modules, therefore the header
00228     //files should be ignored.
00229     #ifndef UNIX_HOST
00230         #include "micro/bootloader-interface.h"
00231         #include "micro/diagnostic.h"
00232         #include "micro/token.h"
00233         //No public HAL code in release 4.0 uses the symbol timer,
00234         //therefore it should not be in doxygen.
00235         #ifndef DOXYGEN_SHOULD_SKIP_THIS
00236             #include "micro/symbol-timer.h"
00237         #endif // DOXYGEN_SHOULD_SKIP_THIS
00238     #endif //UNIX_HOST
00239
00240 #endif // !EMBER_STACK_IP
00241
00242 #endif // !HAL_HOST
00243
00244 #if ((defined(RTOS) && !defined(IP_MODEM_LIBRARY)) \
00245     || (defined(UNIX_HOST) \
00246         || defined(UNIX_HOST_SIM))) \
00247     #define EMBER_HOST
00248     #define emAmHost() true
00249 #else
00250     #define emAmHost() false
00251 #endif
00252
00253 #endif //__HAL_H__
00254

```

8.43 iar-st.h File Reference

```

#include "stm32f10x.h"
#include <stdarg.h>
#include <stdint.h>
#include <stdbool.h>
#include <string.h>
#include "hal/host/generic/compiler/platform-common.h"

```

Macros

- #define [halResetWatchdog\(\)](#)
- #define [SIGNED_ENUM](#)
- #define [_HAL_USE_COMMON_DIVMOD_](#)
- #define [_HAL_USE_COMMON_PGM_](#)
- #define [PLATCOMMONOKTOINCLUDE](#)

Functions

- void [halInternalResetWatchDog](#) (void)

Master Variable Types

These are a set of typedefs to make the size of all variable declarations explicitly known. Since the IAR host code links against the ST Standard peripheral library, we need to map Ember's variable types to ST's variable types.

Note

ST uses IAR's variable types, found in stdint.h.

- typedef bool [boolean](#)
- typedef uint8_t [int8u](#)
- typedef int8_t [int8s](#)
- typedef uint16_t [int16u](#)
- typedef int16_t [int16s](#)
- typedef uint32_t [int32u](#)
- typedef int32_t [int32s](#)
- typedef uint32_t [PointerType](#)

Miscellaneous Macros

- #define [simulatedSerialTimePasses\(\)](#)
- #define [BIGENDIAN_CPU](#)
- #define [MAIN_FUNCTION_PARAMETERS](#)
- #define [MAIN_FUNCTION_ARGUMENTS](#)
- #define [__SOURCEFILE__](#)
- #define [assert](#)(condition)
- #define [simulatedTimePasses\(\)](#)
- #define [simulatedTimePassesMs](#)(x)
- #define [simulatedSerialTimePasses\(\)](#)
- #define [STRINGIZE](#)(X)
- #define [ALIGNMENT](#)(X)
- void [halInternalAssertFailed](#) (const char *filename, int linenumber)

Global Interrupt Manipulation Macros

- `#define DISABLE_INTERRUPTS()`
- `#define RESTORE_INTERRUPTS()`
- `#define INTERRUPTS_ON()`
- `#define INTERRUPTS_OFF()`
- `#define INTERRUPTS_ARE_OFF()`
- `#define INTERRUPTS_WERE_ON()`
- `#define ATOMIC(blah)`
- `#define HANDLE_PENDING_INTERRUPTS()`

Generic Types

- `#define NULL`

8.43.1 Detailed Description

See [Common PLATFORM_HEADER Configuration](#) and [STM32F103RET IAR Specific PLATFORM_HEADER Configuration](#) for documentation.

Definition in file [iar-st.h](#).

8.44 iar-st.h

```

00001
00021 #ifndef __IAR_ST_H__
00022 #define __IAR_ST_H__
00023
00024 #ifndef __ICCARM__
00025     #error Improper PLATFORM_HEADER
00026 #endif
00027
00028 #if (__VER__ < 6040002)
00029     #error Only IAR EWARM versions greater than 6.40.2 are supported
00030 #endif // __VER__
00031
00032 //Pull in the registers, Library, and other critical/useful ST code.
00033 #include "stm32f10x.h"
00034 #include <stdarg.h>
00035 #include <stdint.h>
00036 #include <stdbool.h>
00037
00038 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00039 //The Cortex-M3 does not have zero-page memory
00040 #define XAP2B_PAGEZERO_ON
00041 #define XAP2B_PAGEZERO_OFF
00042 #endif
00043
00055 typedef bool boolean; /*To ease adoption of bool instead of boolean.*/
00056 typedef uint8_t int8u;
00057 typedef int8_t int8s;
00058 typedef uint16_t int16u;
00059 typedef int16_t int16s;
00060 typedef uint32_t int32u;
00061 typedef int32_t int32s;
00062 typedef uint32_t PointerType;
00064
00065
00071 void halInternalResetWatchDog(void);
00072
00078 #define halResetWatchdog() halInternalResetWatchDog()
00079
00080
00084 #define SIGNED_ENUM

```

```

00085
00089 #define simulatedSerialTimePasses()
00090
00091
00095 #define _HAL_USE_COMMON_DIVMOD_
00096
00097
00101 #define _HAL_USE_COMMON_PGM_
00102
00103
00105
00107
00108
00109
00114 #define BIGENDIAN_CPU false
00115
00120 #define MAIN_FUNCTION_PARAMETERS void
00121 #define MAIN_FUNCTION_ARGUMENTS
00122
00123
00124 #ifndef __SOURCEFILE__
00125
00130 #define __SOURCEFILE__ __FILE__
00131 #endif
00132
00133
00134 #undef assert
00135 #if !defined(SIMPLER_ASSERT_REBOOT) || defined(DOXYGEN_SHOULD_SKIP_THIS)
00136
00139 void halInternalAssertFailed(const char * filename,
int linenumber);
00140
00146 #define assert(condition) \
00147     do { \
00148         if (! (condition)) { \
00149             halInternalAssertFailed(__SOURCEFILE__, __LINE__); \
00150         } \
00151     } while(0)
00152 #else
00153 #define assert(condition) \
00154     do { if( !(condition) ) while(1){} } while(0)
00155 #endif
00156
00157
00161 #define simulatedTimePasses()
00162
00165 #define simulatedTimePassesMs(x)
00166
00169 #define simulatedSerialTimePasses()
00170
00174 #define STRINGIZE(X) #X
00175
00179 #define ALIGNMENT(X) \
00180     _Pragma( STRINGIZE( data_alignment=##X## ) )
00181
00183
00184
00185
00186
00188
00190
00191
00192
00193 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00194     //The concept of LITE atomic handling isn't implemented on this platform,
00195     //so just redirect to the normal atomic handling.
00196     #define ATOMIC_LITE(blah) ATOMIC(blah)
00197     #define DECLARE_INTERRUPT_STATE_LITE DECLARE_INTERRUPT_STATE
00198     #define DISABLE_INTERRUPTS_LITE() DISABLE_INTERRUPTS()
00199     #define RESTORE_INTERRUPTS_LITE() RESTORE_INTERRUPTS()
00200
00206     #define DECLARE_INTERRUPT_STATE uint32_t _emIsrState
00207 #endif // DOXYGEN_SHOULD_SKIP_THIS
00208
00209 //The core Global Interrupt Manipulation Macros start here.
00210
00217 #define DISABLE_INTERRUPTS() \
00218     do { \
00219         _emIsrState = __get_PRIMASK(); \
00220         __set_PRIMASK(1); \

```

```

00221     } while(0)
00222
00223
00231 #define RESTORE_INTERRUPTS() \
00232     do { \
00233         __set_PRIMASK(_emIsrState); \
00234     } while(0)
00235
00236
00241 #define INTERRUPTS_ON() \
00242     do { \
00243         __set_PRIMASK(0); \
00244     } while(0)
00245
00246
00251 #define INTERRUPTS_OFF() \
00252     do { \
00253         __set_PRIMASK(1); \
00254     } while(0)
00255
00256
00260 #define INTERRUPTS_ARE_OFF() (__get_PRIMASK() != 0)
00261
00262
00267 #define INTERRUPTS_WERE_ON() (_emIsrState == 0)
00268
00269
00274 #define ATOMIC(blah) \
00275 { \
00276     DECLARE_INTERRUPT_STATE; \
00277     DISABLE_INTERRUPTS(); \
00278     { blah } \
00279     RESTORE_INTERRUPTS(); \
00280 }
00281
00282
00290 #define HANDLE_PENDING_INTERRUPTS() \
00291     do { \
00292         if (INTERRUPTS_ARE_OFF()) { \
00293             INTERRUPTS_ON(); \
00294             INTERRUPTS_OFF(); \
00295         } \
00296     } while (0)
00297
00299
00300
00301
00302
00307 //true and FLASE are defined in ST's HAL Library
00308
00309 #ifndef NULL
00310
00313 #define NULL ((void *)0)
00314 #endif
00315
00317
00318
00323 #include <string.h>
00324
00325
00329 #define PLATCOMMONOKTOINCLUDE
00330     #include "hal/host/generic/compiler/platform-common.h"
00331
00331 #undef PLATCOMMONOKTOINCLUDE
00332
00333 #endif // __IAR_ST_H__
00334

```

8.45 led-common.h File Reference

```
#include "led-specific.h"
```

Typedefs

- typedef enum [HalBoardLedPins](#) [HalBoardLed](#)

Functions

- void [halInternalInitLed](#) (void)
- void [halToggleLed](#) ([HalBoardLed](#) led)
- void [halSetLed](#) ([HalBoardLed](#) led)
- void [halClearLed](#) ([HalBoardLed](#) led)

8.45.1 Detailed Description

See [LED Control](#) and micro specific modules for documentation.

Definition in file [led-common.h](#).

8.46 led-common.h

```

00001
00020 #ifndef __LED_COMMON_H__
00021 #define __LED_COMMON_H__
00022
00023
00026 void halInternalInitLed(void);
00027
00028
00037 typedef enum HalBoardLedPins HalBoardLed;
00038
00039
00045 void halToggleLed(HalBoardLed led);
00046
00047
00053 void halSetLed(HalBoardLed led);
00054
00055
00061 void halClearLed(HalBoardLed led);
00062
00063
00064 //Pull in the micro specific LED definitions. The specific header is chosen
00065 //by the build include path pointing at the appropriate directory.
00066 #include "led-specific.h"
00067
00068
00069 #endif //__LED_COMMON_H__
00070

```

8.47 led-specific.h File Reference

Macros

- #define [BOARDLED0_PIN](#)
- #define [BOARDLED0_PORT](#)
- #define [BOARDLED1_PIN](#)
- #define [BOARDLED1_PORT](#)

Enumerations

- enum [HalBoardLedPins](#) { [BOARDLED0](#), [BOARDLED1](#), [BOARD_ACTIVITY_LED](#), [BOARD_HEARTBEAT_LED](#) }

8.47.1 Detailed Description

See [LED Control](#) and [STM32F103RET Specific LED](#) for documentation.

Definition in file [led-specific.h](#).

8.48 led-specific.h

```

00001
00019 #ifndef __LED_SPECIFIC_H__
00020 #define __LED_SPECIFIC_H__
00021
00022
00028 enum HalBoardLedPins {
00029     BOARDLED0 = 0, //Just a simple identifier for switch statements
00030     BOARDLED1 = 1, //Just a simple identifier for switch statements
00031     BOARD_ACTIVITY_LED = BOARDLED0,
00032     BOARD_HEARTBEAT_LED = BOARDLED1
00033 };
00034
00035
00039 #define BOARDLED0_PIN    GPIO_Pin_8
00040
00044 #define BOARDLED0_PORT    GPIOB
00045
00046
00050 #define BOARDLED1_PIN    GPIO_Pin_9
00051
00055 #define BOARDLED1_PORT    GPIOB
00056
00057
00058 #endif //__LED_SPECIFIC_H__
00059

```

8.49 micro-common.h File Reference

```
#include "micro-specific.h"
```

Macros

- #define [MICRO_DISABLE_WATCH_DOG_KEY](#)
- #define [MILLISECOND_TICKS_PER_SECOND](#)
- #define [MILLISECOND_TICKS_PER_DECISECOND](#)
- #define [MILLISECOND_TICKS_PER_QUARTERSECOND](#)
- #define [MILLISECOND_TICKS_PER_MINUTE](#)
- #define [MILLISECOND_TICKS_PER_HOUR](#)
- #define [MILLISECOND_TICKS_PER_DAY](#)

Enumerations

- enum [SleepModes](#) {
[SLEEPMODE_RUNNING](#), [SLEEPMODE_IDLE](#), [SLEEPMODE_WAKETIMER](#),
[SLEEPMODE_MAINTAINTIMER](#),
[SLEEPMODE_NOTIMER](#), [SLEEPMODE_RESERVED](#), [SLEEPMODE_POWERDOWN](#), [SLEEPMODE_POWERSAVE](#) }

Functions

- void [halInit](#) (void)
- void [halReboot](#) (void)
- void [halPowerUp](#) (void)
- void [halPowerDown](#) (void)
- void [halInternalEnableWatchDog](#) (void)
- void [halInternalDisableWatchDog](#) (uint8_t magicKey)
- void [halCommonDelayMicroseconds](#) (uint16_t us)
- void [halCommonDelayMilliseconds](#) (uint16_t ms)
- uint8_t [halGetResetInfo](#) (void)
- PGM_P [halGetResetString](#) (void)
- void [halStackSeedRandom](#) (uint32_t seed)
- uint16_t [halCommonGetRandom](#) (void)
- void [halSleep](#) ([SleepModes](#) sleepMode)

8.49.1 Detailed Description

See [Microcontroller General Functionality](#) and micro specific modules for documentation.

Definition in file [micro-common.h](#).

8.50 micro-common.h

```
00001
00017 #ifndef __MICRO_COMMON_H__
00018 #define __MICRO_COMMON_H__
00019
00020
00023 void halInit (void);
00024
00027 void halReboot (void);
00028
00031 void halPowerUp (void);
00032
00035 void halPowerDown (void);
00036
00041 #define MICRO_DISABLE_WATCH_DOG_KEY 0xA5
00042
00046 void halInternalEnableWatchDog (void);
00047
00057 void halInternalDisableWatchDog (uint8_t magicKey);
00058
00073 void halCommonDelayMicroseconds (uint16_t us);
00074
00082 void halCommonDelayMilliseconds (uint16_t ms);
00083
00088 uint8_t halGetResetInfo (void);
00089
00094 PGM_P halGetResetString (void);
00095
```



```

00101 void halStackSeedRandom(uint32_t seed);
00102
00105 uint16_t halCommonGetRandom(void);
00106
00107
00108 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00109
00145 enum SleepModes
00146 #else
00147 typedef uint8_t SleepModes;
00148 enum
00149 #endif
00150 {
00151     SLEEPMODE_RUNNING = 0,
00152     SLEEPMODE_IDLE = 1,
00153     SLEEPMODE_WAKETIMER = 2,
00154     SLEEPMODE_MAINTAINTIMER = 3,
00155     SLEEPMODE_NOTIMER = 4,
00156
00157     //The following SleepModes are deprecated. Each micro's halSleep()
00158     //function will remap these modes to the appropriate replacement, as
00159     //necessary.
00160     SLEEPMODE_RESERVED = 6,
00161     SLEEPMODE_POWERDOWN = 7,
00162     SLEEPMODE_POWERSAVE = 8,
00163 };
00164
00171 void halSleep(SleepModes sleepMode);
00172
00173
00174 //Pull in the micro specific micro definitions. The specific header is chosen
00175 //by the build include path pointing at the appropriate directory.
00176 #include "micro-specific.h"
00177
00178 // Since the STM32 host doesn't have a micro.h add these defines here.
00179 // the number of ticks (as returned from halCommonGetInt32uMillisecondTick)
00180 // that represent an actual second. This can vary on different platforms.
00181 // It must be defined by the host system.
00182 #ifndef MILLISECOND_TICKS_PER_SECOND
00183     #define MILLISECOND_TICKS_PER_SECOND 1024UL
00184 // See bug 10232
00185 // #error "MILLISECOND_TICKS_PER_SECOND is not defined in micro.h!"
00186 #endif
00187
00188 #ifndef MILLISECOND_TICKS_PER_DECISECOND
00189     #define MILLISECOND_TICKS_PER_DECISECOND (MILLISECOND_TICKS_PER_SECOND / 10)
00190 #endif
00191
00192 #ifndef MILLISECOND_TICKS_PER_QUARTERSECOND
00193     #define MILLISECOND_TICKS_PER_QUARTERSECOND (MILLISECOND_TICKS_PER_SECOND >>
2)
00194 #endif
00195
00196 #ifndef MILLISECOND_TICKS_PER_MINUTE
00197     #define MILLISECOND_TICKS_PER_MINUTE (60UL * MILLISECOND_TICKS_PER_SECOND)
00198 #endif
00199
00200 #ifndef MILLISECOND_TICKS_PER_HOUR
00201     #define MILLISECOND_TICKS_PER_HOUR (60UL * MILLISECOND_TICKS_PER_MINUTE)
00202 #endif
00203
00204 #ifndef MILLISECOND_TICKS_PER_DAY
00205     #define MILLISECOND_TICKS_PER_DAY (24UL * MILLISECOND_TICKS_PER_HOUR)
00206 #endif
00207
00208 #endif //__MICRO_COMMON_H__
00209
00210

```

8.51 micro-specific.h File Reference

Macros

- #define [MILLISECOND_TICKS_PER_SECOND](#)

Functions

- void [halInternalInitSysTick](#) (void)
- #define [RESET_UNKNOWN](#)
- #define [RESET_LOW_POWER](#)
- #define [RESET_WINDOW_WATCHDOG](#)
- #define [RESET_INDEPENDENT_WATCHDOG](#)
- #define [RESET_SOFTWARE](#)
- #define [RESET_POR_PDR](#)
- #define [RESET_PIN](#)
- #define [RESET_UNSET](#)

8.51.1 Detailed Description

[Microcontroller General Functionality](#) and [STM32F103RET General Functionality](#) for documentation.

Definition in file [micro-specific.h](#).

8.52 micro-specific.h

```

00001
00019 #ifndef __MICRO_SPECIFIC_H__
00020 #define __MICRO_SPECIFIC_H__
00021
00022
00027 #define MILLISECOND_TICKS_PER_SECOND 1024UL
00028
00029
00030
00035 #define RESET_UNKNOWN          0
00036 #define RESET_LOW_POWER        1
00037 #define RESET_WINDOW_WATCHDOG  2
00038 #define RESET_INDEPENDENT_WATCHDOG 3
00039 #define RESET_SOFTWARE         4
00040 #define RESET_POR_PDR          5
00041 #define RESET_PIN              6
00042 #define RESET_UNSET            255
00043
00049 void halInternalInitSysTick(void);
00050
00051
00052 #endif //__MICRO_SPECIFIC_H__
00053

```

8.53 network-manager.h File Reference

```
#include <CONFIGURATION_HEADER>
```

Macros

- #define [NM_WARNING_LIMIT](#)
- #define [NM_WINDOW_SIZE](#)
- #define [NM_CHANNEL_MASK](#)
- #define [NM_WATCHLIST_SIZE](#)

Functions

- void [nmUtilWarningHandler](#) (void)
- bool [nmUtilProcessIncoming](#) ([EmberApsFrame](#) *apsFrame, uint8_t messageLength, uint8_t *message)
- [EmberStatus](#) [nmUtilChangeChannelRequest](#) (void)

8.53.1 Detailed Description

Utilities for use by the ZigBee network manager. See [Network Manager](#) for documentation.

Definition in file [network-manager.h](#).

8.54 network-manager.h

```

00001
00090 #include CONFIGURATION_HEADER
00091
00092 // The application is notified via nmUtilWarningHandler
00093 // if NM_WARNING_LIMIT unsolicited scan reports are received
00094 // within NM_WINDOW_SIZE minutes. To save flash and RAM,
00095 // the actual timing is approximate.
00096 #ifndef NM_WARNING_LIMIT
00097     #define NM_WARNING_LIMIT 16
00098 #endif
00099
00100 #ifndef NM_WINDOW_SIZE
00101     #define NM_WINDOW_SIZE 4
00102 #endif
00103
00104 // The channels that should be used by the network manager.
00105
00106 #ifndef NM_CHANNEL_MASK
00107     #define NM_CHANNEL_MASK EMBER_ALL_802_15_4_CHANNELS_MASK
00108 #endif
00109
00110 // The number of channels used in the NM_CHANNEL_MASK.
00111
00112 #ifndef NM_WATCHLIST_SIZE
00113     #define NM_WATCHLIST_SIZE 16
00114 #endif
00115
00122 void nmUtilWarningHandler(void);
00123
00132 bool nmUtilProcessIncoming(EmberApsFrame *
apsFrame,
                                uint8_t messageLength,
                                uint8_t* message);
00135
00139 EmberStatus nmUtilChangeChannelRequest (
void);
00140

```

8.55 platform-common.h File Reference

Macros

- #define [MEMSET](#)(d, v, l)
- #define [MEMCOPY](#)(d, s, l)
- #define [MEMMOVE](#)(d, s, l)
- #define [MEMCOMPARE](#)(s0, s1, l)
- #define [MEMPGMCOMPARE](#)(s0, s1, l)

Generic Types

- #define [TRUE](#)
- #define [FALSE](#)
- #define [NULL](#)

Bit Manipulation Macros

- #define [BIT](#)(x)
- #define [BIT32](#)(x)
- #define [SETBIT](#)(reg, bit)
- #define [SETBITS](#)(reg, bits)
- #define [CLEARBIT](#)(reg, bit)
- #define [CLEARBITS](#)(reg, bits)
- #define [READBIT](#)(reg, bit)
- #define [READBITS](#)(reg, bits)

Byte Manipulation Macros

- #define [LOW_BYTE](#)(n)
- #define [HIGH_BYTE](#)(n)
- #define [HIGH_LOW_TO_INT](#)(high, low)
- #define [BYTE_0](#)(n)
- #define [BYTE_1](#)(n)
- #define [BYTE_2](#)(n)
- #define [BYTE_3](#)(n)
- #define [COUNTOF](#)(a)

Time Manipulation Macros

- #define [elapsedTimeInt8u](#)(oldTime, newTime)
- #define [elapsedTimeInt16u](#)(oldTime, newTime)
- #define [elapsedTimeInt32u](#)(oldTime, newTime)
- #define [MAX_INT8U_VALUE](#)
- #define [HALF_MAX_INT8U_VALUE](#)
- #define [timeGTorEqualInt8u](#)(t1, t2)
- #define [MAX_INT16U_VALUE](#)
- #define [HALF_MAX_INT16U_VALUE](#)
- #define [timeGTorEqualInt16u](#)(t1, t2)
- #define [MAX_INT32U_VALUE](#)
- #define [HALF_MAX_INT32U_VALUE](#)
- #define [timeGTorEqualInt32u](#)(t1, t2)

Miscellaneous Macros

- #define [UNUSED_VAR](#)(x)
- #define [DEBUG_LEVEL](#)

8.55.1 Detailed Description

See [Common PLATFORM_HEADER Configuration](#) and micro specific modules for documentation.

Definition in file [platform-common.h](#).

8.56 platform-common.h

```

00001
00020 #ifndef PLATCOMMONOKTOINCLUDE
00021     // This header should only be included by a PLATFORM_HEADER
00022     #error platform-common.h should not be included directly
00023 #endif
00024
00025 #ifndef __PLATFORMCOMMON_H__
00026 #define __PLATFORMCOMMON_H__
00027
00028 // Many of the common definitions must be explicitly enabled by the
00029 // particular PLATFORM_HEADER being used
00031
00032
00034 #ifdef _HAL_USE_COMMON_PGM_
00035
00042     #define PGM        const
00043
00047     #define PGM_P      const char *
00048
00052     #define PGM_PU     const unsigned char *
00053
00054
00060     #define PGM_NO_CONST
00061
00062 #endif // _HAL_USE_COMMON_PGM_
00063
00064
00066 #ifdef _HAL_USE_COMMON_DIVMOD_
00067
00080     #define halCommonUDiv32By16(x, y) ((uint16_t) (((uint32_t) (x)) / ((uint16_t)
        (y))))
00081
00087     #define halCommonSDiv32By16(x, y) ((int16_t) (((int32_t) (x)) / ((int16_t)
        (y))))
00088
00094     #define halCommonUMod32By16(x, y) ((uint16_t) (((uint32_t) (x)) % ((uint16_t)
        (y))))
00095
00101     #define halCommonSMod32By16(x, y) ((int16_t) (((int32_t) (x)) % ((int16_t)
        (y))))
00102
00103 #endif // _HAL_USE_COMMON_DIVMOD_
00104
00105
00107 #ifdef _HAL_USE_COMMON_MEMUTILS_
00108
00120
00124     void halCommonMemMove(void *dest, const void *src, uint16_t bytes);
00125
00126
00130     void halCommonMemSet(void *dest, uint8_t val, uint16_t bytes);
00131
00132
00136     int16_t halCommonMemCompare(const void *source0, const void *source1,
        uint16_t bytes);
00137
00138
00143     int8_t halCommonMemPGMCompare(const void *source0, void PGM *source1,
        uint16_t bytes);
00144
00149     void halCommonMemPGMCopy(void* dest, void PGM *source, uint16_t bytes);
00150
00154     #define MEMSET(d,v,l)    halCommonMemSet(d,v,l)
00155     #define MEMCOPY(d,s,l)   halCommonMemMove(d,s,l)
00156     #define MEMMOVE(d,s,l)   halCommonMemMove(d,s,l)

```

```

00157 #define MEMCOMPARE(s0,s1,l) halCommonMemCompare(s0, s1, l)
00158 #define MEMPGMCOMPARE(s0,s1,l) halCommonMemPGMCompare(s0, s1, l)
00159
00161 #else
00162
00165 #define MEMSET(d,v,l) memset(d,v,l)
00166 #define MEMCOPY(d,s,l) memcpy(d,s,l)
00167 #define MEMMOVE(d,s,l) memmove(d,s,l)
00168 #define MEMCOMPARE(s0,s1,l) memcmp(s0, s1, l)
00169 #define MEMPGMCOMPARE(s0,s1,l) memcmp(s0, s1, l)
00170 #endif //__HAL_USE_COMMON_MEMUTILS_
00171
00172
00173
00174
00175
00176
00177
00178
00179
00181 // The following sections are common on all platforms
00183
00185
00193 #define TRUE 1
00194
00198 #define FALSE 0
00199
00200 #ifndef NULL
00201
00204 #define NULL ((void *)0)
00205 #endif
00206
00208
00209
00214
00218 #define BIT(x) (1U << (x)) // Unsigned avoids compiler warnings re BIT(15)
00219
00223 #define BIT32(x) (((uint32_t) 1) << (x))
00224
00230 #define SETBIT(reg, bit)      reg |= BIT(bit)
00231
00237 #define SETBITS(reg, bits)    reg |= (bits)
00238
00244 #define CLEARBIT(reg, bit)    reg &= ~(BIT(bit))
00245
00251 #define CLEARBITS(reg, bits)  reg &= ~(bits)
00252
00256 #define READBIT(reg, bit)     (reg & (BIT(bit)))
00257
00262 #define READBITS(reg, bits)   (reg & (bits))
00263
00265
00266
00268
00272
00276 #define LOW_BYTE(n)           ((uint8_t)((n) & 0xFF))
00277
00281 #define HIGH_BYTE(n)          ((uint8_t)(LOW_BYTE((n) >> 8)))
00282
00287 #define HIGH_LOW_TO_INT(high, low) ( \
00288     (( (uint16_t) (high) ) << 8) + \
00289     ( (uint16_t) ( (low) & 0xFF) ) \
00290 )
00291
00295 #define BYTE_0(n)             ((uint8_t)((n) & 0xFF))
00296
00300 #define BYTE_1(n)             ((uint8_t)(BYTE_0((n) >> 8)))
00301
00305 #define BYTE_2(n)             ((uint8_t)(BYTE_0((n) >> 16)))
00306
00310 #define BYTE_3(n)             ((uint8_t)(BYTE_0((n) >> 24)))
00311
00315 #define COUNTOF(a) (sizeof(a)/sizeof(a[0]))
00316
00318
00319
00321
00325
00330 #define elapsedTimeInt8u(oldTime, newTime) \
00331     ((uint8_t) ((uint8_t)newTime - (uint8_t)oldTime))

```

```

00332
00337 #define elapsedTimeInt16u(oldTime, newTime)      \
00338     ((uint16_t) ((uint16_t)(newTime) - (uint16_t)(oldTime)))
00339
00344 #define elapsedTimeInt32u(oldTime, newTime)      \
00345     ((uint32_t) ((uint32_t)(newTime) - (uint32_t)(oldTime)))
00346
00351 #define MAX_INT8U_VALUE      (0xFF)
00352 #define HALF_MAX_INT8U_VALUE (0x80)
00353 #define timeGTorEqualInt8u(t1, t2)              \
00354     (elapsedTimeInt8u(t2, t1) <= (HALF_MAX_INT8U_VALUE))
00355
00360 #define MAX_INT16U_VALUE      (0xFFFF)
00361 #define HALF_MAX_INT16U_VALUE (0x8000)
00362 #define timeGTorEqualInt16u(t1, t2)             \
00363     (elapsedTimeInt16u(t2, t1) <= (HALF_MAX_INT16U_VALUE))
00364
00369 #define MAX_INT32U_VALUE      (0xFFFFFFFFL)
00370 #define HALF_MAX_INT32U_VALUE (0x80000000L)
00371 #define timeGTorEqualInt32u(t1, t2)             \
00372     (elapsedTimeInt32u(t2, t1) <= (HALF_MAX_INT32U_VALUE))
00373
00375
00376
00378
00382
00383 #ifndef UNUSED_VAR
00384
00388 #define UNUSED_VAR(x) (void)(x)
00389 #endif
00390
00394 #ifndef DEBUG_LEVEL
00395     #if defined(DEBUG) && defined(DEBUG_STRIPPED)
00396         #error "DEBUG and DEBUG_OFF cannot be both be defined!"
00397     #elif defined(DEBUG)
00398         #define DEBUG_LEVEL FULL_DEBUG
00399     #elif defined(DEBUG_STRIPPED)
00400         #define DEBUG_LEVEL NO_DEBUG
00401     #else
00402         #define DEBUG_LEVEL BASIC_DEBUG
00403     #endif
00404 #endif
00405
00407
00408
00409 #endif //__PLATFORMCOMMON_H__
00410

```

8.57 serial.h File Reference

```
#include <yfuncs.h>
```

Enumerations

- enum [SerialBaudRate](#) {
[DEFINE_BAUD](#), [DEFINE_BAUD](#), [DEFINE_BAUD](#), [DEFINE_BAUD](#),
[DEFINE_BAUD](#), [DEFINE_BAUD](#), [DEFINE_BAUD](#), [DEFINE_BAUD](#),
[DEFINE_BAUD](#), [DEFINE_BAUD](#), [DEFINE_BAUD](#), [DEFINE_BAUD](#),
[DEFINE_BAUD](#), [DEFINE_BAUD](#), [DEFINE_BAUD](#), [DEFINE_BAUD](#),
[DEFINE_BAUD](#), [DEFINE_BAUD](#), [DEFINE_BAUD](#) }
- enum [NameOfType](#) { [DEFINE_PARITY](#), [DEFINE_PARITY](#), [DEFINE_PARITY](#) }

Serial HAL APIs

These functions must be implemented by the HAL in order for the serial code to operate. Only the higher-level serial code uses these functions, so they should not be called directly. The HAL should also implement the appropriate interrupt handlers to drain the TX queues and fill the RX FIFO queue, as necessary.

- [EmberStatus halInternalUartInit](#) (uint8_t port, [SerialBaudRate](#) rate, SerialParity parity, uint8_t stopBits)
- uint16_t [halInternalPrintfWriteAvailable](#) (void)
- uint16_t [halInternalPrintfReadAvailable](#) (void)
- void [halInternalForcePrintf](#) (bool onOff)

8.57.1 Detailed Description

See [Serial UART Communication](#) and micro specific modules for documentation.

Definition in file [hal/host/serial.h](#).

8.58 hal/host/serial.h

```

00001
00022 #ifndef __HAL_SERIAL_H__
00023 #define __HAL_SERIAL_H__
00024
00025 #include <yfuns.h>
00026
00027
00028 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00029
00033 enum SerialBaudRate
00034 #else
00035 #ifndef DEFINE_BAUD
00036 #define DEFINE_BAUD(num) BAUD_##num
00037 #endif
00038 typedef uint8_t SerialBaudRate;
00039 enum
00040 #endif //DOXYGEN_SHOULD_SKIP_THIS
00041 {
00042     DEFINE_BAUD(300) = 0, // BAUD_300
00043     DEFINE_BAUD(600) = 1, // BAUD_600
00044     DEFINE_BAUD(900) = 2, // etc...
00045     DEFINE_BAUD(1200) = 3,
00046     DEFINE_BAUD(2400) = 4,
00047     DEFINE_BAUD(4800) = 5,
00048     DEFINE_BAUD(9600) = 6,
00049     DEFINE_BAUD(14400) = 7,
00050     DEFINE_BAUD(19200) = 8,
00051     DEFINE_BAUD(28800) = 9,
00052     DEFINE_BAUD(38400) = 10,
00053     DEFINE_BAUD(50000) = 11,
00054     DEFINE_BAUD(57600) = 12,
00055     DEFINE_BAUD(76800) = 13,
00056     DEFINE_BAUD(100000) = 14,
00057     DEFINE_BAUD(115200) = 15,
00058     DEFINE_BAUD(230400) = 16,
00059     DEFINE_BAUD(460800) = 17,
00060     DEFINE_BAUD(CUSTOM) = 18
00061 };
00062
00063
00064 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00065
00069 enum NameOfType
00070 #else
00071 #ifndef DEFINE_PARITY
00072 #define DEFINE_PARITY(val) PARITY_##val

```



```

00073 #endif
00074 typedef uint8_t SerialParity;
00075 enum
00076 #endif //DOXYGEN_SHOULD_SKIP_THIS
00077 {
00078     DEFINE_PARITY(NONE) = 0, // PARITY_NONE
00079     DEFINE_PARITY(ODD) = 1,  // PARITY_ODD
00080     DEFINE_PARITY(EVEN) = 2  // PARITY_EVEN
00081 };
00082
00108 EmberStatus halInternalUartInit(uint8_t port,
00109                                SerialBaudRate rate,
00110                                SerialParity parity,
00111                                uint8_t stopBits);
00112
00118 uint16_t halInternalPrintfWriteAvailable(void);
00119
00125 uint16_t halInternalPrintfReadAvailable(void);
00126
00132 void halInternalForcePrintf(bool onOff);
00133
00134
00135 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00136 //Refer to uart.h for better documentation of fflush and stdout.
00137
00138 #if defined(__IAR_SYSTEMS_ICC__) && !defined(fflush)
00139     size_t fflush(int handle);
00140 #endif
00141
00142 #if defined(__IAR_SYSTEMS_ICC__) && !defined(stdout)
00143     #define stdout _LLIO_STDOUT
00144 #endif
00145
00146 #endif
00147
00150 #endif //__HAL_SERIAL_H__
00151

```

8.59 serial.h File Reference

Functions

- [EmberStatus emberSerialInit](#) (uint8_t port, [SerialBaudRate](#) rate, SerialParity parity, uint8_t stopBits)
- uint16_t [emberSerialReadAvailable](#) (uint8_t port)
- [EmberStatus emberSerialReadByte](#) (uint8_t port, uint8_t *dataByte)
- [EmberStatus emberSerialReadData](#) (uint8_t port, uint8_t *data, uint16_t length, uint16_t *bytesRead)
- [EmberStatus emberSerialReadDataTimeout](#) (uint8_t port, uint8_t *data, uint16_t length, uint16_t *bytesRead, uint16_t firstByteTimeout, uint16_t subsequentByteTimeout)
- [EmberStatus emberSerialReadLine](#) (uint8_t port, char *data, uint8_t max)
- [EmberStatus emberSerialReadPartialLine](#) (uint8_t port, char *data, uint8_t max, uint8_t *index)
- uint16_t [emberSerialWriteAvailable](#) (uint8_t port)
- uint16_t [emberSerialWriteUsed](#) (uint8_t port)
- [EmberStatus emberSerialWriteByte](#) (uint8_t port, uint8_t dataByte)
- [EmberStatus emberSerialWriteHex](#) (uint8_t port, uint8_t dataByte)
- [EmberStatus emberSerialWriteString](#) (uint8_t port, PGM_P string)
- XAP2B_PAGEZERO_ON [EmberStatus emberSerialPrintf](#) (uint8_t port, PGM_P formatString,...)
- XAP2B_PAGEZERO_OFF
- XAP2B_PAGEZERO_ON [EmberStatus emberSerialPrintfLine](#) (uint8_t port, PGM_P formatString,...)

- XAP2B_PAGEZERO_OFF
XAP2B_PAGEZERO_ON [EmberStatus emberSerialPrintCarriageReturn](#) (uint8_t port)
- XAP2B_PAGEZERO_OFF [EmberStatus emberSerialPrintfVarArg](#) (uint8_t port, PGM_P formatString, va_list ap)
- [EmberStatus emberSerialWriteData](#) (uint8_t port, uint8_t *data, uint8_t length)
- XAP2B_PAGEZERO_ON [EmberStatus emberSerialWaitSend](#) (uint8_t port)
- XAP2B_PAGEZERO_OFF [EmberStatus emberSerialGuaranteedPrintf](#) (uint8_t port, PGM_P formatString,...)
- void [emberSerialBufferTick](#) (void)
- void [emberSerialFlushRx](#) (uint8_t port)
- bool [emberSerialUnused](#) (uint8_t port)

8.59.1 Detailed Description

High-level serial communication functions. See [Serial Communication](#) for documentation.

Definition in file [app/util/serial/serial.h](#).

8.60 app/util/serial/serial.h

```

00001
00012 #ifndef __SERIAL_H__
00013 #define __SERIAL_H__
00014
00015 #ifndef __HAL_H__
00016     #error hal/hal.h should be included first
00017 #endif
00018
00019 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00020 #include <stdarg.h>
00021
00022 //Rx FIFO Full indicator
00023 #define RX_FIFO_FULL (0xFFFF)
00024
00025 #endif // DOXYGEN_SHOULD_SKIP_THIS
00026
00136 EmberStatus emberSerialInit(uint8_t port,
00137                             SerialBaudRate rate,
00138                             SerialParity parity,
00139                             uint8_t stopBits);
00140
00148 uint16_t emberSerialReadAvailable(uint8_t port);
00149
00167 EmberStatus emberSerialReadByte(uint8_t port,
00168                                 uint8_t *dataByte);
00169
00193 EmberStatus emberSerialReadData(uint8_t port,
00194                                 uint8_t *data,
00195                                 uint16_t length,
00196                                 uint16_t *bytesRead);
00197
00232 EmberStatus emberSerialReadDataTimeout(
00233     uint8_t port,
00234                                     uint8_t *data,
00235                                     uint16_t length,
00236                                     uint16_t *bytesRead,
00237                                     uint16_t firstByteTimeout,
00238                                     uint16_t subsequentByteTimeout);
00253 EmberStatus emberSerialReadLine(uint8_t port,
00254                                 char *data, uint8_t max);
00255
00278 EmberStatus emberSerialReadPartialLine(
00279     uint8_t port, char *data, uint8_t max, uint8_t *index);
00279
00288 uint16_t emberSerialWriteAvailable(uint8_t port);
00289

```

```

00297 uint16_t emberSerialWriteUsed(uint8_t port);
00298
00312 EmberStatus emberSerialWriteByte(uint8_t port,
uint8_t dataByte);
00313
00328 EmberStatus emberSerialWriteHex(uint8_t port,
uint8_t dataByte);
00329
00342 EmberStatus emberSerialWriteString(uint8_t
port, PGM_P string);
00343
00368 XAP2B_PAGEZERO_ON
00369 EmberStatus emberSerialPrintf(uint8_t port, PGM_P
formatString, ...);
00370 XAP2B_PAGEZERO_OFF
00371
00387 XAP2B_PAGEZERO_ON
00388 EmberStatus emberSerialPrintfLine(uint8_t port,
PGM_P formatString, ...);
00389 XAP2B_PAGEZERO_OFF
00390
00401 XAP2B_PAGEZERO_ON
00402 EmberStatus emberSerialPrintCarriageReturn
(uint8_t port);
00403 XAP2B_PAGEZERO_OFF
00404
00405
00418 EmberStatus emberSerialPrintfVarArg(uint8_t
port, PGM_P formatString, va_list ap);
00419
00435 EmberStatus emberSerialWriteData(uint8_t port,
uint8_t *data, uint8_t length);
00436
00437 //Host HALs do not use stack buffers.
00438 #ifndef HAL_HOST
00439
00457 EmberStatus emberSerialWriteBuffer(uint8_t port, EmberMessageBuffer
buffer, uint8_t start, uint8_t length);
00458 #endif //HAL_HOST
00459
00472 XAP2B_PAGEZERO_ON
00473 EmberStatus emberSerialWaitSend(uint8_t port);
00474 XAP2B_PAGEZERO_OFF
00475
00496 EmberStatus emberSerialGuaranteedPrintf(
uint8_t port, PGM_P formatString, ...);
00497
00503 void emberSerialBufferTick(void);
00504
00510 void emberSerialFlushRx(uint8_t port);
00511
00518 bool emberSerialUnused(uint8_t port);
00519
00520
00521
00522
00523
00526 #endif // __SERIAL_H__
00527

```

8.61 spi-protocol-common.h File Reference

```

#include "app/util/ezsp/ezsp-enum.h"
#include "spi-protocol-specific.h"

```

Functions

- void [halNcpSerialInit](#) (void)
- void [halNcpSerialPowerup](#) (void)
- void [halNcpSerialPowerdown](#) (void)

- EzspStatus [halNcpHardReset](#) (void)
- EzspStatus [halNcpHardResetReqBootload](#) (bool requestBootload)
- void [halNcpWakeUp](#) (void)
- void [halNcpSendCommand](#) (void)
- void [halNcpSendRawCommand](#) (void)
- EzspStatus [halNcpPollForResponse](#) (void)
- void [halNcpIsAwakeIsr](#) (bool isAwake)
- bool [halNcpHasData](#) (void)
- bool [halNcpVerifySpiProtocolVersion](#) (void)
- bool [halNcpVerifySpiProtocolActive](#) (void)

Variables

- uint8_t * [halNcpFrame](#)
- uint8_t [halNcpSpiErrorByte](#)

8.61.1 Detailed Description

See [SPI Protocol](#) and micro specific modules for documentation.

Definition in file [spi-protocol-common.h](#).

8.62 spi-protocol-common.h

```

00001
00020 #ifndef __SPI_PROTOCOL_COMMON_H__
00021 #define __SPI_PROTOCOL_COMMON_H__
00022
00023 #include "app/util/ezsp/ezsp-enum.h"
00024
00032 extern uint8_t *halNcpFrame;
00033
00041 extern uint8_t halNcpSpiErrorByte;
00042
00046 void halNcpSerialInit(void);
00047
00052 void halNcpSerialPowerup(void);
00053
00057 void halNcpSerialPowerdown(void);
00058
00071 EzspStatus halNcpHardReset(void);
00072
00087 EzspStatus halNcpHardResetReqBootload(bool
    requestBootload);
00088
00098 void halNcpWakeUp(void);
00099
00111 void halNcpSendCommand(void);
00112
00124 void halNcpSendRawCommand(void);
00125
00134 EzspStatus halNcpPollForResponse(void);
00135
00143 void halNcpIsAwakeIsr(bool isAwake);
00144
00149 bool halNcpHasData(void);
00150
00151
00159 bool halNcpVerifySpiProtocolVersion(void);
00160
00168 bool halNcpVerifySpiProtocolActive(void);
00169
00171

```

```

00172
00173 //Pull in the micro specific spi protocol definitions. The specific header is
00174 //chosen by the build include path pointing at the appropriate directory.
00175 #include "spi-protocol-specific.h"
00176
00177
00178 #endif // __SPI_PROTOCOL_COMMON_H__
00179

```

8.63 spi-protocol-specific.h File Reference

SPI Protocol Interface

- #define [SPIP_nSSEL_PORT](#)
- #define [SPIP_nSSEL_PIN](#)
- #define [SPIP_MOSI_PORT](#)
- #define [SPIP_MOSI_PIN](#)
- #define [SPIP_MISO_PORT](#)
- #define [SPIP_MISO_PIN](#)
- #define [SPIP_SCLK_PORT](#)
- #define [SPIP_SCLK_PIN](#)
- #define [SPIP_nHOST_INT_PORT](#)
- #define [SPIP_nHOST_INT_PIN](#)
- #define [SPIP_nWAKE_PORT](#)
- #define [SPIP_nWAKE_PIN](#)
- #define [SPIP_nRESET_PORT](#)
- #define [SPIP_nRESET_PIN](#)

SPI Protocol timing parameters.

Note

Remember: TIM2 is configured to produce a 125us tick.

- #define [WAIT_SECTION_TIMEOUT](#)
- #define [WAKE_HANDSHAKE_TIMEOUT](#)
- #define [STARTUP_TIMEOUT](#)
- #define [INTER_COMMAND_SPACING](#)
- #define [NCP_RESET_DELAY](#)

8.63.1 Detailed Description

See [SPI Protocol](#) and [STM32F103RET Specific SPI Protocol](#) for documentation.

Definition in file [spi-protocol-specific.h](#).

8.64 spi-protocol-specific.h

```

00001
00022 #ifndef __SPI_PROTOCOL_SPECIFIC_H__
00023 #define __SPI_PROTOCOL_SPECIFIC_H__
00024

```

```

00029
00033 #define SPIP_nSSEL_PORT      GPIOA
00034
00037 #define SPIP_nSSEL_PIN        GPIO_Pin_4
00038
00039
00043 #define SPIP_MOSI_PORT        GPIOA
00044
00047 #define SPIP_MOSI_PIN         GPIO_Pin_7
00048
00052 #define SPIP_MISO_PORT        GPIOA
00053
00056 #define SPIP_MISO_PIN         GPIO_Pin_6
00057
00061 #define SPIP_SCLK_PORT        GPIOA
00062
00065 #define SPIP_SCLK_PIN         GPIO_Pin_5
00066
00070 #define SPIP_nHOST_INT_PORT    GPIOC
00071
00074 #define SPIP_nHOST_INT_PIN     GPIO_Pin_4
00075
00079 #define SPIP_nWAKE_PORT        GPIOC
00080
00083 #define SPIP_nWAKE_PIN         GPIO_Pin_5
00084
00088 #define SPIP_nRESET_PORT       GPIOB
00089
00092 #define SPIP_nRESET_PIN        GPIO_Pin_0
00093
00106 #define WAIT_SECTION_TIMEOUT    (2400) //300ms //Increased per BugzId:14656
00107
00110 #define WAKE_HANDSHAKE_TIMEOUT    (2400) //300ms //Increased per BugzId:14655,6
00111
00114 #define STARTUP_TIMEOUT          (60000) //7500ms
00115
00118 #define INTER_COMMAND_SPACING      (8) //1ms
00119
00122 #define NCP_RESET_DELAY    (26)
00123
00127 #endif // __SPI_PROTOCOL_SPECIFIC_H__
00128

```

8.65 stm32f10x_conf.h File Reference

```
#include "stm32f10x_adc.h"
```

```

#include "stm32f10x_bkp.h"
#include "stm32f10x_can.h"
#include "stm32f10x_cec.h"
#include "stm32f10x_crc.h"
#include "stm32f10x_dac.h"
#include "stm32f10x_dbgmcu.h"
#include "stm32f10x_dma.h"
#include "stm32f10x_exti.h"
#include "stm32f10x_flash.h"
#include "stm32f10x_fsmc.h"
#include "stm32f10x_gpio.h"
#include "stm32f10x_i2c.h"
#include "stm32f10x_iwdg.h"
#include "stm32f10x_pwr.h"
#include "stm32f10x_rcc.h"
#include "stm32f10x_rtc.h"
#include "stm32f10x_sdio.h"
#include "stm32f10x_spi.h"
#include "stm32f10x_tim.h"
#include "stm32f10x_usart.h"
#include "stm32f10x_wwdg.h"
#include "misc.h"

```

Macros

- `#define assert_param(condition)`

Functions

- void [halInternalAssertFailed](#) (const char *filename, int linenumber)

8.65.1 Detailed Description

[Microcontroller General Functionality](#) and [STM32F103RET General Functionality](#) for documentation.

Definition in file [stm32f10x_conf.h](#).

8.66 stm32f10x_conf.h

```

00001
00027 #ifndef __STM32F10x_CONF_H
00028 #define __STM32F10x_CONF_H
00029
00030
00031 //Peripheral header file inclusion. There is a header per peripheral source
00032 //found in the library.
00033 #include "stm32f10x_adc.h"
00034 #include "stm32f10x_bkp.h"
00035 #include "stm32f10x_can.h"
00036 #include "stm32f10x_cec.h"
00037 #include "stm32f10x_crc.h"
00038 #include "stm32f10x_dac.h"
00039 #include "stm32f10x_dbgmcu.h"

```

```

00040 #include "stm32f10x_dma.h"
00041 #include "stm32f10x_exti.h"
00042 #include "stm32f10x_flash.h"
00043 #include "stm32f10x_fsmc.h"
00044 #include "stm32f10x_gpio.h"
00045 #include "stm32f10x_i2c.h"
00046 #include "stm32f10x_iwdg.h"
00047 #include "stm32f10x_pwr.h"
00048 #include "stm32f10x_rcc.h"
00049 #include "stm32f10x_rtc.h"
00050 #include "stm32f10x_sdio.h"
00051 #include "stm32f10x_spi.h"
00052 #include "stm32f10x_tim.h"
00053 #include "stm32f10x_usart.h"
00054 #include "stm32f10x_wwdg.h"
00055 //misc.h is for High level functions for NVIC and SysTick, which
00056 //are add-on to CMSIS functions.
00057 #include "misc.h"
00058
00059
00060 //The library uses it's own assert macro (assert_param), so link the library's
00061 //assert to our usual assert.
00062 #if !defined(SIMPLER_ASSERT_REBOOT)
00063
00067     void halInternalAssertFailed(const char * filename,
                                int linenumber);
00068
00075     #define assert_param(condition) \
00076         do { \
00077             if (! (condition)) { \
00078                 halInternalAssertFailed(__SOURCEFILE__, __LINE__); \
00079             } \
00080         } while(0)
00081 #else
00082     #define assert_param(condition) \
00083         do { if( !(condition) ) while(1){} } while(0)
00084 #endif
00085
00086
00087 #endif /* __STM32F10x_CONF_H */
00088

```

8.67 system-timer.h File Reference

Functions

- uint16_t [halInternalStartSystemTimer](#) (void)
- uint16_t [halCommonGetInt16uMillisecondTick](#) (void)
- uint32_t [halCommonGetInt32uMillisecondTick](#) (void)
- uint16_t [halCommonGetInt16uQuarterSecondTick](#) (void)
- void [halCommonSetSystemTime](#) (uint32_t time)

8.67.1 Detailed Description

See [System Timer](#) for documentation.

Definition in file [system-timer.h](#).

8.68 system-timer.h

```

00001
00029 #ifndef __SYSTEM_TIMER_H__
00030 #define __SYSTEM_TIMER_H__
00031
00032

```



```

00039 uint16_t halInternalStartSystemTimer(void);
00040
00041
00049 uint16_t halCommonGetInt16uMillisecondTick(
void);
00050
00058 uint32_t halCommonGetInt32uMillisecondTick(
void);
00059
00067 uint16_t halCommonGetInt16uQuarterSecondTick
(void);
00068
00074 void halCommonSetSystemTime(uint32_t time);
00075
00076
00077 #endif //__SYSTEM_TIMER_H__
00078

```

8.69 uart.h File Reference

```
#include <yfuncs.h>
```

Macros

- #define [stdout](#)

Functions

- size_t [fflush](#) (int handle)

8.69.1 Detailed Description

See [Serial UART Communication](#) and [STM32F103RET Specific UART](#) for documentation.

Definition in file [uart.h](#).

8.70 uart.h

```

00001
00018 #ifndef __UART_H__
00019 #define __UART_H__
00020 #include <yfuncs.h>
00021
00033 size_t fflush(int handle);
00034
00041 #define stdout _LLIO_STDOUT
00042
00043 #endif //__UART_H__
00044

```

8.71 zigbee-device-common.h File Reference

Macros

- `#define ZDO_MESSAGE_OVERHEAD`

Service Discovery Functions

- `EmberStatus emberNodeDescriptorRequest` (`EmberNodeId` target, `EmberApsOption` options)
- `EmberStatus emberPowerDescriptorRequest` (`EmberNodeId` target, `EmberApsOption` options)
- `EmberStatus emberSimpleDescriptorRequest` (`EmberNodeId` target, `uint8_t` target-Endpoint, `EmberApsOption` options)
- `EmberStatus emberActiveEndpointsRequest` (`EmberNodeId` target, `EmberApsOption` options)

Binding Manager Functions

- `EmberStatus emberBindRequest` (`EmberNodeId` target, `EmberEUI64` source, `uint8_t` sourceEndpoint, `uint16_t` clusterId, `uint8_t` type, `EmberEUI64` destination, `EmberMulticastId` groupAddress, `uint8_t` destinationEndpoint, `EmberApsOption` options)
- `EmberStatus emberUnbindRequest` (`EmberNodeId` target, `EmberEUI64` source, `uint8_t` sourceEndpoint, `uint16_t` clusterId, `uint8_t` type, `EmberEUI64` destination, `EmberMulticastId` groupAddress, `uint8_t` destinationEndpoint, `EmberApsOption` options)

Node Manager Functions

- `EmberStatus emberLqiTableRequest` (`EmberNodeId` target, `uint8_t` startIndex, `EmberApsOption` options)
- `EmberStatus emberRoutingTableRequest` (`EmberNodeId` target, `uint8_t` startIndex, `EmberApsOption` options)
- `EmberStatus emberBindingTableRequest` (`EmberNodeId` target, `uint8_t` startIndex, `EmberApsOption` options)
- `EmberStatus emberLeaveRequest` (`EmberNodeId` target, `EmberEUI64` deviceAddress, `uint8_t` leaveRequestFlags, `EmberApsOption` options)
- `EmberStatus emberPermitJoiningRequest` (`EmberNodeId` target, `uint8_t` duration, `uint8_t` authentication, `EmberApsOption` options)
- `void emberSetZigDevRequestRadius` (`uint8_t` radius)
- `uint8_t emberGetZigDevRequestRadius` (void)
- `uint8_t emberGetLastZigDevRequestSequence` (void)
- `uint8_t emberGetLastAppZigDevRequestSequence` (void)

8.71.1 Detailed Description

ZigBee Device Object (ZDO) functions available on all platforms. See [ZigBee Device Object \(ZDO\) Information](#) for documentation.

Definition in file [zigbee-device-common.h](#).

8.72 zigbee-device-common.h

```

00001
00016 #define ZDO_MESSAGE_OVERHEAD 1
00017
00036 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00037 EmberStatus emberNodeDescriptorRequest(
00038     EmberNodeId target,
00039                                     EmberApsOption options);
00039 #else
00040 // Macroized to save code space.
00041 EmberStatus emberSendZigDevRequestTarget(EmberNodeId
00042     target,
00043                                     uint16_t clusterId,
00044                                     EmberApsOption options);
00044 #define emberNodeDescriptorRequest(target, opts) \
00045     (emberSendZigDevRequestTarget((target), NODE_DESCRIPTOR_REQUEST, (opts)))
00046 #endif
00047
00063 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00064 EmberStatus emberPowerDescriptorRequest(
00065     EmberNodeId target,
00066                                     EmberApsOption options);
00066 #else
00067 // Macroized to save code space.
00068 #define emberPowerDescriptorRequest(target, opts) \
00069     (emberSendZigDevRequestTarget((target), POWER_DESCRIPTOR_REQUEST, (opts)))
00070 #endif
00071
00090 EmberStatus emberSimpleDescriptorRequest
00091     (EmberNodeId target,
00092                                     uint8_t targetEndpoint,
00093                                     EmberApsOption options);
00093
00106 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00107 EmberStatus emberActiveEndpointsRequest(
00108     EmberNodeId target,
00109                                     EmberApsOption options);
00109 #else
00110 // Macroized to save code space.
00111 #define emberActiveEndpointsRequest(target, opts) \
00112     (emberSendZigDevRequestTarget((target), ACTIVE_ENDPOINTS_REQUEST, (opts)))
00113 #endif
00114
00144 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00145 EmberStatus emberBindRequest(EmberNodeId
00146     target,
00147                                     EmberEUI64 source,
00148                                     uint8_t sourceEndpoint,
00149                                     uint16_t clusterId,
00150                                     uint8_t type,
00151                                     EmberEUI64 destination,
00152                                     EmberMulticastId groupAddress,
00153                                     uint8_t destinationEndpoint,
00154                                     EmberApsOption options);
00154 #else
00155 // Macroized to save code space.
00156 #define emberBindRequest(target,
00157     src,
00158     srcEndpt,
00159     cluster,
00160     type,
00161     dest,
00162     groupAddress,
00163     destEndpt,
00164     opts)
00165     (emberSendZigDevBindRequest((target),
00166     BIND_REQUEST,
00167     (src), (srcEndpt), (cluster),
00168     (type), (dest), (groupAddress),
00169     (destEndpt), (opts)))
00170 #endif
00171
00172 EmberStatus emberSendZigDevBindRequest(EmberNodeId target
00173     ,
00174                                     uint16_t bindClusterId,
00175                                     EmberEUI64 source,
00176                                     uint8_t sourceEndpoint,
00177                                     uint16_t clusterId,

```

```

00177             uint8_t type,
00178             EmberEUI64 destination,
00179             EmberMulticastId
groupAddress,
00180             uint8_t destinationEndpoint,
00181             EmberApsOption options);
00182 #endif
00183
00210 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00211 EmberStatus emberUnbindRequest(EmberNodeId
target,
00212             EmberEUI64 source,
00213             uint8_t sourceEndpoint,
00214             uint16_t clusterId,
00215             uint8_t type,
00216             EmberEUI64 destination,
00217             EmberMulticastId groupAddress,
00218             uint8_t destinationEndpoint,
00219             EmberApsOption options);
00220 #else
00221 // Macroized to save code space.
00222 #define emberUnbindRequest(target,
00223             src,
00224             srcEndpt,
00225             cluster,
00226             type,
00227             dest,
00228             groupAddress,
00229             destEndpt,
00230             opts)
00231
00232 (emberSendZigDevBindRequest((target),
00233             UNBIND_REQUEST,
00234             (src), (srcEndpt), (cluster),
00235             (type), (dest), (groupAddress),
00236             (destEndpt), (opts)))
00237 #endif
00238
00261 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00262 EmberStatus emberLqiTableRequest(EmberNodeId
target,
00263             uint8_t startIndex,
00264             EmberApsOption options);
00265 #else
00266 #define emberLqiTableRequest(target, startIndex, options) \
00267 (emberTableRequest(LQI_TABLE_REQUEST, (target), (startIndex), (options)))
00268
00269 EmberStatus emberTableRequest(uint16_t clusterId,
00270             EmberNodeId target,
00271             uint8_t startIndex,
00272             EmberApsOption options);
00273 #endif
00274
00291 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00292 EmberStatus emberRoutingTableRequest(
EmberNodeId target,
00293             uint8_t startIndex,
00294             EmberApsOption options);
00295 #else
00296 #define emberRoutingTableRequest(target, startIndex, options) \
00297 (emberTableRequest(ROUTING_TABLE_REQUEST, (target), (startIndex), (options)))
00298 #endif
00299
00317 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00318 EmberStatus emberBindingTableRequest(
EmberNodeId target,
00319             uint8_t startIndex,
00320             EmberApsOption options);
00321 #else
00322 #define emberBindingTableRequest(target, startIndex, options) \
00323 (emberTableRequest(BINDING_TABLE_REQUEST, (target), (startIndex), (options)))
00324 #endif
00325
00345 EmberStatus emberLeaveRequest(EmberNodeId
target,
00346             EmberEUI64 deviceAddress,
00347             uint8_t leaveRequestFlags,
00348             EmberApsOption options);
00349
00366 EmberStatus emberPermitJoiningRequest(

```

```

    EmberNodeId target,
00367                                     uint8_t duration,
00368                                     uint8_t authentication,
00369                                     EmberApsOption options);
00370
00371 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00372
00377 void emberSetZigDevRequestRadius(uint8_t radius);
00378
00384 uint8_t emberGetZigDevRequestRadius(void);
00392 uint8_t emberGetLastZigDevRequestSequence(void
    );
00393 #else
00394 extern uint8_t zigDevRequestRadius;
00395 #define emberGetZigDevRequestRadius() (zigDevRequestRadius)
00396 #define emberSetZigDevRequestRadius(x) (zigDevRequestRadius=x)
00397 #define emberGetLastZigDevRequestSequence() \
00398     (emberGetLastAppZigDevRequestSequence())
00399 #endif
00400
00407 uint8_t emberGetLastAppZigDevRequestSequence
    (void);
00408
00411 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00412 //
    -----
00413 // Utility functions used by the library code.
00414
00415 EmberStatus emberSendZigDevRequest(EmberNodeId
    destination,
00416                                     uint16_t clusterId,
00417                                     EmberApsOption options,
00418                                     uint8_t *contents,
00419                                     uint8_t length);
00420
00430 uint8_t emberNextZigDevRequestSequence(void);
00431
00432 #endif // DOXYGEN_SHOULD_SKIP_THIS
00433

```

8.73 zigbee-device-host.h File Reference

Device Discovery Functions

- [EmberStatus emberNetworkAddressRequest](#) ([EmberEUI64](#) target, bool reportKids, uint8_t childStartIndex)
- [EmberStatus emberIeeeAddressRequest](#) ([EmberNodeId](#) target, bool reportKids, uint8_t childStartIndex, [EmberApsOption](#) options)

Service Discovery Functions

- [EmberStatus ezspMatchDescriptorsRequest](#) ([EmberNodeId](#) target, uint16_t profile, uint8_t inCount, uint8_t outCount, uint16_t *inClusters, uint16_t *outClusters, [EmberApsOption](#) options)

Binding Manager Functions

- [EmberStatus ezspEndDeviceBindRequest](#) ([EmberNodeId](#) localNodeId, [EmberEUI64](#) localEui64, uint8_t endpoint, uint16_t profile, uint8_t inCount, uint8_t outCount, uint16_t *inClusters, uint16_t *outClusters, [EmberApsOption](#) options)

Function to Decode Address Response Messages

- [EmberNodeId ezspDecodeAddressResponse](#) (uint8_t *response, [EmberEUI64](#) eui64-Return)

8.73.1 Detailed Description

ZigBee Device Object (ZDO) functions not provided by the stack. See [ZigBee Device Object \(ZDO\) Information](#) for documentation.

Definition in file [zigbee-device-host.h](#).

8.74 zigbee-device-host.h

```

00001
00104 EmberStatus emberNetworkAddressRequest (
    EmberEUI64 target,
00105                                     bool reportKids,
00106                                     uint8\_t childStartIndex);
00107
00125 EmberStatus emberIeeeAddressRequest (
    EmberNodeId target,
00126                                     bool reportKids,
00127                                     uint8\_t childStartIndex,
00128                                     EmberApsOption options);
00157 EmberStatus ezspMatchDescriptorsRequest (
    EmberNodeId target,
00158                                     uint16\_t profile,
00159                                     uint8\_t inCount,
00160                                     uint8\_t outCount,
00161                                     uint16\_t *inClusters,
00162                                     uint16\_t *outClusters,
00163                                     EmberApsOption options);
00189 EmberStatus ezspEndDeviceBindRequest (
    EmberNodeId localNodeId,
00190                                     EmberEUI64 localEui64,
00191                                     uint8\_t endpoint,
00192                                     uint16\_t profile,
00193                                     uint8\_t inCount,
00194                                     uint8\_t outCount,
00195                                     uint16\_t *inClusters,
00196                                     uint16\_t *outClusters,
00197                                     EmberApsOption options);
00216 EmberNodeId ezspDecodeAddressResponse (
    uint8\_t *response,
00217                                     EmberEUI64 eui64Return);
00218

```