SILICON LABS

EM35x
EZSP

EmberZNet 5.7.0 API Reference:

For the PC Host

February 21, 2016
120-3026-000-5700

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

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Document 120-3024-000A, *EmberZNet API Reference: For the EM35x Network Co--Processor*, has been obsoleted and superseded by this document with respect to the PC Host functionality. STM32F103RET Host functionality is now documented in 120-3025-000.

The EmberZNet API Reference documentation for the PC 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:

Chapter 4

Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Module Documentation

6.1 Ember Common

Modules

- Ember Common Data Types
- Sending and Receiving Messages
- Ember Status Codes
- 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 ::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_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_TABL-E, EMBER_OUTGOING_VIA_BINDING, EMBER_OUTGOING_MULTICAST, EMBER_OUTGOING_MULTICAST_WITH_ALIAS, EMBER_OUTGOING_BR-OADCAST_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_T-YPE_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_BIN-DING }
- enum EmberJoinDecision { EMBER_USE_PRECONFIGURED_KEY, EMBER_S-END_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_STAND-ARD_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_REJOI-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_OUTP-UT_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_COUNT-ER_MAC_TX_UNICAST_SUCCESS,
  EMBER_COUNTER_MAC_TX_UNICAST_RETRY, EMBER_COUNTER_MA-C_TX_UNICAST_FAILED, EMBER_COUNTER_APS_DATA_RX_BROADCA-

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-

MBER_APP_MASTER_KEY_ESTABLISHED, EMBER_TRUST_CENTER_LIN-K_KEY_ESTABLISHED,
EMBER_KEY_ESTABLISHMENT_TIMEOUT, EMBER_KEY_TABLE_FULL, E-MBER_TC_RESPONDED_TO_KEY_REQUEST, EMBER_TC_APP_KEY_SEN-T_TO_REQUESTER,
EMBER_TC_RESPONSE_TO_KEY_REQUEST_FAILED, EMBER_TC_REQUE-ST_KEY_TYPE_NOT_SUPPORTED, EMBER_TC_NO_LINK_KEY_FOR_REQ-UESTER, EMBER_TC_REQUESTER_EUI64_UNKNOWN,
EMBER_TC_RECEIVED_FIRST_APP_KEY_REQUEST, EMBER_TC_TIMEO-UT_WAITING_FOR_SECOND_APP_KEY_REQUEST, EMBER_TC_NON_MA-TCHING_APP_KEY_REQUEST_RECEIVED, EMBER_TC_FAILED_TO_SEND-_APP_KEYS,
EMBER_TC_FAILED_TO_STORE_APP_KEY_REQUEST, EMBER_TC_REJE-CTED_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, EM-BER_TC_REQUESTER_VERIFY_KEY_SUCCESS,
EMBER_VERIFY_LINK_KEY_FAILURE, EMBER_VERIFY_LINK_KEY_SUC-CESS }
- enum EmberLinkKeyRequestPolicy { EMBER_DENY_KEY_REQUESTS, EMBE-R_ALLOW_KEY_REQUESTS, EMBER_GENERATE_NEW_TC_LINK_KEY }
- enum EmberKeySettings { EMBER_KEY_PERMISSIONS_NONE, EMBER_KE-Y_PERMISSIONS_READING_ALLOWED, EMBER_KEY_PERMISSIONS_HA-SHING_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_PASSTHRO-UGH_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_ALPH-A_1, EMBER_VERSION_TYPE_ALPHA_2, EMBER_VERSION_TYPE_ALPH-

A_3,
EMBER_VERSION_TYPE_BETA_1, EMBER_VERSION_TYPE_BETA_2, EM-
BER_VERSION_TYPE_BETA_3, EMBER_VERSION_TYPE_GA }
- enum EmberLeaveRequestFlags { EMBER_ZIGBEE_LEAVE_AND_REJOIN, E-
MBER_ZIGBEE_LEAVE_AND_REMOVE_CHILDREN }
- enum EmberLeaveReason {
EMBER_LEAVE_REASON_NONE, EMBER_LEAVE_DUE_TO_NWK_LEAV-
E_MESSAGE, EMBER_LEAVE_DUE_TO_APS_REMOVE_MESSAGE, EMBE-
R_LEAVE_DUE_TO_ZDO_LEAVE_MESSAGE,
EMBER_LEAVE_DUE_TO_ZLL_TOUCHLINK, EMBER_LEAVE_DUE_TO_A-
PP_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_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

```
<br>

@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

```
<br>

@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_DESCRIPTORS_REQUEST
- #define MATCH_DESCRIPTORS_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_TR-UST_CENTER, EMBER_ZDP_PRIMARY_BINDING_TABLE_CACHE, EMBE-R_ZDP_SECONDARY_BINDING_TABLE_CACHE,
  EMBER_ZDP_PRIMARY_DISCOVERY_CACHE, EMBER_ZDP_SECONDAR-Y_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_APP-LICATION_HANDLES_UNSUPPORTED_ZDO_REQUESTS EMBER_APPLICATIO-N_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_REQUES-TS, 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 ::emberIncomingMessageHandler(). 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_R-EQUEST.

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_DESCRIPTORS_REQUEST

Definition at line 2382 of file ember-types.h.

6.2.2.99 #define MATCH_DESCRIPTORS_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 E-CC 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 M-AC 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 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 EMBER_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 is 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

key	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

cert	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

key	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

key	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

cert	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

key	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

key	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 AP-S 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 emberMacPassthrough-MessageHandler(...). The header is parsed and the various fields are written to the InterPan-Header. 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

symbol	The name of the constant being defined. All Ember returns begin with EMBER_. For example, ::EMBER_CONNECTION_OPEN.
value	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 relevent 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_MESSAG-E_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_LIN-K_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 ::emberSet-InitialSecurityState() 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(*xAA*)

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 Configuration

Macros

- #define EZSP_HOST_SOURCE_ROUTE_TABLE_SIZE
- #define EZSP_HOST_RX_POOL_SIZE
- #define EZSP_HOST_FORM_AND_JOIN_BUFFER_SIZE

6.5.1 Detailed Description

See ezsp-host-configuration-defaults.h for source code.

6.5.2 Macro Definition Documentation

6.5.2.1 #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.

6.5.2.2 #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.

6.5.2.3 #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.

6.6 Hardware Abstraction Layer (HAL) API Reference

Modules

- HAL Configuration
- Asynchronous Serial Host (ASH) Framework
- EM2xx-compatible Resets
- System Timer
- HAL Utilities

6.6.1 Detailed Description

```
PC Host
```

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 functionalty of any halStack or halCommon APIs.

See also hal.h.

6.7 HAL Configuration

Modules

- Common PLATFORM_HEADER Configuration

6.7.1 Detailed Description

Configuration information that affects the entire HAL.

6.8 Common PLATFORM_HEADER Configuration

Modules

- Unix GCC Specific PLATFORM_HEADER Configuration

Macros

- #define MEMSET(d, v, l)
- #define MEMCOPY(d, s, l)
- #define MEMMOVE(d, s, l)
- #define MEMPGMCOPY(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

6.8.1 Detailed Description

Compiler and Platform specific definitions and typedefs common to all platforms. 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.

6.8.2 Macro Definition Documentation

6.8.2.1 #define MEMSET(*d*, *v*, *l*)

Friendly convenience macro pointing to the C Stdlib functions.

Note that the casting below ensures that IAR will not attempt to optimize these calls away and inline the code. This optimization can cause faults when accessing structs allocated on the heap since they are not always word aligned. Versions greater than 6.40 don't seem to have this issue.

Definition at line 184 of file platform-common.h.

6.8.2.2 #define MEMCOPY(*d*, *s*, *l*)

Definition at line 185 of file platform-common.h.

6.8.2.3 #define MEMMOVE(*d*, *s*, *l*)

Definition at line 186 of file platform-common.h.

6.8.2.4 #define MEMPGMCOPY(*d*, *s*, *l*)

Definition at line 187 of file platform-common.h.

6.8.2.5 #define MEMCOMPARE(*s0*, *s1*, *l*)

Definition at line 188 of file platform-common.h.

6.8.2.6 #define MEMPGMCOMPARE(*s0*, *s1*, *l*)

Definition at line 189 of file platform-common.h.

6.8.2.7 #define TRUE

An alias for one, used for clarity.

Definition at line 213 of file platform-common.h.

6.8.2.8 #define FALSE

An alias for zero, used for clarity.

Definition at line 218 of file platform-common.h.

6.8.2.9 #define NULL

The null pointer.

Definition at line 224 of file platform-common.h.

6.8.2.10 #define BIT(*x*)

Useful to reference a single bit of a byte.

Definition at line 238 of file platform-common.h.

6.8.2.11 #define BIT32(*x*)

Useful to reference a single bit of an uint32_t type.

Definition at line 243 of file platform-common.h.

6.8.2.12 #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 250 of file platform-common.h.

6.8.2.13 #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 257 of file platform-common.h.

6.8.2.14 #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 264 of file platform-common.h.

6.8.2.15 #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 271 of file platform-common.h.

6.8.2.16 #define READBIT(*reg,* *bit*)

Returns the value of `bit` within the register or byte `reg`.

Definition at line 276 of file platform-common.h.

6.8.2.17 #define READBITS(*reg,* *bits*)

Returns the value of the bitmask `bits` within the register or byte `reg`.

Definition at line 282 of file platform-common.h.

6.8.2.18 #define LOW_BYTE(*n*)

Returns the low byte of the 16-bit value `n` as an `uint8_t`.

Definition at line 296 of file platform-common.h.

6.8.2.19 #define HIGH_BYTE(*n*)

Returns the high byte of the 16-bit value `n` as an `uint8_t`.

Definition at line 301 of file platform-common.h.

6.8.2.20 #define HIGH_LOW_TO_INT(*high, low*)

Returns the value built from the two `uint8_t` values `high` and `low`.

Definition at line 307 of file platform-common.h.

6.8.2.21 #define BYTE_0(*n*)

Returns the low byte of the 32-bit value `n` as an `uint8_t`.

Definition at line 315 of file platform-common.h.

6.8.2.22 #define BYTE_1(*n*)

Returns the second byte of the 32-bit value `n` as an `uint8_t`.

Definition at line 320 of file platform-common.h.

6.8.2.23 #define BYTE_2(*n*)

Returns the third byte of the 32-bit value `n` as an `uint8_t`.

Definition at line 325 of file platform-common.h.

6.8.2.24 #define BYTE_3(*n*)

Returns the high byte of the 32-bit value `n` as an `uint8_t`.

Definition at line 330 of file platform-common.h.

6.8.2.25 #define COUNTOF(*a*)

Returns the number of entries in an array.

Definition at line 335 of file platform-common.h.

6.8.2.26 #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 350 of file platform-common.h.

6.8.2.27 #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 357 of file platform-common.h.

6.8.2.28 #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 364 of file platform-common.h.

6.8.2.29 #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 371 of file platform-common.h.

6.8.2.30 #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 372 of file platform-common.h.

6.8.2.31 #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 373 of file platform-common.h.

6.8.2.32 #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 380 of file platform-common.h.

6.8.2.33 #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 381 of file platform-common.h.

6.8.2.34 #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 382 of file platform-common.h.

6.8.2.35 #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 389 of file platform-common.h.

6.8.2.36 #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 390 of file platform-common.h.

6.8.2.37 #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 391 of file platform-common.h.

6.8.2.38 #define UNUSED_VAR(*x*)

Description:

Useful macro for avoiding compiler warnings related to unused function arguments or unused variables.

Definition at line 408 of file platform-common.h.

6.8.2.39 #define DEBUG_LEVEL

Set debug level based on whether DEBUG or DEBUG_OFF are defined.

Definition at line 422 of file platform-common.h.

6.9 Unix GCC Specific PLATFORM_HEADER Configuration

Macros

- #define HAL_HAS_INT64
- #define _HAL_USE_COMMON_PGM_
- #define BIGENDIAN_CPU
- #define ALIGNMENT(__alignmentBytes)
- #define WEAK(__symbol)
- #define _HAL_USE_COMMON_DIVMOD_
- #define PLATCOMMONOKTOINCLUDE

Master Variable Types

These are a set of typedefs to make the size of all variable declarations explicitly known.

- typedef bool boolean
- typedef unsigned char int8u
- typedef signed char int8s
- typedef unsigned short int16u
- typedef signed short int16s
- typedef unsigned int int32u
- typedef signed int int32s
- typedef unsigned long long int64u
- typedef signed long long int64s
- typedef unsigned long PointerType

Watchdog Prototypes

Define the watchdog macro and internal function to simply be stubs to satisfy those programs that have no HAL (i.e. scripted tests) and those that want to reference real HAL functions (simulation binaries and Unix host applications) we define both halResetWatchdog() and halInternalResetWatchdog(). The former is used by most of the scripted tests while the latter is used by simulation and real host applications.

- void halInternalResetWatchDog (void)
- #define halResetWatchdog()

6.9.1 Detailed Description

Compiler and Platform specific definitions and typedefs for the the Unix GCC compiler.

Note

ATOMIC and interrupt manipulation macros are defined to have no affect.
gcc.h should be included first in all source files by setting the preprocessor macro PLATFORM_HEADER to point to it. gcc.h automatically includes platform-common.h.

See Common PLATFORM_HEADER Configuration for common documentation.

See gcc.h for source code.

6.9.2 Macro Definition Documentation

6.9.2.1 #define HAL_HAS_INT64

8051 memory segments stubs.

Denotes that this platform supports 64-bit data-types.

Definition at line 79 of file gcc.h.

6.9.2.2 #define _HAL_USE_COMMON_PGM_

Use the Master Program Memory Declarations from platform-common.h.

Definition at line 84 of file gcc.h.

6.9.2.3 #define BIGENDIAN_CPU

A definition stating what the endianess of the platform is.

Definition at line 90 of file gcc.h.

6.9.2.4 #define ALIGNMENT(*__alignmentBytes*)

Provide a portable way to align data.

Definition at line 198 of file gcc.h.

6.9.2.5 #define WEAK(*__symbol*)

Provide a portable way to specify a symbol as weak.

Definition at line 204 of file gcc.h.

6.9.2.6 #define halResetWatchdog( )

Watchdog stub prototype.

Definition at line 222 of file gcc.h.

6.9.2.7 #define _HAL_USE_COMMON_DIVMOD_

Use the Divide and Modulus Operations from platform-common.h.

Include string.h for the C Standard Library memory routines used in platform-common.

Definition at line 236 of file gcc.h.

6.9.2.8 #define PLATCOMMONOKTOINCLUDE

Include platform-common.h last to pick up defaults and common definitions.

Definition at line 241 of file gcc.h.

6.9.3 Typedef Documentation

6.9.3.1 typedef bool boolean

A typedef to make the size of the variable explicitly known.

Definition at line 40 of file gcc.h.

6.9.3.2 typedef unsigned char int8u

A typedef to make the size of the variable explicitly known.

Definition at line 43 of file gcc.h.

6.9.3.3 typedef signed char int8s

A typedef to make the size of the variable explicitly known.

Definition at line 44 of file gcc.h.

6.9.3.4 typedef unsigned short int16u

A typedef to make the size of the variable explicitly known.

Definition at line 45 of file gcc.h.

6.9.3.5 typedef signed short int16s

A typedef to make the size of the variable explicitly known.

Definition at line 46 of file gcc.h.

6.9.3.6 typedef unsigned int int32u

A typedef to make the size of the variable explicitly known.

Definition at line 47 of file gcc.h.

6.9.3.7 typedef signed int int32s

A typedef to make the size of the variable explicitly known.

Definition at line 48 of file gcc.h.

6.9.3.8 typedef unsigned long long int64u

A typedef to make the size of the variable explicitly known.

Definition at line 49 of file gcc.h.

6.9.3.9 typedef signed long long int64s

A typedef to make the size of the variable explicitly known.

Definition at line 50 of file gcc.h.

6.9.3.10 typedef unsigned long PointerType

A typedef to make the size of the variable explicitly known.

Definition at line 51 of file gcc.h.

6.9.4 Function Documentation

6.9.4.1 void halInternalResetWatchDog (void)

Watchdog stub prototype.

6.10 Asynchronous Serial Host (ASH) Framework

Macros

- #define ashStopAckTimer(void)
- #define ashAckTimerIsRunning()
- #define ashAckTimerIsNotRunning()
- #define ashSetAckPeriod(msec)
- #define ashGetAckPeriod()
- #define ashSetAndStartAckTimer(msec)
- #define ASH_NR_TIMER_BIT
- #define ashStopNrTimer()
- #define ashNrTimerIsNotRunning()
- #define ASH_VERSION
- #define ASH_FLAG
- #define ASH_ESC
- #define ASH_XON
- #define ASH_XOFF
- #define ASH_SUB
- #define ASH_CAN
- #define ASH_WAKE
- #define ASH_FLIP
- #define ASH_MIN_DATA_FIELD_LEN
- #define ASH_MAX_DATA_FIELD_LEN
- #define ASH_MIN_DATA_FRAME_LEN
- #define ASH_MIN_FRAME_LEN
- #define ASH_MAX_FRAME_LEN
- #define ASH_CRC_LEN
- #define ASH_MIN_FRAME_WITH_CRC_LEN
- #define ASH_MAX_FRAME_WITH_CRC_LEN
- #define ASH_NCP_SHFRAME_RX_LEN
- #define ASH_NCP_SHFRAME_TX_LEN
- #define ASH_HOST_SHFRAME_RX_LEN
- #define ASH_HOST_SHFRAME_TX_LEN
- #define ASH_DFRAME_MASK
- #define ASH_CONTROL_DATA
- #define ASH_SHFRAME_MASK
- #define ASH_CONTROL_ACK
- #define ASH_CONTROL_NAK
- #define ASH_CONTROL_RST
- #define ASH_CONTROL_RSTACK
- #define ASH_CONTROL_ERROR
- #define ASH_ACKNUM_MASK
- #define ASH_ACKNUM_BIT
- #define ASH_RFLAG_MASK
- #define ASH_RFLAG_BIT
- #define ASH_NFLAG_MASK
- #define ASH_NFLAG_BIT
- #define ASH_PFLAG_MASK
- #define ASH_PFLAG_BIT

- #define ASH_FRMNUM_MASK
- #define ASH_FRMNUM_BIT
- #define ASH_GET_RFLAG(ctl)
- #define ASH_GET_NFLAG(ctl)
- #define ASH_GET_FRMNUM(ctl)
- #define ASH_GET_ACKNUM(ctl)
- #define ASH_FRAME_LEN_DATA_MIN
- #define ASH_FRAME_LEN_ACK
- #define ASH_FRAME_LEN_NAK
- #define ASH_FRAME_LEN_RST
- #define ASH_FRAME_LEN_RSTACK
- #define ASH_FRAME_LEN_ERROR
- #define MOD8(n)
- #define INC8(n)
- #define WITHIN_RANGE(lo, n, hi)

Functions

- uint8_t ashEncodeByte (uint8_t len, uint8_t byte, uint8_t ∗offset)
- EzspStatus ashDecodeByte (uint8_t byte, uint8_t ∗out, uint8_t ∗outLen)
- uint8_t ashRandomizeArray (uint8_t seed, uint8_t ∗buf, uint8_t len)
- void ashStartAckTimer (void)
- bool ashAckTimerHasExpired (void)
- void ashAdjustAckPeriod (bool expired)
- void ashStartNrTimer (void)
- bool ashNrTimerHasExpired (void)

Variables

- bool ashDecodeInProgress
- uint16_t ashAckTimer
- uint16_t ashAckPeriod
- uint8_t ashNrTimer

6.10.1 Detailed Description

Use the Asynchronous Serial Host (ASH) Framework interfaces on a host microcontroller when it communicates with an Ember chip via EZSP-UART.

See ash-common.h for source code.

See ash-protocol.h for source code.

6.10.2 Macro Definition Documentation

6.10.2.1 void ashStopAckTimer(*void*)

Stops and clears ashAckTimer.

Definition at line 98 of file ash-common.h.

6.10.2.2 #define ashAckTimerIsRunning( )

Indicates whether or not ashAckTimer is currently running. The timer may be running even if expired.

Definition at line 104 of file ash-common.h.

6.10.2.3 #define ashAckTimerIsNotRunning( )

Indicates whether or not ashAckTimer is currently running. The timer may be running even if expired.

Definition at line 110 of file ash-common.h.

6.10.2.4 #define ashSetAckPeriod(*msec*)

Sets the acknowledgement timer period (in msec) and stops the timer.

Definition at line 140 of file ash-common.h.

6.10.2.5 #define ashGetAckPeriod( )

Returns the acknowledgement timer period (in msec).

Definition at line 146 of file ash-common.h.

6.10.2.6 #define ashSetAndStartAckTimer(*msec*)

Sets the acknowledgement timer period (in msec), and starts the timer running.

Definition at line 151 of file ash-common.h.

6.10.2.7 #define ASH_NR_TIMER_BIT

Definition at line 155 of file ash-common.h.

6.10.2.8 #define ashStopNrTimer( )

Stops the Not Ready timer.

Definition at line 168 of file ash-common.h.

6.10.2.9 #define ashNrTimerIsNotRunning( )

Indicates whether or not ashNrTimer is currently running.

Definition at line 180 of file ash-common.h.

6.10.2.10 #define ASH_VERSION

Definition at line 21 of file ash-protocol.h.

6.10.2.11 #define ASH_FLAG

frame delimiter

Definition at line 25 of file ash-protocol.h.

6.10.2.12 #define ASH_ESC

byte stuffing escape byte

Definition at line 26 of file ash-protocol.h.

6.10.2.13 #define ASH_XON

flow control byte - means resume transmission

Definition at line 27 of file ash-protocol.h.

6.10.2.14 #define ASH_XOFF

flow control byte - means suspend transmission

Definition at line 28 of file ash-protocol.h.

6.10.2.15 #define ASH_SUB

replaces bytes w framing, overrun or overflow errors

Definition at line 29 of file ash-protocol.h.

6.10.2.16 #define ASH_CAN

frame cancel byte

Definition at line 30 of file ash-protocol.h.

6.10.2.17 #define ASH_WAKE

wake signal byte (also means NCP data pending)

Definition at line 34 of file ash-protocol.h.

6.10.2.18 #define ASH_FLIP

XOR mask used in byte stuffing

Definition at line 37 of file ash-protocol.h.

6.10.2.19 #define ASH_MIN_DATA_FIELD_LEN

Definition at line 41 of file ash-protocol.h.

6.10.2.20 #define ASH_MAX_DATA_FIELD_LEN

Definition at line 42 of file ash-protocol.h.

6.10.2.21 #define ASH_MIN_DATA_FRAME_LEN

Definition at line 43 of file ash-protocol.h.

6.10.2.22 #define ASH_MIN_FRAME_LEN

Definition at line 44 of file ash-protocol.h.

6.10.2.23 #define ASH_MAX_FRAME_LEN

Definition at line 45 of file ash-protocol.h.

6.10.2.24 #define ASH_CRC_LEN

Definition at line 46 of file ash-protocol.h.

6.10.2.25 #define ASH_MIN_FRAME_WITH_CRC_LEN

Definition at line 47 of file ash-protocol.h.

6.10.2.26 #define ASH_MAX_FRAME_WITH_CRC_LEN

Definition at line 48 of file ash-protocol.h.

6.10.2.27 #define ASH_NCP_SHFRAME_RX_LEN

longest non-data frame received

Definition at line 51 of file ash-protocol.h.

6.10.2.28 #define ASH_NCP_SHFRAME_TX_LEN

longest non-data frame sent

Definition at line 52 of file ash-protocol.h.

6.10.2.29 #define ASH_HOST_SHFRAME_RX_LEN

longest non-data frame received

Definition at line 53 of file ash-protocol.h.

6.10.2.30 #define ASH_HOST_SHFRAME_TX_LEN

longest non-data frame sent

Definition at line 54 of file ash-protocol.h.

6.10.2.31 #define ASH_DFRAME_MASK

Definition at line 75 of file ash-protocol.h.

6.10.2.32 #define ASH_CONTROL_DATA

Definition at line 76 of file ash-protocol.h.

6.10.2.33 #define ASH_SHFRAME_MASK

Definition at line 78 of file ash-protocol.h.

6.10.2.34 #define ASH_CONTROL_ACK

Definition at line 79 of file ash-protocol.h.

6.10.2.35 #define ASH_CONTROL_NAK

Definition at line 80 of file ash-protocol.h.

6.10.2.36 #define ASH_CONTROL_RST

Definition at line 81 of file ash-protocol.h.

6.10.2.37 #define ASH_CONTROL_RSTACK

Definition at line 82 of file ash-protocol.h.

6.10.2.38 #define ASH_CONTROL_ERROR

Definition at line 83 of file ash-protocol.h.

6.10.2.39 #define ASH_ACKNUM_MASK

acknowledge frame number

Definition at line 85 of file ash-protocol.h.

6.10.2.40 #define ASH_ACKNUM_BIT

Definition at line 86 of file ash-protocol.h.

6.10.2.41 #define ASH_RFLAG_MASK

retransmitted frame flag

Definition at line 87 of file ash-protocol.h.

6.10.2.42 #define ASH_RFLAG_BIT

Definition at line 88 of file ash-protocol.h.

6.10.2.43 #define ASH_NFLAG_MASK

receiver not ready flag

Definition at line 89 of file ash-protocol.h.

6.10.2.44 #define ASH_NFLAG_BIT

Definition at line 90 of file ash-protocol.h.

6.10.2.45 #define ASH_PFLAG_MASK

flag reserved for future use

Definition at line 91 of file ash-protocol.h.

6.10.2.46 #define ASH_PFLAG_BIT

Definition at line 92 of file ash-protocol.h.

6.10.2.47 #define ASH_FRMNUM_MASK

DATA frame number

Definition at line 93 of file ash-protocol.h.

6.10.2.48 #define ASH_FRMNUM_BIT

Definition at line 94 of file ash-protocol.h.

6.10.2.49 #define ASH_GET_RFLAG(*ctl*)

Definition at line 95 of file ash-protocol.h.

6.10.2.50 #define ASH_GET_NFLAG(*ctl*)

Definition at line 96 of file ash-protocol.h.

6.10.2.51 #define ASH_GET_FRMNUM(*ctl*)

Definition at line 97 of file ash-protocol.h.

6.10.2.52 #define ASH_GET_ACKNUM(*ctl*)

Definition at line 98 of file ash-protocol.h.

6.10.2.53 #define ASH_FRAME_LEN_DATA_MIN

Definition at line 102 of file ash-protocol.h.

6.10.2.54 #define ASH_FRAME_LEN_ACK

Definition at line 103 of file ash-protocol.h.

6.10.2.55 #define ASH_FRAME_LEN_NAK

Definition at line 104 of file ash-protocol.h.

6.10.2.56 #define ASH_FRAME_LEN_RST

Definition at line 105 of file ash-protocol.h.

6.10.2.57 #define ASH_FRAME_LEN_RSTACK

Definition at line 106 of file ash-protocol.h.

6.10.2.58 #define ASH_FRAME_LEN_ERROR

Definition at line 107 of file ash-protocol.h.

6.10.2.59 #define MOD8(*n*)

mask to frame number modulus

Definition at line 110 of file ash-protocol.h.

6.10.2.60 #define INC8(*n*)

increment in frame number modulus

Definition at line 111 of file ash-protocol.h.

6.10.2.61 #define WITHIN_RANGE(*lo*, *n*, *hi*)

Definition at line 113 of file ash-protocol.h.

6.10.3 Function Documentation

6.10.3.1 uint8_t ashEncodeByte (uint8_t *len,* uint8_t *byte,* uint8_t ∗ *offset*)

Builds an ASH frame. Byte stuffs the control and data fields as required, computes and appends the CRC and adds the ending flag byte. Called with the next byte to encode, this function may return several output bytes before it's ready for the next byte.

Parameters

len	new frame flag / length of the frame to be encoded. A non-zero value begins a new frame, so all subsequent calls must use zero. The length includes control byte and data field, but not the flag or crc. This function does not validate the length.
byte	the next byte of data to add to the frame. Note that in general the same byte of data may have to be passed more than once as escape bytes, the CRC and the end flag byte are output.
offset	pointer to the offset of the next input byte. (If the frame data is the array data[], the next byte would be data[offset].) Is set to 0 when starting a new frame (ie, len is non-zero). Is set to 0xFF when the last byte of the frame is returned.

Returns

next encoded output byte in frame.

6.10.3.2 EzspStatus ashDecodeByte (uint8_t *byte,* uint8_t ∗ *out,* uint8_t ∗ *outLen*)

Decodes and validates an ASH frame. Data is passed to it one byte at a time. Decodes byte stuffing, checks crc, finds the end flag and (if enabled) terminates the frame early on CAN or SUB bytes. The number of bytes output will not exceed the max valid frame length, which does not include the flag or the crc.

Parameters

byte	the next byte of data to add to the frame
out	pointer to where to write an output byte
outLen	number of bytes output so far

Returns

status of frame decoding

- ::EZSP_SUCCESS
- ::EZSP_ASH_IN_PROGRESS
- ::EZSP_ASH_CANCELLED
- ::EZSP_ASH_BAD_CRC
- ::EZSP_ASH_COMM_ERROR
- ::EZSP_ASH_TOO_SHORT
- ::EZSP_ASH_TOO_LONG

6.10.3.3 uint8_t ashRandomizeArray (uint8_t *seed,* uint8_t ∗ *buf,* uint8_t *len*)

Randomizes array contents by XORing with an 8-bit pseudo random sequence. This reduces the likelihood that byte-stuffing will greatly increase the size of the payload. (This could happen if a DATA frame contained repeated instances of the same reserved byte value.)

Parameters

seed	zero initializes the random sequence a non-zero value continues from a previous invocation
buf	pointer to the array whose contents will be randomized
len	number of bytes in the array to modify

Returns

last value of the sequence. If a buffer is processed in two or more chunks, as with linked buffers, this value should be passed back as the value of the seed argument

6.10.3.4 void ashStartAckTimer (void)

Sets ashAckTimer to the specified period and starts it running.

6.10.3.5 bool ashAckTimerHasExpired (void)

Indicates whether or not ashAckTimer has expired. If the timer is stopped then it is not expired.

6.10.3.6 void ashAdjustAckPeriod (bool *expired*)

Adapts the acknowledgement timer period to the observed ACK delay. If the timer is not running, it does nothing. If the timer has expired, the timeourt period is doubled. If the timer has not expired, the elapsed time is fed into simple IIR filter: T[n+1] = (7∗T[n] + elapsedTime) / 8 The timeout period, ashAckPeriod, is limited such that: ASH_xxx_T-IME_DATA_MIN <= ashAckPeriod <= ASH_xxx_TIME_DATA_MAX, where xxx is either HOST or NCP.

The acknowledgement timer is always stopped by this function.

Parameters

expired	true if timer has expired

6.10.3.7 void ashStartNrTimer (void)

Starts the Not Ready timer.

On the host, this times nFlag refreshing when the host doesn't have room for callbacks for a prolonged period.

On the NCP, if this times out the NCP resumes sending callbacks.

6.10.3.8 bool ashNrTimerHasExpired (void)

Tests whether the Not Ready timer has expired or has stopped. If expired, it is stopped.

Returns

true if the Not Ready timer has expired or stopped

6.10.4 Variable Documentation

6.10.4.1 bool ashDecodeInProgress

6.10.4.2 uint16_t ashAckTimer

6.10.4.3 uint16_t ashAckPeriod

6.10.4.4 uint8_t ashNrTimer

6.11 EM2xx-compatible Resets

- #define EM2XX_RESET_UNKNOWN
- #define EM2XX_RESET_EXTERNAL
- #define EM2XX_RESET_POWERON
- #define EM2XX_RESET_WATCHDOG
- #define EM2XX_RESET_ASSERT
- #define EM2XX_RESET_BOOTLOADER
- #define EM2XX_RESET_SOFTWARE

6.11.1 Detailed Description

6.11.2 Macro Definition Documentation

6.11.2.1 #define EM2XX_RESET_UNKNOWN

EM2xx-compatible reset code returned by halGetEm2xxResetInfo()

Definition at line 17 of file em2xx-reset-defs.h.

6.11.2.2 #define EM2XX_RESET_EXTERNAL

EM2xx-compatible reset code returned by halGetEm2xxResetInfo()

Definition at line 18 of file em2xx-reset-defs.h.

6.11.2.3 #define EM2XX_RESET_POWERON

EM2xx-compatible reset code returned by halGetEm2xxResetInfo()

Definition at line 19 of file em2xx-reset-defs.h.

6.11.2.4 #define EM2XX_RESET_WATCHDOG

EM2xx-compatible reset code returned by halGetEm2xxResetInfo()

Definition at line 20 of file em2xx-reset-defs.h.

6.11.2.5 #define EM2XX_RESET_ASSERT

EM2xx-compatible reset code returned by halGetEm2xxResetInfo()

Definition at line 21 of file em2xx-reset-defs.h.

6.11.2.6 #define EM2XX_RESET_BOOTLOADER

EM2xx-compatible reset code returned by halGetEm2xxResetInfo()

Definition at line 22 of file em2xx-reset-defs.h.

6.11.2.7 #define EM2XX_RESET_SOFTWARE

EM2xx-compatible reset code returned by halGetEm2xxResetInfo()

Definition at line 23 of file em2xx-reset-defs.h.

6.12 System Timer

Macros

- #define halIdleForMilliseconds(duration)

Functions

- uint16_t halInternalStartSystemTimer (void)
- uint16_t halCommonGetInt16uMillisecondTick (void)
- uint32_t halCommonGetInt32uMillisecondTick (void)
- uint16_t halCommonGetInt16uQuarterSecondTick (void)
- EmberStatus halSleepForQuarterSeconds (uint32_t ∗duration)
- EmberStatus halSleepForMilliseconds (uint32_t ∗duration)
- EmberStatus halCommonIdleForMilliseconds (uint32_t ∗duration)

6.12.1 Detailed Description

Functions that provide access to the system clock. A single system tick (as returned by halCommonGetInt16uMillisecondTick() and halCommonGetInt32uMillisecondTick()) is approximately 1 millisecond.

- When used with a 32.768kHz crystal, the system tick is 0.976 milliseconds.

- When used with a 3.6864MHz crystal, the system tick is 1.111 milliseconds.

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.

6.12.2 Macro Definition Documentation

6.12.2.1 #define halIdleForMilliseconds(*duration*)

Definition at line 193 of file system-timer.h.

6.12.3 Function Documentation

6.12.3.1 uint16_t halInternalStartSystemTimer (void)

Initializes the system tick.

Returns

Time to update the async registers after RTC is started (units of 100 microseconds).

6.12.3.2 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.

6.12.3.3 uint32_t halCommonGetInt32uMillisecondTick (void)

Returns the current system time in system ticks, as a 32-bit value.

EmberStack Usage:

Unused, implementation optional.

Returns

The least significant 32 bits of the current system time, in system ticks.

6.12.3.4 uint16_t halCommonGetInt16uQuarterSecondTick (void)

Returns the current system time in quarter second ticks, as a 16-bit value.

EmberStack Usage:

Unused, implementation optional.

Returns

The least significant 16 bits of the current system time, in system ticks multiplied by 256.

6.12.3.5 EmberStatus halSleepForQuarterSeconds (uint32_t * *duration*)

Uses the system timer to enter ::SLEEPMODE_WAKETIMER for approximately the specified amount of time (provided in quarter seconds).

This function returns EMBER_SUCCESS and the duration parameter is decremented to 0 after sleeping for the specified amount of time. If an interrupt occurs that brings the chip out of sleep, the function returns EMBER_SLEEP_INTERRUPTED and the duration parameter reports the amount of time remaining out of the original request.

Note

This routine always enables interrupts.
The maximum sleep time of the hardware is limited on AVR-based platforms to 8 seconds, on EM2XX-based platforms to 64 seconds, and on EM35x platforms to 48.5 days. Any sleep duration greater than this limit will wake up briefly (e.g. 16 microseconds) to reenable another sleep cycle.

The EM2xx has a 16 bit sleep timer, which normally runs at 1024Hz. In order to support long sleep durations, the chip will periodically wake up to manage a larger timer in software. This periodic wakeup is normally triggered once every 32 seconds. However, this period can be extended to once every 2.275 hours by building with **ENABLE_LONG_SLEEP_CYCLES** defined. This definition enables the use of a prescaler when sleeping for more than 63 seconds at a time. However, this define also imposes the following limitations:

1. The chip may only wake up from the sleep timer. (External GPIO wake events may not be used)
2. Each time a sleep cycle is performed, a loss of accuracy up to +/-750ms will be observed in the system timer.

EmberStack Usage:

Unused, implementation optional.

Parameters

duration	The amount of time, expressed in quarter seconds, that the micro should be placed into ::SLEEPMODE_WAKETIMER. When the function returns, this parameter provides the amount of time remaining out of the original sleep time request (normally the return value will be 0).

Returns

An EmberStatus value indicating the success or failure of the command.

6.12.3.6 EmberStatus halSleepForMilliseconds (uint32_t * *duration*)

Uses the system timer to enter ::SLEEPMODE_WAKETIMER for approximately the specified amount of time (provided in milliseconds). Note that since the system timer ticks at a rate of 1024Hz, a second is comprised of 1024 milliseconds in this function.

This function returns EMBER_SUCCESS and the duration parameter is decremented to 0 after sleeping for the specified amount of time. If an interrupt occurs that brings the chip out of sleep, the function returns EMBER_SLEEP_INTERRUPTED and the duration parameter reports the amount of time remaining out of the original request.

Note

This routine always enables interrupts.
This function is not implemented on AVR-based platforms.
Sleep durations less than 3 milliseconds are not allowed on on EM2XX-based platforms. Any attempt to sleep for less than 3 milliseconds on EM2XX-based platforms will cause the function to immediately exit without sleeping and return EMBER_SLEEP_INTERRUPTED.
The maximum sleep time of the hardware is limited on EM2XX-based platforms to 32 seconds. Any sleep duration greater than this limit will wake up briefly (e.g. 16 microseconds) to reenable another sleep cycle. Due to this limitation, this function should not be used with durations within 3 milliseconds of a multiple 32 seconds. The short sleep cycle that results from such durations is not handled reliably by the

system timer on EM2XX-based platforms. If a sleep duration within 3 milliseconds of a multiple of 32 seconds is desired, halSleepForQuarterSeconds should be used.

EmberStack Usage:

Unused, implementation optional.

Parameters

duration	The amount of time, expressed in milliseconds (1024 milliseconds = 1 second), that the micro should be placed into ::SLEEPMODE_WAKETIMER. When the function returns, this parameter provides the amount of time remaining out of the original sleep time request (normally the return value will be 0).

Returns

An EmberStatus value indicating the success or failure of the command.

6.12.3.7 EmberStatus halCommonIdleForMilliseconds (uint32_t * *duration*)

Uses the system timer to enter ::SLEEPMODE_IDLE for approximately the specified amount of time (provided in milliseconds).

This function returns EMBER_SUCCESS and the duration parameter is decremented to 0 after idling for the specified amount of time. If an interrupt occurs that brings the chip out of idle, the function returns EMBER_SLEEP_INTERRUPTED and the duration parameter reports the amount of time remaining out of the original request.

Note

This routine always enables interrupts.

EmberStack Usage:

Unused, implementation optional.

Parameters

duration	The amount of time, expressed in milliseconds, that the micro should be placed into ::SLEEPMODE_IDLE. When the function returns, this parameter provides the amount of time remaining out of the original idle time request (normally the return value will be 0).

Returns

An EmberStatus value indicating the success or failure of the command.

6.13 HAL Utilities

Modules

- Cyclic Redundancy Code (CRC)

6.13.1 Detailed Description

6.14 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)

6.14.1 Detailed Description

Functions that provide access to cyclic redundancy code (CRC) calculation. See crc.h for source code.

6.14.2 Macro Definition Documentation

6.14.2.1 #define INITIAL_CRC

Definition at line 49 of file crc.h.

6.14.2.2 #define CRC32_START

Definition at line 50 of file crc.h.

6.14.2.3 #define CRC32_END

Definition at line 51 of file crc.h.

6.14.3 Function Documentation

6.14.3.1 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

newByte	The new byte to be run through CRC.
prevResult	The previous CRC result.

Returns

The new CRC result.

6.14.3.2 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

newByte	The new byte to be run through CRC.
prevResult	The previous CRC result.

Returns

The new CRC result.

6.15 Application Utilities API Reference

Modules

- Forming and Joining Networks
- Command Interpreter 2
- ZigBee Device Object (ZDO) Information
- Message Fragmentation
- Network Manager
- Serial Communication
- ASH Application Utility

6.15.1 Detailed Description

6.16 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 ∗network-Found, 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

6.16.1 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.

6.16.2 Macro Definition Documentation

6.16.2.1 #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.

6.16.2.2 #define NETWORK_STORAGE_SIZE_SHIFT

Log_base2 of NETWORK_STORAGE_SIZE.

Definition at line 72 of file form-and-join.h.

6.16.2.3 #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.

6.16.3 Function Documentation

6.16.3.1 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

channelMask	
duration	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.

6.16.3.2 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

channelMask	
extendedPan-Id	

Returns

EMBER_LIBRARY_NOT_PRESENT if the form and join library is not available.

6.16.3.3 EmberStatus emberScanForNextJoinableNetwork (void)

See emberScanForJoinableNetwork().

6.16.3.4 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.

6.16.3.5 bool emberFormAndJoinCanContinueJoinableNetworkScan (void)

Returns true if and only if the application can continue a joinable network scan by calling emberScanForNextJoinableNetwork(). See emberScanForJoinableNetwork().

6.16.3.6 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 emberScan-ForUnusedPanId().

Parameters

panId	
channel	

6.16.3.7 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

network-Found	
lqi	The lqi value of the received beacon.
rssi	The rssi value of the received beacon.

6.16.3.8 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 ember-ScanForNextJoinableNetwork().
- 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

status	

6.16.3.9 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.

6.16.3.10 bool emberFormAndJoinNetworkFoundHandler (EmberZigbeeNetwork ∗ *networkFound,* uint8_t *lqi,* int8_t *rssi*)

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.

6.16.3.11 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.

6.16.3.12 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.

6.16.3.13 void emberFormAndJoinTaskInit (void)

When processor idling is desired on the SOC, this must be called to properly initialize the form and join library.

6.16.3.14 void emberFormAndJoinRunTask (void)

When processor idling is desired on the SOC, this should be called regularly instead of emberFormAndJoinTick()

6.16.3.15 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.

6.16.4 Variable Documentation

6.16.4.1 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.

6.17 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_NUMBE-R_OF_ARGUMENTS,
  EMBER_CMD_ERR_ARGUMENT_OUT_OF_RANGE, EMBER_CMD_ERR_A-RGUMENT_SYNTAX_ERROR, EMBER_CMD_ERR_STRING_TOO_LONG, E-MBER_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 destina-tion)
- #define emberCopyKeyArgument(index, keyDataPointer)
- #define emberCopyEui64Argument(index, eui64)
- #define emberGetEui64Argument(index, eui64)

6.17.1 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 initalize, 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.

6.17.2 Macro Definition Documentation

6.17.2.1 #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.

6.17.2.2 #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.

6.17.2.3 #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.

6.17.2.4 #define MAX_TOKEN_COUNT

Definition at line 122 of file command-interpreter2.h.

6.17.2.5 #define emberCommandEntryAction(*name*, *action*, *argumentTypes*, *description*)

Definition at line 187 of file command-interpreter2.h.

6.17.2.6 #define emberCommandEntryActionWithDetails(*name*, *action*, *argumentTypes*, *description*, *argumentDescriptionArray*)

Definition at line 190 of file command-interpreter2.h.

6.17.2.7 #define emberCommandEntrySubMenu(*name*, *subMenu*, *description*)

Definition at line 198 of file command-interpreter2.h.

6.17.2.8 #define emberCommandEntryTerminator( )

Definition at line 202 of file command-interpreter2.h.

6.17.2.9 #define EMBER_COMMAND_INTERPRETER_CONFIGURATION_ECHO

Definition at line 243 of file command-interpreter2.h.

6.17.2.10 #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.

6.17.2.11 #define emberCopyEui64Argument(*index,* *eui64*)

A convenience macro for copying eui64 arguments to an EmberEUI64.

Definition at line 336 of file command-interpreter2.h.

6.17.2.12 #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.

6.17.2.13 #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.

6.17.2.14 #define emberCommandInterpreterEchoOn( )

Turn echo of command line on.

Definition at line 389 of file command-interpreter2.h.

6.17.2.15 #define emberCommandInterpreterEchoOff( )

Turn echo of command line off.

Definition at line 395 of file command-interpreter2.h.

6.17.2.16 #define emberCommandInterpreterIsEchoOn( )

Returns true if echo is on, false otherwise.

Definition at line 401 of file command-interpreter2.h.

6.17.3 Typedef Documentation

6.17.3.1 typedef void(* CommandAction)(void)

Definition at line 124 of file command-interpreter2.h.

6.17.4 Enumeration Type Documentation

6.17.4.1 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.

6.17.5 Function Documentation

6.17.5.1 uint8_t emberCommandArgumentCount (void)

Returns the number of arguments for the current command.

6.17.5.2 uint32_t emberUnsignedCommandArgument (uint8_t *argNum*)

Retrieves unsigned integer arguments.

6.17.5.3 int32_t emberSignedCommandArgument (uint8_t *argNum*)

Retrieves signed integer arguments.

6.17.5.4 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.

6.17.5.5 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.

6.17.5.6 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].

6.17.5.7 const char∗ emberCommandName (void)

A convenience macro for copying security key arguments to an EmberKeyData pointer.

6.17.5.8 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.

6.17.5.9 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.

6.17.5.10 void emberCommandReaderSetDefaultBase (uint8_t *base*)

6.17.5.11 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.

6.17.5.12 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().

6.17.5.13 void emberPrintCommandUsage (EmberCommandEntry * *entry*)

6.17.5.14 void emberPrintCommandUsageNotes (void)

6.17.5.15 void emberPrintCommandTable (void)

6.17.5.16 void emberCommandClearBuffer (void)

6.17.5.17 void emberCommandReaderInit (void)

Initialize the command interpreter.

6.17.5.18 bool emberProcessCommandString (uint8_t * *input,* uint8_t *sizeOrPort*)

Process the given string as a command.

6.17.6 Variable Documentation

6.17.6.1 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.

6.17.6.2 EmberCommandEntry emberCommandTable[ ]

6.17.6.3 uint8_t emberCommandInterpreter2Configuration

Configuration byte.

6.18 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, Ember-ApsOption options)

Binding Manager Functions

- EmberStatus ezspEndDeviceBindRequest (EmberNodeId localNodeId, EmberEU-I64 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 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, Ember-MulticastId groupAddress, uint8_t destinationEndpoint, EmberApsOption options)
- EmberStatus emberUnbindRequest (EmberNodeId target, EmberEUI64 source, uint8-_t sourceEndpoint, uint16_t clusterId, uint8_t type, EmberEUI64 destination, Ember-MulticastId 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)

6.18.1 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.
  }
}
```

6.18.2 Macro Definition Documentation

6.18.2.1 #define ZDO_MESSAGE_OVERHEAD

ZDO messages start with a sequence number.

Definition at line 16 of file zigbee-device-common.h.

6.18.3 Function Documentation

6.18.3.1 EmberStatus emberNetworkAddressRequest (EmberEUI64 *target,* bool *reportKids,* uint8_t *childStartIndex*)

Request the 16 bit network address of a node whose EUI64 is known.

Parameters

target	The EUI64 of the node.
reportKids	true to request that the target list their children in the response.
childStartIndex	The index of the first child to list in the response. Ignored if `reportKids` is false.

Returns

An EmberStatus value.

- EMBER_SUCCESS - The request was transmitted successfully.
- EMBER_NO_BUFFERS - Insuffient 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.

6.18.3.2 EmberStatus emberIeeeAddressRequest (EmberNodeId *target*, bool *reportKids*, uint8_t *childStartIndex*, EmberApsOption *options*)

Request the EUI64 of a node whose 16 bit network address is known.

Parameters

target	The network address of the node.
reportKids	true to request that the target list their children in the response.
childStart-Index	The index of the first child to list in the response. Ignored if reportKids is false.
options	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

6.18.3.3 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

target	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.
profile	The application profile to match.
inCount	The number of input clusters. To not match any input clusters, set this value to 0.
outCount	The number of output clusters. To not match any output clusters, set this value to 0.
inClusters	The list of input clusters.
outClusters	The list of output clusters.
options	The options to use when sending the unicast request. See emberSend-Unicast() for a description. This parameter is ignored if the target is a broadcast address.

Returns

An EmberStatus value. EMBER_SUCCESS, EMBER_NO_BUFFERS, EMBER_N-ETWORK_DOWN or EMBER_NETWORK_BUSY.

6.18.3.4 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

localNodeId	The node ID of the local device.
localEui64	The EUI64 of the local device.
endpoint	The endpoint to be bound.
profile	The application profile of the endpoint.
inCount	The number of input clusters.
outCount	The number of output clusters.
inClusters	The list of input clusters.
outClusters	The list of output clusters.
options	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.

6.18.3.5 EmberNodeId ezspDecodeAddressResponse (uint8_t ∗ *response,* EmberEUI64 *eui64Return*)

Extracts the EUI64 and the node ID from an address response message.

Parameters

response	The received ZDO message with cluster ID NETWORK_ADDRESS_R-ESPONSE or IEEE_ADDRESS_RESPONSE.
eui64Return	The EUI64 from the response is copied here.

Returns

Returns the node ID from the response if the response status was EMBER_ZDP_SU-CCESS. Otherwise, returns EMBER_NULL_NODE_ID.

6.18.3.6 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

target	The node whose node descriptor is desired.
options	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.

6.18.3.7 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

target	The node whose power descriptor is desired.
options	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.

6.18.3.8 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

target	The node of interest.
target-Endpoint	The endpoint on the target node whose simple descriptor is desired.
options	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.

6.18.3.9 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

target	The node whose active endpoints are desired.
options	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.

6.18.3.10 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

target	The node on which the binding will be created.
source	The source EUI64 in the binding entry.
source-Endpoint	The source endpoint in the binding entry.
clusterId	The cluster ID in the binding entry.
type	The type of binding, either UNICAST_BINDING, MULTICAST_BIND-ING, 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.
destination	The destination EUI64 in the binding entry for UNICAST_BINDING or UNICAST_MANY_TO_ONE_BINDING.
group-Address	The group address for the MULTICAST_BINDING.
destination-Endpoint	The destination endpoint in the binding entry for the UNICAST_BINDI-NG or UNICAST_MANY_TO_ONE_BINDING.
options	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.

6.18.3.11 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

target	The node on which the binding will be removed.
source	The source EUI64 in the binding entry.
source-Endpoint	The source endpoint in the binding entry.
clusterId	The cluster ID in the binding entry.
type	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.
destination	The destination EUI64 in the binding entry for the UNICAST_BINDING or UNICAST_MANY_TO_ONE_BINDING.
group-Address	The group address for the MULTICAST_BINDING.
destination-Endpoint	The destination endpoint in the binding entry for the UNICAST_BINDING or UNICAST_MANY_TO_ONE_BINDING.
options	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

6.18.3.12 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

target	The node whose LQI table is desired.
startIndex	The index of the first neighbor to include in the response.
options	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.

6.18.3.13 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

target	The node whose routing table is desired.
startIndex	The index of the first route entry to include in the response.
options	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.

6.18.3.14 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

target	The node whose binding table is desired.
startIndex	The index of the first binding entry to include in the response.
options	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.

6.18.3.15 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

target	The node which will remove the device.
device-Address	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.
leave-Request-Flags	A bitmask of leave options. Include LEAVE_REQUEST_REMOVE_-CHILDREN_FLAG if the target is to remove their children and/or LE-AVE_REQUEST_REJOIN_FLAG if the target is to rejoin the network immediately after leaving.
options	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.

6.18.3.16 EmberStatus emberPermitJoiningRequest (EmberNodeId *target,* uint8_t *duration,* uint8_t *authentication,* EmberApsOption *options*)

Request the specified node to allow or disallow association.

Parameters

target	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.
duration	A value of 0x00 disables joining. A value of 0xFF enables joining. Any other value enables joining for that number of seconds.
authentica-tion	Controls Trust Center authentication behavior.
options	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_N-ETWORK_DOWN or EMBER_NETWORK_BUSY.

6.18.3.17 void emberSetZigDevRequestRadius (uint8_t *radius*)

Change the default radius for broadcast ZDO requests.

Parameters

radius	The radius to be used for future ZDO request broadcasts.

6.18.3.18 uint8_t emberGetZigDevRequestRadius (void)

Retrieve the default radius for broadcast ZDO requests.

Returns

The radius to be used for future ZDO request broadcasts.

6.18.3.19 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

6.18.3.20 uint8_t emberGetLastAppZigDevRequestSequence (void)

Provide access to the application ZDO transaction sequence number for last request.

Returns

Last application ZDO transaction sequence number used

6.19 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)

6.19.1 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 ezspFragment-Init().

To send a long message, the application calls ezspFragmentSendUnicast(). The application must add a call to ezspFragmentMessageSent() at the start of its ezspMessageSent-Handler(). 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 ezspFragmentIncoming-Message() at the start of its ezspIncomingMessageHandler(). If ezspFragmentIncoming-Message() returns true, the fragmentation code has handled the message and the application must not process it further. The application must also call ezspFragmentTick() regularly.

6.19.2 Function Documentation

6.19.2.1 void ezspFragmentInit (uint16_t *receiveBufferLength,* uint8_t ∗ *receiveBuffer*)

Initialize variables and buffers used for sending and receiving long messages. This functions reads the values of ::EZSP_CONFIG_MAX_HOPS and ::EZSP_CONFIG_FRAGM-ENT_WINDOW_SIZE. The application must set these values before calling this function.

Parameters

receive-BufferLength	The length of receiveBuffer. Incoming messages longer than this will be dropped.
receiveBuffer	The buffer used to reassemble incoming long messages. Once the message is complete, this buffer will be passed back to the application by ezsp-FragmentIncomingMessage().

6.19.2.2 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

type	Specifies the outgoing message type. Must be one of EMBER_OUTGO-ING_DIRECT, EMBER_OUTGOING_VIA_ADDRESS_TABLE, or E-MBER_OUTGOING_VIA_BINDING.
indexOr-Destination	Depending on the type of addressing used, this is either the EmberNode-Id of the destination, an index into the address table, or an index into the binding table.
apsFrame	The APS frame for the message.
max-Fragment-Size	The message will be broken into blocks no larger than this.
message-Length	The length of the messageContents parameter in bytes.
message-Contents	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.

6.19.2.3 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 ezspSet-SourceRoute() in it.

The application must define EZSP_APPLICATION_HAS_FRAGMENT_SOURCE_RO-UTE_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.

6.19.2.4 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

apsFrame	The APS frame passed to ezspMessageSentHandler().
status	The status passed to ezspMessageSentHandler().

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.

6.19.2.5 void ezspFragmentMessageSentHandler (EmberStatus *status*)

The fragmentation code calls this application-defined handler when it finishes sending a long message.

Parameters

status	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.

6.19.2.6 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

apsFrame	The APS frame passed to ezspIncomingMessageHandler().
sender	The sender passed to ezspIncomingMessageHandler().
messageLength	A pointer to the message length passed to ezspIncomingMessageHandler().

message-Contents	A pointer to the message contents passed to ezspIncomingMessage-Handler().

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.

6.19.2.7 void ezspFragmentTick (void)

Used by the fragmentation code to time incoming blocks. The application must call this function regularly.

6.20 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)

6.20.1 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.

6.20.2 Macro Definition Documentation

6.20.2.1 #define NM_WARNING_LIMIT

Definition at line 97 of file network-manager.h.

6.20.2.2 #define NM_WINDOW_SIZE

Definition at line 101 of file network-manager.h.

6.20.2.3 #define NM_CHANNEL_MASK

Definition at line 107 of file network-manager.h.

6.20.2.4 #define NM_WATCHLIST_SIZE

Definition at line 113 of file network-manager.h.

6.20.3 Function Documentation

6.20.3.1 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.

6.20.3.2 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

apsFrame	
message-Length	
message	

6.20.3.3 EmberStatus nmUtilChangeChannelRequest (void)

Chooses a new channel and broadcasts a ZDO channel change request.

6.21 Serial Communication

Macros

- #define SERIAL_PORT_RAW
- #define SERIAL_PORT_CLI

Functions

- void emberSerialSetPrompt (const char ∗thePrompt)
- void emberSerialCleanup (void)
- int emberSerialGetInputFd (uint8_t port)
- void emberSerialSendReadyToRead (uint8_t port)
- void emberSerialCommandCompletionInit (EmberCommandEntry ∗listOfCommands)
- 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 subsequentByte-Timeout)
- 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 format-String,...)
- 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, P-GM_P formatString, va_list ap)
- EmberStatus emberSerialWriteData (uint8_t port, uint8_t ∗data, uint8_t length)
- EmberStatus emberSerialWriteBuffer (uint8_t port, EmberMessageBuffer buffer, uint8-_t start, 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)

6.21.1 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 E-MBER_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_S-ERIAL_FIFO

  - ::EMBER_SERIAL1_MODE=::EMBER_SERIAL_BUFFER or ::EMBER_S-ERIAL_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.

6.21.2 Macro Definition Documentation

6.21.2.1 #define SERIAL_PORT_RAW

Definition at line 17 of file linux-serial.h.

6.21.2.2 #define SERIAL_PORT_CLI

Definition at line 18 of file linux-serial.h.

6.21.3 Function Documentation

6.21.3.1 void emberSerialSetPrompt (const char * *thePrompt*)

6.21.3.2 void emberSerialCleanup (void)

6.21.3.3 int emberSerialGetInputFd (uint8_t *port*)

6.21.3.4 void emberSerialSendReadyToRead (uint8_t *port*)

6.21.3.5 void emberSerialCommandCompletionInit (EmberCommandEntry * *listOfCommands*)

6.21.3.6 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

port	A serial port number (0 or 1).
rate	The baud rate (see SerialBaudRate).
parity	The parity value (see SerialParity).
stopBits	The number of stop bits.

Returns

An error code if initialization failed (such as invalid baudrate), or EMBER_SUCCESS.

6.21.3.7 uint16_t emberSerialReadAvailable (uint8_t *port*)

Returns the number of bytes currently available for reading in the specified RX queue.

Parameters

port	A serial port number (0 or 1).

Returns

The number of bytes available.

6.21.3.8 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

port	A serial port number (0 or 1).
dataByte	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

6.21.3.9 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

port	A serial port number (0 or 1).
data	A pointer to storage location for the data. It must be at least `length` in size.
length	The number of bytes to read.
bytesRead	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

6.21.3.10 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

port	A serial port number (0 or 1).
data	A pointer to storage location for the data. It must be at least `length` in size.
length	The maximum number of bytes to read.
bytesRead	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 `length`. This parameter may be NULL, in which case it is ignored.
firstByte-Timeout	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.
subsequent-ByteTimeout	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

6.21.3.11 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

port	A serial port number (0 or 1).
data	A pointer to storage location for the read line. There must be `max` contiguous bytes available at this location.
max	The maximum number of bytes to read.

Returns

EMBER_SUCCESS

6.21.3.12 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

port	A serial port number (0 or 1).
data	A pointer to storage location for the read line. There must be `max` contiguous bytes available at this location.
max	The maximum number of bytes to read.
index	The address of a variable that holds the place in the `data` 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 ine is ready.

6.21.3.13 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

port	A serial port number (0 or 1).

Returns

The number of bytes or messages available for queueing.

6.21.3.14 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

port	A serial port number (0 or 1).

Returns

The number of bytes or messages available for queueing.

6.21.3.15 EmberStatus emberSerialWriteByte (uint8_t *port*, uint8_t *dataByte*)

Queues a single byte of data for transmission on the specified port.

Parameters

port	A serial port number (0 or 1).
dataByte	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.

6.21.3.16 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

port	A serial port number (0 or 1).
dataByte	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.

6.21.3.17 EmberStatus emberSerialWriteString (uint8_t *port*, PGM_P *string*)

Queues a string for transmission on the specified port.

Parameters

port	A serial port number (0 or 1).
string	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.

6.21.3.18 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

port	A serial port number (0 or 1).
formatString	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.

6.21.3.19 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

port	A serial port number (0 or 1).
formatString	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.

6.21.3.20 XAP2B_PAGEZERO_OFF XAP2B_PAGEZERO_ON EmberStatus emberSerialPrintCarriageReturn (uint8_t *port*)

Prints "\r\n" to the specified serial port.

Parameters

port	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.

6.21.3.21 XAP2B_PAGEZERO_OFF EmberStatus emberSerialPrintfVarArg (uint8_t *port,* PGM_P *formatString,* va_list *ap*)

Prints a format string with a variable argument list.

Parameters

port	A serial port number (0 or 1).
formatString	A printf style format string.
ap	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.

6.21.3.22 EmberStatus emberSerialWriteData (uint8_t *port,* uint8_t $*$ *data,* uint8_t *length*)

Queues an arbitrary chunk of data for transmission on a specified port.

Parameters

port	A serial port number (0 or 1).
data	A pointer to data.
length	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.

6.21.3.23 EmberStatus emberSerialWriteBuffer (uint8_t *port*, EmberMessageBuffer *buffer*, uint8_t *start*, uint8_t *length*)

Queues data contained in linked stack buffers for transmission on a specified port. Can specify an arbitrary initial offset within the linked buffer chain.

Parameters

port	A serial port number (0 or 1).
buffer	The starting buffer in linked buffer chain.
start	The offset from first buffer in chain.
length	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.

6.21.3.24 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

port	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.

6.21.3.25 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

port	A serial port number (0 or 1).
formatString	The string to print.
...	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.

6.21.3.26 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.

6.21.3.27 void emberSerialFlushRx (uint8_t *port*)

Flushes the receive buffer in case none of the incoming serial data is wanted.

Parameters

port	A serial port number (0 or 1).

6.21.3.28 bool emberSerialUnused (uint8_t *port*)

Indicates whether the port is unused or invalid.

Parameters

port	A serial port number.

Returns

true if the port is unused or invalid.

6.22 ASH Application Utility

See also Asynchronous Serial Host (ASH) Framework.

6.23 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 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-interpreter2.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 hexidecimal. A 0x prefix indicates a hexidecimal 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 *EMBER_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 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 EmberInitial-SecurityState::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_N-ETWORK_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 ::emberNetwork-InitExtended() 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_283-K1_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 nwkUpdateId

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 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

7.32.1 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.

7.32.2 Field Documentation

7.32.2.1 uint8_t InterPanHeader::messageType

Definition at line 52 of file ami-inter-pan.h.

7.32.2.2 uint16_t InterPanHeader::panId

Definition at line 57 of file ami-inter-pan.h.

7.32.2.3 bool InterPanHeader::hasLongAddress

Definition at line 58 of file ami-inter-pan.h.

7.32.2.4 EmberNodeId InterPanHeader::shortAddress

Definition at line 59 of file ami-inter-pan.h.

7.32.2.5 EmberEUI64 InterPanHeader::longAddress

Definition at line 60 of file ami-inter-pan.h.

7.32.2.6 uint16_t InterPanHeader::profileId

Definition at line 63 of file ami-inter-pan.h.

7.32.2.7 uint16_t InterPanHeader::clusterId

Definition at line 64 of file ami-inter-pan.h.

7.32.2.8 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 _PC_Host_API.top File Reference

8.1.1 Detailed Description

Starting page for the Ember API documentation for the PC Host, exclusively for building documentation. This file is used by Doxygen to generate the main page for the Ember API documentation, PC Host.

Definition in file _PC_Host_API.top.

8.2 _PC_Host_API.top

```
00001
```

8.3 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)

8.3.1 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.

8.4 ami-inter-pan-host.h

```

#ifndef AMI_INTER_PAN_HOST_H
#define AMI_INTER_PAN_HOST_H

#define INTER_PAN_UNICAST   0x03
#define INTER_PAN_BROADCAST 0x0B
#define INTER_PAN_MULTICAST 0x0F


// Frame control, sequence, dest PAN ID, dest, source PAN ID, source.
#define MAX_INTER_PAN_MAC_SIZE (2 + 1 + 2 + 8 + 2 + 8)
//Short form has a short destination.

// NWK stub frame has two control bytes.
#define STUB_NWK_SIZE 2
#define STUB_NWK_FRAME_CONTROL 0x000B

// APS frame control, group ID, cluster ID, profile ID
#define MAX_STUB_APS_SIZE (1 + 2 + 2 + 2)

// Short form has no group ID.
#define MAX_INTER_PAN_HEADER_SIZE \
 (MAX_INTER_PAN_MAC_SIZE + STUB_NWK_SIZE + MAX_STUB_APS_SIZE)

typedef struct {
  uint8_t messageType;            // one of the INTER_PAN_...CAST values

  // MAC addressing
  // For outgoing messages this is the destination.  For incoming messages
  // it is the source, which always has a long address.
  uint16_t panId;
  bool hasLongAddress;       // always true for incoming messages
  EmberNodeId shortAddress;
  EmberEUI64 longAddress;

  // APS data
  uint16_t profileId;
  uint16_t clusterId;
  uint16_t groupId;               // only used for INTER_PAN_MULTICAST
} InterPanHeader;

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);

#endif // AMI_INTER_PAN_HOST_H

```

8.5 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)

8.5.1 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.h.

8.6 ami-inter-pan.h

```

#ifndef AMI_INTER_PAN_H
#define AMI_INTER_PAN_H

// 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.
//

#define INTER_PAN_UNICAST   0x03
#define INTER_PAN_BROADCAST 0x0B
#define INTER_PAN_MULTICAST 0x0F

// Frame control, sequence, dest PAN ID, dest, source PAN ID, source.
#define MAX_INTER_PAN_MAC_SIZE (2 + 1 + 2 + 8 + 2 + 8)
// Short form has a short destination.

// NWK stub frame has two control bytes.
#define STUB_NWK_SIZE 2
#define STUB_NWK_FRAME_CONTROL 0x000B

// APS frame control, group ID, cluster ID, profile ID
#define MAX_STUB_APS_SIZE (1 + 2 + 2 + 2)
// Short form has no group ID.

#define MAX_INTER_PAN_HEADER_SIZE \
 (MAX_INTER_PAN_MAC_SIZE + STUB_NWK_SIZE + MAX_STUB_APS_SIZE)

typedef struct {
  uint8_t messageType;            // one of the INTER_PAN_...CAST values

  // MAC addressing
  // For outgoing messages this is the destination.  For incoming messages
  // it is the source, which always has a long address.
  uint16_t panId;
  bool hasLongAddress;       // always true for incoming messages
  EmberNodeId shortAddress;
  EmberEUI64 longAddress;

  // APS data
  uint16_t profileId;
  uint16_t clusterId;
  uint16_t groupId;               // only used for INTER_PAN_MULTICAST
} InterPanHeader;


EmberMessageBuffer makeInterPanMessage(InterPanHeader *headerData,
                                       EmberMessageBuffer payload);

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

#endif // AMI_INTER_PAN_H

```

8.7 ash-common.h File Reference

Macros

- #define ashStopAckTimer(void)
- #define ashAckTimerIsRunning()
- #define ashAckTimerIsNotRunning()
- #define ashSetAckPeriod(msec)
- #define ashGetAckPeriod()
- #define ashSetAndStartAckTimer(msec)
- #define ASH_NR_TIMER_BIT
- #define ashStopNrTimer()
- #define ashNrTimerIsNotRunning()

Functions

- uint8_t ashEncodeByte (uint8_t len, uint8_t byte, uint8_t ∗offset)
- EzspStatus ashDecodeByte (uint8_t byte, uint8_t ∗out, uint8_t ∗outLen)
- uint8_t ashRandomizeArray (uint8_t seed, uint8_t ∗buf, uint8_t len)
- void ashStartAckTimer (void)
- bool ashAckTimerHasExpired (void)
- void ashAdjustAckPeriod (bool expired)
- void ashStartNrTimer (void)
- bool ashNrTimerHasExpired (void)

Variables

- bool ashDecodeInProgress
- uint16_t ashAckTimer
- uint16_t ashAckPeriod
- uint8_t ashNrTimer

8.7.1 Detailed Description

Header for ASH common functions. See Asynchronous Serial Host (ASH) Framework for documentation.

Definition in file ash-common.h.

8.8 ash-common.h

```

#ifndef __ASH_COMMON_H__
#define __ASH_COMMON_H__

uint8_t ashEncodeByte(uint8_t len, uint8_t byte, uint8_t *offset);

EzspStatus ashDecodeByte(uint8_t byte, uint8_t *out, uint8_t *
outLen);

uint8_t ashRandomizeArray(uint8_t seed, uint8_t *buf, uint8_t
len);

void ashStartAckTimer(void);

void ashStopAckTimer(void);
#define ashStopAckTimer() do {ashAckTimer = 0;} while (false)

#define ashAckTimerIsRunning() (ashAckTimer != 0)

#define ashAckTimerIsNotRunning() (ashAckTimer == 0)

bool ashAckTimerHasExpired(void);

void ashAdjustAckPeriod(bool expired);

#define ashSetAckPeriod(msec)  \
    do {ashAckPeriod = msec; ashAckTimer = 0;} while (false)

#define ashGetAckPeriod() (ashAckPeriod)

#define ashSetAndStartAckTimer(msec) \
    do {ashSetAckPeriod(msec); ashStartAckTimer();}  while (false)

// Define the units used by the Not Ready timer as 2**n msecs
#define ASH_NR_TIMER_BIT    4 // log2 of msecs per NR timer unit
```

```

void ashStartNrTimer(void);

#define ashStopNrTimer()  do {ashNrTimer = 0;} while (false)

bool ashNrTimerHasExpired(void);

#define ashNrTimerIsNotRunning() (ashAckTimer == 0)

extern bool ashDecodeInProgress; // set false to start decoding a new frame

// ASH timers (units)
extern uint16_t ashAckTimer;        // rec'd ack timer (msecs)
extern uint16_t ashAckPeriod;       // rec'd ack timer period (msecs)
extern uint8_t ashNrTimer;          // not ready timer (16 msec units)

#endif //__ASH_COMMON_H__

```

8.9 ash-protocol.h File Reference

`#include "app/util/ezsp/ezsp-protocol.h"`

Macros

- #define ASH_VERSION
- #define ASH_FLAG
- #define ASH_ESC
- #define ASH_XON
- #define ASH_XOFF
- #define ASH_SUB
- #define ASH_CAN
- #define ASH_WAKE
- #define ASH_FLIP
- #define ASH_MIN_DATA_FIELD_LEN
- #define ASH_MAX_DATA_FIELD_LEN
- #define ASH_MIN_DATA_FRAME_LEN
- #define ASH_MIN_FRAME_LEN
- #define ASH_MAX_FRAME_LEN
- #define ASH_CRC_LEN
- #define ASH_MIN_FRAME_WITH_CRC_LEN
- #define ASH_MAX_FRAME_WITH_CRC_LEN
- #define ASH_NCP_SHFRAME_RX_LEN
- #define ASH_NCP_SHFRAME_TX_LEN
- #define ASH_HOST_SHFRAME_RX_LEN
- #define ASH_HOST_SHFRAME_TX_LEN
- #define ASH_DFRAME_MASK
- #define ASH_CONTROL_DATA
- #define ASH_SHFRAME_MASK
- #define ASH_CONTROL_ACK
- #define ASH_CONTROL_NAK
- #define ASH_CONTROL_RST

- #define ASH_CONTROL_RSTACK
- #define ASH_CONTROL_ERROR
- #define ASH_ACKNUM_MASK
- #define ASH_ACKNUM_BIT
- #define ASH_RFLAG_MASK
- #define ASH_RFLAG_BIT
- #define ASH_NFLAG_MASK
- #define ASH_NFLAG_BIT
- #define ASH_PFLAG_MASK
- #define ASH_PFLAG_BIT
- #define ASH_FRMNUM_MASK
- #define ASH_FRMNUM_BIT
- #define ASH_GET_RFLAG(ctl)
- #define ASH_GET_NFLAG(ctl)
- #define ASH_GET_FRMNUM(ctl)
- #define ASH_GET_ACKNUM(ctl)
- #define ASH_FRAME_LEN_DATA_MIN
- #define ASH_FRAME_LEN_ACK
- #define ASH_FRAME_LEN_NAK
- #define ASH_FRAME_LEN_RST
- #define ASH_FRAME_LEN_RSTACK
- #define ASH_FRAME_LEN_ERROR
- #define MOD8(n)
- #define INC8(n)
- #define WITHIN_RANGE(lo, n, hi)

8.9.1 Detailed Description

ASH protocol header. See Asynchronous Serial Host (ASH) Framework for documentation.

Definition in file ash-protocol.h.

8.10 ash-protocol.h

```

#ifndef __ASH_PROTOCOL_H__
#define __ASH_PROTOCOL_H__

#include "app/util/ezsp/ezsp-protocol.h"

#define ASH_VERSION 2   // protocol version

// Special byte values for ASH protocol and/or low-level comm
// Bytes with these values must be escaped (byte-stuffed) before transmission
#define ASH_FLAG  0x7E
#define ASH_ESC   0x7D
#define ASH_XON   0x11
#define ASH_XOFF  0x13
#define ASH_SUB   0x18
#define ASH_CAN   0x1A
// The wake byte special function applies only when in between frames, so it
// does not need to be escaped within a frame.
#define ASH_WAKE  0xFF
// Constant used in byte-stuffing
#define ASH_FLIP  0x20
// Field and frame lengths, excluding flag byte and any byte stuffing overhead
// All limits are inclusive
```

```
#define ASH_MIN_DATA_FIELD_LEN    EZSP_MIN_FRAME_LENGTH
#define ASH_MAX_DATA_FIELD_LEN    EZSP_MAX_FRAME_LENGTH
#define ASH_MIN_DATA_FRAME_LEN    (ASH_MIN_DATA_FIELD_LEN + 1) // with control
#define ASH_MIN_FRAME_LEN         1    // control plus data field, but not CRC
#define ASH_MAX_FRAME_LEN         (ASH_MAX_DATA_FIELD_LEN + 1)
#define ASH_CRC_LEN               2
#define ASH_MIN_FRAME_WITH_CRC_LEN  (ASH_MIN_FRAME_LEN + ASH_CRC_LEN)
#define ASH_MAX_FRAME_WITH_CRC_LEN  (ASH_MAX_FRAME_LEN + ASH_CRC_LEN)

// Define lengths of short frames - includes control byte and data field
#define ASH_NCP_SHFRAME_RX_LEN    2
#define ASH_NCP_SHFRAME_TX_LEN    3
#define ASH_HOST_SHFRAME_RX_LEN   3
#define ASH_HOST_SHFRAME_TX_LEN   2
// Control byte formats
//  +---------+----+----+----+----+----+----+----+----++---------+
//  |         | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 ||  Range  |
//  +---------+----+----+----+----+----+----+----+----++---------+
//  | DATA    |  0 |   frameNum   | rF |    ackNum    ||0x00-0x7F|
//  +---------+----+----+----+----+----+----+----+----++---------+
//  | ACK     |  1 |  0 |  0 | pF | nF |    ackNum    ||0x80-0x9F|
//  | NAK     |  1 |  0 |  1 | pF | nF |    ackNum    ||0xA0-0xBF|
//  +---------+----+----+----+----+----+----+----+----++---------+
//  | RST     |  1 |  1 |  0 |  0 |  0 |  0 |  0 |  0 ||   0xC0  |
//  | RSTACK  |  1 |  1 |  0 |  0 |  0 |  0 |  0 |  1 ||   0xC1  |
//  | ERROR   |  1 |  1 |  0 |  0 |  0 |  0 |  1 |  0 ||   0xC2  |
//  +---------+----+----+----+----+----+----+----+----++---------+
//           rF = rFlag (retransmission flag)
//           nF = nFlag (receiver not ready flag)
//           pF = flag reserved for future use
//  Control byte values 0xC3-0xFE are unused, 0xFF is reserved.

// Define frame control byte codes
#define ASH_DFRAME_MASK     0x80
#define ASH_CONTROL_DATA    0x00

#define ASH_SHFRAME_MASK    0xE0
#define ASH_CONTROL_ACK     0x80
#define ASH_CONTROL_NAK     0xA0
#define ASH_CONTROL_RST     0xC0
#define ASH_CONTROL_RSTACK  0xC1
#define ASH_CONTROL_ERROR   0xC2

#define ASH_ACKNUM_MASK     0x07
#define ASH_ACKNUM_BIT         0
#define ASH_RFLAG_MASK      0x08
#define ASH_RFLAG_BIT          3
#define ASH_NFLAG_MASK      0x08
#define ASH_NFLAG_BIT          3
#define ASH_PFLAG_MASK      0x10
#define ASH_PFLAG_BIT          4
#define ASH_FRMNUM_MASK     0x70
#define ASH_FRMNUM_BIT         4
#define ASH_GET_RFLAG(ctl)  ((ctl & ASH_RFLAG_MASK ) >> ASH_RFLAG_BIT )
#define ASH_GET_NFLAG(ctl)  ((ctl & ASH_NFLAG_MASK ) >> ASH_NFLAG_BIT )
#define ASH_GET_FRMNUM(ctl) ((ctl & ASH_FRMNUM_MASK) >> ASH_FRMNUM_BIT)
#define ASH_GET_ACKNUM(ctl) ((ctl & ASH_ACKNUM_MASK) >> ASH_ACKNUM_BIT)

// Lengths for each frame type: includes control and data field (if any),
// excludes the CRC and flag bytes
#define ASH_FRAME_LEN_DATA_MIN  (ASH_MIN_DATA_FIELD_LEN + 1)
#define ASH_FRAME_LEN_ACK       1     // control
#define ASH_FRAME_LEN_NAK       1     // control
#define ASH_FRAME_LEN_RST       1     // control
#define ASH_FRAME_LEN_RSTACK    3     // control, version, reset reason
#define ASH_FRAME_LEN_ERROR     3     // control, version, error

// Define macros for handling 3-bit frame numbers modulo 8
#define MOD8(n)      ((n) & 7)
#define INC8(n)      (n=(MOD8(n+1)))
// Return true if n is within the range lo through hi, computed (mod 8)
#define WITHIN_RANGE(lo, n, hi) (MOD8(n-lo)<=MOD8(hi-lo))

#endif //__ASH_PROTOCOL_H__

```

8.11 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_NUMBE-R_OF_ARGUMENTS,
  EMBER_CMD_ERR_ARGUMENT_OUT_OF_RANGE, EMBER_CMD_ERR_A-RGUMENT_SYNTAX_ERROR, EMBER_CMD_ERR_STRING_TOO_LONG, E-MBER_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)

8.11.1 Detailed Description

Processes commands coming from the serial port. See Command Interpreter 2 for documentation.

Definition in file command-interpreter2.h.

8.12 command-interpreter2.h

```

#ifndef __COMMAND_INTERPRETER2_H__
#define __COMMAND_INTERPRETER2_H__

#ifndef EMBER_MAX_COMMAND_ARGUMENTS

#define EMBER_MAX_COMMAND_ARGUMENTS 16
#endif
```

```

#ifndef EMBER_COMMAND_BUFFER_LENGTH
#define EMBER_COMMAND_BUFFER_LENGTH 100
#endif

#if defined(DOXYGEN_SHOULD_SKIP_THIS)
#define EMBER_COMMAND_INTEPRETER_HAS_DESCRIPTION_FIELD
#endif

// The (+ 1) takes into account the leading command.
#define MAX_TOKEN_COUNT (EMBER_MAX_COMMAND_ARGUMENTS + 1)

typedef void (*CommandAction)(void);

#ifdef DOXYGEN_SHOULD_SKIP_THIS

typedef struct {
#else
typedef PGM struct {
#endif

  PGM_P name;
  CommandAction action;
  PGM_P argumentTypes;
#if defined(EMBER_COMMAND_INTEPRETER_HAS_DESCRIPTION_FIELD)
  PGM_P description;

  PGM_P const * argumentDescriptions;
#endif
} EmberCommandEntry;


#if defined(EMBER_COMMAND_INTEPRETER_HAS_DESCRIPTION_FIELD)
  /* @brief Macro to define a CLI action */
  #define emberCommandEntryAction(name, action, argumentTypes, description) \
    { (name), (action), (argumentTypes), (description), NULL }

  #define emberCommandEntryActionWithDetails(name, \
                                             action,           \
                                             argumentTypes,    \
                                             description,                \
                                             argumentDescriptionArray)   \
    { (name), (action), (argumentTypes), (description),
 (argumentDescriptionArray) }

  /* @brief Macro to define a CLI sub-menu (nested command) */
  #define emberCommandEntrySubMenu(name, subMenu, description)  \
    { (name), NULL, (PGM_P)(subMenu), (description), NULL }

  /* @briefy Macro to define a command entry array terminator.*/
  #define emberCommandEntryTerminator() \
    { NULL, NULL, NULL, NULL, NULL }

#else  // Don't include description data in struct

  /* @brief Macro to define a CLI action */
  #define emberCommandEntryAction(name, action, argumentTypes, description) \
    { (name), (action), (argumentTypes) }

  #define emberCommandEntryActionWithDetails(name, \
                                             action,           \
                                             argumentTypes,    \
                                             description,                \
                                             argumentDescriptionArray)   \
    { (name), (action), (argumentTypes) }

  /* @brief Macro to define a CLI sub-menu (nested command) */
  #define emberCommandEntrySubMenu(name, subMenu, description) \
    { (name), NULL, (PGM_P)(subMenu) }

  /* @briefy Macro to define a command entry array terminator.*/
  #define emberCommandEntryTerminator() \
    { NULL, NULL, NULL }

#endif

extern EmberCommandEntry *emberCurrentCommand
;

extern EmberCommandEntry emberCommandTable[];

extern uint8_t emberCommandInterpreter2Configuration;

#define EMBER_COMMAND_INTERPRETER_CONFIGURATION_ECHO (0x01)

#ifdef DOXYGEN_SHOULD_SKIP_THIS

enum EmberCommandStatus
#else
typedef uint8_t EmberCommandStatus;
enum
#endif
{
  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
};

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);

#define emberCopyKeyArgument(index, keyDataPointer)            \
  (emberCopyStringArgument((index),                            \
                           emberKeyContents((keyDataPointer)), \
                           EMBER_ENCRYPTION_KEY_SIZE,          \
                           true))

#define emberCopyEui64Argument(index, eui64) \
  (emberCopyStringArgument((index), (eui64), EUI64_SIZE, true))
#define emberGetEui64Argument(index, eui64) \
  (emberCopyStringArgument((index), (eui64), EUI64_SIZE, true))

uint8_t emberCopyBigEndianEui64Argument(int8_t index, EmberEUI64 destination);

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);

#define emberProcessCommandInput(port) \
  emberProcessCommandString(NULL, port)

#define emberCommandInterpreterEchoOn()                 \
  (emberCommandInterpreter2Configuration                \
   |= EMBER_COMMAND_INTERPRETER_CONFIGURATION_ECHO)

#define emberCommandInterpreterEchoOff()                \
  (emberCommandInterpreter2Configuration                \
   &= (~EMBER_COMMAND_INTERPRETER_CONFIGURATION_ECHO))

#define emberCommandInterpreterIsEchoOn()               \
  (emberCommandInterpreter2Configuration                \
   & EMBER_COMMAND_INTERPRETER_CONFIGURATION_ECHO)

#endif // __COMMAND_INTERPRETER2_H__
```

8.13 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)

8.13.1 Detailed Description

See Cyclic Redundancy Code (CRC) for detailed documentation.

Definition in file crc.h.

8.14 crc.h

```

#ifndef __CRC_H__
#define __CRC_H__

uint16_t halCommonCrc16(uint8_t newByte, uint16_t prevResult);


uint32_t halCommonCrc32(uint8_t newByte, uint32_t prevResult);

// Commonly used initial and expected final CRC32 values
#define INITIAL_CRC              0xFFFFFFFFL
#define CRC32_START              INITIAL_CRC
#define CRC32_END                0xDEBB20E3L  // For CRC32 POLYNOMIAL run
 LSB-MSB


#endif //__CRC_H__

```

8.15 em2xx-reset-defs.h File Reference

- #define EM2XX_RESET_UNKNOWN
- #define EM2XX_RESET_EXTERNAL
- #define EM2XX_RESET_POWERON
- #define EM2XX_RESET_WATCHDOG
- #define EM2XX_RESET_ASSERT
- #define EM2XX_RESET_BOOTLOADER
- #define EM2XX_RESET_SOFTWARE

8.15.1 Detailed Description

Definitions of reset types compatible with EM2xx usage.

Definition in file em2xx-reset-defs.h.

8.16 em2xx-reset-defs.h

```

#ifndef __EM2XX_RESET_DEFS_H__
#define __EM2XX_RESET_DEFS_H__


#define EM2XX_RESET_UNKNOWN               0
#define EM2XX_RESET_EXTERNAL              1   // EM2XX reports POWERON instead
#define EM2XX_RESET_POWERON               2
#define EM2XX_RESET_WATCHDOG              3
#define EM2XX_RESET_ASSERT                6
#define EM2XX_RESET_BOOTLOADER            9
#define EM2XX_RESET_SOFTWARE              11

#endif    //__EM2XX_RESET_DEFS_H__
```

8.17 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_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_TABL-
E, EMBER_OUTGOING_VIA_BINDING, EMBER_OUTGOING_MULTICAST,
EMBER_OUTGOING_MULTICAST_WITH_ALIAS, EMBER_OUTGOING_BR-
OADCAST_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_T-
YPE_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_BIN-
DING }

- enum EmberJoinDecision { EMBER_USE_PRECONFIGURED_KEY, EMBER_S-END_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_STAND-ARD_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_REJOI-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_OUTP-UT_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_COUNT-ER_MAC_TX_UNICAST_SUCCESS, EMBER_COUNTER_MAC_TX_UNICAST_RETRY, EMBER_COUNTER_MA-C_TX_UNICAST_FAILED, EMBER_COUNTER_APS_DATA_RX_BROADCA-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-MBER_APP_MASTER_KEY_ESTABLISHED, EMBER_TRUST_CENTER_LIN-K_KEY_ESTABLISHED,
EMBER_KEY_ESTABLISHMENT_TIMEOUT, EMBER_KEY_TABLE_FULL, E-MBER_TC_RESPONDED_TO_KEY_REQUEST, EMBER_TC_APP_KEY_SEN-T_TO_REQUESTER,
EMBER_TC_RESPONSE_TO_KEY_REQUEST_FAILED, EMBER_TC_REQUE-ST_KEY_TYPE_NOT_SUPPORTED, EMBER_TC_NO_LINK_KEY_FOR_REQ-UESTER, EMBER_TC_REQUESTER_EUI64_UNKNOWN,
EMBER_TC_RECEIVED_FIRST_APP_KEY_REQUEST, EMBER_TC_TIMEO-UT_WAITING_FOR_SECOND_APP_KEY_REQUEST, EMBER_TC_NON_MA-TCHING_APP_KEY_REQUEST_RECEIVED, EMBER_TC_FAILED_TO_SEND-_APP_KEYS,
EMBER_TC_FAILED_TO_STORE_APP_KEY_REQUEST, EMBER_TC_REJE-CTED_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, EM-BER_TC_REQUESTER_VERIFY_KEY_SUCCESS,
EMBER_VERIFY_LINK_KEY_FAILURE, EMBER_VERIFY_LINK_KEY_SUC-CESS }
- enum EmberLinkKeyRequestPolicy { EMBER_DENY_KEY_REQUESTS, EMBE-R_ALLOW_KEY_REQUESTS, EMBER_GENERATE_NEW_TC_LINK_KEY }

- enum EmberKeySettings { EMBER_KEY_PERMISSIONS_NONE, EMBER_KE-Y_PERMISSIONS_READING_ALLOWED, EMBER_KEY_PERMISSIONS_HA-SHING_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_PASSTHRO-UGH_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_ALPH-A_1, EMBER_VERSION_TYPE_ALPHA_2, EMBER_VERSION_TYPE_ALPH-A_3,
  EMBER_VERSION_TYPE_BETA_1, EMBER_VERSION_TYPE_BETA_2, EM-BER_VERSION_TYPE_BETA_3, EMBER_VERSION_TYPE_GA }
- enum EmberLeaveRequestFlags { EMBER_ZIGBEE_LEAVE_AND_REJOIN, E-MBER_ZIGBEE_LEAVE_AND_REMOVE_CHILDREN }
- enum EmberLeaveReason {
  EMBER_LEAVE_REASON_NONE, EMBER_LEAVE_DUE_TO_NWK_LEAV-E_MESSAGE, EMBER_LEAVE_DUE_TO_APS_REMOVE_MESSAGE, EMBE-R_LEAVE_DUE_TO_ZDO_LEAVE_MESSAGE,
  EMBER_LEAVE_DUE_TO_ZLL_TOUCHLINK, EMBER_LEAVE_DUE_TO_A-PP_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

```
<br>

@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

```
<br>

@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_DESCRIPTORS_REQUEST
- #define MATCH_DESCRIPTORS_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_TR-UST_CENTER, EMBER_ZDP_PRIMARY_BINDING_TABLE_CACHE, EMBE-R_ZDP_SECONDARY_BINDING_TABLE_CACHE,
  EMBER_ZDP_PRIMARY_DISCOVERY_CACHE, EMBER_ZDP_SECONDAR-Y_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 }

8.17.1 Detailed Description

Ember data type definitions. See Ember Common Data Types for details.

Definition in file ember-types.h.

8.18 ember-types.h

```
#ifndef EMBER_TYPES_H
#define EMBER_TYPES_H

#ifndef DOXYGEN_SHOULD_SKIP_THIS
#include "stack/config/ember-configuration-defaults.h"
#include "stack/include/ember-static-struct.h"
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS
enum EmberVersionType
#else
typedef uint8_t EmberVersionType;
enum
#endif
{
  EMBER_VERSION_TYPE_PRE_RELEASE = 0x00,

        //Alpha, should be used rarely
  EMBER_VERSION_TYPE_ALPHA_1     = 0x11,
  EMBER_VERSION_TYPE_ALPHA_2     = 0x12,
  EMBER_VERSION_TYPE_ALPHA_3     = 0x13,
  // Leave space in case we decide to add other types in the future.
  EMBER_VERSION_TYPE_BETA_1      = 0x21,
  EMBER_VERSION_TYPE_BETA_2      = 0x22,
  EMBER_VERSION_TYPE_BETA_3      = 0x23,



  // Anything other than 0xAA is considered pre-release
  // We may define other types in the future (e.g. beta, alpha)
  // We chose an arbitrary number (0xAA) to allow for expansion, but
  // to prevent ambiguity in case 0x00 or 0xFF is accidentally retrieved
  // as the version type.
  EMBER_VERSION_TYPE_GA = 0xAA,
};

typedef struct {
  EmberVersionType typeNum;
  PGM_P typeString;
} EmberReleaseTypeStruct;

#define EMBER_RELEASE_TYPE_TO_STRING_STRUCT_DATA     \
  { EMBER_VERSION_TYPE_PRE_RELEASE, "Pre-Release" }, \
  { EMBER_VERSION_TYPE_ALPHA_1,     "Alpha 1" },     \
  { EMBER_VERSION_TYPE_ALPHA_2,     "Alpha 2" },     \
  { EMBER_VERSION_TYPE_ALPHA_3,     "Alpha 3" },     \
  { EMBER_VERSION_TYPE_BETA_1,      "Beta 1" },      \
  { EMBER_VERSION_TYPE_BETA_2,      "Beta 2" },      \
  { EMBER_VERSION_TYPE_BETA_3,      "Beta 3" },      \
  { EMBER_VERSION_TYPE_GA,          "GA" },          \
  { 0xFF, NULL },


typedef struct {
  uint16_t build;
  uint8_t major;
  uint8_t minor;
  uint8_t patch;
  uint8_t special;
  EmberVersionType type;
} EmberVersion;

extern const EmberVersion emberVersion;

#define EUI64_SIZE 8

#define EXTENDED_PAN_ID_SIZE 8

#define EMBER_ENCRYPTION_KEY_SIZE 16

#define EMBER_CERTIFICATE_SIZE 48

#define EMBER_PUBLIC_KEY_SIZE 22

#define EMBER_PRIVATE_KEY_SIZE 21

#define EMBER_SMAC_SIZE 16

#define EMBER_SIGNATURE_SIZE 42

#define EMBER_AES_HASH_BLOCK_SIZE 16

#define EMBER_CERTIFICATE_283K1_SIZE 74

#define EMBER_PUBLIC_KEY_283K1_SIZE 37

#define EMBER_PRIVATE_KEY_283K1_SIZE 36

#define EMBER_SIGNATURE_283K1_SIZE 72

#ifndef __EMBERSTATUS_TYPE__
#define __EMBERSTATUS_TYPE__
  typedef uint8_t EmberStatus;
#endif //__EMBERSTATUS_TYPE__

#include "stack/include/error.h"

typedef uint8_t EmberEUI64[EUI64_SIZE];

typedef uint8_t EmberMessageBuffer;

typedef uint16_t EmberNodeId;

typedef uint16_t EmberMulticastId;

typedef uint16_t EmberPanId;

#define EMBER_MAX_802_15_4_CHANNEL_NUMBER 26

#define EMBER_MIN_802_15_4_CHANNEL_NUMBER 11

#define EMBER_NUM_802_15_4_CHANNELS \
  (EMBER_MAX_802_15_4_CHANNEL_NUMBER - EMBER_MIN_802_15_4_CHANNEL_NUMBER + 1)

#define EMBER_ALL_802_15_4_CHANNELS_MASK 0x07FFF800UL

#define EMBER_ZIGBEE_COORDINATOR_ADDRESS 0x0000

#define EMBER_NULL_NODE_ID 0xFFFF

#define EMBER_NULL_BINDING 0xFF

#define EMBER_TABLE_ENTRY_UNUSED_NODE_ID  0xFFFF

#define EMBER_MULTICAST_NODE_ID           0xFFFE

```

```
#define EMBER_UNKNOWN_NODE_ID             0xFFFD

#define EMBER_DISCOVERY_ACTIVE_NODE_ID    0xFFFC

#define EMBER_NULL_ADDRESS_TABLE_INDEX 0xFF

#define EMBER_ZDO_ENDPOINT 0

#define EMBER_BROADCAST_ENDPOINT 0xFF

#define EMBER_ZDO_PROFILE_ID  0x0000

#define EMBER_WILDCARD_PROFILE_ID  0xFFFF

#define EMBER_MAXIMUM_STANDARD_PROFILE_ID  0x7FFF

#define EMBER_BROADCAST_TABLE_TIMEOUT_QS (20 * 4)


#define EMBER_MANUFACTURER_ID 0x1002


#ifdef DOXYGEN_SHOULD_SKIP_THIS
enum EmberLeaveRequestFlags
#else
typedef uint8_t EmberLeaveRequestFlags;
enum
#endif
{
  EMBER_ZIGBEE_LEAVE_AND_REJOIN          = 0x80,

  EMBER_ZIGBEE_LEAVE_AND_REMOVE_CHILDREN
= 0x40,
};

#ifdef DOXYGEN_SHOULD_SKIP_THIS
enum EmberLeaveReason
#else
typedef uint8_t EmberLeaveReason;
enum
#endif
{
  EMBER_LEAVE_REASON_NONE               = 0,
  EMBER_LEAVE_DUE_TO_NWK_LEAVE_MESSAGE  = 1
,
  EMBER_LEAVE_DUE_TO_APS_REMOVE_MESSAGE =
2,
  // Currently, the stack does not process the ZDO leave message since it is
 optional
  EMBER_LEAVE_DUE_TO_ZDO_LEAVE_MESSAGE  = 3
,
  EMBER_LEAVE_DUE_TO_ZLL_TOUCHLINK      = 4,

  EMBER_LEAVE_DUE_TO_APP_EVENT_1        = 0xFF,
};



#define EMBER_BROADCAST_ADDRESS 0xFFFC

#define EMBER_RX_ON_WHEN_IDLE_BROADCAST_ADDRESS 0xFFFD

#define EMBER_SLEEPY_BROADCAST_ADDRESS 0xFFFF

// From table 3.51 of 053474r14
#define EMBER_MIN_BROADCAST_ADDRESS 0xFFF8

#define emberIsZigbeeBroadcastAddress(address) \
 (EMBER_MIN_BROADCAST_ADDRESS <= ((uint16_t) (address)))


#ifdef DOXYGEN_SHOULD_SKIP_THIS
enum EmberNodeType
#else
typedef uint8_t EmberNodeType;
enum
#endif
{
  EMBER_UNKNOWN_DEVICE = 0,
  EMBER_COORDINATOR = 1,
  EMBER_ROUTER = 2,
  EMBER_END_DEVICE = 3,
  EMBER_SLEEPY_END_DEVICE = 4,
  EMBER_MOBILE_END_DEVICE = 5,
  EMBER_RF4CE_TARGET = 6,
  EMBER_RF4CE_CONTROLLER = 7,
};

#ifdef DOXYGEN_SHOULD_SKIP_THIS
enum EmberEndDeviceConfiguration
#else
typedef uint8_t EmberEndDeviceConfiguration;
enum
#endif
{
  EMBER_END_DEVICE_CONFIG_NONE                     =
0x00,
  EMBER_END_DEVICE_CONFIG_PERSIST_DATA_ON_PARENT
 = 0x01,
};

typedef struct {
  uint16_t panId;
  uint8_t channel;
  bool allowingJoin;
  uint8_t extendedPanId[8];
  uint8_t stackProfile;
  uint8_t nwkUpdateId;
} EmberZigbeeNetwork;


#ifdef DOXYGEN_SHOULD_SKIP_THIS
enum EmberNetworkInitBitmask
#else
typedef uint16_t EmberNetworkInitBitmask;
enum
#endif
{
  EMBER_NETWORK_INIT_NO_OPTIONS           = 0x0000
,
  EMBER_NETWORK_INIT_PARENT_INFO_IN_TOKEN
 = 0x0001,
};


typedef struct {
  EmberNetworkInitBitmask bitmask;
} EmberNetworkInitStruct;


#ifdef DOXYGEN_SHOULD_SKIP_THIS
enum EmberApsOption
#else
typedef uint16_t EmberApsOption;
enum
#endif
{
  EMBER_APS_OPTION_NONE                     = 0x0000,

#ifndef DOXYGEN_SHOULD_SKIP_THIS
  EMBER_APS_OPTION_ENCRYPT_WITH_TRANSIENT_KEY = 0x0001,
  EMBER_APS_OPTION_USE_ALIAS_SEQUENCE_NUMBER = 0x0002,
#endif

  EMBER_APS_OPTION_DSA_SIGN                 = 0x0010,
  EMBER_APS_OPTION_ENCRYPTION               = 0x0020
,
  EMBER_APS_OPTION_RETRY                    = 0x0040,
  EMBER_APS_OPTION_ENABLE_ROUTE_DISCOVERY
   = 0x0100,
  EMBER_APS_OPTION_FORCE_ROUTE_DISCOVERY
   = 0x0200,
  EMBER_APS_OPTION_SOURCE_EUI64             =
0x0400,
  EMBER_APS_OPTION_DESTINATION_EUI64        =
 0x0800,
  EMBER_APS_OPTION_ENABLE_ADDRESS_DISCOVERY
 = 0x1000,
  EMBER_APS_OPTION_POLL_RESPONSE            =
0x2000,
  EMBER_APS_OPTION_ZDO_RESPONSE_REQUIRED
   = 0x4000,
  EMBER_APS_OPTION_FRAGMENT                  =
SIGNED_ENUM 0x8000
};



#ifdef DOXYGEN_SHOULD_SKIP_THIS
enum EmberIncomingMessageType
#else
typedef uint8_t EmberIncomingMessageType;
enum
#endif
{
  EMBER_INCOMING_UNICAST,
  EMBER_INCOMING_UNICAST_REPLY,
  EMBER_INCOMING_MULTICAST,
  EMBER_INCOMING_MULTICAST_LOOPBACK,
  EMBER_INCOMING_BROADCAST,
  EMBER_INCOMING_BROADCAST_LOOPBACK
};


#ifdef DOXYGEN_SHOULD_SKIP_THIS
enum EmberOutgoingMessageType
#else
typedef uint8_t EmberOutgoingMessageType;
enum
#endif
{
  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
};

#ifdef DOXYGEN_SHOULD_SKIP_THIS
enum EmberZigbeeCommandType
#else
typedef uint8_t EmberZigbeeCommandType;
enum
#endif
{
  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,
};

#ifdef DOXYGEN_SHOULD_SKIP_THIS
enum EmberNetworkStatus
#else
typedef uint8_t EmberNetworkStatus;
enum
#endif
{
  EMBER_NO_NETWORK,
  EMBER_JOINING_NETWORK,
  EMBER_JOINED_NETWORK,
  EMBER_JOINED_NETWORK_NO_PARENT,
  EMBER_LEAVING_NETWORK
};


#ifdef DOXYGEN_SHOULD_SKIP_THIS
enum EmberNetworkScanType
#else
typedef uint8_t EmberNetworkScanType;
enum
#endif
{
  EMBER_ENERGY_SCAN,
  EMBER_ACTIVE_SCAN
```

```
};


#ifdef DOXYGEN_SHOULD_SKIP_THIS
enum EmberBindingType
#else
typedef uint8_t EmberBindingType;
enum
#endif
{
  EMBER_UNUSED_BINDING         = 0,
  EMBER_UNICAST_BINDING        = 1,
  EMBER_MANY_TO_ONE_BINDING    = 2,
  EMBER_MULTICAST_BINDING      = 3,
};


#define EMBER_LOW_RAM_CONCENTRATOR 0xFFF8

#define EMBER_HIGH_RAM_CONCENTRATOR 0xFFF9



#ifdef DOXYGEN_SHOULD_SKIP_THIS
enum EmberJoinDecision
#else
typedef uint8_t EmberJoinDecision;
enum
#endif
{
  EMBER_USE_PRECONFIGURED_KEY = 0,
  EMBER_SEND_KEY_IN_THE_CLEAR,
  EMBER_DENY_JOIN,
  EMBER_NO_ACTION
};

#define EMBER_JOIN_DECISION_STRINGS \
  "use preconfigured key",          \
  "send key in the clear",          \
  "deny join",                      \
  "no action",


// These map to the actual values within the APS Command frame so they cannot
// be arbitrarily changed.
#ifdef DOXYGEN_SHOULD_SKIP_THIS
enum EmberDeviceUpdate
#else
typedef uint8_t EmberDeviceUpdate;
enum
#endif
{
  EMBER_STANDARD_SECURITY_SECURED_REJOIN
  = 0,
  EMBER_STANDARD_SECURITY_UNSECURED_JOIN
  = 1,
  EMBER_DEVICE_LEFT                        = 2,
  EMBER_STANDARD_SECURITY_UNSECURED_REJOIN
 = 3,
  EMBER_HIGH_SECURITY_SECURED_REJOIN       =
4,
  EMBER_HIGH_SECURITY_UNSECURED_JOIN       =
5,
  /* 6 Reserved */
  EMBER_HIGH_SECURITY_UNSECURED_REJOIN
= 7,
  /* 8 - 15 Reserved */
};

#define EMBER_DEVICE_UPDATE_STRINGS                                         \
    "secured rejoin",                                                       \
    "UNsecured join",                                                       \
    "device left",                                                          \
    "UNsecured rejoin",                                                     \
    "high secured rejoin",                                                  \
    "high UNsecured join",                                                  \
    "RESERVED",                   /* reserved status code, per the spec. */ \
    "high UNsecured rejoin",

#ifdef DOXYGEN_SHOULD_SKIP_THIS
```

```
enum EmberRejoinReason
#else
typedef uint8_t EmberRejoinReason;
enum
#endif
{
  EMBER_REJOIN_REASON_NONE           = 0,
  EMBER_REJOIN_DUE_TO_NWK_KEY_UPDATE = 1,
  EMBER_REJOIN_DUE_TO_LEAVE_MESSAGE  = 2,
  EMBER_REJOIN_DUE_TO_NO_PARENT      = 3,
  EMBER_REJOIN_DUE_TO_ZLL_TOUCHLINK  = 4,

  // App. Framework events
  // 0xA0 - 0xE0

  // Customer Defined Events
  //   I numbered these backwards in case there is ever request
  //   for more application events. We can expand them
  //   without renumbering the previous ones.
  EMBER_REJOIN_DUE_TO_APP_EVENT_5       = 0xFB,
  EMBER_REJOIN_DUE_TO_APP_EVENT_4       = 0xFC,
  EMBER_REJOIN_DUE_TO_APP_EVENT_3       = 0xFD,
  EMBER_REJOIN_DUE_TO_APP_EVENT_2       = 0xFE,
  EMBER_REJOIN_DUE_TO_APP_EVENT_1       = 0xFF,
};

#ifdef DOXYGEN_SHOULD_SKIP_THIS
enum EmberClusterListId
#else
typedef uint8_t EmberClusterListId;
enum
#endif
{
  EMBER_INPUT_CLUSTER_LIST            = 0,
  EMBER_OUTPUT_CLUSTER_LIST           = 1
};


#ifdef DOXYGEN_SHOULD_SKIP_THIS
enum EmberEventUnits
#else
typedef uint8_t EmberEventUnits;
enum
#endif
{
  EMBER_EVENT_INACTIVE = 0,
  EMBER_EVENT_MS_TIME,
  EMBER_EVENT_QS_TIME,
  EMBER_EVENT_MINUTE_TIME,
  EMBER_EVENT_ZERO_DELAY
};


#ifdef DOXYGEN_SHOULD_SKIP_THIS
enum EmberJoinMethod
#else
typedef uint8_t EmberJoinMethod;
enum
#endif
{
  EMBER_USE_MAC_ASSOCIATION         = 0,

  EMBER_USE_NWK_REJOIN              = 1,


  /* For those networks where the "permit joining" flag is never turned
   * on, they will need to use a NWK Rejoin.  If those devices have been
   * preconfigured with the  NWK key (including sequence number) they can use
   * a secured rejoin.  This is only necessary for end devices since they need
   * a parent.  Routers can simply use the ::EMBER_USE_NWK_COMMISSIONING
   * join method below.
   */
  EMBER_USE_NWK_REJOIN_HAVE_NWK_KEY = 2,

  EMBER_USE_NWK_COMMISSIONING       = 3,
};


typedef struct {
  uint8_t   extendedPanId[8];
  uint16_t  panId;
  int8_t   radioTxPower;
  uint8_t   radioChannel;
  EmberJoinMethod joinMethod;

  EmberNodeId nwkManagerId;
  uint8_t nwkUpdateId;
  uint32_t channels;
} EmberNetworkParameters;


#ifdef DOXYGEN_SHOULD_SKIP_THIS
#define emberInitializeNetworkParameters(parameters) \
  (MEMSET(parameters, 0, sizeof(EmberNetworkParameters)))
#else
void emberInitializeNetworkParameters(
EmberNetworkParameters* parameters);
#endif

typedef struct {
  uint16_t profileId;
  uint16_t clusterId;
  uint8_t sourceEndpoint;
  uint8_t destinationEndpoint;
  EmberApsOption options;
  uint16_t groupId;
  uint8_t sequence;
} EmberApsFrame;


typedef struct {
  EmberBindingType type;
  uint8_t local;
  uint16_t clusterId;
  uint8_t remote;
  EmberEUI64 identifier;
  uint8_t networkIndex;
} EmberBindingTableEntry;


typedef struct {
  uint16_t shortId;
  uint8_t  averageLqi;
  uint8_t  inCost;
  uint8_t  outCost;
  uint8_t  age;
  EmberEUI64 longId;
} EmberNeighborTableEntry;

typedef struct {
  uint16_t destination;
  uint16_t nextHop;
  uint8_t status;
  uint8_t age;
  uint8_t concentratorType;
  uint8_t routeRecordState;
} EmberRouteTableEntry;

typedef struct {
  EmberMulticastId multicastId;
  uint8_t endpoint;
  uint8_t networkIndex;
} EmberMulticastTableEntry;

#ifdef DOXYGEN_SHOULD_SKIP_THIS
enum EmberCounterType
#else
typedef uint8_t EmberCounterType;
enum
#endif
{
  EMBER_COUNTER_MAC_RX_BROADCAST = 0,
  EMBER_COUNTER_MAC_TX_BROADCAST = 1,
  EMBER_COUNTER_MAC_RX_UNICAST = 2,
  EMBER_COUNTER_MAC_TX_UNICAST_SUCCESS = 3,
  EMBER_COUNTER_MAC_TX_UNICAST_RETRY = 4,
  EMBER_COUNTER_MAC_TX_UNICAST_FAILED = 5,

  EMBER_COUNTER_APS_DATA_RX_BROADCAST = 6,
  EMBER_COUNTER_APS_DATA_TX_BROADCAST = 7,
  EMBER_COUNTER_APS_DATA_RX_UNICAST = 8,
  EMBER_COUNTER_APS_DATA_TX_UNICAST_SUCCESS
 = 9,
  EMBER_COUNTER_APS_DATA_TX_UNICAST_RETRY
 = 10,
  EMBER_COUNTER_APS_DATA_TX_UNICAST_FAILED
 = 11,

  EMBER_COUNTER_ROUTE_DISCOVERY_INITIATED
 = 12,

  EMBER_COUNTER_NEIGHBOR_ADDED = 13,
  EMBER_COUNTER_NEIGHBOR_REMOVED = 14,
  EMBER_COUNTER_NEIGHBOR_STALE = 15,

  EMBER_COUNTER_JOIN_INDICATION = 16,
  EMBER_COUNTER_CHILD_REMOVED = 17,

  EMBER_COUNTER_ASH_OVERFLOW_ERROR = 18,
  EMBER_COUNTER_ASH_FRAMING_ERROR = 19,
  EMBER_COUNTER_ASH_OVERRUN_ERROR = 20,

  EMBER_COUNTER_NWK_FRAME_COUNTER_FAILURE
 = 21,

  EMBER_COUNTER_APS_FRAME_COUNTER_FAILURE
 = 22,

  EMBER_COUNTER_ASH_XOFF = 23,

  EMBER_COUNTER_APS_LINK_KEY_NOT_AUTHORIZED
 = 24,

  EMBER_COUNTER_NWK_DECRYPTION_FAILURE = 25
,

  EMBER_COUNTER_APS_DECRYPTION_FAILURE = 26
,

  EMBER_COUNTER_ALLOCATE_PACKET_BUFFER_FAILURE
 = 27,

  EMBER_COUNTER_RELAYED_UNICAST = 28,

  EMBER_COUNTER_PHY_TO_MAC_QUEUE_LIMIT_REACHED
 = 29,

  EMBER_COUNTER_PACKET_VALIDATE_LIBRARY_DROPPED_COUNT
 = 30,

  EMBER_COUNTER_TYPE_NWK_RETRY_OVERFLOW =
31,

  EMBER_COUNTER_PHY_CCA_FAIL_COUNT = 32,

  EMBER_COUNTER_BROADCAST_TABLE_FULL = 33,

  EMBER_COUNTER_TYPE_COUNT = 34,
};

#define EMBER_COUNTER_STRINGS                   \
    "Mac Rx Bcast",                             \
    "Mac Tx Bcast",                             \
    "Mac Rx Ucast",                             \
    "Mac Tx Ucast",                             \
    "Mac Tx Ucast Retry",                       \
    "Mac Tx Ucast Fail",                        \
    "APS Rx Bcast",                             \
    "APS Tx Bcast",                             \
    "APS Rx Ucast",                             \
    "APS Tx Ucast Success",                     \
    "APS Tx Ucast Retry",                       \
    "APS Tx Ucast Fail",                        \
    "Route Disc Initiated",                     \
    "Neighbor Added",                           \
    "Neighbor Removed",                         \
    "Neighbor Stale",                           \
    "Join Indication",                          \
    "Child Moved",                              \
    "ASH Overflow",                             \
    "ASH Frame Error",                          \
    "ASH Overrun Error",                        \
    "NWK FC Failure",                           \
    "APS FC Failure",                           \
    "ASH XOff",                                 \
    "APS Unauthorized Key",                     \
    "NWK Decrypt Failures",                     \
    "APS Decrypt Failures",                     \
    "Packet Buffer Allocate Failures",          \
    "Relayed Ucast",                            \
    "Phy to MAC queue limit reached",           \
    "Packet Validate drop count",               \
    "NWK retry overflow",                       \
    "CCA Failures",                             \
    "Broadcast table full",                     \
    NULL

typedef uint8_t EmberTaskId;

typedef struct {
  EmberEventUnits status;
  EmberTaskId taskid;
  uint32_t timeToExecute;
} EmberEventControl;

typedef PGM struct EmberEventData_S {
  EmberEventControl *control;
  void (*handler)(void);
} EmberEventData;

typedef struct {
  // The time when the next event associated with this task will fire
  uint32_t nextEventTime;
  // The list of events associated with this task
  EmberEventData *events;
  // A flag that indicates the task has something to do other than events
  bool busy;
} EmberTaskControl;


#define EMBER_TX_POWER_MODE_DEFAULT             0x0000

#define EMBER_TX_POWER_MODE_BOOST               0x0001

#define EMBER_TX_POWER_MODE_ALTERNATE           0x0002

#define EMBER_TX_POWER_MODE_BOOST_AND_ALTERNATE (EMBER_TX_POWER_MODE_BOOST
\
                                                |EMBER_TX_POWER_MODE_ALTERNATE)
#ifndef DOXYGEN_SHOULD_SKIP_THIS
// The application does not ever need to call emberSetTxPowerMode() with the
// txPowerMode parameter set to this value.  This value is used internally by
// the stack to indicate that the default token configuration has not been
// overridden by a prior call to emberSetTxPowerMode().
#define EMBER_TX_POWER_MODE_USE_TOKEN           0x8000
#endif//DOXYGEN_SHOULD_SKIP_THIS




#define EMBER_PRIVATE_PROFILE_ID  0xC00E

#define EMBER_PRIVATE_PROFILE_ID_START 0xC00D

#define EMBER_PRIVATE_PROFILE_ID_END   0xC016

#define EMBER_BROADCAST_ALARM_CLUSTER       0x0000

#define EMBER_UNICAST_ALARM_CLUSTER         0x0001

#define EMBER_CACHED_UNICAST_ALARM_CLUSTER  0x0002

#define EMBER_REPORT_COUNTERS_REQUEST 0x0003

#define EMBER_REPORT_COUNTERS_RESPONSE 0x8003

#define EMBER_REPORT_AND_CLEAR_COUNTERS_REQUEST 0x0004

#define EMBER_REPORT_AND_CLEAR_COUNTERS_RESPONSE 0x8004

#define EMBER_OTA_CERTIFICATE_UPGRADE_CLUSTER 0x0005
```

```
typedef struct {
  uint8_t contents[EMBER_ENCRYPTION_KEY_SIZE];
} EmberKeyData;

typedef struct {
  uint8_t contents[EMBER_CERTIFICATE_SIZE];
} EmberCertificateData;

typedef struct {
  uint8_t contents[EMBER_PUBLIC_KEY_SIZE];
} EmberPublicKeyData;

typedef struct {
  uint8_t contents[EMBER_PRIVATE_KEY_SIZE];
} EmberPrivateKeyData;

typedef struct {
  uint8_t contents[EMBER_SMAC_SIZE];
} EmberSmacData;

typedef struct {
  uint8_t contents[EMBER_SIGNATURE_SIZE];
} EmberSignatureData;

typedef struct {
  uint8_t contents[EMBER_AES_HASH_BLOCK_SIZE];
} EmberMessageDigest;

typedef struct {
  uint8_t result[EMBER_AES_HASH_BLOCK_SIZE];
  uint32_t length;
} EmberAesMmoHashContext;

typedef struct {
  /* This is the certificate byte data. */
  uint8_t contents[EMBER_CERTIFICATE_283K1_SIZE];
} EmberCertificate283k1Data;

typedef struct {
  uint8_t contents[EMBER_PUBLIC_KEY_283K1_SIZE];
} EmberPublicKey283k1Data;

typedef struct {
  uint8_t contents[EMBER_PRIVATE_KEY_283K1_SIZE];
} EmberPrivateKey283k1Data;

typedef struct {
  uint8_t contents[EMBER_SIGNATURE_283K1_SIZE];
} EmberSignature283k1Data;

#define EMBER_STANDARD_SECURITY_MODE 0x0000

#define EMBER_TRUST_CENTER_NODE_ID 0x0000


#ifdef DOXYGEN_SHOULD_SKIP_THIS
enum EmberInitialSecurityBitmask
#else
typedef uint16_t EmberInitialSecurityBitmask;
enum
#endif
{
  EMBER_DISTRIBUTED_TRUST_CENTER_MODE
= 0x0002,
  EMBER_TRUST_CENTER_GLOBAL_LINK_KEY         =
 0x0004,
  EMBER_PRECONFIGURED_NETWORK_KEY_MODE
 = 0x0008,

#if !defined DOXYGEN_SHOULD_SKIP_THIS
  // Hidden fields used internally.
  EMBER_HAVE_TRUST_CENTER_UNKNOWN_KEY_TOKEN = 0x0010,
  EMBER_HAVE_TRUST_CENTER_LINK_KEY_TOKEN    = 0x0020,
  EMBER_HAVE_TRUST_CENTER_MASTER_KEY_TOKEN  = 0x0030,
#endif

  EMBER_HAVE_TRUST_CENTER_EUI64              =
0x0040,

  EMBER_TRUST_CENTER_USES_HASHED_LINK_KEY
   = 0x0084,

  EMBER_HAVE_PRECONFIGURED_KEY               =
0x0100,
  EMBER_HAVE_NETWORK_KEY                     = 0x0200,
  EMBER_GET_LINK_KEY_WHEN_JOINING            =
0x0400,
  EMBER_REQUIRE_ENCRYPTED_KEY                = 0x0800
,
  EMBER_NO_FRAME_COUNTER_RESET               =
0x1000,
  EMBER_GET_PRECONFIGURED_KEY_FROM_INSTALL_CODE
 = 0x2000,

#if !defined DOXYGEN_SHOULD_SKIP_THIS
  // Internal data
  EM_SAVED_IN_TOKEN                          = 0x4000,
  #define EM_SECURITY_INITIALIZED            0x00008000L

  // This is only used internally.  High security is not released or supported
  // except for golden unit compliance.
  #define EMBER_HIGH_SECURITY_MODE           0x0001
#else
  /* All other bits are reserved and must be zero. */
#endif
};

#ifdef DOXYGEN_SHOULD_SKIP_THIS
enum EmberExtendedSecurityBitmask
#else
typedef uint16_t EmberExtendedSecurityBitmask;
enum
#endif
{
#ifndef DOXYGEN_SHOULD_SKIP_THIS
  // If this bit is set, we set the 'key token data' field in the Initial
  // Security Bitmask to 0 (No Preconfig Key token), otherwise we leave the
  // field as it is.
  EMBER_PRECONFIG_KEY_NOT_VALID       = 0x0001,
#endif

  // bits 1-3 are unused.

  EMBER_JOINER_GLOBAL_LINK_KEY        = 0x0010,

  EMBER_EXT_NO_FRAME_COUNTER_RESET    = 0x0020,

  // bit 6-7 reserved for future use (stored in TOKEN).

  EMBER_NWK_LEAVE_REQUEST_NOT_ALLOWED =
0x0100,

#ifndef DOXYGEN_SHOULD_SKIP_THIS

  EMBER_R18_STACK_BEHAVIOR            = 0x0200,
#endif

  // bit 10 and 11 are stored in RAM only.
  // bit 11 is reserved for future use.
  // bits 12-15 are unused.
};

#define EMBER_NO_TRUST_CENTER_MODE   EMBER_DISTRIBUTED_TRUST_CENTER_MODE

#define EMBER_GLOBAL_LINK_KEY   EMBER_TRUST_CENTER_GLOBAL_LINK_KEY


#if !defined DOXYGEN_SHOULD_SKIP_THIS
  #define NO_TRUST_CENTER_KEY_TOKEN        0x0000
  #define TRUST_CENTER_KEY_TOKEN_MASK      0x0030
  #define SECURITY_BIT_TOKEN_MASK          0x71FF

  #define SECURITY_LOWER_BIT_MASK          0x000000FF  // ""
  #define SECURITY_UPPER_BIT_MASK          0x00FF0000L // ""
#endif

typedef struct {
  uint16_t bitmask;
  EmberKeyData preconfiguredKey;
  EmberKeyData networkKey;
  uint8_t networkKeySequenceNumber;
  EmberEUI64 preconfiguredTrustCenterEui64
;
} EmberInitialSecurityState;


#ifdef DOXYGEN_SHOULD_SKIP_THIS
enum EmberCurrentSecurityBitmask
#else
typedef uint16_t EmberCurrentSecurityBitmask;
enum
#endif
{
#if defined DOXYGEN_SHOULD_SKIP_THIS
  // These options are the same for Initial and Current Security state

  EMBER_STANDARD_SECURITY_MODE_             =
0x0000,
  EMBER_DISTRIBUTED_TRUST_CENTER_MODE_
 = 0x0002,
  EMBER_TRUST_CENTER_GLOBAL_LINK_KEY_
= 0x0004,
#else
  // Bit 3 reserved
#endif

  EMBER_HAVE_TRUST_CENTER_LINK_KEY          =
0x0010,

  EMBER_TRUST_CENTER_USES_HASHED_LINK_KEY_
   = 0x0084,

  // Bits 1,5,6, 8-15 reserved
};

#if !defined DOXYGEN_SHOULD_SKIP_THIS
  #define INITIAL_AND_CURRENT_BITMASK         0x00FF
#endif


typedef struct {
  EmberCurrentSecurityBitmask bitmask;
  EmberEUI64 trustCenterLongAddress;
} EmberCurrentSecurityState;


#ifdef DOXYGEN_SHOULD_SKIP_THIS
enum EmberKeyStructBitmask
#else
typedef uint16_t EmberKeyStructBitmask;
enum
#endif
{
  EMBER_KEY_HAS_SEQUENCE_NUMBER        = 0x0001,
  EMBER_KEY_HAS_OUTGOING_FRAME_COUNTER =
0x0002,
  EMBER_KEY_HAS_INCOMING_FRAME_COUNTER =
0x0004,
  EMBER_KEY_HAS_PARTNER_EUI64          = 0x0008,
  EMBER_KEY_IS_AUTHORIZED              = 0x0010,
  EMBER_KEY_PARTNER_IS_SLEEPY          = 0x0020,

};

#ifdef DOXYGEN_SHOULD_SKIP_THIS
enum EmberKeyType
#else
typedef uint8_t EmberKeyType;
enum
#endif
{
  EMBER_TRUST_CENTER_LINK_KEY         = 1,
  EMBER_TRUST_CENTER_MASTER_KEY       = 2,
  EMBER_CURRENT_NETWORK_KEY           = 3,
  EMBER_NEXT_NETWORK_KEY              = 4,
  EMBER_APPLICATION_LINK_KEY          = 5,
  EMBER_APPLICATION_MASTER_KEY        = 6,
};

typedef struct {
  EmberKeyStructBitmask bitmask;
  EmberKeyType type;
  EmberKeyData key;
  uint32_t outgoingFrameCounter;
  uint32_t incomingFrameCounter;
  uint8_t sequenceNumber;
  EmberEUI64 partnerEUI64;
} EmberKeyStruct;


#ifdef DOXYGEN_SHOULD_SKIP_THIS
enum EmberKeyStatus
#else
typedef uint8_t EmberKeyStatus;
enum
#endif
{
  EMBER_KEY_STATUS_NONE                       = 0,
  EMBER_APP_LINK_KEY_ESTABLISHED              = 1,
  EMBER_APP_MASTER_KEY_ESTABLISHED            = 2,
  EMBER_TRUST_CENTER_LINK_KEY_ESTABLISHED
  = 3,

  EMBER_KEY_ESTABLISHMENT_TIMEOUT             = 4,
  EMBER_KEY_TABLE_FULL                        = 5,

  // These are success status values applying only to the
  // Trust Center answering key requests
  EMBER_TC_RESPONDED_TO_KEY_REQUEST           = 6
,
  EMBER_TC_APP_KEY_SENT_TO_REQUESTER          =
7,

  // These are failure status values applying only to the
  // Trust Center answering key requests
  EMBER_TC_RESPONSE_TO_KEY_REQUEST_FAILED
  = 8,
  EMBER_TC_REQUEST_KEY_TYPE_NOT_SUPPORTED
  = 9,
  EMBER_TC_NO_LINK_KEY_FOR_REQUESTER          =
10,
  EMBER_TC_REQUESTER_EUI64_UNKNOWN            = 11
,
  EMBER_TC_RECEIVED_FIRST_APP_KEY_REQUEST
  = 12,
  EMBER_TC_TIMEOUT_WAITING_FOR_SECOND_APP_KEY_REQUEST
 = 13,
  EMBER_TC_NON_MATCHING_APP_KEY_REQUEST_RECEIVED
      = 14,
  EMBER_TC_FAILED_TO_SEND_APP_KEYS            = 15
,
  EMBER_TC_FAILED_TO_STORE_APP_KEY_REQUEST
 = 16,
  EMBER_TC_REJECTED_APP_KEY_REQUEST           =
17,
  EMBER_TC_FAILED_TO_GENERATE_NEW_KEY         =
 18,
  EMBER_TC_FAILED_TO_SEND_TC_KEY              = 19,

  // These are generic status values for a key requester.
  EMBER_TRUST_CENTER_IS_PRE_R21               = 30,

  // These are status values applying only to the Trust Center
  // verifying link keys.
  EMBER_TC_REQUESTER_VERIFY_KEY_TIMEOUT
 = 50,
  EMBER_TC_REQUESTER_VERIFY_KEY_FAILURE
 = 51,
  EMBER_TC_REQUESTER_VERIFY_KEY_SUCCESS
 = 52,

  // These are status values applying only to the key requester
  // verifying link keys.
  EMBER_VERIFY_LINK_KEY_FAILURE               = 100,
  EMBER_VERIFY_LINK_KEY_SUCCESS               = 101,
};

```

```
#ifdef DOXYGEN_SHOULD_SKIP_THIS
enum EmberLinkKeyRequestPolicy
#else
typedef uint8_t EmberLinkKeyRequestPolicy;
enum
#endif
{
  EMBER_DENY_KEY_REQUESTS  = 0x00,
  EMBER_ALLOW_KEY_REQUESTS = 0x01,
  EMBER_GENERATE_NEW_TC_LINK_KEY = 0x02,
};


#if defined DOXYGEN_SHOULD_SKIP_THIS
uint8_t* emberKeyContents(EmberKeyData* key);
#else
#define emberKeyContents(key) ((key)->contents)
#endif

#if defined DOXYGEN_SHOULD_SKIP_THIS
uint8_t* emberCertificateContents(EmberCertificateData
* cert);
#else
#define emberCertificateContents(cert) ((cert)->contents)
#endif

#if defined DOXYGEN_SHOULD_SKIP_THIS
uint8_t* emberPublicKeyContents(EmberPublicKeyData
* key);
#else
#define emberPublicKeyContents(key) ((key)->contents)
#endif

#if defined DOXYGEN_SHOULD_SKIP_THIS
uint8_t* emberPrivateKeyContents(EmberPrivateKeyData
* key);
#else
#define emberPrivateKeyContents(key) ((key)->contents)
#endif

#if defined DOXYGEN_SHOULD_SKIP_THIS
uint8_t* emberSmacContents(EmberSmacData* key);
#else
#define emberSmacContents(key) ((key)->contents)
#endif

#if defined DOXYGEN_SHOULD_SKIP_THIS
uint8_t* emberSignatureContents(EmberSignatureData
* sig);
#else
#define emberSignatureContents(sig) ((sig)->contents)
#endif

#if defined DOXYGEN_SHOULD_SKIP_THIS
uint8_t* emberCertificate283k1Contents(
EmberCertificate283k1Data* cert);
#else
#define emberCertificate283k1Contents(cert) ((cert)->contents)
#endif

#if defined DOXYGEN_SHOULD_SKIP_THIS
uint8_t* emberPublicKey283k1Contents(
EmberPublicKey283k1Data* key);
#else
#define emberPublicKey283k1Contents(key) ((key)->contents)
#endif

#if defined DOXYGEN_SHOULD_SKIP_THIS
uint8_t* emberPrivateKey283k1Contents(
EmberPrivateKey283k1Data* key);
#else
#define emberPrivateKey283k1Contents(key) ((key)->contents)
#endif

#if defined DOXYGEN_SHOULD_SKIP_THIS
uint8_t* ember283k1SignatureContents(
Ember283k1SignatureData* sig);
#else
#define ember283k1SignatureContents(sig) ((sig)->contents)
#endif
```

```

#ifdef DOXYGEN_SHOULD_SKIP_THIS
enum EmberKeySettings
#else
typedef uint16_t EmberKeySettings;
enum
#endif
{
  EMBER_KEY_PERMISSIONS_NONE            = 0x0000,
  EMBER_KEY_PERMISSIONS_READING_ALLOWED =
0x0001,
  EMBER_KEY_PERMISSIONS_HASHING_ALLOWED =
0x0002,
};


typedef struct {
  EmberKeySettings keySettings;
} EmberMfgSecurityStruct;


#define EMBER_MFG_SECURITY_CONFIG_MAGIC_NUMBER 0xCABAD11FUL


#ifdef DOXYGEN_SHOULD_SKIP_THIS
enum EmberMacPassthroughType
#else
typedef uint8_t EmberMacPassthroughType;
enum
#endif
{
  EMBER_MAC_PASSTHROUGH_NONE             = 0x00,
  EMBER_MAC_PASSTHROUGH_SE_INTERPAN      =
0x01,
  EMBER_MAC_PASSTHROUGH_EMBERNET         = 0x02,
  EMBER_MAC_PASSTHROUGH_EMBERNET_SOURCE  =
 0x04,
  EMBER_MAC_PASSTHROUGH_APPLICATION      =
0x08,
  EMBER_MAC_PASSTHROUGH_CUSTOM           = 0x10,

#if !defined DOXYGEN_SHOULD_SKIP_THIS

  EM_MAC_PASSTHROUGH_INTERNAL_ZLL        = 0x80,
  EM_MAC_PASSTHROUGH_INTERNAL_GP         = 0x40
#endif
};

typedef uint16_t EmberMacFilterMatchData;

#define EMBER_MAC_FILTER_MATCH_ENABLED_MASK            0x0001
#define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_MASK        0x0003
#define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_MASK      0x000C
#define EMBER_MAC_FILTER_MATCH_ON_DEST_MASK            0x0030
#define EMBER_MAC_FILTER_MATCH_ON_SOURCE_MASK          0x0080

// Globally turn on/off this filter
#define EMBER_MAC_FILTER_MATCH_ENABLED                 0x0000
#define EMBER_MAC_FILTER_MATCH_DISABLED                0x0001

// Pick either one of these
#define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_NONE        0x0000
#define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_LOCAL       0x0001
#define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_BROADCAST   0x0002

// and one of these
#define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_NONE      0x0000
#define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_NON_LOCAL 0x0004
#define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_LOCAL     0x0008

// and one of these
#define EMBER_MAC_FILTER_MATCH_ON_DEST_BROADCAST_SHORT 0x0000
#define EMBER_MAC_FILTER_MATCH_ON_DEST_UNICAST_SHORT   0x0010
#define EMBER_MAC_FILTER_MATCH_ON_DEST_UNICAST_LONG    0x0020

// and one of these
#define EMBER_MAC_FILTER_MATCH_ON_SOURCE_LONG          0x0000
#define EMBER_MAC_FILTER_MATCH_ON_SOURCE_SHORT         0x0080
#define EMBER_MAC_FILTER_MATCH_ON_SOURCE_NONE          0x0100

```

```
// Last entry should set this and nothing else.  No other bits will be
 examined.
#define EMBER_MAC_FILTER_MATCH_END                     0x8000

typedef struct {
  uint8_t filterIndexMatch;
  EmberMacPassthroughType legacyPassthroughType
;
  EmberMessageBuffer message;
} EmberMacFilterMatchStruct;


typedef uint8_t EmberLibraryStatus;


#ifdef DOXYGEN_SHOULD_SKIP_THIS
enum EmberZdoStatus
#else
typedef uint8_t EmberZdoStatus;
enum
#endif
{
  // These values are taken from Table 48 of ZDP Errata 043238r003 and Table 2
  // of NWK 02130r10.
  EMBER_ZDP_SUCCESS               = 0x00,
  // 0x01 to 0x7F are reserved
  EMBER_ZDP_INVALID_REQUEST_TYPE = 0x80,
  EMBER_ZDP_DEVICE_NOT_FOUND     = 0x81,
  EMBER_ZDP_INVALID_ENDPOINT     = 0x82,
  EMBER_ZDP_NOT_ACTIVE           = 0x83,
  EMBER_ZDP_NOT_SUPPORTED        = 0x84,
  EMBER_ZDP_TIMEOUT              = 0x85,
  EMBER_ZDP_NO_MATCH             = 0x86,
  // 0x87 is reserved            = 0x87,
  EMBER_ZDP_NO_ENTRY             = 0x88,
  EMBER_ZDP_NO_DESCRIPTOR        = 0x89,
  EMBER_ZDP_INSUFFICIENT_SPACE   = 0x8a,
  EMBER_ZDP_NOT_PERMITTED        = 0x8b,
  EMBER_ZDP_TABLE_FULL           = 0x8c,
  EMBER_ZDP_NOT_AUTHORIZED       = 0x8d,

  EMBER_NWK_ALREADY_PRESENT      = 0xC5,
  EMBER_NWK_TABLE_FULL           = 0xC7,
  EMBER_NWK_UNKNOWN_DEVICE       = 0xC8
};
















#define NETWORK_ADDRESS_REQUEST      0x0000
#define NETWORK_ADDRESS_RESPONSE     0x8000
#define IEEE_ADDRESS_REQUEST         0x0001
#define IEEE_ADDRESS_RESPONSE        0x8001


  //            <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      0x0002
#define NODE_DESCRIPTOR_RESPONSE     0x8002


  //     See ZigBee document 053474, Section 2.3.2.4 for more details.
#define POWER_DESCRIPTOR_REQUEST     0x0003
#define POWER_DESCRIPTOR_RESPONSE    0x8003


#define SIMPLE_DESCRIPTOR_REQUEST    0x0004
#define SIMPLE_DESCRIPTOR_RESPONSE   0x8004


#define ACTIVE_ENDPOINTS_REQUEST     0x0005
#define ACTIVE_ENDPOINTS_RESPONSE    0x8005


#define MATCH_DESCRIPTORS_REQUEST    0x0006
#define MATCH_DESCRIPTORS_RESPONSE   0x8006


#define DISCOVERY_CACHE_REQUEST      0x0012
#define DISCOVERY_CACHE_RESPONSE     0x8012


#define END_DEVICE_ANNOUNCE          0x0013
#define END_DEVICE_ANNOUNCE_RESPONSE 0x8013


#define SYSTEM_SERVER_DISCOVERY_REQUEST  0x0015
#define SYSTEM_SERVER_DISCOVERY_RESPONSE 0x8015


#define PARENT_ANNOUNCE           0x001F
#define PARENT_ANNOUNCE_RESPONSE  0x801F


#ifdef DOXYGEN_SHOULD_SKIP_THIS
enum EmberZdoServerMask
#else
typedef uint16_t EmberZdoServerMask;
enum
#endif
{
  EMBER_ZDP_PRIMARY_TRUST_CENTER          =
0x0001,
  EMBER_ZDP_SECONDARY_TRUST_CENTER        =
0x0002,
  EMBER_ZDP_PRIMARY_BINDING_TABLE_CACHE
= 0x0004,
  EMBER_ZDP_SECONDARY_BINDING_TABLE_CACHE
 = 0x0008,
  EMBER_ZDP_PRIMARY_DISCOVERY_CACHE       =
0x0010,
  EMBER_ZDP_SECONDARY_DISCOVERY_CACHE     =
0x0020,
  EMBER_ZDP_NETWORK_MANAGER               = 0x0040,
  // Bits 0x0080 to 0x8000 are reserved.
};

#define FIND_NODE_CACHE_REQUEST         0x001C
#define FIND_NODE_CACHE_RESPONSE        0x801C


#define END_DEVICE_BIND_REQUEST      0x0020
#define END_DEVICE_BIND_RESPONSE     0x8020


#define UNICAST_BINDING             0x03
#define UNICAST_MANY_TO_ONE_BINDING 0x83
#define MULTICAST_BINDING           0x01

#define BIND_REQUEST                 0x0021
#define BIND_RESPONSE                0x8021
#define UNBIND_REQUEST               0x0022
#define UNBIND_RESPONSE              0x8022
```

```


#define LQI_TABLE_REQUEST            0x0031
#define LQI_TABLE_RESPONSE           0x8031


#define ROUTING_TABLE_REQUEST        0x0032
#define ROUTING_TABLE_RESPONSE       0x8032


#define BINDING_TABLE_REQUEST        0x0033
#define BINDING_TABLE_RESPONSE       0x8033


#define LEAVE_REQUEST                0x0034
#define LEAVE_RESPONSE               0x8034

#define LEAVE_REQUEST_REMOVE_CHILDREN_FLAG 0x40
#define LEAVE_REQUEST_REJOIN_FLAG          0x80


#define PERMIT_JOINING_REQUEST       0x0036
#define PERMIT_JOINING_RESPONSE      0x8036


#define NWK_UPDATE_REQUEST           0x0038
#define NWK_UPDATE_RESPONSE          0x8038


#define COMPLEX_DESCRIPTOR_REQUEST   0x0010
#define COMPLEX_DESCRIPTOR_RESPONSE  0x8010
#define USER_DESCRIPTOR_REQUEST      0x0011
#define USER_DESCRIPTOR_RESPONSE     0x8011
#define DISCOVERY_REGISTER_REQUEST   0x0012
#define DISCOVERY_REGISTER_RESPONSE  0x8012
#define USER_DESCRIPTOR_SET          0x0014
#define USER_DESCRIPTOR_CONFIRM      0x8014
#define NETWORK_DISCOVERY_REQUEST    0x0030
#define NETWORK_DISCOVERY_RESPONSE   0x8030
#define DIRECT_JOIN_REQUEST          0x0035
#define DIRECT_JOIN_RESPONSE         0x8035


#define CLUSTER_ID_RESPONSE_MINIMUM  0x8000


#ifdef DOXYGEN_SHOULD_SKIP_THIS
enum EmberZdoConfigurationFlags
#else
typedef uint8_t EmberZdoConfigurationFlags;
enum
#endif

{
 EMBER_APP_RECEIVES_SUPPORTED_ZDO_REQUESTS
  = 0x01,
 EMBER_APP_HANDLES_UNSUPPORTED_ZDO_REQUESTS
 = 0x02,
 EMBER_APP_HANDLES_ZDO_ENDPOINT_REQUESTS
    = 0x04,
 EMBER_APP_HANDLES_ZDO_BINDING_REQUESTS
   = 0x08
};


#if defined(EMBER_TEST)
#define WEAK_TEST WEAK()//__attribute__((weak))
#else
#define WEAK_TEST
#endif



#endif // EMBER_TYPES_H

#include "stack/include/zll-types.h"
#include "stack/include/rf4ce-types.h"
#include "stack/include/gp-types.h"

```

8.19 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)

8.19.1 Detailed Description

Return-code definitions for EmberZNet stack API functions. See Ember Status Codes for documentation.

Definition in file error-def.h.

8.20 error-def.h

```


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_SUCCESS(0x00)
#else
DEFINE_ERROR(SUCCESS, 0)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_ERR_FATAL(0x01)
#else
DEFINE_ERROR(ERR_FATAL, 0x01)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_BAD_ARGUMENT(0x02)
#else
DEFINE_ERROR(BAD_ARGUMENT, 0x02)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_NOT_FOUND(0x03)
#else
DEFINE_ERROR(NOT_FOUND, 0x03)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_EEPROM_MFG_STACK_VERSION_MISMATCH(0x04)
#else
DEFINE_ERROR(EEPROM_MFG_STACK_VERSION_MISMATCH, 0x04)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_INCOMPATIBLE_STATIC_MEMORY_DEFINITIONS(0x05)
#else
DEFINE_ERROR(INCOMPATIBLE_STATIC_MEMORY_DEFINITIONS, 0x05)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_EEPROM_MFG_VERSION_MISMATCH(0x06)
#else
DEFINE_ERROR(EEPROM_MFG_VERSION_MISMATCH, 0x06)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_EEPROM_STACK_VERSION_MISMATCH(0x07)
#else
DEFINE_ERROR(EEPROM_STACK_VERSION_MISMATCH, 0x07)
#endif //DOXYGEN_SHOULD_SKIP_THIS

```

```



#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_NO_BUFFERS(0x18)
#else
DEFINE_ERROR(NO_BUFFERS, 0x18)
#endif //DOXYGEN_SHOULD_SKIP_THIS



#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_SERIAL_INVALID_BAUD_RATE(0x20)
#else
DEFINE_ERROR(SERIAL_INVALID_BAUD_RATE, 0x20)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_SERIAL_INVALID_PORT(0x21)
#else
DEFINE_ERROR(SERIAL_INVALID_PORT, 0x21)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_SERIAL_TX_OVERFLOW(0x22)
#else
DEFINE_ERROR(SERIAL_TX_OVERFLOW, 0x22)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_SERIAL_RX_OVERFLOW(0x23)
#else
DEFINE_ERROR(SERIAL_RX_OVERFLOW, 0x23)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_SERIAL_RX_FRAME_ERROR(0x24)
#else
DEFINE_ERROR(SERIAL_RX_FRAME_ERROR, 0x24)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_SERIAL_RX_PARITY_ERROR(0x25)
#else
DEFINE_ERROR(SERIAL_RX_PARITY_ERROR, 0x25)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_SERIAL_RX_EMPTY(0x26)
#else
DEFINE_ERROR(SERIAL_RX_EMPTY, 0x26)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_SERIAL_RX_OVERRUN_ERROR(0x27)
#else
DEFINE_ERROR(SERIAL_RX_OVERRUN_ERROR, 0x27)
#endif //DOXYGEN_SHOULD_SKIP_THIS



#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_MAC_TRANSMIT_QUEUE_FULL(0x39)
```

```
#else
// Internal
DEFINE_ERROR(MAC_TRANSMIT_QUEUE_FULL, 0x39)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_MAC_UNKNOWN_HEADER_TYPE(0x3A)
#else
DEFINE_ERROR(MAC_UNKNOWN_HEADER_TYPE, 0x3A)
#endif //DOXYGEN_SHOULD_SKIP_THIS

#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_MAC_ACK_HEADER_TYPE(0x3B)
#else
DEFINE_ERROR(MAC_ACK_HEADER_TYPE,  0x3B)
#endif //DOXYGEN_SHOULD_SKIP_THIS



#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_MAC_SCANNING(0x3D)
#else
DEFINE_ERROR(MAC_SCANNING, 0x3D)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_MAC_NO_DATA(0x31)
#else
DEFINE_ERROR(MAC_NO_DATA, 0x31)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_MAC_JOINED_NETWORK(0x32)
#else
DEFINE_ERROR(MAC_JOINED_NETWORK, 0x32)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_MAC_BAD_SCAN_DURATION(0x33)
#else
DEFINE_ERROR(MAC_BAD_SCAN_DURATION, 0x33)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_MAC_INCORRECT_SCAN_TYPE(0x34)
#else
DEFINE_ERROR(MAC_INCORRECT_SCAN_TYPE, 0x34)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_MAC_INVALID_CHANNEL_MASK(0x35)
#else
DEFINE_ERROR(MAC_INVALID_CHANNEL_MASK, 0x35)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_MAC_COMMAND_TRANSMIT_FAILURE(0x36)
#else
DEFINE_ERROR(MAC_COMMAND_TRANSMIT_FAILURE, 0x36)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

```

```
#define EMBER_MAC_NO_ACK_RECEIVED(0x40)
#else
DEFINE_ERROR(MAC_NO_ACK_RECEIVED, 0x40)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_MAC_RADIO_NETWORK_SWITCH_FAILED(0x41)
#else
DEFINE_ERROR(MAC_RADIO_NETWORK_SWITCH_FAILED, 0x41)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_MAC_INDIRECT_TIMEOUT(0x42)
#else
DEFINE_ERROR(MAC_INDIRECT_TIMEOUT, 0x42)
#endif //DOXYGEN_SHOULD_SKIP_THIS






#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_SIM_EEPROM_ERASE_PAGE_GREEN(0x43)
#else
DEFINE_ERROR(SIM_EEPROM_ERASE_PAGE_GREEN, 0x43)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_SIM_EEPROM_ERASE_PAGE_RED(0x44)
#else
DEFINE_ERROR(SIM_EEPROM_ERASE_PAGE_RED, 0x44)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_SIM_EEPROM_FULL(0x45)
#else
DEFINE_ERROR(SIM_EEPROM_FULL, 0x45)
#endif //DOXYGEN_SHOULD_SKIP_THIS


//  Errors 46 and 47 are now defined below in the
//    flash error block (was attempting to prevent renumbering)


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_SIM_EEPROM_INIT_1_FAILED(0x48)
#else
DEFINE_ERROR(SIM_EEPROM_INIT_1_FAILED, 0x48)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_SIM_EEPROM_INIT_2_FAILED(0x49)
#else
DEFINE_ERROR(SIM_EEPROM_INIT_2_FAILED, 0x49)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_SIM_EEPROM_INIT_3_FAILED(0x4A)
#else
DEFINE_ERROR(SIM_EEPROM_INIT_3_FAILED, 0x4A)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_SIM_EEPROM_REPAIRING(0x4D)
```

```
#else
DEFINE_ERROR(SIM_EEPROM_REPAIRING, 0x4D)
#endif //DOXYGEN_SHOULD_SKIP_THIS




#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_ERR_FLASH_WRITE_INHIBITED(0x46)
#else
DEFINE_ERROR(ERR_FLASH_WRITE_INHIBITED, 0x46)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_ERR_FLASH_VERIFY_FAILED(0x47)
#else
DEFINE_ERROR(ERR_FLASH_VERIFY_FAILED, 0x47)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_ERR_FLASH_PROG_FAIL(0x4B)
#else
DEFINE_ERROR(ERR_FLASH_PROG_FAIL, 0x4B)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_ERR_FLASH_ERASE_FAIL(0x4C)
#else
DEFINE_ERROR(ERR_FLASH_ERASE_FAIL, 0x4C)
#endif //DOXYGEN_SHOULD_SKIP_THIS





#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_ERR_BOOTLOADER_TRAP_TABLE_BAD(0x58)
#else
DEFINE_ERROR(ERR_BOOTLOADER_TRAP_TABLE_BAD, 0x58)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_ERR_BOOTLOADER_TRAP_UNKNOWN(0x59)
#else
DEFINE_ERROR(ERR_BOOTLOADER_TRAP_UNKNOWN, 0x59)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_ERR_BOOTLOADER_NO_IMAGE(0x05A)
#else
DEFINE_ERROR(ERR_BOOTLOADER_NO_IMAGE, 0x5A)
#endif //DOXYGEN_SHOULD_SKIP_THIS




#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_DELIVERY_FAILED(0x66)
#else
DEFINE_ERROR(DELIVERY_FAILED, 0x66)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_BINDING_INDEX_OUT_OF_RANGE(0x69)
#else
```

```
DEFINE_ERROR(BINDING_INDEX_OUT_OF_RANGE, 0x69)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_ADDRESS_TABLE_INDEX_OUT_OF_RANGE(0x6A)
#else
DEFINE_ERROR(ADDRESS_TABLE_INDEX_OUT_OF_RANGE, 0x6A)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_INVALID_BINDING_INDEX(0x6C)
#else
DEFINE_ERROR(INVALID_BINDING_INDEX, 0x6C)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_INVALID_CALL(0x70)
#else
DEFINE_ERROR(INVALID_CALL, 0x70)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_COST_NOT_KNOWN(0x71)
#else
DEFINE_ERROR(COST_NOT_KNOWN, 0x71)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_MAX_MESSAGE_LIMIT_REACHED(0x72)
#else
DEFINE_ERROR(MAX_MESSAGE_LIMIT_REACHED, 0x72)
#endif //DOXYGEN_SHOULD_SKIP_THIS

#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_MESSAGE_TOO_LONG(0x74)
#else
DEFINE_ERROR(MESSAGE_TOO_LONG, 0x74)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_BINDING_IS_ACTIVE(0x75)
#else
DEFINE_ERROR(BINDING_IS_ACTIVE, 0x75)
#endif //DOXYGEN_SHOULD_SKIP_THIS

#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_ADDRESS_TABLE_ENTRY_IS_ACTIVE(0x76)
#else
DEFINE_ERROR(ADDRESS_TABLE_ENTRY_IS_ACTIVE, 0x76)
#endif //DOXYGEN_SHOULD_SKIP_THIS

//


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_MATCH(0x78)
#else
DEFINE_ERROR(MATCH, 0x78)
#endif //DOXYGEN_SHOULD_SKIP_THIS
#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_DROP_FRAME(0x79)
#else
DEFINE_ERROR(DROP_FRAME, 0x79)
#endif //DOXYGEN_SHOULD_SKIP_THIS
```

```

#ifdef DOXYGEN_SHOULD_SKIP_THIS
#define EMBER_PASS_UNPROCESSED(0x7A)
#else
DEFINE_ERROR(PASS_UNPROCESSED, 0x7A)
#endif //DOXYGEN_SHOULD_SKIP_THIS

#ifdef DOXYGEN_SHOULD_SKIP_THIS
#define EMBER_TX_THEN_DROP(0x7B)
#else
DEFINE_ERROR(TX_THEN_DROP, 0x7B)
#endif //DOXYGEN_SHOULD_SKIP_THIS

#ifdef DOXYGEN_SHOULD_SKIP_THIS
#define EMBER_NO_SECURITY(0x7C)
#else
DEFINE_ERROR(NO_SECURITY, 0x7C)
#endif //DOXYGEN_SHOULD_SKIP_THIS

#ifdef DOXYGEN_SHOULD_SKIP_THIS
#define EMBER_COUNTER_FAILURE(0x7D)
#else
DEFINE_ERROR(COUNTER_FAILURE, 0x7D)
#endif //DOXYGEN_SHOULD_SKIP_THIS

#ifdef DOXYGEN_SHOULD_SKIP_THIS
#define EMBER_AUTH_FAILURE(0x7E)
#else
DEFINE_ERROR(AUTH_FAILURE, 0x7E)
#endif //DOXYGEN_SHOULD_SKIP_THIS

#ifdef DOXYGEN_SHOULD_SKIP_THIS
#define EMBER_UNPROCESSED(0x7F)
#else
DEFINE_ERROR(UNPROCESSED, 0x7F)
#endif //DOXYGEN_SHOULD_SKIP_THIS

//



#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_ADC_CONVERSION_DONE(0x80)
#else
DEFINE_ERROR(ADC_CONVERSION_DONE, 0x80)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_ADC_CONVERSION_BUSY(0x81)
#else
DEFINE_ERROR(ADC_CONVERSION_BUSY, 0x81)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_ADC_CONVERSION_DEFERRED(0x82)
#else
DEFINE_ERROR(ADC_CONVERSION_DEFERRED, 0x82)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_ADC_NO_CONVERSION_PENDING(0x84)
#else
DEFINE_ERROR(ADC_NO_CONVERSION_PENDING, 0x84)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_SLEEP_INTERRUPTED(0x85)
#else
DEFINE_ERROR(SLEEP_INTERRUPTED, 0x85)
#endif //DOXYGEN_SHOULD_SKIP_THIS

```

```



#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_PHY_TX_UNDERFLOW(0x88)
#else
DEFINE_ERROR(PHY_TX_UNDERFLOW, 0x88)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_PHY_TX_INCOMPLETE(0x89)
#else
DEFINE_ERROR(PHY_TX_INCOMPLETE, 0x89)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_PHY_INVALID_CHANNEL(0x8A)
#else
DEFINE_ERROR(PHY_INVALID_CHANNEL, 0x8A)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_PHY_INVALID_POWER(0x8B)
#else
DEFINE_ERROR(PHY_INVALID_POWER, 0x8B)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_PHY_TX_BUSY(0x8C)
#else
DEFINE_ERROR(PHY_TX_BUSY, 0x8C)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_PHY_TX_CCA_FAIL(0x8D)
#else
DEFINE_ERROR(PHY_TX_CCA_FAIL, 0x8D)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_PHY_OSCILLATOR_CHECK_FAILED(0x8E)
#else
DEFINE_ERROR(PHY_OSCILLATOR_CHECK_FAILED, 0x8E)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_PHY_ACK_RECEIVED(0x8F)
#else
DEFINE_ERROR(PHY_ACK_RECEIVED, 0x8F)
#endif //DOXYGEN_SHOULD_SKIP_THIS




#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_NETWORK_UP(0x90)
#else
DEFINE_ERROR(NETWORK_UP, 0x90)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_NETWORK_DOWN(0x91)
```

```
#else
DEFINE_ERROR(NETWORK_DOWN, 0x91)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_JOIN_FAILED(0x94)
#else
DEFINE_ERROR(JOIN_FAILED, 0x94)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_MOVE_FAILED(0x96)
#else
DEFINE_ERROR(MOVE_FAILED, 0x96)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_CANNOT_JOIN_AS_ROUTER(0x98)
#else
DEFINE_ERROR(CANNOT_JOIN_AS_ROUTER, 0x98)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_NODE_ID_CHANGED(0x99)
#else
DEFINE_ERROR(NODE_ID_CHANGED, 0x99)
#endif


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_PAN_ID_CHANGED(0x9A)
#else
DEFINE_ERROR(PAN_ID_CHANGED, 0x9A)
#endif

#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_CHANNEL_CHANGED(0x9B)
#else
DEFINE_ERROR(CHANNEL_CHANGED, 0x9B)
#endif

#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_NO_BEACONS(0xAB)
#else
DEFINE_ERROR(NO_BEACONS, 0xAB)
#endif


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_RECEIVED_KEY_IN_THE_CLEAR(0xAC)
#else
DEFINE_ERROR(RECEIVED_KEY_IN_THE_CLEAR, 0xAC)
#endif


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_NO_NETWORK_KEY_RECEIVED(0xAD)
#else
DEFINE_ERROR(NO_NETWORK_KEY_RECEIVED, 0xAD)
#endif


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_NO_LINK_KEY_RECEIVED(0xAE)
#else
DEFINE_ERROR(NO_LINK_KEY_RECEIVED, 0xAE)
```

```
#endif


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_PRECONFIGURED_KEY_REQUIRED(0xAF)
#else
DEFINE_ERROR(PRECONFIGURED_KEY_REQUIRED, 0xAF)
#endif



#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_KEY_INVALID(0xB2)
#else
DEFINE_ERROR(KEY_INVALID, 0xB2)
#endif // DOXYGEN_SHOULD_SKIP_THIS

#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_INVALID_SECURITY_LEVEL(0x95)
#else
DEFINE_ERROR(INVALID_SECURITY_LEVEL, 0x95)
#endif //DOXYGEN_SHOULD_SKIP_THIS

#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_APS_ENCRYPTION_ERROR(0xA6)
#else
     DEFINE_ERROR(APS_ENCRYPTION_ERROR, 0xA6)
#endif //DOXYGEN_SHOULD_SKIP_THIS

#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_TRUST_CENTER_MASTER_KEY_NOT_SET(0xA7)
#else
     DEFINE_ERROR(TRUST_CENTER_MASTER_KEY_NOT_SET, 0xA7)
#endif //DOXYGEN_SHOULD_SKIP_THIS

#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_SECURITY_STATE_NOT_SET(0xA8)
#else
     DEFINE_ERROR(SECURITY_STATE_NOT_SET, 0xA8)
#endif //DOXYGEN_SHOULD_SKIP_THIS

#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_KEY_TABLE_INVALID_ADDRESS(0xB3)
#else
DEFINE_ERROR(KEY_TABLE_INVALID_ADDRESS, 0xB3)
#endif //DOXYGEN_SHOULD_SKIP_THIS

#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_SECURITY_CONFIGURATION_INVALID(0xB7)
#else
DEFINE_ERROR(SECURITY_CONFIGURATION_INVALID, 0xB7)
#endif //DOXYGEN_SHOULD_SKIP_THIS

#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_TOO_SOON_FOR_SWITCH_KEY(0xB8)
#else
     DEFINE_ERROR(TOO_SOON_FOR_SWITCH_KEY, 0xB8)
#endif

#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_SIGNATURE_VERIFY_FAILURE(0xB9)
#else
     DEFINE_ERROR(SIGNATURE_VERIFY_FAILURE, 0xB9)
#endif

#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_KEY_NOT_AUTHORIZED(0xBB)
#else
     DEFINE_ERROR(KEY_NOT_AUTHORIZED, 0xBB)
```

```
#endif


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_SECURITY_DATA_INVALID(0xBD)
#else
     DEFINE_ERROR(SECURITY_DATA_INVALID, 0xBD)
#endif





#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_NOT_JOINED(0x93)
#else
DEFINE_ERROR(NOT_JOINED, 0x93)
#endif //DOXYGEN_SHOULD_SKIP_THIS

#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_NETWORK_BUSY(0xA1)
#else
DEFINE_ERROR(NETWORK_BUSY, 0xA1)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_INVALID_ENDPOINT(0xA3)
#else
DEFINE_ERROR(INVALID_ENDPOINT, 0xA3)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_BINDING_HAS_CHANGED(0xA4)
#else
DEFINE_ERROR(BINDING_HAS_CHANGED, 0xA4)
#endif //DOXYGEN_SHOULD_SKIP_THIS

#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_INSUFFICIENT_RANDOM_DATA(0xA5)
#else
     DEFINE_ERROR(INSUFFICIENT_RANDOM_DATA, 0xA5)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_SOURCE_ROUTE_FAILURE(0xA9)
#else
     DEFINE_ERROR(SOURCE_ROUTE_FAILURE, 0xA9)
#endif

#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_MANY_TO_ONE_ROUTE_FAILURE(0xAA)
#else
     DEFINE_ERROR(MANY_TO_ONE_ROUTE_FAILURE, 0xAA)
#endif





#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_STACK_AND_HARDWARE_MISMATCH(0xB0)
#else
DEFINE_ERROR(STACK_AND_HARDWARE_MISMATCH, 0xB0)
#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef DOXYGEN_SHOULD_SKIP_THIS

```

```
#define EMBER_INDEX_OUT_OF_RANGE(0xB1)
#else
DEFINE_ERROR(INDEX_OUT_OF_RANGE, 0xB1)
#endif

#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_TABLE_FULL(0xB4)
#else
DEFINE_ERROR(TABLE_FULL, 0xB4)
#endif //DOXYGEN_SHOULD_SKIP_THIS

#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_TABLE_ENTRY_ERASED(0xB6)
#else
DEFINE_ERROR(TABLE_ENTRY_ERASED, 0xB6)
#endif

#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_LIBRARY_NOT_PRESENT(0xB5)
#else
DEFINE_ERROR(LIBRARY_NOT_PRESENT, 0xB5)
#endif

#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_OPERATION_IN_PROGRESS(0xBA)
#else
DEFINE_ERROR(OPERATION_IN_PROGRESS, 0xBA)
#endif

#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_TRUST_CENTER_EUI_HAS_CHANGED(0xBC)
#else
     DEFINE_ERROR(TRUST_CENTER_EUI_HAS_CHANGED, 0xBC)
#endif




#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_NO_RESPONSE(0xC0)
#else
     DEFINE_ERROR(NO_RESPONSE, 0xC0)
#endif

#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_DUPLICATE_ENTRY(0xC1)
#else
     DEFINE_ERROR(DUPLICATE_ENTRY, 0xC1)
#endif

#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_NOT_PERMITTED(0xC2)
#else
     DEFINE_ERROR(NOT_PERMITTED, 0xC2)
#endif

#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_DISCOVERY_TIMEOUT(0xC3)
#else
     DEFINE_ERROR(DISCOVERY_TIMEOUT, 0xC3)
#endif


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_DISCOVERY_ERROR(0xC4)
#else
     DEFINE_ERROR(DISCOVERY_ERROR, 0xC4)
#endif


#ifdef DOXYGEN_SHOULD_SKIP_THIS
```

```

#define EMBER_SECURITY_TIMEOUT(0xC5)
#else
     DEFINE_ERROR(SECURITY_TIMEOUT, 0xC5)
#endif


#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_SECURITY_FAILURE(0xC6)
#else
     DEFINE_ERROR(SECURITY_FAILURE, 0xC6)
#endif



#ifdef DOXYGEN_SHOULD_SKIP_THIS

#define EMBER_APPLICATION_ERROR_0(0xF0)
#define EMBER_APPLICATION_ERROR_1(0xF1)
#define EMBER_APPLICATION_ERROR_2(0xF2)
#define EMBER_APPLICATION_ERROR_3(0xF3)
#define EMBER_APPLICATION_ERROR_4(0xF4)
#define EMBER_APPLICATION_ERROR_5(0xF5)
#define EMBER_APPLICATION_ERROR_6(0xF6)
#define EMBER_APPLICATION_ERROR_7(0xF7)
#define EMBER_APPLICATION_ERROR_8(0xF8)
#define EMBER_APPLICATION_ERROR_9(0xF9)
#define EMBER_APPLICATION_ERROR_10(0xFA)
#define EMBER_APPLICATION_ERROR_11(0xFB)
#define EMBER_APPLICATION_ERROR_12(0xFC)
#define EMBER_APPLICATION_ERROR_13(0xFD)
#define EMBER_APPLICATION_ERROR_14(0xFE)
#define EMBER_APPLICATION_ERROR_15(0xFF)
#else
DEFINE_ERROR( APPLICATION_ERROR_0, 0xF0)
DEFINE_ERROR( APPLICATION_ERROR_1, 0xF1)
DEFINE_ERROR( APPLICATION_ERROR_2, 0xF2)
DEFINE_ERROR( APPLICATION_ERROR_3, 0xF3)
DEFINE_ERROR( APPLICATION_ERROR_4, 0xF4)
DEFINE_ERROR( APPLICATION_ERROR_5, 0xF5)
DEFINE_ERROR( APPLICATION_ERROR_6, 0xF6)
DEFINE_ERROR( APPLICATION_ERROR_7, 0xF7)
DEFINE_ERROR( APPLICATION_ERROR_8, 0xF8)
DEFINE_ERROR( APPLICATION_ERROR_9, 0xF9)
DEFINE_ERROR( APPLICATION_ERROR_10, 0xFA)
DEFINE_ERROR( APPLICATION_ERROR_11, 0xFB)
DEFINE_ERROR( APPLICATION_ERROR_12, 0xFC)
DEFINE_ERROR( APPLICATION_ERROR_13, 0xFD)
DEFINE_ERROR( APPLICATION_ERROR_14, 0xFE)
DEFINE_ERROR( APPLICATION_ERROR_15, 0xFF)
#endif //DOXYGEN_SHOULD_SKIP_THIS


```

8.21 error.h File Reference

Macros

- #define __EMBERSTATUS_TYPE__
- #define DEFINE_ERROR(symbol, value)

Typedefs

- typedef uint8_t EmberStatus

Enumerations

- enum { EMBER_ERROR_CODE_COUNT }

8.21.1 Detailed Description

Return codes for Ember API functions and module definitions. See Ember Status Codes for documentation.

Definition in file error.h.

8.21.2 Macro Definition Documentation

8.21.2.1 #define __EMBERSTATUS_TYPE__

Return type for Ember functions.

Definition at line 18 of file error.h.

8.21.3 Typedef Documentation

8.21.3.1 typedef uint8_t EmberStatus

Definition at line 19 of file error.h.

8.22 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
00046   EMBER_ERROR_CODE_COUNT
00047
00048 };
00049
00050 #undef DEFINE_ERROR
00051
00052 #endif // __ERRORS_H__
00053
```

8.23 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

8.23.1 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.

8.24 ezsp-host-configuration-defaults.h

```

#ifdef CONFIGURATION_HEADER
  #include CONFIGURATION_HEADER
#endif

#ifndef EZSP_HOST_SOURCE_ROUTE_TABLE_SIZE

  #define EZSP_HOST_SOURCE_ROUTE_TABLE_SIZE 32
#endif

#ifndef EZSP_HOST_RX_POOL_SIZE

  #define EZSP_HOST_RX_POOL_SIZE 20
#endif

#ifndef EZSP_HOST_FORM_AND_JOIN_BUFFER_SIZE

  #define EZSP_HOST_FORM_AND_JOIN_BUFFER_SIZE 40
#endif

```

8.25 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 maxRssiValue)
- void emberFormAndJoinTick (void)
- void emberFormAndJoinTaskInit (void)
- void emberFormAndJoinRunTask (void)
- void emberFormAndJoinCleanup (EmberStatus status)

Variables

- bool emberEnableDualChannelScan

8.25.1 Detailed Description

Utilities for forming and joining networks. See Forming and Joining Networks for documentation.

Definition in file form-and-join.h.

8.26 form-and-join.h

```

#define NETWORK_STORAGE_SIZE  16

#define NETWORK_STORAGE_SIZE_SHIFT 4

#ifndef FORM_AND_JOIN_MAX_NETWORKS
  #ifdef EZSP_HOST
    // the host's buffer is 16-bit array, so translate to bytes for comparison
    #define FORM_AND_JOIN_MAX_NETWORKS  \
      (EZSP_HOST_FORM_AND_JOIN_BUFFER_SIZE * 2 / NETWORK_STORAGE_SIZE)
  #else
    // use highest value that won't exceed max EmberMessageBuffer length
    #define FORM_AND_JOIN_MAX_NETWORKS 15
  #endif
#endif

// Check that this value isn't too large for the SoC implementation to handle
#ifndef EZSP_HOST
  #if (FORM_AND_JOIN_MAX_NETWORKS > 15)
    #error "FORM_AND_JOIN_MAX_NETWORKS can't exceed 15 on SoC platform"
  #endif
#endif

EmberStatus emberScanForUnusedPanId(uint32_t
channelMask, uint8_t duration);

EmberStatus emberScanForJoinableNetwork(
uint32_t channelMask, uint8_t* extendedPanId);

EmberStatus emberScanForNextJoinableNetwork
(void);

extern bool emberEnableDualChannelScan;

bool emberFormAndJoinIsScanning(void);

bool emberFormAndJoinCanContinueJoinableNetworkScan
(void);

//
-----------------------------------------------------------------------------
// Callbacks the application needs to implement.

void emberUnusedPanIdFoundHandler(EmberPanId
 panId, uint8_t channel);

void emberJoinableNetworkFoundHandler(
EmberZigbeeNetwork *networkFound,
                                      uint8_t lqi,
                                      int8_t rssi);

void emberScanErrorHandler(EmberStatus status);

//
-----------------------------------------------------------------------------
// Library functions the application must call from within the
// corresponding EmberZNet or EZSP callback.

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);



```

8.27 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)

8.27.1 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.

8.28 fragment-host.h

```

void ezspFragmentInit(uint16_t receiveBufferLength, uint8_t *
      receiveBuffer);

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);

bool ezspFragmentIncomingMessage(EmberApsFrame
       *apsFrame,
                                    EmberNodeId sender,
                                    uint16_t *messageLength,
                                    uint8_t **messageContents);

void ezspFragmentTick(void);

```

8.29 gcc.h File Reference

```
#include <stdint.h>
#include <stdbool.h>
#include <string.h>
#include "hal/micro/generic/compiler/platform-common.h"
```

Macros

- #define HAL_HAS_INT64
- #define _HAL_USE_COMMON_PGM_

- #define BIGENDIAN_CPU
- #define ALIGNMENT(__alignmentBytes)
- #define WEAK(__symbol)
- #define _HAL_USE_COMMON_DIVMOD_
- #define PLATCOMMONOKTOINCLUDE

Master Variable Types

These are a set of typedefs to make the size of all variable declarations explicitly known.

- typedef bool boolean
- typedef unsigned char int8u
- typedef signed char int8s
- typedef unsigned short int16u
- typedef signed short int16s
- typedef unsigned int int32u
- typedef signed int int32s
- typedef unsigned long long int64u
- typedef signed long long int64s
- typedef unsigned long PointerType

Watchdog Prototypes

Define the watchdog macro and internal function to simply be stubs to satisfy those programs that have no HAL (i.e. scripted tests) and those that want to reference real HAL functions (simulation binaries and Unix host applications) we define both halResetWatchdog() and halInternalResetWatchdog(). The former is used by most of the scripted tests while the latter is used by simulation and real host applications.

- #define halResetWatchdog()
- void halInternalResetWatchDog (void)

8.29.1 Detailed Description

See Common PLATFORM_HEADER Configuration and Unix GCC Specific PLATFORM_HEADER Configuration for documentation.

Definition in file gcc.h.

8.30 gcc.h

```
#ifndef __GCC_H__
#define __GCC_H__

#include <stdint.h>
#include <stdbool.h>

#ifndef EMBER_USE_WINDOWS_BOOLEAN // prevent duplicate definition of boolean
typedef bool boolean; /*To ease adoption of bool instead of boolean.*/
#endif

typedef unsigned char  int8u;
typedef signed   char  int8s;
typedef unsigned short int16u;
typedef signed   short int16s;
typedef unsigned int   int32u;
typedef signed   int   int32s;
typedef unsigned long long int64u;
typedef signed   long long int64s;
typedef unsigned long  PointerType;


#if defined(C8051)
typedef boolean        bit;
typedef bit            BIT;

#define SEG_DATA
#define SEG_IDATA
#define SEG_XDATA
#define SEG_PDATA
#define SEG_CODE
#define SEG_BDATA

#define idata SEG_IDATA
#define xdata SEG_XDATA
#define pdata SEG_PDATA
#define code  SEG_CODE
#define bdata SEG_BDATA
#endif // C8051

#define HAL_HAS_INT64

#define _HAL_USE_COMMON_PGM_

#ifdef DOXYGEN_SHOULD_SKIP_THIS
#define BIGENDIAN_CPU false
#else
  #if defined(__i386__)
    #define BIGENDIAN_CPU  false
  #elif defined(__ARM7__)
    #define BIGENDIAN_CPU  false
  #elif defined(__x86_64__)
    #define BIGENDIAN_CPU  false
  #elif defined(__arm__)
    #if defined(__BIG_ENDIAN)
      #define BIGENDIAN_CPU  true
    #else
      #define BIGENDIAN_CPU  false
    #endif
  #elif defined(__LITTLE_ENDIAN__)
    #define BIGENDIAN_CPU  false
  #elif defined(__BIG_ENDIAN__)
    #define BIGENDIAN_CPU  true
  #elif defined(__APPLE__)
    #define BIGENDIAN_CPU  true
  #else
    #error endianess not defined
  #endif
#endif

#ifndef DOXYGEN_SHOULD_SKIP_THIS
  #define NO_STRIPPING
  #define EEPROM

  // Always include stdio.h and assert.h if running under Unix so that they
  // can be used when debugging.
  #include <stdio.h>
  #include <assert.h>
  #include <stdarg.h>

  #define NOP()
  #define DECLARE_INTERRUPT_STATE
  #define DECLARE_INTERRUPT_STATE_LITE
  #define DISABLE_INTERRUPTS() do { } while(0)
  #define DISABLE_INTERRUPTS_LITE() do { } while(0)
  #define RESTORE_INTERRUPTS() do { } while(0)
  #define RESTORE_INTERRUPTS_LITE() do { } while(0)
  #define INTERRUPTS_ON() do { } while(0)
  #define INTERRUPTS_OFF() do { } while(0)
#if defined(EMBER_TEST)
  #define INTERRUPTS_ARE_OFF() (true)
#else
```

```
  #define INTERRUPTS_ARE_OFF() (false)
#endif
  #define ATOMIC(blah) { blah }
  #define ATOMIC_LITE(blah) { blah }
  #define HANDLE_PENDING_INTERRUPTS() do { } while(0)

  #define LOG_MESSAGE_DUMP

  #define UNUSED __attribute__ ((unused))
  #define SIGNED_ENUM

  // think different
  #ifdef __APPLE__
  #define __unix__
  #endif

  #ifdef WIN32
  // undefine this here too
    // See bug EMSTACK-2808
    #if !defined(IMAGE_BUILDER)
      #define __attribute__(foo)
    #endif
  #endif

  #if defined(EMBER_TEST)
    #define MAIN_FUNCTION_PARAMETERS void
    #define MAIN_FUNCTION_ARGUMENTS

    // Called by application main loops to let the simulator simulate.
    // Not used on real hardware.
    void simulatedTimePasses(void);          // time moves forward 4ms
    void simulatedTimePassesUs(uint32_t us);  // time moves forward us microseconds

    // This moves time forward for the minimum of:
    //   - timeToNextAppEvent milliseconds
    //   - time until the next stack event fires
    //   - time until next serial character is read or written
    // This is used to allow time to pass more efficiently - if there is nothing
    // to do it can move the clock forward by a large amount.
    void simulatedTimePassesMs(uint32_t timeToNextAppEvent);

  #else
    #define MAIN_FUNCTION_PARAMETERS int argc, char* argv[]
    #define MAIN_FUNCTION_ARGUMENTS  argc, argv
    #define MAIN_FUNCTION_HAS_STANDARD_ARGUMENTS

    // Stub for code not running in simulation.
    #define simulatedTimePasses()
    #define simulatedTimePassesUs(x)
    #define simulatedTimePassesMs(x)

  #endif

  // Called by the serial code when it wants to block.
  void simulatedSerialTimePasses(void);

#endif //DOXYGEN_SHOULD_SKIP_THIS

#define ALIGNMENT(__alignmentBytes) \
  __attribute__ ((aligned (__alignmentBytes)))

#define WEAK(__symbol) \
  __attribute__ ((weak)) __symbol

void halInternalResetWatchDog(void);
#if defined(EMBER_SCRIPTED_TEST)
  #define halResetWatchdog()
#else
  #define halResetWatchdog() \
    halInternalResetWatchDog()
#endif


#include <string.h>

#define _HAL_USE_COMMON_DIVMOD_

#define PLATCOMMONOKTOINCLUDE
```

```
  #include "hal/micro/generic/compiler/platform-common.h
"
#undef PLATCOMMONOKTOINCLUDE

#endif //__GCC_H__

```

8.31 hal.h File Reference

```
#include "micro/micro.h"
#include "micro/antenna.h"
#include "micro/adc.h"
#include "micro/button.h"
#include "micro/buzzer.h"
#include "micro/crc.h"
#include "micro/endian.h"
#include "micro/led.h"
#include "micro/random.h"
#include "micro/serial.h"
#include "micro/spi.h"
#include "micro/system-timer.h"
#include "micro/bootloader-eeprom.h"
#include "micro/bootloader-interface.h"
#include "micro/diagnostic.h"
#include "micro/token.h"
```

Macros

- #define emAmHost()

8.31.1 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.

8.31.2 Macro Definition Documentation

8.31.2.1 #define emAmHost( )

Definition at line 250 of file hal.h.

8.32 hal.h

```

#ifndef __HAL_H__
#define __HAL_H__
```

```

#ifdef HAL_HOST

#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"
//Pull in the micro specific ADC, buzzer, and clocks headers.  The
//specific header is chosen by the build include path pointing at
//the appropriate directory.
#include "adc.h"
#include "buzzer.h"

#else //HAL_MICRO

// Keep micro and board first for specifics used by other headers
#include "micro/micro.h"
#include "micro/antenna.h"
#if !defined(STACK) && defined(BOARD_HEADER)
#include BOARD_HEADER
#endif

#if (defined(EMBER_STACK_CONNECT))
  #if (defined(UNIX_HOST) && !defined(EMBER_TEST))
    #include "micro/adc.h"
    #include "micro/button.h"
    #include "micro/buzzer.h"
    #include "micro/crc.h"
    #include "micro/endian.h"
    #include "micro/led.h"
    #include "micro/random.h"
    #include "micro/serial.h"
    #include "micro/spi.h"
    #include "micro/system-timer.h"
  #else
    // TODO: here we include only the functionalities that we will have on mustang
    #if (defined(CORTEXM3))
      #include "micro/adc.h"
      #include "micro/bootloader-eeprom.h"
      #if ((defined _EFR_DEVICE) || (defined CORTEXM3_EMBER_MICRO))
      #include "micro/bootloader-interface.h"
      #endif
      #include "micro/button.h"
      #include "micro/led.h"
      #include "micro/buzzer.h"
      #include "micro/crc.h"
      #include "micro/diagnostic.h"
      #include "micro/endian.h"
    #endif //CORTEXM3
    #include "micro/flash.h"
    #include "micro/sim-eeprom.h"
    #include "micro/system-timer.h"
    #include "micro/symbol-timer.h"
    #include "micro/spi.h"
    #if (defined(CORTEXM3) || defined(EMBER_TEST))
      #include "micro/serial.h"
    #else
      #include "micro/serial-minimal.h"
    #endif
    #include "micro/random.h"
    #include "micro/token.h"
    #ifdef EMBER_TEST
      #include "micro/adc.h"
      #include "micro/bootloader-interface.h"
      #include "micro/button.h"
      #include "micro/led.h"
    #endif
  #endif // UNIX_HOST && !EMBER_TEST
#elif (defined(EMBER_STACK_OWL_RX))
  // TODO: here we include only the functionalities that we will have on OWL-RX
  #include "micro/button.h"
  #include "micro/flash.h"
  #include "micro/led.h"
  #include "micro/dog_glcd.h"
  #include "micro/system-timer.h"
  #include "micro/symbol-timer.h"
  #include "micro/spi.h"
  #include "micro/serial-minimal.h"
  #include "micro/endian.h"
//  #include "micro/random.h"
//  #include "micro/token.h"
//  #ifdef EMBER_TEST
//    #include "micro/adc.h"
//    #include "micro/bootloader-interface.h"
//    #include "micro/button.h"
//    #include "micro/led.h"
//  #endif
#elif (defined(EMBER_STACK_OWL_TX))
  // TODO: here we include only the functionalities that we will have on OWL-TX
//  #include "micro/button.h"
//  #include "micro/flash.h"
//  #include "micro/led.h"
//  #include "micro/dog_glcd.h"
//  #include "micro/system-timer.h"
//  #include "micro/symbol-timer.h"
//  #include "micro/spi.h"
//  #include "micro/serial-minimal.h"
//  #include "micro/random.h"
//  #include "micro/token.h"
//  #ifdef EMBER_TEST
//    #include "micro/adc.h"
//    #include "micro/bootloader-interface.h"
//    #include "micro/button.h"
//    #include "micro/led.h"
//  #endif
#elif (defined(EMBER_STACK_WASP))
  // TODO: here we include only the functionalities that we will have on mustang
  #if (defined(CORTEXM3))
//    #include "micro/adc.h"
//    #include "micro/bootloader-eeprom.h"
    #include "micro/button.h"
    #include "micro/buzzer.h"
    #include "micro/led.h"
    #include "micro/diagnostic.h"
  #endif //CORTEXM3
  #include "micro/flash.h"
  #include "micro/system-timer.h"
  #include "micro/symbol-timer.h"
  #include "micro/spi.h"
  #if (defined(CORTEXM3))
    #include "micro/serial.h"
  #else
    #include "micro/serial-minimal.h"
  #endif
  #include "micro/random.h"
  #include "micro/token.h"
  #ifdef EMBER_TEST
    #include "micro/adc.h"
    #include "micro/bootloader-interface.h"
    #include "micro/button.h"
    #include "micro/led.h"
  #endif
#elif (! defined(EMBER_STACK_IP))
  // Pro Stack
  #include "micro/adc.h"
  #include "micro/button.h"
  #include "micro/buzzer.h"
  #include "micro/crc.h"
  #include "micro/endian.h"
  #include "micro/led.h"
  #include "micro/random.h"
  #include "micro/serial.h"
  #include "micro/spi.h"
  #include "micro/system-timer.h"
  #include "micro/bootloader-eeprom.h"

  //Host processors do not use the following modules, therefore the header
  //files should be ignored.
  #ifndef EZSP_HOST
    #include "micro/bootloader-interface.h"
    #include "micro/diagnostic.h"
    #include "micro/token.h"
    //No public HAL code in release 4.0 uses the symbol timer,
    //therefore it should not be in doxygen.
    #ifndef DOXYGEN_SHOULD_SKIP_THIS
```

```
      #include "micro/symbol-timer.h"
    #endif // DOXYGEN_SHOULD_SKIP_THIS
  #endif //EZSP_HOST

#else
  // IP Stack
  #include "micro/adc.h"
  #include "micro/button.h"
  #include "micro/buzzer.h"
  #include "micro/crc.h"
  #include "micro/endian.h"
  #include "micro/led.h"
  #include "micro/random.h"
  #include "micro/serial.h"
  #include "micro/spi.h"
  #include "micro/system-timer.h"
  //Host processors do not use the following modules, therefore the header
  //files should be ignored.
  #ifndef UNIX_HOST
    #include "micro/bootloader-interface.h"
    #include "micro/diagnostic.h"
    #include "micro/token.h"
    //No public HAL code in release 4.0 uses the symbol timer,
    //therefore it should not be in doxygen.
    #ifndef DOXYGEN_SHOULD_SKIP_THIS
      #include "micro/symbol-timer.h"
    #endif // DOXYGEN_SHOULD_SKIP_THIS
  #endif //UNIX_HOST

#endif // !EMBER_STACK_IP

#endif // !HAL_HOST

#if ((defined(RTOS) && !defined(IP_MODEM_LIBRARY)) \
     || (defined(UNIX_HOST)                        \
     || defined(UNIX_HOST_SIM)))
  #define EMBER_HOST
  #define emAmHost() true
#else
  #define emAmHost() false
#endif

#endif //__HAL_H__

```

8.33 linux-serial.h File Reference

Macros

- #define SERIAL_PORT_RAW
- #define SERIAL_PORT_CLI

Functions

- void emberSerialSetPrompt (const char ∗thePrompt)
- void emberSerialCleanup (void)
- int emberSerialGetInputFd (uint8_t port)
- void emberSerialSendReadyToRead (uint8_t port)
- void emberSerialCommandCompletionInit (EmberCommandEntry ∗listOfCommands)

8.33.1 Detailed Description

Ember serial functionality specific to a PC with Unix library support. See Serial Communication for documentation.

Definition in file linux-serial.h.

8.34 linux-serial.h

```
// The normal CLI is accessible via port 0 while port 1 is usable for
// raw input.  This is often used by applications to receive a 260
// image for bootloading.
#define SERIAL_PORT_RAW 0
#define SERIAL_PORT_CLI 1

void emberSerialSetPrompt(const char* thePrompt);
void emberSerialCleanup(void);
int emberSerialGetInputFd(uint8_t port);
void emberSerialSendReadyToRead(uint8_t port);

// For users of app/util/serial/command-interpreter.h
void emberSerialCommandCompletionInit(
EmberCommandEntry* listOfCommands);

#if defined(GATEWAY_APP) && !defined(EMBER_AF_PLUGIN_GATEWAY)
// For users of app/util/serial/cli.h
void emberSerialCommandCompletionInitCli(cliSerialCmdEntry* cliCmdList,
                                         int cliCmdListLength);
#endif
```

8.35 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)

8.35.1 Detailed Description

Utilities for use by the ZigBee network manager. See Network Manager for documentation.

Definition in file network-manager.h.

8.36 network-manager.h

```

#include CONFIGURATION_HEADER

// The application is notified via nmUtilWarningHandler
// if NM_WARNING_LIMIT unsolicited scan reports are received
// within NM_WINDOW_SIZE minutes.  To save flash and RAM,
// the actual timing is approximate.
#ifndef NM_WARNING_LIMIT
  #define NM_WARNING_LIMIT 16
#endif

#ifndef NM_WINDOW_SIZE
  #define NM_WINDOW_SIZE 4
#endif

// The channels that should be used by the network manager.

#ifndef NM_CHANNEL_MASK
  #define NM_CHANNEL_MASK EMBER_ALL_802_15_4_CHANNELS_MASK
#endif

// The number of channels used in the NM_CHANNEL_MASK.

#ifndef NM_WATCHLIST_SIZE
  #define NM_WATCHLIST_SIZE 16
#endif

void nmUtilWarningHandler(void);

bool nmUtilProcessIncoming(EmberApsFrame *
apsFrame,
                              uint8_t messageLength,
                              uint8_t* message);

EmberStatus nmUtilChangeChannelRequest(
void);

```

8.37 platform-common.h File Reference

Macros

- #define MEMSET(d, v, l)
- #define MEMCOPY(d, s, l)
- #define MEMMOVE(d, s, l)
- #define MEMPGMCOPY(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

8.37.1 Detailed Description

See Common PLATFORM_HEADER Configuration for detailed documentation.

Definition in file platform-common.h.

8.38 platform-common.h

```

#ifndef PLATCOMMONOKTOINCLUDE
  //  This header should only be included by a PLATFORM_HEADER
  #error  platform-common.h should not be included directly
#endif

#ifndef __PLATFORMCOMMON_H__
#define __PLATFORMCOMMON_H__

// Many of the common definitions must be explicitly enabled by the
//  particular PLATFORM_HEADER being used


#ifndef DOXYGEN_SHOULD_SKIP_THIS

// The XAP2b compiler uses these macros to enable and disable placement
// in zero-page memory.  All other platforms do not have zero-page memory
// so these macros define to nothing.
#ifndef _HAL_USING_XAP2B_PRAGMAS_
  #define XAP2B_PAGEZERO_ON
  #define XAP2B_PAGEZERO_OFF
#endif

#endif //DOXYGEN_SHOULD_SKIP_THIS


#ifdef _HAL_USE_COMMON_PGM_

  #define PGM     const

  #define PGM_P   const char *

  #define PGM_PU  const unsigned char *


  #define PGM_NO_CONST

#endif //_HAL_USE_COMMON_PGM_


#ifdef _HAL_USE_COMMON_DIVMOD_

  #define halCommonUDiv32By16(x, y) ((uint16_t) (((uint32_t) (x)) / ((uint16_t) (y))))

  #define halCommonSDiv32By16(x, y) ((int16_t) (((int32_t) (x)) / ((int16_t) (y))))

  #define halCommonUMod32By16(x, y) ((uint16_t) (((uint32_t) (x)) % ((uint16_t) (y))))

  #define halCommonSMod32By16(x, y) ((int16_t) (((int32_t) (x)) % ((int16_t) (y))))

#endif //_HAL_USE_COMMON_DIVMOD_


#ifdef _HAL_USE_COMMON_MEMUTILS_


  void halCommonMemMove(void *dest, const void *src, uint16_t bytes);


  void halCommonMemSet(void *dest, uint8_t val, uint16_t bytes);


  int16_t halCommonMemCompare(const void *source0, const void *source1, uint16_t bytes);


  int8_t halCommonMemPGMCompare(const void *source0, const void PGM_NO_CONST * source1, uint16_t bytes);

  void halCommonMemPGMCopy(void* dest, const void PGM_NO_CONST *source, uint16_t bytes);

  #define MEMSET(d,v,l)  halCommonMemSet(d,v,l)
  #define MEMCOPY(d,s,l) halCommonMemMove(d,s,l)
  #define MEMMOVE(d,s,l) halCommonMemMove(d,s,l)
  #define MEMPGMCOPY(d,s,l) halCommonMemPGMCopy(d,s,l)
  #define MEMCOMPARE(s0,s1,l) halCommonMemCompare(s0, s1, l)
  #define MEMPGMCOMPARE(s0,s1,l) halCommonMemPGMCompare(s0, s1, l)

#else

  #define MEMSET(d,v,l)  memset((void*)(d),v,l)
  #define MEMCOPY(d,s,l) memcpy((void*)(d),(void*)(s),l)
  #define MEMMOVE(d,s,l) memmove((void*)(d),(void*)(s),l)
  #define MEMPGMCOPY(d,s,l) memcpy((void*)(d),(void*)(s),l)
  #define MEMCOMPARE(s0,s1,l) memcmp(s0, s1, l)
  #define MEMPGMCOMPARE(s0,s1,l) memcmp(s0, s1, l)
#endif //_HAL_USE_COMMON_MEMUTILS_









//  The following sections are common on all platforms


#define TRUE  1

#define FALSE 0

#ifndef NULL

#define NULL ((void *)0)
#endif




#define BIT(x) (1U << (x))  // Unsigned avoids compiler warnings re BIT(15)

#define BIT32(x) (((uint32_t) 1) << (x))

#define SETBIT(reg, bit)      reg |= BIT(bit)

#define SETBITS(reg, bits)    reg |= (bits)

#define CLEARBIT(reg, bit)    reg &= ~(BIT(bit))

#define CLEARBITS(reg, bits)  reg &= ~(bits)

#define READBIT(reg, bit)     (reg & (BIT(bit)))

#define READBITS(reg, bits)   (reg & (bits))





#define LOW_BYTE(n)                     ((uint8_t)((n) & 0xFF))

#define HIGH_BYTE(n)                    ((uint8_t)(LOW_BYTE((n) >> 8)))

#define HIGH_LOW_TO_INT(high, low) (                              \
                                    ((uint16_t) (( (uint16_t) (high) ) << 8)) +
   \
                                    (  (uint16_t) ( (low) & 0xFF))
\
                                   )

#define BYTE_0(n)                    ((uint8_t)((n) & 0xFF))

#define BYTE_1(n)                    ((uint8_t)(BYTE_0((n) >> 8)))

#define BYTE_2(n)                    ((uint8_t)(BYTE_0((n) >> 16)))

#define BYTE_3(n)                    ((uint8_t)(BYTE_0((n) >> 24)))

#define COUNTOF(a) (sizeof(a)/sizeof(a[0]))

```

```





#define elapsedTimeInt8u(oldTime, newTime)       \
  ((uint8_t) ((uint8_t)(newTime) - (uint8_t)(oldTime)))

#define elapsedTimeInt16u(oldTime, newTime)      \
  ((uint16_t) ((uint16_t)(newTime) - (uint16_t)(oldTime)))

#define elapsedTimeInt32u(oldTime, newTime)      \
  ((uint32_t) ((uint32_t)(newTime) - (uint32_t)(oldTime)))

#define MAX_INT8U_VALUE       (0xFF)
#define HALF_MAX_INT8U_VALUE  (0x80)
#define timeGTorEqualInt8u(t1, t2)           \
  (elapsedTimeInt8u(t2, t1) <= (HALF_MAX_INT8U_VALUE))

#define MAX_INT16U_VALUE      (0xFFFF)
#define HALF_MAX_INT16U_VALUE (0x8000)
#define timeGTorEqualInt16u(t1, t2)           \
  (elapsedTimeInt16u(t2, t1) <= (HALF_MAX_INT16U_VALUE))

#define MAX_INT32U_VALUE      (0xFFFFFFFFL)
#define HALF_MAX_INT32U_VALUE (0x80000000L)
#define timeGTorEqualInt32u(t1, t2)           \
  (elapsedTimeInt32u(t2, t1) <= (HALF_MAX_INT32U_VALUE))






#ifndef UNUSED_VAR

#define UNUSED_VAR(x) (void)(x)
#endif

#ifndef DEBUG_LEVEL
  #if defined(DEBUG) && defined(DEBUG_STRIPPED)
    #error "DEBUG and DEBUG_OFF cannot be both be defined!"
  #elif defined(DEBUG)
    #define DEBUG_LEVEL FULL_DEBUG
  #elif defined(DEBUG_STRIPPED)
    #define DEBUG_LEVEL NO_DEBUG
  #else
    #define DEBUG_LEVEL BASIC_DEBUG
  #endif
#endif


#endif //__PLATFORMCOMMON_H__

```

8.39 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 subsequentByte-Timeout)
- 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 format-String,...)
- 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, P-GM_P formatString, va_list ap)
- EmberStatus emberSerialWriteData (uint8_t port, uint8_t ∗data, uint8_t length)
- EmberStatus emberSerialWriteBuffer (uint8_t port, EmberMessageBuffer buffer, uint8-_t start, 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)

8.39.1 Detailed Description

High-level serial communication functions. See Serial Communication for documentation.

Definition in file serial.h.

8.40 serial.h

```

#ifndef __SERIAL_H__
#define __SERIAL_H__

#ifndef __HAL_H__
  #error hal/hal.h should be included first
#endif

#ifndef DOXYGEN_SHOULD_SKIP_THIS
#include <stdarg.h>

//Rx FIFO Full indicator
#define RX_FIFO_FULL (0xFFFF)

#endif // DOXYGEN_SHOULD_SKIP_THIS

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);

//Host HALs do not use stack buffers.
#ifndef HAL_HOST

EmberStatus emberSerialWriteBuffer(uint8_t
port, EmberMessageBuffer buffer, uint8_t start, uint8_t length);
#endif //HAL_HOST

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);





#endif // __SERIAL_H__

```

8.41 system-timer.h File Reference

Macros

- #define halIdleForMilliseconds(duration)

Functions

- uint16_t halInternalStartSystemTimer (void)
- uint16_t halCommonGetInt16uMillisecondTick (void)
- uint32_t halCommonGetInt32uMillisecondTick (void)
- uint16_t halCommonGetInt16uQuarterSecondTick (void)
- EmberStatus halSleepForQuarterSeconds (uint32_t ∗duration)
- EmberStatus halSleepForMilliseconds (uint32_t ∗duration)
- EmberStatus halCommonIdleForMilliseconds (uint32_t ∗duration)

8.41.1 Detailed Description

See System Timer for documentation.

Definition in file system-timer.h.

8.42 system-timer.h

```

#ifndef __SYSTEM_TIMER_H__
#define __SYSTEM_TIMER_H__

#ifndef DOXYGEN_SHOULD_SKIP_THIS

#if defined( EMBER_TEST )
  #include "unix/simulation/system-timer-sim.h"
#endif

#endif // DOXYGEN_SHOULD_SKIP_THIS


uint16_t halInternalStartSystemTimer(void);


uint16_t halCommonGetInt16uMillisecondTick(
void);

uint32_t halCommonGetInt32uMillisecondTick(
void);

uint16_t halCommonGetInt16uQuarterSecondTick
(void);

EmberStatus halSleepForQuarterSeconds(
uint32_t *duration);

EmberStatus halSleepForMilliseconds(uint32_t
*duration);

EmberStatus halCommonIdleForMilliseconds
(uint32_t *duration);
// Maintain the previous API for backwards compatibility
#define halIdleForMilliseconds(duration)
 halCommonIdleForMilliseconds((duration))

#ifdef EMBER_STACK_COBRA
EmberStatus halCobraIdleForMicroseconds(uint32_t *duration);
```

```
#endif

#endif //__SYSTEM_TIMER_H__

```

8.43 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 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)

8.43.1 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.

8.44 zigbee-device-common.h

```

#define ZDO_MESSAGE_OVERHEAD 1

#ifdef DOXYGEN_SHOULD_SKIP_THIS
EmberStatus emberNodeDescriptorRequest(EmberNodeId target,
                                       EmberApsOption options);
#else
// Macroized to save code space.
EmberStatus emberSendZigDevRequestTarget(EmberNodeId target,
                                         uint16_t clusterId,
                                         EmberApsOption options);
#define emberNodeDescriptorRequest(target, opts)      \
(emberSendZigDevRequestTarget((target), NODE_DESCRIPTOR_REQUEST, (opts)))
#endif

#ifdef DOXYGEN_SHOULD_SKIP_THIS
EmberStatus emberPowerDescriptorRequest(EmberNodeId target,
                                        EmberApsOption options);
#else
// Macroized to save code space.
#define emberPowerDescriptorRequest(target, opts)     \
(emberSendZigDevRequestTarget((target), POWER_DESCRIPTOR_REQUEST, (opts)))
#endif

EmberStatus emberSimpleDescriptorRequest(EmberNodeId target,
                                         uint8_t targetEndpoint,
                                         EmberApsOption options);

#ifdef DOXYGEN_SHOULD_SKIP_THIS
EmberStatus emberActiveEndpointsRequest(EmberNodeId target,
                                        EmberApsOption options);
#else
// Macroized to save code space.
#define emberActiveEndpointsRequest(target, opts)     \
(emberSendZigDevRequestTarget((target), ACTIVE_ENDPOINTS_REQUEST, (opts)))
#endif

#ifdef DOXYGEN_SHOULD_SKIP_THIS
EmberStatus emberBindRequest(EmberNodeId target,
                             EmberEUI64 source,
                             uint8_t sourceEndpoint,
                             uint16_t clusterId,
                             uint8_t type,
                             EmberEUI64 destination,
                             EmberMulticastId groupAddress,
                             uint8_t destinationEndpoint,
                             EmberApsOption options);
#else
// Macroized to save code space.
#define emberBindRequest(target,                                      \
                         src,                                         \
                         srcEndpt,                                    \
                         cluster,                                     \
                         type,                                        \
                         dest,                                        \
                         groupAddress,                                \
                         destEndpt,                                   \
                         opts)                                        \
                                                                      \
 (emberSendZigDevBindRequest((target),                                  \
                             BIND_REQUEST,                              \
                             (src), (srcEndpt), (cluster),              \
                             (type), (dest), (groupAddress),            \
                             (destEndpt), (opts)))

EmberStatus emberSendZigDevBindRequest(EmberNodeId target,
                                       uint16_t bindClusterId,
                                       EmberEUI64 source,
                                       uint8_t sourceEndpoint,
                                       uint16_t clusterId,
                                       uint8_t type,
                                       EmberEUI64 destination,
                                       EmberMulticastId groupAddress,
                                       uint8_t destinationEndpoint,
                                       EmberApsOption options);
#endif

#ifdef DOXYGEN_SHOULD_SKIP_THIS
EmberStatus emberUnbindRequest(EmberNodeId target,
                               EmberEUI64 source,
                               uint8_t sourceEndpoint,
                               uint16_t clusterId,
                               uint8_t type,
                               EmberEUI64 destination,
                               EmberMulticastId groupAddress,
                               uint8_t destinationEndpoint,
                               EmberApsOption options);
#else
// Macroized to save code space.
#define emberUnbindRequest(target,                                      \
                           src,                                         \
                           srcEndpt,                                    \
                           cluster,                                     \
                           type,                                        \
                           dest,                                        \
                           groupAddress,                                \
                           destEndpt,                                   \
                           opts)                                        \
                                                                        \
 (emberSendZigDevBindRequest((target),                                  \
                             UNBIND_REQUEST,                            \
                             (src), (srcEndpt), (cluster),              \
                             (type), (dest), (groupAddress),            \
                             (destEndpt), (opts)))
#endif

#ifdef DOXYGEN_SHOULD_SKIP_THIS
EmberStatus emberLqiTableRequest(EmberNodeId target,
                                 uint8_t startIndex,
                                 EmberApsOption options);
#else
#define emberLqiTableRequest(target, startIndex, options) \
  (emberTableRequest(LQI_TABLE_REQUEST, (target), (startIndex), (options)))

EmberStatus emberTableRequest(uint16_t clusterId,
                              EmberNodeId target,
                              uint8_t startIndex,
                              EmberApsOption options);
#endif

#ifdef DOXYGEN_SHOULD_SKIP_THIS
EmberStatus emberRoutingTableRequest(EmberNodeId target,
                                     uint8_t startIndex,
                                     EmberApsOption options);
#else
#define emberRoutingTableRequest(target, startIndex, options) \
  (emberTableRequest(ROUTING_TABLE_REQUEST, (target), (startIndex), (options)))
#endif

#ifdef DOXYGEN_SHOULD_SKIP_THIS
EmberStatus emberBindingTableRequest(EmberNodeId target,
                                     uint8_t startIndex,
                                     EmberApsOption options);
```

```
#else
#define emberBindingTableRequest(target, startIndex, options) \
  (emberTableRequest(BINDING_TABLE_REQUEST, (target), (startIndex), (options)))
#endif

EmberStatus emberLeaveRequest(EmberNodeId target,
                              EmberEUI64 deviceAddress,
                              uint8_t leaveRequestFlags,
                              EmberApsOption options);

EmberStatus emberPermitJoiningRequest(EmberNodeId target,
                                      uint8_t duration,
                                      uint8_t authentication,
                                      EmberApsOption options);

#ifdef DOXYGEN_SHOULD_SKIP_THIS

void emberSetZigDevRequestRadius(uint8_t radius);

uint8_t emberGetZigDevRequestRadius(void);
uint8_t emberGetLastZigDevRequestSequence(void);
#else
extern uint8_t zigDevRequestRadius;
#define emberGetZigDevRequestRadius() (zigDevRequestRadius)
#define emberSetZigDevRequestRadius(x)        (zigDevRequestRadius=x)
#define emberGetLastZigDevRequestSequence() \
  (emberGetLastAppZigDevRequestSequence())
#endif

uint8_t emberGetLastAppZigDevRequestSequence(void);

#ifndef DOXYGEN_SHOULD_SKIP_THIS
// ----------------------------------------------------------------------------
// Utility functions used by the library code.

EmberStatus emberSendZigDevRequest(EmberNodeId destination,
                                   uint16_t clusterId,
                                   EmberApsOption options,
                                   uint8_t *contents,
                                   uint8_t length);

uint8_t emberNextZigDevRequestSequence(void);

#endif // DOXYGEN_SHOULD_SKIP_THIS

```

8.45 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 eui64Return)

8.45.1 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.

8.46 zigbee-device-host.h

```

EmberStatus emberNetworkAddressRequest(
EmberEUI64 target,
                                             bool reportKids,
                                             uint8_t childStartIndex);

EmberStatus emberIeeeAddressRequest(
EmberNodeId target,
                                          bool reportKids,
                                          uint8_t childStartIndex,
                                          EmberApsOption options);
EmberStatus ezspMatchDescriptorsRequest(
EmberNodeId target,
                                              uint16_t profile,
                                              uint8_t inCount,
                                              uint8_t outCount,
                                              uint16_t *inClusters,
                                              uint16_t *outClusters,
                                              EmberApsOption options);
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);
EmberNodeId ezspDecodeAddressResponse(
uint8_t *response,
                                            EmberEUI64 eui64Return);

```