EmberZNet API Reference: For the STM32F103RET Host

March 23 2011

120-3025-000-4500

ember

Ember Corporation
47 Farnsworth Street
Boston, MA 02210
+1 (617) 951-0200
www.ember.com

wireless semiconductor solutions

ember

Ember Corporation
47 Farnsworth Street
Boston, MA 02210
+1 (617) 951-0200
www.ember.com

wireless semiconductor solutions

About this Guide

Purpose

This document is a unified collection of API reference documentation covering the EmberZNet 3.x. Ember 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 Stack Software 3.x.

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

Developer kit customers are eligible for training and technical support. You can use the Ember web site www.ember.com to obtain information about all Ember products and services, and to sign up for product support.

You can also contact Ember technical support at http://portal.ember.com.

If you have any questions about your Developer Kit, please contact Ember at esales@ember.com.

Introduction

EmberZNet 4.5.0 - Document 120-3025-000-45xx

Note:

Document 120-3024-000A, *EmberZNet API Reference: For the EM35x Network Co-Processor*, has been obsoleted and superseded by this document with respect to the STM32F103RET Host functionality.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

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

- **Ember Common**
- **Hardware Abstraction Layer (HAL) API Reference**
- **Application Utilities API Reference**

Modules

Here is a list of all modules:

Data Structures

Here are the data structures with brief descriptions:

EmberAesMmoHashContext	This data structure contains the context data when calculating an AES MMO hash (message digest)
EmberApsFrame	An in-memory representation of a ZigBee APS frame of an incoming or outgoing message
EmberBindingTableEntry	Defines an entry in the binding table
EmberCertificateData	This data structure contains the certificate data that is used for Certificate Based Key Exchange (CBKE)
EmberCommandEntry	Command entry for a command table
EmberCurrentSecurityState	This describes the security features used by the stack for a joined device
EmberEventControl	Control structure for events
EmberInitialSecurityState	This describes the Initial Security features and requirements that will be used when forming or joining the network
EmberKeyData	This data structure contains the key data that is passed into various other functions
EmberKeyStruct	This describes a one of several different types of keys and its associated data
EmberMacFilterMatchStruct	This structure indicates a matching raw MAC message has been received by the application configured MAC filters
EmberMessageDigest	This data structure contains an AES-MMO Hash (the message digest)
EmberMulticastTableEntry	Defines an entry in the multicast table
EmberNeighborTableEntry	Defines an entry in the neighbor table
EmberNetworkParameters	Holds network parameters
EmberPrivateKeyData	This data structure contains the private key data that is used for Certificate Based Key Exchange (CBKE)
EmberPublicKeyData	This data structure contains the public key data that is used for Certificate Based Key Exchange (CBKE)
EmberRouteTableEntry	Defines an entry in the route table
EmberSignatureData	This data structure contains a DSA signature. It is the bit concatenation of the 'r' and 's' components of the signature
EmberSmacData	This data structure contains the Shared Message Authentication Code (SMAC) data that is used for Certificate Based Key Exchange (CBKE)
EmberTaskControl	Control structure for tasks
EmberZigbeeNetwork	Defines a ZigBee network and the associated parameters
InterPanHeader	A struct for keeping track of all of the header info

File List

Here is a list of all files with brief descriptions:

_STM32F103RET_Host_API.top [code]	Starting page for the Ember API documentation for the STM32F103RET Host, exclusively for building documentation
adc.h [code]	
ami-inter-pan-host.h [code]	Utilities for sending and receiving ZigBee AMI InterPAN messages. See **Sending and Receiving Messages** for documentation
ami-inter-pan.h [code]	Utilities for sending and receiving ZigBee AMI InterPAN messages. See **Sending and Receiving Messages** for documentation
bootload-ezsp-utils.h [code]	Utilities used for performing stand-alone bootloading over EZSP. See **Bootloading** for documentation
bootload-utils.h [code]	Utilities used for performing stand-alone bootloading. See **Bootloading** for documentation
bootloader-eeprom.h [code]	
button-common.h [code]	
button-specific.h [code]	
buzzer.h [code]	
cbke-crypto-engine.h [code]	EmberZNet Smart Energy security API. See **Smart Energy Security** for documention
command-interpreter2.h [code]	Processes commands coming from the serial port. See **Command Interpreter 2** for documentation
crc.h [code]	
ember-configuration-defaults.h [code]	User-configurable stack memory allocation defaults
ember-types.h [code]	Ember data type definitions
error-def.h [code]	Return-code definitions for EmberZNet stack API functions
error.h [code]	Return codes for Ember API functions and module definitions
ezsp-host-configuration-defaults.h [code]	User-configurable parameters for host applications
form-and-join.h [code]	Utilities for forming and joining networks
form-and-join3_2.h [code]	Utilities for forming and joining networks. Deprecated and will be removed from a future release. Use **form-and-join.h** instead
fragment-host.h [code]	Fragmented message support for EZSP Hosts. Splits long messages into smaller blocks for transmission and reassembles received blocks. See **Message Fragmentation** for documentation
hal.h [code]	Generic set of HAL includes for all platforms
iar-st.h [code]	
led-common.h [code]	
led-specific.h [code]	
micro-common.h [code]	
micro-specific.h [code]	
network-manager.h [code]	Utilities for use by the ZigBee network manager. See **Network Manager** for documentation
platform-common.h [code]	
hal/host/serial.h [code]	
app/util/serial/serial.h [code]	High-level serial communication functions
spi-protocol-common.h [code]	
spi-protocol-specific.h [code]	

stm32f10x_conf.h [code]	
system-timer.h [code]	
uart.h [code]	
zigbee-device-common.h [code]	ZigBee Device Object (ZDO) functions available on all platforms. See **ZigBee Device Object (ZDO) Information** for documentation
zigbee-device-host.h [code]	ZigBee Device Object (ZDO) functions not provided by the stack. See **ZigBee Device Object (ZDO) Information** for documentation

Directories

The directory hierarchy:

- app
  - util
    - bootload
    - common
    - ezsp
    - serial
    - zigbee-framework
- hal
  - host
    - cortexm3
      - stm32f103ret
        - compiler
    - generic
      - compiler
- stack
  - config
  - include

Ember Common

Modules

Hardware Abstraction Layer (HAL) API Reference

Modules

Detailed Description

STM32F103RET Host Microcontroller

HAL function names have the following prefix conventions:

halCommon: API that is used by the EmberZNet stack and can also be called from an application. This API must be implemented. Custom applications can change the implementation of the API but its functionality must remain the same.

hal: API that is used by sample applications. Custom applications can remove this API or change its implementation as they see fit.

halStack: API used only by the EmberZNet stack. This API must be implemented and should not be directly called from any application. Custom applications can change the implementation of the API, but its functionality must remain the same.

halInternal: API that is internal to the HAL. The EmberZNet stack and applications must never call this API directly. Custom applications can change this API as they see fit. However, be careful not to impact the functionalty of any halStack or halCommon APIs.

See also **hal.h**.

Application Utilities API Reference

Modules

Ember Common Data Types
[Ember Common]

Data Structures

struct	**EmberZigbeeNetwork** Defines a ZigBee network and the associated parameters. More...
struct	**EmberNetworkParameters** Holds network parameters. More...
struct	**EmberApsFrame** An in-memory representation of a ZigBee APS frame of an incoming or outgoing message. More...
struct	**EmberBindingTableEntry** Defines an entry in the binding table. More...
struct	**EmberNeighborTableEntry** Defines an entry in the neighbor table. More...
struct	**EmberRouteTableEntry** Defines an entry in the route table. More...
struct	**EmberMulticastTableEntry** Defines an entry in the multicast table. More...
struct	**EmberEventControl** Control structure for events. More...
struct	**EmberTaskControl** Control structure for tasks. More...
struct	**EmberKeyData** This data structure contains the key data that is passed into various other functions. More...
struct	**EmberCertificateData** This data structure contains the certificate data that is used for Certificate Based Key Exchange (CBKE). More...
struct	**EmberPublicKeyData** This data structure contains the public key data that is used for Certificate Based Key Exchange (CBKE). More...
struct	**EmberPrivateKeyData** This data structure contains the private key data that is used for Certificate Based Key Exchange (CBKE). More...
struct	**EmberSmacData** This data structure contains the Shared Message Authentication Code (SMAC) data that is used for Certificate Based Key Exchange (CBKE). More...
struct	**EmberSignatureData** This data structure contains a DSA signature. It is the bit concatenation of the 'r' and 's' components of the signature. More...
struct	**EmberMessageDigest** This data structure contains an AES-MMO Hash (the message digest). More...
struct	**EmberAesMmoHashContext** This data structure contains the context data when calculating an AES MMO hash (message digest). More...
struct	**EmberInitialSecurityState** This describes the Initial Security features and requirements that will be used when forming or joining the network. More...
struct	**EmberCurrentSecurityState** This describes the security features used by the stack for a joined device. More...
struct	**EmberKeyStruct** This describes a one of several different types of keys and its associated data. More...
struct	**EmberMacFilterMatchStruct** This structure indicates a matching raw MAC message has been received by the application configured MAC filters. More...

Defines

#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_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_END**

Typedefs

typedef **int8u**	**EmberTaskId**
struct {	
EmberEventControl * **control**	
void(* **handler**)(void)	
}	**EmberEventData**
typedef **int16u**	**EmberMacFilterMatchData**
typedef **int8u**	**EmberLibraryStatus**

Enumerations

enum	**EmberNodeType** {
	EMBER_UNKNOWN_DEVICE,
	EMBER_COORDINATOR,
	EMBER_ROUTER,
	EMBER_END_DEVICE,
	EMBER_SLEEPY_END_DEVICE,
	EMBER_MOBILE_END_DEVICE
	}

enum	**EmberApsOption** {
	EMBER_APS_OPTION_NONE,
	EMBER_APS_OPTION_DSA_SIGN,
	EMBER_APS_OPTION_ENCRYPTION,
	EMBER_APS_OPTION_RETRY,
	EMBER_APS_OPTION_ENABLE_ROUTE_DISCOVERY,
	EMBER_APS_OPTION_FORCE_ROUTE_DISCOVERY,
	EMBER_APS_OPTION_SOURCE_EUI64,
	EMBER_APS_OPTION_DESTINATION_EUI64,
	EMBER_APS_OPTION_ENABLE_ADDRESS_DISCOVERY,
	EMBER_APS_OPTION_POLL_RESPONSE,
	EMBER_APS_OPTION_ZDO_RESPONSE_REQUIRED,
	EMBER_APS_OPTION_FRAGMENT
	}

enum	**EmberIncomingMessageType** {
	EMBER_INCOMING_UNICAST,
	EMBER_INCOMING_UNICAST_REPLY,
	EMBER_INCOMING_MULTICAST,
	EMBER_INCOMING_MULTICAST_LOOPBACK,
	EMBER_INCOMING_BROADCAST,
	EMBER_INCOMING_BROADCAST_LOOPBACK
	}

enum	**EmberOutgoingMessageType** { **EMBER_OUTGOING_DIRECT**, **EMBER_OUTGOING_VIA_ADDRESS_TABLE**, **EMBER_OUTGOING_VIA_BINDING**,


```
    EMBER_OUTGOING_MULTICAST,
    EMBER_OUTGOING_BROADCAST
  }
enum EmberNetworkStatus {
    EMBER_NO_NETWORK,
    EMBER_JOINING_NETWORK,
    EMBER_JOINED_NETWORK,
    EMBER_JOINED_NETWORK_NO_PARENT,
    EMBER_LEAVING_NETWORK
  }
enum EmberNetworkScanType {
    EMBER_ENERGY_SCAN,
    EMBER_ACTIVE_SCAN
  }
enum EmberBindingType {
    EMBER_UNUSED_BINDING,
    EMBER_UNICAST_BINDING,
    EMBER_MANY_TO_ONE_BINDING,
    EMBER_MULTICAST_BINDING
  }
enum EmberJoinDecision {
    EMBER_USE_PRECONFIGURED_KEY,
    EMBER_SEND_KEY_IN_THE_CLEAR,
    EMBER_DENY_JOIN,
    EMBER_NO_ACTION
  }
enum EmberDeviceUpdate {
    EMBER_STANDARD_SECURITY_SECURED_REJOIN,
    EMBER_STANDARD_SECURITY_UNSECURED_JOIN,
    EMBER_DEVICE_LEFT,
    EMBER_STANDARD_SECURITY_UNSECURED_REJOIN,
    EMBER_HIGH_SECURITY_SECURED_REJOIN,
    EMBER_HIGH_SECURITY_UNSECURED_JOIN,
    EMBER_HIGH_SECURITY_UNSECURED_REJOIN
  }
enum EmberClusterListId {
    EMBER_INPUT_CLUSTER_LIST,
    EMBER_OUTPUT_CLUSTER_LIST
  }
enum EmberEventUnits {
    EMBER_EVENT_INACTIVE,
    EMBER_EVENT_MS_TIME,
    EMBER_EVENT_QS_TIME,
    EMBER_EVENT_MINUTE_TIME,
    EMBER_EVENT_ZERO_DELAY
  }
enum EmberJoinMethod {
    EMBER_USE_MAC_ASSOCIATION,
    EMBER_USE_NWK_REJOIN,
    EMBER_USE_NWK_REJOIN_HAVE_NWK_KEY,
    EMBER_USE_NWK_COMMISSIONING
  }
enum EmberCounterType {
    EMBER_COUNTER_MAC_RX_BROADCAST,
    EMBER_COUNTER_MAC_TX_BROADCAST,
    EMBER_COUNTER_MAC_RX_UNICAST,
    EMBER_COUNTER_MAC_TX_UNICAST_SUCCESS,
    EMBER_COUNTER_MAC_TX_UNICAST_RETRY,
    EMBER_COUNTER_MAC_TX_UNICAST_FAILED,
    EMBER_COUNTER_APS_DATA_RX_BROADCAST,
    EMBER_COUNTER_APS_DATA_TX_BROADCAST,
    EMBER_COUNTER_APS_DATA_RX_UNICAST,
    EMBER_COUNTER_APS_DATA_TX_UNICAST_SUCCESS,
    EMBER_COUNTER_APS_DATA_TX_UNICAST_RETRY,
    EMBER_COUNTER_APS_DATA_TX_UNICAST_FAILED,
    EMBER_COUNTER_ROUTE_DISCOVERY_INITIATED,
    EMBER_COUNTER_NEIGHBOR_ADDED,
    EMBER_COUNTER_NEIGHBOR_REMOVED,
    EMBER_COUNTER_NEIGHBOR_STALE,
    EMBER_COUNTER_JOIN_INDICATION,
    EMBER_COUNTER_CHILD_REMOVED,
    EMBER_COUNTER_ASH_OVERFLOW_ERROR,
```



```
  EMBER_COUNTER_ASH_FRAMING_ERROR,
  EMBER_COUNTER_ASH_OVERRUN_ERROR,
  EMBER_COUNTER_NWK_FRAME_COUNTER_FAILURE,
  EMBER_COUNTER_APS_FRAME_COUNTER_FAILURE,
  EMBER_COUNTER_ASH_XOFF,
  EMBER_COUNTER_APS_LINK_KEY_NOT_AUTHORIZED,
  EMBER_COUNTER_NWK_DECRYPTION_FAILURE,
  EMBER_COUNTER_APS_DECRYPTION_FAILURE,
  EMBER_COUNTER_ALLOCATE_PACKET_BUFFER_FAILURE,
  EMBER_COUNTER_RELAYED_UNICAST,
  EMBER_COUNTER_PHY_TO_MAC_QUEUE_LIMIT_REACHED,
  EMBER_COUNTER_TYPE_COUNT
}
enum EmberInitialSecurityBitmask {
  EMBER_DISTRIBUTED_TRUST_CENTER_MODE,
  EMBER_GLOBAL_LINK_KEY,
  EMBER_PRECONFIGURED_NETWORK_KEY_MODE,
  EMBER_HAVE_TRUST_CENTER_EUI64,
  EMBER_TRUST_CENTER_USES_HASHED_LINK_KEY,
  EMBER_HAVE_PRECONFIGURED_KEY,
  EMBER_HAVE_NETWORK_KEY,
  EMBER_GET_LINK_KEY_WHEN_JOINING,
  EMBER_REQUIRE_ENCRYPTED_KEY,
  EMBER_NO_FRAME_COUNTER_RESET,
  EMBER_GET_PRECONFIGURED_KEY_FROM_INSTALL_CODE
}
enum EmberCurrentSecurityBitmask {
  EMBER_STANDARD_SECURITY_MODE_,
  EMBER_DISTRIBUTED_TRUST_CENTER_MODE_,
  EMBER_GLOBAL_LINK_KEY_,
  EMBER_HAVE_TRUST_CENTER_LINK_KEY,
  EMBER_TRUST_CENTER_USES_HASHED_LINK_KEY_
}
enum EmberKeyStructBitmask {
  EMBER_KEY_HAS_SEQUENCE_NUMBER,
  EMBER_KEY_HAS_OUTGOING_FRAME_COUNTER,
  EMBER_KEY_HAS_INCOMING_FRAME_COUNTER,
  EMBER_KEY_HAS_PARTNER_EUI64,
  EMBER_KEY_IS_AUTHORIZED,
  EMBER_KEY_PARTNER_IS_SLEEPY
}
enum EmberKeyType {
  EMBER_TRUST_CENTER_LINK_KEY,
  EMBER_TRUST_CENTER_MASTER_KEY,
  EMBER_CURRENT_NETWORK_KEY,
  EMBER_NEXT_NETWORK_KEY,
  EMBER_APPLICATION_LINK_KEY,
  EMBER_APPLICATION_MASTER_KEY
}
enum EmberKeyStatus {
  EMBER_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
}
enum EmberLinkKeyRequestPolicy {
  EMBER_DENY_KEY_REQUESTS,
  EMBER_ALLOW_KEY_REQUESTS
}
```



```
enum  EmberMacPassthroughType {
        EMBER_MAC_PASSTHROUGH_NONE,
        EMBER_MAC_PASSTHROUGH_SE_INTERPAN,
        EMBER_MAC_PASSTHROUGH_EMBERNET,
        EMBER_MAC_PASSTHROUGH_EMBERNET_SOURCE,
        EMBER_MAC_PASSTHROUGH_APPLICATION,
        EMBER_MAC_PASSTHROUGH_CUSTOM
      }
```

Functions

```
int8u *  emberKeyContents (EmberKeyData *key)
int8u *  emberCertificateContents (EmberCertificateData *cert)
int8u *  emberPublicKeyContents (EmberPublicKeyData *key)
int8u *  emberPrivateKeyContents (EmberPrivateKeyData *key)
int8u *  emberSmacContents (EmberSmacData *key)
int8u *  emberSignatureContents (EmberSignatureData *sig)
```

Miscellaneous Ember Types

```
enum  EmberLeaveRequestFlags {
        EMBER_ZIGBEE_LEAVE_AND_REJOIN,
        EMBER_ZIGBEE_LEAVE_AND_REMOVE_CHILDREN
      }
typedef int8u  EmberStatus
typedef int8u  EmberEUI64 [EUI64_SIZE]
typedef int8u  EmberMessageBuffer
typedef int16u  EmberNodeId
typedef int16u  EmberMulticastId
typedef int16u  EmberPanId
#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_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
```

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
}
```

ZDO server mask bits

These are used in server discovery requests and responses.

```
enum EmberZdoServerMask {
  EMBER_ZDP_PRIMARY_TRUST_CENTER,
  EMBER_ZDP_SECONDARY_TRUST_CENTER,
  EMBER_ZDP_PRIMARY_BINDING_TABLE_CACHE,
  EMBER_ZDP_SECONDARY_BINDING_TABLE_CACHE,
  EMBER_ZDP_PRIMARY_DISCOVERY_CACHE,
  EMBER_ZDP_SECONDARY_DISCOVERY_CACHE,
  EMBER_ZDP_NETWORK_MANAGER
}
```

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
}
```

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_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
```

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

```
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

```
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
```

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**

Detailed Description

See **ember-types.h** for source code.

Define Documentation

#define EUI64_SIZE

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

Definition at line **37** of file **ember-types.h**.

#define EXTENDED_PAN_ID_SIZE

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

Definition at line **42** of file **ember-types.h**.

#define EMBER_ENCRYPTION_KEY_SIZE

Size of an encryption key in bytes (16).

Definition at line **47** of file **ember-types.h**.

#define EMBER_CERTIFICATE_SIZE

Size of Implicit Certificates used for Certificate Based Key Exchange.

Definition at line **53** of file **ember-types.h**.

#define EMBER_PUBLIC_KEY_SIZE

Size of Public Keys used in Elliptical Cryptography ECMQV algorithms.

Definition at line **58** of file **ember-types.h**.

#define EMBER_PRIVATE_KEY_SIZE

Size of Private Keys used in Elliptical Cryptography ECMQV algorithms.

Definition at line **63** of file **ember-types.h**.

#define EMBER_SMAC_SIZE

Size of the SMAC used in Elliptical Cryptography ECMQV algorithms.

Definition at line **68** of file **ember-types.h**.

#define EMBER_SIGNATURE_SIZE

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

Definition at line **74** of file **ember-types.h**.

#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 **79** of file **ember-types.h**.

#define EMBER_MAX_802_15_4_CHANNEL_NUMBER

The maximum 802.15.4 channel number is 26.

Definition at line **122** of file **ember-types.h**.

#define EMBER_MIN_802_15_4_CHANNEL_NUMBER

The minimum 802.15.4 channel number is 11.

Definition at line **127** of file **ember-types.h**.

#define EMBER_NUM_802_15_4_CHANNELS

There are sixteen 802.15.4 channels.

Definition at line **132** of file **ember-types.h**.

#define EMBER_ALL_802_15_4_CHANNELS_MASK

Bitmask to scan all 802.15.4 channels.

Definition at line **138** of file **ember-types.h**.

#define EMBER_ZIGBEE_COORDINATOR_ADDRESS

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

Definition at line **143** of file **ember-types.h**.

#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 **149** of file **ember-types.h**.

#define EMBER_NULL_BINDING

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

Definition at line **155** of file **ember-types.h**.

#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 **166** of file **ember-types.h**.

#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 **174** of file **ember-types.h**.

#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 **183** of file **ember-types.h**.

#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 **192** of file **ember-types.h**.

#define EMBER_NULL_ADDRESS_TABLE_INDEX

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

Definition at line **198** of file **ember-types.h**.

#define EMBER_ZDO_ENDPOINT

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

Definition at line **203** of file **ember-types.h**.

#define EMBER_BROADCAST_ENDPOINT

The broadcast endpoint, as defined in the ZigBee spec.

Definition at line **208** of file **ember-types.h**.

#define EMBER_ZDO_PROFILE_ID

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

Definition at line **213** of file **ember-types.h**.

#define EMBER_BROADCAST_ADDRESS

Broadcast to all routers.

Definition at line **245** of file **ember-types.h**.

#define EMBER_RX_ON_WHEN_IDLE_BROADCAST_ADDRESS

Broadcast to all non-sleepy devices.

Definition at line **247** of file **ember-types.h**.

#define EMBER_SLEEPY_BROADCAST_ADDRESS

Broadcast to all devices, including sleepy end devices.

Definition at line **249** of file **ember-types.h**.

#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 **488** of file **ember-types.h**.

#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 **493** of file **ember-types.h**.

#define EMBER_JOIN_DECISION_STRINGS

@ brief Defines the CLI enumerations for the **EmberJoinDecision** enum

Definition at line **521** of file **ember-types.h**.

#define EMBER_DEVICE_UPDATE_STRINGS

@ brief Defines the CLI enumerations for the **EmberDeviceUpdate** enum.

Definition at line **556** of file **ember-types.h**.

#define emberInitializeNetworkParameters (parameters)

Definition at line **698** of file **ember-types.h**.

#define EMBER_COUNTER_STRINGS

@ brief Defines the CLI enumerations for the **EmberCounterType** enum.

Definition at line **948** of file **ember-types.h**.

#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 **1055** of file **ember-types.h**.

#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 **1059** of file **ember-types.h**.

#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 **1064** of file **ember-types.h**.

#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 **1069** of file **ember-types.h**.

#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 **1093** of file **ember-types.h**.

#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 **1133** of file **ember-types.h**.

#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 **1171** of file **ember-types.h**.

#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 **1188** of file **ember-types.h**.

#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 **1193** of file **ember-types.h**.

#define EMBER_REPORT_COUNTERS_RESPONSE

The cluster id used to respond to an EMBER_REPORT_COUNTERS_REQUEST.

Definition at line **1196** of file **ember-types.h**.

#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 **1202** of file **ember-types.h**.

#define EMBER_REPORT_AND_CLEAR_COUNTERS_RESPONSE

The cluster id used to respond to an EMBER_REPORT_AND_CLEAR_COUNTERS_REQUEST.

Definition at line **1205** of file **ember-types.h**.

#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 **1211** of file **ember-types.h**.

#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 **1275** of file **ember-types.h**.

#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 **1280** of file **ember-types.h**.

#define EMBER_NO_TRUST_CENTER_MODE

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

Definition at line **1378** of file **ember-types.h**.

#define EMBER_MAC_FILTER_MATCH_ENABLED_MASK

Definition at line **1728** of file **ember-types.h**.

#define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_MASK

Definition at line **1729** of file **ember-types.h**.

#define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_MASK

Definition at line **1730** of file **ember-types.h**.

#define EMBER_MAC_FILTER_MATCH_ON_DEST_MASK

Definition at line **1731** of file **ember-types.h**.

#define EMBER_MAC_FILTER_MATCH_ON_SOURCE_MASK

Definition at line **1732** of file **ember-types.h**.

#define EMBER_MAC_FILTER_MATCH_ENABLED

Definition at line **1735** of file **ember-types.h**.

#define EMBER_MAC_FILTER_MATCH_DISABLED

Definition at line **1736** of file **ember-types.h**.

#define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_NONE

Definition at line **1739** of file **ember-types.h**.

#define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_LOCAL

Definition at line **1740** of file **ember-types.h**.

#define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_BROADCAST

Definition at line **1741** of file **ember-types.h**.

#define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_NONE

Definition at line **1744** of file **ember-types.h**.

#define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_NON_LOCAL

Definition at line 1745 of file ember-types.h.

#define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_LOCAL

Definition at line 1746 of file ember-types.h.

#define EMBER_MAC_FILTER_MATCH_ON_DEST_BROADCAST_SHORT

Definition at line 1749 of file ember-types.h.

#define EMBER_MAC_FILTER_MATCH_ON_DEST_UNICAST_SHORT

Definition at line 1750 of file ember-types.h.

#define EMBER_MAC_FILTER_MATCH_ON_DEST_UNICAST_LONG

Definition at line 1751 of file ember-types.h.

#define EMBER_MAC_FILTER_MATCH_ON_SOURCE_LONG

Definition at line 1754 of file ember-types.h.

#define EMBER_MAC_FILTER_MATCH_ON_SOURCE_SHORT

Definition at line 1755 of file ember-types.h.

#define EMBER_MAC_FILTER_MATCH_END

Definition at line 1758 of file ember-types.h.

#define NETWORK_ADDRESS_REQUEST

Definition at line 1842 of file ember-types.h.

#define NETWORK_ADDRESS_RESPONSE

Definition at line 1843 of file ember-types.h.

#define IEEE_ADDRESS_REQUEST

Definition at line 1844 of file ember-types.h.

#define IEEE_ADDRESS_RESPONSE

Definition at line 1845 of file ember-types.h.

#define NODE_DESCRIPTOR_REQUEST

Definition at line 1873 of file ember-types.h.

#define NODE_DESCRIPTOR_RESPONSE

Definition at line 1874 of file ember-types.h.

#define POWER_DESCRIPTOR_REQUEST

Definition at line 1887 of file ember-types.h.

#define POWER_DESCRIPTOR_RESPONSE

Definition at line 1888 of file ember-types.h.

#define SIMPLE_DESCRIPTOR_REQUEST

Definition at line 1904 of file ember-types.h.

#define SIMPLE_DESCRIPTOR_RESPONSE

Definition at line 1905 of file ember-types.h.

#define ACTIVE_ENDPOINTS_REQUEST

Definition at line 1916 of file ember-types.h.

#define ACTIVE_ENDPOINTS_RESPONSE

Definition at line 1917 of file ember-types.h.

#define MATCH_DESCRIPTORS_REQUEST

Definition at line 1931 of file ember-types.h.

#define MATCH_DESCRIPTORS_RESPONSE

Definition at line 1932 of file ember-types.h.

#define DISCOVERY_CACHE_REQUEST

Definition at line 1944 of file ember-types.h.

#define DISCOVERY_CACHE_RESPONSE

Definition at line 1945 of file ember-types.h.

#define END_DEVICE_ANNOUNCE

Definition at line 1956 of file ember-types.h.

#define END_DEVICE_ANNOUNCE_RESPONSE

Definition at line 1957 of file ember-types.h.

#define SYSTEM_SERVER_DISCOVERY_REQUEST

Definition at line 1971 of file ember-types.h.

#define SYSTEM_SERVER_DISCOVERY_RESPONSE

Definition at line **1972** of file **ember-types.h**.

#define FIND_NODE_CACHE_REQUEST

Definition at line **2009** of file **ember-types.h**.

#define FIND_NODE_CACHE_RESPONSE

Definition at line **2010** of file **ember-types.h**.

#define END_DEVICE_BIND_REQUEST

Definition at line **2023** of file **ember-types.h**.

#define END_DEVICE_BIND_RESPONSE

Definition at line **2024** of file **ember-types.h**.

#define UNICAST_BINDING

Definition at line **2044** of file **ember-types.h**.

#define UNICAST_MANY_TO_ONE_BINDING

Definition at line **2045** of file **ember-types.h**.

#define MULTICAST_BINDING

Definition at line **2046** of file **ember-types.h**.

#define BIND_REQUEST

Definition at line **2048** of file **ember-types.h**.

#define BIND_RESPONSE

Definition at line **2049** of file **ember-types.h**.

#define UNBIND_REQUEST

Definition at line **2050** of file **ember-types.h**.

#define UNBIND_RESPONSE

Definition at line **2051** of file **ember-types.h**.

#define LQI_TABLE_REQUEST

Definition at line **2101** of file **ember-types.h**.

#define LQI_TABLE_RESPONSE

Definition at line **2102** of file **ember-types.h**.

#define ROUTING_TABLE_REQUEST

Definition at line 2137 of file ember-types.h.

#define ROUTING_TABLE_RESPONSE

Definition at line 2138 of file ember-types.h.

#define BINDING_TABLE_REQUEST

Definition at line 2159 of file ember-types.h.

#define BINDING_TABLE_RESPONSE

Definition at line 2160 of file ember-types.h.

#define LEAVE_REQUEST

Definition at line 2173 of file ember-types.h.

#define LEAVE_RESPONSE

Definition at line 2174 of file ember-types.h.

#define LEAVE_REQUEST_REMOVE_CHILDREN_FLAG

Definition at line 2176 of file ember-types.h.

#define LEAVE_REQUEST_REJOIN_FLAG

Definition at line 2177 of file ember-types.h.

#define PERMIT_JOINING_REQUEST

Definition at line 2188 of file ember-types.h.

#define PERMIT_JOINING_RESPONSE

Definition at line 2189 of file ember-types.h.

#define NWK_UPDATE_REQUEST

Definition at line 2217 of file ember-types.h.

#define NWK_UPDATE_RESPONSE

Definition at line 2218 of file ember-types.h.

#define COMPLEX_DESCRIPTOR_REQUEST

Definition at line 2224 of file ember-types.h.

#define COMPLEX_DESCRIPTOR_RESPONSE

Definition at line 2225 of file ember-types.h.

#define USER_DESCRIPTOR_REQUEST

Definition at line **2226** of file **ember-types.h**.

#define USER_DESCRIPTOR_RESPONSE

Definition at line **2227** of file **ember-types.h**.

#define DISCOVERY_REGISTER_REQUEST

Definition at line **2228** of file **ember-types.h**.

#define DISCOVERY_REGISTER_RESPONSE

Definition at line **2229** of file **ember-types.h**.

#define USER_DESCRIPTOR_SET

Definition at line **2230** of file **ember-types.h**.

#define USER_DESCRIPTOR_CONFIRM

Definition at line **2231** of file **ember-types.h**.

#define NETWORK_DISCOVERY_REQUEST

Definition at line **2232** of file **ember-types.h**.

#define NETWORK_DISCOVERY_RESPONSE

Definition at line **2233** of file **ember-types.h**.

#define DIRECT_JOIN_REQUEST

Definition at line **2234** of file **ember-types.h**.

#define DIRECT_JOIN_RESPONSE

Definition at line **2235** of file **ember-types.h**.

#define CLUSTER_ID_RESPONSE_MINIMUM

Definition at line **2238** of file **ember-types.h**.

Typedef Documentation

typedef int8u EmberStatus

Return type for Ember functions.

Definition at line **87** of file **ember-types.h**.

typedef int8u EmberEUI64[EUI64_SIZE]

EUI 64-bit ID (an IEEE address).

Definition at line **93** of file **ember-types.h**.

typedef int8u 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 **104** of file **ember-types.h**.

typedef int16u EmberNodeId

16-bit ZigBee network address.

Definition at line **109** of file **ember-types.h**.

typedef int16u EmberMulticastId

16-bit ZigBee multicast group identifier.

Definition at line **112** of file **ember-types.h**.

typedef int16u EmberPanId

802.15.4 PAN ID.

Definition at line **117** of file **ember-types.h**.

typedef int8u EmberTaskId

brief An identifier for a task

Definition at line **982** of file **ember-types.h**.

typedef { ... } EmberEventData

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.

typedef int16u 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 **1726** of file **ember-types.h**.

typedef int8u EmberLibraryStatus

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

Definition at line **1773** of file **ember-types.h**.

Enumeration Type Documentation

enum EmberLeaveRequestFlags

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

Enumerator:

EMBER_ZIGBEE_LEAVE_AND_REJOIN	Leave and rejoin
EMBER_ZIGBEE_LEAVE_AND_REMOVE_CHILDREN	Send all children leave command

Definition at line **217** of file **ember-types.h**.

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.

Definition at line **259** of file **ember-types.h**.

enum EmberApsOption

Options to use when sending a message.

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

Enumerator:

EMBER_APS_OPTION_NONE	No options.
EMBER_APS_OPTION_DSA_SIGN	This signs the application layer message body (APS Frame not included) and appends the ECDSA signature to the end of the message. Needed by Smart Energy applications. This requires the CBKE and ECC libraries. The **emberDsaSignHandler()** function is called after DSA signing is complete but before the message has been sent by the APS layer. Note that when passing a buffer to the stack for DSA signing, the final byte in the buffer has special significance as an indicator of how many leading bytes should be ignored for signature purposes. Refer to API documentation of **emberDsaSign()** or the dsaSign EZSP command for further details about this requirement.
EMBER_APS_OPTION_ENCRYPTION	Send the message using APS Encryption, using the Link Key shared with the destination node to encrypt the data at the APS Level.
EMBER_APS_OPTION_RETRY	Resend the message using the APS retry mechanism. In the mesh stack, this option and the enable route discovery option must be enabled for an existing route to be repaired automatically.
EMBER_APS_OPTION_ENABLE_ROUTE_DISCOVERY	Send the message with the NWK 'enable route discovery' flag, which causes a route discovery to be initiated if no route to the destination is known. Note that in the mesh stack, this option and the APS retry option must be enabled an existing route to be repaired automatically.
EMBER_APS_OPTION_FORCE_ROUTE_DISCOVERY	Send the message with the NWK 'force route discovery' flag, which causes a route discovery to be initiated even if one is known.
EMBER_APS_OPTION_SOURCE_EUI64	Include the source EUI64 in the network frame.
EMBER_APS_OPTION_DESTINATION_EUI64	Include the destination EUI64 in the network frame.
EMBER_APS_OPTION_ENABLE_ADDRESS_DISCOVERY	Send a ZDO request to discover the node ID of the destination, if it is not already know.
EMBER_APS_OPTION_POLL_RESPONSE	This message is being sent in response to a call to emberPollHandler(). It causes the message to be sent immediately instead of being queued up until the next poll from the (end device) destination.
EMBER_APS_OPTION_ZDO_RESPONSE_REQUIRED	This incoming message is a valid ZDO request and the application is responsible for sending a ZDO response. This flag is used only within emberIncomingMessageHandler() when EMBER_APPLICATION_RECEIVES_UNSUPPORTED_ZDO_REQUESTS

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

Definition at line **301** of file **ember-types.h**.

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 **368** of file **ember-types.h**.

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

Definition at line **393** of file **ember-types.h**.

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 **418** of file **ember-types.h**.

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 **442** of file **ember-types.h**.

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 **459** of file **ember-types.h**.

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 **502** of file **ember-types.h**.

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 **536** of file **ember-types.h**.

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 **570** of file **ember-types.h**.

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 **588** of file **ember-types.h**.

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 **613** of file **ember-types.h**.

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_TYPE_COUNT	A placeholder giving the number of Ember counter types.

Definition at line **832** of file **ember-types.h**.

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_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 only necessary if the device does not have a pre-configured key.
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.
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 **1287** of file **ember-types.h**.

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_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 **1438** of file **ember-types.h**.

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 **1490** of file **ember-types.h**.

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 **1525** of file **ember-types.h**.

enum EmberKeyStatus

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

Enumerator:

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

Definition at line **1575** of file **ember-types.h**.

enum EmberLinkKeyRequestPolicy

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

Enumerator:

EMBER_DENY_KEY_REQUESTS
EMBER_ALLOW_KEY_REQUESTS

Definition at line **1610** of file **ember-types.h**.

enum EmberMacPassthroughType

The types of MAC passthrough messages that an application may receive. This is a bitmask.

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 **1697** of file **ember-types.h**.

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 **1786** of file **ember-types.h**.

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 **1980** of file **ember-types.h**.

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 **2254** of file **ember-types.h**.

Function Documentation

int8u* 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:
int8u* Returns a pointer to the first byte of the Key data.

int8u* 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:
int8u* Returns a pointer to the first byte of the certificate data.

int8u* 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:

int8u* Returns a pointer to the first byte of the public key data.

int8u* 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:

int8u* Returns a pointer to the first byte of the private key data.

int8u* emberSmacContents (EmberSmacData * key)

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

int8u* emberSignatureContents (EmberSignatureData * sig)

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

Sending and Receiving Messages
[Ember Common]

Data Structures

struct	**InterPanHeader** A struct for keeping track of all of the header info. More...

Defines

#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)
int8u	**parseInterPanMessage** (**EmberMessageBuffer** message, **int8u** startOffset, **InterPanHeader** *headerData)
int8u	**makeInterPanMessage** (**InterPanHeader** *headerData, **int8u** *message, **int8u** maxLength, **int8u** *payload, **int8u** payloadLength)
int8u	**parseInterPanMessage** (**int8u** *message, **int8u** messageLength, **InterPanHeader** *headerData)

Detailed Description

See also **ami-inter-pan.h** for source code.

See also **ami-inter-pan-host.h** for source code.

Define Documentation

#define INTER_PAN_UNICAST

Definition at line **25** of file **ami-inter-pan.h**.

#define INTER_PAN_BROADCAST

Definition at line **26** of file **ami-inter-pan.h**.

#define INTER_PAN_MULTICAST

Definition at line **27** of file **ami-inter-pan.h**.

#define MAX_INTER_PAN_MAC_SIZE

Definition at line **30** of file **ami-inter-pan.h**.

#define STUB_NWK_SIZE

Definition at line **34** of file **ami-inter-pan.h**.

#define STUB_NWK_FRAME_CONTROL

Definition at line **35** of file **ami-inter-pan.h**.

#define MAX_STUB_APS_SIZE

Definition at line **38** of file **ami-inter-pan.h**.

#define MAX_INTER_PAN_HEADER_SIZE

Definition at line **41** of file **ami-inter-pan.h**.

#define INTER_PAN_UNICAST

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

Definition at line **24** of file **ami-inter-pan-host.h**.

#define INTER_PAN_BROADCAST

Definition at line **25** of file **ami-inter-pan-host.h**.

#define INTER_PAN_MULTICAST

Definition at line **26** of file **ami-inter-pan-host.h**.

#define MAX_INTER_PAN_MAC_SIZE

Definition at line **30** of file **ami-inter-pan-host.h**.

#define STUB_NWK_SIZE

Definition at line **34** of file **ami-inter-pan-host.h**.

#define STUB_NWK_FRAME_CONTROL

Definition at line **35** of file **ami-inter-pan-host.h**.

#define MAX_STUB_APS_SIZE

Definition at line **38** of file **ami-inter-pan-host.h**.

#define MAX_INTER_PAN_HEADER_SIZE

Definition at line **41** of file **ami-inter-pan-host.h**.

Function Documentation

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

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

```
int8u parseInterPanMessage ( EmberMessageBuffer message,
                             int8u              startOffset,
                             InterPanHeader *   headerData
                           )
```

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

```
int8u makeInterPanMessage ( InterPanHeader * headerData,
                            int8u *          message,
                            int8u            maxLength,
                            int8u *          payload,
                            int8u            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.

```
int8u parseInterPanMessage ( int8u *          message,
                             int8u            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.

Ember Status Codes
[Ember Common]

Defines

```
#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_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_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)

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)

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)

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)

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.

Define Documentation

#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**.

#define EMBER_SUCCESS (x00)

The generic "no error" message.

Definition at line **43** of file **error-def.h**.

#define EMBER_ERR_FATAL (x01)

The generic "fatal error" message.

Definition at line **53** of file **error-def.h**.

#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**.

#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 **74** of file **error-def.h**.

#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 **85** of file **error-def.h**.

#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 **96** of file **error-def.h**.

#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 **107** of file **error-def.h**.

#define EMBER_NO_BUFFERS (x18)

There are no more buffers.

Definition at line **124** of file **error-def.h**.

#define EMBER_SERIAL_INVALID_BAUD_RATE (x20)

Specified an invalid baud rate.

Definition at line **140** of file **error-def.h**.

#define EMBER_SERIAL_INVALID_PORT (x21)

Specified an invalid serial port.

Definition at line **150** of file **error-def.h**.

#define EMBER_SERIAL_TX_OVERFLOW (x22)

Tried to send too much data.

Definition at line **160** of file **error-def.h**.

#define EMBER_SERIAL_RX_OVERFLOW (x23)

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

Definition at line **171** of file **error-def.h**.

#define EMBER_SERIAL_RX_FRAME_ERROR (x24)

Detected a UART framing error.

Definition at line **181** of file **error-def.h**.

#define EMBER_SERIAL_RX_PARITY_ERROR (x25)

Detected a UART parity error.

Definition at line **191** of file **error-def.h**.

#define EMBER_SERIAL_RX_EMPTY (x26)

There is no received data to process.

Definition at line **201** of file **error-def.h**.

#define EMBER_SERIAL_RX_OVERRUN_ERROR (x27)

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

Definition at line **212** of file **error-def.h**.

#define EMBER_MAC_TRANSMIT_QUEUE_FULL (x39)

The MAC transmit queue is full.

Definition at line **228** of file **error-def.h**.

#define EMBER_MAC_UNKNOWN_HEADER_TYPE (x3A)

MAC header FCF error on receive.

Definition at line **239** of file **error-def.h**.

#define EMBER_MAC_ACK_HEADER_TYPE (x3B)

MAC ACK header received.

Definition at line **248** of file **error-def.h**.

#define EMBER_MAC_SCANNING (x3D)

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

Definition at line **259** of file **error-def.h**.

#define EMBER_MAC_NO_DATA (x31)

No pending data exists for device doing a data poll.

Definition at line **269** of file **error-def.h**.

#define EMBER_MAC_JOINED_NETWORK (x32)

Attempt to scan when we are joined to a network.

Definition at line **279** of file **error-def.h**.

#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 **290** of file **error-def.h**.

#define EMBER_MAC_INCORRECT_SCAN_TYPE (x34)

emberStartScan was called with an incorrect scan type.

Definition at line **300** of file **error-def.h**.

#define EMBER_MAC_INVALID_CHANNEL_MASK (x35)

emberStartScan was called with an invalid channel mask.

Definition at line **310** of file **error-def.h**.

#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 **321** of file **error-def.h**.

#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 **332** of file **error-def.h**.

#define EMBER_MAC_INDIRECT_TIMEOUT (x42)

Indirect data message timed out before polled.

Definition at line **342** of file **error-def.h**.

#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 **365** of file **error-def.h**.

#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 **381** of file **error-def.h**.

#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 **396** of file **error-def.h**.

#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 **414** of file **error-def.h**.

#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 **427** of file **error-def.h**.

#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 **441** of file **error-def.h**.

#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 **459** of file **error-def.h**.

#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 **480** of file **error-def.h**.

#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 **493** of file **error-def.h**.

#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 **506** of file **error-def.h**.

#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 **519** of file **error-def.h**.

#define EMBER_ERR_BOOTLOADER_TRAP_TABLE_BAD (x58)

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

Definition at line **538** of file **error-def.h**.

#define EMBER_ERR_BOOTLOADER_TRAP_UNKNOWN (x59)

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

Definition at line **549** of file **error-def.h**.

#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 **560** of file **error-def.h**.

#define EMBER_DELIVERY_FAILED (x66)

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

Definition at line **578** of file **error-def.h**.

#define EMBER_BINDING_INDEX_OUT_OF_RANGE (x69)

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

Definition at line **588** of file **error-def.h**.

#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 **599** of file **error-def.h**.

#define EMBER_INVALID_BINDING_INDEX (x6C)

An invalid binding table index was given to a function.

Definition at line **609** of file **error-def.h**.

#define EMBER_INVALID_CALL (x70)

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

Definition at line **620** of file **error-def.h**.

#define EMBER_COST_NOT_KNOWN (x71)

The link cost to a node is not known.

Definition at line **630** of file **error-def.h**.

#define EMBER_MAX_MESSAGE_LIMIT_REACHED (x72)

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

Definition at line **641** of file **error-def.h**.

#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 **651** of file **error-def.h**.

#define EMBER_BINDING_IS_ACTIVE (x75)

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

Definition at line **662** of file **error-def.h**.

#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 **672** of file **error-def.h**.

#define EMBER_ADC_CONVERSION_DONE (x80)

Conversion is complete.

Definition at line **689** of file **error-def.h**.

#define EMBER_ADC_CONVERSION_BUSY (x81)

Conversion cannot be done because a request is being processed.

Definition at line **700** of file **error-def.h**.

#define EMBER_ADC_CONVERSION_DEFERRED (x82)

Conversion is deferred until the current request has been processed.

Definition at line **711** of file **error-def.h**.

#define EMBER_ADC_NO_CONVERSION_PENDING (x84)

No results are pending.

Definition at line **721** of file **error-def.h**.

#define EMBER_SLEEP_INTERRUPTED (x85)

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

Definition at line **732** of file **error-def.h**.

#define EMBER_PHY_TX_UNDERFLOW (x88)

The transmit hardware buffer underflowed.

Definition at line **749** of file **error-def.h**.

#define EMBER_PHY_TX_INCOMPLETE (x89)

The transmit hardware did not finish transmitting a packet.

Definition at line **759** of file **error-def.h**.

#define EMBER_PHY_INVALID_CHANNEL (x8A)

An unsupported channel setting was specified.

Definition at line **769** of file **error-def.h**.

#define EMBER_PHY_INVALID_POWER (x8B)

An unsupported power setting was specified.

Definition at line **779** of file **error-def.h**.

#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 **790** of file **error-def.h**.

#define EMBER_PHY_TX_CCA_FAIL (x8D)

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

Definition at line **801** of file **error-def.h**.

#define EMBER_PHY_OSCILLATOR_CHECK_FAILED (x8E)

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

Definition at line **812** of file **error-def.h**.

#define EMBER_PHY_ACK_RECEIVED (x8F)

The expected ACK was received after the last transmission.

Definition at line **822** of file **error-def.h**.

#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 **841** of file **error-def.h**.

#define EMBER_NETWORK_DOWN (x91)

The network is not operating.

Definition at line **851** of file **error-def.h**.

#define EMBER_JOIN_FAILED (x94)

An attempt to join a network failed.

Definition at line **861** of file **error-def.h**.

#define EMBER_MOVE_FAILED (x96)

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

Definition at line **872** of file **error-def.h**.

#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 **884** of file **error-def.h**.

#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 **894** of file **error-def.h**.

#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 **904** of file **error-def.h**.

#define EMBER_CHANNEL_CHANGED (x9B)

The channel has changed.

Definition at line **912** of file **error-def.h**.

#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 **921** of file **error-def.h**.

#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 **932** of file **error-def.h**.

#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 **942** of file **error-def.h**.

#define EMBER_NO_LINK_KEY_RECEIVED (xAE)

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

Definition at line **952** of file **error-def.h**.

#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 **963** of file **error-def.h**.

#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 **979** of file **error-def.h**.

#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 **989** of file **error-def.h**.

#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 **1003** of file **error-def.h**.

#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 **1012** of file **error-def.h**.

#define EMBER_SECURITY_STATE_NOT_SET (xA8)

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

Definition at line **1021** of file **error-def.h**.

#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 **1034** of file **error-def.h**.

#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 **1043** of file **error-def.h**.

#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 **1054** of file **error-def.h**.

#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 **1063** of file **error-def.h**.

#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 **1075** of file **error-def.h**.

#define EMBER_NOT_JOINED (x93)

The node has not joined a network.

Definition at line **1094** of file **error-def.h**.

#define EMBER_NETWORK_BUSY (xA1)

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

Definition at line **1104** of file **error-def.h**.

#define EMBER_INVALID_ENDPOINT (xA3)

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

Definition at line **1115** of file **error-def.h**.

#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 **1126** of file **error-def.h**.

#define EMBER_INSUFFICIENT_RANDOM_DATA (xA5)

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

Definition at line **1136** of file **error-def.h**.

#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 **1146** of file **error-def.h**.

#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 **1157** of file **error-def.h**.

#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 **1178** of file **error-def.h**.

#define EMBER_INDEX_OUT_OF_RANGE (xB1)

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

Definition at line **1189** of file **error-def.h**.

#define EMBER_TABLE_FULL (xB4)

There are no empty entries left in the table.

Definition at line **1198** of file **error-def.h**.

#define EMBER_TABLE_ENTRY_ERASED (xB6)

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

Definition at line **1208** of file **error-def.h**.

#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 **1218** of file **error-def.h**.

#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 **1228** of file **error-def.h**.

#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 **1239** of file **error-def.h**.

#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 **1257** of file **error-def.h**.

#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 **1258** of file **error-def.h**.

#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 **1259** of file **error-def.h**.

#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 **1260** of file **error-def.h**.

#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 **1261** of file **error-def.h**.

#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 **1262** of file **error-def.h**.

#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 **1263** of file **error-def.h**.

#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 **1264** of file **error-def.h**.

#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 **1265** of file **error-def.h**.

#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 **1266** of file **error-def.h**.

#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 **1267** of file **error-def.h**.

#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 **1268** of file **error-def.h**.

#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 **1269** of file **error-def.h**.

#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 **1270** of file **error-def.h**.

#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 **1271** of file **error-def.h**.

#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 **1272** of file **error-def.h**.

Enumeration Type Documentation

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**.

Smart Energy Security
[Ember Common]

Functions

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

Detailed Description

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

See **cbke-crypto-engine.h** for source code.

Function Documentation

EmberStatus emberGetCertificate (EmberCertificateData * result)

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

Parameters:

result A pointer to an **EmberCertificateData** structure where the retrieved certificate will be stored.

Returns:

EMBER_SUCCESS if the certificate was successfully retrieved. **EMBER_ERR_FATAL** if the token contains uninitialized data.

EmberStatus emberGenerateCbkeKeys (void)

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

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

The generated results of the key generation is returned via **emberGenerateCbkeKeysHandler()**.

Returns:

EMBER_OPERATION_IN_PROGRESS if the stack has queued up the operation for execution.

```
EmberStatus emberCalculateSmacs ( boolean               amInitiator,
                                  EmberCertificateData * partnerCert,
                                  EmberPublicKeyData *   partnerEphemeralPublicKey
                                )
```


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

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

The calculated SMACS are returned via **emberCalculateSmacsHandler()**.

Parameters:

amInitiator	This boolean indicates whether or not the device is the one that initiated the CBKE with the remote device, or whether it was the responder to the exchange.
partnerCert	A pointer to an **EmberCertificateData** structure that contains the CBKE partner's implicit certificate.
partnerEphemeralPublicKey	A pointer to an **EmberPublicKeyData** structure that contains the CBKE partner's ephemeral public key.

Returns:

EMBER_OPERATION_IN_PROGRESS if the stack has queued up the operation for execution.

EmberStatus emberClearTemporaryDataMaybeStoreLinkKey (boolean storeLinkKey)

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

Parameters:

storeLinkKey This tells the stack whether to store the newly generated link key, or discard it.

Returns:

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

EmberStatus emberDsaSign (EmberMessageBuffer messageToSign)

```
void emberGenerateCbkeKeysHandler ( EmberStatus          status,
                                    EmberPublicKeyData * ephemeralPublicKey
                                  )
```

```
void emberCalculateSmacsHandler ( EmberStatus       status,
                                  EmberSmacData *   initiatorSmac,
                                  EmberSmacData *   responderSmac
                                )
```

```
void emberDsaSignHandler ( EmberStatus        status,
                           EmberMessageBuffer signedMessage
                         )
```

```
EmberStatus emberSetPreinstalledCbkeData ( EmberPublicKeyData *   caPublic,
                                           EmberCertificateData * myCert,
                                           EmberPrivateKeyData *  myKey
                                         )
```

boolean emberGetStackCertificateEui64 (EmberEUI64 certEui64)

```
EmberStatus emberDsaVerify ( EmberMessageDigest *  digest,
                             EmberCertificateData * signerCertificate,
```



```
                                EmberSignatureData *  receivedSig
                              )
```

```
void emberDsaVerifyHandler ( EmberStatus  status )
```

Configuration
[Ember Common]

Defines

#define	EMBER_API_MAJOR_VERSION
#define	EMBER_API_MINOR_VERSION
#define	EMBER_STACK_PROFILE
#define	EMBER_MAX_END_DEVICE_CHILDREN
#define	EMBER_SECURITY_LEVEL
#define	EMBER_CHILD_TABLE_SIZE
#define	EMBER_KEY_TABLE_SIZE
#define	EMBER_CERTIFICATE_TABLE_SIZE
#define	EMBER_MAX_DEPTH
#define	EMBER_MAX_HOPS
#define	EMBER_PACKET_BUFFER_COUNT
#define	EMBER_MAX_NEIGHBOR_TABLE_SIZE
#define	EMBER_NEIGHBOR_TABLE_SIZE
#define	EMBER_INDIRECT_TRANSMISSION_TIMEOUT
#define	EMBER_MAX_INDIRECT_TRANSMISSION_TIMEOUT
#define	EMBER_END_DEVICE_POLL_TIMEOUT
#define	EMBER_END_DEVICE_POLL_TIMEOUT_SHIFT
#define	EMBER_MOBILE_NODE_POLL_TIMEOUT
#define	EMBER_APS_UNICAST_MESSAGE_COUNT
#define	EMBER_BINDING_TABLE_SIZE
#define	EMBER_ADDRESS_TABLE_SIZE
#define	EMBER_RESERVED_MOBILE_CHILD_ENTRIES
#define	EMBER_ROUTE_TABLE_SIZE
#define	EMBER_DISCOVERY_TABLE_SIZE
#define	EMBER_MULTICAST_TABLE_SIZE
#define	EMBER_SOURCE_ROUTE_TABLE_SIZE
#define	EMBER_BROADCAST_TABLE_SIZE
#define	EMBER_ASSERT_SERIAL_PORT
#define	EMBER_MAXIMUM_ALARM_DATA_SIZE
#define	EMBER_BROADCAST_ALARM_DATA_SIZE
#define	EMBER_UNICAST_ALARM_DATA_SIZE
#define	EMBER_FRAGMENT_DELAY_MS
#define	EMBER_FRAGMENT_MAX_WINDOW_SIZE
#define	EMBER_FRAGMENT_WINDOW_SIZE
#define	EMBER_BINDING_TABLE_TOKEN_SIZE
#define	EMBER_CHILD_TABLE_TOKEN_SIZE
#define	EMBER_KEY_TABLE_TOKEN_SIZE
#define	EMBER_REQUEST_KEY_TIMEOUT
#define	EMBER_END_DEVICE_BIND_TIMEOUT
#define	EMBER_PAN_ID_CONFLICT_REPORT_THRESHOLD
#define	EMBER_TASK_COUNT
#define	EZSP_HOST_SOURCE_ROUTE_TABLE_SIZE
#define	EZSP_HOST_ASH_RX_POOL_SIZE
#define	EZSP_HOST_FORM_AND_JOIN_BUFFER_SIZE

Detailed Description

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

```
#define EMBER_PACKET_BUFFER_COUNT 8
```

The convenience stubs provided in `hal/ember-configuration.c` can be overridden by defining the appropriate macro and providing the corresponding callback function. For example, an application with custom debug channel input must

implement emberDebugHandler() to process it. Along with the function definition, the application should provide the following line in its CONFIGURATION_HEADER:

```
#define EMBER_APPLICATION_HAS_DEBUG_HANDLER
```

See **ember-configuration-defaults.h** for source code.

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

Define Documentation

#define EMBER_API_MAJOR_VERSION

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

Definition at line **58** of file **ember-configuration-defaults.h**.

#define EMBER_API_MINOR_VERSION

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

Definition at line **65** of file **ember-configuration-defaults.h**.

#define EMBER_STACK_PROFILE

Specifies the stack profile. The default is Profile 0.

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

```
#define EMBER_STACK_PROFILE 1
```

or

```
#define EMBER_STACK_PROFILE 2
```

Definition at line **81** of file **ember-configuration-defaults.h**.

#define EMBER_MAX_END_DEVICE_CHILDREN

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

Definition at line **98** of file **ember-configuration-defaults.h**.

#define EMBER_SECURITY_LEVEL

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

Definition at line **116** of file **ember-configuration-defaults.h**.

#define EMBER_CHILD_TABLE_SIZE

The maximum number of children that a node may have.

For the tree stack this values defaults to the sum of **EMBER_MAX_END_DEVICE_CHILDREN** and EMBER_MAX_ROUTER_CHILDREN. For the mesh stack this defaults to the value of **EMBER_MAX_END_DEVICE_CHILDREN**. In the mesh stack router children are not stored in the child table.

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

Application definitions for **EMBER_CHILD_TABLE_SIZE** that are larger than the default value are ignored and the default value used instead.

Definition at line **145** of file **ember-configuration-defaults.h**.

#define EMBER_KEY_TABLE_SIZE

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

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

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

Definition at line **162** of file **ember-configuration-defaults.h**.

#define EMBER_CERTIFICATE_TABLE_SIZE

The number of entries for the field upgradeable certificate table. Normally certificates (such as SE certs) are stored in the runtime-unmodifiable MFG area. However for those devices wishing to add new certificates after manufacturing, they will have to use the normal token space. This defines the size of that table. For most devices 0 is appropriate since there is no need to change certificates in the field. For those wishing to field upgrade devices with new certificates, 1 is the correct size. Anything more is simply wasting SimEEPROM.

Definition at line **175** of file **ember-configuration-defaults.h**.

#define EMBER_MAX_DEPTH

The maximum depth of the tree in ZigBee 2006. This implicitly determines the maximum diameter of the network (**EMBER_MAX_HOPS**) if that value is not overridden.

Definition at line **188** of file **ember-configuration-defaults.h**.

#define EMBER_MAX_HOPS

The maximum number of hops for a message.

When the radius is not supplied by the Application (i.e. 0) or the stack is sending a message, then the default is two times the max depth (**EMBER_MAX_DEPTH**).

Definition at line **201** of file **ember-configuration-defaults.h**.

#define EMBER_PACKET_BUFFER_COUNT

The number of Packet Buffers available to the Stack. The default is 24.

Each buffer requires 40 bytes of RAM (32 for the buffer itself plus 8 bytes of overhead).

Definition at line **211** of file **ember-configuration-defaults.h**.

#define EMBER_MAX_NEIGHBOR_TABLE_SIZE

The maximum number of router neighbors the stack can keep track of.

A neighbor is a node within radio range. The maximum allowed value is 16. End device children are kept track of in the child table, not the neighbor table. The default is 16. Setting this value lower than 8 is not recommended.

Each neighbor table entry consumes 18 bytes of RAM (6 for the table itself and 12 bytes of security data).

Definition at line **225** of file **ember-configuration-defaults.h**.

#define EMBER_NEIGHBOR_TABLE_SIZE

Definition at line **227** of file **ember-configuration-defaults.h**.

#define EMBER_INDIRECT_TRANSMISSION_TIMEOUT

The maximum amount of time (in milliseconds) that the MAC will hold a message for indirect transmission to a child.

The default is 3000 milliseconds (3 sec). The maximum value is 30 seconds (30000 milliseconds).larger values will cause rollover confusion.

Definition at line **237** of file **ember-configuration-defaults.h**.

#define EMBER_MAX_INDIRECT_TRANSMISSION_TIMEOUT

Definition at line **239** of file **ember-configuration-defaults.h**.

#define EMBER_END_DEVICE_POLL_TIMEOUT

The maximum amount of time, in units determined by **EMBER_END_DEVICE_POLL_TIMEOUT_SHIFT**, that an **EMBER_END_DEVICE** or **EMBER_SLEEPY_END_DEVICE** can wait between polls. The timeout value in seconds is **EMBER_END_DEVICE_POLL_TIMEOUT** << **EMBER_END_DEVICE_POLL_TIMEOUT_SHIFT**. If no poll is heard within this time, then the parent removes the end device from its tables. Note: there is a separate **EMBER_MOBILE_NODE_POLL_TIMEOUT** for mobile end devices.

Using the default values of both **EMBER_END_DEVICE_POLL_TIMEOUT** and **EMBER_END_DEVICE_POLL_TIMEOUT_SHIFT** results in a timeout of 320 seconds, or just over five minutes. The maximum value for **EMBER_END_DEVICE_POLL_TIMEOUT** is 255.

Definition at line **259** of file **ember-configuration-defaults.h**.

#define EMBER_END_DEVICE_POLL_TIMEOUT_SHIFT

The units used for timing out end devices on their parents. See **EMBER_END_DEVICE_POLL_TIMEOUT** for an explanation of how this value is used.

The default value of 6 means gives **EMBER_END_DEVICE_POLL_TIMEOUT** a default unit of 64 seconds, or approximately one minute. The maximum value for **EMBER_END_DEVICE_POLL_TIMEOUT_SHIFT** is 14.

Definition at line **270** of file **ember-configuration-defaults.h**.

#define EMBER_MOBILE_NODE_POLL_TIMEOUT

The maximum amount of time (in quarter-seconds) that a mobile node can wait between polls. If no poll is heard within this timeout, then the parent removes the mobile node from its tables. The default is 20 quarter seconds (5 seconds). The maximum is 255 quarter seconds.

Definition at line **280** of file **ember-configuration-defaults.h**.

#define EMBER_APS_UNICAST_MESSAGE_COUNT

The maximum number of APS retried messages that the stack can be transmitting at any time. Here, "transmitting" means the time between the call to emberSendUnicast() and the subsequent callback to emberMessageSentHandler().

Note:
A message will typically use one packet buffer for the message header and one or more packet buffers for the payload. The default is 10 messages.

Each APS retried message consumes 6 bytes of RAM, in addition to two or more packet buffers.

Definition at line **296** of file **ember-configuration-defaults.h**.

#define EMBER_BINDING_TABLE_SIZE

The maximum number of bindings supported by the stack. The default is 0 bindings. Each binding consumes 2 bytes of RAM.

Definition at line **302** of file **ember-configuration-defaults.h**.

#define EMBER_ADDRESS_TABLE_SIZE

The maximum number of EUI64<->network address associations that the stack can maintain. The default value is 8.

Address table entries are 10 bytes in size.

Definition at line **310** of file **ember-configuration-defaults.h**.

#define EMBER_RESERVED_MOBILE_CHILD_ENTRIES

The number of child table entries reserved for use only by mobile nodes. The default value is 0.

The maximum number of non-mobile children for a parent is **EMBER_CHILD_TABLE_SIZE** - **EMBER_RESERVED_MOBILE_CHILD_ENTRIES**.

Definition at line **320** of file **ember-configuration-defaults.h**.

#define EMBER_ROUTE_TABLE_SIZE

The maximum number of destinations to which a node can route messages. This include both messages originating at this node and those relayed for others. The default value is 16.

Route table entries are 6 bytes in size.

Definition at line **333** of file **ember-configuration-defaults.h**.

#define EMBER_DISCOVERY_TABLE_SIZE

The number of simultaneous route discoveries that a node will support.

Discovery table entries are 9 bytes in size.

Definition at line **349** of file **ember-configuration-defaults.h**.

#define EMBER_MULTICAST_TABLE_SIZE

The maximum number of multicast groups that the device may be a member of. The default value is 8.

Multicast table entries are 3 bytes in size.

Definition at line **362** of file **ember-configuration-defaults.h**.

#define EMBER_SOURCE_ROUTE_TABLE_SIZE

The maximum number of source route table entries supported by the utility code in `app/util/source-route.c`. The maximum source route table size is 255 entries, since a one-byte index is used, and the index 0xFF is reserved. The default value is 32.

Source route table entries are 4 bytes in size.

Definition at line **372** of file **ember-configuration-defaults.h**.

#define EMBER_BROADCAST_TABLE_SIZE

The maximum number broadcasts during a single broadcast timeout period. The minimum and default value is 15 and can only be changed only on compatible Ember stacks.

Broadcast table entries are 5 bytes in size.

Definition at line **381** of file **ember-configuration-defaults.h**.

#define EMBER_ASSERT_SERIAL_PORT

Settings to control if and where assert information will be printed.

The output can be suppressed by defining EMBER_ASSERT_OUTPUT_DISABLED. The serial port to which the output is sent can be changed by defining **EMBER_ASSERT_SERIAL_PORT** as the desired port.

The default is to have assert output on and sent to serial port 1.

Definition at line **399** of file **ember-configuration-defaults.h**.

#define EMBER_MAXIMUM_ALARM_DATA_SIZE

The absolute maximum number of payload bytes in an alarm message.

The three length bytes in **EMBER_UNICAST_ALARM_CLUSTER** messages do not count towards this limit.

EMBER_MAXIMUM_ALARM_DATA_SIZE is defined to be 16.

The maximum payload on any particular device is determined by the configuration parameters, **EMBER_BROADCAST_ALARM_DATA_SIZE** and **EMBER_UNICAST_ALARM_DATA_SIZE**, neither of which may be greater than MBER_MAXIMUM_ALARM_DATA_SIZE.

Definition at line **415** of file **ember-configuration-defaults.h**.

#define EMBER_BROADCAST_ALARM_DATA_SIZE

The sizes of the broadcast and unicast alarm buffers in bytes.

Devices have a single broadcast alarm buffer. Routers have one unicast alarm buffer for each child table entry. The total RAM used for alarms is

```
   EMBER_BROADCAST_ALARM_DATA_SIZE
   + (EMBER_UNICAST_ALARM_DATA_SIZE *
      EMBER_CHILD_TABLE_SIZE)
```

EMBER_BROADCAST_ALARM_DATA_SIZE is the size of the alarm broadcast buffer. Broadcast alarms whose length is larger will not be buffered or forwarded to sleepy end device children. This parameter must be in the inclusive range 0 ... **EMBER_MAXIMUM_ALARM_DATA_SIZE**. The default value is 0.

Definition at line **435** of file **ember-configuration-defaults.h**.

#define EMBER_UNICAST_ALARM_DATA_SIZE

The size of the unicast alarm buffers allocated for end device children.

Unicast alarms whose length is larger will not be buffered or forwarded to sleepy end device children. This parameter must be in the inclusive range 0 ... **EMBER_MAXIMUM_ALARM_DATA_SIZE**. The default value is 0.

Definition at line **449** of file **ember-configuration-defaults.h**.

#define EMBER_FRAGMENT_DELAY_MS

The time the stack will wait (in milliseconds) between sending blocks of a fragmented message. The default value is 0.

Definition at line **458** of file **ember-configuration-defaults.h**.

#define EMBER_FRAGMENT_MAX_WINDOW_SIZE

The maximum number of blocks of a fragmented message that can be sent in a single window is defined to be 8.

Definition at line **464** of file **ember-configuration-defaults.h**.

#define EMBER_FRAGMENT_WINDOW_SIZE

The number of blocks of a fragmented message that can be sent in a single window. The maximum is **EMBER_FRAGMENT_MAX_WINDOW_SIZE**. The default value is 1.

Definition at line **471** of file **ember-configuration-defaults.h**.

#define EMBER_BINDING_TABLE_TOKEN_SIZE

Definition at line **477** of file **ember-configuration-defaults.h**.

#define EMBER_CHILD_TABLE_TOKEN_SIZE

Definition at line **480** of file **ember-configuration-defaults.h**.

#define EMBER_KEY_TABLE_TOKEN_SIZE

Definition at line **483** of file **ember-configuration-defaults.h**.

#define EMBER_REQUEST_KEY_TIMEOUT

The length of time that the device will wait for an answer to its Application Key Request. For the Trust Center this is the time it will hold the first request and wait for a second matching request. If both arrive within this time period, the Trust Center will reply to both with the new key. If both requests are not received then the Trust Center will discard the request. The time is in minutes. The maximum time is 10 minutes. A value of 0 minutes indicates that the Trust Center will not buffer the request but instead respond immediately. Only 1 outstanding request is supported at a time.

The Zigbee Pro Compliant value is 0.

Definition at line **499** of file **ember-configuration-defaults.h**.

#define EMBER_END_DEVICE_BIND_TIMEOUT

The time the coordinator will wait (in seconds) for a second end device bind request to arrive. The default value is 60.

Definition at line **508** of file **ember-configuration-defaults.h**.

#define EMBER_PAN_ID_CONFLICT_REPORT_THRESHOLD

The number of PAN id conflict reports that must be received by the network manager within one minute to trigger a PAN id change. Very rarely, a corrupt beacon can pass the CRC check and trigger a false PAN id conflict. This is more likely to happen in very large dense networks. Setting this value to 2 or 3 dramatically reduces the chances of a spurious PAN id change. The maximum value is 63. The default value is 1.

Definition at line **520** of file **ember-configuration-defaults.h**.

#define EMBER_TASK_COUNT

The number of event tasks that can be tracked for the purpose of processor idling. The EmberZNet stack requires 1, an application and associated libraries may use additional tasks, though typically no more than 3 are needed for most applications.

Definition at line **529** of file **ember-configuration-defaults.h**.

#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**.

#define EZSP_HOST_ASH_RX_POOL_SIZE

Define the size of the ASH 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**.

#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 int16u[], 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**.

HAL Configuration

[Hardware Abstraction Layer (HAL) API Reference]

Modules

- Common PLATFORM_HEADER Configuration

Detailed Description

Configuration information that affects the entire HAL.

Common PLATFORM_HEADER Configuration
[HAL Configuration]

Compiler and Platform specific definitions and typedefs common to all platforms. Some definitions can be overriden by the specific PLATFORM_HEADER for your platform. More...

Modules

STM32F103RET IAR Specific PLATFORM_HEADER Configuration

Master Program Memory Declarations

These are a set of defines for simple declarations of program memory.

#define **PGM**
#define **PGM_P**
#define **PGM_PU**
#define **PGM_NO_CONST**

Divide and Modulus Operations

Some platforms can perform divide and modulus operations on 32 bit quantities more efficiently when the divisor is only a 16 bit quantity. C compilers will always promote the divisor to 32 bits before performing the operation, so the following utility functions are instead required to take advantage of this optimisation.

#define **halCommonUDiv32By16**(x, y)
#define **halCommonSDiv32By16**(x, y)
#define **halCommonUMod32By16**(x, y)
#define **halCommonSMod32By16**(x, y)

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)

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)

Detailed Description

Compiler and Platform specific definitions and typedefs common to all platforms. Some definitions can be overriden by the specific PLATFORM_HEADER for your platform.

platform-common.h provides PLATFORM_HEADER defaults and common definitions. This head should never be included directly, it should only be included by the specific PLATFORM_HEADER used by your platform.

See **platform-common.h** for source code.

Define Documentation

#define PGM

Standard program memory delcaration.

Definition at line **42** of file **platform-common.h**.

#define PGM_P

Char pointer to program memory declaration.

Definition at line **47** of file **platform-common.h**.

#define PGM_PU

Unsigned char pointer to program memory declaration.

Definition at line **52** of file **platform-common.h**.

#define PGM_NO_CONST

Sometimes a second PGM is needed in a declaration. Having two 'const' declarations generates a warning so we have a second PGM that turns into nothing under gcc.

Definition at line **60** of file **platform-common.h**.

#define halCommonUDiv32By16 (x, y)

Provide a portable name for the int32u by int16u division library function (which can perform the division with only a single assembly instruction on some platforms).

Definition at line **80** of file **platform-common.h**.

#define halCommonSDiv32By16 (x, y)

Provide a portable name for the int32s by int16s division library function (which can perform the division with only a single assembly instruction on some platforms).

Definition at line **87** of file **platform-common.h**.

#define halCommonUMod32By16 (x,
y)

Provide a portable name for the int32u by int16u modulo library function (which can perform the division with only a single assembly instruction on some platforms).

Definition at line **94** of file **platform-common.h**.

#define halCommonSMod32By16 (x,
y)

Provide a portable name for the int32s by int16s modulo library function (which can perform the division with only a single assembly instruction on some platforms).

Definition at line **101** of file **platform-common.h**.

#define BIT (x)

Useful to reference a single bit of a byte.

Definition at line **183** of file **platform-common.h**.

#define BIT32 (x)

Useful to reference a single bit of an int32u type.

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

#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 **195** of file **platform-common.h**.

#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 **202** of file **platform-common.h**.

#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 **209** of file **platform-common.h**.

#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 **216** of file **platform-common.h**.

#define READBIT (reg,
bit)

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

Definition at line **221** of file **platform-common.h**.

#define READBITS (reg,
bits)

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

Definition at line **227** of file **platform-common.h**.

#define LOW_BYTE (n)

Returns the low byte of the 16-bit value `n` as an `int8u`.

Definition at line **241** of file **platform-common.h**.

#define HIGH_BYTE (n)

Returns the high byte of the 16-bit value `n` as an `int8u`.

Definition at line **246** of file **platform-common.h**.

#define HIGH_LOW_TO_INT (high,
low)

Returns the value built from the two `int8u` values `high` and `low`.

Definition at line **252** of file **platform-common.h**.

#define BYTE_0 (n)

Returns the low byte of the 32-bit value `n` as an `int8u`.

Definition at line **260** of file **platform-common.h**.

#define BYTE_1 (n)

Returns the second byte of the 32-bit value `n` as an `int8u`.

Definition at line **265** of file **platform-common.h**.

#define BYTE_2 (n)

Returns the third byte of the 32-bit value n as an int8u.

Definition at line **270** of file **platform-common.h**.

#define BYTE_3 (n)

Returns the high byte of the 32-bit value n as an int8u.

Definition at line **275** of file **platform-common.h**.

#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 **290** of file **platform-common.h**.

#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 **297** of file **platform-common.h**.

#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 **304** of file **platform-common.h**.

#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 **311** of file **platform-common.h**.

#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 **312** of file **platform-common.h**.

#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 **313** of file **platform-common.h**.

#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 **320** of file **platform-common.h**.

#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 **321** of file **platform-common.h**.

#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 **322** of file **platform-common.h**.

#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 **329** of file **platform-common.h**.

#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 **330** of file **platform-common.h**.

#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 **331** of file **platform-common.h**.

STM32F103RET IAR Specific PLATFORM_HEADER Configuration
[Common PLATFORM_HEADER Configuration]

Compiler and Platform specific definitions and typedefs for the STM32F103RET Host built with the IAR ARM C compiler. More...

Defines

#define	**halResetWatchdog**()
#define	**SIGNED_ENUM**
#define	**_HAL_USE_COMMON_DIVMOD_**
#define	**_HAL_USE_COMMON_PGM_**
#define	**PLATCOMMONOKTOINCLUDE**

Functions

void	**halInternalResetWatchDog** (void)

Master Variable Types

These are a set of typedefs to make the size of all variable declarations explicitly known. Since the IAR host code links against the ST Standard peripheral library, we need to map Ember's variable types to ST's variable types.

Note:
ST uses IAR's variable types, found in stdint.h.

typedef uint8_t	**boolean**
typedef uint8_t	**int8u**
typedef int8_t	**int8s**
typedef uint16_t	**int16u**
typedef int16_t	**int16s**
typedef uint32_t	**int32u**
typedef int32_t	**int32s**
typedef uint32_t	**PointerType**

Miscellaneous Macros

void	**halInternalAssertFailed** (const char *filename, int linenumber)
#define	**simulatedSerialTimePasses**()
#define	**simulatedSerialTimePasses**()
#define	**BIGENDIAN_CPU**
#define	**MAIN_FUNCTION_PARAMETERS**
#define	**MAIN_FUNCTION_ARGUMENTS**
#define	**__SOURCEFILE__**
#define	**assert**(condition)
#define	**simulatedTimePasses**()
#define	**simulatedTimePassesMs**(x)

Global Interrupt Manipulation Macros

#define	**DISABLE_INTERRUPTS**()
#define	**RESTORE_INTERRUPTS**()
#define	**INTERRUPTS_ON**()
#define	**INTERRUPTS_OFF**()
#define	**INTERRUPTS_ARE_OFF**()
#define	**INTERRUPTS_WERE_ON**()
#define	**ATOMIC**(blah)
#define	**HANDLE_PENDING_INTERRUPTS**()

Generic Types

#define	**NULL**

C Standard Library Memory Utilities

These should be used in place of the standard library functions.

#define	**halCommonMemSet**(d, v, l)
#define	**halCommonMemCopy**(d, s, l)
#define	**halCommonMemCompare**(s0, s1, l)
#define	**halCommonMemPGMCompare**(s0, s1, l)
#define	**halCommonMemPGMCopy**(d, s, l)
#define	**MEMSET**(d, v, l)
#define	**MEMCOPY**(d, s, l)
#define	**MEMCOMPARE**(s0, s1, l)
#define	**MEMPGMCOMPARE**(s0, s1, l)

Detailed Description

Compiler and Platform specific definitions and typedefs for the STM32F103RET Host built with the IAR ARM C compiler.

Note:
iar-st.h should be included first in all source files by setting the preprocessor macro PLATFORM_HEADER to point to it. **iar-st.h** automatically includes **platform-common.h**.

See **Common PLATFORM_HEADER Configuration** for common documentation.

See **iar-st.h** for source code.

Define Documentation

#define halResetWatchdog ( )

Macro to reset the watchdog timer.

Note:
Be very very careful when using this as you can easily get into an infinite loop if you are not careful.

Definition at line **77** of file **iar-st.h**.

#define SIGNED_ENUM

Some platforms need to cast enum values that have the high bit set.

Definition at line **83** of file **iar-st.h**.

#define simulatedSerialTimePasses ( )

Stub for code not running in simulation.

Definition at line **168** of file **iar-st.h**.

#define simulatedSerialTimePasses ( )

Stub for code not running in simulation.

Definition at line **168** of file **iar-st.h**.

#define _HAL_USE_COMMON_DIVMOD_

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

Definition at line **94** of file **iar-st.h**.

#define _HAL_USE_COMMON_PGM_

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

Definition at line **100** of file **iar-st.h**.

#define BIGENDIAN_CPU

A convenient method for code to know what endiannes processor it is running on. For the Cortex-M3, we are little endian.

Definition at line **113** of file **iar-st.h**.

#define MAIN_FUNCTION_PARAMETERS

Define the parameters to main(), and for those functions that are passed the arguments from main().

Definition at line **119** of file **iar-st.h**.

#define MAIN_FUNCTION_ARGUMENTS

Stub for code not running in simulation.

Definition at line **120** of file **iar-st.h**.

#define __SOURCEFILE__

The __SOURCEFILE__ macro is used by asserts to list the filename if it isn't otherwise defined, set it to the compiler intrinsic which specifies the whole filename and path of the sourcefile.

Definition at line **129** of file **iar-st.h**.

#define assert (condition)

A custom implementation of the C language assert macro. This macro implements the conditional evaluation and calls the function **halInternalAssertFailed()**.

Definition at line **145** of file **iar-st.h**.

#define simulatedTimePasses ( )

Stub for code not running in simulation.

Definition at line **160** of file **iar-st.h**.

#define simulatedTimePassesMs (x)

Stub for code not running in simulation.

Definition at line **164** of file **iar-st.h**.

#define DISABLE_INTERRUPTS ( )

Disable interrupts, saving the previous state so it can be later restored with **RESTORE_INTERRUPTS()**.

Note:
Do not fail to call **RESTORE_INTERRUPTS()**.
It is safe to nest this call.

Definition at line **205** of file **iar-st.h**.

#define RESTORE_INTERRUPTS ( )

Restore the global interrupt state previously saved by **DISABLE_INTERRUPTS()**.

Note:
Do not call without having first called **DISABLE_INTERRUPTS()** to have saved the state.
It is safe to nest this call.

Definition at line **219** of file **iar-st.h**.

#define INTERRUPTS_ON ( )

Enable global interrupts without regard to the current or previous state.

Definition at line **229** of file **iar-st.h**.

#define INTERRUPTS_OFF ( )

Disable global interrupts without regard to the current or previous state.

Definition at line **239** of file **iar-st.h**.

#define INTERRUPTS_ARE_OFF ( )

Returns:
TRUE if global interrupts are disabled.

Definition at line **248** of file **iar-st.h**.

#define INTERRUPTS_WERE_ON ( )

Returns:
TRUE if global interrupt flag was enabled when **DISABLE_INTERRUPTS()** was called.

Definition at line **255** of file **iar-st.h**.

#define ATOMIC (blah)

A block of code may be made atomic by wrapping it with this macro. Something which is atomic cannot be interrupted by interrupts.

Definition at line **262** of file **iar-st.h**.

#define HANDLE_PENDING_INTERRUPTS ( )

Allows any pending interrupts to be executed. Usually this would be called at a safe point while interrupts are disabled (such as within an ISR).

Takes no action if interrupts are already enabled.

Definition at line **278** of file **iar-st.h**.

#define NULL

The null pointer.

Definition at line **301** of file **iar-st.h**.

```
#define halCommonMemSet ( d,
                          v,
                          l    )
```

All of the ember defined macros/functions simply redirect to the full C Standard Library as supplied by IAR.

Definition at line **316** of file **iar-st.h**.

```
#define halCommonMemCopy ( d,
                           s,
                           l    )
```

All of the ember defined macros/functions simply redirect to the full C Standard Library as supplied by IAR.

Definition at line **317** of file **iar-st.h**.

```
#define halCommonMemCompare ( s0,
                              s1,
                              l     )
```

All of the ember defined macros/functions simply redirect to the full C Standard Library as supplied by IAR.

Definition at line **318** of file **iar-st.h**.

```
#define halCommonMemPGMCompare ( s0,
                                 s1,
                                 l     )
```

All of the ember defined macros/functions simply redirect to the full C Standard Library as supplied by IAR.

Definition at line **319** of file **iar-st.h**.

```
#define halCommonMemPGMCopy ( d,
                              s,
                              l    )
```

All of the ember defined macros/functions simply redirect to the full C Standard Library as supplied by IAR.

Definition at line **320** of file **iar-st.h**.

```
#define MEMSET ( d,
                 v,
```



```
                    l    )
```

All of the ember defined macros/functions simply redirect to the full C Standard Library as supplied by IAR.

Definition at line **322** of file **iar-st.h**.

```
#define MEMCOPY ( d,
                  s,
                  l    )
```

All of the ember defined macros/functions simply redirect to the full C Standard Library as supplied by IAR.

Definition at line **323** of file **iar-st.h**.

```
#define MEMCOMPARE ( s0,
                     s1,
                     l     )
```

All of the ember defined macros/functions simply redirect to the full C Standard Library as supplied by IAR.

Definition at line **324** of file **iar-st.h**.

```
#define MEMPGMCOMPARE ( s0,
                        s1,
                        l     )
```

All of the ember defined macros/functions simply redirect to the full C Standard Library as supplied by IAR.

Definition at line **325** of file **iar-st.h**.

```
#define PLATCOMMONOKTOINCLUDE
```

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

Definition at line **333** of file **iar-st.h**.

Typedef Documentation

```
typedef uint8_t boolean
```

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

Definition at line **54** of file **iar-st.h**.

```
typedef uint8_t int8u
```

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

Definition at line **55** of file **iar-st.h**.

```
typedef int8_t int8s
```

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

Definition at line **56** of file **iar-st.h**.

```
typedef uint16_t int16u
```


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

Definition at line **57** of file **iar-st.h**.

typedef int16_t int16s

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

Definition at line **58** of file **iar-st.h**.

typedef uint32_t int32u

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

Definition at line **59** of file **iar-st.h**.

typedef int32_t int32s

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

Definition at line **60** of file **iar-st.h**.

typedef uint32_t PointerType

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

Definition at line **61** of file **iar-st.h**.

Function Documentation

void halInternalResetWatchDog (void)

Internal function to reset the watchdog timer.

Note:
Be very very careful when using this as you can easily get into an infinite loop if you are not careful.

void halInternalAssertFailed (const char * filename,
int linenumber
)

A prototype definition for use by the assert macro.

Microcontroller General Functionality
[Hardware Abstraction Layer (HAL) API Reference]

HAL functions common across all microcontroller-specific files. More...

Modules

	STM32F103RET General Functionality
	ST Microcontroller Standard Peripherals Library Inclusions and Definitions

Defines

#define	**MICRO_DISABLE_WATCH_DOG_KEY**

Enumerations

```
enum   SleepModes {
         SLEEPMODE_RUNNING,
         SLEEPMODE_IDLE,
         SLEEPMODE_WAKETIMER,
         SLEEPMODE_MAINTAINTIMER,
         SLEEPMODE_NOTIMER,
         SLEEPMODE_RESERVED,
         SLEEPMODE_POWERDOWN,
         SLEEPMODE_POWERSAVE
       }
```

Functions

void	**halInit** (void)
void	**halReboot** (void)
void	**halPowerUp** (void)
void	**halPowerDown** (void)
void	**halInternalEnableWatchDog** (void)
void	**halInternalDisableWatchDog** (**int8u** magicKey)
void	**halCommonDelayMicroseconds** (**int16u** us)
void	**halCommonDelayMilliseconds** (**int16u** ms)
void	**halInternalAssertFailed** (PGM_P filename, int linenumber)
int8u	**halGetResetInfo** (void)
PGM_P	**halGetResetString** (void)
void	**halStackSeedRandom** (**int32u** seed)
int16u	**halCommonGetRandom** (void)
void	**halSleep** (**SleepModes** sleepMode)

Detailed Description

HAL functions common across all microcontroller-specific files.

Note:
The micro specific definitions, **STM32F103RET General Functionality**, is chosen by the build include path pointing at the appropriate directoy.

See **micro-common.h** for source code.

Define Documentation

#define MICRO_DISABLE_WATCH_DOG_KEY

The value that must be passed as the single parameter to **halInternalDisableWatchDog()** in order to sucessfully disable the watchdog timer.

Definition at line **41** of file **micro-common.h**.

Enumeration Type Documentation

enum SleepModes

Enumerations for the possible microcontroller sleep modes.

NOTE: Refer to a specific micro's implementation of **halSleep()** to see what modes are actually supported.

- SLEEPMODE_RUNNING Everything is active and running. In practice this mode is not used, but it is defined for completeness of information.
- SLEEPMODE_IDLE Only the CPU is idled. The rest of the chip continues runing normally. The chip will wake from any interrupt.
- SLEEPMODE_WAKETIMER The sleep timer clock sources remain running. The RC is always running and the 32kHz XTAL depends on system timer config. Wakeup is possible from both GPIO and the sleep timer. System time is maintained. The sleep timer is assumed to be configured properly for wake events.
- SLEEPMODE_MAINTAINTIMER The sleep timer clock sources remain running. The RC is always running and the 32kHz XTAL depends on the board header. Wakeup is possible from only GPIO. System time is maintained.
- SLEEPMODE_NOTIMER The sleep timer clock sources (both RC and XTAL) are turned off. Wakeup is possible from only GPIO. System time is lost.
- SLEEPMODE_RESERVED Reserved/Unused
- SLEEPMODE_POWERDOWN Deprecated
- SLEEPMODE_POWERSAVE Deprecated

Enumerator:

SLEEPMODE_RUNNING
SLEEPMODE_IDLE
SLEEPMODE_WAKETIMER
SLEEPMODE_MAINTAINTIMER
SLEEPMODE_NOTIMER
SLEEPMODE_RESERVED
SLEEPMODE_POWERDOWN
SLEEPMODE_POWERSAVE

Definition at line **157** of file **micro-common.h**.

Function Documentation

void halInit (void)

Initializes microcontroller-specific peripherals.

void halReboot (void)

Restarts the microcontroller.

void halPowerUp (void)

Powers up microcontroller peripherals.

void halPowerDown (void)

Powers down microcontroller peripherals.

void halInternalEnableWatchDog (void)

Enables the watchdog timer, if there is one and it is reasonable to be enabled.

void halInternalDisableWatchDog (int8u magicKey)

Disables the watchdog timer, if there is one and it can be disabled.

Note:
To prevent the watchdog from being disabled accidentally, a magic key must be provided.

Parameters:
magicKey A value (**MICRO_DISABLE_WATCH_DOG_KEY**) that enables the function.

void halCommonDelayMicroseconds (int16u us)

Blocks the current thread of execution for the specified amount of time, in microseconds.

The function is implemented with either cycle-counted busy loops or a convenient timer. It is intended to create the short blocking delays such as when interfacing with hardware peripherals.

The accuracy of the timing provided by this function is not specified, but a best faith effort is obtain an accurate delay. The implementation may be changed, but this function should be reasonably accurate.

Parameters:
us The specified time, in microseconds. Values should be between 1 and 65535 microseconds.

void halCommonDelayMilliseconds (int16u ms)

Blocks the current thread of execution for the specified amount of time, in milliseconds..

This function depends on **halCommonDelayMicroseconds()**.

Parameters:
ms The specified time, in milliseconds.

void halInternalAssertFailed (PGM_P filename,
int linenumber
)

Called implicitly through the standard C language **assert()** macro. An implementation where notification is, for instance, sent over the serial port can provide meaningful and useful debugging information.

Note:
Liberal usage of **assert()** consumes flash space.

Parameters:
filename Name of the file throwing the assert.
linenumber Line number that threw the assert.

int8u halGetResetInfo (void)

Gets information about what caused the microcontroller to reset.

Returns:
A code identifying the cause of the reset.

PGM_P halGetResetString (void)

Calls **halGetResetInfo()** and supplies a string describing it.

Returns:
A pointer to a program space string.

void halStackSeedRandom (int32u seed)

Seeds the **halCommonGetRandom()** pseudorandom number generator.

Parameters:

seed A seed for the pseudorandom number generator.

int16u halCommonGetRandom (void)

Generate pseudorandom numbers. Implementation is host specific.

void halSleep (SleepModes sleepMode)

Puts the microcontroller to sleep in a specified mode.

Note:

This routine always enables interrupts.

Parameters:

sleepMode A microcontroller sleep mode

STM32F103RET General Functionality
[Microcontroller General Functionality]

HAL functions specific to this micro. More...

Defines

#define	**MILLISECOND_TICKS_PER_SECOND**

Functions

void	**halInternalInitSysTick** (void)

#define	**RESET_UNKNOWN**
#define	**RESET_LOW_POWER**
#define	**RESET_WINDOW_WATCHDOG**
#define	**RESET_INDEPENDENT_WATCHDOG**
#define	**RESET_SOFTWARE**
#define	**RESET_POR_PDR**
#define	**RESET_PIN**
#define	**RESET_UNSET**

Detailed Description

HAL functions specific to this micro.

See **Microcontroller General Functionality** for common documentation.

The definitions in the micro specific header provide the necessary pieces to link the common functionality to a specific micro.

See **micro-specific.h** for source code.

Define Documentation

#define MILLISECOND_TICKS_PER_SECOND

The number of ticks specific to this host (as returned from **halCommonGetInt32uMillisecondTick**) that represent an actual second.

Definition at line **27** of file **micro-specific.h**.

#define RESET_UNKNOWN

A name given to a reset event. The name is derived from the datasheet and the value is the index into the resetString structure.

Definition at line **35** of file **micro-specific.h**.

#define RESET_LOW_POWER

A name given to a reset event. The name is derived from the datasheet and the value is the index into the resetString structure.

Definition at line **36** of file **micro-specific.h**.

#define RESET_WINDOW_WATCHDOG

A name given to a reset event. The name is derived from the datasheet and the value is the index into the resetString structure.

Definition at line **37** of file **micro-specific.h**.

#define RESET_INDEPENDENT_WATCHDOG

A name given to a reset event. The name is derived from the datasheet and the value is the index into the resetString structure.

Definition at line **38** of file **micro-specific.h**.

#define RESET_SOFTWARE

A name given to a reset event. The name is derived from the datasheet and the value is the index into the resetString structure.

Definition at line **39** of file **micro-specific.h**.

#define RESET_POR_PDR

A name given to a reset event. The name is derived from the datasheet and the value is the index into the resetString structure.

Definition at line **40** of file **micro-specific.h**.

#define RESET_PIN

A name given to a reset event. The name is derived from the datasheet and the value is the index into the resetString structure.

Definition at line **41** of file **micro-specific.h**.

#define RESET_UNSET

A name given to a reset event. The name is derived from the datasheet and the value is the index into the resetString structure.

Definition at line **42** of file **micro-specific.h**.

Function Documentation

void halInternalInitSysTick (void)

Initialize the SysTick timer to provide a microsecond time base for use by **halCommonDelayMicroseconds()**.

ST Microcontroller Standard Peripherals Library Inclusions and Definitions

[Microcontroller General Functionality]

ST Microcontroller's Standard Peripherals Library inclusions and definitions. More...

Defines

```
#define assert_param(condition)
```

Functions

```
void halInternalAssertFailed (const char *filename, int linenumber)
```

Detailed Description

ST Microcontroller's Standard Peripherals Library inclusions and definitions.

This file is included from ST's Standard Peripherals Library and includes the headers for the peripherals found in ST's Library. It also defines the assert macro used by ST's Library. The actual documentation for ST's Standard Peripherals Library is beyond the scope of Ember's documentation.

Note:

While this file's name, **stm32f10x_conf.h**, does not conform to Ember's file naming convention, this file is included from ST's Standard Peripherals Library. Not renaming this file means the library does not have to be modified.

See **Microcontroller General Functionality** for common documentation.

See **stm32f10x_conf.h** for source code.

Define Documentation

#define assert_param (condition)

The assert_param macro is used by ST's Library to check a function's parameters. This macro redirects to Ember's assert function. This macro redirect is the same definition of assert as used in the the PLATFORM_HEADER.

Definition at line **75** of file **stm32f10x_conf.h**.

Function Documentation

void halInternalAssertFailed (const char * filename, int linenumber)

A prototype definition of the Ember assert function for use by the assert_param macro.

SPI Protocol

[Hardware Abstraction Layer (HAL) API Reference]

Example host common SPI Protocol implementation for interfacing with a NCP. More...

Modules

	STM32F103RET Specific SPI Protocol

Functions

void	**halNcpSerialInit** (void)
void	**halNcpSerialPowerup** (void)
void	**halNcpSerialPowerdown** (void)
EzspStatus	**halNcpHardReset** (void)
EzspStatus	**halNcpHardResetReqBootload** (**boolean** requestBootload)
void	**halNcpWakeUp** (void)
void	**halNcpSendCommand** (void)
void	**halNcpSendRawCommand** (void)
EzspStatus	**halNcpPollForResponse** (void)
void	**halNcpIsAwakeIsr** (**boolean** isAwake)
boolean	**halNcpHasData** (void)
boolean	**halNcpVerifySpiProtocolVersion** (void)
boolean	**halNcpVerifySpiProtocolActive** (void)

Variables

int8u *	**halNcpFrame**
int8u	**halNcpSpipErrorByte**

Detailed Description

Example host common SPI Protocol implementation for interfacing with a NCP.

For complete documentation of the SPI Protocol, refer to the NCP docs.

Note:
The micro specific definitions, **STM32F103RET Specific SPI Protocol**, is chosen by the build include path pointing at the appropriate directoy.

See **spi-protocol-common.h** for source code.

Function Documentation

void halNcpSerialInit (void)

Initializes the SPI Protocol.

void halNcpSerialPowerup (void)

Reinitializes the SPI Protocol when coming out of sleep (powerdown).

void halNcpSerialPowerdown (void)

Shuts down the SPI Protocol when entering sleep (powerdown).

EzspStatus halNcpHardReset (void)

Forcefully resets the NCP by pulling on the nRESET line; waits for the NCP to boot; verifies that is has booted; verifies the NCP is active; verifies the SPI Protocol version. When this function returns, the NCP is ready to accept all commands.

This function is the same as **halNcpHardResetReqBootload()**, except that the NCP cannot be told to enter bootload mode through the nWAKE signal.

Returns:
A EzspStatus value indicating the success or failure of the command.

EzspStatus halNcpHardResetReqBootload (boolean requestBootload)

Forcefully resets the NCP by pulling on the nRESET line; sets the nWAKE signal based upon the state of the requestBootload boolean; waits for the NCP to boot; verifies that is has booted; verifies the NCP is active; verifies the SPI Protocol version. When this function returns, the NCP is ready to accept all commands.

This function is the same as **halNcpHardReset()**, except that the ability to request the NCP enter bootload mode through the nWAKE signal is made available.

Returns:
A EzspStatus value indicating the success or failure of the command.

void halNcpWakeUp (void)

If the Host thinks that the NCP is sleeping and wants to wake it up, the EZSP calls **halNcpWakeUp()**.

Waking up can take some time (milliseconds) so **halNcpWakeUp()** returns immediately and the SPI Protocol calls **halNcpIsAwakeIsr()** once the wakeup handshaking is complete and the NCP is ready to accept commands.

void halNcpSendCommand (void)

The EZSP writes a command into the command buffer and then calls **halNcpSendCommand()**.

This function assumes the command being sent is an EZSP frame and therefore sets the SPI Byte for an EZSP Frame. If sending a command other than EZSP, use **halNcpSendRawCommand()**. This function returns immediately after transmission of the Command has completed and the transaction has entered the Wait section. The EZSP must now call **halNcpPollForResponse()** until the Response is received.

void halNcpSendRawCommand (void)

The upper layer writes a command into the command buffer and then calls **halNcpSendRawCommand()**.

This function makes no assumption about the data in the SpipBuffer, it will just faithly try to perform the transaction. This function returns immediately after transmission of the Command has completed and the transaction has entered the Wait section. The upper layer must now call **halNcpPollForResponse()** until the the Response is received.

EzspStatus halNcpPollForResponse (void)

After sending a Command with **halNcpSendCommand()**, the upper layer repeatedly calls this function until the SPI Protocol has finished reception of a Response.

Returns:
A EzspStatus value indicating the success or failure of the command.

void halNcpIsAwakeIsr (boolean isAwake)

The SPI Protocol calls **halNcpIsAwakeIsr()** once the wakeup handshaking is complete and the NCP is ready to accept a command.

Parameters:
isAwake TRUE if the wake handshake completed and the NCP is awake. FALSE is the wake handshake failed and the NCP is unresponsive.

boolean halNcpHasData (void)

If the Host wants to find out whether the NCP has a pending callback, the EZSP calls **halNcpHasData()**. If this function returns TRUE then the EZSP will send a callback command.

boolean halNcpVerifySpiProtocolVersion (void)

Transmits the SPI Protocol Version Command and checks the response against a literal value to verify the SPI Protocol version.

Returns:

TRUE if the SPI Protocol Version used in this function matches the version returned by the NCP. FALSE is the versions do not match.

boolean halNcpVerifySpiProtocolActive (void)

Transmits the SPI Status Command and checks the response against a literal value to verify the SPI Protocol is active.

Returns:

TRUE if the SPI Protocol is active. FALSE if the SPI Protocol is not active.

Variable Documentation

int8u* halNcpFrame

A pointer to the length byte at the start of the Payload. Upper layers will write the command to this location before starting a transaction. The upper layer will read the response from this location after a transaction completes. This pointer is the upper layers' primary access into the command/response buffer.

int8u halNcpSpipErrorByte

This error byte is the third byte found in a special SPI Protocol error case. It provides more detail concerning the error. Refer to the NCP docs for a more detailed description of this byte. The application does not need to work with this byte, but it can be useful information when developing.

STM32F103RET Specific SPI Protocol
[SPI Protocol]

Example host specific SPI Protocol implementation for interfacing with a NCP. More...

SPI Protocol Interface

#define	**SPIP_nSSEL_PORT**
#define	**SPIP_nSSEL_PIN**
#define	**SPIP_MOSI_PORT**
#define	**SPIP_MOSI_PIN**
#define	**SPIP_MISO_PORT**
#define	**SPIP_MISO_PIN**
#define	**SPIP_SCLK_PORT**
#define	**SPIP_SCLK_PIN**
#define	**SPIP_nHOST_INT_PORT**
#define	**SPIP_nHOST_INT_PIN**
#define	**SPIP_nWAKE_PORT**
#define	**SPIP_nWAKE_PIN**
#define	**SPIP_nRESET_PORT**
#define	**SPIP_nRESET_PIN**

SPI Protocol timing parameters.

Note:
Remember: TIM2 is configured to produce a 125us tick.

#define	**WAIT_SECTION_TIMEOUT**
#define	**WAKE_HANDSHAKE_TIMEOUT**
#define	**STARTUP_TIMEOUT**
#define	**INTER_COMMAND_SPACING**
#define	**NCP_RESET_DELAY**

Detailed Description

Example host specific SPI Protocol implementation for interfacing with a NCP.

For complete documentation of the SPI Protocol, refer to the NCP docs.

See **SPI Protocol** for common documentation.

The definitions in the micro specific header provide the necessary pieces to link the common functionality to a specific micro.

See **spi-protocol-specific.h** for source code.

Define Documentation

#define SPIP_nSSEL_PORT

The actual port that nSSEL is connected to, PA4, which is configured as a general purpose output.

Definition at line **33** of file **spi-protocol-specific.h**.

#define SPIP_nSSEL_PIN

The actual pin that nSSEL is connected to, PA4, which is configured as a general purpose output.

Definition at line **37** of file **spi-protocol-specific.h**.

#define SPIP_MOSI_PORT

The actual port that MOSI is connected to, PA7, which is configured as alternate function push-pull.

Definition at line **43** of file **spi-protocol-specific.h**.

#define SPIP_MOSI_PIN

The actual pin that MOSI is connected to, PA7, which is configured as alternate function push-pull.

Definition at line **47** of file **spi-protocol-specific.h**.

#define SPIP_MISO_PORT

The actual port that MISO is connected to, PA6, which is configured as input with pull-up.

Definition at line **52** of file **spi-protocol-specific.h**.

#define SPIP_MISO_PIN

The actual pin that MISO is connected to, PA6, which is configured as input with pull-up.

Definition at line **56** of file **spi-protocol-specific.h**.

#define SPIP_SCLK_PORT

The actual port that SCLK is connected to, PA5, which is configured as alternate functon push-pull.

Definition at line **61** of file **spi-protocol-specific.h**.

#define SPIP_SCLK_PIN

The actual pin that SCLK is connected to, PA5, which is configured as alternate functon push-pull.

Definition at line **65** of file **spi-protocol-specific.h**.

#define SPIP_nHOST_INT_PORT

The actual port that nHOST_INT is connected to, PC4, which is configured as input with pull-up; EXTI4 interrupt, falling edge.

Definition at line **70** of file **spi-protocol-specific.h**.

#define SPIP_nHOST_INT_PIN

The actual pin that nHOST_INT is connected to, PC4, which is configured as input with pull-up; EXTI4 interrupt, falling edge.

Definition at line **74** of file **spi-protocol-specific.h**.

#define SPIP_nWAKE_PORT

The actual port that nWAKE is connected to, PC5, which is configured as general purpose output.

Definition at line **79** of file **spi-protocol-specific.h**.

#define SPIP_nWAKE_PIN

The actual pin that nWAKE is connected to, PC5, which is configured as general purpose output.

Definition at line **83** of file **spi-protocol-specific.h**.

#define SPIP_nRESET_PORT

The actual port that nRESET is connected to, PB0, which is configured as general purpose output.

Definition at line **88** of file **spi-protocol-specific.h**.

#define SPIP_nRESET_PIN

The actual pin that nRESET is connected to, PB0, which is configured as general purpose output.

Definition at line **92** of file **spi-protocol-specific.h**.

#define WAIT_SECTION_TIMEOUT

Wait section timeout is 200ms.

Definition at line **106** of file **spi-protocol-specific.h**.

#define WAKE_HANDSHAKE_TIMEOUT

Wait handshake timeout is 10ms.

Definition at line **110** of file **spi-protocol-specific.h**.

#define STARTUP_TIMEOUT

Startup timeout is 7500ms.

Definition at line **114** of file **spi-protocol-specific.h**.

#define INTER_COMMAND_SPACING

Intercommand spacing is 1ms.

Definition at line **118** of file **spi-protocol-specific.h**.

#define NCP_RESET_DELAY

The time to assert nRESET is 26 microseconds.

Definition at line **122** of file **spi-protocol-specific.h**.

System Timer

[Hardware Abstraction Layer (HAL) API Reference]

Functions that provide access to the system timer. More...

Functions

int16u	halInternalStartSystemTimer (void)
int16u	halCommonGetInt16uMillisecondTick (void)
int32u	halCommonGetInt32uMillisecondTick (void)
int16u	halCommonGetInt16uQuarterSecondTick (void)
void	halCommonSetSystemTime (int32u time)

Detailed Description

Functions that provide access to the system timer.

A single system tick (as returned by **halCommonGetInt16uMillisecondTick()** and **halCommonGetInt32uMillisecondTick()**) is approximately 1 millisecond.

Note:
The actual time of a tick is specific to each micro.

A single quarter-second tick (as returned by **halCommonGetInt16uQuarterSecondTick()**) is approximately 0.25 seconds.

The values used by the time support functions will wrap after an interval. The length of the interval depends on the length of the tick and the number of bits in the value. However, there is no issue when comparing time deltas of less than half this interval with a subtraction, if all data types are the same.

See **system-timer.h** for source code.

Function Documentation

int16u halInternalStartSystemTimer (void)

Initializes the system tick.

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

int16u 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.

int32u halCommonGetInt32uMillisecondTick (void)

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

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

int16u halCommonGetInt16uQuarterSecondTick (void)

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

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

void halCommonSetSystemTime (int32u time)

Set the current system time.

Parameters:

time A 32 bit value, expressed in milliseconds, that will become the current system time.

Sample APIs for Peripheral Access

[Hardware Abstraction Layer (HAL) API Reference]

Modules

- Serial UART Communication
- ADC Control
- Button Control
- Buzzer Control
- LED Control
- Bootloader EEPROM Control

Detailed Description

These are sample API for accessing peripherals and can be modified as needed for your applications.

Serial UART Communication

[Sample APIs for Peripheral Access]

This API contains the common HAL interfaces that hosts must implement for the high-level serial code. More...

Modules

	STM32F103RET Specific UART

Enumerations

enum	**SerialBaudRate** { **DEFINE_BAUD**, **DEFINE_BAUD**, **DEFINE_BAUD**, **DEFINE_BAUD**, **DEFINE_BAUD**, **DEFINE_BAUD**, **DEFINE_BAUD**, **DEFINE_BAUD**, **DEFINE_BAUD**, **DEFINE_BAUD**, **DEFINE_BAUD**, **DEFINE_BAUD**, **DEFINE_BAUD**, **DEFINE_BAUD**, **DEFINE_BAUD**, **DEFINE_BAUD**, **DEFINE_BAUD**, **DEFINE_BAUD** }
enum	**NameOfType** { **DEFINE_PARITY**, **DEFINE_PARITY**, **DEFINE_PARITY** }

Serial HAL APIs

These functions must be implemented by the HAL in order for the serial code to operate. Only the higher-level serial code uses these functions, so they should not be called directly. The HAL should also implement the appropriate interrupt handlers to drain the TX queues and fill the RX FIFO queue, as necessary.

EmberStatus	**halInternalUartInit** (**int8u** port, **SerialBaudRate** rate, SerialParity parity, **int8u** stopBits)
int16u	**halInternalPrintfWriteAvailable** (void)
int16u	**halInternalPrintfReadAvailable** (void)
void	**halInternalForcePrintf** (**boolean** onOff)

Detailed Description

This API contains the common HAL interfaces that hosts must implement for the high-level serial code.

This header describes the interface between the high-level serial APIs in **app/util/serial/serial.h** and the low level UART implementation.

Some functions in this file return an **EmberStatus** value. See **error-def.h** for definitions of all **EmberStatus** return values.

See serial.h for source code.

Enumeration Type Documentation

enum SerialBaudRate

Assign numerical values for variables that hold Baud Rate parameters.

Enumerator:

DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD

Definition at line **33** of file **hal/host/serial.h**.

enum NameOfType

Assign numerical values for the types of parity. Use for variables that hold Parity parameters.

Enumerator:

DEFINE_PARITY
DEFINE_PARITY
DEFINE_PARITY

Definition at line **69** of file **hal/host/serial.h**.

Function Documentation

EmberStatus halInternalUartInit (int8u port, SerialBaudRate rate, SerialParity parity, int8u stopBits)

Initializes the UART to the given settings (same parameters as **emberSerialInit()**).

Parameters:

port	Serial port number (0 or 1).
rate	Baud rate (see SerialBaudRate).
parity	Parity value (see SerialParity).
stopBits	Number of stop bits.

Returns:

An error code if initialization failed (such as invalid baud rate), otherise EMBER_SUCCESS.

int16u halInternalPrintfWriteAvailable (void)

Returns the number bytes available in the transmit queue when using the EMBER_SERIAL_USE_STDIO variant of the Ember serial library.

Returns:
Number of bytes available in the transmit queue.

int16u halInternalPrintfReadAvailable (void)

Returns the number bytes available in the receive queue when using the EMBER_SERIAL_USE_STDIO variant of the Ember serial library.

Returns:
Number of bytes available in the receive queue.

void halInternalForcePrintf (boolean onOff)

This function enables/disables EMBER_SERIAL_USE_STDIO printing behavior that is compatible with **emberSerialGuaranteedPrintf()** and a replacement for halInternalForceWriteUartData(). (blocking, bypass queue, and polling).

STM32F103RET Specific UART
[Serial UART Communication]

STM32F102RET host uart driver operating on top of ST's Standard Peripheral Library; supporting IAR's standard library IO routines. More...

Defines

#define **stdout**

Functions

size_t **fflush** (int handle)

Detailed Description

STM32F102RET host uart driver operating on top of ST's Standard Peripheral Library; supporting IAR's standard library IO routines.

See **Serial UART Communication** for common documentation.

See **uart.h** for source code.

Define Documentation

#define stdout

Define the stdout stream. Since we compile with DLib_Config_Normal.h it does not define 'stdout'. There is a low-level IO define '_LLIO_STDOUT' which is equivalent to stdout. Therefore, we define 'stdout' to be '_LLIO_STDOUT'.

Definition at line **41** of file **uart.h**.

Function Documentation

size_t fflush (int handle)

Flush the output stream. DLib_Config_Full.h defines **fflush()**, but this library includes too much code so we compile with DLib_Config_Normal.h instead which does not define **fflush()**. Therefore, we manually define **fflush()** in the low level UART driver. This function simply redirects to the __write() function with a NULL buffer, triggering a flush.

Parameters:

handle The output stream. Should be set to 'stdout' like normal.

Returns:

Zero, indicating success.

ADC Control

[Sample APIs for Peripheral Access]

Modules

- STM32F103RET Specific ADC

Detailed Description

There is no common ADC functionality, only micro specific functionality.

STM32F103RET Specific ADC
[ADC Control]

Example API functions for operating an ADC. More...

Defines

#define	**TEMP_SENSOR_PIN**
#define	**TEMP_SENSOR_PORT**
#define	**TEMP_SENSOR_ADC**
#define	**TEMP_SENSOR_ADC_CHAN**
#define	**TEMP_ENABLE_PIN**
#define	**TEMP_ENABLE_PORT**

Functions

void	**halInternalInitAdc** (void)
int16u	**halSampleAdc** (void)
int16s	**halConvertValueToVolts** (**int16u** value)

Detailed Description

Example API functions for operating an ADC.

Note:
On the STM32F103RET example host, this driver is written specifically to interact with the breakout board temp sensor.

See **adc.h** for source code.

Define Documentation

#define TEMP_SENSOR_PIN

The actual pin that the temp sensor is connected to.

Definition at line **22** of file **adc.h**.

#define TEMP_SENSOR_PORT

The actual port that the temp sensor is connected to.

Definition at line **25** of file **adc.h**.

#define TEMP_SENSOR_ADC

The actual ADC that the temp sensor is connected to.

Definition at line **28** of file **adc.h**.

#define TEMP_SENSOR_ADC_CHAN

The actual ADC channel that the temp sensor is connected to.

Definition at line **31** of file **adc.h**.

#define TEMP_ENABLE_PIN

The actual pin that the temp sensor enable is connected to.

Definition at line **35** of file **adc.h**.

#define TEMP_ENABLE_PORT

The actual port that the temp sensor enable is connected to.

Definition at line **38** of file **adc.h**.

Function Documentation

void halInternalInitAdc (void)

Initialize the ADC.

int16u halSampleAdc (void)

Take a raw reading of the ADC.

Note:
This function is blocking.

Returns:
The raw value read from the ADC.

int16s halConvertValueToVolts (int16u value)

Convert the raw register value (the unaltered value taken directly from the ADC's data register) into a signed fixed point value with units 10^-4 Volts.

Parameters:
value An int16u to be converted.

Returns:
Volts as signed fixed point with units 10^-4 Volts.

Button Control
[Sample APIs for Peripheral Access]

Sample generic API funtions for using push-buttons. More...

Modules

	STM32F103RET Specific Button

Functions

void	**halInternalInitButton** (void)
int8u	**halButtonState** (**int8u** button)
int8u	**halButtonPinState** (**int8u** button)
void	**halButtonIsr** (**int8u** button, **int8u** state)

Button State Definitions

A set of numerical definitions for use with the button APIs indicating the state of a button.

#define	**BUTTON_PRESSED**
#define	**BUTTON_RELEASED**

Detailed Description

Sample generic API funtions for using push-buttons.

Note:
The micro specific definitions, **STM32F103RET Specific Button**, is chosen by the build include path pointing at the appropriate directoy.

See **button-common.h** for source code.

Define Documentation

#define BUTTON_PRESSED

Button state is pressed.

Definition at line **29** of file **button-common.h**.

#define BUTTON_RELEASED

Button state is released.

Definition at line **33** of file **button-common.h**.

Function Documentation

void halInternalInitButton (void)

Initializes the buttons. Must be called before the buttons can be used.

int8u halButtonState (int8u button)

Returns the current state (pressed or released) of a button.

Note:
This function is correlated with **halButtonIsr()** and so returns the shadow state rather than reading the actual state of the pin.

Parameters:

button The button being queried, either BUTTON0 or BUTTON1 as defined in **button-specific.h**.

Returns:

BUTTON_PRESSED if the button is pressed or **BUTTON_RELEASED** if the button is not pressed.

int8u halButtonPinState (int8u button)

Returns the current state (pressed or released) of the pin associated with a button.

This reads the actual state of the pin and can be used on startup to determine the initial position of the buttons.

Parameters:

button The button being queried, either BUTTON0 or BUTTON1 as defined in **button-specific.h**.

Returns:

BUTTON_PRESSED if the button is pressed or **BUTTON_RELEASED** if the button is not pressed.

void halButtonIsr (int8u button, int8u state)

A callback called in interrupt context whenever a button changes its state.

Application Usage:

Must be implemented by the application. This function should contain the functionality to be executed in response to changes of state in each of the buttons, or callbacks to the appropriate functionality.

Parameters:

button The button which has changed state, either BUTTON0 or BUTTON1 as defined in **button-specific.h**.

state The new state of the button referenced by the button parameter, either **BUTTON_PRESSED** if the button has been pressed or **BUTTON_RELEASED** if the button has been released.

STM32F103RET Specific Button
[Button Control]

Sample micro specific API funtions and defines for using push-buttons. More...

Defines

#define	**BUTTON0**
#define	**BUTTON0_PIN**
#define	**BUTTON0_PORT**
#define	**BUTTON0_EXTI_SOURCE_PORT**
#define	**BUTTON0_EXTI_SOURCE_PIN**
#define	**BUTTON0_IRQ**
#define	**BUTTON1**
#define	**BUTTON1_PIN**
#define	**BUTTON1_PORT**
#define	**BUTTON1_EXTI_SOURCE_PORT**
#define	**BUTTON1_EXTI_SOURCE_PIN**
#define	**BUTTON1_IRQ**
#define	**BUTTON01_ISR**

Detailed Description

Sample micro specific API funtions and defines for using push-buttons.

See **Button Control** for common documentation.

The definitions in the micro specific header provide the necessary pieces to link the common functionality to a specific micro.

See **button-specific.h** for source code.

Define Documentation

#define BUTTON0

Simple numerical definition of BUTTON0.

Definition at line **26** of file **button-specific.h**.

#define BUTTON0_PIN

The actual pin that BUTTON0 is connected to.

Definition at line **30** of file **button-specific.h**.

#define BUTTON0_PORT

The actual port that BUTTON0 is connected to.

Definition at line **34** of file **button-specific.h**.

#define BUTTON0_EXTI_SOURCE_PORT

The actual source port that BUTTON0 is connected to for external interrupts.

Definition at line **39** of file **button-specific.h**.

#define BUTTON0_EXTI_SOURCE_PIN

The actual source pin that BUTTON0 is connected to for external interrupts.

Definition at line **44** of file **button-specific.h**.

#define BUTTON0_IRQ

The actual external interrupt IRQ number for BUTTON0.

Definition at line **48** of file **button-specific.h**.

#define BUTTON1

Simple numerical definition of BUTTON1.

Definition at line **53** of file **button-specific.h**.

#define BUTTON1_PIN

The actual pin that BUTTON1 is connected to.

Definition at line **57** of file **button-specific.h**.

#define BUTTON1_PORT

The actual port that BUTTON1 is connected to.

Definition at line **61** of file **button-specific.h**.

#define BUTTON1_EXTI_SOURCE_PORT

The actual source port that BUTTON1 is connected to for external interrupts.

Definition at line **66** of file **button-specific.h**.

#define BUTTON1_EXTI_SOURCE_PIN

The actual source pin that BUTTON1 is connected to for external interrupts.

Definition at line **71** of file **button-specific.h**.

#define BUTTON1_IRQ

The actual external interrupt IRQ number for BUTTON1.

Definition at line **75** of file **button-specific.h**.

#define BUTTON01_ISR

The actual external interrupt ISR handler. Due to the choice of GPIO, BUTTON0 and BUTTON1 share the same ISR handler.

Definition at line **81** of file **button-specific.h**.

Buzzer Control

[Sample APIs for Peripheral Access]

Modules

- STM32F103RET Specific Buzzer

Detailed Description

There is no common buzzer functionality, only micro specific functionality.

STM32F103RET Specific Buzzer
[Buzzer Control]

Example API funtions for operating a piezo buzzer. More...

Functions

void	**halPlayTune_P** (**int8u** PGM *tune, **boolean** bkg)
void	**halStartBuzzerTone** (**int16u** frequency)
void	**halStopBuzzerTone** (void)

Variables

int8u PGM	**hereIamTune** []

Note Definitions

Flats are used instead of sharps because # is a special character.

#define	**NOTE_C3**
#define	**NOTE_Db3**
#define	**NOTE_D3**
#define	**NOTE_Eb3**
#define	**NOTE_E3**
#define	**NOTE_F3**
#define	**NOTE_Gb3**
#define	**NOTE_G3**
#define	**NOTE_Ab3**
#define	**NOTE_A3**
#define	**NOTE_Bb3**
#define	**NOTE_B3**
#define	**NOTE_C4**
#define	**NOTE_Db4**
#define	**NOTE_D4**
#define	**NOTE_Eb4**
#define	**NOTE_E4**
#define	**NOTE_F4**
#define	**NOTE_Gb4**
#define	**NOTE_G4**
#define	**NOTE_Ab4**
#define	**NOTE_A4**
#define	**NOTE_Bb4**
#define	**NOTE_B4**
#define	**NOTE_C5**
#define	**NOTE_Db5**
#define	**NOTE_D5**
#define	**NOTE_Eb5**
#define	**NOTE_E5**
#define	**NOTE_F5**
#define	**NOTE_Gb5**
#define	**NOTE_G5**
#define	**NOTE_Ab5**
#define	**NOTE_A5**
#define	**NOTE_Bb5**
#define	**NOTE_B5**

Detailed Description

Example API funtions for operating a piezo buzzer.

Note:

On the STM32F103RET example host, the buzzer is tied to GPIO PC6 using TIM3 Channel 1.

See **buzzer.h** for source code.

Define Documentation

#define NOTE_C3

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **32** of file **buzzer.h**.

#define NOTE_Db3

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **33** of file **buzzer.h**.

#define NOTE_D3

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **34** of file **buzzer.h**.

#define NOTE_Eb3

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **35** of file **buzzer.h**.

#define NOTE_E3

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **36** of file **buzzer.h**.

#define NOTE_F3

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **37** of file **buzzer.h**.

#define NOTE_Gb3

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **38** of file **buzzer.h**.

#define NOTE_G3

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **39** of file **buzzer.h**.

#define NOTE_Ab3

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **40** of file **buzzer.h**.

#define NOTE_A3

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **41** of file **buzzer.h**.

#define NOTE_Bb3

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **42** of file **buzzer.h**.

#define NOTE_B3

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **43** of file **buzzer.h**.

#define NOTE_C4

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **44** of file **buzzer.h**.

#define NOTE_Db4

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **45** of file **buzzer.h**.

#define NOTE_D4

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **46** of file **buzzer.h**.

#define NOTE_Eb4

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **47** of file **buzzer.h**.

#define NOTE_E4

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **48** of file **buzzer.h**.

#define NOTE_F4

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **49** of file **buzzer.h**.

#define NOTE_Gb4

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **50** of file **buzzer.h**.

#define NOTE_G4

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **51** of file **buzzer.h**.

#define NOTE_Ab4

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **52** of file **buzzer.h**.

#define NOTE_A4

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **53** of file **buzzer.h**.

#define NOTE_Bb4

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **54** of file **buzzer.h**.

#define NOTE_B4

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **55** of file **buzzer.h**.

#define NOTE_C5

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The

division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **56** of file **buzzer.h**.

#define NOTE_Db5

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **57** of file **buzzer.h**.

#define NOTE_D5

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **58** of file **buzzer.h**.

#define NOTE_Eb5

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **59** of file **buzzer.h**.

#define NOTE_E5

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **60** of file **buzzer.h**.

#define NOTE_F5

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **61** of file **buzzer.h**.

#define NOTE_Gb5

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **62** of file **buzzer.h**.

#define NOTE_G5

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **63** of file **buzzer.h**.

#define NOTE_Ab5

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **64** of file **buzzer.h**.

#define NOTE_A5

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **65** of file **buzzer.h**.

#define NOTE_Bb5

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **66** of file **buzzer.h**.

#define NOTE_B5

A note which can be used in tune structure definitions. These definitions are simply the actual note frequencies. The division by 4 is necessary to get the frequencies to fit in a byte.

Definition at line **67** of file **buzzer.h**.

Function Documentation

void halPlayTune_P (int8u PGM * tune, boolean bkg)

Plays a tune on the piezo buzzer.

The tune is played in the background if bkg is TRUE. Otherwise, the API blocks until the playback of the tune is complete.

Parameters:

tune A pointer to tune to play.

bkg Determines whether the tune plays in the background. If TRUE, tune plays in background; if FALSE, tune plays in foreground.

A tune is implemented as follows:

```
int8u PGM hereIamTune[] = {     //All tunes are stored in flash.
   NOTE_B4,  1,                 //Plays the note B4 for 100 milliseconds.
   0,        1,                 //Pause for 100 milliseconds.
   NOTE_B4,  1,                 //Plays the note B4 for 100 milliseconds.
   0,        1,                 //Pause for 100 milliseconds.
   NOTE_B4,  1,                 //Plays the note B4 for 100 milliseconds.
   0,        1,                 //Pause for 100 milliseconds.
   NOTE_B5,  5,                 //Plays the note B5 for 500 milliseconds.
   0,        0                  //NULL terminates the tune.
 };
```

void halStartBuzzerTone (int16u frequency)

Plays a tone on the piezo buzzer. The tone will play continuously until **halStopBuzzerTone()** is called.

Parameters:

frequency The frequency of the tone to play.

void halStopBuzzerTone (void)

Stops playing a tone that was started by **halStartBuzzerTone()**.

Variable Documentation

int8u PGM hereIamTune[]

Extern definition of Ember's traditional little "here I am" announcement tune, which lives in the buzzer module.

LED Control
[Sample APIs for Peripheral Access]

Sample generic API funtions for controlling LEDs. More...

Modules

	STM32F103RET Specific LED

Typedefs

typedef enum **HalBoardLedPins**	**HalBoardLed**

Functions

void	**halInternalInitLed** (void)
void	**halToggleLed** (**HalBoardLed** led)
void	**halSetLed** (**HalBoardLed** led)
void	**halClearLed** (**HalBoardLed** led)

Detailed Description

Sample generic API funtions for controlling LEDs.

When specifying an LED to use, always use the BOARDLEDx definitions that are defined in the HalBoardLedPins enum in the micro specific led header.

Note:
The micro specific definitions, **STM32F103RET Specific LED**, is chosen by the build include path pointing at the appropriate directoy.

See **led-common.h** for source code.

Typedef Documentation

typedef enum HalBoardLedPins HalBoardLed

Ensures that the definitions for the LEDs are always used as parameters to the LED functions.

Note:
Even though many compilers will use 16 bits for an enum instead of 8, we choose to use an enum here. The possible compiler inefficiency does not affect stack-based parameters and local variables, which is the general case for led paramters.

Definition at line **37** of file **led-common.h**.

Function Documentation

void halInternalInitLed (void)

Configures GPIOs pertaining to the control of LEDs.

void halToggleLed (HalBoardLed led)

Atomically wraps an XOR or similar operation for a single GPIO pin attached to an LED.

Parameters:
led Identifier for the LED to be toggled.

void halSetLed (HalBoardLed led)

Turns on (sets) a GPIO pin connected to an LED so that the LED turns on.

Parameters:

led Identifier for the LED to turn on.

void halClearLed (HalBoardLed led)

Turns off (clears) a GPIO pin connected to an LED, which turns off the LED.

Parameters:

led Identifier for the LED to turn off.

STM32F103RET Specific LED
[LED Control]

Sample micro specific API funtions and defines for controlling LEDs. More...

Defines

```
#define  BOARDLED0_PIN
#define  BOARDLED0_PORT
#define  BOARDLED1_PIN
#define  BOARDLED1_PORT
```

Enumerations

```
enum  HalBoardLedPins {
        BOARDLED0,
        BOARDLED1,
        BOARD_ACTIVITY_LED,
        BOARD_HEARTBEAT_LED
      }
```

Detailed Description

Sample micro specific API funtions and defines for controlling LEDs.

See **LED Control** for common documentation.

The definitions in the micro specific header provide the necessary pieces to link the common functionality to a specific micro.

See **led-specific.h** for source code.

Define Documentation

#define BOARDLED0_PIN

The actual pin that BOARDLED0 is connected to.

Definition at line **39** of file **led-specific.h**.

#define BOARDLED0_PORT

The actual port that BOARDLED0 is connected to.

Definition at line **44** of file **led-specific.h**.

#define BOARDLED1_PIN

The actual pin that BOARDLE1 is connected to.

Definition at line **50** of file **led-specific.h**.

#define BOARDLED1_PORT

The actual port that BOARDLED1 is connected to.

Definition at line **55** of file **led-specific.h**.

Enumeration Type Documentation

enum HalBoardLedPins

Assign each LED to a convenient name that is a simple identifier. BOARD_ACTIVITY_LED and BOARD_HEARTBEAT_LED provide a further layer of abstraction ontop of the LEDs for verbose coding.

Enumerator:

BOARDLED0
BOARDLED1
BOARD_ACTIVITY_LED
BOARD_HEARTBEAT_LED

Definition at line **28** of file **led-specific.h**.

Bootloader EEPROM Control
[Sample APIs for Peripheral Access]

Functions and definitions for generic EEPROM operation. More...

Defines

#define	**EEPROM_PAGE_SIZE**
#define	**EEPROM_FIRST_PAGE**
#define	**EEPROM_IMAGE_START**
#define	**EEPROM_SUCCESS**
#define	**EEPROM_ERR**
#define	**EEPROM_ERR_MASK**
#define	**EEPROM_ERR_PG_BOUNDARY**
#define	**EEPROM_ERR_PG_SZ**
#define	**EEPROM_ERR_WRT_DATA**
#define	**EEPROM_ERR_IMG_SZ**
#define	**EEPROM_ERR_ADDR**

EEPROM interaction functions.

void	**halEepromInit** (void)
void	**halEepromShutdown** (void)
int8u	**halEepromRead** (**int32u** address, **int8u** *data, **int16u** len)
int8u	**halEepromWrite** (**int32u** address, const **int8u** *data, **int16u** len)

Detailed Description

Functions and definitions for generic EEPROM operation.

Changing EEPROM size will change the size of the application image space without changing the size or relative location of the recovery and reserved sections. See eeprom.c for more information on modifying EEPROM functionality.

See **bootloader-eeprom.h** for source code.

Define Documentation

#define EEPROM_PAGE_SIZE

Definition of an EEPROM page size, in bytes. The current interface assumes all eeproms have 128 byte pages. If a device has a different physical page size, the driver needs to abstract it to a 128 byte page.

Definition at line **24** of file **bootloader-eeprom.h**.

#define EEPROM_FIRST_PAGE

Define the location of the first page in EEPROM.

Definition at line **28** of file **bootloader-eeprom.h**.

#define EEPROM_IMAGE_START

Define the location of the image start in EEPROM as a function of the **EEPROM_FIRST_PAGE** and **EEPROM_PAGE_SIZE**.

Definition at line **33** of file **bootloader-eeprom.h**.

#define EEPROM_SUCCESS

Define EEPROM success status.

Definition at line **37** of file **bootloader-eeprom.h**.

#define EEPROM_ERR

Define EEPROM error status.

Definition at line **41** of file **bootloader-eeprom.h**.

#define EEPROM_ERR_MASK

Define EEPROM error mask.

Definition at line **45** of file **bootloader-eeprom.h**.

#define EEPROM_ERR_PG_BOUNDARY

Define EEPROM page boundary error.

Definition at line **49** of file **bootloader-eeprom.h**.

#define EEPROM_ERR_PG_SZ

Define EEPROM page size error.

Definition at line **53** of file **bootloader-eeprom.h**.

#define EEPROM_ERR_WRT_DATA

Define EEPROM write data error.

Definition at line **57** of file **bootloader-eeprom.h**.

#define EEPROM_ERR_IMG_SZ

Define EEPROM image too large error.

Definition at line **61** of file **bootloader-eeprom.h**.

#define EEPROM_ERR_ADDR

Define EEPROM invalid address error.

Definition at line **65** of file **bootloader-eeprom.h**.

Function Documentation

void halEepromInit (void)

Initialize EEPROM.

void halEepromShutdown (void)

Shutdown the EEPROM to conserve power.


```
int8u halEepromRead ( int32u  address,
                      int8u * data,
                      int16u  len
                    )
```

Read from the external EEPROM.

This is the standard external EEPROM read function. The format of this call must not be altered. However, the content can be changed to work with a different device. To more easily work with a larger variety of external EEPROM/flash parts, the app bootloader will always call this function with addresses that are a multiple of 128 bytes.

Parameters:

address The address to start reading from.
data A pointer to where read data is stored.
len The length of data to read.

Returns:

EEPROM_SUCCESS

```
int8u halEepromWrite ( int32u        address,
                       const int8u * data,
                       int16u        len
                     )
```

Write to the external EEPROM.

This is the standard external EEPROM write function. The format of this call must not be altered. However, the content can be changed to work with a different device. To more easily work with a larger variety of external EEPROM/flash parts, the app bootloader will always call this function with addresses that are a multiple of 128 bytes.

Parameters:

address The address to start writing to.
data A pointer to the data to write.
len The length of data to write.

Returns:

EEPROM_SUCCESS

HAL Utilities

[Hardware Abstraction Layer (HAL) API Reference]

Modules

- Cyclic Redundancy Code (CRC)

Cyclic Redundancy Code (CRC)
[HAL Utilities]

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

Defines

#define	**INITIAL_CRC**
#define	**CRC32_START**
#define	**CRC32_END**

Functions

int16u	**halCommonCrc16** (**int8u** newByte, **int16u** prevResult)
int32u	**halCommonCrc32** (**int8u** newByte, **int32u** prevResult)

Detailed Description

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

Define Documentation

#define INITIAL_CRC

Commonly used initial CRC32 value.

Definition at line **51** of file **crc.h**.

#define CRC32_START

Commonly used initial CRC32 value.

Definition at line **56** of file **crc.h**.

#define CRC32_END

Commonly used end CRC32 value for polynomial run LSB-MSB.

Definition at line **61** of file **crc.h**.

Function Documentation

```
int16u halCommonCrc16 ( int8u  newByte,
                        int16u 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.

```
int32u halCommonCrc32 ( int8u  newByte,
                        int32u 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.

Forming and Joining Networks
[Application Utilities API Reference]

Defines

#define	**NETWORK_STORAGE_SIZE**
#define	**NETWORK_STORAGE_SIZE_SHIFT**
#define	**FORM_AND_JOIN_MAX_NETWORKS**

Functions

EmberStatus	**emberScanForUnusedPanId** (**int32u** channelMask, **int8u** duration)
EmberStatus	**emberScanForJoinableNetwork** (**int32u** channelMask, **int8u** *extendedPanId)
EmberStatus	**emberScanForNextJoinableNetwork** (void)
boolean	**emberFormAndJoinIsScanning** (void)
void	**emberUnusedPanIdFoundHandler** (**EmberPanId** panId, **int8u** channel)
void	**emberJoinableNetworkFoundHandler** (**EmberZigbeeNetwork** *networkFound, **int8u** lqi, **int8s** rssi)
void	**emberScanErrorHandler** (**EmberStatus** status)
boolean	**emberFormAndJoinScanCompleteHandler** (**int8u** channel, **EmberStatus** status)
boolean	**emberFormAndJoinNetworkFoundHandler** (**EmberZigbeeNetwork** *networkFound, **int8u** lqi, **int8s** rssi)
boolean	**emberFormAndJoinEnergyScanResultHandler** (**int8u** channel, **int8s** maxRssiValue)
void	**emberFormAndJoinTick** (void)
void	**emberFormAndJoinTaskInit** (void)
void	**emberFormAndJoinRunTask** (void)

Variables

boolean	**emberEnableDualChannelScan**

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.

This library improves upon the form-and-join library from EmberZNet 3.2 and prior. The old library (form-and-join3_2) was removed from the release. The currently provided 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 new library can also be used on the host processor.

Define Documentation

#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 **71** of file **form-and-join.h**.

#define NETWORK_STORAGE_SIZE_SHIFT

Log_base2 of **NETWORK_STORAGE_SIZE**.

Definition at line **75** of file **form-and-join.h**.

#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 **97** of file **form-and-join.h**.

Function Documentation

EmberStatus emberScanForUnusedPanId (int32u channelMask, int8u 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.

EmberStatus emberScanForJoinableNetwork (int32u channelMask, int8u * 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
extendedPanId

Returns:

EMBER_LIBRARY_NOT_PRESENT if the form and join library is not available.

EmberStatus emberScanForNextJoinableNetwork (void)

See **emberScanForJoinableNetwork()**.

---

boolean 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.

---

void emberUnusedPanIdFoundHandler (EmberPanId panId,
int8u channel
)

A callback the application needs to implement.

Notifies the application of the PAN id and channel found following a call to **emberScanForUnusedPanId()**.

Parameters:

panId
channel

---

void emberJoinableNetworkFoundHandler (EmberZigbeeNetwork * networkFound,
int8u lqi,
int8s rssi
)

A callback the application needs to implement.

Notifies the application of the network found after a call to **emberScanForJoinableNetwork()** or **emberScanForNextJoinableNetwork()**.

Parameters:

networkFound
lqi The lqi value of the received beacon.
rssi The rssi value of the received beacon.

---

void emberScanErrorHandler (EmberStatus status)

A callback the application needs to implement.

If an error occurs while scanning, this function is called and the scan effort is aborted.

Possible return status values are:

- EMBER_INVALID_CALL: if **emberScanForNextJoinableNetwork()** is called more than 30 seconds after a previous call to **emberScanForJoinableNetwork()** or **emberScanForNextJoinableNetwork()**.
- EMBER_NO_BUFFERS: if there is not enough memory to start a scan.
- EMBER_NO_BEACONS: if no joinable beacons are found.
- EMBER_MAC_SCANNING: if a scan is already in progress.

Parameters:

status

boolean emberFormAndJoinScanCompleteHandler (int8u 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.

boolean emberFormAndJoinNetworkFoundHandler (EmberZigbeeNetwork * networkFound, int8u lqi, int8s 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.

boolean emberFormAndJoinEnergyScanResultHandler (int8u channel, int8s 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.

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.

void emberFormAndJoinTaskInit (void)

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

void emberFormAndJoinRunTask (void)

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

Variable Documentation

boolean 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.

Bootloading

[Application Utilities API Reference]

Modules

- Stand-Alone Bootloader for EZSP
- Stand-Alone Bootloader Library

Detailed Description

For a thorough discussion of bootloading, see the Bootloading chapter of the *EmberZNet Application Developer's Guide*. There are three forms of the bootloading API.

Stand-Alone Bootloader for EZSP
[Bootloading]

Defines

#define	**TICKS_PER_QUARTER_SECOND**

Functions

boolean	**hostBootloadUtilLaunchRequestHandler** (**int8u** lqi, **int8s** rssi, **int16u** manufacturerId, **int8u** *hardwareTag, **EmberEUI64** sourceEui)
void	**hostBootloadUtilQueryResponseHandler** (**int8u** lqi, **int8s** rssi, **boolean** bootloaderActive, **int16u** manufacturerId, **int8u** *hardwareTag, **EmberEUI64** targetEui, **int8u** bootloaderCapabilities, **int8u** platform, **int8u** micro, **int8u** phy, **int16u** blVersion)
void	**hostBootloadReinitHandler** (void)
boolean	**isTheSameEui64** (**EmberEUI64** sourceEui, **EmberEUI64** targetEui)
void	**printLittleEndianEui64** (**int8u** port, **EmberEUI64** eui64)
void	**printBigEndianEui64** (**int8u** port, **EmberEUI64** eui64)
EmberStatus	**debugPrintf** (**int8u** port, PGM_P formatString,...)

Variables

int16u	**nodeBlVersion**
int8u	**nodePlat**
int8u	**nodeMicro**
int8u	**nodePhy**
EzspStatus	**bootloadEzspLastError**
EzspStatus	**ignoreNextEzspError**

Detailed Description

All functions and variables defined here can be used by applications. See **bootload-ezsp-utils.h** for source code.

Define Documentation

#define TICKS_PER_QUARTER_SECOND

Definition at line **23** of file **bootload-ezsp-utils.h**.

Function Documentation

boolean hostBootloadUtilLaunchRequestHandler (**int8u** **lqi,**
int8s **rssi,**
int16u **manufacturerId,**
int8u * **hardwareTag,**
EmberEUI64 **sourceEui**
)

A callback function invoked by bootload-ezsp-utils when a bootload launch request message is received.

The application may choose whether or not to enter the bootloader by checking the manufacturerId, hardwareTag, and sourceEui. If the application chooses to launch the bootloader, the bootloader will launch after successful completion of the bootloader launch authentication protocol.

Parameters:

lqi	The link quality from the node that generated this bootload launch request.
rssi	The energy level (in units of dBm) observed during the reception.
manufacturerId	The manufacturer specification (vendor specific) of the sending node.
hardwareTag	The hardware specification (vendor specific) of the sending node.
sourceEui	The EUI64 of the sending node.

Returns:

TRUE if the application wishes to launch the bootloader, FALSE if the application does not wish to launch the bootloader.

```
void hostBootloadUtilQueryResponseHandler ( int8u      lqi,
                                            int8s      rssi,
                                            boolean    bootloaderActive,
                                            int16u     manufacturerId,
                                            int8u *    hardwareTag,
                                            EmberEUI64 targetEui,
                                            int8u      bootloaderCapabilities,
                                            int8u      platform,
                                            int8u      micro,
                                            int8u      phy,
                                            int16u     blVersion
                                          )
```

A callback function invoked by bootload-ezsp-utils when a bootload query response message is received.

This is particularly useful when the application needs to decide which node to bootload. Several attributes of the responding node are provided to the application. The application can use these attributes to decide whether to bootload or how to bootload a given node.

Parameters:

lqi	The link quality from the node that generated this bootload query response.
rssi	The energy level (in units of dBm) observed during the reception.
bootloaderActive	TRUE if the responding node is running the bootloader, FALSE if not.
manufacturerId	The manufacturer specification (vendor specific) of the responding node.
hardwareTag	The hardware specification (vendor specific) of the responding node.
targetEui	The EUI64 of the responding node.
bootloaderCapabilities	If the lsb is 1, the bootloader on the responding node supports encrypted bootloader message payloads.
platform	The type of platform of the responding node. 1 is avr-atmega, 2 is xap2b.
micro	The type of microcontroller on the responding node. Value depends on platform. 1 is the avr-atmega 64, 2 is the avr-atmega 128, 1 is the xap2b em250.
phy	The type of phy of the responding node. 1 is em2420, 2 is em250.
blVersion	The version of standalone bootloader of the responding node. This is a 2 byte field. The high byte is the version and the low byte is the build. A value of 0xFFFF means unknown. For example, a version field of 0x1234 is version 1.2, build 34.

```
void hostBootloadReinitHandler ( void )
```

A callback function invoked by bootload-ezsp-utils when a NCP has finished being bootloaded.

The application can handle this as simply as calling on **halReboot()** or as complex as needed.

```
boolean isTheSameEui64 ( EmberEUI64 sourceEui,
                         EmberEUI64 targetEui
                       )
```

A function to compare EUI64s.

Compare two EUI64s.

Parameters:

sourceEui The EUI64 of the sending node.

targetEui The EUI64 of the responding node.

Returns:

TRUE if the EUI64s are the same. FALSE if the EUI64s are different.

void printLittleEndianEui64 (int8u port,
EmberEUI64 eui64
)

A function to display an EUI64.

Display an EUI64 in little endian format.

Parameters:

port The serial port to use. 0 for Mega128 port. 0 or 1 for Linux ports.
eui64 The EUI64 to display.

void printBigEndianEui64 (int8u port,
EmberEUI64 eui64
)

A function to display an EUI64.

Display an EUI64 in big endian format.

Parameters:

port The serial port to use. 0 for Mega128 port. 0 or 1 for Linux ports.
eui64 The EUI64 to display.

EmberStatus debugPrintf (int8u port,
PGM_P formatString,
...
)

A function to simular to **emberSerialPrintf()**.

Output to local ports.

Parameters:

port The serial port to use. 0 for Mega128 port. 0 or 1 for Linux ports.
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.

Variable Documentation

int16u nodeBlVersion

int8u nodePlat

int8u nodeMicro

int8u nodePhy

EzspStatus bootloadEzspLastError

EzspStatus ignoreNextEzspError

Stand-Alone Bootloader Library
[Bootloading]

Defines

#define	**BOOTLOAD_HARDWARE_TAG_SIZE**

Enumerations

enum	**bootloadMode** {
	BOOTLOAD_MODE_NONE,
	BOOTLOAD_MODE_PASSTHRU
	}

enum	**bootloadState** {
	BOOTLOAD_STATE_NORMAL,
	BOOTLOAD_STATE_QUERY,
	BOOTLOAD_STATE_WAIT_FOR_AUTH_CHALLENGE,
	BOOTLOAD_STATE_WAIT_FOR_AUTH_RESPONSE,
	BOOTLOAD_STATE_DELAY_BEFORE_START,
	BOOTLOAD_STATE_START_UNICAST_BOOTLOAD,
	BOOTLOAD_STATE_START_BROADCAST_BOOTLOAD,
	BOOTLOAD_STATE_START_SENDING_IMAGE,
	BOOTLOAD_STATE_SENDING_IMAGE,
	BOOTLOAD_STATE_WAIT_FOR_IMAGE_ACK,
	BOOTLOAD_STATE_WAIT_FOR_COMPLETE_ACK,
	BOOTLOAD_STATE_DONE
	}

Functions

void	**bootloadUtilInit** (**int8u** appPort, **int8u** bootloadPort)
EmberStatus	**bootloadUtilSendRequest** (**EmberEUI64** targetEui, **int16u** mfgId, **int8u** hardwareTag[BOOTLOAD_HARDWARE_TAG_SIZE], **int8u** encryptKey[BOOTLOAD_AUTH_COMMON_SIZE], **bootloadMode** mode)
void	**bootloadUtilSendQuery** (**EmberEUI64** target)
void	**bootloadUtilStartBootload** (**EmberEUI64** target, **bootloadMode** mode)
void	**bootloadUtilTick** (void)
boolean	**bootloadUtilLaunchRequestHandler** (**int16u** manufacturerId, **int8u** hardwareTag[BOOTLOAD_HARDWARE_TAG_SIZE], **EmberEUI64** sourceEui)
void	**bootloadUtilQueryResponseHandler** (**boolean** bootloaderActive, **int16u** manufacturerId, **int8u** hardwareTag[BOOTLOAD_HARDWARE_TAG_SIZE], **EmberEUI64** targetEui, **int8u** bootloaderCapabilities, **int8u** platform, **int8u** micro, **int8u** phy, **int16u** blVersion)
void	**bootloadUtilSendAuthResponse** (**EmberEUI64** target)

Bootload State Variables

Used to check whether a bootloading process is currently happening.

bootloadState	**blState**
#define	**IS_BOOTLOADING**

Authentication Challenge and Response

The authentication challenge and response must be the same size. The size is chosen to be evenly divisible by the size of a 128-bit AES block.

#define	**BOOTLOAD_AUTH_COMMON_SIZE**
#define	**BOOTLOAD_AUTH_CHALLENGE_SIZE**
#define	**BOOTLOAD_AUTH_RESPONSE_SIZE**

Detailed Description

All functions and variables defined here can be used by applications. See **bootload-utils.h** for source code.

Applications can use this stand-alone bootload library to:

1. Load a new (application) image on itself via serial bootload through uart port 1 using the xmodem protocol.
2. Load a new image on a remote node over-the-air (OTA) from a host (PC), also known as a passthru bootload.
3. Recover a node that failed during the bootloading process, also known as a recovery bootload.

Note from the diagrams below that with over-the-air bootloading the source node (node transmitting bootload packets) and the target node (node being loaded with a new image) need to be one hop away because bootload packets are IEEE 802.15.4 packets.

In case of recovery, the source (recovery) node does not need to be part of the network since all recovery packets are 802.15.4 packets.

A diagram for typical serial bootloading:

[host pc] --(RS232 or Ethernet/IP network)-- {uart1 or port 4901}[node]

A diagram for typical passthru bootloading:

[host pc] --(RS232 or Ethernet)-- [source node]--(OTA)--[target node]

A diagram for typical recovery bootloading:

[source node] --(OTA)--[target node]

Note:

Applications that use the bootload utilities need to `#define EMBER_APPLICATION_HAS_BOOTLOAD_HANDLERS` within their `CONFIGURATION_HEADER` .

Define Documentation

#define BOOTLOAD_AUTH_COMMON_SIZE

Definition at line **66** of file **bootload-utils.h**.

#define BOOTLOAD_AUTH_CHALLENGE_SIZE

Definition at line **67** of file **bootload-utils.h**.

#define BOOTLOAD_AUTH_RESPONSE_SIZE

Definition at line **68** of file **bootload-utils.h**.

#define BOOTLOAD_HARDWARE_TAG_SIZE

Size of hardware tag which is an array of int8u.

// End set of defines

Definition at line **76** of file **bootload-utils.h**.

#define IS_BOOTLOADING

Definition at line **300** of file **bootload-utils.h**.

Enumeration Type Documentation

enum bootloadMode

Bootload modes supported by the bootload utility library.

Enumerator:

BOOTLOAD_MODE_NONE Used when we are not currently doing any bootloading.

BOOTLOAD_MODE_PASSTHRU Used when doing normal and recovery passthru bootload.

Definition at line **82** of file **bootload-utils.h**.

enum bootloadState

A bootload state is a value that an application can check to see if bootloading is in progress.

This is necessary because we want the application to be aware that bootloading is going on and it needs to limit its activities. For example, when passthru bootloading is going on, do not print anything to a serial port because it may violate the XModem protocol. Also, try to limit radio activities to a minimum to avoid any interruptions to bootload progress. Used in a bootload state machine.

Enumerator:

BOOTLOAD_STATE_NORMAL	Start state
BOOTLOAD_STATE_QUERY	After send query message
BOOTLOAD_STATE_WAIT_FOR_AUTH_CHALLENGE	Wait for authentication challenge
BOOTLOAD_STATE_WAIT_FOR_AUTH_RESPONSE	Wait for authentication response
BOOTLOAD_STATE_DELAY_BEFORE_START	Delay state before start new action
BOOTLOAD_STATE_START_UNICAST_BOOTLOAD	After start unicast bootloading
BOOTLOAD_STATE_START_BROADCAST_BOOTLOAD	After start broadcast bootloading
BOOTLOAD_STATE_START_SENDING_IMAGE	Need to start XMODEM code
BOOTLOAD_STATE_SENDING_IMAGE	During sending OTA data messages
BOOTLOAD_STATE_WAIT_FOR_IMAGE_ACK	Wait for OTA data ack
BOOTLOAD_STATE_WAIT_FOR_COMPLETE_ACK	Wait for OTA end transmission ack
BOOTLOAD_STATE_DONE	Finish bootloading

Definition at line **106** of file **bootload-utils.h**.

Function Documentation

```
void bootloadUtilInit ( int8u  appPort,
                        int8u  bootloadPort
                      )
```

Bootload library initialization.

The application needs to define the ports to be used for printing information and for a (passthru) bootload.

Note:
Generally it's a good idea to use different ports for the application and for bootloading because when doing passthru bootloading, we do not want to print any additional data that can cause an XModem transaction to fail.

Parameters:

appPort	Port used for printing information.
bootloadPort	Port used for passthru bootloading.

```
EmberStatus bootloadUtilSendRequest ( EmberEUI64    targetEui,
                                      int16u         mfgId,
                                      int8u          hardwareTag[BOOTLOAD_HARDWARE_TAG_SIZE],
                                      int8u          encryptKey[BOOTLOAD_AUTH_COMMON_SIZE],
                                      bootloadMode   mode
                                    )
```

Start the bootload process on a remote node that is currently running stack/application.

The source node sends a bootload request message to initiate the bootload authentication process. The source node then enters a state waiting for the target node to send an authentication challenge, which it will encrypt and send back as a response. MfgId and harwareTag information is sent over the air to the target node to verify whether to go into bootload mode. The encryption key is saved on the source node for later authentication. The mode indicates the bootload mode that the source will be using.

Parameters:

targetEui	Node to be bootloaded.
mfgId	Manufacturer ID (vendor specific).
hardwareTag	Hardware ID, such as a board (vendor specific).
encryptKey	Key used in the authentication process.
mode	Bootload mode to be used is passthru (0x01).

Returns:

EMBER_SUCESS if successful, or EMBER_NO_BUFFERS, or EMBER_ERR_FATAL if the function was called too soon after a previous call to it.

void bootloadUtilSendQuery (EmberEUI64 target)

A function to send query message to gather basic information about the node(s).

There are two types of query messages: broadcast and unicast. Broadcast query is generally used to gather information regarding a neighboring node, especially the eui64 of the node. Unicast query is used when we already know the eui64 of the target node that we needs information from.

Parameters:

target	The node we want to gather information from. If the value is NULL, that means we want to do a broadcast query.

void bootloadUtilStartBootload (EmberEUI64 target, bootloadMode mode)

Start the bootload process on a remote node that is already running in bootload mode.

This is generally to recover a node that failed during bootload. The failure can be caused by the source node resetting, the network being too busy, a software reset, and so on. However, the failure is not caused by a target node losing power. After the failure, the node stays in bootload mode on the same (current) channel.

Parameters:

target	remote node to be bootloaded. If the value is NULL, that means we do not know the eui64 of the target node. A broadcast (start bootload) packet is sent and the first node that replies will be bootloaded.
mode	bootload mode to be used, such as passthru (0x01).

void bootloadUtilTick (void)

A function in the application's heartbeat or tick function that contains basic bootloading state machine and also manages the bootload timer.

boolean bootloadUtilLaunchRequestHandler (int16u manufacturerId, int8u hardwareTag[BOOTLOAD_HARDWARE_TAG_SIZE], EmberEUI64 sourceEui)

A callback function invoked by bootload-utils when a bootload request message is received.

The application may choose whether or not to enter the bootloader by checking the manufacturerId, hardwareTag, and sourceEui. If the application chooses to launch the bootloader, the bootloader will launch after successful completion of the bootloader launch authentication protocol.

Parameters:

manufacturerId	The manufacturer specification (vendor specific) of the sending node.
hardwareTag	The hardware specification (vendor specific) of the sending node.
sourceEui	The EUI64 of the sending node.

Returns:

TRUE if the application wishes to launch the bootloader, FALSE if the application does not wish to launch the

bootloader.

```
void bootloadUtilQueryResponseHandler ( boolean      bootloaderActive,
                                        int16u       manufacturerId,
                                        int8u        hardwareTag[BOOTLOAD_HARDWARE_TAG_SIZE],
                                        EmberEUI64   targetEui,
                                        int8u        bootloaderCapabilities,
                                        int8u        platform,
                                        int8u        micro,
                                        int8u        phy,
                                        int16u       blVersion
                                      )
```

A callback function invoked by bootload-utils when a bootload query response message is received.

This is particularly useful when the application needs to decide which node to bootload. Several attributes of the responding node are provided to the application. The application can use these attributes to decide whether to bootload or how to bootload a given node.

Parameters:

bootloaderActive	TRUE if the responding node is running the bootloader, FALSE if not.
manufacturerId	The manufacturer specification (vendor specific) of the responding node.
hardwareTag	The hardware specification (vendor specific) of the responding node.
targetEui	The EUI64 of the responding node.
bootloaderCapabilities	If the lsb is 1, the bootloader on the responding node supports encrypted bootloader message payloads.
platform	The type of platform of the responding node. 1 is avr-atmega, 2 is xap2b.
micro	The type of microcontroller on the responding node. Value depends on platform. 1 is the avr-atmega 64, 2 is the avr-atmega 128, 1 is the xap2b em250.
phy	The type of phy of the responding node. 1 is em2420, 2 is em250.
blVersion	The version of standalone bootloader of the responding node. This is a 2 byte field. The high byte is the version and the low byte is the build. A value of 0xFFFF means unknown. For example, a version field of 0x1234 is version 1.2, build 34.

```
void bootloadUtilSendAuthResponse ( EmberEUI64 target )
```

A function called by a parent node to send an authentication response message to the sleepy or mobile end-device target node.

The message is sent as a Just-In-Time (JIT) message, hence, the end-device target needs to poll for the message.

The bootload utility library will call this function automatically if bootloading the router node.

Parameters:

target The end-device target node being bootloaded.

Variable Documentation

```
bootloadState blState
```


Command Interpreters
[Application Utilities API Reference]

Modules

- Command Interpreter 2

Command Interpreter 2
[Command Interpreters]

Data Structures

struct	**EmberCommandEntry** Command entry for a command table. More...

Defines

#define	**MAX_TOKEN_COUNT**
#define	**EMBER_COMMAND_INTERPRETER_CONFIGURATION_ECHO**
#define	**emberProcessCommandInput**(port)
#define	**emberCommandInterpreterEchoOn**()
#define	**emberCommandInterpreterEchoOff**()
#define	**emberCommandInterpreterIsEchoOn**()

Typedefs

typedef void(*	**CommandAction**)(void)

Enumerations

enum	**EmberCommandStatus** {
	EMBER_CMD_SUCCESS,
	EMBER_CMD_ERR_PORT_PROBLEM,
	EMBER_CMD_ERR_NO_SUCH_COMMAND,
	EMBER_CMD_ERR_WRONG_NUMBER_OF_ARGUMENTS,
	EMBER_CMD_ERR_ARGUMENT_OUT_OF_RANGE,
	EMBER_CMD_ERR_ARGUMENT_SYNTAX_ERROR,
	EMBER_CMD_ERR_STRING_TOO_LONG,
	EMBER_CMD_ERR_INVALID_ARGUMENT_TYPE
	}

Functions

void	**emberCommandErrorHandler** (**EmberCommandStatus** status)
void	**emberPrintCommandUsage** (**EmberCommandEntry** *entry)
void	**emberPrintCommandUsageNotes** (void)
void	**emberPrintCommandTable** (void)
void	**emberCommandReaderInit** (void)
boolean	**emberProcessCommandString** (**int8u** *input, **int8u** size)

Variables

EmberCommandEntry *	**emberCurrentCommand**
EmberCommandEntry	**emberCommandTable** []
int8u	**emberCommandInterpreter2Configuration**

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.

int32u	**emberUnsignedCommandArgument** (**int8u** index)
int16s	**emberSignedCommandArgument** (**int8u** index)
int8u *	**emberStringCommandArgument** (**int8s** index, **int8u** *length)
int8u	**emberCopyStringArgument** (**int8s** index, **int8u** *destination, **int8u** maxLength, **boolean** leftPad)
#define	**emberCopyKeyArgument**(index, keyDataPointer)
#define	**emberCopyEui64Argument**(index, eui64)

Command Table Settings

#define	**EMBER_MAX_COMMAND_ARGUMENTS**

#define **EMBER_COMMAND_BUFFER_LENGTH**

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)
{
  int8u channel = emberUnsignedCommandArgument(0);
  int16u panId  = emberUnsignedCommandArgument(1);
  int8s power   = emberSignedCommandArgument(2);
  int8u length;
  int8u *eui64  = emberStringCommandArgument(3, &length);
  ...
  ... call emberFormNetwork() etc
  ...
}

// The main command table.
EmberCommandEntry emberCommandTable[] = {
  { "network",  (CommandAction)networkCommands, "n",  "network commands" },
  { "status",   statusCommand,                  "",   "app status" },
  ...
  { NULL }
 };

// The table of network commands.
EmberCommandEntry networkCommands[] = {
  { "form",       formCommand, "uvsh" },
  { "join",       joinCommand, "uvsh" },
  ...
  { NULL }
 };

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.

Define Documentation

#define EMBER_MAX_COMMAND_ARGUMENTS

The maximum number of arguments a command can have. A nested command counts as an argument.

Definition at line **101** of file **command-interpreter2.h**.

#define EMBER_COMMAND_BUFFER_LENGTH

The maximum number of arguments a command can have. A nested command counts as an argument.

Definition at line **105** of file **command-interpreter2.h**.

#define MAX_TOKEN_COUNT

// END name group

Definition at line **112** of file **command-interpreter2.h**.

#define EMBER_COMMAND_INTERPRETER_CONFIGURATION_ECHO

Definition at line **180** of file **command-interpreter2.h**.

#define emberCopyKeyArgument (index, keyDataPointer)

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

Definition at line **248** of file **command-interpreter2.h**.

#define emberCopyEui64Argument (index, eui64)

A convenience macro for copying eui64 arguments to an EmberEUI64.

Definition at line **255** of file **command-interpreter2.h**.

#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(int8u port);
```

Definition at line **288** of file **command-interpreter2.h**.

#define emberCommandInterpreterEchoOn ( )

Turn echo of command line on.

Definition at line **293** of file **command-interpreter2.h**.

#define emberCommandInterpreterEchoOff ( )

Turn echo of command line off.

Definition at line 299 of file command-interpreter2.h.

#define emberCommandInterpreterIsEchoOn ( )

Returns true if echo is on, false otherwise.

Definition at line 305 of file command-interpreter2.h.

Typedef Documentation

typedef void(* CommandAction)(void)

Definition at line 114 of file command-interpreter2.h.

Enumeration Type Documentation

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 188 of file command-interpreter2.h.

Function Documentation

int32u emberUnsignedCommandArgument (int8u index)

Retrieves unsigned integer arguments.

int16s emberSignedCommandArgument (int8u index)

Retrieves signed integer arguments.

int8u* emberStringCommandArgument (int8s index,
int8u * 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 index of -1. For example, to retrieve the first character of the command, do: int8u 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.

```
int8u emberCopyStringArgument ( int8s     index,
                                int8u *   destination,
                                int8u     maxLength,
                                boolean   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.

```
void emberCommandErrorHandler ( EmberCommandStatus status )
```

// END name group 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()**.

```
void emberPrintCommandUsage ( EmberCommandEntry * entry )
```

```
void emberPrintCommandUsageNotes ( void )
```

```
void emberPrintCommandTable ( void )
```

```
void emberCommandReaderInit ( void )
```

Initialize the command interpreter.

```
boolean emberProcessCommandString ( int8u *  input,
                                    int8u    size
                                  )
```

Process the given string as a command.

Variable Documentation

```
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.

```
EmberCommandEntry emberCommandTable[]
```

```
int8u emberCommandInterpreter2Configuration
```

Configuration byte.

ZigBee Device Object (ZDO) Information
[Application Utilities API Reference]

Defines

#define **ZDO_MESSAGE_OVERHEAD**

Device Discovery Functions

EmberStatus **emberNetworkAddressRequest** (**EmberEUI64** target, **boolean** reportKids, **int8u** childStartIndex)

EmberStatus **emberIeeeAddressRequest** (**EmberNodeId** target, **boolean** reportKids, **int8u** childStartIndex, **EmberApsOption** options)

Service Discovery Functions

EmberStatus **ezspMatchDescriptorsRequest** (**EmberNodeId** target, **int16u** profile, **int8u** inCount, **int8u** outCount, **int16u** *inClusters, **int16u** *outClusters, **EmberApsOption** options)

Binding Manager Functions

EmberStatus **ezspEndDeviceBindRequest** (**EmberNodeId** localNodeId, **EmberEUI64** localEui64, **int8u** endpoint, **int16u** profile, **int8u** inCount, **int8u** outCount, **int16u** *inClusters, **int16u** *outClusters, **EmberApsOption** options)

Function to Decode Address Response Messages

EmberNodeId **ezspDecodeAddressResponse** (**int8u** *response, **EmberEUI64** eui64Return)

Service Discovery Functions

EmberStatus **emberNodeDescriptorRequest** (**EmberNodeId** target, **EmberApsOption** options)

EmberStatus **emberPowerDescriptorRequest** (**EmberNodeId** target, **EmberApsOption** options)

EmberStatus **emberSimpleDescriptorRequest** (**EmberNodeId** target, **int8u** targetEndpoint, **EmberApsOption** options)

EmberStatus **emberActiveEndpointsRequest** (**EmberNodeId** target, **EmberApsOption** options)

Binding Manager Functions

EmberStatus **emberBindRequest** (**EmberNodeId** target, **EmberEUI64** source, **int8u** sourceEndpoint, **int16u** clusterId, **int8u** type, **EmberEUI64** destination, **EmberMulticastId** groupAddress, **int8u** destinationEndpoint, **EmberApsOption** options)

EmberStatus **emberUnbindRequest** (**EmberNodeId** target, **EmberEUI64** source, **int8u** sourceEndpoint, **int16u** clusterId, **int8u** type, **EmberEUI64** destination, **EmberMulticastId** groupAddress, **int8u** destinationEndpoint, **EmberApsOption** options)

Node Manager Functions

EmberStatus **emberLqiTableRequest** (**EmberNodeId** target, **int8u** startIndex, **EmberApsOption** options)

EmberStatus **emberRoutingTableRequest** (**EmberNodeId** target, **int8u** startIndex, **EmberApsOption** options)

EmberStatus **emberBindingTableRequest** (**EmberNodeId** target, **int8u** startIndex, **EmberApsOption** options)

EmberStatus **emberLeaveRequest** (**EmberNodeId** target, **EmberEUI64** deviceAddress, **int8u** leaveRequestFlags, **EmberApsOption** options)

EmberStatus	**emberPermitJoiningRequest** (**EmberNodeId** target, **int8u** duration, **int8u** authentication, **EmberApsOption** options)
void	**emberSetZigDevRequestRadius** (**int8u** radius)
int8u	**emberGetZigDevRequestRadius** (void)
int8u	**emberGetLastZigDevRequestSequence** (void)

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);
  int8u 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,
                                int8u lastHopLqi,
                                int8s lastHopRssi,
                                EmberNodeId sender,
                                int8u bindingIndex,
                                int8u addressIndex,
                                int8u messageLength,
                                int8u *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.
  }
}
```

Define Documentation

#define ZDO_MESSAGE_OVERHEAD

ZDO messages start with a sequence number.

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

Function Documentation

```
EmberStatus emberNetworkAddressRequest ( EmberEUI64 target,
                                         boolean    reportKids,
                                         int8u      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.

```
EmberStatus emberIeeeAddressRequest ( EmberNodeId     target,
                                      boolean         reportKids,
                                      int8u           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.
childStartIndex	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

```
EmberStatus ezspMatchDescriptorsRequest ( EmberNodeId     target,
                                          int16u          profile,
                                          int8u           inCount,
                                          int8u           outCount,
                                          int16u *        inClusters,
                                          int16u *        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 emberSendUnicast() for a description. This parameter is ignored if the target is a broadcast address.

Returns:

An EmberStatus value. EMBER_SUCCESS, EMBER_NO_BUFFERS, EMBER_NETWORK_DOWN or EMBER_NETWORK_BUSY.

```
EmberStatus ezspEndDeviceBindRequest ( EmberNodeId     localNodeId,
                                       EmberEUI64      localEui64,
                                       int8u           endpoint,
                                       int16u          profile,
                                       int8u           inCount,
                                       int8u           outCount,
                                       int16u *        inClusters,
                                       int16u *        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_NETWORK_DOWN or EMBER_NETWORK_BUSY.

```
EmberNodeId ezspDecodeAddressResponse ( int8u *       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_RESPONSE 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_SUCCESS. Otherwise, returns EMBER_NULL_NODE_ID.

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_NETWORK_DOWN** or **EMBER_NETWORK_BUSY**.

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_NETWORK_DOWN** or **EMBER_NETWORK_BUSY**.

EmberStatus emberSimpleDescriptorRequest (EmberNodeId target, int8u 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.

targetEndpoint 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_NETWORK_DOWN** or **EMBER_NETWORK_BUSY**.

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_NETWORK_DOWN** or **EMBER_NETWORK_BUSY**.

```
EmberStatus emberBindRequest ( EmberNodeId      target,
                               EmberEUI64       source,
                               int8u            sourceEndpoint,
                               int16u           clusterId,
                               int8u            type,
                               EmberEUI64       destination,
                               EmberMulticastId groupAddress,
                               int8u            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.
sourceEndpoint	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 **UNICAST_BINDING** or **UNICAST_MANY_TO_ONE_BINDING**.
groupAddress	The group address for the **MULTICAST_BINDING**.
destinationEndpoint	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** or **EMBER_NETWORK_BUSY**.

```
EmberStatus emberUnbindRequest ( EmberNodeId      target,
                                 EmberEUI64       source,
                                 int8u            sourceEndpoint,
                                 int16u           clusterId,
                                 int8u            type,
                                 EmberEUI64       destination,
                                 EmberMulticastId groupAddress,
                                 int8u            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.
sourceEndpoint	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**.
groupAddress	The group address for the **MULTICAST_BINDING**.
destinationEndpoint	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**

EmberStatus emberLqiTableRequest (**EmberNodeId** **target,**
int8u **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_NETWORK_DOWN** or **EMBER_NETWORK_BUSY**.

EmberStatus emberRoutingTableRequest (**EmberNodeId** **target,**
int8u **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_NETWORK_DOWN** or **EMBER_NETWORK_BUSY**.

EmberStatus emberBindingTableRequest (**EmberNodeId** **target,**
int8u **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_NETWORK_DOWN** or **EMBER_NETWORK_BUSY**.

EmberStatus emberLeaveRequest (EmberNodeId target, EmberEUI64 deviceAddress, int8u 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.
deviceAddress 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.
leaveRequestFlags A bitmask of leave options. Include **LEAVE_REQUEST_REMOVE_CHILDREN_FLAG** if the target is to remove their children and/or **LEAVE_REQUEST_REJOIN_FLAG** if the target is to rejoin the network immediately after leaving.
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** or **EMBER_NETWORK_BUSY**.

EmberStatus emberPermitJoiningRequest (EmberNodeId target, int8u duration, int8u 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.
authentication 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_NETWORK_DOWN** or **EMBER_NETWORK_BUSY**.

---

void emberSetZigDevRequestRadius (int8u radius)

Change the default radius for broadcast ZDO requests.

Parameters:

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

---

int8u emberGetZigDevRequestRadius (void)

Retrieve the default radius for broadcast ZDO requests.

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

int8u emberGetLastZigDevRequestSequence (void)

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

Returns:
Last ZDO transaction sequence number used

Message Fragmentation
[Application Utilities API Reference]

Initialization

void	**ezspFragmentInit** (**int16u** receiveBufferLength, **int8u** *receiveBuffer)

Transmitting

EmberStatus	**ezspFragmentSendUnicast** (**EmberOutgoingMessageType** type, **int16u** indexOrDestination, **EmberApsFrame** *apsFrame, **int8u** maxFragmentSize, **int16u** messageLength, **int8u** *messageContents)
EmberStatus	**ezspFragmentSourceRouteHandler** (void)
boolean	**ezspFragmentMessageSent** (**EmberApsFrame** *apsFrame, **EmberStatus** status)
void	**ezspFragmentMessageSentHandler** (**EmberStatus** status)

Receiving

boolean	**ezspFragmentIncomingMessage** (**EmberApsFrame** *apsFrame, **EmberNodeId** sender, **int16u** *messageLength, **int8u** **messageContents)
void	**ezspFragmentTick** (void)

Detailed Description

Fragmented message support for EZSP Hosts. Splits long messages into smaller blocks for transmission and reassembles received blocks. See fragment-host.c for source code.

EZSP_CONFIG_FRAGMENT_WINDOW_SIZE controls how many blocks are sent at a time.
EZSP_CONFIG_FRAGMENT_DELAY_MS controls the spacing between blocks.

Before calling any of the other functions listed here, the application must call **ezspFragmentInit()**.

To send a long message, the application calls **ezspFragmentSendUnicast()**. The application must add a call to **ezspFragmentMessageSent()** at the start of its ezspMessageSentHandler(). If **ezspFragmentMessageSent()** returns TRUE, the fragmentation code has handled the event and the application must not process it further. The fragmentation code calls the application-defined **ezspFragmentMessageSentHandler()** when it has finished sending the long message.

To receive a long message, the application must add a call to **ezspFragmentIncomingMessage()** at the start of its ezspIncomingMessageHandler(). If **ezspFragmentIncomingMessage()** returns TRUE, the fragmentation code has handled the message and the application must not process it further. The application must also call **ezspFragmentTick()** regularly.

Function Documentation

void ezspFragmentInit (int16u receiveBufferLength,
int8u * 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_FRAGMENT_WINDOW_SIZE. The application must set these values before calling this function.

Parameters:

receiveBufferLength	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 **ezspFragmentIncomingMessage()**.

EmberStatus ezspFragmentSendUnicast (EmberOutgoingMessageType type, int16u indexOrDestination, EmberApsFrame * apsFrame, int8u maxFragmentSize, int16u messageLength, int8u * 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_OUTGOING_DIRECT**, **EMBER_OUTGOING_VIA_ADDRESS_TABLE**, or **EMBER_OUTGOING_VIA_BINDING**.
indexOrDestination	Depending on the type of addressing used, this is either the EmberNodeId of the destination, an index into the address table, or an index into the binding table.
apsFrame	The APS frame for the message.
maxFragmentSize	The message will be broken into blocks no larger than this.
messageLength	The length of the messageContents parameter in bytes.
messageContents	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.

EmberStatus ezspFragmentSourceRouteHandler (void)

A callback invoked just before each block of the current long message is sent. If the message is to be source routed, the application must define this callback and call ezspSetSourceRoute() in it.

The application must define EZSP_APPLICATION_HAS_FRAGMENT_SOURCE_ROUTE_HANDLER in its configuration header if it defines this callback.

Returns:

EMBER_SUCCESS if the source route has been set. Any other value will abort transmission of the current long message.

boolean 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.

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**.

```
boolean ezspFragmentIncomingMessage ( EmberApsFrame *  apsFrame,
                                      EmberNodeId      sender,
                                      int16u *         messageLength,
                                      int8u **         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().
messageContents	A pointer to the message contents passed to ezspIncomingMessageHandler().

Returns:

TRUE if the incoming message was a block of an incomplete long message. The fragmentation code has handled the message so the application must return immediately from its ezspIncomingMessageHandler(). Returns FALSE if the incoming message was not part of a long message. The fragmentation code has not handled the message so the application must continue to process it. Returns FALSE if the incoming message was a block that completed a long message. The fragmentation code replaces the message with the reassembled long message so the application must continue to process it.

```
void ezspFragmentTick ( void )
```

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

Network Manager
[Application Utilities API Reference]

Defines

#define	**NM_WARNING_LIMIT**
#define	**NM_WINDOW_SIZE**
#define	**NM_CHANNEL_MASK**
#define	**NM_WATCHLIST_SIZE**

Functions

void	**nmUtilWarningHandler** (void)
boolean	**nmUtilProcessIncoming** (**EmberApsFrame** *apsFrame, **int8u** messageLength, **int8u** *message)
EmberStatus	**nmUtilChangeChannelRequest** (void)

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.

Define Documentation

#define NM_WARNING_LIMIT

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

#define NM_WINDOW_SIZE

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

#define NM_CHANNEL_MASK

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

#define NM_WATCHLIST_SIZE

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

Function Documentation

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.

boolean nmUtilProcessIncoming (EmberApsFrame * apsFrame, int8u messageLength, int8u * message)

Called from the app in emberIncomingMessageHandler. Returns TRUE if and only if the library processed the message.

Parameters:

apsFrame
messageLength
message

EmberStatus nmUtilChangeChannelRequest (void)

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

Serial Communication
[Application Utilities API Reference]

Defines

#define	**emberSerialWriteUsed**(port)

Functions

EmberStatus	**emberSerialInit** (**int8u** port, **SerialBaudRate** rate, SerialParity parity, **int8u** stopBits)
int16u	**emberSerialReadAvailable** (**int8u** port)
EmberStatus	**emberSerialReadByte** (**int8u** port, **int8u** *dataByte)
EmberStatus	**emberSerialReadLine** (**int8u** port, char *data, **int8u** max)
EmberStatus	**emberSerialReadPartialLine** (**int8u** port, char *data, **int8u** max, **int8u** *index)
int16u	**emberSerialWriteAvailable** (**int8u** port)
EmberStatus	**emberSerialWriteByte** (**int8u** port, **int8u** dataByte)
EmberStatus	**emberSerialWriteHex** (**int8u** port, **int8u** dataByte)
EmberStatus	**emberSerialWriteString** (**int8u** port, PGM_P string)
XAP2B_PAGEZERO_ON **EmberStatus**	**emberSerialPrintf** (**int8u** port, PGM_P formatString,...)
XAP2B_PAGEZERO_OFF XAP2B_PAGEZERO_ON **EmberStatus**	**emberSerialPrintfLine** (**int8u** port, PGM_P formatString,...)
XAP2B_PAGEZERO_OFF XAP2B_PAGEZERO_ON **EmberStatus**	**emberSerialPrintCarriageReturn** (**int8u** port)
XAP2B_PAGEZERO_OFF **EmberStatus**	**emberSerialPrintfVarArg** (**int8u** port, PGM_P formatString, va_list ap)
EmberStatus	**emberSerialWriteData** (**int8u** port, **int8u** *data, **int8u** length)
XAP2B_PAGEZERO_ON **EmberStatus**	**emberSerialWaitSend** (**int8u** port)
XAP2B_PAGEZERO_OFF **EmberStatus**	**emberSerialGuaranteedPrintf** (**int8u** port, PGM_P formatString,...)
void	**emberSerialBufferTick** (void)
void	**emberSerialFlushRx** (**int8u** port)

Printf Prototypes

These prototypes are for the internal printf implementation, in case it is desired to use it elsewhere. See the code for **emberSerialPrintf()** for an example of printf usage.

typedef **EmberStatus**(	**emPrintfFlushHandler**)(**int8u** flushVar, **int8u** *contents, **int8u** length)
int8u	**emPrintfInternal** (**emPrintfFlushHandler handler**, **int8u** port, PGM_P buff, va_list list)

Detailed Description

Unless otherwise noted, the EmberNet stack does not use these functions, and therefore the HAL is not required to implement them. However, many of the supplied example applications do use them. On some platforms, they are also required by DEBUG builds of the stack

Many of these functions return an **EmberStatus** value. See stack/include/error-defs.h for definitions of all **EmberStatus** return values. See **app/util/serial/serial.h** for source code. To use these serial routines, they must be properly configured.

If the Ember serial library is built using EMBER_SERIAL_USE_STDIO, then the Ember serial code will redirect to stdio.h. EMBER_SERIAL_USE_STDIO will not consume any of the usual Ember serial library buffers and does not require use of any of the other EMBER_SERIALx definitions described here. In this mode, the only required lower layers are:

- putchar()
- getchar()
- fflush(stdout)
- **halInternalUartInit()**
- **halInternalPrintfWriteAvailable()**
- **halInternalPrintfReadAvailable()**
- **halInternalForcePrintf()**

The functions can work in two ways, depending on how messages waiting for transmission are stored:

- Buffered mode: Uses stack linked buffers. This method can be more efficient if many messages received over the air also need to be transmitted over the serial interface.
- FIFO mode: Uses a statically allocated queue of bytes, and data to be transmitted is copied into the queue.

(These modes deal only with data transmission. Data **reception** always occurs in a FIFO mode.)

The current version of these sources provides support for as many as two serial ports, but it can be easily extended. The ports are numbered 0 and 1 and should be accessed using those numbers. The ports can be set up independently of each other.

To enable a port, a Use mode (buffered or FIFO) and a Queue Size must be declared on the port. In FIFO mode, the Queue Size is the size of the FIFO and represents the number of bytes that can be waiting for transmission at any given time. In buffered mode, the Queue Size represents the number of whole messages that can be waiting for transmission at any given time. A single message is created for each call to any of the serial APIs.

To specify a Use mode and Queue Size, place declarations in the compiler preprocessor options when building your application:

- **Use Mode:**
  - EMBER_SERIAL0_MODE=EMBER_SERIAL_BUFFER or EMBER_SERIAL_FIFO
  - EMBER_SERIAL1_MODE=EMBER_SERIAL_BUFFER or EMBER_SERIAL_FIFO
- **Queue Size:**
  - EMBER_SERIAL0_TX_QUEUE_SIZE=2
  - EMBER_SERIAL0_RX_QUEUE_SIZE=4
  - EMBER_SERIAL1_TX_QUEUE_SIZE=8
  - EMBER_SERIAL1_RX_QUEUE_SIZE=16

Note the following:

- If buffered mode is declared, **emberSerialBufferTick()** should be called in the application's main event loop.
- If buffered mode is declared, the Tx queue size **MUST** be <= 255
- On the AVR platform, Rx & Tx queue sizes are limited to powers of 2 <= 128
- By default, both ports are unused.

You can also use declarations to specify what should be done if an attempt is made to send more data than the queue can accommodate:

- EMBER_SERIAL0_BLOCKING
- EMBER_SERIAL1_BLOCKING

Be aware that since blocking spins in a loop, doing nothing until space is available, it can adversely affect any code that has tight timing requirements.

If EMBER_SERIAL0_BLOCKING or EMBER_SERIAL1_BLOCKING is defined, then the call to the port will block until space is available, guaranteeing that the entire message is sent. Note that in buffered mode, even if blocking mode is in effect entire messages may be dropped if insufficient stack buffers are available to hold them. When this happens, **EMBER_NO_BUFFERS** is returned.

If no blocking mode is defined, the serial code defaults to non-blocking mode. In this event, when the queue is too short, the data that don't fit are dropped. In FIFO mode, this may result bytes being dropped, starting in the middle of message. In buffered mode, the entire message is dropped. When data is dropped, EMBER_SERIALTX_OVERFLOW is returned.

To minimize code size, very little error checking is done on the given parameters. Specifying an invalid or unused serial port may result in unexplained behavior. In some cases **EMBER_ERR_FATAL** may be returned.

Define Documentation

#define emberSerialWriteUsed (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.

Definition at line **227** of file **app/util/serial/serial.h**.

Typedef Documentation

typedef EmberStatus(emPrintfFlushHandler)(int8u flushVar, int8u *contents, int8u length)

Typedefine to cast a function into the appropriate format to be used inside the `emPrintfInternal` function below, for performing the actual flushing of a formatted string to a destination such as a serial port.

Parameters:

flushVar,: The destination of the flush, most commonly a serial port number (0 or 1).
contents A pointer to the string to flush.
length The number of bytes to flush.

Returns:

The EmberStatus value of the typedefined function.

Definition at line **466** of file **app/util/serial/serial.h**.

Function Documentation

EmberStatus emberSerialInit (int8u port, SerialBaudRate rate, SerialParity parity, int8u 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**.

int16u emberSerialReadAvailable (int8u 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.

EmberStatus emberSerialReadByte (int8u port, int8u * 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

EmberStatus emberSerialReadLine (int8u port, char * data, int8u 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

EmberStatus emberSerialReadPartialLine (int8u port, char * data, int8u max, int8u * 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.

int16u emberSerialWriteAvailable (int8u 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.

EmberStatus emberSerialWriteByte (int8u port, int8u 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**.

EmberStatus emberSerialWriteHex (int8u port, int8u 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**.

EmberStatus emberSerialWriteString (int8u 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**.

XAP2B_PAGEZERO_ON EmberStatus emberSerialPrintf (int8u 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**.

XAP2B_PAGEZERO_OFF XAP2B_PAGEZERO_ON EmberStatus emberSerialPrintfLine (int8u 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**.

XAP2B_PAGEZERO_OFF XAP2B_PAGEZERO_ON EmberStatus emberSerialPrintCarriageReturn (int8u 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**.

XAP2B_PAGEZERO_OFF EmberStatus emberSerialPrintfVarArg (int8u 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**.

EmberStatus emberSerialWriteData (int8u port, int8u * data, int8u 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**.

XAP2B_PAGEZERO_ON EmberStatus emberSerialWaitSend (int8u 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**.

XAP2B_PAGEZERO_OFF EmberStatus emberSerialGuaranteedPrintf (int8u 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.

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.

void emberSerialFlushRx (int8u port)

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

Parameters:

port A serial port number (0 or 1).

int8u emPrintfInternal (emPrintfFlushHandler handler,
int8u port,
PGM_P buff,
va_list list)

The internal printf function, which scans the string for the format specifiers and appropriately implants the passed data into the string.

Parameters:

handler,: The name of an internal function, which has parameters matching the function `emPrintfFlushHandler` above, responsible for flushing a string formatted by this function, `emPrintfInternal`, to the appropriate buffer or function that performs the actual transmission.

port The destination of the flush performed above, most commonly serial port number (0 or 1).

buff The string to print.

list The list of arguments for the format specifiers.

Returns:

The number of characters written.

Deprecated Files

form-and-join3_2.h

EmberAesMmoHashContext Struct Reference
[Ember Common Data Types]

This data structure contains the context data when calculating an AES MMO hash (message digest). More...

`#include <ember-types.h>`

Data Fields

int8u	result [EMBER_AES_HASH_BLOCK_SIZE]
int32u	length

Detailed Description

This data structure contains the context data when calculating an AES MMO hash (message digest).

Definition at line 1264 of file ember-types.h.

Field Documentation

int8u EmberAesMmoHashContext::result[EMBER_AES_HASH_BLOCK_SIZE]

Definition at line 1265 of file ember-types.h.

int32u EmberAesMmoHashContext::length

Definition at line 1266 of file ember-types.h.

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

- ember-types.h

EmberApsFrame Struct Reference
[Ember Common Data Types]

An in-memory representation of a ZigBee APS frame of an incoming or outgoing message. More...

`#include <ember-types.h>`

Data Fields

int16u	profileId
int16u	clusterId
int8u	sourceEndpoint
int8u	destinationEndpoint
EmberApsOption	options
int16u	groupId
int8u	sequence

Detailed Description

An in-memory representation of a ZigBee APS frame of an incoming or outgoing message.

Definition at line 707 of file ember-types.h.

Field Documentation

int16u EmberApsFrame::profileId

The application profile ID that describes the format of the message.

Definition at line 709 of file ember-types.h.

int16u EmberApsFrame::clusterId

The cluster ID for this message.

Definition at line 711 of file ember-types.h.

int8u EmberApsFrame::sourceEndpoint

The source endpoint.

Definition at line 713 of file ember-types.h.

int8u EmberApsFrame::destinationEndpoint

The destination endpoint.

Definition at line 715 of file ember-types.h.

EmberApsOption EmberApsFrame::options

A bitmask of options from the enumeration above.

Definition at line 717 of file ember-types.h.

int16u EmberApsFrame::groupId

The group ID for this message, if it is multicast mode.

Definition at line **719** of file **ember-types.h**.

int8u EmberApsFrame::sequence

The sequence number.

Definition at line **721** of file **ember-types.h**.

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

- **ember-types.h**

EmberBindingTableEntry Struct Reference
[Ember Common Data Types]

Defines an entry in the binding table. More...

`#include <ember-types.h>`

Data Fields

EmberBindingType	type
int8u	local
int16u	clusterId
int8u	remote
EmberEUI64	identifier

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 731 of file ember-types.h.

Field Documentation

EmberBindingType EmberBindingTableEntry::type

The type of binding.

Definition at line 733 of file ember-types.h.

int8u EmberBindingTableEntry::local

The endpoint on the local node.

Definition at line 735 of file ember-types.h.

int16u 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 743 of file ember-types.h.

int8u EmberBindingTableEntry::remote

The endpoint on the remote node (specified by `identifier`).

Definition at line 745 of file ember-types.h.

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 750 of file ember-types.h.

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

- **ember-types.h**

EmberCertificateData Struct Reference
[Ember Common Data Types]

This data structure contains the certificate data that is used for Certificate Based Key Exchange (CBKE). More...

`#include <ember-types.h>`

Data Fields

int8u	**contents** [EMBER_CERTIFICATE_SIZE]

Detailed Description

This data structure contains the certificate data that is used for Certificate Based Key Exchange (CBKE).

Definition at line **1225** of file **ember-types.h**.

Field Documentation

int8u EmberCertificateData::contents[EMBER_CERTIFICATE_SIZE]

Definition at line **1227** of file **ember-types.h**.

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

- **ember-types.h**

EmberCommandEntry Struct Reference
[Command Interpreter 2]

Command entry for a command table. More...

`#include <command-interpreter2.h>`

Data Fields

PGM_P	**name**
CommandAction	**action**
PGM_P	**argumentTypes**
PGM_P	**description**

Detailed Description

Command entry for a command table.

Definition at line **119** of file **command-interpreter2.h**.

Field Documentation

PGM_P EmberCommandEntry::name

Use letters, digits, and underscores, '_', for the command name. Command names are case-sensitive.

Definition at line **126** of file **command-interpreter2.h**.

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 **132** of file **command-interpreter2.h**.

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
- 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 **159** of file **command-interpreter2.h**.

PGM_P EmberCommandEntry::description

A description of the command.

Definition at line **162** of file **command-interpreter2.h**.

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

- **command-interpreter2.h**

EmberCurrentSecurityState Struct Reference
[Ember Common Data Types]

This describes the security features used by the stack for a joined device. More...

`#include <ember-types.h>`

Data Fields

EmberCurrentSecurityBitmask	bitmask
EmberEUI64	trustCenterLongAddress

Detailed Description

This describes the security features used by the stack for a joined device.

Definition at line 1475 of file ember-types.h.

Field Documentation

EmberCurrentSecurityBitmask EmberCurrentSecurityState::bitmask

This bitmask indicates the security features currently in use on this node.

Definition at line 1478 of file ember-types.h.

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 1482 of file ember-types.h.

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

- ember-types.h

EmberEventControl Struct Reference
[Ember Common Data Types]

Control structure for events. More...

`#include <ember-types.h>`

Data Fields

EmberEventUnits	status
EmberTaskId	taskid
int32u	timeToExecute

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 **991** of file **ember-types.h**.

Field Documentation

EmberEventUnits EmberEventControl::status

The event's status, either inactive or the units for timeToExecute.

Definition at line **993** of file **ember-types.h**.

EmberTaskId EmberEventControl::taskid

The id of the task this event belongs to.

Definition at line **995** of file **ember-types.h**.

int32u EmberEventControl::timeToExecute

How long before the event fires. Units are always in milliseconds

Definition at line **999** of file **ember-types.h**.

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

- **ember-types.h**

EmberInitialSecurityState Struct Reference
[Ember Common Data Types]

This describes the Initial Security features and requirements that will be used when forming or joining the network. More...

#include <ember-types.h>

Data Fields

int16u	bitmask
EmberKeyData	preconfiguredKey
EmberKeyData	networkKey
int8u	networkKeySequenceNumber
EmberEUI64	preconfiguredTrustCenterEui64

Detailed Description

This describes the Initial Security features and requirements that will be used when forming or joining the network.

Definition at line 1395 of file ember-types.h.

Field Documentation

int16u 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 1400 of file ember-types.h.

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 EmberInitialSecurityState::bitmask sets the bit indicating EMBER_HAVE_PRECONFIGURED_KEY

Definition at line 1409 of file ember-types.h.

EmberKeyData EmberInitialSecurityState::networkKey

This is the Network Key used when initially forming the network. This must be set on the Trust Center. It is not needed for devices joining the network. This parameter will only be used if the EmberInitialSecurityState::bitmask sets the bit indicating EMBER_HAVE_NETWORK_KEY.

Definition at line 1415 of file ember-types.h.

int8u 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 1422 of file ember-types.h.

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 **1430** of file **ember-types.h**.

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

- **ember-types.h**

EmberKeyData Struct Reference
[Ember Common Data Types]

This data structure contains the key data that is passed into various other functions. More...

`#include <ember-types.h>`

Data Fields

int8u contents [EMBER_ENCRYPTION_KEY_SIZE]

Detailed Description

This data structure contains the key data that is passed into various other functions.

Definition at line **1218** of file **ember-types.h**.

Field Documentation

int8u EmberKeyData::contents[EMBER_ENCRYPTION_KEY_SIZE]

This is the key byte data.

Definition at line **1220** of file **ember-types.h**.

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

- **ember-types.h**

EmberKeyStruct Struct Reference
[Ember Common Data Types]

This describes a one of several different types of keys and its associated data. More...

`#include <ember-types.h>`

Data Fields

EmberKeyStructBitmask	bitmask
EmberKeyType	type
EmberKeyData	key
int32u	outgoingFrameCounter
int32u	incomingFrameCounter
int8u	sequenceNumber
EmberEUI64	partnerEUI64

Detailed Description

This describes a one of several different types of keys and its associated data.

Definition at line **1548** of file **ember-types.h**.

Field Documentation

EmberKeyStructBitmask EmberKeyStruct::bitmask

This bitmask indicates whether various fields in the structure contain valid data.

Definition at line **1551** of file **ember-types.h**.

EmberKeyType EmberKeyStruct::type

This indicates the type of the security key.

Definition at line **1553** of file **ember-types.h**.

EmberKeyData EmberKeyStruct::key

This is the actual key data.

Definition at line **1555** of file **ember-types.h**.

int32u EmberKeyStruct::outgoingFrameCounter

This is the outgoing frame counter associated with the key. It will contain valid data based on the **EmberKeyStructBitmask**.

Definition at line **1558** of file **ember-types.h**.

int32u EmberKeyStruct::incomingFrameCounter

This is the incoming frame counter associated with the key. It will contain valid data based on the **EmberKeyStructBitmask**.

Definition at line **1561** of file **ember-types.h**.

int8u EmberKeyStruct::sequenceNumber

This is the sequence number associated with the key. It will contain valid data based on the **EmberKeyStructBitmask**.

Definition at line **1564** of file **ember-types.h**.

EmberEUI64 EmberKeyStruct::partnerEUI64

This is the Partner EUI64 associated with the key. It will contain valid data based on the **EmberKeyStructBitmask**.

Definition at line **1567** of file **ember-types.h**.

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

- **ember-types.h**

EmberMacFilterMatchStruct Struct Reference
[Ember Common Data Types]

This structure indicates a matching raw MAC message has been received by the application configured MAC filters. More...

`#include <ember-types.h>`

Data Fields

int8u	filterIndexMatch
EmberMacPassthroughType	legacyPassthroughType
EmberMessageBuffer	message

Detailed Description

This structure indicates a matching raw MAC message has been received by the application configured MAC filters.

Definition at line 1763 of file ember-types.h.

Field Documentation

int8u EmberMacFilterMatchStruct::filterIndexMatch

Definition at line 1764 of file ember-types.h.

EmberMacPassthroughType EmberMacFilterMatchStruct::legacyPassthroughType

Definition at line 1765 of file ember-types.h.

EmberMessageBuffer EmberMacFilterMatchStruct::message

Definition at line 1766 of file ember-types.h.

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

- ember-types.h

EmberMessageDigest Struct Reference
[Ember Common Data Types]

This data structure contains an AES-MMO Hash (the message digest). More...

`#include <ember-types.h>`

Data Fields

int8u	**contents** [EMBER_AES_HASH_BLOCK_SIZE]

Detailed Description

This data structure contains an AES-MMO Hash (the message digest).

Definition at line **1257** of file **ember-types.h**.

Field Documentation

int8u EmberMessageDigest::contents[EMBER_AES_HASH_BLOCK_SIZE]

Definition at line **1258** of file **ember-types.h**.

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

- **ember-types.h**

EmberMulticastTableEntry Struct Reference
[Ember Common Data Types]

Defines an entry in the multicast table. More...

`#include <ember-types.h>`

Data Fields

EmberMulticastId	multicastId
int8u	endpoint

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 **818** of file **ember-types.h**.

Field Documentation

EmberMulticastId EmberMulticastTableEntry::multicastId

The multicast group ID.

Definition at line **820** of file **ember-types.h**.

int8u 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 **824** of file **ember-types.h**.

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

- **ember-types.h**

EmberNeighborTableEntry Struct Reference
[Ember Common Data Types]

Defines an entry in the neighbor table. More...

#include <ember-types.h>

Data Fields

int16u	shortId
int8u	averageLqi
int8u	inCost
int8u	outCost
int8u	age
EmberEUI64	longId

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 759 of file ember-types.h.

Field Documentation

int16u EmberNeighborTableEntry::shortId

The neighbor's two byte network id.

Definition at line 761 of file ember-types.h.

int8u 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 764 of file ember-types.h.

int8u 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 767 of file ember-types.h.

int8u 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 774 of file ember-types.h.

int8u 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 780 of file ember-types.h.

EmberEUI64 EmberNeighborTableEntry::longId

The 8 byte EUI64 of the neighbor.

Definition at line **782** of file **ember-types.h**.

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

- **ember-types.h**

EmberNetworkParameters Struct Reference
[Ember Common Data Types]

Holds network parameters. More...

#include <ember-types.h>

Data Fields

int8u	extendedPanId [8]
int16u	panId
int8s	radioTxPower
int8u	radioChannel
EmberJoinMethod	joinMethod
EmberNodeId	nwkManagerId
int8u	nwkUpdateId
int32u	channels

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 662 of file ember-types.h.

Field Documentation

int8u EmberNetworkParameters::extendedPanId[8]

The network's extended PAN identifier.

Definition at line 664 of file ember-types.h.

int16u EmberNetworkParameters::panId

The network's PAN identifier.

Definition at line 666 of file ember-types.h.

int8s EmberNetworkParameters::radioTxPower

A power setting, in dBm.

Definition at line 668 of file ember-types.h.

int8u EmberNetworkParameters::radioChannel

A radio channel. Be sure to specify a channel supported by the radio.

Definition at line 670 of file ember-types.h.

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 675 of file ember-types.h.

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 **681** of file **ember-types.h**.

int8u 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 **687** of file **ember-types.h**.

int32u 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 **693** of file **ember-types.h**.

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

- **ember-types.h**

EmberPrivateKeyData Struct Reference
[Ember Common Data Types]

This data structure contains the private key data that is used for Certificate Based Key Exchange (CBKE). More...

`#include <ember-types.h>`

Data Fields

int8u contents [EMBER_PRIVATE_KEY_SIZE]

Detailed Description

This data structure contains the private key data that is used for Certificate Based Key Exchange (CBKE).

Definition at line **1238** of file **ember-types.h**.

Field Documentation

int8u EmberPrivateKeyData::contents[EMBER_PRIVATE_KEY_SIZE]

Definition at line **1239** of file **ember-types.h**.

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

- **ember-types.h**

EmberPublicKeyData Struct Reference
[Ember Common Data Types]

This data structure contains the public key data that is used for Certificate Based Key Exchange (CBKE). More...

`#include <ember-types.h>`

Data Fields

int8u contents [EMBER_PUBLIC_KEY_SIZE]

Detailed Description

This data structure contains the public key data that is used for Certificate Based Key Exchange (CBKE).

Definition at line **1232** of file **ember-types.h**.

Field Documentation

int8u EmberPublicKeyData::contents[EMBER_PUBLIC_KEY_SIZE]

Definition at line **1233** of file **ember-types.h**.

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

- **ember-types.h**

EmberRouteTableEntry Struct Reference
[Ember Common Data Types]

Defines an entry in the route table. More...

#include <ember-types.h>

Data Fields

int16u	destination
int16u	nextHop
int8u	status
int8u	age
int8u	concentratorType
int8u	routeRecordState

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 790 of file ember-types.h.

Field Documentation

int16u EmberRouteTableEntry::destination

The short id of the destination.

Definition at line 792 of file ember-types.h.

int16u EmberRouteTableEntry::nextHop

The short id of the next hop to this destination.

Definition at line 794 of file ember-types.h.

int8u EmberRouteTableEntry::status

Indicates whether this entry is active (0), being discovered (1), or unused (3).

Definition at line 797 of file ember-types.h.

int8u EmberRouteTableEntry::age

The number of seconds since this route entry was last used to send a packet.

Definition at line 800 of file ember-types.h.

int8u 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 803 of file ember-types.h.

int8u 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 **808** of file **ember-types.h**.

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

- **ember-types.h**

EmberSignatureData Struct Reference
[Ember Common Data Types]

This data structure contains a DSA signature. It is the bit concatenation of the 'r' and 's' components of the signature. More...

`#include <ember-types.h>`

Data Fields

int8u	**contents** [EMBER_SIGNATURE_SIZE]

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 **1251** of file **ember-types.h**.

Field Documentation

int8u EmberSignatureData::contents[EMBER_SIGNATURE_SIZE]

Definition at line **1252** of file **ember-types.h**.

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

- **ember-types.h**

EmberSmacData Struct Reference
[Ember Common Data Types]

This data structure contains the Shared Message Authentication Code (SMAC) data that is used for Certificate Based Key Exchange (CBKE). More...

`#include <ember-types.h>`

Data Fields

int8u	**contents** [EMBER_SMAC_SIZE]

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 **1244** of file **ember-types.h**.

Field Documentation

int8u EmberSmacData::contents[EMBER_SMAC_SIZE]

Definition at line **1245** of file **ember-types.h**.

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

- **ember-types.h**

EmberTaskControl Struct Reference
[Ember Common Data Types]

Control structure for tasks. More...

`#include <ember-types.h>`

Data Fields

int32u	nextEventTime
EmberEventData *	events
boolean	busy

Detailed Description

Control structure for tasks.

This structure should not be accessed directly.

Definition at line 1037 of file ember-types.h.

Field Documentation

int32u EmberTaskControl::nextEventTime

Definition at line 1039 of file ember-types.h.

EmberEventData* EmberTaskControl::events

Definition at line 1041 of file ember-types.h.

boolean EmberTaskControl::busy

Definition at line 1043 of file ember-types.h.

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

- ember-types.h

EmberZigbeeNetwork Struct Reference
[Ember Common Data Types]

Defines a ZigBee network and the associated parameters. More...

`#include <ember-types.h>`

Data Fields

int16u	panId
int8u	channel
boolean	allowingJoin
int8u	extendedPanId [8]
int8u	stackProfile
int8u	nwkUpdateId

Detailed Description

Defines a ZigBee network and the associated parameters.

Definition at line 284 of file ember-types.h.

Field Documentation

int16u EmberZigbeeNetwork::panId

Definition at line 285 of file ember-types.h.

int8u EmberZigbeeNetwork::channel

Definition at line 286 of file ember-types.h.

boolean EmberZigbeeNetwork::allowingJoin

Definition at line 287 of file ember-types.h.

int8u EmberZigbeeNetwork::extendedPanId[8]

Definition at line 288 of file ember-types.h.

int8u EmberZigbeeNetwork::stackProfile

Definition at line 289 of file ember-types.h.

int8u EmberZigbeeNetwork::nwkUpdateId

Definition at line 290 of file ember-types.h.

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

- ember-types.h

InterPanHeader Struct Reference
[Sending and Receiving Messages]

A struct for keeping track of all of the header info. More...

#include <ami-inter-pan.h>

Data Fields

int8u	messageType
int16u	panId
boolean	hasLongAddress
EmberNodeId	shortAddress
EmberEUI64	longAddress
int16u	profileId
int16u	clusterId
int16u	groupId

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 47 of file ami-inter-pan.h.

Field Documentation

int8u InterPanHeader::messageType

Definition at line 48 of file ami-inter-pan.h.

int16u InterPanHeader::panId

Definition at line 53 of file ami-inter-pan.h.

boolean InterPanHeader::hasLongAddress

Definition at line 54 of file ami-inter-pan.h.

EmberNodeId InterPanHeader::shortAddress

Definition at line 55 of file ami-inter-pan.h.

EmberEUI64 InterPanHeader::longAddress

Definition at line 56 of file ami-inter-pan.h.

int16u InterPanHeader::profileId

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

int16u InterPanHeader::clusterId

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

int16u InterPanHeader::groupId

Definition at line **61** 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**

_STM32F103RET_Host_API.top File Reference

Starting page for the Ember API documentation for the STM32F103RET Host, exclusively for building documentation. More...

Go to the source code of this file.

Detailed Description

Starting page for the Ember API documentation for the STM32F103RET Host, exclusively for building documentation.

This file is used by Doxygen to generate the main page for the Ember API documentation, STM32F103RET Host.

Definition in file **_STM32F103RET_Host_API.top**.

_STM32F103RET_Host_API.top

Go to the documentation of this file.

```
00001
```


adc.h File Reference

Go to the source code of this file.

Defines

#define	**TEMP_SENSOR_PIN**
#define	**TEMP_SENSOR_PORT**
#define	**TEMP_SENSOR_ADC**
#define	**TEMP_SENSOR_ADC_CHAN**
#define	**TEMP_ENABLE_PIN**
#define	**TEMP_ENABLE_PORT**

Functions

void	**halInternalInitAdc** (void)
int16u	**halSampleAdc** (void)
int16s	**halConvertValueToVolts** (**int16u** value)

Detailed Description

See **STM32F103RET Specific ADC** for documentation.

Definition in file **adc.h**.

adc.h

Go to the documentation of this file.

```
00001
00017 #ifndef __ADC_H__
00018 #define __ADC_H__
00019
00022 #define TEMP_SENSOR_PIN       GPIO_Pin_0
00023
00025 #define TEMP_SENSOR_PORT      GPIOC
00026
00028 #define TEMP_SENSOR_ADC       ADC1
00029
00031 #define TEMP_SENSOR_ADC_CHAN  10
00032
00035 #define TEMP_ENABLE_PIN       GPIO_Pin_8
00036
00038 #define TEMP_ENABLE_PORT      GPIOA
00039
00040
00043 void halInternalInitAdc(void);
00044
00051 int16u halSampleAdc(void);
00052
00061 int16s halConvertValueToVolts(int16u value);
00062
00063 #endif //__ADC_H__
00064
```


ami-inter-pan-host.h File Reference

Utilities for sending and receiving ZigBee AMI InterPAN messages. See **Sending and Receiving Messages** for documentation. More...

Go to the source code of this file.

Data Structures

struct	**InterPanHeader** A struct for keeping track of all of the header info. More...

Defines

#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

int8u	**makeInterPanMessage** (**InterPanHeader** *headerData, **int8u** *message, **int8u** maxLength, **int8u** *payload, **int8u** payloadLength)
int8u	**parseInterPanMessage** (**int8u** *message, **int8u** messageLength, **InterPanHeader** *headerData)

Detailed Description

Utilities for sending and receiving ZigBee AMI InterPAN messages. See **Sending and Receiving Messages** for documentation.

Definition in file **ami-inter-pan-host.h**.

ami-inter-pan-host.h

Go to the documentation of this file.

```
00001
00015 #ifndef AMI_INTER_PAN_HOST_H
00016 #define AMI_INTER_PAN_HOST_H
00017
00024 #define INTER_PAN_UNICAST   0x03
00025 #define INTER_PAN_BROADCAST 0x0B
00026 #define INTER_PAN_MULTICAST 0x0F
00027
00028
00029 // Frame control, sequence, dest PAN ID, dest, source PAN ID, source.
00030 #define MAX_INTER_PAN_MAC_SIZE (2 + 1 + 2 + 8 + 2 + 8)
00031 //Short form has a short destination.
00032
00033 // NWK stub frame has two control bytes.
00034 #define STUB_NWK_SIZE 2
00035 #define STUB_NWK_FRAME_CONTROL 0x000B
00036
00037 // APS frame control, group ID, cluster ID, profile ID
00038 #define MAX_STUB_APS_SIZE (1 + 2 + 2 + 2)
00039
00040 // Short form has no group ID.
00041 #define MAX_INTER_PAN_HEADER_SIZE \
00042  (MAX_INTER_PAN_MAC_SIZE + STUB_NWK_SIZE + MAX_STUB_APS_SIZE)
00043
00048 typedef struct {
00049   int8u messageType;            // one of the INTER_PAN_...CAST values
00050
00051   // MAC addressing
00052   // For outgoing messages this is the destination.  For incoming messages
00053   // it is the source, which always has a long address.
00054   int16u panId;
00055   boolean hasLongAddress;       // always TRUE for incoming messages
00056   EmberNodeId shortAddress;
00057   EmberEUI64 longAddress;
00058
00059   // APS data
00060   int16u profileId;
00061   int16u clusterId;
00062   int16u groupId;               // only used for INTER_PAN_MULTICAST
00063 } InterPanHeader;
00064
00071 int8u makeInterPanMessage(InterPanHeader *headerData,
00072                           int8u *message,
00073                           int8u maxLength,
00074                           int8u *payload,
00075                           int8u payloadLength);
00076
00084 int8u parseInterPanMessage(int8u *message,
00085                            int8u messageLength,
00086                            InterPanHeader *headerData);
00087
00088 #endif // AMI_INTER_PAN_HOST_H
00089
```


ami-inter-pan.h File Reference

Utilities for sending and receiving ZigBee AMI InterPAN messages. See **Sending and Receiving Messages** for documentation. More...

Go to the source code of this file.

Data Structures

struct	**InterPanHeader** A struct for keeping track of all of the header info. More...

Defines

#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)
int8u	**parseInterPanMessage** (**EmberMessageBuffer** message, **int8u** startOffset, **InterPanHeader** *headerData)

Detailed Description

Utilities for sending and receiving ZigBee AMI InterPAN messages. See **Sending and Receiving Messages** for documentation.

Definition in file **ami-inter-pan.h**.

ami-inter-pan.h

Go to the documentation of this file.

```
00001
00015 #ifndef AMI_INTER_PAN_H
00016 #define AMI_INTER_PAN_H
00017
00018 // The three types of inter-PAN messages.  The values are actually the
00019 // corresponding APS frame controls.
00020 //
00021 // 0x03 is the special interPAN message type.  Unicast mode is 0x00,
00022 // broadcast mode is 0x08, and multicast mode is 0x0C.
00023 //
00024
00025 #define INTER_PAN_UNICAST   0x03
00026 #define INTER_PAN_BROADCAST 0x0B
00027 #define INTER_PAN_MULTICAST 0x0F
00028
00029 // Frame control, sequence, dest PAN ID, dest, source PAN ID, source.
00030 #define MAX_INTER_PAN_MAC_SIZE (2 + 1 + 2 + 8 + 2 + 8)
00031 // Short form has a short destination.
00032
00033 // NWK stub frame has two control bytes.
00034 #define STUB_NWK_SIZE 2
00035 #define STUB_NWK_FRAME_CONTROL 0x000B
00036
00037 // APS frame control, group ID, cluster ID, profile ID
00038 #define MAX_STUB_APS_SIZE (1 + 2 + 2 + 2)
00039 // Short form has no group ID.
00040
00041 #define MAX_INTER_PAN_HEADER_SIZE \
00042  (MAX_INTER_PAN_MAC_SIZE + STUB_NWK_SIZE + MAX_STUB_APS_SIZE)
00043
00047 typedef struct {
00048   int8u messageType;            // one of the INTER_PAN_...CAST values
00049
00050   // MAC addressing
00051   // For outgoing messages this is the destination.  For incoming messages
00052   // it is the source, which always has a long address.
00053   int16u panId;
00054   boolean hasLongAddress;       // always TRUE for incoming messages
00055   EmberNodeId shortAddress;
00056   EmberEUI64 longAddress;
00057
00058   // APS data
00059   int16u profileId;
00060   int16u clusterId;
00061   int16u groupId;               // only used for INTER_PAN_MULTICAST
00062 } InterPanHeader;
00063
00064
00068 EmberMessageBuffer makeInterPanMessage(InterPanHeader *headerData,
00069                                        EmberMessageBuffer payload);
00070
00078 int8u parseInterPanMessage(EmberMessageBuffer message,
00079                            int8u startOffset,
00080                            InterPanHeader *headerData);
00081
00082 #endif // AMI_INTER_PAN_H
00083
```


bootload-ezsp-utils.h File Reference

Utilities used for performing stand-alone bootloading over EZSP. See **Bootloading** for documentation. More...

Go to the source code of this file.

Defines

#define	**TICKS_PER_QUARTER_SECOND**

Functions

boolean	**hostBootloadUtilLaunchRequestHandler** (**int8u** lqi, **int8s** rssi, **int16u** manufacturerId, **int8u** *hardwareTag, **EmberEUI64** sourceEui)
void	**hostBootloadUtilQueryResponseHandler** (**int8u** lqi, **int8s** rssi, **boolean** bootloaderActive, **int16u** manufacturerId, **int8u** *hardwareTag, **EmberEUI64** targetEui, **int8u** bootloaderCapabilities, **int8u** platform, **int8u** micro, **int8u** phy, **int16u** blVersion)
void	**hostBootloadReinitHandler** (void)
boolean	**isTheSameEui64** (**EmberEUI64** sourceEui, **EmberEUI64** targetEui)
void	**printLittleEndianEui64** (**int8u** port, **EmberEUI64** eui64)
void	**printBigEndianEui64** (**int8u** port, **EmberEUI64** eui64)
EmberStatus	**debugPrintf** (**int8u** port, PGM_P formatString,...)

Variables

int16u	**nodeBlVersion**
int8u	**nodePlat**
int8u	**nodeMicro**
int8u	**nodePhy**
EzspStatus	**bootloadEzspLastError**
EzspStatus	**ignoreNextEzspError**

Detailed Description

Utilities used for performing stand-alone bootloading over EZSP. See **Bootloading** for documentation.

Definition in file **bootload-ezsp-utils.h**.

bootload-ezsp-utils.h

Go to the documentation of this file.

```
00001
00016 // application timers are based on quarter second intervals, each
00017 // quarter second is measured in millisecond ticks. This value defines
00018 // the approximate number of millisecond ticks in a quarter second.
00019 // Account for variations in system timers.
00020 #ifdef AVR_ATMEGA_32
00021 #define TICKS_PER_QUARTER_SECOND 225
00022 #else
00023 #define TICKS_PER_QUARTER_SECOND 250
00024 #endif
00025
00026 // Node build info
00027 extern int16u nodeBlVersion;
00028 extern int8u nodePlat;
00029 extern int8u nodeMicro;
00030 extern int8u nodePhy;
00031
00032
00033 // Both of these need to be correctly handled in the applications's
00034 // ezspErrorHandler().
00035 // ezsp error info
00036 extern EzspStatus bootloadEzspLastError;
00037 // If this is not EZSP_SUCCESS, the next call to ezspErrorHandler()
00038 // will ignore this error.
00039 extern EzspStatus ignoreNextEzspError;
00040
00041
00042 // ****************************************************************
00043 // Callback functions used by the bootload library.
00044
00072 boolean hostBootloadUtilLaunchRequestHandler(int8u lqi,
00073                                              int8s rssi,
00074                                              int16u manufacturerId,
00075                                              int8u *hardwareTag,
00076                                              EmberEUI64 sourceEui);
00077
00121 void hostBootloadUtilQueryResponseHandler(int8u lqi,
00122                                           int8s rssi,
00123                                           boolean bootloaderActive,
00124                                           int16u manufacturerId,
00125                                           int8u *hardwareTag,
00126                                           EmberEUI64 targetEui,
00127                                           int8u bootloaderCapabilities,
00128                                           int8u platform,
00129                                           int8u micro,
00130                                           int8u phy,
00131                                           int16u blVersion);
00132
00140 void hostBootloadReinitHandler(void);
00141
00142
00143 // Suppport routines in the bootloader utils library
00144
00156 boolean isTheSameEui64(EmberEUI64 sourceEui, EmberEUI64 targetEui);
00157
00168 void printLittleEndianEui64(int8u port, EmberEUI64 eui64);
00169
00180 void printBigEndianEui64(int8u port, EmberEUI64 eui64);
00181
00199 EmberStatus debugPrintf(int8u port, PGM_P formatString, ...);
00200
```


bootload-utils.h File Reference

Utilities used for performing stand-alone bootloading. See **Bootloading** for documentation. More...

Go to the source code of this file.

Defines

#define	**BOOTLOAD_HARDWARE_TAG_SIZE**

Enumerations

enum	**bootloadMode** { **BOOTLOAD_MODE_NONE**, **BOOTLOAD_MODE_PASSTHRU** }
enum	**bootloadState** { **BOOTLOAD_STATE_NORMAL**, **BOOTLOAD_STATE_QUERY**, **BOOTLOAD_STATE_WAIT_FOR_AUTH_CHALLENGE**, **BOOTLOAD_STATE_WAIT_FOR_AUTH_RESPONSE**, **BOOTLOAD_STATE_DELAY_BEFORE_START**, **BOOTLOAD_STATE_START_UNICAST_BOOTLOAD**, **BOOTLOAD_STATE_START_BROADCAST_BOOTLOAD**, **BOOTLOAD_STATE_START_SENDING_IMAGE**, **BOOTLOAD_STATE_SENDING_IMAGE**, **BOOTLOAD_STATE_WAIT_FOR_IMAGE_ACK**, **BOOTLOAD_STATE_WAIT_FOR_COMPLETE_ACK**, **BOOTLOAD_STATE_DONE** }

Functions

void	**bootloadUtilInit** (**int8u** appPort, **int8u** bootloadPort)
EmberStatus	**bootloadUtilSendRequest** (**EmberEUI64** targetEui, **int16u** mfgId, **int8u** hardwareTag[BOOTLOAD_HARDWARE_TAG_SIZE], **int8u** encryptKey[BOOTLOAD_AUTH_COMMON_SIZE], **bootloadMode** mode)
void	**bootloadUtilSendQuery** (**EmberEUI64** target)
void	**bootloadUtilStartBootload** (**EmberEUI64** target, **bootloadMode** mode)
void	**bootloadUtilTick** (void)
boolean	**bootloadUtilLaunchRequestHandler** (**int16u** manufacturerId, **int8u** hardwareTag[BOOTLOAD_HARDWARE_TAG_SIZE], **EmberEUI64** sourceEui)
void	**bootloadUtilQueryResponseHandler** (**boolean** bootloaderActive, **int16u** manufacturerId, **int8u** hardwareTag[BOOTLOAD_HARDWARE_TAG_SIZE], **EmberEUI64** targetEui, **int8u** bootloaderCapabilities, **int8u** platform, **int8u** micro, **int8u** phy, **int16u** blVersion)
void	**bootloadUtilSendAuthResponse** (**EmberEUI64** target)

Authentication Challenge and Response

The authentication challenge and response must be the same size. The size is chosen to be evenly divisible by the size of a 128-bit AES block.

#define	**BOOTLOAD_AUTH_COMMON_SIZE**
#define	**BOOTLOAD_AUTH_CHALLENGE_SIZE**
#define	**BOOTLOAD_AUTH_RESPONSE_SIZE**

Bootload State Variables

Used to check whether a bootloading process is currently happening.

#define	**IS_BOOTLOADING**
bootloadState	**blState**

Detailed Description

Utilities used for performing stand-alone bootloading. See **Bootloading** for documentation.

Definition in file **bootload-utils.h**.

---

bootload-utils.h

Go to the documentation of this file.

```
00001
00058 // *******************************************************************
00059 // Literals that are needed by the application.
00060
00066 #define BOOTLOAD_AUTH_COMMON_SIZE    16
00067 #define BOOTLOAD_AUTH_CHALLENGE_SIZE BOOTLOAD_AUTH_COMMON_SIZE
00068 #define BOOTLOAD_AUTH_RESPONSE_SIZE  BOOTLOAD_AUTH_COMMON_SIZE
00069
00076 #define BOOTLOAD_HARDWARE_TAG_SIZE 16
00077
00078 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00079
00082 enum bootloadMode
00083 #else
00084 typedef int8u bootloadMode;
00085 enum
00086 #endif
00087 {
00089   BOOTLOAD_MODE_NONE,
00091   BOOTLOAD_MODE_PASSTHRU,
00092 };
00093
00094 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00095
00106 enum bootloadState
00107 #else
00108 typedef int8u bootloadState;
00109 enum
00110 #endif
00111 {
00112   BOOTLOAD_STATE_NORMAL,
00113   BOOTLOAD_STATE_QUERY,
00114   BOOTLOAD_STATE_WAIT_FOR_AUTH_CHALLENGE,
00115   BOOTLOAD_STATE_WAIT_FOR_AUTH_RESPONSE,
00116   BOOTLOAD_STATE_DELAY_BEFORE_START,
00117   BOOTLOAD_STATE_START_UNICAST_BOOTLOAD,
00118   BOOTLOAD_STATE_START_BROADCAST_BOOTLOAD,
00119   BOOTLOAD_STATE_START_SENDING_IMAGE,
00120   BOOTLOAD_STATE_SENDING_IMAGE,
00121   BOOTLOAD_STATE_WAIT_FOR_IMAGE_ACK,
00122   BOOTLOAD_STATE_WAIT_FOR_COMPLETE_ACK,
00123   BOOTLOAD_STATE_DONE
00124 };
00125
00126
00127 // *******************************************************************
00128 // Public functions that are called by the application.
00129
00143 void bootloadUtilInit(int8u appPort, int8u bootloadPort);
00144
00166 EmberStatus bootloadUtilSendRequest(EmberEUI64 targetEui,
00167                                     int16u mfgId,
00168                                     int8u hardwareTag[BOOTLOAD_HARDWARE_TAG_SIZE],
00169                                     int8u encryptKey[BOOTLOAD_AUTH_COMMON_SIZE],
00170                                     bootloadMode mode);
00171
00184 void bootloadUtilSendQuery(EmberEUI64 target);
00185
00203 void bootloadUtilStartBootload(EmberEUI64 target, bootloadMode mode);
00204
00210 void bootloadUtilTick(void);
00211
00212
00213 // *******************************************************************
00214 // Callback functions used by the bootload library.
00215
00234 boolean bootloadUtilLaunchRequestHandler(int16u manufacturerId,
00235                                          int8u hardwareTag[BOOTLOAD_HARDWARE_TAG_SIZE],
00236                                          EmberEUI64 sourceEui);
00237
00267 void bootloadUtilQueryResponseHandler(boolean bootloaderActive,
00268                                       int16u manufacturerId,
00269                                       int8u hardwareTag[BOOTLOAD_HARDWARE_TAG_SIZE],
```



```
00270                                       EmberEUI64 targetEui,
00271                                       int8u bootloaderCapabilities,
00272                                       int8u platform,
00273                                       int8u micro,
00274                                       int8u phy,
00275                                       int16u blVersion);
00276
00289 void bootloadUtilSendAuthResponse(EmberEUI64 target);
00290
00291
00292 // ****************************************************************
00293 // Bootload state variables
00294
00299 extern bootloadState blState;
00300 #define IS_BOOTLOADING ((blState != BOOTLOAD_STATE_NORMAL) && \
00301                         (blState != BOOTLOAD_STATE_DONE))
00302
```


bootloader-eeprom.h File Reference

Go to the source code of this file.

Defines

#define	EEPROM_PAGE_SIZE
#define	EEPROM_FIRST_PAGE
#define	EEPROM_IMAGE_START
#define	EEPROM_SUCCESS
#define	EEPROM_ERR
#define	EEPROM_ERR_MASK
#define	EEPROM_ERR_PG_BOUNDARY
#define	EEPROM_ERR_PG_SZ
#define	EEPROM_ERR_WRT_DATA
#define	EEPROM_ERR_IMG_SZ
#define	EEPROM_ERR_ADDR

EEPROM interaction functions.

void	halEepromInit (void)
void	halEepromShutdown (void)
int8u	halEepromRead (int32u address, int8u *data, int16u len)
int8u	halEepromWrite (int32u address, const int8u *data, int16u len)

Detailed Description

Definition in file bootloader-eeprom.h.

bootloader-eeprom.h

Go to the documentation of this file.

```
00001
00024 #define EEPROM_PAGE_SIZE      (128ul)
00025
00028 #define EEPROM_FIRST_PAGE   (0)
00029
00033 #define EEPROM_IMAGE_START  (EEPROM_FIRST_PAGE*EEPROM_PAGE_SIZE)
00034
00037 #define EEPROM_SUCCESS 0
00038
00041 #define EEPROM_ERR 1
00042
00045 #define EEPROM_ERR_MASK 0x80
00046
00049 #define EEPROM_ERR_PG_BOUNDARY 0x81
00050
00053 #define EEPROM_ERR_PG_SZ 0x82
00054
00057 #define EEPROM_ERR_WRT_DATA 0x83
00058
00061 #define EEPROM_ERR_IMG_SZ 0x84
00062
00065 #define EEPROM_ERR_ADDR 0x85
00066
00067
00074 void halEepromInit(void);
00075
00078 void halEepromShutdown(void);
00079
00095 int8u halEepromRead(int32u address, int8u *data, int16u len);
00096
00112 int8u halEepromWrite(int32u address, const int8u *data, int16u len);
00113
```


button-common.h File Reference

```
#include "button-specific.h"
```

Go to the source code of this file.

Functions

void	**halInternalInitButton** (void)
int8u	**halButtonState** (**int8u** button)
int8u	**halButtonPinState** (**int8u** button)
void	**halButtonIsr** (**int8u** button, **int8u** state)

Button State Definitions

A set of numerical definitions for use with the button APIs indicating the state of a button.

#define	**BUTTON_PRESSED**
#define	**BUTTON_RELEASED**

Detailed Description

See **Button Control** and micro specific modules for documentation.

Definition in file **button-common.h**.

button-common.h

Go to the documentation of this file.

```
00001
00017 #ifndef __BUTTON_COMMON_H__
00018 #define __BUTTON_COMMON_H__
00019
00020
00029 #define BUTTON_PRESSED  1
00030
00033 #define BUTTON_RELEASED 0
00034
00040 void halInternalInitButton(void);
00041
00053 int8u halButtonState(int8u button);
00054
00067 int8u halButtonPinState(int8u button);
00068
00083 void halButtonIsr(int8u button, int8u state);
00084
00085
00086 //Pull in the micro specific button definitions.  The specific header is chosen
00087 //by the build include path pointing at the appropriate directory.
00088 #include "button-specific.h"
00089
00090
00091 #endif //__BUTTON_COMMON_H__
00092
```


button-specific.h File Reference

Go to the source code of this file.

Defines

#define	BUTTON0
#define	BUTTON0_PIN
#define	BUTTON0_PORT
#define	BUTTON0_EXTI_SOURCE_PORT
#define	BUTTON0_EXTI_SOURCE_PIN
#define	BUTTON0_IRQ
#define	BUTTON1
#define	BUTTON1_PIN
#define	BUTTON1_PORT
#define	BUTTON1_EXTI_SOURCE_PORT
#define	BUTTON1_EXTI_SOURCE_PIN
#define	BUTTON1_IRQ
#define	BUTTON01_ISR

Detailed Description

See **Button Control** and **STM32F103RET Specific Button** for documentation.

Definition in file **button-specific.h**.

button-specific.h

Go to the documentation of this file.

```
00001
00020 #ifndef __BUTTON_SPECIFIC_H__
00021 #define __BUTTON_SPECIFIC_H__
00022
00023
00026 #define BUTTON0                   0  //Just a simple identifier for comparisons
00027
00030 #define BUTTON0_PIN               GPIO_Pin_10
00031
00034 #define BUTTON0_PORT              GPIOB
00035
00039 #define BUTTON0_EXTI_SOURCE_PORT  GPIO_PortSourceGPIOB
00040
00044 #define BUTTON0_EXTI_SOURCE_PIN   GPIO_PinSource10
00045
00048 #define BUTTON0_IRQ               EXTI15_10_IRQn
00049
00050
00053 #define BUTTON1                   1  //Just a simple identifier for comparisons
00054
00057 #define BUTTON1_PIN               GPIO_Pin_11
00058
00061 #define BUTTON1_PORT              GPIOB
00062
00066 #define BUTTON1_EXTI_SOURCE_PORT  GPIO_PortSourceGPIOB
00067
00071 #define BUTTON1_EXTI_SOURCE_PIN   GPIO_PinSource11
00072
00075 #define BUTTON1_IRQ               EXTI15_10_IRQn
00076
00077
00081 #define BUTTON01_ISR              EXTI15_10_IRQHandler
00082
00083
00084 #endif //__BUTTON_SPECIFIC_H__
00085
```


buzzer.h File Reference

Go to the source code of this file.

Functions

void	**halPlayTune_P** (**int8u** PGM *tune, **boolean** bkg)
void	**halStartBuzzerTone** (**int16u** frequency)
void	**halStopBuzzerTone** (void)

Variables

int8u PGM	**hereIamTune** []

Note Definitions

Flats are used instead of sharps because # is a special character.

#define	**NOTE_C3**
#define	**NOTE_Db3**
#define	**NOTE_D3**
#define	**NOTE_Eb3**
#define	**NOTE_E3**
#define	**NOTE_F3**
#define	**NOTE_Gb3**
#define	**NOTE_G3**
#define	**NOTE_Ab3**
#define	**NOTE_A3**
#define	**NOTE_Bb3**
#define	**NOTE_B3**
#define	**NOTE_C4**
#define	**NOTE_Db4**
#define	**NOTE_D4**
#define	**NOTE_Eb4**
#define	**NOTE_E4**
#define	**NOTE_F4**
#define	**NOTE_Gb4**
#define	**NOTE_G4**
#define	**NOTE_Ab4**
#define	**NOTE_A4**
#define	**NOTE_Bb4**
#define	**NOTE_B4**
#define	**NOTE_C5**
#define	**NOTE_Db5**
#define	**NOTE_D5**
#define	**NOTE_Eb5**
#define	**NOTE_E5**
#define	**NOTE_F5**
#define	**NOTE_Gb5**
#define	**NOTE_G5**
#define	**NOTE_Ab5**
#define	**NOTE_A5**
#define	**NOTE_Bb5**
#define	**NOTE_B5**

Detailed Description

See **STM32F103RET Specific Buzzer** for documentation.

Definition in file **buzzer.h**.

buzzer.h

Go to the documentation of this file.

```
00001
00017 #ifndef __BUZZER_H__
00018 #define __BUZZER_H__
00019
00020
00032 #define NOTE_C3  (130/4)
00033 #define NOTE_Db3 (138/4)
00034 #define NOTE_D3  (146/4)
00035 #define NOTE_Eb3 (155/4)
00036 #define NOTE_E3  (164/4)
00037 #define NOTE_F3  (174/4)
00038 #define NOTE_Gb3 (185/4)
00039 #define NOTE_G3  (196/4)
00040 #define NOTE_Ab3 (207/4)
00041 #define NOTE_A3  (220/4)
00042 #define NOTE_Bb3 (233/4)
00043 #define NOTE_B3  (246/4)
00044 #define NOTE_C4  (261/4)
00045 #define NOTE_Db4 (277/4)
00046 #define NOTE_D4  (293/4)
00047 #define NOTE_Eb4 (311/4)
00048 #define NOTE_E4  (329/4)
00049 #define NOTE_F4  (349/4)
00050 #define NOTE_Gb4 (369/4)
00051 #define NOTE_G4  (392/4)
00052 #define NOTE_Ab4 (415/4)
00053 #define NOTE_A4  (440/4)
00054 #define NOTE_Bb4 (466/4)
00055 #define NOTE_B4  (493/4)
00056 #define NOTE_C5  (523/4)
00057 #define NOTE_Db5 (554/4)
00058 #define NOTE_D5  (587/4)
00059 #define NOTE_Eb5 (622/4)
00060 #define NOTE_E5  (659/4)
00061 #define NOTE_F5  (698/4)
00062 #define NOTE_Gb5 (739/4)
00063 #define NOTE_G5  (783/4)
00064 #define NOTE_Ab5 (830/4)
00065 #define NOTE_A5  (880/4)
00066 #define NOTE_Bb5 (932/4)
00067 #define NOTE_B5  (987/4)
00068
00096 void halPlayTune_P(int8u PGM *tune, boolean bkg);
00097
00098
00104 void halStartBuzzerTone(int16u frequency);
00105
00106
00109 void halStopBuzzerTone(void);
00110
00111
00115 extern int8u PGM hereIamTune[];
00116
00117 #endif //__BUZZER_H__
00118
```


cbke-crypto-engine.h File Reference

EmberZNet Smart Energy security API. See **Smart Energy Security** for documention. More...

Go to the source code of this file.

Functions

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

Detailed Description

EmberZNet Smart Energy security API. See **Smart Energy Security** for documention.

Definition in file **cbke-crypto-engine.h**.

cbke-crypto-engine.h

Go to the documentation of this file.

```
00001
00030 EmberStatus emberGetCertificate(EmberCertificateData* result);
00031
00047 EmberStatus emberGenerateCbkeKeys(void);
00048
00071 EmberStatus emberCalculateSmacs(boolean amInitiator,
00072                                  EmberCertificateData* partnerCert,
00073                                  EmberPublicKeyData* partnerEphemeralPublicKey);
00074
00089 EmberStatus emberClearTemporaryDataMaybeStoreLinkKey(boolean storeLinkKey);
00090
00091 /* @brief LEGACY FUNCTION:  This functionality has been replaced by a single
00092  * bit in the ::EmberApsFrame, ::EMBER_APS_OPTION_DSA_SIGN.  Devices wishing
00093  * to send signed messages should use that as it requires fewer function calls
00094  * and message buffering.  emberDsaSignHandler() is still called when using
00095  * ::EMBER_APS_OPTION_DSA_SIGN.  However, this function is still supported.
00096  *
00097  * This function begins the process of signing the passed message
00098  * contained within the buffer.  If no other ECC operation is going on,
00099  * it will immediately return with ::EMBER_OPERATION_IN_PROGRESS.
00100  * It will delay a period of time to let APS retries take place, but then it
00101  * will shutdown the radio and consume the CPU processing until the signing
00102  * is complete.  This may take up to 1 second.
00103  *
00104  * The signed message will be returned in ::emberDsaSignHandler().
00105  *
00106  * Note that the last byte of the buffer contents passed to this function has
00107  * special significance.  As the typical use case for DSA signing is to sign the
00108  * ZCL payload of a DRLC Report Event Status message in SE 1.0, there is often
00109  * both a signed portion (ZCL payload) and an unsigned portion (ZCL header).
00110  * The last byte in the content of messageToSign is therefore used as a
00111  * special indicator to signify how many bytes of leading data in the buffer
00112  * should be excluded from consideration during the signing process.  If the
00113  * signature needs to cover the entire buffer (all bytes except last one),
00114  * the caller should ensure that the last byte of the buffer contents is 0.
00115  * When the signature operation is complete, this final byte will be replaced
00116  * by the signature type indicator (0x01 for ECDSA signatures), and the
00117  * actual signature will be appended to the buffer after this byte.
00118  *
00119  * @param messageToSign The message buffer containing the complete message,
00120  *   both the to-be-signed portion as well as any leading data excluded from
00121  *   the signing operation.  See note above regarding special requirements
00122  *   for this buffer.
00123  *
00124  * @return ::EMBER_OPERATION_IN_PROGRESS if the stack has queued up the
00125  *   operation for execution.  ::EMBER_INVALID_CALL if the operation can't be
00126  *   performed in this context (possibly because another ECC operation is
00127  *   pending.)
00128  */
00129 EmberStatus emberDsaSign(EmberMessageBuffer messageToSign);
00130
00131
00132 /* @brief This function is an application callback that must be defined
00133  * when using CBKE.  It is called when the ephemeral key generation operation
00134  * is complete.  The newly generated public key is passed back to the
00135  * application to be sent to the CBKE partner over-the-air.  Internally
00136  * the stack saves the public and private key pair until it the function
00137  * ::emberClearTemporaryDataMaybeStoreLinkKey() is called by the application.
00138  *
00139  * @param status This is the ::EmberStatus value indicating the success or
00140  *   failure of the operation.
00141  * @param ephemeralPublicKey A pointer to an ::EmberPublicKeyData structure
00142  *   containing the newly generated public key.
00143  */
00144 void emberGenerateCbkeKeysHandler(EmberStatus status,
00145                                   EmberPublicKeyData* ephemeralPublicKey);
00146
00147 /* @brief This function is an application callback that must be defined
00148  * when using CBKE.  It is called when the shared secret generation is
00149  * complete and the link key and SMACs have been derived.  The link key is
00150  * stored in a temporary location until the application decides to
00151  * store or discard the key by calling
00152  * ::emberClearTemporaryDataMaybeStoreLinkKey().
```



```
00153  *
00154  * @param status This is the ::EmberStatus value indicating the success or
00155  *   failure of the operation.
00156  * @param initiatorSmac This is a pointer to the ::EmberSmacData structure
00157  *   to the initiator's version of the SMAC.
00158  * @param responderSmac This is a pointer to the ::EmberSmacData structure
00159  *   to the responder's version of the SMAC.
00160  */
00161 void emberCalculateSmacsHandler(EmberStatus status,
00162                                 EmberSmacData* initiatorSmac,
00163                                 EmberSmacData* responderSmac);
00164
00165 /* @brief This function is an application callback that must be defined
00166  * when using CBKE.  This callback is provided to the application to let
00167  * it know that the ECC operations have completed and the radio has been turned
00168  * back on.  When using the sign-and-send option of the ::EmberApsFrame,
00169  * ::EMBER_APS_OPTION_DSA_SIGN, the handler will NOT return the complete
00170  * signed message.  This callback is merely informative.  If ::emberDsaSign()
00171  * has been called, the message plus signature will be returned to the caller
00172  * and it must be sent separately by one of the message send primitives
00173  * (such as ::emberSendUnicast()).
00174  *
00175  * @param status This is the ::EmberStatus value indicating the success or
00176  *   failure of the operation.
00177  * @param signedMessage This is the ::EmberMessageBuffer indicating the newly
00178  *   signed message, if ::emberDsaSign() was called.  If message was signed
00179  *   using ::EMBER_APS_OPTION_DSA_SIGN then this will be
00180  *    ::EMBER_NULL_MESSAGE_BUFFER.
00181  */
00182 void emberDsaSignHandler(EmberStatus status,
00183                          EmberMessageBuffer signedMessage);
00184
00185
00186 /* @brief This function is used to update the Smart Energy certificate,
00187  * CA public key, and local private key that the device uses for CBKE.
00188  * The preferred method for adding certificates is to pre-install them
00189  * in MFG tokens when the chip is manufactured.  However this function
00190  * allows the certificate to be updated at runtime after the device has
00191  * been deployed.
00192  *   The behavior of this function differs based on the hardware platform.
00193  *
00194  * For the 2xx:
00195  * To use this functionality the application must also set
00196  * the stack configuration value ::EMBER_CERTIFICATE_TABLE_SIZE to 1.
00197  * Attempts to call this function with ::EMBER_CERTIFICATE_TABLE_SIZE of 0
00198  * will return ::EMBER_SECURITY_CONFIGURATION_INVALID.
00199  * The passed security data will be persistently stored in stack tokens.
00200  * The certificate contains the EUI64 it is associated with.  If that
00201  * EUI64 matches the EUI64 currently in use by the device, this
00202  * function may be called at any time, even while running in a network.
00203  * If the EUI64 does not match, this function may only be called when the
00204  * network is in a state of ::EMBER_NO_NETWORK.  Attempts to do otherwise
00205  * will result in a return value of ::EMBER_INVALID_CALL.
00206  *
00207  * For the 3xx:
00208  * This function allows a one-time write of the MFG token if it has not
00209  * already been set.  It does NOT utilize the ::EMBER_CERTIFICATE_TABLE_SIZE
00210  * so that should remain set at 0.  Attempts to write the certificate that
00211  * has already been written will return a result of
00212  * ::EMBER_ERR_FLASH_WRITE_INHIBITED.  If the EUI64 in the certificate is
00213  * the same as the current EUI of the device then this function may be called
00214  * while the stack is up.  If the EUI in the certificate is different than
00215  * the current value, this function may only be called when the network is in
00216  * a state of ::EMBER_NO_NETWORK.  Attempts to do otherwise will result in a
00217  * return value of ::EMBER_INVALID_CALL.  If the EUI in the certificate is
00218  * different than the current value this function will also write the
00219  * Custom EUI64 MFG token.  If that token has already been written the operation
00220  * will fail and return a result of ::EMBER_BAD_ARGUMENT.
00221  * If all the above criteria is met the token will be written and
00222  * ::EMBER_SUCCESS will be returned.
00223  *
00224  * @note The device will immediately and persistently <b>change its EUI64
00225  * to match the value in the certificate</b>.
00226  *
00227  * @param caPublic A pointer to the CA public key data that will be stored
00228  *   in stack tokens.
00229  * @param myCert A pointer to the certificate data that will be stored in
00230  *   stack tokens.
00231  * @param mykey A pointer to the private key data that will be stored in
00232  *   stack tokens.
00233  * @return The ::EmberStatus value indicating success or failure of the
```



```
00234  *   operation.
00235  */
00236 EmberStatus emberSetPreinstalledCbkeData(EmberPublicKeyData* caPublic,
00237                                          EmberCertificateData* myCert,
00238                                          EmberPrivateKeyData* myKey);
00239
00240 /* @brief This function retrieves the EUI64 from the stack token
00241  * Smart Energy Certificate (it does not examine the MFG token certificate)
00242  * and returns the value in the "Subject" field (the EUI64) to the caller.
00243  * If no stack token is set, the ::EMBER_CERTIFICATE_TABLE_SIZE is zero
00244  * or if the CBKE library is not present, this function returns FALSE
00245  * and the EUI64 for the return value is not set.
00246  *
00247  * @param certEui64 The location of the return value for the EUI64.
00248  * @return TRUE if the stack token certificate is set and the EUI64
00249  *   return value is valid.  FALSE otherwise.
00250  */
00251 boolean emberGetStackCertificateEui64(EmberEUI64 certEui64);
00252
00253 /* @brief This function verifies the ECDSA signature of the
00254  * calculated digest and the associated received signature, using
00255  * the signerCertificate passed in.  It is expected that the application
00256  * obtains the signerCertificate and performs the message digest calculation
00257  * on its own.
00258  */
00259 EmberStatus emberDsaVerify(EmberMessageDigest* digest,
00260                            EmberCertificateData* signerCertificate,
00261                            EmberSignatureData* receivedSig);
00262
00263 /* @brief This callback is executed by the stack when the DSA verification
00264  * has completed and has a result.  If the result is EMBER_SUCCESS, the
00265  * signature is valid.  If the result is EMBER_SIGNATURE_VERIFY_FAILURE
00266  * then the signature is invalid.  If the result is anything else then the
00267  * signature verify operation failed and the validity is unknown.
00268  */
00269 void emberDsaVerifyHandler(EmberStatus status);
00270
00271
```


command-interpreter2.h File Reference

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

Go to the source code of this file.

Data Structures

struct	**EmberCommandEntry** Command entry for a command table. More...

Defines

#define	**MAX_TOKEN_COUNT**
#define	**EMBER_COMMAND_INTERPRETER_CONFIGURATION_ECHO**
#define	**emberProcessCommandInput**(port)
#define	**emberCommandInterpreterEchoOn**()
#define	**emberCommandInterpreterEchoOff**()
#define	**emberCommandInterpreterIsEchoOn**()

Typedefs

typedef void(*	**CommandAction**)(void)

Enumerations

enum	**EmberCommandStatus** {
	EMBER_CMD_SUCCESS,
	EMBER_CMD_ERR_PORT_PROBLEM,
	EMBER_CMD_ERR_NO_SUCH_COMMAND,
	EMBER_CMD_ERR_WRONG_NUMBER_OF_ARGUMENTS,
	EMBER_CMD_ERR_ARGUMENT_OUT_OF_RANGE,
	EMBER_CMD_ERR_ARGUMENT_SYNTAX_ERROR,
	EMBER_CMD_ERR_STRING_TOO_LONG,
	EMBER_CMD_ERR_INVALID_ARGUMENT_TYPE
	}

Functions

void	**emberCommandErrorHandler** (**EmberCommandStatus** status)
void	**emberPrintCommandUsage** (**EmberCommandEntry** *entry)
void	**emberPrintCommandUsageNotes** (void)
void	**emberPrintCommandTable** (void)
void	**emberCommandReaderInit** (void)
boolean	**emberProcessCommandString** (**int8u** *input, **int8u** size)

Variables

EmberCommandEntry *	**emberCurrentCommand**
EmberCommandEntry	**emberCommandTable** []
int8u	**emberCommandInterpreter2Configuration**

Command Table Settings

#define	**EMBER_MAX_COMMAND_ARGUMENTS**
#define	**EMBER_COMMAND_BUFFER_LENGTH**

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)

int32u	**emberUnsignedCommandArgument** (**int8u** index)
int16s	**emberSignedCommandArgument** (**int8u** index)
int8u *	**emberStringCommandArgument** (**int8s** index, **int8u** *length)
int8u	**emberCopyStringArgument** (**int8s** index, **int8u** *destination, **int8u** maxLength, **boolean** leftPad)

Detailed Description

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

Definition in file **command-interpreter2.h**.

command-interpreter2.h

Go to the documentation of this file.

```
00001
00097 #ifndef EMBER_MAX_COMMAND_ARGUMENTS
00098
00101 #define EMBER_MAX_COMMAND_ARGUMENTS 10
00102 #endif
00103
00104 #ifndef EMBER_COMMAND_BUFFER_LENGTH
00105 #define EMBER_COMMAND_BUFFER_LENGTH 100
00106 #endif
00107
00111 // The (+ 1) takes into account the leading command.
00112 #define MAX_TOKEN_COUNT (EMBER_MAX_COMMAND_ARGUMENTS + 1)
00113
00114 typedef void (*CommandAction)(void);
00115
00116 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00117
00119 typedef struct {
00120 #else
00121 typedef PGM struct {
00122 #endif
00123
00126   PGM_P name;
00132   CommandAction action;
00159   PGM_P argumentTypes;
00162   PGM_P description;
00163 } EmberCommandEntry;
00164
00171 extern EmberCommandEntry *emberCurrentCommand;
00172
00173 extern EmberCommandEntry emberCommandTable[];
00174
00178 extern int8u emberCommandInterpreter2Configuration;
00179
00180 #define EMBER_COMMAND_INTERPRETER_CONFIGURATION_ECHO (0x01)
00181
00182 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00183
00188 enum EmberCommandStatus
00189 #else
00190 typedef int8u EmberCommandStatus;
00191 enum
00192 #endif
00193 {
00194   EMBER_CMD_SUCCESS,
00195   EMBER_CMD_ERR_PORT_PROBLEM,
00196   EMBER_CMD_ERR_NO_SUCH_COMMAND,
00197   EMBER_CMD_ERR_WRONG_NUMBER_OF_ARGUMENTS,
00198   EMBER_CMD_ERR_ARGUMENT_OUT_OF_RANGE,
00199   EMBER_CMD_ERR_ARGUMENT_SYNTAX_ERROR,
00200   EMBER_CMD_ERR_STRING_TOO_LONG,
00201   EMBER_CMD_ERR_INVALID_ARGUMENT_TYPE
00202 };
00203
00213 int32u emberUnsignedCommandArgument(int8u index);
00214
00216 int16s emberSignedCommandArgument(int8u index);
00217
00226 int8u *emberStringCommandArgument(int8s index, int8u *length);
00227
00240 int8u emberCopyStringArgument(int8s index,
00241                               int8u *destination,
00242                               int8u maxLength,
00243                               boolean leftPad);
00244
00248 #define emberCopyKeyArgument(index, keyDataPointer)            \
00249   (emberCopyStringArgument((index),                            \
00250                            emberKeyContents((keyDataPointer)), \
00251                            EMBER_ENCRYPTION_KEY_SIZE,          \
00252                            TRUE))
00253
00255 #define emberCopyEui64Argument(index, eui64) \
00256   (emberCopyStringArgument((index), (eui64), EUI64_SIZE, TRUE))
```



```
00257
00267 void emberCommandErrorHandler(EmberCommandStatus status);
00268 void emberPrintCommandUsage(EmberCommandEntry *entry);
00269 void emberPrintCommandUsageNotes(void);
00270 void emberPrintCommandTable(void);
00271
00274 void emberCommandReaderInit(void);
00275
00278 boolean emberProcessCommandString(int8u *input, int8u size);
00279
00288 #define emberProcessCommandInput(port) \
00289   emberProcessCommandString(NULL, port)
00290
00293 #define emberCommandInterpreterEchoOn()                 \
00294   (emberCommandInterpreter2Configuration                \
00295    |= EMBER_COMMAND_INTERPRETER_CONFIGURATION_ECHO)
00296
00299 #define emberCommandInterpreterEchoOff()                \
00300   (emberCommandInterpreter2Configuration                \
00301    &= (~EMBER_COMMAND_INTERPRETER_CONFIGURATION_ECHO))
00302
00305 #define emberCommandInterpreterIsEchoOn()               \
00306   (emberCommandInterpreter2Configuration                \
00307    & EMBER_COMMAND_INTERPRETER_CONFIGURATION_ECHO)
00308
```


crc.h File Reference

Go to the source code of this file.

Defines

#define	**INITIAL_CRC**
#define	**CRC32_START**
#define	**CRC32_END**

Functions

int16u	**halCommonCrc16** (**int8u** newByte, **int16u** prevResult)
int32u	**halCommonCrc32** (**int8u** newByte, **int32u** prevResult)

Detailed Description

See **Cyclic Redundancy Code (CRC)** for detailed documentation.

Definition in file **crc.h**.

crc.h

Go to the documentation of this file.

```
00001
00007 #ifndef __CRC_H__
00008 #define __CRC_H__
00009
00028 int16u halCommonCrc16(int8u newByte, int16u prevResult);
00029
00030
00046 int32u halCommonCrc32(int8u newByte, int32u prevResult);
00047
00048
00051 #define INITIAL_CRC             0xFFFFFFFFL
00052
00053
00056 #define CRC32_START             INITIAL_CRC
00057
00058
00061 #define CRC32_END               0xDEBB20E3L
00062
00063
00067 #endif //__CRC_H__
00068
```


ember-configuration-defaults.h File Reference

User-configurable stack memory allocation defaults. More...

Go to the source code of this file.

Defines

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

Detailed Description

User-configurable stack memory allocation defaults.

Note:

Application developers should **not** modify any portion of this file. Doing so may cause mysterious bugs. Allocations should be adjusted only by defining the appropriate macros in the application's CONFIGURATION_HEADER.

See **Configuration** for documentation.

Definition in file **ember-configuration-defaults.h**.

ember-configuration-defaults.h

Go to the documentation of this file.

```
00001
00014 //  Todo:
00015 //  - explain how to use a configuration header
00016 //  - the documentation of the custom handlers should
00017 //    go in hal/ember-configuration.c, not here
00018 //  - the stack profile documentation is out of date
00019
00047 #ifndef __EMBER_CONFIGURATION_DEFAULTS_H__
00048 #define __EMBER_CONFIGURATION_DEFAULTS_H__
00049
00050 #ifdef CONFIGURATION_HEADER
00051   #include CONFIGURATION_HEADER
00052 #endif
00053
00054 #ifndef EMBER_API_MAJOR_VERSION
00055
00058   #define EMBER_API_MAJOR_VERSION 2
00059 #endif
00060
00061 #ifndef EMBER_API_MINOR_VERSION
00062
00065   #define EMBER_API_MINOR_VERSION 0
00066 #endif
00067
00080 #ifndef EMBER_STACK_PROFILE
00081   #define EMBER_STACK_PROFILE 0
00082 #endif
00083
00084 #if (EMBER_STACK_PROFILE == 2)
00085   #define EMBER_MAX_DEPTH                 15
00086   #define EMBER_SECURITY_LEVEL             5
00087   #define EMBER_MIN_ROUTE_TABLE_SIZE      10
00088   #define EMBER_MIN_DISCOVERY_TABLE_SIZE   4
00089   #define EMBER_INDIRECT_TRANSMISSION_TIMEOUT 7680
00090   #define EMBER_BROADCAST_TABLE_SIZE      15
00091 #endif
00092
00093 #ifndef EMBER_MAX_END_DEVICE_CHILDREN
00094
00098   #define EMBER_MAX_END_DEVICE_CHILDREN 6
00099 #endif
00100
00101 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00102 /* Need to put in a compile time check to make sure that we aren't specifying
00103  * too many child devices.  The current scheme is limited to 32 children
00104  */
00105 #if EMBER_MAX_END_DEVICE_CHILDREN > 32
00106   #error "EMBER_MAX_END_DEVICE_CHILDREN can not exceed 32."
00107 #endif
00108
00109 #endif // DOXYGEN_SHOULD_SKIP_THIS
00110
00111 #ifndef EMBER_SECURITY_LEVEL
00112
00116   #define EMBER_SECURITY_LEVEL 5
00117 #endif
00118
00119 #if ! (EMBER_SECURITY_LEVEL == 0          \
00120        || EMBER_SECURITY_LEVEL == 5)
00121   #error "Unsupported security level"
00122 #endif
00123
00124 #ifdef EMBER_CHILD_TABLE_SIZE
00125   #if (EMBER_MAX_END_DEVICE_CHILDREN < EMBER_CHILD_TABLE_SIZE)
00126     #undef EMBER_CHILD_TABLE_SIZE
00127   #endif
00128 #endif
00129
00130 #ifndef EMBER_CHILD_TABLE_SIZE
00131
00145   #define EMBER_CHILD_TABLE_SIZE EMBER_MAX_END_DEVICE_CHILDREN
00146 #endif
00147
```



```
00161 #ifndef EMBER_KEY_TABLE_SIZE
00162   #define EMBER_KEY_TABLE_SIZE 0
00163 #endif
00164
00174 #ifndef EMBER_CERTIFICATE_TABLE_SIZE
00175   #define EMBER_CERTIFICATE_TABLE_SIZE 0
00176 #else
00177   #if EMBER_CERTIFICATE_TABLE_SIZE > 1
00178     #error "EMBER_CERTIFICATE_TABLE_SIZE > 1 is not supported!"
00179   #endif
00180 #endif
00181
00187 #ifndef EMBER_MAX_DEPTH
00188   #define EMBER_MAX_DEPTH               15
00189 #elif (EMBER_MAX_DEPTH > 15)
00190   // Depth is a 4-bit field
00191   #error "EMBER_MAX_DEPTH cannot be greater than 15"
00192 #endif
00193
00200 #ifndef EMBER_MAX_HOPS
00201   #define EMBER_MAX_HOPS (2 * EMBER_MAX_DEPTH)
00202 #endif
00203
00210 #ifndef EMBER_PACKET_BUFFER_COUNT
00211   #define EMBER_PACKET_BUFFER_COUNT 24
00212 #endif
00213
00225 #define EMBER_MAX_NEIGHBOR_TABLE_SIZE 16
00226 #ifndef EMBER_NEIGHBOR_TABLE_SIZE
00227   #define EMBER_NEIGHBOR_TABLE_SIZE 16
00228 #endif
00229
00236 #ifndef EMBER_INDIRECT_TRANSMISSION_TIMEOUT
00237   #define EMBER_INDIRECT_TRANSMISSION_TIMEOUT 3000
00238 #endif
00239 #define EMBER_MAX_INDIRECT_TRANSMISSION_TIMEOUT 30000
00240 #if (EMBER_INDIRECT_TRANSMISSION_TIMEOUT                                    \
00241      > EMBER_MAX_INDIRECT_TRANSMISSION_TIMEOUT)
00242   #error "Indirect transmission timeout too large."
00243 #endif
00244
00258 #ifndef EMBER_END_DEVICE_POLL_TIMEOUT
00259   #define EMBER_END_DEVICE_POLL_TIMEOUT 5
00260 #endif
00261
00269 #ifndef EMBER_END_DEVICE_POLL_TIMEOUT_SHIFT
00270   #define EMBER_END_DEVICE_POLL_TIMEOUT_SHIFT 6
00271 #endif
00272
00279 #ifndef EMBER_MOBILE_NODE_POLL_TIMEOUT
00280   #define EMBER_MOBILE_NODE_POLL_TIMEOUT 20
00281 #endif
00282
00295 #ifndef EMBER_APS_UNICAST_MESSAGE_COUNT
00296   #define EMBER_APS_UNICAST_MESSAGE_COUNT 10
00297 #endif
00298
00301 #ifndef EMBER_BINDING_TABLE_SIZE
00302   #define EMBER_BINDING_TABLE_SIZE 0
00303 #endif
00304
00309 #ifndef EMBER_ADDRESS_TABLE_SIZE
00310   #define EMBER_ADDRESS_TABLE_SIZE 8
00311 #endif
00312
00319 #ifndef EMBER_RESERVED_MOBILE_CHILD_ENTRIES
00320   #define EMBER_RESERVED_MOBILE_CHILD_ENTRIES 0
00321 #endif
00322
00329 #ifndef EMBER_ROUTE_TABLE_SIZE
00330   #ifdef EMBER_MIN_ROUTE_TABLE_SIZE
00331     #define EMBER_ROUTE_TABLE_SIZE EMBER_MIN_ROUTE_TABLE_SIZE
00332   #else
00333     #define EMBER_ROUTE_TABLE_SIZE 16
00334   #endif
00335 #elif defined(EMBER_MIN_ROUTE_TABLE_SIZE) \
00336       && EMBER_ROUTE_TABLE_SIZE < EMBER_MIN_ROUTE_TABLE_SIZE
00337   #error "EMBER_ROUTE_TABLE_SIZE is less than required by stack profile."
00338 #endif
00339
00345 #ifndef EMBER DISCOVERY TABLE SIZE
```



```
00346   #ifdef EMBER_MIN_DISCOVERY_TABLE_SIZE
00347     #define EMBER_DISCOVERY_TABLE_SIZE EMBER_MIN_DISCOVERY_TABLE_SIZE
00348   #else
00349     #define EMBER_DISCOVERY_TABLE_SIZE 8
00350   #endif
00351 #elif defined(EMBER_MIN_DISCOVERY_TABLE_SIZE) \
00352       && EMBER_DISCOVERY_TABLE_SIZE < EMBER_MIN_DISCOVERY_TABLE_SIZE
00353   #error "EMBER_DISCOVERY_TABLE_SIZE is less than required by stack profile."
00354 #endif
00355
00361 #ifndef EMBER_MULTICAST_TABLE_SIZE
00362   #define EMBER_MULTICAST_TABLE_SIZE 8
00363 #endif
00364
00371 #ifndef EMBER_SOURCE_ROUTE_TABLE_SIZE
00372   #define EMBER_SOURCE_ROUTE_TABLE_SIZE 32
00373 #endif
00374
00380 #ifndef EMBER_BROADCAST_TABLE_SIZE
00381   #define EMBER_BROADCAST_TABLE_SIZE 15
00382 #elif EMBER_BROADCAST_TABLE_SIZE < 15
00383   #error "EMBER_BROADCAST__TABLE_SIZE is less than the minimum value of 15."
00384 #elif 254 < EMBER_BROADCAST_TABLE_SIZE
00385   #error "EMBER_BROADCAST__TABLE_SIZE is larger than the maximum value of 254."
00386 #endif
00387
00397 #if !defined(EMBER_ASSERT_OUTPUT_DISABLED) \
00398     && !defined(EMBER_ASSERT_SERIAL_PORT)
00399   #define EMBER_ASSERT_SERIAL_PORT 1
00400 #endif
00401
00415 #define EMBER_MAXIMUM_ALARM_DATA_SIZE 16
00416
00434 #ifndef EMBER_BROADCAST_ALARM_DATA_SIZE
00435   #define EMBER_BROADCAST_ALARM_DATA_SIZE 0
00436 #elif EMBER_MAXIMUM_ALARM_DATA_SIZE < EMBER_BROADCAST_ALARM_DATA_SIZE
00437   #error "EMBER_BROADCAST_ALARM_DATA_SIZE is too large."
00438 #endif
00439
00448 #ifndef EMBER_UNICAST_ALARM_DATA_SIZE
00449   #define EMBER_UNICAST_ALARM_DATA_SIZE 0
00450 #elif EMBER_MAXIMUM_ALARM_DATA_SIZE < EMBER_UNICAST_ALARM_DATA_SIZE
00451   #error "EMBER_UNICAST_ALARM_DATA_SIZE is too large."
00452 #endif
00453
00457 #ifndef EMBER_FRAGMENT_DELAY_MS
00458   #define EMBER_FRAGMENT_DELAY_MS 0
00459 #endif
00460
00464 #define EMBER_FRAGMENT_MAX_WINDOW_SIZE 8
00465
00470 #ifndef EMBER_FRAGMENT_WINDOW_SIZE
00471   #define EMBER_FRAGMENT_WINDOW_SIZE 1
00472 #elif EMBER_FRAGMENT_MAX_WINDOW_SIZE < EMBER_FRAGMENT_WINDOW_SIZE
00473   #error "EMBER_FRAGMENT_WINDOW_SIZE is too large."
00474 #endif
00475
00476 #ifndef EMBER_BINDING_TABLE_TOKEN_SIZE
00477   #define EMBER_BINDING_TABLE_TOKEN_SIZE EMBER_BINDING_TABLE_SIZE
00478 #endif
00479 #ifndef EMBER_CHILD_TABLE_TOKEN_SIZE
00480   #define EMBER_CHILD_TABLE_TOKEN_SIZE EMBER_CHILD_TABLE_SIZE
00481 #endif
00482 #ifndef EMBER_KEY_TABLE_TOKEN_SIZE
00483   #define EMBER_KEY_TABLE_TOKEN_SIZE EMBER_KEY_TABLE_SIZE
00484 #endif
00485
00498 #ifndef EMBER_REQUEST_KEY_TIMEOUT
00499   #define EMBER_REQUEST_KEY_TIMEOUT 0
00500 #elif EMBER_REQUEST_KEY_TIMEOUT > 10
00501   #error "EMBER_REQUEST_KEY_TIMEOUT is too large."
00502 #endif
00503
00507 #ifndef EMBER_END_DEVICE_BIND_TIMEOUT
00508   #define EMBER_END_DEVICE_BIND_TIMEOUT 60
00509 #endif
00510
00519 #ifndef EMBER_PAN_ID_CONFLICT_REPORT_THRESHOLD
00520   #define EMBER_PAN_ID_CONFLICT_REPORT_THRESHOLD 1
00521 #endif
00522
```



```
00528 #ifndef EMBER_TASK_COUNT
00529  #define EMBER_TASK_COUNT (3)
00530 #endif
00531
00532
00533 #if defined(EMBER_PSL_STACK)
00534
00535   #ifndef EMBER_PSL_GROUP_ADDRESSES
00536
00538     #define EMBER_PSL_GROUP_ADDRESSES 1
00539   #endif
00540
00541   #ifndef EMBER_PSL_RSSI_THRESHOLD
00542
00544     #define EMBER_PSL_RSSI_THRESHOLD -128
00545   #endif
00546
00547 #endif // EMBER_PSL_STACK
00548
00549
00554 #endif //__EMBER_CONFIGURATION_DEFAULTS_H__
```


ember-types.h File Reference

Ember data type definitions. More...

Go to the source code of this file.

Data Structures

struct	**EmberZigbeeNetwork** Defines a ZigBee network and the associated parameters. More...
struct	**EmberNetworkParameters** Holds network parameters. More...
struct	**EmberApsFrame** An in-memory representation of a ZigBee APS frame of an incoming or outgoing message. More...
struct	**EmberBindingTableEntry** Defines an entry in the binding table. More...
struct	**EmberNeighborTableEntry** Defines an entry in the neighbor table. More...
struct	**EmberRouteTableEntry** Defines an entry in the route table. More...
struct	**EmberMulticastTableEntry** Defines an entry in the multicast table. More...
struct	**EmberEventControl** Control structure for events. More...
struct	**EmberTaskControl** Control structure for tasks. More...
struct	**EmberKeyData** This data structure contains the key data that is passed into various other functions. More...
struct	**EmberCertificateData** This data structure contains the certificate data that is used for Certificate Based Key Exchange (CBKE). More...
struct	**EmberPublicKeyData** This data structure contains the public key data that is used for Certificate Based Key Exchange (CBKE). More...
struct	**EmberPrivateKeyData** This data structure contains the private key data that is used for Certificate Based Key Exchange (CBKE). More...
struct	**EmberSmacData** This data structure contains the Shared Message Authentication Code (SMAC) data that is used for Certificate Based Key Exchange (CBKE). More...
struct	**EmberSignatureData** This data structure contains a DSA signature. It is the bit concatenation of the 'r' and 's' components of the signature. More...
struct	**EmberMessageDigest** This data structure contains an AES-MMO Hash (the message digest). More...
struct	**EmberAesMmoHashContext** This data structure contains the context data when calculating an AES MMO hash (message digest). More...
struct	**EmberInitialSecurityState** This describes the Initial Security features and requirements that will be used when forming or joining the network. More...
struct	**EmberCurrentSecurityState** This describes the security features used by the stack for a joined device. More...
struct	**EmberKeyStruct** This describes a one of several different types of keys and its associated data. More...

struct	**EmberMacFilterMatchStruct** This structure indicates a matching raw MAC message has been received by the application configured MAC filters. More...

Defines

#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_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_END**

Typedefs

typedef **int8u**	**EmberTaskId**
struct {	
EmberEventControl * **control**	
void(* **handler**)(void)	
}	**EmberEventData**
typedef **int16u**	**EmberMacFilterMatchData**
typedef **int8u**	**EmberLibraryStatus**

Enumerations

enum	**EmberNodeType** {
	EMBER_UNKNOWN_DEVICE,
	EMBER_COORDINATOR,
	EMBER_ROUTER,
	EMBER_END_DEVICE,
	EMBER_SLEEPY_END_DEVICE,
	EMBER_MOBILE_END_DEVICE
	}

enum	**EmberApsOption** {
	EMBER_APS_OPTION_NONE,
	EMBER_APS_OPTION_DSA_SIGN,
	EMBER_APS_OPTION_ENCRYPTION,
	EMBER_APS_OPTION_RETRY,
	EMBER_APS_OPTION_ENABLE_ROUTE_DISCOVERY,
	EMBER_APS_OPTION_FORCE_ROUTE_DISCOVERY,
	EMBER_APS_OPTION_SOURCE_EUI64,
	EMBER_APS_OPTION_DESTINATION_EUI64,
	EMBER_APS_OPTION_ENABLE_ADDRESS_DISCOVERY,
	EMBER_APS_OPTION_POLL_RESPONSE,
	EMBER_APS_OPTION_ZDO_RESPONSE_REQUIRED,
	EMBER_APS_OPTION_FRAGMENT
	}


```
enum EmberIncomingMessageType {
  EMBER_INCOMING_UNICAST,
  EMBER_INCOMING_UNICAST_REPLY,
  EMBER_INCOMING_MULTICAST,
  EMBER_INCOMING_MULTICAST_LOOPBACK,
  EMBER_INCOMING_BROADCAST,
  EMBER_INCOMING_BROADCAST_LOOPBACK
}
enum EmberOutgoingMessageType {
  EMBER_OUTGOING_DIRECT,
  EMBER_OUTGOING_VIA_ADDRESS_TABLE,
  EMBER_OUTGOING_VIA_BINDING,
  EMBER_OUTGOING_MULTICAST,
  EMBER_OUTGOING_BROADCAST
}
enum EmberNetworkStatus {
  EMBER_NO_NETWORK,
  EMBER_JOINING_NETWORK,
  EMBER_JOINED_NETWORK,
  EMBER_JOINED_NETWORK_NO_PARENT,
  EMBER_LEAVING_NETWORK
}
enum EmberNetworkScanType {
  EMBER_ENERGY_SCAN,
  EMBER_ACTIVE_SCAN
}
enum EmberBindingType {
  EMBER_UNUSED_BINDING,
  EMBER_UNICAST_BINDING,
  EMBER_MANY_TO_ONE_BINDING,
  EMBER_MULTICAST_BINDING
}
enum EmberJoinDecision {
  EMBER_USE_PRECONFIGURED_KEY,
  EMBER_SEND_KEY_IN_THE_CLEAR,
  EMBER_DENY_JOIN,
  EMBER_NO_ACTION
}
enum EmberDeviceUpdate {
  EMBER_STANDARD_SECURITY_SECURED_REJOIN,
  EMBER_STANDARD_SECURITY_UNSECURED_JOIN,
  EMBER_DEVICE_LEFT,
  EMBER_STANDARD_SECURITY_UNSECURED_REJOIN,
  EMBER_HIGH_SECURITY_SECURED_REJOIN,
  EMBER_HIGH_SECURITY_UNSECURED_JOIN,
  EMBER_HIGH_SECURITY_UNSECURED_REJOIN
}
enum EmberClusterListId {
  EMBER_INPUT_CLUSTER_LIST,
  EMBER_OUTPUT_CLUSTER_LIST
}
enum EmberEventUnits {
  EMBER_EVENT_INACTIVE,
  EMBER_EVENT_MS_TIME,
  EMBER_EVENT_QS_TIME,
  EMBER_EVENT_MINUTE_TIME,
  EMBER_EVENT_ZERO_DELAY
}
enum EmberJoinMethod {
  EMBER_USE_MAC_ASSOCIATION,
  EMBER_USE_NWK_REJOIN,
  EMBER_USE_NWK_REJOIN_HAVE_NWK_KEY,
  EMBER_USE_NWK_COMMISSIONING
}
enum EmberCounterType {
  EMBER_COUNTER_MAC_RX_BROADCAST,
  EMBER_COUNTER_MAC_TX_BROADCAST,
  EMBER_COUNTER_MAC_RX_UNICAST,
```



```
  EMBER_COUNTER_MAC_TX_UNICAST_SUCCESS,
  EMBER_COUNTER_MAC_TX_UNICAST_RETRY,
  EMBER_COUNTER_MAC_TX_UNICAST_FAILED,
  EMBER_COUNTER_APS_DATA_RX_BROADCAST,
  EMBER_COUNTER_APS_DATA_TX_BROADCAST,
  EMBER_COUNTER_APS_DATA_RX_UNICAST,
  EMBER_COUNTER_APS_DATA_TX_UNICAST_SUCCESS,
  EMBER_COUNTER_APS_DATA_TX_UNICAST_RETRY,
  EMBER_COUNTER_APS_DATA_TX_UNICAST_FAILED,
  EMBER_COUNTER_ROUTE_DISCOVERY_INITIATED,
  EMBER_COUNTER_NEIGHBOR_ADDED,
  EMBER_COUNTER_NEIGHBOR_REMOVED,
  EMBER_COUNTER_NEIGHBOR_STALE,
  EMBER_COUNTER_JOIN_INDICATION,
  EMBER_COUNTER_CHILD_REMOVED,
  EMBER_COUNTER_ASH_OVERFLOW_ERROR,
  EMBER_COUNTER_ASH_FRAMING_ERROR,
  EMBER_COUNTER_ASH_OVERRUN_ERROR,
  EMBER_COUNTER_NWK_FRAME_COUNTER_FAILURE,
  EMBER_COUNTER_APS_FRAME_COUNTER_FAILURE,
  EMBER_COUNTER_ASH_XOFF,
  EMBER_COUNTER_APS_LINK_KEY_NOT_AUTHORIZED,
  EMBER_COUNTER_NWK_DECRYPTION_FAILURE,
  EMBER_COUNTER_APS_DECRYPTION_FAILURE,
  EMBER_COUNTER_ALLOCATE_PACKET_BUFFER_FAILURE,
  EMBER_COUNTER_RELAYED_UNICAST,
  EMBER_COUNTER_PHY_TO_MAC_QUEUE_LIMIT_REACHED,
  EMBER_COUNTER_TYPE_COUNT
}
enum EmberInitialSecurityBitmask {
  EMBER_DISTRIBUTED_TRUST_CENTER_MODE,
  EMBER_GLOBAL_LINK_KEY,
  EMBER_PRECONFIGURED_NETWORK_KEY_MODE,
  EMBER_HAVE_TRUST_CENTER_EUI64,
  EMBER_TRUST_CENTER_USES_HASHED_LINK_KEY,
  EMBER_HAVE_PRECONFIGURED_KEY,
  EMBER_HAVE_NETWORK_KEY,
  EMBER_GET_LINK_KEY_WHEN_JOINING,
  EMBER_REQUIRE_ENCRYPTED_KEY,
  EMBER_NO_FRAME_COUNTER_RESET,
  EMBER_GET_PRECONFIGURED_KEY_FROM_INSTALL_CODE
}
enum EmberCurrentSecurityBitmask {
  EMBER_STANDARD_SECURITY_MODE_,
  EMBER_DISTRIBUTED_TRUST_CENTER_MODE_,
  EMBER_GLOBAL_LINK_KEY_,
  EMBER_HAVE_TRUST_CENTER_LINK_KEY,
  EMBER_TRUST_CENTER_USES_HASHED_LINK_KEY_
}
enum EmberKeyStructBitmask {
  EMBER_KEY_HAS_SEQUENCE_NUMBER,
  EMBER_KEY_HAS_OUTGOING_FRAME_COUNTER,
  EMBER_KEY_HAS_INCOMING_FRAME_COUNTER,
  EMBER_KEY_HAS_PARTNER_EUI64,
  EMBER_KEY_IS_AUTHORIZED,
  EMBER_KEY_PARTNER_IS_SLEEPY
}
enum EmberKeyType {
  EMBER_TRUST_CENTER_LINK_KEY,
  EMBER_TRUST_CENTER_MASTER_KEY,
  EMBER_CURRENT_NETWORK_KEY,
  EMBER_NEXT_NETWORK_KEY,
  EMBER_APPLICATION_LINK_KEY,
  EMBER_APPLICATION_MASTER_KEY
}
enum EmberKeyStatus {
  EMBER_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
}
enum EmberLinkKeyRequestPolicy {
  EMBER_DENY_KEY_REQUESTS,
  EMBER_ALLOW_KEY_REQUESTS
}
enum EmberMacPassthroughType {
  EMBER_MAC_PASSTHROUGH_NONE,
  EMBER_MAC_PASSTHROUGH_SE_INTERPAN,
  EMBER_MAC_PASSTHROUGH_EMBERNET,
  EMBER_MAC_PASSTHROUGH_EMBERNET_SOURCE,
  EMBER_MAC_PASSTHROUGH_APPLICATION,
  EMBER_MAC_PASSTHROUGH_CUSTOM
}
```

Functions

```
int8u * emberKeyContents (EmberKeyData *key)
int8u * emberCertificateContents (EmberCertificateData *cert)
int8u * emberPublicKeyContents (EmberPublicKeyData *key)
int8u * emberPrivateKeyContents (EmberPrivateKeyData *key)
int8u * emberSmacContents (EmberSmacData *key)
int8u * emberSignatureContents (EmberSignatureData *sig)
```

Miscellaneous Ember Types

```
#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_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
```


enum	**EmberLeaveRequestFlags** {
	EMBER_ZIGBEE_LEAVE_AND_REJOIN,
	EMBER_ZIGBEE_LEAVE_AND_REMOVE_CHILDREN
	}

typedef **int8u**	**EmberStatus**
typedef **int8u**	**EmberEUI64** [EUI64_SIZE]
typedef **int8u**	**EmberMessageBuffer**
typedef **int16u**	**EmberNodeId**
typedef **int16u**	**EmberMulticastId**
typedef **int16u**	**EmberPanId**

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_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**

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

```
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

```
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
```

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 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
}
```

ZDO server mask bits

These are used in server discovery requests and responses.

```
enum EmberZdoServerMask {
  EMBER_ZDP_PRIMARY_TRUST_CENTER,
  EMBER_ZDP_SECONDARY_TRUST_CENTER,
  EMBER_ZDP_PRIMARY_BINDING_TABLE_CACHE,
  EMBER_ZDP_SECONDARY_BINDING_TABLE_CACHE,
  EMBER_ZDP_PRIMARY_DISCOVERY_CACHE,
  EMBER_ZDP_SECONDARY_DISCOVERY_CACHE,
  EMBER_ZDP_NETWORK_MANAGER
}
```

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
}
```

Detailed Description

Ember data type definitions.

See **Ember Common Data Types** for details.

Definition in file **ember-types.h**.

Variable Documentation

EmberEventControl* control

The control structure for the event.

Definition at line **1028** of file **ember-types.h**.

void(* handler)(void)

The procedure to call when the event fires.

ember-types.h

Go to the documentation of this file.

```
00001
00020 #ifndef EMBER_TYPES_H
00021 #define EMBER_TYPES_H
00022
00023 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00024 #include "stack/config/ember-configuration-defaults.h"
00025 #include "stack/include/ember-static-struct.h"
00026 #endif //DOXYGEN_SHOULD_SKIP_THIS
00027
00032
00033
00037 #define EUI64_SIZE 8
00038
00042 #define EXTENDED_PAN_ID_SIZE 8
00043
00047 #define EMBER_ENCRYPTION_KEY_SIZE 16
00048
00053 #define EMBER_CERTIFICATE_SIZE 48
00054
00058 #define EMBER_PUBLIC_KEY_SIZE 22
00059
00063 #define EMBER_PRIVATE_KEY_SIZE 21
00064
00068 #define EMBER_SMAC_SIZE 16
00069
00074 #define EMBER_SIGNATURE_SIZE 42
00075
00079 #define EMBER_AES_HASH_BLOCK_SIZE 16
00080
00081
00085 #ifndef __EMBERSTATUS_TYPE__
00086 #define __EMBERSTATUS_TYPE__
00087   typedef int8u EmberStatus;
00088 #endif //__EMBERSTATUS_TYPE__
00089
00093 typedef int8u EmberEUI64[EUI64_SIZE];
00094
00104 typedef int8u EmberMessageBuffer;
00105
00109 typedef int16u EmberNodeId;
00110
00112 typedef int16u EmberMulticastId;
00113
00117 typedef int16u EmberPanId;
00118
00122 #define EMBER_MAX_802_15_4_CHANNEL_NUMBER 26
00123
00127 #define EMBER_MIN_802_15_4_CHANNEL_NUMBER 11
00128
00132 #define EMBER_NUM_802_15_4_CHANNELS \
00133   (EMBER_MAX_802_15_4_CHANNEL_NUMBER - EMBER_MIN_802_15_4_CHANNEL_NUMBER + 1)
00134
00138 #define EMBER_ALL_802_15_4_CHANNELS_MASK 0x07FFF800UL
00139
00143 #define EMBER_ZIGBEE_COORDINATOR_ADDRESS 0x0000
00144
00149 #define EMBER_NULL_NODE_ID 0xFFFF
00150
00155 #define EMBER_NULL_BINDING 0xFF
00156
00166 #define EMBER_TABLE_ENTRY_UNUSED_NODE_ID  0xFFFF
00167
00174 #define EMBER_MULTICAST_NODE_ID           0xFFFE
00175
00183 #define EMBER_UNKNOWN_NODE_ID             0xFFFD
00184
00192 #define EMBER_DISCOVERY_ACTIVE_NODE_ID    0xFFFC
00193
00198 #define EMBER_NULL_ADDRESS_TABLE_INDEX 0xFF
00199
00203 #define EMBER_ZDO_ENDPOINT 0
00204
00208 #define EMBER_BROADCAST_ENDPOINT 0xFF
```



```
00209
00213 #define EMBER_ZDO_PROFILE_ID  0x0000
00214
00215
00216 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00217 enum EmberLeaveRequestFlags
00218 #else
00219 typedef int8u EmberLeaveRequestFlags;
00220 enum
00221 #endif
00222 {
00224   EMBER_ZIGBEE_LEAVE_AND_REJOIN          = 0x20,
00225
00227   EMBER_ZIGBEE_LEAVE_AND_REMOVE_CHILDREN = 0x40,
00228 };
00229
00231
00232
00245 #define EMBER_BROADCAST_ADDRESS 0xFFFC
00246
00247 #define EMBER_RX_ON_WHEN_IDLE_BROADCAST_ADDRESS 0xFFFD
00248
00249 #define EMBER_SLEEPY_BROADCAST_ADDRESS 0xFFFF
00250
00258 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00259 enum EmberNodeType
00260 #else
00261 typedef int8u EmberNodeType;
00262 enum
00263 #endif
00264 {
00266   EMBER_UNKNOWN_DEVICE = 0,
00268   EMBER_COORDINATOR = 1,
00270   EMBER_ROUTER = 2,
00272   EMBER_END_DEVICE = 3,
00276   EMBER_SLEEPY_END_DEVICE = 4,
00278   EMBER_MOBILE_END_DEVICE = 5
00279 };
00280
00284 typedef struct {
00285   int16u panId;
00286   int8u channel;
00287   boolean allowingJoin;
00288   int8u extendedPanId[8];
00289   int8u stackProfile;
00290   int8u nwkUpdateId;
00291 } EmberZigbeeNetwork;
00292
00293
00300 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00301 enum EmberApsOption
00302 #else
00303 typedef int16u EmberApsOption;
00304 enum
00305 #endif
00306 {
00308   EMBER_APS_OPTION_NONE                     = 0x0000,
00320   EMBER_APS_OPTION_DSA_SIGN                 = 0x0010,
00323   EMBER_APS_OPTION_ENCRYPTION               = 0x0020,
00327   EMBER_APS_OPTION_RETRY                    = 0x0040,
00333   EMBER_APS_OPTION_ENABLE_ROUTE_DISCOVERY   = 0x0100,
00336   EMBER_APS_OPTION_FORCE_ROUTE_DISCOVERY    = 0x0200,
00338   EMBER_APS_OPTION_SOURCE_EUI64             = 0x0400,
00340   EMBER_APS_OPTION_DESTINATION_EUI64        = 0x0800,
00343   EMBER_APS_OPTION_ENABLE_ADDRESS_DISCOVERY = 0x1000,
00348   EMBER_APS_OPTION_POLL_RESPONSE            = 0x2000,
00353   EMBER_APS_OPTION_ZDO_RESPONSE_REQUIRED    = 0x4000,
00359   EMBER_APS_OPTION_FRAGMENT                 = SIGNED_ENUM 0x8000
00360 };
00361
00362
00363
00367 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00368 enum EmberIncomingMessageType
00369 #else
00370 typedef int8u EmberIncomingMessageType;
00371 enum
00372 #endif
00373 {
00375   EMBER_INCOMING_UNICAST,
00377   EMBER_INCOMING_UNICAST_REPLY,
```



```
00379   EMBER_INCOMING_MULTICAST,
00381   EMBER_INCOMING_MULTICAST_LOOPBACK,
00383   EMBER_INCOMING_BROADCAST,
00385   EMBER_INCOMING_BROADCAST_LOOPBACK
00386 };
00387
00388
00392 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00393 enum EmberOutgoingMessageType
00394 #else
00395 typedef int8u EmberOutgoingMessageType;
00396 enum
00397 #endif
00398 {
00400   EMBER_OUTGOING_DIRECT,
00402   EMBER_OUTGOING_VIA_ADDRESS_TABLE,
00404   EMBER_OUTGOING_VIA_BINDING,
00407   EMBER_OUTGOING_MULTICAST,
00410   EMBER_OUTGOING_BROADCAST
00411 };
00412
00413
00417 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00418 enum EmberNetworkStatus
00419 #else
00420 typedef int8u EmberNetworkStatus;
00421 enum
00422 #endif
00423 {
00425   EMBER_NO_NETWORK,
00427   EMBER_JOINING_NETWORK,
00429   EMBER_JOINED_NETWORK,
00432   EMBER_JOINED_NETWORK_NO_PARENT,
00434   EMBER_LEAVING_NETWORK
00435 };
00436
00437
00441 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00442 enum EmberNetworkScanType
00443 #else
00444 typedef int8u EmberNetworkScanType;
00445 enum
00446 #endif
00447 {
00449   EMBER_ENERGY_SCAN,
00451   EMBER_ACTIVE_SCAN
00452 };
00453
00454
00458 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00459 enum EmberBindingType
00460 #else
00461 typedef int8u EmberBindingType;
00462 enum
00463 #endif
00464 {
00466   EMBER_UNUSED_BINDING         = 0,
00468   EMBER_UNICAST_BINDING        = 1,
00472   EMBER_MANY_TO_ONE_BINDING    = 2,
00476   EMBER_MULTICAST_BINDING      = 3,
00477 };
00478
00479
00488 #define EMBER_LOW_RAM_CONCENTRATOR 0xFFF8
00489
00493 #define EMBER_HIGH_RAM_CONCENTRATOR 0xFFF9
00494
00496
00497
00501 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00502 enum EmberJoinDecision
00503 #else
00504 typedef int8u EmberJoinDecision;
00505 enum
00506 #endif
00507 {
00509   EMBER_USE_PRECONFIGURED_KEY = 0,
00511   EMBER_SEND_KEY_IN_THE_CLEAR,
00513   EMBER_DENY_JOIN,
00515   EMBER_NO_ACTION
00516 };
```



```
00517
00521 #define EMBER_JOIN_DECISION_STRINGS \
00522   "use preconfigured key",          \
00523   "send key in the clear",          \
00524   "deny join",                      \
00525   "no action",
00526
00527
00533 // These map to the actual values within the APS Command frame so they cannot
00534 // be arbitrarily changed.
00535 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00536 enum EmberDeviceUpdate
00537 #else
00538 typedef int8u EmberDeviceUpdate;
00539 enum
00540 #endif
00541 {
00542   EMBER_STANDARD_SECURITY_SECURED_REJOIN   = 0,
00543   EMBER_STANDARD_SECURITY_UNSECURED_JOIN   = 1,
00544   EMBER_DEVICE_LEFT                         = 2,
00545   EMBER_STANDARD_SECURITY_UNSECURED_REJOIN = 3,
00546   EMBER_HIGH_SECURITY_SECURED_REJOIN       = 4,
00547   EMBER_HIGH_SECURITY_UNSECURED_JOIN       = 5,
00548   /* 6 Reserved */
00549   EMBER_HIGH_SECURITY_UNSECURED_REJOIN     = 7,
00550   /* 8 - 15 Reserved */
00551 };
00552
00556 #define EMBER_DEVICE_UPDATE_STRINGS                                    \
00557     "secured rejoin",                                                  \
00558     "UNsecured join",                                                  \
00559     "device left",                                                     \
00560     "UNsecured rejoin",                                                \
00561     "high secured rejoin",                                             \
00562     "high UNsecured join",                                             \
00563     "RESERVED",                 /* reserved status code, per the spec. */ \
00564     "high UNsecured rejoin",
00565
00569 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00570 enum EmberClusterListId
00571 #else
00572 typedef int8u EmberClusterListId;
00573 enum
00574 #endif
00575 {
00577   EMBER_INPUT_CLUSTER_LIST            = 0,
00579   EMBER_OUTPUT_CLUSTER_LIST           = 1
00580 };
00581
00582
00587 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00588 enum EmberEventUnits
00589 #else
00590 typedef int8u EmberEventUnits;
00591 enum
00592 #endif
00593 {
00595   EMBER_EVENT_INACTIVE = 0,
00597   EMBER_EVENT_MS_TIME,
00600   EMBER_EVENT_QS_TIME,
00603   EMBER_EVENT_MINUTE_TIME,
00605   EMBER_EVENT_ZERO_DELAY
00606 };
00607
00608
00612 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00613 enum EmberJoinMethod
00614 #else
00615 typedef int8u EmberJoinMethod;
00616 enum
00617 #endif
00618 {
00624   EMBER_USE_MAC_ASSOCIATION         = 0,
00625
00636   EMBER_USE_NWK_REJOIN              = 1,
00637
00638
00639   /* For those networks where the "permit joining" flag is never turned
00640    * on, they will need to use a NWK Rejoin.  If those devices have been
00641    * preconfigured with the  NWK key (including sequence number) they can use
00642    * a secured rejoin.  This is only necessary for end devices since they need
```



```
00643    * a parent.  Routers can simply use the ::EMBER_USE_NWK_COMMISSIONING
00644    * join method below.
00645    */
00646   EMBER_USE_NWK_REJOIN_HAVE_NWK_KEY = 2,
00647
00652   EMBER_USE_NWK_COMMISSIONING       = 3,
00653 };
00654
00655
00662 typedef struct {
00664   int8u   extendedPanId[8];
00666   int16u  panId;
00668   int8s   radioTxPower;
00670   int8u   radioChannel;
00675   EmberJoinMethod joinMethod;
00676
00681   EmberNodeId nwkManagerId;
00687   int8u nwkUpdateId;
00693   int32u channels;
00694 } EmberNetworkParameters;
00695
00696
00697 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00698 #define emberInitializeNetworkParameters(parameters) \
00699   (MEMSET(parameters, 0, sizeof(EmberNetworkParameters)))
00700 #else
00701 void emberInitializeNetworkParameters(EmberNetworkParameters* parameters);
00702 #endif
00703
00707 typedef struct {
00709   int16u profileId;
00711   int16u clusterId;
00713   int8u sourceEndpoint;
00715   int8u destinationEndpoint;
00717   EmberApsOption options;
00719   int16u groupId;
00721   int8u sequence;
00722 } EmberApsFrame;
00723
00724
00731 typedef struct {
00733   EmberBindingType type;
00735   int8u local;
00743   int16u clusterId;
00745   int8u remote;
00750   EmberEUI64 identifier;
00751 } EmberBindingTableEntry;
00752
00753
00759 typedef struct {
00761   int16u shortId;
00764   int8u  averageLqi;
00767   int8u  inCost;
00774   int8u  outCost;
00780   int8u  age;
00782   EmberEUI64 longId;
00783 } EmberNeighborTableEntry;
00784
00790 typedef struct {
00792   int16u destination;
00794   int16u nextHop;
00797   int8u status;
00800   int8u age;
00803   int8u concentratorType;
00808   int8u routeRecordState;
00809 } EmberRouteTableEntry;
00810
00818 typedef struct {
00820   EmberMulticastId multicastId;
00824   int8u  endpoint;
00825 } EmberMulticastTableEntry;
00826
00831 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00832 enum EmberCounterType
00833 #else
00834 typedef int8u EmberCounterType;
00835 enum
00836 #endif
00837 {
00839   EMBER_COUNTER_MAC_RX_BROADCAST = 0,
00841   EMBER_COUNTER_MAC_TX_BROADCAST = 1,
```



```
00843   EMBER_COUNTER_MAC_RX_UNICAST = 2,
00845   EMBER_COUNTER_MAC_TX_UNICAST_SUCCESS = 3,
00851   EMBER_COUNTER_MAC_TX_UNICAST_RETRY = 4,
00853   EMBER_COUNTER_MAC_TX_UNICAST_FAILED = 5,
00854
00856   EMBER_COUNTER_APS_DATA_RX_BROADCAST = 6,
00858   EMBER_COUNTER_APS_DATA_TX_BROADCAST = 7,
00860   EMBER_COUNTER_APS_DATA_RX_UNICAST = 8,
00862   EMBER_COUNTER_APS_DATA_TX_UNICAST_SUCCESS = 9,
00868   EMBER_COUNTER_APS_DATA_TX_UNICAST_RETRY = 10,
00870   EMBER_COUNTER_APS_DATA_TX_UNICAST_FAILED = 11,
00871
00874   EMBER_COUNTER_ROUTE_DISCOVERY_INITIATED = 12,
00875
00877   EMBER_COUNTER_NEIGHBOR_ADDED = 13,
00879   EMBER_COUNTER_NEIGHBOR_REMOVED = 14,
00881   EMBER_COUNTER_NEIGHBOR_STALE = 15,
00882
00884   EMBER_COUNTER_JOIN_INDICATION = 16,
00886   EMBER_COUNTER_CHILD_REMOVED = 17,
00887
00889   EMBER_COUNTER_ASH_OVERFLOW_ERROR = 18,
00891   EMBER_COUNTER_ASH_FRAMING_ERROR = 19,
00893   EMBER_COUNTER_ASH_OVERRUN_ERROR = 20,
00894
00897   EMBER_COUNTER_NWK_FRAME_COUNTER_FAILURE = 21,
00898
00901   EMBER_COUNTER_APS_FRAME_COUNTER_FAILURE = 22,
00902
00904   EMBER_COUNTER_ASH_XOFF = 23,
00905
00909   EMBER_COUNTER_APS_LINK_KEY_NOT_AUTHORIZED = 24,
00910
00913   EMBER_COUNTER_NWK_DECRYPTION_FAILURE = 25,
00914
00917   EMBER_COUNTER_APS_DECRYPTION_FAILURE = 26,
00918
00923   EMBER_COUNTER_ALLOCATE_PACKET_BUFFER_FAILURE = 27,
00924
00926   EMBER_COUNTER_RELAYED_UNICAST = 28,
00927
00939   EMBER_COUNTER_PHY_TO_MAC_QUEUE_LIMIT_REACHED = 29,
00940
00942   EMBER_COUNTER_TYPE_COUNT = 30
00943 };
00944
00948 #define EMBER_COUNTER_STRINGS                    \
00949     "Mac Rx Bcast",                              \
00950     "Mac Tx Bcast",                              \
00951     "Mac Rx Ucast",                              \
00952     "Mac Tx Ucast",                              \
00953     "Mac Tx Ucast Retry",                        \
00954     "Mac Tx Ucast Fail",                         \
00955     "APS Rx Bcast",                              \
00956     "APS Tx Bcast",                              \
00957     "APS Rx Ucast",                              \
00958     "APS Tx Ucast Success",                      \
00959     "APS Tx Ucast Retry",                        \
00960     "APS Tx Ucast Fail",                         \
00961     "Route Disc Initiated",                      \
00962     "Neighbor Added",                            \
00963     "Neighbor Removed",                          \
00964     "Neighbor Stale",                            \
00965     "Join Indication",                           \
00966     "Child Moved",                               \
00967     "ASH Overflow",                              \
00968     "ASH Frame Error",                           \
00969     "ASH Overrun Error",                         \
00970     "NWK FC Failure",                            \
00971     "APS FC Failure",                            \
00972     "ASH XOff",                                  \
00973     "APS Unauthorized Key",                      \
00974     "NWK Decrypt Failures",                      \
00975     "APS Decrypt Failures",                      \
00976     "Packet Buffer Allocate Failures",           \
00977     "Relayed Ucast",                             \
00978     "Phy to MAC queue limit reached",            \
00979     NULL
00980
00982 typedef int8u EmberTaskId;
00983
```



```
00984 #ifndef EZSP_HOST
00985
00991   typedef struct {
00993     EmberEventUnits status;
00995     EmberTaskId taskid;
00999     int32u timeToExecute;
01000   } EmberEventControl;
01001 #else
01002   // host applications use an older, basic form of the event system
01009   typedef struct {
01011     EmberEventUnits status;
01015     int16u timeToExecute;
01016   } EmberEventControl;
01017 #endif
01018
01026 typedef PGM struct {
01028   EmberEventControl *control;
01030   void (*handler)(void);
01031 } EmberEventData;
01032
01037 typedef struct {
01038   // The time when the next event associated with this task will fire
01039   int32u nextEventTime;
01040   // The list of events associated with this task
01041   EmberEventData *events;
01042   // A flag that indicates the task has something to do other than events
01043   boolean busy;
01044 } EmberTaskControl;
01045
01050
01055 #define EMBER_TX_POWER_MODE_DEFAULT             0x0000
01056
01059 #define EMBER_TX_POWER_MODE_BOOST               0x0001
01060
01064 #define EMBER_TX_POWER_MODE_ALTERNATE           0x0002
01065
01069 #define EMBER_TX_POWER_MODE_BOOST_AND_ALTERNATE (EMBER_TX_POWER_MODE_BOOST     \
01070                                                 |EMBER_TX_POWER_MODE_ALTERNATE)
01071 #ifndef DOXYGEN_SHOULD_SKIP_THIS
01072 // The application does not ever need to call emberSetTxPowerMode() with the
01073 // txPowerMode parameter set to this value.  This value is used internally by
01074 // the stack to indicate that the default token configuration has not been
01075 // overridden by a prior call to emberSetTxPowerMode().
01076 #define EMBER_TX_POWER_MODE_USE_TOKEN           0x8000
01077 #endif//DOXYGEN_SHOULD_SKIP_THIS
01078
01080
01085
01093 #define EMBER_PRIVATE_PROFILE_ID  0xC00E
01094
01133 #define EMBER_BROADCAST_ALARM_CLUSTER       0x0000
01134
01171 #define EMBER_UNICAST_ALARM_CLUSTER         0x0001
01172
01188 #define EMBER_CACHED_UNICAST_ALARM_CLUSTER  0x0002
01189
01193 #define EMBER_REPORT_COUNTERS_REQUEST 0x0003
01194
01196 #define EMBER_REPORT_COUNTERS_RESPONSE 0x8003
01197
01202 #define EMBER_REPORT_AND_CLEAR_COUNTERS_REQUEST 0x0004
01203
01205 #define EMBER_REPORT_AND_CLEAR_COUNTERS_RESPONSE 0x8004
01206
01211 #define EMBER_OTA_CERTIFICATE_UPGRADE_CLUSTER 0x0005
01212
01214
01215
01218 typedef struct {
01220   int8u contents[EMBER_ENCRYPTION_KEY_SIZE];
01221 } EmberKeyData;
01222
01225 typedef struct {
01226   /* This is the certificate byte data. */
01227   int8u contents[EMBER_CERTIFICATE_SIZE];
01228 } EmberCertificateData;
01229
01232 typedef struct {
01233   int8u contents[EMBER_PUBLIC_KEY_SIZE];
01234 } EmberPublicKeyData;
01235
```



```
01238 typedef struct {
01239   int8u contents[EMBER_PRIVATE_KEY_SIZE];
01240 } EmberPrivateKeyData;
01241
01244 typedef struct {
01245   int8u contents[EMBER_SMAC_SIZE];
01246 } EmberSmacData;
01247
01251 typedef struct {
01252   int8u contents[EMBER_SIGNATURE_SIZE];
01253 } EmberSignatureData;
01254
01257 typedef struct {
01258   int8u contents[EMBER_AES_HASH_BLOCK_SIZE];
01259 } EmberMessageDigest;
01260
01264 typedef struct {
01265   int8u result[EMBER_AES_HASH_BLOCK_SIZE];
01266   int32u length;
01267 } EmberAesMmoHashContext;
01268
01269
01275 #define EMBER_STANDARD_SECURITY_MODE 0x0000
01276
01280 #define EMBER_TRUST_CENTER_NODE_ID 0x0000
01281
01282
01286 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01287 enum EmberInitialSecurityBitmask
01288 #else
01289 typedef int16u EmberInitialSecurityBitmask;
01290 enum
01291 #endif
01292 {
01295   EMBER_DISTRIBUTED_TRUST_CENTER_MODE       = 0x0002,
01298   EMBER_GLOBAL_LINK_KEY                     = 0x0004,
01301   EMBER_PRECONFIGURED_NETWORK_KEY_MODE      = 0x0008,
01302
01303 #if !defined DOXYGEN_SHOULD_SKIP_THIS
01304   // Hidden fields used internally.
01305   EMBER_HAVE_TRUST_CENTER_UNKNOWN_KEY_TOKEN = 0x0010,
01306   EMBER_HAVE_TRUST_CENTER_LINK_KEY_TOKEN    = 0x0020,
01307   EMBER_HAVE_TRUST_CENTER_MASTER_KEY_TOKEN  = 0x0030,
01308 #endif
01309
01319   EMBER_HAVE_TRUST_CENTER_EUI64             = 0x0040,
01320
01327   EMBER_TRUST_CENTER_USES_HASHED_LINK_KEY   = 0x0084,
01328
01332   EMBER_HAVE_PRECONFIGURED_KEY              = 0x0100,
01336   EMBER_HAVE_NETWORK_KEY                    = 0x0200,
01341   EMBER_GET_LINK_KEY_WHEN_JOINING           = 0x0400,
01347   EMBER_REQUIRE_ENCRYPTED_KEY               = 0x0800,
01355   EMBER_NO_FRAME_COUNTER_RESET              = 0x1000,
01361   EMBER_GET_PRECONFIGURED_KEY_FROM_INSTALL_CODE = 0x2000,
01362
01363 #if !defined DOXYGEN_SHOULD_SKIP_THIS
01364   // Internal data
01365   EM_SAVED_IN_TOKEN                         = 0x4000,
01366   #define EM_SECURITY_INITIALIZED           0x00008000L
01367
01368   // This is only used internally.  High security is not released or supported
01369   // except for golden unit compliance.
01370   #define EMBER_HIGH_SECURITY_MODE          0x0001
01371 #else
01372   /* All other bits are reserved and must be zero. */
01373 #endif
01374 };
01375
01378 #define EMBER_NO_TRUST_CENTER_MODE   EMBER_DISTRIBUTED_TRUST_CENTER_MODE
01379
01380 #if !defined DOXYGEN_SHOULD_SKIP_THIS
01381   #define NO_TRUST_CENTER_KEY_TOKEN        0x0000
01382   #define TRUST_CENTER_KEY_TOKEN_MASK      0x0030
01383   #define SECURITY_BIT_TOKEN_MASK          0x01FF
01384
01385   // This is negative logic to support all devices currently in the field
01386   // without this bit set.
01387   #define KEY_IS_NOT_AUTHORIZED            0x00010000L // RAM bitmask value
01388
01389   #define SECURITY LOWER BIT MASK          0x000000FF  // ""
```



```
01390   #define SECURITY_UPPER_BIT_MASK          0x00FF0000L // ""
01391 #endif
01392
01395 typedef struct {
01400   int16u bitmask;
01409   EmberKeyData preconfiguredKey;
01415   EmberKeyData networkKey;
01422   int8u networkKeySequenceNumber;
01430   EmberEUI64 preconfiguredTrustCenterEui64;
01431 } EmberInitialSecurityState;
01432
01433
01437 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01438 enum EmberCurrentSecurityBitmask
01439 #else
01440 typedef int16u EmberCurrentSecurityBitmask;
01441 enum
01442 #endif
01443 {
01444 #if defined DOXYGEN_SHOULD_SKIP_THIS
01445   // These options are the same for Initial and Current Security state
01446
01449   EMBER_STANDARD_SECURITY_MODE_             = 0x0000,
01452   EMBER_DISTRIBUTED_TRUST_CENTER_MODE_      = 0x0002,
01455   EMBER_GLOBAL_LINK_KEY_                    = 0x0004,
01456 #else
01457   // Bit 3 reserved
01458 #endif
01459
01460   EMBER_HAVE_TRUST_CENTER_LINK_KEY          = 0x0010,
01461
01463   EMBER_TRUST_CENTER_USES_HASHED_LINK_KEY_   = 0x0084,
01464
01465   // Bits 1,5,6, 8-15 reserved
01466 };
01467
01468 #if !defined DOXYGEN_SHOULD_SKIP_THIS
01469   #define INITIAL_AND_CURRENT_BITMASK         0x00FF
01470 #endif
01471
01472
01475 typedef struct {
01478   EmberCurrentSecurityBitmask bitmask;
01482   EmberEUI64 trustCenterLongAddress;
01483 } EmberCurrentSecurityState;
01484
01485
01489 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01490 enum EmberKeyStructBitmask
01491 #else
01492 typedef int16u EmberKeyStructBitmask;
01493 enum
01494 #endif
01495 {
01498   EMBER_KEY_HAS_SEQUENCE_NUMBER         = 0x0001,
01502   EMBER_KEY_HAS_OUTGOING_FRAME_COUNTER = 0x0002,
01506   EMBER_KEY_HAS_INCOMING_FRAME_COUNTER = 0x0004,
01510   EMBER_KEY_HAS_PARTNER_EUI64           = 0x0008,
01514   EMBER_KEY_IS_AUTHORIZED               = 0x0010,
01519   EMBER_KEY_PARTNER_IS_SLEEPY           = 0x0020,
01520
01521 };
01522
01524 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01525 enum EmberKeyType
01526 #else
01527 typedef int8u EmberKeyType;
01528 enum
01529 #endif
01530 {
01532   EMBER_TRUST_CENTER_LINK_KEY         = 1,
01534   EMBER_TRUST_CENTER_MASTER_KEY       = 2,
01536   EMBER_CURRENT_NETWORK_KEY           = 3,
01538   EMBER_NEXT_NETWORK_KEY              = 4,
01540   EMBER_APPLICATION_LINK_KEY          = 5,
01542   EMBER_APPLICATION_MASTER_KEY        = 6,
01543 };
01544
01548 typedef struct {
01551   EmberKeyStructBitmask bitmask;
01553   EmberKeyType type;
```



```
01555   EmberKeyData key;
01558   int32u outgoingFrameCounter;
01561   int32u incomingFrameCounter;
01564   int8u sequenceNumber;
01567   EmberEUI64 partnerEUI64;
01568 } EmberKeyStruct;
01569
01570
01574 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01575 enum EmberKeyStatus
01576 #else
01577 typedef int8u EmberKeyStatus;
01578 enum
01579 #endif
01580 {
01581   EMBER_APP_LINK_KEY_ESTABLISHED           = 1,
01582   EMBER_APP_MASTER_KEY_ESTABLISHED         = 2,
01583   EMBER_TRUST_CENTER_LINK_KEY_ESTABLISHED  = 3,
01584   EMBER_KEY_ESTABLISHMENT_TIMEOUT          = 4,
01585   EMBER_KEY_TABLE_FULL                      = 5,
01586
01587   // These are success status values applying only to the
01588   // Trust Center answering key requests
01589   EMBER_TC_RESPONDED_TO_KEY_REQUEST        = 6,
01590   EMBER_TC_APP_KEY_SENT_TO_REQUESTER       = 7,
01591
01592   // These are failure status values applying only to the
01593   // Trust Center answering key requests
01594   EMBER_TC_RESPONSE_TO_KEY_REQUEST_FAILED  = 8,
01595   EMBER_TC_REQUEST_KEY_TYPE_NOT_SUPPORTED  = 9,
01596   EMBER_TC_NO_LINK_KEY_FOR_REQUESTER       = 10,
01597   EMBER_TC_REQUESTER_EUI64_UNKNOWN         = 11,
01598   EMBER_TC_RECEIVED_FIRST_APP_KEY_REQUEST  = 12,
01599   EMBER_TC_TIMEOUT_WAITING_FOR_SECOND_APP_KEY_REQUEST = 13,
01600   EMBER_TC_NON_MATCHING_APP_KEY_REQUEST_RECEIVED      = 14,
01601   EMBER_TC_FAILED_TO_SEND_APP_KEYS         = 15,
01602   EMBER_TC_FAILED_TO_STORE_APP_KEY_REQUEST = 16,
01603   EMBER_TC_REJECTED_APP_KEY_REQUEST        = 17,
01604 };
01605
01609 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01610 enum EmberLinkKeyRequestPolicy
01611 #else
01612 typedef int8u EmberLinkKeyRequestPolicy;
01613 enum
01614 #endif
01615 {
01616   EMBER_DENY_KEY_REQUESTS  = 0x00,
01617   EMBER_ALLOW_KEY_REQUESTS = 0x01,
01618 };
01619
01620
01628 #if defined DOXYGEN_SHOULD_SKIP_THIS
01629 int8u* emberKeyContents(EmberKeyData* key);
01630 #else
01631 #define emberKeyContents(key) ((key)->contents)
01632 #endif
01633
01641 #if defined DOXYGEN_SHOULD_SKIP_THIS
01642 int8u* emberCertificateContents(EmberCertificateData* cert);
01643 #else
01644 #define emberCertificateContents(cert) ((cert)->contents)
01645 #endif
01646
01654 #if defined DOXYGEN_SHOULD_SKIP_THIS
01655 int8u* emberPublicKeyContents(EmberPublicKeyData* key);
01656 #else
01657 #define emberPublicKeyContents(key) ((key)->contents)
01658 #endif
01659
01667 #if defined DOXYGEN_SHOULD_SKIP_THIS
01668 int8u* emberPrivateKeyContents(EmberPrivateKeyData* key);
01669 #else
01670 #define emberPrivateKeyContents(key) ((key)->contents)
01671 #endif
01672
01677 #if defined DOXYGEN_SHOULD_SKIP_THIS
01678 int8u* emberSmacContents(EmberSmacData* key);
01679 #else
01680 #define emberSmacContents(key) ((key)->contents)
01681 #endif
```



```
01682
01686 #if defined DOXYGEN_SHOULD_SKIP_THIS
01687 int8u* emberSignatureContents(EmberSignatureData* sig);
01688 #else
01689 #define emberSignatureContents(sig) ((sig)->contents)
01690 #endif
01691
01696 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01697 enum EmberMacPassthroughType
01698 #else
01699 typedef int8u EmberMacPassthroughType;
01700 enum
01701 #endif
01702 {
01704   EMBER_MAC_PASSTHROUGH_NONE             = 0x00,
01706   EMBER_MAC_PASSTHROUGH_SE_INTERPAN      = 0x01,
01708   EMBER_MAC_PASSTHROUGH_EMBERNET         = 0x02,
01710   EMBER_MAC_PASSTHROUGH_EMBERNET_SOURCE  = 0x04,
01712   EMBER_MAC_PASSTHROUGH_APPLICATION      = 0x08,
01714   EMBER_MAC_PASSTHROUGH_CUSTOM           = 0x10,
01715
01716 #if !defined DOXYGEN_SHOULD_SKIP_THIS
01717
01718   EM_MAC_PASSTHROUGH_INTERNAL            = 0x80
01719 #endif
01720 };
01721
01726 typedef int16u EmberMacFilterMatchData;
01727
01728 #define EMBER_MAC_FILTER_MATCH_ENABLED_MASK             0x0001
01729 #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_MASK         0x0003
01730 #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_MASK       0x000C
01731 #define EMBER_MAC_FILTER_MATCH_ON_DEST_MASK             0x0030
01732 #define EMBER_MAC_FILTER_MATCH_ON_SOURCE_MASK           0x0080
01733
01734 // Globally turn on/off this filter
01735 #define EMBER_MAC_FILTER_MATCH_ENABLED                  0x0000
01736 #define EMBER_MAC_FILTER_MATCH_DISABLED                 0x0001
01737
01738 // Pick either one of these
01739 #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_NONE         0x0000
01740 #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_LOCAL        0x0001
01741 #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_BROADCAST    0x0002
01742
01743 // and one of these
01744 #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_NONE       0x0000
01745 #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_NON_LOCAL  0x0004
01746 #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_LOCAL      0x0008
01747
01748 // and one of these
01749 #define EMBER_MAC_FILTER_MATCH_ON_DEST_BROADCAST_SHORT  0x0000
01750 #define EMBER_MAC_FILTER_MATCH_ON_DEST_UNICAST_SHORT    0x0010
01751 #define EMBER_MAC_FILTER_MATCH_ON_DEST_UNICAST_LONG     0x0020
01752
01753 // and one of these
01754 #define EMBER_MAC_FILTER_MATCH_ON_SOURCE_LONG           0x0000
01755 #define EMBER_MAC_FILTER_MATCH_ON_SOURCE_SHORT          0x0080
01756
01757 // Last entry should set this and nothing else.  No other bits will be examined.
01758 #define EMBER_MAC_FILTER_MATCH_END                      0x8000
01759
01763 typedef struct {
01764   int8u filterIndexMatch;
01765   EmberMacPassthroughType legacyPassthroughType;
01766   EmberMessageBuffer message;
01767 } EmberMacFilterMatchStruct;
01768
01769
01773 typedef int8u EmberLibraryStatus;
01774
01779
01785 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01786 enum EmberZdoStatus
01787 #else
01788 typedef int8u EmberZdoStatus;
01789 enum
01790 #endif
01791 {
01792   // These values are taken from Table 48 of ZDP Errata 043238r003 and Table 2
01793   // of NWK 02130r10.
01794   EMBER_ZDP_SUCCESS               = 0x00,
```



```
01795   // 0x01 to 0x7F are reserved
01796   EMBER_ZDP_INVALID_REQUEST_TYPE = 0x80,
01797   EMBER_ZDP_DEVICE_NOT_FOUND     = 0x81,
01798   EMBER_ZDP_INVALID_ENDPOINT     = 0x82,
01799   EMBER_ZDP_NOT_ACTIVE           = 0x83,
01800   EMBER_ZDP_NOT_SUPPORTED        = 0x84,
01801   EMBER_ZDP_TIMEOUT              = 0x85,
01802   EMBER_ZDP_NO_MATCH             = 0x86,
01803   // 0x87 is reserved            = 0x87,
01804   EMBER_ZDP_NO_ENTRY             = 0x88,
01805   EMBER_ZDP_NO_DESCRIPTOR        = 0x89,
01806   EMBER_ZDP_INSUFFICIENT_SPACE   = 0x8a,
01807   EMBER_ZDP_NOT_PERMITTED        = 0x8b,
01808   EMBER_ZDP_TABLE_FULL           = 0x8c,
01809   EMBER_ZDP_NOT_AUTHORIZED       = 0x8d,
01810
01811   EMBER_NWK_ALREADY_PRESENT      = 0xC5,
01812   EMBER_NWK_TABLE_FULL           = 0xC7,
01813   EMBER_NWK_UNKNOWN_DEVICE       = 0xC8
01814 };
01815
01828
01829
01830
01831
01832
01833
01834
01835
01836
01837
01838
01839
01840
01841
01842 #define NETWORK_ADDRESS_REQUEST      0x0000
01843 #define NETWORK_ADDRESS_RESPONSE     0x8000
01844 #define IEEE_ADDRESS_REQUEST         0x0001
01845 #define IEEE_ADDRESS_RESPONSE        0x8001
01847
01854   //            <node descriptor: 13>
01855   //
01856   //  Node Descriptor field is divided into subfields of bitmasks as follows:
01857   //      (Note: All lengths below are given in bits rather than bytes.)
01858   //            Logical Type:                    3
01859   //            Complex Descriptor Available:    1
01860   //            User Descriptor Available:       1
01861   //            (reserved/unused):               3
01862   //            APS Flags:                       3
01863   //            Frequency Band:                  5
01864   //            MAC capability flags:            8
01865   //            Manufacturer Code:              16
01866   //            Maximum buffer size:             8
01867   //            Maximum incoming transfer size: 16
01868   //            Server mask:                    16
01869   //            Maximum outgoing transfer size: 16
01870   //            Descriptor Capability Flags:     8
01871   //     See ZigBee document 053474, Section 2.3.2.3 for more details.
01873 #define NODE_DESCRIPTOR_REQUEST      0x0002
01874 #define NODE_DESCRIPTOR_RESPONSE     0x8002
01876
01885   //     See ZigBee document 053474, Section 2.3.2.4 for more details.
01887 #define POWER_DESCRIPTOR_REQUEST     0x0003
01888 #define POWER_DESCRIPTOR_RESPONSE    0x8003
01890
01904 #define SIMPLE_DESCRIPTOR_REQUEST    0x0004
01905 #define SIMPLE_DESCRIPTOR_RESPONSE   0x8004
01907
01916 #define ACTIVE_ENDPOINTS_REQUEST     0x0005
01917 #define ACTIVE_ENDPOINTS_RESPONSE    0x8005
01919
01931 #define MATCH_DESCRIPTORS_REQUEST    0x0006
01932 #define MATCH_DESCRIPTORS_RESPONSE   0x8006
01934
01944 #define DISCOVERY_CACHE_REQUEST      0x0012
01945 #define DISCOVERY_CACHE_RESPONSE     0x8012
01947
01956 #define END_DEVICE_ANNOUNCE          0x0013
01957 #define END_DEVICE_ANNOUNCE_RESPONSE 0x8013
01959
01971 #define SYSTEM SERVER DISCOVERY REQUEST  0x0015
```



```
01972 #define SYSTEM_SERVER_DISCOVERY_RESPONSE 0x8015
01974
01979 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01980 enum EmberZdoServerMask
01981 #else
01982 typedef int16u EmberZdoServerMask;
01983 enum
01984 #endif
01985 {
01986   EMBER_ZDP_PRIMARY_TRUST_CENTER          = 0x0001,
01987   EMBER_ZDP_SECONDARY_TRUST_CENTER        = 0x0002,
01988   EMBER_ZDP_PRIMARY_BINDING_TABLE_CACHE   = 0x0004,
01989   EMBER_ZDP_SECONDARY_BINDING_TABLE_CACHE = 0x0008,
01990   EMBER_ZDP_PRIMARY_DISCOVERY_CACHE       = 0x0010,
01991   EMBER_ZDP_SECONDARY_DISCOVERY_CACHE     = 0x0020,
01992   EMBER_ZDP_NETWORK_MANAGER               = 0x0040,
01993   // Bits 0x0080 to 0x8000 are reserved.
01994 };
01995
02009 #define FIND_NODE_CACHE_REQUEST         0x001C
02010 #define FIND_NODE_CACHE_RESPONSE        0x801C
02012
02023 #define END_DEVICE_BIND_REQUEST      0x0020
02024 #define END_DEVICE_BIND_RESPONSE     0x8020
02026
02044 #define UNICAST_BINDING              0x03
02045 #define UNICAST_MANY_TO_ONE_BINDING 0x83
02046 #define MULTICAST_BINDING            0x01
02047
02048 #define BIND_REQUEST                  0x0021
02049 #define BIND_RESPONSE                 0x8021
02050 #define UNBIND_REQUEST                0x0022
02051 #define UNBIND_RESPONSE               0x8022
02053
02101 #define LQI_TABLE_REQUEST             0x0031
02102 #define LQI_TABLE_RESPONSE            0x8031
02104
02137 #define ROUTING_TABLE_REQUEST         0x0032
02138 #define ROUTING_TABLE_RESPONSE        0x8032
02140
02159 #define BINDING_TABLE_REQUEST         0x0033
02160 #define BINDING_TABLE_RESPONSE        0x8033
02162
02173 #define LEAVE_REQUEST                 0x0034
02174 #define LEAVE_RESPONSE                0x8034
02175
02176 #define LEAVE_REQUEST_REMOVE_CHILDREN_FLAG 0x40
02177 #define LEAVE_REQUEST_REJOIN_FLAG          0x80
02179
02188 #define PERMIT_JOINING_REQUEST        0x0036
02189 #define PERMIT_JOINING_RESPONSE       0x8036
02191
02217 #define NWK_UPDATE_REQUEST            0x0038
02218 #define NWK_UPDATE_RESPONSE           0x8038
02220
02224 #define COMPLEX_DESCRIPTOR_REQUEST   0x0010
02225 #define COMPLEX_DESCRIPTOR_RESPONSE  0x8010
02226 #define USER_DESCRIPTOR_REQUEST       0x0011
02227 #define USER_DESCRIPTOR_RESPONSE      0x8011
02228 #define DISCOVERY_REGISTER_REQUEST   0x0012
02229 #define DISCOVERY_REGISTER_RESPONSE  0x8012
02230 #define USER_DESCRIPTOR_SET           0x0014
02231 #define USER_DESCRIPTOR_CONFIRM       0x8014
02232 #define NETWORK_DISCOVERY_REQUEST     0x0030
02233 #define NETWORK_DISCOVERY_RESPONSE   0x8030
02234 #define DIRECT_JOIN_REQUEST           0x0035
02235 #define DIRECT_JOIN_RESPONSE          0x8035
02236
02237
02238 #define CLUSTER_ID_RESPONSE_MINIMUM  0x8000
02240
02241
02253 #ifdef DOXYGEN_SHOULD_SKIP_THIS
02254 enum EmberZdoConfigurationFlags
02255 #else
02256 typedef int8u EmberZdoConfigurationFlags;
02257 enum
02258 #endif
02259
02260 {
02261  EMBER_APP_RECEIVES_SUPPORTED_ZDO_REQUESTS  = 0x01,
```



```
02262  EMBER_APP_HANDLES_UNSUPPORTED_ZDO_REQUESTS = 0x02,
02263  EMBER_APP_HANDLES_ZDO_ENDPOINT_REQUESTS    = 0x04,
02264  EMBER_APP_HANDLES_ZDO_BINDING_REQUESTS     = 0x08
02265 };
02266
02268
02269 #endif // EMBER_TYPES_H
02270
```


error-def.h File Reference

Return-code definitions for EmberZNet stack API functions. More...

Go to the source code of this file.

Generic Messages

These messages are system wide.

#define **EMBER_SUCCESS**(x00)
#define **EMBER_ERR_FATAL**(x01)
#define **EMBER_BAD_ARGUMENT**(x02)
#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_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)

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)

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)

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)

Detailed Description

Return-code definitions for EmberZNet stack API functions.

See **Ember Status Codes** for documentation.

Definition in file **error-def.h**.

error-def.h

Go to the documentation of this file.

```
00001
00038
00039 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00040
00043 #define EMBER_SUCCESS(0x00)
00044 #else
00045 DEFINE_ERROR(SUCCESS, 0)
00046 #endif //DOXYGEN_SHOULD_SKIP_THIS
00047
00048
00049 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00050
00053 #define EMBER_ERR_FATAL(0x01)
00054 #else
00055 DEFINE_ERROR(ERR_FATAL, 0x01)
00056 #endif //DOXYGEN_SHOULD_SKIP_THIS
00057
00058
00059 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00060
00063 #define EMBER_BAD_ARGUMENT(0x02)
00064 #else
00065 DEFINE_ERROR(BAD_ARGUMENT, 0x02)
00066 #endif //DOXYGEN_SHOULD_SKIP_THIS
00067
00068
00069 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00070
00074 #define EMBER_EEPROM_MFG_STACK_VERSION_MISMATCH(0x04)
00075 #else
00076 DEFINE_ERROR(EEPROM_MFG_STACK_VERSION_MISMATCH, 0x04)
00077 #endif //DOXYGEN_SHOULD_SKIP_THIS
00078
00079
00080 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00081
00085 #define EMBER_INCOMPATIBLE_STATIC_MEMORY_DEFINITIONS(0x05)
00086 #else
00087 DEFINE_ERROR(INCOMPATIBLE_STATIC_MEMORY_DEFINITIONS, 0x05)
00088 #endif //DOXYGEN_SHOULD_SKIP_THIS
00089
00090
00091 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00092
00096 #define EMBER_EEPROM_MFG_VERSION_MISMATCH(0x06)
00097 #else
00098 DEFINE_ERROR(EEPROM_MFG_VERSION_MISMATCH, 0x06)
00099 #endif //DOXYGEN_SHOULD_SKIP_THIS
00100
00101
00102 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00103
00107 #define EMBER_EEPROM_STACK_VERSION_MISMATCH(0x07)
00108 #else
00109 DEFINE_ERROR(EEPROM_STACK_VERSION_MISMATCH, 0x07)
00110 #endif //DOXYGEN_SHOULD_SKIP_THIS
00111
00113
00114
00119
00120 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00121
00124 #define EMBER_NO_BUFFERS(0x18)
00125 #else
00126 DEFINE_ERROR(NO_BUFFERS, 0x18)
00127 #endif //DOXYGEN_SHOULD_SKIP_THIS
00128
00130
00135
00136 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00137
00140 #define EMBER_SERIAL_INVALID_BAUD_RATE(0x20)
00141 #else
```



```
00142 DEFINE_ERROR(SERIAL_INVALID_BAUD_RATE, 0x20)
00143 #endif //DOXYGEN_SHOULD_SKIP_THIS
00144
00145
00146 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00147
00150 #define EMBER_SERIAL_INVALID_PORT(0x21)
00151 #else
00152 DEFINE_ERROR(SERIAL_INVALID_PORT, 0x21)
00153 #endif //DOXYGEN_SHOULD_SKIP_THIS
00154
00155
00156 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00157
00160 #define EMBER_SERIAL_TX_OVERFLOW(0x22)
00161 #else
00162 DEFINE_ERROR(SERIAL_TX_OVERFLOW, 0x22)
00163 #endif //DOXYGEN_SHOULD_SKIP_THIS
00164
00165
00166 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00167
00171 #define EMBER_SERIAL_RX_OVERFLOW(0x23)
00172 #else
00173 DEFINE_ERROR(SERIAL_RX_OVERFLOW, 0x23)
00174 #endif //DOXYGEN_SHOULD_SKIP_THIS
00175
00176
00177 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00178
00181 #define EMBER_SERIAL_RX_FRAME_ERROR(0x24)
00182 #else
00183 DEFINE_ERROR(SERIAL_RX_FRAME_ERROR, 0x24)
00184 #endif //DOXYGEN_SHOULD_SKIP_THIS
00185
00186
00187 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00188
00191 #define EMBER_SERIAL_RX_PARITY_ERROR(0x25)
00192 #else
00193 DEFINE_ERROR(SERIAL_RX_PARITY_ERROR, 0x25)
00194 #endif //DOXYGEN_SHOULD_SKIP_THIS
00195
00196
00197 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00198
00201 #define EMBER_SERIAL_RX_EMPTY(0x26)
00202 #else
00203 DEFINE_ERROR(SERIAL_RX_EMPTY, 0x26)
00204 #endif //DOXYGEN_SHOULD_SKIP_THIS
00205
00206
00207 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00208
00212 #define EMBER_SERIAL_RX_OVERRUN_ERROR(0x27)
00213 #else
00214 DEFINE_ERROR(SERIAL_RX_OVERRUN_ERROR, 0x27)
00215 #endif //DOXYGEN_SHOULD_SKIP_THIS
00216
00218
00223
00224 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00225
00228 #define EMBER_MAC_TRANSMIT_QUEUE_FULL(0x39)
00229 #else
00230 // Internal
00231 DEFINE_ERROR(MAC_TRANSMIT_QUEUE_FULL, 0x39)
00232 #endif //DOXYGEN_SHOULD_SKIP_THIS
00233
00234
00235 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00236
00239 #define EMBER_MAC_UNKNOWN_HEADER_TYPE(0x3A)
00240 #else
00241 DEFINE_ERROR(MAC_UNKNOWN_HEADER_TYPE, 0x3A)
00242 #endif //DOXYGEN_SHOULD_SKIP_THIS
00243
00244 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00245
00248 #define EMBER_MAC_ACK_HEADER_TYPE(0x3B)
00249 #else
```



```
00250 DEFINE_ERROR(MAC_ACK_HEADER_TYPE,  0x3B)
00251 #endif //DOXYGEN_SHOULD_SKIP_THIS
00252
00253
00254
00255 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00256
00259 #define EMBER_MAC_SCANNING(0x3D)
00260 #else
00261 DEFINE_ERROR(MAC_SCANNING, 0x3D)
00262 #endif //DOXYGEN_SHOULD_SKIP_THIS
00263
00264
00265 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00266
00269 #define EMBER_MAC_NO_DATA(0x31)
00270 #else
00271 DEFINE_ERROR(MAC_NO_DATA, 0x31)
00272 #endif //DOXYGEN_SHOULD_SKIP_THIS
00273
00274
00275 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00276
00279 #define EMBER_MAC_JOINED_NETWORK(0x32)
00280 #else
00281 DEFINE_ERROR(MAC_JOINED_NETWORK, 0x32)
00282 #endif //DOXYGEN_SHOULD_SKIP_THIS
00283
00284
00285 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00286
00290 #define EMBER_MAC_BAD_SCAN_DURATION(0x33)
00291 #else
00292 DEFINE_ERROR(MAC_BAD_SCAN_DURATION, 0x33)
00293 #endif //DOXYGEN_SHOULD_SKIP_THIS
00294
00295
00296 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00297
00300 #define EMBER_MAC_INCORRECT_SCAN_TYPE(0x34)
00301 #else
00302 DEFINE_ERROR(MAC_INCORRECT_SCAN_TYPE, 0x34)
00303 #endif //DOXYGEN_SHOULD_SKIP_THIS
00304
00305
00306 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00307
00310 #define EMBER_MAC_INVALID_CHANNEL_MASK(0x35)
00311 #else
00312 DEFINE_ERROR(MAC_INVALID_CHANNEL_MASK, 0x35)
00313 #endif //DOXYGEN_SHOULD_SKIP_THIS
00314
00315
00316 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00317
00321 #define EMBER_MAC_COMMAND_TRANSMIT_FAILURE(0x36)
00322 #else
00323 DEFINE_ERROR(MAC_COMMAND_TRANSMIT_FAILURE, 0x36)
00324 #endif //DOXYGEN_SHOULD_SKIP_THIS
00325
00326
00327 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00328
00332 #define EMBER_MAC_NO_ACK_RECEIVED(0x40)
00333 #else
00334 DEFINE_ERROR(MAC_NO_ACK_RECEIVED, 0x40)
00335 #endif //DOXYGEN_SHOULD_SKIP_THIS
00336
00337
00338 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00339
00342 #define EMBER_MAC_INDIRECT_TIMEOUT(0x42)
00343 #else
00344 DEFINE_ERROR(MAC_INDIRECT_TIMEOUT, 0x42)
00345 #endif //DOXYGEN_SHOULD_SKIP_THIS
00346
00348
00349
00354
00355
00356 #ifdef DOXYGEN SHOULD SKIP THIS
```



```
00357
00365 #define EMBER_SIM_EEPROM_ERASE_PAGE_GREEN(0x43)
00366 #else
00367 DEFINE_ERROR(SIM_EEPROM_ERASE_PAGE_GREEN, 0x43)
00368 #endif //DOXYGEN_SHOULD_SKIP_THIS
00369
00370
00371 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00372
00381 #define EMBER_SIM_EEPROM_ERASE_PAGE_RED(0x44)
00382 #else
00383 DEFINE_ERROR(SIM_EEPROM_ERASE_PAGE_RED, 0x44)
00384 #endif //DOXYGEN_SHOULD_SKIP_THIS
00385
00386
00387 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00388
00396 #define EMBER_SIM_EEPROM_FULL(0x45)
00397 #else
00398 DEFINE_ERROR(SIM_EEPROM_FULL, 0x45)
00399 #endif //DOXYGEN_SHOULD_SKIP_THIS
00400
00401
00402 //  Errors 46 and 47 are now defined below in the
00403 //    flash error block (was attempting to prevent renumbering)
00404
00405
00406 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00407
00414 #define EMBER_SIM_EEPROM_INIT_1_FAILED(0x48)
00415 #else
00416 DEFINE_ERROR(SIM_EEPROM_INIT_1_FAILED, 0x48)
00417 #endif //DOXYGEN_SHOULD_SKIP_THIS
00418
00419
00420 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00421
00427 #define EMBER_SIM_EEPROM_INIT_2_FAILED(0x49)
00428 #else
00429 DEFINE_ERROR(SIM_EEPROM_INIT_2_FAILED, 0x49)
00430 #endif //DOXYGEN_SHOULD_SKIP_THIS
00431
00432
00433 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00434
00441 #define EMBER_SIM_EEPROM_INIT_3_FAILED(0x4A)
00442 #else
00443 DEFINE_ERROR(SIM_EEPROM_INIT_3_FAILED, 0x4A)
00444 #endif //DOXYGEN_SHOULD_SKIP_THIS
00445
00446
00447 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00448
00459 #define EMBER_SIM_EEPROM_REPAIRING(0x4D)
00460 #else
00461 DEFINE_ERROR(SIM_EEPROM_REPAIRING, 0x4D)
00462 #endif //DOXYGEN_SHOULD_SKIP_THIS
00463
00465
00466
00471
00472 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00473
00480 #define EMBER_ERR_FLASH_WRITE_INHIBITED(0x46)
00481 #else
00482 DEFINE_ERROR(ERR_FLASH_WRITE_INHIBITED, 0x46)
00483 #endif //DOXYGEN_SHOULD_SKIP_THIS
00484
00485
00486 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00487
00493 #define EMBER_ERR_FLASH_VERIFY_FAILED(0x47)
00494 #else
00495 DEFINE_ERROR(ERR_FLASH_VERIFY_FAILED, 0x47)
00496 #endif //DOXYGEN_SHOULD_SKIP_THIS
00497
00498
00499 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00500
00506 #define EMBER_ERR_FLASH_PROG_FAIL(0x4B)
00507 #else
```



```
00508 DEFINE_ERROR(ERR_FLASH_PROG_FAIL, 0x4B)
00509 #endif //DOXYGEN_SHOULD_SKIP_THIS
00510
00511
00512 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00513
00519 #define EMBER_ERR_FLASH_ERASE_FAIL(0x4C)
00520 #else
00521 DEFINE_ERROR(ERR_FLASH_ERASE_FAIL, 0x4C)
00522 #endif //DOXYGEN_SHOULD_SKIP_THIS
00523
00525
00526
00531
00532
00533 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00534
00538 #define EMBER_ERR_BOOTLOADER_TRAP_TABLE_BAD(0x58)
00539 #else
00540 DEFINE_ERROR(ERR_BOOTLOADER_TRAP_TABLE_BAD, 0x58)
00541 #endif //DOXYGEN_SHOULD_SKIP_THIS
00542
00543
00544 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00545
00549 #define EMBER_ERR_BOOTLOADER_TRAP_UNKNOWN(0x59)
00550 #else
00551 DEFINE_ERROR(ERR_BOOTLOADER_TRAP_UNKNOWN, 0x59)
00552 #endif //DOXYGEN_SHOULD_SKIP_THIS
00553
00554
00555 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00556
00560 #define EMBER_ERR_BOOTLOADER_NO_IMAGE(0x05A)
00561 #else
00562 DEFINE_ERROR(ERR_BOOTLOADER_NO_IMAGE, 0x5A)
00563 #endif //DOXYGEN_SHOULD_SKIP_THIS
00564
00566
00567
00572
00573 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00574
00578 #define EMBER_DELIVERY_FAILED(0x66)
00579 #else
00580 DEFINE_ERROR(DELIVERY_FAILED, 0x66)
00581 #endif //DOXYGEN_SHOULD_SKIP_THIS
00582
00583
00584 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00585
00588 #define EMBER_BINDING_INDEX_OUT_OF_RANGE(0x69)
00589 #else
00590 DEFINE_ERROR(BINDING_INDEX_OUT_OF_RANGE, 0x69)
00591 #endif //DOXYGEN_SHOULD_SKIP_THIS
00592
00593
00594 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00595
00599 #define EMBER_ADDRESS_TABLE_INDEX_OUT_OF_RANGE(0x6A)
00600 #else
00601 DEFINE_ERROR(ADDRESS_TABLE_INDEX_OUT_OF_RANGE, 0x6A)
00602 #endif //DOXYGEN_SHOULD_SKIP_THIS
00603
00604
00605 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00606
00609 #define EMBER_INVALID_BINDING_INDEX(0x6C)
00610 #else
00611 DEFINE_ERROR(INVALID_BINDING_INDEX, 0x6C)
00612 #endif //DOXYGEN_SHOULD_SKIP_THIS
00613
00614
00615 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00616
00620 #define EMBER_INVALID_CALL(0x70)
00621 #else
00622 DEFINE_ERROR(INVALID_CALL, 0x70)
00623 #endif //DOXYGEN_SHOULD_SKIP_THIS
00624
00625
```



```
00626 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00627
00630 #define EMBER_COST_NOT_KNOWN(0x71)
00631 #else
00632 DEFINE_ERROR(COST_NOT_KNOWN, 0x71)
00633 #endif //DOXYGEN_SHOULD_SKIP_THIS
00634
00635
00636 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00637
00641 #define EMBER_MAX_MESSAGE_LIMIT_REACHED(0x72)
00642 #else
00643 DEFINE_ERROR(MAX_MESSAGE_LIMIT_REACHED, 0x72)
00644 #endif //DOXYGEN_SHOULD_SKIP_THIS
00645
00646 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00647
00651 #define EMBER_MESSAGE_TOO_LONG(0x74)
00652 #else
00653 DEFINE_ERROR(MESSAGE_TOO_LONG, 0x74)
00654 #endif //DOXYGEN_SHOULD_SKIP_THIS
00655
00656
00657 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00658
00662 #define EMBER_BINDING_IS_ACTIVE(0x75)
00663 #else
00664 DEFINE_ERROR(BINDING_IS_ACTIVE, 0x75)
00665 #endif //DOXYGEN_SHOULD_SKIP_THIS
00666
00667 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00668
00672 #define EMBER_ADDRESS_TABLE_ENTRY_IS_ACTIVE(0x76)
00673 #else
00674 DEFINE_ERROR(ADDRESS_TABLE_ENTRY_IS_ACTIVE, 0x76)
00675 #endif //DOXYGEN_SHOULD_SKIP_THIS
00676
00678
00683
00684
00685 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00686
00689 #define EMBER_ADC_CONVERSION_DONE(0x80)
00690 #else
00691 DEFINE_ERROR(ADC_CONVERSION_DONE, 0x80)
00692 #endif //DOXYGEN_SHOULD_SKIP_THIS
00693
00694
00695 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00696
00700 #define EMBER_ADC_CONVERSION_BUSY(0x81)
00701 #else
00702 DEFINE_ERROR(ADC_CONVERSION_BUSY, 0x81)
00703 #endif //DOXYGEN_SHOULD_SKIP_THIS
00704
00705
00706 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00707
00711 #define EMBER_ADC_CONVERSION_DEFERRED(0x82)
00712 #else
00713 DEFINE_ERROR(ADC_CONVERSION_DEFERRED, 0x82)
00714 #endif //DOXYGEN_SHOULD_SKIP_THIS
00715
00716
00717 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00718
00721 #define EMBER_ADC_NO_CONVERSION_PENDING(0x84)
00722 #else
00723 DEFINE_ERROR(ADC_NO_CONVERSION_PENDING, 0x84)
00724 #endif //DOXYGEN_SHOULD_SKIP_THIS
00725
00726
00727 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00728
00732 #define EMBER_SLEEP_INTERRUPTED(0x85)
00733 #else
00734 DEFINE_ERROR(SLEEP_INTERRUPTED, 0x85)
00735 #endif //DOXYGEN_SHOULD_SKIP_THIS
00736
00738
00743
```



```
00744
00745 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00746
00749 #define EMBER_PHY_TX_UNDERFLOW(0x88)
00750 #else
00751 DEFINE_ERROR(PHY_TX_UNDERFLOW, 0x88)
00752 #endif //DOXYGEN_SHOULD_SKIP_THIS
00753
00754
00755 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00756
00759 #define EMBER_PHY_TX_INCOMPLETE(0x89)
00760 #else
00761 DEFINE_ERROR(PHY_TX_INCOMPLETE, 0x89)
00762 #endif //DOXYGEN_SHOULD_SKIP_THIS
00763
00764
00765 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00766
00769 #define EMBER_PHY_INVALID_CHANNEL(0x8A)
00770 #else
00771 DEFINE_ERROR(PHY_INVALID_CHANNEL, 0x8A)
00772 #endif //DOXYGEN_SHOULD_SKIP_THIS
00773
00774
00775 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00776
00779 #define EMBER_PHY_INVALID_POWER(0x8B)
00780 #else
00781 DEFINE_ERROR(PHY_INVALID_POWER, 0x8B)
00782 #endif //DOXYGEN_SHOULD_SKIP_THIS
00783
00784
00785 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00786
00790 #define EMBER_PHY_TX_BUSY(0x8C)
00791 #else
00792 DEFINE_ERROR(PHY_TX_BUSY, 0x8C)
00793 #endif //DOXYGEN_SHOULD_SKIP_THIS
00794
00795
00796 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00797
00801 #define EMBER_PHY_TX_CCA_FAIL(0x8D)
00802 #else
00803 DEFINE_ERROR(PHY_TX_CCA_FAIL, 0x8D)
00804 #endif //DOXYGEN_SHOULD_SKIP_THIS
00805
00806
00807 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00808
00812 #define EMBER_PHY_OSCILLATOR_CHECK_FAILED(0x8E)
00813 #else
00814 DEFINE_ERROR(PHY_OSCILLATOR_CHECK_FAILED, 0x8E)
00815 #endif //DOXYGEN_SHOULD_SKIP_THIS
00816
00817
00818 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00819
00822 #define EMBER_PHY_ACK_RECEIVED(0x8F)
00823 #else
00824 DEFINE_ERROR(PHY_ACK_RECEIVED, 0x8F)
00825 #endif //DOXYGEN_SHOULD_SKIP_THIS
00826
00828
00834
00835
00836 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00837
00841 #define EMBER_NETWORK_UP(0x90)
00842 #else
00843 DEFINE_ERROR(NETWORK_UP, 0x90)
00844 #endif //DOXYGEN_SHOULD_SKIP_THIS
00845
00846
00847 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00848
00851 #define EMBER_NETWORK_DOWN(0x91)
00852 #else
00853 DEFINE_ERROR(NETWORK_DOWN, 0x91)
00854 #endif //DOXYGEN_SHOULD_SKIP_THIS
```



```
00855
00856
00857 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00858
00861 #define EMBER_JOIN_FAILED(0x94)
00862 #else
00863 DEFINE_ERROR(JOIN_FAILED, 0x94)
00864 #endif //DOXYGEN_SHOULD_SKIP_THIS
00865
00866
00867 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00868
00872 #define EMBER_MOVE_FAILED(0x96)
00873 #else
00874 DEFINE_ERROR(MOVE_FAILED, 0x96)
00875 #endif //DOXYGEN_SHOULD_SKIP_THIS
00876
00877
00878 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00879
00884 #define EMBER_CANNOT_JOIN_AS_ROUTER(0x98)
00885 #else
00886 DEFINE_ERROR(CANNOT_JOIN_AS_ROUTER, 0x98)
00887 #endif //DOXYGEN_SHOULD_SKIP_THIS
00888
00889
00890 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00891
00894 #define EMBER_NODE_ID_CHANGED(0x99)
00895 #else
00896 DEFINE_ERROR(NODE_ID_CHANGED, 0x99)
00897 #endif
00898
00899
00900 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00901
00904 #define EMBER_PAN_ID_CHANGED(0x9A)
00905 #else
00906 DEFINE_ERROR(PAN_ID_CHANGED, 0x9A)
00907 #endif
00908
00909 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00910
00912 #define EMBER_CHANNEL_CHANGED(0x9B)
00913 #else
00914 DEFINE_ERROR(CHANNEL_CHANGED, 0x9B)
00915 #endif
00916
00917 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00918
00921 #define EMBER_NO_BEACONS(0xAB)
00922 #else
00923 DEFINE_ERROR(NO_BEACONS, 0xAB)
00924 #endif
00925
00926
00927 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00928
00932 #define EMBER_RECEIVED_KEY_IN_THE_CLEAR(0xAC)
00933 #else
00934 DEFINE_ERROR(RECEIVED_KEY_IN_THE_CLEAR, 0xAC)
00935 #endif
00936
00937
00938 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00939
00942 #define EMBER_NO_NETWORK_KEY_RECEIVED(0xAD)
00943 #else
00944 DEFINE_ERROR(NO_NETWORK_KEY_RECEIVED, 0xAD)
00945 #endif
00946
00947
00948 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00949
00952 #define EMBER_NO_LINK_KEY_RECEIVED(0xAE)
00953 #else
00954 DEFINE_ERROR(NO_LINK_KEY_RECEIVED, 0xAE)
00955 #endif
00956
00957
00958 #ifdef DOXYGEN_SHOULD_SKIP_THIS
```



```
00959
00963 #define EMBER_PRECONFIGURED_KEY_REQUIRED(0xAF)
00964 #else
00965 DEFINE_ERROR(PRECONFIGURED_KEY_REQUIRED, 0xAF)
00966 #endif
00967
00968
00970
00974 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00975
00979 #define EMBER_KEY_INVALID(0xB2)
00980 #else
00981 DEFINE_ERROR(KEY_INVALID, 0xB2)
00982 #endif // DOXYGEN_SHOULD_SKIP_THIS
00983
00984 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00985
00989 #define EMBER_INVALID_SECURITY_LEVEL(0x95)
00990 #else
00991 DEFINE_ERROR(INVALID_SECURITY_LEVEL, 0x95)
00992 #endif //DOXYGEN_SHOULD_SKIP_THIS
00993
00994 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00995
01003 #define EMBER_APS_ENCRYPTION_ERROR(0xA6)
01004 #else
01005      DEFINE_ERROR(APS_ENCRYPTION_ERROR, 0xA6)
01006 #endif //DOXYGEN_SHOULD_SKIP_THIS
01007
01008 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01009
01012 #define EMBER_TRUST_CENTER_MASTER_KEY_NOT_SET(0xA7)
01013 #else
01014      DEFINE_ERROR(TRUST_CENTER_MASTER_KEY_NOT_SET, 0xA7)
01015 #endif //DOXYGEN_SHOULD_SKIP_THIS
01016
01017 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01018
01021 #define EMBER_SECURITY_STATE_NOT_SET(0xA8)
01022 #else
01023      DEFINE_ERROR(SECURITY_STATE_NOT_SET, 0xA8)
01024 #endif //DOXYGEN_SHOULD_SKIP_THIS
01025
01026 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01027
01034 #define EMBER_KEY_TABLE_INVALID_ADDRESS(0xB3)
01035 #else
01036 DEFINE_ERROR(KEY_TABLE_INVALID_ADDRESS, 0xB3)
01037 #endif //DOXYGEN_SHOULD_SKIP_THIS
01038
01039 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01040
01043 #define EMBER_SECURITY_CONFIGURATION_INVALID(0xB7)
01044 #else
01045 DEFINE_ERROR(SECURITY_CONFIGURATION_INVALID, 0xB7)
01046 #endif //DOXYGEN_SHOULD_SKIP_THIS
01047
01048 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01049
01054 #define EMBER_TOO_SOON_FOR_SWITCH_KEY(0xB8)
01055 #else
01056      DEFINE_ERROR(TOO_SOON_FOR_SWITCH_KEY, 0xB8)
01057 #endif
01058
01059 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01060
01063 #define EMBER_SIGNATURE_VERIFY_FAILURE(0xB9)
01064 #else
01065      DEFINE_ERROR(SIGNATURE_VERIFY_FAILURE, 0xB9)
01066 #endif
01067
01068 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01069
01075 #define EMBER_KEY_NOT_AUTHORIZED(0xBB)
01076 #else
01077      DEFINE_ERROR(KEY_NOT_AUTHORIZED, 0xBB)
01078 #endif
01079
01080
01082
01083
```



```
01088
01089
01090 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01091
01094 #define EMBER_NOT_JOINED(0x93)
01095 #else
01096 DEFINE_ERROR(NOT_JOINED, 0x93)
01097 #endif //DOXYGEN_SHOULD_SKIP_THIS
01098
01099 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01100
01104 #define EMBER_NETWORK_BUSY(0xA1)
01105 #else
01106 DEFINE_ERROR(NETWORK_BUSY, 0xA1)
01107 #endif //DOXYGEN_SHOULD_SKIP_THIS
01108
01109
01110 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01111
01115 #define EMBER_INVALID_ENDPOINT(0xA3)
01116 #else
01117 DEFINE_ERROR(INVALID_ENDPOINT, 0xA3)
01118 #endif //DOXYGEN_SHOULD_SKIP_THIS
01119
01120
01121 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01122
01126 #define EMBER_BINDING_HAS_CHANGED(0xA4)
01127 #else
01128 DEFINE_ERROR(BINDING_HAS_CHANGED, 0xA4)
01129 #endif //DOXYGEN_SHOULD_SKIP_THIS
01130
01131 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01132
01136 #define EMBER_INSUFFICIENT_RANDOM_DATA(0xA5)
01137 #else
01138      DEFINE_ERROR(INSUFFICIENT_RANDOM_DATA, 0xA5)
01139 #endif //DOXYGEN_SHOULD_SKIP_THIS
01140
01141
01142 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01143
01146 #define EMBER_SOURCE_ROUTE_FAILURE(0xA9)
01147 #else
01148      DEFINE_ERROR(SOURCE_ROUTE_FAILURE, 0xA9)
01149 #endif
01150
01151 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01152
01157 #define EMBER_MANY_TO_ONE_ROUTE_FAILURE(0xAA)
01158 #else
01159      DEFINE_ERROR(MANY_TO_ONE_ROUTE_FAILURE, 0xAA)
01160 #endif
01161
01162
01164
01169
01170
01171 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01172
01178 #define EMBER_STACK_AND_HARDWARE_MISMATCH(0xB0)
01179 #else
01180 DEFINE_ERROR(STACK_AND_HARDWARE_MISMATCH, 0xB0)
01181 #endif //DOXYGEN_SHOULD_SKIP_THIS
01182
01183
01184 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01185
01189 #define EMBER_INDEX_OUT_OF_RANGE(0xB1)
01190 #else
01191 DEFINE_ERROR(INDEX_OUT_OF_RANGE, 0xB1)
01192 #endif
01193
01194 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01195
01198 #define EMBER_TABLE_FULL(0xB4)
01199 #else
01200 DEFINE_ERROR(TABLE_FULL, 0xB4)
01201 #endif //DOXYGEN_SHOULD_SKIP_THIS
01202
01203 #ifdef DOXYGEN_SHOULD_SKIP_THIS
```



```
01204
01208 #define EMBER_TABLE_ENTRY_ERASED(0xB6)
01209 #else
01210 DEFINE_ERROR(TABLE_ENTRY_ERASED, 0xB6)
01211 #endif
01212
01213 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01214
01218 #define EMBER_LIBRARY_NOT_PRESENT(0xB5)
01219 #else
01220 DEFINE_ERROR(LIBRARY_NOT_PRESENT, 0xB5)
01221 #endif
01222
01223 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01224
01228 #define EMBER_OPERATION_IN_PROGRESS(0xBA)
01229 #else
01230 DEFINE_ERROR(OPERATION_IN_PROGRESS, 0xBA)
01231 #endif
01232
01233 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01234
01239 #define EMBER_TRUST_CENTER_EUI_HAS_CHANGED(0xBC)
01240 #else
01241      DEFINE_ERROR(TRUST_CENTER_EUI_HAS_CHANGED, 0xBC)
01242 #endif
01243
01245
01251
01252 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01253
01257 #define EMBER_APPLICATION_ERROR_0(0xF0)
01258 #define EMBER_APPLICATION_ERROR_1(0xF1)
01259 #define EMBER_APPLICATION_ERROR_2(0xF2)
01260 #define EMBER_APPLICATION_ERROR_3(0xF3)
01261 #define EMBER_APPLICATION_ERROR_4(0xF4)
01262 #define EMBER_APPLICATION_ERROR_5(0xF5)
01263 #define EMBER_APPLICATION_ERROR_6(0xF6)
01264 #define EMBER_APPLICATION_ERROR_7(0xF7)
01265 #define EMBER_APPLICATION_ERROR_8(0xF8)
01266 #define EMBER_APPLICATION_ERROR_9(0xF9)
01267 #define EMBER_APPLICATION_ERROR_10(0xFA)
01268 #define EMBER_APPLICATION_ERROR_11(0xFB)
01269 #define EMBER_APPLICATION_ERROR_12(0xFC)
01270 #define EMBER_APPLICATION_ERROR_13(0xFD)
01271 #define EMBER_APPLICATION_ERROR_14(0xFE)
01272 #define EMBER_APPLICATION_ERROR_15(0xFF)
01273 #else
01274 DEFINE_ERROR( APPLICATION_ERROR_0, 0xF0)
01275 DEFINE_ERROR( APPLICATION_ERROR_1, 0xF1)
01276 DEFINE_ERROR( APPLICATION_ERROR_2, 0xF2)
01277 DEFINE_ERROR( APPLICATION_ERROR_3, 0xF3)
01278 DEFINE_ERROR( APPLICATION_ERROR_4, 0xF4)
01279 DEFINE_ERROR( APPLICATION_ERROR_5, 0xF5)
01280 DEFINE_ERROR( APPLICATION_ERROR_6, 0xF6)
01281 DEFINE_ERROR( APPLICATION_ERROR_7, 0xF7)
01282 DEFINE_ERROR( APPLICATION_ERROR_8, 0xF8)
01283 DEFINE_ERROR( APPLICATION_ERROR_9, 0xF9)
01284 DEFINE_ERROR( APPLICATION_ERROR_10, 0xFA)
01285 DEFINE_ERROR( APPLICATION_ERROR_11, 0xFB)
01286 DEFINE_ERROR( APPLICATION_ERROR_12, 0xFC)
01287 DEFINE_ERROR( APPLICATION_ERROR_13, 0xFD)
01288 DEFINE_ERROR( APPLICATION_ERROR_14, 0xFE)
01289 DEFINE_ERROR( APPLICATION_ERROR_15, 0xFF)
01290 #endif //DOXYGEN_SHOULD_SKIP_THIS
01291
01293
```


stack » include

error.h File Reference

Return codes for Ember API functions and module definitions. More...

Go to the source code of this file.

Defines

#define DEFINE_ERROR(symbol, value)

Typedefs

typedef int8u EmberStatus

Enumerations

enum { EMBER_ERROR_CODE_COUNT }

Detailed Description

Return codes for Ember API functions and module definitions.

See Ember Status Codes for documentation.

Definition in file error.h.

Typedef Documentation

typedef int8u EmberStatus

Return type for Ember functions.

Definition at line 19 of file error.h.

---

error.h

Go to the documentation of this file.

```
00001
00011 #ifndef __ERRORS_H__
00012 #define __ERRORS_H__
00013
00017 #ifndef __EMBERSTATUS_TYPE__
00018 #define __EMBERSTATUS_TYPE__
00019   typedef int8u 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
```


ezsp-host-configuration-defaults.h File Reference

User-configurable parameters for host applications. More...

Go to the source code of this file.

Defines

#define	**EZSP_HOST_SOURCE_ROUTE_TABLE_SIZE**
#define	**EZSP_HOST_ASH_RX_POOL_SIZE**
#define	**EZSP_HOST_FORM_AND_JOIN_BUFFER_SIZE**

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**.

ezsp-host-configuration-defaults.h

Go to the documentation of this file.

```
00001
00019 #ifdef CONFIGURATION_HEADER
00020   #include CONFIGURATION_HEADER
00021 #endif
00022
00023 #ifndef EZSP_HOST_SOURCE_ROUTE_TABLE_SIZE
00024
00032   #define EZSP_HOST_SOURCE_ROUTE_TABLE_SIZE 32
00033 #endif
00034
00035 #ifndef EZSP_HOST_ASH_RX_POOL_SIZE
00036
00043   #define EZSP_HOST_ASH_RX_POOL_SIZE 20
00044 #endif
00045
00046 #ifndef EZSP_HOST_FORM_AND_JOIN_BUFFER_SIZE
00047
00055   #define EZSP_HOST_FORM_AND_JOIN_BUFFER_SIZE 40
00056 #endif
00057
```


form-and-join.h File Reference

Utilities for forming and joining networks. More...

Go to the source code of this file.

Defines

#define	NETWORK_STORAGE_SIZE
#define	NETWORK_STORAGE_SIZE_SHIFT
#define	FORM_AND_JOIN_MAX_NETWORKS

Functions

EmberStatus	emberScanForUnusedPanId (int32u channelMask, int8u duration)
EmberStatus	emberScanForJoinableNetwork (int32u channelMask, int8u *extendedPanId)
EmberStatus	emberScanForNextJoinableNetwork (void)
boolean	emberFormAndJoinIsScanning (void)
void	emberUnusedPanIdFoundHandler (EmberPanId panId, int8u channel)
void	emberJoinableNetworkFoundHandler (EmberZigbeeNetwork *networkFound, int8u lqi, int8s rssi)
void	emberScanErrorHandler (EmberStatus status)
boolean	emberFormAndJoinScanCompleteHandler (int8u channel, EmberStatus status)
boolean	emberFormAndJoinNetworkFoundHandler (EmberZigbeeNetwork *networkFound, int8u lqi, int8s rssi)
boolean	emberFormAndJoinEnergyScanResultHandler (int8u channel, int8s maxRssiValue)
void	emberFormAndJoinTick (void)
void	emberFormAndJoinTaskInit (void)
void	emberFormAndJoinRunTask (void)

Variables

boolean	emberEnableDualChannelScan

Detailed Description

Utilities for forming and joining networks.

See **Forming and Joining Networks** for documentation.

Definition in file **form-and-join.h**.

form-and-join.h

Go to the documentation of this file.

```
00001
00071 #define NETWORK_STORAGE_SIZE  16
00072
00075 #define NETWORK_STORAGE_SIZE_SHIFT 4
00076
00090 #ifndef FORM_AND_JOIN_MAX_NETWORKS
00091   #ifdef EZSP_HOST
00092     // the host's buffer is 16-bit array, so translate to bytes for comparison
00093     #define FORM_AND_JOIN_MAX_NETWORKS  \
00094       (EZSP_HOST_FORM_AND_JOIN_BUFFER_SIZE * 2 / NETWORK_STORAGE_SIZE)
00095   #else
00096     // use highest value that won't exceed max EmberMessageBuffer length
00097     #define FORM_AND_JOIN_MAX_NETWORKS 15
00098   #endif
00099 #endif
00100
00101 // Check that this value isn't too large for the SoC implementation to handle
00102 #ifndef EZSP_HOST
00103   #if (FORM_AND_JOIN_MAX_NETWORKS > 15)
00104     #error "FORM_AND_JOIN_MAX_NETWORKS can't exceed 15 on SoC platform"
00105   #endif
00106 #endif
00107
00124 EmberStatus emberScanForUnusedPanId(int32u channelMask, int8u duration);
00125
00152 EmberStatus emberScanForJoinableNetwork(int32u channelMask, int8u* extendedPanId);
00153
00155 EmberStatus emberScanForNextJoinableNetwork(void);
00156
00172 extern boolean emberEnableDualChannelScan;
00173
00178 boolean emberFormAndJoinIsScanning(void);
00179
00180 //------------------------------------------------------------------------------
00181 // Callbacks the application needs to implement.
00182
00191 void emberUnusedPanIdFoundHandler(EmberPanId panId, int8u channel);
00192
00203 void emberJoinableNetworkFoundHandler(EmberZigbeeNetwork *networkFound,
00204                                       int8u lqi,
00205                                       int8s rssi);
00206
00224 void emberScanErrorHandler(EmberStatus status);
00225
00226 //------------------------------------------------------------------------------
00227 // Library functions the application must call from within the
00228 // corresponding EmberZNet or EZSP callback.
00229
00237 boolean emberFormAndJoinScanCompleteHandler(int8u channel, EmberStatus status);
00238
00246 boolean emberFormAndJoinNetworkFoundHandler(EmberZigbeeNetwork *networkFound,
00247                                             int8u lqi,
00248                                             int8s rssi);
00249
00257 boolean emberFormAndJoinEnergyScanResultHandler(int8u channel, int8s maxRssiValue);
00258
00263 void emberFormAndJoinTick(void);
00264
00268 void emberFormAndJoinTaskInit(void);
00269
00273 void emberFormAndJoinRunTask(void);
00274
00275
```


form-and-join3_2.h File Reference

Utilities for forming and joining networks. Deprecated and will be removed from a future release. Use **form-and-join.h** instead. More...

Go to the source code of this file.

Enumerations

enum	**formAndJoinScanType** { **FORM_AND_JOIN_NOT_SCANNING**, **FORM_AND_JOIN_ENERGY_SCAN**, **FORM_AND_JOIN_PAN_ID_SCAN**, **FORM_AND_JOIN_JOINABLE_SCAN**, **FORM_AND_JOIN_CROSSTALK_SCAN** }

Functions

void	**formZigbeeNetwork3_2** (**int32u** channelMask, **int8s** radioTxPower, **int8u** *extendedPanIdDesired)
void	**joinZigbeeNetwork3_2** (**EmberNodeType** nodeType, **int32u** channelMask, **int8s** radioTxPower, **int8u** *extendedPanIdDesired)
void	**scanError** (**EmberStatus** status)

Detailed Description

Utilities for forming and joining networks. Deprecated and will be removed from a future release. Use **form-and-join.h** instead.

See **Forming and Joining Networks** for documentation.

Definition in file **form-and-join3_2.h**.

Enumeration Type Documentation

enum formAndJoinScanType

The current reason for scanning.

Enumerator:

FORM_AND_JOIN_NOT_SCANNING	
FORM_AND_JOIN_ENERGY_SCAN	Energy scan for finding a quiet channel.
FORM_AND_JOIN_PAN_ID_SCAN	Active scan to see which PAN IDs are in use.
FORM_AND_JOIN_JOINABLE_SCAN	Active scan for a network to join.
FORM_AND_JOIN_CROSSTALK_SCAN	Active scan to work around channel crosstalk.

Definition at line **21** of file **form-and-join3_2.h**.

Function Documentation

void formZigbeeNetwork3_2 (int32u channelMask, int8s radioTxPower, int8u * extendedPanIdDesired)

Form a network.

This performs the following actions:

1. Do an energy scan on the indicated channels and randomly choose one from amongst those with the least average energy.
2. Randomly pick a short PAN ID that does not appear during an active scan on the chosen channel.

3. use the Extended PAN ID passed in or pick a random one if the Extended PAN ID passed in is "0" or a null pointer.
4. Form a network using the chosen channel, short PAN ID, and extended PAN ID.

If any errors occur, the status code is passed to **scanError()** and no network is formed. Success is indicated by calling emberStackStatusHandler() with the EMBER_NETWORK_UP status value.

Parameters:

channelMask
radioTxPower
extendedPanIdDesired

void joinZigbeeNetwork3_2 (EmberNodeType nodeType,
int32u channelMask,
int8s radioTxPower,
int8u * extendedPanIdDesired
)

Join a network.

This tries to join the first network found on the indicated channels that

1. currently permits joining
2. matches the stack profile of the application
3. matches the Extended PAN ID passed in, or if "0" is passed in it matches any Extended PAN ID.

If any errors occur, the status code is passed to **scanError()** and no network is joined. Success is indicated by calling emberStackStatusHandler() with the EMBER_NETWORK_UP status value.

With some board layouts, the em250 is susceptible to a dual channel issue in which packets from 12 channels above or below can sometimes be heard faintly. This affects channels 11, 12, 13, 14, 23, 24, 25, and 26. Hardware reference designs EM250_REF_DES_LAT, version C0 and EM250_REF_DES_CER, version B0 solve the problem.

This function also implements a software workaround. After discovering a network on one of the susceptible channels, joinZigbeeNetwork also scans the channel 12 up or down. If the same network is found there, it chooses the correct one by comparing the link quality of the received beacons.

Parameters:

nodeType
channelMask
radioTxPower
extendedPanIdDesired

void scanError (EmberStatus status)

A callback the application needs to provided.

If an error occurs while attempting to form or join a network, this procedure is called and the form or join effort is aborted.

Parameters:

status

form-and-join3_2.h

Go to the documentation of this file.

```
00001
00018 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00019
00021 enum formAndJoinScanType
00022 #else
00023 extern int8u formAndJoinScanType;
00024 enum
00025 #endif
00026 {
00027   FORM_AND_JOIN_NOT_SCANNING,
00028   FORM_AND_JOIN_ENERGY_SCAN,
00029   FORM_AND_JOIN_PAN_ID_SCAN,
00030   FORM_AND_JOIN_JOINABLE_SCAN,
00031   FORM_AND_JOIN_CROSSTALK_SCAN
00032 };
00033
00034
00055 void formZigbeeNetwork3_2(int32u channelMask,
00056                           int8s radioTxPower,
00057                           int8u* extendedPanIdDesired);
00058
00088 void joinZigbeeNetwork3_2(EmberNodeType nodeType,
00089                           int32u channelMask,
00090                           int8s radioTxPower,
00091                           int8u* extendedPanIdDesired);
00092
00100 void scanError(EmberStatus status);
00101
```


fragment-host.h File Reference

Fragmented message support for EZSP Hosts. Splits long messages into smaller blocks for transmission and reassembles received blocks. See **Message Fragmentation** for documentation. More...

Go to the source code of this file.

Initialization

void	**ezspFragmentInit** (**int16u** receiveBufferLength, **int8u** *receiveBuffer)

Transmitting

EmberStatus	**ezspFragmentSendUnicast** (**EmberOutgoingMessageType** type, **int16u** indexOrDestination, **EmberApsFrame** *apsFrame, **int8u** maxFragmentSize, **int16u** messageLength, **int8u** *messageContents)
EmberStatus	**ezspFragmentSourceRouteHandler** (void)
boolean	**ezspFragmentMessageSent** (**EmberApsFrame** *apsFrame, **EmberStatus** status)
void	**ezspFragmentMessageSentHandler** (**EmberStatus** status)

Receiving

boolean	**ezspFragmentIncomingMessage** (**EmberApsFrame** *apsFrame, **EmberNodeId** sender, **int16u** *messageLength, **int8u** **messageContents)
void	**ezspFragmentTick** (void)

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.

Definition in file **fragment-host.h**.

fragment-host.h

Go to the documentation of this file.

```
00001
00055 void ezspFragmentInit(int16u receiveBufferLength, int8u *receiveBuffer);
00056
00091 EmberStatus ezspFragmentSendUnicast(EmberOutgoingMessageType type,
00092                                       int16u indexOrDestination,
00093                                       EmberApsFrame *apsFrame,
00094                                       int8u maxFragmentSize,
00095                                       int16u messageLength,
00096                                       int8u *messageContents);
00097
00110 EmberStatus ezspFragmentSourceRouteHandler(void);
00111
00126 boolean ezspFragmentMessageSent(EmberApsFrame *apsFrame, EmberStatus status);
00127
00136 void ezspFragmentMessageSentHandler(EmberStatus status);
00137
00169 boolean ezspFragmentIncomingMessage(EmberApsFrame *apsFrame,
00170                                       EmberNodeId sender,
00171                                       int16u *messageLength,
00172                                       int8u **messageContents);
00173
00178 void ezspFragmentTick(void);
00179
```


hal

hal.h File Reference

Generic set of HAL includes for all platforms. More...

```
#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 "adc.h"
#include "buzzer.h"
```

Go to the source code of this file.

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**.

---

hal

hal.h

Go to the documentation of this file.

```
00001
00063 #ifndef __HAL_H__
00064 #define __HAL_H__
00065
00066 #ifdef HAL_HOST
00067
00068 #include "host/button-common.h"
00069 #include "host/crc.h"
00070 #include "host/led-common.h"
00071 #include "host/micro-common.h"
00072 #include "host/serial.h"
00073 #include "host/system-timer.h"
00074 //Pull in the micro specific ADC, buzzer, and clocks headers.  The
00075 //specific header is chosen by the build include path pointing at
00076 //the appropriate directory.
00077 #include "adc.h"
00078 #include "buzzer.h"
00079
00080 #else //HAL_MICRO
00081
00082 // Keep micro and board first for specifics used by other headers
00083 #include "micro/micro.h"
00084 #if !defined(STACK) && defined(BOARD_HEADER)
00085 #include BOARD_HEADER
00086 #endif
00087
00088 #include "micro/adc.h"
00089 #include "micro/button.h"
00090 #include "micro/buzzer.h"
00091 #include "micro/crc.h"
00092 #include "micro/endian.h"
00093 #include "micro/led.h"
00094 #include "micro/random.h"
00095 #include "micro/serial.h"
00096 #include "micro/spi.h"
00097 #include "micro/system-timer.h"
00098 //Host processors do not use the following modules, therefore the header
00099 //files should be ignored.
00100 #ifndef EZSP_HOST
00101   #include "micro/bootloader-interface.h"
00102   #include "micro/diagnostic.h"
00103   #include "micro/token.h"
00104   //No public HAL code in release 4.0 uses the symbol timer,
00105   //therefore it should not be in doxygen.
00106   #ifndef DOXYGEN_SHOULD_SKIP_THIS
00107     #include "micro/symbol-timer.h"
00108   #endif // DOXYGEN_SHOULD_SKIP_THIS
00109 #endif //EZSP_HOST
00110
00111 #endif
00112
00113 #endif //__HAL_H__
00114
```


iar-st.h File Reference

```
#include "stm32f10x.h"
#include <stdarg.h>
#include <stdint.h>
#include <string.h>
#include "hal/host/generic/compiler/platform-common.h"
```

Go to the source code of this file.

Defines

#define	**halResetWatchdog**()
#define	**SIGNED_ENUM**
#define	**_HAL_USE_COMMON_DIVMOD_**
#define	**_HAL_USE_COMMON_PGM_**
#define	**PLATCOMMONOKTOINCLUDE**

Functions

void	**halInternalResetWatchDog** (void)

Miscellaneous Macros

#define	**simulatedSerialTimePasses**()
#define	**BIGENDIAN_CPU**
#define	**MAIN_FUNCTION_PARAMETERS**
#define	**MAIN_FUNCTION_ARGUMENTS**
#define	**__SOURCEFILE__**
#define	**assert**(condition)
#define	**simulatedTimePasses**()
#define	**simulatedTimePassesMs**(x)
#define	**simulatedSerialTimePasses**()
void	**halInternalAssertFailed** (const char *filename, int linenumber)

Global Interrupt Manipulation Macros

#define	**DISABLE_INTERRUPTS**()
#define	**RESTORE_INTERRUPTS**()
#define	**INTERRUPTS_ON**()
#define	**INTERRUPTS_OFF**()
#define	**INTERRUPTS_ARE_OFF**()
#define	**INTERRUPTS_WERE_ON**()
#define	**ATOMIC**(blah)
#define	**HANDLE_PENDING_INTERRUPTS**()

Generic Types

#define	**NULL**

C Standard Library Memory Utilities

These should be used in place of the standard library functions.

#define	**halCommonMemSet**(d, v, l)
#define	**halCommonMemCopy**(d, s, l)
#define	**halCommonMemCompare**(s0, s1, l)
#define	**halCommonMemPGMCompare**(s0, s1, l)

#define	**halCommonMemPGMCopy**(d, s, l)
#define	**MEMSET**(d, v, l)
#define	**MEMCOPY**(d, s, l)
#define	**MEMCOMPARE**(s0, s1, l)
#define	**MEMPGMCOMPARE**(s0, s1, l)

Master Variable Types

These are a set of typedefs to make the size of all variable declarations explicitly known. Since the IAR host code links against the ST Standard peripheral library, we need to map Ember's variable types to ST's variable types.

Note:
ST uses IAR's variable types, found in stdint.h.

typedef uint8_t	**boolean**
typedef uint8_t	**int8u**
typedef int8_t	**int8s**
typedef uint16_t	**int16u**
typedef int16_t	**int16s**
typedef uint32_t	**int32u**
typedef int32_t	**int32s**
typedef uint32_t	**PointerType**

Detailed Description

See **Common PLATFORM_HEADER Configuration** and **STM32F103RET IAR Specific PLATFORM_HEADER Configuration** for documentation.

Definition in file **iar-st.h**.

iar-st.h

Go to the documentation of this file.

```
00001
00021 #ifndef __IAR_ST_H__
00022 #define __IAR_ST_H__
00023
00024 #ifndef __ICCARM__
00025   #error Improper PLATFORM_HEADER
00026 #endif
00027
00028 #if (__VER__ < 5040005) || (__VER__ > 5050006)
00029   #error Only IAR EWARM versions >= 5.40.5 and <= 5.50.6 are supported
00030 #endif // __VER__
00031
00032 //Pull in the registers, Library, and other critical/useful ST code.
00033 #include "stm32f10x.h"
00034 #include <stdarg.h>
00035 #include <stdint.h>
00036
00037 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00038 //The Cortex-M3 does not have zero-page memory
00039 #define XAP2B_PAGEZERO_ON
00040 #define XAP2B_PAGEZERO_OFF
00041 #endif
00042
00054 typedef uint8_t  boolean;
00055 typedef uint8_t  int8u;
00056 typedef int8_t   int8s;
00057 typedef uint16_t int16u;
00058 typedef int16_t  int16s;
00059 typedef uint32_t int32u;
00060 typedef int32_t  int32s;
00061 typedef uint32_t PointerType;
00063
00064
00070 void halInternalResetWatchDog(void);
00071
00077 #define halResetWatchdog()  halInternalResetWatchDog()
00078
00079
00083 #define SIGNED_ENUM
00084
00088 #define simulatedSerialTimePasses()
00089
00090
00094 #define _HAL_USE_COMMON_DIVMOD_
00095
00096
00100 #define _HAL_USE_COMMON_PGM_
00101
00102
00104
00106
00107
00108
00113 #define BIGENDIAN_CPU  FALSE
00114
00119 #define MAIN_FUNCTION_PARAMETERS void
00120 #define MAIN_FUNCTION_ARGUMENTS
00121
00122
00123 #ifndef __SOURCEFILE__
00124
00129   #define __SOURCEFILE__ __FILE__
00130 #endif
00131
00132
00133 #undef assert
00134 #if !defined(SIMPLER_ASSERT_REBOOT) || defined(DOXYGEN_SHOULD_SKIP_THIS)
00135
00138   void halInternalAssertFailed(const char * filename, int linenumber);
00139
00145   #define assert(condition)                                          \
00146         do {                                                         \
00147           if (! (condition)) {                                       \
```



```
00148             halInternalAssertFailed(__SOURCEFILE__, __LINE__); \
00149           }                                                     \
00150         } while(0)
00151 #else
00152   #define assert(condition) \
00153             do { if( !(condition) ) while(1){} } while(0)
00154 #endif
00155
00156
00160 #define simulatedTimePasses()
00161
00164 #define simulatedTimePassesMs(x)
00165
00168 #define simulatedSerialTimePasses()
00169
00171
00172
00173
00174
00176
00178
00179
00180
00181 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00182     //The concept of LITE atomic handling isn't implemented on this platform,
00183     //so just redirect to the normal atomic handling.
00184     #define ATOMIC_LITE(blah)             ATOMIC(blah)
00185     #define DECLARE_INTERRUPT_STATE_LITE  DECLARE_INTERRUPT_STATE
00186     #define DISABLE_INTERRUPTS_LITE()     DISABLE_INTERRUPTS()
00187     #define RESTORE_INTERRUPTS_LITE()     RESTORE_INTERRUPTS()
00188
00194     #define DECLARE_INTERRUPT_STATE int32u _emIsrState
00195 #endif  // DOXYGEN_SHOULD_SKIP_THIS
00196
00197 //The core Global Interrupt Manipulation Macros start here.
00198
00205 #define DISABLE_INTERRUPTS()       \
00206   do {                             \
00207     _emIsrState = __get_PRIMASK(); \
00208     __set_PRIMASK(1);              \
00209   } while(0)
00210
00211
00219 #define RESTORE_INTERRUPTS()    \
00220   do {                          \
00221     __set_PRIMASK(_emIsrState); \
00222   } while(0)
00223
00224
00229 #define INTERRUPTS_ON() \
00230   do {                  \
00231     __set_PRIMASK(0);   \
00232   } while(0)
00233
00234
00239 #define INTERRUPTS_OFF() \
00240   do {                   \
00241     __set_PRIMASK(1);    \
00242   } while(0)
00243
00244
00248 #define INTERRUPTS_ARE_OFF() (__get_PRIMASK() != 0)
00249
00250
00255 #define INTERRUPTS_WERE_ON() (_emIsrState == 0)
00256
00257
00262 #define ATOMIC(blah)       \
00263 {                          \
00264   DECLARE_INTERRUPT_STATE; \
00265   DISABLE_INTERRUPTS();    \
00266   { blah }                 \
00267   RESTORE_INTERRUPTS();    \
00268 }
00269
00270
00278 #define HANDLE_PENDING_INTERRUPTS() \
00279   do {                              \
00280     if (INTERRUPTS_ARE_OFF()) {     \
00281       INTERRUPTS_ON();              \
00282       INTERRUPTS_OFF();             \
```



```
00283     }                                 \
00284   } while (0)
00285
00287
00288
00289
00290
00295 //TRUE and FLASE are defined in ST's HAL Library
00296
00297 #ifndef NULL
00298
00301 #define NULL ((void *)0)
00302 #endif
00303
00305
00306
00315 #include <string.h>
00316 #define halCommonMemSet(d,v,l) memset(d,v,l)
00317 #define halCommonMemCopy(d,s,l) memcpy(d,s,l)
00318 #define halCommonMemCompare(s0,s1,l) memcmp(s0, s1, l)
00319 #define halCommonMemPGMCompare(s0,s1,l) memcmp(s0, s1, l)
00320 #define halCommonMemPGMCopy(d,s,l) memcpy(d,s,l)
00321
00322 #define MEMSET(d,v,l)  halCommonMemSet(d,v,l)
00323 #define MEMCOPY(d,s,l) halCommonMemCopy(d,s,l)
00324 #define MEMCOMPARE(s0,s1,l) halCommonMemCompare(s0, s1, l)
00325 #define MEMPGMCOMPARE(s0,s1,l) halCommonMemPGMCompare(s0, s1, l)
00326
00327
00328
00329
00333 #define PLATCOMMONOKTOINCLUDE
00334   #include "hal/host/generic/compiler/platform-common.h"
00335 #undef PLATCOMMONOKTOINCLUDE
00336
00337 #endif // __IAR_ST_H__
00338
```


led-common.h File Reference

```
#include "led-specific.h"
```

Go to the source code of this file.

Typedefs

typedef enum **HalBoardLedPins**	**HalBoardLed**

Functions

void	**halInternalInitLed** (void)
void	**halToggleLed** (**HalBoardLed** led)
void	**halSetLed** (**HalBoardLed** led)
void	**halClearLed** (**HalBoardLed** led)

Detailed Description

See **LED Control** and micro specific modules for documentation.

Definition in file **led-common.h**.

led-common.h

Go to the documentation of this file.

```
00001
00020 #ifndef __LED_COMMON_H__
00021 #define __LED_COMMON_H__
00022
00023
00026 void halInternalInitLed(void);
00027
00028
00037 typedef enum HalBoardLedPins HalBoardLed;
00038
00039
00045 void halToggleLed(HalBoardLed led);
00046
00047
00053 void halSetLed(HalBoardLed led);
00054
00055
00061 void halClearLed(HalBoardLed led);
00062
00063
00064 //Pull in the micro specific LED definitions.  The specific header is chosen
00065 //by the build include path pointing at the appropriate directory.
00066 #include "led-specific.h"
00067
00068
00069 #endif //__LED_COMMON_H__
00070
```


led-specific.h File Reference

Go to the source code of this file.

Defines

#define	BOARDLED0_PIN
#define	BOARDLED0_PORT
#define	BOARDLED1_PIN
#define	BOARDLED1_PORT

Enumerations

```
enum HalBoardLedPins {
  BOARDLED0,
  BOARDLED1,
  BOARD_ACTIVITY_LED,
  BOARD_HEARTBEAT_LED
}
```

Detailed Description

See **LED Control** and **STM32F103RET Specific LED** for documentation.

Definition in file **led-specific.h**.

led-specific.h

Go to the documentation of this file.

```
00001
00019 #ifndef __LED_SPECIFIC_H__
00020 #define __LED_SPECIFIC_H__
00021
00022
00028 enum HalBoardLedPins {
00029   BOARDLED0 = 0, //Just a simple identifier for switch statements
00030   BOARDLED1 = 1, //Just a simple identifier for switch statements
00031   BOARD_ACTIVITY_LED  = BOARDLED0,
00032   BOARD_HEARTBEAT_LED = BOARDLED1
00033 };
00034
00035
00039 #define BOARDLED0_PIN   GPIO_Pin_8
00040
00044 #define BOARDLED0_PORT  GPIOB
00045
00046
00050 #define BOARDLED1_PIN   GPIO_Pin_9
00051
00055 #define BOARDLED1_PORT  GPIOB
00056
00057
00058 #endif //__LED_SPECIFIC_H__
00059
```


micro-common.h File Reference

```
#include "micro-specific.h"
```

Go to the source code of this file.

Defines

```
#define  MICRO_DISABLE_WATCH_DOG_KEY
```

Enumerations

```
enum  SleepModes {
        SLEEPMODE_RUNNING,
        SLEEPMODE_IDLE,
        SLEEPMODE_WAKETIMER,
        SLEEPMODE_MAINTAINTIMER,
        SLEEPMODE_NOTIMER,
        SLEEPMODE_RESERVED,
        SLEEPMODE_POWERDOWN,
        SLEEPMODE_POWERSAVE
      }
```

Functions

void	halInit (void)
void	halReboot (void)
void	halPowerUp (void)
void	halPowerDown (void)
void	halInternalEnableWatchDog (void)
void	halInternalDisableWatchDog (int8u magicKey)
void	halCommonDelayMicroseconds (int16u us)
void	halCommonDelayMilliseconds (int16u ms)
void	halInternalAssertFailed (PGM_P filename, int linenumber)
int8u	halGetResetInfo (void)
PGM_P	halGetResetString (void)
void	halStackSeedRandom (int32u seed)
int16u	halCommonGetRandom (void)
void	halSleep (SleepModes sleepMode)

Detailed Description

See **Microcontroller General Functionality** and micro specific modules for documentation.

Definition in file **micro-common.h**.

micro-common.h

Go to the documentation of this file.

```
00001
00017 #ifndef __MICRO_COMMON_H__
00018 #define __MICRO_COMMON_H__
00019
00020
00023 void halInit(void);
00024
00027 void halReboot(void);
00028
00031 void halPowerUp(void);
00032
00035 void halPowerDown(void);
00036
00041 #define MICRO_DISABLE_WATCH_DOG_KEY 0xA5
00042
00046 void halInternalEnableWatchDog(void);
00047
00057 void halInternalDisableWatchDog(int8u magicKey);
00058
00073 void halCommonDelayMicroseconds(int16u us);
00074
00082 void halCommonDelayMilliseconds(int16u ms);
00083
00094 void halInternalAssertFailed(PGM_P filename, int linenumber);
00095
00100 int8u halGetResetInfo(void);
00101
00106 PGM_P halGetResetString(void);
00107
00113 void halStackSeedRandom(int32u seed);
00114
00117 int16u halCommonGetRandom(void);
00118
00119
00120 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00121
00157 enum SleepModes
00158 #else
00159 typedef int8u SleepModes;
00160 enum
00161 #endif
00162 {
00163   SLEEPMODE_RUNNING = 0,
00164   SLEEPMODE_IDLE = 1,
00165   SLEEPMODE_WAKETIMER = 2,
00166   SLEEPMODE_MAINTAINTIMER = 3,
00167   SLEEPMODE_NOTIMER = 4,
00168
00169   //The following SleepModes are deprecated.  Each micro's halSleep()
00170   //function will remap these modes to the appropriate replacement, as
00171   //necessary.
00172   SLEEPMODE_RESERVED = 6,
00173   SLEEPMODE_POWERDOWN = 7,
00174   SLEEPMODE_POWERSAVE = 8,
00175 };
00176
00183 void halSleep(SleepModes sleepMode);
00184
00185
00186 //Pull in the micro specific micro definitions.  The specific header is chosen
00187 //by the build include path pointing at the appropriate directory.
00188 #include "micro-specific.h"
00189
00190
00191 #endif //__MICRO_COMMON_H__
00192
00193
```


micro-specific.h File Reference

Go to the source code of this file.

Defines

#define	MILLISECOND_TICKS_PER_SECOND

Functions

void	halInternalInitSysTick (void)

#define	RESET_UNKNOWN
#define	RESET_LOW_POWER
#define	RESET_WINDOW_WATCHDOG
#define	RESET_INDEPENDENT_WATCHDOG
#define	RESET_SOFTWARE
#define	RESET_POR_PDR
#define	RESET_PIN
#define	RESET_UNSET

Detailed Description

Microcontroller General Functionality and STM32F103RET General Functionality for documentation.

Definition in file micro-specific.h.

micro-specific.h

Go to the documentation of this file.

```
00001
00019 #ifndef __MICRO_SPECIFIC_H__
00020 #define __MICRO_SPECIFIC_H__
00021
00022
00027 #define MILLISECOND_TICKS_PER_SECOND 1024
00028
00029
00030
00035 #define RESET_UNKNOWN                0
00036 #define RESET_LOW_POWER              1
00037 #define RESET_WINDOW_WATCHDOG        2
00038 #define RESET_INDEPENDENT_WATCHDOG   3
00039 #define RESET_SOFTWARE               4
00040 #define RESET_POR_PDR                5
00041 #define RESET_PIN                    6
00042 #define RESET_UNSET                255
00043
00049 void halInternalInitSysTick(void);
00050
00051
00052 #endif //__MICRO_SPECIFIC_H__
00053
```


network-manager.h File Reference

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

```
#include <CONFIGURATION_HEADER>
```

Go to the source code of this file.

Defines

#define	**NM_WARNING_LIMIT**
#define	**NM_WINDOW_SIZE**
#define	**NM_CHANNEL_MASK**
#define	**NM_WATCHLIST_SIZE**

Functions

void	**nmUtilWarningHandler** (void)
boolean	**nmUtilProcessIncoming** (**EmberApsFrame** *apsFrame, **int8u** messageLength, **int8u** *message)
EmberStatus	**nmUtilChangeChannelRequest** (void)

Detailed Description

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

Definition in file **network-manager.h**.

network-manager.h

Go to the documentation of this file.

```
00001
00090 #include CONFIGURATION_HEADER
00091
00092 // The application is notified via nmUtilWarningHandler
00093 // if NM_WARNING_LIMIT unsolicited scan reports are received
00094 // within NM_WINDOW_SIZE minutes.  To save flash and RAM,
00095 // the actual timing is approximate.
00096 #ifndef NM_WARNING_LIMIT
00097   #define NM_WARNING_LIMIT 16
00098 #endif
00099
00100 #ifndef NM_WINDOW_SIZE
00101   #define NM_WINDOW_SIZE 4
00102 #endif
00103
00104 // The channels that should be used by the network manager.
00105
00106 #ifndef NM_CHANNEL_MASK
00107   #define NM_CHANNEL_MASK EMBER_ALL_802_15_4_CHANNELS_MASK
00108 #endif
00109
00110 // The number of channels used in the NM_CHANNEL_MASK.
00111
00112 #ifndef NM_WATCHLIST_SIZE
00113   #define NM_WATCHLIST_SIZE 16
00114 #endif
00115
00122 void nmUtilWarningHandler(void);
00123
00132 boolean nmUtilProcessIncoming(EmberApsFrame *apsFrame,
00133                               int8u messageLength,
00134                               int8u* message);
00135
00139 EmberStatus nmUtilChangeChannelRequest(void);
00140
```


platform-common.h File Reference

Go to the source code of this file.

Master Program Memory Declarations

These are a set of defines for simple declarations of program memory.

#define	**PGM**
#define	**PGM_P**
#define	**PGM_PU**
#define	**PGM_NO_CONST**

Divide and Modulus Operations

Some platforms can perform divide and modulus operations on 32 bit quantities more efficiently when the divisor is only a 16 bit quantity. C compilers will always promote the divisor to 32 bits before performing the operation, so the following utility functions are instead required to take advantage of this optimisation.

#define	**halCommonUDiv32By16**(x, y)
#define	**halCommonSDiv32By16**(x, y)
#define	**halCommonUMod32By16**(x, y)
#define	**halCommonSMod32By16**(x, y)

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)

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)

Detailed Description

See **Common PLATFORM_HEADER Configuration** and micro specific modules for documentation.

Definition in file **platform-common.h**.

platform-common.h

Go to the documentation of this file.

```
00001
00020 #ifndef PLATCOMMONOKTOINCLUDE
00021   //  This header should only be included by a PLATFORM_HEADER
00022   #error  platform-common.h should not be included directly
00023 #endif
00024
00025 #ifndef __PLATFORMCOMMON_H__
00026 #define __PLATFORMCOMMON_H__
00028 // Many of the common definitions must be explicitly enabled by the
00029 //  particular PLATFORM_HEADER being used
00031
00032
00034 #ifdef _HAL_USE_COMMON_PGM_
00035
00042   #define PGM     const
00043
00047   #define PGM_P   const char *
00048
00052   #define PGM_PU  const unsigned char *
00053
00054
00060   #define PGM_NO_CONST
00061
00062 #endif //_HAL_USE_COMMON_PGM_
00063
00064
00066 #ifdef _HAL_USE_COMMON_DIVMOD_
00067
00080   #define halCommonUDiv32By16(x, y) ((int16u) (((int32u) (x)) / ((int16u) (y))))
00081
00087   #define halCommonSDiv32By16(x, y) ((int16s) (((int32s) (x)) / ((int16s) (y))))
00088
00094   #define halCommonUMod32By16(x, y) ((int16u) (((int32u) (x)) % ((int16u) (y))))
00095
00101   #define halCommonSMod32By16(x, y) ((int16s) (((int32s) (x)) % ((int16s) (y))))
00102
00103 #endif //_HAL_USE_COMMON_DIVMOD_
00104
00105
00107 #ifdef _HAL_USE_COMMON_MEMUTILS_
00108
00120
00124   void halCommonMemCopy(void *dest, const void *src, int16u bytes);
00125
00126
00130   void halCommonMemSet(void *dest, int8u val, int16u bytes);
00131
00132
00136   int8s halCommonMemCompare(const void *source0, const void *source1, int16u bytes);
00137
00138
00143   int8s halCommonMemPGMCompare(const void *source0, void PGM *source1, int16u bytes);
00144
00149   void halCommonMemPGMCopy(void* dest, void PGM *source, int16u bytes);
00150
00154   #define MEMSET(d,v,l)  halCommonMemSet(d,v,l)
00155   #define MEMCOPY(d,s,l) halCommonMemCopy(d,s,l)
00156   #define MEMCOMPARE(s0,s1,l) halCommonMemCompare(s0, s1, l)
00157   #define MEMPGMCOMPARE(s0,s1,l) halCommonMemPGMCompare(s0, s1, l)
00158
00160 #endif //_HAL_USE_COMMON_MEMUTILS_
00161
00162
00163
00164
00165
00166
00167
00168
00169
00171 //  The following sections are common on all platforms
00173
00175
```



```
00179
00183 #define BIT(x) (1U << (x))  // Unsigned avoids compiler warnings re BIT(15)
00184
00188 #define BIT32(x) (((int32u) 1) << (x))
00189
00195 #define SETBIT(reg, bit)      reg |= BIT(bit)
00196
00202 #define SETBITS(reg, bits)    reg |= (bits)
00203
00209 #define CLEARBIT(reg, bit)    reg &= ~(BIT(bit))
00210
00216 #define CLEARBITS(reg, bits)  reg &= ~(bits)
00217
00221 #define READBIT(reg, bit)     (reg & (BIT(bit)))
00222
00227 #define READBITS(reg, bits)   (reg & (bits))
00228
00230
00231
00233
00237
00241 #define LOW_BYTE(n)                    ((int8u)((n) & 0xFF))
00242
00246 #define HIGH_BYTE(n)                   ((int8u)(LOW_BYTE((n) >> 8)))
00247
00252 #define HIGH_LOW_TO_INT(high, low) (                              \
00253                                     (( (int16u) (high) ) << 8) +  \
00254                                     (  (int16u) ( (low) & 0xFF))  \
00255                                    )
00256
00260 #define BYTE_0(n)                     ((int8u)((n) & 0xFF))
00261
00265 #define BYTE_1(n)                     ((int8u)(BYTE_0((n) >> 8)))
00266
00270 #define BYTE_2(n)                     ((int8u)(BYTE_0((n) >> 16)))
00271
00275 #define BYTE_3(n)                     ((int8u)(BYTE_0((n) >> 24)))
00276
00278
00279
00281
00285
00290 #define elapsedTimeInt8u(oldTime, newTime)       \
00291   ((int8u) ((int8u)(newTime) - (int8u)(oldTime)))
00292
00297 #define elapsedTimeInt16u(oldTime, newTime)      \
00298   ((int16u) ((int16u)(newTime) - (int16u)(oldTime)))
00299
00304 #define elapsedTimeInt32u(oldTime, newTime)      \
00305   ((int32u) ((int32u)(newTime) - (int32u)(oldTime)))
00306
00311 #define MAX_INT8U_VALUE       (0xFF)
00312 #define HALF_MAX_INT8U_VALUE  (0x80)
00313 #define timeGTorEqualInt8u(t1, t2)           \
00314   (elapsedTimeInt8u(t2, t1) <= (HALF_MAX_INT8U_VALUE))
00315
00320 #define MAX_INT16U_VALUE      (0xFFFF)
00321 #define HALF_MAX_INT16U_VALUE (0x8000)
00322 #define timeGTorEqualInt16u(t1, t2)          \
00323   (elapsedTimeInt16u(t2, t1) <= (HALF_MAX_INT16U_VALUE))
00324
00329 #define MAX_INT32U_VALUE      (0xFFFFFFFFL)
00330 #define HALF_MAX_INT32U_VALUE (0x80000000L)
00331 #define timeGTorEqualInt32u(t1, t2)          \
00332   (elapsedTimeInt32u(t2, t1) <= (HALF_MAX_INT32U_VALUE))
00333
00335
00336
00337
00338 #endif //__PLATFORMCOMMON_H__
00339
```


hal/host/serial.h File Reference

```
#include <yfuns.h>
```

Go to the source code of this file.

Enumerations

```
enum   SerialBaudRate {
         DEFINE_BAUD,
         DEFINE_BAUD,
         DEFINE_BAUD,
         DEFINE_BAUD,
         DEFINE_BAUD,
         DEFINE_BAUD,
         DEFINE_BAUD,
         DEFINE_BAUD,
         DEFINE_BAUD,
         DEFINE_BAUD,
         DEFINE_BAUD,
         DEFINE_BAUD,
         DEFINE_BAUD,
         DEFINE_BAUD,
         DEFINE_BAUD,
         DEFINE_BAUD,
         DEFINE_BAUD,
         DEFINE_BAUD,
         DEFINE_BAUD
       }
enum   NameOfType {
         DEFINE_PARITY,
         DEFINE_PARITY,
         DEFINE_PARITY
       }
```

Serial HAL APIs

These functions must be implemented by the HAL in order for the serial code to operate. Only the higher-level serial code uses these functions, so they should not be called directly. The HAL should also implement the appropriate interrupt handlers to drain the TX queues and fill the RX FIFO queue, as necessary.

EmberStatus	halInternalUartInit (int8u port, SerialBaudRate rate, SerialParity parity, int8u stopBits)
int16u	halInternalPrintfWriteAvailable (void)
int16u	halInternalPrintfReadAvailable (void)
void	halInternalForcePrintf (boolean onOff)

Detailed Description

See Serial UART Communication and micro specific modules for documentation.

Definition in file hal/host/serial.h.

hal / host / serial.h

Go to the documentation of this file.

```
00001
00022 #ifndef __HAL_SERIAL_H__
00023 #define __HAL_SERIAL_H__
00024
00025 #include <yfuns.h>
00026
00027
00028 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00029
00033 enum SerialBaudRate
00034 #else
00035 #ifndef DEFINE_BAUD
00036 #define DEFINE_BAUD(num) BAUD_##num
00037 #endif
00038 typedef int8u SerialBaudRate;
00039 enum
00040 #endif //DOXYGEN_SHOULD_SKIP_THIS
00041 {
00042   DEFINE_BAUD(300) = 0,  // BAUD_300
00043   DEFINE_BAUD(600) = 1,  // BAUD_600
00044   DEFINE_BAUD(900) = 2,  // etc...
00045   DEFINE_BAUD(1200) = 3,
00046   DEFINE_BAUD(2400) = 4,
00047   DEFINE_BAUD(4800) = 5,
00048   DEFINE_BAUD(9600) = 6,
00049   DEFINE_BAUD(14400) = 7,
00050   DEFINE_BAUD(19200) = 8,
00051   DEFINE_BAUD(28800) = 9,
00052   DEFINE_BAUD(38400) = 10,
00053   DEFINE_BAUD(50000) = 11,
00054   DEFINE_BAUD(57600) = 12,
00055   DEFINE_BAUD(76800) = 13,
00056   DEFINE_BAUD(100000) = 14,
00057   DEFINE_BAUD(115200) = 15,
00058   DEFINE_BAUD(230400) = 16,
00059   DEFINE_BAUD(460800) = 17,
00060   DEFINE_BAUD(CUSTOM) = 18
00061 };
00062
00063
00064 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00065
00069 enum NameOfType
00070 #else
00071 #ifndef DEFINE_PARITY
00072 #define DEFINE_PARITY(val) PARITY_##val
00073 #endif
00074 typedef int8u SerialParity;
00075 enum
00076 #endif //DOXYGEN_SHOULD_SKIP_THIS
00077 {
00078   DEFINE_PARITY(NONE) = 0,  // PARITY_NONE
00079   DEFINE_PARITY(ODD) = 1,   // PARITY_ODD
00080   DEFINE_PARITY(EVEN) = 2   // PARITY_EVEN
00081 };
00082
00108 EmberStatus halInternalUartInit(int8u port,
00109                                 SerialBaudRate rate,
00110                                 SerialParity parity,
00111                                 int8u stopBits);
00112
00118 int16u halInternalPrintfWriteAvailable(void);
00119
00125 int16u halInternalPrintfReadAvailable(void);
00126
00132 void halInternalForcePrintf(boolean onOff);
00133
00134
00135 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00136 //Refer to uart.h for better documentation of fflush and stdout.
00137
00138 #ifndef fflush
00139   size_t fflush(int handle);
```



```
00140 #endif
00141
00142 #ifndef stdout
00143   #define stdout _LLIO_STDOUT
00144 #endif
00145
00146 #endif
00147
00150 #endif //__HAL_SERIAL_H__
00151
```


app/util/serial/serial.h File Reference

High-level serial communication functions. More...

Go to the source code of this file.

Defines

#define	**emberSerialWriteUsed**(port)

Functions

EmberStatus	**emberSerialInit** (**int8u** port, **SerialBaudRate** rate, SerialParity parity, **int8u** stopBits)
int16u	**emberSerialReadAvailable** (**int8u** port)
EmberStatus	**emberSerialReadByte** (**int8u** port, **int8u** *dataByte)
EmberStatus	**emberSerialReadLine** (**int8u** port, char *data, **int8u** max)
EmberStatus	**emberSerialReadPartialLine** (**int8u** port, char *data, **int8u** max, **int8u** *index)
int16u	**emberSerialWriteAvailable** (**int8u** port)
EmberStatus	**emberSerialWriteByte** (**int8u** port, **int8u** dataByte)
EmberStatus	**emberSerialWriteHex** (**int8u** port, **int8u** dataByte)
EmberStatus	**emberSerialWriteString** (**int8u** port, PGM_P string)
XAP2B_PAGEZERO_ON **EmberStatus**	**emberSerialPrintf** (**int8u** port, PGM_P formatString,...)
XAP2B_PAGEZERO_OFF XAP2B_PAGEZERO_ON **EmberStatus**	**emberSerialPrintfLine** (**int8u** port, PGM_P formatString,...)
XAP2B_PAGEZERO_OFF XAP2B_PAGEZERO_ON **EmberStatus**	**emberSerialPrintCarriageReturn** (**int8u** port)
XAP2B_PAGEZERO_OFF **EmberStatus**	**emberSerialPrintfVarArg** (**int8u** port, PGM_P formatString, va_list ap)
EmberStatus	**emberSerialWriteData** (**int8u** port, **int8u** *data, **int8u** length)
XAP2B_PAGEZERO_ON **EmberStatus**	**emberSerialWaitSend** (**int8u** port)
XAP2B_PAGEZERO_OFF **EmberStatus**	**emberSerialGuaranteedPrintf** (**int8u** port, PGM_P formatString,...)
void	**emberSerialBufferTick** (void)
void	**emberSerialFlushRx** (**int8u** port)

Printf Prototypes

These prototypes are for the internal printf implementation, in case it is desired to use it elsewhere. See the code for **emberSerialPrintf()** for an example of printf usage.

typedef **EmberStatus**(	**emPrintfFlushHandler**)(**int8u** flushVar, **int8u** *contents, **int8u** length)
int8u	**emPrintfInternal** (**emPrintfFlushHandler handler**, **int8u** port, PGM_P buff, va_list list)

Detailed Description

High-level serial communication functions.

See **Serial Communication** for documentation.

Definition in file **app/util/serial/serial.h**.

app/util/serial/serial.h

Go to the documentation of this file.

```
00001
00012 #ifndef __SERIAL_H__
00013 #define __SERIAL_H__
00014
00015 #ifndef __HAL_H__
00016   #error hal/hal.h should be included first
00017 #endif
00018
00019 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00020 #include <stdarg.h>
00021
00022 //Rx FIFO Full indicator
00023 #define RX_FIFO_FULL (0xFFFF)
00024
00025 #endif // DOXYGEN_SHOULD_SKIP_THIS
00026
00136 EmberStatus emberSerialInit(int8u port,
00137                             SerialBaudRate rate,
00138                             SerialParity parity,
00139                             int8u stopBits);
00140
00148 int16u emberSerialReadAvailable(int8u port);
00149
00167 EmberStatus emberSerialReadByte(int8u port, int8u *dataByte);
00168
00183 EmberStatus emberSerialReadLine(int8u port, char *data, int8u max);
00184
00208 EmberStatus emberSerialReadPartialLine(int8u port, char *data, int8u max, int8u
*index);
00209
00218 int16u emberSerialWriteAvailable(int8u port);
00219
00227 #define emberSerialWriteUsed(port)  \
00228   (emSerialTxQueueSizes[port] - emberSerialWriteAvailable(port))
00229
00243 EmberStatus emberSerialWriteByte(int8u port, int8u dataByte);
00244
00259 EmberStatus emberSerialWriteHex(int8u port, int8u dataByte);
00260
00273 EmberStatus emberSerialWriteString(int8u port, PGM_P string);
00274
00299 XAP2B_PAGEZERO_ON
00300 EmberStatus emberSerialPrintf(int8u port, PGM_P formatString, ...);
00301 XAP2B_PAGEZERO_OFF
00302
00318 XAP2B_PAGEZERO_ON
00319 EmberStatus emberSerialPrintfLine(int8u port, PGM_P formatString, ...);
00320 XAP2B_PAGEZERO_OFF
00321
00332 XAP2B_PAGEZERO_ON
00333 EmberStatus emberSerialPrintCarriageReturn(int8u port);
00334 XAP2B_PAGEZERO_OFF
00335
00336
00349 EmberStatus emberSerialPrintfVarArg(int8u port, PGM_P formatString, va_list ap);
00350
00366 EmberStatus emberSerialWriteData(int8u port, int8u *data, int8u length);
00367
00368 //Host HALs do not use stack buffers.
00369 #ifndef HAL_HOST
00370
00388 EmberStatus emberSerialWriteBuffer(int8u port, EmberMessageBuffer buffer, int8u start,
int8u length);
00389 #endif //HAL_HOST
00390
00403 XAP2B_PAGEZERO_ON
00404 EmberStatus emberSerialWaitSend(int8u port);
00405 XAP2B_PAGEZERO_OFF
00406
00427 EmberStatus emberSerialGuaranteedPrintf(int8u port, PGM_P formatString, ...);
00428
00434 void emberSerialBufferTick(void);
00435
```



```
00441 void emberSerialFlushRx(int8u port);
00442
00443
00444
00445
00466 typedef EmberStatus (emPrintfFlushHandler)(int8u flushVar,
00467                                             int8u *contents,
00468                                             int8u length);
00469
00470
00488 int8u emPrintfInternal(emPrintfFlushHandler handler, int8u port, PGM_P buff, va_list
list);
00489
00490
00495 #endif // __SERIAL_H__
00496
```


spi-protocol-common.h File Reference

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

Go to the source code of this file.

Functions

void	**halNcpSerialInit** (void)
void	**halNcpSerialPowerup** (void)
void	**halNcpSerialPowerdown** (void)
EzspStatus	**halNcpHardReset** (void)
EzspStatus	**halNcpHardResetReqBootload** (**boolean** requestBootload)
void	**halNcpWakeUp** (void)
void	**halNcpSendCommand** (void)
void	**halNcpSendRawCommand** (void)
EzspStatus	**halNcpPollForResponse** (void)
void	**halNcpIsAwakeIsr** (**boolean** isAwake)
boolean	**halNcpHasData** (void)
boolean	**halNcpVerifySpiProtocolVersion** (void)
boolean	**halNcpVerifySpiProtocolActive** (void)

Variables

int8u *	**halNcpFrame**
int8u	**halNcpSpipErrorByte**

Detailed Description

See **SPI Protocol** and micro specific modules for documentation.

Definition in file **spi-protocol-common.h**.

spi-protocol-common.h

Go to the documentation of this file.

```
00001
00020 #ifndef __SPI_PROTOCOL_COMMON_H__
00021 #define __SPI_PROTOCOL_COMMON_H__
00022
00023 #include "app/util/ezsp/ezsp-enum.h"
00024
00032 extern int8u *halNcpFrame;
00033
00041 extern int8u halNcpSpipErrorByte;
00042
00046 void halNcpSerialInit(void);
00047
00052 void halNcpSerialPowerup(void);
00053
00057 void halNcpSerialPowerdown(void);
00058
00071 EzspStatus halNcpHardReset(void);
00072
00087 EzspStatus halNcpHardResetReqBootload(boolean requestBootload);
00088
00098 void halNcpWakeUp(void);
00099
00111 void halNcpSendCommand(void);
00112
00124 void halNcpSendRawCommand(void);
00125
00134 EzspStatus halNcpPollForResponse(void);
00135
00143 void halNcpIsAwakeIsr(boolean isAwake);
00144
00149 boolean halNcpHasData(void);
00150
00151
00159 boolean halNcpVerifySpiProtocolVersion(void);
00160
00168 boolean halNcpVerifySpiProtocolActive(void);
00169
00171
00172
00173 //Pull in the micro specific spi protocol definitions.  The specific header is
00174 //chosen by the build include path pointing at the appropriate directory.
00175 #include "spi-protocol-specific.h"
00176
00177
00178 #endif // __SPI_PROTOCOL_COMMON_H__
00179
```


spi-protocol-specific.h File Reference

Go to the source code of this file.

SPI Protocol Interface

#define	SPIP_nSSEL_PORT
#define	SPIP_nSSEL_PIN
#define	SPIP_MOSI_PORT
#define	SPIP_MOSI_PIN
#define	SPIP_MISO_PORT
#define	SPIP_MISO_PIN
#define	SPIP_SCLK_PORT
#define	SPIP_SCLK_PIN
#define	SPIP_nHOST_INT_PORT
#define	SPIP_nHOST_INT_PIN
#define	SPIP_nWAKE_PORT
#define	SPIP_nWAKE_PIN
#define	SPIP_nRESET_PORT
#define	SPIP_nRESET_PIN

SPI Protocol timing parameters.

Note:
Remember: TIM2 is configured to produce a 125us tick.

#define	WAIT_SECTION_TIMEOUT
#define	WAKE_HANDSHAKE_TIMEOUT
#define	STARTUP_TIMEOUT
#define	INTER_COMMAND_SPACING
#define	NCP_RESET_DELAY

Detailed Description

See **SPI Protocol** and **STM32F103RET Specific SPI Protocol** for documentation.

Definition in file **spi-protocol-specific.h**.

---

spi-protocol-specific.h

Go to the documentation of this file.

```
00001
00022 #ifndef __SPI_PROTOCOL_SPECIFIC_H__
00023 #define __SPI_PROTOCOL_SPECIFIC_H__
00024
00029
00033 #define SPIP_nSSEL_PORT     GPIOA
00034
00037 #define SPIP_nSSEL_PIN      GPIO_Pin_4
00038
00039
00043 #define SPIP_MOSI_PORT      GPIOA
00044
00047 #define SPIP_MOSI_PIN       GPIO_Pin_7
00048
00052 #define SPIP_MISO_PORT      GPIOA
00053
00056 #define SPIP_MISO_PIN       GPIO_Pin_6
00057
00061 #define SPIP_SCLK_PORT      GPIOA
00062
00065 #define SPIP_SCLK_PIN       GPIO_Pin_5
00066
00070 #define SPIP_nHOST_INT_PORT GPIOC
00071
00074 #define SPIP_nHOST_INT_PIN  GPIO_Pin_4
00075
00079 #define SPIP_nWAKE_PORT     GPIOC
00080
00083 #define SPIP_nWAKE_PIN      GPIO_Pin_5
00084
00088 #define SPIP_nRESET_PORT    GPIOB
00089
00092 #define SPIP_nRESET_PIN     GPIO_Pin_0
00093
00106 #define WAIT_SECTION_TIMEOUT    (1600) //200ms
00107
00110 #define WAKE_HANDSHAKE_TIMEOUT    (80) //10ms
00111
00114 #define STARTUP_TIMEOUT        (60000) //7500ms
00115
00118 #define INTER_COMMAND_SPACING      (8) //1ms
00119
00122 #define NCP_RESET_DELAY  (26)
00123
00127 #endif // __SPI_PROTOCOL_SPECIFIC_H__
00128
```


stm32f10x_conf.h File Reference

```
#include "stm32f10x_adc.h"
#include "stm32f10x_bkp.h"
#include "stm32f10x_can.h"
#include "stm32f10x_cec.h"
#include "stm32f10x_crc.h"
#include "stm32f10x_dac.h"
#include "stm32f10x_dbgmcu.h"
#include "stm32f10x_dma.h"
#include "stm32f10x_exti.h"
#include "stm32f10x_flash.h"
#include "stm32f10x_fsmc.h"
#include "stm32f10x_gpio.h"
#include "stm32f10x_i2c.h"
#include "stm32f10x_iwdg.h"
#include "stm32f10x_pwr.h"
#include "stm32f10x_rcc.h"
#include "stm32f10x_rtc.h"
#include "stm32f10x_sdio.h"
#include "stm32f10x_spi.h"
#include "stm32f10x_tim.h"
#include "stm32f10x_usart.h"
#include "stm32f10x_wwdg.h"
#include "misc.h"
```

Go to the source code of this file.

Defines

```
#define assert_param(condition)
```

Functions

```
void halInternalAssertFailed (const char *filename, int linenumber)
```

Detailed Description

Microcontroller General Functionality and **STM32F103RET General Functionality** for documentation.

Definition in file **stm32f10x_conf.h**.

---

stm32f10x_conf.h

Go to the documentation of this file.

```
00001
00027 #ifndef __STM32F10x_CONF_H
00028 #define __STM32F10x_CONF_H
00029
00030
00031 //Peripheral header file inclusion.  There is a header per peripheral source
00032 //found in the library.
00033 #include "stm32f10x_adc.h"
00034 #include "stm32f10x_bkp.h"
00035 #include "stm32f10x_can.h"
00036 #include "stm32f10x_cec.h"
00037 #include "stm32f10x_crc.h"
00038 #include "stm32f10x_dac.h"
00039 #include "stm32f10x_dbgmcu.h"
00040 #include "stm32f10x_dma.h"
00041 #include "stm32f10x_exti.h"
00042 #include "stm32f10x_flash.h"
00043 #include "stm32f10x_fsmc.h"
00044 #include "stm32f10x_gpio.h"
00045 #include "stm32f10x_i2c.h"
00046 #include "stm32f10x_iwdg.h"
00047 #include "stm32f10x_pwr.h"
00048 #include "stm32f10x_rcc.h"
00049 #include "stm32f10x_rtc.h"
00050 #include "stm32f10x_sdio.h"
00051 #include "stm32f10x_spi.h"
00052 #include "stm32f10x_tim.h"
00053 #include "stm32f10x_usart.h"
00054 #include "stm32f10x_wwdg.h"
00055 //misc.h is for High level functions for NVIC and SysTick, which
00056 //are add-on to CMSIS functions.
00057 #include "misc.h"
00058
00059
00060 //The library uses it's own assert macro (assert_param), so link the library's
00061 //assert to our usual assert.
00062 #if !defined(SIMPLER_ASSERT_REBOOT)
00063
00067   void halInternalAssertFailed(const char * filename, int linenumber);
00068
00075   #define assert_param(condition)                                    \
00076         do {                                                         \
00077           if (! (condition)) {                                       \
00078             halInternalAssertFailed(__SOURCEFILE__, __LINE__); \
00079           }                                                          \
00080         } while(0)
00081 #else
00082   #define assert_param(condition) \
00083             do { if( !(condition) ) while(1){} } while(0)
00084 #endif
00085
00086
00087 #endif /* __STM32F10x_CONF_H */
00088
```


system-timer.h File Reference

Go to the source code of this file.

Functions

int16u	halInternalStartSystemTimer (void)
int16u	halCommonGetInt16uMillisecondTick (void)
int32u	halCommonGetInt32uMillisecondTick (void)
int16u	halCommonGetInt16uQuarterSecondTick (void)
void	halCommonSetSystemTime (int32u time)

Detailed Description

See System Timer for documentation.

Definition in file system-timer.h.

system-timer.h

Go to the documentation of this file.

```
00001
00029 #ifndef __SYSTEM_TIMER_H__
00030 #define __SYSTEM_TIMER_H__
00031
00032
00039 int16u halInternalStartSystemTimer(void);
00040
00041
00049 int16u halCommonGetInt16uMillisecondTick(void);
00050
00058 int32u halCommonGetInt32uMillisecondTick(void);
00059
00067 int16u halCommonGetInt16uQuarterSecondTick(void);
00068
00074 void halCommonSetSystemTime(int32u time);
00075
00076
00077 #endif //__SYSTEM_TIMER_H__
00078
```


uart.h File Reference

```
#include <yfuns.h>
```

Go to the source code of this file.

Defines

```
#define  stdout
```

Functions

```
size_t  fflush (int handle)
```

Detailed Description

See **Serial UART Communication** and **STM32F103RET Specific UART** for documentation.

Definition in file **uart.h**.

---

uart.h

Go to the documentation of this file.

```
00001
00018 #ifndef __UART_H__
00019 #define __UART_H__
00020 #include <yfuns.h>
00021
00033 size_t fflush(int handle);
00034
00041 #define stdout _LLIO_STDOUT
00042
00043 #endif //__UART_H__
00044
```


zigbee-device-common.h File Reference

ZigBee Device Object (ZDO) functions available on all platforms. See **ZigBee Device Object (ZDO) Information** for documentation. More...

Go to the source code of this file.

Defines

#define	**ZDO_MESSAGE_OVERHEAD**

Service Discovery Functions

EmberStatus	**emberNodeDescriptorRequest** (**EmberNodeId** target, **EmberApsOption** options)
EmberStatus	**emberPowerDescriptorRequest** (**EmberNodeId** target, **EmberApsOption** options)
EmberStatus	**emberSimpleDescriptorRequest** (**EmberNodeId** target, **int8u** targetEndpoint, **EmberApsOption** options)
EmberStatus	**emberActiveEndpointsRequest** (**EmberNodeId** target, **EmberApsOption** options)

Binding Manager Functions

EmberStatus	**emberBindRequest** (**EmberNodeId** target, **EmberEUI64** source, **int8u** sourceEndpoint, **int16u** clusterId, **int8u** type, **EmberEUI64** destination, **EmberMulticastId** groupAddress, **int8u** destinationEndpoint, **EmberApsOption** options)
EmberStatus	**emberUnbindRequest** (**EmberNodeId** target, **EmberEUI64** source, **int8u** sourceEndpoint, **int16u** clusterId, **int8u** type, **EmberEUI64** destination, **EmberMulticastId** groupAddress, **int8u** destinationEndpoint, **EmberApsOption** options)

Node Manager Functions

EmberStatus	**emberLqiTableRequest** (**EmberNodeId** target, **int8u** startIndex, **EmberApsOption** options)
EmberStatus	**emberRoutingTableRequest** (**EmberNodeId** target, **int8u** startIndex, **EmberApsOption** options)
EmberStatus	**emberBindingTableRequest** (**EmberNodeId** target, **int8u** startIndex, **EmberApsOption** options)
EmberStatus	**emberLeaveRequest** (**EmberNodeId** target, **EmberEUI64** deviceAddress, **int8u** leaveRequestFlags, **EmberApsOption** options)
EmberStatus	**emberPermitJoiningRequest** (**EmberNodeId** target, **int8u** duration, **int8u** authentication, **EmberApsOption** options)
void	**emberSetZigDevRequestRadius** (**int8u** radius)
int8u	**emberGetZigDevRequestRadius** (void)
int8u	**emberGetLastZigDevRequestSequence** (void)

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**.

zigbee-device-common.h

Go to the documentation of this file.

```
00001
00016 #define ZDO_MESSAGE_OVERHEAD 1
00017
00036 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00037 EmberStatus emberNodeDescriptorRequest(EmberNodeId target,
00038                                        EmberApsOption options);
00039 #else
00040 // Macroized to save code space.
00041 EmberStatus emberSendZigDevRequestTarget(EmberNodeId target,
00042                                          int16u clusterId,
00043                                          EmberApsOption options);
00044 #define emberNodeDescriptorRequest(target, opts)       \
00045 (emberSendZigDevRequestTarget((target), NODE_DESCRIPTOR_REQUEST, (opts)))
00046 #endif
00047
00063 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00064 EmberStatus emberPowerDescriptorRequest(EmberNodeId target,
00065                                        EmberApsOption options);
00066 #else
00067 // Macroized to save code space.
00068 #define emberPowerDescriptorRequest(target, opts)      \
00069 (emberSendZigDevRequestTarget((target), POWER_DESCRIPTOR_REQUEST, (opts)))
00070 #endif
00071
00090 EmberStatus emberSimpleDescriptorRequest(EmberNodeId target,
00091                                          int8u targetEndpoint,
00092                                          EmberApsOption options);
00093
00106 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00107 EmberStatus emberActiveEndpointsRequest(EmberNodeId target,
00108                                        EmberApsOption options);
00109 #else
00110 // Macroized to save code space.
00111 #define emberActiveEndpointsRequest(target, opts)      \
00112 (emberSendZigDevRequestTarget((target), ACTIVE_ENDPOINTS_REQUEST, (opts)))
00113 #endif
00114
00144 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00145 EmberStatus emberBindRequest(EmberNodeId target,
00146                              EmberEUI64 source,
00147                              int8u sourceEndpoint,
00148                              int16u clusterId,
00149                              int8u type,
00150                              EmberEUI64 destination,
00151                              EmberMulticastId groupAddress,
00152                              int8u destinationEndpoint,
00153                              EmberApsOption options);
00154 #else
00155 // Macroized to save code space.
00156 #define emberBindRequest(target,                                        \
00157                          src,                                           \
00158                          srcEndpt,                                      \
00159                          cluster,                                       \
00160                          type,                                          \
00161                          dest,                                          \
00162                          groupAddress,                                  \
00163                          destEndpt,                                     \
00164                          opts)                                          \
00165                                                                         \
00166  (emberSendZigDevBindRequest((target),                                 \
00167                              BIND_REQUEST,                              \
00168                              (src), (srcEndpt), (cluster),              \
00169                              (type), (dest), (groupAddress),            \
00170                              (destEndpt), (opts)))
00171
00172 EmberStatus emberSendZigDevBindRequest(EmberNodeId target,
00173                                        int16u bindClusterId,
00174                                        EmberEUI64 source,
00175                                        int8u sourceEndpoint,
00176                                        int16u clusterId,
00177                                        int8u type,
00178                                        EmberEUI64 destination,
00179                                        EmberMulticastId groupAddress,
```



```
00180                                       int8u destinationEndpoint,
00181                                       EmberApsOption options);
00182 #endif
00183
00210 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00211 EmberStatus emberUnbindRequest(EmberNodeId target,
00212                                EmberEUI64 source,
00213                                int8u sourceEndpoint,
00214                                int16u clusterId,
00215                                int8u type,
00216                                EmberEUI64 destination,
00217                                EmberMulticastId groupAddress,
00218                                int8u destinationEndpoint,
00219                                EmberApsOption options);
00220 #else
00221 // Macroized to save code space.
00222 #define emberUnbindRequest(target,                                      \
00223                            src,                                         \
00224                            srcEndpt,                                    \
00225                            cluster,                                     \
00226                            type,                                        \
00227                            dest,                                        \
00228                            groupAddress,                                \
00229                            destEndpt,                                   \
00230                            opts)                                        \
00231                                                                         \
00232  (emberSendZigDevBindRequest((target),                                  \
00233                              UNBIND_REQUEST,                            \
00234                              (src), (srcEndpt), (cluster),              \
00235                              (type), (dest), (groupAddress),            \
00236                              (destEndpt), (opts)))
00237 #endif
00238
00261 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00262 EmberStatus emberLqiTableRequest(EmberNodeId target,
00263                                  int8u startIndex,
00264                                  EmberApsOption options);
00265 #else
00266 #define emberLqiTableRequest(target, startIndex, options) \
00267   (emberTableRequest(LQI_TABLE_REQUEST, (target), (startIndex), (options)))
00268
00269 EmberStatus emberTableRequest(int16u clusterId,
00270                               EmberNodeId target,
00271                               int8u startIndex,
00272                               EmberApsOption options);
00273 #endif
00274
00291 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00292 EmberStatus emberRoutingTableRequest(EmberNodeId target,
00293                                      int8u startIndex,
00294                                      EmberApsOption options);
00295 #else
00296 #define emberRoutingTableRequest(target, startIndex, options) \
00297   (emberTableRequest(ROUTING_TABLE_REQUEST, (target), (startIndex), (options)))
00298 #endif
00299
00317 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00318 EmberStatus emberBindingTableRequest(EmberNodeId target,
00319                                      int8u startIndex,
00320                                      EmberApsOption options);
00321 #else
00322 #define emberBindingTableRequest(target, startIndex, options) \
00323   (emberTableRequest(BINDING_TABLE_REQUEST, (target), (startIndex), (options)))
00324 #endif
00325
00345 EmberStatus emberLeaveRequest(EmberNodeId target,
00346                               EmberEUI64 deviceAddress,
00347                               int8u leaveRequestFlags,
00348                               EmberApsOption options);
00349
00366 EmberStatus emberPermitJoiningRequest(EmberNodeId target,
00367                                       int8u duration,
00368                                       int8u authentication,
00369                                       EmberApsOption options);
00370
00371 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00372
00377 void emberSetZigDevRequestRadius(int8u radius);
00378
00384 int8u emberGetZigDevRequestRadius(void);
00385 #else
```



```
00386 extern int8u zigDevRequestRadius;
00387 #define emberGetZigDevRequestRadius() (zigDevRequestRadius)
00388 #define emberSetZigDevRequestRadius(x)        (zigDevRequestRadius=x)
00389 #endif
00390
00396 int8u emberGetLastZigDevRequestSequence(void);
00397
00400 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00401 //------------------------------------------------------------------------------
00402 // Utility functions used by the library code.
00403
00404 EmberStatus emberSendZigDevRequest(EmberNodeId destination,
00405                                    int16u clusterId,
00406                                    EmberApsOption options,
00407                                    int8u *contents,
00408                                    int8u length);
00409
00419 int8u emberNextZigDevRequestSequence(void);
00420
00421 #endif // DOXYGEN_SHOULD_SKIP_THIS
00422
```


zigbee-device-host.h File Reference

ZigBee Device Object (ZDO) functions not provided by the stack. See **ZigBee Device Object (ZDO) Information** for documentation. More...

Go to the source code of this file.

Device Discovery Functions

EmberStatus	**emberNetworkAddressRequest** (**EmberEUI64** target, **boolean** reportKids, **int8u** childStartIndex)
EmberStatus	**emberIeeeAddressRequest** (**EmberNodeId** target, **boolean** reportKids, **int8u** childStartIndex, **EmberApsOption** options)

Service Discovery Functions

EmberStatus	**ezspMatchDescriptorsRequest** (**EmberNodeId** target, **int16u** profile, **int8u** inCount, **int8u** outCount, **int16u** *inClusters, **int16u** *outClusters, **EmberApsOption** options)

Binding Manager Functions

EmberStatus	**ezspEndDeviceBindRequest** (**EmberNodeId** localNodeId, **EmberEUI64** localEui64, **int8u** endpoint, **int16u** profile, **int8u** inCount, **int8u** outCount, **int16u** *inClusters, **int16u** *outClusters, **EmberApsOption** options)

Function to Decode Address Response Messages

EmberNodeId	**ezspDecodeAddressResponse** (**int8u** *response, **EmberEUI64** eui64Return)

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**.

zigbee-device-host.h

Go to the documentation of this file.

```
00001
00104 EmberStatus emberNetworkAddressRequest(EmberEUI64 target,
00105                                        boolean reportKids,
00106                                        int8u childStartIndex);
00107
00125 EmberStatus emberIeeeAddressRequest(EmberNodeId target,
00126                                     boolean reportKids,
00127                                     int8u childStartIndex,
00128                                     EmberApsOption options);
00157 EmberStatus ezspMatchDescriptorsRequest(EmberNodeId target,
00158                                         int16u profile,
00159                                         int8u inCount,
00160                                         int8u outCount,
00161                                         int16u *inClusters,
00162                                         int16u *outClusters,
00163                                         EmberApsOption options);
00189 EmberStatus ezspEndDeviceBindRequest(EmberNodeId localNodeId,
00190                                      EmberEUI64 localEui64,
00191                                      int8u endpoint,
00192                                      int16u profile,
00193                                      int8u inCount,
00194                                      int8u outCount,
00195                                      int16u *inClusters,
00196                                      int16u *outClusters,
00197                                      EmberApsOption options);
00216 EmberNodeId ezspDecodeAddressResponse(int8u *response,
00217                                       EmberEUI64 eui64Return);
00218
```


app

app Directory Reference

Directories

directory	util

util Directory Reference

Directories

directory	**bootload**
directory	**common**
directory	**ezsp**
directory	**serial**
directory	**zigbee-framework**

bootload Directory Reference

Files

file	**bootload-ezsp-utils.h** [code]
file	**bootload-utils.h** [code]

common Directory Reference

Files

file	**form-and-join.h** [code]
file	**form-and-join3_2.h** [code]

ezsp Directory Reference

Files

file **ezsp-host-configuration-defaults.h** [code]

serial Directory Reference

Files

file	**command-interpreter2.h** [code]
file	**app/util/serial/serial.h** [code]

zigbee-framework Directory Reference

Files

file	**ami-inter-pan-host.h** [code]
file	**ami-inter-pan.h** [code]
file	**fragment-host.h** [code]
file	**network-manager.h** [code]
file	**zigbee-device-common.h** [code]
file	**zigbee-device-host.h** [code]

hal Directory Reference

Directories

directory	**host**

Files

file	**hal.h** [code]

host Directory Reference

Directories

directory	**cortexm3**
directory	**generic**

Files

file	**bootloader-eeprom.h** [code]
file	**button-common.h** [code]
file	**crc.h** [code]
file	**led-common.h** [code]
file	**micro-common.h** [code]
file	**hal/host/serial.h** [code]
file	**spi-protocol-common.h** [code]
file	**system-timer.h** [code]

cortexm3 Directory Reference

Directories

directory	stm32f103ret

stm32f103ret Directory Reference

Directories

directory	compiler

Files

file	adc.h [code]
file	button-specific.h [code]
file	buzzer.h [code]
file	led-specific.h [code]
file	micro-specific.h [code]
file	spi-protocol-specific.h [code]
file	stm32f10x_conf.h [code]
file	uart.h [code]

compiler Directory Reference

Files

file	**iar-st.h** [code]

generic Directory Reference

Directories

directory	compiler

compiler Directory Reference

Files

file **platform-common.h** [code]

stack Directory Reference

Directories

directory	**config**
directory	**include**

config Directory Reference

Files

file	**ember-configuration-defaults.h** [code]

include Directory Reference

Files

file	**cbke-crypto-engine.h** [code]
file	**ember-types.h** [code]
file	**error-def.h** [code]
file	**error.h** [code]

- a -

- b -

- BYTE_3 : **platform-common.h**

- c -

Index - _ - **a** - **b** - **c** - **d** - **e** - **f** - **h** - **i** - **j** - **l** - **m** - **n** - **p** - **r** - **s** - **t** - **u** - **w** - **z** -

- d -

- e -

- EMBER_APS_UNICAST_MESSAGE_COUNT : **ember-configuration-defaults.h**
- EMBER_ASSERT_SERIAL_PORT : **ember-configuration-defaults.h**
- EMBER_BAD_ARGUMENT : **error-def.h**
- EMBER_BINDING_HAS_CHANGED : **error-def.h**
- EMBER_BINDING_INDEX_OUT_OF_RANGE : **error-def.h**
- EMBER_BINDING_IS_ACTIVE : **error-def.h**
- EMBER_BINDING_TABLE_SIZE : **ember-configuration-defaults.h**
- EMBER_BINDING_TABLE_TOKEN_SIZE : **ember-configuration-defaults.h**
- EMBER_BROADCAST_ADDRESS : **ember-types.h**
- EMBER_BROADCAST_ALARM_CLUSTER : **ember-types.h**
- EMBER_BROADCAST_ALARM_DATA_SIZE : **ember-configuration-defaults.h**
- EMBER_BROADCAST_ENDPOINT : **ember-types.h**
- EMBER_BROADCAST_TABLE_SIZE : **ember-configuration-defaults.h**
- EMBER_CACHED_UNICAST_ALARM_CLUSTER : **ember-types.h**
- EMBER_CANNOT_JOIN_AS_ROUTER : **error-def.h**
- EMBER_CERTIFICATE_SIZE : **ember-types.h**
- EMBER_CERTIFICATE_TABLE_SIZE : **ember-configuration-defaults.h**
- EMBER_CHANNEL_CHANGED : **error-def.h**
- EMBER_CHILD_TABLE_SIZE : **ember-configuration-defaults.h**
- EMBER_CHILD_TABLE_TOKEN_SIZE : **ember-configuration-defaults.h**
- EMBER_CMD_ERR_ARGUMENT_OUT_OF_RANGE : **command-interpreter2.h**
- EMBER_CMD_ERR_ARGUMENT_SYNTAX_ERROR : **command-interpreter2.h**
- EMBER_CMD_ERR_INVALID_ARGUMENT_TYPE : **command-interpreter2.h**
- EMBER_CMD_ERR_NO_SUCH_COMMAND : **command-interpreter2.h**
- EMBER_CMD_ERR_PORT_PROBLEM : **command-interpreter2.h**
- EMBER_CMD_ERR_STRING_TOO_LONG : **command-interpreter2.h**
- EMBER_CMD_ERR_WRONG_NUMBER_OF_ARGUMENTS : **command-interpreter2.h**
- EMBER_CMD_SUCCESS : **command-interpreter2.h**
- EMBER_COMMAND_BUFFER_LENGTH : **command-interpreter2.h**
- EMBER_COMMAND_INTERPRETER_CONFIGURATION_ECHO : **command-interpreter2.h**
- EMBER_COORDINATOR : **ember-types.h**
- EMBER_COST_NOT_KNOWN : **error-def.h**
- EMBER_COUNTER_ALLOCATE_PACKET_BUFFER_FAILURE : **ember-types.h**
- EMBER_COUNTER_APS_DATA_RX_BROADCAST : **ember-types.h**
- EMBER_COUNTER_APS_DATA_RX_UNICAST : **ember-types.h**
- EMBER_COUNTER_APS_DATA_TX_BROADCAST : **ember-types.h**
- EMBER_COUNTER_APS_DATA_TX_UNICAST_FAILED : **ember-types.h**
- EMBER_COUNTER_APS_DATA_TX_UNICAST_RETRY : **ember-types.h**
- EMBER_COUNTER_APS_DATA_TX_UNICAST_SUCCESS : **ember-types.h**
- EMBER_COUNTER_APS_DECRYPTION_FAILURE : **ember-types.h**
- EMBER_COUNTER_APS_FRAME_COUNTER_FAILURE : **ember-types.h**
- EMBER_COUNTER_APS_LINK_KEY_NOT_AUTHORIZED : **ember-types.h**
- EMBER_COUNTER_ASH_FRAMING_ERROR : **ember-types.h**
- EMBER_COUNTER_ASH_OVERFLOW_ERROR : **ember-types.h**
- EMBER_COUNTER_ASH_OVERRUN_ERROR : **ember-types.h**
- EMBER_COUNTER_ASH_XOFF : **ember-types.h**
- EMBER_COUNTER_CHILD_REMOVED : **ember-types.h**
- EMBER_COUNTER_JOIN_INDICATION : **ember-types.h**
- EMBER_COUNTER_MAC_RX_BROADCAST : **ember-types.h**
- EMBER_COUNTER_MAC_RX_UNICAST : **ember-types.h**
- EMBER_COUNTER_MAC_TX_BROADCAST : **ember-types.h**
- EMBER_COUNTER_MAC_TX_UNICAST_FAILED : **ember-types.h**
- EMBER_COUNTER_MAC_TX_UNICAST_RETRY : **ember-types.h**
- EMBER_COUNTER_MAC_TX_UNICAST_SUCCESS : **ember-types.h**
- EMBER_COUNTER_NEIGHBOR_ADDED : **ember-types.h**
- EMBER_COUNTER_NEIGHBOR_REMOVED : **ember-types.h**
- EMBER_COUNTER_NEIGHBOR_STALE : **ember-types.h**
- EMBER_COUNTER_NWK_DECRYPTION_FAILURE : **ember-types.h**
- EMBER_COUNTER_NWK_FRAME_COUNTER_FAILURE : **ember-types.h**
- EMBER_COUNTER_PHY_TO_MAC_QUEUE_LIMIT_REACHED : **ember-types.h**
- EMBER_COUNTER_RELAYED_UNICAST : **ember-types.h**
- EMBER_COUNTER_ROUTE_DISCOVERY_INITIATED : **ember-types.h**
- EMBER_COUNTER_STRINGS : **ember-types.h**
- EMBER_COUNTER_TYPE_COUNT : **ember-types.h**
- EMBER_CURRENT_NETWORK_KEY : **ember-types.h**
- EMBER_DELIVERY_FAILED : **error-def.h**
- EMBER_DENY_JOIN : **ember-types.h**
- EMBER_DENY_KEY_REQUESTS : **ember-types.h**
- EMBER_DEVICE_LEFT : **ember-types.h**

- EMBER_DEVICE_UPDATE_STRINGS : **ember-types.h**
- EMBER_DISCOVERY_ACTIVE_NODE_ID : **ember-types.h**
- EMBER_DISCOVERY_TABLE_SIZE : **ember-configuration-defaults.h**
- EMBER_DISTRIBUTED_TRUST_CENTER_MODE : **ember-types.h**
- EMBER_DISTRIBUTED_TRUST_CENTER_MODE_ : **ember-types.h**
- EMBER_EEPROM_MFG_STACK_VERSION_MISMATCH : **error-def.h**
- EMBER_EEPROM_MFG_VERSION_MISMATCH : **error-def.h**
- EMBER_EEPROM_STACK_VERSION_MISMATCH : **error-def.h**
- EMBER_ENCRYPTION_KEY_SIZE : **ember-types.h**
- EMBER_END_DEVICE : **ember-types.h**
- EMBER_END_DEVICE_BIND_TIMEOUT : **ember-configuration-defaults.h**
- EMBER_END_DEVICE_POLL_TIMEOUT : **ember-configuration-defaults.h**
- EMBER_END_DEVICE_POLL_TIMEOUT_SHIFT : **ember-configuration-defaults.h**
- EMBER_ENERGY_SCAN : **ember-types.h**
- EMBER_ERR_BOOTLOADER_NO_IMAGE : **error-def.h**
- EMBER_ERR_BOOTLOADER_TRAP_TABLE_BAD : **error-def.h**
- EMBER_ERR_BOOTLOADER_TRAP_UNKNOWN : **error-def.h**
- EMBER_ERR_FATAL : **error-def.h**
- EMBER_ERR_FLASH_ERASE_FAIL : **error-def.h**
- EMBER_ERR_FLASH_PROG_FAIL : **error-def.h**
- EMBER_ERR_FLASH_VERIFY_FAILED : **error-def.h**
- EMBER_ERR_FLASH_WRITE_INHIBITED : **error-def.h**
- EMBER_ERROR_CODE_COUNT : **error.h**
- EMBER_EVENT_INACTIVE : **ember-types.h**
- EMBER_EVENT_MINUTE_TIME : **ember-types.h**
- EMBER_EVENT_MS_TIME : **ember-types.h**
- EMBER_EVENT_QS_TIME : **ember-types.h**
- EMBER_EVENT_ZERO_DELAY : **ember-types.h**
- EMBER_FRAGMENT_DELAY_MS : **ember-configuration-defaults.h**
- EMBER_FRAGMENT_MAX_WINDOW_SIZE : **ember-configuration-defaults.h**
- EMBER_FRAGMENT_WINDOW_SIZE : **ember-configuration-defaults.h**
- EMBER_GET_LINK_KEY_WHEN_JOINING : **ember-types.h**
- EMBER_GET_PRECONFIGURED_KEY_FROM_INSTALL_CODE : **ember-types.h**
- EMBER_GLOBAL_LINK_KEY : **ember-types.h**
- EMBER_GLOBAL_LINK_KEY_ : **ember-types.h**
- EMBER_HAVE_NETWORK_KEY : **ember-types.h**
- EMBER_HAVE_PRECONFIGURED_KEY : **ember-types.h**
- EMBER_HAVE_TRUST_CENTER_EUI64 : **ember-types.h**
- EMBER_HAVE_TRUST_CENTER_LINK_KEY : **ember-types.h**
- EMBER_HIGH_RAM_CONCENTRATOR : **ember-types.h**
- EMBER_HIGH_SECURITY_SECURED_REJOIN : **ember-types.h**
- EMBER_HIGH_SECURITY_UNSECURED_JOIN : **ember-types.h**
- EMBER_HIGH_SECURITY_UNSECURED_REJOIN : **ember-types.h**
- EMBER_INCOMING_BROADCAST : **ember-types.h**
- EMBER_INCOMING_BROADCAST_LOOPBACK : **ember-types.h**
- EMBER_INCOMING_MULTICAST : **ember-types.h**
- EMBER_INCOMING_MULTICAST_LOOPBACK : **ember-types.h**
- EMBER_INCOMING_UNICAST : **ember-types.h**
- EMBER_INCOMING_UNICAST_REPLY : **ember-types.h**
- EMBER_INCOMPATIBLE_STATIC_MEMORY_DEFINITIONS : **error-def.h**
- EMBER_INDEX_OUT_OF_RANGE : **error-def.h**
- EMBER_INDIRECT_TRANSMISSION_TIMEOUT : **ember-configuration-defaults.h**
- EMBER_INPUT_CLUSTER_LIST : **ember-types.h**
- EMBER_INSUFFICIENT_RANDOM_DATA : **error-def.h**
- EMBER_INVALID_BINDING_INDEX : **error-def.h**
- EMBER_INVALID_CALL : **error-def.h**
- EMBER_INVALID_ENDPOINT : **error-def.h**
- EMBER_INVALID_SECURITY_LEVEL : **error-def.h**
- EMBER_JOIN_DECISION_STRINGS : **ember-types.h**
- EMBER_JOIN_FAILED : **error-def.h**
- EMBER_JOINED_NETWORK : **ember-types.h**
- EMBER_JOINED_NETWORK_NO_PARENT : **ember-types.h**
- EMBER_JOINING_NETWORK : **ember-types.h**
- EMBER_KEY_ESTABLISHMENT_TIMEOUT : **ember-types.h**
- EMBER_KEY_HAS_INCOMING_FRAME_COUNTER : **ember-types.h**
- EMBER_KEY_HAS_OUTGOING_FRAME_COUNTER : **ember-types.h**
- EMBER_KEY_HAS_PARTNER_EUI64 : **ember-types.h**
- EMBER_KEY_HAS_SEQUENCE_NUMBER : **ember-types.h**
- EMBER_KEY_INVALID : **error-def.h**

- EMBER_KEY_IS_AUTHORIZED : **ember-types.h**
- EMBER_KEY_NOT_AUTHORIZED : **error-def.h**
- EMBER_KEY_PARTNER_IS_SLEEPY : **ember-types.h**
- EMBER_KEY_TABLE_FULL : **ember-types.h**
- EMBER_KEY_TABLE_INVALID_ADDRESS : **error-def.h**
- EMBER_KEY_TABLE_SIZE : **ember-configuration-defaults.h**
- EMBER_KEY_TABLE_TOKEN_SIZE : **ember-configuration-defaults.h**
- EMBER_LEAVING_NETWORK : **ember-types.h**
- EMBER_LIBRARY_NOT_PRESENT : **error-def.h**
- EMBER_LOW_RAM_CONCENTRATOR : **ember-types.h**
- EMBER_MAC_ACK_HEADER_TYPE : **error-def.h**
- EMBER_MAC_BAD_SCAN_DURATION : **error-def.h**
- EMBER_MAC_COMMAND_TRANSMIT_FAILURE : **error-def.h**
- EMBER_MAC_FILTER_MATCH_DISABLED : **ember-types.h**
- EMBER_MAC_FILTER_MATCH_ENABLED : **ember-types.h**
- EMBER_MAC_FILTER_MATCH_ENABLED_MASK : **ember-types.h**
- EMBER_MAC_FILTER_MATCH_END : **ember-types.h**
- EMBER_MAC_FILTER_MATCH_ON_DEST_BROADCAST_SHORT : **ember-types.h**
- EMBER_MAC_FILTER_MATCH_ON_DEST_MASK : **ember-types.h**
- EMBER_MAC_FILTER_MATCH_ON_DEST_UNICAST_LONG : **ember-types.h**
- EMBER_MAC_FILTER_MATCH_ON_DEST_UNICAST_SHORT : **ember-types.h**
- EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_BROADCAST : **ember-types.h**
- EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_LOCAL : **ember-types.h**
- EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_MASK : **ember-types.h**
- EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_NONE : **ember-types.h**
- EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_LOCAL : **ember-types.h**
- EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_MASK : **ember-types.h**
- EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_NON_LOCAL : **ember-types.h**
- EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_NONE : **ember-types.h**
- EMBER_MAC_FILTER_MATCH_ON_SOURCE_LONG : **ember-types.h**
- EMBER_MAC_FILTER_MATCH_ON_SOURCE_MASK : **ember-types.h**
- EMBER_MAC_FILTER_MATCH_ON_SOURCE_SHORT : **ember-types.h**
- EMBER_MAC_INCORRECT_SCAN_TYPE : **error-def.h**
- EMBER_MAC_INDIRECT_TIMEOUT : **error-def.h**
- EMBER_MAC_INVALID_CHANNEL_MASK : **error-def.h**
- EMBER_MAC_JOINED_NETWORK : **error-def.h**
- EMBER_MAC_NO_ACK_RECEIVED : **error-def.h**
- EMBER_MAC_NO_DATA : **error-def.h**
- EMBER_MAC_PASSTHROUGH_APPLICATION : **ember-types.h**
- EMBER_MAC_PASSTHROUGH_CUSTOM : **ember-types.h**
- EMBER_MAC_PASSTHROUGH_EMBERNET : **ember-types.h**
- EMBER_MAC_PASSTHROUGH_EMBERNET_SOURCE : **ember-types.h**
- EMBER_MAC_PASSTHROUGH_NONE : **ember-types.h**
- EMBER_MAC_PASSTHROUGH_SE_INTERPAN : **ember-types.h**
- EMBER_MAC_SCANNING : **error-def.h**
- EMBER_MAC_TRANSMIT_QUEUE_FULL : **error-def.h**
- EMBER_MAC_UNKNOWN_HEADER_TYPE : **error-def.h**
- EMBER_MANY_TO_ONE_BINDING : **ember-types.h**
- EMBER_MANY_TO_ONE_ROUTE_FAILURE : **error-def.h**
- EMBER_MAX_802_15_4_CHANNEL_NUMBER : **ember-types.h**
- EMBER_MAX_COMMAND_ARGUMENTS : **command-interpreter2.h**
- EMBER_MAX_DEPTH : **ember-configuration-defaults.h**
- EMBER_MAX_END_DEVICE_CHILDREN : **ember-configuration-defaults.h**
- EMBER_MAX_HOPS : **ember-configuration-defaults.h**
- EMBER_MAX_INDIRECT_TRANSMISSION_TIMEOUT : **ember-configuration-defaults.h**
- EMBER_MAX_MESSAGE_LIMIT_REACHED : **error-def.h**
- EMBER_MAX_NEIGHBOR_TABLE_SIZE : **ember-configuration-defaults.h**
- EMBER_MAXIMUM_ALARM_DATA_SIZE : **ember-configuration-defaults.h**
- EMBER_MESSAGE_TOO_LONG : **error-def.h**
- EMBER_MIN_802_15_4_CHANNEL_NUMBER : **ember-types.h**
- EMBER_MOBILE_END_DEVICE : **ember-types.h**
- EMBER_MOBILE_NODE_POLL_TIMEOUT : **ember-configuration-defaults.h**
- EMBER_MOVE_FAILED : **error-def.h**
- EMBER_MULTICAST_BINDING : **ember-types.h**
- EMBER_MULTICAST_NODE_ID : **ember-types.h**
- EMBER_MULTICAST_TABLE_SIZE : **ember-configuration-defaults.h**
- EMBER_NEIGHBOR_TABLE_SIZE : **ember-configuration-defaults.h**
- EMBER_NETWORK_BUSY : **error-def.h**
- EMBER_NETWORK_DOWN : **error-def.h**

- EMBER_NETWORK_UP : **error-def.h**
- EMBER_NEXT_NETWORK_KEY : **ember-types.h**
- EMBER_NO_ACTION : **ember-types.h**
- EMBER_NO_BEACONS : **error-def.h**
- EMBER_NO_BUFFERS : **error-def.h**
- EMBER_NO_FRAME_COUNTER_RESET : **ember-types.h**
- EMBER_NO_LINK_KEY_RECEIVED : **error-def.h**
- EMBER_NO_NETWORK : **ember-types.h**
- EMBER_NO_NETWORK_KEY_RECEIVED : **error-def.h**
- EMBER_NO_TRUST_CENTER_MODE : **ember-types.h**
- EMBER_NODE_ID_CHANGED : **error-def.h**
- EMBER_NOT_JOINED : **error-def.h**
- EMBER_NULL_ADDRESS_TABLE_INDEX : **ember-types.h**
- EMBER_NULL_BINDING : **ember-types.h**
- EMBER_NULL_NODE_ID : **ember-types.h**
- EMBER_NUM_802_15_4_CHANNELS : **ember-types.h**
- EMBER_NWK_ALREADY_PRESENT : **ember-types.h**
- EMBER_NWK_TABLE_FULL : **ember-types.h**
- EMBER_NWK_UNKNOWN_DEVICE : **ember-types.h**
- EMBER_OPERATION_IN_PROGRESS : **error-def.h**
- EMBER_OTA_CERTIFICATE_UPGRADE_CLUSTER : **ember-types.h**
- EMBER_OUTGOING_BROADCAST : **ember-types.h**
- EMBER_OUTGOING_DIRECT : **ember-types.h**
- EMBER_OUTGOING_MULTICAST : **ember-types.h**
- EMBER_OUTGOING_VIA_ADDRESS_TABLE : **ember-types.h**
- EMBER_OUTGOING_VIA_BINDING : **ember-types.h**
- EMBER_OUTPUT_CLUSTER_LIST : **ember-types.h**
- EMBER_PACKET_BUFFER_COUNT : **ember-configuration-defaults.h**
- EMBER_PAN_ID_CHANGED : **error-def.h**
- EMBER_PAN_ID_CONFLICT_REPORT_THRESHOLD : **ember-configuration-defaults.h**
- EMBER_PHY_ACK_RECEIVED : **error-def.h**
- EMBER_PHY_INVALID_CHANNEL : **error-def.h**
- EMBER_PHY_INVALID_POWER : **error-def.h**
- EMBER_PHY_OSCILLATOR_CHECK_FAILED : **error-def.h**
- EMBER_PHY_TX_BUSY : **error-def.h**
- EMBER_PHY_TX_CCA_FAIL : **error-def.h**
- EMBER_PHY_TX_INCOMPLETE : **error-def.h**
- EMBER_PHY_TX_UNDERFLOW : **error-def.h**
- EMBER_PRECONFIGURED_KEY_REQUIRED : **error-def.h**
- EMBER_PRECONFIGURED_NETWORK_KEY_MODE : **ember-types.h**
- EMBER_PRIVATE_KEY_SIZE : **ember-types.h**
- EMBER_PRIVATE_PROFILE_ID : **ember-types.h**
- EMBER_PUBLIC_KEY_SIZE : **ember-types.h**
- EMBER_RECEIVED_KEY_IN_THE_CLEAR : **error-def.h**
- EMBER_REPORT_AND_CLEAR_COUNTERS_REQUEST : **ember-types.h**
- EMBER_REPORT_AND_CLEAR_COUNTERS_RESPONSE : **ember-types.h**
- EMBER_REPORT_COUNTERS_REQUEST : **ember-types.h**
- EMBER_REPORT_COUNTERS_RESPONSE : **ember-types.h**
- EMBER_REQUEST_KEY_TIMEOUT : **ember-configuration-defaults.h**
- EMBER_REQUIRE_ENCRYPTED_KEY : **ember-types.h**
- EMBER_RESERVED_MOBILE_CHILD_ENTRIES : **ember-configuration-defaults.h**
- EMBER_ROUTE_TABLE_SIZE : **ember-configuration-defaults.h**
- EMBER_ROUTER : **ember-types.h**
- EMBER_RX_ON_WHEN_IDLE_BROADCAST_ADDRESS : **ember-types.h**
- EMBER_SECURITY_CONFIGURATION_INVALID : **error-def.h**
- EMBER_SECURITY_LEVEL : **ember-configuration-defaults.h**
- EMBER_SECURITY_STATE_NOT_SET : **error-def.h**
- EMBER_SEND_KEY_IN_THE_CLEAR : **ember-types.h**
- EMBER_SERIAL_INVALID_BAUD_RATE : **error-def.h**
- EMBER_SERIAL_INVALID_PORT : **error-def.h**
- EMBER_SERIAL_RX_EMPTY : **error-def.h**
- EMBER_SERIAL_RX_FRAME_ERROR : **error-def.h**
- EMBER_SERIAL_RX_OVERFLOW : **error-def.h**
- EMBER_SERIAL_RX_OVERRUN_ERROR : **error-def.h**
- EMBER_SERIAL_RX_PARITY_ERROR : **error-def.h**
- EMBER_SERIAL_TX_OVERFLOW : **error-def.h**
- EMBER_SIGNATURE_SIZE : **ember-types.h**
- EMBER_SIGNATURE_VERIFY_FAILURE : **error-def.h**
- EMBER_SIM_EEPROM_ERASE_PAGE_GREEN : **error-def.h**

- EMBER_SIM_EEPROM_ERASE_PAGE_RED : **error-def.h**
- EMBER_SIM_EEPROM_FULL : **error-def.h**
- EMBER_SIM_EEPROM_INIT_1_FAILED : **error-def.h**
- EMBER_SIM_EEPROM_INIT_2_FAILED : **error-def.h**
- EMBER_SIM_EEPROM_INIT_3_FAILED : **error-def.h**
- EMBER_SIM_EEPROM_REPAIRING : **error-def.h**
- EMBER_SLEEP_INTERRUPTED : **error-def.h**
- EMBER_SLEEPY_BROADCAST_ADDRESS : **ember-types.h**
- EMBER_SLEEPY_END_DEVICE : **ember-types.h**
- EMBER_SMAC_SIZE : **ember-types.h**
- EMBER_SOURCE_ROUTE_FAILURE : **error-def.h**
- EMBER_SOURCE_ROUTE_TABLE_SIZE : **ember-configuration-defaults.h**
- EMBER_STACK_AND_HARDWARE_MISMATCH : **error-def.h**
- EMBER_STACK_PROFILE : **ember-configuration-defaults.h**
- EMBER_STANDARD_SECURITY_MODE : **ember-types.h**
- EMBER_STANDARD_SECURITY_MODE_ : **ember-types.h**
- EMBER_STANDARD_SECURITY_SECURED_REJOIN : **ember-types.h**
- EMBER_STANDARD_SECURITY_UNSECURED_JOIN : **ember-types.h**
- EMBER_STANDARD_SECURITY_UNSECURED_REJOIN : **ember-types.h**
- EMBER_SUCCESS : **error-def.h**
- EMBER_TABLE_ENTRY_ERASED : **error-def.h**
- EMBER_TABLE_ENTRY_UNUSED_NODE_ID : **ember-types.h**
- EMBER_TABLE_FULL : **error-def.h**
- EMBER_TASK_COUNT : **ember-configuration-defaults.h**
- EMBER_TC_APP_KEY_SENT_TO_REQUESTER : **ember-types.h**
- EMBER_TC_FAILED_TO_SEND_APP_KEYS : **ember-types.h**
- EMBER_TC_FAILED_TO_STORE_APP_KEY_REQUEST : **ember-types.h**
- EMBER_TC_NO_LINK_KEY_FOR_REQUESTER : **ember-types.h**
- EMBER_TC_NON_MATCHING_APP_KEY_REQUEST_RECEIVED : **ember-types.h**
- EMBER_TC_RECEIVED_FIRST_APP_KEY_REQUEST : **ember-types.h**
- EMBER_TC_REJECTED_APP_KEY_REQUEST : **ember-types.h**
- EMBER_TC_REQUEST_KEY_TYPE_NOT_SUPPORTED : **ember-types.h**
- EMBER_TC_REQUESTER_EUI64_UNKNOWN : **ember-types.h**
- EMBER_TC_RESPONDED_TO_KEY_REQUEST : **ember-types.h**
- EMBER_TC_RESPONSE_TO_KEY_REQUEST_FAILED : **ember-types.h**
- EMBER_TC_TIMEOUT_WAITING_FOR_SECOND_APP_KEY_REQUEST : **ember-types.h**
- EMBER_TOO_SOON_FOR_SWITCH_KEY : **error-def.h**
- EMBER_TRUST_CENTER_EUI_HAS_CHANGED : **error-def.h**
- EMBER_TRUST_CENTER_LINK_KEY : **ember-types.h**
- EMBER_TRUST_CENTER_LINK_KEY_ESTABLISHED : **ember-types.h**
- EMBER_TRUST_CENTER_MASTER_KEY : **ember-types.h**
- EMBER_TRUST_CENTER_MASTER_KEY_NOT_SET : **error-def.h**
- EMBER_TRUST_CENTER_NODE_ID : **ember-types.h**
- EMBER_TRUST_CENTER_USES_HASHED_LINK_KEY : **ember-types.h**
- EMBER_TRUST_CENTER_USES_HASHED_LINK_KEY_ : **ember-types.h**
- EMBER_TX_POWER_MODE_ALTERNATE : **ember-types.h**
- EMBER_TX_POWER_MODE_BOOST : **ember-types.h**
- EMBER_TX_POWER_MODE_BOOST_AND_ALTERNATE : **ember-types.h**
- EMBER_TX_POWER_MODE_DEFAULT : **ember-types.h**
- EMBER_UNICAST_ALARM_CLUSTER : **ember-types.h**
- EMBER_UNICAST_ALARM_DATA_SIZE : **ember-configuration-defaults.h**
- EMBER_UNICAST_BINDING : **ember-types.h**
- EMBER_UNKNOWN_DEVICE : **ember-types.h**
- EMBER_UNKNOWN_NODE_ID : **ember-types.h**
- EMBER_UNUSED_BINDING : **ember-types.h**
- EMBER_USE_MAC_ASSOCIATION : **ember-types.h**
- EMBER_USE_NWK_COMMISSIONING : **ember-types.h**
- EMBER_USE_NWK_REJOIN : **ember-types.h**
- EMBER_USE_NWK_REJOIN_HAVE_NWK_KEY : **ember-types.h**
- EMBER_USE_PRECONFIGURED_KEY : **ember-types.h**
- EMBER_ZDO_ENDPOINT : **ember-types.h**
- EMBER_ZDO_PROFILE_ID : **ember-types.h**
- EMBER_ZDP_DEVICE_NOT_FOUND : **ember-types.h**
- EMBER_ZDP_INSUFFICIENT_SPACE : **ember-types.h**
- EMBER_ZDP_INVALID_ENDPOINT : **ember-types.h**
- EMBER_ZDP_INVALID_REQUEST_TYPE : **ember-types.h**
- EMBER_ZDP_NETWORK_MANAGER : **ember-types.h**
- EMBER_ZDP_NO_DESCRIPTOR : **ember-types.h**
- EMBER_ZDP_NO_ENTRY : **ember-types.h**

- EMBER_ZDP_NO_MATCH : **ember-types.h**
- EMBER_ZDP_NOT_ACTIVE : **ember-types.h**
- EMBER_ZDP_NOT_AUTHORIZED : **ember-types.h**
- EMBER_ZDP_NOT_PERMITTED : **ember-types.h**
- EMBER_ZDP_NOT_SUPPORTED : **ember-types.h**
- EMBER_ZDP_PRIMARY_BINDING_TABLE_CACHE : **ember-types.h**
- EMBER_ZDP_PRIMARY_DISCOVERY_CACHE : **ember-types.h**
- EMBER_ZDP_PRIMARY_TRUST_CENTER : **ember-types.h**
- EMBER_ZDP_SECONDARY_BINDING_TABLE_CACHE : **ember-types.h**
- EMBER_ZDP_SECONDARY_DISCOVERY_CACHE : **ember-types.h**
- EMBER_ZDP_SECONDARY_TRUST_CENTER : **ember-types.h**
- EMBER_ZDP_SUCCESS : **ember-types.h**
- EMBER_ZDP_TABLE_FULL : **ember-types.h**
- EMBER_ZDP_TIMEOUT : **ember-types.h**
- EMBER_ZIGBEE_COORDINATOR_ADDRESS : **ember-types.h**
- EMBER_ZIGBEE_LEAVE_AND_REJOIN : **ember-types.h**
- EMBER_ZIGBEE_LEAVE_AND_REMOVE_CHILDREN : **ember-types.h**
- emberActiveEndpointsRequest() : **zigbee-device-common.h**
- EmberApsOption : **ember-types.h**
- emberBindingTableRequest() : **zigbee-device-common.h**
- EmberBindingType : **ember-types.h**
- emberBindRequest() : **zigbee-device-common.h**
- emberCalculateSmacs() : **cbke-crypto-engine.h**
- emberCalculateSmacsHandler() : **cbke-crypto-engine.h**
- emberCertificateContents() : **ember-types.h**
- emberClearTemporaryDataMaybeStoreLinkKey() : **cbke-crypto-engine.h**
- EmberClusterListId : **ember-types.h**
- emberCommandErrorHandler() : **command-interpreter2.h**
- emberCommandInterpreter2Configuration : **command-interpreter2.h**
- emberCommandInterpreterEchoOff : **command-interpreter2.h**
- emberCommandInterpreterEchoOn : **command-interpreter2.h**
- emberCommandInterpreterIsEchoOn : **command-interpreter2.h**
- emberCommandReaderInit() : **command-interpreter2.h**
- EmberCommandStatus : **command-interpreter2.h**
- emberCommandTable : **command-interpreter2.h**
- emberCopyEui64Argument : **command-interpreter2.h**
- emberCopyKeyArgument : **command-interpreter2.h**
- emberCopyStringArgument() : **command-interpreter2.h**
- EmberCounterType : **ember-types.h**
- emberCurrentCommand : **command-interpreter2.h**
- EmberCurrentSecurityBitmask : **ember-types.h**
- EmberDeviceUpdate : **ember-types.h**
- emberDsaSign() : **cbke-crypto-engine.h**
- emberDsaSignHandler() : **cbke-crypto-engine.h**
- emberDsaVerify() : **cbke-crypto-engine.h**
- emberDsaVerifyHandler() : **cbke-crypto-engine.h**
- emberEnableDualChannelScan : **form-and-join.h**
- EmberEUI64 : **ember-types.h**
- EmberEventData : **ember-types.h**
- EmberEventUnits : **ember-types.h**
- emberFormAndJoinEnergyScanResultHandler() : **form-and-join.h**
- emberFormAndJoinIsScanning() : **form-and-join.h**
- emberFormAndJoinNetworkFoundHandler() : **form-and-join.h**
- emberFormAndJoinRunTask() : **form-and-join.h**
- emberFormAndJoinScanCompleteHandler() : **form-and-join.h**
- emberFormAndJoinTaskInit() : **form-and-join.h**
- emberFormAndJoinTick() : **form-and-join.h**
- emberGenerateCbkeKeys() : **cbke-crypto-engine.h**
- emberGenerateCbkeKeysHandler() : **cbke-crypto-engine.h**
- emberGetCertificate() : **cbke-crypto-engine.h**
- emberGetLastZigDevRequestSequence() : **zigbee-device-common.h**
- emberGetStackCertificateEui64() : **cbke-crypto-engine.h**
- emberGetZigDevRequestRadius() : **zigbee-device-common.h**
- emberIeeeAddressRequest() : **zigbee-device-host.h**
- EmberIncomingMessageType : **ember-types.h**
- emberInitializeNetworkParameters : **ember-types.h**
- EmberInitialSecurityBitmask : **ember-types.h**
- emberJoinableNetworkFoundHandler() : **form-and-join.h**
- EmberJoinDecision : **ember-types.h**

- EmberJoinMethod : **ember-types.h**
- emberKeyContents() : **ember-types.h**
- EmberKeyStatus : **ember-types.h**
- EmberKeyStructBitmask : **ember-types.h**
- EmberKeyType : **ember-types.h**
- emberLeaveRequest() : **zigbee-device-common.h**
- EmberLeaveRequestFlags : **ember-types.h**
- EmberLibraryStatus : **ember-types.h**
- EmberLinkKeyRequestPolicy : **ember-types.h**
- emberLqiTableRequest() : **zigbee-device-common.h**
- EmberMacFilterMatchData : **ember-types.h**
- EmberMacPassthroughType : **ember-types.h**
- EmberMessageBuffer : **ember-types.h**
- EmberMulticastId : **ember-types.h**
- emberNetworkAddressRequest() : **zigbee-device-host.h**
- EmberNetworkScanType : **ember-types.h**
- EmberNetworkStatus : **ember-types.h**
- emberNodeDescriptorRequest() : **zigbee-device-common.h**
- EmberNodeId : **ember-types.h**
- EmberNodeType : **ember-types.h**
- EmberOutgoingMessageType : **ember-types.h**
- EmberPanId : **ember-types.h**
- emberPermitJoiningRequest() : **zigbee-device-common.h**
- emberPowerDescriptorRequest() : **zigbee-device-common.h**
- emberPrintCommandTable() : **command-interpreter2.h**
- emberPrintCommandUsage() : **command-interpreter2.h**
- emberPrintCommandUsageNotes() : **command-interpreter2.h**
- emberPrivateKeyContents() : **ember-types.h**
- emberProcessCommandInput : **command-interpreter2.h**
- emberProcessCommandString() : **command-interpreter2.h**
- emberPublicKeyContents() : **ember-types.h**
- emberRoutingTableRequest() : **zigbee-device-common.h**
- emberScanErrorHandler() : **form-and-join.h**
- emberScanForJoinableNetwork() : **form-and-join.h**
- emberScanForNextJoinableNetwork() : **form-and-join.h**
- emberScanForUnusedPanId() : **form-and-join.h**
- emberSerialBufferTick() : **app/util/serial/serial.h**
- emberSerialFlushRx() : **app/util/serial/serial.h**
- emberSerialGuaranteedPrintf() : **app/util/serial/serial.h**
- emberSerialInit() : **app/util/serial/serial.h**
- emberSerialPrintCarriageReturn() : **app/util/serial/serial.h**
- emberSerialPrintf() : **app/util/serial/serial.h**
- emberSerialPrintfLine() : **app/util/serial/serial.h**
- emberSerialPrintfVarArg() : **app/util/serial/serial.h**
- emberSerialReadAvailable() : **app/util/serial/serial.h**
- emberSerialReadByte() : **app/util/serial/serial.h**
- emberSerialReadLine() : **app/util/serial/serial.h**
- emberSerialReadPartialLine() : **app/util/serial/serial.h**
- emberSerialWaitSend() : **app/util/serial/serial.h**
- emberSerialWriteAvailable() : **app/util/serial/serial.h**
- emberSerialWriteByte() : **app/util/serial/serial.h**
- emberSerialWriteData() : **app/util/serial/serial.h**
- emberSerialWriteHex() : **app/util/serial/serial.h**
- emberSerialWriteString() : **app/util/serial/serial.h**
- emberSerialWriteUsed : **app/util/serial/serial.h**
- emberSetPreinstalledCbkeData() : **cbke-crypto-engine.h**
- emberSetZigDevRequestRadius() : **zigbee-device-common.h**
- emberSignatureContents() : **ember-types.h**
- emberSignedCommandArgument() : **command-interpreter2.h**
- emberSimpleDescriptorRequest() : **zigbee-device-common.h**
- emberSmacContents() : **ember-types.h**
- EmberStatus : **error.h** , **ember-types.h**
- emberStringCommandArgument() : **command-interpreter2.h**
- EmberTaskId : **ember-types.h**
- emberUnbindRequest() : **zigbee-device-common.h**
- emberUnsignedCommandArgument() : **command-interpreter2.h**
- emberUnusedPanIdFoundHandler() : **form-and-join.h**
- EmberZdoConfigurationFlags : **ember-types.h**
- EmberZdoServerMask : **ember-types.h**

- EmberZdoStatus : **ember-types.h**
- emPrintfFlushHandler : **app/util/serial/serial.h**
- emPrintfInternal() : **app/util/serial/serial.h**
- END_DEVICE_ANNOUNCE : **ember-types.h**
- END_DEVICE_ANNOUNCE_RESPONSE : **ember-types.h**
- END_DEVICE_BIND_REQUEST : **ember-types.h**
- END_DEVICE_BIND_RESPONSE : **ember-types.h**
- EUI64_SIZE : **ember-types.h**
- EXTENDED_PAN_ID_SIZE : **ember-types.h**
- EZSP_HOST_ASH_RX_POOL_SIZE : **ezsp-host-configuration-defaults.h**
- EZSP_HOST_FORM_AND_JOIN_BUFFER_SIZE : **ezsp-host-configuration-defaults.h**
- EZSP_HOST_SOURCE_ROUTE_TABLE_SIZE : **ezsp-host-configuration-defaults.h**
- ezspDecodeAddressResponse() : **zigbee-device-host.h**
- ezspEndDeviceBindRequest() : **zigbee-device-host.h**
- ezspFragmentIncomingMessage() : **fragment-host.h**
- ezspFragmentInit() : **fragment-host.h**
- ezspFragmentMessageSent() : **fragment-host.h**
- ezspFragmentMessageSentHandler() : **fragment-host.h**
- ezspFragmentSendUnicast() : **fragment-host.h**
- ezspFragmentSourceRouteHandler() : **fragment-host.h**
- ezspFragmentTick() : **fragment-host.h**
- ezspMatchDescriptorsRequest() : **zigbee-device-host.h**

- f -

- h -

- halPowerDown() : **micro-common.h**
- halPowerUp() : **micro-common.h**
- halReboot() : **micro-common.h**
- halResetWatchdog : **iar-st.h**
- halSampleAdc() : **adc.h**
- halSetLed() : **led-common.h**
- halSleep() : **micro-common.h**
- halStackSeedRandom() : **micro-common.h**
- halStartBuzzerTone() : **buzzer.h**
- halStopBuzzerTone() : **buzzer.h**
- halToggleLed() : **led-common.h**
- HANDLE_PENDING_INTERRUPTS : **iar-st.h**
- handler : **ember-types.h**
- hereIamTune : **buzzer.h**
- HIGH_BYTE : **platform-common.h**
- HIGH_LOW_TO_INT : **platform-common.h**
- hostBootloadReinitHandler() : **bootload-ezsp-utils.h**
- hostBootloadUtilLaunchRequestHandler() : **bootload-ezsp-utils.h**
- hostBootloadUtilQueryResponseHandler() : **bootload-ezsp-utils.h**

- i -

- j -

- | -

- m -

- n -

- p -

- r -

- s -

- t -

- u -

- w -

Index - _ - **a** - **b** - **c** - **d** - **e** - **f** - **h** - **i** - **j** - **l** - **m** - **n** - **p** - **r** - **s** - **t** - **u** - **w** - **z** -

- z -

- ZDO_MESSAGE_OVERHEAD : **zigbee-device-common.h**

