SILICON LABS

EM35x

EmberZNet 4.7.2 API Reference:

For the EM35x SoC Platform

December 18, 2012
120-3022-000-4720

Disclaimer

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

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

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

About This Guide

Purpose

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

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

- Everything is contained in this single document.

- This document is fully searchable using the Adobe Acrobat search engine that is part of the free Acrobat Reader (available from www.adobe.com).

- This document can be easily printed.

Audience

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

Getting Help

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

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

Chapter 1

Introduction

The EmberZNet API Reference documentation for the EM35x includes the following API sets:

- EmberZNet Stack API Reference
- Hardware Abstraction Layer (HAL) API Reference
- Application Utilities API Reference

Chapter 2

Deprecated List

File ami-inter-pan-host.h

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

File ami-inter-pan.h

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

File fragment-host.h

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

File fragment.h

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

Chapter 3

Module Index

3.1   Modules

Here is a list of all modules:

Chapter 4

Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Module Documentation

6.1 EmberZNet Stack API Reference

Modules

- Stack Information
- Ember Common Data Types
- Network Formation
- Packet Buffers
- Sending and Receiving Messages
- End Devices
- Security
- Binding Table
- Configuration
- Status Codes
- Stack Tokens
- ZigBee Device Object
- Bootloader
- Event Scheduling
- Manufacturing and Functional Test Library
- Debugging Utilities

6.1.1 Detailed Description

This documentation describes the application programming interface (API) for the EmberZNet stack. The file ember.h is the master include file for the EmberZNet API modules.

6.2   Stack Information

Data Structures

- struct EmberEndpointDescription

    Endpoint information (a ZigBee Simple Descriptor).
- struct EmberEndpoint

    Gives the endpoint information for a particular endpoint.

Macros

- #define EMBER_MAJOR_VERSION
- #define EMBER_MINOR_VERSION
- #define EMBER_PATCH_VERSION
- #define EMBER_SPECIAL_VERSION
- #define EMBER_BUILD_NUMBER
- #define EMBER_FULL_VERSION
- #define EMBER_VERSION_TYPE
- #define SOFTWARE_VERSION

Functions

- void emberStackStatusHandler (EmberStatus status)
- EmberNetworkStatus emberNetworkState (void)
- boolean emberStackIsUp (void)
- EmberEUI64 emberGetEui64 (void)
- boolean emberIsLocalEui64 (EmberEUI64 eui64)
- EmberNodeId emberGetNodeId (void)
- EmberNodeId emberRadioGetNodeId (void)
- void emberSetManufacturerCode (int16u code)
- void emberSetPowerDescriptor (int16u descriptor)
- void emberSetMaximumIncomingTransferSize (int16u size)
- void emberSetMaximumOutgoingTransferSize (int16u size)
- void emberSetDescriptorCapability (int8u capability)
- EmberStatus emberGetNetworkParameters (EmberNetworkParameters ∗parameters)
- EmberStatus emberGetNodeType (EmberNodeType ∗resultLocation)
- EmberStatus emberSetRadioChannel (int8u channel)
- int8u emberGetRadioChannel (void)
- EmberStatus emberSetRadioPower (int8s power)
- int8s emberGetRadioPower (void)
- EmberPanId emberGetPanId (void)
- EmberPanId emberRadioGetPanId (void)
- void emberGetExtendedPanId (int8u ∗resultLocation)
- int8u emberGetEndpoint (int8u index)
- boolean emberGetEndpointDescription (int8u endpoint, EmberEndpointDescription ∗result)
- int16u emberGetEndpointCluster (int8u endpoint, EmberClusterListId listId, int8u listIndex)
- boolean emberIsNodeIdValid (EmberNodeId nodeId)
- EmberNodeId emberLookupNodeIdByEui64 (EmberEUI64 eui64)
- EmberStatus emberLookupEui64ByNodeId (EmberNodeId nodeId, EmberEUI64 eui64Return)

- void emberCounterHandler (EmberCounterType type, int8u data)
- EmberStatus emberGetNeighbor (int8u index, EmberNeighborTableEntry ∗result)
- EmberStatus emberGetRouteTableEntry (int8u index, EmberRouteTableEntry ∗result)
- int8u emberStackProfile (void)
- int8u emberTreeDepth (void)
- int8u emberNeighborCount (void)
- int8u emberRouteTableSize (void)
- int8u emberNextZigbeeSequenceNumber (void)

Variables

- PGM int8u emberStackProfileId [ ]
- int8u emberEndpointCount
- EmberEndpoint emberEndpoints [ ]

Radio-specific Functions

- EmberStatus emberSetTxPowerMode (int16u txPowerMode)
- int16u emberGetTxPowerMode (void)
- EmberStatus emberSetNodeId (EmberNodeId nodeId)
- void emberRadioNeedsCalibratingHandler (void)
- void emberCalibrateCurrentChannel (void)

General Functions

- void emberReverseMemCopy (int8u ∗dest, const int8u ∗src, int8u length)
- XAP2B_PAGEZERO_ON int16u emberFetchLowHighInt16u (int8u ∗contents)
- XAP2B_PAGEZERO_OFF void emberStoreLowHighInt16u (int8u ∗contents, int16u value)
- int32u emberFetchLowHighInt32u (int8u ∗contents)
- int32u emberFetchHighLowInt32u (int8u ∗contents)
- void emberStoreLowHighInt32u (int8u ∗contents, int32u value)
- void emberStoreHighLowInt32u (int8u ∗contents, int32u value)

6.2.1 Detailed Description

See stack-info.h for source code.

See also config.h.

This documentation was produced from the following software release and build.

SOFTWARE_VERSION	0x4700	High byte = release number, low byte = patch number

6.2.2 Macro Definition Documentation

6.2.2.1 #define EMBER_MAJOR_VERSION

Definition at line 19 of file config.h.

6.2.2.2 #define EMBER_MINOR_VERSION

Definition at line 20 of file config.h.

6.2.2.3 #define EMBER_PATCH_VERSION

Definition at line 21 of file config.h.

6.2.2.4 #define EMBER_SPECIAL_VERSION

Definition at line 22 of file config.h.

6.2.2.5 #define EMBER_BUILD_NUMBER

Definition at line 25 of file config.h.

6.2.2.6 #define EMBER_FULL_VERSION

Definition at line 27 of file config.h.

6.2.2.7 #define EMBER_VERSION_TYPE

Definition at line 32 of file config.h.

6.2.2.8 #define SOFTWARE_VERSION

Software version. High byte First nibble: Major Version Second nibble: Minor Version Low Byte First nibble: Patch Version Second nibble: Special Version

Definition at line 43 of file config.h.

6.2.3 Function Documentation

6.2.3.1 void emberStackStatusHandler (EmberStatus *status*)

A callback invoked when the status of the stack changes. If the status parameter equals EMBER_NE-TWORK_UP, then the emberGetNetworkParameters() function can be called to obtain the new network parameters. If any of the parameters are being stored in nonvolatile memory by the application, the stored values should be updated.

The application is free to begin messaging once it receives the EMBER_NETWORK_UP status. However, routes discovered immediately after the stack comes up may be suboptimal. This is because the routes are based on the neighbor table's information about two-way links with neighboring nodes, which is obtained from periodic ZigBee Link Status messages. It can take two or three link status exchange periods (of 16 seconds each) before the neighbor table has a good estimate of link quality to neighboring nodes. Therefore, the application may improve the quality of initially discovered routes by waiting after startup to give the neighbor table time to be populated.

Parameters

status	Stack status. One of the following: • EMBER_NETWORK_UP • EMBER_NETWORK_DOWN • EMBER_JOIN_FAILED • EMBER_MOVE_FAILED • EMBER_CANNOT_JOIN_AS_ROUTER • EMBER_NODE_ID_CHANGED • EMBER_PAN_ID_CHANGED • EMBER_CHANNEL_CHANGED • EMBER_NO_BEACONS • EMBER_RECEIVED_KEY_IN_THE_CLEAR • EMBER_NO_NETWORK_KEY_RECEIVED • EMBER_NO_LINK_KEY_RECEIVED • EMBER_PRECONFIGURED_KEY_REQUIRED

6.2.3.2 EmberNetworkStatus emberNetworkState (void)

Returns the current join status.

Returns a value indicating whether the node is joining, joined to, or leaving a network.

Returns

An EmberNetworkStatus value indicating the current join status.

6.2.3.3 boolean emberStackIsUp (void)

Indicates whether the stack is currently up.

Returns true if the stack is joined to a network and ready to send and receive messages. This reflects only the state of the local node; it does not indicate whether or not other nodes are able to communicate with this node.

Returns

True if the stack is up, false otherwise.

6.2.3.4 EmberEUI64 emberGetEui64 (void)

Returns the EUI64 ID of the local node.

Returns

The 64-bit ID.

6.2.3.5 boolean emberIsLocalEui64 (EmberEUI64 *eui64*)

Determines whether `eui64` is the local node's EUI64 ID.

Parameters

eui64	An EUI64 ID.

Returns

TRUE if `eui64` is the local node's ID, otherwise FALSE.

6.2.3.6 EmberNodeId emberGetNodeId (void)

Returns the 16-bit node ID of local node on the current logical network.

Returns

The 16-bit ID.

6.2.3.7 EmberNodeId emberRadioGetNodeId (void)

Returns the 16-bit node ID of local node on the network it is currently tuned on.

Returns

The 16-bit ID.

6.2.3.8 void emberSetManufacturerCode (int16u *code*)

Sets the manufacturer code to the specified value. The manufacturer code is one of the fields of the node descriptor.

Parameters

code	The manufacturer code for the local node.

6.2.3.9 void emberSetPowerDescriptor (int16u *descriptor*)

Sets the power descriptor to the specified value. The power descriptor is a dynamic value, therefore you should call this function whenever the value changes.

Parameters

descriptor	The new power descriptor for the local node.

6.2.3.10   void emberSetMaximumIncomingTransferSize (int16u *size*)

Sets the maximum incoming transfer size to the specified value. The maximum incoming transfer size is one of the fields of the node descriptor.

Parameters

size	The maximum incoming transfer size for the local node.

6.2.3.11   void emberSetMaximumOutgoingTransferSize (int16u *size*)

Sets the maximum outgoing transfer size to the specified value. The maximum outgoing transfer size is one of the fields of the node descriptor.

Parameters

size	The maximum outgoing transfer size for the local node.

6.2.3.12   void emberSetDescriptorCapability (int8u *capability*)

Sets the descriptor capability field of the node.

Parameters

capability	The descriptor capability of the local node.

6.2.3.13   EmberStatus emberGetNetworkParameters (EmberNetworkParameters ∗ *parameters*)

Copies the current network parameters into the structure provided by the caller.

Parameters

parameters	A pointer to an EmberNetworkParameters value into which the current network parameters will be copied.

Returns

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

6.2.3.14   EmberStatus emberGetNodeType (EmberNodeType ∗ *resultLocation*)

Copies the current node type into the location provided by the caller.

Parameters

resultLocation	A pointer to an EmberNodeType value into which the current node type will be copied.

Returns

> An EmberStatus value that indicates the success or failure of the command.

6.2.3.15   EmberStatus emberSetRadioChannel (int8u *channel*)

Sets the channel to use for sending and receiving messages on the current logical network. For a list of available radio channels, see the technical specification for the RF communication module in your Developer Kit.

Note: Care should be taken when using this API, as all devices on a network must use the same channel.

Parameters

channel	Desired radio channel.

Returns

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

6.2.3.16   int8u emberGetRadioChannel (void   )

Gets the radio channel to which a node is set on the current logical network. The possible return values depend on the radio in use. For a list of available radio channels, see the technical specification for the RF communication module in your Developer Kit.

Returns

> Current radio channel.

6.2.3.17   EmberStatus emberSetRadioPower (int8s *power*)

Sets the radio output power at which a node is to operate for the current logical network. Ember radios have discrete power settings. For a list of available power settings, see the technical specification for the RF communication module in your Developer Kit. Note: Care should be taken when using this API on a running network, as it will directly impact the established link qualities neighboring nodes have with the node on which it is called. This can lead to disruption of existing routes and erratic network behavior. Note: If the requested power level is not available on a given radio, this function will use the next higher available power level.

Parameters

power	Desired radio output power, in dBm.

Returns

> An EmberStatus value indicating the success or failure of the command. Failure indicates that the requested power level is out of range.

6.2.3.18 int8s emberGetRadioPower (void)

Gets the radio output power of the current logical network at which a node is operating. Ember radios have discrete power settings. For a list of available power settings, see the technical specification for the RF communication module in your Developer Kit.

Returns

Current radio output power, in dBm.

6.2.3.19 EmberPanId emberGetPanId (void)

Returns the local node's PAN ID of the current logical network.

Returns

A PAN ID.

6.2.3.20 EmberPanId emberRadioGetPanId (void)

Returns the local node's PAN ID of the current radio network.

Returns

A PAN ID.

6.2.3.21 void emberGetExtendedPanId (int8u ∗ resultLocation)

Fetches a node's 8 byte Extended PAN identifier.

6.2.3.22 int8u emberGetEndpoint (int8u index)

Retrieves the endpoint number for the index'th endpoint. index must be less than the value of ember-EndpointCount.

This function is provided by the stack, using the data from emberEndpoints, unless the application defines EMBER_APPLICATION_HAS_GET_ENDPOINT in its CONFIGURATION_HEADER.

Parameters

index	The index of an endpoint (as distinct from its endpoint number). This must be less than the value of emberEndpointCount.

Returns

The endpoint number for the index'th endpoint.

6.2.3.23 boolean emberGetEndpointDescription (int8u *endpoint,* EmberEndpointDescription ∗ *result*)

Retrieves the endpoint description for the given endpoint.

This function is provided by the stack, using the data from emberEndpoints, unless the application defines ::EMBER_APPLICATION_HAS_GET_ENDPOINT in its ::CONFIGURATION_HEADER.

Parameters

endpoint	The endpoint whose description is to be returned.
result	A pointer to the location to which to copy the endpoint description.

Returns

TRUE if the description was copied to result or FALSE if the endpoint is not active.

6.2.3.24 int16u emberGetEndpointCluster (int8u *endpoint,* EmberClusterListId *listId,* int8u *listIndex*)

Retrieves a cluster ID from one of the cluster lists associated with the given endpoint.

This function is provided by the stack, using the data from emberEndpoints, unless the application defines ::EMBER_APPLICATION_HAS_GET_ENDPOINT in its CONFIGURATION_HEADER.

Parameters

endpoint	The endpoint from which the cluster ID is to be read.
listId	The list from which the cluster ID is to be read.
listIndex	The index of the desired cluster ID in the list. This value must be less than the length of the list. The length can be found in the EmberEndpointDescription for this endpoint.

Returns

The cluster ID at position listIndex in the specified endpoint cluster list.

6.2.3.25 boolean emberIsNodeIdValid (EmberNodeId *nodeId*)

Determines whether `nodeId` is valid.

Parameters

nodeId	A node ID.

Returns

TRUE if `nodeId` is valid, FALSE otherwise.

6.2.3.26 EmberNodeId emberLookupNodeIdByEui64 (EmberEUI64 *eui64*)

Returns the node ID that corresponds to the specified EUI64. The node ID is found by searching through all stack tables for the specified EUI64.

Parameters

eui64	The EUI64 of the node to look up.

Returns

The short ID of the node or EMBER_NULL_NODE_ID if the short ID is not known.

6.2.3.27 EmberStatus emberLookupEui64ByNodeId (EmberNodeId *nodeId,* EmberEUI64 *eui64Return*)

Returns the EUI64 that corresponds to the specified node ID. The EUI64 is found by searching through all stack tables for the specified node ID.

Parameters

nodeId	The short ID of the node to look up.
eui64Return	The EUI64 of the node is copied here if it is known.

Returns

An EmberStatus value:

- EMBER_SUCCESS - eui64Return has been set to the EUI64 of the node.
- EMBER_ERR_FATAL - The EUI64 of the node is not known.

6.2.3.28 void emberCounterHandler (EmberCounterType *type,* int8u *data*)

A callback invoked to inform the application of the occurrence of an event defined by EmberCounterType, for example, transmissions and receptions at different layers of the stack.

The application must define ::EMBER_APPLICATION_HAS_COUNTER_HANDLER in its CONFIGU-RATION_HEADER to use this. This function may be called in ISR context, so processing should be kept to a minimum.

Parameters

type	The type of the event.
data	For transmission events, the number of retries used. For other events, this parameter is unused and is set to zero.

6.2.3.29 EmberStatus emberGetNeighbor (int8u *index,* EmberNeighborTableEntry ∗ *result*)

Copies a neighbor table entry to the structure that `result` points to. Neighbor table entries are stored in ascending order by node id, with all unused entries at the end of the table. The number of active neighbors

can be obtained using emberNeighborCount().

Parameters

index	The index of a neighbor table entry.
result	A pointer to the location to which to copy the neighbor table entry.

Returns

EMBER_ERR_FATAL if the index is greater or equal to the number of active neighbors, or if the device is an end device. Returns EMBER_SUCCESS otherwise.

6.2.3.30 EmberStatus emberGetRouteTableEntry (int8u *index,* EmberRouteTableEntry ∗ *result*)

Copies a route table entry to the structure that `result` points to. Unused route table entries have destination 0xFFFF. The route table size can be obtained via emberRouteTableSize().

Parameters

index	The index of a route table entry.
result	A pointer to the location to which to copy the route table entry.

Returns

EMBER_ERR_FATAL if the index is out of range or the device is an end device, and EMBER_SUC-CESS otherwise.

6.2.3.31 int8u emberStackProfile (void)

Returns the stack profile of the network which the node has joined.

Returns

stack profile

6.2.3.32 int8u emberTreeDepth (void)

Returns the depth of the node in the network.

Returns

current depth

6.2.3.33 int8u emberNeighborCount (void)

Returns the number of active entries in the neighbor table.

Returns

number of active entries in the neighbor table

6.2.3.34   int8u emberRouteTableSize (void)

Returns the size of the route table.

Returns

the size of the route table

6.2.3.35   int8u emberNextZigbeeSequenceNumber (void)

Increments and returns the ZigBee sequence number.

Returns

the next ZigBee sequence number

6.2.3.36   EmberStatus emberSetTxPowerMode (int16u *txPowerMode*)

Enables boost power mode and/or the alternate transmit path.

Boost power mode is a high performance radio mode which offers increased transmit power and receive sensitivity at the cost of an increase in power consumption. The alternate transmit output path allows for simplified connection to an external power amplifier via the RF_TX_ALT_P and RF_TX_ALT_N pins on the em250. emberInit() calls this function using the power mode and transmitter output settings as specified in the MFG_PHY_CONFIG token (with each bit inverted so that the default token value of 0xffff corresponds to normal power mode and bi-directional RF transmitter output). The application only needs to call emberSetTxPowerMode() if it wishes to use a power mode or transmitter output setting different from that specified in the MFG_PHY_CONFIG token. After this initial call to emberSetTxPowerMode(), the stack will automatically maintain the specified power mode configuration across sleep/wake cycles.

Note

This function does not alter the MFG_PHY_CONFIG token. The MFG_PHY_CONFIG token must be properly configured to ensure optimal radio performance when the standalone bootloader runs in recovery mode. The MFG_PHY_CONFIG can only be set using external tools. IF YOUR PRODUCT USES BOOST MODE OR THE ALTERNATE TRANSMITTER OUTPUT AND THE STANDAL-ONE BOOTLOADER YOU MUST SET THE PHY_CONFIG TOKEN INSTEAD OF USING THIS FUNCTION. Contact `support@ember.com` for instructions on how to set the MFG_PHY_CON-FIG token appropriately.

Parameters

txPowerMode	Specifies which of the transmit power mode options are to be activated. This parameter should be set to one of the literal values described in stack/include/ember-types.h. Any power option not specified in the txPowerMode parameter will be deactivated.

Returns

EMBER_SUCCESS if successful; an error code otherwise.

6.2.3.37 int16u emberGetTxPowerMode (void)

Returns the current configuration of boost power mode and alternate transmitter output.

Returns

the current tx power mode.

6.2.3.38 EmberStatus emberSetNodeId (EmberNodeId *nodeId*)

It allows to set the short node ID of the node. Notice that it can only be set if the stack is in the INITAL state.

Parameters

nodeId	Specifies the short ID to be assigned to the node.

Returns

EMBER_SUCCESS if successful; an error code otherwise.

6.2.3.39 void emberRadioNeedsCalibratingHandler (void)

The radio calibration callback function.

The Voltage Controlled Oscillator (VCO) can drift with temperature changes. During every call to ember-Tick(), the stack will check to see if the VCO has drifted. If the VCO has drifted, the stack will call emberRadioNeedsCalibratingHandler() to inform the application that it should perform calibration of the current channel as soon as possible. Calibration can take up to 150ms. The default callback function implementation provided here performs calibration immediately. If the application wishes, it can define its own callback by defining ::EMBER_APPLICATION_HAS_CUSTOM_RADIO_CALIBRATION_CALLBACK in its CONFIGURATION_HEADER. It can then failsafe any critical processes or peripherals before calling emberCalibrateCurrentChannel(). The application must call emberCalibrateCurrentChannel() in response to this callback to maintain expected radio performance.

6.2.3.40 void emberCalibrateCurrentChannel (void)

Calibrates the current channel. The stack will notify the application of the need for channel calibration via the emberRadioNeedsCalibratingHandler() callback function during emberTick(). This function should only be called from within the context of the emberRadioNeedsCalibratingHandler() callback function. Calibration can take up to 150ms. Note if this function is called when the radio is off, it will turn the radio on and leave it on.

6.2.3.41 void emberReverseMemCopy (int8u ∗ *dest,* const int8u ∗ *src,* int8u *length*)

This function copies an array of bytes and reverses the order before writing the data to the destination.

Parameters

dest	A pointer to the location where the data will be copied to.
src	A pointer to the location where the data will be copied from.
length	The length (in bytes) of the data to be copied.

6.2.3.42   XAP2B_PAGEZERO_ON int16u emberFetchLowHighInt16u (int8u ∗ *contents*)

Returns the value built from the two `int8u` values `contents`[0] and `contents`[1]. `contents`[0] is the low byte.

6.2.3.43   XAP2B_PAGEZERO_OFF void emberStoreLowHighInt16u (int8u ∗ *contents,* int16u *value*)

Stores `value` in `contents`[0] and `contents`[1]. `contents`[0] is the low byte.

6.2.3.44   int32u emberFetchLowHighInt32u (int8u ∗ *contents*)

Returns the value built from the four `int8u` values `contents`[0], `contents`[1], `contents`[2] and `contents`[3]. `contents`[0] is the low byte.

6.2.3.45   int32u emberFetchHighLowInt32u (int8u ∗ *contents*)

Description:

   Returns the value built from the four `int8u` values `contents`[0], `contents`[1], `contents`[2] and `contents`[3]. `contents`[3] is the low byte.

6.2.3.46   void emberStoreLowHighInt32u (int8u ∗ *contents,* int32u *value*)

Stores `value` in `contents`[0], `contents`[1], `contents`[2] and `contents`[3]. `contents`[0] is the low byte.

6.2.3.47   void emberStoreHighLowInt32u (int8u ∗ *contents,* int32u *value*)

Description:

   Stores `value` in `contents`[0], `contents`[1], `contents`[2] and `contents`[3]. `contents`[3] is the low byte.

6.2.4   Variable Documentation

6.2.4.1   PGM int8u emberStackProfileId[ ]

The application must provide a definition for this variable.

6.2.4.2   int8u emberEndpointCount

The application must provide a definition for this variable.

6.2.4.3   EmberEndpoint emberEndpoints[ ]

If emberEndpointCount is nonzero, the application must provide descriptions for each endpoint.

This can be done either by providing a definition of emberEndpoints or by providing definitions of ember-GetEndpoint(), emberGetEndpointDescription() and emberGetEndpointCluster(). Using the array is often simpler, but consumes large amounts of memory if emberEndpointCount is large.

If the application provides definitions for the three functions, it must define EMBER_APPLICATION_H-AS_GET_ENDPOINT in its CONFIGURATION_HEADER.

6.3 Ember Common Data Types

Data Structures

- struct EmberVersion

  Version struct containing all version information.

- struct EmberZigbeeNetwork

  Defines a ZigBee network and the associated parameters.

- struct EmberNetworkInitStruct

  Defines the network initialization configuration that should be used when emberNetworkInitExtended() is called by the application.

- struct EmberNetworkParameters

  Holds network parameters.

- struct EmberApsFrame

  An in-memory representation of a ZigBee APS frame of an incoming or outgoing message.

- struct EmberBindingTableEntry

  Defines an entry in the binding table.

- struct EmberNeighborTableEntry

  Defines an entry in the neighbor table.

- struct EmberRouteTableEntry

  Defines an entry in the route table.

- struct EmberMulticastTableEntry

  Defines an entry in the multicast table.

- struct EmberEventControl

  Control structure for events.

- struct EmberTaskControl

  Control structure for tasks.

- struct EmberKeyData

  This data structure contains the key data that is passed into various other functions.

- struct EmberCertificateData

  This data structure contains the certificate data that is used for Certificate Based Key Exchange (CBKE).

- struct EmberPublicKeyData

  This data structure contains the public key data that is used for Certificate Based Key Exchange (CBKE).

- struct EmberPrivateKeyData

  This data structure contains the private key data that is used for Certificate Based Key Exchange (CBKE).

- struct EmberSmacData

  This data structure contains the Shared Message Authentication Code (SMAC) data that is used for Certificate Based Key Exchange (CBKE).

- struct EmberSignatureData

  This data structure contains a DSA signature. It is the bit concatenation of the 'r' and 's' components of the signature.

- struct EmberMessageDigest

  This data structure contains an AES-MMO Hash (the message digest).

- struct EmberAesMmoHashContext

  This data structure contains the context data when calculating an AES MMO hash (message digest).

- struct EmberInitialSecurityState

  This describes the Initial Security features and requirements that will be used when forming or joining the network.

- struct EmberCurrentSecurityState

    This describes the security features used by the stack for a joined device.
- struct EmberKeyStruct

    This describes a one of several different types of keys and its associated data.
- struct EmberMfgSecurityStruct

    This structure is used to get/set the security config that is stored in manufacturing tokens.
- struct EmberMacFilterMatchStruct

    This structure indicates a matching raw MAC message has been received by the application configured MAC filters.

Macros

- #define EMBER_JOIN_DECISION_STRINGS
- #define EMBER_DEVICE_UPDATE_STRINGS
- #define emberInitializeNetworkParameters(parameters)
- #define EMBER_COUNTER_STRINGS
- #define EMBER_STANDARD_SECURITY_MODE
- #define EMBER_TRUST_CENTER_NODE_ID
- #define EMBER_NO_TRUST_CENTER_MODE
- #define EMBER_GLOBAL_LINK_KEY
- #define EMBER_MFG_SECURITY_CONFIG_MAGIC_NUMBER
- #define EMBER_MAC_FILTER_MATCH_ENABLED_MASK
- #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_MASK
- #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_MASK
- #define EMBER_MAC_FILTER_MATCH_ON_DEST_MASK
- #define EMBER_MAC_FILTER_MATCH_ON_SOURCE_MASK
- #define EMBER_MAC_FILTER_MATCH_ENABLED
- #define EMBER_MAC_FILTER_MATCH_DISABLED
- #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_NONE
- #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_LOCAL
- #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_BROADCAST
- #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_NONE
- #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_NON_LOCAL
- #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_LOCAL
- #define EMBER_MAC_FILTER_MATCH_ON_DEST_BROADCAST_SHORT
- #define EMBER_MAC_FILTER_MATCH_ON_DEST_UNICAST_SHORT
- #define EMBER_MAC_FILTER_MATCH_ON_DEST_UNICAST_LONG
- #define EMBER_MAC_FILTER_MATCH_ON_SOURCE_LONG
- #define EMBER_MAC_FILTER_MATCH_ON_SOURCE_SHORT
- #define EMBER_MAC_FILTER_MATCH_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_EN-D_DEVICE,
  EMBER_SLEEPY_END_DEVICE, EMBER_MOBILE_END_DEVICE }
- enum EmberNetworkInitBitmask { EMBER_NETWORK_INIT_NO_OPTIONS, EMBER_NETW-ORK_INIT_PARENT_INFO_IN_TOKEN }
- enum EmberApsOption {
  EMBER_APS_OPTION_NONE, EMBER_APS_OPTION_DSA_SIGN, EMBER_APS_OPTION-_ENCRYPTION, EMBER_APS_OPTION_RETRY,
  EMBER_APS_OPTION_ENABLE_ROUTE_DISCOVERY, EMBER_APS_OPTION_FORCE_R-OUTE_DISCOVERY, EMBER_APS_OPTION_SOURCE_EUI64, EMBER_APS_OPTION_DE-STINATION_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_INCOM-ING_MULTICAST, EMBER_INCOMING_MULTICAST_LOOPBACK,
  EMBER_INCOMING_BROADCAST, EMBER_INCOMING_BROADCAST_LOOPBACK }
- enum EmberOutgoingMessageType {
  EMBER_OUTGOING_DIRECT, EMBER_OUTGOING_VIA_ADDRESS_TABLE, EMBER_O-UTGOING_VIA_BINDING, EMBER_OUTGOING_MULTICAST,
  EMBER_OUTGOING_BROADCAST }
- enum EmberNetworkStatus {
  EMBER_NO_NETWORK, EMBER_JOINING_NETWORK, EMBER_JOINED_NETWORK, E-MBER_JOINED_NETWORK_NO_PARENT,
  EMBER_LEAVING_NETWORK }
- enum EmberNetworkScanType { EMBER_ENERGY_SCAN, EMBER_ACTIVE_SCAN }
- enum EmberBindingType { EMBER_UNUSED_BINDING, EMBER_UNICAST_BINDING, EM-BER_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_UNSECU-RED_REJOIN,
  EMBER_HIGH_SECURITY_SECURED_REJOIN, EMBER_HIGH_SECURITY_UNSECURED-_JOIN, EMBER_HIGH_SECURITY_UNSECURED_REJOIN, EMBER_REJOIN_REASON_NO-NE,
  EMBER_REJOIN_DUE_TO_NWK_KEY_UPDATE, EMBER_REJOIN_DUE_TO_LEAVE_ME-SSAGE, EMBER_REJOIN_DUE_TO_NO_PARENT, EMBER_REJOIN_DUE_TO_ZLL_TOUC-HLINK,
  EMBER_REJOIN_DUE_TO_APP_EVENT_5, EMBER_REJOIN_DUE_TO_APP_EVENT_4, E-MBER_REJOIN_DUE_TO_APP_EVENT_3, EMBER_REJOIN_DUE_TO_APP_EVENT_2,
  EMBER_REJOIN_DUE_TO_APP_EVENT_1 }
- enum EmberDeviceUpdate {
  EMBER_STANDARD_SECURITY_SECURED_REJOIN, EMBER_STANDARD_SECURITY_-UNSECURED_JOIN, EMBER_DEVICE_LEFT, EMBER_STANDARD_SECURITY_UNSECU-RED_REJOIN,
  EMBER_HIGH_SECURITY_SECURED_REJOIN, EMBER_HIGH_SECURITY_UNSECURED-_JOIN, EMBER_HIGH_SECURITY_UNSECURED_REJOIN, EMBER_REJOIN_REASON_NO-

NE,
EMBER_REJOIN_DUE_TO_NWK_KEY_UPDATE, EMBER_REJOIN_DUE_TO_LEAVE_ME-
SSAGE, EMBER_REJOIN_DUE_TO_NO_PARENT, EMBER_REJOIN_DUE_TO_ZLL_TOUC-
HLINK,
EMBER_REJOIN_DUE_TO_APP_EVENT_5, EMBER_REJOIN_DUE_TO_APP_EVENT_4, E-
MBER_REJOIN_DUE_TO_APP_EVENT_3, EMBER_REJOIN_DUE_TO_APP_EVENT_2,
EMBER_REJOIN_DUE_TO_APP_EVENT_1 }

- enum EmberClusterListId { EMBER_INPUT_CLUSTER_LIST, EMBER_OUTPUT_CLUSTER_-
LIST }
- enum EmberEventUnits {
EMBER_EVENT_INACTIVE, EMBER_EVENT_MS_TIME, EMBER_EVENT_QS_TIME, EM-
BER_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_BROADCAS-
T, EMBER_COUNTER_MAC_RX_UNICAST, EMBER_COUNTER_MAC_TX_UNICAST_SU-
CCESS,
EMBER_COUNTER_MAC_TX_UNICAST_RETRY, EMBER_COUNTER_MAC_TX_UNICAS-
T_FAILED, EMBER_COUNTER_APS_DATA_RX_BROADCAST, EMBER_COUNTER_APS-
_DATA_TX_BROADCAST,
EMBER_COUNTER_APS_DATA_RX_UNICAST, EMBER_COUNTER_APS_DATA_TX_UN-
ICAST_SUCCESS, EMBER_COUNTER_APS_DATA_TX_UNICAST_RETRY, EMBER_COU-
NTER_APS_DATA_TX_UNICAST_FAILED,
EMBER_COUNTER_ROUTE_DISCOVERY_INITIATED, EMBER_COUNTER_NEIGHBOR_-
ADDED, EMBER_COUNTER_NEIGHBOR_REMOVED, EMBER_COUNTER_NEIGHBOR_S-
TALE,
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_COUN-
TER_FAILURE, EMBER_COUNTER_APS_FRAME_COUNTER_FAILURE, EMBER_COUN-
TER_ASH_XOFF,
EMBER_COUNTER_APS_LINK_KEY_NOT_AUTHORIZED, EMBER_COUNTER_NWK_DE-
CRYPTION_FAILURE, EMBER_COUNTER_APS_DECRYPTION_FAILURE, EMBER_COU-
NTER_ALLOCATE_PACKET_BUFFER_FAILURE,
EMBER_COUNTER_RELAYED_UNICAST, EMBER_COUNTER_PHY_TO_MAC_QUEUE_L-
IMIT_REACHED, EMBER_COUNTER_PACKET_VALIDATE_LIBRARY_DROPPED_COUN-
T, EMBER_COUNTER_TYPE_COUNT }
- enum EmberInitialSecurityBitmask {
EMBER_DISTRIBUTED_TRUST_CENTER_MODE, EMBER_TRUST_CENTER_GLOBAL_L-
INK_KEY, EMBER_PRECONFIGURED_NETWORK_KEY_MODE, EMBER_HAVE_TRUST-
_CENTER_EUI64,
EMBER_TRUST_CENTER_USES_HASHED_LINK_KEY, EMBER_HAVE_PRECONFIGURE-
D_KEY, EMBER_HAVE_NETWORK_KEY, EMBER_GET_LINK_KEY_WHEN_JOINING,
EMBER_REQUIRE_ENCRYPTED_KEY, EMBER_NO_FRAME_COUNTER_RESET, EMBER-
_GET_PRECONFIGURED_KEY_FROM_INSTALL_CODE }
- enum EmberExtendedSecurityBitmask { EMBER_JOINER_GLOBAL_LINK_KEY, EMBER_N-
WK_LEAVE_REQUEST_NOT_ALLOWED }
- enum EmberCurrentSecurityBitmask {
EMBER_STANDARD_SECURITY_MODE_, EMBER_DISTRIBUTED_TRUST_CENTER_MO-
DE_, EMBER_TRUST_CENTER_GLOBAL_LINK_KEY_, EMBER_HAVE_TRUST_CENTER-
_LINK_KEY,
EMBER_TRUST_CENTER_USES_HASHED_LINK_KEY_ }

- enum EmberKeyStructBitmask {
  EMBER_KEY_HAS_SEQUENCE_NUMBER, EMBER_KEY_HAS_OUTGOING_FRAME_CO-
  UNTER, EMBER_KEY_HAS_INCOMING_FRAME_COUNTER, EMBER_KEY_HAS_PARTN-
  ER_EUI64,
  EMBER_KEY_IS_AUTHORIZED, EMBER_KEY_PARTNER_IS_SLEEPY }
- enum EmberKeyType {
  EMBER_TRUST_CENTER_LINK_KEY, EMBER_TRUST_CENTER_MASTER_KEY, EMBE-
  R_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, E-
  MBER_TRUST_CENTER_LINK_KEY_ESTABLISHED, EMBER_KEY_ESTABLISHMENT_T-
  IMEOUT,
  EMBER_KEY_TABLE_FULL, EMBER_TC_RESPONDED_TO_KEY_REQUEST, EMBER_T-
  C_APP_KEY_SENT_TO_REQUESTER, EMBER_TC_RESPONSE_TO_KEY_REQUEST_FAIL-
  ED,
  EMBER_TC_REQUEST_KEY_TYPE_NOT_SUPPORTED, EMBER_TC_NO_LINK_KEY_FO-
  R_REQUESTER, EMBER_TC_REQUESTER_EUI64_UNKNOWN, EMBER_TC_RECEIVED_-
  FIRST_APP_KEY_REQUEST,
  EMBER_TC_TIMEOUT_WAITING_FOR_SECOND_APP_KEY_REQUEST, EMBER_TC_NO-
  N_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_K-
  EY_REQUESTS }
- enum EmberKeySettings { EMBER_KEY_PERMISSIONS_NONE, EMBER_KEY_PERMISSIO-
  NS_READING_ALLOWED, EMBER_KEY_PERMISSIONS_HASHING_ALLOWED }
- enum EmberMacPassthroughType {
  EMBER_MAC_PASSTHROUGH_NONE, EMBER_MAC_PASSTHROUGH_SE_INTERPAN, E-
  MBER_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 EmberVersionType { EMBER_VERSION_TYPE_PRE_RELEASE, EMBER_VERSION_T-
  YPE_GA }
- enum EmberLeaveRequestFlags { EMBER_ZIGBEE_LEAVE_AND_REJOIN, EMBER_ZIGBE-
  E_LEAVE_AND_REMOVE_CHILDREN }

- enum EmberLeaveReason {
  EMBER_LEAVE_REASON_NONE, EMBER_LEAVE_DUE_TO_NWK_LEAVE_MESSAGE, E-
  MBER_LEAVE_DUE_TO_APS_REMOVE_MESSAGE, EMBER_LEAVE_DUE_TO_ZDO_LE-
  AVE_MESSAGE,
  EMBER_LEAVE_DUE_TO_ZLL_TOUCHLINK, EMBER_LEAVE_DUE_TO_APP_EVENT_1 }
- typedef int8u EmberStatus
- typedef int8u EmberEUI64 [EUI64_SIZE]
- typedef int8u EmberMessageBuffer
- typedef int16u EmberNodeId
- typedef int16u EmberMulticastId
- typedef int16u EmberPanId
- const EmberVersion emberVersion
- #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 __EMBERSTATUS_TYPE__
- #define EMBER_MAX_802_15_4_CHANNEL_NUMBER
- #define EMBER_MIN_802_15_4_CHANNEL_NUMBER
- #define EMBER_NUM_802_15_4_CHANNELS
- #define EMBER_ALL_802_15_4_CHANNELS_MASK
- #define EMBER_ZIGBEE_COORDINATOR_ADDRESS
- #define EMBER_NULL_NODE_ID
- #define EMBER_NULL_BINDING
- #define EMBER_TABLE_ENTRY_UNUSED_NODE_ID
- #define EMBER_MULTICAST_NODE_ID
- #define EMBER_UNKNOWN_NODE_ID
- #define EMBER_DISCOVERY_ACTIVE_NODE_ID
- #define EMBER_NULL_ADDRESS_TABLE_INDEX
- #define EMBER_ZDO_ENDPOINT
- #define EMBER_BROADCAST_ENDPOINT
- #define EMBER_ZDO_PROFILE_ID
- #define EMBER_WILDCARD_PROFILE_ID
- #define EMBER_MAXIMUM_STANDARD_PROFILE_ID
- #define EMBER_BROADCAST_TABLE_TIMEOUT_QS

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

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_DEVIC-
  E_NOT_FOUND, EMBER_ZDP_INVALID_ENDPOINT,
  EMBER_ZDP_NOT_ACTIVE, EMBER_ZDP_NOT_SUPPORTED, EMBER_ZDP_TIMEOUT, E-
  MBER_ZDP_NO_MATCH,
  EMBER_ZDP_NO_ENTRY, EMBER_ZDP_NO_DESCRIPTOR, EMBER_ZDP_INSUFFICIEN-
  T_SPACE, EMBER_ZDP_NOT_PERMITTED,
  EMBER_ZDP_TABLE_FULL, EMBER_ZDP_NOT_AUTHORIZED, EMBER_NWK_ALREAD-
  Y_PRESENT, EMBER_NWK_TABLE_FULL,
  EMBER_NWK_UNKNOWN_DEVICE }

Network and IEEE Address Request/Response

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

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

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

- #define NETWORK_ADDRESS_REQUEST
- #define NETWORK_ADDRESS_RESPONSE
- #define IEEE_ADDRESS_REQUEST
- #define IEEE_ADDRESS_RESPONSE

Node Descriptor Request/Response

```
<br>

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

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

- #define NODE_DESCRIPTOR_REQUEST
- #define NODE_DESCRIPTOR_RESPONSE

Power Descriptor Request / Response

```
<br>

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

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

- #define POWER_DESCRIPTOR_REQUEST
- #define POWER_DESCRIPTOR_RESPONSE

Simple Descriptor Request / Response

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

- #define SIMPLE_DESCRIPTOR_REQUEST
- #define SIMPLE_DESCRIPTOR_RESPONSE

Active Endpoints Request / Response

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

- #define ACTIVE_ENDPOINTS_REQUEST
- #define ACTIVE_ENDPOINTS_RESPONSE

Match Descriptors Request / Response

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

- #define MATCH_DESCRIPTORS_REQUEST
- #define MATCH_DESCRIPTORS_RESPONSE

Discovery Cache Request / Response

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

- #define DISCOVERY_CACHE_REQUEST
- #define DISCOVERY_CACHE_RESPONSE

End Device Announce and End Device Announce Response

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

- #define END_DEVICE_ANNOUNCE
- #define END_DEVICE_ANNOUNCE_RESPONSE

System Server Discovery Request / Response

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

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

- #define SYSTEM_SERVER_DISCOVERY_REQUEST
- #define SYSTEM_SERVER_DISCOVERY_RESPONSE

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_BINDI-
  NG_TABLE_CACHE,
  EMBER_ZDP_PRIMARY_DISCOVERY_CACHE, EMBER_ZDP_SECONDARY_DISCOVER-
  Y_CACHE, EMBER_ZDP_NETWORK_MANAGER }

Find Node Cache Request / Response

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

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

- #define FIND_NODE_CACHE_REQUEST
- #define FIND_NODE_CACHE_RESPONSE

End Device Bind Request / Response

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

- #define END_DEVICE_BIND_REQUEST
- #define END_DEVICE_BIND_RESPONSE

Binding types and Request / Response

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

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

- #define UNICAST_BINDING
- #define UNICAST_MANY_TO_ONE_BINDING
- #define MULTICAST_BINDING
- #define BIND_REQUEST
- #define BIND_RESPONSE
- #define UNBIND_REQUEST
- #define UNBIND_RESPONSE

LQI Table Request / Response

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

The device-type byte has the following fields:

```
   Name            Mask        Values

device type        0x03        0x00 coordinator
                               0x01 router
                               0x02 end device
                               0x03 unknown

rx mode            0x0C        0x00 off when idle
                               0x04 on when idle
                               0x08 unknown

relationship       0x70        0x00 parent
                               0x10 child
                               0x20 sibling
                               0x30 other
                               0x40 previous child
reserved           0x10
```

The permit-joining byte has the following fields

```
   Name            Mask        Values

permit joining     0x03        0x00 not accepting join requests
                               0x01 accepting join requests
                               0x02 unknown
reserved           0xFC
```

- #define LQI_TABLE_REQUEST
- #define LQI_TABLE_RESPONSE

Routing Table Request / Response

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

The status byte has the following fields:

```
   Name            Mask        Values

status             0x07        0x00 active
                               0x01 discovery underway
                               0x02 discovery failed
                               0x03 inactive
                               0x04 validation underway

flags              0x38
                               0x08 memory constrained
                               0x10 many-to-one
                               0x20 route record required

reserved           0xC0
```

- #define ROUTING_TABLE_REQUEST
- #define ROUTING_TABLE_RESPONSE

Binding Table Request / Response

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

Note

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

- #define BINDING_TABLE_REQUEST
- #define BINDING_TABLE_RESPONSE

Leave Request / Response

```
Request:  <transaction sequence number: 1> <EUI64:8> <flags:1>
          The flag bits are:
          0x40 remove children
          0x80 rejoin
Response: <transaction sequence number: 1> <status:1>
```

- #define LEAVE_REQUEST
- #define LEAVE_RESPONSE
- #define LEAVE_REQUEST_REMOVE_CHILDREN_FLAG
- #define LEAVE_REQUEST_REJOIN_FLAG

Permit Joining Request / Response

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

- #define PERMIT_JOINING_REQUEST
- #define PERMIT_JOINING_RESPONSE

Network Update Request / Response

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

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

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

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

```
   Unicast requests always get a response, which is INVALID_REQUEST if the
   duration is not a legal value.

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

- #define NWK_UPDATE_REQUEST
- #define NWK_UPDATE_RESPONSE

Unsupported

Not mandatory and not supported.

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

ZDO configuration flags.

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

EMBER_APPLICATION_RECEIVES_SUPPORTED_ZDO_REQUESTS EMBER_APPLICATION_H-ANDLES_UNSUPPORTED_ZDO_REQUESTS EMBER_APPLICATION_HANDLES_ENDPOINT_Z-DO_REQUESTS EMBER_APPLICATION_HANDLES_BINDING_ZDO_REQUESTS

See ember-configuration.h for more information.

- enum EmberZdoConfigurationFlags { EMBER_APP_RECEIVES_SUPPORTED_ZDO_REQUES-TS, EMBER_APP_HANDLES_UNSUPPORTED_ZDO_REQUESTS, EMBER_APP_HANDLE-S_ZDO_ENDPOINT_REQUESTS, EMBER_APP_HANDLES_ZDO_BINDING_REQUESTS }

6.3.1 Detailed Description

See ember-types.h for source code.

6.3.2 Macro Definition Documentation

6.3.2.1 #define EUI64_SIZE

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

Definition at line 72 of file ember-types.h.

6.3.2.2 #define EXTENDED_PAN_ID_SIZE

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

Definition at line 77 of file ember-types.h.

6.3.2.3 #define EMBER_ENCRYPTION_KEY_SIZE

Size of an encryption key in bytes (16).

Definition at line 82 of file ember-types.h.

6.3.2.4 #define EMBER_CERTIFICATE_SIZE

Size of Implicit Certificates used for Certificate Based Key Exchange.

Definition at line 88 of file ember-types.h.

6.3.2.5 #define EMBER_PUBLIC_KEY_SIZE

Size of Public Keys used in Elliptical Cryptography ECMQV algorithms.

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

6.3.2.6 #define EMBER_PRIVATE_KEY_SIZE

Size of Private Keys used in Elliptical Cryptography ECMQV algorithms.

Definition at line 98 of file ember-types.h.

6.3.2.7 #define EMBER_SMAC_SIZE

Size of the SMAC used in Elliptical Cryptography ECMQV algorithms.

Definition at line 103 of file ember-types.h.

6.3.2.8 #define EMBER_SIGNATURE_SIZE

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

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

6.3.2.9   #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 114 of file ember-types.h.

6.3.2.10   #define __EMBERSTATUS_TYPE__

Return type for Ember functions.

Definition at line 121 of file ember-types.h.

6.3.2.11   #define EMBER_MAX_802_15_4_CHANNEL_NUMBER

The maximum 802.15.4 channel number is 26.

Definition at line 157 of file ember-types.h.

6.3.2.12   #define EMBER_MIN_802_15_4_CHANNEL_NUMBER

The minimum 802.15.4 channel number is 11.

Definition at line 162 of file ember-types.h.

6.3.2.13   #define EMBER_NUM_802_15_4_CHANNELS

There are sixteen 802.15.4 channels.

Definition at line 167 of file ember-types.h.

6.3.2.14   #define EMBER_ALL_802_15_4_CHANNELS_MASK

Bitmask to scan all 802.15.4 channels.

Definition at line 173 of file ember-types.h.

6.3.2.15   #define EMBER_ZIGBEE_COORDINATOR_ADDRESS

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

Definition at line 178 of file ember-types.h.

6.3.2.16   #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 184 of file ember-types.h.

6.3.2.17   #define EMBER_NULL_BINDING

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

Definition at line 190 of file ember-types.h.

6.3.2.18 #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 201 of file ember-types.h.

6.3.2.19 #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 209 of file ember-types.h.

6.3.2.20 #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 218 of file ember-types.h.

6.3.2.21 #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 227 of file ember-types.h.

6.3.2.22 #define EMBER_NULL_ADDRESS_TABLE_INDEX

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

Definition at line 233 of file ember-types.h.

6.3.2.23 #define EMBER_ZDO_ENDPOINT

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

Definition at line 238 of file ember-types.h.

6.3.2.24 #define EMBER_BROADCAST_ENDPOINT

The broadcast endpoint, as defined in the ZigBee spec.

Definition at line 243 of file ember-types.h.

6.3.2.25 #define EMBER_ZDO_PROFILE_ID

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

Definition at line 248 of file ember-types.h.

6.3.2.26 #define EMBER_WILDCARD_PROFILE_ID

The profile ID used to address all the public profiles.

Definition at line 253 of file ember-types.h.

6.3.2.27 #define EMBER_MAXIMUM_STANDARD_PROFILE_ID

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

Definition at line 258 of file ember-types.h.

6.3.2.28 #define EMBER_BROADCAST_TABLE_TIMEOUT_QS

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

Definition at line 266 of file ember-types.h.

6.3.2.29 #define EMBER_BROADCAST_ADDRESS

Broadcast to all routers.

Definition at line 315 of file ember-types.h.

6.3.2.30 #define EMBER_RX_ON_WHEN_IDLE_BROADCAST_ADDRESS

Broadcast to all non-sleepy devices.

Definition at line 317 of file ember-types.h.

6.3.2.31 #define EMBER_SLEEPY_BROADCAST_ADDRESS

Broadcast to all devices, including sleepy end devices.

Definition at line 319 of file ember-types.h.

6.3.2.32 #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 586 of file ember-types.h.

6.3.2.33   #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 591 of file ember-types.h.

6.3.2.34   #define EMBER_JOIN_DECISION_STRINGS

@ brief Defines the CLI enumerations for the EmberJoinDecision enum

Definition at line 619 of file ember-types.h.

6.3.2.35   #define EMBER_DEVICE_UPDATE_STRINGS

@ brief Defines the CLI enumerations for the EmberDeviceUpdate enum.

Definition at line 654 of file ember-types.h.

6.3.2.36   #define emberInitializeNetworkParameters(*parameters*)

Definition at line 826 of file ember-types.h.

6.3.2.37   #define EMBER_COUNTER_STRINGS

@ brief Defines the CLI enumerations for the EmberCounterType enum.

Definition at line 1086 of file ember-types.h.

6.3.2.38   #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 1194 of file ember-types.h.

6.3.2.39   #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 1198 of file ember-types.h.

6.3.2.40   #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 1203 of file ember-types.h.

6.3.2.41   #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 1208 of file ember-types.h.

6.3.2.42   #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 1232 of file ember-types.h.

6.3.2.43   #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_TIME-OUT). 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 emberIncomingMessage-Handler(). 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 1272 of file ember-types.h.

6.3.2.44   #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 emberIncoming-MessageHandler(). 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 1310 of file ember-types.h.

6.3.2.45   #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 1327 of file ember-types.h.

6.3.2.46   #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 1332 of file ember-types.h.

6.3.2.47 #define EMBER_REPORT_COUNTERS_RESPONSE

The cluster id used to respond to an EMBER_REPORT_COUNTERS_REQUEST.

Definition at line 1335 of file ember-types.h.

6.3.2.48 #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 1341 of file ember-types.h.

6.3.2.49 #define EMBER_REPORT_AND_CLEAR_COUNTERS_RESPONSE

The cluster id used to respond to an EMBER_REPORT_AND_CLEAR_COUNTERS_REQUEST.

Definition at line 1344 of file ember-types.h.

6.3.2.50 #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 1350 of file ember-types.h.

6.3.2.51 #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 1414 of file ember-types.h.

6.3.2.52 #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 1419 of file ember-types.h.

6.3.2.53 #define EMBER_NO_TRUST_CENTER_MODE

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

Definition at line 1559 of file ember-types.h.

6.3.2.54 #define EMBER_GLOBAL_LINK_KEY

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

Definition at line 1563 of file ember-types.h.

6.3.2.55 #define EMBER_MFG_SECURITY_CONFIG_MAGIC_NUMBER

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

Definition at line 1901 of file ember-types.h.

6.3.2.56 #define EMBER_MAC_FILTER_MATCH_ENABLED_MASK

Definition at line 1940 of file ember-types.h.

6.3.2.57 #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_MASK

Definition at line 1941 of file ember-types.h.

6.3.2.58 #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_MASK

Definition at line 1942 of file ember-types.h.

6.3.2.59 #define EMBER_MAC_FILTER_MATCH_ON_DEST_MASK

Definition at line 1943 of file ember-types.h.

6.3.2.60 #define EMBER_MAC_FILTER_MATCH_ON_SOURCE_MASK

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

6.3.2.61 #define EMBER_MAC_FILTER_MATCH_ENABLED

Definition at line 1947 of file ember-types.h.

6.3.2.62 #define EMBER_MAC_FILTER_MATCH_DISABLED

Definition at line 1948 of file ember-types.h.

6.3.2.63 #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_NONE

Definition at line 1951 of file ember-types.h.

6.3.2.64 #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_LOCAL

Definition at line 1952 of file ember-types.h.

6.3.2.65 #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_BROADCAST

Definition at line 1953 of file ember-types.h.

6.3.2.66 #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_NONE

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

6.3.2.67 #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_NON_LOCAL

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

6.3.2.68 #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_LOCAL

Definition at line 1958 of file ember-types.h.

6.3.2.69 #define EMBER_MAC_FILTER_MATCH_ON_DEST_BROADCAST_SHORT

Definition at line 1961 of file ember-types.h.

6.3.2.70 #define EMBER_MAC_FILTER_MATCH_ON_DEST_UNICAST_SHORT

Definition at line 1962 of file ember-types.h.

6.3.2.71 #define EMBER_MAC_FILTER_MATCH_ON_DEST_UNICAST_LONG

Definition at line 1963 of file ember-types.h.

6.3.2.72 #define EMBER_MAC_FILTER_MATCH_ON_SOURCE_LONG

Definition at line 1966 of file ember-types.h.

6.3.2.73 #define EMBER_MAC_FILTER_MATCH_ON_SOURCE_SHORT

Definition at line 1967 of file ember-types.h.

6.3.2.74 #define EMBER_MAC_FILTER_MATCH_END

Definition at line 1970 of file ember-types.h.

6.3.2.75 #define NETWORK_ADDRESS_REQUEST

Definition at line 2054 of file ember-types.h.

6.3.2.76 #define NETWORK_ADDRESS_RESPONSE

Definition at line 2055 of file ember-types.h.

6.3.2.77 #define IEEE_ADDRESS_REQUEST

Definition at line 2056 of file ember-types.h.

6.3.2.78 #define IEEE_ADDRESS_RESPONSE

Definition at line 2057 of file ember-types.h.

6.3.2.79 #define NODE_DESCRIPTOR_REQUEST

Definition at line 2085 of file ember-types.h.

6.3.2.80 #define NODE_DESCRIPTOR_RESPONSE

Definition at line 2086 of file ember-types.h.

6.3.2.81 #define POWER_DESCRIPTOR_REQUEST

Definition at line 2099 of file ember-types.h.

6.3.2.82 #define POWER_DESCRIPTOR_RESPONSE

Definition at line 2100 of file ember-types.h.

6.3.2.83 #define SIMPLE_DESCRIPTOR_REQUEST

Definition at line 2116 of file ember-types.h.

6.3.2.84 #define SIMPLE_DESCRIPTOR_RESPONSE

Definition at line 2117 of file ember-types.h.

6.3.2.85 #define ACTIVE_ENDPOINTS_REQUEST

Definition at line 2128 of file ember-types.h.

6.3.2.86 #define ACTIVE_ENDPOINTS_RESPONSE

Definition at line 2129 of file ember-types.h.

6.3.2.87 #define MATCH_DESCRIPTORS_REQUEST

Definition at line 2143 of file ember-types.h.

6.3.2.88 #define MATCH_DESCRIPTORS_RESPONSE

Definition at line 2144 of file ember-types.h.

6.3.2.89 #define DISCOVERY_CACHE_REQUEST

Definition at line 2156 of file ember-types.h.

6.3.2.90 #define DISCOVERY_CACHE_RESPONSE

Definition at line 2157 of file ember-types.h.

6.3.2.91 #define END_DEVICE_ANNOUNCE

Definition at line 2168 of file ember-types.h.

6.3.2.92 #define END_DEVICE_ANNOUNCE_RESPONSE

Definition at line 2169 of file ember-types.h.

6.3.2.93 #define SYSTEM_SERVER_DISCOVERY_REQUEST

Definition at line 2183 of file ember-types.h.

6.3.2.94 #define SYSTEM_SERVER_DISCOVERY_RESPONSE

Definition at line 2184 of file ember-types.h.

6.3.2.95 #define FIND_NODE_CACHE_REQUEST

Definition at line 2221 of file ember-types.h.

6.3.2.96 #define FIND_NODE_CACHE_RESPONSE

Definition at line 2222 of file ember-types.h.

6.3.2.97 #define END_DEVICE_BIND_REQUEST

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

6.3.2.98 #define END_DEVICE_BIND_RESPONSE

Definition at line 2236 of file ember-types.h.

6.3.2.99 #define UNICAST_BINDING

Definition at line 2256 of file ember-types.h.

6.3.2.100 #define UNICAST_MANY_TO_ONE_BINDING

Definition at line 2257 of file ember-types.h.

6.3.2.101 #define MULTICAST_BINDING

Definition at line 2258 of file ember-types.h.

6.3.2.102 #define BIND_REQUEST

Definition at line 2260 of file ember-types.h.

6.3.2.103 #define BIND_RESPONSE

Definition at line 2261 of file ember-types.h.

6.3.2.104 #define UNBIND_REQUEST

Definition at line 2262 of file ember-types.h.

6.3.2.105 #define UNBIND_RESPONSE

Definition at line 2263 of file ember-types.h.

6.3.2.106 #define LQI_TABLE_REQUEST

Definition at line 2313 of file ember-types.h.

6.3.2.107 #define LQI_TABLE_RESPONSE

Definition at line 2314 of file ember-types.h.

6.3.2.108 #define ROUTING_TABLE_REQUEST

Definition at line 2349 of file ember-types.h.

6.3.2.109 #define ROUTING_TABLE_RESPONSE

Definition at line 2350 of file ember-types.h.

6.3.2.110 #define BINDING_TABLE_REQUEST

Definition at line 2371 of file ember-types.h.

6.3.2.111 #define BINDING_TABLE_RESPONSE

Definition at line 2372 of file ember-types.h.

6.3.2.112 #define LEAVE_REQUEST

Definition at line 2385 of file ember-types.h.

6.3.2.113 #define LEAVE_RESPONSE

Definition at line 2386 of file ember-types.h.

6.3.2.114 #define LEAVE_REQUEST_REMOVE_CHILDREN_FLAG

Definition at line 2388 of file ember-types.h.

6.3.2.115 #define LEAVE_REQUEST_REJOIN_FLAG

Definition at line 2389 of file ember-types.h.

6.3.2.116 #define PERMIT_JOINING_REQUEST

Definition at line 2400 of file ember-types.h.

6.3.2.117 #define PERMIT_JOINING_RESPONSE

Definition at line 2401 of file ember-types.h.

6.3.2.118 #define NWK_UPDATE_REQUEST

Definition at line 2429 of file ember-types.h.

6.3.2.119 #define NWK_UPDATE_RESPONSE

Definition at line 2430 of file ember-types.h.

6.3.2.120 #define COMPLEX_DESCRIPTOR_REQUEST

Definition at line 2436 of file ember-types.h.

6.3.2.121 #define COMPLEX_DESCRIPTOR_RESPONSE

Definition at line 2437 of file ember-types.h.

6.3.2.122 #define USER_DESCRIPTOR_REQUEST

Definition at line 2438 of file ember-types.h.

6.3.2.123 #define USER_DESCRIPTOR_RESPONSE

Definition at line 2439 of file ember-types.h.

6.3.2.124 #define DISCOVERY_REGISTER_REQUEST

Definition at line 2440 of file ember-types.h.

6.3.2.125 #define DISCOVERY_REGISTER_RESPONSE

Definition at line 2441 of file ember-types.h.

6.3.2.126 #define USER_DESCRIPTOR_SET

Definition at line 2442 of file ember-types.h.

6.3.2.127 #define USER_DESCRIPTOR_CONFIRM

Definition at line 2443 of file ember-types.h.

6.3.2.128 #define NETWORK_DISCOVERY_REQUEST

Definition at line 2444 of file ember-types.h.

6.3.2.129 #define NETWORK_DISCOVERY_RESPONSE

Definition at line 2445 of file ember-types.h.

6.3.2.130 #define DIRECT_JOIN_REQUEST

Definition at line 2446 of file ember-types.h.

6.3.2.131 #define DIRECT_JOIN_RESPONSE

Definition at line 2447 of file ember-types.h.

6.3.2.132 #define CLUSTER_ID_RESPONSE_MINIMUM

Definition at line 2450 of file ember-types.h.

6.3.3 Typedef Documentation

6.3.3.1 typedef int8u EmberStatus

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

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

6.3.3.2 typedef int8u EmberEUI64[EUI64_SIZE]

EUI 64-bit ID (an IEEE address).

Definition at line 128 of file ember-types.h.

6.3.3.3 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 139 of file ember-types.h.

6.3.3.4 typedef int16u EmberNodeId

16-bit ZigBee network address.

Definition at line 144 of file ember-types.h.

6.3.3.5 typedef int16u EmberMulticastId

16-bit ZigBee multicast group identifier.

Definition at line 147 of file ember-types.h.

6.3.3.6 typedef int16u EmberPanId

802.15.4 PAN ID.

Definition at line 152 of file ember-types.h.

6.3.3.7 typedef int8u EmberTaskId

brief An identifier for a task

Definition at line 1121 of file ember-types.h.

6.3.3.8 typedef { ... } EmberEventData

6.3.3.9 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 1938 of file ember-types.h.

6.3.3.10 typedef int8u EmberLibraryStatus

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

Definition at line 1985 of file ember-types.h.

6.3.4 Enumeration Type Documentation

6.3.4.1 enum EmberVersionType

Type of Ember software version.

Enumerator:

EMBER_VERSION_TYPE_PRE_RELEASE
EMBER_VERSION_TYPE_GA

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

6.3.4.2 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 270 of file ember-types.h.

6.3.4.3 enum EmberLeaveReason

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

Enumerator:

EMBER_LEAVE_REASON_NONE
EMBER_LEAVE_DUE_TO_NWK_LEAVE_MESSAGE
EMBER_LEAVE_DUE_TO_APS_REMOVE_MESSAGE
EMBER_LEAVE_DUE_TO_ZDO_LEAVE_MESSAGE
EMBER_LEAVE_DUE_TO_ZLL_TOUCHLINK
EMBER_LEAVE_DUE_TO_APP_EVENT_1

Definition at line 284 of file ember-types.h.

6.3.4.4 enum EmberNodeType

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

Enumerator:

> **EMBER_UNKNOWN_DEVICE**   Device is not joined
>
> **EMBER_COORDINATOR**   Will relay messages and can act as a parent to other nodes.
>
> **EMBER_ROUTER**   Will relay messages and can act as a parent to other nodes.
>
> **EMBER_END_DEVICE**   Communicates only with its parent and will not relay messages.
>
> **EMBER_SLEEPY_END_DEVICE**   An end device whose radio can be turned off to save power. The application must call emberPollForData() to receive messages.
>
> **EMBER_MOBILE_END_DEVICE**   A sleepy end device that can move through the network.

Definition at line 329 of file ember-types.h.

6.3.4.5 enum EmberNetworkInitBitmask

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

Enumerator:

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

Definition at line 369 of file ember-types.h.

6.3.4.6 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 399 of file ember-types.h.

6.3.4.7 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 466 of file ember-types.h.

6.3.4.8 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 emberMessage-SentHandler() only. It may not be passed to emberSendUnicast().

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

Definition at line 491 of file ember-types.h.

6.3.4.9 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 516 of file ember-types.h.

6.3.4.10 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 540 of file ember-types.h.

6.3.4.11 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 557 of file ember-types.h.

6.3.4.12 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 600 of file ember-types.h.

6.3.4.13 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*
> *EMBER_REJOIN_REASON_NONE*
> *EMBER_REJOIN_DUE_TO_NWK_KEY_UPDATE*
> *EMBER_REJOIN_DUE_TO_LEAVE_MESSAGE*
> *EMBER_REJOIN_DUE_TO_NO_PARENT*
> *EMBER_REJOIN_DUE_TO_ZLL_TOUCHLINK*
> *EMBER_REJOIN_DUE_TO_APP_EVENT_5*
> *EMBER_REJOIN_DUE_TO_APP_EVENT_4*
> *EMBER_REJOIN_DUE_TO_APP_EVENT_3*
> *EMBER_REJOIN_DUE_TO_APP_EVENT_2*
> *EMBER_REJOIN_DUE_TO_APP_EVENT_1*

Definition at line 634 of file ember-types.h.

6.3.4.14 enum EmberDeviceUpdate

Notes the last rejoin reason.

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

EMBER_REJOIN_REASON_NONE

EMBER_REJOIN_DUE_TO_NWK_KEY_UPDATE

EMBER_REJOIN_DUE_TO_LEAVE_MESSAGE

EMBER_REJOIN_DUE_TO_NO_PARENT

EMBER_REJOIN_DUE_TO_ZLL_TOUCHLINK

EMBER_REJOIN_DUE_TO_APP_EVENT_5

EMBER_REJOIN_DUE_TO_APP_EVENT_4

EMBER_REJOIN_DUE_TO_APP_EVENT_3

EMBER_REJOIN_DUE_TO_APP_EVENT_2

EMBER_REJOIN_DUE_TO_APP_EVENT_1

Definition at line 668 of file ember-types.h.

6.3.4.15   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 698 of file ember-types.h.

6.3.4.16   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 716 of file ember-types.h.

6.3.4.17 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 emberTrust-CenterJoinHandler(). 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 741 of file ember-types.h.

6.3.4.18 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_COUN-TER_APS_DATA_TX_UNICAST_SUCCESS` and `EMBER_COUNTER_APS_DATA_TX_UNI-CAST_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 ember-Tick(). After that limit is hit, packets will be dropped. The number of dropped packets will be recorded in this counter.

NOTE: For each call to emberCounterHandler() there may be more than 1 packet that was dropped due to the limit reached. The actual number of packets dropped will be returned in the 'data' parameter passed to that function.

EMBER_COUNTER_PACKET_VALIDATE_LIBRARY_DROPPED_COUNT   The number of times we dropped a packet due to the packet-validate library checking a packet and rejecting it due to length or other formatting problems.

EMBER_COUNTER_TYPE_COUNT   A placeholder giving the number of Ember counter types.

Definition at line 964 of file ember-types.h.

6.3.4.19 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_M-ODE)

EMBER_TRUST_CENTER_GLOBAL_LINK_KEY   This enables a Global Link Key for the Trust Center. All nodes will share the same Trust Center Link Key.

EMBER_PRECONFIGURED_NETWORK_KEY_MODE   This enables devices that perform MAC Association with a pre-configured Network Key to join the network. It is only set on the Trust Center.

EMBER_HAVE_TRUST_CENTER_EUI64   This denotes that the EmberInitialSecurityState::preconfigured-TrustCenterEui64 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::preconfiguredTrustCenter-Eui64 must be populated with the appropriate EUI64.

EMBER_TRUST_CENTER_USES_HASHED_LINK_KEY   This denotes that the EmberInitialSecurity-State::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::preconfigured-Key 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 1426 of file ember-types.h.

6.3.4.20 enum EmberExtendedSecurityBitmask

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

Enumerator:

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

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

Definition at line 1519 of file ember-types.h.

6.3.4.21 enum EmberCurrentSecurityBitmask

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

Enumerator:

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

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

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

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

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

Definition at line 1620 of file ember-types.h.

6.3.4.22 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 1672 of file ember-types.h.

6.3.4.23 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 1707 of file ember-types.h.

6.3.4.24 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 1758 of file ember-types.h.

6.3.4.25 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 1793 of file ember-types.h.

6.3.4.26 enum EmberKeySettings

Enumerator:

EMBER_KEY_PERMISSIONS_NONE
EMBER_KEY_PERMISSIONS_READING_ALLOWED
EMBER_KEY_PERMISSIONS_HASHING_ALLOWED

Definition at line 1877 of file ember-types.h.

6.3.4.27 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 boot-loader 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 1909 of file ember-types.h.

6.3.4.28 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 1998 of file ember-types.h.

6.3.4.29 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 2192 of file ember-types.h.

6.3.4.30 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 2466 of file ember-types.h.

6.3.5 Function Documentation

6.3.5.1 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.

6.3.5.2 int8u∗ emberCertificateContents (EmberCertificateData ∗ *cert*)

This function allows the programmer to gain access to the actual certificate data bytes of the Ember-CertificateData struct.

Parameters

cert	A Pointer to an EmberCertificateData structure.

Returns

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

6.3.5.3 int8u∗ emberPublicKeyContents (EmberPublicKeyData ∗ *key*)

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

Parameters

key	A Pointer to an EmberPublicKeyData structure.

Returns

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

6.3.5.4 int8u∗ emberPrivateKeyContents (EmberPrivateKeyData ∗ *key*)

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

Parameters

key	A Pointer to an EmberPrivateKeyData structure.

Returns

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

6.3.5.5 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.

6.3.5.6 int8u∗ emberSignatureContents (EmberSignatureData ∗ *sig*)

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

6.3.6 Variable Documentation

6.3.6.1 const EmberVersion emberVersion

Struct containing the version info.

6.4 Network Formation

Functions

- EmberStatus emberInit (void)
- void emberTick (void)
- EmberStatus emberNetworkInit (void)
- EmberStatus emberNetworkInitExtended (EmberNetworkInitStruct ∗networkInitStruct)
- EmberStatus emberFormNetwork (EmberNetworkParameters ∗parameters)
- EmberStatus emberPermitJoining (int8u duration)
- EmberStatus emberJoinNetwork (EmberNodeType nodeType, EmberNetworkParameters ∗parameters)
- EmberStatus emberLeaveNetwork (void)
- EmberStatus emberSendZigbeeLeave (EmberNodeId destination, EmberLeaveRequestFlags flags)
- EmberStatus emberFindAndRejoinNetworkWithReason (boolean haveCurrentNetworkKey, int32u channelMask, EmberRejoinReason reason)
- EmberStatus emberFindAndRejoinNetwork (boolean haveCurrentNetworkKey, int32u channelMask)
- EmberRejoinReason emberGetLastRejoinReason (void)
- EmberStatus emberRejoinNetwork (boolean haveCurrentNetworkKey)
- EmberStatus emberStartScan (EmberNetworkScanType scanType, int32u channelMask, int8u duration)
- EmberStatus emberStopScan (void)
- void emberScanCompleteHandler (int8u channel, EmberStatus status)
- void emberEnergyScanResultHandler (int8u channel, int8s maxRssiValue)
- void emberNetworkFoundHandler (EmberZigbeeNetwork ∗networkFound)
- boolean emberStackIsPerformingRejoin (void)
- EmberLeaveReason emberGetLastLeaveReason (EmberNodeId ∗returnNodeIdThatSentLeave)

6.4.1 Detailed Description

EmberZNet API for finding, forming, joining, and leaving ZigBee networks. See network-formation.h for source code.

6.4.2 Function Documentation

6.4.2.1 EmberStatus emberInit (void)

Initializes the radio and the EmberZNet stack.)

Device configuration functions must be called before emberInit() is called.

Note

The application must check the return value of this function. If the initialization fails, normal messaging functions will not be available. Some failure modes are not fatal, but the application must follow certain procedures to permit recovery. Ignoring the return code will result in unpredictable radio and API behavior. (In particular, problems with association will occur.)

Returns

An EmberStatus value indicating successful initialization or the reason for failure.

6.4.2.2 void emberTick (void)

A periodic tick routine that should be called:

- in the application's main event loop,

- as soon as possible after any radio interrupts, and

- after emberInit().

6.4.2.3 EmberStatus emberNetworkInit (void)

Resume network operation after a reboot.

It is required that this be called on boot prior to ANY network operations. This will initialize the networking system and attempt to resume the previous network identity and configuration. If the node was not previously this routine should still be called.

If the node was previously joined to a network it will retain its original type (e.g. coordinator, router, end device, etc.)

EMBER_NOT_JOINED is returned if the node is not part of a network.

Returns

An EmberStatus value that indicates one of the following:
- successful initialization,
- EMBER_NOT_JOINED if the node is not part of a network, or
- the reason for failure.

6.4.2.4 EmberStatus emberNetworkInitExtended (EmberNetworkInitStruct * *networkInitStruct*)

Resume network operation based on passed parameters.

This routine behaves similar to emberNetworkInit() however the caller can control the operation of the initialization. Either this routine or emberNetworkInit() must be called to initialize the network before any network operations are performed.

6.4.2.5 EmberStatus emberFormNetwork (EmberNetworkParameters * *parameters*)

Forms a new network by becoming the coordinator.

Note

If using security, the application must call emberSetInitialSecurityState() prior to joining the network. This also applies when a device leaves a network and wants to form another one.

Parameters

parameters	Specification of the new network.

Returns

An EmberStatus value that indicates either the successful formation of the new network, or the reason that the network formation failed.

6.4.2.6   EmberStatus emberPermitJoining (int8u *duration*)

Tells the stack to allow other nodes to join the network with this node as their parent. Joining is initially disabled by default. This function may only be called after the node is part of a network and the stack is up.

Parameters

duration	A value of 0x00 disables joining. A value of 0xFF enables joining. Any other value enables joining for that number of seconds.

6.4.2.7   EmberStatus emberJoinNetwork (EmberNodeType *nodeType,* EmberNetworkParameters ∗ *parameters*)

Causes the stack to associate with the network using the specified network parameters. It can take several seconds for the stack to associate with the local network. Do not send messages until a call to the emberStackStatusHandler() callback informs you that the stack is up.

Note

If using security, the application must call emberSetInitialSecurityState() prior to joining the network. This also applies when a device leaves a network and wants to join another one.

Parameters

nodeType	Specification of the role that this node will have in the network. This role must not be EMBER_COORDINATOR. To be a coordinator, call emberFormNetwork().
parameters	Specification of the network with which the node should associate.

Returns

An EmberStatus value that indicates either:

- that the association process began successfully, or
- the reason for failure.

6.4.2.8   EmberStatus emberLeaveNetwork (void)

Causes the stack to leave the current network. This generates a call to the emberStackStatusHandler() callback to indicate that the network is down. The radio will not be used until after a later call to emberFormNetwork() or emberJoinNetwork().

Returns

An EmberStatus value indicating success or reason for failure. A status of EMBER_INVALID_CALL indicates that the node is either not joined to a network or is already in the process of leaving.

6.4.2.9 EmberStatus emberSendZigbeeLeave (EmberNodeId *destination,* EmberLeaveRequestFlags *flags*)

Sends a ZigBee NWK leave command to the specified destination.

Parameters

destination	is the node Id of the device that is being told to leave.
flags	is an bitmask indicating additional considerations for the leave request. See the Ember-LeaveRequestFlags enum for more information. Multiple bits may be set.

Returns

An EmberStatus value indicating success or reason for failure. A status of EMBER_INVALID_CALL indicates that the node not currently joined to the network, or the destination is the local node. To tell the local device to leave, use the emberLeaveNetwork() API.

6.4.2.10 EmberStatus emberFindAndRejoinNetworkWithReason (boolean *haveCurrentNetworkKey,* int32u *channelMask,* EmberRejoinReason *reason*)

The application may call this function when contact with the network has been lost. The most common usage case is when an end device can no longer communicate with its parent and wishes to find a new one. Another case is when a device has missed a Network Key update and no longer has the current Network Key.

Note that a call to emberPollForData() on an end device that has lost contact with its parent will automatically call ::emberRejoinNetwork(TRUE).

The stack will call emberStackStatusHandler() to indicate that the network is down, then try to re-establish contact with the network by performing an active scan, choosing a network with matching extended pan id, and sending a ZigBee network rejoin request. A second call to the emberStackStatusHandler() callback indicates either the success or the failure of the attempt. The process takes approximately 150 milliseconds per channel to complete.

This call replaces the ::emberMobileNodeHasMoved() API from EmberZNet 2.x, which used MAC association and consequently took half a second longer to complete.

Parameters

haveCurrent-NetworkKey	This parameter determines whether the request to rejoin the Network is sent encrypted (TRUE) or unencrypted (FALSE). The application should first try to use encryption. If that fails, the application should call this function again and use no encryption. If the unencrypted rejoin is successful then device will be in the joined but unauthenticated state. The Trust Center will be notified of the rejoin and send an updated Network encrypted using the device's Link Key. Sending the rejoin unencrypted is only supported on networks using Standard Security with link keys (i.e. ZigBee 2006 networks do not support it).
channelMask	A mask indicating the channels to be scanned. See emberStartScan() for format details.
reason	An enemuration indicating why the rejoin occurred. The stack will set the reason based on the ::EmberRejoinReason. An application should use one of the APP_EVENT rejoin reasons. The stack will never use these. Only if the function return code is EMBER_-SUCCESS will the rejoin reason be set.

Returns

    An EmberStatus value indicating success or reason for failure.

6.4.2.11 EmberStatus emberFindAndRejoinNetwork (boolean *haveCurrentNetworkKey,* int32u *channelMask*)

This call is the same emberFindAndRejoinNetworkWithReason() however the reason is assumed to be ::EMBER_REJOIN_REASON_APP_EVENT_1.

6.4.2.12 EmberRejoinReason emberGetLastRejoinReason (void)

Returns the enumeration for why a rejoin previously occurred..

6.4.2.13 EmberStatus emberRejoinNetwork (boolean *haveCurrentNetworkKey*)

A convenience function which calls emberFindAndRejoinNetwork() with a channel mask value for scanning only the current channel. Included for back-compatibility.

6.4.2.14 EmberStatus emberStartScan (EmberNetworkScanType *scanType,* int32u *channelMask,* int8u *duration*)

This function will start a scan. EMBER_SUCCESS signals that the scan successfully started. Note that while a scan can be initiated while the node is currently joined to a network, the node will generally be unable to communication with its PAN during the scan period, so care should be taken when performing scans of any significant duration while presently joined to an existing PAN.

Possible error responses and their meanings:

- EMBER_MAC_SCANNING, we are already scanning.

- EMBER_MAC_BAD_SCAN_DURATION, we have set a duration value that is not 0..14 inclusive.

- EMBER_MAC_INCORRECT_SCAN_TYPE, we have requested an undefined scanning type;

- EMBER_MAC_INVALID_CHANNEL_MASK, our channel mask did not specify any valid channels on the current platform.

Parameters

scanType	Indicates the type of scan to be performed. Possible values: EMBER_ENERGY_SCAN, EMBER_ACTIVE_SCAN.
channelMask	Bits set as 1 indicate that this particular channel should be scanned. Bits set to 0 indicate that this particular channel should not be scanned. For example, a channelMask value of 0x00000001 would indicate that only channel 0 should be scanned. Valid channels range from 11 to 26 inclusive. This translates to a channel mask value of 0x07 FF F8 00. As a convenience, a channelMask of 0 is reinterpreted as the mask for the current channel.
duration	Sets the exponent of the number of scan periods, where a scan period is 960 symbols, and a symbol is 16 microseconds. The scan will occur for $((2^{duration}) + 1)$ scan periods. The value of duration must be less than 15. The time corresponding to the first few values are as follows: 0 = 31 msec, 1 = 46 msec, 2 = 77 msec, 3 = 138 msec, 4 = 261 msec, 5 = 507 msec, 6 = 998 msec.

6.4.2.15   EmberStatus emberStopScan (void)

Terminates a scan in progress.

Returns

> Returns EMBER_SUCCESS if successful.

6.4.2.16   void emberScanCompleteHandler (int8u *channel,* EmberStatus *status*)

Indicates the status of the current scan. When the scan has completed the stack will call this function
with status set to EMBER_SUCCESS. Prior to the scan completing the stack may call this function with
other status values. Non-EMBER_SUCCESS status values indicate that the scan failed to start successfully
on the channel indicated by the channel parameter. The current scan is ongoing until the stack calls this
function with status set to EMBER_SUCCESS.

Parameters

channel	The channel on which the current error occurred. Undefined for the case of EMBER_-SUCCESS.
status	The error condition that occurred on the current channel. Value will be EMBER_SUC-CESS when the scan has completed.

6.4.2.17   void emberEnergyScanResultHandler (int8u *channel,* int8s *maxRssiValue*)

Reports the maximum RSSI value measured on the channel.

Parameters

channel	The 802.15.4 channel number on which the RSSI value was measured.
maxRssiValue	The maximum RSSI value measured (in units of dBm).

6.4.2.18   void emberNetworkFoundHandler (EmberZigbeeNetwork ∗ *networkFound*)

Reports that a network was found, and gives the network parameters useful for deciding which network to
join.

Parameters

networkFound	A pointer to a EmberZigbeeNetwork structure that contains the discovered network and its associated parameters.

6.4.2.19   boolean emberStackIsPerformingRejoin (void)

Indicates whether the stack is in the process of performing a rejoin.

Returns

> Returns TRUE if the device is in the process of performing a rejoin. Returns FALSE otherwise.

6.4.2.20 EmberLeaveReason emberGetLastLeaveReason (EmberNodeId ∗ *returnNodeIdThatSentLeave*)

Indicates the reason why the device left the network (if any). This also will return the device that sent the leave message, if the leave was due to an over-the-air message.

If returnNodeIdThatSentLeave is a non-NULL pointer, then the node Id of the device that sent the leave message will be written to the value pointed to be the pointer. If the leave was not due to an over-the-air message (but an internal API call instead) then EMBER_UNKNOWN_NODE_ID is returned.

Returns

Returns EmberLeaveReason enumeration, or EMBER_LEAVE_REASON_NONE if the device has not left the network.

6.5   Packet Buffers

Macros

- #define LOG_PACKET_BUFFER_SIZE
- #define PACKET_BUFFER_SIZE
- #define EMBER_NULL_MESSAGE_BUFFER
- #define emberMessageBufferLength(buffer)

Functions

- XAP2B_PAGEZERO_ON int8u ∗ emberMessageBufferContents (EmberMessageBuffer buffer)
- void emberSetMessageBufferLength (EmberMessageBuffer buffer, int8u newLength)
- void emberHoldMessageBuffer (EmberMessageBuffer buffer)
- void emberReleaseMessageBuffer (EmberMessageBuffer buffer)
- int8u emberPacketBufferFreeCount (void)

Buffer Functions

- EmberMessageBuffer emberAllocateLinkedBuffers (int8u count)
- EmberMessageBuffer emberFillStackBuffer (int16u count,...)
- #define emberStackBufferLink(buffer)
- #define emberSetStackBufferLink(buffer, newLink)
- #define emberAllocateStackBuffer()

Linked Buffer Utilities

The plural "buffers" in the names of these procedures is a reminder that they deal with linked chains of buffers.

- EmberMessageBuffer emberFillLinkedBuffers (int8u ∗contents, int8u length)
- void emberCopyToLinkedBuffers (int8u ∗contents, EmberMessageBuffer buffer, int8u startIndex, int8u length)
- void emberCopyFromLinkedBuffers (EmberMessageBuffer buffer, int8u startIndex, int8u ∗contents, int8u length)
- void emberCopyBufferBytes (EmberMessageBuffer to, int16u toIndex, EmberMessageBuffer from, int16u fromIndex, int16u count)
- EmberStatus emberAppendToLinkedBuffers (EmberMessageBuffer buffer, int8u ∗contents, int8u length)
- EmberStatus emberAppendPgmToLinkedBuffers (EmberMessageBuffer buffer, PGM_P contents, int8u length)
- EmberStatus emberAppendPgmStringToLinkedBuffers (EmberMessageBuffer buffer, PGM_P suffix)
- EmberStatus emberSetLinkedBuffersLength (EmberMessageBuffer buffer, int8u length)
- int8u ∗ emberGetLinkedBuffersPointer (EmberMessageBuffer buffer, int8u index)
- XAP2B_PAGEZERO_ON int8u emberGetLinkedBuffersByte (EmberMessageBuffer buffer, int8u index)
- XAP2B_PAGEZERO_OFF void emberSetLinkedBuffersByte (EmberMessageBuffer buffer, int8u index, int8u byte)

- int16u emberGetLinkedBuffersLowHighInt16u (EmberMessageBuffer buffer, int8u index)
- void emberSetLinkedBuffersLowHighInt16u (EmberMessageBuffer buffer, int8u index, int16u value)
- int32u emberGetLinkedBuffersLowHighInt32u (EmberMessageBuffer buffer, int8u index)
- void emberSetLinkedBuffersLowHighInt32u (EmberMessageBuffer buffer, int8u index, int32u value)
- EmberMessageBuffer emberCopyLinkedBuffers (EmberMessageBuffer buffer)
- EmberMessageBuffer emberMakeUnsharedLinkedBuffer (EmberMessageBuffer buffer, boolean is-Shared)

6.5.1 Detailed Description

These functions implement a fixed-block-size memory management scheme to store and manipulate Ember-ZNet packets. Buffers are identified to clients with a 1-byte ID.

Buffers can be linked together to create longer packets. The utility procedures allow you to treat a linked chain of buffers as a single buffer.

Freeing a buffer automatically decrements the reference count of any following buffer, possibly freeing the following buffer as well.

Packet buffers may be allocated, held, and released.

See packet-buffer.h for source code.

6.5.2 Macro Definition Documentation

6.5.2.1 #define LOG_PACKET_BUFFER_SIZE

Buffers hold 32 bytes. Defined as the log to ensure it is a power of 2.

Definition at line 38 of file packet-buffer.h.

6.5.2.2 #define PACKET_BUFFER_SIZE

Buffers hold 32 bytes.

Definition at line 42 of file packet-buffer.h.

6.5.2.3 #define EMBER_NULL_MESSAGE_BUFFER

Provides the message buffer equivalent of NULL.

Definition at line 45 of file packet-buffer.h.

6.5.2.4 #define emberMessageBufferLength(*buffer*)

Returns the length of a buffer. Implemented as a macro for improved efficiency.

Parameters

buffer	A buffer.

Returns

Buffer length.

Definition at line 73 of file packet-buffer.h.

6.5.2.5   #define emberStackBufferLink(*buffer*)

Returns the buffer that follows this one in the message. EMBER_NULL_MESSAGE_BUFFER is returned if there is no following buffer.

Parameters

buffer	The buffer whose following buffer is desired.

Returns

Returns the buffer that follows `buffer` in the message. EMBER_NULL_MESSAGE_BUFFER is returned if there is no following buffer.

Definition at line 185 of file packet-buffer.h.

6.5.2.6   #define emberSetStackBufferLink(*buffer, newLink*)

Sets the buffer following this one in the message. The final buffer in the message has EMBER_NULL_M-ESSAGE_BUFFER as its link.

Parameters

buffer	The buffer whose link is to be set.
newLink	The buffer that is to follow `buffer`.

Definition at line 196 of file packet-buffer.h.

6.5.2.7   #define emberAllocateStackBuffer(   )

Allocates a stack buffer.

Returns

A newly allocated buffer, or EMBER_NULL_MESSAGE_BUFFER if no buffer is available.

Definition at line 205 of file packet-buffer.h.

6.5.3   Function Documentation

6.5.3.1   XAP2B_PAGEZERO_ON int8u∗ emberMessageBufferContents (EmberMessageBuffer *buffer*)

Gets a pointer to a buffer's contents. This pointer can be used to access only the first PACKET_BUFFER-_SIZE bytes in the buffer. To read a message composed of multiple buffers, use emberCopyFromLinked-Buffers().

Parameters

buffer	The buffer whose contents are desired.

Returns

Returns a pointer to the contents of `buffer`.

6.5.3.2 void emberSetMessageBufferLength (EmberMessageBuffer *buffer,* int8u *newLength*)

Sets the length of a buffer.

When asserts are enabled, attempting to set a length greater than the size of the buffer triggers an assert.

Parameters

buffer	A buffer
newLength	The length to set the buffer to.

6.5.3.3 void emberHoldMessageBuffer (EmberMessageBuffer *buffer*)

Holds a message buffer by incrementing its reference count. Implemented as a macro for improved efficiency.

Parameters

buffer	A buffer.

6.5.3.4 void emberReleaseMessageBuffer (EmberMessageBuffer *buffer*)

Releases a message buffer by decrementing its reference count. Implemented as a macro for improved efficiency.

Parameters

buffer	A buffer.

6.5.3.5 EmberMessageBuffer emberAllocateLinkedBuffers (int8u *count*)

Allocates one or more linked buffers.

Parameters

count	The number of buffers to allocate.

Returns

The first buffer in the newly allocated chain of buffers, or EMBER_NULL_MESSAGE_BUFFER if insufficient buffers are available.

6.5.3.6   EmberMessageBuffer emberFillStackBuffer (int16u *count,* ...)

Allocates a stack buffer and fills the buffer with data passed in the function call.

Parameters

count	Buffer length.
...	`count` bytes, which will be placed in the buffer.

Returns

A newly allocated buffer, or EMBER_NULL_MESSAGE_BUFFER if no buffer is available.

6.5.3.7   EmberMessageBuffer emberFillLinkedBuffers (int8u * *contents,* int8u *length*)

Allocates a chain of stack buffers sufficient to hold `length` bytes of data and fills the buffers with the data in contents. If the value of contents is NULL, the buffers are allocated but not filled.

Parameters

contents	A pointer to data to place in the allocated buffers.
length	The buffer length.

Returns

The first buffer in a series of linked stack buffers, or EMBER_NULL_MESSAGE_BUFFER if insufficient buffers are available.

6.5.3.8   void emberCopyToLinkedBuffers (int8u * *contents,* EmberMessageBuffer *buffer,* int8u *startIndex,* int8u *length*)

Copies a specified number of bytes of data into a buffer, starting at a specified index in the buffer array. Buffer links are followed as required. No buffers are allocated or released.

Parameters

contents	A pointer to data to copy into the buffer.
buffer	The buffer to copy data into.
startIndex	The buffer index at which copying should start.
length	The number of bytes of data to copy.

6.5.3.9   void emberCopyFromLinkedBuffers (EmberMessageBuffer *buffer,* int8u *startIndex,* int8u * *contents,* int8u *length*)

Copies `length` bytes of data from a buffer to `contents`, starting at a specified index in the buffer array. Buffer links are followed as required.

Parameters

buffer	The buffer to copy data from.
startIndex	The buffer index at which copying should start.
contents	A pointer to data to copy from the buffer.
length	The number of bytes of data to copy.

6.5.3.10 void emberCopyBufferBytes (EmberMessageBuffer *to,* int16u *toIndex,* EmberMessageBuffer *from,* int16u *fromIndex,* int16u *count*)

Copies a specified number of bytes of data from one buffer into another. Buffer links are followed as required. No buffers are allocated or released.

Parameters

to	The buffer to copy data into.
toIndex	The position in the destination buffer at which copying should start.
from	The buffer to copy data from.
fromIndex	The position in the source buffer at which copying should start.
count	The number of bytes of data to copy.

6.5.3.11 EmberStatus emberAppendToLinkedBuffers (EmberMessageBuffer *buffer,* int8u ∗ *contents,* int8u *length*)

Appends `length` bytes from `contents` onto a buffer. Combines the functionality of ::setPacketBuffersLength() and ::copyToPacketBuffers().

Parameters

buffer	The buffer to append data to.
contents	A pointer to data to append.
length	The number of bytes of data to append.

Returns

EMBER_SUCCESS if sufficient buffers are available, and EMBER_NO_BUFFERS if not.

6.5.3.12 EmberStatus emberAppendPgmToLinkedBuffers (EmberMessageBuffer *buffer,* PGM_P *contents,* int8u *length*)

Appends `length` bytes from `contents`, a pointer into program space (flash) to buffer.

Parameters

buffer	The buffer to append data to.
contents	The data to append.
length	The number of bytes of data to append.

Returns

EMBER_SUCCESS if sufficient buffers are available, and EMBER_NO_BUFFERS if not.

6.5.3.13 EmberStatus emberAppendPgmStringToLinkedBuffers (EmberMessageBuffer *buffer,* PGM_P *suffix*)

Appends a string from program space (flash) to a buffer.

Parameters

buffer	The buffer to append data to.
suffix	The string in program space to append.

Returns

EMBER_SUCCESS if sufficient buffers are available, and EMBER_NO_BUFFERS if not.

6.5.3.14 EmberStatus emberSetLinkedBuffersLength (EmberMessageBuffer *buffer,* int8u *length*)

Sets the length of a chain of buffers, adding or removing trailing buffers as needed.

Parameters

buffer	The buffer whose length is to be set.
length	The length to set.

Returns

EMBER_SUCCESS if changing `buffer's` length by `length` bytes does not require additional buffers or if sufficient buffers are available, and EMBER_NO_BUFFERS if not.

6.5.3.15 int8u∗ emberGetLinkedBuffersPointer (EmberMessageBuffer *buffer,* int8u *index*)

Gets a pointer to a specified byte in a linked list of buffers.

Parameters

buffer	The buffer that the requested byte must come from.
index	The index of the requested byte.

Returns

A pointer to the requested byte.

6.5.3.16 XAP2B_PAGEZERO_ON int8u emberGetLinkedBuffersByte (EmberMessageBuffer *buffer,* int8u *index*)

Gets a specified byte in a linked list of buffers.

Parameters

buffer	The buffer that the requested byte must come from.
index	The index of the requested byte.

Returns

A byte.

6.5.3.17 XAP2B_PAGEZERO_OFF void emberSetLinkedBuffersByte (EmberMessageBuffer *buffer,* int8u *index,* int8u *byte*)

Sets the indexed byte in a linked list of buffers.

Parameters

buffer	The buffer holding the byte to be set.
index	The index of the byte to set.
byte	The value to set the byte to.

6.5.3.18 int16u emberGetLinkedBuffersLowHighInt16u (EmberMessageBuffer *buffer,* int8u *index*)

Gets a little endian 2-byte value from a linked list of buffers.

Parameters

buffer	The buffer containing the 2-byte value.
index	The index of the low byte.

Returns

The 2-byte value.

6.5.3.19 void emberSetLinkedBuffersLowHighInt16u (EmberMessageBuffer *buffer,* int8u *index,* int16u *value*)

Sets a little endian 2-byte value in a linked list of buffers.

Parameters

buffer	The buffer to set the 2-byte value in.
index	The index of the low byte.
value	The 2-byte value to set.

6.5.3.20 int32u emberGetLinkedBuffersLowHighInt32u (EmberMessageBuffer *buffer,* int8u *index*)

Gets a little endian 4-byte value from a linked list of buffers.

Parameters

buffer	The buffer containing the 4-byte value.
index	The index of the low byte.

Returns

The 4-byte value.

6.5.3.21 void emberSetLinkedBuffersLowHighInt32u (EmberMessageBuffer *buffer,* int8u *index,* int32u *value*)

Sets a little endian 4-byte value in a linked list of buffers.

Parameters

buffer	The buffer to set the 2-byte value in.
index	The index of the low byte.
value	The 4-byte value to set.

6.5.3.22 EmberMessageBuffer emberCopyLinkedBuffers (EmberMessageBuffer *buffer*)

Copies a chain of linked buffers.

Parameters

buffer	The first buffer in the chain to copy.

Returns

A newly created copy of the `buffer` chain.

6.5.3.23 EmberMessageBuffer emberMakeUnsharedLinkedBuffer (EmberMessageBuffer *buffer,* boolean *isShared*)

Creates a new, unshared copy of a specified buffer, if that buffer is shared. If it isn't shared, increments the reference count by 1 so that the user of the returned buffer can release it in either case.

Parameters

buffer	The buffer to copy.
isShared	A flag indicating whether the buffer is shared.

Returns

A fresh copy of `buffer` if `isShared` is true, and `buffer` if `isShared` is not true.

6.5.3.24   int8u emberPacketBufferFreeCount (void)

Retrieves the current number of free packet buffers.

Returns

The number of free packet buffers.

6.6 Sending and Receiving Messages

Data Structures

- struct InterPanHeader

  A struct for keeping track of all of the header info.

Macros

- #define EMBER_APSC_MAX_ACK_WAIT_HOPS_MULTIPLIER_MS
- #define EMBER_APSC_MAX_ACK_WAIT_TERMINAL_SECURITY_MS
- #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

- int8u emberMaximumApsPayloadLength (void)
- EmberStatus emberSendMulticast (EmberApsFrame ∗apsFrame, int8u radius, int8u nonmember-Radius, EmberMessageBuffer message)
- EmberStatus emberSendUnicast (EmberOutgoingMessageType type, int16u indexOrDestination, EmberApsFrame ∗apsFrame, EmberMessageBuffer message)
- EmberStatus emberSendBroadcast (EmberNodeId destination, EmberApsFrame ∗apsFrame, int8u radius, EmberMessageBuffer message)
- EmberStatus emberProxyBroadcast (EmberNodeId source, EmberNodeId destination, int8u sequence, EmberApsFrame ∗apsFrame, int8u radius, EmberMessageBuffer message)
- EmberStatus emberSendManyToOneRouteRequest (int16u concentratorType, int8u radius)
- int8u emberAppendSourceRouteHandler (EmberNodeId destination, EmberMessageBuffer header)
- void emberIncomingRouteRecordHandler (EmberNodeId source, EmberEUI64 sourceEui, int8u relay-Count, EmberMessageBuffer header, int8u relayListIndex)
- void emberIncomingManyToOneRouteRequestHandler (EmberNodeId source, EmberEUI64 long-Id, int8u cost)
- void emberIncomingRouteErrorHandler (EmberStatus status, EmberNodeId target)
- EmberStatus emberCancelMessage (EmberMessageBuffer message)
- void emberMessageSentHandler (EmberOutgoingMessageType type, int16u indexOrDestination, EmberApsFrame ∗apsFrame, EmberMessageBuffer message, EmberStatus status)

- void emberIncomingMessageHandler (EmberIncomingMessageType type, EmberApsFrame ∗aps-Frame, EmberMessageBuffer message)
- EmberStatus emberGetLastHopLqi (int8u ∗lastHopLqi)
- EmberStatus emberGetLastHopRssi (int8s ∗lastHopRssi)
- EmberNodeId emberGetSender (void)
- EmberStatus emberGetSenderEui64 (EmberEUI64 senderEui64)
- EmberStatus emberSendReply (int16u clusterId, EmberMessageBuffer reply)
- void emberSetReplyFragmentData (int16u fragmentData)
- boolean emberAddressTableEntryIsActive (int8u addressTableIndex)
- EmberStatus emberSetAddressTableRemoteEui64 (int8u addressTableIndex, EmberEUI64 eui64)
- void emberSetAddressTableRemoteNodeId (int8u addressTableIndex, EmberNodeId id)
- void emberGetAddressTableRemoteEui64 (int8u addressTableIndex, EmberEUI64 eui64)
- EmberNodeId emberGetAddressTableRemoteNodeId (int8u addressTableIndex)
- void emberSetExtendedTimeout (EmberEUI64 remoteEui64, boolean extendedTimeout)
- boolean emberGetExtendedTimeout (EmberEUI64 remoteEui64)
- void emberIdConflictHandler (EmberNodeId conflictingId)
- boolean emberPendingAckedMessages (void)
- EmberMessageBuffer makeInterPanMessage (InterPanHeader ∗headerData, EmberMessageBuffer payload)
- int8u parseInterPanMessage (EmberMessageBuffer message, int8u startOffset, InterPanHeader ∗header-Data)
- int8u makeInterPanMessage (InterPanHeader ∗headerData, int8u ∗message, int8u maxLength, int8u ∗payload, int8u payloadLength)
- int8u parseInterPanMessage (int8u ∗message, int8u messageLength, InterPanHeader ∗headerData)

Variables

- int16u emberApsAckTimeoutMs
- EmberMulticastTableEntry ∗ emberMulticastTable
- int8u emberMulticastTableSize

6.6.1 Detailed Description

See message.h for source code.

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

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

6.6.2 Macro Definition Documentation

6.6.2.1 #define EMBER_APSC_MAX_ACK_WAIT_HOPS_MULTIPLIER_MS

The per-hop delay allowed for in the calculation of the APS ACK timeout value. This is defined in the ZigBee specification. This times the maximum number of hops (EMBER_MAX_HOPS) plus the terminal encrypt/decrypt time is the timeout between retries of an APS acked message, in milliseconds.

Definition at line 32 of file message.h.

6.6.2.2   #define EMBER_APSC_MAX_ACK_WAIT_TERMINAL_SECURITY_MS

The terminal encrypt/decrypt time allowed for in the calculation of the APS ACK timeout value. This is defined in the ZigBee specification.

Definition at line 37 of file message.h.

6.6.2.3   #define INTER_PAN_UNICAST

Definition at line 29 of file ami-inter-pan.h.

6.6.2.4   #define INTER_PAN_BROADCAST

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

6.6.2.5   #define INTER_PAN_MULTICAST

Definition at line 31 of file ami-inter-pan.h.

6.6.2.6   #define MAX_INTER_PAN_MAC_SIZE

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

6.6.2.7   #define STUB_NWK_SIZE

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

6.6.2.8   #define STUB_NWK_FRAME_CONTROL

Definition at line 39 of file ami-inter-pan.h.

6.6.2.9   #define MAX_STUB_APS_SIZE

Definition at line 42 of file ami-inter-pan.h.

6.6.2.10   #define MAX_INTER_PAN_HEADER_SIZE

Definition at line 45 of file ami-inter-pan.h.

6.6.2.11   #define INTER_PAN_UNICAST

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

Definition at line 28 of file ami-inter-pan-host.h.

6.6.2.12 #define INTER_PAN_BROADCAST

Definition at line 29 of file ami-inter-pan-host.h.

6.6.2.13 #define INTER_PAN_MULTICAST

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

6.6.2.14 #define MAX_INTER_PAN_MAC_SIZE

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

6.6.2.15 #define STUB_NWK_SIZE

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

6.6.2.16 #define STUB_NWK_FRAME_CONTROL

Definition at line 39 of file ami-inter-pan-host.h.

6.6.2.17 #define MAX_STUB_APS_SIZE

Definition at line 42 of file ami-inter-pan-host.h.

6.6.2.18 #define MAX_INTER_PAN_HEADER_SIZE

Definition at line 45 of file ami-inter-pan-host.h.

6.6.3 Function Documentation

6.6.3.1 int8u emberMaximumApsPayloadLength (void)

Returns the maximum size of the payload that the Application Support sub-layer will accept.

The size depends on the security level in use. The value is the same as that found in the node descriptor.

Returns

The maximum APS payload length.

6.6.3.2 EmberStatus emberSendMulticast (EmberApsFrame ∗ *apsFrame,* int8u *radius,* int8u *nonmemberRadius,* EmberMessageBuffer *message*)

Sends a multicast message to all endpoints that share a specific multicast ID and are within a specified number of hops of the sender.

Parameters

apsFrame	The APS frame for the message. The multicast will be sent to the groupId in this frame.
radius	The message will be delivered to all nodes within this number of hops of the sender. A value of zero is converted to EMBER_MAX_HOPS.
nonmember-Radius	The number of hops that the message will be forwarded by devices that are not members of the group. A value of 7 or greater is treated as infinite.
message	A message.

Returns

An EmberStatus value. For any result other than EMBER_SUCCESS, the message will not be sent.

- EMBER_SUCCESS - The message has been submitted for transmission.
- EMBER_INVALID_BINDING_INDEX - The `bindingTableIndex` refers to a non-multicast binding.
- EMBER_NETWORK_DOWN - The node is not part of a network.
- EMBER_MESSAGE_TOO_LONG - The message is too large to fit in a MAC layer frame.
- EMBER_NO_BUFFERS - The free packet buffer pool is empty.
- EMBER_NETWORK_BUSY - Insufficient resources available in Network or MAC layers to send message.

6.6.3.3   EmberStatus emberSendUnicast (EmberOutgoingMessageType *type,* int16u *indexOrDestination,* EmberApsFrame ∗ *apsFrame,* EmberMessageBuffer *message*)

Sends a unicast message as per the ZigBee specification.

The message will arrive at its destination only if there is a known route to the destination node. Setting the ::ENABLE_ROUTE_DISCOVERY option will cause a route to be discovered if none is known. Setting the ::FORCE_ROUTE_DISCOVERY option will force route discovery. Routes to end-device children of the local node are always known.

Setting the APS_RETRY option will cause the message to be retransmitted until either a matching acknowledgment is received or three transmissions have been made.

Note

Using the ::FORCE_ROUTE_DISCOVERY option will cause the first transmission to be consumed by a route request as part of discovery, so the application payload of this packet will not reach its destination on the first attempt. If you want the packet to reach its destination, the APS_RETRY option must be set so that another attempt is made to transmit the message with its application payload after the route has been constructed.

Setting the ::DESTINATION_EUI64 option will cause the long ID of the destination to be included in the network header. This is the only way to absolutely guarantee that the message is delivered to the correct node. Without it, a message may on occasion be delivered to the wrong destination in the event of an id conflict that has not yet been detected and resolved by the network layer.

Note

When sending fragmented messages, the stack will only assign a new APS sequence number for the first fragment of the message (i.e., EMBER_APS_OPTION_FRAGMENT is set and the low-order byte of the groupId field in the APS frame is zero). For all subsequent fragments of the same message, the application must set the sequence number field in the APS frame to the sequence number assigned by the stack to the first fragment.

Parameters

type	Specifies the outgoing message type. Must be one of EMBER_OUTGOING_DIRE-CT, EMBER_OUTGOING_VIA_ADDRESS_TABLE, or EMBER_OUTGOING_VIA_BINDING.
indexOr-Destination	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 which is to be added to the message.
message	Contents of the message.

Returns

An EmberStatus value. For any result other than EMBER_SUCCESS, the message will not be sent.

- EMBER_SUCCESS - The message has been submitted for transmission.
- EMBER_INVALID_BINDING_INDEX - The `bindingTableIndex` refers to a non-unicast binding.
- EMBER_NETWORK_DOWN - The node is not part of a network.
- EMBER_MESSAGE_TOO_LONG - The message is too large to fit in a MAC layer frame.
- EMBER_MAX_MESSAGE_LIMIT_REACHED - The EMBER_APS_UNICAST_MESSAGE_COUNT limit has been reached.

6.6.3.4   EmberStatus emberSendBroadcast (EmberNodeId *destination,* EmberApsFrame ∗ *apsFrame,* int8u *radius,* EmberMessageBuffer *message*)

Sends a broadcast message as per the ZigBee specification.

The message will be delivered to all nodes within `radius` hops of the sender. A radius of zero is converted to EMBER_MAX_HOPS.

Parameters

destination	The destination to which to send the broadcast. This must be one of three ZigBee broadcast addresses.
apsFrame	The APS frame data to be included in the message.
radius	The maximum number of hops the message will be relayed.
message	The actual message to be sent.

Returns

An EmberStatus value.

6.6.3.5   EmberStatus emberProxyBroadcast (EmberNodeId *source,* EmberNodeId *destination,* int8u *sequence,* EmberApsFrame ∗ *apsFrame,* int8u *radius,* EmberMessageBuffer *message*)

Proxies a broadcast message for another node.

The message will be delivered to all nodes within `radius` hops of the local node. A radius of zero is converted to EMBER_MAX_HOPS.

Parameters

source	The source from which to send the broadcast.
destination	The destination to which to send the broadcast. This must be one of three ZigBee broadcast addresses.
sequence	The NWK sequence number for the message.
apsFrame	The APS frame data to be included in the message.
radius	The maximum number of hops the message will be relayed.
message	The actual message to be sent.

Returns

An EmberStatus value.

6.6.3.6 EmberStatus emberSendManyToOneRouteRequest (int16u *concentratorType,* int8u *radius*)

Sends a route request packet that creates routes from every node in the network back to this node.

This function should be called by an application that wishes to communicate with many nodes, for example, a gateway, central monitor, or controller. A device using this function was referred to as an "aggregator" in EmberZNet 2.x and earlier, and is referred to as a "concentrator" in the ZigBee specification and EmberZ-Net 3.

This function enables large scale networks, because the other devices do not have to individually perform bandwidth-intensive route discoveries. Instead, when a remote node sends an APS unicast to a concentrator, its network layer automatically delivers a special route record packet first, which lists the network ids of all the intermediate relays. The concentrator can then use source routing to send outbound APS unicasts. (A source routed message is one in which the entire route is listed in the network layer header.) This allows the concentrator to communicate with thousands of devices without requiring large route tables on neighboring nodes.

This function is only available in ZigBee Pro (stack profile 2), and cannot be called on end devices. Any router can be a concentrator (not just the coordinator), and there can be multiple concentrators on a network.

Note that a concentrator does not automatically obtain routes to all network nodes after calling this function. Remote applications must first initiate an inbound APS unicast.

Many-to-one routes are not repaired automatically. Instead, the concentrator application must call this function to rediscover the routes as necessary, for example, upon failure of a retried APS message. The reason for this is that there is no scalable one-size-fits-all route repair strategy. A common and recommended strategy is for the concentrator application to refresh the routes by calling this function periodically.

Parameters

concentrator-Type	Must be either EMBER_HIGH_RAM_CONCENTRATOR or EMBER_LOW_RAM-_CONCENTRATOR. The former is used when the caller has enough memory to store source routes for the whole network. In that case, remote nodes stop sending route records once the concentrator has successfully received one. The latter is used when the concentrator has insufficient RAM to store all outbound source routes. In that case, route records are sent to the concentrator prior to every inbound APS unicast.
radius	The maximum number of hops the route request will be relayed. A radius of zero is converted to EMBER_MAX_HOPS.

Returns

> EMBER_SUCCESS if the route request was successfully submitted to the transmit queue, and EMB-ER_ERR_FATAL otherwise.

6.6.3.7 int8u emberAppendSourceRouteHandler (EmberNodeId *destination,* EmberMessageBuffer *header*)

The application can implement this callback to supply source routes to outgoing messages.

The application must define :EMBER_APPLICATION_HAS_SOURCE_ROUTING in its configuration header to use this. The application uses the supplied destination to look up a source route. If available, the application appends the source route to the supplied header using the proper frame format, as described in section 3.4.1.9 "Source Route Subframe Field" of the ZigBee specification. If a source route is appended, the stack takes care of setting the proper flag in the network frame control field. See app/util/source-route.c for a sample implementation.

If header is :EMBER_NULL_MESSAGE_BUFFER the only action is to return the size of the source route frame needed to the destination.

Parameters

destination	The network destination of the message.
header	The message buffer containing the partially complete packet header. The application appends the source route frame to this header.

Returns

> The size in bytes of the source route frame, or zero if there is not one available.

6.6.3.8 void emberIncomingRouteRecordHandler (EmberNodeId *source,* EmberEUI64 *sourceEui,* int8u *relayCount,* EmberMessageBuffer *header,* int8u *relayListIndex*)

Reports the arrival of a route record command frame to the application.

The route record command frame lists the short IDs of the relays that were used along the route from the source to us. This information is used by aggregators to be able to initiate source routed messages. The application must define EMBER_APPLICATION_HAS_SOURCE_ROUTING in its configuration header to use this.

Parameters

source	The id of the node that initiated the route record.
sourceEui	The EUI64 of the node that initiated the route record.
relayCount	The number of relays in the list.
header	The message buffer containing the route record frame.
relayListIndex	The starting index of the relay list. The relay closest to the source is listed first, and the relay closest to us is listed last. Short ids are stored low byte first. Be careful to use buffer-boundary-safe APIs to read the list.

6.6.3.9 void emberIncomingManyToOneRouteRequestHandler (EmberNodeId *source,* EmberEUI64 *longId,* int8u *cost*)

A callback indicating that a many-to-one route to the concentrator with the given short and long id is available for use.

The application must define `EMBER_APPLICATION_HAS_INCOMING_MANY_TO_ONE_ROUTE_RE-QUEST_HANDLER` in its configuration header to use this.

Parameters

source	The short id of the concentrator that initiated the many-to-one route request.
longId	The EUI64 of the concentrator.
cost	The path cost to the concentrator.

6.6.3.10 void emberIncomingRouteErrorHandler (EmberStatus *status,* EmberNodeId *target*)

A callback invoked when a route error message is received.

A status of EMBER_SOURCE_ROUTE_FAILURE indicates that a source-routed unicast sent from this node encountered a broken link. Note that this case occurs only if this node is a concentrator using many-to-one routing for inbound messages and source-routing for outbound messages. The node prior to the broken link generated the route error message and returned it to us along the many-to-one route.

A status of EMBER_MANY_TO_ONE_ROUTE_FAILURE also occurs only if we are a concentrator, and indicates that a unicast sent to us along a many-to-one route encountered a broken link. The node prior to the broken link generated the route error message and forwarded it to us via a randomly chosen neighbor, taking advantage of the many-to-one nature of the route.

A status of EMBER_MAC_INDIRECT_TIMEOUT indicates that a message sent to the target end device could not be delivered by the parent because the indirect transaction timer expired. Upon receipt of the route error, the stack sets the extended timeout for the target node in the address table, if present. It then calls this handler to indicate receipt of the error.

Note that if the original unicast data message is sent using the EMBER_APS_OPTION_RETRY option, a new route error message is generated for each failed retry. Thus it is not unusual to receive three route error messages in succession for a single failed retried APS unicast. On the other hand, it is also not guaranteed that any route error messages will be delivered successfully at all. The only sure way to detect a route failure is to use retried APS messages and to check the status of the emberMessageSentHandler().

If the application includes this callback, it must define `EMBER_APPLICATION_HAS_INCOMING_RO-UTE_ERROR_HANDLER` in its configuration header.

Parameters

status	EMBER_SOURCE_ROUTE_FAILURE, EMBER_MANY_TO_ONE_ROUTE_FAI-LURE, EMBER_MAC_INDIRECT_TIMEOUT
target	The short id of the remote node.

6.6.3.11 EmberStatus emberCancelMessage (EmberMessageBuffer *message*)

DEPRECATED.

Parameters

message	A message.

Returns

Always returns EMBER_SUCCESS.

6.6.3.12 **void emberMessageSentHandler (EmberOutgoingMessageType *type*, int16u *indexOrDestination*, EmberApsFrame ∗ *apsFrame*, EmberMessageBuffer *message*, EmberStatus *status*)**

A callback invoked by the stack when it has completed sending a message.

Parameters

type	The type of message sent.
indexOr-Destination	The destination to which the message was sent, for direct unicasts, or the address table or binding index for other unicasts. The value is unspecified for multicasts and broadcasts.
apsFrame	The APS frame for the message.
message	The message that was sent.
status	An EmberStatus value of EMBER_SUCCESS if an ACK was received from the destination or EMBER_DELIVERY_FAILED if no ACK was received.

6.6.3.13 **void emberIncomingMessageHandler (EmberIncomingMessageType *type*, EmberApsFrame ∗ *apsFrame*, EmberMessageBuffer *message*)**

A callback invoked by the EmberZNet stack when a message is received.

The following functions may be called from emberIncomingMessageHandler():

- emberGetLastHopLqi()

- emberGetLastHopRssi()

- emberGetSender()

- emberGetSenderEui64()

- emberGetBindingIndex()

- emberSendReply() (for incoming APS retried unicasts only)

- emberSetReplyBinding()

- emberNoteSendersBinding()

Parameters

type	The type of the incoming message. One of the following: • EMBER_INCOMING_UNICAST • EMBER_INCOMING_UNICAST_REPLY • EMBER_INCOMING_MULTICAST • EMBER_INCOMING_MULTICAST_LOOPBACK • EMBER_INCOMING_BROADCAST • EMBER_INCOMING_BROADCAST_LOOPBACK
apsFrame	The APS frame from the incoming message.
message	The message that was sent.

6.6.3.14   EmberStatus emberGetLastHopLqi (int8u ∗ *lastHopLqi*)

Gets the link quality from the node that last relayed the current message.

Note

This function may only be called from within

- emberIncomingMessageHandler()
- emberNetworkFoundHandler()
- emberIncomingRouteRecordHandler()
- ::emberMacPassthroughMessageHandler()
- emberIncomingBootloadMessageHandler()

When this function is called from within one of these handler functions the link quality reported corresponds to the header being processed in that hander function. If this function is called outside of these handler functions the link quality reported will correspond to a message that was processed earlier.

This function is not available from within emberPollHandler() or emberPollCompleteHandler(). The link quality information of interest during the emberPollHandler() is from the data request packet itself. This message must be handled quickly due to strict 15.4 timing requirements, and the link quality information is not recorded by the stack. The link quality information of interest during the emberPollCompleteHandler() is from the ACK to the data request packet. The ACK is handled by the hardware and the link quality information does not make it up to the stack.

Parameters

lastHopLqi	The link quality for the last incoming message processed.

Returns

This function always returns EMBER_SUCCESS. It is not necessary to check this return value.

6.6.3.15   EmberStatus emberGetLastHopRssi (int8s ∗ *lastHopRssi*)

Gets the receive signal strength indication (RSSI) for the current message.

After a successful call to this function, the quantity referenced by lastHopRssi will contain the energy level (in units of dBm) observed during the last packet received.

Note

This function may only be called from within:

- emberIncomingMessageHandler()
- emberNetworkFoundHandler()
- emberIncomingRouteRecordHandler()
- ::emberMacPassthroughMessageHandler()
- emberIncomingBootloadMessageHandler()

When this function is called from within one of these handler functions the RSSI reported corresponds to the header being processed in that hander function. If this function is called outside of these handler functions the RSSI reported will correspond to a message that was processed earlier.

This function is not available from within emberPollHandler() or emberPollCompleteHandler(). The RSSI information of interest during the emberPollHandler() is from the data request packet itself. This message must be handled quickly due to strict 15.4 timing requirements, and the RSSI information is not recorded by the stack. The RSSI information of interest during the emberPollCompleteHandler() is from the ACK to the data request packet. The ACK is handled by the hardware and the RSSI information does not make it up to the stack.

Parameters

lastHopRssi	The RSSI for the last incoming message processed.

Returns

This function always returns EMBER_SUCCESS. It is not necessary to check this return value.

6.6.3.16   EmberNodeId emberGetSender (void   )

Returns the node ID of the sender of the current incoming message.

Note

This function can be called only from within emberIncomingMessageHandler().

Returns

The sender of the current incoming message.

6.6.3.17   EmberStatus emberGetSenderEui64 (EmberEUI64 *senderEui64*)

Returns the EUI64 of the sender of the current incoming message, if the sender chose to include this information in the message. The EMBER_APS_OPTION_SOURCE_EUI64 bit in the options field of the APS frame of the incoming message indicates that the EUI64 is present in the message.

Note

This function can be called only from within emberIncomingMessageHandler().

Parameters

senderEui64	The EUI64 of the sender.

Returns

An EmberStatus value:

- EMBER_SUCCESS - senderEui64 has been set to the EUI64 of the sender of the current incoming message.
- EMBER_INVALID_CALL - Either:
  1. This function was called outside of the context of the emberIncomingMessageHandler() callback
  2. It was called in the context of emberIncomingMessageHandler() but the incoming message did not include the EUI64 of the sender.

6.6.3.18   EmberStatus emberSendReply (int16u *clusterId,* EmberMessageBuffer *reply*)

Sends a reply for an application that has received a unicast message.

The reply will be included with the ACK that the stack automatically sends back.

Note

This function may be called only from within emberIncomingMessageHandler().

Parameters

clusterId	The cluster ID to use for the reply.
reply	A reply message.

Returns

An EmberStatus value. For any result other than EMBER_SUCCESS, the message will not be sent.

- EMBER_SUCCESS - The message has been submitted for transmission.
- EMBER_INVALID_CALL - Either:
  1. This function was called outside of the context of the emberIncomingMessageHandler() callback
  2. It was called in the context of emberIncomingMessageHandler() but the incoming message was not a unicast
  3. It was called more than once in the context of emberIncomingMessageHandler().
- EMBER_NETWORK_BUSY - Either:
  1. No route available.
  2. Insufficient resources available in Network or MAC layers to send message.

6.6.3.19 void emberSetReplyFragmentData (int16u *fragmentData*)

Sets the fragment data to be used when sending a reply to a unicast message.

Note

> This function may be called only from within emberIncomingMessageHandler().

Parameters

fragmentData	The low byte is the block number of the reply. The high byte is the ack bitfield of the reply.

6.6.3.20 boolean emberAddressTableEntryIsActive (int8u *addressTableIndex*)

Indicates whether any messages are currently being sent using this address table entry.

Note that this function does not indicate whether the address table entry is unused. To determine whether an address table entry is unused, check the remote node ID. The remote node ID will have the value EMB-ER_TABLE_ENTRY_UNUSED_NODE_ID when the address table entry is not in use.

Parameters

addressTable-Index	The index of an address table entry.

Returns

> TRUE if the address table entry is active, FALSE otherwise.

6.6.3.21 EmberStatus emberSetAddressTableRemoteEui64 (int8u *addressTableIndex,* EmberEUI64 *eui64*)

Sets the EUI64 of an address table entry.

This function will also check other address table entries, the child table and the neighbor table to see if the node ID for the given EUI64 is already known. If known then this function will also set the node ID. If not known it will set the node ID to EMBER_UNKNOWN_NODE_ID.

Parameters

addressTable-Index	The index of an address table entry.
eui64	The EUI64 to use for the address table entry.

Returns

> EMBER_SUCCESS if the EUI64 was successfully set, and EMBER_ADDRESS_TABLE_ENTRY-_IS_ACTIVE otherwise.

6.6.3.22   void emberSetAddressTableRemoteNodeId (int8u *addressTableIndex,* EmberNodeId *id*)

Sets the short ID of an address table entry.

Usually the application will not need to set the short ID in the address table. Once the remote EUI64 is set the stack is capable of figuring out the short ID on its own. However, in cases where the application does set the short ID, the application must set the remote EUI64 prior to setting the short ID.

Parameters

addressTable-Index	The index of an address table entry.
id	The short ID corresponding to the remote node whose EUI64 is stored in the address table at the given index or EMBER_TABLE_ENTRY_UNUSED_NODE_ID which indicates that the entry stored in the address table at the given index is not in use.

6.6.3.23   void emberGetAddressTableRemoteEui64 (int8u *addressTableIndex,* EmberEUI64 *eui64*)

Gets the EUI64 of an address table entry.

Parameters

addressTable-Index	The index of an address table entry.
eui64	The EUI64 of the address table entry is copied to this location.

6.6.3.24   EmberNodeId emberGetAddressTableRemoteNodeId (int8u *addressTableIndex*)

Gets the short ID of an address table entry.

Parameters

addressTable-Index	The index of an address table entry.

Returns

One of the following:

- The short ID corresponding to the remote node whose EUI64 is stored in the address table at the given index.
- EMBER_UNKNOWN_NODE_ID - Indicates that the EUI64 stored in the address table at the given index is valid but the short ID is currently unknown.
- EMBER_DISCOVERY_ACTIVE_NODE_ID - Indicates that the EUI64 stored in the address table at the given location is valid and network address discovery is underway.
- EMBER_TABLE_ENTRY_UNUSED_NODE_ID - Indicates that the entry stored in the address table at the given index is not in use.

6.6.3.25   void emberSetExtendedTimeout (EmberEUI64 *remoteEui64,* boolean *extendedTimeout*)

Tells the stack whether or not the normal interval between retransmissions of a retried unicast message should be increased by EMBER_INDIRECT_TRANSMISSION_TIMEOUT.

The interval needs to be increased when sending to a sleepy node so that the message is not retransmitted until the destination has had time to wake up and poll its parent. The stack will automatically extend the timeout:

- For our own sleepy children.

- When an address response is received from a parent on behalf of its child.

- When an indirect transaction expiry route error is received.

- When an end device announcement is received from a sleepy node.

Parameters

remoteEui64	The address of the node for which the timeout is to be set.
extended-Timeout	TRUE if the retry interval should be increased by EMBER_INDIRECT_TRANSMIS-SION_TIMEOUT. FALSE if the normal retry interval should be used.

6.6.3.26 boolean emberGetExtendedTimeout (EmberEUI64 *remoteEui64*)

Indicates whether or not the stack will extend the normal interval between retransmissions of a retried unicast message by EMBER_INDIRECT_TRANSMISSION_TIMEOUT.

Parameters

remoteEui64	The address of the node for which the timeout is to be returned.

Returns

TRUE if the retry interval will be increased by EMBER_INDIRECT_TRANSMISSION_TIMEOUT and FALSE if the normal retry interval will be used.

6.6.3.27 void emberIdConflictHandler (EmberNodeId *conflictingId*)

A callback invoked by the EmberZNet stack when an ID conflict is discovered, that is, two different nodes in the network were found to be using the same short ID.

The stack automatically removes the conflicting short ID from its internal tables (address, binding, route, neighbor, and child tables). The application should discontinue any other use of the ID. If the application includes this callback, it must define ::EMBER_APPLICATION_HAS_ID_CONFLICT_HANDLER in its configuration header.

Parameters

conflictingId	The short ID for which a conflict was detected.

6.6.3.28 boolean emberPendingAckedMessages (void)

Indicates whether there are pending messages in the APS retry queue.

Returns

TRUE if there is at least a pending message belonging to the current network in the APS retry queue, FALSE otherwise.

6.6.3.29 EmberMessageBuffer makeInterPanMessage (InterPanHeader ∗ *headerData,* EmberMessageBuffer *payload*)

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

6.6.3.30 int8u parseInterPanMessage (EmberMessageBuffer *message,* int8u *startOffset,* InterPanHeader ∗ *headerData*)

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

6.6.3.31 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.

6.6.3.32 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.

6.6.4 Variable Documentation

6.6.4.1 int16u emberApsAckTimeoutMs

The APS ACK timeout value. The stack waits this amount of time between resends of APS retried messages. The default value is:

```
((EMBER_APSC_MAX_ACK_WAIT_HOPS_MULTIPLIER_MS
  * EMBER_MAX_HOPS)
 + EMBER_APSC_MAX_ACK_WAIT_TERMINAL_SECURITY_MS)
```

6.6.4.2 EmberMulticastTableEntry∗ emberMulticastTable

The multicast table.

Each entry contains a multicast ID and an endpoint, indicating that the endpoint is a member of the multicast group. Only devices with an endpoint in a multicast group will receive messages sent to that multicast group.

Entries with with an endpoint of 0 are ignored (the ZDO does not a member of any multicast groups). All endpoints are initialized to 0 on startup.

6.6.4.3   int8u emberMulticastTableSize

The number of entries in the multicast table.

6.7 End Devices

Functions

- EmberNodeId emberChildId (int8u childIndex)
- int8u emberChildIndex (EmberNodeId childId)
- EmberStatus emberGetChildData (int8u index, EmberEUI64 childEui64Return, EmberNodeType
  ∗childTypeReturn)
- void emberChildJoinHandler (int8u index, boolean joining)
- EmberStatus emberPollForData (void)
- void emberPollCompleteHandler (EmberStatus status)
- EmberStatus emberSetMessageFlag (EmberNodeId childId)
- EmberStatus emberClearMessageFlag (EmberNodeId childId)
- void emberPollHandler (EmberNodeId childId, boolean transmitExpected)
- int8u emberChildCount (void)
- int8u emberRouterChildCount (void)
- int8u emberMaxChildCount (void)
- int8u emberMaxRouterChildCount (void)
- EmberNodeId emberGetParentNodeId (void)
- EmberEUI64 emberGetParentEui64 (void)

Power Management

- enum {
  EMBER_OUTGOING_MESSAGES, EMBER_INCOMING_MESSAGES, EMBER_RADIO_IS_-
  ON, EMBER_TRANSPORT_ACTIVE,
  EMBER_APS_LAYER_ACTIVE, EMBER_ASSOCIATING, EMBER_ZLL_TOUCH_LINKING
  }
- int16u emberCurrentStackTasks (void)
- boolean emberOkToNap (void)
- boolean emberOkToHibernate (void)
- boolean emberOkToLongPoll (void)
- void emberStackPowerDown (void)
- void emberStackPowerUp (void)
- #define EMBER_HIGH_PRIORITY_TASKS

6.7.1 Detailed Description

EmberZNet API relating to end device children. See child.h for source code.

6.7.2 Macro Definition Documentation

6.7.2.1 #define EMBER_HIGH_PRIORITY_TASKS

A mask of the tasks that prevent a device from sleeping.

Definition at line 256 of file child.h.

6.7.3 Enumeration Type Documentation

6.7.3.1 anonymous enum

Defines tasks that prevent the stack from sleeping.

Enumerator:

    EMBER_OUTGOING_MESSAGES There are messages waiting for transmission.

    EMBER_INCOMING_MESSAGES One or more incoming messages are being processed.

    EMBER_RADIO_IS_ON The radio is currently powered on. On sleepy devices the radio is turned off when not in use. On non-sleepy devices (EMBER_COORDINATOR, EMBER_ROUTER, or EMBER_END_DEVICE) the radio is always on.

    EMBER_TRANSPORT_ACTIVE The transport layer has messages awaiting an ACK.

    EMBER_APS_LAYER_ACTIVE The ZigBee APS layer has messages awaiting an ACK.

    EMBER_ASSOCIATING The node is currently trying to associate with a network.

    EMBER_ZLL_TOUCH_LINKING The node is currently touch linking.

Definition at line 233 of file child.h.

6.7.4 Function Documentation

6.7.4.1 EmberNodeId emberChildId (int8u *childIndex*)

Converts a child index to a node ID.

Parameters

childIndex	The index.

Returns

The node ID of the child or EMBER_NULL_NODE_ID if there isn't a child at the childIndex specified.

6.7.4.2 int8u emberChildIndex (EmberNodeId *childId*)

Converts a node ID to a child index.

Parameters

childId	The node ID of the child.

Returns

The child index or 0xFF if the node ID does not belong to a child.

6.7.4.3 EmberStatus emberGetChildData (int8u *index,* EmberEUI64 *childEui64Return,* EmberNodeType * *childTypeReturn*)

If there is a child at 'index' this copies its EUI64 and node type into the return variables and returns EM-BER_SUCCESS. If there is no child at 'index' it returns EMBER_NOT_JOINED. Possible child indexes run from zero to emberMaxChildCount() - 1.

Parameters

index	The index of the child of interest.
childEui64-Return	The child's EUI64 is copied into here.
childType-Return	The child's node type is copied into here.

Returns

Returns EMBER_SUCCESS if a child is found at that index, EMBER_NOT_JOINED if not.

6.7.4.4 void emberChildJoinHandler (int8u *index,* boolean *joining*)

This is called by the stack when a child joins or leaves. 'Joining' is TRUE if the child is joining and FALSE if leaving.

The index is the same as the value that should be passed to emberGetChildData() to get this child's data. Note that if the child is leaving emberGetChildData() will now return EMBER_NOT_JOINED if called with this index. If the application includes emberChildJoinHandler(), it must define EMBER_APPLICA-TION_HAS_CHILD_JOIN_HANDLER in its CONFIGURATION_HEADER

Parameters

index	The index of the child of interest.
joining	True if the child is joining, false if the child is leaving.

6.7.4.5 EmberStatus emberPollForData (void)

Function to request any pending data from the parent node. This function allows an end device to query its parent for any pending data.

End devices must call this function periodically to maintain contact with their parent. The parent will remove a mobile end device from its child table if it has not received a poll from it within the last EM-BER_MOBILE_NODE_POLL_TIMEOUT quarter seconds. It will remove a sleepy or non-sleepy end device if it has not received a poll from it within the last EMBER_END_DEVICE_POLL_TIMEOUT << EMBER_END_DEVICE_POLL_TIMEOUT_SHIFT seconds.

If the end device has lost contact with its parent, emberPollForData() calls ::emberRejoinNetwork(TRUE) and returns EMBER_ERR_FATAL.

The default values for the timeouts are set in config/ember-configuration-defaults.h, and can be overridden in the application's configuration header.

Returns

An EmberStatus value:

- EMBER_SUCCESS - The poll message has been submitted for transmission.
- EMBER_INVALID_CALL - The node is not an end device.
- EMBER_NOT_JOINED - The node is not part of a network.
- EMBER_MAC_TRANSMIT_QUEUE_FULL - The MAC layer transmit queue is full.
- EMBER_NO_BUFFERS - There were no buffers available to create the data poll message.
- EMBER_ERR_FATAL - Too much time has elapsed since the last successful poll. A rejoin attempt has been initiated. This error is not "fatal". The command can be retried until successful.

6.7.4.6 void emberPollCompleteHandler (EmberStatus *status*)

@ brief This is called by the stack when a data poll to the parent is complete.

If the application includes emberPollCompleteHandler(), it must define EMBER_APPLICATION_HAS_-POLL_COMPLETE_HANDLER within its CONFIGURATION_HEADER

Parameters

status	An `EmberStatus` value: • EMBER_SUCCESS - Data was received in response to the poll. • EMBER_MAC_NO_DATA - No data was pending. • EMBER_DELIVERY_FAILED - The poll message could not be sent. • EMBER_MAC_NO_ACK_RECEIVED - The poll message was sent but not acknowledged by the parent.

6.7.4.7 EmberStatus emberSetMessageFlag (EmberNodeId *childId*)

Sets a flag to indicate that there is a message pending for a child. The next time that the child polls, it will be informed that it has a pending message. The message is sent from emberPollHandler, which is called when the child requests the data.

Parameters

childId	The ID of the child that just polled for data.

Returns

An EmberStatus value.

- EMBER_SUCCESS - The next time that the child polls, it will be informed that it has pending data.
- EMBER_NOT_JOINED - The child identified by childId is not our child (it is not in the PAN).

6.7.4.8 EmberStatus emberClearMessageFlag (EmberNodeId *childId*)

Clears a flag to indicate that there are no more messages for a child. The next time the child polls, it will be informed that it does not have any pending messages.

Parameters

childId	The ID of the child that no longer has pending messages.

Returns

An EmberStatus value.

- EMBER_SUCCESS - The next time that the child polls, it will be informed that it does not have any pending messages.
- EMBER_NOT_JOINED - The child identified by childId is not our child (it is not in the PAN).

6.7.4.9 void emberPollHandler (EmberNodeId *childId,* boolean *transmitExpected*)

A callback that allows the application to send a message in response to a poll from a child.

This function is called when a child polls, provided that the pending message flag is set for that child (see emberSetMessageFlag(). The message should be sent to the child using emberSendUnicast() with the EMBER_APS_OPTION_POLL_RESPONSE option.

If the application includes ::emberPollHanlder(), it must define EMBER_APPLICATION_HAS_POLL_-HANDLER in its CONFIGURATION_HEADER.

Parameters

childId	The ID of the child that is requesting data.
transmit-Expected	TRUE if the child is expecting an application- supplied data message. FALSE otherwise.

6.7.4.10 int8u emberChildCount (void)

Returns the number of children the node currently has.

Returns

number of children

6.7.4.11 int8u emberRouterChildCount (void)

Returns the number of router children that the node currently has.

Returns

number of router children

6.7.4.12 int8u emberMaxChildCount (void)

Returns the maximum number of children for this node. The return value is undefined for nodes that are not joined to a network.

Returns

maximum number of children

6.7.4.13 int8u emberMaxRouterChildCount (void)

Returns the maximum number of router children for this node. The return value is undefined for nodes that are not joined to a network.

Returns

maximum number of router children

6.7.4.14 EmberNodeId emberGetParentNodeId (void)

Returns the parent's node ID. The return value is undefined for nodes without parents (coordinators and nodes that are not joined to a network).

Returns

parent's node ID

6.7.4.15 EmberEUI64 emberGetParentEui64 (void)

Returns the parent's EUI64. The return value is undefined for nodes without parents (coordinators and nodes that are not joined to a network).

Returns

parent's EUI64

6.7.4.16 int16u emberCurrentStackTasks (void)

Returns a bitmask indicating the stack's current tasks.

The mask EMBER_HIGH_PRIORITY_TASKS defines which tasks are high priority. Devices should not sleep if any high priority tasks are active. Active tasks that are not high priority are waiting for messages to arrive from other devices. If there are active tasks, but no high priority ones, the device may sleep but should periodically wake up and call emberPollForData() in order to receive messages. Parents will hold messages for EMBER_INDIRECT_TRANSMISSION_TIMEOUT milliseconds before discarding them.

Returns

A bitmask of the stack's active tasks.

6.7.4.17   boolean emberOkToNap (void)

Indicates whether the stack is currently in a state where there are no high priority tasks and may sleep.

There may be tasks expecting incoming messages, in which case the device should periodically wake up and call emberPollForData() in order to receive messages. This function can only be called when the node type is EMBER_SLEEPY_END_DEVICE or EMBER_MOBILE_END_DEVICE.

Returns

TRUE if the application may sleep but the stack may be expecting incoming messages.

6.7.4.18   boolean emberOkToHibernate (void)

Indicates whether the stack currently has any tasks pending.

If there are no pending tasks then emberTick() does not need to be called until the next time a stack API function is called. This function can only be called when the node type is EMBER_SLEEPY_END_DEV-ICE or EMBER_MOBILE_END_DEVICE.

Returns

TRUE if the application may sleep for as long as it wishes.

6.7.4.19   boolean emberOkToLongPoll (void)

Indicates whether the stack is currently in a state that does not require the application to periodically poll.

Returns

TRUE if the application may stop polling periodically.

6.7.4.20   void emberStackPowerDown (void)

Immediately turns the radio power completely off.

After calling this function, you must not call any other stack function except emberStackPowerUp(). This is because all other stack functions require that the radio is powered on for their proper operation.

6.7.4.21   void emberStackPowerUp (void)

Initializes the radio. Typically called coming out of deep sleep.

For non-sleepy devices, also turns the radio on and leaves it in rx mode.

6.8 Security

Modules

- Trust Center

Macros

- #define EMBER_JOIN_NO_PRECONFIG_KEY_BITMASK
- #define EMBER_JOIN_PRECONFIG_KEY_BITMASK

Functions

- EmberStatus emberSetInitialSecurityState (EmberInitialSecurityState ∗state)
- EmberStatus emberSetExtendedSecurityBitmask (EmberExtendedSecurityBitmask mask)
- EmberStatus emberGetExtendedSecurityBitmask (EmberExtendedSecurityBitmask ∗mask)
- EmberStatus emberGetCurrentSecurityState (EmberCurrentSecurityState ∗state)
- EmberStatus emberGetKey (EmberKeyType type, EmberKeyStruct ∗keyStruct)
- boolean emberHaveLinkKey (EmberEUI64 remoteDevice)
- EmberStatus emberGenerateRandomKey (EmberKeyData ∗keyAddress)
- void emberSwitchNetworkKeyHandler (int8u sequenceNumber)
- EmberStatus emberRequestLinkKey (EmberEUI64 partner)
- void emberZigbeeKeyEstablishmentHandler (EmberEUI64 partner, EmberKeyStatus status)
- EmberStatus emberGetKeyTableEntry (int8u index, EmberKeyStruct ∗result)
- EmberStatus emberSetKeyTableEntry (int8u index, EmberEUI64 address, boolean linkKey, Ember-KeyData ∗keyData)
- EmberStatus emberAddOrUpdateKeyTableEntry (EmberEUI64 address, boolean linkKey, Ember-KeyData ∗keyData)
- int8u emberFindKeyTableEntry (EmberEUI64 address, boolean linkKey)
- EmberStatus emberEraseKeyTableEntry (int8u index)
- EmberStatus emberClearKeyTable (void)
- EmberStatus emberStopWritingStackTokens (void)
- EmberStatus emberStartWritingStackTokens (void)
- boolean emberWritingStackTokensEnabled (void)
- EmberStatus emberApsCryptMessage (boolean encrypt, EmberMessageBuffer buffer, int8u aps-HeaderEndIndex, EmberEUI64 remoteEui64)
- EmberStatus emberGetMfgSecurityConfig (EmberMfgSecurityStruct ∗settings)
- EmberStatus emberSetMfgSecurityConfig (int32u magicNumber, const EmberMfgSecurityStruct ∗settings)

6.8.1 Detailed Description

This file describes the functions necessary to manage security for a regular device. There are three major modes for security and applications should link in the appropriate library:

- Residential security uses only network keys. This is the only supported option for ZigBee 2006 devices.

- Standard security uses network keys with optional link keys. Ember strongly recommends using Link Keys. It is possible for 2006 devices to run on a network that uses Standard Security.

- High Security uses network keys and requires link keys derived via SKKE. Devices that do not support link keys (i.e. 2006 devices) are not allowed. High Security also uses Entity Authentication to synchronize frame counters between neighboring devices and children.

See security.h for source code.

6.8.2 Macro Definition Documentation

6.8.2.1 #define EMBER_JOIN_NO_PRECONFIG_KEY_BITMASK

A non-Trust Center Device configuration bitmask example. There is no Preconfigured Link Key, so the NWK key is expected to be sent in-the-clear. The device will request a Trust Center Link key after getting the Network Key.

Definition at line 96 of file security.h.

6.8.2.2 #define EMBER_JOIN_PRECONFIG_KEY_BITMASK

A non-Trust Center device configuration bitmask example. The device has a Preconfigured Link Key and expects to receive a NWK Key encrypted at the APS Layer. A NWK key sent in-the-clear will be rejected.

Definition at line 106 of file security.h.

6.8.3 Function Documentation

6.8.3.1 EmberStatus emberSetInitialSecurityState (EmberInitialSecurityState ∗ *state*)

This function sets the initial security state that will be used by the device when it forms or joins a network. If security is enabled then this function **must** be called prior to forming or joining the network. It must also be called if the device left the network and wishes to form or join another network.

This call **should not** be used when restoring prior network operation from saved state via emberNetwork-Init as this will cause saved security settings and keys table from the prior network to be erased, resulting in improper keys and/or frame counter values being used, which will prevent proper communication with other devices in the network. Calling emberNetworkInit is sufficient to restore all saved security settings after a reboot and re-enter the network.

The call may be used by either the Trust Center or non Trust Center devices, the options that are set are different depending on which role the device will assume. See the EmberInitialSecurityState structure for more explanation about the various security settings.

The function will return EMBER_SECURITY_CONFIGURATION_INVALID in the following cases:

- Distributed Trust Center Mode was enabled with Hashed Link Keys.

- High Security was specified.

Parameters

state	The security configuration to be set.

Returns

An EmberStatus value. EMBER_SUCCESS if the security state has been set successfully. EMBER_-INVALID_CALL if the device is not in the EMBER_NO_NETWORK state. The value EMBER_SE-CURITY_CONFIGURATION_INVALID is returned if the combination of security parameters is not valid. EMBER_KEY_INVALID is returned if a reserved or invalid key value was passed in the key structure for one of the keys.

6.8.3.2 EmberStatus emberSetExtendedSecurityBitmask (EmberExtendedSecurityBitmask *mask*)

Sets the extended initial security bitmask.

Parameters

mask	An object of type EmberExtendedSecurityBitmask that indicates what the extended security bitmask should be set to.

Returns

EMBER_SUCCESS if the security settings were successfully retrieved. EMBER_INVALID_CALL otherwise.

6.8.3.3 EmberStatus emberGetExtendedSecurityBitmask (EmberExtendedSecurityBitmask ∗ *mask*)

Gets the extended security bitmask that is being used by a device.

Parameters

mask	A pointer to an EmberExtendedSecurityBitmask value into which the extended security bitmask will be copied.

Returns

EMBER_SUCCESS if the security settings were successfully retrieved. EMBER_INVALID_CALL otherwise.

6.8.3.4 EmberStatus emberGetCurrentSecurityState (EmberCurrentSecurityState ∗ *state*)

Gets the security state that is being used by a device joined into the Network.

Parameters

state	A pointer to an EmberCurrentSecurityState value into which the security configuration will be copied.

Returns

EMBER_SUCCESS if the security settings were successfully retrieved. EMBER_NOT_JOINED if the device is not currently joined in the network.

6.8.3.5 EmberStatus emberGetKey (EmberKeyType *type,* EmberKeyStruct ∗ *keyStruct*)

Gets the specified key and its associated data. This can retrieve the Trust Center Link Key, Current Network Key, or Next Network Key. On the 35x series chips, the data returned by this call is governed by the security policy set in the manufacturing token for TOKEN_MFG_SECURITY_CONFIG. See the API calls ember-SetMfgSecurityConfig() and emberGetMfgSecurityConfig() for more information. If the security policy is not set to EMBER_KEY_PERMISSIONS_READING_ALLOWED, then the actual encryption key value will not be returned. Other meta-data about the key will be returned. The 2xx series chips have no such restrictions.

Parameters

type	The Type of key to get (e.g. Trust Center Link or Network).
keyStruct	A pointer to the EmberKeyStruct data structure that will be populated with the pertinent information.

Returns

EMBER_SUCCESS if the key was retrieved successfully. EMBER_INVALID_CALL if an attempt was made to retrieve an EMBER_APPLICATION_LINK_KEY or EMBER_APPLICATION_MAS-TER_KEY.

6.8.3.6 boolean emberHaveLinkKey (EmberEUI64 *remoteDevice*)

Returns TRUE if a link key is available for securing messages sent to the remote device.

Parameters

remoteDevice	The long address of a some other device in the network.

Returns

boolean Returns TRUE if a link key is available.

6.8.3.7 EmberStatus emberGenerateRandomKey (EmberKeyData ∗ *keyAddress*)

Generates a Random Key (link, network, or master) and returns the result.

It copies the key into the array that `result` points to. This is an time-expensive operation as it needs to obtain truly random numbers.

Parameters

keyAddress	A pointer to the location in which to copy the generated key.

Returns

EMBER_SUCCESS on success, EMBER_INSUFFICIENT_RANDOM_DATA on failure.

6.8.3.8 void emberSwitchNetworkKeyHandler (int8u *sequenceNumber*)

A callback to inform the application that the Network Key has been updated and the node has been switched over to use the new key. The actual key being used is not passed up, but the sequence number is.

Parameters

sequence-Number	The sequence number of the new network key.

6.8.3.9 EmberStatus emberRequestLinkKey (EmberEUI64 *partner*)

A function to request a Link Key from the Trust Center with another device device on the Network (which could be the Trust Center). A Link Key with the Trust Center is possible but the requesting device cannot be the Trust Center. Link Keys are optional in ZigBee Standard Security and thus the stack cannot know whether the other device supports them.

If the partner device is the Trust Center, then only that device needs to request the key. The Trust Center will immediately respond to those requests and send the key back to the device.

If the partner device is not the Trust Center, then both devices must request an Application Link Key with each other. The requests will be sent to the Trust Center for it to answer. The Trust Center will store the first request and wait EMBER_REQUEST_KEY_TIMEOUT for the second request to be received. The Trust Center only supports one outstanding Application key request at a time and therefore will ignore other requests that are not associated with the first request.

Sleepy devices should poll at a higher rate until a response is received or the request times out.

The success or failure of the request is returned via emberZigbeeKeyEstablishmentHandler(...)

Parameters

partner	The IEEE address of the partner device. If NULL is passed in then the Trust Center IEEE Address is assumed.

Returns

> EMBER_SUCCESS if the call succeeds, or EMBER_NO_BUFFERS.

6.8.3.10 void emberZigbeeKeyEstablishmentHandler (EmberEUI64 *partner,* EmberKeyStatus *status*)

A callback to the application to notify it of the status of the request for a Link Key. The application should define EMBER_APPLICATION_HAS_ZIGBEE_KEY_ESTABLISHMENT_HANDLER in order to implement its own handler.

Parameters

partner	The IEEE address of the partner device. Or all zeros if the Key establishment failed.
status	The status of the key establishment.

6.8.3.11 EmberStatus emberGetKeyTableEntry (int8u *index,* EmberKeyStruct ∗ *result*)

A function used to obtain data from the Key Table. On the 35x series chips, the data returned by this call is governed by the security policy set in the manufacturing token for TOKEN_MFG_SECURITY_-CONFIG. See the API calls emberSetMfgSecurityConfig() and emberGetMfgSecurityConfig() for more information. If the security policy is not set to EMBER_KEY_PERMISSIONS_READING_ALLOWED, then the actual encryption key value will not be returned. Other meta-data about the key will be returned. The 2xx series chips have no such restrictions.

Parameters

index	The index in the key table of the entry to get.
result	A pointer to the location of an EmberKeyStruct that will contain the results retrieved by the stack.

Returns

EMBER_TABLE_ENTRY_ERASED if the index is an erased key entry. EMBER_INDEX_OUT_O-F_RANGE if the passed index is not valid. EMBER_SUCCESS on success.

6.8.3.12 EmberStatus emberSetKeyTableEntry (int8u *index,* EmberEUI64 *address,* boolean *linkKey,* EmberKeyData ∗ *keyData*)

A function to set an entry in the key table.

Parameters

index	The index in the key table of the entry to set.
address	The address of the partner device associated with the key.
keyData	A pointer to the key data associated with the key entry.
linkKey	A boolean indicating whether this is a Link or Master Key.

Returns

EMBER_KEY_INVALID if the passed key data is using one of the reserved key values. EMBER_I-NDEX_OUT_OF_RANGE if passed index is not valid. EMBER_SUCCESS on success.

6.8.3.13 EmberStatus emberAddOrUpdateKeyTableEntry (EmberEUI64 *address,* boolean *linkKey,* EmberKeyData ∗ *keyData*)

This function add a new entry in the key table or updates an existing entry with a new key. It first searches the key table for an entry that has a matching EUI64. If it does not find one it searches for the first free entry. If it is successful in either case, it sets the entry with the EUI64, key data, and flag that indicates if it is a Link or Master Key. The Incoming Frame Counter for that key is also reset to 0. If no existing entry was found, and there was not a free entry in the table, then the call will fail.

Parameters

address	The IEEE Address of the partner device that shares the key.
linkKey	A boolean indicating whether this is a Link or Master key.
keyData	A pointer to the actual key data.

Returns

EMBER_TABLE_FULL if no free entry was found to add. EMBER_KEY_INVALID if the passed key was a reserved value. ::EMBER_KEY_TABLE_ADDRESS_NOT_VALID if the passed address is reserved or invalid. EMBER_SUCCESS on success.

6.8.3.14 int8u emberFindKeyTableEntry (EmberEUI64 *address,* boolean *linkKey*)

A function to search the key table and find an entry matching the specified IEEE address and key type.

Parameters

address	The IEEE Address of the partner device that shares the key. To find the first empty entry pass in an address of all zeros.
linkKey	A boolean indicating whether to search for an entry containing a Link or Master Key.

Returns

The index that matches the search criteria, or 0xFF if no matching entry was found.

6.8.3.15 EmberStatus emberEraseKeyTableEntry (int8u *index*)

A function to clear a single entry in the key table.

Parameters

index	The index in the key table of the entry to erase.

Returns

EMBER_SUCCESS if the index is valid and the key data was erased. EMBER_KEY_INVALID if the index is out of range for the size of the key table.

6.8.3.16 EmberStatus emberClearKeyTable (void)

This function clears the key table of the current network.

Returns

EMBER_SUCCESS if the key table was successfully cleared. EMBER_INVALID_CALL otherwise.

6.8.3.17 EmberStatus emberStopWritingStackTokens (void)

This function suppresses normal write operations of the stack tokens. This is only done in rare circumstances when the device already has network parameters but needs to conditionally rejoin a network in order to perform a security message exchange (i.e. key establishment). If the network is not authenticated properly, it will need to forget any stack data it used and return to the old network. By suppressing writing of the stack tokens the device will not have stored any persistent data about the network and a reboot will clear the RAM copies. The Smart Energy profile feature Trust Center Swap-out uses this in order to securely replace the Trust Center and re-authenticate to it.

Returns

EMBER_SUCCESS if it could allocate temporary buffers to store network information. EMBER_N-O_BUFFERS otherwise.

6.8.3.18 EmberStatus emberStartWritingStackTokens (void)

This function will immediately write the value of stack tokens and then resume normal network operation by writing the stack tokens at appropriate intervals or changes in state. It has no effect unless a previous call was made to emberStopWritingStackTokens().

Returns

EMBER_SUCCESS if it had previously unwritten tokens from a call to emberStopWritingStack-Tokens() that it now wrote to the token system. EMBER_INVALID_CALL otherwise.

6.8.3.19 boolean emberWritingStackTokensEnabled (void)

This function will determine whether stack tokens will be written to persistent storage when they change. By default it is set to TRUE meaning the stack will update its internal tokens via HAL calls when the associated RAM values change.

Returns

TRUE if the device will update the persistent storage for tokens when values in RAM change. FALSE otherwise.

6.8.3.20 EmberStatus emberApsCryptMessage (boolean *encrypt,* EmberMessageBuffer *buffer,* int8u *apsHeaderEndIndex,* EmberEUI64 *remoteEui64*)

This function performs APS encryption/decryption of messages directly. Normally the stack handles all APS encryption/decryption automatically and there is no need to call this function. If APS data is sent or received via some other means (such as over interpan) then APS encryption/decryption must be done manually. If decryption is performed then the Auxiliary security header and MIC will be removed from the message. If encrypting, then the auxiliary header and MIC will be added to the message. This is only available on SOC platforms.

Parameters

encrypt	a boolean indicating whether perform encryption (TRUE) or decryption (FALSE).
buffer	An EmberMessageBuffer containing the APS frame to decrypt or encrypt.
apsHeaderEnd-Index	The index into the buffer where the APS header ends. If encryption is being performed this should point to the APS payload, where an Auxiliary header will be inserted. If decryption is being performed, this should point to the start of the Auxiliary header frame.
remoteEui64	the EmberEUI64 of the remote device the message was received from (decryption) or being sent to (encryption).

Returns

EMBER_SUCCESS if encryption/decryption was performed successfully. An appropriate Ember-Status code on failure.

6.8.3.21 EmberStatus emberGetMfgSecurityConfig (EmberMfgSecurityStruct * *settings*)

This function will retrieve the security configuration stored in manufacturing tokens. It is only available on the 35x series. See emberSetMfgSecurityConfig() for more details.

Parameters

settings	A pointer to the EmberMfgSecurityStruct variable that will contain the returned data.

Returns

EMBER_SUCCESS if the tokens were successfully read. EmberStatus error code otherwise.

6.8.3.22 EmberStatus emberSetMfgSecurityConfig (int32u *magicNumber,* const EmberMfgSecurityStruct * *settings*)

This function will set the security configuration to be stored in manufacturing tokens. It is only available on the 35x series. This API must be called with care. Once set, a manufacturing token CANNOT BE UNSET without using the ISA3 tools and connecting the chip via JTAG. Additionally, a chip with read protection enabled cannot have its configuration changed without a full chip erase. Thus this provides a way to disallow access to the keys at runtime that cannot be undone.

To call this API the magic number must be passed in corresponding to EMBER_MFG_SECURITY_C-ONFIG_MAGIC_NUMBER. This prevents accidental calls to this function when emberGetMfgSecurity-Config() was actually intended.

This function will disable external access to the actual key data used for decryption/encryption outside the stack. Attempts to call emberGetKey() or emberGetKeyTableEntry() will return the meta-data (e.g. sequence number, associated EUI64, frame counters) but the key value may be modified, see below.

The stack always has access to the actual key data.

If the EmberKeySettings within the EmberMfgSecurityStruct are set to EMBER_KEY_PERMISSIONS_-NONE then the key value with be set to zero when emberGetKey() or emberGetKeyTableEntry() is called. If the EmberKeySettings within the EmberMfgSecurityStruct are set to EMBER_KEY_PERMISSIONS_-HASHING_ALLOWED, then the AES-MMO hash of the key will replace the actual key data when calls to emberGetKey() or emberGetKeyTableEntry() are made. If the EmberKeySettings within the EmberMfg-SecurityStruct are set to EMBER_KEY_PERMISSIONS_READING_ALLOWED, then the actual key data is returned. This is the default value of the token.

Parameters

magicNumber	A 32-bit value that must correspond to EMBER_MFG_SECURITY_CONFIG_MAG-IC_NUMBER, otherwise EMBER_INVALID_CALL will be returned.
settings	The security settings that are intended to be set by the application and written to manufacturing token.

Returns

EMBER_BAD_ARGUMENT if the passed magic number is invalid. EMBER_INVALID_CALL if the chip does not support writing MFG tokens (i.e. em2xx) EMBER_SECURITY_CONFIGURATI-ON_INVALID if there was an attempt to write an unerased manufacturing token (i.e. the token has already been set).

6.9 Trust Center

Macros

- #define EMBER_FORM_TRUST_CENTER_NETWORK_BITMASK
- #define EMBER_FORM_DISTRIBUTED_TRUST_CENTER_NETWORk_BITMASK

Functions

- EmberStatus emberBroadcastNextNetworkKey (EmberKeyData ∗key)
- EmberStatus emberSendUnicastNetworkKeyUpdate (EmberNodeId targetShort, EmberEUI64 target-Long, EmberKeyData ∗newKey)
- EmberStatus emberBroadcastNetworkKeySwitch (void)
- EmberJoinDecision emberTrustCenterJoinHandler (EmberNodeId newNodeId, EmberEUI64 new-NodeEui64, EmberDeviceUpdate status, EmberNodeId parentOfNewNode)
- EmberStatus emberBecomeTrustCenter (EmberKeyData ∗newNetworkKey)
- EmberStatus emberSendRemoveDevice (EmberNodeId destShort, EmberEUI64 destLong, Ember-EUI64 deviceToRemoveLong)

Variables

- EmberLinkKeyRequestPolicy emberTrustCenterLinkKeyRequestPolicy
- EmberLinkKeyRequestPolicy emberAppLinkKeyRequestPolicy

6.9.1 Detailed Description

This file describes the routines used by the Trust Center to manage devices in the network. The Trust center decides whether to use preconfigured keys or not and manages passing out keys to joining and rejoining devices. The Trust Center also sends out new keys and decides when to start using them.

See trust-center.h for source code

6.9.2 Macro Definition Documentation

6.9.2.1 #define EMBER_FORM_TRUST_CENTER_NETWORK_BITMASK

A Trust Center device configuration bitmask example. The Trust Center is expected to be setup with a Network Key Preconfigured Link Key that is global throughout all devices on the Network. The decision whether or not to send the key in-the-clear is NOT controlled through this bitmask. That is controlled via the emberTrustCenterJoinHandler(...) function.

Definition at line 29 of file trust-center.h.

6.9.2.2 #define EMBER_FORM_DISTRIBUTED_TRUST_CENTER_NETWORk_BITMASK

A coordinator device configuration bitmask example. The coordinator is expected to be setup with a Network Key and a Preconfigured Link Key that is global throughout all devices on the Network. The decision whether or not to send the key in-the-clear is decentralized, and each individual router can make this decision via the emberTrustCenterJoinHandler(...) function.

Definition at line 42 of file trust-center.h.

6.9.3 Function Documentation

6.9.3.1 EmberStatus emberBroadcastNextNetworkKey (EmberKeyData ∗ *key*)

This function broadcasts a new encryption key, but does not tell the nodes in the network to start using it.

To tell nodes to switch to the new key, use emberBroadcastNetworkKeySwitch(). This is only valid for the Trust Center/Coordinator. It is not valid when operating in Distributed Trust Center mode.

It is up to the application to determine how quickly to send the Switch Key after sending the alternate encryption key. The factors to consider are the polling rate of sleepy end devices, and the buffer size of their parent nodes. Sending too quickly may cause a sleepy end device to miss the Alternate Encryption Key and only get the Switch Key message, which means it will be unable to change to the new network key.

Parameters

key	A pointer to a 16-byte encryption key (EMBER_ENCRYPTION_KEY_SIZE). A NULL (or all zero key) may be passed in, which will cause the stack to randomly generate a new key.

Returns

An EmberStatus value that indicates the success or failure of the command.

6.9.3.2 EmberStatus emberSendUnicastNetworkKeyUpdate (EmberNodeId *targetShort,* EmberEUI64 *targetLong,* EmberKeyData ∗ *newKey*)

This function sends a unicast update of the network key to the target device. The APS command will be encrypted using the device's current APS link key. On success, the bit ::EMBER_KEY_UNICAST_NWK_KEY_UPDATE_SENT will be set in the link key table entry for the device. When a successful call is made to emberBroadcastNetworkKeySwitch(), the bit will be cleared for all entries.

On the first call to this function the trust center's local copy of the alternate NWK key will be updated with the new value.

Both the short and long address of the device must be known ahead of time and passed in as parameters. It is assumed that the application has already generated the new network key and will pass the same key value on subsequent calls to send the key to different nodes in the network.

Parameters

targetStort	the short node ID of the device to send a NWK key update to.
targetLong	the EUI64 of the node to send a key update NWK key update to.
nwkKey	a pointer to the new NWK key value.

Returns

an EmberStatus value that indicates the success or failure of the command.

6.9.3.3 EmberStatus emberBroadcastNetworkKeySwitch (void)

This function broadcasts a switch key message to tell all nodes to change to the sequence number of the previously sent Alternate Encryption Key.

This function is only valid for the Trust Center/Coordinator, and will also cause the Trust Center/Coordinator to change its Network Key. It is not valid when operating in Distributed Trust Center mode.

Returns

An EmberStatus value that indicates the success or failure of the command.

6.9.3.4 EmberJoinDecision emberTrustCenterJoinHandler (EmberNodeId *newNodeId*, EmberEUI64 *newNodeEui64*, EmberDeviceUpdate *status*, EmberNodeId *parentOfNewNode*)

A callback that allows the application running on the Trust Center (which is the coordinator for ZigBee networks) to control which nodes are allowed to join the network. If the node is allowed to join, the trust center must decide how to send it the Network Key, encrypted or unencrypted.

A default handler is provided and its behavior is as follows. A status of ::EMBER_DEVICE_SECURED_REJOIN means that the device has the Network Key, no action is required from the Trust Center. A status of EMBER_DEVICE_LEFT also requires no action. In both cases EMBER_NO_ACTION is returned.

When operating in a network with a Trust Center and there is a Global Link Key configured, EMBER_USE_PRECONFIGURED_KEY will be returned which means the Trust Center is using a pre-configured Link Key. The Network Key will be sent to the joining node encrypted with the Link Key. If a Link Key has not been set on the Trust Center, EMBER_DENY_JOIN is returned.

The ::EMBER_ASK_TRUST_CENTER decision has been deprecated. This function will not be called for a router or end device when operating in a Network With a Trust Center.

If the device is a router in a network that is operating in a Distributed Trust Center Security mode, then the handler will be called by the stack.

The default handler in a Distributed Trust Center Security mode network is as follows: If the router received an encrypted Network Key when it joined, then a pre-configured Link key will be used to send the Network Key Encrypted to the joining device (EMBER_USE_PRECONFIGURED_KEY). If the router received the Network Key in the clear, then it will also send the key in the clear to the joining node (EMBER_SEND_KEY_IN_THE_CLEAR).

Parameters

newNodeId	The node id of the device wishing to join.
newNodeEui64	The EUI64 of the device wishing to join.
status	The EmberUpdateDeviceStatus indicating whether the device is joining/rejoining or leaving.
parentOfNewNode	The node id of the parent of device wishing to join.

Returns

EMBER_USE_PRECONFIGURED_KEY to allow the node to join without sending it the key. EMBER_SEND_KEY_IN_THE_CLEAR to allow the node to join and send it the key. EMBER_DENY_JOIN to reject the join attempt. value should not be returned if the local node is itself the trust center).

6.9.3.5 EmberStatus emberBecomeTrustCenter (EmberKeyData ∗ *newNetworkKey*)

This function causes a coordinator to become the Trust Center when it is operating in a network that is not using one. It will send out an updated Network Key to all devices that will indicate a transition of

the network to now use a Trust Center. The Trust Center should also switch all devices to using this new network key with a call to emberBroadcastNetworkKeySwitch().

Parameters

newNetwork-Key	The key data for the Updated Network Key.

Returns

An EmberStatus value that indicates the success or failure of the command.

6.9.3.6 EmberStatus emberSendRemoveDevice (EmberNodeId *destShort,* EmberEUI64 *destLong,* EmberEUI64 *deviceToRemoveLong*)

This sends an APS remove device command to the destination. If the destination is an end device then, this must be sent to the parent of the end device. In that case the deviceToRemoveLong and the destLong will be different values. Otherwise if a router is being asked to leave, then those parameters will be the same. This command will be APS encrypted with the destination device's link key, which means a link key must be present.

Parameters

destShort	The short node ID of the destination of the command.
destLong	The EUI64 of the destination of the command.
deviceTo-RemoveLong	The EUI64 of the target device being asked to leave.

Returns

An EmberStatus value indicating success or failure of the operation.

6.9.4 Variable Documentation

6.9.4.1 EmberLinkKeyRequestPolicy emberTrustCenterLinkKeyRequestPolicy

This variable controls the policy that the Trust Center uses for determining whether to allow or deny requests for Trust Center link keys.

The following is a good set of guidelines for TC Link key requests:

- If preconfigured TC link keys are setup on devices, requests for the TC key should never be allowed (EMBER_DENY_KEY_REQUESTS).

- If devices request link keys during joining (i.e. join in the clear and set EMBER_GET_LINK_KEY_WHEN_JOINING) then it is advisable to allow requesting keys from the TC for a short period of time (e.g. the same amount of time "permit joining" is turned on). Afterwards requests for the TC link key should be denied.

6.9.4.2 EmberLinkKeyRequestPolicy emberAppLinkKeyRequestPolicy

This variable controls the policy that the Trust Center uses for determining whether to allow or deny requests for application link keys between device pairs. When a request is received and the policy is EMBER_ALLOW_KEY_REQUESTS, the TC will generate a random key and send a copy to both devices encrypted with their individual link keys.

Generally application link key requests may always be allowed.

6.10   Binding Table

Functions

- EmberStatus emberSetBinding (int8u index, EmberBindingTableEntry ∗value)
- EmberStatus emberGetBinding (int8u index, EmberBindingTableEntry ∗result)
- EmberStatus emberDeleteBinding (int8u index)
- boolean emberBindingIsActive (int8u index)
- EmberNodeId emberGetBindingRemoteNodeId (int8u index)
- void emberSetBindingRemoteNodeId (int8u index, EmberNodeId id)
- EmberStatus emberClearBindingTable (void)
- EmberStatus emberRemoteSetBindingHandler (EmberBindingTableEntry ∗entry)
- EmberStatus emberRemoteDeleteBindingHandler (int8u index)
- int8u emberGetBindingIndex (void)
- EmberStatus emberSetReplyBinding (int8u index, EmberBindingTableEntry ∗entry)
- EmberStatus emberNoteSendersBinding (int8u index)

6.10.1   Detailed Description

EmberZNet binding table API. See binding-table.h for source code.

6.10.2   Function Documentation

6.10.2.1   EmberStatus **emberSetBinding (int8u *index,* EmberBindingTableEntry ∗ *value*)**

Sets an entry in the binding table by copying the structure pointed to by `value` into the binding table.

Note

> You do not need to reserve memory for `value`.

Parameters

index	The index of a binding table entry.
value	A pointer to a structure.

Returns

> An EmberStatus value that indicates the success or failure of the command.

6.10.2.2   EmberStatus **emberGetBinding (int8u *index,* EmberBindingTableEntry ∗ *result*)**

Copies a binding table entry to the structure that `result` points to.

Parameters

index	The index of a binding table entry.
result	A pointer to the location to which to copy the binding table entry.

Returns

An EmberStatus value that indicates the success or failure of the command.

6.10.2.3 EmberStatus emberDeleteBinding (int8u *index*)

Deletes a binding table entry.

Parameters

index	The index of a binding table entry.

Returns

An EmberStatus value that indicates the success or failure of the command.

6.10.2.4 boolean emberBindingIsActive (int8u *index*)

Indicates whether any messages are currently being sent using this binding table entry.

Note that this function does not indicate whether a binding is clear. To determine whether a binding is clear, check the EmberBindingTableEntry structure that defines the binding. The type field should have the value EMBER_UNUSED_BINDING.

Parameters

index	The index of a binding table entry.

Returns

TRUE if the binding table entry is active, FALSE otherwise.

6.10.2.5 EmberNodeId emberGetBindingRemoteNodeId (int8u *index*)

Returns the node ID for the binding's destination, if the ID is known.

If a message is sent using the binding and the destination's ID is not known, the stack will discover the ID by broadcasting a ZDO address request. The application can avoid the need for this discovery be calling emberNoteSendersBinding() whenever a message arrives from the binding's destination, or by calling emberSetBindingRemoteNodeId() when it knows the correct ID via some other means, such as having saved it in nonvolatile memory.

The destination's node ID is forgotten when the binding is changed, when the local node reboots or, much more rarely, when the destination node changes its ID in response to an ID conflict.

Parameters

index	The index of a binding table entry.

Returns

The short ID of the destination node or EMBER_NULL_NODE_ID if no destination is known.

6.10.2.6   void emberSetBindingRemoteNodeId (int8u *index,* EmberNodeId *id*)

Set the node ID for the binding's destination. See emberGetBindingRemoteNodeId() for a description.

Parameters

index	The index of a binding table entry.
id	The ID of the binding's destination.

6.10.2.7   EmberStatus emberClearBindingTable (void)

Deletes all binding table entries.

Returns

An EmberStatus value that indicates the success or failure of the command.

6.10.2.8   EmberStatus emberRemoteSetBindingHandler (EmberBindingTableEntry ∗ *entry*)

A callback invoked when a remote node requests that a binding be added to the local binding table (via the ZigBee Device Object at endpoint 0).

The application is free to add the binding to the binding table, ignore the request, or take some other action. It is recommended that nonvolatile bindings be used for remote provisioning applications.

The binding's type defaults to EMBER_UNICAST_BINDING. The application should set the type as appropriate for the binding's local endpoint and cluster ID.

If the application includes emberRemoteSetBindingHandler(), it must define EMBER_APPLICATION-_HAS_REMOTE_BINDING_HANDLER in its CONFIGURATION_HEADER and also include ember-RemoteDeleteBindingHandler().

Parameters

entry	A pointer to a new binding table entry.

Returns

EMBER_SUCCESS if the binding was added to the table and any other status if not.

6.10.2.9   EmberStatus emberRemoteDeleteBindingHandler (int8u *index*)

A callback invoked when a remote node requests that a binding be removed from the local binding table (via the ZigBee Device Object at endpoint 0).

The application is free to remove the binding from the binding table, ignore the request, or take some other action.

If the application includes emberRemoteDeleteBindingHandler(), it must define EMBER_APPLICATIO-N_HAS_REMOTE_BINDING_HANDLER in its CONFIGURATION_HEADER and also include ember-RemoteSetBindingHandler().

Parameters

index	The index of the binding entry to be removed.

Returns

EMBER_SUCCESS if the binding was removed from the table and any other status if not.

6.10.2.10 int8u emberGetBindingIndex (void)

Returns a binding index that matches the current incoming message, if known.

A binding matches the incoming message if:

- The binding's source endpoint is the same as the message's destination endpoint.

- The binding's destination endpoint is the same as the message's source endpoint.

- The source of the message has been previously identified as the the binding's remote node by a successful address discovery or by the application via a call to either emberSetReplyBinding() or emberNoteSendersBinding().

Note

This function can be called only from within emberIncomingMessageHandler().

Returns

The index of a binding that matches the current incoming message or `0xFF` if there is no matching binding.

6.10.2.11 EmberStatus emberSetReplyBinding (int8u *index,* EmberBindingTableEntry ∗ *entry*)

Creates a binding table entry for the sender of a message, which can be used to send messages to that sender.

This function is identical to emberSetBinding() except that calling it tells the stack that this binding corresponds to the sender of the current message. The stack uses this information to associate the sender's routing info with the binding table entry.

Note

This function may only be called from within emberIncomingMessageHandler().

Parameters

index	The index of the binding to set.
entry	A pointer to data for the binding.

Returns

An EmberStatus value that indicates the success or failure of the command.

6.10.2.12   EmberStatus emberNoteSendersBinding (int8u *index*)

Updates the routing information associated with a binding table entry for the sender of a message.

This function should be used in place of emberSetReplyBinding() when a message arrives from a remote endpoint for which a binding already exists.

Parameters

index	The index of the binding to update.

Returns

An EmberStatus value that indicates the success or failure of the command.

6.11 Configuration

Macros

- #define EMBER_API_MAJOR_VERSION
- #define EMBER_API_MINOR_VERSION
- #define EMBER_STACK_PROFILE
- #define EMBER_MAX_END_DEVICE_CHILDREN
- #define EMBER_SECURITY_LEVEL
- #define EMBER_CHILD_TABLE_SIZE
- #define EMBER_KEY_TABLE_SIZE
- #define EMBER_CERTIFICATE_TABLE_SIZE
- #define EMBER_MAX_DEPTH
- #define EMBER_MAX_HOPS
- #define EMBER_PACKET_BUFFER_COUNT
- #define EMBER_MAX_NEIGHBOR_TABLE_SIZE
- #define EMBER_NEIGHBOR_TABLE_SIZE
- #define EMBER_INDIRECT_TRANSMISSION_TIMEOUT
- #define EMBER_MAX_INDIRECT_TRANSMISSION_TIMEOUT
- #define EMBER_SEND_MULTICASTS_TO_SLEEPY_ADDRESS
- #define EMBER_END_DEVICE_POLL_TIMEOUT
- #define EMBER_END_DEVICE_POLL_TIMEOUT_SHIFT
- #define EMBER_MOBILE_NODE_POLL_TIMEOUT
- #define EMBER_APS_UNICAST_MESSAGE_COUNT
- #define EMBER_BINDING_TABLE_SIZE
- #define EMBER_ADDRESS_TABLE_SIZE
- #define EMBER_RESERVED_MOBILE_CHILD_ENTRIES
- #define EMBER_ROUTE_TABLE_SIZE
- #define EMBER_DISCOVERY_TABLE_SIZE
- #define EMBER_MULTICAST_TABLE_SIZE
- #define EMBER_SOURCE_ROUTE_TABLE_SIZE
- #define EMBER_DEFAULT_BROADCAST_TABLE_SIZE
- #define EMBER_BROADCAST_TABLE_SIZE
- #define EMBER_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 EMBER_SUPPORTED_NETWORKS

6.11.1   Detailed Description

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

```
#define EMBER_PACKET_BUFFER_COUNT 8
```

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

```
#define EMBER_APPLICATION_HAS_DEBUG_HANDLER
```

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

6.11.2   Macro Definition Documentation

6.11.2.1   #define EMBER_API_MAJOR_VERSION

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

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

6.11.2.2   #define EMBER_API_MINOR_VERSION

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

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

6.11.2.3   #define EMBER_STACK_PROFILE

Specifies the stack profile. The default is Profile 0.

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

```
#define EMBER_STACK_PROFILE 1
```

or

```
#define EMBER_STACK_PROFILE 2
```

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

6.11.2.4   #define EMBER_MAX_END_DEVICE_CHILDREN

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

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

6.11.2.5 #define EMBER_SECURITY_LEVEL

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

Definition at line 123 of file ember-configuration-defaults.h.

6.11.2.6 #define EMBER_CHILD_TABLE_SIZE

The maximum number of children that a node may have.

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

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

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

Definition at line 152 of file ember-configuration-defaults.h.

6.11.2.7 #define EMBER_KEY_TABLE_SIZE

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

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

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

Definition at line 169 of file ember-configuration-defaults.h.

6.11.2.8 #define EMBER_CERTIFICATE_TABLE_SIZE

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

Definition at line 182 of file ember-configuration-defaults.h.

6.11.2.9 #define EMBER_MAX_DEPTH

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

Definition at line 195 of file ember-configuration-defaults.h.

6.11.2.10 #define EMBER_MAX_HOPS

The maximum number of hops for a message.

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

Definition at line 208 of file ember-configuration-defaults.h.

6.11.2.11 #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 218 of file ember-configuration-defaults.h.

6.11.2.12 #define EMBER_MAX_NEIGHBOR_TABLE_SIZE

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

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

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

Definition at line 232 of file ember-configuration-defaults.h.

6.11.2.13 #define EMBER_NEIGHBOR_TABLE_SIZE

Definition at line 234 of file ember-configuration-defaults.h.

6.11.2.14 #define EMBER_INDIRECT_TRANSMISSION_TIMEOUT

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

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

Definition at line 244 of file ember-configuration-defaults.h.

6.11.2.15 #define EMBER_MAX_INDIRECT_TRANSMISSION_TIMEOUT

Definition at line 246 of file ember-configuration-defaults.h.

6.11.2.16 #define EMBER_SEND_MULTICASTS_TO_SLEEPY_ADDRESS

This defines the behavior for what address multicasts are sent to The normal address is RxOnWhenIdle=T-RUE (0xFFFD). However setting this to true can change locally generated multicasts to be sent to the sleepy broadcast address (0xFFFF). Changing the default is NOT ZigBee Pro compliant and may not be interoperable.

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

6.11.2.17 #define EMBER_END_DEVICE_POLL_TIMEOUT

The maximum amount of time, in units determined by EMBER_END_DEVICE_POLL_TIMEOUT_SH-IFT, 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_DEVI-CE_POLL_TIMEOUT_SHIFT results in a timeout of 320 seconds, or just over five minutes. The maximum value for EMBER_END_DEVICE_POLL_TIMEOUT is 255.

Definition at line 278 of file ember-configuration-defaults.h.

6.11.2.18 #define EMBER_END_DEVICE_POLL_TIMEOUT_SHIFT

The units used for timing out end devices on their parents. See EMBER_END_DEVICE_POLL_TIMEO-UT 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_TIMEOU-T_SHIFT is 14.

Definition at line 289 of file ember-configuration-defaults.h.

6.11.2.19 #define EMBER_MOBILE_NODE_POLL_TIMEOUT

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

Definition at line 299 of file ember-configuration-defaults.h.

6.11.2.20 #define EMBER_APS_UNICAST_MESSAGE_COUNT

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

Note

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

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

Definition at line 315 of file ember-configuration-defaults.h.

6.11.2.21 #define EMBER_BINDING_TABLE_SIZE

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

Definition at line 321 of file ember-configuration-defaults.h.

6.11.2.22 #define EMBER_ADDRESS_TABLE_SIZE

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

Address table entries are 10 bytes in size.

Definition at line 329 of file ember-configuration-defaults.h.

6.11.2.23 #define EMBER_RESERVED_MOBILE_CHILD_ENTRIES

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

The maximum number of non-mobile children for a parent is EMBER_CHILD_TABLE_SIZE - EMBER-_RESERVED_MOBILE_CHILD_ENTRIES.

Definition at line 339 of file ember-configuration-defaults.h.

6.11.2.24 #define EMBER_ROUTE_TABLE_SIZE

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

Route table entries are 6 bytes in size.

Definition at line 352 of file ember-configuration-defaults.h.

6.11.2.25 #define EMBER_DISCOVERY_TABLE_SIZE

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

Discovery table entries are 9 bytes in size.

Definition at line 368 of file ember-configuration-defaults.h.

6.11.2.26 #define EMBER_MULTICAST_TABLE_SIZE

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

Multicast table entries are 3 bytes in size.

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

6.11.2.27 #define EMBER_SOURCE_ROUTE_TABLE_SIZE

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

Source route table entries are 4 bytes in size.

Definition at line 391 of file ember-configuration-defaults.h.

6.11.2.28 #define EMBER␣DEFAULT␣BROADCAST␣TABLE␣SIZE

The maximum number broadcasts during a single broadcast timeout period. The minimum and default value is 15 and can only be changed only on compatible Ember stacks. Be very careful when changing the broadcast table size as it effects timing of the broadcasts as well as number of possible broadcasts. Additionally, this value must be universal for all devices in the network otherwise a single router can overwhelm all its neighbors with more broadcasts than they can support. In general, this value should be left alone.

Broadcast table entries are 5 bytes in size.

Definition at line 411 of file ember-configuration-defaults.h.

6.11.2.29 #define EMBER␣BROADCAST␣TABLE␣SIZE

Definition at line 414 of file ember-configuration-defaults.h.

6.11.2.30 #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 432 of file ember-configuration-defaults.h.

6.11.2.31 #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 448 of file ember-configuration-defaults.h.

6.11.2.32 #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 pa-

rameter must be in the inclusive range 0 ... EMBER_MAXIMUM_ALARM_DATA_SIZE. The default value is 0.

Definition at line 468 of file ember-configuration-defaults.h.

6.11.2.33 #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 482 of file ember-configuration-defaults.h.

6.11.2.34 #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 491 of file ember-configuration-defaults.h.

6.11.2.35 #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 497 of file ember-configuration-defaults.h.

6.11.2.36 #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 504 of file ember-configuration-defaults.h.

6.11.2.37 #define EMBER_BINDING_TABLE_TOKEN_SIZE

Definition at line 510 of file ember-configuration-defaults.h.

6.11.2.38 #define EMBER_CHILD_TABLE_TOKEN_SIZE

Definition at line 513 of file ember-configuration-defaults.h.

6.11.2.39 #define EMBER_KEY_TABLE_TOKEN_SIZE

Definition at line 516 of file ember-configuration-defaults.h.

6.11.2.40 #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 532 of file ember-configuration-defaults.h.

6.11.2.41 #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 541 of file ember-configuration-defaults.h.

6.11.2.42 #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 553 of file ember-configuration-defaults.h.

6.11.2.43 #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 562 of file ember-configuration-defaults.h.

6.11.2.44 #define EMBER_SUPPORTED_NETWORKS

The number of networks supported by the stack.

Definition at line 571 of file ember-configuration-defaults.h.

6.12   Status Codes

Macros

- #define DEFINE_ERROR(symbol, value)

Enumerations

- enum { EMBER_ERROR_CODE_COUNT }

Generic Messages

These messages are system wide.

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

Packet Buffer Module Errors

- #define EMBER_NO_BUFFERS(x18)

Serial Manager Errors

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

MAC Errors

- #define EMBER_MAC_TRANSMIT_QUEUE_FULL(x39)
- #define EMBER_MAC_UNKNOWN_HEADER_TYPE(x3A)
- #define EMBER_MAC_ACK_HEADER_TYPE(x3B)
- #define EMBER_MAC_SCANNING(x3D)
- #define EMBER_MAC_NO_DATA(x31)
- #define EMBER_MAC_JOINED_NETWORK(x32)
- #define EMBER_MAC_BAD_SCAN_DURATION(x33)
- #define EMBER_MAC_INCORRECT_SCAN_TYPE(x34)
- #define EMBER_MAC_INVALID_CHANNEL_MASK(x35)

- #define EMBER_MAC_COMMAND_TRANSMIT_FAILURE(x36)
- #define EMBER_MAC_NO_ACK_RECEIVED(x40)
- #define EMBER_MAC_RADIO_NETWORK_SWITCH_FAILED(x41)
- #define EMBER_MAC_INDIRECT_TIMEOUT(x42)

Simulated EEPROM Errors

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

Flash Errors

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

Bootloader Errors

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

Transport Errors

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

HAL Module Errors

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

PHY Errors

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

Return Codes Passed to emberStackStatusHandler()

See also emberStackStatusHandler().

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

Security Errors

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

Miscellaneous Network Errors

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

Miscellaneous Utility Errors

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

Application Errors

These error codes are available for application use.

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

6.12.1   Detailed Description

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

See error-def.h for source code.

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

Note

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

6.12.2   Macro Definition Documentation

6.12.2.1   #define DEFINE_ERROR(*symbol, value*)

Macro used by error-def.h to define all of the return codes.

Parameters

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

Definition at line 35 of file error.h.

6.12.2.2 #define EMBER_SUCCESS(*x00*)

The generic "no error" message.

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

6.12.2.3 #define EMBER_ERR_FATAL(*x01*)

The generic "fatal error" message.

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

6.12.2.4 #define EMBER_BAD_ARGUMENT(*x02*)

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

Definition at line 63 of file error-def.h.

6.12.2.5 #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.

6.12.2.6 #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.

6.12.2.7 #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.

6.12.2.8 #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.

6.12.2.9   #define EMBER_NO_BUFFERS(*x18*)

There are no more buffers.

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

6.12.2.10   #define EMBER_SERIAL_INVALID_BAUD_RATE(*x20*)

Specified an invalid baud rate.

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

6.12.2.11   #define EMBER_SERIAL_INVALID_PORT(*x21*)

Specified an invalid serial port.

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

6.12.2.12   #define EMBER_SERIAL_TX_OVERFLOW(*x22*)

Tried to send too much data.

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

6.12.2.13   #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.

6.12.2.14   #define EMBER_SERIAL_RX_FRAME_ERROR(*x24*)

Detected a UART framing error.

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

6.12.2.15   #define EMBER_SERIAL_RX_PARITY_ERROR(*x25*)

Detected a UART parity error.

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

6.12.2.16   #define EMBER_SERIAL_RX_EMPTY(*x26*)

There is no received data to process.

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

6.12.2.17   #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.

6.12.2.18   #define EMBER_MAC_TRANSMIT_QUEUE_FULL(*x39*)

The MAC transmit queue is full.

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

6.12.2.19   #define EMBER_MAC_UNKNOWN_HEADER_TYPE(*x3A*)

MAC header FCF error on receive.

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

6.12.2.20   #define EMBER_MAC_ACK_HEADER_TYPE(*x3B*)

MAC ACK header received.

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

6.12.2.21   #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.

6.12.2.22   #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.

6.12.2.23   #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.

6.12.2.24   #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.

6.12.2.25   #define EMBER_MAC_INCORRECT_SCAN_TYPE(*x34*)

emberStartScan was called with an incorrect scan type.

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

6.12.2.26   #define EMBER_MAC_INVALID_CHANNEL_MASK(*x35*)

emberStartScan was called with an invalid channel mask.

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

6.12.2.27   #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.

6.12.2.28   #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.

6.12.2.29   #define EMBER_MAC_RADIO_NETWORK_SWITCH_FAILED(*x41*)

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

Definition at line 343 of file error-def.h.

6.12.2.30   #define EMBER_MAC_INDIRECT_TIMEOUT(*x42*)

Indirect data message timed out before polled.

Definition at line 353 of file error-def.h.

6.12.2.31   #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 376 of file error-def.h.

6.12.2.32   #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 392 of file error-def.h.

6.12.2.33   #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 407 of file error-def.h.

6.12.2.34   #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 425 of file error-def.h.

6.12.2.35   #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 438 of file error-def.h.

6.12.2.36   #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_STAC-K_NVDATA_VERSION were incorrect and the token system failed to properly reload default tokens and reset the Simulated EEPROM.

Definition at line 452 of file error-def.h.

6.12.2.37   #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 470 of file error-def.h.

6.12.2.38   #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 491 of file error-def.h.

6.12.2.39 #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 504 of file error-def.h.

6.12.2.40 #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 517 of file error-def.h.

6.12.2.41 #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 530 of file error-def.h.

6.12.2.42 #define EMBER_ERR_BOOTLOADER_TRAP_TABLE_BAD(*x58*)

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

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

6.12.2.43 #define EMBER_ERR_BOOTLOADER_TRAP_UNKNOWN(*x59*)

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

Definition at line 560 of file error-def.h.

6.12.2.44 #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 571 of file error-def.h.

6.12.2.45 #define EMBER_DELIVERY_FAILED(*x66*)

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

Definition at line 589 of file error-def.h.

6.12.2.46 #define EMBER_BINDING_INDEX_OUT_OF_RANGE(*x69*)

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

Definition at line 599 of file error-def.h.

6.12.2.47 #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 610 of file error-def.h.

6.12.2.48 #define EMBER_INVALID_BINDING_INDEX(*x6C*)

An invalid binding table index was given to a function.

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

6.12.2.49 #define EMBER_INVALID_CALL(*x70*)

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

Definition at line 631 of file error-def.h.

6.12.2.50 #define EMBER_COST_NOT_KNOWN(*x71*)

The link cost to a node is not known.

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

6.12.2.51 #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 652 of file error-def.h.

6.12.2.52 #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 662 of file error-def.h.

6.12.2.53 #define EMBER_BINDING_IS_ACTIVE(*x75*)

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

Definition at line 673 of file error-def.h.

6.12.2.54 #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 683 of file error-def.h.

6.12.2.55 #define EMBER_ADC_CONVERSION_DONE(*x80*)

Conversion is complete.

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

6.12.2.56 #define EMBER_ADC_CONVERSION_BUSY(*x81*)

Conversion cannot be done because a request is being processed.

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

6.12.2.57 #define EMBER_ADC_CONVERSION_DEFERRED(*x82*)

Conversion is deferred until the current request has been processed.

Definition at line 722 of file error-def.h.

6.12.2.58 #define EMBER_ADC_NO_CONVERSION_PENDING(*x84*)

No results are pending.

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

6.12.2.59 #define EMBER_SLEEP_INTERRUPTED(*x85*)

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

Definition at line 743 of file error-def.h.

6.12.2.60 #define EMBER_PHY_TX_UNDERFLOW(*x88*)

The transmit hardware buffer underflowed.

Definition at line 760 of file error-def.h.

6.12.2.61 #define EMBER_PHY_TX_INCOMPLETE(*x89*)

The transmit hardware did not finish transmitting a packet.

Definition at line 770 of file error-def.h.

6.12.2.62 #define EMBER_PHY_INVALID_CHANNEL(*x8A*)

An unsupported channel setting was specified.

Definition at line 780 of file error-def.h.

6.12.2.63 #define EMBER_PHY_INVALID_POWER(*x8B*)

An unsupported power setting was specified.

Definition at line 790 of file error-def.h.

6.12.2.64 #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 801 of file error-def.h.

6.12.2.65 #define EMBER_PHY_TX_CCA_FAIL(*x8D*)

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

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

6.12.2.66 #define EMBER_PHY_OSCILLATOR_CHECK_FAILED(*x8E*)

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

Definition at line 823 of file error-def.h.

6.12.2.67 #define EMBER_PHY_ACK_RECEIVED(*x8F*)

The expected ACK was received after the last transmission.

Definition at line 833 of file error-def.h.

6.12.2.68 #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 852 of file error-def.h.

6.12.2.69 #define EMBER_NETWORK_DOWN(*x91*)

The network is not operating.

Definition at line 862 of file error-def.h.

6.12.2.70 #define EMBER_JOIN_FAILED(*x94*)

An attempt to join a network failed.

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

6.12.2.71 #define EMBER_MOVE_FAILED(*x96*)

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

Definition at line 883 of file error-def.h.

6.12.2.72 #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 895 of file error-def.h.

6.12.2.73 #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 905 of file error-def.h.

6.12.2.74 #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 915 of file error-def.h.

6.12.2.75 #define EMBER_CHANNEL_CHANGED(x9B)

The channel has changed.

Definition at line 923 of file error-def.h.

6.12.2.76 #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 932 of file error-def.h.

6.12.2.77 #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_ENCR-YPTED_KEY).

Definition at line 943 of file error-def.h.

6.12.2.78 #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 953 of file error-def.h.

6.12.2.79 #define EMBER_NO_LINK_KEY_RECEIVED(xAE)

After a device joined a Secured Network, a Link Key was requested (EMBER_GET_LINK_KEY_WHE-N_JOINING) but no response was ever received.

Definition at line 963 of file error-def.h.

6.12.2.80   #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 974 of file error-def.h.

6.12.2.81   #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 990 of file error-def.h.

6.12.2.82   #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 1000 of file error-def.h.

6.12.2.83   #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 1014 of file error-def.h.

6.12.2.84   #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 1023 of file error-def.h.

6.12.2.85   #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 1032 of file error-def.h.

6.12.2.86   #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 1045 of file error-def.h.

6.12.2.87 #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 1054 of file error-def.h.

6.12.2.88 #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 1065 of file error-def.h.

6.12.2.89 #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 1074 of file error-def.h.

6.12.2.90 #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 1086 of file error-def.h.

6.12.2.91 #define EMBER_SECURITY_DATA_INVALID(*xBD*)

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

Definition at line 1096 of file error-def.h.

6.12.2.92 #define EMBER_NOT_JOINED(*x93*)

The node has not joined a network.

Definition at line 1114 of file error-def.h.

6.12.2.93 #define EMBER_NETWORK_BUSY(*xA1*)

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

Definition at line 1124 of file error-def.h.

6.12.2.94 #define EMBER_INVALID_ENDPOINT(*xA3*)

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

Definition at line 1135 of file error-def.h.

6.12.2.95 #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 1146 of file error-def.h.

6.12.2.96 #define EMBER_INSUFFICIENT_RANDOM_DATA(*xA5*)

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

Definition at line 1156 of file error-def.h.

6.12.2.97 #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 1166 of file error-def.h.

6.12.2.98 #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 1177 of file error-def.h.

6.12.2.99 #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 1198 of file error-def.h.

6.12.2.100 #define EMBER_INDEX_OUT_OF_RANGE(*xB1*)

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

Definition at line 1209 of file error-def.h.

6.12.2.101 #define EMBER_TABLE_FULL(*xB4*)

There are no empty entries left in the table.

Definition at line 1218 of file error-def.h.

6.12.2.102 #define EMBER_TABLE_ENTRY_ERASED(*xB6*)

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

Definition at line 1228 of file error-def.h.

6.12.2.103   #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 1238 of file error-def.h.

6.12.2.104   #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 1248 of file error-def.h.

6.12.2.105   #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 1259 of file error-def.h.

6.12.2.106   #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 1277 of file error-def.h.

6.12.2.107   #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 1278 of file error-def.h.

6.12.2.108   #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 1279 of file error-def.h.

6.12.2.109   #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 1280 of file error-def.h.

6.12.2.110   #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 1281 of file error-def.h.

6.12.2.111   #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 1282 of file error-def.h.

6.12.2.112   #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 1283 of file error-def.h.

6.12.2.113   #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 1284 of file error-def.h.

6.12.2.114   #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 1285 of file error-def.h.

6.12.2.115   #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 1286 of file error-def.h.

6.12.2.116   #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 1287 of file error-def.h.

6.12.2.117   #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 1288 of file error-def.h.

6.12.2.118 #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 1289 of file error-def.h.

6.12.2.119 #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 1290 of file error-def.h.

6.12.2.120 #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 1291 of file error-def.h.

6.12.2.121 #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 1292 of file error-def.h.

6.12.3 Enumeration Type Documentation

6.12.3.1 anonymous enum

Enumerator:

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

Definition at line 39 of file error.h.

6.13 Stack Tokens

Macros

- #define TOKEN_NEXT_ADDRESS(region, address)
- #define CURRENT_STACK_TOKEN_VERSION

Convenience Macros

The following convenience macros are used to simplify the definition process for commonly specified parameters to the basic TOKEN_DEF macro. Please see hal/micro/token.h for a more complete explanation.

- #define DEFINE_BASIC_TOKEN(name, type,...)
- #define DEFINE_COUNTER_TOKEN(name, type,...)
- #define DEFINE_INDEXED_TOKEN(name, type, arraysize,...)
- #define DEFINE_FIXED_BASIC_TOKEN(name, type, address,...)
- #define DEFINE_FIXED_COUNTER_TOKEN(name, type, address,...)
- #define DEFINE_FIXED_INDEXED_TOKEN(name, type, arraysize, address,...)
- #define DEFINE_MFG_TOKEN(name, type, address,...)

Creator Codes

The CREATOR is used as a distinct identifier tag for the token.

The CREATOR is necessary because the token name is defined differently depending on the hardware platform, therefore the CREATOR makes sure that token definitions and data stay tagged and known. The only requirement is that each creator definition must be unique. Please see hal/micro/token.h for a more complete explanation.

- #define CREATOR_STACK_NVDATA_VERSION
- #define CREATOR_STACK_BOOT_COUNTER
- #define CREATOR_STACK_NONCE_COUNTER
- #define CREATOR_STACK_ANALYSIS_REBOOT
- #define CREATOR_STACK_KEYS
- #define CREATOR_STACK_NODE_DATA
- #define CREATOR_STACK_CLASSIC_DATA
- #define CREATOR_STACK_ALTERNATE_KEY
- #define CREATOR_STACK_APS_FRAME_COUNTER
- #define CREATOR_STACK_TRUST_CENTER
- #define CREATOR_STACK_NETWORK_MANAGEMENT
- #define CREATOR_STACK_PARENT_INFO
- #define CREATOR_MULTI_NETWORK_STACK_KEYS
- #define CREATOR_MULTI_NETWORK_STACK_NODE_DATA
- #define CREATOR_MULTI_NETWORK_STACK_ALTERNATE_KEY
- #define CREATOR_MULTI_NETWORK_STACK_TRUST_CENTER
- #define CREATOR_MULTI_NETWORK_STACK_NETWORK_MANAGEMENT
- #define CREATOR_MULTI_NETWORK_STACK_PARENT_INFO
- #define CREATOR_MULTI_NETWORK_STACK_NONCE_COUNTER
- #define CREATOR_STACK_BINDING_TABLE
- #define CREATOR_STACK_CHILD_TABLE

- #define CREATOR_STACK_KEY_TABLE
- #define CREATOR_STACK_CERTIFICATE_TABLE
- #define CREATOR_STACK_ZLL_DATA
- #define CREATOR_STACK_ZLL_SECURITY

6.13.1 Detailed Description

The tokens listed here are divided into three sections (the three main types of tokens mentioned in token.h):

- manufacturing

- stack

- application

For a full explanation of the tokens, see hal/micro/token.h. See token-stack.h for source code.

There is a set of tokens predefined in the APPLICATION DATA section at the end of token-stack.h because these tokens are required by the stack, but they are classified as application tokens since they are sized by the application via its CONFIGURATION_HEADER.

The user application can include its own tokens in a header file similar to this one. The macro ::APPLI-CATION_TOKEN_HEADER should be defined to equal the name of the header file in which application tokens are defined. See the APPLICATION DATA section at the end of token-stack.h for examples of token definitions.

Since token-stack.h contains both the typedefs and the token defs, there are two #defines used to select which one is needed when this file is included. #define DEFINETYPES is used to select the type definitions and #define DEFINETOKENS is used to select the token definitions. Refer to token.h and token.c to see how these are used.

6.13.2 Macro Definition Documentation

6.13.2.1   #define TOKEN_NEXT_ADDRESS(*region, address*)

By default, tokens are automatically located after the previous token.

If a token needs to be placed at a specific location, one of the DEFINE_FIXED_∗ definitions should be used. This macro is inherently used in the DEFINE_FIXED_∗ definition to locate a token, and under special circumstances (such as manufacturing tokens) it may be explicitly used.

Parameters

region	A name for the next region being located.
address	The address of the beginning of the next region.

Definition at line 59 of file token-stack.h.

6.13.2.2   #define DEFINE_BASIC_TOKEN(*name, type, ...*)

Definition at line 95 of file token-stack.h.

6.13.2.3 #define DEFINE_COUNTER_TOKEN(*name, type, ...*)

Definition at line 98 of file token-stack.h.

6.13.2.4 #define DEFINE_INDEXED_TOKEN(*name, type, arraysize, ...*)

Definition at line 101 of file token-stack.h.

6.13.2.5 #define DEFINE_FIXED_BASIC_TOKEN(*name, type, address, ...*)

Definition at line 104 of file token-stack.h.

6.13.2.6 #define DEFINE_FIXED_COUNTER_TOKEN(*name, type, address, ...*)

Definition at line 108 of file token-stack.h.

6.13.2.7 #define DEFINE_FIXED_INDEXED_TOKEN(*name, type, arraysize, address, ...*)

Definition at line 112 of file token-stack.h.

6.13.2.8 #define DEFINE_MFG_TOKEN(*name, type, address, ...*)

Definition at line 116 of file token-stack.h.

6.13.2.9 #define CREATOR_STACK_NVDATA_VERSION

Definition at line 146 of file token-stack.h.

6.13.2.10 #define CREATOR_STACK_BOOT_COUNTER

Definition at line 147 of file token-stack.h.

6.13.2.11 #define CREATOR_STACK_NONCE_COUNTER

Definition at line 148 of file token-stack.h.

6.13.2.12 #define CREATOR_STACK_ANALYSIS_REBOOT

Definition at line 149 of file token-stack.h.

6.13.2.13 #define CREATOR_STACK_KEYS

Definition at line 150 of file token-stack.h.

6.13.2.14 #define CREATOR_STACK_NODE_DATA

Definition at line 151 of file token-stack.h.

6.13.2.15 #define CREATOR_STACK_CLASSIC_DATA

Definition at line 152 of file token-stack.h.

6.13.2.16 #define CREATOR_STACK_ALTERNATE_KEY

Definition at line 153 of file token-stack.h.

6.13.2.17 #define CREATOR_STACK_APS_FRAME_COUNTER

Definition at line 154 of file token-stack.h.

6.13.2.18 #define CREATOR_STACK_TRUST_CENTER

Definition at line 155 of file token-stack.h.

6.13.2.19 #define CREATOR_STACK_NETWORK_MANAGEMENT

Definition at line 156 of file token-stack.h.

6.13.2.20 #define CREATOR_STACK_PARENT_INFO

Definition at line 157 of file token-stack.h.

6.13.2.21 #define CREATOR_MULTI_NETWORK_STACK_KEYS

Definition at line 159 of file token-stack.h.

6.13.2.22 #define CREATOR_MULTI_NETWORK_STACK_NODE_DATA

Definition at line 160 of file token-stack.h.

6.13.2.23 #define CREATOR_MULTI_NETWORK_STACK_ALTERNATE_KEY

Definition at line 161 of file token-stack.h.

6.13.2.24 #define CREATOR_MULTI_NETWORK_STACK_TRUST_CENTER

Definition at line 162 of file token-stack.h.

6.13.2.25 #define CREATOR_MULTI_NETWORK_STACK_NETWORK_MANAGEMENT

Definition at line 163 of file token-stack.h.

6.13.2.26 #define CREATOR_MULTI_NETWORK_STACK_PARENT_INFO

Definition at line 164 of file token-stack.h.

6.13.2.27 #define CREATOR_MULTI_NETWORK_STACK_NONCE_COUNTER

Definition at line 167 of file token-stack.h.

6.13.2.28 #define CREATOR_STACK_BINDING_TABLE

Definition at line 170 of file token-stack.h.

6.13.2.29 #define CREATOR_STACK_CHILD_TABLE

Definition at line 171 of file token-stack.h.

6.13.2.30 #define CREATOR_STACK_KEY_TABLE

Definition at line 172 of file token-stack.h.

6.13.2.31 #define CREATOR_STACK_CERTIFICATE_TABLE

Definition at line 173 of file token-stack.h.

6.13.2.32 #define CREATOR_STACK_ZLL_DATA

Definition at line 174 of file token-stack.h.

6.13.2.33 #define CREATOR_STACK_ZLL_SECURITY

Definition at line 175 of file token-stack.h.

6.13.2.34 #define CURRENT_STACK_TOKEN_VERSION

The current version number of the stack tokens. MSB is the version, LSB is a complement.

Please see hal/micro/token.h for a more complete explanation.

Definition at line 209 of file token-stack.h.

6.14   ZigBee Device Object

Functions

- EmberStatus emberNetworkAddressRequest (EmberEUI64 target, boolean reportKids, int8u childStartIndex)
- EmberStatus emberIeeeAddressRequest (EmberNodeId target, boolean reportKids, int8u childStartIndex, EmberApsOption options)
- EmberStatus emberEnergyScanRequest (EmberNodeId target, int32u scanChannels, int8u scanDuration, int16u scanCount)
- EmberStatus emberSetNetworkManagerRequest (EmberNodeId networkManager, int32u activeChannels)
- EmberStatus emberChannelChangeRequest (int8u channel)
- EmberStatus emberSendDeviceAnnouncement (void)
- int8u emberGetLastStackZigDevRequestSequence (void)

6.14.1   Detailed Description

See zigbee-device-stack.h for source code.

6.14.2   Function Documentation

6.14.2.1   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.

6.14.2.2   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

6.14.2.3 EmberStatus emberEnergyScanRequest (EmberNodeId *target,* int32u *scanChannels,* int8u *scanDuration,* int16u *scanCount*)

Request that an energy scan be performed and its results returned. This request may only be sent by the current network manager and must be unicast, not broadcast.

Parameters

target	The network address of the node to perform the scan.
scanChannels	A mask of the channels to be scanned.
scanDuration	How long to scan on each channel. Allowed values are 0..5, with the scan times as specified by 802.15.4 (0 = 31ms, 1 = 46ms, 2 = 77 ms, 3 = 138ms, 4 = 261ms, 5 = 507ms).
scanCount	The number of scans to be performed on each channel (1 .. 8).

Returns

An EmberStatus value.

- EMBER_SUCCESS
- EMBER_NO_BUFFERS
- EMBER_NETWORK_DOWN
- EMBER_NETWORK_BUSY

6.14.2.4 EmberStatus emberSetNetworkManagerRequest (EmberNodeId *networkManager,* int32u *activeChannels*)

Broadcasts a request to set the identity of the network manager and the active channel mask. The mask is used when scanning for the network after missing a channel update.

Parameters

network-Manager	The network address of the network manager.
activeChannels	The new active channel mask.

Returns

An EmberStatus value.

- EMBER_SUCCESS
- EMBER_NO_BUFFERS

- EMBER_NETWORK_DOWN
- EMBER_NETWORK_BUSY

6.14.2.5 EmberStatus emberChannelChangeRequest (int8u *channel*)

Broadcasts a request to change the channel. This request may only be sent by the current network manager. There is a delay of several seconds from receipt of the broadcast to changing the channel, to allow time for the broadcast to propagate.

Parameters

channel	The channel to change to.

Returns

An EmberStatus value.
- EMBER_SUCCESS
- EMBER_NO_BUFFERS
- EMBER_NETWORK_DOWN
- EMBER_NETWORK_BUSY

6.14.2.6 EmberStatus emberSendDeviceAnnouncement (void)

Sends a broadcast for a ZDO Device announcement. Normally it is NOT required to call this as the stack automatically sends a device announcement during joining or rejoining, as per the spec. However if the device wishes to re-send its device announcement they can use this call.

Returns

An EmberStatus value.
- EMBER_SUCCESS
- EMBER_INVALID_CALL

6.14.2.7 int8u emberGetLastStackZigDevRequestSequence (void)

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

Returns

Last stack ZDO transaction sequence number used

6.15   Bootloader

Functions

- EmberStatus emberSendBootloadMessage (boolean broadcast, EmberEUI64 destEui64, EmberMessage-Buffer message)
- void emberIncomingBootloadMessageHandler (EmberEUI64 longId, EmberMessageBuffer message)
- void emberBootloadTransmitCompleteHandler (EmberMessageBuffer message, EmberStatus status)

6.15.1   Detailed Description

EmberZNet bootload API. See bootload.h for source code.

6.15.2   Function Documentation

6.15.2.1   EmberStatus emberSendBootloadMessage (boolean *broadcast,* EmberEUI64 *destEui64,* EmberMessageBuffer *message*)

Transmits the given bootload message to a neighboring node using a specific 802.15.4 header that allows the EmberZNet stack as well as the bootloader to recognize the message, but will not interfere with other ZigBee stacks.

Parameters

broadcast	If TRUE, the destination address and pan id are both set to the broadcast address.
destEui64	The EUI64 of the target node. Ignored if the broadcast field is set to TRUE.
message	The bootloader message to send.

Returns

EMBER_SUCCESS if the message was successfully submitted to the transmit queue, and EMBER_-ERR_FATAL otherwise.

6.15.2.2   void emberIncomingBootloadMessageHandler (EmberEUI64 *longId,* EmberMessageBuffer *message*)

A callback invoked by the EmberZNet stack when a bootload message is received. If the application includes emberIncomingBootloadMessageHandler(), it must define EMBER_APPLICATION_HAS_BO-OTLOAD_HANDLERS in its CONFIGURATION_HEADER.

Parameters

longId	The EUI64 of the sending node.
message	The bootload message that was sent.

6.15.2.3 void emberBootloadTransmitCompleteHandler (EmberMessageBuffer *message,* EmberStatus *status*)

A callback invoked by the EmberZNet stack when the MAC has finished transmitting a bootload message. If the application includes this callback, it must define EMBER_APPLICATION_HAS_BOOTLOAD_H-ANDLERS in its CONFIGURATION_HEADER.

Parameters

message	The message that was sent.
status	EMBER_SUCCESS if the transmission was successful, or EMBER_DELIVERY_FA-ILED if not.

6.16 Event Scheduling

Macros

- #define __EVENT_H__
- #define emberEventControlSetInactive(control)
- #define emberEventControlGetActive(control)
- #define emberEventControlSetActive(control)
- #define emberEventControlSetDelayMS(control, delay)
- #define emberEventControlSetDelayQS(control, delay)
- #define emberEventControlSetDelayMinutes(control, delay)
- #define emberEventControlGetRemainingMS(control)
- #define emberTaskEnableIdling(allow)
- #define emberMarkTaskActive(taskid)

Functions

- void emEventControlSetActive (EmberEventControl ∗event)
- void emEventControlSetDelayMS (EmberEventControl ∗event, int16u delay)
- void emEventControlSetDelayQS (EmberEventControl ∗event, int16u delay)
- void emEventControlSetDelayMinutes (EmberEventControl ∗event, int16u delay)
- int32u emEventControlGetRemainingMS (EmberEventControl ∗event)
- void emberRunEvents (EmberEventData ∗events)
- void emberRunTask (EmberTaskId taskid)
- int32u emberMsToNextEvent (EmberEventData ∗events, int32u maxMs)
- int32u emberMsToNextEventExtended (EmberEventData ∗events, int32u maxMs, int8u ∗return-Index)
- int32u emberMsToNextStackEvent (void)
- EmberTaskId emberTaskInit (EmberEventData ∗events)
- boolean emberMarkTaskIdle (EmberTaskId taskid)
- void emTaskEnableIdling (boolean allow)
- void emMarkTaskActive (EmberTaskId taskid)

6.16.1 Detailed Description

These macros implement an event abstraction that allows the application to schedule code to run after some specified time interval. An event consists of a procedure to be called at some point in the future and a control object that determines the procedure should be called. Events are also useful for when an ISR needs to initiate an action that should run outside of ISR context.

See event.h for source code.

Note that while not required, it is recommended that the event-handling procedure explicitly define the recurrence of the next event, either by rescheduling it via some kind of *emberEventControlSetDelayXX()* call or by deactivating it via a call to emberEventControlSetInactive(). In cases where the handler does not explicitly reschedule or cancel the event, the default behavior of the event control system is to keep the event immediately active as if the handler function had called emberEventControlSetActive(someEvent) or emberEventControlSetDelayMS(someEvent, EMBER_EVENT_ZERO_DELAY)

The time units are all in 'binary' One 'millisecond' is 1/1024 of a second. One 'quarter second' is 256 milliseconds (which happens to work out to be the same as a real quarter second) One 'minute' is 65536

(0x10000) milliseconds (64 ∗ 4 'quarter seconds'). The accuracy of the base 'millisecond' depends on your timer source.

Following are some brief usage examples.

```c
EmberEventControl delayEvent;
EmberEventControl signalEvent;
EmberEventControl periodicEvent;

void delayEventHandler(void)
{
  // Disable this event until its next use.
  emberEventControlSetInactive(delayEvent);
}

void signalEventHandler(void)
{
  // Disable this event until its next use.
  emberEventControlSetInactive(signalEvent);

  // Sometimes we need to do something 100 ms later.
  if (somethingIsExpected)
    emberEventControlSetDelayMS(delayEvent, 100);
}

void periodicEventHandler(void)
{
  emberEventControlSetDelayQS(periodicEvent, 4);
}

void someIsr(void)
{
  // Set the signal event to run at the first opportunity.
  emberEventControlSetActive(signalEvent);
}

// Put the controls and handlers in an array.  They will be run in
// this order.
EmberEventData events[] =
 {
   { &delayEvent,    delayEventHandler },
   { &signalEvent,   signalEentHandler },
   { &periodicEvent, periodicEventHandler },
   { NULL, NULL }                              // terminator
 };

void main(void)
{
  // Cause the periodic event to occur once a second.
  emberEventControlSetDelayQS(periodicEvent, 4);

  while (TRUE) {
    emberRunEvents(events);
  }
}
```

6.16.2 Macro Definition Documentation

6.16.2.1 #define __EVENT_H__

Definition at line 106 of file event.h.

6.16.2.2 #define emberEventControlSetInactive(control)

Sets this EmberEventControl as inactive (no pending event).

Definition at line 110 of file event.h.

6.16.2.3   #define emberEventControlGetActive(control)

Returns TRUE if the event is active, FALSE otherwise.

Definition at line 115 of file event.h.

6.16.2.4   #define emberEventControlSetActive(control)

Sets this EmberEventControl to run at the next available opportunity.

Definition at line 121 of file event.h.

6.16.2.5   #define emberEventControlSetDelayMS(control, delay)

Sets this EmberEventControl to run "delay" milliseconds in the future.

Definition at line 131 of file event.h.

6.16.2.6   #define emberEventControlSetDelayQS(control, delay)

Sets this EmberEventControl to run "delay" quarter seconds in the future. The 'quarter seconds' are actually 256 milliseconds long.

Definition at line 141 of file event.h.

6.16.2.7   #define emberEventControlSetDelayMinutes(control, delay)

Sets this EmberEventControl to run "delay" minutes in the future. The 'minutes' are actually 65536 (0x10000) milliseconds long.

Definition at line 152 of file event.h.

6.16.2.8   #define emberEventControlGetRemainingMS(control)

Returns The amount of milliseconds remaining before the event is scheduled to run. If the event is inactive, MAX_INT32U_VALUE is returned.

Definition at line 163 of file event.h.

6.16.2.9   #define emberTaskEnableIdling(allow)

Call this to indicate that an application supports processor idling.

Definition at line 229 of file event.h.

6.16.2.10   #define emberMarkTaskActive(taskid)

Indicates that a task has something to do, so the CPU should not be idled until emberMarkTaskIdle is next called on this task.

Definition at line 237 of file event.h.

6.16.3 Function Documentation

6.16.3.1 void emEventControlSetActive (EmberEventControl ∗ *event*)

Sets this EmberEventControl to run at the next available opportunity.

6.16.3.2 void emEventControlSetDelayMS (EmberEventControl ∗ *event,* int16u *delay*)

Sets this EmberEventControl to run "delay" milliseconds in the future.

6.16.3.3 void emEventControlSetDelayQS (EmberEventControl ∗ *event,* int16u *delay*)

Sets this EmberEventControl to run "delay" quarter seconds in the future. The 'quarter seconds' are actually 256 milliseconds long.

6.16.3.4 void emEventControlSetDelayMinutes (EmberEventControl ∗ *event,* int16u *delay*)

Sets this EmberEventControl to run "delay" minutes in the future. The 'minutes' are actually 65536 (0x10000) milliseconds long.

6.16.3.5 int32u emEventControlGetRemainingMS (EmberEventControl ∗ *event*)

Returns The amount of milliseconds remaining before the event is scheduled to run. If the event is inactive, MAX_INT32U_VALUE is returned.

6.16.3.6 void emberRunEvents (EmberEventData ∗ *events*)

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.

6.16.3.7 void emberRunTask (EmberTaskId *taskid*)

If an application has initialized a task via emberTaskInit, to run the events associated with that task, it should could emberRunTask() instead of emberRunEvents().

6.16.3.8 int32u emberMsToNextEvent (EmberEventData ∗ *events,* int32u *maxMs*)

Returns the number of milliseconds before the next event is scheduled to expire, or maxMs if no event is scheduled to expire within that time. NOTE: If any events are modified within an interrupt, in order to guarantee the accuracy of this API, it must be called with interrupts disabled or from within an ATOMIC() block.

6.16.3.9 int32u emberMsToNextEventExtended (EmberEventData ∗ *events,* int32u *maxMs,* int8u ∗ *returnIndex*)

This function does the same as emberMsToNextEvent() with the following addition. If the returnIndex is non-NULL, it will set the value pointed to by the pointer to be equal to the index of the event that is ready

to fire next. If no events are active, then it returns 0xFF.

6.16.3.10   int32u emberMsToNextStackEvent (void)

Returns the number of milliseconds before the next stack event is scheduled to expire.

6.16.3.11   EmberTaskId emberTaskInit (EmberEventData ∗ *events*)

Initializes a task to be used for managing events and processor idling state. Returns the EmberTaskId which represents the newly created task.

6.16.3.12   boolean emberMarkTaskIdle (EmberTaskId *taskid*)

Indicates that a task has nothing to do (unless any events are pending) and that it would be safe to idle the CPU if all other tasks also have nothing to do. This API should always be called with interrupts disabled. It will forcibly re-enable interrupts before returning Returns TRUE if the processor was idled, FALSE if idling wasn't permitted because some other task has something to do.

6.16.3.13   void emTaskEnableIdling (boolean *allow*)

6.16.3.14   void emMarkTaskActive (EmberTaskId *taskid*)

6.17 Manufacturing and Functional Test Library

Functions

- EmberStatus mfglibStart (void(∗mfglibRxCallback)(int8u ∗packet, int8u linkQuality, int8s rssi))
- EmberStatus mfglibEnd (void)
- EmberStatus mfglibStartTone (void)
- EmberStatus mfglibStopTone (void)
- EmberStatus mfglibStartStream (void)
- EmberStatus mfglibStopStream (void)
- EmberStatus mfglibSendPacket (int8u ∗packet, int16u repeat)
- EmberStatus mfglibSetChannel (int8u chan)
- int8u mfglibGetChannel (void)
- EmberStatus mfglibSetPower (int16u txPowerMode, int8s power)
- int8s mfglibGetPower (void)
- void mfglibSetSynOffset (int8s synOffset)
- int8s mfglibGetSynOffset (void)
- void mfglibTestContModCal (int8u channel, int32u duration)

6.17.1 Detailed Description

This is a manufacturing and functional test library for testing and verifying the RF component of products at manufacture time. See mfglib.h for source code.

Developers can optionally include this library in their application code. The goal is that in most cases, this will eliminate the need for developers to load multiple images into their hardware at manufacturing time.

This library can optionally be compiled into the developer's production code and run at manufacturing time. Any interface to the library is handled by the application.

This library cannot assist in hardware start up.

Many functions in this file return an EmberStatus value. See error-def.h for definitions of all EmberStatus return values.

6.17.2 Function Documentation

6.17.2.1 EmberStatus mfglibStart (void(∗)(int8u ∗packet, int8u linkQuality, int8s rssi) *mfglibRxCallback*)

Activates use of mfglib test routines and enables the radio receiver to report packets it receives to the caller-specified ::mfglibRxCallback() routine.

It is legal to pass in a NULL. These packets will not be passed up with a CRC failure. The first byte of the packet in the callback is the length. All other functions will return an error until mfglibStart() has been called.

Application Usage:

Use this function to enter test mode.

Note: This function should only be called shortly after initialization and prior to forming or joining a network.

Parameters

mfglibRx- *Callback*	Function pointer to callback routine invoked whenever a valid packet is received. emberTick() must be called routinely for this callback to function correctly.

Returns

One of the following:

- EMBER_SUCCESS if the mfg test mode has been enabled.
- EMBER_ERR_FATAL if the mfg test mode is not available.

6.17.2.2   EmberStatus mfglibEnd (void)

Deactivates use of Manufacturing and Functional Test Library test routines.

This restores the hardware to the state it was in prior to mfglibStart() and stops receiving packets started by mfglibStart() at the same time.

Application Usage:

Use this function to exit the mfg test mode.

Note: It may be desirable to also reboot after use of manufacturing mode to ensure all application state is properly re-initialized.

Returns

One of the following:

- EMBER_SUCCESS if the mfg test mode has been exited.
- EMBER_ERR_FATAL if the mfg test mode cannot be exited.

6.17.2.3   EmberStatus mfglibStartTone (void)

Starts transmitting the tone feature of the radio.

In this mode, the radio will transmit an unmodulated tone on the currently set channel and power level. Upon successful return, the tone will be transmitting. To stop transmitting a tone, the application must call mfglibStopTone(), allowing it the flexibility to determine its own criteria for tone duration, such as time, event, and so on.

Application Usage:

Use this function to transmit a tone.

Returns

One of the following:

- EMBER_SUCCESS if the transmit tone has started.
- EMBER_ERR_FATAL if the tone cannot be started.

6.17.2.4   EmberStatus mfglibStopTone (void)

Stops transmitting a tone started by mfglibStartTone().

Application Usage:

Use this function to stop transmitting a tone.

Returns

One of the following:
- EMBER_SUCCESS if the transmit tone has stopped.
- EMBER_ERR_FATAL if the tone cannot be stopped.

6.17.2.5   EmberStatus mfglibStartStream (void)

Starts transmitting a random stream of characters. This is so that the radio modulation can be measured.

Application Usage:

Use this function to enable the measurement of radio modulation.

Returns

One of the following:
- EMBER_SUCCESS if the transmit stream has started.
- EMBER_ERR_FATAL if the stream cannot be started.

6.17.2.6   EmberStatus mfglibStopStream (void)

Stops transmitting a random stream of characters started by mfglibStartStream().

Application Usage:

Use this function to end the measurement of radio modulation.

Returns

One of the following:
- EMBER_SUCCESS if the transmit stream has stopped.
- EMBER_ERR_FATAL if the stream cannot be stopped.

6.17.2.7 EmberStatus mfglibSendPacket (int8u ∗ *packet,* int16u *repeat*)

Sends a single packet, (repeat + 1) times.

Application Usage:

Use this function to send raw data. Note that *packet* array must be word-aligned (begin at even address), such that *((((int16u)packet) & 1) == 0)* holds true. (This is generally done by either declaring *packet* as a local variable or putting it in a global declaration immediately following the declaration of an int16u.)

Parameters

packet	Packet to be sent. First byte of the packet is always the length byte, whose value does not include itself but does include the 16-bit CRC in the length calculation. The CRC gets appended automatically by the radio as it transmits the packet, so the host does not need to provide this as part of packetContents. The total length of packet contents (Length Byte+1) going out the radio should not be >128 or <6 bytes. Note that the packet array should not include the CRC, as this appended by the radio automatically.
repeat	Number of times to repeat sending the packet after having been sent once. A value of 0 means send once and don't repeat.

Returns

One of the following:

- EMBER_SUCCESS if the packet was sent.
- EMBER_ERR_FATAL if the mfg test mode is not available or TONE or STREAM test is running.

6.17.2.8 EmberStatus mfglibSetChannel (int8u *chan*)

Selects the radio channel. The channel range is from 11 to 26.

Customers can set any valid channel they want. Calibration occurs if this is the first time after power up.

Application Usage:

Use this function to change channels.

Parameters

chan	Valid values depend upon the radio used.

Returns

One of the following:

- EMBER_SUCCESS if the channel has been set.
- ::EMBER_ERROR_INVALID_CHANNEL if the channel requested is invalid.
- EMBER_ERR_FATAL if the mfg test mode is not available or TONE or STREAM test is running.

6.17.2.9 int8u mfglibGetChannel (void)

Returns the current radio channel, as previously set via mfglibSetChannel().

Application Usage:

Use this function to get current channel.

Returns

Current channel.

6.17.2.10 EmberStatus mfglibSetPower (int16u *txPowerMode,* int8s *power*)

First select the transmit power mode, and then include a method for selecting the radio transmit power.

Valid power settings depend upon the specific radio in use. Ember radios have discrete power settings, and then requested power is rounded to a valid power setting. The actual power output is available to the caller via mfglibGetPower().

Application Usage:

Use this function to adjust the transmit power.

Parameters

txPowerMode	boost mode or external PA.
power	Power in units of dBm, which can be negative.

Returns

One of the following:
- EMBER_SUCCESS if the power has been set.
- ::EMBER_ERROR_INVALID_POWER if the power requested is invalid.
- EMBER_ERR_FATAL if the mfg test mode is not available or TONE or STREAM test is running.

6.17.2.11 int8s mfglibGetPower (void)

returns the current radio power setting as previously set via mfglibSetPower().

Application Usage:

Use this function to get current power setting.

Returns

current power setting.

6.17.2.12   void mfglibSetSynOffset (int8s *synOffset*)

set the synth offset in 11.7kHz steps. This function does NOT write the new synth offset to the token, it only changes it in memory. It can be changed as many times as you like, and the setting will be lost when a reset occurs. The value will survive deep sleep, but will not survive a reset, thus it will not take effect in the bootloader. If you would like it to be permanent (and accessible to the bootloader), you must write the TOKEN_MFG_SYNTH_FREQ_OFFSET token using the token API or em3xx_load –patch.

Application Usage:

Use this function to compensate for tolerances in the crystal oscillator or capacitors. This function does not effect a permanent change; once you have found the offset you want, you must write it to a token using the token API for it to be permanent.

Parameters

synOffset	the number of 11.7kHz steps to offset the carrier frequency (may be negative)

6.17.2.13   int8s mfglibGetSynOffset (void)

get the current synth offset in 11.7kHz steps. see mfglibSetSynOffset() for details

Returns

the synth offset in 11.7kHz steps

6.17.2.14   void mfglibTestContModCal (int8u *channel,* int32u *duration*)

Run mod DAC calibration on the given channel for the given amount of time.

If the duration argument == 0, this test will run forever (until the chip is reset).

Application Usage:

Use this function to run the active transmit part of mod DAC calibration.

Parameters

channel	Selects the channel to transmit on.
duration	Duration in ms, 0 == infinite.

Returns

None.

6.18 Debugging Utilities

Macros

- #define NO_DEBUG
- #define BASIC_DEBUG
- #define FULL_DEBUG
- #define emberDebugInit(port)

Functions

- void emberDebugAssert (PGM_P filename, int linenumber)
- void emberDebugMemoryDump (int8u ∗start, int8u ∗end)
- void emberDebugBinaryPrintf (PGM_P formatString,...)
- void emDebugSendVuartMessage (int8u ∗buff, int8u len)
- void emberDebugError (EmberStatus code)
- boolean emberDebugReportOff (void)
- void emberDebugReportRestore (boolean state)
- void emberDebugPrintf (PGM_P formatString,...)

6.18.1 Detailed Description

EmberZNet debugging utilities. See ember-debug.h for source code.

6.18.2 Macro Definition Documentation

6.18.2.1 #define NO_DEBUG

Definition at line 20 of file ember-debug.h.

6.18.2.2 #define BASIC_DEBUG

Definition at line 21 of file ember-debug.h.

6.18.2.3 #define FULL_DEBUG

Definition at line 22 of file ember-debug.h.

6.18.2.4 #define emberDebugInit(*port*)

This function is obsolete and no longer required to initialize the debug system.

Parameters

port	Ignored because the port used for debug communication is automatically determined for each platform.

Definition at line 30 of file ember-debug.h.

6.18.3   Function Documentation

6.18.3.1   void emberDebugAssert (PGM_P *filename,* int *linenumber*)

Prints the filename and line number to the debug serial port.

Parameters

filename	The name of the file where the assert occurred.
linenumber	The line number in the file where the assert occurred.

6.18.3.2   void emberDebugMemoryDump (int8u ∗ *start,* int8u ∗ *end*)

Prints the contents of RAM to the debug serial port.

Parameters

start	The start address of the block of RAM to dump.
end	The end address of the block of RAM to dump (address of the last byte).

6.18.3.3   void emberDebugBinaryPrintf (PGM_P *formatString,* ...)

Prints binary data to the debug channel.

This function does not use the normal printf format conventions. To print text debug messages, use ember-DebugPrintf(). The format string must contain only these conversion specification characters:

- B - int8u value.

- W - int16u value, printed least significant byte first.

- D - int32u value, printed least significant byte first.

- F - pointer to null terminated string in Flash (PGM_P).

- xxxp - pointer to RAM, length is xxx (max 255).

- lp - pointer to RAM, length is int8u argument.

- xxxf - pointer to Flash (PGM_P), length is xxx (max 255).

- lf - pointer to Flash (PGM_P), length is int8u argument.

- b - EmberMessageBuffer.

Examples:

```
emberDebugBinaryPrintf("BWD", status, panId, channelMask)
    ;
emberDebugBinaryPrintf("F8p", "string example", eui64);
emberDebugBinaryPrintf("lp64fb", length, bytes, dataTable
    , buffer);
```

Parameters

formatString	A string of conversion specification characters describing the arguments to be printed.
...	The arguments to be printed.

6.18.3.4   void emDebugSendVuartMessage (int8u ∗ *buff,* int8u *len*)

internal debug command used by the HAL to send vuart data out the the debug channel

Parameters

buff	pointer to the data to send
len	lenght of the data to send

6.18.3.5   void emberDebugError (EmberStatus *code*)

Prints an EmberStatus return code to the serial port.

Parameters

code	The EmberStatus code to print.

6.18.3.6   boolean emberDebugReportOff (void)

Turns off all debug output.

Returns

The current state (TRUE for on, FALSE for off).

6.18.3.7   void emberDebugReportRestore (boolean *state*)

Restores the state of the debug output.

Parameters

state	The state returned from emberDebugReportOff(). This is done so that debug output is not blindly turned on.

6.18.3.8   void emberDebugPrintf (PGM_P *formatString, ...*)

Prints text debug messages.

Parameters

formatString	Takes the following:

%%	Percent sign
%c	Single-byte char
%s	RAM string

%p	Flash string (does not follow the printf standard)
%u	Two-byte unsigned decimal
%d	Two-byte signed decimal
%x, %%2x, %%4x	1-, 2-, 4-byte hex value (always 0 padded; does not follow the printf standard)

6.19   Hardware Abstraction Layer (HAL) API Reference

Modules

- Common Microcontroller Functions
- Token Access
- Sample APIs for Peripheral Access
- System Timer Control
- HAL Configuration
- HAL Utilities
- Bootloader Interfaces
- Custom Bootloader HAL

6.19.1   Detailed Description

```
<center><h1>EM35x Microprocessors</h1></center>
```

HAL function names have the following prefix conventions:

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

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

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

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

See also hal.h.

6.20   Common Microcontroller Functions

Macros

- #define halGetEm2xxResetInfo()

Functions

- void halStackProcessBootCount (void)
- int8u halGetResetInfo (void)
- PGM_P halGetResetString (void)
- void halInternalAssertFailed (PGM_P filename, int linenumber)
- void halInternalSysReset (int16u extendedCause)
- int16u halGetExtendedResetInfo (void)
- PGM_P halGetExtendedResetString (void)

Vector Table Index Definitions

These are numerical definitions for vector table. Indices 0 through 15 are Cortex-M3 standard exception vectors and indices 16 through 32 are EM3XX specific interrupt vectors.

- #define STACK_VECTOR_INDEX
- #define RESET_VECTOR_INDEX
- #define NMI_VECTOR_INDEX
- #define HARD_FAULT_VECTOR_INDEX
- #define MEMORY_FAULT_VECTOR_INDEX
- #define BUS_FAULT_VECTOR_INDEX
- #define USAGE_FAULT_VECTOR_INDEX
- #define RESERVED07_VECTOR_INDEX
- #define RESERVED08_VECTOR_INDEX
- #define RESERVED09_VECTOR_INDEX
- #define RESERVED10_VECTOR_INDEX
- #define SVCALL_VECTOR_INDEX
- #define DEBUG_MONITOR_VECTOR_INDEX
- #define RESERVED13_VECTOR_INDEX
- #define PENDSV_VECTOR_INDEX
- #define SYSTICK_VECTOR_INDEX
- #define TIMER1_VECTOR_INDEX
- #define TIMER2_VECTOR_INDEX
- #define MANAGEMENT_VECTOR_INDEX
- #define BASEBAND_VECTOR_INDEX
- #define SLEEP_TIMER_VECTOR_INDEX
- #define SC1_VECTOR_INDEX
- #define SC2_VECTOR_INDEX
- #define SECURITY_VECTOR_INDEX
- #define MAC_TIMER_VECTOR_INDEX
- #define MAC_TX_VECTOR_INDEX
- #define MAC_RX_VECTOR_INDEX
- #define ADC_VECTOR_INDEX

- #define IRQA_VECTOR_INDEX
- #define IRQB_VECTOR_INDEX
- #define IRQC_VECTOR_INDEX
- #define IRQD_VECTOR_INDEX
- #define DEBUG_VECTOR_INDEX
- #define VECTOR_TABLE_LENGTH

6.20.1   Detailed Description

Many of the supplied example applications use these microcontroller functions. See hal/micro/micro.h for source code.

Note

> The term SFD refers to the Start Frame Delimiter.

See also hal/micro/cortexm3/micro.h for source code.

6.20.2   Macro Definition Documentation

6.20.2.1   #define halGetEm2xxResetInfo(   )

Calls halGetExtendedResetInfo() and translates the EM35x reset code to the corresponding value used by the EM2XX HAL. EM35x reset codes not present in the EM2XX are returned after being OR'ed with 0x80.

Application Usage:

> Used by the EZSP host as a platform-independent NCP reset code.

Returns

> The EM2XX reset code, or a new EM3xx code if B7 is set.

Definition at line 167 of file micro.h.

6.20.2.2   #define STACK_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 111 of file cortexm3/micro.h.

6.20.2.3   #define RESET_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 112 of file cortexm3/micro.h.

6.20.2.4   #define NMI_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 113 of file cortexm3/micro.h.

6.20.2.5 #define HARD_FAULT_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 114 of file cortexm3/micro.h.

6.20.2.6 #define MEMORY_FAULT_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 115 of file cortexm3/micro.h.

6.20.2.7 #define BUS_FAULT_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 116 of file cortexm3/micro.h.

6.20.2.8 #define USAGE_FAULT_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 117 of file cortexm3/micro.h.

6.20.2.9 #define RESERVED07_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 118 of file cortexm3/micro.h.

6.20.2.10 #define RESERVED08_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 119 of file cortexm3/micro.h.

6.20.2.11 #define RESERVED09_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 120 of file cortexm3/micro.h.

6.20.2.12 #define RESERVED10_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 121 of file cortexm3/micro.h.

6.20.2.13 #define SVCALL_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 122 of file cortexm3/micro.h.

6.20.2.14   #define DEBUG_MONITOR_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 123 of file cortexm3/micro.h.

6.20.2.15   #define RESERVED13_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 124 of file cortexm3/micro.h.

6.20.2.16   #define PENDSV_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 125 of file cortexm3/micro.h.

6.20.2.17   #define SYSTICK_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 126 of file cortexm3/micro.h.

6.20.2.18   #define TIMER1_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 127 of file cortexm3/micro.h.

6.20.2.19   #define TIMER2_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 128 of file cortexm3/micro.h.

6.20.2.20   #define MANAGEMENT_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 129 of file cortexm3/micro.h.

6.20.2.21   #define BASEBAND_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 130 of file cortexm3/micro.h.

6.20.2.22   #define SLEEP_TIMER_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 131 of file cortexm3/micro.h.

6.20.2.23   #define SC1_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 132 of file cortexm3/micro.h.

6.20.2.24   #define SC2_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 133 of file cortexm3/micro.h.

6.20.2.25   #define SECURITY_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 134 of file cortexm3/micro.h.

6.20.2.26   #define MAC_TIMER_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 135 of file cortexm3/micro.h.

6.20.2.27   #define MAC_TX_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 136 of file cortexm3/micro.h.

6.20.2.28   #define MAC_RX_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 137 of file cortexm3/micro.h.

6.20.2.29   #define ADC_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 138 of file cortexm3/micro.h.

6.20.2.30   #define IRQA_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 139 of file cortexm3/micro.h.

6.20.2.31   #define IRQB_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 140 of file cortexm3/micro.h.

6.20.2.32 #define IRQC_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 141 of file cortexm3/micro.h.

6.20.2.33 #define IRQD_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 142 of file cortexm3/micro.h.

6.20.2.34 #define DEBUG_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 143 of file cortexm3/micro.h.

6.20.2.35 #define VECTOR_TABLE_LENGTH

Number of vectors.

Definition at line 147 of file cortexm3/micro.h.

6.20.3 Function Documentation

6.20.3.1 void halStackProcessBootCount (void)

Called from emberInit and provides a means for the HAL to increment a boot counter, most commonly in non-volatile memory.

This is useful while debugging to determine the number of resets that might be seen over a period of time. Exposing this functionality allows the application to disable or alter processing of the boot counter if, for example, the application is expecting a lot of resets that could wear out non-volatile storage or some

EmberStack Usage:

Called from emberInit only as helpful debugging information. This should be left enabled by default, but this function can also be reduced to a simple return statement if boot counting is not desired.

6.20.3.2 int8u halGetResetInfo (void)

Gets information about what caused the microcontroller to reset.

Returns

A code identifying the cause of the reset.

6.20.3.3 PGM_P halGetResetString (void)

Calls halGetResetInfo() and supplies a string describing it.

Application Usage:

Useful for diagnostic printing of text just after program initialization.

Returns

A pointer to a program space string.

6.20.3.4 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.

6.20.3.5 void halInternalSysReset (int16u *extendedCause*)

Records the specified reset cause then forces a reboot.

6.20.3.6 int16u halGetExtendedResetInfo (void)

Returns the Extended Reset Cause information.

Returns

A 16-bit code identifying the base and extended cause of the reset

6.20.3.7 PGM_P halGetExtendedResetString (void)

Calls halGetExtendedResetInfo() and supplies a string describing the extended cause of the reset. halGet-ResetString() should also be called to get the string for the base reset cause.

Application Usage:

Useful for diagnostic printing of text just after program initialization.

Returns

A pointer to a program space string.

6.21   Token Access

Modules

- Tokens
- Simulated EEPROM
- Simulated EEPROM 2

6.21.1   Detailed Description

The token system stores such non-volatile information as the manufacturing ID, channel number, transmit power, and various pieces of information that the application needs to be persistent between device power cycles. The token system is design to abstract implementation details and simplify interacting with differing non-volatile systems. The majority of tokens are stored in Simulated EEPROM (in Flash) where they can be rewritten. Manufacturing tokens are stored in dedicated regions of flash and are not designed to be rewritten.

Refer to the Tokens module for a detailed description of the token system.

Refer to the Simulated EEPROM module for a detailed description of the necessary support functions for Simulated EEPROM.

Refer to the Simulated EEPROM 2 module for a detailed description of the necessary support functions for Simulated EEPROM, version 2.

Refer to token-stack.h for stack token definitions.

Refer to token-manufacturing.h for manufaturing token definitions.

Note

Simulated EEPROM, version 2 is only supported on EM335x chips.

6.22 Tokens

Macros

- #define halCommonGetToken(data, token)
- #define halCommonGetMfgToken(data, token)
- #define halCommonGetIndexedToken(data, token, index)
- #define halCommonSetToken(token, data)
- #define halCommonSetIndexedToken(token, index, data)
- #define halCommonIncrementCounterToken(token)

Functions

- EmberStatus halStackInitTokens (void)

6.22.1 Detailed Description

There are three main types of tokens:

- **Manufacturing tokens:** Tokens that are set at the factory and must not be changed through software operations.

- **Stack-level tokens:** Tokens that can be changed via the appropriate stack API calls.

- **Application level tokens:** Tokens that can be set via the token system API calls in this file.

The token system API controls writing tokens to non-volatile data and reading tokens from non-volatile data. If an application wishes to use application specific normal tokens, it must do so by creating its own token header file similar to token-stack.h. The macro APPLICATION_TOKEN_HEADER should be defined to equal the name of the header file in which application tokens are defined. If an application wishes to use application specific manufacturing tokens, it must do so by creating its own manufacturing token header file similar to token-manufacturing.h. The macro APPLICATION_MFG_TOKEN_HEADER should be defined to equal the name of the header file in which manufacturing tokens are defined.

Because the token system is based on memory locations within non-volatile storage, the token information could become out of sync without some kind of version tracking. The two defines, CURRENT_MFG_-TOKEN_VERSION and CURRENT_STACK_TOKEN_VERSION, are used to make sure the stack stays in sync with the proper token set. If the application defines its own tokens, it is recommended that the application also define an application token to be a application version to ensure the application stays in sync with the proper token set.

The most general format of a token definition is:

```
#define CREATOR_name 16bit_value
#ifdef DEFINETYPES
   typedef data_type type
#endif //DEFINETYPES
#ifdef DEFINETOKENS
   DEFINE_*_TOKEN(name, type, ... ,defaults)
#endif //DEFINETOKENS
```

The defined CREATOR is used as a distinct identifier tag for the token. The CREATOR is necessary because the token name is defined differently depending on underlying implementation, so the CREAT-OR makes sure token definitions and data stay tagged and known. The only requirement on these creator definitions is that they all must be unique. A favorite method for picking creator codes is to use two ASCII

characters inorder to make the codes more memorable. The 'name' part of the `#define CREATOR-_name` must match the 'name' provided in the `DEFINE_*_TOKEN` because the token system uses this name to automatically link the two.

The typedef provides a convenient and efficient abstraction of the token data. Since some tokens are structs with multiple pieces of data inside of them, type defining the token type allows more efficient and readable local copies of the tokens throughout the code.

The typedef is wrapped with an `#ifdef DEFINETYPES` because the typdefs and token defs live in the same file, and DEFINETYPES is used to select only the typedefs when the file is included. Similarly, the `DEFINE_*_TOKEN` is wrapped with an `#ifdef DEFINETOKENS` as a method for selecting only the token definitions when the file is included.

The abstract definition, `DEFINE_*_TOKEN(name, type, ...  ,defaults)`, has seven possible complete definitions:

- `DEFINE_BASIC_TOKEN(name, type, ...)`

- `DEFINE_INDEXED_TOKEN(name, type, arraysize, ...)`

- `DEFINE_COUNTER_TOKEN(name, type, ...)`

- `DEFINE_MFG_TOKEN(name, type, address, ...)`

The three fields common to all `DEFINE_*_TOKEN` are:

name - The name of the token, which all information is tied to.

type - Type of the token which is the same as the typedef mentioned before.

... - The default value to which the token is set upon initialization.

Note

> The old DEFINE_FIXED* token definitions are no longer used. They remain defined for backwards compatibility. In current systems, the Simulated EEPROM is used for storing non-manufacturing tokens and the Simulated EEPROM intelligently manages where tokens are stored to provide wear leveling across the flash memory and increase the number of write cycles. Manufacturing tokens live at a fixed address, but they must use DEFINE_MFG_TOKEN so the token system knows they are manufacturing tokens.

DEFINE_BASIC_TOKEN is the simplest definition and will be used for the majority of tokens (tokens that are not indexed, not counters, and not manufacturing). Basic tokens are designed for data storage that is always accessed as a single element.

DEFINE_INDEXED_TOKEN should be used on tokens that look like arrays. For example, data storage that looks like:

```
int32u myData[5]
```

This example data storage can be a token with typedef of int32u and defined as INDEXED with arraysize of 5. The extra field in this token definition is: arraysize - The number of elements in the indexed token. Indexed tokens are designed for data storage that is logically grouped together, but elements are accessed individually.

DEFINE_COUNTER_TOKEN should be used on tokens that are simple numbers where the majority of operations on the token is to increment the count. The reason for using DEFINE_COUNTER_TOKEN instead of DEFINE_BASIC_TOKEN is the special support that the token system provides for incrementing counters. The function call `halCommonIncrementCounterToken()` only operates on counter

tokens and is more efficient in terms of speed, data compression, and write cyles for incrementing simple numbers in the token system.

DEFINE_MFG_TOKEN is a DEFINE_BASIC_TOKEN token at a specific address and the token is manufacturing data that is written only once. The major difference is this token is designated manufacturing, which means the token system treats it differently from stack or app tokens. Primarily, a manufacturing token is written only once and lives at a fixed address outside of the Simulated EEPROM system. Being a write once token, the token system will also aid in debugging by asserting if there is an attempt to write a manufacturing token.

Here is an example of two application tokens:

```
#define CREATOR_SENSOR_NAME        0x5354
#define CREATOR_SENSOR_PARAMETERS  0x5350
#ifdef DEFINETYPES
  typedef int8u tokTypeSensorName[10];
  typedef struct {
    int8u initValues[5];
    int8u reportInterval;
    int16u calibrationValue;
  } tokTypeSensorParameters;
#endif //DEFINETYPES
#ifdef DEFINETOKENS
  DEFINE_BASIC_TOKEN(SENSOR_NAME,
                     tokTypeSensorName,
                     {'U','N','A','M','E','D',' ',' ',' ',' '})
  DEFINE_BASIC_TOKEN(SENSOR_PARAMETERS,
                     tokTypeSensorParameters,
                     {{0x01,0x02,0x03,0x04,0x05},5,0x0000})
#endif //DEFINETOKENS
```

Here is an example of how to use the two application tokens:

```
{
  tokTypeSensorName sensor;
  tokTypeSensorParameters params;

  halCommonGetToken(&sensor, TOKEN_SENSOR_NAME);
  halCommonGetToken(&params, TOKEN_SENSOR_PARAMETERS);
  if(params.calibrationValue == 0xBEEF) {
    params.reportInterval = 5;
  }
  halCommonSetToken(TOKEN_SENSOR_PARAMETERS, &params);
}
```

See token-stack.h to see the default set of tokens and their values.

The nodetest utility app can be used for generic manipulation such as loading default token values, viewing tokens, and writing tokens. **The nodetest utility cannot work with customer defined application tokens or manufacturing tokens. Using the nodetest utility will erase customer defined application tokens in the Simulated EEPROM.**

The Simulated EEPROM will initialize tokens to their default values if the token does not yet exist, the token's creator code is changed, or the token's size changes.

Changing the number indexes in an INDEXED token will not alter existing entries. If the number of indexes is reduced, the entires that still fit in the token will retain their data and the entries that no longer fit will be erased. If the number of indexes is increased, the existing entries retain their data and the new entries are initialized to the token's defaults.

Further details on exact implementation can be found in code comments in token-stack.h file, the platform specific token-manufacturing.h file, the platform specific token.h file, and the platform specific token.c file.

Some functions in this file return an EmberStatus value. See error-def.h for definitions of all EmberStatus return values.

See hal/micro/token.h for source code.

6.22.2 Macro Definition Documentation

6.22.2.1 #define halCommonGetToken(*data, token*)

Macro that copies the token value from non-volatile storage into a RAM location. This macro can only be used with tokens that are defined using DEFINE_BASIC_TOKEN.

Note

> To better understand the parameters of this macro, refer to the example of token usage above.

Parameters

data	A pointer to where the token data should be placed.
token	The token name used in `DEFINE_*_TOKEN`, prepended with `TOKEN_`.

Definition at line 276 of file token.h.

6.22.2.2 #define halCommonGetMfgToken(*data, token*)

Macro that copies the token value from non-volatile storage into a RAM location. This macro can only be used with tokens that are defined using DEFINE_MFG_TOKEN.

Note

> To better understand the parameters of this macro, refer to the example of token usage above.

Parameters

data	A pointer to where the token data should be placed.
token	The token name used in `DEFINE_*_TOKEN`, prepended with `TOKEN_`.

Definition at line 291 of file token.h.

6.22.2.3 #define halCommonGetIndexedToken(*data, token, index*)

Macro that copies the token value from non-volatile storage into a RAM location. This macro can only be used with tokens that are defined using DEFINE_INDEXED_TOKEN.

Note

> To better understand the parameters of this macro, refer to the example of token usage above.

Parameters

data	A pointer to where the token data should be placed.
token	The token name used in `DEFINE_*_TOKEN`, prepended with `TOKEN_`.
index	The index to access in the indexed token.

Definition at line 307 of file token.h.

6.22.2.4   #define halCommonSetToken(*token, data*)

Macro that sets the value of a token in non-volatile storage. This macro can only be used with tokens that are defined using DEFINE_BASIC_TOKEN.

Note

> To better understand the parameters of this macro, refer to the example of token usage above.

Parameters

token	The token name used in `DEFINE_*_TOKEN`, prepended with `TOKEN_`.
data	A pointer to the data being written.

Definition at line 321 of file token.h.

6.22.2.5   #define halCommonSetIndexedToken(*token, index, data*)

Macro that sets the value of a token in non-volatile storage. This macro can only be used with tokens that are defined using DEFINE_INDEXED_TOKEN.

Note

> To better understand the parameters of this macro, refer to the example of token usage above.

Parameters

token	The token name used in `DEFINE_*_TOKEN`, prepended with `TOKEN_`.
index	The index to access in the indexed token.
data	A pointer to where the token data should be placed.

Definition at line 338 of file token.h.

6.22.2.6   #define halCommonIncrementCounterToken(*token*)

Macro that increments the value of a token that is a counter. This macro can only be used with tokens that are defined using either DEFINE_COUNTER_TOKEN.

Note

> To better understand the parameters of this macro, refer to the example of token usage above.

Parameters

token	The token name used in `DEFINE_*_TOKEN`, prepended with `TOKEN_`.

Definition at line 351 of file token.h.

6.22.3   Function Documentation

6.22.3.1 EmberStatus halStackInitTokens (void)

Initializes and enables the token system. Checks if the manufacturing and stack non-volatile data versions are correct.

Returns

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

6.23 Simulated EEPROM

6.24   Simulated EEPROM 2

6.25   Sample APIs for Peripheral Access

Modules

- Serial UART Communication
- Button Control
- Buzzer Control
- LED Control
- Flash Memory Control

6.25.1   Detailed Description

These are sample API for accessing peripherals and can be modified as needed for your applications.

6.26 Serial UART Communication

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 Mode Definitions

These are numerical definitions for the possible serial modes so that code can test for the one being used. There may be additional modes defined in the micro-specific micro.h.

- #define EMBER_SERIAL_UNUSED
- #define EMBER_SERIAL_FIFO
- #define EMBER_SERIAL_BUFFER
- #define EMBER_SERIAL_LOWLEVEL

FIFO Utility Macros

These macros manipulate the FIFO queue data structures to add and remove data.

- #define FIFO_ENQUEUE(queue, data, size)
- #define FIFO_DEQUEUE(queue, size)

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.

- EmberStatus halInternalUartInit (int8u port, SerialBaudRate rate, SerialParity parity, int8u stopBits)
- void halInternalPowerDownUart (void)
- void halInternalPowerUpUart (void)
- void halInternalStartUartTx (int8u port)
- void halInternalStopUartTx (int8u port)
- EmberStatus halInternalForceWriteUartData (int8u port, int8u ∗data, int8u length)
- EmberStatus halInternalForceReadUartByte (int8u port, int8u ∗dataByte)
- void halInternalWaitUartTxComplete (int8u port)
- void halInternalRestartUart (void)
- boolean halInternalUart1FlowControlRxIsEnabled (void)
- boolean halInternalUart1XonRefreshDone (void)
- boolean halInternalUart1TxIsIdle (void)
- #define halInternalUartFlowControl(port)
- #define halInternalUartRxPump(port)

Buffered Serial Utility APIs

The higher-level serial code implements these APIs, which the HAL uses to deal with buffered serial output.

- void emSerialBufferNextMessageIsr (EmSerialBufferQueue ∗q)
- void emSerialBufferNextBlockIsr (EmSerialBufferQueue ∗q, int8u port)

Virtual UART API

API used by the stack in debug builds to receive data arriving over the virtual UART.

- void halStackReceiveVuartMessage (int8u ∗data, int8u length)

6.26.1   Detailed Description

This API contains the HAL interfaces that applications 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 hal/micro/serial.h for source code.

6.26.2   Macro Definition Documentation

6.26.2.1   #define EMBER_SERIAL_UNUSED

A numerical definition for a possible serial mode the code can test for.

Definition at line 55 of file hal/micro/serial.h.

6.26.2.2   #define EMBER_SERIAL_FIFO

A numerical definition for a possible serial mode the code can test for.

Definition at line 56 of file hal/micro/serial.h.

6.26.2.3   #define EMBER_SERIAL_BUFFER

A numerical definition for a possible serial mode the code can test for.

Definition at line 57 of file hal/micro/serial.h.

6.26.2.4   #define EMBER_SERIAL_LOWLEVEL

A numerical definition for a possible serial mode the code can test for.

Definition at line 58 of file hal/micro/serial.h.

6.26.2.5 #define FIFO_ENQUEUE(*queue, data, size*)

Macro that enqueues a byte of data in a FIFO queue.

Parameters

queue	Pointer to the FIFO queue.
data	Data byte to be enqueued.
size	Size used to control the wrap-around of the FIFO pointers.

Definition at line 233 of file hal/micro/serial.h.

6.26.2.6 #define FIFO_DEQUEUE(*queue, size*)

Macro that de-queues a byte of data from a FIFO queue.

Parameters

queue	Pointer to the FIFO queue.
size	Size used to control the wrap-around of the FIFO pointers.

Definition at line 255 of file hal/micro/serial.h.

6.26.2.7 #define halInternalUartFlowControl(*port*)

This function is used in FIFO mode when flow control is enabled. It is called from emberSerialReadByte(), and based on the number of bytes used in the uart receive queue, decides when to tell the host it may resume transmission.

Parameters

port	Serial port number (0 or 1). (Does nothing for port 0)

Definition at line 424 of file hal/micro/serial.h.

6.26.2.8 #define halInternalUartRxPump(*port*)

This function exists only in Buffer Mode on the EM2xx and in software UART (SOFTUART) mode on the EM3xx. This function is called by emberSerialReadByte(). It is responsible for maintaining synchronization between the emSerialRxQueue and the UART DMA.

Parameters

port	Serial port number (0 or 1).

Definition at line 437 of file hal/micro/serial.h.

6.26.3 Enumeration Type Documentation

6.26.3.1 enum SerialBaudRate

Assign numerical values for variables that hold Baud Rate parameters.

Enumerator:

DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD
DEFINE_BAUD

Definition at line 269 of file hal/micro/serial.h.

6.26.3.2 enum NameOfType

Assign numerical values for the types of parity. Use for variables that hold Parity parameters.

Enumerator:

DEFINE_PARITY
DEFINE_PARITY
DEFINE_PARITY

Definition at line 309 of file hal/micro/serial.h.

6.26.4 Function Documentation

6.26.4.1 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.

6.26.4.2 void halInternalPowerDownUart (void)

This function is typically called by ::halPowerDown() and it is responsible for performing all the work internal to the UART needed to stop the UART before a sleep cycle.

6.26.4.3 void halInternalPowerUpUart (void)

This function is typically called by ::halPowerUp() and it is responsible for performing all the work internal to the UART needed to restart the UART after a sleep cycle.

6.26.4.4 void halInternalStartUartTx (int8u *port*)

Called by serial code whenever anything is queued for transmission to start any interrupt-driven transmission. May be called when transmission is already in progess.

Parameters

port	Serial port number (0 or 1).

6.26.4.5 void halInternalStopUartTx (int8u *port*)

Called by serial code to stop any interrupt-driven serial transmission currently in progress.

Parameters

port	Serial port number (0 or 1).

6.26.4.6 EmberStatus halInternalForceWriteUartData (int8u *port,* int8u ∗ *data,* int8u *length*)

Directly writes a byte to the UART for transmission, regardless of anything currently queued for transmission. Should wait for anything currently in the UART hardware registers to finish transmission first, and block until data is finished being sent.

Parameters

port	Serial port number (0 or 1).
data	Pointer to the data to be transmitted.
length	The length of data to be transmitted

6.26.4.7   EmberStatus halInternalForceReadUartByte (int8u *port,* int8u ∗ *dataByte*)

Directly reads a byte from the UART for reception, regardless of anything currently queued for reception. Does not block if a data byte has not been received.

Parameters

port	Serial port number (0 or 1).
dataByte	The byte to receive data into.

6.26.4.8   void halInternalWaitUartTxComplete (int8u *port*)

Blocks until the UART has finished transmitting any data in its hardware registers.

Parameters

port	Serial port number (0 or 1).

6.26.4.9   void halInternalRestartUart (void)

This function is typically called by halInternalPowerUpBoard() and it is responsible for performing all the work internal to the UART needed to restart the UART after a sleep cycle. (For example, resyncing the DMA hardware and the serial FIFO.)

6.26.4.10   boolean halInternalUart1FlowControlRxIsEnabled (void)

Checks to see if the host is allowed to send serial data to the ncp - i.e., it is not being held off by nCTS or an XOFF. Returns TRUE is the host is able to send.

6.26.4.11   boolean halInternalUart1XonRefreshDone (void)

When Xon/Xoff flow control is used, returns TRUE if the host is not being held off and XON refreshing is complete.

6.26.4.12   boolean halInternalUart1TxIsIdle (void)

Returns true if the uart transmitter is idle, including the transmit shift register.

6.26.4.13   void emSerialBufferNextMessageIsr (EmSerialBufferQueue ∗ *q*)

When new serial transmission is started and `bufferQueue->nextByte` is equal to NULL, this can be called to set up `nextByte` and `lastByte` for the next message.

Parameters

q	Pointer to the buffer queue structure for the port.

6.26.4.14 void emSerialBufferNextBlockIsr (EmSerialBufferQueue ∗ *q,* int8u *port*)

When a serial transmission is in progress and `bufferQueue->nextByte` has been sent and incremented leaving it equal to lastByte, this should be called to set up `nextByte` and `lastByte` for the next block.

Parameters

q	Pointer to the buffer queue structure for the port.
port	Serial port number (0 or 1).

6.26.4.15 void halStackReceiveVuartMessage (int8u ∗ *data,* int8u *length*)

When using a debug build with virtual UART support, this API is called by the stack when virtual UART data has been received over the debug channel.

Parameters

data	Pointer to the the data received
length	Length of the data received

6.27 Button Control

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

6.27.1 Detailed Description

Sample API functions for using push-buttons. See button.h for source code.

6.27.2 Macro Definition Documentation

6.27.2.1 #define BUTTON_PRESSED

Button state is pressed.

Definition at line 24 of file button.h.

6.27.2.2 #define BUTTON_RELEASED

Button state is released.

Definition at line 28 of file button.h.

6.27.3 Function Documentation

6.27.3.1 void halInternalInitButton (void)

Initializes the buttons. This function is automatically called by ::halInit().

6.27.3.2 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 the appropriate BOARD_HEADER.

Returns

BUTTON_PRESSED if the button is pressed or BUTTON_RELEASED if the button is not pressed.

6.27.3.3 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 the appropriate BOARD_HEADER.

Returns

BUTTON_PRESSED if the button is pressed or BUTTON_RELEASED if the button is not pressed.

6.27.3.4 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 the appropriate BOARD_HEADER.
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.

6.28   Buzzer Control

Functions

- void halPlayTune_P (int8u PGM ∗tune, boolean bkg)
- void halStackIndicatePresence (void)

Note Definitions

Flats are used instead of sharps because # is a special character.

- #define NOTE_C3
- #define NOTE_Db3
- #define NOTE_D3
- #define NOTE_Eb3
- #define NOTE_E3
- #define NOTE_F3
- #define NOTE_Gb3
- #define NOTE_G3
- #define NOTE_Ab3
- #define NOTE_A3
- #define NOTE_Bb3
- #define NOTE_B3
- #define NOTE_C4
- #define NOTE_Db4
- #define NOTE_D4
- #define NOTE_Eb4
- #define NOTE_E4
- #define NOTE_F4
- #define NOTE_Gb4
- #define NOTE_G4
- #define NOTE_Ab4
- #define NOTE_A4
- #define NOTE_Bb4
- #define NOTE_B4
- #define NOTE_C5
- #define NOTE_Db5
- #define NOTE_D5
- #define NOTE_Eb5
- #define NOTE_E5
- #define NOTE_F5
- #define NOTE_Gb5
- #define NOTE_G5
- #define NOTE_Ab5
- #define NOTE_A5
- #define NOTE_Bb5
- #define NOTE_B5

6.28.1 Detailed Description

Sample API functions for playing tunes on a piezo buzzer. See buzzer.h for source code.

6.28.2 Macro Definition Documentation

6.28.2.1 #define NOTE_C3

A note which can be used in tune structure definitions.

Definition at line 23 of file buzzer.h.

6.28.2.2 #define NOTE_Db3

A note which can be used in tune structure definitions.

Definition at line 24 of file buzzer.h.

6.28.2.3 #define NOTE_D3

A note which can be used in tune structure definitions.

Definition at line 25 of file buzzer.h.

6.28.2.4 #define NOTE_Eb3

A note which can be used in tune structure definitions.

Definition at line 26 of file buzzer.h.

6.28.2.5 #define NOTE_E3

A note which can be used in tune structure definitions.

Definition at line 27 of file buzzer.h.

6.28.2.6 #define NOTE_F3

A note which can be used in tune structure definitions.

Definition at line 28 of file buzzer.h.

6.28.2.7 #define NOTE_Gb3

A note which can be used in tune structure definitions.

Definition at line 29 of file buzzer.h.

6.28.2.8 #define NOTE_G3

A note which can be used in tune structure definitions.

Definition at line 30 of file buzzer.h.

6.28.2.9   #define NOTE_Ab3

A note which can be used in tune structure definitions.

Definition at line 31 of file buzzer.h.

6.28.2.10   #define NOTE_A3

A note which can be used in tune structure definitions.

Definition at line 32 of file buzzer.h.

6.28.2.11   #define NOTE_Bb3

A note which can be used in tune structure definitions.

Definition at line 33 of file buzzer.h.

6.28.2.12   #define NOTE_B3

A note which can be used in tune structure definitions.

Definition at line 34 of file buzzer.h.

6.28.2.13   #define NOTE_C4

A note which can be used in tune structure definitions.

Definition at line 35 of file buzzer.h.

6.28.2.14   #define NOTE_Db4

A note which can be used in tune structure definitions.

Definition at line 36 of file buzzer.h.

6.28.2.15   #define NOTE_D4

A note which can be used in tune structure definitions.

Definition at line 37 of file buzzer.h.

6.28.2.16   #define NOTE_Eb4

A note which can be used in tune structure definitions.

Definition at line 38 of file buzzer.h.

6.28.2.17 #define NOTE_E4

A note which can be used in tune structure definitions.

Definition at line 39 of file buzzer.h.

6.28.2.18 #define NOTE_F4

A note which can be used in tune structure definitions.

Definition at line 40 of file buzzer.h.

6.28.2.19 #define NOTE_Gb4

A note which can be used in tune structure definitions.

Definition at line 41 of file buzzer.h.

6.28.2.20 #define NOTE_G4

A note which can be used in tune structure definitions.

Definition at line 42 of file buzzer.h.

6.28.2.21 #define NOTE_Ab4

A note which can be used in tune structure definitions.

Definition at line 43 of file buzzer.h.

6.28.2.22 #define NOTE_A4

A note which can be used in tune structure definitions.

Definition at line 44 of file buzzer.h.

6.28.2.23 #define NOTE_Bb4

A note which can be used in tune structure definitions.

Definition at line 45 of file buzzer.h.

6.28.2.24 #define NOTE_B4

A note which can be used in tune structure definitions.

Definition at line 46 of file buzzer.h.

6.28.2.25 #define NOTE_C5

A note which can be used in tune structure definitions.

Definition at line 47 of file buzzer.h.

6.28.2.26 #define NOTE_Db5

A note which can be used in tune structure definitions.

Definition at line 48 of file buzzer.h.

6.28.2.27 #define NOTE_D5

A note which can be used in tune structure definitions.

Definition at line 49 of file buzzer.h.

6.28.2.28 #define NOTE_Eb5

A note which can be used in tune structure definitions.

Definition at line 50 of file buzzer.h.

6.28.2.29 #define NOTE_E5

A note which can be used in tune structure definitions.

Definition at line 51 of file buzzer.h.

6.28.2.30 #define NOTE_F5

A note which can be used in tune structure definitions.

Definition at line 52 of file buzzer.h.

6.28.2.31 #define NOTE_Gb5

A note which can be used in tune structure definitions.

Definition at line 53 of file buzzer.h.

6.28.2.32 #define NOTE_G5

A note which can be used in tune structure definitions.

Definition at line 54 of file buzzer.h.

6.28.2.33 #define NOTE_Ab5

A note which can be used in tune structure definitions.

Definition at line 55 of file buzzer.h.

6.28.2.34 #define NOTE_A5

A note which can be used in tune structure definitions.

Definition at line 56 of file buzzer.h.

6.28.2.35   #define NOTE_Bb5

A note which can be used in tune structure definitions.

Definition at line 57 of file buzzer.h.

6.28.2.36   #define NOTE_B5

A note which can be used in tune structure definitions.

Definition at line 58 of file buzzer.h.

6.28.3   Function Documentation

6.28.3.1   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_B5,  5,                 //Plays the note B5 for 500 milliseconds.
  0,        0                  //NULL terminates the tune.
};
```


6.28.3.2   void halStackIndicatePresence (void)

Causes something to happen on a node (such as playing a tune on the buzzer) that can be used to indicate where it physically is.

6.29   LED Control

Typedefs

- typedef enum HalBoardLedPins HalBoardLed

Functions

- void halInternalInitLed (void)
- void halToggleLed (HalBoardLed led)
- void halSetLed (HalBoardLed led)
- void halClearLed (HalBoardLed led)
- void halStackIndicateActivity (boolean turnOn)

6.29.1   Detailed Description

Sample API funtions for controlling LEDs. When specifying an LED to use, always use the BOARDLEDx definitions that are defined within the BOARD_HEADER.

See led.h for source code.

6.29.2   Typedef Documentation

6.29.2.1   typedef enum HalBoardLedPins HalBoardLed

Ensures that the definitions from the BOARD_HEADER are always used as parameters to the LED functions.

Definition at line 30 of file led.h.

6.29.3   Function Documentation

6.29.3.1   void halInternalInitLed (void)

Configures GPIOs pertaining to the control of LEDs.

6.29.3.2   void halToggleLed (HalBoardLed *led*)

Atomically wraps an XOR or similar operation for a single GPIO pin attached to an LED.

Parameters

led	Identifier (from BOARD_HEADER) for the LED to be toggled.

6.29.3.3   void halSetLed (HalBoardLed *led*)

Turns on (sets) a GPIO pin connected to an LED so that the LED turns on.

Parameters

led	Identifier (from BOARD_HEADER) for the LED to turn on.

6.29.3.4 void halClearLed (**HalBoardLed** *led*)

Turns off (clears) a GPIO pin connected to an LED, which turns off the LED.

Parameters

led	Identifier (from BOARD_HEADER) for the LED to turn off.

6.29.3.5 void halStackIndicateActivity (**boolean** *turnOn*)

Called by the stack to indicate activity over the radio (for both transmission and reception). It is called once with `turnOn` TRUE and shortly thereafter with `turnOn` FALSE.

Typically does something interesting, such as change the state of an LED.

Parameters

turnOn	See Usage.

6.30   Flash Memory Control

Functions

- boolean halFlashEraseIsActive (void)

6.30.1   Detailed Description

Definition and description of public flash manipulation routines.

Note

During an erase or a write the flash is not available, which means code will not be executable from flash. These routines still execute from flash, though, since the bus architecture can support doing so. **Additonally, this also means all interrupts will be disabled.**

Hardware documentation indicates 40us for a write and 21ms for an erase.

See flash.h for source code.

6.30.2   Function Documentation

6.30.2.1   boolean halFlashEraseIsActive (void)

Tells the calling code if a Flash Erase operation is active.

This state is import to know because Flash Erasing is ATOMIC for 21ms and could disrupt interrupt latency. But if an ISR can know that it wasn't serviced immediately due to Flash Erasing, then the ISR has the opportunity to correct in whatever manner it needs to.

Returns

A boolean flag: TRUE if Flash Erase is active, FALSE otherwise.

6.31   System Timer Control

Macros

- #define halIdleForMilliseconds(duration)

Functions

- int16u halInternalStartSystemTimer (void)
- int16u halCommonGetInt16uMillisecondTick (void)
- int32u halCommonGetInt32uMillisecondTick (void)
- int16u halCommonGetInt16uQuarterSecondTick (void)
- EmberStatus halSleepForQuarterSeconds (int32u *duration)
- EmberStatus halSleepForMilliseconds (int32u *duration)
- EmberStatus halCommonIdleForMilliseconds (int32u *duration)

6.31.1   Detailed Description

Functions that provide access to the system clock. A single system tick (as returned by halCommonGet-Int16uMillisecondTick() and halCommonGetInt32uMillisecondTick()) is approximately 1 millisecond.

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

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

A single quarter-second tick (as returned by halCommonGetInt16uQuarterSecondTick()) is approximately 0.25 seconds.

The values used by the time support functions will wrap after an interval. The length of the interval depends on the length of the tick and the number of bits in the value. However, there is no issue when comparing time deltas of less than half this interval with a subtraction, if all data types are the same.

See system-timer.h for source code.

6.31.2   Macro Definition Documentation

6.31.2.1   #define halIdleForMilliseconds(*duration*)

Definition at line 203 of file system-timer.h.

6.31.3   Function Documentation

6.31.3.1   int16u halInternalStartSystemTimer (void)

Initializes the system tick.

Returns

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

6.31.3.2 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.

6.31.3.3 int32u halCommonGetInt32uMillisecondTick (void)

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

EmberStack Usage:

Unused, implementation optional.

Returns

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

6.31.3.4 int16u halCommonGetInt16uQuarterSecondTick (void)

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

EmberStack Usage:

Unused, implementation optional.

Returns

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

6.31.3.5 EmberStatus halSleepForQuarterSeconds (int32u * *duration*)

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

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

Note

This routine always enables interrupts.
The maximum sleep time of the hardware is limited on AVR-based platforms to 8 seconds, on EM2-XX-based platforms to 64 seconds, and on EM35x platforms to 48.5 days. Any sleep duration greater than this limit will wake up briefly (e.g. 16 microseconds) to reenable another sleep cycle.

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

1. The chip may only wake up from the sleep timer. (External GPIO wake events may not be used)

2. Each time a sleep cycle is performed, a loss of accuracy up to +/-750ms will be observed in the system timer.

EmberStack Usage:

Unused, implementation optional.

Parameters

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

Returns

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

6.31.3.6   EmberStatus halSleepForMilliseconds (int32u ∗ *duration*)

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

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

Note

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

EmberStack Usage:

Unused, implementation optional.

Parameters

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

Returns

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

6.31.3.7 EmberStatus halCommonIdleForMilliseconds (int32u ∗ *duration*)

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

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

Note

This routine always enables interrupts.

EmberStack Usage:

Unused, implementation optional.

Parameters

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

Returns

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

6.32 HAL Configuration

Modules

- Sample Breakout Board Configuration
- IAR PLATFORM_HEADER Configuration
- Common PLATFORM_HEADER Configuration
- NVIC Configuration
- Reset Cause Type Definitions

6.32.1 Detailed Description

6.33   Sample Breakout Board Configuration

Custom Baud Rate Definitions

The following define is used with defining a custom baud rate for the UART. This define provides a simple hook into the definition of the baud rates used with the UART. The baudSettings[] array in uart.c links the BAUD_* defines with the actual register values needed for operating the UART. The array baudSettings[] can be edited directly for a custom baud rate or another entry (the register settings) can be provided here with this define.

- #define EMBER_SERIAL_BAUD_CUSTOM

LED Definitions

The following are used to aid in the abstraction with the LED connections. The microcontroller-specific sources use these definitions so they are able to work across a variety of boards which could have different connections. The names and ports/pins used below are intended to match with a schematic of the system to provide the abstraction.

The HalBoardLedPins enum values should always be used when manipulating the state of LEDs, as they directly refer to the GPIOs to which the LEDs are connected.

Note: LEDs 0 and 1 are on the RCM.

Note: LED 2 is on the breakout board (dev0680).

Note: LED 3 simply redirects to LED 2.

- enum HalBoardLedPins {
  BOARDLED0, BOARDLED1, BOARDLED2, BOARDLED3,
  BOARD_ACTIVITY_LED, BOARD_HEARTBEAT_LED }

Button Definitions

The following are used to aid in the abstraction with the Button connections. The microcontroller-specific sources use these definitions so they are able to work across a variety of boards which could have different connections. The names and ports/pins used below are intended to match with a schematic of the system to provide the abstraction.

The BUTTONn macros should always be used with manipulating the buttons as they directly refer to the GPIOs to which the buttons are connected.

Note

The GPIO number must match the IRQ letter

- #define BUTTON0
- #define BUTTON0_IN
- #define BUTTON0_SEL()
- #define BUTTON0_ISR
- #define BUTTON0_INTCFG
- #define BUTTON0_INT_EN_BIT
- #define BUTTON0_FLAG_BIT
- #define BUTTON0_MISS_BIT

- #define BUTTON1
- #define BUTTON1_IN
- #define BUTTON1_SEL()
- #define BUTTON1_ISR
- #define BUTTON1_INTCFG
- #define BUTTON1_INT_EN_BIT
- #define BUTTON1_FLAG_BIT
- #define BUTTON1_MISS_BIT

Radio HoldOff Configuration Definitions

This define does not equate to anything. It is used as a trigger to enable Radio HoldOff support.

The following are used to aid in the abstraction with Radio HoldOff (RHO). The microcontroller-specific sources use these definitions so they are able to work across a variety of boards which could have different connections. The names and ports/pins used below are intended to match with a schematic of the system to provide the abstraction.

The Radio HoldOff input GPIO is abstracted like BUTTON0/1.

- int8u WAKE_ON_LED_RHO_VAR
- #define RHO_GPIO
- #define RHO_ASSERTED
- #define RHO_CFG
- #define RHO_IN
- #define RHO_OUT
- #define RHO_SEL()
- #define RHO_ISR
- #define RHO_INTCFG
- #define RHO_INT_EN_BIT
- #define RHO_FLAG_BIT
- #define RHO_MISS_BIT
- #define PWRUP_CFG_LED_RHO_FOR_RHO
- #define PWRUP_OUT_LED_RHO_FOR_RHO
- #define PWRDN_CFG_LED_RHO_FOR_RHO
- #define PWRDN_OUT_LED_RHO_FOR_RHO
- #define WAKE_ON_LED_RHO_FOR_RHO
- #define PWRUP_CFG_LED_RHO_FOR_LED
- #define PWRUP_OUT_LED_RHO_FOR_LED
- #define PWRDN_CFG_LED_RHO_FOR_LED
- #define PWRDN_OUT_LED_RHO_FOR_LED
- #define WAKE_ON_LED_RHO_FOR_LED
- #define PWRUP_CFG_LED_RHO
- #define PWRUP_OUT_LED_RHO
- #define PWRDN_CFG_LED_RHO
- #define PWRDN_OUT_LED_RHO
- #define WAKE_ON_LED_RHO
- #define halInternalInitRadioHoldOff()
- #define WAKE_ON_LED_RHO_VAR
- #define ADJUST_GPIO_CONFIG_LED_RHO(enableRadioHoldOff)

Temperature sensor ADC channel

Define the analog input channel connected to the LM-20 temperature sensor. The scale factor compensates for different platform input ranges. PB5/ADC0 must be an analog input. PC7 must be an output and set to a high level to power the sensor.

- #define TEMP_SENSOR_ADC_CHANNEL
- #define TEMP_SENSOR_SCALE_FACTOR

Packet Trace

When PACKET_TRACE is defined, ::GPIO_PACFGH will automatically be setup by halInit() to enable Packet Trace support on PA4 and PA5, in addition to the configuration specified below.

Note

This define will override any settings for PA4 and PA5.

- #define PACKET_TRACE

ENABLE_OSC32K

When ENABLE_OSC32K is defined, halInit() will configure system timekeeping to utilize the external 32.768 kHz crystal oscillator rather than the internal 1 kHz RC oscillator.

Note

ENABLE_OSC32K is mutually exclusive with ENABLE_ALT_FUNCTION_NTX_ACTIVE since they define conflicting usage of GPIO PC6.

On initial powerup the 32.768 kHz crystal oscillator will take a little while to start stable oscillation. This only happens on initial powerup, not on wake-from-sleep, since the crystal usually stays running in deep sleep mode.

When ENABLE_OSC32K is defined the crystal oscillator is started as part of halInit(). After the crystal is started we delay for OSC32K_STARTUP_DELAY_MS (time in milliseconds). This delay allows the crystal oscillator to stabilize before we start using it for system timing.

If you set OSC32K_STARTUP_DELAY_MS to less than the crystal's startup time:

- The system timer won't produce a reliable one millisecond tick before the crystal is stable.

- You may see some number of ticks of unknown period occur before the crystal is stable.

- halInit() will complete and application code will begin running, but any events based on the system timer will not be accurate until the crystal is stable.

- An unstable system timer will only affect the APIs in system-timer.h.

Typical 32.768 kHz crystals measured by Ember take about 400 milliseconds to stabilize. Be sure to characterize your particular crystal's stabilization time since crystal behavior can vary.

- #define OSC32K_STARTUP_DELAY_MS

Packet Trace Configuration Defines

Provide the proper set of pin configuration for when the Packet Trace is enabled (look above for the define which enables it). When Packet Trace is not enabled, leave the two PTI pins in their default configuration. If Packet Trace is not being used, feel free to set the pin configurations as desired. The config shown here is simply the Power On Reset defaults.

- #define PWRUP_CFG_PTI_EN
- #define PWRUP_OUT_PTI_EN
- #define PWRDN_CFG_PTI_EN
- #define PWRDN_OUT_PTI_EN
- #define PWRUP_CFG_PTI_DATA
- #define PWRUP_OUT_PTI_DATA
- #define PWRDN_CFG_PTI_DATA
- #define PWRDN_OUT_PTI_DATA

32kHz Oscillator and nTX_ACTIVE Configuration Defines

Since the 32kHz Oscillator and nTX_ACTIVE both share PC6, their configuration defines are linked and instantiated together. Look above for the defines that enable the 32kHz Oscillator and nTX_ACTIVE.

Note

ENABLE_OSC32K is mutually exclusive with ENABLE_ALT_FUNCTION_NTX_ACTIVE since they define conflicting usage of GPIO PC6.

When using the 32kHz, configure PC6 and PC7 for analog for the XTAL.

When using nTX_ACTIVE, configure PC6 for alternate output while awake and a low output when deep-sleeping. Also, configure PC7 for TEMP_EN.

When not using the 32kHz or nTX_ACTIVE, configure PC6 and PC7 for Button1 and TEMP_EN.

- #define PWRUP_CFG_BUTTON1
- #define PWRUP_OUT_BUTTON1
- #define PWRDN_CFG_BUTTON1
- #define PWRDN_OUT_BUTTON1
- #define CFG_TEMPEN

TX_ACTIVE Configuration Defines

Provide the proper set of pin (PC5) configurations for when TX_ACTIVE is enabled (look above for the define which enables it). When TX_ACTIVE is not enabled, configure the pin for LED2.

- #define PWRUP_CFG_LED2
- #define PWRUP_OUT_LED2
- #define PWRDN_CFG_LED2
- #define PWRDN_OUT_LED2

GPIO Configuration Macros

These macros define the GPIO configuration and initial state of the output registers for all the GPIO in the powerup and powerdown modes.

- int16u gpioCfgPowerUp [6]
- int16u gpioCfgPowerDown [6]
- int8u gpioOutPowerUp [3]
- int8u gpioOutPowerDown [3]
- int32u gpioRadioPowerBoardMask
- #define DEFINE_GPIO_RADIO_POWER_BOARD_MASK_VARIABLE()
- #define DEFINE_POWERUP_GPIO_CFG_VARIABLES()
- #define DEFINE_POWERUP_GPIO_OUTPUT_DATA_VARIABLES()
- #define DEFINE_POWERDOWN_GPIO_CFG_VARIABLES()
- #define DEFINE_POWERDOWN_GPIO_OUTPUT_DATA_VARIABLES()
- #define SET_POWERUP_GPIO_CFG_REGISTERS()
- #define SET_POWERUP_GPIO_OUTPUT_DATA_REGISTERS()
- #define SET_POWERDOWN_GPIO_CFG_REGISTERS()
- #define SET_POWERDOWN_GPIO_OUTPUT_DATA_REGISTERS()
- #define CONFIGURE_EXTERNAL_REGULATOR_ENABLE()

GPIO Wake Source Definitions

A convenient define that chooses if this external signal can be used as source to wake from deep sleep. Any change in the state of the signal will wake up the CPU.

- #define WAKE_ON_PA0
- #define WAKE_ON_PA1
- #define WAKE_ON_PA2
- #define WAKE_ON_PA3
- #define WAKE_ON_PA4
- #define WAKE_ON_PA5
- #define WAKE_ON_PA6
- #define WAKE_ON_PA7
- #define WAKE_ON_PB0
- #define WAKE_ON_PB1
- #define WAKE_ON_PB2
- #define WAKE_ON_PB3
- #define WAKE_ON_PB4
- #define WAKE_ON_PB5
- #define WAKE_ON_PB6
- #define WAKE_ON_PB7
- #define WAKE_ON_PC0
- #define WAKE_ON_PC1
- #define WAKE_ON_PC2
- #define WAKE_ON_PC3
- #define WAKE_ON_PC4
- #define WAKE_ON_PC5
- #define WAKE_ON_PC6
- #define WAKE_ON_PC7

6.33.1   Detailed Description

Functions and definitions specific to the breakout board.

Note

> The file dev0680.h is intended to be copied, renamed, and customized for customer-specific hardware.

The file dev0680.h is the default BOARD_HEADER file used with the breakout board of the development kit.

The EM35x on a dev0680 BoB has the following example GPIO configuration. This board file and the default HAL setup reflects this configuration.

- PA0 - SC2MOSI

- PA1 - SC2MISO

- PA2 - SC2SCLK

- PA3 - SC2nSSEL

- PA4 - PTI_EN

- PA5 - PTI_DATA

- PA6 - LED (on RCM), or Radio HoldOff

- PA7 - LED (on RCM)

- PB0 - Power Amplifier shutdown control

- PB1 - SC1TXD

- PB2 - SC1RXD

- PB3 - SC1nCTS

- PB4 - SC1nRTS

- PB5 - TEMP_SENSE

- PB6 - Button (IRQB fixed to PB6)

- PB7 - Buzzer (also used for DataFlash Enable)

- PC0 - JTAG (JRST)

- PC1 - Power Amplifier antenna select control

- PC2 - JTAG (JTDO) / SWO

- PC3 - JTAG (JTDI)

- PC4 - JTAG (JTMS)

- PC5 - LED (on BoB)

- PC6 - Button (IRQC pointed to PC6)

- PC7 - TEMP_EN

6.33.2 Macro Definition Documentation

6.33.2.1 #define EMBER_SERIAL_BAUD_CUSTOM

This define is the register setting for generating a baud of.

1. Refer to the EM35x datasheet's discussion on UART baud rates for the equation used to derive this value.

Definition at line 65 of file dev0680.h.

6.33.2.2 #define BUTTON0

The actual GPIO BUTTON0 is connected to. This define should be used whenever referencing BUTTON0.

Definition at line 124 of file dev0680.h.

6.33.2.3 #define BUTTON0_IN

The GPIO input register for BUTTON0.

Definition at line 128 of file dev0680.h.

6.33.2.4 #define BUTTON0_SEL( )

Point the proper IRQ at the desired pin for BUTTON0.

Note

IRQB is fixed and as such does not need any selection operation.

Definition at line 133 of file dev0680.h.

6.33.2.5 #define BUTTON0_ISR

The interrupt service routine for BUTTON0.

Definition at line 137 of file dev0680.h.

6.33.2.6 #define BUTTON0_INTCFG

The interrupt configuration register for BUTTON0.

Definition at line 141 of file dev0680.h.

6.33.2.7 #define BUTTON0_INT_EN_BIT

The interrupt enable bit for BUTTON0.

Definition at line 145 of file dev0680.h.

6.33.2.8 #define BUTTON0_FLAG_BIT

The interrupt flag bit for BUTTON0.

Definition at line 149 of file dev0680.h.

6.33.2.9 #define BUTTON0_MISS_BIT

The missed interrupt bit for BUTTON0.

Definition at line 153 of file dev0680.h.

6.33.2.10 #define BUTTON1

The actual GPIO BUTTON1 is connected to. This define should be used whenever referencing BUTTON1.

Definition at line 159 of file dev0680.h.

6.33.2.11 #define BUTTON1_IN

The GPIO input register for BUTTON1.

Definition at line 163 of file dev0680.h.

6.33.2.12 #define BUTTON1_SEL( )

Point the proper IRQ at the desired pin for BUTTON1.

Note

For this board, IRQC is pointed at PC6

Definition at line 168 of file dev0680.h.

6.33.2.13 #define BUTTON1_ISR

The interrupt service routine for BUTTON1.

Definition at line 172 of file dev0680.h.

6.33.2.14 #define BUTTON1_INTCFG

The interrupt configuration register for BUTTON1.

Definition at line 176 of file dev0680.h.

6.33.2.15 #define BUTTON1_INT_EN_BIT

The interrupt enable bit for BUTTON1.

Definition at line 180 of file dev0680.h.

6.33.2.16 #define BUTTON1␣FLAG␣BIT

The interrupt flag bit for BUTTON1.

Definition at line 184 of file dev0680.h.

6.33.2.17 #define BUTTON1␣MISS␣BIT

The missed interrupt bit for BUTTON1.

Definition at line 188 of file dev0680.h.

6.33.2.18 #define RHO␣GPIO

The actual GPIO used to control Radio HoldOff.

Note

If RHO_GPIO is not defined, then Radio HoldOff support will not be built in even for runtime use.

Definition at line 234 of file dev0680.h.

6.33.2.19 #define RHO␣ASSERTED

The GPIO signal level to assert Radio HoldOff (1=high, 0=low).

Definition at line 238 of file dev0680.h.

6.33.2.20 #define RHO␣CFG

The GPIO configuration register for Radio HoldOff.

Definition at line 242 of file dev0680.h.

6.33.2.21 #define RHO␣IN

The GPIO input register for Radio HoldOff.

Definition at line 246 of file dev0680.h.

6.33.2.22 #define RHO␣OUT

The GPIO output register for Radio HoldOff.

Definition at line 250 of file dev0680.h.

6.33.2.23 #define RHO␣SEL( )

Point the proper IRQ at the desired pin for Radio HoldOff.

Note

>For this board, IRQD is pointed at PA6

Definition at line 255 of file dev0680.h.

6.33.2.24 #define RHO_ISR

The interrupt service routine for Radio HoldOff.

Definition at line 259 of file dev0680.h.

6.33.2.25 #define RHO_INTCFG

The interrupt configuration register for Radio HoldOff.

Definition at line 263 of file dev0680.h.

6.33.2.26 #define RHO_INT_EN_BIT

The interrupt enable bit for Radio HoldOff.

Definition at line 267 of file dev0680.h.

6.33.2.27 #define RHO_FLAG_BIT

The interrupt flag bit for Radio HoldOff.

Definition at line 271 of file dev0680.h.

6.33.2.28 #define RHO_MISS_BIT

The missed interrupt bit for Radio HoldOff.

Definition at line 275 of file dev0680.h.

6.33.2.29 #define PWRUP_CFG_LED_RHO_FOR_RHO

Configuration of GPIO for Radio HoldOff operation.

Definition at line 279 of file dev0680.h.

6.33.2.30 #define PWRUP_OUT_LED_RHO_FOR_RHO

The actual GPIO used to control Radio HoldOff.

Note

>If RHO_GPIO is not defined, then Radio HoldOff support will not be built in even for runtime use.

Definition at line 280 of file dev0680.h.

6.33.2.31 #define PWRDN_CFG_LED_RHO_FOR_RHO

The actual GPIO used to control Radio HoldOff.

Note

If RHO_GPIO is not defined, then Radio HoldOff support will not be built in even for runtime use.

Definition at line 281 of file dev0680.h.

6.33.2.32 #define PWRDN_OUT_LED_RHO_FOR_RHO

The actual GPIO used to control Radio HoldOff.

Note

If RHO_GPIO is not defined, then Radio HoldOff support will not be built in even for runtime use.

Definition at line 282 of file dev0680.h.

6.33.2.33 #define WAKE_ON_LED_RHO_FOR_RHO

The actual GPIO used to control Radio HoldOff.

Note

If RHO_GPIO is not defined, then Radio HoldOff support will not be built in even for runtime use.

Definition at line 283 of file dev0680.h.

6.33.2.34 #define PWRUP_CFG_LED_RHO_FOR_LED

Configuration of GPIO for LED operation.

Definition at line 287 of file dev0680.h.

6.33.2.35 #define PWRUP_OUT_LED_RHO_FOR_LED

The actual GPIO used to control Radio HoldOff.

Note

If RHO_GPIO is not defined, then Radio HoldOff support will not be built in even for runtime use.

Definition at line 288 of file dev0680.h.

6.33.2.36 #define PWRDN_CFG_LED_RHO_FOR_LED

The actual GPIO used to control Radio HoldOff.

Note

If RHO_GPIO is not defined, then Radio HoldOff support will not be built in even for runtime use.

Definition at line 289 of file dev0680.h.

6.33.2.37 #define PWRDN_OUT_LED_RHO_FOR_LED

The actual GPIO used to control Radio HoldOff.

Note

If RHO_GPIO is not defined, then Radio HoldOff support will not be built in even for runtime use.

Definition at line 290 of file dev0680.h.

6.33.2.38 #define WAKE_ON_LED_RHO_FOR_LED

The actual GPIO used to control Radio HoldOff.

Note

If RHO_GPIO is not defined, then Radio HoldOff support will not be built in even for runtime use.

Definition at line 291 of file dev0680.h.

6.33.2.39 #define PWRUP_CFG_LED_RHO

The following definitions are helpers for managing Radio HoldOff and should not be modified.

(defined(RADIO_HOLDOFF) && defined(RHO_GPIO))

Definition at line 306 of file dev0680.h.

6.33.2.40 #define PWRUP_OUT_LED_RHO

The actual GPIO used to control Radio HoldOff.

Note

If RHO_GPIO is not defined, then Radio HoldOff support will not be built in even for runtime use.

Definition at line 307 of file dev0680.h.

6.33.2.41 #define PWRDN_CFG_LED_RHO

The actual GPIO used to control Radio HoldOff.

Note

If RHO_GPIO is not defined, then Radio HoldOff support will not be built in even for runtime use.

Definition at line 308 of file dev0680.h.

6.33.2.42 #define PWRDN_OUT_LED_RHO

The actual GPIO used to control Radio HoldOff.

Note

> If RHO_GPIO is not defined, then Radio HoldOff support will not be built in even for runtime use.

Definition at line 309 of file dev0680.h.

6.33.2.43 #define WAKE_ON_LED_RHO

The actual GPIO used to control Radio HoldOff.

Note

> If RHO_GPIO is not defined, then Radio HoldOff support will not be built in even for runtime use.

Definition at line 310 of file dev0680.h.

6.33.2.44 #define halInternalInitRadioHoldOff( )

The actual GPIO used to control Radio HoldOff.

Note

> If RHO_GPIO is not defined, then Radio HoldOff support will not be built in even for runtime use.

Definition at line 311 of file dev0680.h.

6.33.2.45 #define WAKE_ON_LED_RHO_VAR

The actual GPIO used to control Radio HoldOff.

Note

> If RHO_GPIO is not defined, then Radio HoldOff support will not be built in even for runtime use.

Definition at line 315 of file dev0680.h.

6.33.2.46 #define ADJUST_GPIO_CONFIG_LED_RHO(*enableRadioHoldOff*)

The actual GPIO used to control Radio HoldOff.

Note

> If RHO_GPIO is not defined, then Radio HoldOff support will not be built in even for runtime use.

Definition at line 317 of file dev0680.h.

6.33.2.47 #define TEMP_SENSOR_ADC_CHANNEL

The analog input channel to use for the temperature sensor.
Definition at line 385 of file dev0680.h.

6.33.2.48 #define TEMP_SENSOR_SCALE_FACTOR

The scale factor to compensate for different input ranges.

Definition at line 389 of file dev0680.h.

6.33.2.49 #define PACKET_TRACE

This define does not equate to anything. It is used as a trigger to enable Packet Trace support on the breakout board (dev0680).

Definition at line 405 of file dev0680.h.

6.33.2.50 #define OSC32K_STARTUP_DELAY_MS

Definition at line 443 of file dev0680.h.

6.33.2.51 #define PWRUP_CFG_PTI_EN

Give the packet trace configuration a friendly name.

Definition at line 603 of file dev0680.h.

6.33.2.52 #define PWRUP_OUT_PTI_EN

Give the packet trace configuration a friendly name.

Definition at line 604 of file dev0680.h.

6.33.2.53 #define PWRDN_CFG_PTI_EN

Give the packet trace configuration a friendly name.

Definition at line 605 of file dev0680.h.

6.33.2.54 #define PWRDN_OUT_PTI_EN

Give the packet trace configuration a friendly name.

Definition at line 606 of file dev0680.h.

6.33.2.55 #define PWRUP_CFG_PTI_DATA

Give the packet trace configuration a friendly name.

Definition at line 607 of file dev0680.h.

6.33.2.56 #define PWRUP_OUT_PTI_DATA

Give the packet trace configuration a friendly name.

Definition at line 608 of file dev0680.h.

6.33.2.57 #define PWRDN_CFG_PTI_DATA

Give the packet trace configuration a friendly name.

Definition at line 609 of file dev0680.h.

6.33.2.58 #define PWRDN_OUT_PTI_DATA

Give the packet trace configuration a friendly name.

Definition at line 610 of file dev0680.h.

6.33.2.59 #define PWRUP_CFG_BUTTON1

Give GPIO PC6 configuration a friendly name.

Definition at line 667 of file dev0680.h.

6.33.2.60 #define PWRUP_OUT_BUTTON1

Give GPIO PC6 configuration a friendly name.

Definition at line 668 of file dev0680.h.

6.33.2.61 #define PWRDN_CFG_BUTTON1

Give GPIO PC6 configuration a friendly name.

Definition at line 669 of file dev0680.h.

6.33.2.62 #define PWRDN_OUT_BUTTON1

Give GPIO PC6 configuration a friendly name.

Definition at line 670 of file dev0680.h.

6.33.2.63 #define CFG_TEMPEN

Give GPIO PC7 configuration a friendly name.
ENABLE_OSC32K

Definition at line 680 of file dev0680.h.

6.33.2.64 #define PWRUP_CFG_LED2

Give the TX_ACTIVE configuration a friendly name.
ENABLE_ALT_FUNCTION_TX_ACTIVE

Definition at line 701 of file dev0680.h.

6.33.2.65   #define PWRUP_OUT_LED2

Give the TX_ACTIVE configuration a friendly name.

ENABLE_ALT_FUNCTION_TX_ACTIVE

Definition at line 702 of file dev0680.h.

6.33.2.66   #define PWRDN_CFG_LED2

Give the TX_ACTIVE configuration a friendly name.

ENABLE_ALT_FUNCTION_TX_ACTIVE

Definition at line 703 of file dev0680.h.

6.33.2.67   #define PWRDN_OUT_LED2

Give the TX_ACTIVE configuration a friendly name.

ENABLE_ALT_FUNCTION_TX_ACTIVE

Definition at line 704 of file dev0680.h.

6.33.2.68   #define DEFINE_GPIO_RADIO_POWER_BOARD_MASK_VARIABLE(   )

Define the mask for GPIO relevant to the radio in the context of power state. Each bit in the mask indicates the corresponding GPIO which should be affected when invoking halStackRadioPowerUpBoard() or hal-StackRadioPowerDownBoard().

Definition at line 736 of file dev0680.h.

6.33.2.69   #define DEFINE_POWERUP_GPIO_CFG_VARIABLES(   )

Initialize GPIO powerup configuration variables.

Definition at line 743 of file dev0680.h.

6.33.2.70   #define DEFINE_POWERUP_GPIO_OUTPUT_DATA_VARIABLES(   )

Initialize GPIO powerup output variables.

Definition at line 775 of file dev0680.h.

6.33.2.71   #define DEFINE_POWERDOWN_GPIO_CFG_VARIABLES(   )

Initialize powerdown GPIO configuration variables.

Definition at line 811 of file dev0680.h.

6.33.2.72   #define DEFINE_POWERDOWN_GPIO_OUTPUT_DATA_VARIABLES(   )

Initialize powerdown GPIO output variables.

Definition at line 845 of file dev0680.h.

6.33.2.73 #define SET_POWERUP_GPIO_CFG_REGISTERS( )

Set powerup GPIO configuration registers.

Definition at line 885 of file dev0680.h.

6.33.2.74 #define SET_POWERUP_GPIO_OUTPUT_DATA_REGISTERS( )

Set powerup GPIO output registers.

Definition at line 897 of file dev0680.h.

6.33.2.75 #define SET_POWERDOWN_GPIO_CFG_REGISTERS( )

Set powerdown GPIO configuration registers.

Definition at line 906 of file dev0680.h.

6.33.2.76 #define SET_POWERDOWN_GPIO_OUTPUT_DATA_REGISTERS( )

Set powerdown GPIO output registers.

Definition at line 918 of file dev0680.h.

6.33.2.77 #define CONFIGURE_EXTERNAL_REGULATOR_ENABLE( )

External regulator enable/disable macro.

Definition at line 931 of file dev0680.h.

6.33.2.78 #define WAKE_ON_PA0

TRUE if this GPIO can wake the chip from deep sleep, FALSE if not.

Definition at line 946 of file dev0680.h.

6.33.2.79 #define WAKE_ON_PA1

TRUE if this GPIO can wake the chip from deep sleep, FALSE if not.

Definition at line 947 of file dev0680.h.

6.33.2.80 #define WAKE_ON_PA2

TRUE if this GPIO can wake the chip from deep sleep, FALSE if not.

Definition at line 948 of file dev0680.h.

6.33.2.81 #define WAKE_ON_PA3

TRUE if this GPIO can wake the chip from deep sleep, FALSE if not.

Definition at line 949 of file dev0680.h.

6.33.2.82 #define WAKE_ON_PA4

TRUE if this GPIO can wake the chip from deep sleep, FALSE if not.

Definition at line 950 of file dev0680.h.

6.33.2.83 #define WAKE_ON_PA5

TRUE if this GPIO can wake the chip from deep sleep, FALSE if not.

Definition at line 951 of file dev0680.h.

6.33.2.84 #define WAKE_ON_PA6

TRUE if this GPIO can wake the chip from deep sleep, FALSE if not.

Definition at line 952 of file dev0680.h.

6.33.2.85 #define WAKE_ON_PA7

TRUE if this GPIO can wake the chip from deep sleep, FALSE if not.

Definition at line 953 of file dev0680.h.

6.33.2.86 #define WAKE_ON_PB0

TRUE if this GPIO can wake the chip from deep sleep, FALSE if not.

Definition at line 954 of file dev0680.h.

6.33.2.87 #define WAKE_ON_PB1

TRUE if this GPIO can wake the chip from deep sleep, FALSE if not.

Definition at line 955 of file dev0680.h.

6.33.2.88 #define WAKE_ON_PB2

TRUE if this GPIO can wake the chip from deep sleep, FALSE if not.

Definition at line 959 of file dev0680.h.

6.33.2.89 #define WAKE_ON_PB3

TRUE if this GPIO can wake the chip from deep sleep, FALSE if not.

Definition at line 961 of file dev0680.h.

6.33.2.90 #define WAKE_ON_PB4

TRUE if this GPIO can wake the chip from deep sleep, FALSE if not.

Definition at line 962 of file dev0680.h.

6.33.2.91 #define WAKE_ON_PB5

TRUE if this GPIO can wake the chip from deep sleep, FALSE if not.

Definition at line 963 of file dev0680.h.

6.33.2.92 #define WAKE_ON_PB6

TRUE if this GPIO can wake the chip from deep sleep, FALSE if not.

Definition at line 964 of file dev0680.h.

6.33.2.93 #define WAKE_ON_PB7

TRUE if this GPIO can wake the chip from deep sleep, FALSE if not.

Definition at line 965 of file dev0680.h.

6.33.2.94 #define WAKE_ON_PC0

TRUE if this GPIO can wake the chip from deep sleep, FALSE if not.

Definition at line 966 of file dev0680.h.

6.33.2.95 #define WAKE_ON_PC1

TRUE if this GPIO can wake the chip from deep sleep, FALSE if not.

Definition at line 967 of file dev0680.h.

6.33.2.96 #define WAKE_ON_PC2

TRUE if this GPIO can wake the chip from deep sleep, FALSE if not.

Definition at line 968 of file dev0680.h.

6.33.2.97 #define WAKE_ON_PC3

TRUE if this GPIO can wake the chip from deep sleep, FALSE if not.

Definition at line 969 of file dev0680.h.

6.33.2.98 #define WAKE_ON_PC4

TRUE if this GPIO can wake the chip from deep sleep, FALSE if not.

Definition at line 970 of file dev0680.h.

6.33.2.99 #define WAKE_ON_PC5

TRUE if this GPIO can wake the chip from deep sleep, FALSE if not.

Definition at line 971 of file dev0680.h.

6.33.2.100 #define WAKE_ON_PC6

TRUE if this GPIO can wake the chip from deep sleep, FALSE if not.

Definition at line 972 of file dev0680.h.

6.33.2.101 #define WAKE_ON_PC7

TRUE if this GPIO can wake the chip from deep sleep, FALSE if not.

Definition at line 973 of file dev0680.h.

6.33.3 Enumeration Type Documentation

6.33.3.1 enum HalBoardLedPins

Assign each GPIO with an LED connected to a convenient name. BOARD_ACTIVITY_LED and BOARD_HEARTBEAT_LED provide a further layer of abstraction on top of the 3 LEDs for verbose coding.

Enumerator:

> **BOARDLED0**
> **BOARDLED1**
> **BOARDLED2**
> **BOARDLED3**
> **BOARD_ACTIVITY_LED**
> **BOARD_HEARTBEAT_LED**

Definition at line 94 of file dev0680.h.

6.33.4 Variable Documentation

6.33.4.1 int8u WAKE_ON_LED_RHO_VAR

The actual GPIO used to control Radio HoldOff.

Note

> If RHO_GPIO is not defined, then Radio HoldOff support will not be built in even for runtime use.

6.33.4.2 int16u gpioCfgPowerUp[6]

Define the mask for GPIO relevant to the radio in the context of power state. Each bit in the mask indicates the corresponding GPIO which should be affected when invoking halStackRadioPowerUpBoard() or halStackRadioPowerDownBoard().

6.33.4.3 int16u gpioCfgPowerDown[6]

Define the mask for GPIO relevant to the radio in the context of power state. Each bit in the mask indicates the corresponding GPIO which should be affected when invoking halStackRadioPowerUpBoard() or halStackRadioPowerDownBoard().

6.33.4.4 int8u gpioOutPowerUp[3]

Define the mask for GPIO relevant to the radio in the context of power state. Each bit in the mask indicates the corresponding GPIO which should be affected when invoking halStackRadioPowerUpBoard() or hal-StackRadioPowerDownBoard().

6.33.4.5 int8u gpioOutPowerDown[3]

Define the mask for GPIO relevant to the radio in the context of power state. Each bit in the mask indicates the corresponding GPIO which should be affected when invoking halStackRadioPowerUpBoard() or hal-StackRadioPowerDownBoard().

6.33.4.6 int32u gpioRadioPowerBoardMask

Define the mask for GPIO relevant to the radio in the context of power state. Each bit in the mask indicates the corresponding GPIO which should be affected when invoking halStackRadioPowerUpBoard() or hal-StackRadioPowerDownBoard().

6.34   IAR PLATFORM_HEADER Configuration

Macros

- #define HAL_HAS_INT64
- #define _HAL_USE_COMMON_PGM_
- #define _HAL_USE_COMMON_MEMUTILS_
- #define PLATCOMMONOKTOINCLUDE
- #define MAIN_FUNCTION_PARAMETERS

Functions

- void _executeBarrierInstructions (void)

Master Variable Types

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

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

Miscellaneous Macros

- void halInternalAssertFailed (const char ∗filename, int linenumber)
- void halInternalResetWatchDog (void)
- #define BIGENDIAN_CPU
- #define NTOHS(val)
- #define NTOHL(val)
- #define NO_STRIPPING
- #define EEPROM
- #define __SOURCEFILE__
- #define assert(condition)
- #define __delay_cycles(x)
- #define DEBUG_LEVEL
- #define halResetWatchdog()
- #define __attribute__(nothing)
- #define UNUSED
- #define SIGNED_ENUM
- #define STACK_FILL_VALUE
- #define RAMFUNC
- #define NO_OPERATION()

- #define SET_REG_FIELD(reg, field, value)
- #define simulatedTimePasses()
- #define simulatedTimePassesMs(x)
- #define simulatedSerialTimePasses()
- #define _HAL_USE_COMMON_DIVMOD_
- #define VAR_AT_SEGMENT(__variableDeclaration, __segmentName)

Portable segment names

- #define __NO_INIT__
- #define __DEBUG_CHANNEL__
- #define __INTVEC__
- #define __CSTACK__
- #define __RESETINFO__
- #define __DATA_INIT__
- #define __DATA__
- #define __BSS__
- #define __APP_RAM__
- #define __CONST__
- #define __TEXT__
- #define __TEXTRW_INIT__
- #define __TEXTRW__
- #define __AAT__
- #define __BAT__
- #define __FAT__
- #define __RAT__
- #define __NVM__
- #define __SIMEE__
- #define __EMHEAP__
- #define __EMHEAP_OVERLAY__
- #define __GUARD_REGION__
- #define __DLIB_PERTHREAD_INIT__
- #define __DLIB_PERTHREAD_INITIALIZED_DATA__
- #define __DLIB_PERTHREAD_ZERO_DATA__
- #define STACK_SEGMENT_BEGIN
- #define STACK_SEGMENT_END
- #define EMHEAP_SEGMENT_BEGIN
- #define EMHEAP_SEGMENT_END
- #define EMHEAP_OVERLAY_SEGMENT_END
- #define RESETINFO_SEGMENT_END
- #define CODE_SEGMENT_BEGIN
- #define CODE_SEGMENT_END

Global Interrupt Manipulation Macros

Note: The special purpose BASEPRI register is used to enable and disable interrupts while permitting faults. When BASEPRI is set to 1 no interrupts can trigger. The configurable faults (usage, memory management, and bus faults) can trigger if enabled as well as the always-enabled exceptions (reset, NMI and hard fault). When BASEPRI is set to 0, it is disabled, so any interrupt can triggger if its priority is higher than the current priority.

- #define ATOMIC_LITE(blah)
- #define DECLARE_INTERRUPT_STATE_LITE
- #define DISABLE_INTERRUPTS_LITE()
- #define RESTORE_INTERRUPTS_LITE()
- #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()
- #define SET_BASE_PRIORITY_LEVEL(basepri)

External Declarations

These are routines that are defined in certain header files that we don't want to include, e.g. stdlib.h

- int abs (int I)

6.34.1 Detailed Description

Compiler and Platform specific definitions and typedefs for the IAR ARM C compiler.

Note

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

See iar.h and platform-common.h for source code.

6.34.2 Macro Definition Documentation

6.34.2.1 #define HAL_HAS_INT64

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

Definition at line 104 of file iar.h.

6.34.2.2 #define _HAL_USE_COMMON_PGM_

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

Definition at line 109 of file iar.h.

6.34.2.3   #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 123 of file iar.h.

6.34.2.4   #define NTOHS(*val*)

Define intrinsics for NTOHL and NTOHS to save code space by making endian.c compile to nothing.

Definition at line 130 of file iar.h.

6.34.2.5   #define NTOHL(*val*)

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 131 of file iar.h.

6.34.2.6   #define NO_STRIPPING

A friendlier name for the compiler's intrinsic for not stripping.

Definition at line 138 of file iar.h.

6.34.2.7   #define EEPROM

A friendlier name for the compiler's intrinsic for eeprom reference.

Definition at line 145 of file iar.h.

6.34.2.8   #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 154 of file iar.h.

6.34.2.9   #define assert(*condition*)

A custom implementation of the C language assert macro. This macro implements the conditional evaluation and calls the function halInternalAssertFailed(). (see hal/micro/micro.h)

Definition at line 171 of file iar.h.

6.34.2.10   #define __delay_cycles(*x*)

__delay_cycles() is an intrinsic IAR call; however, we have explicity disallowed it since it is too specific to the system clock.

Note

> Please use halCommonDelayMicroseconds() instead, because it correctly accounts for various system clock speeds.

Definition at line 198 of file iar.h.

6.34.2.11 #define DEBUG_LEVEL

Set debug level based on whether DEBUG or DEBUG_OFF are defined. For the EM35x, basic debugging support is included if DEBUG is not defined.

Definition at line 213 of file iar.h.

6.34.2.12 #define halResetWatchdog( )

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 223 of file iar.h.

6.34.2.13 #define __attribute__(nothing)

Define **attribute** to nothing since it isn't handled by IAR.

Definition at line 229 of file iar.h.

6.34.2.14 #define UNUSED

Declare a variable as unused to avoid a warning. Has no effect in IAR builds.

Definition at line 236 of file iar.h.

6.34.2.15 #define SIGNED_ENUM

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

Definition at line 241 of file iar.h.

6.34.2.16 #define STACK_FILL_VALUE

Define the magic value that is interpreted by IAR C-SPY's Stack View.

Definition at line 247 of file iar.h.

6.34.2.17 #define RAMFUNC

Define a generic RAM function identifier to a compiler specific one.

Definition at line 257 of file iar.h.

6.34.2.18 #define NO_OPERATION( )

Define a generic no operation identifier to a compiler specific one.

Definition at line 263 of file iar.h.

6.34.2.19 #define SET_REG_FIELD(*reg, field, value*)

A convenience macro that makes it easy to change the field of a register to any value.

Definition at line 269 of file iar.h.

6.34.2.20 #define simulatedTimePasses( )

Stub for code not running in simulation.

Definition at line 278 of file iar.h.

6.34.2.21 #define simulatedTimePassesMs(*x*)

Stub for code not running in simulation.

Definition at line 282 of file iar.h.

6.34.2.22 #define simulatedSerialTimePasses( )

Stub for code not running in simulation.

Definition at line 286 of file iar.h.

6.34.2.23 #define _HAL_USE_COMMON_DIVMOD_

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

Definition at line 292 of file iar.h.

6.34.2.24 #define VAR_AT_SEGMENT(__*variableDeclaration*, __*segmentName*)

Provide a portable way to specify the segment where a variable lives.

Definition at line 299 of file iar.h.

6.34.2.25 #define __NO_INIT__

Portable segment names.

Definition at line 312 of file iar.h.

6.34.2.26 #define __DEBUG_CHANNEL__

Portable segment names.

Definition at line 313 of file iar.h.

6.34.2.27 #define __INTVEC__

Portable segment names.

Definition at line 314 of file iar.h.

6.34.2.28 #define __CSTACK__

Portable segment names.

Definition at line 315 of file iar.h.

6.34.2.29 #define __RESETINFO__

Portable segment names.

Definition at line 316 of file iar.h.

6.34.2.30 #define __DATA_INIT__

Portable segment names.

Definition at line 317 of file iar.h.

6.34.2.31 #define __DATA__

Portable segment names.

Definition at line 318 of file iar.h.

6.34.2.32 #define __BSS__

Portable segment names.

Definition at line 319 of file iar.h.

6.34.2.33 #define __APP_RAM__

Portable segment names.

Definition at line 320 of file iar.h.

6.34.2.34 #define __CONST__

Portable segment names.

Definition at line 321 of file iar.h.

6.34.2.35 #define __TEXT__

Portable segment names.

Definition at line 322 of file iar.h.

6.34.2.36 #define __TEXTRW_INIT__

Portable segment names.

Definition at line 323 of file iar.h.

6.34.2.37 #define __TEXTRW__

Portable segment names.

Definition at line 324 of file iar.h.

6.34.2.38 #define __AAT__

Portable segment names.

Definition at line 325 of file iar.h.

6.34.2.39 #define __BAT__

Portable segment names.

Definition at line 326 of file iar.h.

6.34.2.40 #define __FAT__

Portable segment names.

Definition at line 327 of file iar.h.

6.34.2.41 #define __RAT__

Portable segment names.

Definition at line 328 of file iar.h.

6.34.2.42 #define __NVM__

Portable segment names.

Definition at line 329 of file iar.h.

6.34.2.43 #define __SIMEE__

Portable segment names.

Definition at line 330 of file iar.h.

6.34.2.44 #define __EMHEAP__

Portable segment names.

Definition at line 331 of file iar.h.

6.34.2.45 #define __EMHEAP_OVERLAY__

Portable segment names.

Definition at line 332 of file iar.h.

6.34.2.46 #define __GUARD_REGION__

Portable segment names.

Definition at line 333 of file iar.h.

6.34.2.47 #define __DLIB_PERTHREAD_INIT__

Portable segment names.

Definition at line 334 of file iar.h.

6.34.2.48 #define __DLIB_PERTHREAD_INITIALIZED_DATA__

Portable segment names.

Definition at line 335 of file iar.h.

6.34.2.49 #define __DLIB_PERTHREAD_ZERO_DATA__

Portable segment names.

Definition at line 336 of file iar.h.

6.34.2.50 #define STACK_SEGMENT_BEGIN

Portable segment names.

Definition at line 373 of file iar.h.

6.34.2.51 #define STACK_SEGMENT_END

Portable segment names.

Definition at line 374 of file iar.h.

6.34.2.52 #define EMHEAP_SEGMENT_BEGIN

Portable segment names.

Definition at line 376 of file iar.h.

6.34.2.53 #define EMHEAP_SEGMENT_END

Portable segment names.

Definition at line 377 of file iar.h.

6.34.2.54 #define EMHEAP_OVERLAY_SEGMENT_END

Portable segment names.

Definition at line 379 of file iar.h.

6.34.2.55 #define RESETINFO_SEGMENT_END

Portable segment names.

Definition at line 381 of file iar.h.

6.34.2.56 #define CODE_SEGMENT_BEGIN

Portable segment names.

Definition at line 383 of file iar.h.

6.34.2.57 #define CODE_SEGMENT_END

Portable segment names.

Definition at line 384 of file iar.h.

6.34.2.58 #define ATOMIC_LITE(*blah*)

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 408 of file iar.h.

6.34.2.59 #define DECLARE_INTERRUPT_STATE_LITE

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 409 of file iar.h.

6.34.2.60 #define DISABLE_INTERRUPTS_LITE( )

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 410 of file iar.h.

6.34.2.61 #define RESTORE_INTERRUPTS_LITE( )

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 411 of file iar.h.

6.34.2.62 #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 497 of file iar.h.

6.34.2.63 #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 514 of file iar.h.

6.34.2.64 #define INTERRUPTS_ON( )

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

Definition at line 528 of file iar.h.

6.34.2.65 #define INTERRUPTS_OFF( )

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

Definition at line 542 of file iar.h.

6.34.2.66 #define INTERRUPTS_ARE_OFF( )

Returns

TRUE if global interrupts are disabled.

Definition at line 554 of file iar.h.

6.34.2.67 #define INTERRUPTS_WERE_ON( )

Returns

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

Definition at line 560 of file iar.h.

6.34.2.68 #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 566 of file iar.h.

6.34.2.69 #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 582 of file iar.h.

6.34.2.70 #define SET_BASE_PRIORITY_LEVEL(basepri)

Sets the base priority mask (BASEPRI) to the value passed, bit shifted up by PRIGROUP_POSITION+1. This will inhibit the core from taking all interrupts with a preemptive priority equal to or less than the BASEPRI mask. This macro is dependent on the value of PRIGROUP_POSITION in nvic-config.h. Note that the value 0 disables the the base priority mask.

Refer to the "PRIGROUP" table in nvic-config.h to know the valid values for this macro depending on the value of PRIGROUP_POSITION. With respect to the table, this macro can only take the preemptive priority group numbers denoted by the parenthesis.

Definition at line 604 of file iar.h.

6.34.2.71 #define _HAL_USE_COMMON_MEMUTILS_

Use the C Standard Library Memory Utilities from platform-common.h.

Definition at line 617 of file iar.h.

6.34.2.72 #define PLATCOMMONOKTOINCLUDE

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

Definition at line 646 of file iar.h.

6.34.2.73   #define MAIN_FUNCTION_PARAMETERS

The kind of arguments the main function takes.

Definition at line 653 of file iar.h.

6.34.3   Typedef Documentation

6.34.3.1   typedef unsigned char boolean

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

Definition at line 89 of file iar.h.

6.34.3.2   typedef unsigned char int8u

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

Definition at line 90 of file iar.h.

6.34.3.3   typedef signed char int8s

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

Definition at line 91 of file iar.h.

6.34.3.4   typedef unsigned short int16u

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

Definition at line 92 of file iar.h.

6.34.3.5   typedef signed short int16s

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

Definition at line 93 of file iar.h.

6.34.3.6   typedef unsigned int int32u

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

Definition at line 94 of file iar.h.

6.34.3.7   typedef signed int int32s

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

Definition at line 95 of file iar.h.

6.34.3.8 typedef unsigned long long int64u

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

Definition at line 96 of file iar.h.

6.34.3.9 typedef signed long long int64s

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

Definition at line 97 of file iar.h.

6.34.3.10 typedef unsigned int PointerType

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

Definition at line 98 of file iar.h.

6.34.4 Function Documentation

6.34.4.1 void halInternalAssertFailed (const char ∗ *filename,* int *linenumber*)

A prototype definition for use by the assert macro. (see hal/micro/micro.h)

6.34.4.2 void halInternalResetWatchDog (void)

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.

6.34.4.3 void _executeBarrierInstructions (void)

6.34.4.4 int abs (int *I*)

Returns the absolute value of I (also called the magnitude of I). That is, if I is negative, the result is the opposite of I, but if I is nonnegative the result is I.

Parameters

I	An integer.

Returns

A nonnegative integer.

6.35 Common PLATFORM_HEADER Configuration

Generic Types

- #define TRUE
- #define FALSE
- #define NULL

Bit Manipulation Macros

- #define BIT(x)
- #define BIT32(x)
- #define SETBIT(reg, bit)
- #define SETBITS(reg, bits)
- #define CLEARBIT(reg, bit)
- #define CLEARBITS(reg, bits)
- #define READBIT(reg, bit)
- #define READBITS(reg, bits)

Byte Manipulation Macros

- #define LOW_BYTE(n)
- #define HIGH_BYTE(n)
- #define HIGH_LOW_TO_INT(high, low)
- #define BYTE_0(n)
- #define BYTE_1(n)
- #define BYTE_2(n)
- #define BYTE_3(n)
- #define COUNTOF(a)

Time Manipulation Macros

- #define elapsedTimeInt8u(oldTime, newTime)
- #define elapsedTimeInt16u(oldTime, newTime)
- #define elapsedTimeInt32u(oldTime, newTime)
- #define MAX_INT8U_VALUE
- #define HALF_MAX_INT8U_VALUE
- #define timeGTorEqualInt8u(t1, t2)
- #define MAX_INT16U_VALUE
- #define HALF_MAX_INT16U_VALUE
- #define timeGTorEqualInt16u(t1, t2)
- #define MAX_INT32U_VALUE
- #define HALF_MAX_INT32U_VALUE
- #define timeGTorEqualInt32u(t1, t2)

6.35.1 Detailed Description

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

See platform-common.h for source code.

6.35.2 Macro Definition Documentation

6.35.2.1 #define TRUE

An alias for one, used for clarity.

Definition at line 195 of file platform-common.h.

6.35.2.2 #define FALSE

An alias for zero, used for clarity.

Definition at line 200 of file platform-common.h.

6.35.2.3 #define NULL

The null pointer.

Definition at line 206 of file platform-common.h.

6.35.2.4 #define BIT(*x*)

Useful to reference a single bit of a byte.

Definition at line 220 of file platform-common.h.

6.35.2.5 #define BIT32(*x*)

Useful to reference a single bit of an int32u type.

Definition at line 225 of file platform-common.h.

6.35.2.6 #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 232 of file platform-common.h.

6.35.2.7 #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 239 of file platform-common.h.

6.35.2.8 #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 246 of file platform-common.h.

6.35.2.9 #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 253 of file platform-common.h.

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

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

Definition at line 258 of file platform-common.h.

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

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

Definition at line 264 of file platform-common.h.

6.35.2.12 #define LOW_BYTE(*n* **)**

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

Definition at line 278 of file platform-common.h.

6.35.2.13 #define HIGH_BYTE(*n* **)**

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

Definition at line 283 of file platform-common.h.

6.35.2.14 #define HIGH_LOW_TO_INT(*high, low*)

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

Definition at line 289 of file platform-common.h.

6.35.2.15 #define BYTE_0(*n*)

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

Definition at line 297 of file platform-common.h.

6.35.2.16 #define BYTE_1(*n*)

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

Definition at line 302 of file platform-common.h.

6.35.2.17 #define BYTE_2(*n*)

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

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

6.35.2.18 #define BYTE_3(*n*)

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

Definition at line 312 of file platform-common.h.

6.35.2.19 #define COUNTOF(*a*)

Returns the number of entries in an array.

Definition at line 317 of file platform-common.h.

6.35.2.20 #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 332 of file platform-common.h.

6.35.2.21 #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 339 of file platform-common.h.

6.35.2.22   #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 346 of file platform-common.h.

6.35.2.23   #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 353 of file platform-common.h.

6.35.2.24   #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 354 of file platform-common.h.

6.35.2.25   #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 355 of file platform-common.h.

6.35.2.26   #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 362 of file platform-common.h.

6.35.2.27   #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 363 of file platform-common.h.

6.35.2.28   #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 364 of file platform-common.h.

6.35.2.29   #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 371 of file platform-common.h.

6.35.2.30 #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 372 of file platform-common.h.

6.35.2.31 #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 373 of file platform-common.h.

6.36 NVIC Configuration

Nested Vectored Interrupt Controller configuration header.

This header defines the functions called by all of the NVIC exceptions/ interrupts. The following are the nine peripheral ISRs available for modification. To use one of these ISRs, it must be instantiated somewhere in the HAL. Each ISR may only be instantiated once. It is not necessary to instantiate all or any of these ISRs (a stub will be automatically generated if an ISR is not instantiated).

- `void halTimer1Isr(void);`
- `void halTimer2Isr(void);`
- `void halSc1Isr(void);`
- `void halSc1Isr(void);`
- `void halAdcIsr(void);`
- `void halIrqAIsr(void);`
- `void halIrqBIsr(void);`
- `void halIrqCIsr(void);`
- `void halIrqDIsr(void);`

Note

This file should **not** be modified.

6.37　Reset Cause Type Definitions

Definitions for all the reset cause types.

Reset cause types are built from a base definition and an extended. definition. The base definitions allow working with entire categories of resets while the extended definitions allow drilling down to very specific causes. The macros for the base and extended definitions are combined for use in the code and equated to their combined numberical equivalents. In addition, exach base and extended definition is given a corresponding 3 letter ASCII string to facilitate printing. The ASCII strings are best use with halGet-ExtendedResetString.

For example:

```
RESET_BASE_DEF(EXTERNAL,             0x03,     "EXT")
  RESET_EXT_DEF(EXTERNAL, UNKNOWN,   0x00,     "UNK")
  RESET_EXT_DEF(EXTERNAL, PIN,       0x01,     "PIN")
```

results in enums which includes the entries:

```
RESET_EXTERNAL = 0x03
RESET_EXTERNAL_PIN = 0x0301
```

For a complete listing of all reset base and extended definitions, see reset-def.h for source code.

6.38 HAL Utilities

Modules

- Crash and Watchdog Diagnostics
- Cyclic Redundancy Code (CRC)
- Random Number Generation
- Network to Host Byte Order Conversion

6.38.1 Detailed Description

6.39   Crash and Watchdog Diagnostics

Macros

- #define halResetWasCrash()

Functions

- int32u halGetMainStackBytesUsed (void)
- void halPrintCrashSummary (int8u port)
- void halPrintCrashDetails (int8u port)
- void halPrintCrashData (int8u port)

6.39.1   Detailed Description

Crash and watchdog diagnostic functions. See diagnostic.h for source code.

6.39.2   Macro Definition Documentation

6.39.2.1   #define halResetWasCrash(   )

Macro evaluating to TRUE if the last reset was a crash, FALSE otherwise.

Definition at line 266 of file diagnostic.h.

6.39.3   Function Documentation

6.39.3.1   int32u halGetMainStackBytesUsed (void)

Returns the number of bytes used in the main stack.

Returns

The number of bytes used in the main stack.

6.39.3.2   void halPrintCrashSummary (int8u *port*)

Print a summary of crash details.

Parameters

port	Serial port number (0 or 1).

6.39.3.3   void halPrintCrashDetails (int8u *port*)

Print the complete, decoded crash details.

Parameters

port	Serial port number (0 or 1).

6.39.3.4 void halPrintCrashData (int8u *port*)

Print the complete crash data.

Parameters

port	Serial port number (0 or 1).

6.40 Cyclic Redundancy Code (CRC)

Macros

- #define INITIAL_CRC
- #define CRC32_START
- #define CRC32_END

Functions

- int16u halCommonCrc16 (int8u newByte, int16u prevResult)
- int32u halCommonCrc32 (int8u newByte, int32u prevResult)

6.40.1 Detailed Description

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

6.40.2 Macro Definition Documentation

6.40.2.1 #define INITIAL_CRC

Definition at line 49 of file crc.h.

6.40.2.2 #define CRC32_START

Definition at line 50 of file crc.h.

6.40.2.3 #define CRC32_END

Definition at line 51 of file crc.h.

6.40.3 Function Documentation

6.40.3.1 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.

6.40.3.2 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.

6.41 Random Number Generation

Functions

- void halStackSeedRandom (int32u seed)
- int16u halCommonGetRandom (void)

6.41.1 Detailed Description

Functions that provide access to random numbers. These functions may be hardware accelerated, though often are not.

See random.h for source code.

6.41.2 Function Documentation

6.41.2.1 void halStackSeedRandom (int32u *seed*)

Seeds the halCommonGetRandom() pseudorandom number generator.

Called by the stack during initialization with a seed from the radio.

Parameters

seed	A seed for the pseudorandom number generator.

6.41.2.2 int16u halCommonGetRandom (void)

Runs a standard LFSR to generate pseudorandom numbers.

Called by the MAC in the stack to choose random backoff slots.

Complicated implementations may improve the MAC's ability to avoid collisions in large networks, but it is **critical** to implement this function to return quickly.

6.42 Network to Host Byte Order Conversion

Functions

- int16u NTOHS (int16u val)
- int32u NTOHL (int32u val)

6.42.1 Detailed Description

Functions that provide conversions from network to host byte order. Network byte order is big endian, so these APIs are only necessary on platforms which have a natural little endian byte order. On big-endian platforms, the APIs are macro'd away to nothing. See endian.h for source code.

6.42.2 Function Documentation

6.42.2.1 int16u NTOHS (int16u *val*)

Converts a short (16-bit) value from network to host byte order.

6.42.2.2 int32u NTOHL (int32u *val*)

Converts a long (32-bit) value from network to host byte order.

6.43 Bootloader Interfaces

Modules

- Common
- Standalone
- Application

6.43.1 Detailed Description

6.44 Common

Macros

- #define BOOTLOADER_BASE_TYPE(extendedType)
- #define BOOTLOADER_MAKE_EXTENDED_TYPE(baseType, extendedSpecifier)
- #define BL_EXT_TYPE_NULL
- #define BL_EXT_TYPE_STANDALONE_UNKNOWN
- #define BL_EXT_TYPE_SERIAL_UART
- #define BL_EXT_TYPE_SERIAL_UART_OTA
- #define BL_EXT_TYPE_EZSP_SPI
- #define BL_EXT_TYPE_EZSP_SPI_OTA
- #define BL_EXT_TYPE_APP_UNKNOWN
- #define BL_EXT_TYPE_APP_SPI
- #define BL_EXT_TYPE_APP_I2C
- #define BOOTLOADER_INVALID_VERSION

Functions

- BlBaseType halBootloaderGetType (void)
- BlExtendedType halBootloaderGetInstalledType (void)
- int16u halGetBootloaderVersion (void)
- void halGetExtendedBootloaderVersion (int32u ∗emberVersion, int32u ∗customerVersion)

Bootloader Numerical Definitions

These are numerical definitions for the possible bootloader types and a typedef of the bootloader base type.

- #define BL_TYPE_NULL
- #define BL_TYPE_STANDALONE
- #define BL_TYPE_APPLICATION
- #define BL_TYPE_BOOTLOADER
- #define BL_TYPE_SMALL_BOOTLOADER

Bootloader type definitions

These are the type definitions for the bootloader.

- typedef int8u BlBaseType
- typedef int16u BlExtendedType

6.44.1 Detailed Description

Common bootloader interface defines and functions. See bootloader-interface.h for source code.

6.44.2 Macro Definition Documentation

6.44.2.1 #define BL_TYPE_NULL

Numerical definition for a bootloader type.

Definition at line 27 of file bootloader-interface.h.

6.44.2.2 #define BL_TYPE_STANDALONE

Numerical definition for a bootloader type.

Definition at line 28 of file bootloader-interface.h.

6.44.2.3 #define BL_TYPE_APPLICATION

Numerical definition for a bootloader type.

Definition at line 29 of file bootloader-interface.h.

6.44.2.4 #define BL_TYPE_BOOTLOADER

Numerical definition for a bootloader type.

Definition at line 30 of file bootloader-interface.h.

6.44.2.5 #define BL_TYPE_SMALL_BOOTLOADER

Numerical definition for a bootloader type.

Definition at line 31 of file bootloader-interface.h.

6.44.2.6 #define BOOTLOADER_BASE_TYPE(*extendedType*)

Macro returning the base type of a bootloader when given an extended type.

Definition at line 58 of file bootloader-interface.h.

6.44.2.7 #define BOOTLOADER_MAKE_EXTENDED_TYPE(*baseType,* *extendedSpecifier*)

Macro returning the extended type of a bootloader when given a base type and extendedSpecifier.

Definition at line 64 of file bootloader-interface.h.

6.44.2.8 #define BL_EXT_TYPE_NULL

Macro defining the extended NULL bootloader type.

Definition at line 69 of file bootloader-interface.h.

6.44.2.9 #define BL_EXT_TYPE_STANDALONE_UNKNOWN

Macro defining the extended standalone unknown bootloader type.

Definition at line 73 of file bootloader-interface.h.

6.44.2.10 #define BL_EXT_TYPE_SERIAL_UART

Macro defining the extended standalone UART bootloader type.

Definition at line 77 of file bootloader-interface.h.

6.44.2.11 #define BL_EXT_TYPE_SERIAL_UART_OTA

Macro defining the extended standalone OTA and UART bootloader type.

Definition at line 84 of file bootloader-interface.h.

6.44.2.12 #define BL_EXT_TYPE_EZSP_SPI

Definition at line 85 of file bootloader-interface.h.

6.44.2.13 #define BL_EXT_TYPE_EZSP_SPI_OTA

Definition at line 86 of file bootloader-interface.h.

6.44.2.14 #define BL_EXT_TYPE_APP_UNKNOWN

Macro defining the extended application unknown bootloader type.

Definition at line 90 of file bootloader-interface.h.

6.44.2.15 #define BL_EXT_TYPE_APP_SPI

Macro defining the extended application SPI bootloader type.

Definition at line 94 of file bootloader-interface.h.

6.44.2.16 #define BL_EXT_TYPE_APP_I2C

Macro defining the extended application I2C bootloader type.

Definition at line 98 of file bootloader-interface.h.

6.44.2.17 #define BOOTLOADER_INVALID_VERSION

Define an invalid bootloader version.

Definition at line 109 of file bootloader-interface.h.

6.44.3   Typedef Documentation

6.44.3.1   typedef int8u BlBaseType

Define the bootloader base type.

Definition at line 41 of file bootloader-interface.h.

6.44.3.2   typedef int16u BlExtendedType

Define the bootloader extended type.

Definition at line 44 of file bootloader-interface.h.

6.44.4   Function Documentation

6.44.4.1   BlBaseType halBootloaderGetType (void)

Returns the bootloader base type the application was built for.

Returns

> BL_TYPE_NULL, BL_TYPE_STANDALONE, or BL_TYPE_APPLICATION

6.44.4.2   BlExtendedType halBootloaderGetInstalledType (void)

Returns the extended bootloader type of the bootloader that is present on the chip.

6.44.4.3   int16u halGetBootloaderVersion (void)

Returns the version of the installed bootloader, regardless of its type.

Returns

> Version if bootloader installed, or BOOTLOADER_INVALID_VERSION. A returned version of 0x1234 would indicate version 1.2 build 34

6.44.4.4   void halGetExtendedBootloaderVersion (int32u ∗ *emberVersion,* int32u ∗ *customerVersion*)

Return extended bootloader version information, if supported. This API is not supported for EM2XX chips and only returns extra information on bootloaders built on or after the 4.7 release.

Parameters

emberVersion	If specified, we will return the full 32bit ember version for this bootloader. Format is major, minor, patch, doc (4bit nibbles) followed by a 16bit build number.
customer-Version	This will return the 32bit value specified in CUSTOMER_BOOTLOADER_VERSION at build time.

6.45 Standalone

Macros

- #define NO_BOOTLOADER_MODE
- #define STANDALONE_BOOTLOADER_NORMAL_MODE
- #define STANDALONE_BOOTLOADER_RECOVERY_MODE

Functions

- int16u halGetStandaloneBootloaderVersion (void)
- EmberStatus halLaunchStandaloneBootloader (int8u mode)

6.45.1 Detailed Description

Definition of the standalone bootloader interface. Some functions in this file return an EmberStatus value. See error-def.h for definitions of all EmberStatus return values.

See bootloader-interface-standalone.h for source code.

6.45.2 Macro Definition Documentation

6.45.2.1 #define NO_BOOTLOADER_MODE

Define a numerical value for NO BOOTLOADER mode. In other words, the bootloader should not be run.

Definition at line 33 of file bootloader-interface-standalone.h.

6.45.2.2 #define STANDALONE_BOOTLOADER_NORMAL_MODE

Define a numerical value for the normal bootloader mode.

Definition at line 37 of file bootloader-interface-standalone.h.

6.45.2.3 #define STANDALONE_BOOTLOADER_RECOVERY_MODE

Define a numerical value for the recovery bootloader mode.

Definition at line 41 of file bootloader-interface-standalone.h.

6.45.3 Function Documentation

6.45.3.1 int16u halGetStandaloneBootloaderVersion (void)

Detects if the standalone bootloader is installed, and if so returns the installed version.

A returned version of 0x1234 would indicate version 1.2 build 34

Returns

BOOTLOADER_INVALID_VERSION if the standalone bootloader is not present, or the version of the installed standalone bootloader.

6.45.3.2 EmberStatus halLaunchStandaloneBootloader (int8u *mode*)

Quits the current application and launches the standalone bootloader (if installed). The function returns an error if the standalone bootloader is not present.

Parameters

mode	Controls the mode in which the standalone bootloader will run. See the bootloader Application Note for full details. Options are: • STANDALONE_BOOTLOADER_NORMAL_MODE Will listen for an over-the-air image transfer on the current channel with current power settings. • STANDALONE_BOOTLOADER_RECOVERY_MODE Will listen for an over-the-air image transfer on the default channel with default power settings. Both modes also allow an image transfer to begin via serial xmodem.

Returns

An EmberStatus error if the standalone bootloader is not present, or EMBER_SUCCESS.

6.46 Application

Macros

- #define BOOTLOADER_SEGMENT_SIZE_LOG2
- #define BOOTLOADER_SEGMENT_SIZE
- #define BL_IMAGE_IS_VALID_CONTINUE

Functions

- int8u halAppBootloaderInit (void)
- const HalEepromInformationType ∗ halAppBootloaderInfo (void)
- void halAppBootloaderShutdown (void)
- void halAppBootloaderImageIsValidReset (void)
- int16u halAppBootloaderImageIsValid (void)
- EmberStatus halAppBootloaderInstallNewImage (void)
- int8u halAppBootloaderWriteRawStorage (int32u address, const int8u ∗data, int16u len)
- int8u halAppBootloaderReadRawStorage (int32u address, int8u ∗data, int16u len)
- int8u halAppBootloaderEraseRawStorage (int32u address, int32u len)
- boolean halAppBootloaderStorageBusy (void)
- int8u halAppBootloaderReadDownloadSpace (int16u pageToBeRead, int8u ∗destRamBuffer)
- int8u halAppBootloaderWriteDownloadSpace (int16u pageToBeWritten, int8u ∗RamPtr)
- int8u halAppBootloaderGetImageData (int32u ∗timestamp, int8u ∗userData)
- int16u halAppBootloaderGetVersion (void)
- int16u halAppBootloaderGetRecoveryVersion (void)

6.46.1 Detailed Description

Defiunition of the application bootloader interface. Some functions in this file return an EmberStatus value. See error-def.h for definitions of all EmberStatus return values.

See bootloader-interface-app.h for source code.

6.46.2 Macro Definition Documentation

6.46.2.1 #define BOOTLOADER_SEGMENT_SIZE_LOG2

This is the working unit of data for the app bootloader. We want it as big as possible, but it must be a factor of the NVM page size and fit into a single Zigbee packet. We choose $2^6 = 64$ bytes.

Definition at line 27 of file bootloader-interface-app.h.

6.46.2.2 #define BOOTLOADER_SEGMENT_SIZE

This is the working unit of data for the app bootloader. We want it as big as possible, but it must be a factor of the NVM page size and fit into a single Zigbee packet. We choose $2^6 = 64$ bytes.

Definition at line 32 of file bootloader-interface-app.h.

6.46.2.3 #define BL␣IMAGE␣IS␣VALID␣CONTINUE

Define a numerical value for checking image validity when calling the image interface functions.

Definition at line 65 of file bootloader-interface-app.h.

6.46.3 Function Documentation

6.46.3.1 int8u halAppBootloaderInit (void)

Call this function as part of your application initialization to ensure the storage mechanism is ready to use. Note: some earlier drivers may assert instead of returning an error if initialization fails.

Returns

> ::EEPROM_SUCCESS or ::EEPROM_ERR_INVALID_CHIP

6.46.3.2 const HalEepromInformationType∗ halAppBootloaderInfo (void)

Call this function to get information about the attached storage device and its capabilities.

Returns

> A pointer to a HalEepromInformationType data structure, or NULL if the driver does not support this API

6.46.3.3 void halAppBootloaderShutdown (void)

Call this function when you are done accessing the storage mechanism to ensure that it is returned to its lowest power state.

6.46.3.4 void halAppBootloaderImageIsValidReset (void)

Call this function once before checking for a valid image to reset the call flag.

6.46.3.5 int16u halAppBootloaderImageIsValid (void)

Reads the app image out of storage, calculates the total file CRC to verify the image is intact.

Caller should loop calling this function while it returns BL_IMAGE_IS_VALID_CONTINUE to get final result. This allows caller to service system needs during validation.

Call halAppBootloaderImageIsValidReset() before calling halAppBootloaderImageIsValid() to reset the call flag.

Here is an example application call:

```
halAppBootloaderImageIsValidReset();
while ( (pages = halAppBootloaderImageIsValid() )
        == BL_IMAGE_IS_VALID_CONTINUE) {
  // make app specific calls here, if any
  emberTick();
}
```

Returns

One of the following:

- Number of pages in a valid image
- 0 for an invalid image
- BL_IMAGE_IS_VALID_CONTINUE (-1) to continue to iterate for the final result.

6.46.3.6   EmberStatus halAppBootloaderInstallNewImage (void)

Invokes the bootloader to install the application in storage. This function resets the device to start the bootloader code and does not return!

6.46.3.7   int8u halAppBootloaderWriteRawStorage (int32u *address,* const int8u ∗ *data,* int16u *len*)

Writes data to the specified raw storage address and length without being restricted to any page size Note-: Not all storage implementations support accesses that are not page aligned, refer to the HalEeprom-InformationType structure for more information. Note: Some storage devices require contents to be erased before new data can be written, and will return an ::EEPROM_ERR_ERASE_REQUIRED error if write is called on a location that is not already erased. Refer to the HalEepromInformationType structure to see if the attached storage device requires erasing.

Parameters

address	Address to start writing data
data	A pointer to the buffer of data to write.
len	Length of the data to write

Returns

::EEPROM_SUCCESS or ::EEPROM_ERR.

6.46.3.8   int8u halAppBootloaderReadRawStorage (int32u *address,* int8u ∗ *data,* int16u *len*)

Reads data from the specified raw storage address and length without being restricted to any page size Note: Not all storage implementations support accesses that are not page aligned, refer to the HalEeprom-InformationType structure for more information.

Parameters

address	Address from which to start reading data
data	A pointer to a buffer where data should be read into
len	Length of the data to read

Returns

::EEPROM_SUCCESS or ::EEPROM_ERR.

6.46.3.9   int8u halAppBootloaderEraseRawStorage (int32u *address,* int32u *len*)

Erases the specified region of the storage device. Note: Most devices require the specified region to be page aligned, and will return an error if an unaligned region is specified. Note: Many devices take an extremely long time to perform an erase operation. When erasing a large region, it may be preferable to make multiple calls to this API so that other application functionality can be performed while the erase is in progress. The halAppBootloaderStorageBusy() API may be used to determine when the last erase operation has completed. Erase timing information can be found in the HalEepromInformationType structure.

Parameters

address	Address to start erasing
len	Length of the region to be erased

Returns

::EEPROM_SUCCESS or ::EEPROM_ERR.

6.46.3.10   boolean halAppBootloaderStorageBusy (void)

Determine if the attached storage device is still busy performing the last operation, such as a write or an erase.

Returns

TRUE if still busy or FALSE if not.

6.46.3.11   int8u halAppBootloaderReadDownloadSpace (int16u *pageToBeRead,* int8u ∗ *destRamBuffer*)

Converts pageToBeRead to an address and the calls storage read function. Note: This function is deprecated. It has been replaced by halAppBootloaderReadRawStorage()

Parameters

pageToBeRead	pass in the page to be read. This will be converted to the appropriate address. Pages are ::EEPROM_PAGE_SIZE long.
destRamBuffer	a pointer to the buffer to write to.

Returns

::EEPROM_SUCCESS or ::EEPROM_ERR.

6.46.3.12   int8u halAppBootloaderWriteDownloadSpace (int16u *pageToBeWritten,* int8u ∗ *RamPtr*)

Converts pageToBeWritten to an address and calls the storage write function. Note: This function is deprecated. It has been replaced by halAppBootloaderWriteRawStorage()

Parameters

pageToBe-Written	pass in the page to be written. This will be converted to the appropriate address. Pages are ::EEPROM_PAGE_SIZE long.
RamPtr	a pointer to the data to be written.

Returns

    ::EEPROM_SUCCESS or ::EEPROM_ERR

6.46.3.13 int8u halAppBootloaderGetImageData (int32u ∗ *timestamp,* int8u ∗ *userData*)

Read the application image data from storage.

Parameters

timestamp	write the image timestamp to this data pointer.
userData	write the user data field to this buffer.

Returns

    ::EEPROM_SUCCESS or ::EEPROM_ERR

6.46.3.14 int16u halAppBootloaderGetVersion (void)

Returns the application bootloader version.

6.46.3.15 int16u halAppBootloaderGetRecoveryVersion (void)

Returns the recovery image version.

6.47 Custom Bootloader HAL

Modules

- Common
- Standalone
- Application

6.47.1 Detailed Description

6.48   Common

Modules

- GPIO
- Serial

Typedefs

- typedef int8u BL_Status

Enumerations

- enum { COMM_SERIAL, COMM_RADIO }

Bootloader Status Definitions

These are numerical definitions for the possible bootloader status codes.

- #define BL_SUCCESS
- #define BL_CRC_MATCH
- #define BL_IMG_FLASHED
- #define BL_ERR
- #define BL_ERR_MASK
- #define BL_ERR_HEADER_EXP
- #define BL_ERR_HEADER_WRITE_CRC
- #define BL_ERR_CRC
- #define BL_ERR_UNKNOWN_TAG
- #define BL_ERR_SIG
- #define BL_ERR_ODD_LEN
- #define BL_ERR_BLOCK_INDEX
- #define BL_ERR_OVWR_BL
- #define BL_ERR_OVWR_SIMEE
- #define BL_ERR_ERASE_FAIL
- #define BL_ERR_WRITE_FAIL
- #define BL_ERR_CRC_LEN
- #define BL_ERR_NO_QUERY
- #define BL_ERR_BAD_LEN
- #define BL_ERR_TAGBUF
- #define BL_EBL_CONTINUE

Bootloader State Flags

These are numerical flags for the possible bootloader states. These values are used in the bootloader code for making the current state more verbose.

Note

> The flags do not start at 0 so that they can be output via the serial port during debug and easily screened
> out of normal xmodem traffic which depends only on ACK (0x06) and NAK (0x15).

- #define TIMEOUT
- #define FILEDONE
- #define FILEABORT
- #define BLOCKOK
- #define QUERYFOUND
- #define START_TIMEOUT
- #define BLOCK_TIMEOUT
- #define BLOCKERR_MASK
- #define BLOCKERR_SOH
- #define BLOCKERR_CHK
- #define BLOCKERR_CRCH
- #define BLOCKERR_CRCL
- #define BLOCKERR_SEQUENCE
- #define BLOCKERR_PARTIAL
- #define BLOCKERR_DUPLICATE

6.48.1   Detailed Description

EM35x common bootloader definitions. See bootloader-common.h for source code.

6.48.2   Macro Definition Documentation

6.48.2.1   #define BL_SUCCESS

Numerical definition for a bootloader status code: Success.

Definition at line 45 of file bootloader-common.h.

6.48.2.2   #define BL_CRC_MATCH

Numerical definition for a bootloader status code: CRC match.

Definition at line 48 of file bootloader-common.h.

6.48.2.3   #define BL_IMG_FLASHED

Numerical definition for a bootloader status code: Image flashed.

Definition at line 51 of file bootloader-common.h.

6.48.2.4   #define BL_ERR

Numerical definition for a bootloader status code: serial error.

Definition at line 54 of file bootloader-common.h.

6.48.2.5 #define BL_ERR_MASK

Numerical definition for a bootloader status code: Error mask.

Definition at line 57 of file bootloader-common.h.

6.48.2.6 #define BL_ERR_HEADER_EXP

Numerical definition for a bootloader status code: Failed in header state. Header expected.

Definition at line 61 of file bootloader-common.h.

6.48.2.7 #define BL_ERR_HEADER_WRITE_CRC

Numerical definition for a bootloader status code: Failed write/CRC of header.

Definition at line 65 of file bootloader-common.h.

6.48.2.8 #define BL_ERR_CRC

Numerical definition for a bootloader status code: Failed file CRC.

Definition at line 68 of file bootloader-common.h.

6.48.2.9 #define BL_ERR_UNKNOWN_TAG

Numerical definition for a bootloader status code: Unknown tag.

Definition at line 71 of file bootloader-common.h.

6.48.2.10 #define BL_ERR_SIG

Numerical definition for a bootloader status code: EBL header error.

Definition at line 74 of file bootloader-common.h.

6.48.2.11 #define BL_ERR_ODD_LEN

Numerical definition for a bootloader status code: Trying to flash odd length bytes.

Definition at line 78 of file bootloader-common.h.

6.48.2.12 #define BL_ERR_BLOCK_INDEX

Numerical definition for a bootloader status code: Indexed past end of block buffer.

Definition at line 82 of file bootloader-common.h.

6.48.2.13 #define BL_ERR_OVWR_BL

Numerical definition for a bootloader status code: Attempt to overwrite bootloader flash.

Definition at line 86 of file bootloader-common.h.

6.48.2.14 #define BL_ERR_OVWR_SIMEE

Numerical definition for a bootloader status code: Attempt to overwrite Simulated EEPROM flash.

Definition at line 90 of file bootloader-common.h.

6.48.2.15 #define BL_ERR_ERASE_FAIL

Numerical definition for a bootloader status code: Flash erase failed.

Definition at line 94 of file bootloader-common.h.

6.48.2.16 #define BL_ERR_WRITE_FAIL

Numerical definition for a bootloader status code: Flash write failed.

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

6.48.2.17 #define BL_ERR_CRC_LEN

Numerical definition for a bootloader status code: END tag CRC wrong length.

Definition at line 102 of file bootloader-common.h.

6.48.2.18 #define BL_ERR_NO_QUERY

Numerical definition for a bootloader status code: Received data before query request/response.

Definition at line 106 of file bootloader-common.h.

6.48.2.19 #define BL_ERR_BAD_LEN

Numerical definition for a bootloader status code: Invalid length detected.

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

6.48.2.20 #define BL_ERR_TAGBUF

Numerical definition for a bootloader status code: Problem with tagBuf detected.

Definition at line 114 of file bootloader-common.h.

6.48.2.21 #define BL_EBL_CONTINUE

Numerical definition for a bootloader status code: processEbl deferred, call again to continue.

Definition at line 118 of file bootloader-common.h.

6.48.2.22 #define TIMEOUT

Bootloader state flag.

Definition at line 132 of file bootloader-common.h.

6.48.2.23 #define FILEDONE

Bootloader state flag.

Definition at line 133 of file bootloader-common.h.

6.48.2.24 #define FILEABORT

Bootloader state flag.

Definition at line 134 of file bootloader-common.h.

6.48.2.25 #define BLOCKOK

Bootloader state flag.

Definition at line 135 of file bootloader-common.h.

6.48.2.26 #define QUERYFOUND

Bootloader state flag.

Definition at line 136 of file bootloader-common.h.

6.48.2.27 #define START_TIMEOUT

Bootloader state flag.

Definition at line 137 of file bootloader-common.h.

6.48.2.28 #define BLOCK_TIMEOUT

Bootloader state flag.

Definition at line 138 of file bootloader-common.h.

6.48.2.29 #define BLOCKERR_MASK

Bootloader state flag.

Definition at line 139 of file bootloader-common.h.

6.48.2.30 #define BLOCKERR_SOH

Bootloader state flag: Start Of Header not received.

Definition at line 142 of file bootloader-common.h.

6.48.2.31 #define BLOCKERR_CHK

Bootloader state flag: Sequence of bytes don't match.

Definition at line 145 of file bootloader-common.h.

6.48.2.32 #define BLOCKERR_CRCH

Bootloader state flag: CRC High byte failure.

Definition at line 148 of file bootloader-common.h.

6.48.2.33 #define BLOCKERR_CRCL

Bootloader state flag: CRC Low byte failure.

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

6.48.2.34 #define BLOCKERR_SEQUENCE

Bootloader state flag: Block received out of sequence.

Definition at line 154 of file bootloader-common.h.

6.48.2.35 #define BLOCKERR_PARTIAL

Bootloader state flag: Partial block received.

Definition at line 157 of file bootloader-common.h.

6.48.2.36 #define BLOCKERR_DUPLICATE

Bootloader state flag: Duplicate of previous block.

Definition at line 160 of file bootloader-common.h.

6.48.3 Typedef Documentation

6.48.3.1 typedef int8u BL_Status

Define the bootloader status type.

Definition at line 37 of file bootloader-common.h.

6.48.4 Enumeration Type Documentation

6.48.4.1 anonymous enum

Enumerator:

COMM_SERIAL
COMM_RADIO

Definition at line 165 of file bootloader-common.h.

6.49 GPIO

Enumerations

- enum blState_e {
  BL_ST_UP, BL_ST_DOWN, BL_ST_POLLING_LOOP, BL_ST_DOWNLOAD_LOOP,
  BL_ST_DOWNLOAD_FAILURE, BL_ST_DOWNLOAD_SUCCESS }

Functions

- void bootloadGpioInit (void)
- void bootloadStateIndicator (enum blState_e state)
- boolean bootloadForceActivation (void)

State Indicator Macros

The bootloader indicates which state it is in by calling these macros. Map them to the ::halBootloadState-Indicator function (in bootloder-gpio.c) if you want to display that bootloader state. Used to blink the LED's or otherwise signal bootloader activity.

- #define BL_STATE_UP()
- #define BL_STATE_DOWN()
- #define BL_STATE_POLLING_LOOP()
- #define BL_STATE_DOWNLOAD_LOOP()
- #define BL_STATE_DOWNLOAD_SUCCESS()
- #define BL_STATE_DOWNLOAD_FAILURE()

6.49.1 Detailed Description

EM35x bootloader GPIO definitions. See bootloader-gpio.h for source code.

6.49.2 Macro Definition Documentation

6.49.2.1 #define BL_STATE_UP( )

Finished init sequence, ready for bootload.

Definition at line 28 of file bootloader-gpio.h.

6.49.2.2 #define BL_STATE_DOWN( )

Called right before bootloader resets to application. Use to cleanup and reset GPIO's to leave node in known state for app start, if necessary.

Definition at line 34 of file bootloader-gpio.h.

6.49.2.3 #define BL_STATE_POLLING_LOOP( )

Standalone bootloader polling serial/radio interface.

Definition at line 38 of file bootloader-gpio.h.

6.49.2.4 #define BL_STATE_DOWNLOAD_LOOP( )

Processing download image.

Definition at line 42 of file bootloader-gpio.h.

6.49.2.5 #define BL_STATE_DOWNLOAD_SUCCESS( )

Download process was a success.

Definition at line 46 of file bootloader-gpio.h.

6.49.2.6 #define BL_STATE_DOWNLOAD_FAILURE( )

Download process failed.

Definition at line 50 of file bootloader-gpio.h.

6.49.3 Enumeration Type Documentation

6.49.3.1 enum blState_e

Defines various bootloader states. Use in LED code to signal bootload activity.

Enumerator:

> *BL_ST_UP* bootloader up
> *BL_ST_DOWN* bootloader going down
> *BL_ST_POLLING_LOOP* polling interfaces
> *BL_ST_DOWNLOAD_LOOP* downloading
> *BL_ST_DOWNLOAD_FAILURE* download failure
> *BL_ST_DOWNLOAD_SUCCESS* download success

Definition at line 56 of file bootloader-gpio.h.

6.49.4 Function Documentation

6.49.4.1 void bootloadGpioInit (void)

Initialize GPIO.

6.49.4.2 void bootloadStateIndicator (enum blState_e *state*)

Helper function used for displaying bootloader state (for example: with LEDs).

6.49.4.3   boolean bootloadForceActivation (void)

Force activation of bootloader.

6.50 Serial

Functions

- void serInit (void)
- void serPutFlush (void)
- void serPutChar (int8u ch)
- void serPutStr (const char ∗str)
- void serPutBuf (const int8u buf[ ], int8u size)
- void serPutDecimal (int16u val)
- void serPutHex (int8u byte)
- void serPutHexInt (int16u word)
- boolean serCharAvailable (void)
- int8u serGetChar (int8u ∗ch)
- void serGetFlush (void)

6.50.1 Detailed Description

EM35x common bootloader serial definitions. See bootloader-serial.h for source code.

6.50.2 Function Documentation

6.50.2.1 void serInit (void)

Initialize serial port.

6.50.2.2 void serPutFlush (void)

Flush the transmiter.

6.50.2.3 void serPutChar (int8u *ch*)

Transmit a character.

Parameters

ch	A character.

6.50.2.4 void serPutStr (const char ∗ *str*)

Transmit a string.

Parameters

str	A string.

6.50.2.5 void serPutBuf (const int8u *buf[ ],* int8u *size*)

Transmit a buffer.

Parameters

buf	A buffer.
size	Length of buffer.

6.50.2.6 void serPutDecimal (int16u *val*)

Transmit a 16bit value in decimal.

Parameters

val	The data to print.

6.50.2.7 void serPutHex (int8u *byte*)

Transmit a byte as hex.

Parameters

byte	A byte.

6.50.2.8 void serPutHexInt (int16u *word*)

Transmit a 16bit integer as hex.

Parameters

word	A 16bit integer.

6.50.2.9 boolean serCharAvailable (void)

Determine if a character is available.

Returns

TRUE if a character is available, FALSE otherwise.

6.50.2.10 int8u serGetChar (int8u ∗ *ch*)

Get a character if available, otherwise return an error.

Parameters

ch	Pointer to a location where the received byte will be placed.

Returns

   [BL_SUCCESS](#) if a character was obtained, [BL_ERR](#) otherwise.

6.50.2.11   void serGetFlush (void)

Flush the receiver.

6.51 Standalone

Required Custom Functions

- void bootloaderMenu (void)

Available Bootloader Library Functions

Functions implemented by the bootloader library that may be used by custom functions.

- BL_Status receiveImage (int8u commState)
- boolean checkDebugMenuOption (int8u ch)
- BL_Status initOtaState (void)
- BL_Status checkOtaStart (void)
- BL_Status receiveOtaImage (void)
- boolean paIsPresent (void)

6.51.1 Detailed Description

EM35x standalone bootloader public definitions. See standalone-bootloader.h for source code.

6.51.2 Function Documentation

6.51.2.1 void bootloaderMenu (void)

This function must be implemented, providing a bootloader menu.

6.51.2.2 BL_Status receiveImage (int8u *commState*)

Puts the bootloader into a mode where it will receive an image. commState indicates whether the image is received via serial (COMM_SERIAL) or over the air (COMM_RADIO)

6.51.2.3 boolean checkDebugMenuOption (int8u *ch*)

A hook to the bootloader library for it to check for extra menu options. Only used for ember internal debug builds, not normally needed.

Returns

TRUE if the option was handled, FALSE if not.

6.51.2.4 BL_Status initOtaState (void)

Initialize OTA Bootloader state.

Note

OTA support hooks are subject to change!

Returns

>   BL_Status of the success of the function.

6.51.2.5   BL_Status checkOtaStart (void)

Check to see if the bootloader has detected an OTA upload start.

Note

>   OTA support hooks are subject to change!

Returns

>   BL_Status of the success of the function.

6.51.2.6   BL_Status receiveOtaImage (void)

Puts the bootloader into a mode where it will receive an image over the air. The function checkOtaStart() should have been called first and it should have returned with a status of BL_SUCCESS before calling this function.

Note

>   OTA support hooks are subject to change!

Returns

>   BL_Status of the success of the function.

6.51.2.7   boolean palsPresent (void)

Uses the information in the PHY_CONFIG token to determine if a power amplifier is present in the node design.

Note

>   This function must not be called before emBootloaderRadioBoot().

Returns

>   TRUE if a power amplifier is present, FALSE otherwise.

6.52   Application

Required Custom Functions

- void bootloaderAction (boolean runRecovery)

Available Bootloader Library Functions

Functions implemented by the bootloader library that may be used by custom functions.

- BL_Status recoveryMode (void)
- BL_Status processImage (boolean install)

6.52.1   Detailed Description

EM35x application bootloader and generic EEPROM Interface. See app-bootloader.h for source code.

6.52.2   Function Documentation

6.52.2.1   void bootloaderAction (boolean *runRecovery*)

Drives the app bootloader. If the ::runRecovery parameter is TRUE, the recovery mode should be activated, otherwise it should attempt to install an image. This function should not return. It should always exit by resetting the the bootloader.

Parameters

runRecovery	If TRUE, recover mode is activated. Otherwise, normal image installation is activated.

6.52.2.2   BL_Status recoveryMode (void)

Activates recoveryMode to receive a new image over xmodem.

Returns

BL_SUCCESS if an image was successfully received.

6.52.2.3   BL_Status processImage (boolean *install*)

Processes an image in the external eeprom.

Parameters

install	If FALSE, it will simply validate the image without touching main flash. If TRUE, the image will be programmed to main flash.

Returns

BL_SUCCESS if an image was successfully installed/validated

6.53 Application Utilities API Reference

Modules

- Forming and Joining Networks
- ZigBee Device Object (ZDO) Information
- Message Fragmentation
- Network Manager
- Serial Communication

6.53.1 Detailed Description

The Application Utilities API consists of sample utilities you can modify and use in your applications.

6.54   Forming and Joining Networks

Macros

- #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)
- void emberFormAndJoinCleanup (EmberStatus status)

Variables

- boolean emberEnableDualChannelScan

6.54.1   Detailed Description

Functions for finding an existing network to join and for finding an unused PAN id with which to form a network.

Summary of application requirements:

For the SOC:

- Define ::EMBER_APPLICATION_HAS_ENERGY_SCAN_RESULT_HANDLER in the configuration header.

- Call emberFormAndJoinTick() regularly in the main loop.

- Include form-and-join.c and form-and-join-node-adapter.c in the build.

- Optionally include form-and-join-node-callbacks.c in the build.

- If processor idling is desired: – Call emberFormAndJoinTaskInit() to initialize the form and join task – Call emberFormAndJoinRunTask() regularly in the main loop instead of emberFormAndJoinTick()

For an EZSP Host:

- Define ::EZSP_APPLICATION_HAS_ENERGY_SCAN_RESULT_HANDLER in the configuration header.

- Include form-and-join.c and form-and-join-host-adapter.c in the build.

- Optionally include form-and-join-host-callbacks.c in the build.

For either platform, the application can omit the form-and-join-∗-callback.c file from the build and implement the callbacks itself if necessary. In this case the appropriate form-and-join callback function must be called from within each callback, as is done within the form-and-join-∗-callback.c files.

On either platform, FORM_AND_JOIN_MAX_NETWORKS can be explicitly defined to limit (or expand) the number of joinable networks that the library will save for consideration during the scan process.

The library is able to resume scanning for joinable networks from where it left off, via a call to emberScanForNextJoinableNetwork(). Thus if the first joinable network found is not the correct one, the application can continue scanning without starting from the beginning and without finding the same network that it has already rejected. The library can also be used on the host processor.

6.54.2   Macro Definition Documentation

6.54.2.1   #define NETWORK_STORAGE_SIZE

Number of bytes required to store relevant info for a saved network.

This constant represents the minimum number of bytes required to store all members of the NetworkInfo struct used in the adapter code. Its value should not be changed unless the underlying adapter code is updated accordingly. Note that this constant's value may be different than sizeof(NetworkInfo) because some compilers pad the structs to align on word boundaries. Thus, the adapter code stores/retrieves these pieces of data individually (to be platform-agnostic) rather than as a struct.

For efficiency's sake, this number should be kept to a power of 2 and not and not exceed 32 (PACKET_BUFFER_SIZE).

Definition at line 68 of file form-and-join.h.

6.54.2.2   #define NETWORK_STORAGE_SIZE_SHIFT

Log_base2 of NETWORK_STORAGE_SIZE.

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

6.54.2.3   #define FORM_AND_JOIN_MAX_NETWORKS

Number of joinable networks that can be remembered during the scan process.

Note for SoC Platforms: This is currently limited to a maximum of 15 due to the size of each network entry (16 bytes) and the EmberMessageBuffer API's requirement that total buffer storage length be kept to an 8-bit quantity (less than 256).

Note for EZSP Host Platforms: In the host implementation of this library, the storage size for the detected networks buffer is controlled by ::EZSP_HOST_FORM_AND_JOIN_BUFFER_SIZE, so that limits the highest value that the host can set for FORM_AND_JOIN_MAX_NETWORKS.

Definition at line 94 of file form-and-join.h.

6.54.3 Function Documentation

6.54.3.1 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.

6.54.3.2 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.

6.54.3.3 EmberStatus emberScanForNextJoinableNetwork (void)

See emberScanForJoinableNetwork().

6.54.3.4 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.

6.54.3.5 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	

6.54.3.6 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 ember-ScanForNextJoinableNetwork().

Parameters

networkFound	
lqi	The lqi value of the received beacon.
rssi	The rssi value of the received beacon.

6.54.3.7 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	

6.54.3.8 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.

6.54.3.9 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.

6.54.3.10 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.

6.54.3.11 void emberFormAndJoinTick (void)

Used by the form and join code on the node to time out a joinable scan after 30 seconds of inactivity. The application must call emberFormAndJoinTick() regularly. This function does not exist for the EZSP host library.

6.54.3.12 void emberFormAndJoinTaskInit (void)

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

6.54.3.13 void emberFormAndJoinRunTask (void)

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

6.54.3.14 void emberFormAndJoinCleanup (EmberStatus *status*)

When form-and-join state is no longer needed, the application can call this routine to cleanup and free resources. On the SOC platforms this will free the allocated message buffer.

6.54.4 Variable Documentation

6.54.4.1 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.

6.55   ZigBee Device Object (ZDO) Information

Macros

- #define ZDO_MESSAGE_OVERHEAD

Service Discovery Functions

- EmberStatus emberMatchDescriptorsRequest (EmberNodeId target, int16u profile, EmberMessage-Buffer inClusters, EmberMessageBuffer outClusters, EmberApsOption options)

Binding Manager Functions

- EmberStatus emberEndDeviceBindRequest (int8u endpoint, EmberApsOption options)

Function to Decode Address Response Messages

- EmberNodeId emberDecodeAddressResponse (EmberMessageBuffer response, EmberEUI64 eui64-Return)

Service Discovery Functions

- EmberStatus emberNodeDescriptorRequest (EmberNodeId target, EmberApsOption options)
- EmberStatus emberPowerDescriptorRequest (EmberNodeId target, EmberApsOption options)
- EmberStatus emberSimpleDescriptorRequest (EmberNodeId target, int8u targetEndpoint, Ember-ApsOption 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 destination-Endpoint, EmberApsOption options)
- EmberStatus emberUnbindRequest (EmberNodeId target, EmberEUI64 source, int8u sourceEndpoint, int16u clusterId, int8u type, EmberEUI64 destination, EmberMulticastId groupAddress, int8u destination-Endpoint, 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 leave-RequestFlags, EmberApsOption options)

- EmberStatus emberPermitJoiningRequest (EmberNodeId target, int8u duration, int8u authentication, EmberApsOption options)
- void emberSetZigDevRequestRadius (int8u radius)
- int8u emberGetZigDevRequestRadius (void)
- int8u emberGetLastZigDevRequestSequence (void)
- int8u emberGetLastAppZigDevRequestSequence (void)

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)

6.55.1 Detailed Description

For getting information about nodes of a ZigBee network via a ZigBee Device Object (ZDO). See zigbee-device-library.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 emberSendBroadcast() or emberSendUnicast().

The result of the send is reported to the application as normal via emberMessageSentHandler().

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 emberDecodeAddressResponse().

```
void emberIncomingMessageHandler(
      EmberIncomingMessageType type,
                                EmberApsFrame *apsFrame,
                                EmberMessageBuffer message)
{
  if (apsFrame->profileId == EMBER_ZDO_PROFILE_ID)
      {
    switch (apsFrame->clusterId) {
    case NETWORK_ADDRESS_RESPONSE:
    case IEEE_ADDRESS_RESPONSE:
      {
        EmberEUI64 eui64;
        EmberNodeId nodeId = emberDecodeAddressResponse
      (message, eui64);
        // Use nodeId and eui64 here.
        break;
      }
    default:
      // Handle other incoming ZDO responses here.
    }
  } else {
    // Handle incoming application messages here.
  }
}
```

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.
  }
}
```

6.55.2 Macro Definition Documentation

6.55.2.1 #define ZDO_MESSAGE_OVERHEAD

ZDO messages start with a sequence number.

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

6.55.3 Function Documentation

6.55.3.1 EmberStatus emberMatchDescriptorsRequest (EmberNodeId *target,* int16u *profile,* EmberMessageBuffer *inClusters,* EmberMessageBuffer *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.
inClusters	The list of input clusters. To not match any input clusters, use EMBER_NULL_MESSAGE_BUFFER.
outClusters	The list of output clusters. To not match any output clusters, use EMBER_NULL_MESSAGE_BUFFER.
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.

6.55.3.2 EmberStatus emberEndDeviceBindRequest (int8u *endpoint,* EmberApsOption *options*)

An end device bind request to the coordinator. The simple descriptor of the specified endpoint is used to construct the request. If the coordinator receives a second end device bind request then a binding is created for every matching cluster.

Parameters

endpoint	The endpoint on the local device whose simple descriptor will be used to create the request.
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.

6.55.3.3 EmberNodeId emberDecodeAddressResponse (EmberMessageBuffer *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.

6.55.3.4 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.

6.55.3.5 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.

6.55.3.6 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.

6.55.3.9 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 U-NICAST_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 UNICA-ST_MANY_TO_ONE_BINDING.
groupAddress	The group address for the MULTICAST_BINDING.
destination-Endpoint	The destination endpoint in the binding entry for the UNICAST_BINDING or UNIC-AST_MANY_TO_ONE_BINDING.
options	The options to use when sending the request. See emberSendUnicast() for a description.

Returns

An EmberStatus value.

- EMBER_SUCCESS
- EMBER_NO_BUFFERS _ EMBER_NETWORK_DOWN
- EMBER_NETWORK_BUSY

6.55.3.10 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_DO-WN or EMBER_NETWORK_BUSY.

6.55.3.11 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_DO-WN or EMBER_NETWORK_BUSY.

6.55.3.12 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_B-INDING 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_DO-WN or EMBER_NETWORK_BUSY.

6.55.3.13 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.
leaveRequest-Flags	A bitmask of leave options. Include LEAVE_REQUEST_REMOVE_CHILDREN_F-LAG if the target is to remove their children and/or LEAVE_REQUEST_REJOIN_F-LAG 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_DO-WN or EMBER_NETWORK_BUSY.

6.55.3.14 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_DO-WN or EMBER_NETWORK_BUSY.

6.55.3.15 void emberSetZigDevRequestRadius (int8u *radius*)

Change the default radius for broadcast ZDO requests.

Parameters

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

6.55.3.16 int8u emberGetZigDevRequestRadius (void)

Retrieve the default radius for broadcast ZDO requests.

Returns

The radius to be used for future ZDO request broadcasts.

6.55.3.17 int8u emberGetLastZigDevRequestSequence (void)

Provide access to the application ZDO transaction sequence number for last request. This function has been deprecated and replaced by emberGetLastAppZigDevRequestSequence().

Returns

Last application ZDO transaction sequence number used

6.55.3.18 int8u emberGetLastAppZigDevRequestSequence (void)

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

Returns

Last application ZDO transaction sequence number used

6.55.3.19 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.

6.55.3.20 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

6.55.3.21 **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_DO-WN or EMBER_NETWORK_BUSY.

6.55.3.22 **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_DO-WN or EMBER_NETWORK_BUSY.

6.55.3.23 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.

6.56   Message Fragmentation

Transmitting

- EmberStatus emberFragmentSendUnicast (EmberOutgoingMessageType type, int16u indexOrDestination, EmberApsFrame ∗apsFrame, EmberMessageBuffer payload, int8u maxFragmentSize)
- boolean emberFragmentMessageSent (EmberApsFrame ∗apsFrame, EmberMessageBuffer buffer, EmberStatus status)
- void emberFragmentMessageSentHandler (EmberStatus status)

Receiving

- boolean emberFragmentIncomingMessage (EmberApsFrame ∗apsFrame, EmberMessageBuffer payload)
- void emberFragmentTick (void)

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)

6.56.1   Detailed Description

Splits long messages into smaller blocks for transmission and reassembles received blocks. See fragment.h for source code.

EMBER_FRAGMENT_WINDOW_SIZE controls how many blocks are sent at a time. EMBER_FRAG-MENT_DELAY_MS controls the spacing between blocks.

To send a long message, the application calls emberFragmentSendUnicast(). The application must add a call to emberFragmentMessageSent() at the start of its emberMessageSentHandler(). If emberFragment-MessageSent() returns TRUE, the fragmentation code has handled the event and the application must not process it further. The fragmentation code calls the application-defined emberFragmentMessageSent-Handler() when it has finished sending the long message.

To receive a long message, the application must add a call to emberFragmentIncomingMessage() at the start of its emberIncomingMessageHandler(). If emberFragmentIncomingMessage() returns TRUE, the

fragmentation code has handled the message and the application must not process it further. The application must also call emberFragmentTick() regularly.

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 ezspFragmentMessage-Sent() returns TRUE, the fragmentation code has handled the event and the application must not process it further. The fragmentation code calls the application-defined ezspFragmentMessageSentHandler() when it has finished sending the long message.

To receive a long message, the application must add a call to ezspFragmentIncomingMessage() at the start of its ezspIncomingMessageHandler(). If ezspFragmentIncomingMessage() returns TRUE, the fragmentation code has handled the message and the application must not process it further. The application must also call ezspFragmentTick() regularly.

6.56.2 Function Documentation

6.56.2.1 EmberStatus emberFragmentSendUnicast (EmberOutgoingMessageType *type,* int16u *indexOrDestination,* EmberApsFrame * *apsFrame,* EmberMessageBuffer *payload,* int8u *maxFragmentSize*)

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_DIRE-CT, EMBER_OUTGOING_VIA_ADDRESS_TABLE, or EMBER_OUTGOING_V-IA_BINDING.
indexOr-Destination	Depending on the type of addressing used, this is either the EmberNodeId of the desti-nation, an index into the address table, or an index into the binding table.
apsFrame	The APS frame for the message.
payload	The long message to be sent.
maxFragment-Size	The message will be broken into blocks no larger than this.

Returns

An EmberStatus value.

- EMBER_SUCCESS
- EMBER_MESSAGE_TOO_LONG
- EMBER_NO_BUFFERS
- EMBER_NETWORK_DOWN
- EMBER_NETWORK_BUSY
- EMBER_INVALID_CALL is returned if the payload length is zero or if the window size (EM-BER_FRAGMENT_WINDOW_SIZE) is zero.

6.56.2.2 boolean emberFragmentMessageSent (EmberApsFrame ∗ *apsFrame,* EmberMessageBuffer *buffer,* EmberStatus *status*)

The application must call this function at the start of its emberMessageSentHandler(). 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 emberMessageSentHandler().
buffer	The buffer passed to emberMessageSentHandler().
status	The status passed to emberMessageSentHandler().

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 emberMessageSentHandler(). Returns FALSE otherwise. The fragmentation code has not handled the event so the application must continue to process it.

6.56.2.3 void emberFragmentMessageSentHandler (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_NO_BUFFERS, EMBER_NETWORK_DOWN or EMBER_NETWORK_BUSY.

6.56.2.4 boolean emberFragmentIncomingMessage (EmberApsFrame ∗ *apsFrame,* EmberMessageBuffer *payload*)

The application must call this function at the start of its emberIncomingMessageHandler(). 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 emberIncomingMessageHandler().
payload	The payload passed to emberIncomingMessageHandler().

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 emberIncomingMessageHandler(). Returns FALSE if the incoming message was not part of a long message. The fragmentation code has not handled the message so the application must continue to process it. Returns FALSE if the incoming message was a block that completed a long message. The fragmentation code replaces the message with the reassembled long message so the application must continue to process it.

6.56.2.5 void emberFragmentTick (void)

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

6.56.2.6 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

receiveBuffer-Length	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().

6.56.2.7 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.
indexOr-Destination	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.
maxFragment-Size	The message will be broken into blocks no larger than this.
messageLength	The length of the messageContents parameter in bytes.
message-Contents	The long message to be sent.

Returns

An EmberStatus value.

- EMBER_SUCCESS
- EMBER_MESSAGE_TOO_LONG
- EMBER_NETWORK_DOWN
- EMBER_NETWORK_BUSY
- EMBER_INVALID_CALL is returned if messageLength is zero or if the window size (::EZSP-_CONFIG_FRAGMENT_WINDOW_SIZE) is zero.

6.56.2.8 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_HANDLE-R in its configuration header if it defines this callback.

Returns

EMBER_SUCCESS if the source route has been set. Any other value will abort transmission of the current long message.

6.56.2.9 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 FAL-SE otherwise. The fragmentation code has not handled the event so the application must continue to process it.

6.56.2.10 void ezspFragmentMessageSentHandler (EmberStatus *status*)

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

Parameters

status	EMBER_SUCCESS if all the blocks of the long message were delivered to the destination, otherwise EMBER_DELIVERY_FAILED, EMBER_NETWORK_DOWN or EMBER_NETWORK_BUSY.

6.56.2.11 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 TR-UE, the fragmentation code has handled the message and the application must not process it further. When the final block of a long message is received, this function replaces the message with the reassembled long message and returns FALSE so that the application processes it.

Parameters

apsFrame	The APS frame passed to ezspIncomingMessageHandler().
sender	The sender passed to ezspIncomingMessageHandler().
messageLength	A pointer to the message length passed to ezspIncomingMessageHandler().
message-Contents	A pointer to the message contents passed to 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 ezspIncomingMessage-Handler(). Returns FALSE if the incoming message was not part of a long message. The fragmentation code has not handled the message so the application must continue to process it. Returns FALSE if the incoming message was a block that completed a long message. The fragmentation code replaces the message with the reassembled long message so the application must continue to process it.

6.56.2.12 void ezspFragmentTick (void)

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

6.57   Network Manager

Macros

- #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)

6.57.1   Detailed Description

The network manager is an optional function of one device in the ZigBee network. Devices on the network send unsolicited ZDO energy scan reports to the network manager when more than 25% of unicasts fail within a rolling window, but no more than once every 15 minutes.

See network-manager.h for source code.

The network manager is the coordinator by default but can be changed via emberSetNetworkManager-Request(). 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 ember-FindAndRejoinNetwork to get back to the right channel.

Two implementations of this library are provided: network-manager.c, and network-manager-lite.c. The former keeps track of the mean and deviation of the energy on each channel and uses these stats to choose the channel to change to. This consumes a fair amount of RAM. The latter takes the simpler (and possibly more effective) approach of just avoiding past bad channels. Application developers are encouraged to use and modify either of these solutions to take into account their own application-specific needs.

6.57.2 Macro Definition Documentation

6.57.2.1 #define NM_WARNING_LIMIT

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

6.57.2.2 #define NM_WINDOW_SIZE

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

6.57.2.3 #define NM_CHANNEL_MASK

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

6.57.2.4 #define NM_WATCHLIST_SIZE

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

6.57.3 Function Documentation

6.57.3.1 void nmUtilWarningHandler (void)

callback called when unsolicited scan reports hit limit. This callback must be implemented by the application. It is called when the number of unsolicited scan reports received within NM_WINDOW_LIMIT minutes reaches NM_WARNING_LIMIT.

6.57.3.2 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	

6.57.3.3 EmberStatus nmUtilChangeChannelRequest (void)

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

6.58   Serial Communication

Macros

- #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 emberSerialReadData (int8u port, int8u ∗data, int16u length, int16u ∗bytesRead)
- EmberStatus emberSerialReadDataTimeout (int8u port, int8u ∗data, int16u length, int16u ∗bytes-Read, int16u firstByteTimeout, int16u subsequentByteTimeout)
- 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)
- EmberStatus emberSerialWriteBuffer (int8u port, EmberMessageBuffer buffer, int8u start, int8u length)
- XAP2B_PAGEZERO_ON EmberStatus emberSerialWaitSend (int8u port)
- XAP2B_PAGEZERO_OFF EmberStatus emberSerialGuaranteedPrintf (int8u port, PGM_P format-String,...)
- 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 flushHandler, int8u port, PGM_P string, va_list args)

6.58.1   Detailed Description

Unless otherwise noted, the EmberNet stack does not use these functions, and therefore the HAL is not required to implement them. However, many of the supplied example applications do use them. On some platforms, they are also required by DEBUG builds of the stack

Many of these functions return an EmberStatus value. See stack/include/error-defs.h for definitions of all EmberStatus return values. See app/util/serial/serial.h for source code. To use these serial routines, they must be properly configured.

If the Ember serial library is built using EMBER_SERIAL_USE_STDIO, then the Ember serial code will redirect to stdio.h. EMBER_SERIAL_USE_STDIO will not consume any of the usual Ember serial library buffers and does not require use of any of the other EMBER_SERIALx definitions described here. In this mode, the only required lower layers are:

- putchar()

- getchar()

- fflush(stdout)

- halInternalUartInit()

- halInternalPrintfWriteAvailable()

- halInternalPrintfReadAvailable()

- halInternalForcePrintf()

    The functions can work in two ways, depending on how messages waiting for transmission are stored:

    - Buffered mode: Uses stack linked buffers. This method can be more efficient if many messages received over the air also need to be transmitted over the serial interface.
    - FIFO mode: Uses a statically allocated queue of bytes, and data to be transmitted is copied into the queue.

(These modes deal only with data transmission. Data **reception** always occurs in a FIFO mode.)

The current version of these sources provides support for as many as two serial ports, but it can be easily extended. The ports are numbered 0 and 1 and should be accessed using those numbers. The ports can be set up independently of each other.

To enable a port, a Use mode (buffered or FIFO) and a Queue Size must be declared on the port. In FIFO mode, the Queue Size is the size of the FIFO and represents the number of bytes that can be waiting for transmission at any given time. In buffered mode, the Queue Size represents the number of whole messages that can be waiting for transmission at any given time. A single message is created for each call to any of the serial APIs.

To specify a Use mode and Queue Size, place declarations in the compiler preprocessor options when building your application:

- **Use Mode:**

    - ::EMBER_SERIAL0_MODE=EMBER_SERIAL_BUFFER or EMBER_SERIAL_FIFO
    - ::EMBER_SERIAL1_MODE=EMBER_SERIAL_BUFFER or EMBER_SERIAL_FIFO

- **Queue Size:**

    - ::EMBER_SERIAL0_TX_QUEUE_SIZE=2

- ::EMBER_SERIAL0_RX_QUEUE_SIZE=4
- ::EMBER_SERIAL1_TX_QUEUE_SIZE=8
- ::EMBER_SERIAL1_RX_QUEUE_SIZE=16

Note the following:

- If buffered mode is declared, emberSerialBufferTick() should be called in the application's main event loop.

- If buffered mode is declared, the Tx queue size **MUST** be $<= 255$

- On the AVR platform, Rx & Tx queue sizes are limited to powers of $2 <= 128$

- By default, both ports are unused.

You can also use declarations to specify what should be done if an attempt is made to send more data than the queue can accommodate:

- ::EMBER_SERIAL0_BLOCKING

- ::EMBER_SERIAL1_BLOCKING

Be aware that since blocking spins in a loop, doing nothing until space is available, it can adversely affect any code that has tight timing requirements.

If ::EMBER_SERIAL0_BLOCKING or ::EMBER_SERIAL1_BLOCKING is defined, then the call to the port will block until space is available, guaranteeing that the entire message is sent. Note that in buffered mode, even if blocking mode is in effect entire messages may be dropped if insufficient stack buffers are available to hold them. When this happens, EMBER_NO_BUFFERS is returned.

If no blocking mode is defined, the serial code defaults to non-blocking mode. In this event, when the queue is too short, the data that don't fit are dropped. In FIFO mode, this may result bytes being dropped, starting in the middle of message. In buffered mode, the entire message is dropped. When data is dropped, ::EMBER_SERIALTX_OVERFLOW is returned.

To minimize code size, very little error checking is done on the given parameters. Specifying an invalid or unused serial port may result in unexplained behavior. In some cases EMBER_ERR_FATAL may be returned.

6.58.2   Macro Definition Documentation

6.58.2.1   #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 297 of file app/util/serial/serial.h.

6.58.3 Typedef Documentation

6.58.3.1 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 536 of file app/util/serial/serial.h.

6.58.4 Function Documentation

6.58.4.1 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.

6.58.4.2 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.

6.58.4.3 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

6.58.4.4 EmberStatus emberSerialReadData (int8u *port,* int8u ∗ *data,* int16u *length,* int16u ∗ *bytesRead*)

Reads bytes from the specified RX queue. Blocks until the full length has been read or an error occurs. In the event of an error, some valid data may have already been read before the error occurred, in which case that data will be in the buffer pointed to by `data` and the number of bytes successfully read will be placed in `bytesRead`.

Parameters

port	A serial port number (0 or 1).
data	A pointer to storage location for the data. It must be at least `length` in size.
length	The number of bytes to read.
bytesRead	A pointer to a location that will receive the number of bytes read. If the function returns early due to an error, this value may be less than `length`. This parameter may be N-ULL, in which case it is ignored.

Returns

One of the following (see the Main Page):

- EMBER_SERIAL_RX_OVERFLOW if the serial receive fifo was out of space
- EMBER_SERIAL_RX_FRAME_ERROR if a framing error was received
- EMBER_SERIAL_RX_PARITY_ERROR if a parity error was received
- EMBER_SERIAL_RX_OVERRUN_ERROR if the hardware fifo was out of space
- EMBER_SUCCESS if all the data requested is returned

6.58.4.5 EmberStatus emberSerialReadDataTimeout (int8u *port,* int8u ∗ *data,* int16u *length,* int16u ∗ *bytesRead,* int16u *firstByteTimeout,* int16u *subsequentByteTimeout*)

Reads bytes from the specified RX queue, up to a maximum of `length` bytes. The function may return before `length` bytes is read if a timeout is reached or an error occurs. Returns EMBER_SERIAL_RX_-EMPTY if a timeout occurs.

Parameters

port	A serial port number (0 or 1).
data	A pointer to storage location for the data. It must be at least `length` in size.
length	The maximum number of bytes to read.
bytesRead	A pointer to a location that will receive the number of bytes read. If the function returns early due to an error or timeout, this value may be less than `length`. This parameter may be NULL, in which case it is ignored.
firstByte-Timeout	The amount of time, in milliseconds, to wait for the first byte to arrive (if the queue is empty when the function is called). This value must be a minimum of 2 due to the timer resolution.
subsequent-ByteTimeout	The amount of time, in milliseconds, to wait after the previous byte was received for the next byte to arrive. This value must be a minimum of 2 due to the timer resolution.

Returns

One of the following (see the Main Page):

- EMBER_SERIAL_RX_EMPTY if the timeout was exceeded before the requested amount of data was read
- EMBER_SERIAL_RX_OVERFLOW if the serial receive fifo was out of space
- EMBER_SERIAL_RX_FRAME_ERROR if a framing error was received
- EMBER_SERIAL_RX_PARITY_ERROR if a parity error was received
- EMBER_SERIAL_RX_OVERRUN_ERROR if the hardware fifo was out of space
- EMBER_SUCCESS if all the data requested is returned

6.58.4.6 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

6.58.4.7 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.

6.58.4.8 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.

6.58.4.9 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.

6.58.4.10   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.

6.58.4.11   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.

6.58.4.12   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.

6.58.4.13 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.

6.58.4.14 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.

6.58.4.15 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.

6.58.4.16 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.

6.58.4.17   EmberStatus emberSerialWriteBuffer (int8u *port,* EmberMessageBuffer *buffer,* int8u *start,* int8u *length*)

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

Parameters

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

Returns

One of the following (see the Main Page):

- EMBER_SERIAL_TX_OVERFLOW indicates that data was dropped.
- EMBER_NO_BUFFERS indicates that there was an insufficient number of available stack buffers.
- EMBER_SUCCESS.

6.58.4.18   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.

6.58.4.19   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.

6.58.4.20   void emberSerialBufferTick (void)

When a serial port is used in buffered mode, this must be called in an application's main event loop, similar to emberTick(). It frees buffers that are used to queue messages. **Note:** This function has no effect if FIFO mode is being used.

6.58.4.21   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).

6.58.4.22   int8u emPrintfInternal (emPrintfFlushHandler *flushHandler,* int8u *port,* PGM_P *string,* va_list *args*)

The internal printf function, which scans the string for the format specifiers and appropriately implants the passed data into the string.

Parameters

flushHandler,:	The name of an internal function, which has parameters matching the function `em-PrintfFlushHandler` 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).
string	The string to print.
args	The list of arguments for the format specifiers.

Returns

The number of characters written.

6.59   Deprecated Files

6.60 Multi_network

Functions

- int8u emberGetCurrentNetwork (void)
- EmberStatus emberSetCurrentNetwork (int8u index)
- int8u emberGetCallbackNetwork (void)

6.60.1 Detailed Description

See multi-network.h for source code.

6.60.2 Function Documentation

6.60.2.1 int8u emberGetCurrentNetwork (void)

Returns the current network index.

6.60.2.2 EmberStatus emberSetCurrentNetwork (int8u *index*)

Sets the current network.

Parameters

index	The network index.

Returns

EMBER_INDEX_OUT_OF_RANGE if the index does not correspond to a valid network, and EMBER_SUCCESS otherwise.

6.60.2.3 int8u emberGetCallbackNetwork (void)

Can only be called inside an application callback.

Returns

the index of the network the callback refers to. If this function is called outside of a callback, it returns 0xFF.

6.61 Commands2

Data Structures

- struct EmberCommandEntry

    Command entry for a command table.

Macros

- #define MAX_TOKEN_COUNT
- #define emberCommandEntryAction(name, action, argumentTypes, description)
- #define emberCommandEntrySubMenu(name, subMenu, description)
- #define emberCommandEntryTerminator()
- #define EMBER_COMMAND_INTERPRETER_CONFIGURATION_ECHO
- #define emberProcessCommandInput(port)
- #define emberCommandInterpreterEchoOn()
- #define emberCommandInterpreterEchoOff()
- #define emberCommandInterpreterIsEchoOn()

Typedefs

- typedef void(∗ CommandAction)(void)

Enumerations

- enum EmberCommandStatus {
  EMBER_CMD_SUCCESS, EMBER_CMD_ERR_PORT_PROBLEM, EMBER_CMD_ERR_NO-
  _SUCH_COMMAND, EMBER_CMD_ERR_WRONG_NUMBER_OF_ARGUMENTS,
  EMBER_CMD_ERR_ARGUMENT_OUT_OF_RANGE, EMBER_CMD_ERR_ARGUMENT_S-
  YNTAX_ERROR, EMBER_CMD_ERR_STRING_TOO_LONG, EMBER_CMD_ERR_INVALI-
  D_ARGUMENT_TYPE }

Functions

- void emberCommandActionHandler (const CommandAction action)
- void emberCommandErrorHandler (EmberCommandStatus status)
- void emberPrintCommandUsage (EmberCommandEntry ∗entry)
- void emberPrintCommandUsageNotes (void)
- void emberPrintCommandTable (void)
- void emberCommandReaderInit (void)
- boolean emberProcessCommandString (int8u ∗input, int8u sizeOrPort)

Variables

- EmberCommandEntry ∗ emberCurrentCommand
- EmberCommandEntry emberCommandTable [ ]
- int8u emberCommandInterpreter2Configuration

354

Command Table Settings

- #define EMBER_MAX_COMMAND_ARGUMENTS
- #define EMBER_COMMAND_BUFFER_LENGTH
- #define EMBER_COMMAND_INTEPRETER_HAS_DESCRIPTION_FIELD

Functions to Retrieve Arguments

Use the following functions in your functions that process commands to retrieve arguments from the command interpreter. These functions pull out unsigned integers, signed integers, and strings, and hex strings. Index 0 is the first command argument.

- int8u emberCommandArgumentCount (void)
- int32u emberUnsignedCommandArgument (int8u argNum)
- int16s emberSignedCommandArgument (int8u argNum)
- int8u ∗ emberStringCommandArgument (int8s argNum, int8u ∗length)
- int8u emberCopyStringArgument (int8s argNum, int8u ∗destination, int8u maxLength, boolean left-Pad)
- #define emberCopyKeyArgument(index, keyDataPointer)
- #define emberCopyEui64Argument(index, eui64)

6.61.1   Detailed Description

Interpret serial port commands. See command-interpreter2.c for source code.

See the following application usage example followed by a brief explanation.

```
// Usage: network form 22 0xAB12 -3 { 00 01 02 A3 A4 A5 A6 A7 }
void formCommand(void)
{
  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[] = {
  emberCommandEntrySubMenu("network",  networkCommands,
      "Network form/join commands"),
  emberCommandEntryAction("status",    statusCommand,
      "Prints application status),
  ...
  emberCommandEntryTerminator()
};

// The table of network commands.
EmberCommandEntry networkCommands[] = {
  emberCommandEntryAction("form", formCommand, "uvsh", "Form a network"),
  emberCommandEntryAction("join", joinCommand, "uvsh", "Join a network"),
  ...
  emberCommandEntryTerminator()
};

void main(void)
{
   emberCommandReaderInit();
```

```
    while(0) {
      ...
      // Process input and print prompt if it returns TRUE.
      if (emberProcessCommandInput(serialPort)) {
        emberSerialPrintf(1, "%p>", PROMPT);
      }
      ...
    }
}
```

1. Applications specify the commands that can be interpreted by defining the emberCommandTable array of type EmberCommandEntry. The table includes the following information for each command:

   (a) The full command name.

   (b) Your application's function name that implements the command.

   (c) An EmberCommandEntry::argumentTypes string specifies the number and types of arguments the command accepts. See ::argumentTypes for details.

   (d) A description string explains the command.

2. A default error handler emberCommandErrorHandler() is provided to deal with incorrect command input. Applications may override it.

3. The application calls emberCommandReaderInit() to initalize, and emberProcessCommandInput() in its main loop.

4. Within the application's command functions, use emberXXXCommandArgument() functions to retrieve command arguments.

The command interpreter does extensive processing and validation of the command input before calling the function that implements the command. It checks that the number, type, syntax, and range of all arguments are correct. It performs any conversions necessary (for example, converting integers and strings input in hexadecimal notation into the corresponding bytes), so that no additional parsing is necessary within command functions. If there is an error in the command input, emberCommandErrorHandler() is called rather than a command function.

The command interpreter allows inexact matches of command names. The input command may be either shorter or longer than the actual command. However, if more than one inexact match is found and there is no exact match, an error of type EMBER_CMD_ERR_NO_SUCH_COMMAND will be generated. To disable this feature, define EMBER_REQUIRE_EXACT_COMMAND_NAME in the application configuration header.

6.61.2 Macro Definition Documentation

6.61.2.1 #define EMBER_MAX_COMMAND_ARGUMENTS

The maximum number of arguments a command can have. A nested command counts as an argument.

Definition at line 104 of file command-interpreter2.h.

6.61.2.2 #define EMBER_COMMAND_BUFFER_LENGTH

The maximum number of arguments a command can have. A nested command counts as an argument.

Definition at line 108 of file command-interpreter2.h.

6.61.2.3   #define EMBER_COMMAND_INTEPRETER_HAS_DESCRIPTION_FIELD

Whether or not the command entry structure will include descriptions for the commands. This consumes additional CONST space, which is expensive on the XAP. By default descriptions are not included.

Definition at line 116 of file command-interpreter2.h.

6.61.2.4   #define MAX_TOKEN_COUNT

Definition at line 123 of file command-interpreter2.h.

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

Definition at line 180 of file command-interpreter2.h.

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

Definition at line 184 of file command-interpreter2.h.

6.61.2.7   #define emberCommandEntryTerminator(   )

Definition at line 188 of file command-interpreter2.h.

6.61.2.8   #define EMBER_COMMAND_INTERPRETER_CONFIGURATION_ECHO

Definition at line 221 of file command-interpreter2.h.

6.61.2.9   #define emberCopyKeyArgument(*index, keyDataPointer*)

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

Definition at line 292 of file command-interpreter2.h.

6.61.2.10   #define emberCopyEui64Argument(*index, eui64*)

A convenience macro for copying eui64 arguments to an EmberEUI64.

Definition at line 299 of file command-interpreter2.h.

6.61.2.11   #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 336 of file command-interpreter2.h.

6.61.2.12   #define emberCommandInterpreterEchoOn(   )

Turn echo of command line on.

Definition at line 341 of file command-interpreter2.h.

6.61.2.13   #define emberCommandInterpreterEchoOff(   )

Turn echo of command line off.

Definition at line 347 of file command-interpreter2.h.

6.61.2.14   #define emberCommandInterpreterIsEchoOn(   )

Returns true if echo is on, false otherwise.

Definition at line 353 of file command-interpreter2.h.

6.61.3   Typedef Documentation

6.61.3.1   typedef void(∗ CommandAction)(void)

Definition at line 125 of file command-interpreter2.h.

6.61.4   Enumeration Type Documentation

6.61.4.1   enum EmberCommandStatus

Command error states.

If you change this list, ensure you also change the strings that describe these errors in the array ember-CommandErrorNames[] 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 229 of file command-interpreter2.h.

6.61.5   Function Documentation

6.61.5.1   int8u emberCommandArgumentCount (void)

Returns the number of arguments for the current command.

6.61.5.2   int32u emberUnsignedCommandArgument (int8u *argNum*)

Retrieves unsigned integer arguments.

6.61.5.3   int16s emberSignedCommandArgument (int8u *argNum*)

Retrieves signed integer arguments.

6.61.5.4   int8u∗ emberStringCommandArgument (int8s *argNum,* 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 argNum 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.

6.61.5.5   int8u emberCopyStringArgument (int8s *argNum,* 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.

6.61.5.6   void emberCommandActionHandler (const CommandAction *action*)

The application may implement this handler. To override the default handler, define EMBER_APPLICA-TION_HAS_COMMAND_ACTION_HANDLER in the CONFIGURATION_HEADER.

6.61.5.7   void emberCommandErrorHandler (EmberCommandStatus *status*)

The application may implement this handler. To override the default handler, define EMBER_APPLI-CATION_HAS_COMMAND_ERROR_HANDLER in the CONFIGURATION_HEADER. Defining this will also remove the help functions emberPrintCommandUsage(), emberPrintCommandUsageNotes(), and emberPrintCommandTable().

6.61.5.8   void emberPrintCommandUsage (EmberCommandEntry ∗ *entry*)

6.61.5.9   void emberPrintCommandUsageNotes (void)

6.61.5.10   void emberPrintCommandTable (void)

6.61.5.11   void emberCommandReaderInit (void)

Initialize the command interpreter.

6.61.5.12 boolean emberProcessCommandString (int8u ∗ *input,* int8u *sizeOrPort*)

Process the given string as a command.

6.61.6 Variable Documentation

6.61.6.1 EmberCommandEntry∗ emberCurrentCommand

A pointer to the currently matching command entry. Only valid from within a command function. If the original command was nested, points to the final (non-nested) command entry.

6.61.6.2 EmberCommandEntry emberCommandTable[ ]

6.61.6.3 int8u emberCommandInterpreter2Configuration

Configuration byte.

Chapter 7

Data Structure Documentation

7.1 EmberAesMmoHashContext Struct Reference

```
#include <ember-types.h>
```

Data Fields

- int8u result [EMBER_AES_HASH_BLOCK_SIZE]
- int32u length

7.1.1 Detailed Description

This data structure contains the context data when calculating an AES MMO hash (message digest).

Definition at line 1403 of file ember-types.h.

7.1.2 Field Documentation

7.1.2.1 int8u EmberAesMmoHashContext::result[EMBER_AES_HASH_BLOCK_SIZE]

Definition at line 1404 of file ember-types.h.

7.1.2.2 int32u EmberAesMmoHashContext::length

Definition at line 1405 of file ember-types.h.

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

- ember-types.h

7.2 EmberApsFrame Struct Reference

```
#include <ember-types.h>
```

Data Fields

- int16u profileId
- int16u clusterId
- int8u sourceEndpoint
- int8u destinationEndpoint
- EmberApsOption options
- int16u groupId
- int8u sequence

7.2.1 Detailed Description

An in-memory representation of a ZigBee APS frame of an incoming or outgoing message.

Definition at line 835 of file ember-types.h.

7.2.2 Field Documentation

7.2.2.1 int16u EmberApsFrame::profileId

The application profile ID that describes the format of the message.

Definition at line 837 of file ember-types.h.

7.2.2.2 int16u EmberApsFrame::clusterId

The cluster ID for this message.

Definition at line 839 of file ember-types.h.

7.2.2.3 int8u EmberApsFrame::sourceEndpoint

The source endpoint.

Definition at line 841 of file ember-types.h.

7.2.2.4 int8u EmberApsFrame::destinationEndpoint

The destination endpoint.

Definition at line 843 of file ember-types.h.

7.2.2.5 EmberApsOption EmberApsFrame::options

A bitmask of options from the enumeration above.

Definition at line 845 of file ember-types.h.

7.2.2.6 int16u EmberApsFrame::groupId

The group ID for this message, if it is multicast mode.

Definition at line 847 of file ember-types.h.

7.2.2.7 int8u EmberApsFrame::sequence

The sequence number.

Definition at line 849 of file ember-types.h.

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

- ember-types.h

7.3 EmberBindingTableEntry Struct Reference

```
#include <ember-types.h>
```

Data Fields

- EmberBindingType type
- int8u local
- int16u clusterId
- int8u remote
- EmberEUI64 identifier
- int8u networkIndex

7.3.1 Detailed Description

Defines an entry in the binding table.

A binding entry specifies a local endpoint, a remote endpoint, a cluster ID and either the destination EUI64 (for unicast bindings) or the 64-bit group address (for multicast bindings).

Definition at line 859 of file ember-types.h.

7.3.2 Field Documentation

7.3.2.1 EmberBindingType EmberBindingTableEntry::type

The type of binding.

Definition at line 861 of file ember-types.h.

7.3.2.2 int8u EmberBindingTableEntry::local

The endpoint on the local node.

Definition at line 863 of file ember-types.h.

7.3.2.3 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 871 of file ember-types.h.

7.3.2.4 int8u EmberBindingTableEntry::remote

The endpoint on the remote node (specified by identifier).

Definition at line 873 of file ember-types.h.

7.3.2.5 EmberEUI64 EmberBindingTableEntry::identifier

A 64-bit identifier. This is either:

- The destination EUI64, for unicasts
- A 16-bit multicast group address, for multicasts

Definition at line 878 of file ember-types.h.

7.3.2.6 int8u EmberBindingTableEntry::networkIndex

The index of the network the binding belongs to.

Definition at line 880 of file ember-types.h.

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

- ember-types.h

7.4 EmberCertificateData Struct Reference

```
#include <ember-types.h>
```

Data Fields

- int8u contents [EMBER_CERTIFICATE_SIZE]

7.4.1 Detailed Description

This data structure contains the certificate data that is used for Certificate Based Key Exchange (CBKE).

Definition at line 1364 of file ember-types.h.

7.4.2 Field Documentation

7.4.2.1 int8u EmberCertificateData::contents[EMBER_CERTIFICATE_SIZE]

Definition at line 1366 of file ember-types.h.

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

- ember-types.h

7.5 EmberCommandEntry Struct Reference

```
#include <command-interpreter2.h>
```

Data Fields

- PGM_P name
- CommandAction action
- PGM_P argumentTypes
- PGM_P description

7.5.1 Detailed Description

Command entry for a command table.

Definition at line 130 of file command-interpreter2.h.

7.5.2 Field Documentation

7.5.2.1 PGM_P EmberCommandEntry::name

Use letters, digits, and underscores, '_', for the command name. Command names are case-sensitive.

Definition at line 137 of file command-interpreter2.h.

7.5.2.2 CommandAction EmberCommandEntry::action

A reference to a function in the application that implements the command. If this entry refers to a nested command, then action field has to be set to NULL.

Definition at line 143 of file command-interpreter2.h.

7.5.2.3 PGM_P EmberCommandEntry::argumentTypes

In case of normal (non-nested) commands, argumentTypes is a string that specifies the number and types of arguments the command accepts. The argument specifiers are:

- u: one-byte unsigned integer.

- v: two-byte unsigned integer

- w: four-byte unsigned integer

- s: one-byte signed integer

- 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 Ember-CommandEntry array.

Definition at line 170 of file command-interpreter2.h.

7.5.2.4 PGM_P EmberCommandEntry::description

A description of the command.

Definition at line 174 of file command-interpreter2.h.

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

- command-interpreter2.h

7.6 EmberCurrentSecurityState Struct Reference

```
#include <ember-types.h>
```

Data Fields

- EmberCurrentSecurityBitmask bitmask
- EmberEUI64 trustCenterLongAddress

7.6.1 Detailed Description

This describes the security features used by the stack for a joined device.

Definition at line 1657 of file ember-types.h.

7.6.2 Field Documentation

7.6.2.1 EmberCurrentSecurityBitmask EmberCurrentSecurityState::bitmask

This bitmask indicates the security features currently in use on this node.

Definition at line 1660 of file ember-types.h.

7.6.2.2 EmberEUI64 EmberCurrentSecurityState::trustCenterLongAddress

This indicates the EUI64 of the Trust Center. It will be all zeroes if the Trust Center Address is not known (i.e. the device is in a Distributed Trust Center network).

Definition at line 1664 of file ember-types.h.

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

- ember-types.h

7.7 EmberEndpoint Struct Reference

```
#include <stack-info.h>
```

Data Fields

- int8u endpoint
- EmberEndpointDescription PGM ∗ description
- int16u PGM ∗ inputClusterList
- int16u PGM ∗ outputClusterList

7.7.1 Detailed Description

Gives the endpoint information for a particular endpoint.

Definition at line 283 of file stack-info.h.

7.7.2 Field Documentation

7.7.2.1 int8u EmberEndpoint::endpoint

An endpoint of the application on this node.

Definition at line 285 of file stack-info.h.

7.7.2.2 EmberEndpointDescription PGM∗ EmberEndpoint::description

The endpoint's description.

Definition at line 287 of file stack-info.h.

7.7.2.3 int16u PGM∗ EmberEndpoint::inputClusterList

Input clusters the endpoint will accept.

Definition at line 289 of file stack-info.h.

7.7.2.4 int16u PGM∗ EmberEndpoint::outputClusterList

Output clusters the endpoint may send.

Definition at line 291 of file stack-info.h.

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

- stack-info.h

7.8 EmberEndpointDescription Struct Reference

```
#include <stack-info.h>
```

Data Fields

- int16u profileId
- int16u deviceId
- int8u deviceVersion
- int8u inputClusterCount
- int8u outputClusterCount

7.8.1 Detailed Description

Endpoint information (a ZigBee Simple Descriptor).

This is a ZigBee Simple Descriptor and contains information about an endpoint. This information is shared with other nodes in the network by the ZDO.

Definition at line 267 of file stack-info.h.

7.8.2 Field Documentation

7.8.2.1 int16u EmberEndpointDescription::profileId

Identifies the endpoint's application profile.

Definition at line 269 of file stack-info.h.

7.8.2.2 int16u EmberEndpointDescription::deviceId

The endpoint's device ID within the application profile.

Definition at line 271 of file stack-info.h.

7.8.2.3 int8u EmberEndpointDescription::deviceVersion

The endpoint's device version.

Definition at line 273 of file stack-info.h.

7.8.2.4 int8u EmberEndpointDescription::inputClusterCount

The number of input clusters.

Definition at line 275 of file stack-info.h.

7.8.2.5 int8u EmberEndpointDescription::outputClusterCount

The number of output clusters.

Definition at line 277 of file stack-info.h.

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

- stack-info.h

7.9 EmberEventControl Struct Reference

```
#include <ember-types.h>
```

Data Fields

- EmberEventUnits status
- EmberTaskId taskid
- int32u timeToExecute

7.9.1 Detailed Description

Control structure for events.

This structure should not be accessed directly. This holds the event status (one of the *EMBER_EVENT_* values) and the time left before the event fires.

Definition at line 1130 of file ember-types.h.

7.9.2 Field Documentation

7.9.2.1 EmberEventUnits EmberEventControl::status

The event's status, either inactive or the units for timeToExecute.

Definition at line 1132 of file ember-types.h.

7.9.2.2 EmberTaskId EmberEventControl::taskid

The id of the task this event belongs to.

Definition at line 1134 of file ember-types.h.

7.9.2.3   int32u EmberEventControl::timeToExecute

How long before the event fires. Units are always in milliseconds

Definition at line 1138 of file ember-types.h.

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

- ember-types.h

7.10   EmberInitialSecurityState Struct Reference

```
#include <ember-types.h>
```

Data Fields

- int16u bitmask
- EmberKeyData preconfiguredKey
- EmberKeyData networkKey
- int8u networkKeySequenceNumber
- EmberEUI64 preconfiguredTrustCenterEui64

7.10.1   Detailed Description

This describes the Initial Security features and requirements that will be used when forming or joining the network.

Definition at line 1577 of file ember-types.h.

7.10.2   Field Documentation

7.10.2.1   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 Ember-InitialSecurityBitmask.

Definition at line 1582 of file ember-types.h.

7.10.2.2   EmberKeyData EmberInitialSecurityState::preconfiguredKey

This is the pre-configured key that can used by devices when joining the network if the Trust Center does not send the initial security data in-the-clear. For the Trust Center, it will be the global link key and **must** be set regardless of whether joining devices are expected to have a pre-configured Link Key. This parameter will only be used if the EmberInitialSecurityState::bitmask sets the bit indicating EMBER_HAVE_PREC-ONFIGURED_KEY

Definition at line 1591 of file ember-types.h.

7.10.2.3   EmberKeyData EmberInitialSecurityState::networkKey

This is the Network Key used when initially forming the network. This must be set on the Trust Center. It is not needed for devices joining the network. This parameter will only be used if the EmberInitialSecurity-State::bitmask sets the bit indicating EMBER_HAVE_NETWORK_KEY.

Definition at line 1597 of file ember-types.h.

7.10.2.4   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 1604 of file ember-types.h.

7.10.2.5   EmberEUI64 EmberInitialSecurityState::preconfiguredTrustCenterEui64

This is the long address of the trust center on the network that will be joined. It is usually NOT set prior to joining the network and instead it is learned during the joining message exchange. This field is only examined if EMBER_HAVE_TRUST_CENTER_EUI64 is set in the EmberInitialSecurityState-::bitmask. Most devices should clear that bit and leave this field alone. This field must be set when using commissioning mode. It is required to be in little-endian format.

Definition at line 1612 of file ember-types.h.

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

- ember-types.h

7.11   EmberKeyData Struct Reference

```
#include <ember-types.h>
```

Data Fields

- int8u contents [EMBER_ENCRYPTION_KEY_SIZE]

7.11.1   Detailed Description

This data structure contains the key data that is passed into various other functions.

Definition at line 1357 of file ember-types.h.

7.11.2   Field Documentation

7.11.2.1   int8u EmberKeyData::contents[EMBER_ENCRYPTION_KEY_SIZE]

This is the key byte data.

Definition at line 1359 of file ember-types.h.

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

- ember-types.h

7.12 EmberKeyStruct Struct Reference

```
#include <ember-types.h>
```

Data Fields

- EmberKeyStructBitmask bitmask
- EmberKeyType type
- EmberKeyData key
- int32u outgoingFrameCounter
- int32u incomingFrameCounter
- int8u sequenceNumber
- EmberEUI64 partnerEUI64

7.12.1 Detailed Description

This describes a one of several different types of keys and its associated data.

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

7.12.2 Field Documentation

7.12.2.1 EmberKeyStructBitmask EmberKeyStruct::bitmask

This bitmask indicates whether various fields in the structure contain valid data. It also contains the index of the network the key belongs to.

Definition at line 1734 of file ember-types.h.

7.12.2.2 EmberKeyType EmberKeyStruct::type

This indicates the type of the security key.

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

7.12.2.3 EmberKeyData EmberKeyStruct::key

This is the actual key data.

Definition at line 1738 of file ember-types.h.

7.12.2.4   int32u EmberKeyStruct::outgoingFrameCounter

This is the outgoing frame counter associated with the key. It will contain valid data based on the Ember-KeyStructBitmask.

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

7.12.2.5   int32u EmberKeyStruct::incomingFrameCounter

This is the incoming frame counter associated with the key. It will contain valid data based on the Ember-KeyStructBitmask.

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

7.12.2.6   int8u EmberKeyStruct::sequenceNumber

This is the sequence number associated with the key. It will contain valid data based on the EmberKey-StructBitmask.

Definition at line 1747 of file ember-types.h.

7.12.2.7   EmberEUI64 EmberKeyStruct::partnerEUI64

This is the Partner EUI64 associated with the key. It will contain valid data based on the EmberKeyStruct-Bitmask.

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

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

- ember-types.h

7.13   EmberMacFilterMatchStruct Struct Reference

```
#include <ember-types.h>
```

Data Fields

- int8u filterIndexMatch
- EmberMacPassthroughType legacyPassthroughType
- EmberMessageBuffer message

7.13.1   Detailed Description

This structure indicates a matching raw MAC message has been received by the application configured MAC filters.

Definition at line 1975 of file ember-types.h.

7.13.2 Field Documentation

7.13.2.1 int8u EmberMacFilterMatchStruct::filterIndexMatch

Definition at line 1976 of file ember-types.h.

7.13.2.2 EmberMacPassthroughType EmberMacFilterMatchStruct::legacyPassthroughType

Definition at line 1977 of file ember-types.h.

7.13.2.3 EmberMessageBuffer EmberMacFilterMatchStruct::message

Definition at line 1978 of file ember-types.h.

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

- ember-types.h

7.14 EmberMessageDigest Struct Reference

```
#include <ember-types.h>
```

Data Fields

- int8u contents [EMBER_AES_HASH_BLOCK_SIZE]

7.14.1 Detailed Description

This data structure contains an AES-MMO Hash (the message digest).

Definition at line 1396 of file ember-types.h.

7.14.2 Field Documentation

7.14.2.1 int8u EmberMessageDigest::contents[EMBER_AES_HASH_BLOCK_SIZE]

Definition at line 1397 of file ember-types.h.

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

- ember-types.h

7.15 EmberMfgSecurityStruct Struct Reference

```
#include <ember-types.h>
```

Data Fields

- EmberKeySettings keySettings

7.15.1 Detailed Description

This structure is used to get/set the security config that is stored in manufacturing tokens.

Definition at line 1892 of file ember-types.h.

7.15.2 Field Documentation

7.15.2.1 EmberKeySettings EmberMfgSecurityStruct::keySettings

Definition at line 1893 of file ember-types.h.

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

- ember-types.h

7.16 EmberMulticastTableEntry Struct Reference

```
#include <ember-types.h>
```

Data Fields

- EmberMulticastId multicastId
- int8u endpoint
- int8u networkIndex

7.16.1 Detailed Description

Defines an entry in the multicast table.

A multicast table entry indicates that a particular endpoint is a member of a particular multicast group. Only devices with an endpoint in a multicast group will receive messages sent to that multicast group.

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

7.16.2 Field Documentation

7.16.2.1 EmberMulticastId EmberMulticastTableEntry::multicastId

The multicast group ID.

Definition at line 950 of file ember-types.h.

7.16.2.2 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 954 of file ember-types.h.

7.16.2.3 int8u EmberMulticastTableEntry::networkIndex

The network index of the network the entry is related to.

Definition at line 956 of file ember-types.h.

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

- ember-types.h

7.17 EmberNeighborTableEntry Struct Reference

```
#include <ember-types.h>
```

Data Fields

- int16u shortId
- int8u averageLqi
- int8u inCost
- int8u outCost
- int8u age
- EmberEUI64 longId

7.17.1 Detailed Description

Defines an entry in the neighbor table.

A neighbor table entry stores information about the reliability of RF links to and from neighboring nodes.

Definition at line 889 of file ember-types.h.

7.17.2 Field Documentation

7.17.2.1 int16u EmberNeighborTableEntry::shortId

The neighbor's two byte network id.

Definition at line 891 of file ember-types.h.

7.17.2.2 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 894 of file ember-types.h.

7.17.2.3 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 897 of file ember-types.h.

7.17.2.4 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 904 of file ember-types.h.

7.17.2.5 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 910 of file ember-types.h.

7.17.2.6 EmberEUI64 EmberNeighborTableEntry::longId

The 8 byte EUI64 of the neighbor.

Definition at line 912 of file ember-types.h.

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

- ember-types.h

7.18 EmberNetworkInitStruct Struct Reference

```
#include <ember-types.h>
```

Data Fields

- EmberNetworkInitBitmask bitmask

7.18.1 Detailed Description

Defines the network initialization configuration that should be used when emberNetworkInitExtended() is called by the application.

Definition at line 387 of file ember-types.h.

7.18.2 Field Documentation

7.18.2.1 EmberNetworkInitBitmask EmberNetworkInitStruct::bitmask

Definition at line 388 of file ember-types.h.

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

- ember-types.h

7.19 EmberNetworkParameters Struct Reference

```
#include <ember-types.h>
```

Data Fields

- int8u extendedPanId [8]
- int16u panId
- int8s radioTxPower
- int8u radioChannel
- EmberJoinMethod joinMethod
- EmberNodeId nwkManagerId
- int8u nwkUpdateId
- int32u channels

7.19.1 Detailed Description

Holds network parameters.

For information about power settings and radio channels, see the technical specification for the RF communication module in your Developer Kit.

Definition at line 790 of file ember-types.h.

7.19.2 Field Documentation

7.19.2.1 int8u EmberNetworkParameters::extendedPanId[8]

The network's extended PAN identifier.

Definition at line 792 of file ember-types.h.

7.19.2.2 int16u EmberNetworkParameters::panId

The network's PAN identifier.

Definition at line 794 of file ember-types.h.

7.19.2.3 int8s EmberNetworkParameters::radioTxPower

A power setting, in dBm.

Definition at line 796 of file ember-types.h.

7.19.2.4 int8u EmberNetworkParameters::radioChannel

A radio channel. Be sure to specify a channel supported by the radio.

Definition at line 798 of file ember-types.h.

7.19.2.5 EmberJoinMethod EmberNetworkParameters::joinMethod

Join method: The protocol messages used to establish an initial parent. It is ignored when forming a ZigBee network, or when querying the stack for its network parameters.

Definition at line 803 of file ember-types.h.

7.19.2.6 EmberNodeId EmberNetworkParameters::nwkManagerId

NWK Manager ID. The ID of the network manager in the current network. This may only be set at joining when using EMBER_USE_NWK_COMMISSIONING as the join method.

Definition at line 809 of file ember-types.h.

7.19.2.7 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 815 of file ember-types.h.

7.19.2.8 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_COMMISS-IONING as the join method.

Definition at line 821 of file ember-types.h.

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

- ember-types.h

7.20 EmberPrivateKeyData Struct Reference

```
#include <ember-types.h>
```

Data Fields

- int8u contents [EMBER_PRIVATE_KEY_SIZE]

7.20.1 Detailed Description

This data structure contains the private key data that is used for Certificate Based Key Exchange (CBKE).

Definition at line 1377 of file ember-types.h.

7.20.2 Field Documentation

7.20.2.1 int8u EmberPrivateKeyData::contents[EMBER_PRIVATE_KEY_SIZE]

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

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

- ember-types.h

7.21 EmberPublicKeyData Struct Reference

```
#include <ember-types.h>
```

Data Fields

- int8u contents [EMBER_PUBLIC_KEY_SIZE]

7.21.1 Detailed Description

This data structure contains the public key data that is used for Certificate Based Key Exchange (CBKE).

Definition at line 1371 of file ember-types.h.

7.21.2 Field Documentation

7.21.2.1 int8u EmberPublicKeyData::contents[EMBER_PUBLIC_KEY_SIZE]

Definition at line 1372 of file ember-types.h.

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

- ember-types.h

7.22 EmberRouteTableEntry Struct Reference

```
#include <ember-types.h>
```

Data Fields

- int16u destination
- int16u nextHop
- int8u status
- int8u age
- int8u concentratorType
- int8u routeRecordState

7.22.1 Detailed Description

Defines an entry in the route table.

A route table entry stores information about the next hop along the route to the destination.

Definition at line 920 of file ember-types.h.

7.22.2 Field Documentation

7.22.2.1 int16u EmberRouteTableEntry::destination

The short id of the destination.

Definition at line 922 of file ember-types.h.

7.22.2.2 int16u EmberRouteTableEntry::nextHop

The short id of the next hop to this destination.

Definition at line 924 of file ember-types.h.

7.22.2.3 int8u EmberRouteTableEntry::status

Indicates whether this entry is active (0), being discovered (1), or unused (3).

Definition at line 927 of file ember-types.h.

7.22.2.4 int8u EmberRouteTableEntry::age

The number of seconds since this route entry was last used to send a packet.

Definition at line 930 of file ember-types.h.

7.22.2.5 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 933 of file ember-types.h.

7.22.2.6 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 938 of file ember-types.h.

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

- ember-types.h

7.23 EmberSignatureData Struct Reference

```
#include <ember-types.h>
```

Data Fields

- int8u contents [EMBER_SIGNATURE_SIZE]

7.23.1 Detailed Description

This data structure contains a DSA signature. It is the bit concatenation of the 'r' and 's' components of the signature.

Definition at line 1390 of file ember-types.h.

7.23.2 Field Documentation

7.23.2.1 int8u EmberSignatureData::contents[EMBER_SIGNATURE_SIZE]

Definition at line 1391 of file ember-types.h.

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

- ember-types.h

7.24 EmberSmacData Struct Reference

```
#include <ember-types.h>
```

Data Fields

- int8u contents [EMBER_SMAC_SIZE]

7.24.1 Detailed Description

This data structure contains the Shared Message Authentication Code (SMAC) data that is used for Certificate Based Key Exchange (CBKE).

Definition at line 1383 of file ember-types.h.

7.24.2 Field Documentation

7.24.2.1 int8u EmberSmacData::contents[EMBER_SMAC_SIZE]

Definition at line 1384 of file ember-types.h.

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

- ember-types.h

7.25 EmberTaskControl Struct Reference

```
#include <ember-types.h>
```

Data Fields

- int32u nextEventTime
- EmberEventData * events
- boolean busy

7.25.1 Detailed Description

Control structure for tasks.

This structure should not be accessed directly.

Definition at line 1176 of file ember-types.h.

7.25.2 Field Documentation

7.25.2.1 int32u EmberTaskControl::nextEventTime

Definition at line 1178 of file ember-types.h.

7.25.2.2 EmberEventData∗ EmberTaskControl::events

Definition at line 1180 of file ember-types.h.

7.25.2.3 boolean EmberTaskControl::busy

Definition at line 1182 of file ember-types.h.

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

- ember-types.h

7.26 EmberVersion Struct Reference

```
#include <ember-types.h>
```

Data Fields

- int16u build
- int8u major
- int8u minor
- int8u patch
- int8u special
- EmberVersionType type

7.26.1 Detailed Description

Version struct containing all version information.

Definition at line 55 of file ember-types.h.

7.26.2 Field Documentation

7.26.2.1 int16u EmberVersion::build

Definition at line 56 of file ember-types.h.

7.26.2.2 int8u EmberVersion::major

Definition at line 57 of file ember-types.h.

7.26.2.3 int8u EmberVersion::minor

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

7.26.2.4 int8u EmberVersion::patch

Definition at line 59 of file ember-types.h.

7.26.2.5 int8u EmberVersion::special

Definition at line 60 of file ember-types.h.

7.26.2.6 EmberVersionType EmberVersion::type

Definition at line 61 of file ember-types.h.

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

- ember-types.h

7.27 EmberZigbeeNetwork Struct Reference

```
#include <ember-types.h>
```

Data Fields

- int16u panId
- int8u channel
- boolean allowingJoin
- int8u extendedPanId [8]
- int8u stackProfile
- int8u nwkUpdateId

7.27.1 Detailed Description

Defines a ZigBee network and the associated parameters.

Definition at line 354 of file ember-types.h.

7.27.2 Field Documentation

7.27.2.1 int16u EmberZigbeeNetwork::panId

Definition at line 355 of file ember-types.h.

7.27.2.2 int8u EmberZigbeeNetwork::channel

Definition at line 356 of file ember-types.h.

7.27.2.3 boolean EmberZigbeeNetwork::allowingJoin

Definition at line 357 of file ember-types.h.

7.27.2.4 int8u EmberZigbeeNetwork::extendedPanId[8]

Definition at line 358 of file ember-types.h.

7.27.2.5 int8u EmberZigbeeNetwork::stackProfile

Definition at line 359 of file ember-types.h.

7.27.2.6 int8u EmberZigbeeNetwork::nwkUpdateId

Definition at line 360 of file ember-types.h.

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

- ember-types.h

7.28 InterPanHeader Struct Reference

```
#include <ami-inter-pan.h>
```

Data Fields

- int8u messageType
- int16u panId
- boolean hasLongAddress
- EmberNodeId shortAddress
- EmberEUI64 longAddress
- int16u profileId
- int16u clusterId
- int16u groupId

7.28.1 Detailed Description

A struct for keeping track of all of the header info.

A struct for keeping track of all of the interpan header info.

Definition at line 51 of file ami-inter-pan.h.

7.28.2 Field Documentation

7.28.2.1 int8u InterPanHeader::messageType

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

7.28.2.2 int16u InterPanHeader::panId

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

7.28.2.3 boolean InterPanHeader::hasLongAddress

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

7.28.2.4 EmberNodeId InterPanHeader::shortAddress

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

7.28.2.5 EmberEUI64 InterPanHeader::longAddress

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

7.28.2.6 int16u InterPanHeader::profileId

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

7.28.2.7 int16u InterPanHeader::clusterId

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

7.28.2.8 int16u InterPanHeader::groupId

Definition at line 65 of file ami-inter-pan.h.

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

- ami-inter-pan.h
- ami-inter-pan-host.h

Chapter 8

File Documentation

8.1 ⎽EM35x⎽API.top File Reference

8.1.1 Detailed Description

Starting page for the Ember API documentation for the EM35x exclusively for building documentation. This file is used by Doxygen to generate the main page for the Ember API documentation, EM250.

Definition in file ⎽EM35x_API.top.

8.2 ⎽EM35x⎽API.top

00001

8.3 ami-inter-pan-host.h File Reference

Data Structures

- struct InterPanHeader

  A struct for keeping track of all of the header info.

Macros

- #define INTER_PAN_UNICAST
- #define INTER_PAN_BROADCAST
- #define INTER_PAN_MULTICAST
- #define MAX_INTER_PAN_MAC_SIZE
- #define STUB_NWK_SIZE
- #define STUB_NWK_FRAME_CONTROL
- #define MAX_STUB_APS_SIZE
- #define MAX_INTER_PAN_HEADER_SIZE

Functions

- int8u makeInterPanMessage (InterPanHeader ∗headerData, int8u ∗message, int8u maxLength, int8u ∗payload, int8u payloadLength)
- int8u parseInterPanMessage (int8u ∗message, int8u messageLength, InterPanHeader ∗headerData)

8.3.1 Detailed Description

Utilities for sending and receiving ZigBee AMI InterPAN messages. See Sending and Receiving Messages for documentation.

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

Definition in file ami-inter-pan-host.h.

8.4 ami-inter-pan-host.h

```
00001
00019 #ifndef AMI_INTER_PAN_HOST_H
00020 #define AMI_INTER_PAN_HOST_H
00021
00028 #define INTER_PAN_UNICAST    0x03
00029 #define INTER_PAN_BROADCAST 0x0B
00030 #define INTER_PAN_MULTICAST 0x0F
00031
00032
00033 // Frame control, sequence, dest PAN ID, dest, source PAN ID, source.
00034 #define MAX_INTER_PAN_MAC_SIZE (2 + 1 + 2 + 8 + 2 + 8)
00035 //Short form has a short destination.
00036
00037 // NWK stub frame has two control bytes.
00038 #define STUB_NWK_SIZE 2
00039 #define STUB_NWK_FRAME_CONTROL 0x000B
00040
00041 // APS frame control, group ID, cluster ID, profile ID
00042 #define MAX_STUB_APS_SIZE (1 + 2 + 2 + 2)
00043
00044 // Short form has no group ID.
00045 #define MAX_INTER_PAN_HEADER_SIZE \
00046  (MAX_INTER_PAN_MAC_SIZE + STUB_NWK_SIZE + MAX_STUB_APS_SIZE)
00047
00052 typedef struct {
00053   int8u messageType;              // one of the INTER_PAN_...CAST values
00054
00055   // MAC addressing
00056   // For outgoing messages this is the destination.  For incoming messages
00057   // it is the source, which always has a long address.
00058   int16u panId;
00059   boolean hasLongAddress;        // always TRUE for incoming messages
00060   EmberNodeId shortAddress;
00061   EmberEUI64 longAddress;
00062
00063   // APS data
00064   int16u profileId;
00065   int16u clusterId;
00066   int16u groupId;                // only used for INTER_PAN_MULTICAST
00067 } InterPanHeader;
00068
00075 int8u makeInterPanMessage(InterPanHeader
       *headerData,
00076                             int8u *message,
00077                             int8u maxLength,
00078                             int8u *payload,
00079                             int8u payloadLength);
00080
00088 int8u parseInterPanMessage(int8u *message,
00089                              int8u messageLength,
```

```
00090                                    InterPanHeader *headerData);
00091
00092 #endif // AMI_INTER_PAN_HOST_H
00093
```

8.5   ami-inter-pan.h File Reference

Data Structures

- struct InterPanHeader

    A struct for keeping track of all of the header info.

Macros

- #define INTER_PAN_UNICAST
- #define INTER_PAN_BROADCAST
- #define INTER_PAN_MULTICAST
- #define MAX_INTER_PAN_MAC_SIZE
- #define STUB_NWK_SIZE
- #define STUB_NWK_FRAME_CONTROL
- #define MAX_STUB_APS_SIZE
- #define MAX_INTER_PAN_HEADER_SIZE

Functions

- EmberMessageBuffer makeInterPanMessage (InterPanHeader *headerData, EmberMessageBuffer payload)
- int8u parseInterPanMessage (EmberMessageBuffer message, int8u startOffset, InterPanHeader *header-Data)

8.5.1   Detailed Description

Utilities for sending and receiving ZigBee AMI InterPAN messages. See Sending and Receiving Messages for documentation.

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

Definition in file ami-inter-pan.h.

8.6   ami-inter-pan.h

```
00001
00019 #ifndef AMI_INTER_PAN_H
00020 #define AMI_INTER_PAN_H
00021
00022 // The three types of inter-PAN messages.  The values are actually the
00023 // corresponding APS frame controls.
00024 //
00025 // 0x03 is the special interPAN message type.  Unicast mode is 0x00,
00026 // broadcast mode is 0x08, and multicast mode is 0x0C.
```

```
00027 //
00028
00029 #define INTER_PAN_UNICAST   0x03
00030 #define INTER_PAN_BROADCAST 0x0B
00031 #define INTER_PAN_MULTICAST 0x0F
00032
00033 // Frame control, sequence, dest PAN ID, dest, source PAN ID, source.
00034 #define MAX_INTER_PAN_MAC_SIZE (2 + 1 + 2 + 8 + 2 + 8)
00035 // Short form has a short destination.
00036
00037 // NWK stub frame has two control bytes.
00038 #define STUB_NWK_SIZE 2
00039 #define STUB_NWK_FRAME_CONTROL 0x000B
00040
00041 // APS frame control, group ID, cluster ID, profile ID
00042 #define MAX_STUB_APS_SIZE (1 + 2 + 2 + 2)
00043 // Short form has no group ID.
00044
00045 #define MAX_INTER_PAN_HEADER_SIZE \
00046  (MAX_INTER_PAN_MAC_SIZE + STUB_NWK_SIZE + MAX_STUB_APS_SIZE)
00047
00051 typedef struct {
00052   int8u messageType;            // one of the INTER_PAN_...CAST
       values
00053
00054   // MAC addressing
00055   // For outgoing messages this is the destination.  For incoming messages
00056   // it is the source, which always has a long address.
00057   int16u panId;
00058   boolean hasLongAddress;       // always TRUE for incoming
       messages
00059   EmberNodeId shortAddress;
00060   EmberEUI64 longAddress;
00061
00062   // APS data
00063   int16u profileId;
00064   int16u clusterId;
00065   int16u groupId;               // only used for
       INTER_PAN_MULTICAST
00066 } InterPanHeader;
00067
00068
00072 EmberMessageBuffer makeInterPanMessage(
      InterPanHeader *headerData,
00073                                        EmberMessageBuffer
      payload);
00074
00082 int8u parseInterPanMessage(EmberMessageBuffer
       message,
00083                             int8u startOffset,
00084                             InterPanHeader *headerData);
00085
00086 #endif // AMI_INTER_PAN_H
00087
```

8.7   app-bootloader.h File Reference

Required Custom Functions

- void bootloaderAction (boolean runRecovery)

Available Bootloader Library Functions

Functions implemented by the bootloader library that may be used by custom functions.

- BL_Status recoveryMode (void)
- BL_Status processImage (boolean install)

8.7.1 Detailed Description

See Application for detailed documentation.

Definition in file app-bootloader.h.

8.8 app-bootloader.h

```
00001
00014 #ifndef __EM350_APP_BOOTLOADER_H__
00015 #define __EM350_APP_BOOTLOADER_H__
00016
00018
00029 void bootloaderAction(boolean runRecovery);
00032
00033
00042 BL_Status recoveryMode(void);
00043
00049 BL_Status processImage(boolean install);
00053 #endif //__EM350_APP_BOOTLOADER_H__
00054
```

8.9 binding-table.h File Reference

Functions

- EmberStatus emberSetBinding (int8u index, EmberBindingTableEntry ∗value)
- EmberStatus emberGetBinding (int8u index, EmberBindingTableEntry ∗result)
- EmberStatus emberDeleteBinding (int8u index)
- boolean emberBindingIsActive (int8u index)
- EmberNodeId emberGetBindingRemoteNodeId (int8u index)
- void emberSetBindingRemoteNodeId (int8u index, EmberNodeId id)
- EmberStatus emberClearBindingTable (void)
- EmberStatus emberRemoteSetBindingHandler (EmberBindingTableEntry ∗entry)
- EmberStatus emberRemoteDeleteBindingHandler (int8u index)
- int8u emberGetBindingIndex (void)
- EmberStatus emberSetReplyBinding (int8u index, EmberBindingTableEntry ∗entry)
- EmberStatus emberNoteSendersBinding (int8u index)

8.9.1 Detailed Description

See Binding Table for documentation.

Definition in file binding-table.h.

8.10 binding-table.h

```
00001
00027 EmberStatus emberSetBinding(int8u index,
     EmberBindingTableEntry *value);
00028
00040 EmberStatus emberGetBinding(int8u index,
     EmberBindingTableEntry *result);
00041
00049 EmberStatus emberDeleteBinding(int8u index);
```

```
00050
00063 boolean emberBindingIsActive(int8u index);
00064
00086 EmberNodeId emberGetBindingRemoteNodeId(
      int8u index);
00087
00095 void emberSetBindingRemoteNodeId(int8u index,
      EmberNodeId id);
00096
00102 EmberStatus emberClearBindingTable(void);
00103
00126 EmberStatus emberRemoteSetBindingHandler
      (EmberBindingTableEntry *entry);
00127
00145 EmberStatus emberRemoteDeleteBindingHandler
      (int8u index);
00146
00167 int8u emberGetBindingIndex(void);
00168
00186 EmberStatus emberSetReplyBinding(int8u
      index, EmberBindingTableEntry *entry);
00187
00199 EmberStatus emberNoteSendersBinding(int8u
       index);
00200
00201 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00202 // This is defined in hal/ember-configuration.c.
00203 extern int8u emberBindingTableSize;
00204 #endif
00205
```

8.11   bootload.h File Reference

Functions

- EmberStatus emberSendBootloadMessage (boolean broadcast, EmberEUI64 destEui64, EmberMessage-Buffer message)
- void emberIncomingBootloadMessageHandler (EmberEUI64 longId, EmberMessageBuffer message)
- void emberBootloadTransmitCompleteHandler (EmberMessageBuffer message, EmberStatus status)

8.11.1   Detailed Description

See Bootloader for documentation.

Definition in file bootload.h.

8.12   bootload.h

```
00001
00030 EmberStatus emberSendBootloadMessage(boolean
      broadcast,
00031                                       EmberEUI64 destEui64,
00032                                       EmberMessageBuffer
      message);
00033
00043 void emberIncomingBootloadMessageHandler(
      EmberEUI64 longId,
00044                                       EmberMessageBuffer
      message);
00045
00055 void emberBootloadTransmitCompleteHandler(
      EmberMessageBuffer message,
00056                                       EmberStatus status);
00057
```

8.13 bootloader-common.h File Reference

Typedefs

- typedef int8u BL_Status

Enumerations

- enum { COMM_SERIAL, COMM_RADIO }

Bootloader Status Definitions

These are numerical definitions for the possible bootloader status codes.

- #define BL_SUCCESS
- #define BL_CRC_MATCH
- #define BL_IMG_FLASHED
- #define BL_ERR
- #define BL_ERR_MASK
- #define BL_ERR_HEADER_EXP
- #define BL_ERR_HEADER_WRITE_CRC
- #define BL_ERR_CRC
- #define BL_ERR_UNKNOWN_TAG
- #define BL_ERR_SIG
- #define BL_ERR_ODD_LEN
- #define BL_ERR_BLOCK_INDEX
- #define BL_ERR_OVWR_BL
- #define BL_ERR_OVWR_SIMEE
- #define BL_ERR_ERASE_FAIL
- #define BL_ERR_WRITE_FAIL
- #define BL_ERR_CRC_LEN
- #define BL_ERR_NO_QUERY
- #define BL_ERR_BAD_LEN
- #define BL_ERR_TAGBUF
- #define BL_EBL_CONTINUE

Bootloader State Flags

These are numerical flags for the possible bootloader states. These values are used in the bootloader code for making the current state more verbose.

Note

The flags do not start at 0 so that they can be output via the serial port during debug and easily screened out of normal xmodem traffic which depends only on ACK (0x06) and NAK (0x15).

- #define TIMEOUT
- #define FILEDONE
- #define FILEABORT

- #define BLOCKOK
- #define QUERYFOUND
- #define START_TIMEOUT
- #define BLOCK_TIMEOUT
- #define BLOCKERR_MASK
- #define BLOCKERR_SOH
- #define BLOCKERR_CHK
- #define BLOCKERR_CRCH
- #define BLOCKERR_CRCL
- #define BLOCKERR_SEQUENCE
- #define BLOCKERR_PARTIAL
- #define BLOCKERR_DUPLICATE

8.13.1 Detailed Description

See Common for detailed documentation.

Definition in file bootloader-common.h.

8.14 bootloader-common.h

```
00001
00007 //[[ Author(s): David Iacobone, diacobone@ember.com
00008 //             Lee Taylor, lee@ember.com
00009 //]]
00010
00018 #ifndef __BOOTLOADER_COMMON_H__
00019 #define __BOOTLOADER_COMMON_H__
00020
00021 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00022
00023 //#define BL_DEBUG
00024
00025 #ifdef BL_DEBUG
00026 #define BLDEBUG(x) (x)    // turn debug output on
00027 #define BLDEBUG_PRINT(str) serPutStr(str)
00028 #else
00029 #define BLDEBUG(x)        // turn debug output off
00030 #define BLDEBUG_PRINT(str)
00031 #endif
00032
00033 #endif //DOXYGEN_SHOULD_SKIP_THIS
00034
00037 typedef int8u BL_Status;
00038
00045 #define BL_SUCCESS 0
00046
00048 #define BL_CRC_MATCH 2
00049
00051 #define BL_IMG_FLASHED 3
00052
00054 #define BL_ERR 1
00055
00057 #define BL_ERR_MASK 0x40
00058
00061 #define BL_ERR_HEADER_EXP 0x41
00062
00065 #define BL_ERR_HEADER_WRITE_CRC 0x42
00066
00068 #define BL_ERR_CRC 0x43
00069
00071 #define BL_ERR_UNKNOWN_TAG 0x44
00072
00074 #define BL_ERR_SIG 0x45
00075
00078 #define BL_ERR_ODD_LEN 0x46
```

```
00079
00082 #define BL_ERR_BLOCK_INDEX 0x47
00083
00086 #define BL_ERR_OVWR_BL 0x48
00087
00090 #define BL_ERR_OVWR_SIMEE 0x49
00091
00094 #define BL_ERR_ERASE_FAIL 0x4A
00095
00098 #define BL_ERR_WRITE_FAIL 0x4B
00099
00102 #define BL_ERR_CRC_LEN 0x4C
00103
00106 #define BL_ERR_NO_QUERY 0X4D
00107
00110 #define BL_ERR_BAD_LEN 0x4E
00111
00114 #define BL_ERR_TAGBUF 0x4F
00115
00118 #define BL_EBL_CONTINUE 0x50
00119
00132 #define TIMEOUT           0x16
00133 #define FILEDONE          0x17
00134 #define FILEABORT         0x18
00135 #define BLOCKOK           0x19
00136 #define QUERYFOUND        0x1A
00137 #define START_TIMEOUT     0x1B
00138 #define BLOCK_TIMEOUT     0x1C
00139 #define BLOCKERR_MASK     0x20
00140
00142 #define BLOCKERR_SOH      0x21
00143
00145 #define BLOCKERR_CHK      0x22
00146
00148 #define BLOCKERR_CRCH     0x23
00149
00151 #define BLOCKERR_CRCL     0x24
00152
00154 #define BLOCKERR_SEQUENCE   0x25
00155
00157 #define BLOCKERR_PARTIAL    0x26
00158
00160 #define BLOCKERR_DUPLICATE 0x27
00161
00164 // two possible communication modes: serial mode, or radio/ota mode.
00165 enum {
00166   COMM_SERIAL  = 0x01,  // in serial mode (uart or ezsp spi)
00167   COMM_RADIO   = 0x02,  // in radio mode
00168 };
00169
00170 #endif //__BOOTLOADER_COMMON_H__
00171
00172
```

8.15 bootloader-gpio.h File Reference

Enumerations

- enum blState_e {
  BL_ST_UP, BL_ST_DOWN, BL_ST_POLLING_LOOP, BL_ST_DOWNLOAD_LOOP,
  BL_ST_DOWNLOAD_FAILURE, BL_ST_DOWNLOAD_SUCCESS }

Functions

- void bootloadGpioInit (void)
- void bootloadStateIndicator (enum blState_e state)
- boolean bootloadForceActivation (void)

State Indicator Macros

The bootloader indicates which state it is in by calling these macros. Map them to the ::halBootloadState-Indicator function (in bootloder-gpio.c) if you want to display that bootloader state. Used to blink the LED's or otherwise signal bootloader activity.

- #define BL_STATE_UP()
- #define BL_STATE_DOWN()
- #define BL_STATE_POLLING_LOOP()
- #define BL_STATE_DOWNLOAD_LOOP()
- #define BL_STATE_DOWNLOAD_SUCCESS()
- #define BL_STATE_DOWNLOAD_FAILURE()

8.15.1 Detailed Description

See GPIO for detailed documentation.

Definition in file bootloader-gpio.h.

8.16 bootloader-gpio.h

```
00001
00014 #ifndef __BOOTLOADER_GPIO_H__
00015 #define __BOOTLOADER_GPIO_H__
00016
00024 //
00025
00028 #define BL_STATE_UP() do { bootloadStateIndicator(BL_ST_UP); } while(0)
00029
00034 #define BL_STATE_DOWN() do { bootloadStateIndicator(BL_ST_DOWN); } while(0)
00035
00038 #define BL_STATE_POLLING_LOOP() do {
     bootloadStateIndicator(BL_ST_POLLING_LOOP); } while(0)
00039
00042 #define BL_STATE_DOWNLOAD_LOOP() do {
     bootloadStateIndicator(BL_ST_DOWNLOAD_LOOP); } while(0)
00043
00046 #define BL_STATE_DOWNLOAD_SUCCESS() do {
     bootloadStateIndicator(BL_ST_DOWNLOAD_SUCCESS); } while(0)
00047
00050 #define BL_STATE_DOWNLOAD_FAILURE() do {
     bootloadStateIndicator(BL_ST_DOWNLOAD_FAILURE); } while(0)
00051
00056 enum blState_e {
00058   BL_ST_UP,
00060   BL_ST_DOWN,
00062   BL_ST_POLLING_LOOP,
00064   BL_ST_DOWNLOAD_LOOP,
00066   BL_ST_DOWNLOAD_FAILURE,
00068   BL_ST_DOWNLOAD_SUCCESS
00069 };
00070
00073 void bootloadGpioInit(void);
00074
00078 void bootloadStateIndicator(enum blState_e state
     );
00079
00082 boolean bootloadForceActivation( void );
00083
00084 #endif // __BOOTLOADER_GPIO_H__
00085
```

8.17   bootloader-interface-app.h File Reference

```
#include "hal/micro/bootloader-eeprom.h"
```

Macros

- #define BOOTLOADER_SEGMENT_SIZE_LOG2
- #define BOOTLOADER_SEGMENT_SIZE
- #define BL_IMAGE_IS_VALID_CONTINUE

Functions

- int8u halAppBootloaderInit (void)
- const HalEepromInformationType ∗ halAppBootloaderInfo (void)
- void halAppBootloaderShutdown (void)
- void halAppBootloaderImageIsValidReset (void)
- int16u halAppBootloaderImageIsValid (void)
- EmberStatus halAppBootloaderInstallNewImage (void)
- int8u halAppBootloaderWriteRawStorage (int32u address, const int8u ∗data, int16u len)
- int8u halAppBootloaderReadRawStorage (int32u address, int8u ∗data, int16u len)
- int8u halAppBootloaderEraseRawStorage (int32u address, int32u len)
- boolean halAppBootloaderStorageBusy (void)
- int8u halAppBootloaderReadDownloadSpace (int16u pageToBeRead, int8u ∗destRamBuffer)
- int8u halAppBootloaderWriteDownloadSpace (int16u pageToBeWritten, int8u ∗RamPtr)
- int8u halAppBootloaderGetImageData (int32u ∗timestamp, int8u ∗userData)
- int16u halAppBootloaderGetVersion (void)
- int16u halAppBootloaderGetRecoveryVersion (void)

8.17.1   Detailed Description

See Application for documentation.

Definition in file bootloader-interface-app.h.

8.18   bootloader-interface-app.h

```
00001
00018 #ifndef __BOOTLOADER_INTERFACE_APP_H__
00019 #define __BOOTLOADER_INTERFACE_APP_H__
00020
00021 #include "hal/micro/bootloader-eeprom.h"
00022
00027 #define BOOTLOADER_SEGMENT_SIZE_LOG2   6
00028
00032 #define BOOTLOADER_SEGMENT_SIZE        (1 << BOOTLOADER_SEGMENT_SIZE_LOG2)
00033
00034
00042 int8u halAppBootloaderInit(void);
00043
00050 const HalEepromInformationType *halAppBootloaderInfo(void);
00051
00055 void halAppBootloaderShutdown(void);
00056
00060 void halAppBootloaderImageIsValidReset(void);
```

```
00061
00065 #define BL_IMAGE_IS_VALID_CONTINUE  ((int16u)0xFFFF)
00066
00092 int16u halAppBootloaderImageIsValid(void);
00093
00094
00099 EmberStatus halAppBootloaderInstallNewImage
      (void);
00100
00101
00102
00123 int8u halAppBootloaderWriteRawStorage(
      int32u address,
00124                                    const int8u *data,
00125                                    int16u len);
00126
00141 int8u halAppBootloaderReadRawStorage(int32u
       address, int8u *data, int16u len);
00142
00160 int8u halAppBootloaderEraseRawStorage(
      int32u address, int32u len);
00161
00167 boolean halAppBootloaderStorageBusy(void);
00168
00181 int8u halAppBootloaderReadDownloadSpace(
      int16u pageToBeRead,
00182                                    int8u* destRamBuffer);
00183
00184
00197 int8u halAppBootloaderWriteDownloadSpace
      (int16u pageToBeWritten,
00198                                    int8u* RamPtr);
00199
00208 int8u halAppBootloaderGetImageData(int32u
       *timestamp, int8u *userData);
00209
00210
00213 int16u halAppBootloaderGetVersion(void);
00214
00215
00218 int16u halAppBootloaderGetRecoveryVersion
      (void);
00219
00220
00221 #endif //__BOOTLOADER_INTERFACE_APP_H__
00222
```

8.19   bootloader-interface-standalone.h File Reference

Macros

- #define NO_BOOTLOADER_MODE
- #define STANDALONE_BOOTLOADER_NORMAL_MODE
- #define STANDALONE_BOOTLOADER_RECOVERY_MODE

Functions

- int16u halGetStandaloneBootloaderVersion (void)
- EmberStatus halLaunchStandaloneBootloader (int8u mode)

8.19.1   Detailed Description

See Standalone for documentation.

Definition in file bootloader-interface-standalone.h.

8.20 bootloader-interface-standalone.h

```
00001
00017 #ifndef __BOOTLOADER_STANDALONE_H__
00018 #define __BOOTLOADER_STANDALONE_H__
00019
00028 int16u halGetStandaloneBootloaderVersion
      (void);
00029
00033 #define NO_BOOTLOADER_MODE             0xFF
00034
00037 #define STANDALONE_BOOTLOADER_NORMAL_MODE   1
00038
00041 #define STANDALONE_BOOTLOADER_RECOVERY_MODE 0
00042
00061 EmberStatus halLaunchStandaloneBootloader
      (int8u mode);
00062
00063
00064 #endif //__BOOTLOADER_STANDALONE_H__
00065
```

8.21 bootloader-interface.h File Reference

```
#include "bootloader-interface-app.h"
#include "bootloader-interface-standalone.h"
```

Macros

- #define BOOTLOADER_BASE_TYPE(extendedType)
- #define BOOTLOADER_MAKE_EXTENDED_TYPE(baseType, extendedSpecifier)
- #define BL_EXT_TYPE_NULL
- #define BL_EXT_TYPE_STANDALONE_UNKNOWN
- #define BL_EXT_TYPE_SERIAL_UART
- #define BL_EXT_TYPE_SERIAL_UART_OTA
- #define BL_EXT_TYPE_EZSP_SPI
- #define BL_EXT_TYPE_EZSP_SPI_OTA
- #define BL_EXT_TYPE_APP_UNKNOWN
- #define BL_EXT_TYPE_APP_SPI
- #define BL_EXT_TYPE_APP_I2C
- #define BOOTLOADER_INVALID_VERSION

Functions

- BlBaseType halBootloaderGetType (void)
- BlExtendedType halBootloaderGetInstalledType (void)
- int16u halGetBootloaderVersion (void)
- void halGetExtendedBootloaderVersion (int32u ∗emberVersion, int32u ∗customerVersion)

Bootloader Numerical Definitions

These are numerical definitions for the possible bootloader types and a typedef of the bootloader base type.

- #define BL_TYPE_NULL

- #define BL_TYPE_STANDALONE
- #define BL_TYPE_APPLICATION
- #define BL_TYPE_BOOTLOADER
- #define BL_TYPE_SMALL_BOOTLOADER

Bootloader type definitions

These are the type definitions for the bootloader.

- typedef int8u BlBaseType
- typedef int16u BlExtendedType

8.21.1 Detailed Description

See Common for detailed documentation.

Definition in file bootloader-interface.h.

8.22 bootloader-interface.h

```
00001
00014 #ifndef __BOOTLOADER_INTERFACE_H__
00015 #define __BOOTLOADER_INTERFACE_H__
00016
00017 #include "bootloader-interface-app.h"
00018 #include "bootloader-interface-standalone.h"
00019
00027 #define BL_TYPE_NULL          (0)
00028 #define BL_TYPE_STANDALONE   (1)
00029 #define BL_TYPE_APPLICATION  (2)
00030 #define BL_TYPE_BOOTLOADER    (3)      // Generic bootloader type
00031 #define BL_TYPE_SMALL_BOOTLOADER  (4)  // Generic, but small bootloader type
00032
00041 typedef int8u BlBaseType;
00044 typedef int16u BlExtendedType;
00052 BlBaseType halBootloaderGetType(void);
00053
00054
00058 #define BOOTLOADER_BASE_TYPE(extendedType)  \
00059                               ((int8u)(((extendedType) >> 8) & 0xFF))
00060
00064 #define BOOTLOADER_MAKE_EXTENDED_TYPE(baseType, extendedSpecifier) \
00065         ((int16u)(((int16u)baseType) << 8) |
    (((int16u)extendedSpecifier)&0xFF))
00066
00069 #define BL_EXT_TYPE_NULL                 ((BL_TYPE_NULL << 8) | 0x00)
00070
00073 #define BL_EXT_TYPE_STANDALONE_UNKNOWN  ((BL_TYPE_STANDALONE << 8) | 0x00)
00074
00077 #define BL_EXT_TYPE_SERIAL_UART         ((BL_TYPE_STANDALONE << 8) | 0x01)
00078
00079 // skipping the extSpecifier of 0x02 in case we decide we want it to
00080 //   be a bitmask for "OTA only"
00081
00084 #define BL_EXT_TYPE_SERIAL_UART_OTA     ((BL_TYPE_STANDALONE << 8) | 0x03)
00085 #define BL_EXT_TYPE_EZSP_SPI            ((BL_TYPE_STANDALONE << 8) | 0x04)
00086 #define BL_EXT_TYPE_EZSP_SPI_OTA        ((BL_TYPE_STANDALONE << 8) | 0x06)
00087
00090 #define BL_EXT_TYPE_APP_UNKNOWN         ((BL_TYPE_APPLICATION << 8) | 0x00)
00091
00094 #define BL_EXT_TYPE_APP_SPI             ((BL_TYPE_APPLICATION << 8) | 0x01)
00095
00098 #define BL_EXT_TYPE_APP_I2C             ((BL_TYPE_APPLICATION << 8) | 0x02)
00099
00103 BlExtendedType halBootloaderGetInstalledType
     (void);
```

```
00104
00105
00106
00109 #define BOOTLOADER_INVALID_VERSION 0xFFFF
00110
00117 int16u halGetBootloaderVersion(void);
00118
00119
00130 void halGetExtendedBootloaderVersion(int32u
      * emberVersion, int32u* customerVersion);
00131
00132 #endif // __BOOTLOADER_INTERFACE_H__
00133
```

8.23   bootloader-serial.h File Reference

Functions

- void serInit (void)
- void serPutFlush (void)
- void serPutChar (int8u ch)
- void serPutStr (const char ∗str)
- void serPutBuf (const int8u buf[ ], int8u size)
- void serPutDecimal (int16u val)
- void serPutHex (int8u byte)
- void serPutHexInt (int16u word)
- boolean serCharAvailable (void)
- int8u serGetChar (int8u ∗ch)
- void serGetFlush (void)

8.23.1   Detailed Description

See Serial for detailed documentation.

Definition in file bootloader-serial.h.

8.24   bootloader-serial.h

```
00001
00007 //[[ Author(s): David Iacobone, diacobone@ember.com
00008 //            Lee Taylor, lee@ember.com
00009 //]]
00010
00018 #ifndef __BOOTLOADER_SERIAL_H__
00019 #define __BOOTLOADER_SERIAL_H__
00020
00023 void serInit(void);
00024
00027 void serPutFlush(void);
00031 void serPutChar(int8u ch);
00035 void serPutStr(const char *str);
00040 void serPutBuf(const int8u buf[], int8u size);
00044 void serPutDecimal(int16u val);
00048 void serPutHex(int8u byte);
00052 void serPutHexInt(int16u word);
00053
00057 boolean serCharAvailable(void);
00062 int8u   serGetChar(int8u* ch);
00063
00066 void    serGetFlush(void);
00067
```

```
00068 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00069 #ifdef BTL_HAS_EZSP_SPI
00070 extern int8u preSpiComBufIndex;
00071 #define serResetCharInput() preSpiComBufIndex = 0;
00072 #endif
00073 #endif
00074
00075 #endif //__BOOTLOADER_SERIAL_H__
00076
```

8.25   button.h File Reference

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

8.25.1   Detailed Description

See Button Control for documentation.

Definition in file button.h.

8.26   button.h

```
00001
00024 #define BUTTON_PRESSED  1
00025
00028 #define BUTTON_RELEASED 0
00029
00036 void halInternalInitButton(void);
00037
00049 int8u halButtonState(int8u button);
00050
00063 int8u halButtonPinState(int8u button);
00064
00079 void halButtonIsr(int8u button, int8u state);
00080
```

8.27   buzzer.h File Reference

Functions

- void halPlayTune_P (int8u PGM *tune, boolean bkg)
- void halStackIndicatePresence (void)

Note Definitions

Flats are used instead of sharps because # is a special character.

- #define NOTE_C3
- #define NOTE_Db3
- #define NOTE_D3
- #define NOTE_Eb3
- #define NOTE_E3
- #define NOTE_F3
- #define NOTE_Gb3
- #define NOTE_G3
- #define NOTE_Ab3
- #define NOTE_A3
- #define NOTE_Bb3
- #define NOTE_B3
- #define NOTE_C4
- #define NOTE_Db4
- #define NOTE_D4
- #define NOTE_Eb4
- #define NOTE_E4
- #define NOTE_F4
- #define NOTE_Gb4
- #define NOTE_G4
- #define NOTE_Ab4
- #define NOTE_A4
- #define NOTE_Bb4
- #define NOTE_B4
- #define NOTE_C5
- #define NOTE_Db5
- #define NOTE_D5
- #define NOTE_Eb5
- #define NOTE_E5
- #define NOTE_F5
- #define NOTE_Gb5
- #define NOTE_G5
- #define NOTE_Ab5
- #define NOTE_A5
- #define NOTE_Bb5
- #define NOTE_B5

8.27.1 Detailed Description

See Buzzer Control for documentation.

Definition in file buzzer.h.

8.28 buzzer.h

```
00001
00023 #define NOTE_C3 119
00024 #define NOTE_Db3        112
00025 #define NOTE_D3 106
00026 #define NOTE_Eb3        100
00027 #define NOTE_E3 94
00028 #define NOTE_F3 89
00029 #define NOTE_Gb3        84
00030 #define NOTE_G3 79
00031 #define NOTE_Ab3        74
00032 #define NOTE_A3 70
00033 #define NOTE_Bb3        66
00034 #define NOTE_B3 63
00035 #define NOTE_C4 59
00036 #define NOTE_Db4        55
00037 #define NOTE_D4 52
00038 #define NOTE_Eb4        49
00039 #define NOTE_E4 46
00040 #define NOTE_F4 44
00041 #define NOTE_Gb4        41
00042 #define NOTE_G4 39
00043 #define NOTE_Ab4        37
00044 #define NOTE_A4 35
00045 #define NOTE_Bb4        33
00046 #define NOTE_B4 31
00047 #define NOTE_C5 29
00048 #define NOTE_Db5        27
00049 #define NOTE_D5 26
00050 #define NOTE_Eb5        24
00051 #define NOTE_E5 23
00052 #define NOTE_F5 21
00053 #define NOTE_Gb5        20
00054 #define NOTE_G5 19
00055 #define NOTE_Ab5        18
00056 #define NOTE_A5 17
00057 #define NOTE_Bb5        16
00058 #define NOTE_B5 15
00059
00082 void halPlayTune_P(int8u PGM *tune, boolean bkg);
00083
00084
00089 void halStackIndicatePresence(void);
00090
```

8.29 child.h File Reference

Functions

- EmberNodeId emberChildId (int8u childIndex)
- int8u emberChildIndex (EmberNodeId childId)
- EmberStatus emberGetChildData (int8u index, EmberEUI64 childEui64Return, EmberNodeType ∗childTypeReturn)
- void emberChildJoinHandler (int8u index, boolean joining)
- EmberStatus emberPollForData (void)
- void emberPollCompleteHandler (EmberStatus status)
- EmberStatus emberSetMessageFlag (EmberNodeId childId)
- EmberStatus emberClearMessageFlag (EmberNodeId childId)
- void emberPollHandler (EmberNodeId childId, boolean transmitExpected)
- int8u emberChildCount (void)
- int8u emberRouterChildCount (void)
- int8u emberMaxChildCount (void)
- int8u emberMaxRouterChildCount (void)
- EmberNodeId emberGetParentNodeId (void)
- EmberEUI64 emberGetParentEui64 (void)

Power Management

- #define EMBER_HIGH_PRIORITY_TASKS
- enum {
  EMBER_OUTGOING_MESSAGES, EMBER_INCOMING_MESSAGES, EMBER_RADIO_IS_-
  ON, EMBER_TRANSPORT_ACTIVE,
  EMBER_APS_LAYER_ACTIVE, EMBER_ASSOCIATING, EMBER_ZLL_TOUCH_LINKING
  }
- int16u emberCurrentStackTasks (void)
- boolean emberOkToNap (void)
- boolean emberOkToHibernate (void)
- boolean emberOkToLongPoll (void)
- void emberStackPowerDown (void)
- void emberStackPowerUp (void)

8.29.1 Detailed Description

See End Devices for documentation.

Definition in file child.h.

8.30 child.h

```
00001
00023 EmberNodeId emberChildId(int8u childIndex);
00024
00031 int8u emberChildIndex(EmberNodeId childId);
00032
00047 EmberStatus emberGetChildData(int8u index,
00048                               EmberEUI64 childEui64Return,
00049                               EmberNodeType *childTypeReturn);
00050
00066 void emberChildJoinHandler(int8u index, boolean
     joining);
00067
00098 EmberStatus emberPollForData(void);
00099
00114 void emberPollCompleteHandler(EmberStatus
     status);
00115
00129 EmberStatus emberSetMessageFlag(EmberNodeId
      childId);
00130
00143 EmberStatus emberClearMessageFlag(EmberNodeId
      childId);
00144
00162 void emberPollHandler(EmberNodeId childId, boolean
     transmitExpected);
00163
00164
00165 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00166
00170 int8u emberChildCount(void);
00171
00177 int8u emberRouterChildCount(void);
00178
00184 int8u emberMaxChildCount(void);
00185
00191 int8u emberMaxRouterChildCount(void);
00192
00199 EmberNodeId emberGetParentNodeId(void);
00200
00207 EmberEUI64 emberGetParentEui64(void);
00208
00209 #else    // Doxygen ignores the following
00210
```

```
00211 extern int8u emMaxEndDeviceChildren;     // maximum for this node
00212 extern int8u emEndDeviceChildCount;      // how many we have
00213
00214 // The '+ 0' prevents anyone from accidentally assigning to these.
00215 #define emberChildCount()          (emEndDeviceChildCount + 0)
00216 #define emberRouterChildCount()    0
00217 #define emberMaxChildCount()       (emMaxEndDeviceChildren + 0)
00218 #define emberMaxRouterChildCount() 0
00219
00220 // Implemented in ember-stack-common.c
00221 EmberNodeId emberGetParentNodeId(void);
00222 int8u* emberGetParentEui64(void);
00223
00224
00225 #endif
00226
00230
00233 enum
00234 {
00236    EMBER_OUTGOING_MESSAGES = 0x01,
00238    EMBER_INCOMING_MESSAGES = 0x02,
00243    EMBER_RADIO_IS_ON = 0x04,
00245    EMBER_TRANSPORT_ACTIVE = 0x08,
00247    EMBER_APS_LAYER_ACTIVE = 0x10,
00249    EMBER_ASSOCIATING = 0x20,
00251    EMBER_ZLL_TOUCH_LINKING = 0x40,
00252 };
00253
00256 #define EMBER_HIGH_PRIORITY_TASKS \
00257 (EMBER_OUTGOING_MESSAGES | EMBER_INCOMING_MESSAGES | EMBER_RADIO_IS_ON)
00258
00272 int16u emberCurrentStackTasks(void);
00273
00285 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00286 boolean emberOkToNap(void);
00287 #else
00288 #define emberOkToNap() \
00289  (! (emberCurrentStackTasks() & EMBER_HIGH_PRIORITY_TASKS))
00290 #endif
00291
00301 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00302 boolean emberOkToHibernate(void);
00303 #else
00304 #define emberOkToHibernate() (! emberCurrentStackTasks())
00305 #endif
00306
00312 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00313 boolean emberOkToLongPoll(void);
00314 #else
00315 #define emberOkToLongPoll() (! emberPendingAckedMessages())
00316 #endif
00317
00325 void emberStackPowerDown(void);
00326
00331 void emberStackPowerUp(void);
00332
00333 // @} END Power Management
00334
00335 // @} END addtogroup
00336
```

8.31   command-interpreter2.h File Reference

Data Structures

- struct EmberCommandEntry

  Command entry for a command table.

Macros

- #define MAX_TOKEN_COUNT

- #define emberCommandEntryAction(name, action, argumentTypes, description)
- #define emberCommandEntrySubMenu(name, subMenu, description)
- #define emberCommandEntryTerminator()
- #define EMBER_COMMAND_INTERPRETER_CONFIGURATION_ECHO
- #define emberProcessCommandInput(port)
- #define emberCommandInterpreterEchoOn()
- #define emberCommandInterpreterEchoOff()
- #define emberCommandInterpreterIsEchoOn()

Typedefs

- typedef void(∗ CommandAction)(void)

Enumerations

- enum EmberCommandStatus {
  EMBER_CMD_SUCCESS, EMBER_CMD_ERR_PORT_PROBLEM, EMBER_CMD_ERR_NO_SUCH_COMMAND, EMBER_CMD_ERR_WRONG_NUMBER_OF_ARGUMENTS,
  EMBER_CMD_ERR_ARGUMENT_OUT_OF_RANGE, EMBER_CMD_ERR_ARGUMENT_SYNTAX_ERROR, EMBER_CMD_ERR_STRING_TOO_LONG, EMBER_CMD_ERR_INVALID_ARGUMENT_TYPE }

Functions

- void emberCommandActionHandler (const CommandAction action)
- void emberCommandErrorHandler (EmberCommandStatus status)
- void emberPrintCommandUsage (EmberCommandEntry ∗entry)
- void emberPrintCommandUsageNotes (void)
- void emberPrintCommandTable (void)
- void emberCommandReaderInit (void)
- boolean emberProcessCommandString (int8u ∗input, int8u sizeOrPort)

Variables

- EmberCommandEntry ∗ emberCurrentCommand
- EmberCommandEntry emberCommandTable [ ]
- int8u emberCommandInterpreter2Configuration

Command Table Settings

- #define EMBER_MAX_COMMAND_ARGUMENTS
- #define EMBER_COMMAND_BUFFER_LENGTH
- #define EMBER_COMMAND_INTEPRETER_HAS_DESCRIPTION_FIELD

Functions to Retrieve Arguments

Use the following functions in your functions that process commands to retrieve arguments from the command interpreter. These functions pull out unsigned integers, signed integers, and strings, and hex strings. Index 0 is the first command argument.

- #define emberCopyKeyArgument(index, keyDataPointer)
- #define emberCopyEui64Argument(index, eui64)
- int8u emberCommandArgumentCount (void)
- int32u emberUnsignedCommandArgument (int8u argNum)
- int16s emberSignedCommandArgument (int8u argNum)
- int8u ∗ emberStringCommandArgument (int8s argNum, int8u ∗length)
- int8u emberCopyStringArgument (int8s argNum, int8u ∗destination, int8u maxLength, boolean left-Pad)

8.31.1   Detailed Description

Processes commands coming from the serial port. See Commands2 for documentation.

Definition in file command-interpreter2.h.

8.32   command-interpreter2.h

```
00001
00010 #ifndef __COMMAND_INTERPRETER2_H__
00011 #define __COMMAND_INTERPRETER2_H__
00012
00100 #ifndef EMBER_MAX_COMMAND_ARGUMENTS
00101
00104 #define EMBER_MAX_COMMAND_ARGUMENTS 10
00105 #endif
00106
00107 #ifndef EMBER_COMMAND_BUFFER_LENGTH
00108 #define EMBER_COMMAND_BUFFER_LENGTH 100
00109 #endif
00110
00115 #if defined(DOXYGEN_SHOULD_SKIP_THIS)
00116 #define EMBER_COMMAND_INTEPRETER_HAS_DESCRIPTION_FIELD
00117 #endif
00118
00119
00123 // The (+ 1) takes into account the leading command.
00124 #define MAX_TOKEN_COUNT (EMBER_MAX_COMMAND_ARGUMENTS + 1)
00125
00126 typedef void (*CommandAction)(void);
00127
00128 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00129
00131 typedef struct {
00132 #else
00133 typedef PGM struct {
00134 #endif
00135
00138   PGM_P name;
00144   CommandAction action;
00171   PGM_P argumentTypes;
00175   PGM_P description;
00176 } EmberCommandEntry;
00177
00178
00179 #if defined(EMBER_COMMAND_INTEPRETER_HAS_DESCRIPTION_FIELD)
00180   /* @brief Macro to define a CLI action */
00181   #define emberCommandEntryAction(name, action, argumentTypes, description) \
00182     { (name), (action), (argumentTypes), (description) }
00183
```

```
00184  /* @brief Macro to define a CLI sub-menu (nested command) */
00185  #define emberCommandEntrySubMenu(name, subMenu, description)   \
00186    { (name), NULL, (PGM_P)(subMenu), (description) }
00187
00188  /* @briefy Macro to define a command entry array terminator.*/
00189  #define emberCommandEntryTerminator() \
00190    { NULL, NULL, NULL, NULL }
00191
00192 #else  // Don't include description data in struct
00193  /* @brief Macro to define a CLI action */
00194  #define emberCommandEntryAction(name, action, argumentTypes, description) \
00195    { (name), (action), (argumentTypes), NULL }
00196
00197  /* @brief Macro to define a CLI sub-menu (nested command) */
00198  #define emberCommandEntrySubMenu(name, subMenu, description) \
00199    { (name), NULL, (PGM_P)(subMenu), NULL }
00200
00201  /* @briefy Macro to define a command entry array terminator.*/
00202  #define emberCommandEntryTerminator() \
00203    { NULL, NULL, NULL, NULL }
00204
00205 #endif
00206
00213 extern EmberCommandEntry *emberCurrentCommand
      ;
00214
00215 extern EmberCommandEntry emberCommandTable[];
00216
00220 extern int8u emberCommandInterpreter2Configuration
      ;
00221
00222 #define EMBER_COMMAND_INTERPRETER_CONFIGURATION_ECHO (0x01)
00223
00224 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00225
00230 enum EmberCommandStatus
00231 #else
00232 typedef int8u EmberCommandStatus;
00233 enum
00234 #endif
00235 {
00236   EMBER_CMD_SUCCESS,
00237   EMBER_CMD_ERR_PORT_PROBLEM,
00238   EMBER_CMD_ERR_NO_SUCH_COMMAND,
00239   EMBER_CMD_ERR_WRONG_NUMBER_OF_ARGUMENTS
      ,
00240   EMBER_CMD_ERR_ARGUMENT_OUT_OF_RANGE,

00241   EMBER_CMD_ERR_ARGUMENT_SYNTAX_ERROR,

00242   EMBER_CMD_ERR_STRING_TOO_LONG,
00243   EMBER_CMD_ERR_INVALID_ARGUMENT_TYPE
00244 };
00245
00255 int8u emberCommandArgumentCount(void);
00256
00258 int32u emberUnsignedCommandArgument(int8u
      argNum);
00259
00261 int16s emberSignedCommandArgument(int8u
      argNum);
00262
00271 int8u *emberStringCommandArgument(int8s
      argNum, int8u *length);
00272
00285 int8u emberCopyStringArgument(int8s argNum,
00286                               int8u *destination,
00287                               int8u maxLength,
00288                               boolean leftPad);
00289
00293 #define emberCopyKeyArgument(index, keyDataPointer)       \
00294   (emberCopyStringArgument((index),                       \
00295                            emberKeyContents((keyDataPointer)), \
00296                            EMBER_ENCRYPTION_KEY_SIZE,      \
00297                            TRUE))
00298
00300 #define emberCopyEui64Argument(index, eui64) \
00301   (emberCopyStringArgument((index), (eui64), EUI64_SIZE, TRUE))
00302
00310 void emberCommandActionHandler(const CommandAction
```

```
      action);
00317 void emberCommandErrorHandler(EmberCommandStatus status
      );
00318 void emberPrintCommandUsage(EmberCommandEntry
      *entry);
00319 void emberPrintCommandUsageNotes(void);
00320 void emberPrintCommandTable(void);
00321
00324 void emberCommandReaderInit(void);
00325
00328 boolean emberProcessCommandString(int8u *input,
      int8u sizeOrPort);
00329
00338 #define emberProcessCommandInput(port) \
00339   emberProcessCommandString(NULL, port)
00340
00343 #define emberCommandInterpreterEchoOn()                  \
00344   (emberCommandInterpreter2Configuration               \
00345    |= EMBER_COMMAND_INTERPRETER_CONFIGURATION_ECHO)
00346
00349 #define emberCommandInterpreterEchoOff()                 \
00350   (emberCommandInterpreter2Configuration               \
00351    &= (~EMBER_COMMAND_INTERPRETER_CONFIGURATION_ECHO))
00352
00355 #define emberCommandInterpreterIsEchoOn()                \
00356   (emberCommandInterpreter2Configuration               \
00357    & EMBER_COMMAND_INTERPRETER_CONFIGURATION_ECHO)
00358
00361 #endif // __COMMAND_INTERPRETER2_H__
```

8.33   config.h File Reference

Macros

- #define EMBER_MAJOR_VERSION
- #define EMBER_MINOR_VERSION
- #define EMBER_PATCH_VERSION
- #define EMBER_SPECIAL_VERSION
- #define EMBER_BUILD_NUMBER
- #define EMBER_FULL_VERSION
- #define EMBER_VERSION_TYPE
- #define SOFTWARE_VERSION

8.33.1   Detailed Description

See Stack Information for documentation.

Definition in file config.h.

8.34   config.h

```
00001
00018 // The 4 digit version: A.B.C.D
00019 #define EMBER_MAJOR_VERSION    4
00020 #define EMBER_MINOR_VERSION    7
00021 #define EMBER_PATCH_VERSION    2
00022 #define EMBER_SPECIAL_VERSION 0
00023
00024 // 2 bytes
00025 #define EMBER_BUILD_NUMBER    84
00026
00027 #define EMBER_FULL_VERSION (  ((int16u)EMBER_MAJOR_VERSION << 12) \
00028                             | ((int16u)EMBER_MINOR_VERSION <<  8) \
```

```
00029                                  | ((int16u)EMBER_PATCH_VERSION <<  4) \
00030                                  | ((int16u)EMBER_SPECIAL_VERSION))
00031
00032 #define EMBER_VERSION_TYPE EMBER_VERSION_TYPE_PRE_RELEASE
00033
00043 #define SOFTWARE_VERSION EMBER_FULL_VERSION
00044
```

8.35   crc.h File Reference

Macros

- #define INITIAL_CRC
- #define CRC32_START
- #define CRC32_END

Functions

- int16u halCommonCrc16 (int8u newByte, int16u prevResult)
- int32u halCommonCrc32 (int8u newByte, int32u prevResult)

8.35.1   Detailed Description

See Cyclic Redundancy Code (CRC) for detailed documentation.

Definition in file crc.h.

8.36   crc.h

```
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 // Commonly used initial and expected final CRC32 values
00049 #define INITIAL_CRC              0xFFFFFFFFL
00050 #define CRC32_START              INITIAL_CRC
00051 #define CRC32_END                0xDEBB20E3L  // For CRC32 POLYNOMIAL run
     LSB-MSB
00052
00053
00057 #endif //__CRC_H__
00058
```

8.37   dev0680.h File Reference

Custom Baud Rate Definitions

The following define is used with defining a custom baud rate for the UART. This define provides a simple hook into the definition of the baud rates used with the UART. The baudSettings[] array in uart.c links the

BAUD_∗ defines with the actual register values needed for operating the UART. The array baudSettings[] can be edited directly for a custom baud rate or another entry (the register settings) can be provided here with this define.

- #define EMBER_SERIAL_BAUD_CUSTOM

LED Definitions

The following are used to aid in the abstraction with the LED connections. The microcontroller-specific sources use these definitions so they are able to work across a variety of boards which could have different connections. The names and ports/pins used below are intended to match with a schematic of the system to provide the abstraction.

The HalBoardLedPins enum values should always be used when manipulating the state of LEDs, as they directly refer to the GPIOs to which the LEDs are connected.

Note: LEDs 0 and 1 are on the RCM.

Note: LED 2 is on the breakout board (dev0680).

Note: LED 3 simply redirects to LED 2.

- enum HalBoardLedPins {
  BOARDLED0, BOARDLED1, BOARDLED2, BOARDLED3,
  BOARD_ACTIVITY_LED, BOARD_HEARTBEAT_LED }

Button Definitions

The following are used to aid in the abstraction with the Button connections. The microcontroller-specific sources use these definitions so they are able to work across a variety of boards which could have different connections. The names and ports/pins used below are intended to match with a schematic of the system to provide the abstraction.

The BUTTONn macros should always be used with manipulating the buttons as they directly refer to the GPIOs to which the buttons are connected.

Note

The GPIO number must match the IRQ letter

- #define BUTTON0
- #define BUTTON0_IN
- #define BUTTON0_SEL()
- #define BUTTON0_ISR
- #define BUTTON0_INTCFG
- #define BUTTON0_INT_EN_BIT
- #define BUTTON0_FLAG_BIT
- #define BUTTON0_MISS_BIT
- #define BUTTON1
- #define BUTTON1_IN
- #define BUTTON1_SEL()
- #define BUTTON1_ISR
- #define BUTTON1_INTCFG
- #define BUTTON1_INT_EN_BIT
- #define BUTTON1_FLAG_BIT
- #define BUTTON1_MISS_BIT

Radio HoldOff Configuration Definitions

This define does not equate to anything. It is used as a trigger to enable Radio HoldOff support.

The following are used to aid in the abstraction with Radio HoldOff (RHO). The microcontroller-specific sources use these definitions so they are able to work across a variety of boards which could have different connections. The names and ports/pins used below are intended to match with a schematic of the system to provide the abstraction.

The Radio HoldOff input GPIO is abstracted like BUTTON0/1.

- #define RHO_GPIO
- #define RHO_ASSERTED
- #define RHO_CFG
- #define RHO_IN
- #define RHO_OUT
- #define RHO_SEL()
- #define RHO_ISR
- #define RHO_INTCFG
- #define RHO_INT_EN_BIT
- #define RHO_FLAG_BIT
- #define RHO_MISS_BIT
- #define PWRUP_CFG_LED_RHO_FOR_RHO
- #define PWRUP_OUT_LED_RHO_FOR_RHO
- #define PWRDN_CFG_LED_RHO_FOR_RHO
- #define PWRDN_OUT_LED_RHO_FOR_RHO
- #define WAKE_ON_LED_RHO_FOR_RHO
- #define PWRUP_CFG_LED_RHO_FOR_LED
- #define PWRUP_OUT_LED_RHO_FOR_LED
- #define PWRDN_CFG_LED_RHO_FOR_LED
- #define PWRDN_OUT_LED_RHO_FOR_LED
- #define WAKE_ON_LED_RHO_FOR_LED
- #define PWRUP_CFG_LED_RHO
- #define PWRUP_OUT_LED_RHO
- #define PWRDN_CFG_LED_RHO
- #define PWRDN_OUT_LED_RHO
- #define WAKE_ON_LED_RHO
- #define halInternalInitRadioHoldOff()
- #define WAKE_ON_LED_RHO_VAR
- #define ADJUST_GPIO_CONFIG_LED_RHO(enableRadioHoldOff)
- int8u WAKE_ON_LED_RHO_VAR

Temperature sensor ADC channel

Define the analog input channel connected to the LM-20 temperature sensor. The scale factor compensates for different platform input ranges. PB5/ADC0 must be an analog input. PC7 must be an output and set to a high level to power the sensor.

- #define TEMP_SENSOR_ADC_CHANNEL
- #define TEMP_SENSOR_SCALE_FACTOR

Packet Trace

When PACKET_TRACE is defined, ::GPIO_PACFGH will automatically be setup by halInit() to enable Packet Trace support on PA4 and PA5, in addition to the configuration specified below.

Note

> This define will override any settings for PA4 and PA5.

- #define PACKET_TRACE

ENABLE_OSC32K

When ENABLE_OSC32K is defined, halInit() will configure system timekeeping to utilize the external 32.768 kHz crystal oscillator rather than the internal 1 kHz RC oscillator.

Note

> ENABLE_OSC32K is mutually exclusive with ENABLE_ALT_FUNCTION_NTX_ACTIVE since they define conflicting usage of GPIO PC6.

On initial powerup the 32.768 kHz crystal oscillator will take a little while to start stable oscillation. This only happens on initial powerup, not on wake-from-sleep, since the crystal usually stays running in deep sleep mode.

When ENABLE_OSC32K is defined the crystal oscillator is started as part of halInit(). After the crystal is started we delay for OSC32K_STARTUP_DELAY_MS (time in milliseconds). This delay allows the crystal oscillator to stabilize before we start using it for system timing.

If you set OSC32K_STARTUP_DELAY_MS to less than the crystal's startup time:

- The system timer won't produce a reliable one millisecond tick before the crystal is stable.

- You may see some number of ticks of unknown period occur before the crystal is stable.

- halInit() will complete and application code will begin running, but any events based on the system timer will not be accurate until the crystal is stable.

- An unstable system timer will only affect the APIs in system-timer.h.

Typical 32.768 kHz crystals measured by Ember take about 400 milliseconds to stabilize. Be sure to characterize your particular crystal's stabilization time since crystal behavior can vary.

- #define OSC32K_STARTUP_DELAY_MS

Packet Trace Configuration Defines

Provide the proper set of pin configuration for when the Packet Trace is enabled (look above for the define which enables it). When Packet Trace is not enabled, leave the two PTI pins in their default configuration. If Packet Trace is not being used, feel free to set the pin configurations as desired. The config shown here is simply the Power On Reset defaults.

- #define PWRUP_CFG_PTI_EN
- #define PWRUP_OUT_PTI_EN

- #define PWRDN_CFG_PTI_EN
- #define PWRDN_OUT_PTI_EN
- #define PWRUP_CFG_PTI_DATA
- #define PWRUP_OUT_PTI_DATA
- #define PWRDN_CFG_PTI_DATA
- #define PWRDN_OUT_PTI_DATA

32kHz Oscillator and nTX␣ACTIVE Configuration Defines

Since the 32kHz Oscillator and nTX_ACTIVE both share PC6, their configuration defines are linked and instantiated together. Look above for the defines that enable the 32kHz Oscillator and nTX_ACTIVE.

Note

ENABLE_OSC32K is mutually exclusive with ENABLE_ALT_FUNCTION_NTX_ACTIVE since they define conflicting usage of GPIO PC6.

When using the 32kHz, configure PC6 and PC7 for analog for the XTAL.

When using nTX_ACTIVE, configure PC6 for alternate output while awake and a low output when deep-sleeping. Also, configure PC7 for TEMP_EN.

When not using the 32kHz or nTX_ACTIVE, configure PC6 and PC7 for Button1 and TEMP_EN.

- #define PWRUP_CFG_BUTTON1
- #define PWRUP_OUT_BUTTON1
- #define PWRDN_CFG_BUTTON1
- #define PWRDN_OUT_BUTTON1
- #define CFG_TEMPEN

TX␣ACTIVE Configuration Defines

Provide the proper set of pin (PC5) configurations for when TX_ACTIVE is enabled (look above for the define which enables it). When TX_ACTIVE is not enabled, configure the pin for LED2.

- #define PWRUP_CFG_LED2
- #define PWRUP_OUT_LED2
- #define PWRDN_CFG_LED2
- #define PWRDN_OUT_LED2

GPIO Configuration Macros

These macros define the GPIO configuration and initial state of the output registers for all the GPIO in the powerup and powerdown modes.

- #define DEFINE_GPIO_RADIO_POWER_BOARD_MASK_VARIABLE()
- #define DEFINE_POWERUP_GPIO_CFG_VARIABLES()
- #define DEFINE_POWERUP_GPIO_OUTPUT_DATA_VARIABLES()
- #define DEFINE_POWERDOWN_GPIO_CFG_VARIABLES()
- #define DEFINE_POWERDOWN_GPIO_OUTPUT_DATA_VARIABLES()
- #define SET_POWERUP_GPIO_CFG_REGISTERS()

- #define SET_POWERUP_GPIO_OUTPUT_DATA_REGISTERS()
- #define SET_POWERDOWN_GPIO_CFG_REGISTERS()
- #define SET_POWERDOWN_GPIO_OUTPUT_DATA_REGISTERS()
- #define CONFIGURE_EXTERNAL_REGULATOR_ENABLE()
- int16u gpioCfgPowerUp [6]
- int16u gpioCfgPowerDown [6]
- int8u gpioOutPowerUp [3]
- int8u gpioOutPowerDown [3]
- int32u gpioRadioPowerBoardMask

GPIO Wake Source Definitions

A convenient define that chooses if this external signal can be used as source to wake from deep sleep. Any change in the state of the signal will wake up the CPU.

- #define WAKE_ON_PA0
- #define WAKE_ON_PA1
- #define WAKE_ON_PA2
- #define WAKE_ON_PA3
- #define WAKE_ON_PA4
- #define WAKE_ON_PA5
- #define WAKE_ON_PA6
- #define WAKE_ON_PA7
- #define WAKE_ON_PB0
- #define WAKE_ON_PB1
- #define WAKE_ON_PB2
- #define WAKE_ON_PB3
- #define WAKE_ON_PB4
- #define WAKE_ON_PB5
- #define WAKE_ON_PB6
- #define WAKE_ON_PB7
- #define WAKE_ON_PC0
- #define WAKE_ON_PC1
- #define WAKE_ON_PC2
- #define WAKE_ON_PC3
- #define WAKE_ON_PC4
- #define WAKE_ON_PC5
- #define WAKE_ON_PC6
- #define WAKE_ON_PC7

Board Specific Functions

The following macros exist to aid in the initialization, power up from sleep, and power down to sleep operations. These macros are responsible for either initializing directly, or calling initialization functions for any peripherals that are specific to this board implementation. These macros are called from halInit, halPowerDown, and halPowerUp respectively.

- #define halInternalInitBoard()
- #define halInternalPowerDownBoard()
- #define halInternalPowerUpBoard()

8.37.1   Detailed Description

See Sample Breakout Board Configuration for detailed documentation.

Definition in file dev0680.h.

8.37.2   Macro Definition Documentation

8.37.2.1   #define halInternalInitBoard(   )

Initialize the board. This function is called from ::halInit().

Definition at line 993 of file dev0680.h.

8.37.2.2   #define halInternalPowerDownBoard(   )

Power down the board. This function is called from ::halPowerDown().

Definition at line 1013 of file dev0680.h.

8.37.2.3   #define halInternalPowerUpBoard(   )

Power up the board. This function is called from ::halPowerUp().

Definition at line 1025 of file dev0680.h.

8.38   dev0680.h

```
00001
00046 #ifndef __BOARD_H__
00047 #define __BOARD_H__
00048
00065 #define EMBER_SERIAL_BAUD_CUSTOM  13
00066
00067
00088
00094 enum HalBoardLedPins {
00095   BOARDLED0 = PORTA_PIN(6),
00096   BOARDLED1 = PORTA_PIN(7),
00097   BOARDLED2 = PORTC_PIN(5),
00098   BOARDLED3 = BOARDLED2,
00099   BOARD_ACTIVITY_LED  = BOARDLED0,
00100   BOARD_HEARTBEAT_LED = BOARDLED1,
00101 };
00102
00124 #define BUTTON0             PORTB_PIN(6)
00125
00128 #define BUTTON0_IN          GPIO_PBIN
00129
00133 #define BUTTON0_SEL()       do { } while(0)
00134
00137 #define BUTTON0_ISR         halIrqBIsr
00138
00141 #define BUTTON0_INTCFG      GPIO_INTCFGB
00142
00145 #define BUTTON0_INT_EN_BIT  INT_IRQB
00146
00149 #define BUTTON0_FLAG_BIT    INT_IRQBFLAG
00150
00153 #define BUTTON0_MISS_BIT    INT_MISSIRQB
00154
00159 #define BUTTON1             PORTC_PIN(6)
00160
00163 #define BUTTON1_IN          GPIO_PCIN
```

```
00164
00168 #define BUTTON1_SEL()         do { GPIO_IRQCSEL = PORTC_PIN(6); } while(0)
00169
00172 #define BUTTON1_ISR           halIrqCIsr
00173
00176 #define BUTTON1_INTCFG        GPIO_INTCFGC
00177
00180 #define BUTTON1_INT_EN_BIT    INT_IRQC
00181
00184 #define BUTTON1_FLAG_BIT      INT_IRQCFLAG
00185
00188 #define BUTTON1_MISS_BIT      INT_MISSIRQC
00189
00190
00206
00211 //#define RADIO_HOLDOFF  // Configure Radio HoldOff at bootup
00212
00214
00227
00234 #define RHO_GPIO              PORTA_PIN(6)
00235
00238 #define RHO_ASSERTED          1
00239
00242 #define RHO_CFG               GPIO_PACFGH
00243
00246 #define RHO_IN                GPIO_PAIN
00247
00250 #define RHO_OUT               GPIO_PAOUT
00251
00255 #define RHO_SEL()             do { GPIO_IRQDSEL = RHO_GPIO; } while(0)
00256
00259 #define RHO_ISR               halIrqDIsr
00260
00263 #define RHO_INTCFG            GPIO_INTCFGD
00264
00267 #define RHO_INT_EN_BIT        INT_IRQD
00268
00271 #define RHO_FLAG_BIT          INT_IRQDFLAG
00272
00275 #define RHO_MISS_BIT          INT_MISSIRQD
00276
00279 #define PWRUP_CFG_LED_RHO_FOR_RHO    GPIOCFG_IN_PUD
00280 #define PWRUP_OUT_LED_RHO_FOR_RHO    GPIOOUT_PULLDOWN /* Deassert */
00281 #define PWRDN_CFG_LED_RHO_FOR_RHO    GPIOCFG_IN_PUD
00282 #define PWRDN_OUT_LED_RHO_FOR_RHO    GPIOOUT_PULLDOWN /* Deassert */
00283 #define WAKE_ON_LED_RHO_FOR_RHO      TRUE
00284
00287 #define PWRUP_CFG_LED_RHO_FOR_LED    GPIOCFG_OUT
00288 #define PWRUP_OUT_LED_RHO_FOR_LED    1 /* LED default off */
00289 #define PWRDN_CFG_LED_RHO_FOR_LED    GPIOCFG_OUT
00290 #define PWRDN_OUT_LED_RHO_FOR_LED    1 /* LED off */
00291 #define WAKE_ON_LED_RHO_FOR_LED      FALSE
00292
00296 #if    (defined(RADIO_HOLDOFF) && defined(RHO_GPIO))
00297   // Initial bootup configuration is for Radio HoldOff
00298   #define PWRUP_CFG_LED_RHO                PWRUP_CFG_LED_RHO_FOR_RHO
00299   #define PWRUP_OUT_LED_RHO                PWRUP_OUT_LED_RHO_FOR_RHO
00300   #define PWRDN_CFG_LED_RHO                PWRDN_CFG_LED_RHO_FOR_RHO
00301   #define PWRDN_OUT_LED_RHO                PWRDN_OUT_LED_RHO_FOR_RHO
00302   #define WAKE_ON_LED_RHO                  WAKE_ON_LED_RHO_FOR_RHO
00303   #define halInternalInitRadioHoldOff() halSetRadioHoldOff(TRUE)
00304 #else
00305   // Initial bootup configuration is for LED
00306   #define PWRUP_CFG_LED_RHO                PWRUP_CFG_LED_RHO_FOR_LED
00307   #define PWRUP_OUT_LED_RHO                PWRUP_OUT_LED_RHO_FOR_LED
00308   #define PWRDN_CFG_LED_RHO                PWRDN_CFG_LED_RHO_FOR_LED
00309   #define PWRDN_OUT_LED_RHO                PWRDN_OUT_LED_RHO_FOR_LED
00310   #define WAKE_ON_LED_RHO                  WAKE_ON_LED_RHO_FOR_LED
00311   #define halInternalInitRadioHoldOff() /* no-op */
00312 #endif//(defined(RADIO_HOLDOFF) && defined(RHO_GPIO))
00313
00314 #ifdef  RHO_GPIO
00315   #define WAKE_ON_LED_RHO_VAR         halInternalWakeOnLedOrRho
00316   extern int8u WAKE_ON_LED_RHO_VAR;
00317   #define ADJUST_GPIO_CONFIG_LED_RHO(enableRadioHoldOff)  do {                 \
00318           ATOMIC( /* Must read-modify-write so to be safe, use ATOMIC() */    \
00319             if (enableRadioHoldOff) { /* Radio HoldOff */                      \
00320               /* Actual register state */                                      \
00321               /*halGpioConfig(RHO_CFG, PWRUP_CFG_LED_RHO_FOR_RHO);*/           \
00322              RHO_CFG = RHO_CFG                                                  \
```

```
00323                      & ~(0x000F                     << ((RHO_GPIO&3)*4))              \
00324                      | (PWRUP_CFG_LED_RHO_FOR_RHO << ((RHO_GPIO&3)*4));               \
00325                  RHO_OUT = RHO_OUT                                                    \
00326                      & ~(0x0001                     << ((RHO_GPIO&7)  ))              \
00327                      | (PWRUP_OUT_LED_RHO_FOR_RHO << ((RHO_GPIO&7)  ));               \
00328                  /* Shadow register state */                                         \
00329                  gpioCfgPowerUp[RHO_GPIO>>2] = gpioCfgPowerUp[RHO_GPIO>>2]            \
00330                      & ~(0x000F                     << ((RHO_GPIO&3)*4))              \
00331                      | (PWRUP_CFG_LED_RHO_FOR_RHO << ((RHO_GPIO&3)*4));               \
00332                  gpioOutPowerUp[RHO_GPIO>>3] = gpioOutPowerUp[RHO_GPIO>>3]            \
00333                      & ~(0x0001                     << ((RHO_GPIO&7)  ))              \
00334                      | (PWRUP_OUT_LED_RHO_FOR_RHO << ((RHO_GPIO&7)  ));               \
00335                  gpioCfgPowerDown[RHO_GPIO>>2] = gpioCfgPowerDown[RHO_GPIO>>2]        \
00336                      & ~(0x000F                     << ((RHO_GPIO&3)*4))              \
00337                      | (PWRDN_CFG_LED_RHO_FOR_RHO << ((RHO_GPIO&3)*4));               \
00338                  gpioOutPowerDown[RHO_GPIO>>3] = gpioOutPowerDown[RHO_GPIO>>3]        \
00339                      & ~(0x0001                     << ((RHO_GPIO&7)  ))              \
00340                      | (PWRDN_OUT_LED_RHO_FOR_RHO << ((RHO_GPIO&7)  ));               \
00341                  WAKE_ON_LED_RHO_VAR = WAKE_ON_LED_RHO_FOR_RHO;                       \
00342                } else { /* LED */                                                    \
00343                  /* Actual register state */                                         \
00344                  /*halGpioConfig(RHO_CFG, PWRUP_CFG_LED_RHO_FOR_LED);*/               \
00345                  RHO_CFG = RHO_CFG                                                    \
00346                      & ~(0x000F                     << ((RHO_GPIO&3)*4))              \
00347                      | (PWRUP_CFG_LED_RHO_FOR_LED << ((RHO_GPIO&3)*4));               \
00348                  RHO_OUT = RHO_OUT                                                    \
00349                      & ~(0x0001                     << ((RHO_GPIO&7)  ))              \
00350                      | (PWRUP_OUT_LED_RHO_FOR_LED << ((RHO_GPIO&7)  ));               \
00351                  /* Shadow register state */                                         \
00352                  gpioCfgPowerUp[RHO_GPIO>>2] = gpioCfgPowerUp[RHO_GPIO>>2]            \
00353                      & ~(0x000F                     << ((RHO_GPIO&3)*4))              \
00354                      | (PWRUP_CFG_LED_RHO_FOR_LED << ((RHO_GPIO&3)*4));               \
00355                  gpioOutPowerUp[RHO_GPIO>>3] = gpioOutPowerUp[RHO_GPIO>>3]            \
00356                      & ~(0x0001                     << ((RHO_GPIO&7)  ))              \
00357                      | (PWRUP_OUT_LED_RHO_FOR_LED << ((RHO_GPIO&7)  ));               \
00358                  gpioCfgPowerDown[RHO_GPIO>>2] = gpioCfgPowerDown[RHO_GPIO>>2]        \
00359                      & ~(0x000F                     << ((RHO_GPIO&3)*4))              \
00360                      | (PWRDN_CFG_LED_RHO_FOR_LED << ((RHO_GPIO&3)*4));               \
00361                  gpioOutPowerDown[RHO_GPIO>>3] = gpioOutPowerDown[RHO_GPIO>>3]        \
00362                      & ~(0x0001                     << ((RHO_GPIO&7)  ))              \
00363                      | (PWRDN_OUT_LED_RHO_FOR_LED << ((RHO_GPIO&7)  ));               \
00364                  WAKE_ON_LED_RHO_VAR = WAKE_ON_LED_RHO_FOR_LED;                       \
00365                }                                                                     \
00366            )} while (0)
00367 #else
00368    #define WAKE_ON_LED_RHO_VAR          WAKE_ON_LED_RHO_FOR_LED
00369 #endif//RHO_GPIO
00370
00372
00385 #define TEMP_SENSOR_ADC_CHANNEL ADC_SOURCE_ADC0_VREF2
00386
00389 #define TEMP_SENSOR_SCALE_FACTOR 1
00390
00405 #define PACKET_TRACE  // We do have PACKET_TRACE support
00406
00407
00443 #define OSC32K_STARTUP_DELAY_MS (0)
00444
00445 #if OSC32K_STARTUP_DELAY_MS > MAX_INT16U_VALUE
00446 #error "OSC32K_STARTUP_DELAY_MS must fit in 16 bits."
00447 #endif
00448
00455 //#define ENABLE_OSC32K  // Enable 32.768 kHz osc instead of 1 kHz RC osc
00457
00476 //#define DISABLE_SLEEPTMR_DEEPSLEEP  // Disable internal 10kHz SlowRC
00478
00494 //#define DISABLE_OSC24M_BIAS_TRIM  // Disable 24 MHz bias trim adjustment
00496
00497
00514 //#define ENABLE_ALT_FUNCTION_REG_EN
00516
00517
00533 //#define ENABLE_ALT_FUNCTION_TX_ACTIVE
00535
00536
00555 //#define ENABLE_ALT_FUNCTION_NTX_ACTIVE
00557
00573 //#define EEPROM_USES_SHUTDOWN_CONTROL
00575
00576
```

```
00588
00589
00602 #ifdef  PACKET_TRACE
00603   #define PWRUP_CFG_PTI_EN     GPIOCFG_OUT_ALT
00604   #define PWRUP_OUT_PTI_EN     0
00605   #define PWRDN_CFG_PTI_EN     GPIOCFG_IN_PUD
00606   #define PWRDN_OUT_PTI_EN     GPIOOUT_PULLDOWN
00607   #define PWRUP_CFG_PTI_DATA   GPIOCFG_OUT_ALT
00608   #define PWRUP_OUT_PTI_DATA   1
00609   #define PWRDN_CFG_PTI_DATA   GPIOCFG_IN_PUD
00610   #define PWRDN_OUT_PTI_DATA   GPIOOUT_PULLUP
00611 #else
00612   #define PWRUP_CFG_PTI_EN     GPIOCFG_IN
00613   #define PWRUP_OUT_PTI_EN     0
00614   #define PWRDN_CFG_PTI_EN     GPIOCFG_IN
00615   #define PWRDN_OUT_PTI_EN     0
00616   #define PWRUP_CFG_PTI_DATA   GPIOCFG_IN
00617   #define PWRUP_OUT_PTI_DATA   0
00618   #define PWRDN_CFG_PTI_DATA   GPIOCFG_IN
00619   #define PWRDN_OUT_PTI_DATA   0
00620 #endif//PACKET_TRACE
00621
00622
00623
00646 #if defined(ENABLE_OSC32K) && defined(ENABLE_ALT_FUNCTION_NTX_ACTIVE)
00647   //Oops!  Only one of these can be used at a time!
00648   #error ENABLE_OSC32K and ENABLE_ALT_FUNCTION_NTX_ACTIVE are mutually\
00649  exclusive.  They define conflicting usage for GPIO PC6.
00650
00651 #elif defined(ENABLE_OSC32K) && !defined(ENABLE_ALT_FUNCTION_NTX_ACTIVE)
00652   //Use OCS32K configuration
00653   #define PWRUP_CFG_BUTTON1  GPIOCFG_ANALOG
00654   #define PWRUP_OUT_BUTTON1  0
00655   #define PWRDN_CFG_BUTTON1  GPIOCFG_ANALOG
00656   #define PWRDN_OUT_BUTTON1  0
00657
00658 #elif !defined(ENABLE_OSC32K) && defined(ENABLE_ALT_FUNCTION_NTX_ACTIVE)
00659   //Use nTX_ACTIVE configuration
00660   #define PWRUP_CFG_BUTTON1  GPIOCFG_OUT_ALT
00661   #define PWRUP_OUT_BUTTON1  0
00662   #define PWRDN_CFG_BUTTON1  GPIOCFG_OUT
00663   #define PWRDN_OUT_BUTTON1  0
00664
00665 #else
00666   //Use Button1 configuration
00667   #define PWRUP_CFG_BUTTON1  GPIOCFG_IN_PUD
00668   #define PWRUP_OUT_BUTTON1  GPIOOUT_PULLUP /* Button needs a pullup */
00669   #define PWRDN_CFG_BUTTON1  GPIOCFG_IN_PUD
00670   #define PWRDN_OUT_BUTTON1  GPIOOUT_PULLUP /* Button needs a pullup */
00671
00672 #endif
00673
00677 #ifdef  ENABLE_OSC32K
00678   #define CFG_TEMPEN          GPIOCFG_ANALOG
00679 #else
00680   #define CFG_TEMPEN          GPIOCFG_OUT
00681 #endif//ENABLE_OSC32K
00682
00683
00684
00695 #ifdef  ENABLE_ALT_FUNCTION_TX_ACTIVE
00696   #define PWRUP_CFG_LED2  GPIOCFG_OUT_ALT
00697   #define PWRUP_OUT_LED2  0
00698   #define PWRDN_CFG_LED2  GPIOCFG_OUT
00699   #define PWRDN_OUT_LED2  0
00700 #else
00701   #define PWRUP_CFG_LED2  GPIOCFG_OUT
00702   #define PWRUP_OUT_LED2  1  /* LED default off */
00703   #define PWRDN_CFG_LED2  GPIOCFG_OUT
00704   #define PWRDN_OUT_LED2  1  /* LED default off */
00705 #endif//ENABLE_ALT_FUNCTION_TX_ACTIVE
00706
00707
00708
00717 //Each pin has 4 cfg bits.  There are 3 ports with 2 cfg registers per
00718 //port since the cfg register only holds 2 pins (16bits).  Therefore,
00719 //the cfg arrays need to be 6 entries of 16bits.
00720 extern int16u gpioCfgPowerUp[6];
00721 extern int16u gpioCfgPowerDown[6];
00722 //Each pin has 1 out bit.  There are 3 ports with 1 out register per
```

```
00723 //port (8bits).  Therefore, the out arrays need to be 3 entries of 8bits.
00724 extern int8u gpioOutPowerUp[3];
00725 extern int8u gpioOutPowerDown[3];
00726 //A single mask variable covers all 24 GPIO.
00727 extern int32u gpioRadioPowerBoardMask;
00728
00729
00736 #define DEFINE_GPIO_RADIO_POWER_BOARD_MASK_VARIABLE() \
00737 int32u gpioRadioPowerBoardMask = 0
00738
00739
00743 #define DEFINE_POWERUP_GPIO_CFG_VARIABLES()                            \
00744 int16u gpioCfgPowerUp[6] = {                                          \
00745                         ((GPIOCFG_OUT_ALT    <<PA0_CFG_BIT)|          \
00746                          (GPIOCFG_IN         <<PA1_CFG_BIT)|          \
00747                          (GPIOCFG_OUT_ALT    <<PA2_CFG_BIT)|          \
00748                          (GPIOCFG_OUT        <<PA3_CFG_BIT)),         \
00749                         ((PWRUP_CFG_PTI_EN   <<PA4_CFG_BIT)|          \
00750                          (PWRUP_CFG_PTI_DATA <<PA5_CFG_BIT)|          \
00751                          (PWRUP_CFG_LED_RHO  <<PA6_CFG_BIT)|          \
00752                          (GPIOCFG_OUT        <<PA7_CFG_BIT)),         \
00753                         ((GPIOCFG_OUT        <<PB0_CFG_BIT)|          \
00754                          (GPIOCFG_OUT_ALT    <<PB1_CFG_BIT)| /* SC1TXD  */\
00755                          (GPIOCFG_IN_PUD     <<PB2_CFG_BIT)| /* SC1RXD  */\
00756                          (GPIOCFG_IN_PUD     <<PB3_CFG_BIT)),/* SC1nCTS */\
00757                         ((GPIOCFG_OUT_ALT    <<PB4_CFG_BIT)| /* SC1nRTS */\
00758                          (GPIOCFG_ANALOG     <<PB5_CFG_BIT)|          \
00759                          (GPIOCFG_IN_PUD     <<PB6_CFG_BIT)|          \
00760                          (GPIOCFG_OUT_ALT    <<PB7_CFG_BIT)),         \
00761                         ((GPIOCFG_OUT_ALT    <<PC0_CFG_BIT)|          \
00762                          (GPIOCFG_OUT        <<PC1_CFG_BIT)|          \
00763                          (GPIOCFG_OUT_ALT    <<PC2_CFG_BIT)|          \
00764                          (GPIOCFG_IN         <<PC3_CFG_BIT)),         \
00765                         ((GPIOCFG_IN         <<PC4_CFG_BIT)|          \
00766                          (PWRUP_CFG_LED2     <<PC5_CFG_BIT)|          \
00767                          (PWRUP_CFG_BUTTON1  <<PC6_CFG_BIT)|          \
00768                          (CFG_TEMPEN         <<PC7_CFG_BIT))          \
00769                        }
00770
00771
00775 #define DEFINE_POWERUP_GPIO_OUTPUT_DATA_VARIABLES()                    \
00776 int8u gpioOutPowerUp[3] = {                                           \
00777                         ((0                  <<PA0_BIT)|              \
00778                          (0                  <<PA1_BIT)|              \
00779                          (0                  <<PA2_BIT)|              \
00780                         /* nSSEL is default idle high */             \
00781                          (1                  <<PA3_BIT)|              \
00782                          (PWRUP_OUT_PTI_EN   <<PA4_BIT)|              \
00783                          (PWRUP_OUT_PTI_DATA <<PA5_BIT)|              \
00784                          (PWRUP_OUT_LED_RHO  <<PA6_BIT)|              \
00785                         /* LED default off */                        \
00786                          (1                  <<PA7_BIT)),             \
00787                         ((1                  <<PB0_BIT)|              \
00788                          (1                  <<PB1_BIT)|  /* SC1TXD  */\
00789                          (1                  <<PB2_BIT)|  /* SC1RXD  */\
00790                          (1                  <<PB3_BIT)|  /* SC1nCTS */\
00791                          (0                  <<PB4_BIT)|  /* SC1nRTS */\
00792                          (0                  <<PB5_BIT)|              \
00793                         /* PB6 has button needing a pullup */        \
00794                          (GPIOOUT_PULLUP     <<PB6_BIT)|              \
00795                          (0                  <<PB7_BIT)),             \
00796                         ((1                  <<PC0_BIT)|              \
00797                          (0                  <<PC1_BIT)|              \
00798                          (1                  <<PC2_BIT)|              \
00799                          (0                  <<PC3_BIT)|              \
00800                          (0                  <<PC4_BIT)|              \
00801                          (PWRUP_OUT_LED2     <<PC5_BIT)|              \
00802                          (PWRUP_OUT_BUTTON1  <<PC6_BIT)|              \
00803                         /* Temp Sensor default on */                 \
00804                          (1                  <<PC7_BIT))              \
00805                        }
00806
00807
00811 #define DEFINE_POWERDOWN_GPIO_CFG_VARIABLES()                         \
00812 int16u gpioCfgPowerDown[6] = {                                        \
00813                         ((GPIOCFG_IN_PUD     <<PA0_CFG_BIT)|          \
00814                          (GPIOCFG_IN_PUD     <<PA1_CFG_BIT)|
```

```
                                                 \
00815                              (GPIOCFG_IN_PUD      <<PA2_CFG_BIT)|
       \
00816                              (GPIOCFG_OUT         <<PA3_CFG_BIT)),
       \
00817                             ((PWRDN_CFG_PTI_EN    <<PA4_CFG_BIT)|
       \
00818                              (PWRDN_CFG_PTI_DATA <<PA5_CFG_BIT)|
       \
00819                              (PWRDN_CFG_LED_RHO  <<PA6_CFG_BIT)|
       \
00820                              (GPIOCFG_OUT         <<PA7_CFG_BIT)),
       \
00821                             ((GPIOCFG_OUT         <<PB0_CFG_BIT)|
       \
00822                              (GPIOCFG_OUT         <<PB1_CFG_BIT)| /* SC1TXD
  */\
00823                              (GPIOCFG_IN_PUD      <<PB2_CFG_BIT)| /* SC1RXD
  */\
00824                              (GPIOCFG_IN_PUD      <<PB3_CFG_BIT)),/* SC1nCTS
  */\
00825                             ((GPIOCFG_OUT         <<PB4_CFG_BIT)| /* SC1nRTS
  */\
00826                              /* disable analog for sleep */
       \
00827                              (GPIOCFG_IN_PUD      <<PB5_CFG_BIT)|
       \
00828                              (GPIOCFG_IN_PUD      <<PB6_CFG_BIT)|
       \
00829                              /* need to use pulldown for sleep */
       \
00830                              (GPIOCFG_IN_PUD      <<PB7_CFG_BIT)),
       \
00831                             ((GPIOCFG_IN_PUD      <<PC0_CFG_BIT)|
       \
00832                              (GPIOCFG_OUT         <<PC1_CFG_BIT)|
       \
00833                              (GPIOCFG_OUT         <<PC2_CFG_BIT)|
       \
00834                              (GPIOCFG_IN_PUD      <<PC3_CFG_BIT)),
       \
00835                             ((GPIOCFG_IN_PUD      <<PC4_CFG_BIT)|
       \
00836                              (PWRDN_CFG_LED2      <<PC5_CFG_BIT)|
       \
00837                              (PWRDN_CFG_BUTTON1  <<PC6_CFG_BIT)|
       \
00838                              (CFG_TEMPEN          <<PC7_CFG_BIT))
       \
00839                                 }
00840
00841
00845 #define DEFINE_POWERDOWN_GPIO_OUTPUT_DATA_VARIABLES()                     \
00846 int8u gpioOutPowerDown[3] = {                                           \
00847                             ((GPIOOUT_PULLUP      <<PA0_BIT)|            \
00848                              (GPIOOUT_PULLUP      <<PA1_BIT)|            \
00849                              (GPIOOUT_PULLUP      <<PA2_BIT)|            \
00850                              /* nSSEL is idle high */                   \
00851                              (1                   <<PA3_BIT)|            \
00852                              /* enable is idle low */                   \
00853                              (PWRDN_OUT_PTI_EN    <<PA4_BIT)|            \
00854                              /* data is idle high */                    \
00855                              (PWRDN_OUT_PTI_DATA <<PA5_BIT)|             \
00856                              (PWRDN_OUT_LED_RHO  <<PA6_BIT)|             \
00857                              /* LED off */                              \
00858                              (1                   <<PA7_BIT)),           \
00859                             ((0                   <<PB0_BIT)|            \
00860                              (GPIOOUT_PULLUP      <<PB1_BIT)|  /* SC1TXD  */\
00861                              (GPIOOUT_PULLUP      <<PB2_BIT)|  /* SC1RXD  */\
00862                              (GPIOOUT_PULLDOWN    <<PB3_BIT)|  /* SC1nCTS */\
00863                              (GPIOOUT_PULLUP      <<PB4_BIT)|  /* SC1nRTS */\
00864                              /* tempsense needs pulldown */             \
00865                              (GPIOOUT_PULLDOWN   <<PB5_BIT)|             \
00866                              /* PB6 has button needing a pullup */      \
00867                              (GPIOOUT_PULLUP     <<PB6_BIT)|             \
00868                              /* buzzer needs pulldown for sleep */      \
00869                              (GPIOOUT_PULLDOWN  <<PB7_BIT)),             \
00870                             ((GPIOOUT_PULLUP     <<PC0_BIT)|             \
00871                              (0                   <<PC1_BIT)|            \
00872                              (1                   <<PC2_BIT)|            \
```

```
00873                                 (GPIOOUT_PULLDOWN   <<PC3_BIT)|                  \
00874                                 (GPIOOUT_PULLDOWN   <<PC4_BIT)|                  \
00875                                 (PWRDN_OUT_LED2     <<PC5_BIT)|                  \
00876                                 (PWRDN_OUT_BUTTON1  <<PC6_BIT)|                  \
00877                                 /* Temp Sensor off */                           \
00878                                 (0                  <<PC7_BIT))                  \
00879                               }
00880
00881
00885 #define SET_POWERUP_GPIO_CFG_REGISTERS() \
00886   GPIO_PACFGL = gpioCfgPowerUp[0];        \
00887   GPIO_PACFGH = gpioCfgPowerUp[1];        \
00888   GPIO_PBCFGL = gpioCfgPowerUp[2];        \
00889   GPIO_PBCFGH = gpioCfgPowerUp[3];        \
00890   GPIO_PCCFGL = gpioCfgPowerUp[4];        \
00891   GPIO_PCCFGH = gpioCfgPowerUp[5];
00892
00893
00897 #define SET_POWERUP_GPIO_OUTPUT_DATA_REGISTERS() \
00898   GPIO_PAOUT = gpioOutPowerUp[0];                 \
00899   GPIO_PBOUT = gpioOutPowerUp[1];                 \
00900   GPIO_PCOUT = gpioOutPowerUp[2];
00901
00902
00906 #define SET_POWERDOWN_GPIO_CFG_REGISTERS() \
00907   GPIO_PACFGL = gpioCfgPowerDown[0];        \
00908   GPIO_PACFGH = gpioCfgPowerDown[1];        \
00909   GPIO_PBCFGL = gpioCfgPowerDown[2];        \
00910   GPIO_PBCFGH = gpioCfgPowerDown[3];        \
00911   GPIO_PCCFGL = gpioCfgPowerDown[4];        \
00912   GPIO_PCCFGH = gpioCfgPowerDown[5];
00913
00914
00918 #define SET_POWERDOWN_GPIO_OUTPUT_DATA_REGISTERS() \
00919   GPIO_PAOUT = gpioOutPowerDown[0];                 \
00920   GPIO_PBOUT = gpioOutPowerDown[1];                 \
00921   GPIO_PCOUT = gpioOutPowerDown[2];
00922
00923
00924
00928 #ifdef ENABLE_ALT_FUNCTION_REG_EN
00929   #define CONFIGURE_EXTERNAL_REGULATOR_ENABLE()  GPIO_DBGCFG |= GPIO_EXTREGEN;
00930 #else
00931   #define CONFIGURE_EXTERNAL_REGULATOR_ENABLE()  GPIO_DBGCFG &= ~GPIO_EXTREGEN;
00932 #endif
00933
00934
00935
00946 #define WAKE_ON_PA0   FALSE
00947 #define WAKE_ON_PA1   FALSE
00948 #define WAKE_ON_PA2   FALSE
00949 #define WAKE_ON_PA3   FALSE
00950 #define WAKE_ON_PA4   FALSE
00951 #define WAKE_ON_PA5   FALSE
00952 #define WAKE_ON_PA6   WAKE_ON_LED_RHO_VAR
00953 #define WAKE_ON_PA7   FALSE
00954 #define WAKE_ON_PB0   FALSE
00955 #define WAKE_ON_PB1   FALSE
00956 #ifdef SLEEPY_EZSP_UART  // SC1RXD
00957   #define WAKE_ON_PB2   TRUE
00958 #else
00959   #define WAKE_ON_PB2   FALSE
00960 #endif
00961 #define WAKE_ON_PB3   FALSE
00962 #define WAKE_ON_PB4   FALSE
00963 #define WAKE_ON_PB5   FALSE
00964 #define WAKE_ON_PB6   TRUE   //BUTTON0
00965 #define WAKE_ON_PB7   FALSE
00966 #define WAKE_ON_PC0   FALSE
00967 #define WAKE_ON_PC1   FALSE
00968 #define WAKE_ON_PC2   FALSE
00969 #define WAKE_ON_PC3   FALSE
00970 #define WAKE_ON_PC4   FALSE
00971 #define WAKE_ON_PC5   FALSE
00972 #define WAKE_ON_PC6   TRUE   //BUTTON1
00973 #define WAKE_ON_PC7   FALSE
00974
00975
00976
00978
```

```
00979
00992 #ifndef EZSP_UART
00993   #define halInternalInitBoard()                                    \
00994           do {                                                       \
00995             halInternalPowerUpBoard();                               \
00996             halInternalInitRadioHoldOff();                           \
00997             halInternalRestartUart();                                \
00998             halInternalInitButton();                                 \
00999           } while(0)
01000 #else
01001   #define halInternalInitBoard()                                    \
01002           do {                                                       \
01003             halInternalPowerUpBoard();                               \
01004             halInternalInitRadioHoldOff();                           \
01005             halInternalRestartUart();                                \
01006           } while(0)
01007 #endif
01008
01013 #define halInternalPowerDownBoard()                     \
01014           do {                                          \
01015             /* Board peripheral deactivation */         \
01016             /* halInternalSleepAdc(); */                \
01017             SET_POWERDOWN_GPIO_OUTPUT_DATA_REGISTERS()  \
01018             SET_POWERDOWN_GPIO_CFG_REGISTERS()          \
01019           } while(0)
01020
01025 #define halInternalPowerUpBoard()                                 \
01026           do {                                                    \
01027             SET_POWERUP_GPIO_OUTPUT_DATA_REGISTERS()              \
01028             SET_POWERUP_GPIO_CFG_REGISTERS()                      \
01029             /*The radio GPIO should remain in the powerdown state */ \
01030             /*until the stack specifically powers them up. */     \
01031             halStackRadioPowerDownBoard();                        \
01032             CONFIGURE_EXTERNAL_REGULATOR_ENABLE()                 \
01033             /* Board peripheral reactivation */                  \
01034             halInternalInitAdc();                                 \
01035           } while(0)
01036
01037
01038 #endif //__BOARD_H__
01039
```

8.39 diagnostic.h File Reference

Macros

- #define halResetWasCrash()

Functions

- int32u halGetMainStackBytesUsed (void)
- void halPrintCrashSummary (int8u port)
- void halPrintCrashDetails (int8u port)
- void halPrintCrashData (int8u port)

8.39.1 Detailed Description

See Crash and Watchdog Diagnostics for detailed documentation.

Definition in file diagnostic.h.

8.40 diagnostic.h

```
00001
00014 #ifndef __EM3XX_DIAGNOSTIC_H__
00015 #define __EM3XX_DIAGNOSTIC_H__
00016
00017 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00018
00019 // Define the reset reasons that should print out detailed crash data.
00020 #define RESET_CRASH_REASON_MASK ( (1 << RESET_UNKNOWN) |    \
00021                                   (1 << RESET_WATCHDOG) |   \
00022                                   (1 << RESET_CRASH) |      \
00023                                   (1 << RESET_FLASH) |      \
00024                                   (1 << RESET_FAULT) |      \
00025                                   (1 << RESET_FATAL) )
00026
00027 typedef struct
00028 {
00029   // first two fields must be the same as HalCrashInfoType
00030   int16u resetReason;       // reason written out just before forcing a
        reset
00031   int16u resetSignature;
00032   int16u panId;         // PanId that the bootloader will use
00033   int8u  radioChannel;  // emberGetRadioChannel() - 11
00034   int8u  radioPower;    // emberGetRadioPower()
00035   int8u  radioLnaCal;   // Low Noise Amplifier calibration
00036   int8u  reserved[22]; // (reserved for future use)
00037 } HalBootParamType;
00038
00039 // note that assertInfo and dmaProt are written just before a forced reboot
00040 typedef union
00041 {
00042   struct { PGM_P file; int line; } assertInfo;
00043   struct { int32u channel; int32u address; } dmaProt;
00044 } HalCrashSpecificDataType;
00045
00046 // Define crash registers as structs so a debugger can display their bit fields
00047 typedef union {
00048   struct
00049   {
00050     int32u EXCPT          : 9;  // B0-8
00051     int32u ICIIT_LOW      : 7;  // B9-15
00052     int32u                : 8;  // B16-23
00053     int32u T              : 1;  // B24
00054     int32u ICIIT_HIGH     : 2;  // B25-26
00055     int32u Q              : 1;  // B27
00056     int32u V              : 1;  // B28
00057     int32u C              : 1;  // B29
00058     int32u Z              : 1;  // B30
00059     int32u N              : 1;  // B31
00060   } bits;
00061   int32u word;
00062 } HalCrashxPsrType;
00063
00064 typedef union {
00065   struct
00066   {
00067     int32u VECTACTIVE     : 9;  // B0-8
00068     int32u                : 2;  // B9-10
00069     int32u RETTOBASE      : 1;  // B11
00070     int32u VECTPENDING    : 9;  // B12-20
00071     int32u                : 1;  // B21
00072     int32u ISRPENDING     : 1;  // B22
00073     int32u ISRPREEMPT     : 1;  // B23
00074     int32u                : 1;  // B24
00075     int32u PENDSTCLR      : 1;  // B25
00076     int32u PENDSTSET      : 1;  // B26
00077     int32u PENDSVCLR      : 1;  // B27
00078     int32u PENDSVSET      : 1;  // B28
00079     int32u                : 2;  // B29-30
00080     int32u NMIPENDSET     : 1;  // B31
00081   } bits;
00082   int32u word;
00083 } HalCrashIcsrType;
00084
00085 typedef union {
00086   struct
00087   {
```

```
00088      int32u Timer1        : 1;  // B0
00089      int32u Timer2        : 1;  // B1
00090      int32u Management    : 1;  // B2
00091      int32u Baseband      : 1;  // B3
00092      int32u Sleep_Timer   : 1;  // B4
00093      int32u SC1           : 1;  // B5
00094      int32u SC2           : 1;  // B6
00095      int32u Security      : 1;  // B7
00096      int32u MAC_Timer     : 1;  // B8
00097      int32u MAC_TX        : 1;  // B9
00098      int32u MAC_RX        : 1;  // B10
00099      int32u ADC           : 1;  // B11
00100      int32u IRQ_A         : 1;  // B12
00101      int32u IRQ_B         : 1;  // B13
00102      int32u IRQ_C         : 1;  // B14
00103      int32u IRQ_D         : 1;  // B15
00104      int32u Debug         : 1;  // B16
00105      int32u               : 15; // B17-31
00106    } bits;
00107    int32u word;
00108 } HalCrashIntActiveType;
00109
00110 typedef union {
00111    struct
00112    {
00113      int32u MEMFAULTACT    : 1;  // B0
00114      int32u BUSFAULTACT    : 1;  // B1
00115      int32u               : 1;  // B2
00116      int32u USGFAULTACT    : 1;  // B3
00117      int32u               : 3;  // B4-6
00118      int32u SVCALLACT      : 1;  // B7
00119      int32u MONITORACT     : 1;  // B8
00120      int32u               : 1;  // B9
00121      int32u PENDSVACT      : 1;  // B10
00122      int32u SYSTICKACT     : 1;  // B11
00123      int32u USGFAULTPENDED : 1;  // B12
00124      int32u MEMFAULTPENDED : 1;  // B13
00125      int32u BUSFAULTPENDED : 1;  // B14
00126      int32u SVCALLPENDED   : 1;  // B15
00127      int32u MEMFAULTENA    : 1;  // B16
00128      int32u BUSFAULTENA    : 1;  // B17
00129      int32u USGFAULTENA    : 1;  // B18
00130      int32u               : 13; // B19-31
00131    } bits;
00132    int32u word;
00133 } HalCrashShcsrType;
00134
00135 typedef union {
00136    struct
00137    {
00138      int32u IACCVIOL       : 1;  // B0
00139      int32u DACCVIOL       : 1;  // B1
00140      int32u               : 1;  // B2
00141      int32u MUNSTKERR      : 1;  // B3
00142      int32u MSTKERR        : 1;  // B4
00143      int32u               : 2;  // B5-6
00144      int32u MMARVALID      : 1;  // B7
00145      int32u IBUSERR        : 1;  // B8
00146      int32u PRECISERR      : 1;  // B9
00147      int32u IMPRECISERR    : 1;  // B10
00148      int32u UNSTKERR       : 1;  // B11
00149      int32u STKERR         : 1;  // B12
00150      int32u               : 2;  // B13-14
00151      int32u BFARVALID      : 1;  // B15
00152      int32u UNDEFINSTR     : 1;  // B16
00153      int32u INVSTATE       : 1;  // B17
00154      int32u INVPC          : 1;  // B18
00155      int32u NOCP           : 1;  // B19
00156      int32u               : 4;  // B20-23
00157      int32u UNALIGNED      : 1;  // B24
00158      int32u DIVBYZERO      : 1;  // B25
00159      int32u               : 6;  // B26-31
00160    } bits;
00161    int32u word;
00162 } HalCrashCfsrType;
00163
00164 typedef union {
00165    struct
00166    {
00167      int32u               : 1;  // B0
```

```
00168     int32u VECTTBL        : 1;   // B1
00169     int32u                : 28; // B2-29
00170     int32u FORCED         : 1;   // B30
00171     int32u DEBUGEVT       : 1;   // B31
00172   } bits;
00173   int32u word;
00174 } HalCrashHfsrType;
00175
00176 typedef union {
00177   struct
00178   {
00179     int32u HALTED         : 1;   // B0
00180     int32u BKPT           : 1;   // B1
00181     int32u DWTTRAP        : 1;   // B2
00182     int32u VCATCH         : 1;   // B3
00183     int32u EXTERNAL       : 1;   // B4
00184     int32u                : 27; // B5-31
00185   } bits;
00186   int32u word;
00187 } HalCrashDfsrType;
00188
00189 typedef union {
00190   struct
00191   {
00192     int32u MISSED         : 1;   // B0
00193     int32u RESERVED       : 1;   // B1
00194     int32u PROTECTED      : 1;   // B2
00195     int32u WRONGSIZE      : 1;   // B3
00196     int32u                : 28; // B4-31
00197   } bits;
00198   int32u word;
00199 } HalCrashAfsrType;
00200
00201 #define NUM_RETURNS     6
00202
00203 // Define the crash data structure
00204 typedef struct
00205 {
00206   //
        ****************************************************************************
00207   // The components within this first block are written by the assembly
00208   // language common fault handler, and position and order is critical.
00209   // cstartup-iar-boot-entry.s79 also relies on the position/order here.
00210   // Do not edit without also modifying that code.
00211   //
        ****************************************************************************
00212   int16u resetReason;   // reason written out just before forcing a
        reset
00213   int16u resetSignature;
00214   int32u R0;            // processor registers
00215   int32u R1;
00216   int32u R2;
00217   int32u R3;
00218   int32u R4;
00219   int32u R5;
00220   int32u R6;
00221   int32u R7;
00222   int32u R8;
00223   int32u R9;
00224   int32u R10;
00225   int32u R11;
00226   int32u R12;
00227   int32u LR;
00228   int32u mainSP;        // main and process stack pointers
00229   int32u processSP;
00230   //
        ****************************************************************************
00231   // End of the block written by the common fault handler.
00232   //
        ****************************************************************************
00233
00234   int32u PC;               // stacked return value (if it could be read)
00235   HalCrashxPsrType xPSR;   // stacked processor status reg (if it could be read)
00236   int32u mainSPUsed;       // bytes used in main stack
00237   int32u processSPUsed;    // bytes used in process stack
00238   int32u mainStackBottom;  // address of the bottom of the stack
00239   HalCrashIcsrType icsr;   // interrupt control state register
00240   HalCrashShcsrType shcsr; // system handlers control and state register
00241   HalCrashIntActiveType intActive;  // irq active bit register
00242   HalCrashCfsrType cfsr;   // configurable fault status register
```

```
00243   HalCrashHfsrType hfsr;   // hard fault status register
00244   HalCrashDfsrType dfsr;   // debug fault status register
00245   int32u faultAddress;     // fault address register (MMAR or BFAR)
00246   HalCrashAfsrType afsr;   // auxiliary fault status register
00247   int32u returns[NUM_RETURNS];  // probable return addresses found on the
     stack
00248   HalCrashSpecificDataType data;  // additional data specific to the crash type
00249 } HalCrashInfoType;
00250
00251 typedef union
00252 {
00253   HalCrashInfoType crash;
00254   HalBootParamType boot;
00255 } HalResetInfoType;
00256
00257 #define RESETINFO_WORDS  ((sizeof(HalResetInfoType)+3)/4)
00258
00259 extern HalResetInfoType halResetInfo;
00260
00261 #endif // DOXYGEN_SHOULD_SKIP_THIS
00262
00266 #define halResetWasCrash() \
00267                   ( ( (1 << halGetResetInfo()) & RESET_CRASH_REASON_MASK) != 0)
00268
00273 int32u halGetMainStackBytesUsed(void);
00274
00279 void halPrintCrashSummary(int8u port);
00280
00285 void halPrintCrashDetails(int8u port);
00286
00291 void halPrintCrashData(int8u port);
00292
00293 #endif  //__EM3XX_DIAGNOSTIC_H__
00294
```

8.41   ember-configuration-defaults.h File Reference

Macros

- #define EMBER_API_MAJOR_VERSION
- #define EMBER_API_MINOR_VERSION
- #define EMBER_STACK_PROFILE
- #define EMBER_MAX_END_DEVICE_CHILDREN
- #define EMBER_SECURITY_LEVEL
- #define EMBER_CHILD_TABLE_SIZE
- #define EMBER_KEY_TABLE_SIZE
- #define EMBER_CERTIFICATE_TABLE_SIZE
- #define EMBER_MAX_DEPTH
- #define EMBER_MAX_HOPS
- #define EMBER_PACKET_BUFFER_COUNT
- #define EMBER_MAX_NEIGHBOR_TABLE_SIZE
- #define EMBER_NEIGHBOR_TABLE_SIZE
- #define EMBER_INDIRECT_TRANSMISSION_TIMEOUT
- #define EMBER_MAX_INDIRECT_TRANSMISSION_TIMEOUT
- #define EMBER_SEND_MULTICASTS_TO_SLEEPY_ADDRESS
- #define EMBER_END_DEVICE_POLL_TIMEOUT
- #define EMBER_END_DEVICE_POLL_TIMEOUT_SHIFT
- #define EMBER_MOBILE_NODE_POLL_TIMEOUT
- #define EMBER_APS_UNICAST_MESSAGE_COUNT
- #define EMBER_BINDING_TABLE_SIZE
- #define EMBER_ADDRESS_TABLE_SIZE

- #define EMBER_RESERVED_MOBILE_CHILD_ENTRIES
- #define EMBER_ROUTE_TABLE_SIZE
- #define EMBER_DISCOVERY_TABLE_SIZE
- #define EMBER_MULTICAST_TABLE_SIZE
- #define EMBER_SOURCE_ROUTE_TABLE_SIZE
- #define EMBER_DEFAULT_BROADCAST_TABLE_SIZE
- #define EMBER_BROADCAST_TABLE_SIZE
- #define EMBER_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 EMBER_SUPPORTED_NETWORKS

8.41.1   Detailed Description

User-configurable stack memory allocation defaults.

Note

> Application developers should **not** modify any portion of this file. Doing so may cause mysterious bugs. Allocations should be adjusted only by defining the appropriate macros in the application's CONFIGURATION_HEADER.

See Configuration for documentation.

Definition in file ember-configuration-defaults.h.

8.42   ember-configuration-defaults.h

```
00001
00014 //  Todo:
00015 //  - explain how to use a configuration header
00016 //  - the documentation of the custom handlers should
00017 //     go in hal/ember-configuration.c, not here
00018 //  - the stack profile documentation is out of date
00019
00047 #ifndef __EMBER_CONFIGURATION_DEFAULTS_H__
00048 #define __EMBER_CONFIGURATION_DEFAULTS_H__
00049
00050 #ifdef CONFIGURATION_HEADER
00051   #include CONFIGURATION_HEADER
00052 #endif
00053
00054 #ifndef EMBER_API_MAJOR_VERSION
00055
00058   #define EMBER_API_MAJOR_VERSION 2
```

```
00059 #endif
00060
00061 #ifndef EMBER_API_MINOR_VERSION
00062
00065   #define EMBER_API_MINOR_VERSION 0
00066 #endif
00067
00080 #ifndef EMBER_STACK_PROFILE
00081   #define EMBER_STACK_PROFILE 0
00082 #endif
00083
00084 #if (EMBER_STACK_PROFILE == 2)
00085   #define EMBER_MAX_DEPTH                    15
00086   #define EMBER_SECURITY_LEVEL               5
00087   #define EMBER_MIN_ROUTE_TABLE_SIZE        10
00088   #define EMBER_MIN_DISCOVERY_TABLE_SIZE     4
00089   #define EMBER_INDIRECT_TRANSMISSION_TIMEOUT 7680
00090   #define EMBER_SEND_MULTICASTS_TO_SLEEPY_ADDRESS FALSE
00091 #endif
00092
00093 #ifndef EMBER_MAX_END_DEVICE_CHILDREN
00094
00098   #define EMBER_MAX_END_DEVICE_CHILDREN 6
00099 #endif
00100
00101 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00102 /* Need to put in a compile time check to make sure that we aren't specifying
00103  * too many child devices.  The NCP may or may not support 64 end devices.  But
00104  * the host code doesn't matter.
00105  */
00106 #if defined(HAL_HAS_INT64) || defined(EZSP_HOST)
00107   #if EMBER_MAX_END_DEVICE_CHILDREN > 64
00108     #error "EMBER_MAX_END_DEVICE_CHILDREN can not exceed 64."
00109   #endif
00110 #else
00111   #if EMBER_MAX_END_DEVICE_CHILDREN > 32
00112     #error "EMBER_MAX_END_DEVICE_CHILDREN can not exceed 32."
00113   #endif
00114 #endif
00115
00116 #endif // DOXYGEN_SHOULD_SKIP_THIS
00117
00118 #ifndef EMBER_SECURITY_LEVEL
00119
00123   #define EMBER_SECURITY_LEVEL 5
00124 #endif
00125
00126 #if ! (EMBER_SECURITY_LEVEL == 0            \
00127        || EMBER_SECURITY_LEVEL == 5)
00128   #error "Unsupported security level"
00129 #endif
00130
00131 #ifdef EMBER_CHILD_TABLE_SIZE
00132   #if (EMBER_MAX_END_DEVICE_CHILDREN < EMBER_CHILD_TABLE_SIZE)
00133     #undef EMBER_CHILD_TABLE_SIZE
00134   #endif
00135 #endif
00136
00137 #ifndef EMBER_CHILD_TABLE_SIZE
00138
00152   #define EMBER_CHILD_TABLE_SIZE EMBER_MAX_END_DEVICE_CHILDREN
00153 #endif
00154
00168 #ifndef EMBER_KEY_TABLE_SIZE
00169   #define EMBER_KEY_TABLE_SIZE 0
00170 #endif
00171
00181 #ifndef EMBER_CERTIFICATE_TABLE_SIZE
00182   #define EMBER_CERTIFICATE_TABLE_SIZE 0
00183 #else
00184   #if EMBER_CERTIFICATE_TABLE_SIZE > 1
00185     #error "EMBER_CERTIFICATE_TABLE_SIZE > 1 is not supported!"
00186   #endif
00187 #endif
00188
00194 #ifndef EMBER_MAX_DEPTH
00195   #define EMBER_MAX_DEPTH                    15
00196 #elif (EMBER_MAX_DEPTH > 15)
00197   // Depth is a 4-bit field
00198   #error "EMBER_MAX_DEPTH cannot be greater than 15"
```

```
00199 #endif
00200
00207 #ifndef EMBER_MAX_HOPS
00208   #define EMBER_MAX_HOPS (2 * EMBER_MAX_DEPTH)
00209 #endif
00210
00217 #ifndef EMBER_PACKET_BUFFER_COUNT
00218   #define EMBER_PACKET_BUFFER_COUNT 24
00219 #endif
00220
00232 #define EMBER_MAX_NEIGHBOR_TABLE_SIZE 16
00233 #ifndef EMBER_NEIGHBOR_TABLE_SIZE
00234   #define EMBER_NEIGHBOR_TABLE_SIZE 16
00235 #endif
00236
00243 #ifndef EMBER_INDIRECT_TRANSMISSION_TIMEOUT
00244   #define EMBER_INDIRECT_TRANSMISSION_TIMEOUT 3000
00245 #endif
00246 #define EMBER_MAX_INDIRECT_TRANSMISSION_TIMEOUT 30000
00247 #if (EMBER_INDIRECT_TRANSMISSION_TIMEOUT
        \
00248       > EMBER_MAX_INDIRECT_TRANSMISSION_TIMEOUT)
00249   #error "Indirect transmission timeout too large."
00250 #endif
00251
00258 #ifndef EMBER_SEND_MULTICASTS_TO_SLEEPY_ADDRESS
00259   #define EMBER_SEND_MULTICASTS_TO_SLEEPY_ADDRESS FALSE
00260 #endif
00261
00262
00277 #ifndef EMBER_END_DEVICE_POLL_TIMEOUT
00278   #define EMBER_END_DEVICE_POLL_TIMEOUT 5
00279 #endif
00280
00288 #ifndef EMBER_END_DEVICE_POLL_TIMEOUT_SHIFT
00289   #define EMBER_END_DEVICE_POLL_TIMEOUT_SHIFT 6
00290 #endif
00291
00298 #ifndef EMBER_MOBILE_NODE_POLL_TIMEOUT
00299   #define EMBER_MOBILE_NODE_POLL_TIMEOUT 20
00300 #endif
00301
00314 #ifndef EMBER_APS_UNICAST_MESSAGE_COUNT
00315   #define EMBER_APS_UNICAST_MESSAGE_COUNT 10
00316 #endif
00317
00320 #ifndef EMBER_BINDING_TABLE_SIZE
00321   #define EMBER_BINDING_TABLE_SIZE 0
00322 #endif
00323
00328 #ifndef EMBER_ADDRESS_TABLE_SIZE
00329   #define EMBER_ADDRESS_TABLE_SIZE 8
00330 #endif
00331
00338 #ifndef EMBER_RESERVED_MOBILE_CHILD_ENTRIES
00339   #define EMBER_RESERVED_MOBILE_CHILD_ENTRIES 0
00340 #endif
00341
00348 #ifndef EMBER_ROUTE_TABLE_SIZE
00349   #ifdef EMBER_MIN_ROUTE_TABLE_SIZE
00350     #define EMBER_ROUTE_TABLE_SIZE EMBER_MIN_ROUTE_TABLE_SIZE
00351   #else
00352     #define EMBER_ROUTE_TABLE_SIZE 16
00353   #endif
00354 #elif defined(EMBER_MIN_ROUTE_TABLE_SIZE) \
00355       && EMBER_ROUTE_TABLE_SIZE < EMBER_MIN_ROUTE_TABLE_SIZE
00356   #error "EMBER_ROUTE_TABLE_SIZE is less than required by stack profile."
00357 #endif
00358
00364 #ifndef EMBER_DISCOVERY_TABLE_SIZE
00365   #ifdef EMBER_MIN_DISCOVERY_TABLE_SIZE
00366     #define EMBER_DISCOVERY_TABLE_SIZE EMBER_MIN_DISCOVERY_TABLE_SIZE
00367   #else
00368     #define EMBER_DISCOVERY_TABLE_SIZE 8
00369   #endif
00370 #elif defined(EMBER_MIN_DISCOVERY_TABLE_SIZE) \
00371       && EMBER_DISCOVERY_TABLE_SIZE < EMBER_MIN_DISCOVERY_TABLE_SIZE
00372   #error "EMBER_DISCOVERY_TABLE_SIZE is less than required by stack profile."
00373 #endif
00374
```

```
00380 #ifndef EMBER_MULTICAST_TABLE_SIZE
00381   #define EMBER_MULTICAST_TABLE_SIZE 8
00382 #endif
00383
00390 #ifndef EMBER_SOURCE_ROUTE_TABLE_SIZE
00391   #define EMBER_SOURCE_ROUTE_TABLE_SIZE 32
00392 #endif
00393
00405 #if !defined(EMBER_ZLL_STACK) && (EMBER_STACK_PROFILE == 2) &&
      !defined(EMBER_TEST)
00406   #if defined(EMBER_BROADCAST_TABLE_SIZE)
00407     #error "Cannot override broadcast table size unless (EMBER_STACK_PROFILE !=
      2) or EMBER_ZLL_STACK"
00408   #endif
00409 #endif
00410
00411 #define EMBER_DEFAULT_BROADCAST_TABLE_SIZE 15
00412
00413 #ifndef EMBER_BROADCAST_TABLE_SIZE
00414   #define EMBER_BROADCAST_TABLE_SIZE EMBER_DEFAULT_BROADCAST_TABLE_SIZE
00415 #elif EMBER_BROADCAST_TABLE_SIZE < EMBER_DEFAULT_BROADCAST_TABLE_SIZE
00416   #error "EMBER_BROADCAST_TABLE_SIZE is less than the minimum value of 15."
00417 #elif 254 < EMBER_BROADCAST_TABLE_SIZE
00418   #error "EMBER_BROADCAST_TABLE_SIZE is larger than the maximum value of 254."
00419 #endif
00420
00430 #if !defined(EMBER_ASSERT_OUTPUT_DISABLED) \
00431     && !defined(EMBER_ASSERT_SERIAL_PORT)
00432   #define EMBER_ASSERT_SERIAL_PORT 1
00433 #endif
00434
00448 #define EMBER_MAXIMUM_ALARM_DATA_SIZE 16
00449
00467 #ifndef EMBER_BROADCAST_ALARM_DATA_SIZE
00468   #define EMBER_BROADCAST_ALARM_DATA_SIZE 0
00469 #elif EMBER_MAXIMUM_ALARM_DATA_SIZE < EMBER_BROADCAST_ALARM_DATA_SIZE
00470   #error "EMBER_BROADCAST_ALARM_DATA_SIZE is too large."
00471 #endif
00472
00481 #ifndef EMBER_UNICAST_ALARM_DATA_SIZE
00482   #define EMBER_UNICAST_ALARM_DATA_SIZE 0
00483 #elif EMBER_MAXIMUM_ALARM_DATA_SIZE < EMBER_UNICAST_ALARM_DATA_SIZE
00484   #error "EMBER_UNICAST_ALARM_DATA_SIZE is too large."
00485 #endif
00486
00490 #ifndef EMBER_FRAGMENT_DELAY_MS
00491   #define EMBER_FRAGMENT_DELAY_MS 0
00492 #endif
00493
00497 #define EMBER_FRAGMENT_MAX_WINDOW_SIZE 8
00498
00503 #ifndef EMBER_FRAGMENT_WINDOW_SIZE
00504   #define EMBER_FRAGMENT_WINDOW_SIZE 1
00505 #elif EMBER_FRAGMENT_MAX_WINDOW_SIZE < EMBER_FRAGMENT_WINDOW_SIZE
00506   #error "EMBER_FRAGMENT_WINDOW_SIZE is too large."
00507 #endif
00508
00509 #ifndef EMBER_BINDING_TABLE_TOKEN_SIZE
00510   #define EMBER_BINDING_TABLE_TOKEN_SIZE EMBER_BINDING_TABLE_SIZE
00511 #endif
00512 #ifndef EMBER_CHILD_TABLE_TOKEN_SIZE
00513   #define EMBER_CHILD_TABLE_TOKEN_SIZE EMBER_CHILD_TABLE_SIZE
00514 #endif
00515 #ifndef EMBER_KEY_TABLE_TOKEN_SIZE
00516   #define EMBER_KEY_TABLE_TOKEN_SIZE EMBER_KEY_TABLE_SIZE
00517 #endif
00518
00531 #ifndef EMBER_REQUEST_KEY_TIMEOUT
00532   #define EMBER_REQUEST_KEY_TIMEOUT 0
00533 #elif EMBER_REQUEST_KEY_TIMEOUT > 10
00534   #error "EMBER_REQUEST_KEY_TIMEOUT is too large."
00535 #endif
00536
00540 #ifndef EMBER_END_DEVICE_BIND_TIMEOUT
00541   #define EMBER_END_DEVICE_BIND_TIMEOUT 60
00542 #endif
00543
00552 #ifndef EMBER_PAN_ID_CONFLICT_REPORT_THRESHOLD
00553   #define EMBER_PAN_ID_CONFLICT_REPORT_THRESHOLD 1
00554 #endif
```

```
00555
00561 #ifndef EMBER_TASK_COUNT
00562  #define EMBER_TASK_COUNT (3)
00563 #endif
00564
00567 #ifndef EMBER_SUPPORTED_NETWORKS
00568 #ifdef EMBER_TEST
00569  #define EMBER_SUPPORTED_NETWORKS 2
00570 #else
00571  #define EMBER_SUPPORTED_NETWORKS 1
00572 #endif
00573 #endif
00574
00575
00576 #if defined(EMBER_ZLL_STACK)
00577
00578   #ifndef EMBER_ZLL_GROUP_ADDRESSES
00579
00581     #define EMBER_ZLL_GROUP_ADDRESSES 1
00582   #endif
00583
00584   #ifndef EMBER_ZLL_RSSI_THRESHOLD
00585
00587     #define EMBER_ZLL_RSSI_THRESHOLD -128
00588   #endif
00589
00590 #endif // EMBER_ZLL_STACK
00591
00592
00597 #endif //__EMBER_CONFIGURATION_DEFAULTS_H__
```

8.43 ember-debug.h File Reference

Macros

- #define NO_DEBUG
- #define BASIC_DEBUG
- #define FULL_DEBUG
- #define emberDebugInit(port)

Functions

- void emberDebugAssert (PGM_P filename, int linenumber)
- void emberDebugMemoryDump (int8u *start, int8u *end)
- void emberDebugBinaryPrintf (PGM_P formatString,...)
- void emDebugSendVuartMessage (int8u *buff, int8u len)
- void emberDebugError (EmberStatus code)
- boolean emberDebugReportOff (void)
- void emberDebugReportRestore (boolean state)
- void emberDebugPrintf (PGM_P formatString,...)

8.43.1 Detailed Description

See Debugging Utilities for documentation.

Definition in file ember-debug.h.

8.44   ember-debug.h

```
00001
00008 #ifndef __EMBER_DEBUG_H__
00009 #define __EMBER_DEBUG_H__
00010
00019 // Define the values for DEBUG_LEVEL
00020 #define NO_DEBUG    0
00021 #define BASIC_DEBUG 1
00022 #define FULL_DEBUG  2
00023
00030 #define emberDebugInit(port) do {} while(FALSE)
00031
00032 #if (DEBUG_LEVEL >= BASIC_DEBUG) || defined(DOXYGEN_SHOULD_SKIP_THIS)
00033
00039  void emberDebugAssert(PGM_P filename, int linenumber);
00040
00041
00049  void emberDebugMemoryDump(int8u *start, int8u *
    end);
00050
00077 void emberDebugBinaryPrintf(PGM_P formatString, ...);
00078
00086  void emDebugSendVuartMessage(int8u *buff, int8u
     len);
00087
00088 #else // (DEBUG_LEVEL >= BASIC_DEBUG) || defined(DOXYGEN_SHOULD_SKIP_THIS)
00089   #define emberDebugAssert(filename, linenumber) do {} while(FALSE)
00090   #define emberDebugMemoryDump(start, end) do {} while(FALSE)
00091   #define emberDebugBinaryPrintf(formatstring, ...) do {} while(FALSE)
00092   #define emDebugSendVuartMessage(buff, len) do {} while(FALSE)
00093 #endif // (DEBUG_LEVEL >= BASIC_DEBUG) || defined(DOXYGEN_SHOULD_SKIP_THIS)
00094
00095 #if (DEBUG_LEVEL == FULL_DEBUG) || defined(DOXYGEN_SHOULD_SKIP_THIS)
00096
00100  void emberDebugError(EmberStatus code);
00101
00106  boolean emberDebugReportOff(void);
00107
00113  void emberDebugReportRestore(boolean state);
00114
00115 // Format: Same as emberSerialPrintf
00116 // emberDebugPrintf("format string"[, parameters ...])
00132 void emberDebugPrintf(PGM_P formatString, ...);
00133
00134 #else // (DEBUG_LEVEL == FULL_DEBUG) || defined(DOXYGEN_SHOULD_SKIP_THIS)
00135   #define emberDebugError(code) do {} while(FALSE)
00136   // Note the following doesn't have a do{}while(FALSE)
00137   //   because it has a return value
00138   #define emberDebugReportOff() (FALSE)
00139   #define emberDebugReportRestore(state) do {} while(FALSE)
00140   #define emberDebugPrintf(...) do {} while(FALSE)
00141 #endif // (DEBUG_LEVEL == FULL_DEBUG) || defined(DOXYGEN_SHOULD_SKIP_THIS)
00142
00146 #endif // __EMBER_DEBUG_H__
00147
```

8.45   ember-types.h File Reference

Data Structures

- struct EmberVersion

    Version struct containing all version information.

- struct EmberZigbeeNetwork

    Defines a ZigBee network and the associated parameters.

- struct EmberNetworkInitStruct

    Defines the network initialization configuration that should be used when emberNetworkInitExtended() is called by the application.

- struct EmberNetworkParameters

Holds network parameters.

- struct EmberApsFrame

  An in-memory representation of a ZigBee APS frame of an incoming or outgoing message.

- struct EmberBindingTableEntry

  Defines an entry in the binding table.

- struct EmberNeighborTableEntry

  Defines an entry in the neighbor table.

- struct EmberRouteTableEntry

  Defines an entry in the route table.

- struct EmberMulticastTableEntry

  Defines an entry in the multicast table.

- struct EmberEventControl

  Control structure for events.

- struct EmberTaskControl

  Control structure for tasks.

- struct EmberKeyData

  This data structure contains the key data that is passed into various other functions.

- struct EmberCertificateData

  This data structure contains the certificate data that is used for Certificate Based Key Exchange (CBKE).

- struct EmberPublicKeyData

  This data structure contains the public key data that is used for Certificate Based Key Exchange (CBKE).

- struct EmberPrivateKeyData

  This data structure contains the private key data that is used for Certificate Based Key Exchange (CBKE).

- struct EmberSmacData

  This data structure contains the Shared Message Authentication Code (SMAC) data that is used for Certificate Based Key Exchange (CBKE).

- struct EmberSignatureData

  This data structure contains a DSA signature. It is the bit concatenation of the 'r' and 's' components of the signature.

- struct EmberMessageDigest

  This data structure contains an AES-MMO Hash (the message digest).

- struct EmberAesMmoHashContext

  This data structure contains the context data when calculating an AES MMO hash (message digest).

- struct EmberInitialSecurityState

  This describes the Initial Security features and requirements that will be used when forming or joining the network.

- struct EmberCurrentSecurityState

  This describes the security features used by the stack for a joined device.

- struct EmberKeyStruct

  This describes a one of several different types of keys and its associated data.

- struct EmberMfgSecurityStruct

  This structure is used to get/set the security config that is stored in manufacturing tokens.

- struct EmberMacFilterMatchStruct

  This structure indicates a matching raw MAC message has been received by the application configured MAC filters.

Macros

- #define EMBER_JOIN_DECISION_STRINGS
- #define EMBER_DEVICE_UPDATE_STRINGS
- #define emberInitializeNetworkParameters(parameters)
- #define EMBER_COUNTER_STRINGS
- #define EMBER_STANDARD_SECURITY_MODE
- #define EMBER_TRUST_CENTER_NODE_ID
- #define EMBER_NO_TRUST_CENTER_MODE
- #define EMBER_GLOBAL_LINK_KEY
- #define EMBER_MFG_SECURITY_CONFIG_MAGIC_NUMBER
- #define EMBER_MAC_FILTER_MATCH_ENABLED_MASK
- #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_MASK
- #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_MASK
- #define EMBER_MAC_FILTER_MATCH_ON_DEST_MASK
- #define EMBER_MAC_FILTER_MATCH_ON_SOURCE_MASK
- #define EMBER_MAC_FILTER_MATCH_ENABLED
- #define EMBER_MAC_FILTER_MATCH_DISABLED
- #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_NONE
- #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_LOCAL
- #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_BROADCAST
- #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_NONE
- #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_NON_LOCAL
- #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_LOCAL
- #define EMBER_MAC_FILTER_MATCH_ON_DEST_BROADCAST_SHORT
- #define EMBER_MAC_FILTER_MATCH_ON_DEST_UNICAST_SHORT
- #define EMBER_MAC_FILTER_MATCH_ON_DEST_UNICAST_LONG
- #define EMBER_MAC_FILTER_MATCH_ON_SOURCE_LONG
- #define EMBER_MAC_FILTER_MATCH_ON_SOURCE_SHORT
- #define EMBER_MAC_FILTER_MATCH_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_EN-
  D_DEVICE,
  EMBER_SLEEPY_END_DEVICE, EMBER_MOBILE_END_DEVICE }
- enum EmberNetworkInitBitmask { EMBER_NETWORK_INIT_NO_OPTIONS, EMBER_NETW-
  ORK_INIT_PARENT_INFO_IN_TOKEN }

- enum EmberApsOption {
  EMBER_APS_OPTION_NONE, EMBER_APS_OPTION_DSA_SIGN, EMBER_APS_OPTION_ENCRYPTION, EMBER_APS_OPTION_RETRY,
  EMBER_APS_OPTION_ENABLE_ROUTE_DISCOVERY, EMBER_APS_OPTION_FORCE_ROUTE_DISCOVERY, EMBER_APS_OPTION_SOURCE_EUI64, EMBER_APS_OPTION_DESTINATION_EUI64,
  EMBER_APS_OPTION_ENABLE_ADDRESS_DISCOVERY, EMBER_APS_OPTION_POLL_RESPONSE, EMBER_APS_OPTION_ZDO_RESPONSE_REQUIRED, EMBER_APS_OPTION_FRAGMENT }
- enum EmberIncomingMessageType {
  EMBER_INCOMING_UNICAST, EMBER_INCOMING_UNICAST_REPLY, EMBER_INCOMING_MULTICAST, EMBER_INCOMING_MULTICAST_LOOPBACK,
  EMBER_INCOMING_BROADCAST, EMBER_INCOMING_BROADCAST_LOOPBACK }
- enum EmberOutgoingMessageType {
  EMBER_OUTGOING_DIRECT, EMBER_OUTGOING_VIA_ADDRESS_TABLE, EMBER_OUTGOING_VIA_BINDING, EMBER_OUTGOING_MULTICAST,
  EMBER_OUTGOING_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, EMBER_REJOIN_REASON_NONE,
  EMBER_REJOIN_DUE_TO_NWK_KEY_UPDATE, EMBER_REJOIN_DUE_TO_LEAVE_MESSAGE, EMBER_REJOIN_DUE_TO_NO_PARENT, EMBER_REJOIN_DUE_TO_ZLL_TOUCHLINK,
  EMBER_REJOIN_DUE_TO_APP_EVENT_5, EMBER_REJOIN_DUE_TO_APP_EVENT_4, EMBER_REJOIN_DUE_TO_APP_EVENT_3, EMBER_REJOIN_DUE_TO_APP_EVENT_2,
  EMBER_REJOIN_DUE_TO_APP_EVENT_1 }
- enum 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, EMBER_REJOIN_REASON_NONE,
  EMBER_REJOIN_DUE_TO_NWK_KEY_UPDATE, EMBER_REJOIN_DUE_TO_LEAVE_MESSAGE, EMBER_REJOIN_DUE_TO_NO_PARENT, EMBER_REJOIN_DUE_TO_ZLL_TOUCHLINK,
  EMBER_REJOIN_DUE_TO_APP_EVENT_5, EMBER_REJOIN_DUE_TO_APP_EVENT_4, EMBER_REJOIN_DUE_TO_APP_EVENT_3, EMBER_REJOIN_DUE_TO_APP_EVENT_2,
  EMBER_REJOIN_DUE_TO_APP_EVENT_1 }
- enum EmberClusterListId { EMBER_INPUT_CLUSTER_LIST, EMBER_OUTPUT_CLUSTER_LIST }

- enum EmberEventUnits {
  EMBER_EVENT_INACTIVE, EMBER_EVENT_MS_TIME, EMBER_EVENT_QS_TIME, EM-
  BER_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_BROADCAS-
  T, EMBER_COUNTER_MAC_RX_UNICAST, EMBER_COUNTER_MAC_TX_UNICAST_SU-
  CCESS,
  EMBER_COUNTER_MAC_TX_UNICAST_RETRY, EMBER_COUNTER_MAC_TX_UNICAS-
  T_FAILED, EMBER_COUNTER_APS_DATA_RX_BROADCAST, EMBER_COUNTER_APS-
  _DATA_TX_BROADCAST,
  EMBER_COUNTER_APS_DATA_RX_UNICAST, EMBER_COUNTER_APS_DATA_TX_UN-
  ICAST_SUCCESS, EMBER_COUNTER_APS_DATA_TX_UNICAST_RETRY, EMBER_COU-
  NTER_APS_DATA_TX_UNICAST_FAILED,
  EMBER_COUNTER_ROUTE_DISCOVERY_INITIATED, EMBER_COUNTER_NEIGHBOR_-
  ADDED, EMBER_COUNTER_NEIGHBOR_REMOVED, EMBER_COUNTER_NEIGHBOR_S-
  TALE,
  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_COUN-
  TER_FAILURE, EMBER_COUNTER_APS_FRAME_COUNTER_FAILURE, EMBER_COUN-
  TER_ASH_XOFF,
  EMBER_COUNTER_APS_LINK_KEY_NOT_AUTHORIZED, EMBER_COUNTER_NWK_DE-
  CRYPTION_FAILURE, EMBER_COUNTER_APS_DECRYPTION_FAILURE, EMBER_COU-
  NTER_ALLOCATE_PACKET_BUFFER_FAILURE,
  EMBER_COUNTER_RELAYED_UNICAST, EMBER_COUNTER_PHY_TO_MAC_QUEUE_L-
  IMIT_REACHED, EMBER_COUNTER_PACKET_VALIDATE_LIBRARY_DROPPED_COUN-
  T, EMBER_COUNTER_TYPE_COUNT }
- enum EmberInitialSecurityBitmask {
  EMBER_DISTRIBUTED_TRUST_CENTER_MODE, EMBER_TRUST_CENTER_GLOBAL_L-
  INK_KEY, EMBER_PRECONFIGURED_NETWORK_KEY_MODE, EMBER_HAVE_TRUST-
  _CENTER_EUI64,
  EMBER_TRUST_CENTER_USES_HASHED_LINK_KEY, EMBER_HAVE_PRECONFIGURE-
  D_KEY, EMBER_HAVE_NETWORK_KEY, EMBER_GET_LINK_KEY_WHEN_JOINING,
  EMBER_REQUIRE_ENCRYPTED_KEY, EMBER_NO_FRAME_COUNTER_RESET, EMBER-
  _GET_PRECONFIGURED_KEY_FROM_INSTALL_CODE }
- enum EmberExtendedSecurityBitmask { EMBER_JOINER_GLOBAL_LINK_KEY, EMBER_N-
  WK_LEAVE_REQUEST_NOT_ALLOWED }
- enum EmberCurrentSecurityBitmask {
  EMBER_STANDARD_SECURITY_MODE_, EMBER_DISTRIBUTED_TRUST_CENTER_MO-
  DE_, EMBER_TRUST_CENTER_GLOBAL_LINK_KEY_, EMBER_HAVE_TRUST_CENTER-
  _LINK_KEY,
  EMBER_TRUST_CENTER_USES_HASHED_LINK_KEY_ }
- enum EmberKeyStructBitmask {
  EMBER_KEY_HAS_SEQUENCE_NUMBER, EMBER_KEY_HAS_OUTGOING_FRAME_CO-
  UNTER, EMBER_KEY_HAS_INCOMING_FRAME_COUNTER, EMBER_KEY_HAS_PARTN-
  ER_EUI64,
  EMBER_KEY_IS_AUTHORIZED, EMBER_KEY_PARTNER_IS_SLEEPY }
- enum EmberKeyType {
  EMBER_TRUST_CENTER_LINK_KEY, EMBER_TRUST_CENTER_MASTER_KEY, EMBE-
  R_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, E-
  MBER_TRUST_CENTER_LINK_KEY_ESTABLISHED, EMBER_KEY_ESTABLISHMENT_T-
  IMEOUT,
  EMBER_KEY_TABLE_FULL, EMBER_TC_RESPONDED_TO_KEY_REQUEST, EMBER_T-
  C_APP_KEY_SENT_TO_REQUESTER, EMBER_TC_RESPONSE_TO_KEY_REQUEST_FAIL-
  ED,
  EMBER_TC_REQUEST_KEY_TYPE_NOT_SUPPORTED, EMBER_TC_NO_LINK_KEY_FO-
  R_REQUESTER, EMBER_TC_REQUESTER_EUI64_UNKNOWN, EMBER_TC_RECEIVED_-
  FIRST_APP_KEY_REQUEST,
  EMBER_TC_TIMEOUT_WAITING_FOR_SECOND_APP_KEY_REQUEST, EMBER_TC_NO-
  N_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_K-
  EY_REQUESTS }
- enum EmberKeySettings { EMBER_KEY_PERMISSIONS_NONE, EMBER_KEY_PERMISSIO-
  NS_READING_ALLOWED, EMBER_KEY_PERMISSIONS_HASHING_ALLOWED }
- enum EmberMacPassthroughType {
  EMBER_MAC_PASSTHROUGH_NONE, EMBER_MAC_PASSTHROUGH_SE_INTERPAN, E-
  MBER_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 __EMBERSTATUS_TYPE__
- #define EMBER_MAX_802_15_4_CHANNEL_NUMBER
- #define EMBER_MIN_802_15_4_CHANNEL_NUMBER
- #define EMBER_NUM_802_15_4_CHANNELS
- #define EMBER_ALL_802_15_4_CHANNELS_MASK
- #define EMBER_ZIGBEE_COORDINATOR_ADDRESS

- #define EMBER_NULL_NODE_ID
- #define EMBER_NULL_BINDING
- #define EMBER_TABLE_ENTRY_UNUSED_NODE_ID
- #define EMBER_MULTICAST_NODE_ID
- #define EMBER_UNKNOWN_NODE_ID
- #define EMBER_DISCOVERY_ACTIVE_NODE_ID
- #define EMBER_NULL_ADDRESS_TABLE_INDEX
- #define EMBER_ZDO_ENDPOINT
- #define EMBER_BROADCAST_ENDPOINT
- #define EMBER_ZDO_PROFILE_ID
- #define EMBER_WILDCARD_PROFILE_ID
- #define EMBER_MAXIMUM_STANDARD_PROFILE_ID
- #define EMBER_BROADCAST_TABLE_TIMEOUT_QS
- enum EmberVersionType { EMBER_VERSION_TYPE_PRE_RELEASE, EMBER_VERSION_T-YPE_GA }
- enum EmberLeaveRequestFlags { EMBER_ZIGBEE_LEAVE_AND_REJOIN, EMBER_ZIGBE-E_LEAVE_AND_REMOVE_CHILDREN }
- enum EmberLeaveReason {
  EMBER_LEAVE_REASON_NONE, EMBER_LEAVE_DUE_TO_NWK_LEAVE_MESSAGE, E-MBER_LEAVE_DUE_TO_APS_REMOVE_MESSAGE, EMBER_LEAVE_DUE_TO_ZDO_LE-AVE_MESSAGE,
  EMBER_LEAVE_DUE_TO_ZLL_TOUCHLINK, EMBER_LEAVE_DUE_TO_APP_EVENT_1 }
- typedef int8u EmberStatus
- typedef int8u EmberEUI64 [EUI64_SIZE]
- typedef int8u EmberMessageBuffer
- typedef int16u EmberNodeId
- typedef int16u EmberMulticastId
- typedef int16u EmberPanId
- const EmberVersion emberVersion

ZigBee Broadcast Addresses

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

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

Ember Concentrator Types

- #define EMBER_LOW_RAM_CONCENTRATOR
- #define EMBER_HIGH_RAM_CONCENTRATOR

txPowerModes for emberSetTxPowerMode and mfglibSetPower

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

Alarm Message and Counters Request Definitions

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

ZDO response status.

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

- enum EmberZdoStatus {
  EMBER_ZDP_SUCCESS, EMBER_ZDP_INVALID_REQUEST_TYPE, EMBER_ZDP_DEVIC-
  E_NOT_FOUND, EMBER_ZDP_INVALID_ENDPOINT,
  EMBER_ZDP_NOT_ACTIVE, EMBER_ZDP_NOT_SUPPORTED, EMBER_ZDP_TIMEOUT, E-
  MBER_ZDP_NO_MATCH,
  EMBER_ZDP_NO_ENTRY, EMBER_ZDP_NO_DESCRIPTOR, EMBER_ZDP_INSUFFICIEN-
  T_SPACE, EMBER_ZDP_NOT_PERMITTED,
  EMBER_ZDP_TABLE_FULL, EMBER_ZDP_NOT_AUTHORIZED, EMBER_NWK_ALREAD-
  Y_PRESENT, EMBER_NWK_TABLE_FULL,
  EMBER_NWK_UNKNOWN_DEVICE }

Network and IEEE Address Request/Response

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

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

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

- #define NETWORK_ADDRESS_REQUEST
- #define NETWORK_ADDRESS_RESPONSE
- #define IEEE_ADDRESS_REQUEST
- #define IEEE_ADDRESS_RESPONSE

Node Descriptor Request/Response

```
<br>

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

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

- #define NODE_DESCRIPTOR_REQUEST
- #define NODE_DESCRIPTOR_RESPONSE

Power Descriptor Request / Response

```
<br>

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

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

- #define POWER_DESCRIPTOR_REQUEST
- #define POWER_DESCRIPTOR_RESPONSE

Simple Descriptor Request / Response

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

- #define SIMPLE_DESCRIPTOR_REQUEST
- #define SIMPLE_DESCRIPTOR_RESPONSE

Active Endpoints Request / Response

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

- #define ACTIVE_ENDPOINTS_REQUEST
- #define ACTIVE_ENDPOINTS_RESPONSE

Match Descriptors Request / Response

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

- #define MATCH_DESCRIPTORS_REQUEST
- #define MATCH_DESCRIPTORS_RESPONSE

Discovery Cache Request / Response

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

- #define DISCOVERY_CACHE_REQUEST
- #define DISCOVERY_CACHE_RESPONSE

End Device Announce and End Device Announce Response

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

- #define END_DEVICE_ANNOUNCE
- #define END_DEVICE_ANNOUNCE_RESPONSE

System Server Discovery Request / Response

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

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

- #define SYSTEM_SERVER_DISCOVERY_REQUEST
- #define SYSTEM_SERVER_DISCOVERY_RESPONSE

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_BINDI-
  NG_TABLE_CACHE,
  EMBER_ZDP_PRIMARY_DISCOVERY_CACHE, EMBER_ZDP_SECONDARY_DISCOVER-
  Y_CACHE, EMBER_ZDP_NETWORK_MANAGER }

Find Node Cache Request / Response

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

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

- #define FIND_NODE_CACHE_REQUEST
- #define FIND_NODE_CACHE_RESPONSE

End Device Bind Request / Response

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

- #define END_DEVICE_BIND_REQUEST
- #define END_DEVICE_BIND_RESPONSE

Binding types and Request / Response

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

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

- #define UNICAST_BINDING
- #define UNICAST_MANY_TO_ONE_BINDING
- #define MULTICAST_BINDING
- #define BIND_REQUEST
- #define BIND_RESPONSE
- #define UNBIND_REQUEST
- #define UNBIND_RESPONSE

LQI Table Request / Response

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

The device-type byte has the following fields:

Name	Mask	Values
device type	0x03	0x00 coordinator
		0x01 router
		0x02 end device
		0x03 unknown
rx mode	0x0C	0x00 off when idle
		0x04 on when idle
		0x08 unknown
relationship	0x70	0x00 parent
		0x10 child
		0x20 sibling
		0x30 other
		0x40 previous child
reserved	0x10	

The permit-joining byte has the following fields

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

- #define LQI_TABLE_REQUEST
- #define LQI_TABLE_RESPONSE

Routing Table Request / Response

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

The status byte has the following fields:

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

- #define ROUTING_TABLE_REQUEST
- #define ROUTING_TABLE_RESPONSE

Binding Table Request / Response

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

Note

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

- #define BINDING_TABLE_REQUEST
- #define BINDING_TABLE_RESPONSE

Leave Request / Response

```
Request:  <transaction sequence number: 1> <EUI64:8> <flags:1>
          The flag bits are:
          0x40 remove children
          0x80 rejoin
Response: <transaction sequence number: 1> <status:1>
```

- #define LEAVE_REQUEST
- #define LEAVE_RESPONSE
- #define LEAVE_REQUEST_REMOVE_CHILDREN_FLAG
- #define LEAVE_REQUEST_REJOIN_FLAG

Permit Joining Request / Response

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

- #define PERMIT_JOINING_REQUEST
- #define PERMIT_JOINING_RESPONSE

Network Update Request / Response

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

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

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

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

```
    Unicast requests always get a response, which is INVALID_REQUEST if the
    duration is not a legal value.

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

- #define NWK_UPDATE_REQUEST
- #define NWK_UPDATE_RESPONSE

Unsupported

Not mandatory and not supported.

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

ZDO configuration flags.

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

EMBER_APPLICATION_RECEIVES_SUPPORTED_ZDO_REQUESTS EMBER_APPLICATION_H-ANDLES_UNSUPPORTED_ZDO_REQUESTS EMBER_APPLICATION_HANDLES_ENDPOINT_Z-DO_REQUESTS EMBER_APPLICATION_HANDLES_BINDING_ZDO_REQUESTS

See ember-configuration.h for more information.

- enum EmberZdoConfigurationFlags { EMBER_APP_RECEIVES_SUPPORTED_ZDO_REQUES-TS, EMBER_APP_HANDLES_UNSUPPORTED_ZDO_REQUESTS, EMBER_APP_HANDLE-S_ZDO_ENDPOINT_REQUESTS, EMBER_APP_HANDLES_ZDO_BINDING_REQUESTS }

8.45.1 Detailed Description

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

Definition in file ember-types.h.

8.45.2   Variable Documentation

8.45.2.1   EmberEventControl∗ control

The control structure for the event.

Definition at line 1167 of file ember-types.h.

8.45.2.2   void(∗ handler)(void)

The procedure to call when the event fires.

Definition at line 1169 of file ember-types.h.

8.46   ember-types.h

```
00001
00020 #ifndef EMBER_TYPES_H
00021 #define EMBER_TYPES_H
00022
00023 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00024 #include "stack/config/ember-configuration-defaults.h
      "
00025 #include "stack/include/ember-static-struct.h"
00026 #endif //DOXYGEN_SHOULD_SKIP_THIS
00027
00032
00036 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00037 enum EmberVersionType
00038 #else
00039 typedef int8u EmberVersionType;
00040 enum
00041 #endif
00042 {
00043   EMBER_VERSION_TYPE_PRE_RELEASE = 0x00,
00044   // Anything other than 0xAA is considered pre-release
00045   // We may define other types in the future (e.g. beta, alpha)
00046   // We chose an arbitrary number (0xAA) to allow for expansion, but
00047   // to prevent ambiguity in case 0x00 or 0xFF is accidentally retrieved
00048   // as the version type.
00049   EMBER_VERSION_TYPE_GA = 0xAA,
00050 };
00051
00055 typedef struct {
00056   int16u build;
00057   int8u major;
00058   int8u minor;
00059   int8u patch;
00060   int8u special;
00061   EmberVersionType type;
00062 } EmberVersion;
00063
00067 extern const EmberVersion emberVersion;
00068
00072 #define EUI64_SIZE 8
00073
00077 #define EXTENDED_PAN_ID_SIZE 8
00078
00082 #define EMBER_ENCRYPTION_KEY_SIZE 16
00083
00088 #define EMBER_CERTIFICATE_SIZE 48
00089
00093 #define EMBER_PUBLIC_KEY_SIZE 22
00094
00098 #define EMBER_PRIVATE_KEY_SIZE 21
00099
00103 #define EMBER_SMAC_SIZE 16
00104
00109 #define EMBER_SIGNATURE_SIZE 42
00110
00114 #define EMBER_AES_HASH_BLOCK_SIZE 16
```

```
00115
00116
00120 #ifndef __EMBERSTATUS_TYPE__
00121 #define __EMBERSTATUS_TYPE__
00122   typedef int8u EmberStatus;
00123 #endif //__EMBERSTATUS_TYPE__
00124
00128 typedef int8u EmberEUI64[EUI64_SIZE];
00129
00139 typedef int8u EmberMessageBuffer;
00140
00144 typedef int16u EmberNodeId;
00145
00147 typedef int16u EmberMulticastId;
00148
00152 typedef int16u EmberPanId;
00153
00157 #define EMBER_MAX_802_15_4_CHANNEL_NUMBER 26
00158
00162 #define EMBER_MIN_802_15_4_CHANNEL_NUMBER 11
00163
00167 #define EMBER_NUM_802_15_4_CHANNELS \
00168   (EMBER_MAX_802_15_4_CHANNEL_NUMBER - EMBER_MIN_802_15_4_CHANNEL_NUMBER + 1)
00169
00173 #define EMBER_ALL_802_15_4_CHANNELS_MASK 0x07FFF800UL
00174
00178 #define EMBER_ZIGBEE_COORDINATOR_ADDRESS 0x0000
00179
00184 #define EMBER_NULL_NODE_ID 0xFFFF
00185
00190 #define EMBER_NULL_BINDING 0xFF
00191
00201 #define EMBER_TABLE_ENTRY_UNUSED_NODE_ID  0xFFFF
00202
00209 #define EMBER_MULTICAST_NODE_ID            0xFFFE
00210
00218 #define EMBER_UNKNOWN_NODE_ID              0xFFFD
00219
00227 #define EMBER_DISCOVERY_ACTIVE_NODE_ID    0xFFFC
00228
00233 #define EMBER_NULL_ADDRESS_TABLE_INDEX 0xFF
00234
00238 #define EMBER_ZDO_ENDPOINT 0
00239
00243 #define EMBER_BROADCAST_ENDPOINT 0xFF
00244
00248 #define EMBER_ZDO_PROFILE_ID  0x0000
00249
00253 #define EMBER_WILDCARD_PROFILE_ID  0xFFFF
00254
00258 #define EMBER_MAXIMUM_STANDARD_PROFILE_ID  0x7FFF
00259
00260
00266 #define EMBER_BROADCAST_TABLE_TIMEOUT_QS (20 * 4)
00267
00268
00269 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00270 enum EmberLeaveRequestFlags
00271 #else
00272 typedef int8u EmberLeaveRequestFlags;
00273 enum
00274 #endif
00275 {
00277   EMBER_ZIGBEE_LEAVE_AND_REJOIN            = 0x20,
00278
00280   EMBER_ZIGBEE_LEAVE_AND_REMOVE_CHILDREN
00281   = 0x40,
00281 };
00282
00283 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00284 enum EmberLeaveReason
00285 #else
00286 typedef int8u EmberLeaveReason;
00287 enum
00288 #endif
00289 {
00290   EMBER_LEAVE_REASON_NONE               = 0,
00291   EMBER_LEAVE_DUE_TO_NWK_LEAVE_MESSAGE  = 1
    ,
00292   EMBER_LEAVE_DUE_TO_APS_REMOVE_MESSAGE =
```

```
         2,
00293    // Currently, the stack does not process the ZDO leave message since it is
         optional
00294    EMBER_LEAVE_DUE_TO_ZDO_LEAVE_MESSAGE  = 3
         ,
00295    EMBER_LEAVE_DUE_TO_ZLL_TOUCHLINK      = 4,
00296
00297    EMBER_LEAVE_DUE_TO_APP_EVENT_1        = 0xFF,
00298 };
00299
00301
00302
00315 #define EMBER_BROADCAST_ADDRESS 0xFFFC
00316
00317 #define EMBER_RX_ON_WHEN_IDLE_BROADCAST_ADDRESS 0xFFFD
00318
00319 #define EMBER_SLEEPY_BROADCAST_ADDRESS 0xFFFF
00320
00328 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00329 enum EmberNodeType
00330 #else
00331 typedef int8u EmberNodeType;
00332 enum
00333 #endif
00334 {
00336    EMBER_UNKNOWN_DEVICE = 0,
00338    EMBER_COORDINATOR = 1,
00340    EMBER_ROUTER = 2,
00342    EMBER_END_DEVICE = 3,
00346    EMBER_SLEEPY_END_DEVICE = 4,
00348    EMBER_MOBILE_END_DEVICE = 5
00349 };
00350
00354 typedef struct {
00355    int16u panId;
00356    int8u channel;
00357    boolean allowingJoin;
00358    int8u extendedPanId[8];
00359    int8u stackProfile;
00360    int8u nwkUpdateId;
00361 } EmberZigbeeNetwork;
00362
00363
00368 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00369 enum EmberNetworkInitBitmask
00370 #else
00371 typedef int16u EmberNetworkInitBitmask;
00372 enum
00373 #endif
00374 {
00375    EMBER_NETWORK_INIT_NO_OPTIONS            = 0x0000
         ,
00379    EMBER_NETWORK_INIT_PARENT_INFO_IN_TOKEN
         = 0x0001,
00380 };
00381
00382
00387 typedef struct {
00388    EmberNetworkInitBitmask bitmask;
00389 } EmberNetworkInitStruct;
00390
00391
00398 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00399 enum EmberApsOption
00400 #else
00401 typedef int16u EmberApsOption;
00402 enum
00403 #endif
00404 {
00406    EMBER_APS_OPTION_NONE                    = 0x0000,
00418    EMBER_APS_OPTION_DSA_SIGN                = 0x0010,
00421    EMBER_APS_OPTION_ENCRYPTION              = 0x0020
         ,
00425    EMBER_APS_OPTION_RETRY                   = 0x0040,
00431    EMBER_APS_OPTION_ENABLE_ROUTE_DISCOVERY
         = 0x0100,
00434    EMBER_APS_OPTION_FORCE_ROUTE_DISCOVERY
         = 0x0200,
00436    EMBER_APS_OPTION_SOURCE_EUI64            =
         0x0400,
```

```
00438    EMBER_APS_OPTION_DESTINATION_EUI64        =
     0x0800,
00441    EMBER_APS_OPTION_ENABLE_ADDRESS_DISCOVERY
      = 0x1000,
00446    EMBER_APS_OPTION_POLL_RESPONSE            =
     0x2000,
00451    EMBER_APS_OPTION_ZDO_RESPONSE_REQUIRED
         = 0x4000,
00457    EMBER_APS_OPTION_FRAGMENT                 =
     SIGNED_ENUM 0x8000
00458 };
00459
00460
00461
00465 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00466 enum EmberIncomingMessageType
00467 #else
00468 typedef int8u EmberIncomingMessageType;
00469 enum
00470 #endif
00471 {
00473    EMBER_INCOMING_UNICAST,
00475    EMBER_INCOMING_UNICAST_REPLY,
00477    EMBER_INCOMING_MULTICAST,
00479    EMBER_INCOMING_MULTICAST_LOOPBACK,
00481    EMBER_INCOMING_BROADCAST,
00483    EMBER_INCOMING_BROADCAST_LOOPBACK
00484 };
00485
00486
00490 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00491 enum EmberOutgoingMessageType
00492 #else
00493 typedef int8u EmberOutgoingMessageType;
00494 enum
00495 #endif
00496 {
00498    EMBER_OUTGOING_DIRECT,
00500    EMBER_OUTGOING_VIA_ADDRESS_TABLE,
00502    EMBER_OUTGOING_VIA_BINDING,
00505    EMBER_OUTGOING_MULTICAST,
00508    EMBER_OUTGOING_BROADCAST
00509 };
00510
00511
00515 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00516 enum EmberNetworkStatus
00517 #else
00518 typedef int8u EmberNetworkStatus;
00519 enum
00520 #endif
00521 {
00523    EMBER_NO_NETWORK,
00525    EMBER_JOINING_NETWORK,
00527    EMBER_JOINED_NETWORK,
00530    EMBER_JOINED_NETWORK_NO_PARENT,
00532    EMBER_LEAVING_NETWORK
00533 };
00534
00535
00539 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00540 enum EmberNetworkScanType
00541 #else
00542 typedef int8u EmberNetworkScanType;
00543 enum
00544 #endif
00545 {
00547    EMBER_ENERGY_SCAN,
00549    EMBER_ACTIVE_SCAN
00550 };
00551
00552
00556 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00557 enum EmberBindingType
00558 #else
00559 typedef int8u EmberBindingType;
00560 enum
00561 #endif
00562 {
00564    EMBER_UNUSED_BINDING        = 0,
```

```
00566   EMBER_UNICAST_BINDING       = 1,
00570   EMBER_MANY_TO_ONE_BINDING   = 2,
00574   EMBER_MULTICAST_BINDING     = 3,
00575 };
00576
00577
00586 #define EMBER_LOW_RAM_CONCENTRATOR 0xFFF8
00587
00591 #define EMBER_HIGH_RAM_CONCENTRATOR 0xFFF9
00592
00594
00595
00599 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00600 enum EmberJoinDecision
00601 #else
00602 typedef int8u EmberJoinDecision;
00603 enum
00604 #endif
00605 {
00607   EMBER_USE_PRECONFIGURED_KEY = 0,
00609   EMBER_SEND_KEY_IN_THE_CLEAR,
00611   EMBER_DENY_JOIN,
00613   EMBER_NO_ACTION
00614 };
00615
00619 #define EMBER_JOIN_DECISION_STRINGS \
00620   "use preconfigured key",          \
00621   "send key in the clear",          \
00622   "deny join",                      \
00623   "no action",
00624
00625
00631 // These map to the actual values within the APS Command frame so they cannot
00632 // be arbitrarily changed.
00633 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00634 enum EmberDeviceUpdate
00635 #else
00636 typedef int8u EmberDeviceUpdate;
00637 enum
00638 #endif
00639 {
00640   EMBER_STANDARD_SECURITY_SECURED_REJOIN
         = 0,
00641   EMBER_STANDARD_SECURITY_UNSECURED_JOIN
         = 1,
00642   EMBER_DEVICE_LEFT                       = 2,
00643   EMBER_STANDARD_SECURITY_UNSECURED_REJOIN
        = 3,
00644   EMBER_HIGH_SECURITY_SECURED_REJOIN      =
       4,
00645   EMBER_HIGH_SECURITY_UNSECURED_JOIN      =
       5,
00646   /* 6 Reserved */
00647   EMBER_HIGH_SECURITY_UNSECURED_REJOIN
       = 7,
00648   /* 8 – 15 Reserved */
00649 };
00650
00654 #define EMBER_DEVICE_UPDATE_STRINGS                                         \
00655     "secured rejoin",                                                       \
00656     "UNsecured join",                                                       \
00657     "device left",                                                          \
00658     "UNsecured rejoin",                                                     \
00659     "high secured rejoin",                                                  \
00660     "high UNsecured join",                                                  \
00661     "RESERVED",                    /* reserved status code, per the spec. */ \
00662     "high UNsecured rejoin",
00663
00667 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00668 enum EmberDeviceUpdate
00669 #else
00670 typedef int8u EmberRejoinReason;
00671 enum
00672 #endif
00673 {
00674   EMBER_REJOIN_REASON_NONE          = 0,
00675   EMBER_REJOIN_DUE_TO_NWK_KEY_UPDATE = 1,
00676   EMBER_REJOIN_DUE_TO_LEAVE_MESSAGE  = 2,
00677   EMBER_REJOIN_DUE_TO_NO_PARENT      = 3,
00678   EMBER_REJOIN_DUE_TO_ZLL_TOUCHLINK  = 4,
```

```
00679
00680   // App. Framework events
00681   // 0xA0 - 0xE0
00682
00683   // Customer Defined Events
00684   //   I numbered these backwards in case there is ever request
00685   //   for more application events. We can expand them
00686   //   without renumbering the previous ones.
00687   EMBER_REJOIN_DUE_TO_APP_EVENT_5       = 0xFB,
00688   EMBER_REJOIN_DUE_TO_APP_EVENT_4       = 0xFC,
00689   EMBER_REJOIN_DUE_TO_APP_EVENT_3       = 0xFD,
00690   EMBER_REJOIN_DUE_TO_APP_EVENT_2       = 0xFE,
00691   EMBER_REJOIN_DUE_TO_APP_EVENT_1       = 0xFF,
00692 };
00693
00697 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00698 enum EmberClusterListId
00699 #else
00700 typedef int8u EmberClusterListId;
00701 enum
00702 #endif
00703 {
00705   EMBER_INPUT_CLUSTER_LIST          = 0,
00707   EMBER_OUTPUT_CLUSTER_LIST         = 1
00708 };
00709
00710
00715 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00716 enum EmberEventUnits
00717 #else
00718 typedef int8u EmberEventUnits;
00719 enum
00720 #endif
00721 {
00723   EMBER_EVENT_INACTIVE = 0,
00725   EMBER_EVENT_MS_TIME,
00728   EMBER_EVENT_QS_TIME,
00731   EMBER_EVENT_MINUTE_TIME,
00733   EMBER_EVENT_ZERO_DELAY
00734 };
00735
00736
00740 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00741 enum EmberJoinMethod
00742 #else
00743 typedef int8u EmberJoinMethod;
00744 enum
00745 #endif
00746 {
00752   EMBER_USE_MAC_ASSOCIATION         = 0,
00753
00764   EMBER_USE_NWK_REJOIN              = 1,
00765
00766
00767   /* For those networks where the "permit joining" flag is never turned
00768    * on, they will need to use a NWK Rejoin.  If those devices have been
00769    * preconfigured with the  NWK key (including sequence number) they can use
00770    * a secured rejoin.  This is only necessary for end devices since they need
00771    * a parent.  Routers can simply use the ::EMBER_USE_NWK_COMMISSIONING
00772    * join method below.
00773    */
00774   EMBER_USE_NWK_REJOIN_HAVE_NWK_KEY = 2,
00775
00780   EMBER_USE_NWK_COMMISSIONING       = 3,
00781 };
00782
00783
00790 typedef struct {
00792   int8u   extendedPanId[8];
00794   int16u  panId;
00796   int8s   radioTxPower;
00798   int8u   radioChannel;
00803   EmberJoinMethod joinMethod;
00804
00809   EmberNodeId nwkManagerId;
00815   int8u nwkUpdateId;
00821   int32u channels;
00822 } EmberNetworkParameters;
00823
00824
```

```
00825 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00826 #define emberInitializeNetworkParameters(parameters) \
00827   (MEMSET(parameters, 0, sizeof(EmberNetworkParameters)))
00828 #else
00829 void emberInitializeNetworkParameters(
      EmberNetworkParameters* parameters);
00830 #endif
00831
00835 typedef struct {
00837   int16u profileId;
00839   int16u clusterId;
00841   int8u sourceEndpoint;
00843   int8u destinationEndpoint;
00845   EmberApsOption options;
00847   int16u groupId;
00849   int8u sequence;
00850 } EmberApsFrame;
00851
00852
00859 typedef struct {
00861   EmberBindingType type;
00863   int8u local;
00871   int16u clusterId;
00873   int8u remote;
00878   EmberEUI64 identifier;
00880   int8u networkIndex;
00881 } EmberBindingTableEntry;
00882
00883
00889 typedef struct {
00891   int16u shortId;
00894   int8u  averageLqi;
00897   int8u  inCost;
00904   int8u  outCost;
00910   int8u  age;
00912   EmberEUI64 longId;
00913 } EmberNeighborTableEntry;
00914
00920 typedef struct {
00922   int16u destination;
00924   int16u nextHop;
00927   int8u status;
00930   int8u age;
00933   int8u concentratorType;
00938   int8u routeRecordState;
00939 } EmberRouteTableEntry;
00940
00948 typedef struct {
00950   EmberMulticastId multicastId;
00954   int8u endpoint;
00956   int8u networkIndex;
00957 } EmberMulticastTableEntry;
00958
00963 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00964 enum EmberCounterType
00965 #else
00966 typedef int8u EmberCounterType;
00967 enum
00968 #endif
00969 {
00971   EMBER_COUNTER_MAC_RX_BROADCAST = 0,
00973   EMBER_COUNTER_MAC_TX_BROADCAST = 1,
00975   EMBER_COUNTER_MAC_RX_UNICAST = 2,
00977   EMBER_COUNTER_MAC_TX_UNICAST_SUCCESS = 3,
00983   EMBER_COUNTER_MAC_TX_UNICAST_RETRY = 4,
00985   EMBER_COUNTER_MAC_TX_UNICAST_FAILED = 5,
00986
00988   EMBER_COUNTER_APS_DATA_RX_BROADCAST = 6,
00990   EMBER_COUNTER_APS_DATA_TX_BROADCAST = 7,
00992   EMBER_COUNTER_APS_DATA_RX_UNICAST = 8,
00994   EMBER_COUNTER_APS_DATA_TX_UNICAST_SUCCESS
      = 9,
01000   EMBER_COUNTER_APS_DATA_TX_UNICAST_RETRY
      = 10,
01002   EMBER_COUNTER_APS_DATA_TX_UNICAST_FAILED
      = 11,
01003
01006   EMBER_COUNTER_ROUTE_DISCOVERY_INITIATED
      = 12,
01007
```

```
01009   EMBER_COUNTER_NEIGHBOR_ADDED = 13,
01011   EMBER_COUNTER_NEIGHBOR_REMOVED = 14,
01013   EMBER_COUNTER_NEIGHBOR_STALE = 15,
01014
01016   EMBER_COUNTER_JOIN_INDICATION = 16,
01018   EMBER_COUNTER_CHILD_REMOVED = 17,
01019
01021   EMBER_COUNTER_ASH_OVERFLOW_ERROR = 18,
01023   EMBER_COUNTER_ASH_FRAMING_ERROR = 19,
01025   EMBER_COUNTER_ASH_OVERRUN_ERROR = 20,
01026
01029   EMBER_COUNTER_NWK_FRAME_COUNTER_FAILURE
     = 21,
01030
01033   EMBER_COUNTER_APS_FRAME_COUNTER_FAILURE
     = 22,
01034
01036   EMBER_COUNTER_ASH_XOFF = 23,
01037
01041   EMBER_COUNTER_APS_LINK_KEY_NOT_AUTHORIZED
     = 24,
01042
01045   EMBER_COUNTER_NWK_DECRYPTION_FAILURE = 25
     ,
01046
01049   EMBER_COUNTER_APS_DECRYPTION_FAILURE = 26
     ,
01050
01055   EMBER_COUNTER_ALLOCATE_PACKET_BUFFER_FAILURE
     = 27,
01056
01058   EMBER_COUNTER_RELAYED_UNICAST = 28,
01059
01071   EMBER_COUNTER_PHY_TO_MAC_QUEUE_LIMIT_REACHED
     = 29,
01072
01077   EMBER_COUNTER_PACKET_VALIDATE_LIBRARY_DROPPED_COUNT
     = 30,
01078
01080   EMBER_COUNTER_TYPE_COUNT = 31
01081 };
01082
01086 #define EMBER_COUNTER_STRINGS                \
01087     "Mac Rx Bcast",                         \
01088     "Mac Tx Bcast",                         \
01089     "Mac Rx Ucast",                         \
01090     "Mac Tx Ucast",                         \
01091     "Mac Tx Ucast Retry",                   \
01092     "Mac Tx Ucast Fail",                    \
01093     "APS Rx Bcast",                         \
01094     "APS Tx Bcast",                         \
01095     "APS Rx Ucast",                         \
01096     "APS Tx Ucast Success",                 \
01097     "APS Tx Ucast Retry",                   \
01098     "APS Tx Ucast Fail",                    \
01099     "Route Disc Initiated",                 \
01100     "Neighbor Added",                       \
01101     "Neighbor Removed",                     \
01102     "Neighbor Stale",                       \
01103     "Join Indication",                      \
01104     "Child Moved",                          \
01105     "ASH Overflow",                         \
01106     "ASH Frame Error",                      \
01107     "ASH Overrun Error",                    \
01108     "NWK FC Failure",                       \
01109     "APS FC Failure",                       \
01110     "ASH XOff",                             \
01111     "APS Unauthorized Key",                 \
01112     "NWK Decrypt Failures",                 \
01113     "APS Decrypt Failures",                 \
01114     "Packet Buffer Allocate Failures",      \
01115     "Relayed Ucast",                        \
01116     "Phy to MAC queue limit reached",       \
01117     "Packet Validate drop count",           \
01118     NULL
01119
01121 typedef int8u EmberTaskId;
01122
01123 #ifndef EZSP_HOST
01124
```

```
01130  typedef struct {
01132    EmberEventUnits status;
01134    EmberTaskId taskid;
01138    int32u timeToExecute;
01139  } EmberEventControl;
01140 #else
01141  // host applications use an older, basic form of the event system
01148  typedef struct {
01150    EmberEventUnits status;
01154    int16u timeToExecute;
01155  } EmberEventControl;
01156 #endif
01157
01165 typedef PGM struct {
01167   EmberEventControl *control;
01169   void (*handler)(void);
01170 } EmberEventData;
01171
01176 typedef struct {
01177   // The time when the next event associated with this task will fire
01178   int32u nextEventTime;
01179   // The list of events associated with this task
01180   EmberEventData *events;
01181   // A flag that indicates the task has something to do other than events
01182   boolean busy;
01183 } EmberTaskControl;
01184
01189
01194 #define EMBER_TX_POWER_MODE_DEFAULT             0x0000
01195
01198 #define EMBER_TX_POWER_MODE_BOOST               0x0001
01199
01203 #define EMBER_TX_POWER_MODE_ALTERNATE           0x0002
01204
01208 #define EMBER_TX_POWER_MODE_BOOST_AND_ALTERNATE (EMBER_TX_POWER_MODE_BOOST
      \
01209                                                 |EMBER_TX_POWER_MODE_ALTERNATE)
01210 #ifndef DOXYGEN_SHOULD_SKIP_THIS
01211 // The application does not ever need to call emberSetTxPowerMode() with the
01212 // txPowerMode parameter set to this value.  This value is used internally by
01213 // the stack to indicate that the default token configuration has not been
01214 // overridden by a prior call to emberSetTxPowerMode().
01215 #define EMBER_TX_POWER_MODE_USE_TOKEN           0x8000
01216 #endif//DOXYGEN_SHOULD_SKIP_THIS
01217
01219
01224
01232 #define EMBER_PRIVATE_PROFILE_ID  0xC00E
01233
01272 #define EMBER_BROADCAST_ALARM_CLUSTER       0x0000
01273
01310 #define EMBER_UNICAST_ALARM_CLUSTER         0x0001
01311
01327 #define EMBER_CACHED_UNICAST_ALARM_CLUSTER  0x0002
01328
01332 #define EMBER_REPORT_COUNTERS_REQUEST 0x0003
01333
01335 #define EMBER_REPORT_COUNTERS_RESPONSE 0x8003
01336
01341 #define EMBER_REPORT_AND_CLEAR_COUNTERS_REQUEST 0x0004
01342
01344 #define EMBER_REPORT_AND_CLEAR_COUNTERS_RESPONSE 0x8004
01345
01350 #define EMBER_OTA_CERTIFICATE_UPGRADE_CLUSTER 0x0005
01351
01353
01354
01357 typedef struct {
01359   int8u contents[EMBER_ENCRYPTION_KEY_SIZE];
01360 } EmberKeyData;
01361
01364 typedef struct {
01365   /* This is the certificate byte data. */
01366   int8u contents[EMBER_CERTIFICATE_SIZE];
01367 } EmberCertificateData;
01368
01371 typedef struct {
01372   int8u contents[EMBER_PUBLIC_KEY_SIZE];
01373 } EmberPublicKeyData;
01374
```

```
01377 typedef struct {
01378   int8u contents[EMBER_PRIVATE_KEY_SIZE];
01379 } EmberPrivateKeyData;
01380
01383 typedef struct {
01384   int8u contents[EMBER_SMAC_SIZE];
01385 } EmberSmacData;
01386
01390 typedef struct {
01391   int8u contents[EMBER_SIGNATURE_SIZE];
01392 } EmberSignatureData;
01393
01396 typedef struct {
01397   int8u contents[EMBER_AES_HASH_BLOCK_SIZE];
01398 } EmberMessageDigest;
01399
01403 typedef struct {
01404   int8u result[EMBER_AES_HASH_BLOCK_SIZE];
01405   int32u length;
01406 } EmberAesMmoHashContext;
01407
01408
01414 #define EMBER_STANDARD_SECURITY_MODE 0x0000
01415
01419 #define EMBER_TRUST_CENTER_NODE_ID 0x0000
01420
01421
01425 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01426 enum EmberInitialSecurityBitmask
01427 #else
01428 typedef int16u EmberInitialSecurityBitmask;
01429 enum
01430 #endif
01431 {
01434   EMBER_DISTRIBUTED_TRUST_CENTER_MODE
      = 0x0002,
01437   EMBER_TRUST_CENTER_GLOBAL_LINK_KEY        =
       0x0004,
01440   EMBER_PRECONFIGURED_NETWORK_KEY_MODE
       = 0x0008,
01441
01442 #if !defined DOXYGEN_SHOULD_SKIP_THIS
01443   // Hidden fields used internally.
01444   EMBER_HAVE_TRUST_CENTER_UNKNOWN_KEY_TOKEN = 0x0010,
01445   EMBER_HAVE_TRUST_CENTER_LINK_KEY_TOKEN    = 0x0020,
01446   EMBER_HAVE_TRUST_CENTER_MASTER_KEY_TOKEN  = 0x0030,
01447 #endif
01448
01458   EMBER_HAVE_TRUST_CENTER_EUI64              =
      0x0040,
01459
01466   EMBER_TRUST_CENTER_USES_HASHED_LINK_KEY
        = 0x0084,
01467
01471   EMBER_HAVE_PRECONFIGURED_KEY               =
      0x0100,
01475   EMBER_HAVE_NETWORK_KEY                     = 0x0200,
01480   EMBER_GET_LINK_KEY_WHEN_JOINING           =
      0x0400,
01486   EMBER_REQUIRE_ENCRYPTED_KEY               = 0x0800
      ,
01494   EMBER_NO_FRAME_COUNTER_RESET              =
      0x1000,
01500   EMBER_GET_PRECONFIGURED_KEY_FROM_INSTALL_CODE
       = 0x2000,
01501
01502 #if !defined DOXYGEN_SHOULD_SKIP_THIS
01503   // Internal data
01504   EM_SAVED_IN_TOKEN                         = 0x4000,
01505   #define EM_SECURITY_INITIALIZED        0x00008000L
01506
01507   // This is only used internally.  High security is not released or supported
01508   // except for golden unit compliance.
01509   #define EMBER_HIGH_SECURITY_MODE        0x0001
01510 #else
01511   /* All other bits are reserved and must be zero. */
01512 #endif
01513 };
01514
01518 #ifdef DOXYGEN_SHOULD_SKIP_THIS
```

```
01519 enum EmberExtendedSecurityBitmask
01520 #else
01521 typedef int16u EmberExtendedSecurityBitmask;
01522 enum
01523 #endif
01524 {
01525 #ifndef DOXYGEN_SHOULD_SKIP_THIS
01526   // If this bit is set, we set the 'key token data' field in the Initial
01527   // Security Bitmask to 0 (No Preconfig Key token), otherwise we leave the
01528   // field as it is.
01529   EMBER_PRECONFIG_KEY_NOT_VALID      = 0x0001,
01530 #endif
01531
01532   // bits 1-3 are unused.
01533
01536   EMBER_JOINER_GLOBAL_LINK_KEY       = 0x0010,
01537
01538   // bit 5-7 reserved for future use (stored in TOKEN).
01539
01542   EMBER_NWK_LEAVE_REQUEST_NOT_ALLOWED =
      0x0100,
01543
01544 #ifndef DOXYGEN_SHOULD_SKIP_THIS
01545
01549   EMBER_R18_STACK_BEHAVIOR           = 0x0200,
01550 #endif
01551
01552   // bits 10-11 are reserved for future use (stored in RAM only).
01553
01554   // bits 12-15 are unused.
01555 };
01556
01559 #define EMBER_NO_TRUST_CENTER_MODE   EMBER_DISTRIBUTED_TRUST_CENTER_MODE
01560
01563 #define EMBER_GLOBAL_LINK_KEY   EMBER_TRUST_CENTER_GLOBAL_LINK_KEY
01564
01565
01566 #if !defined DOXYGEN_SHOULD_SKIP_THIS
01567   #define NO_TRUST_CENTER_KEY_TOKEN       0x0000
01568   #define TRUST_CENTER_KEY_TOKEN_MASK     0x0030
01569   #define SECURITY_BIT_TOKEN_MASK         0x71FF
01570
01571   #define SECURITY_LOWER_BIT_MASK         0x000000FF  // ""
01572   #define SECURITY_UPPER_BIT_MASK         0x00FF0000L // ""
01573 #endif
01574
01577 typedef struct {
01582   int16u bitmask;
01591   EmberKeyData preconfiguredKey;
01597   EmberKeyData networkKey;
01604   int8u networkKeySequenceNumber;
01612   EmberEUI64 preconfiguredTrustCenterEui64
      ;
01613 } EmberInitialSecurityState;
01614
01615
01619 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01620 enum EmberCurrentSecurityBitmask
01621 #else
01622 typedef int16u EmberCurrentSecurityBitmask;
01623 enum
01624 #endif
01625 {
01626 #if defined DOXYGEN_SHOULD_SKIP_THIS
01627   // These options are the same for Initial and Current Security state
01628
01631   EMBER_STANDARD_SECURITY_MODE_           =
      0x0000,
01634   EMBER_DISTRIBUTED_TRUST_CENTER_MODE_
       = 0x0002,
01637   EMBER_TRUST_CENTER_GLOBAL_LINK_KEY_
      = 0x0004,
01638 #else
01639   // Bit 3 reserved
01640 #endif
01641
01642   EMBER_HAVE_TRUST_CENTER_LINK_KEY           =
      0x0010,
01643
01645   EMBER_TRUST_CENTER_USES_HASHED_LINK_KEY_
```

```
           = 0x0084,
01646
01647    // Bits 1,5,6, 8-15 reserved
01648 };
01649
01650 #if !defined DOXYGEN_SHOULD_SKIP_THIS
01651    #define INITIAL_AND_CURRENT_BITMASK        0x00FF
01652 #endif
01653
01654
01657 typedef struct {
01660    EmberCurrentSecurityBitmask bitmask;
01664    EmberEUI64 trustCenterLongAddress;
01665 } EmberCurrentSecurityState;
01666
01667
01671 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01672 enum EmberKeyStructBitmask
01673 #else
01674 typedef int16u EmberKeyStructBitmask;
01675 enum
01676 #endif
01677 {
01680    EMBER_KEY_HAS_SEQUENCE_NUMBER         = 0x0001,
01684    EMBER_KEY_HAS_OUTGOING_FRAME_COUNTER =
       0x0002,
01688    EMBER_KEY_HAS_INCOMING_FRAME_COUNTER =
       0x0004,
01692    EMBER_KEY_HAS_PARTNER_EUI64           = 0x0008,
01696    EMBER_KEY_IS_AUTHORIZED               = 0x0010,
01701    EMBER_KEY_PARTNER_IS_SLEEPY           = 0x0020,
01702
01703 };
01704
01706 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01707 enum EmberKeyType
01708 #else
01709 typedef int8u EmberKeyType;
01710 enum
01711 #endif
01712 {
01714    EMBER_TRUST_CENTER_LINK_KEY        = 1,
01716    EMBER_TRUST_CENTER_MASTER_KEY      = 2,
01718    EMBER_CURRENT_NETWORK_KEY          = 3,
01720    EMBER_NEXT_NETWORK_KEY             = 4,
01722    EMBER_APPLICATION_LINK_KEY         = 5,
01724    EMBER_APPLICATION_MASTER_KEY       = 6,
01725 };
01726
01730 typedef struct {
01734    EmberKeyStructBitmask bitmask;
01736    EmberKeyType type;
01738    EmberKeyData key;
01741    int32u outgoingFrameCounter;
01744    int32u incomingFrameCounter;
01747    int8u sequenceNumber;
01750    EmberEUI64 partnerEUI64;
01751 } EmberKeyStruct;
01752
01753
01757 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01758 enum EmberKeyStatus
01759 #else
01760 typedef int8u EmberKeyStatus;
01761 enum
01762 #endif
01763 {
01764    EMBER_APP_LINK_KEY_ESTABLISHED           = 1,
01765    EMBER_APP_MASTER_KEY_ESTABLISHED         = 2,
01766    EMBER_TRUST_CENTER_LINK_KEY_ESTABLISHED
       = 3,
01767    EMBER_KEY_ESTABLISHMENT_TIMEOUT          = 4,
01768    EMBER_KEY_TABLE_FULL                     = 5,
01769
01770    // These are success status values applying only to the
01771    // Trust Center answering key requests
01772    EMBER_TC_RESPONDED_TO_KEY_REQUEST        = 6
       ,
01773    EMBER_TC_APP_KEY_SENT_TO_REQUESTER       =
       7,
```

```
01774
01775   // These are failure status values applying only to the
01776   // Trust Center answering key requests
01777   EMBER_TC_RESPONSE_TO_KEY_REQUEST_FAILED
        = 8,
01778   EMBER_TC_REQUEST_KEY_TYPE_NOT_SUPPORTED
        = 9,
01779   EMBER_TC_NO_LINK_KEY_FOR_REQUESTER        =
      10,
01780   EMBER_TC_REQUESTER_EUI64_UNKNOWN          = 11
      ,
01781   EMBER_TC_RECEIVED_FIRST_APP_KEY_REQUEST
        = 12,
01782   EMBER_TC_TIMEOUT_WAITING_FOR_SECOND_APP_KEY_REQUEST
       = 13,
01783   EMBER_TC_NON_MATCHING_APP_KEY_REQUEST_RECEIVED
          = 14,
01784   EMBER_TC_FAILED_TO_SEND_APP_KEYS          = 15
      ,
01785   EMBER_TC_FAILED_TO_STORE_APP_KEY_REQUEST
       = 16,
01786   EMBER_TC_REJECTED_APP_KEY_REQUEST         =
      17,
01787 };
01788
01792 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01793 enum EmberLinkKeyRequestPolicy
01794 #else
01795 typedef int8u EmberLinkKeyRequestPolicy;
01796 enum
01797 #endif
01798 {
01799   EMBER_DENY_KEY_REQUESTS  = 0x00,
01800   EMBER_ALLOW_KEY_REQUESTS = 0x01,
01801 };
01802
01803
01811 #if defined DOXYGEN_SHOULD_SKIP_THIS
01812 int8u* emberKeyContents(EmberKeyData* key);
01813 #else
01814 #define emberKeyContents(key) ((key)->contents)
01815 #endif
01816
01824 #if defined DOXYGEN_SHOULD_SKIP_THIS
01825 int8u* emberCertificateContents(
      EmberCertificateData* cert);
01826 #else
01827 #define emberCertificateContents(cert) ((cert)->contents)
01828 #endif
01829
01837 #if defined DOXYGEN_SHOULD_SKIP_THIS
01838 int8u* emberPublicKeyContents(EmberPublicKeyData
      * key);
01839 #else
01840 #define emberPublicKeyContents(key) ((key)->contents)
01841 #endif
01842
01850 #if defined DOXYGEN_SHOULD_SKIP_THIS
01851 int8u* emberPrivateKeyContents(EmberPrivateKeyData
      * key);
01852 #else
01853 #define emberPrivateKeyContents(key) ((key)->contents)
01854 #endif
01855
01860 #if defined DOXYGEN_SHOULD_SKIP_THIS
01861 int8u* emberSmacContents(EmberSmacData* key)
      ;
01862 #else
01863 #define emberSmacContents(key) ((key)->contents)
01864 #endif
01865
01869 #if defined DOXYGEN_SHOULD_SKIP_THIS
01870 int8u* emberSignatureContents(EmberSignatureData
      * sig);
01871 #else
01872 #define emberSignatureContents(sig) ((sig)->contents)
01873 #endif
01874
01875
01876 #ifdef DOXYGEN_SHOULD_SKIP_THIS
```

```
01877 enum EmberKeySettings
01878 #else
01879 typedef int16u EmberKeySettings;
01880 enum
01881 #endif
01882 {
01883   EMBER_KEY_PERMISSIONS_NONE            = 0x0000,
01884   EMBER_KEY_PERMISSIONS_READING_ALLOWED =
      0x0001,
01885   EMBER_KEY_PERMISSIONS_HASHING_ALLOWED =
      0x0002,
01886 };
01887
01888
01892 typedef struct {
01893   EmberKeySettings keySettings;
01894 } EmberMfgSecurityStruct;
01895
01896
01901 #define EMBER_MFG_SECURITY_CONFIG_MAGIC_NUMBER 0xCABAD11FUL
01902
01903
01908 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01909 enum EmberMacPassthroughType
01910 #else
01911 typedef int8u EmberMacPassthroughType;
01912 enum
01913 #endif
01914 {
01916   EMBER_MAC_PASSTHROUGH_NONE           = 0x00,
01918   EMBER_MAC_PASSTHROUGH_SE_INTERPAN    =
      0x01,
01920   EMBER_MAC_PASSTHROUGH_EMBERNET       = 0x02,
01922   EMBER_MAC_PASSTHROUGH_EMBERNET_SOURCE =
       0x04,
01924   EMBER_MAC_PASSTHROUGH_APPLICATION    =
      0x08,
01926   EMBER_MAC_PASSTHROUGH_CUSTOM         = 0x10,
01927
01928 #if !defined DOXYGEN_SHOULD_SKIP_THIS
01929
01930   EM_MAC_PASSTHROUGH_INTERNAL          = 0x80
01931 #endif
01932 };
01933
01938 typedef int16u EmberMacFilterMatchData;
01939
01940 #define EMBER_MAC_FILTER_MATCH_ENABLED_MASK          0x0001
01941 #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_MASK      0x0003
01942 #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_MASK    0x000C
01943 #define EMBER_MAC_FILTER_MATCH_ON_DEST_MASK          0x0030
01944 #define EMBER_MAC_FILTER_MATCH_ON_SOURCE_MASK        0x0080
01945
01946 // Globally turn on/off this filter
01947 #define EMBER_MAC_FILTER_MATCH_ENABLED               0x0000
01948 #define EMBER_MAC_FILTER_MATCH_DISABLED              0x0001
01949
01950 // Pick either one of these
01951 #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_NONE      0x0000
01952 #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_LOCAL     0x0001
01953 #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_BROADCAST 0x0002
01954
01955 // and one of these
01956 #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_NONE      0x0000
01957 #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_NON_LOCAL 0x0004
01958 #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_LOCAL     0x0008
01959
01960 // and one of these
01961 #define EMBER_MAC_FILTER_MATCH_ON_DEST_BROADCAST_SHORT 0x0000
01962 #define EMBER_MAC_FILTER_MATCH_ON_DEST_UNICAST_SHORT   0x0010
01963 #define EMBER_MAC_FILTER_MATCH_ON_DEST_UNICAST_LONG    0x0020
01964
01965 // and one of these
01966 #define EMBER_MAC_FILTER_MATCH_ON_SOURCE_LONG        0x0000
01967 #define EMBER_MAC_FILTER_MATCH_ON_SOURCE_SHORT       0x0080
01968
01969 // Last entry should set this and nothing else.  No other bits will be
      examined.
01970 #define EMBER_MAC_FILTER_MATCH_END                   0x8000
01971
```

```
01975 typedef struct {
01976   int8u filterIndexMatch;
01977   EmberMacPassthroughType legacyPassthroughType
      ;
01978   EmberMessageBuffer message;
01979 } EmberMacFilterMatchStruct;
01980
01981
01985 typedef int8u EmberLibraryStatus;
01986
01991
01997 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01998 enum EmberZdoStatus
01999 #else
02000 typedef int8u EmberZdoStatus;
02001 enum
02002 #endif
02003 {
02004   // These values are taken from Table 48 of ZDP Errata 043238r003 and Table 2
02005   // of NWK 02130r10.
02006   EMBER_ZDP_SUCCESS            = 0x00,
02007   // 0x01 to 0x7F are reserved
02008   EMBER_ZDP_INVALID_REQUEST_TYPE = 0x80,
02009   EMBER_ZDP_DEVICE_NOT_FOUND    = 0x81,
02010   EMBER_ZDP_INVALID_ENDPOINT    = 0x82,
02011   EMBER_ZDP_NOT_ACTIVE          = 0x83,
02012   EMBER_ZDP_NOT_SUPPORTED       = 0x84,
02013   EMBER_ZDP_TIMEOUT             = 0x85,
02014   EMBER_ZDP_NO_MATCH            = 0x86,
02015   // 0x87 is reserved           = 0x87,
02016   EMBER_ZDP_NO_ENTRY            = 0x88,
02017   EMBER_ZDP_NO_DESCRIPTOR       = 0x89,
02018   EMBER_ZDP_INSUFFICIENT_SPACE  = 0x8a,
02019   EMBER_ZDP_NOT_PERMITTED       = 0x8b,
02020   EMBER_ZDP_TABLE_FULL          = 0x8c,
02021   EMBER_ZDP_NOT_AUTHORIZED      = 0x8d,
02022
02023   EMBER_NWK_ALREADY_PRESENT     = 0xC5,
02024   EMBER_NWK_TABLE_FULL          = 0xC7,
02025   EMBER_NWK_UNKNOWN_DEVICE       = 0xC8
02026 };
02027
02040
02041
02042
02043
02044
02045
02046
02047
02048
02049
02050
02051
02052
02053
02054 #define NETWORK_ADDRESS_REQUEST      0x0000
02055 #define NETWORK_ADDRESS_RESPONSE     0x8000
02056 #define IEEE_ADDRESS_REQUEST         0x0001
02057 #define IEEE_ADDRESS_RESPONSE        0x8001
02058
02059
02066   //              <node descriptor: 13>
02067   //
02068   //  Node Descriptor field is divided into subfields of bitmasks as follows:
02069   //      (Note: All lengths below are given in bits rather than bytes.)
02070   //              Logical Type:                    3
02071   //              Complex Descriptor Available:    1
02072   //              User Descriptor Available:       1
02073   //              (reserved/unused):               3
02074   //              APS Flags:                       3
02075   //              Frequency Band:                  5
02076   //              MAC capability flags:            8
02077   //              Manufacturer Code:              16
02078   //              Maximum buffer size:             8
02079   //              Maximum incoming transfer size: 16
02080   //              Server mask:                    16
02081   //              Maximum outgoing transfer size: 16
02082   //              Descriptor Capability Flags:     8
02083   //      See ZigBee document 053474, Section 2.3.2.3 for more details.
```

```
02085 #define NODE_DESCRIPTOR_REQUEST      0x0002
02086 #define NODE_DESCRIPTOR_RESPONSE     0x8002
02087
02088
02097   //     See ZigBee document 053474, Section 2.3.2.4 for more details.
02099 #define POWER_DESCRIPTOR_REQUEST      0x0003
02100 #define POWER_DESCRIPTOR_RESPONSE     0x8003
02101
02102
02116 #define SIMPLE_DESCRIPTOR_REQUEST     0x0004
02117 #define SIMPLE_DESCRIPTOR_RESPONSE    0x8004
02118
02119
02128 #define ACTIVE_ENDPOINTS_REQUEST      0x0005
02129 #define ACTIVE_ENDPOINTS_RESPONSE     0x8005
02130
02131
02143 #define MATCH_DESCRIPTORS_REQUEST     0x0006
02144 #define MATCH_DESCRIPTORS_RESPONSE    0x8006
02145
02146
02156 #define DISCOVERY_CACHE_REQUEST       0x0012
02157 #define DISCOVERY_CACHE_RESPONSE      0x8012
02158
02159
02168 #define END_DEVICE_ANNOUNCE           0x0013
02169 #define END_DEVICE_ANNOUNCE_RESPONSE  0x8013
02170
02171
02183 #define SYSTEM_SERVER_DISCOVERY_REQUEST  0x0015
02184 #define SYSTEM_SERVER_DISCOVERY_RESPONSE 0x8015
02185
02186
02191 #ifdef DOXYGEN_SHOULD_SKIP_THIS
02192 enum EmberZdoServerMask
02193 #else
02194 typedef int16u EmberZdoServerMask;
02195 enum
02196 #endif
02197 {
02198   EMBER_ZDP_PRIMARY_TRUST_CENTER          =
      0x0001,
02199   EMBER_ZDP_SECONDARY_TRUST_CENTER        =
      0x0002,
02200   EMBER_ZDP_PRIMARY_BINDING_TABLE_CACHE
      = 0x0004,
02201   EMBER_ZDP_SECONDARY_BINDING_TABLE_CACHE
       = 0x0008,
02202   EMBER_ZDP_PRIMARY_DISCOVERY_CACHE       =
      0x0010,
02203   EMBER_ZDP_SECONDARY_DISCOVERY_CACHE     =
      0x0020,
02204   EMBER_ZDP_NETWORK_MANAGER              = 0x0040,
02205   // Bits 0x0080 to 0x8000 are reserved.
02206 };
02207
02221 #define FIND_NODE_CACHE_REQUEST        0x001C
02222 #define FIND_NODE_CACHE_RESPONSE       0x801C
02223
02224
02235 #define END_DEVICE_BIND_REQUEST      0x0020
02236 #define END_DEVICE_BIND_RESPONSE     0x8020
02237
02238
02256 #define UNICAST_BINDING            0x03
02257 #define UNICAST_MANY_TO_ONE_BINDING 0x83
02258 #define MULTICAST_BINDING          0x01
02259
02260 #define BIND_REQUEST                 0x0021
02261 #define BIND_RESPONSE                0x8021
02262 #define UNBIND_REQUEST               0x0022
02263 #define UNBIND_RESPONSE              0x8022
02264
02265
02313 #define LQI_TABLE_REQUEST            0x0031
02314 #define LQI_TABLE_RESPONSE           0x8031
02315
02316
02349 #define ROUTING_TABLE_REQUEST        0x0032
02350 #define ROUTING_TABLE_RESPONSE       0x8032
```

```
02351
02352
02371 #define BINDING_TABLE_REQUEST          0x0033
02372 #define BINDING_TABLE_RESPONSE         0x8033
02373
02374
02385 #define LEAVE_REQUEST                  0x0034
02386 #define LEAVE_RESPONSE                 0x8034
02387
02388 #define LEAVE_REQUEST_REMOVE_CHILDREN_FLAG 0x40
02389 #define LEAVE_REQUEST_REJOIN_FLAG            0x80
02390
02391
02400 #define PERMIT_JOINING_REQUEST         0x0036
02401 #define PERMIT_JOINING_RESPONSE        0x8036
02402
02403
02429 #define NWK_UPDATE_REQUEST             0x0038
02430 #define NWK_UPDATE_RESPONSE            0x8038
02431
02432
02436 #define COMPLEX_DESCRIPTOR_REQUEST     0x0010
02437 #define COMPLEX_DESCRIPTOR_RESPONSE    0x8010
02438 #define USER_DESCRIPTOR_REQUEST        0x0011
02439 #define USER_DESCRIPTOR_RESPONSE       0x8011
02440 #define DISCOVERY_REGISTER_REQUEST     0x0012
02441 #define DISCOVERY_REGISTER_RESPONSE    0x8012
02442 #define USER_DESCRIPTOR_SET            0x0014
02443 #define USER_DESCRIPTOR_CONFIRM        0x8014
02444 #define NETWORK_DISCOVERY_REQUEST      0x0030
02445 #define NETWORK_DISCOVERY_RESPONSE     0x8030
02446 #define DIRECT_JOIN_REQUEST            0x0035
02447 #define DIRECT_JOIN_RESPONSE           0x8035
02448
02449
02450 #define CLUSTER_ID_RESPONSE_MINIMUM  0x8000
02451
02452
02465 #ifdef DOXYGEN_SHOULD_SKIP_THIS
02466 enum EmberZdoConfigurationFlags
02467 #else
02468 typedef int8u EmberZdoConfigurationFlags;
02469 enum
02470 #endif
02471
02472 {
02473  EMBER_APP_RECEIVES_SUPPORTED_ZDO_REQUESTS
       = 0x01,
02474  EMBER_APP_HANDLES_UNSUPPORTED_ZDO_REQUESTS
       = 0x02,
02475  EMBER_APP_HANDLES_ZDO_ENDPOINT_REQUESTS
        = 0x04,
02476  EMBER_APP_HANDLES_ZDO_BINDING_REQUESTS
       = 0x08
02477 };
02478
02480
02481 #endif // EMBER_TYPES_H
02482
```

8.47 ember.h File Reference

```
#include "ember-types.h"
```

```
#include "stack-info.h"
#include "network-formation.h"
#include "packet-buffer.h"
#include "message.h"
#include "raw-message.h"
#include "child.h"
#include "security.h"
#include "aes-mmo.h"
#include "binding-table.h"
#include "bootload.h"
#include "error.h"
#include "zigbee-device-stack.h"
#include "event.h"
#include "ember-debug.h"
#include "library.h"
#include "multi-network.h"
```

PHY Information

Bit masks for TOKEN_MFG_RADIO_BANDS_SUPPORTED.

- #define RADIO_BANDS_SUPPORTED_2400

8.47.1 Detailed Description

The master include file for the EmberZNet API. See EmberZNet Stack API Reference for documentation.

Definition in file ember.h.

8.47.2 Macro Definition Documentation

8.47.2.1 #define RADIO_BANDS_SUPPORTED_2400

2.4GHz band

Definition at line 69 of file ember.h.

8.48 ember.h

```
00001
00021 #ifndef __EMBER_H__
00022 #define __EMBER_H__
00023
00024 #include "ember-types.h"
00025 #include "stack-info.h"
00026 #include "network-formation.h"
00027 #include "packet-buffer.h"
00028 #include "message.h"
00029 #include "raw-message.h"
00030 #include "child.h"
00031 #include "security.h"
00032 #include "aes-mmo.h"
00033 #include "binding-table.h"
00034 #include "bootload.h"
00035 #include "error.h"
```

```
00036 #include "zigbee-device-stack.h"
00037 #include "event.h"
00038 #include "ember-debug.h"
00039 #include "library.h"
00040 #include "multi-network.h"
00041
00042 #if defined(EMBER_ZLL_STACK)
00043   #ifndef EMBER_SCRIPTED_TEST
00044     #define EMBER_PARAMETERIZED_BROADCAST_TABLE
00045   #endif
00046   #include "zll-types.h"
00047   #include "zll-api.h"
00048 #endif
00049
00050 // We strip the multi-network code for flash saving purposes on the EM2xx and
00051 // EM351 platforms.
00052 #ifndef EMBER_MULTI_NETWORK_STRIPPED
00053 #if defined(XAP2B) || defined(CORTEXM3_EM351)
00054 #define EMBER_MULTI_NETWORK_STRIPPED
00055 #endif // defined(XAP2B) || defined(CORTEXM3_EM351)
00056 #endif // EMBER_MULTI_NETWORK_STRIPPED
00057
00062 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00063 #define RADIO_BANDS_SUPPORTED_868   BIT(0)
00064 #define RADIO_BANDS_SUPPORTED_915   BIT(1)
00065 #define RADIO_BANDS_SUPPORTED_433   BIT(2)
00066 #endif // DOXYGEN_SHOULD_SKIP_THIS
00067
00069 #define RADIO_BANDS_SUPPORTED_2400  BIT(3)
00070
00071 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00072 #define RADIO_BANDS_SUPPORTED_408   BIT(4)
00073 #endif // DOXYGEN_SHOULD_SKIP_THIS
00074
00076
00077 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00078
00081 #include "config/config.h"
00083
00084 #ifdef DEBUG_ASSERT
00085 extern boolean enableFailure;
00086 extern int8u rateOfFailure;
00087 extern int8u failuresInARow;
00088 static int8u bufferFailure;
00089 boolean generateFailure(void);
00090 void dumpFailure(void);
00091 #endif
00092
00093 #endif //DOXYGEN_SHOULD_SKIP_THIS
00094
00095 #endif // __EMBER_H__
00096
```

8.49 endian.h File Reference

Functions

- int16u NTOHS (int16u val)
- int32u NTOHL (int32u val)

8.49.1 Detailed Description

See Network to Host Byte Order Conversion for detailed documentation.

Definition in file endian.h.

8.50   endian.h

```
00001
00007 #ifndef __ENDIAN_H__
00008 #define __ENDIAN_H__
00009
00019 #if BIGENDIAN_CPU == FALSE
00020
00024   #ifndef NTOHS  // some platforms already define this
00025     int16u NTOHS(int16u val);
00026   #endif
00027
00031   #ifndef NTOHL  // some platforms already define this
00032     int32u NTOHL(int32u val);
00033   #endif
00034
00035 #else // BIGENDIAN_CPU == TRUE
00036
00037   #ifndef NTOHS  // some platforms already define this
00038     #define NTOHS(val)  (val)
00039   #endif
00040   #ifndef NTOHL  // some platforms already define this
00041     #define NTOHL(val)  (val)
00042   #endif
00043
00044 #endif
00045
00049 #endif //__ENDIAN_H__
00050
```

8.51   error-def.h File Reference

Generic Messages

These messages are system wide.

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

Packet Buffer Module Errors

- #define EMBER_NO_BUFFERS(x18)

Serial Manager Errors

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

MAC Errors

- #define EMBER_MAC_TRANSMIT_QUEUE_FULL(x39)
- #define EMBER_MAC_UNKNOWN_HEADER_TYPE(x3A)
- #define EMBER_MAC_ACK_HEADER_TYPE(x3B)
- #define EMBER_MAC_SCANNING(x3D)
- #define EMBER_MAC_NO_DATA(x31)
- #define EMBER_MAC_JOINED_NETWORK(x32)
- #define EMBER_MAC_BAD_SCAN_DURATION(x33)
- #define EMBER_MAC_INCORRECT_SCAN_TYPE(x34)
- #define EMBER_MAC_INVALID_CHANNEL_MASK(x35)
- #define EMBER_MAC_COMMAND_TRANSMIT_FAILURE(x36)
- #define EMBER_MAC_NO_ACK_RECEIVED(x40)
- #define EMBER_MAC_RADIO_NETWORK_SWITCH_FAILED(x41)
- #define EMBER_MAC_INDIRECT_TIMEOUT(x42)

Simulated EEPROM Errors

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

Flash Errors

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

Bootloader Errors

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

Transport Errors

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

HAL Module Errors

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

PHY Errors

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

Return Codes Passed to emberStackStatusHandler()

See also emberStackStatusHandler().

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

Security Errors

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

Miscellaneous Network Errors

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

Miscellaneous Utility Errors

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

Application Errors

These error codes are available for application use.

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

8.51.1 Detailed Description

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

Definition in file error-def.h.

8.52   error-def.h

```
00001
00038
00039 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00040
00043 #define EMBER_SUCCESS(0x00)
00044 #else
00045 DEFINE_ERROR(SUCCESS, 0)
00046 #endif //DOXYGEN_SHOULD_SKIP_THIS
00047
00048
00049 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00050
00053 #define EMBER_ERR_FATAL(0x01)
00054 #else
00055 DEFINE_ERROR(ERR_FATAL, 0x01)
00056 #endif //DOXYGEN_SHOULD_SKIP_THIS
00057
00058
00059 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00060
00063 #define EMBER_BAD_ARGUMENT(0x02)
00064 #else
00065 DEFINE_ERROR(BAD_ARGUMENT, 0x02)
00066 #endif //DOXYGEN_SHOULD_SKIP_THIS
00067
00068
00069 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00070
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
00343 #define EMBER_MAC_RADIO_NETWORK_SWITCH_FAILED(0x41)
00344 #else
00345 DEFINE_ERROR(MAC_RADIO_NETWORK_SWITCH_FAILED, 0x41)
00346 #endif //DOXYGEN_SHOULD_SKIP_THIS
00347
00348
00349 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00350
00353 #define EMBER_MAC_INDIRECT_TIMEOUT(0x42)
00354 #else
00355 DEFINE_ERROR(MAC_INDIRECT_TIMEOUT, 0x42)
00356 #endif //DOXYGEN_SHOULD_SKIP_THIS
```

```
00357
00359
00360
00365
00366
00367 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00368
00376 #define EMBER_SIM_EEPROM_ERASE_PAGE_GREEN(0x43)
00377 #else
00378 DEFINE_ERROR(SIM_EEPROM_ERASE_PAGE_GREEN, 0x43)
00379 #endif //DOXYGEN_SHOULD_SKIP_THIS
00380
00381
00382 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00383
00392 #define EMBER_SIM_EEPROM_ERASE_PAGE_RED(0x44)
00393 #else
00394 DEFINE_ERROR(SIM_EEPROM_ERASE_PAGE_RED, 0x44)
00395 #endif //DOXYGEN_SHOULD_SKIP_THIS
00396
00397
00398 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00399
00407 #define EMBER_SIM_EEPROM_FULL(0x45)
00408 #else
00409 DEFINE_ERROR(SIM_EEPROM_FULL, 0x45)
00410 #endif //DOXYGEN_SHOULD_SKIP_THIS
00411
00412
00413 //  Errors 46 and 47 are now defined below in the
00414 //    flash error block (was attempting to prevent renumbering)
00415
00416
00417 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00418
00425 #define EMBER_SIM_EEPROM_INIT_1_FAILED(0x48)
00426 #else
00427 DEFINE_ERROR(SIM_EEPROM_INIT_1_FAILED, 0x48)
00428 #endif //DOXYGEN_SHOULD_SKIP_THIS
00429
00430
00431 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00432
00438 #define EMBER_SIM_EEPROM_INIT_2_FAILED(0x49)
00439 #else
00440 DEFINE_ERROR(SIM_EEPROM_INIT_2_FAILED, 0x49)
00441 #endif //DOXYGEN_SHOULD_SKIP_THIS
00442
00443
00444 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00445
00452 #define EMBER_SIM_EEPROM_INIT_3_FAILED(0x4A)
00453 #else
00454 DEFINE_ERROR(SIM_EEPROM_INIT_3_FAILED, 0x4A)
00455 #endif //DOXYGEN_SHOULD_SKIP_THIS
00456
00457
00458 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00459
00470 #define EMBER_SIM_EEPROM_REPAIRING(0x4D)
00471 #else
00472 DEFINE_ERROR(SIM_EEPROM_REPAIRING, 0x4D)
00473 #endif //DOXYGEN_SHOULD_SKIP_THIS
00474
00476
00477
00482
00483 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00484
00491 #define EMBER_ERR_FLASH_WRITE_INHIBITED(0x46)
00492 #else
00493 DEFINE_ERROR(ERR_FLASH_WRITE_INHIBITED, 0x46)
00494 #endif //DOXYGEN_SHOULD_SKIP_THIS
00495
00496
00497 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00498
00504 #define EMBER_ERR_FLASH_VERIFY_FAILED(0x47)
00505 #else
00506 DEFINE_ERROR(ERR_FLASH_VERIFY_FAILED, 0x47)
```

```
00507 #endif //DOXYGEN_SHOULD_SKIP_THIS
00508
00509
00510 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00511
00517 #define EMBER_ERR_FLASH_PROG_FAIL(0x4B)
00518 #else
00519 DEFINE_ERROR(ERR_FLASH_PROG_FAIL, 0x4B)
00520 #endif //DOXYGEN_SHOULD_SKIP_THIS
00521
00522
00523 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00524
00530 #define EMBER_ERR_FLASH_ERASE_FAIL(0x4C)
00531 #else
00532 DEFINE_ERROR(ERR_FLASH_ERASE_FAIL, 0x4C)
00533 #endif //DOXYGEN_SHOULD_SKIP_THIS
00534
00536
00537
00542
00543
00544 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00545
00549 #define EMBER_ERR_BOOTLOADER_TRAP_TABLE_BAD(0x58)
00550 #else
00551 DEFINE_ERROR(ERR_BOOTLOADER_TRAP_TABLE_BAD, 0x58)
00552 #endif //DOXYGEN_SHOULD_SKIP_THIS
00553
00554
00555 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00556
00560 #define EMBER_ERR_BOOTLOADER_TRAP_UNKNOWN(0x59)
00561 #else
00562 DEFINE_ERROR(ERR_BOOTLOADER_TRAP_UNKNOWN, 0x59)
00563 #endif //DOXYGEN_SHOULD_SKIP_THIS
00564
00565
00566 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00567
00571 #define EMBER_ERR_BOOTLOADER_NO_IMAGE(0x05A)
00572 #else
00573 DEFINE_ERROR(ERR_BOOTLOADER_NO_IMAGE, 0x5A)
00574 #endif //DOXYGEN_SHOULD_SKIP_THIS
00575
00577
00578
00583
00584 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00585
00589 #define EMBER_DELIVERY_FAILED(0x66)
00590 #else
00591 DEFINE_ERROR(DELIVERY_FAILED, 0x66)
00592 #endif //DOXYGEN_SHOULD_SKIP_THIS
00593
00594
00595 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00596
00599 #define EMBER_BINDING_INDEX_OUT_OF_RANGE(0x69)
00600 #else
00601 DEFINE_ERROR(BINDING_INDEX_OUT_OF_RANGE, 0x69)
00602 #endif //DOXYGEN_SHOULD_SKIP_THIS
00603
00604
00605 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00606
00610 #define EMBER_ADDRESS_TABLE_INDEX_OUT_OF_RANGE(0x6A)
00611 #else
00612 DEFINE_ERROR(ADDRESS_TABLE_INDEX_OUT_OF_RANGE, 0x6A)
00613 #endif //DOXYGEN_SHOULD_SKIP_THIS
00614
00615
00616 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00617
00620 #define EMBER_INVALID_BINDING_INDEX(0x6C)
00621 #else
00622 DEFINE_ERROR(INVALID_BINDING_INDEX, 0x6C)
00623 #endif //DOXYGEN_SHOULD_SKIP_THIS
00624
00625
```

```
00626 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00627
00631 #define EMBER_INVALID_CALL(0x70)
00632 #else
00633 DEFINE_ERROR(INVALID_CALL, 0x70)
00634 #endif //DOXYGEN_SHOULD_SKIP_THIS
00635
00636
00637 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00638
00641 #define EMBER_COST_NOT_KNOWN(0x71)
00642 #else
00643 DEFINE_ERROR(COST_NOT_KNOWN, 0x71)
00644 #endif //DOXYGEN_SHOULD_SKIP_THIS
00645
00646
00647 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00648
00652 #define EMBER_MAX_MESSAGE_LIMIT_REACHED(0x72)
00653 #else
00654 DEFINE_ERROR(MAX_MESSAGE_LIMIT_REACHED, 0x72)
00655 #endif //DOXYGEN_SHOULD_SKIP_THIS
00656
00657 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00658
00662 #define EMBER_MESSAGE_TOO_LONG(0x74)
00663 #else
00664 DEFINE_ERROR(MESSAGE_TOO_LONG, 0x74)
00665 #endif //DOXYGEN_SHOULD_SKIP_THIS
00666
00667
00668 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00669
00673 #define EMBER_BINDING_IS_ACTIVE(0x75)
00674 #else
00675 DEFINE_ERROR(BINDING_IS_ACTIVE, 0x75)
00676 #endif //DOXYGEN_SHOULD_SKIP_THIS
00677
00678 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00679
00683 #define EMBER_ADDRESS_TABLE_ENTRY_IS_ACTIVE(0x76)
00684 #else
00685 DEFINE_ERROR(ADDRESS_TABLE_ENTRY_IS_ACTIVE, 0x76)
00686 #endif //DOXYGEN_SHOULD_SKIP_THIS
00687
00689
00694
00695
00696 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00697
00700 #define EMBER_ADC_CONVERSION_DONE(0x80)
00701 #else
00702 DEFINE_ERROR(ADC_CONVERSION_DONE, 0x80)
00703 #endif //DOXYGEN_SHOULD_SKIP_THIS
00704
00705
00706 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00707
00711 #define EMBER_ADC_CONVERSION_BUSY(0x81)
00712 #else
00713 DEFINE_ERROR(ADC_CONVERSION_BUSY, 0x81)
00714 #endif //DOXYGEN_SHOULD_SKIP_THIS
00715
00716
00717 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00718
00722 #define EMBER_ADC_CONVERSION_DEFERRED(0x82)
00723 #else
00724 DEFINE_ERROR(ADC_CONVERSION_DEFERRED, 0x82)
00725 #endif //DOXYGEN_SHOULD_SKIP_THIS
00726
00727
00728 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00729
00732 #define EMBER_ADC_NO_CONVERSION_PENDING(0x84)
00733 #else
00734 DEFINE_ERROR(ADC_NO_CONVERSION_PENDING, 0x84)
00735 #endif //DOXYGEN_SHOULD_SKIP_THIS
00736
00737
```

```
00738 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00739
00743 #define EMBER_SLEEP_INTERRUPTED(0x85)
00744 #else
00745 DEFINE_ERROR(SLEEP_INTERRUPTED, 0x85)
00746 #endif //DOXYGEN_SHOULD_SKIP_THIS
00747
00749
00754
00755
00756 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00757
00760 #define EMBER_PHY_TX_UNDERFLOW(0x88)
00761 #else
00762 DEFINE_ERROR(PHY_TX_UNDERFLOW, 0x88)
00763 #endif //DOXYGEN_SHOULD_SKIP_THIS
00764
00765
00766 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00767
00770 #define EMBER_PHY_TX_INCOMPLETE(0x89)
00771 #else
00772 DEFINE_ERROR(PHY_TX_INCOMPLETE, 0x89)
00773 #endif //DOXYGEN_SHOULD_SKIP_THIS
00774
00775
00776 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00777
00780 #define EMBER_PHY_INVALID_CHANNEL(0x8A)
00781 #else
00782 DEFINE_ERROR(PHY_INVALID_CHANNEL, 0x8A)
00783 #endif //DOXYGEN_SHOULD_SKIP_THIS
00784
00785
00786 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00787
00790 #define EMBER_PHY_INVALID_POWER(0x8B)
00791 #else
00792 DEFINE_ERROR(PHY_INVALID_POWER, 0x8B)
00793 #endif //DOXYGEN_SHOULD_SKIP_THIS
00794
00795
00796 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00797
00801 #define EMBER_PHY_TX_BUSY(0x8C)
00802 #else
00803 DEFINE_ERROR(PHY_TX_BUSY, 0x8C)
00804 #endif //DOXYGEN_SHOULD_SKIP_THIS
00805
00806
00807 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00808
00812 #define EMBER_PHY_TX_CCA_FAIL(0x8D)
00813 #else
00814 DEFINE_ERROR(PHY_TX_CCA_FAIL, 0x8D)
00815 #endif //DOXYGEN_SHOULD_SKIP_THIS
00816
00817
00818 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00819
00823 #define EMBER_PHY_OSCILLATOR_CHECK_FAILED(0x8E)
00824 #else
00825 DEFINE_ERROR(PHY_OSCILLATOR_CHECK_FAILED, 0x8E)
00826 #endif //DOXYGEN_SHOULD_SKIP_THIS
00827
00828
00829 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00830
00833 #define EMBER_PHY_ACK_RECEIVED(0x8F)
00834 #else
00835 DEFINE_ERROR(PHY_ACK_RECEIVED, 0x8F)
00836 #endif //DOXYGEN_SHOULD_SKIP_THIS
00837
00839
00845
00846
00847 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00848
00852 #define EMBER_NETWORK_UP(0x90)
00853 #else
```

```
00854 DEFINE_ERROR(NETWORK_UP, 0x90)
00855 #endif //DOXYGEN_SHOULD_SKIP_THIS
00856
00857
00858 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00859
00862 #define EMBER_NETWORK_DOWN(0x91)
00863 #else
00864 DEFINE_ERROR(NETWORK_DOWN, 0x91)
00865 #endif //DOXYGEN_SHOULD_SKIP_THIS
00866
00867
00868 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00869
00872 #define EMBER_JOIN_FAILED(0x94)
00873 #else
00874 DEFINE_ERROR(JOIN_FAILED, 0x94)
00875 #endif //DOXYGEN_SHOULD_SKIP_THIS
00876
00877
00878 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00879
00883 #define EMBER_MOVE_FAILED(0x96)
00884 #else
00885 DEFINE_ERROR(MOVE_FAILED, 0x96)
00886 #endif //DOXYGEN_SHOULD_SKIP_THIS
00887
00888
00889 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00890
00895 #define EMBER_CANNOT_JOIN_AS_ROUTER(0x98)
00896 #else
00897 DEFINE_ERROR(CANNOT_JOIN_AS_ROUTER, 0x98)
00898 #endif //DOXYGEN_SHOULD_SKIP_THIS
00899
00900
00901 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00902
00905 #define EMBER_NODE_ID_CHANGED(0x99)
00906 #else
00907 DEFINE_ERROR(NODE_ID_CHANGED, 0x99)
00908 #endif
00909
00910
00911 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00912
00915 #define EMBER_PAN_ID_CHANGED(0x9A)
00916 #else
00917 DEFINE_ERROR(PAN_ID_CHANGED, 0x9A)
00918 #endif
00919
00920 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00921
00923 #define EMBER_CHANNEL_CHANGED(0x9B)
00924 #else
00925 DEFINE_ERROR(CHANNEL_CHANGED, 0x9B)
00926 #endif
00927
00928 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00929
00932 #define EMBER_NO_BEACONS(0xAB)
00933 #else
00934 DEFINE_ERROR(NO_BEACONS, 0xAB)
00935 #endif
00936
00937
00938 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00939
00943 #define EMBER_RECEIVED_KEY_IN_THE_CLEAR(0xAC)
00944 #else
00945 DEFINE_ERROR(RECEIVED_KEY_IN_THE_CLEAR, 0xAC)
00946 #endif
00947
00948
00949 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00950
00953 #define EMBER_NO_NETWORK_KEY_RECEIVED(0xAD)
00954 #else
00955 DEFINE_ERROR(NO_NETWORK_KEY_RECEIVED, 0xAD)
00956 #endif
```

```
00957
00958
00959 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00960
00963 #define EMBER_NO_LINK_KEY_RECEIVED(0xAE)
00964 #else
00965 DEFINE_ERROR(NO_LINK_KEY_RECEIVED, 0xAE)
00966 #endif
00967
00968
00969 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00970
00974 #define EMBER_PRECONFIGURED_KEY_REQUIRED(0xAF)
00975 #else
00976 DEFINE_ERROR(PRECONFIGURED_KEY_REQUIRED, 0xAF)
00977 #endif
00978
00979
00981
00985 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00986
00990 #define EMBER_KEY_INVALID(0xB2)
00991 #else
00992 DEFINE_ERROR(KEY_INVALID, 0xB2)
00993 #endif // DOXYGEN_SHOULD_SKIP_THIS
00994
00995 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00996
01000 #define EMBER_INVALID_SECURITY_LEVEL(0x95)
01001 #else
01002 DEFINE_ERROR(INVALID_SECURITY_LEVEL, 0x95)
01003 #endif //DOXYGEN_SHOULD_SKIP_THIS
01004
01005 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01006
01014 #define EMBER_APS_ENCRYPTION_ERROR(0xA6)
01015 #else
01016     DEFINE_ERROR(APS_ENCRYPTION_ERROR, 0xA6)
01017 #endif //DOXYGEN_SHOULD_SKIP_THIS
01018
01019 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01020
01023 #define EMBER_TRUST_CENTER_MASTER_KEY_NOT_SET(0xA7)
01024 #else
01025     DEFINE_ERROR(TRUST_CENTER_MASTER_KEY_NOT_SET, 0xA7)
01026 #endif //DOXYGEN_SHOULD_SKIP_THIS
01027
01028 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01029
01032 #define EMBER_SECURITY_STATE_NOT_SET(0xA8)
01033 #else
01034     DEFINE_ERROR(SECURITY_STATE_NOT_SET, 0xA8)
01035 #endif //DOXYGEN_SHOULD_SKIP_THIS
01036
01037 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01038
01045 #define EMBER_KEY_TABLE_INVALID_ADDRESS(0xB3)
01046 #else
01047 DEFINE_ERROR(KEY_TABLE_INVALID_ADDRESS, 0xB3)
01048 #endif //DOXYGEN_SHOULD_SKIP_THIS
01049
01050 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01051
01054 #define EMBER_SECURITY_CONFIGURATION_INVALID(0xB7)
01055 #else
01056 DEFINE_ERROR(SECURITY_CONFIGURATION_INVALID, 0xB7)
01057 #endif //DOXYGEN_SHOULD_SKIP_THIS
01058
01059 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01060
01065 #define EMBER_TOO_SOON_FOR_SWITCH_KEY(0xB8)
01066 #else
01067     DEFINE_ERROR(TOO_SOON_FOR_SWITCH_KEY, 0xB8)
01068 #endif
01069
01070 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01071
01074 #define EMBER_SIGNATURE_VERIFY_FAILURE(0xB9)
01075 #else
01076     DEFINE_ERROR(SIGNATURE_VERIFY_FAILURE, 0xB9)
```

```
01077 #endif
01078
01079 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01080
01086 #define EMBER_KEY_NOT_AUTHORIZED(0xBB)
01087 #else
01088     DEFINE_ERROR(KEY_NOT_AUTHORIZED, 0xBB)
01089 #endif
01090
01091
01092 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01093
01096 #define EMBER_SECURITY_DATA_INVALID(0xBD)
01097 #else
01098     DEFINE_ERROR(SECURITY_DATA_INVALID, 0xBD)
01099 #endif
01100
01102
01103
01108
01109
01110 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01111
01114 #define EMBER_NOT_JOINED(0x93)
01115 #else
01116 DEFINE_ERROR(NOT_JOINED, 0x93)
01117 #endif //DOXYGEN_SHOULD_SKIP_THIS
01118
01119 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01120
01124 #define EMBER_NETWORK_BUSY(0xA1)
01125 #else
01126 DEFINE_ERROR(NETWORK_BUSY, 0xA1)
01127 #endif //DOXYGEN_SHOULD_SKIP_THIS
01128
01129
01130 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01131
01135 #define EMBER_INVALID_ENDPOINT(0xA3)
01136 #else
01137 DEFINE_ERROR(INVALID_ENDPOINT, 0xA3)
01138 #endif //DOXYGEN_SHOULD_SKIP_THIS
01139
01140
01141 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01142
01146 #define EMBER_BINDING_HAS_CHANGED(0xA4)
01147 #else
01148 DEFINE_ERROR(BINDING_HAS_CHANGED, 0xA4)
01149 #endif //DOXYGEN_SHOULD_SKIP_THIS
01150
01151 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01152
01156 #define EMBER_INSUFFICIENT_RANDOM_DATA(0xA5)
01157 #else
01158     DEFINE_ERROR(INSUFFICIENT_RANDOM_DATA, 0xA5)
01159 #endif //DOXYGEN_SHOULD_SKIP_THIS
01160
01161
01162 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01163
01166 #define EMBER_SOURCE_ROUTE_FAILURE(0xA9)
01167 #else
01168     DEFINE_ERROR(SOURCE_ROUTE_FAILURE, 0xA9)
01169 #endif
01170
01171 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01172
01177 #define EMBER_MANY_TO_ONE_ROUTE_FAILURE(0xAA)
01178 #else
01179     DEFINE_ERROR(MANY_TO_ONE_ROUTE_FAILURE, 0xAA)
01180 #endif
01181
01182
01184
01189
01190
01191 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01192
01198 #define EMBER_STACK_AND_HARDWARE_MISMATCH(0xB0)
```

```
01199 #else
01200 DEFINE_ERROR(STACK_AND_HARDWARE_MISMATCH, 0xB0)
01201 #endif //DOXYGEN_SHOULD_SKIP_THIS
01202
01203
01204 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01205
01209 #define EMBER_INDEX_OUT_OF_RANGE(0xB1)
01210 #else
01211 DEFINE_ERROR(INDEX_OUT_OF_RANGE, 0xB1)
01212 #endif
01213
01214 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01215
01218 #define EMBER_TABLE_FULL(0xB4)
01219 #else
01220 DEFINE_ERROR(TABLE_FULL, 0xB4)
01221 #endif //DOXYGEN_SHOULD_SKIP_THIS
01222
01223 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01224
01228 #define EMBER_TABLE_ENTRY_ERASED(0xB6)
01229 #else
01230 DEFINE_ERROR(TABLE_ENTRY_ERASED, 0xB6)
01231 #endif
01232
01233 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01234
01238 #define EMBER_LIBRARY_NOT_PRESENT(0xB5)
01239 #else
01240 DEFINE_ERROR(LIBRARY_NOT_PRESENT, 0xB5)
01241 #endif
01242
01243 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01244
01248 #define EMBER_OPERATION_IN_PROGRESS(0xBA)
01249 #else
01250 DEFINE_ERROR(OPERATION_IN_PROGRESS, 0xBA)
01251 #endif
01252
01253 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01254
01259 #define EMBER_TRUST_CENTER_EUI_HAS_CHANGED(0xBC)
01260 #else
01261     DEFINE_ERROR(TRUST_CENTER_EUI_HAS_CHANGED, 0xBC)
01262 #endif
01263
01265
01271
01272 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01273
01277 #define EMBER_APPLICATION_ERROR_0(0xF0)
01278 #define EMBER_APPLICATION_ERROR_1(0xF1)
01279 #define EMBER_APPLICATION_ERROR_2(0xF2)
01280 #define EMBER_APPLICATION_ERROR_3(0xF3)
01281 #define EMBER_APPLICATION_ERROR_4(0xF4)
01282 #define EMBER_APPLICATION_ERROR_5(0xF5)
01283 #define EMBER_APPLICATION_ERROR_6(0xF6)
01284 #define EMBER_APPLICATION_ERROR_7(0xF7)
01285 #define EMBER_APPLICATION_ERROR_8(0xF8)
01286 #define EMBER_APPLICATION_ERROR_9(0xF9)
01287 #define EMBER_APPLICATION_ERROR_10(0xFA)
01288 #define EMBER_APPLICATION_ERROR_11(0xFB)
01289 #define EMBER_APPLICATION_ERROR_12(0xFC)
01290 #define EMBER_APPLICATION_ERROR_13(0xFD)
01291 #define EMBER_APPLICATION_ERROR_14(0xFE)
01292 #define EMBER_APPLICATION_ERROR_15(0xFF)
01293 #else
01294 DEFINE_ERROR( APPLICATION_ERROR_0, 0xF0)
01295 DEFINE_ERROR( APPLICATION_ERROR_1, 0xF1)
01296 DEFINE_ERROR( APPLICATION_ERROR_2, 0xF2)
01297 DEFINE_ERROR( APPLICATION_ERROR_3, 0xF3)
01298 DEFINE_ERROR( APPLICATION_ERROR_4, 0xF4)
01299 DEFINE_ERROR( APPLICATION_ERROR_5, 0xF5)
01300 DEFINE_ERROR( APPLICATION_ERROR_6, 0xF6)
01301 DEFINE_ERROR( APPLICATION_ERROR_7, 0xF7)
01302 DEFINE_ERROR( APPLICATION_ERROR_8, 0xF8)
01303 DEFINE_ERROR( APPLICATION_ERROR_9, 0xF9)
01304 DEFINE_ERROR( APPLICATION_ERROR_10, 0xFA)
01305 DEFINE_ERROR( APPLICATION_ERROR_11, 0xFB)
```

```
01306 DEFINE_ERROR( APPLICATION_ERROR_12, 0xFC)
01307 DEFINE_ERROR( APPLICATION_ERROR_13, 0xFD)
01308 DEFINE_ERROR( APPLICATION_ERROR_14, 0xFE)
01309 DEFINE_ERROR( APPLICATION_ERROR_15, 0xFF)
01310 #endif //DOXYGEN_SHOULD_SKIP_THIS
01311
01313
```

8.53   error.h File Reference

Macros

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

Typedefs

- typedef int8u EmberStatus

Enumerations

- enum { EMBER_ERROR_CODE_COUNT }

8.53.1   Detailed Description

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

Definition in file error.h.

8.53.2   Macro Definition Documentation

8.53.2.1   #define __EMBERSTATUS_TYPE__

Return type for Ember functions.

Definition at line 18 of file error.h.

8.53.3   Typedef Documentation

8.53.3.1   typedef int8u EmberStatus

Definition at line 19 of file error.h.

8.54   error.h

```
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
```

8.55   event.h File Reference

Macros

- #define __EVENT_H__
- #define emberEventControlSetInactive(control)
- #define emberEventControlGetActive(control)
- #define emberEventControlSetActive(control)
- #define emberEventControlSetDelayMS(control, delay)
- #define emberEventControlSetDelayQS(control, delay)
- #define emberEventControlSetDelayMinutes(control, delay)
- #define emberEventControlGetRemainingMS(control)
- #define emberTaskEnableIdling(allow)
- #define emberMarkTaskActive(taskid)

Functions

- void emEventControlSetActive (EmberEventControl ∗event)
- void emEventControlSetDelayMS (EmberEventControl ∗event, int16u delay)
- void emEventControlSetDelayQS (EmberEventControl ∗event, int16u delay)
- void emEventControlSetDelayMinutes (EmberEventControl ∗event, int16u delay)
- int32u emEventControlGetRemainingMS (EmberEventControl ∗event)
- void emberRunEvents (EmberEventData ∗events)
- void emberRunTask (EmberTaskId taskid)
- int32u emberMsToNextEvent (EmberEventData ∗events, int32u maxMs)
- int32u emberMsToNextEventExtended (EmberEventData ∗events, int32u maxMs, int8u ∗return-Index)
- int32u emberMsToNextStackEvent (void)
- EmberTaskId emberTaskInit (EmberEventData ∗events)
- boolean emberMarkTaskIdle (EmberTaskId taskid)
- void emTaskEnableIdling (boolean allow)
- void emMarkTaskActive (EmberTaskId taskid)

8.55.1 Detailed Description

Scheduling events for future execution. See Event Scheduling for documentation.

Definition in file event.h.

8.56 event.h

```
00001
00009 #ifdef XAP2B
00010   // The xap2b platform does not support processor idling
00011   #define EMBER_NO_IDLE_SUPPORT
00012 #endif
00013
00100 // Controlling events
00101
00102 // Possible event status values.  Having zero as the 'inactive' value
00103 // causes events to initially be inactive.
00104 //
00105 #ifndef __EVENT_H__
00106 #define __EVENT_H__
00107
00110 #define emberEventControlSetInactive(control)    \
00111   do { (control).status = EMBER_EVENT_INACTIVE; } while(0)
00112
00115 #define emberEventControlGetActive(control)      \
00116   ((control).status != EMBER_EVENT_INACTIVE)
00117
00121 #define emberEventControlSetActive(control)              \
00122   do { emEventControlSetActive(&(control)); } while(0)
00123
00127 void emEventControlSetActive(EmberEventControl
      *event);
00128
00131 #define emberEventControlSetDelayMS(control, delay)        \
00132   do { emEventControlSetDelayMS(&(control), (delay)); } while(0)
00133
00136 void emEventControlSetDelayMS(EmberEventControl
      *event, int16u delay);
00137
00141 #define emberEventControlSetDelayQS(control, delay)           \
00142   do { emEventControlSetDelayQS(&(control), (delay)); } while(0)
00143
00147 void emEventControlSetDelayQS(EmberEventControl
      *event, int16u delay);
00148
00152 #define emberEventControlSetDelayMinutes(control, delay)       \
00153   do { emEventControlSetDelayMinutes(&(control), (delay)); } while(0)
00154
00158 void emEventControlSetDelayMinutes(
      EmberEventControl*event, int16u delay);
00159
00163 #define emberEventControlGetRemainingMS(control)        \
00164   (emEventControlGetRemainingMS(&(control)))
00165
00169 int32u emEventControlGetRemainingMS(
      EmberEventControl *event);
00170
00171
00172 // Running events
00173
00180 void emberRunEvents(EmberEventData *events);
00181
00186 void emberRunTask(EmberTaskId taskid);
00187
00195 int32u emberMsToNextEvent(EmberEventData
      *events, int32u maxMs);
00196
00202 int32u emberMsToNextEventExtended(
      EmberEventData *events, int32u maxMs, int8u* returnIndex);
00203
00207 int32u emberMsToNextStackEvent(void);
00208
00209
00214 EmberTaskId emberTaskInit(EmberEventData
```

```
         *events);
00215
00224 boolean emberMarkTaskIdle(EmberTaskId taskid);
00225
00226 #ifndef EMBER_NO_IDLE_SUPPORT
00227
00229   #define emberTaskEnableIdling(allow)  \
00230     do { emTaskEnableIdling((allow)); } while(0)
00231
00232   void emTaskEnableIdling(boolean allow);
00233
00237   #define emberMarkTaskActive(taskid)  \
00238     do { emMarkTaskActive((taskid)); } while(0)
00239
00240   void emMarkTaskActive(EmberTaskId taskid);
00241 #else
00242   #define emberTaskEnableIdling(allow)  do {} while(0)
00243   #define emberMarkTaskActive(taskid)   do {} while(0)
00244 #endif // EMBER_NO_IDLE_SUPPORT
00245
00246 #endif // __EVENT_H__
00247
00248 // @} END addtogroup
00249
00250
```

8.57   flash.h File Reference

```
#include "memmap.h"
```

Functions

- boolean halFlashEraseIsActive (void)

8.57.1   Detailed Description

See Flash Memory Control for documentation.

Definition in file flash.h.

8.58   flash.h

```
00001
00022 #ifndef __FLASH_H__
00023 #define __FLASH_H__
00024
00025 #include "memmap.h"
00026
00027
00037 boolean halFlashEraseIsActive(void);
00038
00039 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00040
00041 //[[ The following eraseType definitions must match the FIB erase types! ]]
00045 #define MFB_MASS_ERASE 0x01
00046 #define MFB_PAGE_ERASE 0x02
00047 #define CIB_ERASE      0x03
00048
00068 EmberStatus halInternalFlashErase(int8u eraseType, int32u
     address);
00069
00099 EmberStatus halInternalFlashWrite(int32u address, int16u
     * data, int32u length);
00100
```

```
00119 EmberStatus halInternalCibOptionByteWrite(int8u byte, int8u
      data);
00120
00121 #endif //DOXYGEN_SHOULD_SKIP_THIS
00122
00123 #endif //__FLASH_H__
00124
```

8.59   form-and-join.h File Reference

Macros

- #define NETWORK_STORAGE_SIZE
- #define NETWORK_STORAGE_SIZE_SHIFT
- #define FORM_AND_JOIN_MAX_NETWORKS

Functions

- EmberStatus emberScanForUnusedPanId (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)
- void emberFormAndJoinCleanup (EmberStatus status)

Variables

- boolean emberEnableDualChannelScan

8.59.1   Detailed Description

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

Definition in file form-and-join.h.

8.60   form-and-join.h

```
00001
00068 #define NETWORK_STORAGE_SIZE  16
00069
00072 #define NETWORK_STORAGE_SIZE_SHIFT 4
```

```
00073
00087 #ifndef FORM_AND_JOIN_MAX_NETWORKS
00088   #ifdef EZSP_HOST
00089     // the host's buffer is 16-bit array, so translate to bytes for comparison
00090     #define FORM_AND_JOIN_MAX_NETWORKS  \
00091       (EZSP_HOST_FORM_AND_JOIN_BUFFER_SIZE * 2 / NETWORK_STORAGE_SIZE)
00092   #else
00093     // use highest value that won't exceed max EmberMessageBuffer length
00094     #define FORM_AND_JOIN_MAX_NETWORKS 15
00095   #endif
00096 #endif
00097
00098 // Check that this value isn't too large for the SoC implementation to handle
00099 #ifndef EZSP_HOST
00100   #if (FORM_AND_JOIN_MAX_NETWORKS > 15)
00101     #error "FORM_AND_JOIN_MAX_NETWORKS can't exceed 15 on SoC platform"
00102   #endif
00103 #endif
00104
00121 EmberStatus emberScanForUnusedPanId(int32u
       channelMask, int8u duration);
00122
00149 EmberStatus emberScanForJoinableNetwork(
      int32u channelMask, int8u* extendedPanId);
00150
00152 EmberStatus emberScanForNextJoinableNetwork
      (void);
00153
00169 extern boolean emberEnableDualChannelScan;
00170
00175 boolean emberFormAndJoinIsScanning(void);
00176
00177 //
      -----------------------------------------------------------------------------
00178 // Callbacks the application needs to implement.
00179
00188 void emberUnusedPanIdFoundHandler(EmberPanId
       panId, int8u channel);
00189
00200 void emberJoinableNetworkFoundHandler(
      EmberZigbeeNetwork *networkFound,
00201                                     int8u lqi,
00202                                     int8s rssi);
00203
00221 void emberScanErrorHandler(EmberStatus status);
00222
00223 //
      -----------------------------------------------------------------------------
00224 // Library functions the application must call from within the
00225 // corresponding EmberZNet or EZSP callback.
00226
00234 boolean emberFormAndJoinScanCompleteHandler(
      int8u channel, EmberStatus status);
00235
00243 boolean emberFormAndJoinNetworkFoundHandler(
      EmberZigbeeNetwork *networkFound,
00244                                     int8u lqi,
00245                                     int8s rssi);
00246
00254 boolean emberFormAndJoinEnergyScanResultHandler
      (int8u channel, int8s maxRssiValue);
00255
00260 void emberFormAndJoinTick(void);
00261
00265 void emberFormAndJoinTaskInit(void);
00266
00270 void emberFormAndJoinRunTask(void);
00271
00276 void emberFormAndJoinCleanup(EmberStatus
      status);
00277
00278
00279
```

8.61   fragment-host.h File 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)

8.61.1   Detailed Description

Fragmented message support for EZSP Hosts. Splits long messages into smaller blocks for transmission and reassembles received blocks. See Message Fragmentation for documentation.

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

Definition in file fragment-host.h.

8.62   fragment-host.h

```
00001
00059 void ezspFragmentInit(int16u receiveBufferLength, int8u
      *receiveBuffer);
00060
00095 EmberStatus ezspFragmentSendUnicast(
      EmberOutgoingMessageType type,
00096                                     int16u indexOrDestination,
00097                                     EmberApsFrame *apsFrame,
00098                                     int8u maxFragmentSize,
00099                                     int16u messageLength,
00100                                     int8u *messageContents);
00101
00114 EmberStatus ezspFragmentSourceRouteHandler
      (void);
00115
00130 boolean ezspFragmentMessageSent(EmberApsFrame
      *apsFrame, EmberStatus status);
00131
00140 void ezspFragmentMessageSentHandler(EmberStatus
      status);
00141
00173 boolean ezspFragmentIncomingMessage(EmberApsFrame
      *apsFrame,
00174                                     EmberNodeId sender,
00175                                     int16u *messageLength,
00176                                     int8u **messageContents);
00177
00182 void ezspFragmentTick(void);
00183
```

8.63 fragment.h File Reference

Transmitting

- EmberStatus emberFragmentSendUnicast (EmberOutgoingMessageType type, int16u indexOrDestination, EmberApsFrame ∗apsFrame, EmberMessageBuffer payload, int8u maxFragmentSize)
- boolean emberFragmentMessageSent (EmberApsFrame ∗apsFrame, EmberMessageBuffer buffer, EmberStatus status)
- void emberFragmentMessageSentHandler (EmberStatus status)

Receiving

- boolean emberFragmentIncomingMessage (EmberApsFrame ∗apsFrame, EmberMessageBuffer payload)
- void emberFragmentTick (void)

8.63.1 Detailed Description

Splits long messages into smaller blocks for transmission and reassembles received blocks. See Message Fragmentation for documentation.

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

Definition in file fragment.h.

8.64 fragment.h

```
00001
00068 EmberStatus emberFragmentSendUnicast(
      EmberOutgoingMessageType type,
00069                                       int16u indexOrDestination,
00070                                       EmberApsFrame *apsFrame,
00071                                       EmberMessageBuffer
      payload,
00072                                       int8u maxFragmentSize);
00073
00088 boolean emberFragmentMessageSent(EmberApsFrame
       *apsFrame,
00089                                  EmberMessageBuffer buffer,
00090                                  EmberStatus status);
00091
00099 void emberFragmentMessageSentHandler(EmberStatus
       status);
00100
00127 boolean emberFragmentIncomingMessage(EmberApsFrame
       *apsFrame,
00128                                       EmberMessageBuffer
      payload);
00129
00133 void emberFragmentTick(void);
00134
```

8.65   hal.h File Reference

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

8.65.1   Detailed Description

Generic set of HAL includes for all platforms. See also Hardware Abstraction Layer (HAL) API Reference for more documentation.

Some HAL includes are not used or present in builds intended for the Host processor connected to the Ember Network Coprocessor.

Definition in file hal.h.

8.66   hal.h

```
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 #include "host/bootloader-eeprom.h"
00075 //Pull in the micro specific ADC, buzzer, and clocks headers.  The
00076 //specific header is chosen by the build include path pointing at
00077 //the appropriate directory.
00078 #include "adc.h"
00079 #include "buzzer.h"
00080
00081 #else //HAL_MICRO
00082
00083 // Keep micro and board first for specifics used by other headers
00084 #include "micro/micro.h"
00085 #if !defined(STACK) && defined(BOARD_HEADER)
00086 #include BOARD_HEADER
00087 #endif
00088
00089 #if (! defined(EMBER_STACK_IP))
00090   // Pro Stack
00091   #include "micro/adc.h"
00092   #include "micro/button.h"
```

```
00093   #include "micro/buzzer.h"
00094   #include "micro/crc.h"
00095   #include "micro/endian.h"
00096   #include "micro/led.h"
00097   #include "micro/random.h"
00098   #include "micro/serial.h"
00099   #include "micro/spi.h"
00100   #include "micro/system-timer.h"
00101   #include "micro/bootloader-eeprom.h"
00102
00103   //Host processors do not use the following modules, therefore the header
00104   //files should be ignored.
00105   #ifndef EZSP_HOST
00106     #include "micro/bootloader-interface.h"
00107     #include "micro/diagnostic.h"
00108     #include "micro/token.h"
00109     //No public HAL code in release 4.0 uses the symbol timer,
00110     //therefore it should not be in doxygen.
00111     #ifndef DOXYGEN_SHOULD_SKIP_THIS
00112       #include "micro/symbol-timer.h"
00113     #endif // DOXYGEN_SHOULD_SKIP_THIS
00114   #endif //EZSP_HOST
00115
00116 #else
00117   // IP Stack
00118   #include "micro/endian.h"
00119   #include "micro/random.h"
00120   #include "micro/serial.h"
00121   #include "micro/system-timer.h"
00122   //Host processors do not use the following modules, therefore the header
00123   //files should be ignored.
00124   #ifndef UNIX_HOST
00125     #include "micro/adc.h"
00126     #include "micro/button.h"
00127     #include "micro/buzzer.h"
00128     #include "micro/crc.h"
00129     #include "micro/led.h"
00130     #include "micro/spi.h"
00131     #include "micro/bootloader-interface.h"
00132     #include "micro/diagnostic.h"
00133     #include "micro/token.h"
00134     //No public HAL code in release 4.0 uses the symbol timer,
00135     //therefore it should not be in doxygen.
00136     #ifndef DOXYGEN_SHOULD_SKIP_THIS
00137       #include "micro/symbol-timer.h"
00138     #endif // DOXYGEN_SHOULD_SKIP_THIS
00139   #endif //UNIX_HOST
00140
00141 #endif // !EMBER_STACK_IP
00142
00143 #endif
00144
00145 #endif //__HAL_H__
00146
```

8.67   iar.h File Reference

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

Macros

- #define HAL_HAS_INT64
- #define _HAL_USE_COMMON_PGM_
- #define _HAL_USE_COMMON_MEMUTILS_
- #define PLATCOMMONOKTOINCLUDE
- #define MAIN_FUNCTION_PARAMETERS

Functions

- void _executeBarrierInstructions (void)

Master Variable Types

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

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

Miscellaneous Macros

- #define BIGENDIAN_CPU
- #define NTOHS(val)
- #define NTOHL(val)
- #define NO_STRIPPING
- #define EEPROM
- #define __SOURCEFILE__
- #define assert(condition)
- #define __delay_cycles(x)
- #define DEBUG_LEVEL
- #define halResetWatchdog()
- #define __attribute__(nothing)
- #define UNUSED
- #define SIGNED_ENUM
- #define STACK_FILL_VALUE
- #define RAMFUNC
- #define NO_OPERATION()
- #define SET_REG_FIELD(reg, field, value)
- #define simulatedTimePasses()
- #define simulatedTimePassesMs(x)
- #define simulatedSerialTimePasses()
- #define _HAL_USE_COMMON_DIVMOD_
- #define VAR_AT_SEGMENT(__variableDeclaration, __segmentName)
- void halInternalAssertFailed (const char ∗filename, int linenumber)
- void halInternalResetWatchDog (void)

Portable segment names

- #define __NO_INIT__
- #define __DEBUG_CHANNEL__
- #define __INTVEC__
- #define __CSTACK__
- #define __RESETINFO__
- #define __DATA_INIT__
- #define __DATA__
- #define __BSS__
- #define __APP_RAM__
- #define __CONST__
- #define __TEXT__
- #define __TEXTRW_INIT__
- #define __TEXTRW__
- #define __AAT__
- #define __BAT__
- #define __FAT__
- #define __RAT__
- #define __NVM__
- #define __SIMEE__
- #define __EMHEAP__
- #define __EMHEAP_OVERLAY__
- #define __GUARD_REGION__
- #define __DLIB_PERTHREAD_INIT__
- #define __DLIB_PERTHREAD_INITIALIZED_DATA__
- #define __DLIB_PERTHREAD_ZERO_DATA__
- #define STACK_SEGMENT_BEGIN
- #define STACK_SEGMENT_END
- #define EMHEAP_SEGMENT_BEGIN
- #define EMHEAP_SEGMENT_END
- #define EMHEAP_OVERLAY_SEGMENT_END
- #define RESETINFO_SEGMENT_END
- #define CODE_SEGMENT_BEGIN
- #define CODE_SEGMENT_END

Global Interrupt Manipulation Macros

Note: The special purpose BASEPRI register is used to enable and disable interrupts while permitting faults. When BASEPRI is set to 1 no interrupts can trigger. The configurable faults (usage, memory management, and bus faults) can trigger if enabled as well as the always-enabled exceptions (reset, NMI and hard fault). When BASEPRI is set to 0, it is disabled, so any interrupt can triggger if its priority is higher than the current priority.

- #define ATOMIC_LITE(blah)
- #define DECLARE_INTERRUPT_STATE_LITE
- #define DISABLE_INTERRUPTS_LITE()
- #define RESTORE_INTERRUPTS_LITE()
- #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()
- #define SET_BASE_PRIORITY_LEVEL(basepri)

External Declarations

These are routines that are defined in certain header files that we don't want to include, e.g. stdlib.h

- int abs (int I)

8.67.1 Detailed Description

See IAR PLATFORM_HEADER Configuration for detailed documentation.

Definition in file iar.h.

8.68 iar.h

```
00001
00019 #ifndef __IAR_H__
00020 #define __IAR_H__
00021
00022 #ifndef __ICCARM__
00023   #error Improper PLATFORM_HEADER
00024 #endif
00025
00026 #if (__VER__ < 6040002)
00027   #error Only IAR EWARM versions greater than 6.40.2 are supported
00028 #endif // __VER__
00029
00030
00031 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00032   #include <intrinsics.h>
00033   #include <stdarg.h>
00034   #if defined (CORTEXM3_EM357) || defined (CORTEXM3_EM351)
00035     #include "micro/cortexm3/em35x/regs.h"
00036   #elif defined (CORTEXM3_STM32W108)
00037     #include "micro/cortexm3/stm32w108/regs.h"
00038   #else
00039     #error Unknown CORTEXM3 micro
00040   #endif
00041   //Provide a default NVIC configuration file.  The build process can
00042   //override this if it needs to.
00043   #ifndef NVIC_CONFIG
00044     #define NVIC_CONFIG "hal/micro/cortexm3/nvic-config.h"
00045   #endif
00046 //[[
00047 #ifdef  EMBER_EMU_TEST
00048   #ifdef  I_AM_AN_EMULATOR
00049     // This register is defined for both the chip and the emulator with
00050     // with distinct reset values.  Need to undefine to avoid preprocessor
00051     // collision.
00052     #undef DATA_EMU_REGS_BASE
00053     #undef DATA_EMU_REGS_END
00054     #undef DATA_EMU_REGS_SIZE
00055     #undef I_AM_AN_EMULATOR
00056     #undef I_AM_AN_EMULATOR_REG
00057     #undef I_AM_AN_EMULATOR_ADDR
00058     #undef I_AM_AN_EMULATOR_RESET
00059     #undef I_AM_AN_EMULATOR_I_AM_AN_EMULATOR
00060     #undef I_AM_AN_EMULATOR_I_AM_AN_EMULATOR_MASK
```

```
00061     #undef I_AM_AN_EMULATOR_I_AM_AN_EMULATOR_BIT
00062     #undef I_AM_AN_EMULATOR_I_AM_AN_EMULATOR_BITS
00063   #endif//I_AM_AN_EMULATOR
00064   #if defined (CORTEXM3_EM357) || defined (CORTEXM3_EM351)
00065     #include "micro/cortexm3/em35x/regs-emu.h"
00066   #else
00067     #error MICRO currently not supported for emulator builds.
00068   #endif
00069 #endif//EMBER_EMU_TEST
00070 //]]
00071
00072 // suppress warnings about unknown pragmas
00073 //  (as they may be pragmas known to other platforms)
00074 #pragma diag_suppress = pe161
00075
00076 #endif  // DOXYGEN_SHOULD_SKIP_THIS
00077
00078
00079
00080
00089 typedef unsigned char  boolean;
00090 typedef unsigned char  int8u;
00091 typedef signed   char  int8s;
00092 typedef unsigned short int16u;
00093 typedef signed   short int16s;
00094 typedef unsigned int   int32u;
00095 typedef signed   int   int32s;
00096 typedef unsigned long long int64u;
00097 typedef signed   long long int64s;
00098 typedef unsigned int   PointerType;
00100
00104 #define HAL_HAS_INT64
00105
00109 #define _HAL_USE_COMMON_PGM_
00110
00111
00112
00114
00116
00117
00118
00123 #define BIGENDIAN_CPU  FALSE
00124
00125
00130 #define NTOHS(val) (__REV16(val))
00131 #define NTOHL(val) (__REV(val))
00132
00133
00138 #define NO_STRIPPING  __root
00139
00140
00145 #define EEPROM  errorerror
00146
00147
00148 #ifndef __SOURCEFILE__
00149
00154   #define __SOURCEFILE__ __FILE__
00155 #endif
00156
00157
00158 #undef assert
00159
00163 void halInternalAssertFailed(const char *filename, int
      linenumber);
00164
00170 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00171 #define assert(condition)
00172 #else //DOXYGEN_SHOULD_SKIP_THIS
00173 // Don't define PUSH_REGS_BEFORE_ASSERT if it causes problems with the
      compiler.
00174 // For example, in some compilers any inline assembly disables all
      optimization.
00175 //
00176 // For IAR V5.30, inline assembly apparently does not affect compiler output.
00177 #define PUSH_REGS_BEFORE_ASSERT
00178 #ifdef PUSH_REGS_BEFORE_ASSERT
00179 #define assert(condition)  \
00180       do { if (! (condition)) { \
00181         asm("PUSH {R0,R1,R2,LR}"); \
00182         halInternalAssertFailed(__SOURCEFILE__, __LINE__); } } while(0)
```

```
00183 #else
00184 #define assert(condition) \
00185        do { if (! (condition)) { \
00186          halInternalAssertFailed(__SOURCEFILE__, __LINE__); } } while(0)
00187 #endif
00188 #endif //DOXYGEN_SHOULD_SKIP_THIS
00189
00190 #ifndef BOOTLOADER
00191   #undef __delay_cycles
00192
00198   #define __delay_cycles(x)
      please_use_halCommonDelayMicroseconds_instead_of_delay_cycles
00199 #endif
00200
00205 #ifndef DEBUG_LEVEL
00206   #if defined(DEBUG) && defined(DEBUG_OFF)
00207     #error "DEBUG and DEBUG_OFF cannot be both be defined!"
00208   #elif defined(DEBUG)
00209     #define DEBUG_LEVEL FULL_DEBUG
00210   #elif defined(DEBUG_OFF)
00211     #define DEBUG_LEVEL NO_DEBUG
00212   #else
00213     #define DEBUG_LEVEL BASIC_DEBUG
00214   #endif
00215 #endif
00216
00222 void halInternalResetWatchDog(void);
00223 #define halResetWatchdog()  halInternalResetWatchDog()
00224
00225
00229 #define __attribute__(nothing)
00230
00231
00236 #define UNUSED
00237
00241 #define SIGNED_ENUM
00242
00243
00247 #define STACK_FILL_VALUE  0xCDCDCDCD
00248
00252 #ifdef RAMEXE
00253   //If the whole build is running out of RAM, as chosen by the RAMEXE build
00254   //define, then define RAMFUNC to nothing since it's not needed.
00255   #define RAMFUNC
00256 #else //RAMEXE
00257   #define RAMFUNC __ramfunc
00258 #endif //RAMEXE
00259
00263 #define NO_OPERATION() __no_operation()
00264
00269 #define SET_REG_FIELD(reg, field, value)                    \
00270   do{                                                       \
00271     reg = ((reg & (~field##_MASK)) |                        \
00272           (int32u) (((int32u) value) << field##_BIT));      \
00273   }while(0)
00274
00278 #define simulatedTimePasses()
00279
00282 #define simulatedTimePassesMs(x)
00283
00286 #define simulatedSerialTimePasses()
00287
00288
00292 #define _HAL_USE_COMMON_DIVMOD_
00293
00294
00299 #define VAR_AT_SEGMENT(__variableDeclaration, __segmentName) \
00300     __variableDeclaration @ __segmentName
00301
00303
00304
00305
00312 #define __NO_INIT__        ".noinit"
00313 #define __DEBUG_CHANNEL__ "DEBUG_CHANNEL"
00314 #define __INTVEC__ ".intvec"
00315 #define __CSTACK__ "CSTACK"
00316 #define __RESETINFO__ "RESETINFO"
00317 #define __DATA_INIT__ ".data_init"
00318 #define __DATA__ ".data"
00319 #define __BSS__ ".bss"
```

```
00320 #define __APP_RAM__ "APP_RAM"
00321 #define __CONST__ ".rodata"
00322 #define __TEXT__ ".text"
00323 #define __TEXTRW_INIT__ ".textrw_init"
00324 #define __TEXTRW__ ".textrw"
00325 #define __AAT__ "AAT"  // Application address table
00326 #define __BAT__ "BAT"  // Bootloader address table
00327 #define __FAT__ "FAT"  // Fixed address table
00328 #define __RAT__ "RAT"  // Ramexe address table
00329 #define __NVM__ "NVM" //Non-Volatile Memory data storage
00330 #define __SIMEE__ "SIMEE" //Simulated EEPROM storage
00331 #define __EMHEAP__ "EMHEAP" // Heap region for extra memory
00332 #define __EMHEAP_OVERLAY__ "EMHEAP_overlay" // Heap and reset info combined
00333 #define __GUARD_REGION__ "GUARD_REGION" // Guard page between heap and stack
00334 #define __DLIB_PERTHREAD_INIT__ "__DLIB_PERTHREAD_init" // DLIB_PERTHREAD flash
      initialization data
00335 #define __DLIB_PERTHREAD_INITIALIZED_DATA__ "DLIB_PERTHREAD_INITIALIZED_DATA"
      // DLIB_PERTHREAD RAM region to init
00336 #define __DLIB_PERTHREAD_ZERO_DATA__ "DLIB_PERTHREAD_ZERO_DATA" //
      DLIB_PERTHREAD RAM region to zero out
00337
00338
00339 //=============================================================================
00340 // The '#pragma segment=' declaration must be used before attempting to access
00341 // the segments so the compiler properly handles the __segment_*() functions.
00342 //
00343 // The segment names used here are the default segment names used by IAR. Refer
00344 // to the IAR Compiler Reference Guide for a proper description of these
00345 // segments.
00346 //=============================================================================
00347 #pragma segment=__NO_INIT__
00348 #pragma segment=__DEBUG_CHANNEL__
00349 #pragma segment=__INTVEC__
00350 #pragma segment=__CSTACK__
00351 #pragma segment=__RESETINFO__
00352 #pragma segment=__DATA_INIT__
00353 #pragma segment=__DATA__
00354 #pragma segment=__BSS__
00355 #pragma segment=__APP_RAM__
00356 #pragma segment=__CONST__
00357 #pragma segment=__TEXT__
00358 #pragma segment=__TEXTRW_INIT__
00359 #pragma segment=__TEXTRW__
00360 #pragma segment=__AAT__
00361 #pragma segment=__BAT__
00362 #pragma segment=__FAT__
00363 #pragma segment=__RAT__
00364 #pragma segment=__NVM__
00365 #pragma segment=__SIMEE__
00366 #pragma segment=__EMHEAP__
00367 #pragma segment=__EMHEAP_OVERLAY__
00368 #pragma segment=__GUARD_REGION__
00369 #pragma segment=__DLIB_PERTHREAD_INIT__
00370 #pragma segment=__DLIB_PERTHREAD_INITIALIZED_DATA__
00371 #pragma segment=__DLIB_PERTHREAD_ZERO_DATA__
00372
00373 #define STACK_SEGMENT_BEGIN __segment_begin(__CSTACK__)
00374 #define STACK_SEGMENT_END   __segment_end(__CSTACK__)
00375
00376 #define EMHEAP_SEGMENT_BEGIN __segment_begin(__EMHEAP__)
00377 #define EMHEAP_SEGMENT_END   __segment_end(__EMHEAP__)
00378
00379 #define EMHEAP_OVERLAY_SEGMENT_END __segment_end(__EMHEAP_OVERLAY__)
00380
00381 #define RESETINFO_SEGMENT_END __segment_end(__RESETINFO__)
00382
00383 #define CODE_SEGMENT_BEGIN __segment_begin(__TEXT__)
00384 #define CODE_SEGMENT_END   __segment_end(__TEXT__)
00385
00388 //A utility function for inserting barrier instructions.  These
00389 //instructions should be used whenever the MPU is enabled or disabled so
00390 //that all memory/instruction accesses can complete before the MPU changes
00391 //state.
00392 void _executeBarrierInstructions(void);
00393
00395
00405
00406
00407
00408 #define ATOMIC_LITE(blah) ATOMIC(blah)
```

```
00409 #define DECLARE_INTERRUPT_STATE_LITE DECLARE_INTERRUPT_STATE
00410 #define DISABLE_INTERRUPTS_LITE() DISABLE_INTERRUPTS()
00411 #define RESTORE_INTERRUPTS_LITE() RESTORE_INTERRUPTS()
00412
00413 #ifdef BOOTLOADER
00414   #ifndef DOXYGEN_SHOULD_SKIP_THIS
00415     // The bootloader does not use interrupts
00416     #define DECLARE_INTERRUPT_STATE
00417     #define DISABLE_INTERRUPTS() do { } while(0)
00418     #define RESTORE_INTERRUPTS() do { } while(0)
00419     #define INTERRUPTS_ON() do { } while(0)
00420     #define INTERRUPTS_OFF() do { } while(0)
00421     #define INTERRUPTS_ARE_OFF() (FALSE)
00422     #define ATOMIC(blah) { blah }
00423     #define HANDLE_PENDING_INTERRUPTS() do { } while(0)
00424     #define SET_BASE_PRIORITY_LEVEL(basepri) do { } while(0)
00425   #endif  // DOXYGEN_SHOULD_SKIP_THIS
00426 #else  // BOOTLOADER
00427
00428   #ifndef DOXYGEN_SHOULD_SKIP_THIS
00429     // A series of macros for the diagnostics of the global interrupt state.
00430     // These macros either enable or disable the debugging code in the
00431     // Interrupt Manipulation Macros as well as define the two pins used for
00432     // indicating the entry and exit of critical sections.
00433     //UNCOMMENT the below #define to enable interrupt debugging.
00434     //#define INTERRUPT_DEBUGGING
00435     #ifdef INTERRUPT_DEBUGGING
00436       // Designed to use the breakout board LED (PC5)
00437       #define ATOMIC_DEBUG(x)  x
00438       #define I_PIN         5
00439       #define I_PORT        C
00440       #define I_CFG_HL      H
00441       // For the concatenation to work, we need to define the regs via their
      own macros:
00442       #define I_SET_REG(port)          GPIO_P ## port ## SET
00443       #define I_CLR_REG(port)          GPIO_P ## port ## CLR
00444       #define I_OUT_REG(port)          GPIO_P ## port ## OUT
00445       #define I_CFG_MSK(port, pin)     P ## port ## pin ## _CFG_MASK
00446       #define I_CFG_BIT(port, pin)     P ## port ## pin ## _CFG_BIT
00447       #define I_CFG_REG(port, hl)      GPIO_P ## port ## CFG ## hl
00448       // Finally, the macros to actually manipulate the interrupt status IO
00449       #define I_OUT(port, pin, hl) \
00450         do { I_CFG_REG(port, hl) &= ~(I_CFG_MSK(port, pin));       \
00451              I_CFG_REG(port, hl) |= (GPIOCFG_OUT << I_CFG_BIT(port, pin)); \
00452         } while (0)
00453       #define I_SET(port, pin)   do { I_SET_REG(port) = (BIT(pin)); } while (0)
00454       #define I_CLR(port, pin)   do { I_CLR_REG(port) = (BIT(pin)); } while (0)
00455       #define I_STATE(port, pin)    ((I_OUT_REG(port) & BIT(pin)) == BIT(pin))
00456       #define DECLARE_INTERRUPT_STATE int8u _emIsrState, _emIsrDbgIoState
00457     #else
00458       #define ATOMIC_DEBUG(x)
00459
00464       #define DECLARE_INTERRUPT_STATE int8u _emIsrState
00465     #endif//INTERRUPT_DEBUGGING
00466
00467     // Prototypes for the BASEPRI and PRIMASK access functions.  They are very
00468     // basic and instantiated in assembly code in the file spmr.s37 (since
00469     // there are no C functions that cause the compiler to emit code to access
00470     // the BASEPRI/PRIMASK). This will inhibit the core from taking interrupts
00471     // with a priority equal to or less than the BASEPRI value.
00472     // Note that the priority values used by these functions are 5 bits and
00473     // right-aligned
00474     extern int8u _readBasePri(void);
00475     extern void _writeBasePri(int8u priority);
00476
00477     // Prototypes for BASEPRI functions used to disable and enable interrupts
00478     // while still allowing enabled faults to trigger.
00479     extern void _enableBasePri(void);
00480     extern int8u _disableBasePri(void);
00481     extern boolean _basePriIsDisabled(void);
00482
00483     // Prototypes for setting and clearing PRIMASK for global interrupt
00484     // enable/disable.
00485     extern void _setPriMask(void);
00486     extern void _clearPriMask(void);
00487   #endif  // DOXYGEN_SHOULD_SKIP_THIS
00488
00489   //The core Global Interrupt Manipulation Macros start here.
00490
00497   #define DISABLE_INTERRUPTS()                        \
```

```
00498    do {                                            \
00499      _emIsrState = _disableBasePri();              \
00500      ATOMIC_DEBUG(                                 \
00501        _emIsrDbgIoState = I_STATE(I_PORT,I_PIN); \
00502        I_SET(I_PORT, I_PIN);                      \
00503      )                                            \
00504    } while(0)
00505
00506
00514  #define RESTORE_INTERRUPTS()        \
00515    do {                              \
00516      ATOMIC_DEBUG(                   \
00517        if(!_emIsrDbgIoState)        \
00518          I_CLR(I_PORT, I_PIN);      \
00519      )                              \
00520      _writeBasePri(_emIsrState); \
00521    } while(0)
00522
00523
00528  #define INTERRUPTS_ON()                \
00529    do {                                 \
00530      ATOMIC_DEBUG(                      \
00531        I_OUT(I_PORT, I_PIN, I_CFG_HL); \
00532        I_CLR(I_PORT, I_PIN);           \
00533      )                                  \
00534      _enableBasePri();                  \
00535    } while(0)
00536
00537
00542  #define INTERRUPTS_OFF()      \
00543    do {                        \
00544      (void)_disableBasePri();  \
00545      ATOMIC_DEBUG(             \
00546        I_SET(I_PORT, I_PIN);  \
00547      )                        \
00548    } while(0)
00549
00550
00554  #define INTERRUPTS_ARE_OFF() ( _basePriIsDisabled() )
00555
00560  #define INTERRUPTS_WERE_ON() (_emIsrState == 0)
00561
00566  #define ATOMIC(blah)           \
00567  {                              \
00568    DECLARE_INTERRUPT_STATE;   \
00569    DISABLE_INTERRUPTS();      \
00570    { blah }                   \
00571    RESTORE_INTERRUPTS();      \
00572  }
00573
00574
00582  #define HANDLE_PENDING_INTERRUPTS() \
00583    do {                             \
00584      if (INTERRUPTS_ARE_OFF()) {   \
00585        INTERRUPTS_ON();            \
00586        INTERRUPTS_OFF();           \
00587      }                            \
00588    } while (0)
00589
00590
00604  #define SET_BASE_PRIORITY_LEVEL(basepri) \
00605    do {                                   \
00606      _writeBasePri(basepri);             \
00607    } while(0)
00608
00609 #endif // BOOTLOADER
00610
00611
00612
00613
00617 #define _HAL_USE_COMMON_MEMUTILS_
00618
00620
00624
00625
00626
00636 int abs(int I);
00637
00639
00640
```

```
00641
00642
00646 #define PLATCOMMONOKTOINCLUDE
00647   #include "hal/micro/generic/compiler/platform-common.h
      "
00648 #undef PLATCOMMONOKTOINCLUDE
00649
00653 #define MAIN_FUNCTION_PARAMETERS void
00654
00655 #endif // __IAR_H__
00656
```

8.69   led.h File Reference

Typedefs

- typedef enum HalBoardLedPins HalBoardLed

Functions

- void halInternalInitLed (void)
- void halToggleLed (HalBoardLed led)
- void halSetLed (HalBoardLed led)
- void halClearLed (HalBoardLed led)
- void halStackIndicateActivity (boolean turnOn)

8.69.1   Detailed Description

See LED Control for documentation.

Definition in file led.h.

8.70   led.h

```
00001
00022 void halInternalInitLed(void);
00023
00027 #if defined(STACK) || defined(MINIMAL_HAL)
00028   typedef int8u HalBoardLed;
00029 #else
00030   typedef enum HalBoardLedPins HalBoardLed;
00031 #endif
00032 // Note: Even though many compilers will use 16 bits for an enum instead of 8,
00033 //  we choose to use an enum here.  The possible compiler inefficiency does not
00034 //  affect stack-based parameters and local variables, which is the
00035 //  general case for led paramters.
00036
00042 void halToggleLed(HalBoardLed led);
00043
00049 void halSetLed(HalBoardLed led);
00050
00056 void halClearLed(HalBoardLed led);
00057
00067 void halStackIndicateActivity(boolean turnOn);
00068
```

8.71 message.h File Reference

Macros

- #define EMBER_APSC_MAX_ACK_WAIT_HOPS_MULTIPLIER_MS
- #define EMBER_APSC_MAX_ACK_WAIT_TERMINAL_SECURITY_MS

Functions

- int8u emberMaximumApsPayloadLength (void)
- EmberStatus emberSendMulticast (EmberApsFrame ∗apsFrame, int8u radius, int8u nonmember-Radius, EmberMessageBuffer message)
- EmberStatus emberSendUnicast (EmberOutgoingMessageType type, int16u indexOrDestination, Ember-ApsFrame ∗apsFrame, EmberMessageBuffer message)
- EmberStatus emberSendBroadcast (EmberNodeId destination, EmberApsFrame ∗apsFrame, int8u radius, EmberMessageBuffer message)
- EmberStatus emberProxyBroadcast (EmberNodeId source, EmberNodeId destination, int8u sequence, EmberApsFrame ∗apsFrame, int8u radius, EmberMessageBuffer message)
- EmberStatus emberSendManyToOneRouteRequest (int16u concentratorType, int8u radius)
- int8u emberAppendSourceRouteHandler (EmberNodeId destination, EmberMessageBuffer header)
- void emberIncomingRouteRecordHandler (EmberNodeId source, EmberEUI64 sourceEui, int8u relay-Count, EmberMessageBuffer header, int8u relayListIndex)
- void emberIncomingManyToOneRouteRequestHandler (EmberNodeId source, EmberEUI64 long-Id, int8u cost)
- void emberIncomingRouteErrorHandler (EmberStatus status, EmberNodeId target)
- EmberStatus emberCancelMessage (EmberMessageBuffer message)
- void emberMessageSentHandler (EmberOutgoingMessageType type, int16u indexOrDestination, Ember-ApsFrame ∗apsFrame, EmberMessageBuffer message, EmberStatus status)
- void emberIncomingMessageHandler (EmberIncomingMessageType type, EmberApsFrame ∗aps-Frame, EmberMessageBuffer message)
- EmberStatus emberGetLastHopLqi (int8u ∗lastHopLqi)
- EmberStatus emberGetLastHopRssi (int8s ∗lastHopRssi)
- EmberNodeId emberGetSender (void)
- EmberStatus emberGetSenderEui64 (EmberEUI64 senderEui64)
- EmberStatus emberSendReply (int16u clusterId, EmberMessageBuffer reply)
- void emberSetReplyFragmentData (int16u fragmentData)
- boolean emberAddressTableEntryIsActive (int8u addressTableIndex)
- EmberStatus emberSetAddressTableRemoteEui64 (int8u addressTableIndex, EmberEUI64 eui64)
- void emberSetAddressTableRemoteNodeId (int8u addressTableIndex, EmberNodeId id)
- void emberGetAddressTableRemoteEui64 (int8u addressTableIndex, EmberEUI64 eui64)
- EmberNodeId emberGetAddressTableRemoteNodeId (int8u addressTableIndex)
- void emberSetExtendedTimeout (EmberEUI64 remoteEui64, boolean extendedTimeout)
- boolean emberGetExtendedTimeout (EmberEUI64 remoteEui64)
- void emberIdConflictHandler (EmberNodeId conflictingId)
- boolean emberPendingAckedMessages (void)

Variables

- int16u emberApsAckTimeoutMs
- EmberMulticastTableEntry ∗ emberMulticastTable
- int8u emberMulticastTableSize

8.71.1   Detailed Description

EmberZNet API for sending and receiving messages. See Sending and Receiving Messages for documentation.

Definition in file message.h.

8.72   message.h

```
00001
00024 int8u emberMaximumApsPayloadLength(void);
00025
00032 #define EMBER_APSC_MAX_ACK_WAIT_HOPS_MULTIPLIER_MS      50
00033
00037 #define EMBER_APSC_MAX_ACK_WAIT_TERMINAL_SECURITY_MS  100
00038
00048 extern int16u emberApsAckTimeoutMs;
00049
00079 EmberStatus emberSendMulticast(EmberApsFrame
     *apsFrame,
00080                                int8u radius,
00081                                int8u nonmemberRadius,
00082                                EmberMessageBuffer message);
00083
00141 EmberStatus emberSendUnicast(
     EmberOutgoingMessageType type,
00142                               int16u indexOrDestination,
00143                               EmberApsFrame *apsFrame,
00144                               EmberMessageBuffer message);
00145
00162 EmberStatus emberSendBroadcast(EmberNodeId
     destination,
00163                                EmberApsFrame *apsFrame,
00164                                int8u radius,
00165                                EmberMessageBuffer message);
00166
00187 EmberStatus emberProxyBroadcast(EmberNodeId
     source,
00188                                 EmberNodeId destination,
00189                                 int8u sequence,
00190                                 EmberApsFrame *apsFrame,
00191                                 int8u radius,
00192                                 EmberMessageBuffer message);
00193
00246 EmberStatus emberSendManyToOneRouteRequest
     (int16u concentratorType,
00247                                             int8u radius);
00248
00273 int8u emberAppendSourceRouteHandler(
     EmberNodeId destination,
00274                                    EmberMessageBuffer header
     );
00275
00295 void emberIncomingRouteRecordHandler(EmberNodeId
     source,
00296                                     EmberEUI64 sourceEui,
00297                                     int8u relayCount,
00298                                     EmberMessageBuffer
     header,
00299                                     int8u relayListIndex);
00300
00314 void emberIncomingManyToOneRouteRequestHandler
     (EmberNodeId source,
00315                                              EmberEUI64 longId,
00316                                              int8u cost);
00317
00363 void emberIncomingRouteErrorHandler(EmberStatus
     status,
00364                                    EmberNodeId target);
00365
00372 EmberStatus emberCancelMessage(EmberMessageBuffer
     message);
00373
00391 void emberMessageSentHandler(EmberOutgoingMessageType
```

```
       type,
00392                     int16u indexOrDestination,
00393                     EmberApsFrame *apsFrame,
00394                     EmberMessageBuffer message,
00395                     EmberStatus status);
00396
00422 void emberIncomingMessageHandler(
      EmberIncomingMessageType type,
00423                               EmberApsFrame *apsFrame,
00424                               EmberMessageBuffer message);
00425
00456 EmberStatus emberGetLastHopLqi(int8u *
      lastHopLqi);
00457
00491 EmberStatus emberGetLastHopRssi(int8s *
      lastHopRssi);
00492
00500 EmberNodeId emberGetSender(void);
00501
00520 EmberStatus emberGetSenderEui64(EmberEUI64
       senderEui64);
00521
00548 EmberStatus emberSendReply(int16u clusterId,
      EmberMessageBuffer reply);
00549
00558 void emberSetReplyFragmentData(int16u
      fragmentData);
00559
00560
00575 boolean emberAddressTableEntryIsActive(int8u
       addressTableIndex);
00576
00591 EmberStatus emberSetAddressTableRemoteEui64
      (int8u addressTableIndex,
00592                                  EmberEUI64 eui64);
00593
00609 void emberSetAddressTableRemoteNodeId(int8u
       addressTableIndex,
00610                                  EmberNodeId id);
00611
00619 void emberGetAddressTableRemoteEui64(int8u
      addressTableIndex,
00620                                  EmberEUI64 eui64);
00621
00638 EmberNodeId emberGetAddressTableRemoteNodeId
      (int8u addressTableIndex);
00639
00660 void emberSetExtendedTimeout(EmberEUI64
      remoteEui64, boolean extendedTimeout);
00661
00673 boolean emberGetExtendedTimeout(EmberEUI64
      remoteEui64);
00674
00689 void emberIdConflictHandler(EmberNodeId
      conflictingId);
00690
00696 boolean emberPendingAckedMessages(void);
00697
00708 extern EmberMulticastTableEntry *emberMulticastTable
      ;
00709
00712 extern int8u emberMulticastTableSize;
00713
```

8.73 mfglib.h File Reference

Functions

- EmberStatus mfglibStart (void(*mfglibRxCallback)(int8u *packet, int8u linkQuality, int8s rssi))
- EmberStatus mfglibEnd (void)
- EmberStatus mfglibStartTone (void)
- EmberStatus mfglibStopTone (void)

- EmberStatus mfglibStartStream (void)
- EmberStatus mfglibStopStream (void)
- EmberStatus mfglibSendPacket (int8u ∗packet, int16u repeat)
- EmberStatus mfglibSetChannel (int8u chan)
- int8u mfglibGetChannel (void)
- EmberStatus mfglibSetPower (int16u txPowerMode, int8s power)
- int8s mfglibGetPower (void)
- void mfglibSetSynOffset (int8s synOffset)
- int8s mfglibGetSynOffset (void)
- void mfglibTestContModCal (int8u channel, int32u duration)

8.73.1 Detailed Description

See Manufacturing and Functional Test Library for documentation.

Definition in file mfglib.h.

8.74 mfglib.h

```
00001
00030 #ifndef __MFGLIB_H__
00031 #define __MFGLIB_H__
00032
00055 EmberStatus mfglibStart(void (*mfglibRxCallback)(int8u
       *packet, int8u linkQuality, int8s rssi));
00056
00070 EmberStatus mfglibEnd(void);
00071
00086 EmberStatus mfglibStartTone(void);
00087
00095 EmberStatus mfglibStopTone(void);
00096
00106 EmberStatus mfglibStartStream(void);
00107
00117 EmberStatus mfglibStopStream(void);
00118
00143 EmberStatus mfglibSendPacket(int8u * packet,
      int16u repeat);
00144
00158 EmberStatus mfglibSetChannel(int8u chan);
00159
00166 int8u mfglibGetChannel(void);
00167
00186 EmberStatus mfglibSetPower(int16u txPowerMode,
      int8s power);
00187
00194 int8s mfglibGetPower(void);
00195
00212 void mfglibSetSynOffset(int8s synOffset);
00213
00218 int8s mfglibGetSynOffset(void);
00219
00235 void mfglibTestContModCal(int8u channel, int32u
      duration);
00236
00237 #endif // __MFGLIB_H__
00238
```

8.75 micro.h File Reference

```
#include "hal/micro/generic/em2xx-reset-defs.h"
```

Macros

- #define halGetEm2xxResetInfo()

Functions

- void halStackProcessBootCount (void)
- int8u halGetResetInfo (void)
- PGM_P halGetResetString (void)
- void halInternalAssertFailed (PGM_P filename, int linenumber)

8.75.1 Detailed Description

Full HAL functions common across all microcontroller-specific files. See Common Microcontroller Functions for documentation. Some functions in this file return an EmberStatus value. See error-def.h for definitions of all EmberStatus return values.

Definition in file micro.h.

8.76 micro.h

```
00001
00020 #ifndef __MICRO_H__
00021 #define __MICRO_H__
00022
00023 #include "hal/micro/generic/em2xx-reset-defs.h"
00024
00025 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00026
00027 #if ((! defined(EZSP_HOST)) && (! defined(UNIX_HOST)))
00028 // If we are not a Host processor, Define distinct literal values for
00029 // each platform and micro.  The Host processor is not tied to a
00030 // specific platform or micro since it uses EZSP.  These values are
00031 // used in ebl headers and the bootloader query response message.
00032 // Some values may reserved for future use.
00033
00034 // PLAT 1 was the AVR ATMega, no longer supported
00035 // for PLAT 1, MICRO 1 is the AVR ATMega 64
00036 // for PLAT 1, MICRO 2 is the AVR ATMega 128
00037 // for PLAT 1, MICRO 3 is the AVR ATMega 32
00038
00039 // PLAT 2 is the XAP2b
00040 // for PLAT 2, MICRO 1 is the em250
00041 // for PLAT 2, MICRO 2 is the em260
00042
00043 // PLAT 4 is the Cortex-M3
00044 // for PLAT 4, MICRO 1 is the em350
00045 // for PLAT 4, MICRO 2 is the em360
00046 // for PLAT 4, MICRO 3 is the em357
00047 // for PLAT 4, MICRO 4 is the em367
00048 // for PLAT 4, MICRO 5 is the em351
00049 // for PLAT 4, MICRO 6 is the em35x
00050 // for PLAT 4, MICRO 7 is the stm32w108
00051
00052 #if defined( EMBER_TEST )
00053   #define PLAT 1
00054   #define MICRO 2
00055 #elif defined (AVR_ATMEGA)
00056   #define PLAT 1
00057   #if defined (AVR_ATMEGA_64)
00058     #define MICRO 1
00059   #elif defined (AVR_ATMEGA_128)
00060     #define MICRO 2
00061   #elif defined (AVR_ATMEGA_32)
00062     #define MICRO 3
00063   #endif
```

```
00064 #elif defined (XAP2B)
00065   #define PLAT 2
00066   #if defined (XAP2B_EM250)
00067     #define MICRO 1
00068   #elif defined (XAP2B_EM260)
00069     #define MICRO 2
00070   #endif
00071 #elif defined (MSP430)
00072   #define PLAT 3
00073   #if defined (MSP430_1612)
00074     #define MICRO 1
00075   #elif defined (MSP430_1471)
00076     #define MICRO 2
00077   #endif
00078 #elif defined (CORTEXM3)
00079   #define PLAT 4
00080   #if defined (CORTEXM3_EM350)
00081     #define MICRO 1
00082   #elif defined (CORTEXM3_EM360)
00083     #define MICRO 2
00084   #elif defined (CORTEXM3_EM357)
00085     #define MICRO 3
00086   #elif defined (CORTEXM3_EM367)
00087     #define MICRO 4
00088   #elif defined (CORTEXM3_EM351)
00089     #define MICRO 5
00090   #elif defined (CORTEXM3_EM35X)
00091     #define MICRO 6
00092   #elif defined (CORTEXM3_STM32W108)
00093     #define MICRO 7
00094   #endif
00095 #else
00096   #error no platform defined, or unsupported
00097 #endif
00098
00099 #ifndef MICRO
00100   #error no micro defined, or unsupported
00101 #endif
00102
00103 #endif // ((! defined(EZSP_HOST)) && (! defined(UNIX_HOST)))
00104
00105 // Define distinct literal values for each phy.
00106 // These values are used in the bootloader query response message.
00107 // PHY 0 is a null phy
00108 // PHY 1 is the em2420 (no longer supported)
00109 // PHY 2 is the em250
00110 // PHY 3 is the em3xx
00111
00112 #if defined (PHY_EM2420) || defined (PHY_EM2420B)
00113   #error EM2420 is no longer supported.
00114 #elif defined (PHY_EM250) || defined (PHY_EM250B)
00115   #define PHY 2
00116 #elif defined (PHY_EM3XX)
00117   #define PHY 3
00118 #elif defined (PHY_NULL)
00119   #define PHY 0
00120 #else
00121   #error no phy defined, or unsupported
00122 #endif
00123
00124 #endif // DOXYGEN_SHOULD_SKIP_THIS
00125
00139 void halStackProcessBootCount(void);
00140
00145 int8u halGetResetInfo(void);
00146
00154 PGM_P halGetResetString(void);
00155
00164 #ifdef CORTEXM3
00165   int8u halGetEm2xxResetInfo(void);
00166 #else
00167   #define halGetEm2xxResetInfo() halGetResetInfo()
00168 #endif
00169
00170
00180 void halInternalAssertFailed(PGM_P filename, int
      linenumber);
00181
00182
00183 #ifndef DOXYGEN_SHOULD_SKIP_THIS
```

```
00184
00185 #include "micro-common.h"
00186
00187 #if defined( EMBER_TEST )
00188   #include "hal/micro/unix/simulation/micro.h"
00189   #include "hal/micro/unix/simulation/bootloader.h"
00190 #elif defined(AVR_ATMEGA)
00191   #if defined(AVR_ATMEGA_64) || defined(AVR_ATMEGA_128)
00192     #include "avr-atmega/128/micro.h"
00193     #include "avr-atmega/128/bootloader.h"
00194   #elif defined(AVR_ATMEGA_32)
00195     #include "avr-atmega/32/micro.h"
00196   #else
00197     // default, assume 128
00198     #include "avr-atmega/128/micro.h"
00199     #include "avr-atmega/128/bootloader.h"
00200   #endif
00201 #elif defined(MSP430_1612)
00202   #include "msp430/1612/micro.h"
00203   #include "msp430/1612/bootloader.h"
00204 #elif defined(MSP430_1471)
00205   #include "msp430/1471/micro.h"
00206   #include "msp430/1471/bootloader.h"
00207 #elif defined(XAP2B)
00208   #include "xap2b/em250/micro.h"
00209 #elif defined(CORTEXM3)
00210   #include "cortexm3/micro.h"
00211 #elif ((defined(EZSP_HOST) || defined(UNIX_HOST)))
00212   #include "hal/micro/unix/host/micro.h"
00213 #else
00214   #error no platform or micro defined
00215 #endif
00216
00217 // the number of ticks (as returned from halCommonGetInt32uMillisecondTick)
00218 // that represent an actual second. This can vary on different platforms.
00219 // It must be defined by the host system.
00220 #ifndef MILLISECOND_TICKS_PER_SECOND
00221   #define MILLISECOND_TICKS_PER_SECOND 1024UL
00222 // See bug 10232
00223 //  #error "MILLISECOND_TICKS_PER_SECOND is not defined in micro.h!"
00224 #endif
00225
00226 #endif // DOXYGEN_SHOULD_SKIP_THIS
00227
00228 #endif //__MICRO_H__
00229
```

8.77 micro.h File Reference

```
#include "micro-common.h"
```

Functions

- void halInternalSysReset (int16u extendedCause)
- int16u halGetExtendedResetInfo (void)
- PGM_P halGetExtendedResetString (void)

Vector Table Index Definitions

These are numerical definitions for vector table. Indices 0 through 15 are Cortex-M3 standard exception vectors and indices 16 through 32 are EM3XX specific interrupt vectors.

- #define STACK_VECTOR_INDEX
- #define RESET_VECTOR_INDEX

- #define NMI_VECTOR_INDEX
- #define HARD_FAULT_VECTOR_INDEX
- #define MEMORY_FAULT_VECTOR_INDEX
- #define BUS_FAULT_VECTOR_INDEX
- #define USAGE_FAULT_VECTOR_INDEX
- #define RESERVED07_VECTOR_INDEX
- #define RESERVED08_VECTOR_INDEX
- #define RESERVED09_VECTOR_INDEX
- #define RESERVED10_VECTOR_INDEX
- #define SVCALL_VECTOR_INDEX
- #define DEBUG_MONITOR_VECTOR_INDEX
- #define RESERVED13_VECTOR_INDEX
- #define PENDSV_VECTOR_INDEX
- #define SYSTICK_VECTOR_INDEX
- #define TIMER1_VECTOR_INDEX
- #define TIMER2_VECTOR_INDEX
- #define MANAGEMENT_VECTOR_INDEX
- #define BASEBAND_VECTOR_INDEX
- #define SLEEP_TIMER_VECTOR_INDEX
- #define SC1_VECTOR_INDEX
- #define SC2_VECTOR_INDEX
- #define SECURITY_VECTOR_INDEX
- #define MAC_TIMER_VECTOR_INDEX
- #define MAC_TX_VECTOR_INDEX
- #define MAC_RX_VECTOR_INDEX
- #define ADC_VECTOR_INDEX
- #define IRQA_VECTOR_INDEX
- #define IRQB_VECTOR_INDEX
- #define IRQC_VECTOR_INDEX
- #define IRQD_VECTOR_INDEX
- #define DEBUG_VECTOR_INDEX
- #define VECTOR_TABLE_LENGTH

8.77.1 Detailed Description

Utility and convenience functions for EM35x microcontroller. See Common Microcontroller Functions for documentation.

Definition in file cortexm3/micro.h.

8.78 cortexm3/micro.h

```
00001
00013 #ifndef __EM3XX_MICRO_H__
00014 #define __EM3XX_MICRO_H__
00015
00016 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00017
00018 #ifndef __MICRO_H__
00019 #error do not include this file directly - include micro/micro.h
00020 #endif
00021
00022 // Micro specific serial defines
```

```
00023 #define EM_NUM_SERIAL_PORTS 2
00024 #define EMBER_SPI_MASTER 4
00025 #define EMBER_SPI_SLAVE 5
00026 #define EMBER_I2C 6
00027
00028 // Define the priority registers of system handlers and interrupts.
00029 // This example shows how to save the current ADC interrupt priority and
00030 // then set it to 24:
00031 //    int8u oldAdcPriority = INTERRUPT_PRIORITY_REGISTER(ADC);
00032 //    INTERRUPT_PRIORITY_REGISTER(ADC) = 24;
00033
00034 // For Cortex-M3 faults and exceptions
00035 #define HANDLER_PRIORITY_REGISTER(handler) \
00036   (*( ((int8u *)SCS_SHPR_7to4_ADDR) + handler##_VECTOR_INDEX - 4) )
00037
00038 // For EM3XX-specific interrupts
00039 #define INTERRUPT_PRIORITY_REGISTER(interrupt) \
00040   (* ( ((int8u *)NVIC_IPR_3to0_ADDR) + interrupt##_VECTOR_INDEX - 16) )
00041
00042
00043 // The reset types of the EM300 series have both a base type and an
00044 //  extended type.  The extended type is a 16-bit value which has the base
00045 //  type in the upper 8-bits, and the extended classification in the
00046 //  lower 8-bits
00047 // For backwards compatibility with other platforms, only the base type is
00048 //  returned by the halGetResetInfo() API.  For the full extended type, the
00049 //  halGetExtendedResetInfo() API should be called.
00050
00051 #define RESET_BASE_TYPE(extendedType)   ((int8u)(((extendedType) >> 8) & 0xFF))
00052 #define RESET_EXTENDED_FIELD(extendedType) ((int8u)(extendedType & 0xFF))
00053 #define RESET_VALID_SIGNATURE           (0xF00F)
00054 #define RESET_INVALID_SIGNATURE         (0xC33C)
00055
00056 // Define the base reset cause types
00057 #define RESET_BASE_DEF(basename, value, string)  RESET_##basename = value,
00058 #define RESET_EXT_DEF(basename, extname, extvalue, string)  /*nothing*/
00059 enum {
00060   #include "reset-def.h"
00061   NUM_RESET_BASE_TYPES
00062 };
00063 #undef RESET_BASE_DEF
00064 #undef RESET_EXT_DEF
00065
00066 // Define the extended reset cause types
00067 #define RESET_EXT_VALUE(basename, extvalue)  \
00068   (((RESET_##basename)<<8) + extvalue)
00069 #define RESET_BASE_DEF(basename, value, string)  /*nothing*/
00070 #define RESET_EXT_DEF(basename, extname, extvalue, string)  \
00071   RESET_##basename##_##extname = RESET_EXT_VALUE(basename, extvalue),
00072 enum {
00073   #include "reset-def.h"
00074 };
00075 #undef RESET_EXT_VALUE
00076 #undef RESET_BASE_DEF
00077 #undef RESET_EXT_DEF
00078
00079 // These define the size of the GUARD region configured in the MPU that
00080 // sits between the heap and the stack
00081 #define HEAP_GUARD_REGION_SIZE        (SIZE_32B)
00082 #define HEAP_GUARD_REGION_SIZE_BYTES (1<<(HEAP_GUARD_REGION_SIZE+1))
00083
00084 // Define a value to fill the guard region between the heap and stack.
00085 #define HEAP_GUARD_FILL_VALUE (0xE2E2E2E2)
00086
00087 // Resize the CSTACK to add space to the 'heap' that exists below it
00088 int32u halStackModifyCStackSize(int32s stackSizeDeltaWords);
00089
00090 // Initialize the CSTACK/Heap region and the guard page in between them
00091 void halInternalInitCStackRegion(void);
00092
00093 // Helper functions to get the location of the stack/heap
00094 int32u halInternalGetCStackBottom(void);
00095 int32u halInternalGetHeapTop(void);
00096 int32u halInternalGetHeapBottom(void);
00097
00098 #endif // DOXYGEN_SHOULD_SKIP_THIS
00099
00111 #define STACK_VECTOR_INDEX          0 // special case: stack pointer at reset
00112 #define RESET_VECTOR_INDEX          1
00113 #define NMI_VECTOR_INDEX            2
```

```
00114 #define HARD_FAULT_VECTOR_INDEX      3
00115 #define MEMORY_FAULT_VECTOR_INDEX    4
00116 #define BUS_FAULT_VECTOR_INDEX       5
00117 #define USAGE_FAULT_VECTOR_INDEX     6
00118 #define RESERVED07_VECTOR_INDEX      7
00119 #define RESERVED08_VECTOR_INDEX      8
00120 #define RESERVED09_VECTOR_INDEX      9
00121 #define RESERVED10_VECTOR_INDEX      10
00122 #define SVCALL_VECTOR_INDEX          11
00123 #define DEBUG_MONITOR_VECTOR_INDEX   12
00124 #define RESERVED13_VECTOR_INDEX      13
00125 #define PENDSV_VECTOR_INDEX          14
00126 #define SYSTICK_VECTOR_INDEX         15
00127 #define TIMER1_VECTOR_INDEX          16
00128 #define TIMER2_VECTOR_INDEX          17
00129 #define MANAGEMENT_VECTOR_INDEX      18
00130 #define BASEBAND_VECTOR_INDEX        19
00131 #define SLEEP_TIMER_VECTOR_INDEX     20
00132 #define SC1_VECTOR_INDEX             21
00133 #define SC2_VECTOR_INDEX             22
00134 #define SECURITY_VECTOR_INDEX        23
00135 #define MAC_TIMER_VECTOR_INDEX       24
00136 #define MAC_TX_VECTOR_INDEX          25
00137 #define MAC_RX_VECTOR_INDEX          26
00138 #define ADC_VECTOR_INDEX             27
00139 #define IRQA_VECTOR_INDEX            28
00140 #define IRQB_VECTOR_INDEX            29
00141 #define IRQC_VECTOR_INDEX            30
00142 #define IRQD_VECTOR_INDEX            31
00143 #define DEBUG_VECTOR_INDEX           32
00144
00147 #define VECTOR_TABLE_LENGTH          33
00148
00153 void halInternalSysReset(int16u extendedCause);
00154
00160 int16u halGetExtendedResetInfo(void);
00161
00171 PGM_P halGetExtendedResetString(void);
00172
00173
00174 //[[ ram vectors are not public
00175 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00176
00194 int32u halRegisterRamVector(int8u vectorNumber, int32u
      newVector);
00195
00196
00209 int32u halUnRegisterRamVector(int8u vectorNumber);
00210
00211
00219 void halSetRadioHoldOff(boolean enable);
00220
00221
00226 boolean halGetRadioHoldOff(void);
00227
00228
00235 boolean halRadioHoldOffIsActive(void);
00236
00237
00248 void halStackRadioPowerDownBoard(void);
00249
00250
00261 void halStackRadioPowerUpBoard(void);
00262
00263
00264 #endif // DOXYGEN_SHOULD_SKIP_THIS
00265 //]]
00266
00267 #include "micro-common.h"
00268
00269 #endif //__EM3XX_MICRO_H__
00270
```

8.79 multi-network.h File Reference

Functions

- int8u emberGetCurrentNetwork (void)
- EmberStatus emberSetCurrentNetwork (int8u index)
- int8u emberGetCallbackNetwork (void)

8.79.1 Detailed Description

EmberZNet API for multi-network support. See Multi_network for documentation.

Definition in file multi-network.h.

8.80 multi-network.h

```
00001
00018 int8u emberGetCurrentNetwork(void);
00019
00027 EmberStatus emberSetCurrentNetwork(int8u
      index);
00028
00034 int8u emberGetCallbackNetwork(void);
00035
```

8.81 network-formation.h File Reference

Functions

- EmberStatus emberInit (void)
- void emberTick (void)
- EmberStatus emberNetworkInit (void)
- EmberStatus emberNetworkInitExtended (EmberNetworkInitStruct ∗networkInitStruct)
- EmberStatus emberFormNetwork (EmberNetworkParameters ∗parameters)
- EmberStatus emberPermitJoining (int8u duration)
- EmberStatus emberJoinNetwork (EmberNodeType nodeType, EmberNetworkParameters ∗parameters)
- EmberStatus emberLeaveNetwork (void)
- EmberStatus emberSendZigbeeLeave (EmberNodeId destination, EmberLeaveRequestFlags flags)
- EmberStatus emberFindAndRejoinNetworkWithReason (boolean haveCurrentNetworkKey, int32u channelMask, EmberRejoinReason reason)
- EmberStatus emberFindAndRejoinNetwork (boolean haveCurrentNetworkKey, int32u channelMask)
- EmberRejoinReason emberGetLastRejoinReason (void)
- EmberStatus emberRejoinNetwork (boolean haveCurrentNetworkKey)
- EmberStatus emberStartScan (EmberNetworkScanType scanType, int32u channelMask, int8u duration)
- EmberStatus emberStopScan (void)
- void emberScanCompleteHandler (int8u channel, EmberStatus status)
- void emberEnergyScanResultHandler (int8u channel, int8s maxRssiValue)
- void emberNetworkFoundHandler (EmberZigbeeNetwork ∗networkFound)
- boolean emberStackIsPerformingRejoin (void)
- EmberLeaveReason emberGetLastLeaveReason (EmberNodeId ∗returnNodeIdThatSentLeave)

8.81.1 Detailed Description

See Network Formation for documentation.

Definition in file network-formation.h.

8.82 network-formation.h

```
00001
00032 EmberStatus emberInit(void);
00033
00039 void emberTick(void);
00040
00059 EmberStatus emberNetworkInit(void);
00060
00069 EmberStatus emberNetworkInitExtended(
      EmberNetworkInitStruct* networkInitStruct);
00070
00082 EmberStatus emberFormNetwork(EmberNetworkParameters
       *parameters);
00083
00093 EmberStatus emberPermitJoining(int8u duration
      );
00094
00115 EmberStatus emberJoinNetwork(EmberNodeType
       nodeType,
00116                              EmberNetworkParameters *
      parameters);
00117
00127 EmberStatus emberLeaveNetwork(void);
00128
00143 EmberStatus emberSendZigbeeLeave(EmberNodeId
       destination,
00144                                  EmberLeaveRequestFlags
      flags);
00145
00146
00189 EmberStatus emberFindAndRejoinNetworkWithReason
      (boolean haveCurrentNetworkKey,
00190                                        int32u channelMask,
00191                                        EmberRejoinReason reason);
00192
00196 EmberStatus emberFindAndRejoinNetwork(
      boolean haveCurrentNetworkKey,
00197                                  int32u channelMask);
00198
00201 EmberRejoinReason emberGetLastRejoinReason(void);
00202
00207 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00208 EmberStatus emberRejoinNetwork(boolean
      haveCurrentNetworkKey);
00209 #else
00210 #define emberRejoinNetwork(haveKey) emberFindAndRejoinNetwork((haveKey), 0)
00211 #endif
00212
00247 EmberStatus emberStartScan(EmberNetworkScanType
       scanType,
00248                            int32u channelMask,
00249                            int8u duration);
00250
00255 EmberStatus emberStopScan(void);
00256
00271 void emberScanCompleteHandler( int8u channel,
      EmberStatus status );
00272
00281 void emberEnergyScanResultHandler(int8u
      channel, int8s maxRssiValue);
00282
00290 void emberNetworkFoundHandler(EmberZigbeeNetwork
       *networkFound);
00291
00298 boolean emberStackIsPerformingRejoin(void);
00299
00300
00314 EmberLeaveReason emberGetLastLeaveReason
```

```
          (EmberNodeId* returnNodeIdThatSentLeave);
00315
00316
```

8.83   network-manager.h File Reference

```
#include <CONFIGURATION_HEADER>
```

Macros

- #define NM_WARNING_LIMIT
- #define NM_WINDOW_SIZE
- #define NM_CHANNEL_MASK
- #define NM_WATCHLIST_SIZE

Functions

- void nmUtilWarningHandler (void)
- boolean nmUtilProcessIncoming (EmberApsFrame *apsFrame, int8u messageLength, int8u *message)
- EmberStatus nmUtilChangeChannelRequest (void)

8.83.1   Detailed Description

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

Definition in file network-manager.h.

8.84   network-manager.h

```
00001
00090 #include CONFIGURATION_HEADER
00091
00092 // The application is notified via nmUtilWarningHandler
00093 // if NM_WARNING_LIMIT unsolicited scan reports are received
00094 // within NM_WINDOW_SIZE minutes.  To save flash and RAM,
00095 // the actual timing is approximate.
00096 #ifndef NM_WARNING_LIMIT
00097   #define NM_WARNING_LIMIT 16
00098 #endif
00099
00100 #ifndef NM_WINDOW_SIZE
00101   #define NM_WINDOW_SIZE 4
00102 #endif
00103
00104 // The channels that should be used by the network manager.
00105
00106 #ifndef NM_CHANNEL_MASK
00107   #define NM_CHANNEL_MASK EMBER_ALL_802_15_4_CHANNELS_MASK
00108 #endif
00109
00110 // The number of channels used in the NM_CHANNEL_MASK.
00111
00112 #ifndef NM_WATCHLIST_SIZE
00113   #define NM_WATCHLIST_SIZE 16
00114 #endif
00115
00122 void nmUtilWarningHandler(void);
```

```
00123
00132 boolean nmUtilProcessIncoming(EmberApsFrame *
      apsFrame,
00133                                  int8u messageLength,
00134                                  int8u* message);
00135
00139 EmberStatus nmUtilChangeChannelRequest(
      void);
00140
```

8.85   nvic-config.h File Reference

8.86   nvic-config.h

```
00001
00029 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00030
00031 // \b NOTE NOTE NOTE NOTE NOTE NOTE - The physical layout of this file, that
00032 // means the white space, is CRITICAL!  Since this file is \#include'ed by
00033 // the assembly file isr-stubs.s79, the white space in this file translates
00034 // into the white space in that file and the assembler has strict requirements.
00035 // Specifically, there must be white space *before* each "SEGMENT()" and there
00036 // must be an *empty line* between each "EXCEPTION()" and "SEGMENT()".
00037 //
00038 // \b NOTE NOTE NOTE - The order of the EXCEPTIONS in this file is critical
00039 // since it translates to the order they are placed into the vector table in
00040 // cstartup.
00041 //
00042 // The purpose of this file is to consolidate NVIC configuration, this
00043 // includes basic exception handler (ISR) function definitions, into a single
00044 // place where it is easily tracked and changeable.
00045 //
00046 // There are 5 bits of priority yielding 32 levels of priority.  A lower
00047 // priority number yields the highest priority.  As such, 0 is the highest
00048 // priority while 31 is the lowest priority.
00049 //
00050 // NOTE: Do NOT exceed a priority level of 31 as the value will be truncated
00051 // and treated as a much higher priority.  Only the upper 5 of the 8 bit
00052 // priority fields are effective - bits [7:3].  As such, when the priorities
00053 // defined here are used they are bit shifted left by 3 to move them into the
00054 // upper 5 bits.
00055
00056 //The 'PRIGROUP' field is 5 bits large and indicates how many of the most
00057 //significant bits within the 8-bit priority field for a given exception
00058 //number are allocated to preemptive priority grouping versus non-preemptive
00059 //subpriority.  This value sets the "binary" point in the priority field
00060 //where the separation occurs.
00061 //The table below shows for each PRIGROUP value (the PRIGROUP value is the
00062 //number on the far left) the priority groups in ()
00063 //with the subpriorities in [].  When defining an exception's priority, use
00064 //one of the numbers in [].
00065 //0=7:1=(0)[0],(1)[1],(2)[2],(3)[3],(4)[4],(5)[5],(6)[6],(7)[7]
00066 //       (8)[8],(9)[9],(10)[10],(11)[11],(12)[12],(13)[13],(14)[14],(15)[15],
00067 //
      (16)[16],(17)[17],(18)[18],(19)[19],(20)[20],(21)[21],(22)[22],(23)[23],
00068 //       (24)[24],(25)[25],(26)[26],(27)[27],(28)[28],(29)[29],(30)[30],(31)[31]
00069 //1=6:2=(0)[0],(1)[1],(2)[2],(3)[3],(4)[4],(5)[5],(6)[6],(7)[7]
00070 //       (8)[8],(9)[9],(10)[10],(11)[11],(12)[12],(13)[13],(14)[14],(15)[15],
00071 //
      (16)[16],(17)[17],(18)[18],(19)[19],(20)[20],(21)[21],(22)[22],(23)[23],
00072 //       (24)[24],(25)[25],(26)[26],(27)[27],(28)[28],(29)[29],(30)[30],(31)[31]
00073 //2=5:3=(0)[0],(1)[1],(2)[2],(3)[3],(4)[4],(5)[5],(6)[6],(7)[7],
00074 //       (8)[8],(9)[9],(10)[10],(11)[11],(12)[12],(13)[13],(14)[14],(15)[15],
00075 //
      (16)[16],(17)[17],(18)[18],(19)[19],(20)[20],(21)[21],(22)[22],(23)[23],
00076 //       (24)[24],(25)[25],(26)[26],(27)[27],(28)[28],(29)[29],(30)[30],(31)[31]
00077 //3=4:4=(0)[0-1],(1)[2-3],(2)[4-5],(3)[6-7],
00078 //       (4)[8-9],(5)[10-11],(6)[12-13],(7)[14-15],
00079 //       (8)[16-17],(9)[18-19],(10)[20-21],(11)[22-23],
00080 //       (12)[24-25],(13)[26-27],(14)[28-29],(15)[30-31]
00081 //4=3:5=(0)[0-3],(1)[4-7],(2)[8-11],(3)[12-15],
00082 //       (4)[16-19],(5)[20-23],(6)[24-27],(7)[28-31]
00083 //5=2:6=(0)[0-7],(1)[8-15],(2)[16-23],(3)[24-31]
00084 //6=1:7=(0)[0-15],(1)[16-31]
```

```
00085 //7=0:8=(0)[0-31]
00086 //Provide 8 levels of preemptive priority and 4 of "tie-breaker" priority.
00087 //Priorities from highest to lowest are as follows:
00088 // 0  - faults
00089 // 4  - not used
00090 // 8  - PendSV for deep sleep, SysTick for idling, MAC Tmr for idling during
       TX,
00091 //        management interrupt for XTAL biasing..
00092 // 12 - Priority level used by DISABLE_INTERRUPTS() and INTERRUPTS_OFF()
00093 // 16 - normal interrupts
00094 // 20 - not used
00095 // 24 - not used
00096 // 28 - debug
00097 #define PRIGROUP_POSITION 4
00098
00099 // Priority level used by DISABLE_INTERRUPTS() and INTERRUPTS_OFF()
00100 // Must be lower priority than pendsv
00101 // NOTE!! IF THIS VALUE IS CHANGED, SPRM.S79 MUST ALSO BE UPDATED
00102 #define INTERRUPTS_DISABLED_PRIORITY  (12 << 3)  // READ NOTE ABOVE!!
00103
00104
00105 //Exceptions with fixed priorities cannot be changed by software.  Simply make
00106 //them 0 since they are high priorities anyways.
00107 #define FIXED 0
00108 //Reserved exceptions are not instatiated in the hardware.  Therefore
00109 //exception priorities don't exist so just default them to lowest level.
00110 #define NONE  31
00111
00112 #ifndef SEGMENT
00113   #define SEGMENT()
00114 #endif
00115 #ifndef SEGMENT2
00116   #define SEGMENT2()
00117 #endif
00118 #ifndef PERM_EXCEPTION
00119   #define PERM_EXCEPTION(vectorNumber, functionName, priority) \
00120     EXCEPTION(vectorNumber, functionName, priority)
00121 #endif
00122
00123     // SEGMENT()
00124     //   **Place holder required by isr-stubs.s79 to define __CODE__**
00125     // SEGMENT2()
00126     //   **Place holder required by isr-stubs.s79 to define __THUMB__**
00127     // EXCEPTION(vectorNumber, functionName, priorityLevel)
00128     //   vectorNumber = exception number defined by hardware (not used
        anywhere)
00129     //   functionName = name of the function that the exception should trigger
00130     //   priorityLevel = priority level of the function
00131     // PERM_EXCEPTION
00132     //   is used to define an exception that should not be intercepted by the
00133     //   interrupt debugging logic, or that not should have a weak stub
        defined.
00134     //   Otherwise the definition is the same as EXCEPTION
00135
00136 //INTENTIONALLY INDENTED AND SPACED APART! Keep it that way!  See comment
       above!
00137     SEGMENT()
00138     SEGMENT2()
00139     PERM_EXCEPTION(01, halEntryPoint,   FIXED)  //Reset Handler
00140
00141     SEGMENT()
00142     SEGMENT2()
00143     EXCEPTION(02, halNmiIsr,          FIXED)  //NMI Handler
00144
00145     SEGMENT()
00146     SEGMENT2()
00147     EXCEPTION(03, halHardFaultIsr,     FIXED)  //Hard Fault Handler
00148
00149     SEGMENT()
00150     SEGMENT2()
00151     EXCEPTION(04, halMemoryFaultIsr,   0)      //Memory Fault Handler
00152
00153     SEGMENT()
00154     SEGMENT2()
00155     EXCEPTION(05, halBusFaultIsr,      0)      //Bus Fault Handler
00156
00157     SEGMENT()
00158     SEGMENT2()
00159     EXCEPTION(06, halUsageFaultIsr,    0)      //Usage Fault Handler
00160
```

```
00161    SEGMENT()
00162    SEGMENT2()
00163    EXCEPTION(07, halReserved07Isr,    NONE)   //Reserved
00164
00165    SEGMENT()
00166    SEGMENT2()
00167    EXCEPTION(08, halReserved08Isr,    NONE)   //Reserved
00168
00169    SEGMENT()
00170    SEGMENT2()
00171    EXCEPTION(09, halReserved09Isr,    NONE)   //Reserved
00172
00173    SEGMENT()
00174    SEGMENT2()
00175    EXCEPTION(10, halReserved10Isr,    NONE)   //Reserved
00176
00177    SEGMENT()
00178    SEGMENT2()
00179    EXCEPTION(11, halSvCallIsr,        16)     //SVCall Handler
00180
00181    SEGMENT()
00182    SEGMENT2()
00183    EXCEPTION(12, halDebugMonitorIsr,  16)     //Debug Monitor Handler
00184
00185    SEGMENT()
00186    SEGMENT2()
00187    EXCEPTION(13, halReserved13Isr,    NONE)   //Reserved
00188
00189    SEGMENT()
00190    SEGMENT2()
00191    EXCEPTION(14, halPendSvIsr,        8)      //PendSV Handler
00192
00193    SEGMENT()
00194    SEGMENT2()
00195    EXCEPTION(15, halSysTickIsr,       8)      //SysTick Handler
00196
00197    //The following handlers map to "External Interrupts 16 and above"
00198    //In the NVIC Interrupt registers, this corresponds to bits 16:0 with bit 0
00199    //being TIMER1 (exception 16) and bit 15 being IRQD (exception 15)
00200    SEGMENT()
00201    SEGMENT2()
00202    EXCEPTION(16, halTimer1Isr,        16)     //Timer 1 Handler
00203
00204    SEGMENT()
00205    SEGMENT2()
00206    EXCEPTION(17, halTimer2Isr,        16)     //Timer 2 Handler
00207
00208    SEGMENT()
00209    SEGMENT2()
00210    EXCEPTION(18, halManagementIsr,    8)       //Management Handler
00211
00212    SEGMENT()
00213    SEGMENT2()
00214    EXCEPTION(19, halBaseBandIsr,      16)     //BaseBand Handler
00215
00216    SEGMENT()
00217    SEGMENT2()
00218    EXCEPTION(20, halSleepTimerIsr,    16)     //Sleep Timer Handler
00219
00220    SEGMENT()
00221    SEGMENT2()
00222    EXCEPTION(21, halSc1Isr,           16)     //SC1 Handler
00223
00224    SEGMENT()
00225    SEGMENT2()
00226    EXCEPTION(22, halSc2Isr,           16)     //SC2 Handler
00227
00228    SEGMENT()
00229    SEGMENT2()
00230    EXCEPTION(23, halSecurityIsr,      16)     //Security Handler
00231
00232    //MAC Timer Handler must be higher priority than emRadioTransmitIsr
00233    // for idling during managed TX->RX turnaround to function correctly.
00234        // But it is >= 12 so it doesn't run when ATOMIC.
00235    SEGMENT()
00236    SEGMENT2()
00237    EXCEPTION(24, halStackMacTimerIsr, 12)      //MAC Timer Handler
00238
00239    SEGMENT()
00240    SEGMENT2()
```

```
00241    EXCEPTION(25, emRadioTransmitIsr,  16)    //MAC TX Handler
00242
00243    SEGMENT()
00244    SEGMENT2()
00245    EXCEPTION(26, emRadioReceiveIsr,   16)    //MAC RX Handler
00246
00247    SEGMENT()
00248    SEGMENT2()
00249    EXCEPTION(27, halAdcIsr,           16)    //ADC Handler
00250
00251    SEGMENT()
00252    SEGMENT2()
00253    EXCEPTION(28, halIrqAIsr,          16)    //GPIO IRQA Handler
00254
00255    SEGMENT()
00256    SEGMENT2()
00257    EXCEPTION(29, halIrqBIsr,          16)    //GPIO IRQB Handler
00258
00259    SEGMENT()
00260    SEGMENT2()
00261    EXCEPTION(30, halIrqCIsr,          16)    //GPIO IRQC Handler
00262
00263    SEGMENT()
00264    SEGMENT2()
00265    EXCEPTION(31, halIrqDIsr,          16)    //GPIO IRQD Handler
00266
00267    SEGMENT()
00268    SEGMENT2()
00269    EXCEPTION(32, halDebugIsr,         28)    //Debug Handler
00270
00271 #undef SEGMENT
00272 #undef SEGMENT2
00273 #undef PERM_EXCEPTION
00274
00275 #endif //DOXYGEN_SHOULD_SKIP_THIS
00276
```

8.87   packet-buffer.h File Reference

Macros

- #define LOG_PACKET_BUFFER_SIZE
- #define PACKET_BUFFER_SIZE
- #define EMBER_NULL_MESSAGE_BUFFER
- #define emberMessageBufferLength(buffer)

Functions

- XAP2B_PAGEZERO_ON int8u ∗ emberMessageBufferContents (EmberMessageBuffer buffer)
- void emberSetMessageBufferLength (EmberMessageBuffer buffer, int8u newLength)
- void emberHoldMessageBuffer (EmberMessageBuffer buffer)
- void emberReleaseMessageBuffer (EmberMessageBuffer buffer)
- int8u emberPacketBufferFreeCount (void)

Buffer Functions

- #define emberStackBufferLink(buffer)
- #define emberSetStackBufferLink(buffer, newLink)
- #define emberAllocateStackBuffer()
- EmberMessageBuffer emberAllocateLinkedBuffers (int8u count)
- EmberMessageBuffer emberFillStackBuffer (int16u count,...)

Linked Buffer Utilities

The plural "buffers" in the names of these procedures is a reminder that they deal with linked chains of buffers.

- EmberMessageBuffer emberFillLinkedBuffers (int8u ∗contents, int8u length)
- void emberCopyToLinkedBuffers (int8u ∗contents, EmberMessageBuffer buffer, int8u startIndex, int8u length)
- void emberCopyFromLinkedBuffers (EmberMessageBuffer buffer, int8u startIndex, int8u ∗contents, int8u length)
- void emberCopyBufferBytes (EmberMessageBuffer to, int16u toIndex, EmberMessageBuffer from, int16u fromIndex, int16u count)
- EmberStatus emberAppendToLinkedBuffers (EmberMessageBuffer buffer, int8u ∗contents, int8u length)
- EmberStatus emberAppendPgmToLinkedBuffers (EmberMessageBuffer buffer, PGM_P contents, int8u length)
- EmberStatus emberAppendPgmStringToLinkedBuffers (EmberMessageBuffer buffer, PGM_P suffix)
- EmberStatus emberSetLinkedBuffersLength (EmberMessageBuffer buffer, int8u length)
- int8u ∗ emberGetLinkedBuffersPointer (EmberMessageBuffer buffer, int8u index)
- XAP2B_PAGEZERO_ON int8u emberGetLinkedBuffersByte (EmberMessageBuffer buffer, int8u index)
- XAP2B_PAGEZERO_OFF void emberSetLinkedBuffersByte (EmberMessageBuffer buffer, int8u index, int8u byte)
- int16u emberGetLinkedBuffersLowHighInt16u (EmberMessageBuffer buffer, int8u index)
- void emberSetLinkedBuffersLowHighInt16u (EmberMessageBuffer buffer, int8u index, int16u value)
- int32u emberGetLinkedBuffersLowHighInt32u (EmberMessageBuffer buffer, int8u index)
- void emberSetLinkedBuffersLowHighInt32u (EmberMessageBuffer buffer, int8u index, int32u value)
- EmberMessageBuffer emberCopyLinkedBuffers (EmberMessageBuffer buffer)
- EmberMessageBuffer emberMakeUnsharedLinkedBuffer (EmberMessageBuffer buffer, boolean isShared)

8.87.1 Detailed Description

Packet buffer allocation and management routines See Packet Buffers for documentation.

Definition in file packet-buffer.h.

8.88 packet-buffer.h

```
00001
00010 // The old overview was for the wrong audience. A new overview should be
      written.
00011
00032 #ifndef __PACKET_BUFFER_H__
00033 #define __PACKET_BUFFER_H__
00034
00038 #define LOG_PACKET_BUFFER_SIZE 5
00039
00042 #define PACKET_BUFFER_SIZE (1 << LOG_PACKET_BUFFER_SIZE)
00043
00045 #define EMBER_NULL_MESSAGE_BUFFER 0xFF
00046
00057 XAP2B_PAGEZERO_ON
00058 int8u *emberMessageBufferContents(
      EmberMessageBuffer buffer);
```

```
00059 XAP2B_PAGEZERO_OFF
00060
00061 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00062 // An old name.
00063 #define emberLinkedBufferContents(buffer) emberMessageBufferContents(buffer)
00064 #endif
00065
00073 #define emberMessageBufferLength(buffer)    (emMessageBufferLengths[buffer])
00074
00085 void emberSetMessageBufferLength(EmberMessageBuffer
      buffer, int8u newLength);
00086
00087 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00088 #ifdef EM_DEBUG_BUFFER_USE
00089 #define EM_BUFFER_DEBUG(X) X
00090 #define EM_BUFFER_FILE_LOC , __FILE__, __LINE__
00091 #define EM_BUFFER_FILE_DECL , char *file, int line
00092 #define EM_BUFFER_FILE_VALUE , file, line
00093 #else
00094 #define EM_BUFFER_DEBUG(X)
00095 #define EM_BUFFER_FILE_LOC
00096 #define EM_BUFFER_FILE_DECL
00097 #define EM_BUFFER_FILE_VALUE
00098 #endif
00099
00100 #ifdef EMBER_TEST
00101 #define EM_ASSERT_IS_NOT_FREE(buffer) \
00102  assert(emMessageBufferReferenceCounts[(buffer)] > 0)
00103 #define EM_ASSERT_IS_VALID_BUFFER(buffer) \
00104  assert(buffer < emPacketBufferCount)
00105 #else
00106 #define EM_ASSERT_IS_NOT_FREE(buffer)
00107 #define EM_ASSERT_IS_VALID_BUFFER(buffer)
00108 #endif  // EMBER_TEST
00109 #endif  // DOXYGEN_SHOULD_SKIP_THIS
00110
00116 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00117 void emberHoldMessageBuffer(EmberMessageBuffer
      buffer);
00118 #else
00119
00120 #ifdef EMBER_TEST
00121 #define emberHoldMessageBuffer(buffer)                                  \
00122 do {                                                                    \
00123   EmberMessageBuffer EM_HOLD_BUFFER_TEMP_XXX = (buffer);                \
00124   EM_ASSERT_IS_NOT_FREE(EM_HOLD_BUFFER_TEMP_XXX);                       \
00125   EM_ASSERT_IS_VALID_BUFFER(EM_HOLD_BUFFER_TEMP_XXX);                   \
00126   emHoldMessageBuffer(EM_HOLD_BUFFER_TEMP_XXX EM_BUFFER_FILE_LOC);      \
00127 } while (FALSE)
00128
00129 #else
00130 #define emberHoldMessageBuffer(buffer) emHoldMessageBuffer(buffer)
00131 #endif // EMBER_TEST
00132
00133 void emHoldMessageBuffer(EmberMessageBuffer buffer
      EM_BUFFER_FILE_DECL);
00134 #endif
00135
00141 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00142 void emberReleaseMessageBuffer(EmberMessageBuffer
      buffer);
00143 #else
00144
00145 #if defined(EMBER_TEST)
00146 #define emberReleaseMessageBuffer(buffer)                               \
00147 do {                                                                    \
00148   EmberMessageBuffer EM_RELEASE_BUFFER_TEMP_XXX = (buffer);             \
00149   EM_ASSERT_IS_NOT_FREE(EM_RELEASE_BUFFER_TEMP_XXX);                    \
00150   EM_ASSERT_IS_VALID_BUFFER(EM_RELEASE_BUFFER_TEMP_XXX);                \
00151   emReleaseMessageBuffer(EM_RELEASE_BUFFER_TEMP_XXX EM_BUFFER_FILE_LOC);  \
00152 } while (FALSE)
00153 #else
00154 #define emberReleaseMessageBuffer(buffer) emReleaseMessageBuffer(buffer)
00155 #endif // EMBER_TEST
00156
00157 XAP2B_PAGEZERO_ON
00158 void emReleaseMessageBuffer(EmberMessageBuffer buffer
      EM_BUFFER_FILE_DECL);
00159 XAP2B_PAGEZERO_OFF
00160 #endif //DOXYGEN_SHOULD_SKIP_THIS
```

```
00161
00162 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00163 extern int8u emPacketBufferCount;
00164 extern int8u *emPacketBufferData;
00165 extern int8u *emMessageBufferLengths;
00166 extern int8u *emMessageBufferReferenceCounts;
00167 extern EmberMessageBuffer *emPacketBufferLinks;
00168 extern EmberMessageBuffer *emPacketBufferQueueLinks;
00169 #endif //DOXYGEN_SHOULD_SKIP_THIS
00170
00175
00185 #define emberStackBufferLink(buffer) \
00186 (emPacketBufferLinks[(buffer)])
00187
00196 #define emberSetStackBufferLink(buffer, newLink) \
00197 (emPacketBufferLinks[(buffer)] = (newLink))
00198
00205 #define emberAllocateStackBuffer() (emberAllocateLinkedBuffers(1))
00206
00215 EmberMessageBuffer emberAllocateLinkedBuffers
      (int8u count);
00216
00228 EmberMessageBuffer emberFillStackBuffer(
      int16u count, ...);
00230
00237
00251 EmberMessageBuffer emberFillLinkedBuffers
      (int8u *contents, int8u length);
00252
00266 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00267 void
00268 emberCopyToLinkedBuffers(int8u *contents,
00269                          EmberMessageBuffer buffer,
00270                          int8u startIndex,
00271                          int8u length);
00272 #else
00273 #define emberCopyToLinkedBuffers(contents, buffer, startIndex, length) \
00274 emReallyCopyToLinkedBuffers((PGM_P) (contents), (buffer), (startIndex),
      (length), 1)
00275 XAP2B_PAGEZERO_ON
00276 void
00277 emReallyCopyToLinkedBuffers(PGM_P contents,
00278                            EmberMessageBuffer buffer,
00279                            int8u startIndex,
00280                            int8u length,
00281                            int8u direction);
00282 XAP2B_PAGEZERO_OFF
00283 #endif //DOXYGEN_SHOULD_SKIP_THIS
00284
00298 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00299 // To save flash this shares an implementation with emberCopyToLinkedBuffers().
00300 void
00301 emberCopyFromLinkedBuffers(EmberMessageBuffer
      buffer,
00302                            int8u startIndex,
00303                            int8u *contents,
00304                            int8u length);
00305 #else
00306 #define emberCopyFromLinkedBuffers(buffer, startIndex, contents, length) \
00307 emReallyCopyToLinkedBuffers((PGM_P) (contents), (buffer), (startIndex),
      (length), 0)
00308 #endif
00309
00323 void emberCopyBufferBytes(EmberMessageBuffer
      to,
00324                          int16u toIndex,
00325                          EmberMessageBuffer from,
00326                          int16u fromIndex,
00327                          int16u count);
00328
00343 EmberStatus emberAppendToLinkedBuffers(
      EmberMessageBuffer buffer,
00344                                         int8u *contents,
00345                                         int8u length);
00346
00360 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00361 // To save flash this shares an implementation with
00362 // emberAppendToLinkedBuffers().
00363 EmberStatus emberAppendPgmToLinkedBuffers
      (EmberMessageBuffer buffer,
```

```
00364                                                    PGM_P contents,
00365                                                    int8u length);
00366 #else
00367 #define emberAppendPgmToLinkedBuffers(buffer, contents, length) \
00368  (emAppendToLinkedBuffers((buffer), (PGM_P) (contents), (length), 2))
00369
00370 EmberStatus emAppendToLinkedBuffers(EmberMessageBuffer
     buffer,
00371                                          PGM_P contents,
00372                                          int8u length,
00373                                          int8u direction);
00374 #endif
00375
00385 EmberStatus emberAppendPgmStringToLinkedBuffers
     (EmberMessageBuffer buffer, PGM_P suffix);
00386
00399 EmberStatus emberSetLinkedBuffersLength(
     EmberMessageBuffer buffer, int8u length);
00400
00410 int8u *emberGetLinkedBuffersPointer(
     EmberMessageBuffer buffer, int8u index);
00411
00421 XAP2B_PAGEZERO_ON
00422 int8u emberGetLinkedBuffersByte(
     EmberMessageBuffer buffer, int8u index);
00423 XAP2B_PAGEZERO_OFF
00424
00434 void emberSetLinkedBuffersByte(EmberMessageBuffer
     buffer, int8u index, int8u byte);
00435
00445 int16u emberGetLinkedBuffersLowHighInt16u
     (EmberMessageBuffer buffer,
00446                                          int8u index);
00447
00457 void emberSetLinkedBuffersLowHighInt16u(
     EmberMessageBuffer buffer,
00458                                          int8u index,
00459                                          int16u value);
00460
00470 int32u emberGetLinkedBuffersLowHighInt32u
     (EmberMessageBuffer buffer,
00471                                          int8u index);
00472
00482 void emberSetLinkedBuffersLowHighInt32u(
     EmberMessageBuffer buffer,
00483                                          int8u index,
00484                                          int32u value);
00485
00493 EmberMessageBuffer emberCopyLinkedBuffers
     (EmberMessageBuffer buffer);
00494
00507 EmberMessageBuffer emberMakeUnsharedLinkedBuffer
     (EmberMessageBuffer buffer, boolean isShared);
00508
00510
00516 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00517 extern int8u emPacketBufferFreeCount;
00518 #define emberPacketBufferFreeCount() (emPacketBufferFreeCount)
00519 #else
00520 int8u emberPacketBufferFreeCount(void);
00521 #endif
00522
00525 #endif  // __PACKET_BUFFER_H__
```

8.89   platform-common.h File Reference

Generic Types

- #define TRUE
- #define FALSE
- #define NULL

Bit Manipulation Macros

- #define BIT(x)
- #define BIT32(x)
- #define SETBIT(reg, bit)
- #define SETBITS(reg, bits)
- #define CLEARBIT(reg, bit)
- #define CLEARBITS(reg, bits)
- #define READBIT(reg, bit)
- #define READBITS(reg, bits)

Byte Manipulation Macros

- #define LOW_BYTE(n)
- #define HIGH_BYTE(n)
- #define HIGH_LOW_TO_INT(high, low)
- #define BYTE_0(n)
- #define BYTE_1(n)
- #define BYTE_2(n)
- #define BYTE_3(n)
- #define COUNTOF(a)

Time Manipulation Macros

- #define elapsedTimeInt8u(oldTime, newTime)
- #define elapsedTimeInt16u(oldTime, newTime)
- #define elapsedTimeInt32u(oldTime, newTime)
- #define MAX_INT8U_VALUE
- #define HALF_MAX_INT8U_VALUE
- #define timeGTorEqualInt8u(t1, t2)
- #define MAX_INT16U_VALUE
- #define HALF_MAX_INT16U_VALUE
- #define timeGTorEqualInt16u(t1, t2)
- #define MAX_INT32U_VALUE
- #define HALF_MAX_INT32U_VALUE
- #define timeGTorEqualInt32u(t1, t2)

8.89.1   Detailed Description

See Common PLATFORM_HEADER Configuration for detailed documentation.

Definition in file platform-common.h.

8.90 platform-common.h

```
00001
00019 #ifndef PLATCOMMONOKTOINCLUDE
00020   //  This header should only be included by a PLATFORM_HEADER
00021   #error  platform-common.h should not be included directly
00022 #endif
00023
00024 #ifndef __PLATFORMCOMMON_H__
00025 #define __PLATFORMCOMMON_H__
00026
00027 // Many of the common definitions must be explicitly enabled by the
00028 //  particular PLATFORM_HEADER being used
00030
00031
00032 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00033
00034 // The XAP2b compiler uses these macros to enable and disable placement
00035 // in zero-page memory.  All other platforms do not have zero-page memory
00036 // so these macros define to nothing.
00037 #ifndef _HAL_USING_XAP2B_PRAGMAS_
00038   #define XAP2B_PAGEZERO_ON
00039   #define XAP2B_PAGEZERO_OFF
00040 #endif
00041
00042 #endif //DOXYGEN_SHOULD_SKIP_THIS
00043
00044
00046 #ifdef _HAL_USE_COMMON_PGM_
00047
00054   #define PGM     const
00055
00059   #define PGM_P   const char *
00060
00064   #define PGM_PU  const unsigned char *
00065
00066
00072   #define PGM_NO_CONST
00073
00074 #endif //_HAL_USE_COMMON_PGM_
00075
00076
00078 #ifdef _HAL_USE_COMMON_DIVMOD_
00079
00092   #define halCommonUDiv32By16(x, y) ((int16u) (((int32u) (x)) / ((int16u)
      (y))))
00093
00099   #define halCommonSDiv32By16(x, y) ((int16s) (((int32s) (x)) / ((int16s)
      (y))))
00100
00106   #define halCommonUMod32By16(x, y) ((int16u) (((int32u) (x)) % ((int16u)
      (y))))
00107
00113   #define halCommonSMod32By16(x, y) ((int16s) (((int32s) (x)) % ((int16s)
      (y))))
00114
00115 #endif //_HAL_USE_COMMON_DIVMOD_
00116
00117
00119 #ifdef _HAL_USE_COMMON_MEMUTILS_
00120
00132
00136   void halCommonMemCopy(void *dest, const void *src, int16u bytes);
00137
00138
00142   void halCommonMemSet(void *dest, int8u val, int16u bytes);
00143
00144
00148   int8s halCommonMemCompare(const void *source0, const void *source1,
      int16u bytes);
00149
00150
00155   int8s halCommonMemPGMCompare(const void *source0, const void
      PGM_NO_CONST *source1, int16u bytes);
00156
00161   void halCommonMemPGMCopy(void* dest, const void PGM_NO_CONST *source, int16u
       bytes);
00162
00166   #define MEMSET(d,v,l)  halCommonMemSet(d,v,l)
```

```
00167   #define MEMCOPY(d,s,l) halCommonMemCopy(d,s,l)
00168   #define MEMCOMPARE(s0,s1,l) halCommonMemCompare(s0, s1, l)
00169   #define MEMPGMCOMPARE(s0,s1,l) halCommonMemPGMCompare(s0, s1, l)
00170
00172 #endif //_HAL_USE_COMMON_MEMUTILS_
00173
00174
00175
00176
00177
00178
00179
00180
00181
00183 //  The following sections are common on all platforms
00185
00187
00195 #define TRUE  1
00196
00200 #define FALSE 0
00201
00202 #ifndef NULL
00203
00206 #define NULL ((void *)0)
00207 #endif
00208
00210
00211
00216
00220 #define BIT(x) (1U << (x))  // Unsigned avoids compiler warnings re BIT(15)
00221
00225 #define BIT32(x) (((int32u) 1) << (x))
00226
00232 #define SETBIT(reg, bit)      reg |= BIT(bit)
00233
00239 #define SETBITS(reg, bits)    reg |= (bits)
00240
00246 #define CLEARBIT(reg, bit)    reg &= ~(BIT(bit))
00247
00253 #define CLEARBITS(reg, bits)  reg &= ~(bits)
00254
00258 #define READBIT(reg, bit)     (reg & (BIT(bit)))
00259
00264 #define READBITS(reg, bits)   (reg & (bits))
00265
00267
00268
00270
00274
00278 #define LOW_BYTE(n)                   ((int8u)((n) & 0xFF))
00279
00283 #define HIGH_BYTE(n)                  ((int8u)(LOW_BYTE((n) >> 8)))
00284
00289 #define HIGH_LOW_TO_INT(high, low) (                         \
00290                                 ((int16u) (( (int16u) (high) ) << 8)) +  \
00291                                 (  (int16u) ( (low) & 0xFF))             \
00292                              )
00293
00297 #define BYTE_0(n)                     ((int8u)((n) & 0xFF))
00298
00302 #define BYTE_1(n)                     ((int8u)(BYTE_0((n) >> 8)))
00303
00307 #define BYTE_2(n)                     ((int8u)(BYTE_0((n) >> 16)))
00308
00312 #define BYTE_3(n)                     ((int8u)(BYTE_0((n) >> 24)))
00313
00317 #define COUNTOF(a) (sizeof(a)/sizeof(a[0]))
00318
00320
00321
00323
00327
00332 #define elapsedTimeInt8u(oldTime, newTime)       \
00333   ((int8u) ((int8u)(newTime) - (int8u)(oldTime)))
00334
00339 #define elapsedTimeInt16u(oldTime, newTime)      \
00340   ((int16u) ((int16u)(newTime) - (int16u)(oldTime)))
00341
00346 #define elapsedTimeInt32u(oldTime, newTime)      \
00347   ((int32u) ((int32u)(newTime) - (int32u)(oldTime)))
```

```
00348
00353 #define MAX_INT8U_VALUE        (0xFF)
00354 #define HALF_MAX_INT8U_VALUE  (0x80)
00355 #define timeGTorEqualInt8u(t1, t2)           \
00356   (elapsedTimeInt8u(t2, t1) <= (HALF_MAX_INT8U_VALUE))
00357
00362 #define MAX_INT16U_VALUE       (0xFFFF)
00363 #define HALF_MAX_INT16U_VALUE (0x8000)
00364 #define timeGTorEqualInt16u(t1, t2)           \
00365   (elapsedTimeInt16u(t2, t1) <= (HALF_MAX_INT16U_VALUE))
00366
00371 #define MAX_INT32U_VALUE       (0xFFFFFFFFL)
00372 #define HALF_MAX_INT32U_VALUE (0x80000000L)
00373 #define timeGTorEqualInt32u(t1, t2)           \
00374   (elapsedTimeInt32u(t2, t1) <= (HALF_MAX_INT32U_VALUE))
00375
00377
00378
00379
00380 #endif //__PLATFORMCOMMON_H__
00381
```

8.91   random.h File Reference

Functions

- void halStackSeedRandom (int32u seed)
- int16u halCommonGetRandom (void)

8.91.1   Detailed Description

See Random Number Generation for detailed documentation.

Definition in file random.h.

8.92   random.h

```
00001
00016 #ifndef __RANDOM_H__
00017 #define __RANDOM_H__
00018
00026 void halStackSeedRandom(int32u seed);
00027
00036 #if defined( EMBER_TEST )
00037 #define halCommonGetRandom() halCommonGetRandomTraced(__FILE__, __LINE__)
00038 int16u halCommonGetRandomTraced(char *file, int line);
00039 #else
00040 int16u halCommonGetRandom(void);
00041 #endif
00042
00046 #endif //__RANDOM_H__
00047
00048
```

8.93   reset-def.h File Reference

8.93.1   Detailed Description

Definitions for all the reset cause types.

Definition in file reset-def.h.

8.94   reset-def.h

```
00001
00035 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00036
00037 // ********************
00038 // This file is specifically kept wider than 80 columns in order to keep the
        information
00039 //   well organized and easy to read by glancing down the columns
00040 // ********************
00041
00042 // The reset types of the EM300 series have both a base type and an
00043 //  extended type.  The extended type is a 16-bit value which has the base
00044 //  type in the upper 8-bits, and the extended classification in the
00045 //  lower 8-bits
00046 // For backwards compatibility with other platforms, only the base type is
00047 //  returned by the halGetResetInfo() API.  For the full extended type, the
00048 //  halGetExtendedResetInfo() API should be called.
00049
00050
00051 // RESET_BASE_DEF macros take the following parameters:
00052 //    RESET_BASE_DEF(basename, value, string)  // description
00053 //        basename - the name used in the enum definition, expanded as
        RESET_basename
00054 //        value - the value of the enum definition
00055 //        string - the reset string, must be 3 characters
00056 //        description - a comment describing the cause
00057
00058 // RESET_EXT_DEF macros take the following parameters:
00059 //    RESET_EXT_DEF(basename, extname, extvalue, string)  // description
00060 //        basename - the base name used in the enum definition
00061 //        extname - the extended name used in the enum definition, expanded as
        RESET_basename_extname
00062 //        extvalue - the LSB value of the enum definition, combined with the
        value of the base value in the MSB
00063 //        string - the reset string, must be 3 characters
00064 //        description - a comment describing the cause
00065
00066 // ********************
00067 // This file is specifically kept wider than 80 columns in order to keep the
        information
00068 //   well organized and easy to read by glancing down the columns
00069 // (Yes, this comment is mentioned multiple times in the file)
00070 // ********************
00071
00072 //[[
00073 //   The first 3 (unknown, fib, and bootloader) reset base types and their
00074 //    extended values should never be changed, as they are used by
00075 //    bootloaders in the field which can never be changed.
00076 //   The later reset base and extended types may be changed over time
00077 //]]
00078
00079 RESET_BASE_DEF(UNKNOWN,             0x00,   "UNK")   // Underterminable cause
00080   RESET_EXT_DEF(UNKNOWN, UNKNOWN,    0x00,   "UNK")    // Undeterminable cause
00081
00082 RESET_BASE_DEF(FIB,                 0x01,   "FIB")     // Reset originated
        from the FIB bootloader
00083   RESET_EXT_DEF(FIB, GO,            0x00,   "GO ")       // FIB bootloader
        caused a reset to main flash
00084   RESET_EXT_DEF(FIB, BOOTLOADER,    0x01,   "BTL")       // FIB bootloader is
        instructing ember bootloader to run
00085   RESET_EXT_DEF(FIB, GO2,           0x02,   "GO2")       // GO2 (unused)
00086   RESET_EXT_DEF(FIB, GO3,           0x03,   "GO3")       // GO3 (unused)
00087   RESET_EXT_DEF(FIB, GO4,           0x04,   "GO4")       // GO4 (unused)
00088   RESET_EXT_DEF(FIB, GO5,           0x05,   "GO5")       // GO5 (unused)
00089   RESET_EXT_DEF(FIB, GO6,           0x06,   "GO6")       // GO6 (unused)
00090   RESET_EXT_DEF(FIB, GO7,           0x07,   "GO7")       // GO7 (unused)
00091   RESET_EXT_DEF(FIB, GO8,           0x08,   "GO8")       // GO8 (unused)
00092   RESET_EXT_DEF(FIB, GO9,           0x09,   "GO9")       // GO9 (unused)
00093   RESET_EXT_DEF(FIB, GOA,           0x0A,   "GOA")       // GOA (unused)
00094   RESET_EXT_DEF(FIB, GOB,           0x0B,   "GOB")       // GOB (unused)
00095   RESET_EXT_DEF(FIB, GOC,           0x0C,   "GOC")       // GOC (unused)
00096   RESET_EXT_DEF(FIB, GOD,           0x0D,   "GOD")       // GOD (unused)
00097   RESET_EXT_DEF(FIB, GOE,           0x0E,   "GOE")       // GOE (unused)
00098   RESET_EXT_DEF(FIB, GOF,           0x0F,   "GOF")       // GOF (unused)
00099   RESET_EXT_DEF(FIB, JUMP,          0x10,   "JMP")       // FIB bootloader is
        jumping to a specific flash address
00100   RESET_EXT_DEF(FIB, BAUDRATE,      0x11,   "BDR")       // FIB bootloader
        detected a high baud rate, causes ember bootloader to run
```

```
00101  RESET_EXT_DEF(FIB, UNPROTECT,      0x12,   "UPR")      // Read protection
       disabled, flash should be erased
00102  RESET_EXT_DEF(FIB, BOOTMODE,       0x13,   "BTM")      // BOOTMODE was not
       held long enough
00103  RESET_EXT_DEF(FIB, MISMATCH,       0x14,   "MSM")      // MISMATCHED FIB
       Bootloader & Part Data
00104  RESET_EXT_DEF(FIB, FATAL,          0x15,   "FTL")      // FIB Fatal Error
00105
00106 RESET_BASE_DEF(BOOTLOADER,          0x02,   "BTL")   // Reset relates to an
       Ember bootloader
00107  RESET_EXT_DEF(BOOTLOADER, UNKNOWN, 0x00,    "UNK")   // Unknown
       bootloader cause (should never occur)
00108  RESET_EXT_DEF(BOOTLOADER, GO,      0x01,    "GO ")     // Bootloader caused
       reset telling app to run
00109  RESET_EXT_DEF(BOOTLOADER, BOOTLOAD, 0x02,   "BTL")    // Application
       requested that bootloader runs
00110  RESET_EXT_DEF(BOOTLOADER, BADIMAGE, 0x03,   "BAD")    // Application
       bootloader detect bad external upgrade image
00111  RESET_EXT_DEF(BOOTLOADER, FATAL,   0x04,    "FTL")    // Fatal Error or
       assert in bootloader
00112  RESET_EXT_DEF(BOOTLOADER, FORCE,   0x05,    "FRC")    // Forced bootloader
       activation
00113  RESET_EXT_DEF(BOOTLOADER, OTAVALID, 0x06,   "OTA")    // OTA Bootloader
       mode activation
00114  RESET_EXT_DEF(BOOTLOADER, DEEPSLEEP, 0x07,  "DSL")    // Bootloader
       initiated deep sleep
00115
00116 //[[ -- Reset types below here may be changed in the future if absolutely
       necessary -- ]]
00117
00118 RESET_BASE_DEF(EXTERNAL,            0x03,    "EXT")   // External reset
       trigger
00119  RESET_EXT_DEF(EXTERNAL, UNKNOWN,   0x00,    "UNK")    // Unknown external
       cause (should never occur)
00120  RESET_EXT_DEF(EXTERNAL, PIN,       0x01,    "PIN")    // External pin reset
00121
00122 RESET_BASE_DEF(POWERON,             0x04,    "PWR")   // Poweron reset type,
       supply voltage < power-on threshold
00123  RESET_EXT_DEF(POWERON, UNKNOWN,    0x00,    "UNK")    // Unknown poweron
       reset (should never occur)
00124  RESET_EXT_DEF(POWERON, HV,         0x01,    "HV ")    // High voltage
       poweron
00125  RESET_EXT_DEF(POWERON, LV,         0x02,    "LV ")    // Low voltage
       poweron
00126
00127 RESET_BASE_DEF(WATCHDOG,            0x05,    "WDG")   // Watchdog reset
       occurred
00128  RESET_EXT_DEF(WATCHDOG, UNKNWON,   0x00,    "UNK")    // Unknown watchdog
       reset (should never occur)
00129  RESET_EXT_DEF(WATCHDOG, EXPIRED,   0x01,    "EXP")    // Watchdog timer
       expired
00130  RESET_EXT_DEF(WATCHDOG, CAUGHT,    0x02,    "LWM")    // Watchdog low
       watermark expired and caught extended info
00131
00132 RESET_BASE_DEF(SOFTWARE,            0x06,    " SW")   // Software triggered
       reset
00133  RESET_EXT_DEF(SOFTWARE, UNKNOWN,   0x00,    "UNK")    // Unknown software
       cause
00134  RESET_EXT_DEF(SOFTWARE, REBOOT,    0x01,    "RBT")    // General software
       reboot
00135  RESET_EXT_DEF(SOFTWARE, DEEPSLEEP, 0x02,    "DSL")    // App initiated deep
       sleep
00136
00137 RESET_BASE_DEF(CRASH,               0x07,    "CRS")   // Software crash
00138  RESET_EXT_DEF(CRASH, UNKNOWN,      0x00,    "UNK")    // Unknown crash
00139  RESET_EXT_DEF(CRASH, ASSERT,       0x01,    "AST")    // a self-check
       assert in the code failed
00140
00141 RESET_BASE_DEF(FLASH,               0x08,    "FSH")   // Flash failure cause
       reset
00142  RESET_EXT_DEF(FLASH, UNKNWON,      0x00,    "UNK")    // Unknown flash
       failure
00143  RESET_EXT_DEF(FLASH, VERIFY,       0x01,    "VFY")    // Flash write verify
       failure
00144  RESET_EXT_DEF(FLASH, INHIBIT,      0x02,    "INH")    // Flash write
       inhibited: already written
00145
00146 RESET_BASE_DEF(FATAL,               0x09,    "BAD")   // A non-recoverable
       fatal error occurred
00147  RESET_EXT_DEF(FATAL, UNKNOWN,      0x00,    "UNK")    // Unknown fatal
```

```
              error (should never occur)
00148   RESET_EXT_DEF(FATAL, LOCKUP,        0x01,      "LCK")    // CPU Core locked up
00149   RESET_EXT_DEF(FATAL, CRYSTAL,       0x02,      "XTL")    // 24MHz crystal
        failure
00150   RESET_EXT_DEF(FATAL, OPTIONBYTE,    0x03,      "OBF")    // option byte
        complement error
00151
00152 RESET_BASE_DEF(FAULT,                 0x0A,      "FLT")    // A access fault
        occurred
00153   RESET_EXT_DEF(FAULT, UNKNOWN,       0x00,      "UNK")    // An unknown fault
        occurred
00154   RESET_EXT_DEF(FAULT, HARD,          0x01,      "HRD")    // Hard fault
00155   RESET_EXT_DEF(FAULT, MEM,           0x02,      "MEM")    // Memory protection
        violation
00156   RESET_EXT_DEF(FAULT, BUS,           0x03,      "BUS")    // Bus fault
00157   RESET_EXT_DEF(FAULT, USAGE,         0x04,      "USG")    // Usage fault
00158   RESET_EXT_DEF(FAULT, DBGMON,        0x05,      "DBG")    // Debug monitor
        fault
00159   RESET_EXT_DEF(FAULT, PROTDMA,       0x06,      "DMA")    // DMA RAM protection
        violation
00160   RESET_EXT_DEF(FAULT, BADVECTOR,     0x07,      "VCT")    // Uninitialized
        interrupt vector
00161
00162 #endif //DOXYGEN_SHOULD_SKIP_THIS
00163
```

8.95   security.h File Reference

```
#include "stack/include/trust-center.h"
```

Macros

- #define EMBER_JOIN_NO_PRECONFIG_KEY_BITMASK
- #define EMBER_JOIN_PRECONFIG_KEY_BITMASK

Functions

- EmberStatus emberSetInitialSecurityState (EmberInitialSecurityState ∗state)
- EmberStatus emberSetExtendedSecurityBitmask (EmberExtendedSecurityBitmask mask)
- EmberStatus emberGetExtendedSecurityBitmask (EmberExtendedSecurityBitmask ∗mask)
- EmberStatus emberGetCurrentSecurityState (EmberCurrentSecurityState ∗state)
- EmberStatus emberGetKey (EmberKeyType type, EmberKeyStruct ∗keyStruct)
- boolean emberHaveLinkKey (EmberEUI64 remoteDevice)
- EmberStatus emberGenerateRandomKey (EmberKeyData ∗keyAddress)
- void emberSwitchNetworkKeyHandler (int8u sequenceNumber)
- EmberStatus emberRequestLinkKey (EmberEUI64 partner)
- void emberZigbeeKeyEstablishmentHandler (EmberEUI64 partner, EmberKeyStatus status)
- EmberStatus emberGetKeyTableEntry (int8u index, EmberKeyStruct ∗result)
- EmberStatus emberSetKeyTableEntry (int8u index, EmberEUI64 address, boolean linkKey, Ember-KeyData ∗keyData)
- EmberStatus emberAddOrUpdateKeyTableEntry (EmberEUI64 address, boolean linkKey, Ember-KeyData ∗keyData)
- int8u emberFindKeyTableEntry (EmberEUI64 address, boolean linkKey)
- EmberStatus emberEraseKeyTableEntry (int8u index)
- EmberStatus emberClearKeyTable (void)
- EmberStatus emberStopWritingStackTokens (void)

- EmberStatus emberStartWritingStackTokens (void)
- boolean emberWritingStackTokensEnabled (void)
- EmberStatus emberApsCryptMessage (boolean encrypt, EmberMessageBuffer buffer, int8u aps-HeaderEndIndex, EmberEUI64 remoteEui64)
- EmberStatus emberGetMfgSecurityConfig (EmberMfgSecurityStruct ∗settings)
- EmberStatus emberSetMfgSecurityConfig (int32u magicNumber, const EmberMfgSecurityStruct ∗settings)

8.95.1   Detailed Description

EmberZNet security API. See Security for documentation.

Definition in file security.h.

8.96   security.h

```
00001
00029 #include "stack/include/trust-center.h"
00030
00067 EmberStatus emberSetInitialSecurityState
      (EmberInitialSecurityState* state);
00068
00077 EmberStatus emberSetExtendedSecurityBitmask
      (EmberExtendedSecurityBitmask mask);
00078
00087 EmberStatus emberGetExtendedSecurityBitmask
      (EmberExtendedSecurityBitmask* mask);
00088
00089
00096 #define EMBER_JOIN_NO_PRECONFIG_KEY_BITMASK \
00097   ( EMBER_STANDARD_SECURITY_MODE           \
00098     | EMBER_GET_LINK_KEY_WHEN_JOINING )
00099
00106 #define EMBER_JOIN_PRECONFIG_KEY_BITMASK \
00107   ( EMBER_STANDARD_SECURITY_MODE         \
00108     | EMBER_HAVE_PRECONFIGURED_KEY       \
00109     | EMBER_REQUIRE_ENCRYPTED_KEY )
00110
00111
00121 EmberStatus emberGetCurrentSecurityState
      (EmberCurrentSecurityState* state);
00122
00141 EmberStatus emberGetKey(EmberKeyType type,
00142                         EmberKeyStruct* keyStruct);
00143
00144
00151 boolean emberHaveLinkKey(EmberEUI64 remoteDevice);
00152
00153 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00154 extern int8u emberKeyTableSize;
00155 extern int32u* emIncomingApsFrameCounters;
00156 #endif
00157
00171 EmberStatus emberGenerateRandomKey(
      EmberKeyData* keyAddress);
00172
00180 void emberSwitchNetworkKeyHandler(int8u
      sequenceNumber);
00181
00211 EmberStatus emberRequestLinkKey(EmberEUI64
       partner);
00212
00222 void emberZigbeeKeyEstablishmentHandler(
      EmberEUI64 partner, EmberKeyStatus status);
00223
00224
00241 EmberStatus emberGetKeyTableEntry(int8u
      index,
00242                                   EmberKeyStruct* result);
00243
00255 #if defined DOXYGEN_SHOULD_SKIP_THIS
```

```
00256 EmberStatus emberSetKeyTableEntry(int8u
      index,
00257                                     EmberEUI64 address,
00258                                     boolean linkKey,
00259                                     EmberKeyData* keyData);
00260 #else
00261 EmberStatus emSetKeyTableEntry(boolean erase,
00262                               int8u index,
00263                               EmberEUI64 address,
00264                               boolean linkKey,
00265                               EmberKeyData* keyData);
00266 #define emberSetKeyTableEntry(index, address, linkKey, keyData) \
00267   emSetKeyTableEntry(FALSE, index, address, linkKey, keyData)
00268 #endif
00269
00292 EmberStatus emberAddOrUpdateKeyTableEntry
      (EmberEUI64 address,
00293                                     boolean linkKey,
00294                                     EmberKeyData* keyData);
00295
00307 int8u emberFindKeyTableEntry(EmberEUI64
      address,
00308                             boolean linkKey);
00309
00318 #if defined DOXYGEN_SHOULD_SKIP_THIS
00319 EmberStatus emberEraseKeyTableEntry(int8u
       index);
00320 #else
00321 #define emberEraseKeyTableEntry(index) \
00322   emSetKeyTableEntry(TRUE, (index), NULL, FALSE, NULL)
00323 #endif
00324
00325
00332 EmberStatus emberClearKeyTable(void);
00333
00348 EmberStatus emberStopWritingStackTokens(
      void);
00349
00359 EmberStatus emberStartWritingStackTokens
      (void);
00360
00369 boolean emberWritingStackTokensEnabled(void);
00370
00398 EmberStatus emberApsCryptMessage(boolean encrypt
      ,
00399                                     EmberMessageBuffer buffer,
00400                                     int8u apsHeaderEndIndex,
00401                                     EmberEUI64 remoteEui64);
00402
00403
00414 EmberStatus emberGetMfgSecurityConfig(
      EmberMfgSecurityStruct* settings);
00415
00416
00461 EmberStatus emberSetMfgSecurityConfig(
      int32u magicNumber,
00462                                     const EmberMfgSecurityStruct
      * settings);
00463
00464
00465
00466 // @} END addtogroup
00467
```

8.97 serial.h File Reference

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 Mode Definitions

These are numerical definitions for the possible serial modes so that code can test for the one being used. There may be additional modes defined in the micro-specific micro.h.

- #define EMBER_SERIAL_UNUSED
- #define EMBER_SERIAL_FIFO
- #define EMBER_SERIAL_BUFFER
- #define EMBER_SERIAL_LOWLEVEL

FIFO Utility Macros

These macros manipulate the FIFO queue data structures to add and remove data.

- #define FIFO_ENQUEUE(queue, data, size)
- #define FIFO_DEQUEUE(queue, size)

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.

- #define halInternalUartFlowControl(port)
- #define halInternalUartRxPump(port)
- EmberStatus halInternalUartInit (int8u port, SerialBaudRate rate, SerialParity parity, int8u stopBits)
- void halInternalPowerDownUart (void)
- void halInternalPowerUpUart (void)
- void halInternalStartUartTx (int8u port)
- void halInternalStopUartTx (int8u port)
- EmberStatus halInternalForceWriteUartData (int8u port, int8u *data, int8u length)
- EmberStatus halInternalForceReadUartByte (int8u port, int8u *dataByte)
- void halInternalWaitUartTxComplete (int8u port)
- void halInternalRestartUart (void)
- boolean halInternalUart1FlowControlRxIsEnabled (void)
- boolean halInternalUart1XonRefreshDone (void)
- boolean halInternalUart1TxIsIdle (void)

Buffered Serial Utility APIs

The higher-level serial code implements these APIs, which the HAL uses to deal with buffered serial output.

- void emSerialBufferNextMessageIsr (EmSerialBufferQueue *q)
- void emSerialBufferNextBlockIsr (EmSerialBufferQueue *q, int8u port)

Virtual UART API

API used by the stack in debug builds to receive data arriving over the virtual UART.

- void halStackReceiveVuartMessage (int8u ∗data, int8u length)

8.97.1   Detailed Description

Serial hardware abstraction layer interfaces. See Serial UART Communication for documentation.

Definition in file hal/micro/serial.h.

8.98   hal/micro/serial.h

```
00001
00023 #ifndef __HAL_SERIAL_H__
00024 #define __HAL_SERIAL_H__
00025
00026 // EM_NUM_SERIAL_PORTS is inherited from the micro specifc micro.h
00027 #if (EM_NUM_SERIAL_PORTS == 1)
00028   #define FOR_EACH_PORT(cast,prefix,suffix)  \
00029     cast(prefix##0##suffix)
00030 #elif (EM_NUM_SERIAL_PORTS == 2)
00031   #define FOR_EACH_PORT(cast,prefix,suffix)  \
00032     cast(prefix##0##suffix),               \
00033     cast(prefix##1##suffix)
00034 #elif (EM_NUM_SERIAL_PORTS == 3)
00035   #define FOR_EACH_PORT(cast,prefix,suffix)  \
00036     cast(prefix##0##suffix),            \
00037     cast(prefix##1##suffix),          \
00038     cast(prefix##2##suffix)
00039 #else
00040   #error unsupported number of serial ports
00041 #endif
00042
00055 #define EMBER_SERIAL_UNUSED 0
00056 #define EMBER_SERIAL_FIFO   1
00057 #define EMBER_SERIAL_BUFFER 2
00058 #define EMBER_SERIAL_LOWLEVEL 3
00059
00061 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00062
00063 // The following tests for setting of an invalid mode
00064 #ifdef EMBER_SERIAL0_MODE
00065 #if (EMBER_SERIAL0_MODE != EMBER_SERIAL_FIFO)    && \
00066     (EMBER_SERIAL0_MODE != EMBER_SERIAL_BUFFER) && \
00067     (EMBER_SERIAL0_MODE != EMBER_SERIAL_LOWLEVEL) && \
00068     (EMBER_SERIAL0_MODE != EMBER_SERIAL_UNUSED)
00069   #error Invalid Serial 0 Mode
00070 #endif
00071 #else
00072   #define EMBER_SERIAL0_MODE EMBER_SERIAL_UNUSED
00073 #endif
00074 #ifdef EMBER_SERIAL1_MODE
00075 #if (EMBER_SERIAL1_MODE != EMBER_SERIAL_FIFO)    && \
00076     (EMBER_SERIAL1_MODE != EMBER_SERIAL_BUFFER) && \
00077     (EMBER_SERIAL1_MODE != EMBER_SERIAL_LOWLEVEL) && \
00078     (EMBER_SERIAL1_MODE != EMBER_SERIAL_UNUSED)
00079   #error Invalid Serial 1 Mode
00080 #endif
00081 #else
00082   #define EMBER_SERIAL1_MODE EMBER_SERIAL_UNUSED
00083 #endif
00084 #ifdef EMBER_SERIAL2_MODE
00085 #if (EMBER_SERIAL2_MODE != EMBER_SERIAL_FIFO)    && \
00086     (EMBER_SERIAL2_MODE != EMBER_SERIAL_BUFFER) && \
00087     (EMBER_SERIAL2_MODE != EMBER_SERIAL_LOWLEVEL) && \
00088     (EMBER_SERIAL2_MODE != EMBER_SERIAL_UNUSED)
00089   #error Invalid Serial 2 Mode
00090 #endif
00091 #else
```

```
00092   #define EMBER_SERIAL2_MODE EMBER_SERIAL_UNUSED
00093 #endif
00094
00095 // Determine if FIFO and/or Buffer modes are being used, so those sections of
00096 //  code may be disabled if not
00097 #if (EMBER_SERIAL0_MODE == EMBER_SERIAL_FIFO) || \
00098     (EMBER_SERIAL1_MODE == EMBER_SERIAL_FIFO) || \
00099     (EMBER_SERIAL2_MODE == EMBER_SERIAL_FIFO)
00100   #define EM_ENABLE_SERIAL_FIFO
00101 #endif
00102 #if (EMBER_SERIAL0_MODE == EMBER_SERIAL_BUFFER) || \
00103     (EMBER_SERIAL1_MODE == EMBER_SERIAL_BUFFER) || \
00104     (EMBER_SERIAL2_MODE == EMBER_SERIAL_BUFFER)
00105   #define EM_ENABLE_SERIAL_BUFFER
00106 #endif
00107
00108
00116 typedef struct {
00118   int16u head;
00120   int16u tail;
00122   volatile int16u used;
00124   int8u fifo[];
00125 } EmSerialFifoQueue;
00126
00127
00131 typedef struct {
00133   int8u length;
00135   EmberMessageBuffer buffer;
00137   int8u startIndex;
00138 } EmSerialBufferQueueEntry;
00139
00143 typedef struct {
00145   int8u head;
00147   int8u tail;
00149   volatile int8u used;
00151   volatile int8u dead;
00154   EmberMessageBuffer currentBuffer;
00156   int8u *nextByte;
00158   int8u *lastByte;
00160   EmSerialBufferQueueEntry fifo[];
00161 } EmSerialBufferQueue;
00162
00172 extern void *emSerialTxQueues[EM_NUM_SERIAL_PORTS];
00174 extern EmSerialFifoQueue *emSerialRxQueues[EM_NUM_SERIAL_PORTS];
00176 extern int16u PGM emSerialTxQueueMasks[EM_NUM_SERIAL_PORTS];
00178 extern int16u PGM emSerialTxQueueSizes[EM_NUM_SERIAL_PORTS];
00180 extern int16u PGM emSerialRxQueueSizes[EM_NUM_SERIAL_PORTS];
00182 extern int8u emSerialRxError[EM_NUM_SERIAL_PORTS];
00184 extern int16u emSerialRxErrorIndex[EM_NUM_SERIAL_PORTS];
00186 extern int8u PGM emSerialPortModes[EM_NUM_SERIAL_PORTS];
00187
00188 //Compatibility code for the AVR Atmega
00189 #ifdef AVR_ATMEGA
00190 extern int8u PGM emSerialTxQueueWraps[EM_NUM_SERIAL_PORTS];
00191 extern int8u PGM emSerialRxQueueWraps[EM_NUM_SERIAL_PORTS];
00192 #else
00193 extern int16u PGM emSerialTxQueueWraps[EM_NUM_SERIAL_PORTS];
00194 extern int16u PGM emSerialRxQueueWraps[EM_NUM_SERIAL_PORTS];
00195 #endif
00196
00199 #ifdef EZSP_UART
00200   #define HANDLE_ASH_ERROR(type) emberCounterHandler(type, 0)
00201 #else
00202   #define HANDLE_ASH_ERROR(type)
00203 #endif
00204
00205 #endif //DOXYGEN_SHOULD_SKIP_THIS
00206
00207
00224 //Compatibility code for the AVR Atmega
00225 #ifdef AVR_ATMEGA
00226 #define FIFO_ENQUEUE(queue,data,size) \
00227   do {                                \
00228     queue->fifo[queue->head] = data;      \
00229     queue->head = ((queue->head + 1) & size);  \
00230     queue->used++;                          \
00231   } while(0)
00232 #else
00233 #define FIFO_ENQUEUE(queue,data,size) \
00234   do {                                \
```

```
00235     queue->fifo[queue->head] = data;         \
00236     queue->head = ((queue->head + 1) % size);  \
00237     queue->used++;                             \
00238   } while(0)
00239 #endif
00240
00248 //Compatibility code for the AVR Atmega
00249 #ifdef AVR_ATMEGA
00250 #define FIFO_DEQUEUE(queue,size) \
00251   queue->fifo[queue->tail];           \
00252   queue->tail = ((queue->tail + 1) & size);  \
00253   queue->used--
00254 #else
00255 #define FIFO_DEQUEUE(queue,size) \
00256   queue->fifo[queue->tail];           \
00257   queue->tail = ((queue->tail + 1) % size);  \
00258   queue->used--
00259 #endif
00260
00264 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00265
00269 enum SerialBaudRate
00270 #else
00271 #ifndef DEFINE_BAUD
00272 #define DEFINE_BAUD(num) BAUD_##num
00273 #endif
00274 typedef int8u SerialBaudRate;
00275 enum
00276 #endif //DOXYGEN_SHOULD_SKIP_THIS
00277 {
00278   DEFINE_BAUD(300) = 0,   // BAUD_300
00279   DEFINE_BAUD(600) = 1,   // BAUD_600
00280   DEFINE_BAUD(900) = 2,   // etc...
00281   DEFINE_BAUD(1200) = 3,
00282   DEFINE_BAUD(2400) = 4,
00283   DEFINE_BAUD(4800) = 5,
00284   DEFINE_BAUD(9600) = 6,
00285   DEFINE_BAUD(14400) = 7,
00286   DEFINE_BAUD(19200) = 8,
00287   DEFINE_BAUD(28800) = 9,
00288   DEFINE_BAUD(38400) = 10,
00289   DEFINE_BAUD(50000) = 11,
00290   DEFINE_BAUD(57600) = 12,
00291   DEFINE_BAUD(76800) = 13,
00292   DEFINE_BAUD(100000) = 14,
00293   DEFINE_BAUD(115200) = 15,
00294 #ifdef AVR_ATMEGA
00295   DEFINE_BAUD(CUSTOM) = 16
00296 #else
00297   DEFINE_BAUD(230400) = 16,   /*<! define higher baud rates for the
      EM2XX and EM3XX */
00298   DEFINE_BAUD(460800) = 17,   /*<! Note: receiving data at baud
      rates > 115200 */
00299   DEFINE_BAUD(CUSTOM) = 18    /*<! may not be reliable due to
      interrupt latency */
00300 #endif
00301 };
00302
00303
00304 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00305
00309 enum NameOfType
00310 #else
00311 #ifndef DEFINE_PARITY
00312 #define DEFINE_PARITY(val) PARITY_##val
00313 #endif
00314 typedef int8u SerialParity;
00315 enum
00316 #endif //DOXYGEN_SHOULD_SKIP_THIS
00317 {
00318   DEFINE_PARITY(NONE) = 0,  // PARITY_NONE
00319   DEFINE_PARITY(ODD) = 1,   // PARITY_ODD
00320   DEFINE_PARITY(EVEN) = 2   // PARITY_EVEN
00321 };
00322
00348 EmberStatus halInternalUartInit(int8u port,
00349                                 SerialBaudRate rate,
00350                                 SerialParity parity,
00351                                 int8u stopBits);
00352
```

```
00357 void halInternalPowerDownUart(void);
00358
00363 void halInternalPowerUpUart(void);
00364
00371 void halInternalStartUartTx(int8u port);
00372
00373
00379 void halInternalStopUartTx(int8u port);
00380
00381
00393 EmberStatus halInternalForceWriteUartData
      (int8u port, int8u *data, int8u length);
00394
00403 EmberStatus halInternalForceReadUartByte
      (int8u port, int8u *dataByte);
00404
00410 void halInternalWaitUartTxComplete(int8u port
      );
00411
00419 #if (EMBER_SERIAL1_MODE == EMBER_SERIAL_FIFO) &&          \
00420     ( defined(EMBER_SERIAL1_XONXOFF) ||                  \
00421     (defined(XAP2B) && defined(EMBER_SERIAL1_RTSCTS) ) )
00422 void halInternalUartFlowControl(int8u port);
00423 #else
00424 #define halInternalUartFlowControl(port) do {} while(FALSE)
00425 #endif
00426
00434 #if (defined(XAP2B)&&(EMBER_SERIAL1_MODE == EMBER_SERIAL_BUFFER)) ||
      defined(CORTEXM3)
00435 void halInternalUartRxPump(int8u port);
00436 #else
00437 #define halInternalUartRxPump(port) do {} while(FALSE)
00438 #endif
00439
00445 void halInternalRestartUart(void);
00446
00452 boolean halInternalUart1FlowControlRxIsEnabled
      (void);
00453
00458 boolean halInternalUart1XonRefreshDone(void);
00459
00464 boolean halInternalUart1TxIsIdle(void);
00465
00481 void emSerialBufferNextMessageIsr(
      EmSerialBufferQueue *q);
00482
00483
00493 void emSerialBufferNextBlockIsr(EmSerialBufferQueue *
      q, int8u port);
00494
00511 void halStackReceiveVuartMessage(int8u* data,
      int8u length);
00512
00515 #endif //__HAL_SERIAL_H__
00516
```

8.99 serial.h File Reference

Macros

- #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 emberSerialReadData (int8u port, int8u *data, int16u length, int16u *bytesRead)

- EmberStatus emberSerialReadDataTimeout (int8u port, int8u ∗data, int16u length, int16u ∗bytes-Read, int16u firstByteTimeout, int16u subsequentByteTimeout)
- 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)
- EmberStatus emberSerialWriteBuffer (int8u port, EmberMessageBuffer buffer, int8u start, int8u length)
- XAP2B_PAGEZERO_ON EmberStatus emberSerialWaitSend (int8u port)
- XAP2B_PAGEZERO_OFF EmberStatus emberSerialGuaranteedPrintf (int8u port, PGM_P format-String,...)
- 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 flushHandler, int8u port, PGM_P string, va_list args)

8.99.1 Detailed Description

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

Definition in file app/util/serial/serial.h.

8.100 app/util/serial/serial.h

```
00001
00012 #ifndef __SERIAL_H__
00013 #define __SERIAL_H__
00014
00015 #ifndef __HAL_H__
00016    #error hal/hal.h should be included first
00017 #endif
00018
00019 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00020 #include <stdarg.h>
00021
00022 //Rx FIFO Full indicator
00023 #define RX_FIFO_FULL (0xFFFF)
00024
```

```
00025 #endif // DOXYGEN_SHOULD_SKIP_THIS
00026
00136 EmberStatus emberSerialInit(int8u port,
00137                             SerialBaudRate rate,
00138                             SerialParity parity,
00139                             int8u stopBits);
00140
00148 int16u emberSerialReadAvailable(int8u port);
00149
00167 EmberStatus emberSerialReadByte(int8u port,
00168                                 int8u *dataByte);
00193 EmberStatus emberSerialReadData(int8u port,
00194                                 int8u *data,
00195                                 int16u length,
00196                                 int16u *bytesRead);
00197
00232 EmberStatus emberSerialReadDataTimeout(
00233 int8u port,
00234                                        int8u *data,
00235                                        int16u length,
00236                                        int16u *bytesRead,
00237                                        int16u firstByteTimeout,
00238                                        int16u subsequentByteTimeout);
00238
00253 EmberStatus emberSerialReadLine(int8u port,
00254 char *data, int8u max);
00254
00278 EmberStatus emberSerialReadPartialLine(
00279 int8u port, char *data, int8u max, int8u *index);
00279
00288 int16u emberSerialWriteAvailable(int8u port
00289 );
00289
00297 #define emberSerialWriteUsed(port)  \
00298   (emSerialTxQueueSizes[port] - emberSerialWriteAvailable(port))
00299
00313 EmberStatus emberSerialWriteByte(int8u port
00314 , int8u dataByte);
00314
00329 EmberStatus emberSerialWriteHex(int8u port,
00330 int8u dataByte);
00330
00343 EmberStatus emberSerialWriteString(int8u
00344 port, PGM_P string);
00344
00369 XAP2B_PAGEZERO_ON
00370 EmberStatus emberSerialPrintf(int8u port,
00371 PGM_P formatString, ...);
00371 XAP2B_PAGEZERO_OFF
00372
00388 XAP2B_PAGEZERO_ON
00389 EmberStatus emberSerialPrintfLine(int8u
00390 port, PGM_P formatString, ...);
00390 XAP2B_PAGEZERO_OFF
00391
00402 XAP2B_PAGEZERO_ON
00403 EmberStatus emberSerialPrintCarriageReturn
00404 (int8u port);
00404 XAP2B_PAGEZERO_OFF
00405
00406
00419 EmberStatus emberSerialPrintfVarArg(int8u
00420  port, PGM_P formatString, va_list ap);
00420
00436 EmberStatus emberSerialWriteData(int8u port
00437 , int8u *data, int8u length);
00437
00438 //Host HALs do not use stack buffers.
00439 #ifndef HAL_HOST
00440
00458 EmberStatus emberSerialWriteBuffer(int8u
00459 port, EmberMessageBuffer buffer, int8u start, int8u
      length);
00459 #endif //HAL_HOST
00460
00473 XAP2B_PAGEZERO_ON
00474 EmberStatus emberSerialWaitSend(int8u port);
00475 XAP2B_PAGEZERO_OFF
00476
```

```
00497 EmberStatus emberSerialGuaranteedPrintf(
      int8u port, PGM_P formatString, ...);
00498
00504 void emberSerialBufferTick(void);
00505
00511 void emberSerialFlushRx(int8u port);
00512
00513
00514
00515
00536 typedef EmberStatus (emPrintfFlushHandler)(int8u
      flushVar,
00537                                            int8u *contents,
00538                                            int8u length);
00539
00540
00558 int8u emPrintfInternal(emPrintfFlushHandler
      flushHandler,
00559                        int8u port,
00560                        PGM_P string,
00561                        va_list args);
00562
00563
00568 #endif // __SERIAL_H__
00569
```

8.101   stack-info.h File Reference

Data Structures

- struct EmberEndpointDescription

  Endpoint information (a ZigBee Simple Descriptor).
- struct EmberEndpoint

  Gives the endpoint information for a particular endpoint.

Functions

- void emberStackStatusHandler (EmberStatus status)
- EmberNetworkStatus emberNetworkState (void)
- boolean emberStackIsUp (void)
- EmberEUI64 emberGetEui64 (void)
- boolean emberIsLocalEui64 (EmberEUI64 eui64)
- EmberNodeId emberGetNodeId (void)
- EmberNodeId emberRadioGetNodeId (void)
- void emberSetManufacturerCode (int16u code)
- void emberSetPowerDescriptor (int16u descriptor)
- void emberSetMaximumIncomingTransferSize (int16u size)
- void emberSetMaximumOutgoingTransferSize (int16u size)
- void emberSetDescriptorCapability (int8u capability)
- EmberStatus emberGetNetworkParameters (EmberNetworkParameters ∗parameters)
- EmberStatus emberGetNodeType (EmberNodeType ∗resultLocation)
- EmberStatus emberSetRadioChannel (int8u channel)
- int8u emberGetRadioChannel (void)
- EmberStatus emberSetRadioPower (int8s power)
- int8s emberGetRadioPower (void)
- EmberPanId emberGetPanId (void)
- EmberPanId emberRadioGetPanId (void)

- void emberGetExtendedPanId (int8u ∗resultLocation)
- int8u emberGetEndpoint (int8u index)
- boolean emberGetEndpointDescription (int8u endpoint, EmberEndpointDescription ∗result)
- int16u emberGetEndpointCluster (int8u endpoint, EmberClusterListId listId, int8u listIndex)
- boolean emberIsNodeIdValid (EmberNodeId nodeId)
- EmberNodeId emberLookupNodeIdByEui64 (EmberEUI64 eui64)
- EmberStatus emberLookupEui64ByNodeId (EmberNodeId nodeId, EmberEUI64 eui64Return)
- void emberCounterHandler (EmberCounterType type, int8u data)
- EmberStatus emberGetNeighbor (int8u index, EmberNeighborTableEntry ∗result)
- EmberStatus emberGetRouteTableEntry (int8u index, EmberRouteTableEntry ∗result)
- int8u emberStackProfile (void)
- int8u emberTreeDepth (void)
- int8u emberNeighborCount (void)
- int8u emberRouteTableSize (void)
- int8u emberNextZigbeeSequenceNumber (void)

Variables

- PGM int8u emberStackProfileId [ ]
- int8u emberEndpointCount
- EmberEndpoint emberEndpoints [ ]

Radio-specific Functions

- EmberStatus emberSetTxPowerMode (int16u txPowerMode)
- int16u emberGetTxPowerMode (void)
- EmberStatus emberSetNodeId (EmberNodeId nodeId)
- void emberRadioNeedsCalibratingHandler (void)
- void emberCalibrateCurrentChannel (void)

General Functions

- void emberReverseMemCopy (int8u ∗dest, const int8u ∗src, int8u length)
- XAP2B_PAGEZERO_ON int16u emberFetchLowHighInt16u (int8u ∗contents)
- XAP2B_PAGEZERO_OFF void emberStoreLowHighInt16u (int8u ∗contents, int16u value)
- int32u emberFetchLowHighInt32u (int8u ∗contents)
- int32u emberFetchHighLowInt32u (int8u ∗contents)
- void emberStoreLowHighInt32u (int8u ∗contents, int32u value)
- void emberStoreHighLowInt32u (int8u ∗contents, int32u value)

8.101.1 Detailed Description

EmberZNet API for accessing and setting stack information. See Stack Information for documentation.

Definition in file stack-info.h.

8.102   stack-info.h

```
00001
00051 void emberStackStatusHandler(EmberStatus
      status);
00052
00060 EmberNetworkStatus emberNetworkState(void);
00061
00071 boolean emberStackIsUp(void);
00072
00073 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00074
00078 EmberEUI64 emberGetEui64(void);
00079
00086 boolean emberIsLocalEui64(EmberEUI64 eui64);
00087
00093 EmberNodeId emberGetNodeId(void);
00094
00100 EmberNodeId emberRadioGetNodeId(void);
00101
00107 void emberSetManufacturerCode(int16u code);
00108
00115 void emberSetPowerDescriptor(int16u descriptor);
00116
00123 void emberSetMaximumIncomingTransferSize(
      int16u size);
00124
00131 void emberSetMaximumOutgoingTransferSize(
      int16u size);
00132
00137 void emberSetDescriptorCapability(int8u
      capability);
00138
00139
00140 #else    // Doxygen ignores the following
00141 extern EmberEUI64 emLocalEui64;
00142 #define emberGetEui64() (emLocalEui64)
00143 #define emberIsLocalEui64(eui64)                          \
00144 (MEMCOMPARE((eui64), emLocalEui64, EUI64_SIZE) == 0)
00145
00146 XAP2B_PAGEZERO_ON
00147 EmberNodeId emberGetNodeId(void);
00148 EmberNodeId emberRadioGetNodeId(void);
00149 XAP2B_PAGEZERO_OFF
00150
00151 extern int16u emManufacturerCode;
00152 extern int16u emPowerDescriptor;
00153 extern int16u emMaximumIncomingTransferSize;
00154 extern int16u emMaximumOutgoingTransferSize;
00155 extern int8u emDescriptorCapability;
00156 #define emberSetManufacturerCode(code)                    \
00157 (emManufacturerCode = (code))
00158 #define emberSetPowerDescriptor(descriptor)               \
00159 (emPowerDescriptor = (descriptor))
00160 #define emberSetMaximumIncomingTransferSize(size)         \
00161 (emMaximumIncomingTransferSize = (size))
00162 #define emberSetMaximumOutgoingTransferSize(size)         \
00163 (emMaximumOutgoingTransferSize = (size))
00164 #define emberSetDescriptorCapability(capability)          \
00165 (emDescriptorCapability = (capability))
00166 #endif
00167
00177 EmberStatus emberGetNetworkParameters(
      EmberNetworkParameters *parameters);
00178
00188 EmberStatus emberGetNodeType(EmberNodeType
       *resultLocation);
00189
00203 EmberStatus emberSetRadioChannel(int8u
      channel);
00204
00212 int8u emberGetRadioChannel(void);
00213
00231 EmberStatus emberSetRadioPower(int8s power);
00232
00240 int8s emberGetRadioPower(void);
00241
00246 EmberPanId emberGetPanId(void);
00247
```

```
00252 EmberPanId emberRadioGetPanId(void);
00253
00255 void emberGetExtendedPanId(int8u *resultLocation);
00256
00258 extern PGM int8u emberStackProfileId[];
00259
00267 typedef struct {
00269   int16u profileId;
00271   int16u deviceId;
00273   int8u deviceVersion;
00275   int8u inputClusterCount;
00277   int8u outputClusterCount;
00278 } EmberEndpointDescription;
00279
00283 typedef struct {
00285   int8u endpoint;
00287   EmberEndpointDescription PGM * description
      ;
00289   int16u PGM * inputClusterList;
00291   int16u PGM * outputClusterList;
00292 } EmberEndpoint;
00293
00295 extern int8u emberEndpointCount;
00296
00310 extern EmberEndpoint emberEndpoints[];
00311
00325 int8u emberGetEndpoint(int8u index);
00326
00342 boolean emberGetEndpointDescription(int8u
      endpoint,
00343                                     EmberEndpointDescription
       *result);
00344
00363 int16u emberGetEndpointCluster(int8u endpoint
      ,
00364                                 EmberClusterListId listId,
00365                                 int8u listIndex);
00366
00373 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00374 boolean emberIsNodeIdValid(EmberNodeId nodeId);
00375 #else
00376 #define emberIsNodeIdValid(nodeId) ((nodeId) < EMBER_DISCOVERY_ACTIVE_NODE_ID)
00377 #endif
00378
00388 EmberNodeId emberLookupNodeIdByEui64(
      EmberEUI64 eui64);
00389
00402 EmberStatus emberLookupEui64ByNodeId(
      EmberNodeId nodeId,
00403                                     EmberEUI64 eui64Return);
00404
00418 void emberCounterHandler(EmberCounterType
      type, int8u data);
00419
00435 EmberStatus emberGetNeighbor(int8u index,
      EmberNeighborTableEntry *result);
00436
00450 EmberStatus emberGetRouteTableEntry(int8u
       index, EmberRouteTableEntry *result);
00451
00452 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00453
00458 int8u emberStackProfile(void);
00459
00464 int8u emberTreeDepth(void);
00465 #endif
00466
00470 int8u emberNeighborCount(void);
00471 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00472
00476 int8u emberRouteTableSize(void);
00477
00478 #else   // Doxgyen ignores the following
00479 // The '+ 0' prevents anyone from accidentally assigning to these.
00480 #define emberStackProfile()       (emStackProfile    + 0)
00481 #define emberTreeDepth()          (emTreeDepth       + 0)
00482 #define emberMaxDepth()           (emMaxDepth        + 0)
00483 #define emberRouteTableSize()     (emRouteTableSize  + 0)
00484
00485 extern int8u emStackProfile;
```

```
00486 extern int8u emDefaultSecurityLevel;
00487 extern int8u emMaxHops;
00488 extern int8u emRouteTableSize;
00489 extern int8u emMaxDepth;                    // maximum tree depth
00490 extern int8u emTreeDepth;                   // our depth
00491
00492 // This was originally a #define declared here. I moved
00493 // emZigbeeNetworkSecurityLevel to the networkInfo struct.This function is now
00494 // defined in ember-stack-common.c
00495 int8u emberSecurityLevel(void);
00496 // New function (defined in ember-stack-common.c)
00497 void emberSetSecurityLevel(int8u securityLevel);
00498 #endif
00499
00504 int8u emberNextZigbeeSequenceNumber(void);
00505
00508
00542 EmberStatus emberSetTxPowerMode(int16u
      txPowerMode);
00543
00549 int16u emberGetTxPowerMode(void);
00550
00558 EmberStatus emberSetNodeId(EmberNodeId
      nodeId);
00559
00560
00577 void emberRadioNeedsCalibratingHandler(void);
00578
00587 void emberCalibrateCurrentChannel(void);
00589
00590
00593
00601 void emberReverseMemCopy(int8u* dest, const int8u*
       src, int8u length);
00602
00603
00607 XAP2B_PAGEZERO_ON
00608 int16u emberFetchLowHighInt16u(int8u *
      contents);
00609 XAP2B_PAGEZERO_OFF
00610
00614 void emberStoreLowHighInt16u(int8u *contents,
      int16u value);
00615
00616 #if !defined DOXYGEN_SHOULD_SKIP_THIS
00617 int32u emFetchInt32u(boolean lowHigh, int8u* contents);
00618 #endif
00619
00624 #if defined DOXYGEN_SHOULD_SKIP_THIS
00625 int32u emberFetchLowHighInt32u(int8u *
      contents);
00626 #else
00627 #define emberFetchLowHighInt32u(contents) \
00628   (emFetchInt32u(TRUE, contents))
00629 #endif
00630
00635 #if defined DOXYGEN_SHOULD_SKIP_THIS
00636 int32u emberFetchHighLowInt32u(int8u *
      contents);
00637 #else
00638 #define emberFetchHighLowInt32u(contents) \
00639   (emFetchInt32u(FALSE, contents))
00640 #endif
00641
00642
00643
00644 #if !defined DOXYGEN_SHOULD_SKIP_THIS
00645 void emStoreInt32u(boolean lowHigh, int8u* contents, int32u value);
00646 #endif
00647
00651 #if defined DOXYGEN_SHOULD_SKIP_THIS
00652 void emberStoreLowHighInt32u(int8u *contents,
      int32u value);
00653 #else
00654 #define emberStoreLowHighInt32u(contents, value) \
00655   (emStoreInt32u(TRUE, contents, value))
00656 #endif
00657
00661 #if defined DOXYGEN_SHOULD_SKIP_THIS
00662 void emberStoreHighLowInt32u(int8u *contents,
```

```
      int32u value);
00663 #else
00664 #define emberStoreHighLowInt32u(contents, value) \
00665   (emStoreInt32u(FALSE, contents, value))
00666 #endif
00667
```

8.103   standalone-bootloader.h File Reference

Required Custom Functions

- void bootloaderMenu (void)

Available Bootloader Library Functions

Functions implemented by the bootloader library that may be used by custom functions.

- BL_Status receiveImage (int8u commState)
- boolean checkDebugMenuOption (int8u ch)
- BL_Status initOtaState (void)
- BL_Status checkOtaStart (void)
- BL_Status receiveOtaImage (void)
- boolean paIsPresent (void)

8.103.1   Detailed Description

See Standalone for detailed documentation.

Definition in file standalone-bootloader.h.

8.104   standalone-bootloader.h

```
00001
00007 //[[ Author(s): David Iacobone, diacobone@ember.com
00008 //             Lee Taylor, lee@ember.com
00009 //]]
00010
00018 #ifndef __STANDALONE_BOOTLOADER_H__
00019 #define __STANDALONE_BOOTLOADER_H__
00020
00021
00023
00028 void bootloaderMenu(void);
00032
00033
00043 BL_Status receiveImage(int8u commState);
00044
00050 boolean checkDebugMenuOption(int8u ch);
00051
00052
00058 BL_Status initOtaState(void);
00059
00065 BL_Status checkOtaStart(void);
00066
00075 BL_Status receiveOtaImage(void);
00076
00084 boolean paIsPresent(void);
00085
00089 #endif //__STANDALONE_BOOTLOADER_H__
00090
```

8.105 system-timer.h File Reference

Macros

- #define halIdleForMilliseconds(duration)

Functions

- int16u halInternalStartSystemTimer (void)
- int16u halCommonGetInt16uMillisecondTick (void)
- int32u halCommonGetInt32uMillisecondTick (void)
- int16u halCommonGetInt16uQuarterSecondTick (void)
- EmberStatus halSleepForQuarterSeconds (int32u ∗duration)
- EmberStatus halSleepForMilliseconds (int32u ∗duration)
- EmberStatus halCommonIdleForMilliseconds (int32u ∗duration)

8.105.1 Detailed Description

See System Timer Control for documentation.

Definition in file system-timer.h.

8.106 system-timer.h

```
00001
00031 #ifndef __SYSTEM_TIMER_H__
00032 #define __SYSTEM_TIMER_H__
00033
00034 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00035
00036 #if defined( EMBER_TEST )
00037    #include "unix/simulation/system-timer-sim.h"
00038 #elif defined(AVR_ATMEGA_32)
00039    #include "avr-atmega/32/system-timer.h"
00040 #elif defined(AVR_ATMEGA_128)
00041    #include "avr-atmega/128/system-timer.h"
00042 #endif
00043
00044 #endif // DOXYGEN_SHOULD_SKIP_THIS
00045
00046
00053 int16u halInternalStartSystemTimer(void);
00054
00055
00056 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00057 XAP2B_PAGEZERO_ON
00058 #endif
00059
00066 int16u halCommonGetInt16uMillisecondTick
      (void);
00067 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00068 XAP2B_PAGEZERO_OFF
00069 #endif
00070
00080 int32u halCommonGetInt32uMillisecondTick
      (void);
00081
00091 int16u halCommonGetInt16uQuarterSecondTick
      (void);
00092
00134 EmberStatus halSleepForQuarterSeconds(
      int32u *duration);
00135
```

```
00177 EmberStatus halSleepForMilliseconds(int32u
        *duration);
00178
00201 EmberStatus halCommonIdleForMilliseconds
        (int32u *duration);
00202 // Maintain the previous API for backwards compatibility
00203 #define halIdleForMilliseconds(duration)
        halCommonIdleForMilliseconds((duration))
00204
00205
00206 #endif //__SYSTEM_TIMER_H__
00207
```

8.107   token-manufacturing.h File Reference

Macros

- #define TOKEN_NEXT_ADDRESS(region, address)
- #define CREATOR_MFG_CHIP_DATA
- #define CREATOR_MFG_PART_DATA
- #define CREATOR_MFG_TESTER_DATA
- #define CREATOR_MFG_EMBER_EUI_64
- #define CREATOR_MFG_ANALOG_TRIM_NORMAL
- #define CREATOR_MFG_ANALOG_TRIM_BOOST
- #define CREATOR_MFG_ANALOG_TRIM_BOTH
- #define CREATOR_MFG_REG_TRIM
- #define CREATOR_MFG_1V8_REG_VOLTAGE
- #define CREATOR_MFG_VREF_VOLTAGE
- #define CREATOR_MFG_TEMP_CAL
- #define CREATOR_MFG_FIB_VERSION
- #define CREATOR_MFG_FIB_CHECKSUM
- #define CREATOR_MFG_FIB_OBS
- #define CREATOR_MFG_CIB_OBS
- #define CREATOR_MFG_CUSTOM_VERSION
- #define CREATOR_MFG_CUSTOM_EUI_64
- #define CREATOR_MFG_STRING
- #define CREATOR_MFG_BOARD_NAME
- #define CREATOR_MFG_EUI_64
- #define CREATOR_MFG_MANUF_ID
- #define CREATOR_MFG_PHY_CONFIG
- #define CREATOR_MFG_BOOTLOAD_AES_KEY
- #define CREATOR_MFG_EZSP_STORAGE
- #define CREATOR_MFG_ASH_CONFIG
- #define CREATOR_MFG_CBKE_DATA
- #define CREATOR_MFG_INSTALLATION_CODE
- #define CREATOR_MFG_OSC24M_BIAS_TRIM
- #define CREATOR_MFG_SYNTH_FREQ_OFFSET
- #define CREATOR_MFG_OSC24M_SETTLE_DELAY
- #define CREATOR_MFG_SECURITY_CONFIG
- #define CREATOR_MFG_CCA_THRESHOLD
- #define CURRENT_MFG_TOKEN_VERSION
- #define CURRENT_MFG_CUSTOM_VERSION

Convenience Macros

The following convenience macros are used to simplify the definition process for commonly specified parameters to the basic TOKEN_DEF macro. Please see hal/micro/token.h for a more complete explanation.

- #define DEFINE_MFG_TOKEN(name, type, address,...)

8.107.1 Detailed Description

Definitions for manufacturing tokens. This file should not be included directly. It is accessed by the other token files.

Please see stack/config/token-stack.h and hal/micro/token.h for a full explanation of the tokens.

The tokens listed below are the manufacturing tokens. This token definitions file is included from the master definitions file: stack/config/token-stack.h Please see that file for more details.

The user application can include its own manufacturing tokens in a header file similar to this one. The macro ::APPLICATION_MFG_TOKEN_HEADER should be defined to equal the name of the header file in which application manufacturing tokens are defined.

The macro DEFINE_MFG_TOKEN() should be used when instantiating a manufacturing token. Refer to the list of *_LOCATION defines to see what memory is allocated and what memory is unused/available.

REMEMBER: By definition, manufacturing tokens exist at fixed addresses. Tokens should not overlap.

Here is a basic example of a manufacturing token header file:

```
#define CREATOR_MFG_EXAMPLE 0x4242
#ifdef DEFINETYPES
typedef int8u tokTypeMfgExample[8];
#endif //DEFINETYPES
#ifdef DEFINETOKENS
#define MFG_EXAMPLE_LOCATION 0x0980
DEFINE_MFG_TOKEN(MFG_EXAMPLE,
                 tokTypeMfgExample,
                 MFG_EXAMPLE_LOCATION,
                 {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF})
#endif //DEFINETOKENS
```

Since this file contains both the typedefs and the token defs, there are two #defines used to select which one is needed when this file is included. #define DEFINETYPES is used to select the type definitions and #define DEFINETOKENS is used to select the token definitions. Refer to token.h and token.c to see how these are used.

Definition in file token-manufacturing.h.

8.107.2 Macro Definition Documentation

8.107.2.1 #define DEFINE_MFG_TOKEN(*name, type, address, ...*)

Definition at line 58 of file token-manufacturing.h.

8.107.2.2 #define TOKEN_NEXT_ADDRESS(*region, address*)

Definition at line 64 of file token-manufacturing.h.

8.107.2.3 #define CREATOR_MFG_CHIP_DATA

Definition at line 83 of file token-manufacturing.h.

8.107.2.4 #define CREATOR_MFG_PART_DATA

Definition at line 84 of file token-manufacturing.h.

8.107.2.5 #define CREATOR_MFG_TESTER_DATA

Definition at line 85 of file token-manufacturing.h.

8.107.2.6 #define CREATOR_MFG_EMBER_EUI_64

Definition at line 86 of file token-manufacturing.h.

8.107.2.7 #define CREATOR_MFG_ANALOG_TRIM_NORMAL

Definition at line 87 of file token-manufacturing.h.

8.107.2.8 #define CREATOR_MFG_ANALOG_TRIM_BOOST

Definition at line 88 of file token-manufacturing.h.

8.107.2.9 #define CREATOR_MFG_ANALOG_TRIM_BOTH

Definition at line 89 of file token-manufacturing.h.

8.107.2.10 #define CREATOR_MFG_REG_TRIM

Definition at line 90 of file token-manufacturing.h.

8.107.2.11 #define CREATOR_MFG_1V8_REG_VOLTAGE

Definition at line 91 of file token-manufacturing.h.

8.107.2.12 #define CREATOR_MFG_VREF_VOLTAGE

Definition at line 92 of file token-manufacturing.h.

8.107.2.13 #define CREATOR_MFG_TEMP_CAL

Definition at line 93 of file token-manufacturing.h.

8.107.2.14 #define CREATOR_MFG_FIB_VERSION

Definition at line 94 of file token-manufacturing.h.

8.107.2.15 #define CREATOR_MFG_FIB_CHECKSUM

Definition at line 95 of file token-manufacturing.h.

8.107.2.16 #define CREATOR_MFG_FIB_OBS

Definition at line 96 of file token-manufacturing.h.

8.107.2.17 #define CREATOR_MFG_CIB_OBS

Definition at line 98 of file token-manufacturing.h.

8.107.2.18 #define CREATOR_MFG_CUSTOM_VERSION

Definition at line 99 of file token-manufacturing.h.

8.107.2.19 #define CREATOR_MFG_CUSTOM_EUI_64

Definition at line 100 of file token-manufacturing.h.

8.107.2.20 #define CREATOR_MFG_STRING

Definition at line 101 of file token-manufacturing.h.

8.107.2.21 #define CREATOR_MFG_BOARD_NAME

Definition at line 102 of file token-manufacturing.h.

8.107.2.22 #define CREATOR_MFG_EUI_64

Definition at line 103 of file token-manufacturing.h.

8.107.2.23 #define CREATOR_MFG_MANUF_ID

Definition at line 104 of file token-manufacturing.h.

8.107.2.24 #define CREATOR_MFG_PHY_CONFIG

Definition at line 105 of file token-manufacturing.h.

8.107.2.25 #define CREATOR␣MFG␣BOOTLOAD␣AES␣KEY

Definition at line 106 of file token-manufacturing.h.

8.107.2.26 #define CREATOR␣MFG␣EZSP␣STORAGE

Definition at line 107 of file token-manufacturing.h.

8.107.2.27 #define CREATOR␣MFG␣ASH␣CONFIG

Definition at line 108 of file token-manufacturing.h.

8.107.2.28 #define CREATOR␣MFG␣CBKE␣DATA

Definition at line 109 of file token-manufacturing.h.

8.107.2.29 #define CREATOR␣MFG␣INSTALLATION␣CODE

Definition at line 110 of file token-manufacturing.h.

8.107.2.30 #define CREATOR␣MFG␣OSC24M␣BIAS␣TRIM

Definition at line 111 of file token-manufacturing.h.

8.107.2.31 #define CREATOR␣MFG␣SYNTH␣FREQ␣OFFSET

Definition at line 112 of file token-manufacturing.h.

8.107.2.32 #define CREATOR␣MFG␣OSC24M␣SETTLE␣DELAY

Definition at line 113 of file token-manufacturing.h.

8.107.2.33 #define CREATOR␣MFG␣SECURITY␣CONFIG

Definition at line 114 of file token-manufacturing.h.

8.107.2.34 #define CREATOR␣MFG␣CCA␣THRESHOLD

Definition at line 115 of file token-manufacturing.h.

8.107.2.35 #define CURRENT␣MFG␣TOKEN␣VERSION

Definition at line 118 of file token-manufacturing.h.

8.107.2.36 #define CURRENT_MFG_CUSTOM_VERSION

Definition at line 119 of file token-manufacturing.h.

8.108 token-manufacturing.h

```
00001
00058 #define DEFINE_MFG_TOKEN(name, type, address, ...)  \
00059   TOKEN_NEXT_ADDRESS(name,(address))                  \
00060   TOKEN_MFG(name, CREATOR_##name, 0, 0, type, 1,  __VA_ARGS__)
00061
00063 #ifndef TOKEN_NEXT_ADDRESS
00064   #define TOKEN_NEXT_ADDRESS(region, address)
00065 #endif
00066
00068 // MANUFACTURING DATA
00069 // *the addresses of these tokens must not change*
00070
00071
00072 // MANUFACTURING CREATORS
00073 // The creator codes are here in one list instead of next to their token
00074 // definitions so comparision of the codes is easier.  The only requirement
00075 // on these creator definitions is that they all must be unique.  A favorite
00076 // method for picking creator codes is to use two ASCII characters in order
00077 // to make the codes more memorable.  Also, the msb of Stack and Manufacturing
00078 // token creator codes is always 1, while the msb of Application token creator
00079 // codes is always 0.  This distinction allows Application tokens
00080 // to freely use the lower 15 bits for creator codes without the risk of
00081 // duplicating a Stack or Manufacturing token creator code.
00082 //--- Fixed Information Block ---
00083 #define CREATOR_MFG_CHIP_DATA          0xC344 // msb+'C'+'D'
00084 #define CREATOR_MFG_PART_DATA          0xF064 // msb+'p'+'d'
00085 #define CREATOR_MFG_TESTER_DATA        0xF464 // msb+'t'+'d'
00086 #define CREATOR_MFG_EMBER_EUI_64       0xE545 // msb+'e'+'E'
00087 #define CREATOR_MFG_ANALOG_TRIM_NORMAL 0xF46E // msb+'t'+'n'
00088 #define CREATOR_MFG_ANALOG_TRIM_BOOST  0xF442 // msb+'t'+'B'
00089 #define CREATOR_MFG_ANALOG_TRIM_BOTH   0xF462 // msb+'t'+'b'
00090 #define CREATOR_MFG_REG_TRIM           0xF274 // msb+'r'+'t'
00091 #define CREATOR_MFG_1V8_REG_VOLTAGE    0xF276 // msb+'r'+'v'
00092 #define CREATOR_MFG_VREF_VOLTAGE       0xF676 // msb+'v'+'v'
00093 #define CREATOR_MFG_TEMP_CAL           0xF463 // msb+'t'+'c'
00094 #define CREATOR_MFG_FIB_VERSION        0xFF09
00095 #define CREATOR_MFG_FIB_CHECKSUM       0xE663 // msb+'f'+'c'
00096 #define CREATOR_MFG_FIB_OBS            0xE66F // msb+'f'+'o'
00097 //--- Customer Information Block ---
00098 #define CREATOR_MFG_CIB_OBS            0xE36F // msb+'c'+'o'
00099 #define CREATOR_MFG_CUSTOM_VERSION     0xC356
00100 #define CREATOR_MFG_CUSTOM_EUI_64      0xE345
00101 #define CREATOR_MFG_STRING             0xED73
00102 #define CREATOR_MFG_BOARD_NAME         0xC24E // msb+'B'+'N' (Board Name)
00103 #define CREATOR_MFG_EUI_64             0xB634
00104 #define CREATOR_MFG_MANUF_ID           0xC944 // msb+'I'+'D' (Id)
00105 #define CREATOR_MFG_PHY_CONFIG         0xD043 // msb+'P'+'C' (Phy Config)
00106 #define CREATOR_MFG_BOOTLOAD_AES_KEY   0xC24B // msb+'B'+'K' (Bootloader Key)
00107 #define CREATOR_MFG_EZSP_STORAGE       0xCD53
00108 #define CREATOR_MFG_ASH_CONFIG         0xC143 // msb+'A'+'C' (ASH Config)
00109 #define CREATOR_MFG_CBKE_DATA          0xC342 // msb+'C'+'B' (CBke)
00110 #define CREATOR_MFG_INSTALLATION_CODE  0xC943 // msb+'I'+'C' (Installation
       Code)
00111 #define CREATOR_MFG_OSC24M_BIAS_TRIM   0xB254 // msb+'2'+'T' (2[4mHz] Trim)
00112 #define CREATOR_MFG_SYNTH_FREQ_OFFSET  0xD346 // msb+'S'+'F' (Synth Freq)
00113 #define CREATOR_MFG_OSC24M_SETTLE_DELAY 0xB253 // msb+'2'+'S' (2[4mHz] Settle)
00114 #define CREATOR_MFG_SECURITY_CONFIG    0xD343 // msb+'S'+'C' (Security Config)
00115 #define CREATOR_MFG_CCA_THRESHOLD      0xC343 // msb+'C'+'C' (Clear Channel)
00116
00117 // The master defines indicating the verions number these definitions work
       with.
00118 #define CURRENT_MFG_TOKEN_VERSION 0x01FE //MSB is version, LSB is complement
00119 #define CURRENT_MFG_CUSTOM_VERSION 0x01FE //MSB is version, LSB is complement
00120
00121
00122 #ifdef DEFINETYPES
00123 //--- Fixed Information Block ---
00124 typedef int8u tokTypeMfgChipData[24];
00125 typedef int8u tokTypeMfgPartData[6];
```

```
00126 typedef int8u tokTypeMfgTesterData[6];
00127 typedef int8u tokTypeMfgEmberEui64[8];
00128 typedef struct {
00129   int16u iffilterL;
00130   int16u lna;
00131   int16u ifamp;
00132   int16u rxadcH;
00133   int16u prescalar;
00134   int16u phdet;
00135   int16u vco;
00136   int16u loopfilter;
00137   int16u pa;
00138   int16u iqmixer;
00139 } tokTypeMfgAnalogueTrim;
00140 typedef struct {
00141   int16u iffilterH;
00142   int16u biasmaster;
00143   int16u moddac;
00144   int16u auxadc;
00145   int16u caladc;
00146 } tokTypeMfgAnalogueTrimBoth;
00147 typedef struct {
00148   int8u regTrim1V2;
00149   int8u regTrim1V8;
00150 } tokTypeMfgRegTrim;
00151 typedef int16u tokTypeMfgRegVoltage1V8;
00152 typedef int16u tokTypeMfgAdcVrefVoltage;
00153 typedef int16u tokTypeMfgTempCal;
00154 typedef int16u tokTypeMfgFibVersion;
00155 typedef int16u tokTypeMfgFibChecksum;
00156 typedef struct {
00157   int16u ob2;
00158   int16u ob3;
00159   int16u ob0;
00160   int16u ob1;
00161 } tokTypeMfgFibObs;
00162 //--- Customer Information Block ---
00163 typedef struct {
00164   int16u ob0;
00165   int16u ob1;
00166   int16u ob2;
00167   int16u ob3;
00168   int16u ob4;
00169   int16u ob5;
00170   int16u ob6;
00171   int16u ob7;
00172 } tokTypeMfgCibObs;
00173 typedef int16u tokTypeMfgCustomVersion;
00174 typedef int8u tokTypeMfgCustomEui64[8];
00175 typedef int8u tokTypeMfgString[16];
00176 typedef int8u tokTypeMfgBoardName[16];
00177 typedef int16u tokTypeMfgManufId;
00178 typedef int16u tokTypeMfgPhyConfig;
00179 typedef int8u tokTypeMfgBootloadAesKey[16];
00180 typedef int8u tokTypeMfgEui64[8];
00181 typedef int8u tokTypeMfgEzspStorage[8];
00182 typedef int16u tokTypeMfgAshConfig;
00183 typedef struct {
00184   int8u certificate[48];
00185   int8u caPublicKey[22];
00186   int8u privateKey[21];
00187   // The bottom flag bit is 1 for uninitialized, 0 for initialized.
00188   // The other flag bits should be set to 0 at initialization.
00189   int8u flags;
00190 } tokTypeMfgCbkeData;
00191 typedef struct {
00192   // The bottom flag bit is 1 for uninitialized, 0 for initialized.
00193   // Bits 1 and 2 give the size of the value string:
00194   // 0 = 6 bytes, 1 = 8 bytes, 2 = 12 bytes, 3 = 16 bytes.
00195   // The other flag bits should be set to 0 at initialization.
00196   // Special flags support.  Due to a bug in the way some customers
00197   // had programmed the flags field, we will also examine the upper
00198   // bits 9 and 10 for the size field.  Those bits are also reserved.
00199   int16u flags;
00200   int8u value[16];
00201   int16u crc;
00202 } tokTypeMfgInstallationCode;
00203 typedef int16u tokTypeMfgOsc24mBiasTrim;
00204 typedef int16u tokTypeMfgSynthFreqOffset;
00205 typedef int16u tokTypeMfgOsc24mSettleDelay;
```

```
00206 typedef int16u tokTypeMfgSecurityConfig;
00207 typedef int16u tokTypeMfgCcaThreshold;
00208 #endif //DEFINETYPES
00209
00210
00211 #ifdef DEFINETOKENS
00212 //The Manufacturing tokens need to be stored at well-defined locations.
00213 //None of these addresses should ever change without extremely great care.
00214 //All locations are OR'ed with DATA_BIG_INFO_BASE to make a full 32bit address.
00215 //--- Fixed Information Block ---
00216 // FIB Bootloader                       0x0000  //1918 bytes
00217 #define MFG_CHIP_DATA_LOCATION          0x077E  //  24 bytes
00218 #define MFG_PART_DATA_LOCATION          0x0796  //   6 bytes
00219 #define MFG_TESTER_DATA_LOCATION        0x079C  //   6 bytes
00220 #define MFG_EMBER_EUI_64_LOCATION       0x07A2  //   8 bytes
00221 #define MFG_ANALOG_TRIM_NORMAL_LOCATION 0x07AA  //  20 bytes
00222 #define MFG_ANALOG_TRIM_BOOST_LOCATION  0x07BE  //  20 bytes
00223 #define MFG_ANALOG_TRIM_BOTH_LOCATION   0x07D2  //  10 bytes
00224 #define MFG_REG_TRIM_LOCATION           0x07DC  //   2 bytes
00225 #define MFG_1V8_REG_VOLTAGE_LOCATION    0x07DE  //   2 bytes
00226 #define MFG_VREF_VOLTAGE_LOCATION       0x07E0  //   2 bytes
00227 #define MFG_TEMP_CAL_LOCATION           0x07E2  //   2 bytes
00228 //reserved                             0x07E4  //  16 bytes
00229 #define MFG_FIB_VERSION_LOCATION        0x07F4  //   2 bytes
00230 #define MFG_FIB_CHECKSUM_LOCATION        0x07F6  //   2 bytes
00231 #define MFG_FIB_OBS_LOCATION            0x07F8  //   8 bytes
00232 //--- Customer Information Block ---
00233 // The CIB is a 2KB block starting at 0x08040800.
00234 #define MFG_CIB_OBS_LOCATION            0x0800  //  16 bytes (option bytes)
00235 #define MFG_CUSTOM_VERSION_LOCATION     0x0810  //   2 bytes
00236 #define MFG_CUSTOM_EUI_64_LOCATION      0x0812  //   8 bytes
00237 #define MFG_STRING_LOCATION             0x081A  //  16 bytes
00238 #define MFG_BOARD_NAME_LOCATION         0x082A  //  16 bytes
00239 #define MFG_MANUF_ID_LOCATION           0x083A  //   2 bytes
00240 #define MFG_PHY_CONFIG_LOCATION         0x083C  //   2 bytes
00241 #define MFG_BOOTLOAD_AES_KEY_LOCATION   0x083E  //  16 bytes
00242 #define MFG_EZSP_STORAGE_LOCATION       0x084E  //   8 bytes
00243 #define MFG_ASH_CONFIG_LOCATION         0x0856  //  40 bytes
00244 #define MFG_CBKE_DATA_LOCATION          0x087E  //  92 bytes
00245 #define MFG_INSTALLATION_CODE_LOCATION  0x08DA  //  20 bytes
00246 #define MFG_OSC24M_BIAS_TRIM_LOCATION   0x08EE  //   2 bytes
00247 #define MFG_SYNTH_FREQ_OFFSET_LOCATION  0x08F0  //   2 bytes
00248 #define MFG_OSC24M_SETTLE_DELAY_LOCATION 0x08F2  //   2 bytes
00249 #define MFG_SECURITY_CONFIG_LOCATION    0x08F4  //   2 bytes
00250 #define MFG_CCA_THRESHOLD_LOCATION      0x08F6  //   2 bytes
00251 // reserved for future stack use        0x08F8 - 0x0FFF // 1800 bytes free
00252 //--- Virtual MFG Tokens ---
00253 #define MFG_EUI_64_LOCATION             0x8000  // Special Trigger - see
        token.c
00254
00255 // Define the size of indexed token array
00256 #define MFG_ASH_CONFIG_ARRAY_SIZE       20
00257
00258
00259 //--- Fixed Information Block ---
00260 TOKEN_NEXT_ADDRESS(MFG_CHIP_DATA_ADDR,MFG_CHIP_DATA_LOCATION)
00261 TOKEN_MFG(MFG_CHIP_DATA, CREATOR_MFG_CHIP_DATA,
00262           0, 0, tokTypeMfgChipData, 1,
00263           {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
00264            0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
00265            0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF})
00266
00267 TOKEN_NEXT_ADDRESS(MFG_PART_DATA_ADDR,MFG_PART_DATA_LOCATION)
00268 TOKEN_MFG(MFG_PART_DATA, CREATOR_MFG_PART_DATA,
00269           0, 0, tokTypeMfgPartData, 1,
00270           {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF})
00271
00272 TOKEN_NEXT_ADDRESS(MFG_TESTER_DATA_ADDR,
      MFG_TESTER_DATA_LOCATION)
00273 TOKEN_MFG(MFG_TESTER_DATA, CREATOR_MFG_TESTER_DATA,
00274           0, 0, tokTypeMfgTesterData, 1,
00275           {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF})
00276
00277 TOKEN_NEXT_ADDRESS(MFG_EMBER_EUI_64_ADDR,
      MFG_EMBER_EUI_64_LOCATION)
00278 TOKEN_MFG(MFG_EMBER_EUI_64, CREATOR_MFG_EMBER_EUI_64,
00279           0, 0, tokTypeMfgEmberEui64, 1,
00280           {3,0,0,0,0,0,0,3})
00281
00282 TOKEN_NEXT_ADDRESS(MFG_ANALOG_TRIM_NORMAL_ADDR,
```

```
      MFG_ANALOG_TRIM_NORMAL_LOCATION)
00283 TOKEN_MFG(MFG_ANALOG_TRIM_NORMAL, CREATOR_MFG_ANALOG_TRIM_NORMAL
      ,
00284            0, 0, tokTypeMfgAnalogueTrim, 1,
00285           {0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,
00286            0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF})
00287
00288 TOKEN_NEXT_ADDRESS(MFG_ANALOG_TRIM_BOOST_ADDR,
      MFG_ANALOG_TRIM_BOOST_LOCATION)
00289 TOKEN_MFG(MFG_ANALOG_TRIM_BOOST, CREATOR_MFG_ANALOG_TRIM_BOOST
      ,
00290            0, 0, tokTypeMfgAnalogueTrim, 1,
00291           {0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,
00292            0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF})
00293
00294 TOKEN_NEXT_ADDRESS(MFG_ANALOG_TRIM_BOTH_ADDR,
      MFG_ANALOG_TRIM_BOTH_LOCATION)
00295 TOKEN_MFG(MFG_ANALOG_TRIM_BOTH, CREATOR_MFG_ANALOG_TRIM_BOTH
      ,
00296            0, 0, tokTypeMfgAnalogueTrimBoth, 1,
00297           {0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF})
00298
00299 TOKEN_NEXT_ADDRESS(MFG_REG_TRIM_ADDR,MFG_REG_TRIM_LOCATION)
00300 TOKEN_MFG(MFG_REG_TRIM, CREATOR_MFG_REG_TRIM,
00301            0, 0, tokTypeMfgRegTrim, 1,
00302           {0xFF, 0xFF})
00303
00304 TOKEN_NEXT_ADDRESS(MFG_1V8_REG_VOLTAGE_ADDR,
      MFG_1V8_REG_VOLTAGE_LOCATION)
00305 TOKEN_MFG(MFG_1V8_REG_VOLTAGE, CREATOR_MFG_1V8_REG_VOLTAGE
      ,
00306            0, 0, tokTypeMfgRegVoltage1V8, 1,
00307           0xFFFF)
00308
00309 TOKEN_NEXT_ADDRESS(MFG_VREF_VOLTAGE_ADDR,
      MFG_VREF_VOLTAGE_LOCATION)
00310 TOKEN_MFG(MFG_VREF_VOLTAGE, CREATOR_MFG_VREF_VOLTAGE,
00311            0, 0, tokTypeMfgAdcVrefVoltage, 1,
00312           0xFFFF)
00313
00314 TOKEN_NEXT_ADDRESS(MFG_TEMP_CAL_ADDR,MFG_TEMP_CAL_LOCATION)
00315 TOKEN_MFG(MFG_TEMP_CAL, CREATOR_MFG_TEMP_CAL,
00316            0, 0, tokTypeMfgTempCal, 1,
00317           0xFFFF)
00318
00319 TOKEN_NEXT_ADDRESS(MFG_FIB_VERSION_ADDR,
      MFG_FIB_VERSION_LOCATION)
00320 TOKEN_MFG(MFG_FIB_VERSION, CREATOR_MFG_FIB_VERSION,
00321            0, 0, tokTypeMfgFibVersion, 1,
00322           CURRENT_MFG_TOKEN_VERSION)
00323
00324 TOKEN_NEXT_ADDRESS(MFG_FIB_CHECKSUM_ADDR,
      MFG_FIB_CHECKSUM_LOCATION)
00325 TOKEN_MFG(MFG_FIB_CHECKSUM, CREATOR_MFG_FIB_CHECKSUM,
00326            0, 0, tokTypeMfgFibChecksum, 1,
00327           0xFFFF)
00328
00329 TOKEN_NEXT_ADDRESS(MFG_FIB_OBS_ADDR,MFG_FIB_OBS_LOCATION)
00330 TOKEN_MFG(MFG_FIB_OBS, CREATOR_MFG_FIB_OBS,
00331            0, 0, tokTypeMfgFibObs, 1,
00332           {0xFFFF,0x03FC,0xAA55,0xFFFF})
00333
00334
00335 //--- Customer Information Block ---
00336 TOKEN_NEXT_ADDRESS(MFG_CIB_OBS_ADDR,MFG_CIB_OBS_LOCATION)
00337 TOKEN_MFG(MFG_CIB_OBS, CREATOR_MFG_CIB_OBS,
00338            0, 0, tokTypeMfgCibObs, 1,
00339           {0x5AA5,0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF})
00340
00341 TOKEN_NEXT_ADDRESS(MFG_CUSTOM_VERSION_ADDR,
      MFG_CUSTOM_VERSION_LOCATION)
00342 TOKEN_MFG(MFG_CUSTOM_VERSION, CREATOR_MFG_CUSTOM_VERSION
      ,
00343            0, 0, tokTypeMfgCustomVersion, 1,
00344           CURRENT_MFG_CUSTOM_VERSION)
00345
00346 TOKEN_NEXT_ADDRESS(MFG_CUSTOM_EUI_64_ADDR,
      MFG_CUSTOM_EUI_64_LOCATION)
00347 TOKEN_MFG(MFG_CUSTOM_EUI_64, CREATOR_MFG_CUSTOM_EUI_64
      ,
```

```
00348            0, 0, tokTypeMfgCustomEui64, 1,
00349            {0,3,3,3,3,3,3,0})
00350
00351 TOKEN_NEXT_ADDRESS(MFG_STRING_ADDR,MFG_STRING_LOCATION)
00352 TOKEN_MFG(MFG_STRING, CREATOR_MFG_STRING,
00353            0, 0, tokTypeMfgString, 1,
00354            {0,})
00355
00356 TOKEN_NEXT_ADDRESS(MFG_BOARD_NAME_ADDR,
      MFG_BOARD_NAME_LOCATION)
00357 TOKEN_MFG(MFG_BOARD_NAME, CREATOR_MFG_BOARD_NAME,
00358            0, 0, tokTypeMfgBoardName, 1,
00359            {0,})
00360
00361 TOKEN_NEXT_ADDRESS(MFG_MANUF_ID_ADDR,MFG_MANUF_ID_LOCATION)
00362 TOKEN_MFG(MFG_MANUF_ID, CREATOR_MFG_MANUF_ID,
00363            0, 0, tokTypeMfgManufId, 1,
00364            {0x00,0x00,}) // default to 0 for ember
00365
00366 TOKEN_NEXT_ADDRESS(MFG_PHY_CONFIG_ADDR,
      MFG_PHY_CONFIG_LOCATION)
00367 TOKEN_MFG(MFG_PHY_CONFIG, CREATOR_MFG_PHY_CONFIG,
00368            0, 0, tokTypeMfgPhyConfig, 1,
00369            {0x00,0x00,}) // default to non-boost mode, internal pa.
00370
00371 TOKEN_NEXT_ADDRESS(MFG_BOOTLOAD_AES_KEY_ADDR,
      MFG_BOOTLOAD_AES_KEY_LOCATION)
00372 TOKEN_MFG(MFG_BOOTLOAD_AES_KEY, CREATOR_MFG_BOOTLOAD_AES_KEY
      ,
00373            0, 0, tokTypeMfgBootloadAesKey, 1,
00374            {0xFF,}) // default key is all f's
00375
00376 TOKEN_NEXT_ADDRESS(MFG_EZSP_STORAGE_ADDR,
      MFG_EZSP_STORAGE_LOCATION)
00377 TOKEN_MFG(MFG_EZSP_STORAGE, CREATOR_MFG_EZSP_STORAGE,
00378            0, 0, tokTypeMfgEzspStorage, 1,
00379            { 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF })
00380
00381 TOKEN_NEXT_ADDRESS(MFG_ASH_CONFIG_ADDR,
      MFG_ASH_CONFIG_LOCATION)
00382 TOKEN_MFG(MFG_ASH_CONFIG, CREATOR_MFG_ASH_CONFIG,
00383            0, 1, tokTypeMfgAshConfig, MFG_ASH_CONFIG_ARRAY_SIZE,
00384            { 0xFFFF, })
00385
00386 TOKEN_NEXT_ADDRESS(MFG_CBKE_DATA_ADDR,MFG_CBKE_DATA_LOCATION)
00387 TOKEN_MFG(MFG_CBKE_DATA, CREATOR_MFG_CBKE_DATA,
00388            0, 0, tokTypeMfgCbkeData, 1,
00389            {0xFF,})
00390
00391 TOKEN_NEXT_ADDRESS(MFG_INSTALLATION_CODE_ADDR,
      MFG_INSTALLATION_CODE_LOCATION)
00392 TOKEN_MFG(MFG_INSTALLATION_CODE, CREATOR_MFG_INSTALLATION_CODE
      ,
00393            0, 0, tokTypeMfgInstallationCode, 1,
00394            {0xFF,})
00395
00396 TOKEN_NEXT_ADDRESS(MFG_OSC24M_BIAS_TRIM_ADDR,
      MFG_OSC24M_BIAS_TRIM_LOCATION)
00397 TOKEN_MFG(MFG_OSC24M_BIAS_TRIM, CREATOR_MFG_OSC24M_BIAS_TRIM
      ,
00398            0, 0, tokTypeMfgOsc24mBiasTrim, 1,
00399            {0xFF,})
00400
00401 TOKEN_NEXT_ADDRESS(MFG_SYNTH_FREQ_OFFSET_ADDR,
      MFG_SYNTH_FREQ_OFFSET_LOCATION)
00402 TOKEN_MFG(MFG_SYNTH_FREQ_OFFSET, CREATOR_MFG_SYNTH_FREQ_OFFSET
      ,
00403            0, 0, tokTypeMfgSynthFreqOffset, 1,
00404            {0xFF,0xFF,})
00405
00406 TOKEN_NEXT_ADDRESS(MFG_OSC24M_SETTLE_DELAY_ADDR,
      MFG_OSC24M_SETTLE_DELAY_LOCATION)
00407 TOKEN_MFG(MFG_OSC24M_SETTLE_DELAY, CREATOR_MFG_OSC24M_SETTLE_DELAY
      ,
00408            0, 0, tokTypeMfgOsc24mSettleDelay, 1,
00409            100)
00410
00411 TOKEN_NEXT_ADDRESS(MFG_SECURITY_CONFIG_ADDR,
      MFG_SECURITY_CONFIG_LOCATION)
00412 TOKEN_MFG(MFG_SECURITY_CONFIG, CREATOR_MFG_SECURITY_CONFIG
```

```
      ,
00413              0, 0, tokTypeMfgSecurityConfig, 1,
00414              { 0xFF, 0xFF })
00415
00416 TOKEN_NEXT_ADDRESS(MFG_CCA_THRESHOLD_ADDR,
      MFG_CCA_THRESHOLD_LOCATION)
00417 TOKEN_MFG(MFG_CCA_THRESHOLD, CREATOR_MFG_CCA_THRESHOLD
      ,
00418              0, 0, tokTypeMfgCcaThreshold, 1,
00419              {0xFF, 0xFF,})
00420
00421 TOKEN_NEXT_ADDRESS(MFG_EUI_64_ADDR,MFG_EUI_64_LOCATION)
00422 TOKEN_MFG(MFG_EUI_64, CREATOR_MFG_EUI_64,
00423              0, 0, tokTypeMfgEui64, 1,
00424              {3,3,3,3,0,0,0,0})
00425
00426 #endif //DEFINETOKENS
00427
00428
00429 #ifdef APPLICATION_MFG_TOKEN_HEADER
00430   #include APPLICATION_MFG_TOKEN_HEADER
00431 #endif
00432
00433 #undef TOKEN_NEXT_ADDRESS
00434
```

8.109   token-stack.h File Reference

```
#include "token-phy.h"
```

Macros

- #define TOKEN_NEXT_ADDRESS(region, address)
- #define CURRENT_STACK_TOKEN_VERSION

Convenience Macros

The following convenience macros are used to simplify the definition process for commonly specified parameters to the basic TOKEN_DEF macro. Please see hal/micro/token.h for a more complete explanation.

- #define DEFINE_BASIC_TOKEN(name, type,...)
- #define DEFINE_COUNTER_TOKEN(name, type,...)
- #define DEFINE_INDEXED_TOKEN(name, type, arraysize,...)
- #define DEFINE_FIXED_BASIC_TOKEN(name, type, address,...)
- #define DEFINE_FIXED_COUNTER_TOKEN(name, type, address,...)
- #define DEFINE_FIXED_INDEXED_TOKEN(name, type, arraysize, address,...)
- #define DEFINE_MFG_TOKEN(name, type, address,...)

Creator Codes

The CREATOR is used as a distinct identifier tag for the token.

The CREATOR is necessary because the token name is defined differently depending on the hardware platform, therefore the CREATOR makes sure that token definitions and data stay tagged and known. The only requirement is that each creator definition must be unique. Please see hal/micro/token.h for a more complete explanation.

- #define CREATOR_STACK_NVDATA_VERSION
- #define CREATOR_STACK_BOOT_COUNTER
- #define CREATOR_STACK_NONCE_COUNTER
- #define CREATOR_STACK_ANALYSIS_REBOOT
- #define CREATOR_STACK_KEYS
- #define CREATOR_STACK_NODE_DATA
- #define CREATOR_STACK_CLASSIC_DATA
- #define CREATOR_STACK_ALTERNATE_KEY
- #define CREATOR_STACK_APS_FRAME_COUNTER
- #define CREATOR_STACK_TRUST_CENTER
- #define CREATOR_STACK_NETWORK_MANAGEMENT
- #define CREATOR_STACK_PARENT_INFO
- #define CREATOR_MULTI_NETWORK_STACK_KEYS
- #define CREATOR_MULTI_NETWORK_STACK_NODE_DATA
- #define CREATOR_MULTI_NETWORK_STACK_ALTERNATE_KEY
- #define CREATOR_MULTI_NETWORK_STACK_TRUST_CENTER
- #define CREATOR_MULTI_NETWORK_STACK_NETWORK_MANAGEMENT
- #define CREATOR_MULTI_NETWORK_STACK_PARENT_INFO
- #define CREATOR_MULTI_NETWORK_STACK_NONCE_COUNTER
- #define CREATOR_STACK_BINDING_TABLE
- #define CREATOR_STACK_CHILD_TABLE
- #define CREATOR_STACK_KEY_TABLE
- #define CREATOR_STACK_CERTIFICATE_TABLE
- #define CREATOR_STACK_ZLL_DATA
- #define CREATOR_STACK_ZLL_SECURITY

8.109.1 Detailed Description

Definitions for stack tokens. See Stack Tokens for documentation. The file token-stack.h should not be included directly. It is accessed by the other token files.

Definition in file token-stack.h.

8.110 token-stack.h

```
00001
00046 #ifndef DEFINEADDRESSES
00047
00059   #define TOKEN_NEXT_ADDRESS(region, address)
00060 #endif
00061
00062
00063 // The basic TOKEN_DEF macro should not be used directly since the simplified
00064 //  definitions are safer to use.  For completeness of information, the basic
00065 //  macro has the following format:
00066 //
00067 //  TOKEN_DEF(name,creator,iscnt,isidx,type,arraysize,...)
00068 //  name - The root name used for the token
00069 //  creator - a "creator code" used to uniquely identify the token
00070 //  iscnt - a boolean flag that is set to identify a counter token
00071 //  isidx - a boolean flag that is set to identify an indexed token
00072 //  type - the basic type or typdef of the token
00073 //  arraysize - the number of elements making up an indexed token
00074 //  ... - initializers used when resetting the tokens to default values
00075 //
00076 //
00077 // The following convenience macros are used to simplify the definition
00078 //  process for commonly specified parameters to the basic TOKEN_DEF macro
```

```
00079 //  DEFINE_BASIC_TOKEN(name, type, ...)
00080 //  DEFINE_INDEXED_TOKEN(name, type, arraysize, ...)
00081 //  DEFINE_COUNTER_TOKEN(name, type, ...)
00082 //  DEFINE_FIXED_BASIC_TOKEN(name, type, address, ...)
00083 //  DEFINE_FIXED_INDEXED_TOKEN(name, type, arraysize, address, ...)
00084 //  DEFINE_FIXED_COUNTER_TOKEN(name, type, address, ...)
00085 //  DEFINE_MFG_TOKEN(name, type, address, ...)
00086 //
00087
00095 #define DEFINE_BASIC_TOKEN(name, type, ...)  \
00096   TOKEN_DEF(name, CREATOR_##name, 0, 0, type, 1,  __VA_ARGS__)
00097
00098 #define DEFINE_COUNTER_TOKEN(name, type, ...)  \
00099   TOKEN_DEF(name, CREATOR_##name, 1, 0, type, 1,  __VA_ARGS__)
00100
00101 #define DEFINE_INDEXED_TOKEN(name, type, arraysize, ...)  \
00102   TOKEN_DEF(name, CREATOR_##name, 0, 1, type, (arraysize),  __VA_ARGS__)
00103
00104 #define DEFINE_FIXED_BASIC_TOKEN(name, type, address, ...)  \
00105   TOKEN_NEXT_ADDRESS(name,(address))                        \
00106   TOKEN_DEF(name, CREATOR_##name, 0, 0, type, 1,  __VA_ARGS__)
00107
00108 #define DEFINE_FIXED_COUNTER_TOKEN(name, type, address, ...)  \
00109   TOKEN_NEXT_ADDRESS(name,(address))                          \
00110   TOKEN_DEF(name, CREATOR_##name, 1, 0, type, 1,  __VA_ARGS__)
00111
00112 #define DEFINE_FIXED_INDEXED_TOKEN(name, type, arraysize, address, ...)  \
00113   TOKEN_NEXT_ADDRESS(name,(address))                                     \
00114   TOKEN_DEF(name, CREATOR_##name, 0, 1, type, (arraysize),  __VA_ARGS__)
00115
00116 #define DEFINE_MFG_TOKEN(name, type, address, ...)  \
00117   TOKEN_NEXT_ADDRESS(name,(address))                \
00118   TOKEN_MFG(name, CREATOR_##name, 0, 0, type, 1,  __VA_ARGS__)
00119
00123 // The Simulated EEPROM unit tests define all of their own tokens.
00124 #ifndef SIM_EEPROM_TEST
00125
00126 // The creator codes are here in one list instead of next to their token
00127 // definitions so comparision of the codes is easier.  The only requirement
00128 // on these creator definitions is that they all must be unique.  A favorite
00129 // method for picking creator codes is to use two ASCII characters inorder
00130 // to make the codes more memorable.
00131
00145 // STACK CREATORS
00146 #define CREATOR_STACK_NVDATA_VERSION                    0xFF01
00147 #define CREATOR_STACK_BOOT_COUNTER                      0xE263
00148 #define CREATOR_STACK_NONCE_COUNTER                     0xE563
00149 #define CREATOR_STACK_ANALYSIS_REBOOT                   0xE162
00150 #define CREATOR_STACK_KEYS                              0xEB79
00151 #define CREATOR_STACK_NODE_DATA                         0xEE64
00152 #define CREATOR_STACK_CLASSIC_DATA                      0xE364
00153 #define CREATOR_STACK_ALTERNATE_KEY                     0xE475
00154 #define CREATOR_STACK_APS_FRAME_COUNTER                 0xE123
00155 #define CREATOR_STACK_TRUST_CENTER                      0xE124
00156 #define CREATOR_STACK_NETWORK_MANAGEMENT                0xE125
00157 #define CREATOR_STACK_PARENT_INFO                       0xE126
00158 // MULTI-NETWORK STACK CREATORS
00159 #define CREATOR_MULTI_NETWORK_STACK_KEYS                0xE210
00160 #define CREATOR_MULTI_NETWORK_STACK_NODE_DATA           0xE211
00161 #define CREATOR_MULTI_NETWORK_STACK_ALTERNATE_KEY       0xE212
00162 #define CREATOR_MULTI_NETWORK_STACK_TRUST_CENTER        0xE213
00163 #define CREATOR_MULTI_NETWORK_STACK_NETWORK_MANAGEMENT  0xE214
00164 #define CREATOR_MULTI_NETWORK_STACK_PARENT_INFO         0xE215
00165 // Temporary solution for multi-network nwk counters: for now we define
00166 // the following counter which will be used on the network with index 1.
00167 #define CREATOR_MULTI_NETWORK_STACK_NONCE_COUNTER       0xE220
00168
00169 // APP CREATORS
00170 #define CREATOR_STACK_BINDING_TABLE                     0xE274
00171 #define CREATOR_STACK_CHILD_TABLE                       0xFF0D
00172 #define CREATOR_STACK_KEY_TABLE                         0xE456
00173 #define CREATOR_STACK_CERTIFICATE_TABLE                 0xE500
00174 #define CREATOR_STACK_ZLL_DATA                          0xE501
00175 #define CREATOR_STACK_ZLL_SECURITY                      0xE502
00176
00180
00181 // MANUFACTURING DATA
00182 // Since the manufacturing data is platform specific, we pull in the proper
00183 // file here.
00184 #if defined(AVR_ATMEGA)
```

```
00185   #include "hal/micro/avr-atmega/token-manufacturing.h"
00186 #elif defined(MSP430)
00187   #include "hal/micro/msp430/token-manufacturing.h"
00188 #elif defined(XAP2B)
00189   #include "hal/micro/xap2b/token-manufacturing.h"
00190 #elif defined(CORTEXM3)
00191   // cortexm3 handles mfg tokens seperately via mfg-token.h
00192 #elif defined(EMBER_TEST)
00193   #include "hal/micro/avr-atmega/token-manufacturing.h"
00194 #else
00195   #error no platform defined
00196 #endif
00197
00198
00200 // STACK DATA
00201 // *the addresses of these tokens must not change*
00202
00209 #define CURRENT_STACK_TOKEN_VERSION 0x03FC //MSB is version, LSB is complement
00210
00211 #ifdef DEFINETYPES
00212 typedef int16u tokTypeStackNvdataVersion;
00213 #ifdef EMBER_SIMEE2
00214   typedef int32u tokTypeStackBootCounter;
00215 #else //EMBER_SIMEE2
00216   typedef int16u tokTypeStackBootCounter;
00217 #endif //EMBER_SIMEE2
00218 typedef int16u tokTypeStackAnalysisReboot;
00219 typedef int32u tokTypeStackNonceCounter;
00220 typedef struct {
00221   int8u networkKey[16];
00222   int8u activeKeySeqNum;
00223 } tokTypeStackKeys;
00224 typedef struct {
00225   int16u panId;
00226   int8s radioTxPower;
00227   int8u radioFreqChannel;
00228   int8u stackProfile;
00229   int8u nodeType;
00230   int16u zigbeeNodeId;
00231   int8u extendedPanId[8];
00232 } tokTypeStackNodeData;
00233 typedef struct {
00234   int16u mode;
00235   int8u eui64[8];
00236   int8u key[16];
00237 } tokTypeStackTrustCenter;
00238 typedef struct {
00239   int32u activeChannels;
00240   int16u managerNodeId;
00241   int8u updateId;
00242 } tokTypeStackNetworkManagement;
00243 typedef struct {
00244   int8u parentEui[8];
00245   int16u parentNodeId;
00246 } tokTypeStackParentInfo;
00247 #endif //DEFINETYPES
00248
00249 #ifdef DEFINETOKENS
00250 // The Stack tokens also need to be stored at well-defined locations
00251 //  None of these addresses should ever change without extremely great care
00252 #define STACK_VERSION_LOCATION    128 //  2   bytes
00253 #define STACK_APS_NONCE_LOCATION  130 //  4   bytes
00254 #define STACK_ALT_NWK_KEY_LOCATION 134 // 17  bytes (key + sequence number)
00255 // reserved                      151      1   bytes
00256 #define STACK_BOOT_COUNT_LOCATION 152 //  2   bytes
00257 // reserved                      154      2   bytes
00258 #define STACK_NONCE_LOCATION      156 //  4   bytes
00259 // reserved                      160      1   bytes
00260 #define STACK_REBOOT_LOCATION     161 //  2   bytes
00261 // reserved                      163      7   bytes
00262 #define STACK_KEYS_LOCATION       170 //  17  bytes
00263 // reserved                      187      5   bytes
00264 #define STACK_NODE_DATA_LOCATION  192 //  16  bytes
00265 #define STACK_CLASSIC_LOCATION    208 //  26  bytes
00266 #define STACK_TRUST_CENTER_LOCATION 234 //26  bytes
00267 // reserved                      260      8   bytes
00268 #define STACK_NETWORK_MANAGEMENT_LOCATION 268
00269                                     //  7   bytes
00270 // reserved                      275      109 bytes
00271
```

```
00272 DEFINE_FIXED_BASIC_TOKEN(STACK_NVDATA_VERSION,
00273                          tokTypeStackNvdataVersion,
00274                          STACK_VERSION_LOCATION,
00275                          CURRENT_STACK_TOKEN_VERSION
      )
00276 DEFINE_FIXED_COUNTER_TOKEN(STACK_APS_FRAME_COUNTER,
00277                            tokTypeStackNonceCounter,
00278                            STACK_APS_NONCE_LOCATION,
00279                            0x00000000)
00280 DEFINE_FIXED_BASIC_TOKEN(STACK_ALTERNATE_KEY,
00281                          tokTypeStackKeys,
00282                          STACK_ALT_NWK_KEY_LOCATION,
00283                          {0,})
00284 DEFINE_FIXED_COUNTER_TOKEN(STACK_BOOT_COUNTER,
00285                            tokTypeStackBootCounter,
00286                            STACK_BOOT_COUNT_LOCATION,
00287                            0x0000)
00288 DEFINE_FIXED_COUNTER_TOKEN(STACK_NONCE_COUNTER,
00289                            tokTypeStackNonceCounter,
00290                            STACK_NONCE_LOCATION,
00291                            0x00000000)
00292 DEFINE_FIXED_BASIC_TOKEN(STACK_ANALYSIS_REBOOT,
00293                          tokTypeStackAnalysisReboot,
00294                          STACK_REBOOT_LOCATION,
00295                          0x0000)
00296 DEFINE_FIXED_BASIC_TOKEN(STACK_KEYS,
00297                          tokTypeStackKeys,
00298                          STACK_KEYS_LOCATION,
00299                          {0,})
00300 DEFINE_FIXED_BASIC_TOKEN(STACK_NODE_DATA,
00301                          tokTypeStackNodeData,
00302                          STACK_NODE_DATA_LOCATION,
00303                          {0xFFFF,-1,0,0x00,0x00,0x0000})
00304 DEFINE_FIXED_BASIC_TOKEN(STACK_TRUST_CENTER,
00305                          tokTypeStackTrustCenter,
00306                          STACK_TRUST_CENTER_LOCATION,
00307                          {0,})
00308 DEFINE_FIXED_BASIC_TOKEN(STACK_NETWORK_MANAGEMENT,
00309                          tokTypeStackNetworkManagement,
00310                          STACK_NETWORK_MANAGEMENT_LOCATION,
00311                          {0, 0xFFFF, 0})
00312 DEFINE_BASIC_TOKEN(STACK_PARENT_INFO,
00313                    tokTypeStackParentInfo,
00314                    { {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF}, 0xFFFF}
      )
00315
00316 #endif //DEFINETOKENS
00317
00318
00320 // PHY DATA
00321 #include "token-phy.h"
00322
00324 // MULTI-NETWORK STACK TOKENS: Tokens for the networks with index > 0.
00325 // The 0-index network info is stored in the usual tokens.
00326
00327 #ifdef DEFINETOKENS
00328 #if !defined(EMBER_MULTI_NETWORK_STRIPPED)
00329 #define EXTRA_NETWORKS_NUMBER (EMBER_SUPPORTED_NETWORKS - 1)
00330 DEFINE_INDEXED_TOKEN(MULTI_NETWORK_STACK_KEYS,
00331                      tokTypeStackKeys,
00332                      EXTRA_NETWORKS_NUMBER,
00333                      {0,})
00334 DEFINE_INDEXED_TOKEN(MULTI_NETWORK_STACK_NODE_DATA,
00335                      tokTypeStackNodeData,
00336                      EXTRA_NETWORKS_NUMBER,
00337                      {0,})
00338 DEFINE_INDEXED_TOKEN(MULTI_NETWORK_STACK_ALTERNATE_KEY,
00339                      tokTypeStackKeys,
00340                      EXTRA_NETWORKS_NUMBER,
00341                      {0,})
00342 DEFINE_INDEXED_TOKEN(MULTI_NETWORK_STACK_TRUST_CENTER,
00343                      tokTypeStackTrustCenter,
00344                      EXTRA_NETWORKS_NUMBER,
00345                      {0,})
00346 DEFINE_INDEXED_TOKEN(MULTI_NETWORK_STACK_NETWORK_MANAGEMENT
      ,
00347                      tokTypeStackNetworkManagement,
00348                      EXTRA_NETWORKS_NUMBER,
00349                      {0,})
00350 DEFINE_INDEXED_TOKEN(MULTI_NETWORK_STACK_PARENT_INFO,
```

```
00351                         tokTypeStackParentInfo,
00352                         EXTRA_NETWORKS_NUMBER,
00353                         {{0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF}, 0xFFFF}
      )
00354
00355 // Temporary solution for NWK counter token: the following is used for 1-index
00356 // network.
00357 DEFINE_COUNTER_TOKEN(MULTI_NETWORK_STACK_NONCE_COUNTER,
00358                      tokTypeStackNonceCounter,
00359                      0x00000000)
00360 #endif // EMBER_MULTI_NETWORK_STRIPPED
00361 #endif // DEFINETOKENS
00362
00363
00365 // APPLICATION DATA
00366 // *If a fixed application token is desired, its address must be above 384.*
00367
00368 #ifdef DEFINETYPES
00369 typedef int8u tokTypeStackBindingTable[13];
00370 typedef int8u tokTypeStackChildTable[11];
00371 typedef int8u tokTypeStackKeyTable[25];
00372 // Certificate Table Entry
00373 //   Certificate:    48-bytes
00374 //   CA Public Key:  22-bytes
00375 //   Private Key:    21-bytes
00376 //   Flags:          1-byte
00377 #define TOKEN_CERTIFICATE_TABLE_ENTRY_SIZE (48 + 22 + 21 + 1)
00378 #define TOKEN_CERTIFICATE_TABLE_ENTRY_FLAGS_INDEX
      (TOKEN_CERTIFICATE_TABLE_ENTRY_SIZE - 1)
00379 typedef int8u tokTypeStackCertificateTable[
      TOKEN_CERTIFICATE_TABLE_ENTRY_SIZE];
00380 #endif //DEFINETYPES
00381
00382 // The following application tokens are required by the stack, but are sized by

00383 //  the application via its CONFIGURATION_HEADER, which is why they are present
00384 //  within the application data section.  Any special application defined
00385 //  tokens will follow.
00386 // NOTE: changing the size of these tokens within the CONFIGURATION_HEADER
00387 //  WILL move automatically move any custom application tokens that are defined
00388 //  in the APPLICATION_TOKEN_HEADER
00389 #ifdef DEFINETOKENS
00390 // Application tokens start at location 384 and are automatically positioned.
00391 TOKEN_NEXT_ADDRESS(APP,384)
00392 DEFINE_INDEXED_TOKEN(STACK_BINDING_TABLE,
00393                      tokTypeStackBindingTable,
00394                      EMBER_BINDING_TABLE_TOKEN_SIZE
      ,
00395                      {0,})
00396 DEFINE_INDEXED_TOKEN(STACK_CHILD_TABLE,
00397                      tokTypeStackChildTable,
00398                      EMBER_CHILD_TABLE_TOKEN_SIZE,
00399                      {0,})
00400 DEFINE_INDEXED_TOKEN(STACK_KEY_TABLE,
00401                      tokTypeStackKeyTable,
00402                      EMBER_KEY_TABLE_TOKEN_SIZE,
00403                      {0,})
00404 DEFINE_INDEXED_TOKEN(STACK_CERTIFICATE_TABLE,
00405                      tokTypeStackCertificateTable,
00406                      EMBER_CERTIFICATE_TABLE_SIZE,
00407                      {0,})
00408 #endif //DEFINETOKENS
00409
00410 // This must appear before the application header so that the token
00411 // numbering is consistent regardless of whether application tokens are
00412 // defined.
00413 #if defined(EMBER_ZLL_STACK)
00414   #include "stack/zll/zll-token-config.h"
00415 #endif
00416
00417 #ifdef APPLICATION_TOKEN_HEADER
00418   #include APPLICATION_TOKEN_HEADER
00419 #endif
00420
00421 //The tokens defined below are test tokens.  They are normally not used by
00422 //anything but are left here as a convenience so test tokens do not have to
00423 //be recreated.  If test code needs temporary, non-volatile storage, simply
00424 //uncomment and alter the set below as needed.
00425 //#define CREATOR_TT01 1
00426 //#define CREATOR_TT02 2
```

```
00427 //#define CREATOR_TT03 3
00428 //#define CREATOR_TT04 4
00429 //#define CREATOR_TT05 5
00430 //#define CREATOR_TT06 6
00431 //#ifdef DEFINETYPES
00432 //typedef int32u tokTypeTT01;
00433 //typedef int32u tokTypeTT02;
00434 //typedef int32u tokTypeTT03;
00435 //typedef int32u tokTypeTT04;
00436 //typedef int16u tokTypeTT05;
00437 //typedef int16u tokTypeTT06;
00438 //#endif //DEFINETYPES
00439 //#ifdef DEFINETOKENS
00440 //#define TT01_LOCATION 1
00441 //#define TT02_LOCATION 2
00442 //#define TT03_LOCATION 3
00443 //#define TT04_LOCATION 4
00444 //#define TT05_LOCATION 5
00445 //#define TT06_LOCATION 6
00446 //DEFINE_FIXED_BASIC_TOKEN(TT01, tokTypeTT01, TT01_LOCATION, 0x0000)
00447 //DEFINE_FIXED_BASIC_TOKEN(TT02, tokTypeTT02, TT02_LOCATION, 0x0000)
00448 //DEFINE_FIXED_BASIC_TOKEN(TT03, tokTypeTT03, TT03_LOCATION, 0x0000)
00449 //DEFINE_FIXED_BASIC_TOKEN(TT04, tokTypeTT04, TT04_LOCATION, 0x0000)
00450 //DEFINE_FIXED_BASIC_TOKEN(TT05, tokTypeTT05, TT05_LOCATION, 0x0000)
00451 //DEFINE_FIXED_BASIC_TOKEN(TT06, tokTypeTT06, TT06_LOCATION, 0x0000)
00452 //#endif //DEFINETOKENS
00453
00454
00455
00456 #else //SIM_EEPROM_TEST
00457
00458   //The Simulated EEPROM unit tests define all of their tokens via the
00459   //APPLICATION_TOKEN_HEADER macro.
00460   #ifdef APPLICATION_TOKEN_HEADER
00461     #include APPLICATION_TOKEN_HEADER
00462   #endif
00463
00464 #endif //SIM_EEPROM_TEST
00465
00466 #ifndef DEFINEADDRESSES
00467   #undef TOKEN_NEXT_ADDRESS
00468 #endif
00469
```

8.111   token.h File Reference

Macros

- #define halCommonGetToken(data, token)
- #define halCommonGetMfgToken(data, token)
- #define halCommonGetIndexedToken(data, token, index)
- #define halCommonSetToken(token, data)
- #define halCommonSetIndexedToken(token, index, data)
- #define halCommonIncrementCounterToken(token)

Functions

- EmberStatus halStackInitTokens (void)

8.111.1   Detailed Description

Token system for storing non-volatile information. See Tokens for documentation.

Definition in file token.h.

8.112 token.h

```
00001
00222 #ifndef __TOKEN_H__
00223 #define __TOKEN_H__
00224
00225 #if defined(AVR_ATMEGA)
00226   #include "avr-atmega/token.h"
00227 #elif defined(MSP430)
00228   #include "msp430/token.h"
00229 #elif defined(XAP2B)
00230   #include "xap2b/token.h"
00231 #elif defined(CORTEXM3)
00232   #ifdef MINIMAL_HAL
00233     #include "cortexm3/nvm-token.h"
00234     #include "cortexm3/mfg-token.h"
00235   #else //MINIMAL_HAL
00236     #include "cortexm3/token.h"
00237   #endif //MINIMAL_HAL
00238 #elif defined(EMBER_TEST)
00239   #include "generic/token-ram.h"
00240 #else
00241   #error invalid platform
00242 #endif
00243
00244
00252 EmberStatus halStackInitTokens(void);
00253
00254 // NOTE:
00255 // The following API as written below is purely for doxygen
00256 // documentation purposes.  The live API used in code is actually macros
00257 // defined in the platform specific token headers and provide abstraction
00258 // that can allow easy and efficient access to tokens in different
00259 // implementations.
00260
00261 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00262
00276 #define halCommonGetToken( data, token )
00277
00291 #define halCommonGetMfgToken( data, token )
00292
00307 #define halCommonGetIndexedToken( data, token, index )
00308
00321 #define halCommonSetToken( token, data )
00322
00338 #define halCommonSetIndexedToken( token, index, data )
00339
00351 #define halCommonIncrementCounterToken( token )
00352
00353 #endif //DOXYGEN_SHOULD_SKIP_THIS
00354
00355
00356
00357 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00358   // These interfaces serve only as a glue layer
00359   // to link creator codes to tokens (primarily for *test code)
00360   #define INVALID_EE_ADDRESS 0xFFFF
00361   int16u getTokenAddress(int16u creator);
00362   int8u getTokenSize(int16u creator );
00363   int8u getTokenArraySize(int16u creator);
00364 #endif //DOXYGEN_SHOULD_SKIP_THIS
00365
00366
00367 #endif // __TOKEN_H__
00368
```

8.113 token.h File Reference

```
#include "mfg-token.h"
#include "stack/config/token-stack.h"
```

Macros

- #define DEFINETYPES
- #define DEFINETOKENS
- #define TOKEN_DEF(name, creator, iscnt, isidx, type, arraysize,...)
- #define TOKEN_DEF(name, creator, iscnt, isidx, type, arraysize,...)
- #define COUNTER_TOKEN_PAD
- #define TOKEN_DEF(name, creator, iscnt, isidx, type, arraysize,...)
- #define halCommonGetToken(data, token)
- #define halCommonGetIndexedToken(data, token, index)
- #define halStackGetIndexedToken(data, token, index, size)
- #define halStackGetIdxTokenPtrOrData(ptr, token, index)
- #define halCommonSetToken(token, data)
- #define halCommonSetIndexedToken(token, index, data)
- #define halStackSetIndexedToken(token, index, data, size)
- #define halCommonIncrementCounterToken(token)

Enumerations

- enum { TOKEN_COUNT }
- enum

Functions

- void halInternalGetTokenData (void ∗data, int16u token, int8u index, int8u len)
- void halInternalSetTokenData (int16u token, int8u index, void ∗data, int8u len)
- void halInternalIncrementCounterToken (int8u token)
- void halInternalGetIdxTokenPtr (void ∗ptr, int16u ID, int8u index, int8u len)

Variables

- const int16u tokenCreators [ ]
- const boolean tokenIsCnt [ ]
- const int8u tokenSize [ ]
- const int8u tokenArraySize [ ]
- const void ∗const tokenDefaults [ ]

8.113.1 Detailed Description

Cortex-M3 Token system for storing non-volatile information. See Tokens for documentation. DOXY-GEN NOTE: This file contains definitions, functions, and information that are internal only and should not be accessed by applications. This information is still documented, but should not be published in the generated doxygen.

Definition in file cortexm3/token.h.

8.113.2 Macro Definition Documentation

8.113.2.1 #define DEFINETYPES

Description:

Simple declarations of all of the token types so that they can be referenced from anywhere in the code base.

Definition at line 33 of file cortexm3/token.h.

8.113.2.2 #define DEFINETOKENS

Definition at line 40 of file cortexm3/token.h.

8.113.2.3 #define TOKEN_DEF(*name, creator, iscnt, isidx, type, arraysize, ...*)

Description:

Enum for translating token defs into a number. This number is used as an index into the cache of token information the token system and Simulated EEPROM hold.

The special entry TOKEN_COUNT is always at the top of the enum, allowing the token and sim-eeprom system to know how many tokens there are.

Parameters

name,:	The name of the token.

Description:

Macro for translating token definitions into size variables. This provides a convenience for abstracting the 'sizeof(type)' anywhere.

Parameters

name,:	The name of the token.
type,:	The token type. The types are found in token-stack.h.

Description:

Macro for typedef'ing the CamelCase token type found in token-stack.h to a capitalized TOKEN style name that ends in _TYPE. This macro allows other macros below to use 'token##_TYPE' to declare a local copy of that token.

Parameters

name,:	The name of the token.
type,:	The token type. The types are found in token-stack.h.

Definition at line 156 of file cortexm3/token.h.

8.113.2.4 #define TOKEN_DEF(*name, creator, iscnt, isidx, type, arraysize, ...*)

Description:

Enum for translating token defs into a number. This number is used as an index into the cache of token information the token system and Simulated EEPROM hold.

The special entry TOKEN_COUNT is always at the top of the enum, allowing the token and sim-eeprom system to know how many tokens there are.

Parameters

name,:	The name of the token.

Description:

Macro for translating token definitions into size variables. This provides a convenience for abstracting the 'sizeof(type)' anywhere.

Parameters

name,:	The name of the token.
type,:	The token type. The types are found in token-stack.h.

Description:

Macro for typedef'ing the CamelCase token type found in token-stack.h to a capitalized TOKEN style name that ends in _TYPE. This macro allows other macros below to use 'token##_TYPE' to declare a local copy of that token.

Parameters

name,:	The name of the token.
type,:	The token type. The types are found in token-stack.h.

Definition at line 156 of file cortexm3/token.h.

8.113.2.5 #define COUNTER_TOKEN_PAD

Description:

A define for the token and Simulated EEPROM system that specifies, in bytes, the number of +1 marks available for a counter token. Since each mark requires a byte, this also corresponds to the number of words (COUNTER_TOKEN_PAD/2) that automatically pad out the counter tokens.

Definition at line 142 of file cortexm3/token.h.

8.113.2.6 #define TOKEN_DEF(*name, creator, iscnt, isidx, type, arraysize, ...*)

Description:

Enum for translating token defs into a number. This number is used as an index into the cache of token information the token system and Simulated EEPROM hold.

The special entry TOKEN_COUNT is always at the top of the enum, allowing the token and sim-eeprom system to know how many tokens there are.

Parameters

name,:	The name of the token.

Description:

Macro for translating token definitions into size variables. This provides a convenience for abstracting the 'sizeof(type)' anywhere.

Parameters

name,:	The name of the token.
type,:	The token type. The types are found in token-stack.h.

Description:

Macro for typedef'ing the CamelCase token type found in token-stack.h to a capitalized TOKEN style name that ends in _TYPE. This macro allows other macros below to use 'token##_TYPE' to declare a local copy of that token.

Parameters

name,:	The name of the token.
type,:	The token type. The types are found in token-stack.h.

Definition at line 156 of file cortexm3/token.h.

8.113.2.7   #define halCommonGetToken(*data, token*)

Definition at line 229 of file cortexm3/token.h.

8.113.2.8   #define halCommonGetIndexedToken(*data, token, index*)

Definition at line 232 of file cortexm3/token.h.

8.113.2.9   #define halStackGetIndexedToken(*data, token, index, size*)

Definition at line 235 of file cortexm3/token.h.

8.113.2.10   #define halStackGetIdxTokenPtrOrData(*ptr, token, index*)

Definition at line 238 of file cortexm3/token.h.

8.113.2.11   #define halCommonSetToken(*token, data*)

Definition at line 242 of file cortexm3/token.h.

8.113.2.12   #define halCommonSetIndexedToken(*token, index, data*)

Definition at line 245 of file cortexm3/token.h.

8.113.2.13   #define halStackSetIndexedToken(*token, index, data, size*)

Definition at line 248 of file cortexm3/token.h.

8.113.2.14   #define halCommonIncrementCounterToken(*token*)

Definition at line 251 of file cortexm3/token.h.

8.113.3   Enumeration Type Documentation

8.113.3.1   anonymous enum

Enumerator:

> *TOKEN_COUNT*

Definition at line 56 of file cortexm3/token.h.

8.113.3.2   anonymous enum

Definition at line 73 of file cortexm3/token.h.

8.113.4   Function Documentation

8.113.4.1   void halInternalGetTokenData (void $*$ *data,* int16u *token,* int8u *index,* int8u *len*)

Description:

> Copies the token value from non-volatile storage into a RAM location. This is the internal function that the two exposed APIs (halCommonGetToken and halCommonGetIndexedToken) expand out to. The API simplifies the access into this function by hiding the size parameter and hiding the value 0 used for the index parameter in scalar tokens.

Note

> Only the public function should be called since the public function provides the correct parameters.

Parameters

data,:	A pointer to where the data being read should be placed.
token,:	The name of the token to get data from. On this platform that name is defined as an address.
index,:	The index to access. If the token being accessed is not an indexed token, this parameter is set by the API to be 0.
len,:	The length of the token being worked on. This value is automatically set by the API to be the size of the token.

8.113.4.2   void halInternalSetTokenData (int16u *token,* int8u *index,* void ∗ *data,* int8u *len*)

Description:

Sets the value of a token in non-volatile storage. This is the internal function that the two exposed APIs (halCommonSetToken and halCommonSetIndexedToken) expand out to. The API simplifies the access into this function by hiding the size parameter and hiding the value 0 used for the index parameter in scalar tokens.

Note

Only the public function should be called since the public function provides the correct parameters.

Parameters

token,:	The name of the token to get data from. On this platform that name is defined as an address.
index,:	The index to access. If the token being accessed is not an indexed token, this parameter is set by the API to be 0.
data,:	A pointer to the data being written.
len,:	The length of the token being worked on. This value is automatically set by the API to be the size of the token.

8.113.4.3   void halInternalIncrementCounterToken (int8u *token*)

Description:

Increments the value of a token that is a counter. This is the internal function that the exposed API (halCommonIncrementCounterToken) expand out to. This internal function is used as a level of simple redirection providing clean separation from the lower token handler code.

Note

Only the public function should be called since the public function provides the correct parameters.

Parameters

token,:	The name of the token.

8.113.4.4   void halInternalGetIdxTokenPtr (void ∗ *ptr,* int16u *ID,* int8u *index,* int8u *len*)

8.113.5   Variable Documentation

8.113.5.1   const int16u tokenCreators[ ]

Description:

External declaration of an array of creator codes. Since the token and sim-eeprom systems identify tokens through an enum (see below for the enum) and these two systems need to link creator codes to their tokens, this array instantiates that link.

Parameters

creator,:	The creator code type. The codes are found in token-stack.h.

8.113.5.2 const boolean tokenIsCnt[ ]

Description:

External declaration of an array of IsCnt flags. Since the token and sim-eeprom systems identify tokens through an enum (see below for the enum) and these two systems need to know which tokens are counter tokens, this array provides that information.

Parameters

iscnt,:	The flag indicating if the token is a counter. The iscnt's are found in token-stack.h.

8.113.5.3 const int8u tokenSize[ ]

Description:

External declaration of an array of sizes. Since the token and sim-eeprom systems identify tokens through an enum (see below for the enum) and these two systems need to know the size of each token, this array provides that information.

Parameters

type,:	The token type. The types are found in token-stack.h.

8.113.5.4 const int8u tokenArraySize[ ]

Description:

External declaration of an array of array sizes. Since the token and sim-eeprom systems identify tokens through an enum (see below for the enum) and these two systems need to know the array size of each token, this array provides that information.

Parameters

arraysize,:	The array size.

8.113.5.5 const void∗ const tokenDefaults[ ]

Description:

External declaration of an array of all token default values. This array is filled with pointers to the set of constant declarations of all of the token default values. Therefore, the index into this array chooses which token's defaults to access, and the address offset chooses the byte in the defaults to use.

For example, to get the n-th byte of the i-th token, use: int8u byte = ∗(((int8u ∗)tokenDefaults[i])+(n)

Parameters

TOKEN_	*#name##_DE-FAULTS,:*	A constant declaration of the token default values, generated for all tokens.

8.114 cortexm3/token.h

```
00001
00015 #ifndef __PLAT_TOKEN_H__
00016 #define __PLAT_TOKEN_H__
00017
00018 #ifndef __TOKEN_H__
00019 #error do not include this file directly - include micro/token.h
00020 #endif
00021
00022
00023 // The manufacturing tokens live in the Info Blocks, while all other tokens
00024 // live in the Simulated EEPROM.  This means they are defined differently,
00025 // which is covered in mfg-token.h
00026 #include "mfg-token.h"
00027
00028 //-- Build structure defines
00033 #define DEFINETYPES
00034   #include "stack/config/token-stack.h"
00035 #undef DEFINETYPES
00036
00037
00038
00039 //-- Build parameter links
00040 #define DEFINETOKENS
00041
00042 #undef TOKEN_DEF
00043
00054 #define TOKEN_DEF(name,creator,iscnt,isidx,type,arraysize,...) \
00055   TOKEN_##name,
00056   enum{
00057     #include "stack/config/token-stack.h"
00058     TOKEN_COUNT
00059   };
00060 #undef TOKEN_DEF
00061
00062
00071 #define TOKEN_DEF(name,creator,iscnt,isidx,type,arraysize,...) \
00072   TOKEN_##name##_SIZE = sizeof(type),
00073   enum {
00074     #include "stack/config/token-stack.h"
00075   };
00076 #undef TOKEN_DEF
00077
00078
00088 extern const int16u tokenCreators[];
00089
00099 extern const boolean tokenIsCnt[];
00100
00109 extern const int8u tokenSize[];
00110
00119 extern const int8u tokenArraySize[];
00120
00134 extern const void * const tokenDefaults[];
00135
00142 #define COUNTER_TOKEN_PAD        50
00143
00144
00145
00156 #define TOKEN_DEF(name,creator,iscnt,isidx,type,arraysize,...) \
00157   typedef type TOKEN_##name##_TYPE;
00158   #include "stack/config/token-stack.h"
00159 #undef TOKEN_DEF
00160
00161 #undef DEFINETOKENS
00162
00163
00185 void halInternalGetTokenData(void *data, int16u
     token, int8u index, int8u len);
00186
00208 void halInternalSetTokenData(int16u token, int8u
```

```
      index, void *data, int8u len);
00209
00221 void halInternalIncrementCounterToken(int8u
      token);
00222
00223
00224 // See hal/micro/token.h for the full explanation of the token API as
00225 // instantiated below.
00226
00227 //These defines Link the public API to the private internal instance.
00228
00229 #define halCommonGetToken( data, token )                    \
00230   halInternalGetTokenData(data, token, 0x7F, token##_SIZE)
00231
00232 #define halCommonGetIndexedToken( data, token, index )      \
00233   halInternalGetTokenData(data, token, index, token##_SIZE)
00234
00235 #define halStackGetIndexedToken( data, token, index, size ) \
00236   halInternalGetTokenData(data, token, index, size)
00237
00238 #define halStackGetIdxTokenPtrOrData( ptr, token, index ) \
00239   halInternalGetIdxTokenPtr(ptr, token, index, token##_SIZE)
00240 void halInternalGetIdxTokenPtr(void *ptr, int16u
      ID, int8u index, int8u len);
00241
00242 #define halCommonSetToken( token, data )                    \
00243   halInternalSetTokenData(token, 0x7F, data, token##_SIZE)
00244
00245 #define halCommonSetIndexedToken( token, index, data )      \
00246   halInternalSetTokenData(token, index, data, token##_SIZE)
00247
00248 #define halStackSetIndexedToken( token, index, data, size ) \
00249   halInternalSetTokenData(token, index, data, size)
00250
00251 #define halCommonIncrementCounterToken( token )             \
00252   halInternalIncrementCounterToken(token);
00253
00254 // For use only by the EZSP UART protocol
00255 #ifdef EZSP_UART
00256   #ifdef CORTEXM3
00257     #define halInternalMfgTokenPointer( address )  \
00258       ((const void *)(address + DATA_BIG_INFO_BASE))
00259     #define halInternalMfgIndexedToken( type, address, index )  \
00260       (*((const type *)(address + DATA_BIG_INFO_BASE) + index))
00261   #endif
00262 #endif
00263
00264
00265 #undef TOKEN_MFG
00266
00267 #endif // __PLAT_TOKEN_H__
00268
```

8.115   trust-center.h File Reference

Macros

- #define EMBER_FORM_TRUST_CENTER_NETWORK_BITMASK
- #define EMBER_FORM_DISTRIBUTED_TRUST_CENTER_NETWORk_BITMASK

Functions

- EmberStatus emberBroadcastNextNetworkKey (EmberKeyData *key)
- EmberStatus emberSendUnicastNetworkKeyUpdate (EmberNodeId targetShort, EmberEUI64 target-Long, EmberKeyData *newKey)
- EmberStatus emberBroadcastNetworkKeySwitch (void)
- EmberJoinDecision emberTrustCenterJoinHandler (EmberNodeId newNodeId, EmberEUI64 new-NodeEui64, EmberDeviceUpdate status, EmberNodeId parentOfNewNode)

- EmberStatus emberBecomeTrustCenter (EmberKeyData ∗newNetworkKey)
- EmberStatus emberSendRemoveDevice (EmberNodeId destShort, EmberEUI64 destLong, Ember-EUI64 deviceToRemoveLong)

Variables

- EmberLinkKeyRequestPolicy emberTrustCenterLinkKeyRequestPolicy
- EmberLinkKeyRequestPolicy emberAppLinkKeyRequestPolicy

8.115.1 Detailed Description

EmberZNet security API See Security for documentation.

Definition in file trust-center.h.

8.116 trust-center.h

```
00001
00029 #define EMBER_FORM_TRUST_CENTER_NETWORK_BITMASK  \
00030   ( EMBER_STANDARD_SECURITY_MODE               \
00031     | EMBER_TRUST_CENTER_GLOBAL_LINK_KEY        \
00032     | EMBER_HAVE_NETWORK_KEY                    \
00033     | EMBER_HAVE_PRECONFIGURED_KEY )
00034
00042 #define EMBER_FORM_DISTRIBUTED_TRUST_CENTER_NETWORk_BITMASK   \
00043   ( EMBER_STANDARD_SECURITY_MODE                             \
00044     | EMBER_TRUST_CENTER_GLOBAL_LINK_KEY                     \
00045     | EMBER_DISTRIBUTED_TRUST_CENTER_MODE                    \
00046     | EMBER_HAVE_NETWORK_KEY                                 \
00047     | EMBER_HAVE_PRECONFIGURED_KEY )
00048
00071 EmberStatus emberBroadcastNextNetworkKey
    (EmberKeyData* key);
00072
00100 #if defined DOXYGEN_SHOULD_SKIP_THIS
00101 EmberStatus emberSendUnicastNetworkKeyUpdate
    (EmberNodeId targetShort,
00102                                           EmberEUI64  targetLong,
00103                                           EmberKeyData* newKey);
00104 #else
00105 EmberStatus emSendAlternateNetworkKeyToAddress(EmberNodeId
    targetShort,
00106                                               EmberEUI64 targetLong,
00107                                               EmberKeyData* newKey
    );
00108
00109 #define emberSendUnicastNetworkKeyUpdate(shortAddr, longAddr, key) \
00110   emSendAlternateNetworkKeyToAddress((shortAddr), (longAddr), (key))
00111 #endif
00112
00113
00125 #if defined DOXYGEN_SHOULD_SKIP_THIS
00126 EmberStatus emberBroadcastNetworkKeySwitch
    (void);
00127 #else
00128 EmberStatus emSendNetworkKeySwitch(EmberNodeId
    destination);
00129
00130 #define emberBroadcastNetworkKeySwitch() \
00131   emSendNetworkKeySwitch(EMBER_SLEEPY_BROADCAST_ADDRESS)
00132 #endif
00133
00178 EmberJoinDecision emberTrustCenterJoinHandler
    (EmberNodeId newNodeId,
00179                                             EmberEUI64 newNodeEui64
    ,
00180                                             EmberDeviceUpdate
    status,
```

```
00181                                                      EmberNodeId
     parentOfNewNode);
00182
00195 EmberStatus emberBecomeTrustCenter(
     EmberKeyData* newNetworkKey);
00196
00197
00211 extern EmberLinkKeyRequestPolicy
     emberTrustCenterLinkKeyRequestPolicy;
00212
00222 extern EmberLinkKeyRequestPolicy
     emberAppLinkKeyRequestPolicy;
00223
00224
00242 EmberStatus emberSendRemoveDevice(EmberNodeId
      destShort,
00243                                    EmberEUI64 destLong,
00244                                    EmberEUI64 deviceToRemoveLong);
00245
00246
00247 // @} END addtogroup
00248
```

8.117 zigbee-device-common.h File Reference

Macros

- #define ZDO_MESSAGE_OVERHEAD

Service Discovery Functions

- EmberStatus emberNodeDescriptorRequest (EmberNodeId target, EmberApsOption options)
- EmberStatus emberPowerDescriptorRequest (EmberNodeId target, EmberApsOption options)
- EmberStatus emberSimpleDescriptorRequest (EmberNodeId target, 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 destination-Endpoint, EmberApsOption options)
- EmberStatus emberUnbindRequest (EmberNodeId target, EmberEUI64 source, int8u sourceEndpoint, int16u clusterId, int8u type, EmberEUI64 destination, EmberMulticastId groupAddress, int8u destination-Endpoint, 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)
- int8u emberGetLastAppZigDevRequestSequence (void)

8.117.1 Detailed Description

ZigBee Device Object (ZDO) functions available on all platforms. See ZigBee Device Object (ZDO) Information for documentation.

Definition in file zigbee-device-common.h.

8.118 zigbee-device-common.h

```
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);
00392 int8u emberGetLastZigDevRequestSequence(
      void);
00393 #else
00394 extern int8u zigDevRequestRadius;
00395 #define emberGetZigDevRequestRadius() (zigDevRequestRadius)
00396 #define emberSetZigDevRequestRadius(x)       (zigDevRequestRadius=x)
00397 #define emberGetLastZigDevRequestSequence() \
00398   (emberGetLastAppZigDevRequestSequence())
00399 #endif
00400
00407 int8u emberGetLastAppZigDevRequestSequence
      (void);
00408
00411 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00412 //
      --------------------------------------------------------------------------
00413 // Utility functions used by the library code.
00414
00415 EmberStatus emberSendZigDevRequest(EmberNodeId
      destination,
00416                                        int16u clusterId,
00417                                        EmberApsOption options,
00418                                        int8u *contents,
00419                                        int8u length);
00420
00430 int8u emberNextZigDevRequestSequence(void);
00431
00432 #endif // DOXYGEN_SHOULD_SKIP_THIS
00433
```

8.119 zigbee-device-host.h File Reference

Device Discovery Functions

- EmberStatus emberNetworkAddressRequest (EmberEUI64 target, boolean reportKids, int8u child-StartIndex)
- EmberStatus emberIeeeAddressRequest (EmberNodeId target, boolean reportKids, int8u childStart-Index, 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)

8.119.1   Detailed Description

ZigBee Device Object (ZDO) functions not provided by the stack. See ZigBee Device Object (ZDO) Information for documentation.

Definition in file zigbee-device-host.h.

8.120   zigbee-device-host.h

```
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
```

8.121   zigbee-device-library.h File Reference

Service Discovery Functions

- EmberStatus emberMatchDescriptorsRequest (EmberNodeId target, int16u profile, EmberMessage-Buffer inClusters, EmberMessageBuffer outClusters, EmberApsOption options)

Binding Manager Functions

- EmberStatus emberEndDeviceBindRequest (int8u endpoint, EmberApsOption options)

Function to Decode Address Response Messages

- EmberNodeId emberDecodeAddressResponse (EmberMessageBuffer response, EmberEUI64 eui64-Return)

8.121.1   Detailed Description

ZigBee Device Object (ZDO) functions not provided by the stack. See ZigBee Device Object (ZDO) Information for documentation.

Definition in file zigbee-device-library.h.

8.122   zigbee-device-library.h

```
00001
00101 EmberStatus emberMatchDescriptorsRequest
      (EmberNodeId target,
00102                                      int16u profile,
00103                                      EmberMessageBuffer
      inClusters,
00104                                      EmberMessageBuffer
      outClusters,
00105                                      EmberApsOption options);
00125 EmberStatus emberEndDeviceBindRequest(int8u
       endpoint,
00126                                      EmberApsOption options);
00143 EmberNodeId emberDecodeAddressResponse(
      EmberMessageBuffer response,
00144                                      EmberEUI64 eui64Return);
00145
```

8.123   zigbee-device-stack.h File Reference

Functions

- EmberStatus emberNetworkAddressRequest (EmberEUI64 target, boolean reportKids, int8u child-StartIndex)
- EmberStatus emberIeeeAddressRequest (EmberNodeId target, boolean reportKids, int8u childStart-Index, EmberApsOption options)
- EmberStatus emberEnergyScanRequest (EmberNodeId target, int32u scanChannels, int8u scanDuration, int16u scanCount)
- EmberStatus emberSetNetworkManagerRequest (EmberNodeId networkManager, int32u activeChannels)
- EmberStatus emberChannelChangeRequest (int8u channel)
- EmberStatus emberSendDeviceAnnouncement (void)
- int8u emberGetLastStackZigDevRequestSequence (void)

8.123.1 Detailed Description

ZigBee Device Object (ZDO) functions included in the stack. See ZigBee Device Object for documentation.

Definition in file zigbee-device-stack.h.

8.124 zigbee-device-stack.h

```
00001
00010 #ifndef __ZIGBEE_DEVICE_STACK_H__
00011 #define __ZIGBEE_DEVICE_STACK_H__
00012
00035 EmberStatus emberNetworkAddressRequest(
     EmberEUI64 target,
00036                                        boolean reportKids,
00037                                        int8u childStartIndex);
00038
00055 EmberStatus emberIeeeAddressRequest(
     EmberNodeId target,
00056                                       boolean reportKids,
00057                                       int8u childStartIndex,
00058                                       EmberApsOption options);
00059
00079 EmberStatus emberEnergyScanRequest(EmberNodeId
      target,
00080                                     int32u scanChannels,
00081                                     int8u  scanDuration,
00082                                     int16u scanCount);
00083
00097 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00098 EmberStatus emberSetNetworkManagerRequest
     (EmberNodeId networkManager,
00099                                          int32u activeChannels);
00100 #else
00101 #define emberSetNetworkManagerRequest(manager, channels)     \
00102 (emberEnergyScanRequest(EMBER_SLEEPY_BROADCAST_ADDRESS,       \
00103                         (channels),                          \
00104                         0xFF,                                \
00105                         (manager)))
00106 #endif
00107
00121 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00122 EmberStatus emberChannelChangeRequest(int8u
      channel);
00123 #else
00124 #define emberChannelChangeRequest(channel)                   \
00125 (emberEnergyScanRequest(EMBER_SLEEPY_BROADCAST_ADDRESS,       \
00126                         BIT32(channel),                      \
00127                         0xFE,                                \
00128                         0))
00129 #endif
00130
00141 EmberStatus emberSendDeviceAnnouncement(
     void);
00142
00149 int8u emberGetLastStackZigDevRequestSequence
     (void);
00150
00154 #endif  // __ZIGBEE_DEVICE_STACK_H__
```