



EM35x

EmberZNet 5.7.0 API Reference: For the EM35x SoC Platform

February 21, 2016
120-3022-000-5700

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



Disclaimer

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

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

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

About This Guide

Purpose

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

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

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

Audience

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

Getting Help

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

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

Chapter 1

Introduction

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

- [EmberZNet Stack API Reference](#)
- [Ember ZigBee RF4CE 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:

EmberZNet Stack API Reference	15
Stack Information	16
Ember Common Data Types	30
Network Formation	80
Packet Buffers	88
Sending and Receiving Messages	98
End Devices	118
Security	126
Trust Center	138
Binding Table	143
Configuration	148
Status Codes	159
Stack Tokens	182
ZigBee Device Object	188
Bootloader	192
Event Scheduling	194
Multi-Network Manager	200
Ember ZigBee Light Link (ZLL) APIs and Handlers	201
Ember ZigBee Light Link (ZLL) Data Types	206
Manufacturing and Functional Test Library	210
Debugging Utilities	217
Ember ZigBee RF4CE Stack API Reference	221
Ember ZigBee RF4CE APIs and Handlers	222
Ember ZigBee RF4CE Data Types	236
Packet Buffers	88
Configuration	148
Status Codes	159
Stack Tokens	182
Bootloader	192
Event Scheduling	194
Multi-Network Manager	200
Manufacturing and Functional Test Library	210
Debugging Utilities	217

Hardware Abstraction Layer (HAL) API Reference	241
Common Microcontroller Functions	242
Token Access	256
Tokens	257
Simulated EEPROM	263
Simulated EEPROM 2	266
Sample APIs for Peripheral Access	267
Serial UART Communication	268
Button Control	276
Buzzer Control	278
LED Control	284
Flash Memory Control	286
USB Device Stack Library	287
USB_COMMON	290
USB_DEVICE	308
System Timer Control	316
Symbol Timer Control	321
HAL Configuration	323
Sample Breakout Board Configuration	324
IAR PLATFORM_HEADER Configuration	373
Common PLATFORM_HEADER Configuration	398
NVIC Configuration	405
Reset Cause Type Definitions	406
HAL Utilities	407
Crash and Watchdog Diagnostics	408
Cyclic Redundancy Code (CRC)	410
Random Number Generation	412
Network to Host Byte Order Conversion	413
Bootloader Interfaces	414
Common	415
Standalone	420
Application	422
Custom Bootloader HAL	428
Common	429
GPIO	430
Serial	433
Standalone	436
Application	439
Application Utilities API Reference	446
Forming and Joining Networks	447
ZigBee Device Object (ZDO) Information	453
Message Fragmentation	467
Network Manager	473
Serial Communication	476
Command Interpreter	488
Adc	496

Chapter 4

Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

EmberAesMmoHashContext	This data structure contains the context data when calculating an AES MMO hash (message digest)	506
EmberApsFrame	An in-memory representation of a ZigBee APS frame of an incoming or outgoing message	507
EmberBindingTableEntry	Defines an entry in the binding table	508
EmberCertificate283k1Data	This data structure contains the certificate data that is used for Certificate Based Key Exchange (CBKE) in SECT283k1 Elliptical Cryptography	509
EmberCertificateData	This data structure contains the certificate data that is used for Certificate Based Key Exchange (CBKE)	510
EmberCommandEntry	Command entry for a command table	510
EmberCurrentSecurityState	This describes the security features used by the stack for a joined device	512
EmberEndpoint	Gives the endpoint information for a particular endpoint	513
EmberEndpointDescription	Endpoint information (a ZigBee Simple Descriptor)	513
EmberEventControl	Control structure for events	515
EmberEventData_S	Complete events with a control and a handler procedure	515
EmberInitialSecurityState	This describes the Initial Security features and requirements that will be used when forming or joining the network	516
EmberKeyData	This data structure contains the key data that is passed into various other functions	518

EmberKeyStruct	This describes a one of several different types of keys and its associated data	518
EmberMacFilterMatchStruct	This structure indicates a matching raw MAC message has been received by the application configured MAC filters	520
EmberMessageDigest	This data structure contains an AES-MMO Hash (the message digest) .	520
EmberMfgSecurityStruct	This structure is used to get/set the security config that is stored in manufacturing tokens	521
EmberMulticastTableEntry	Defines an entry in the multicast table	521
EmberNeighborTableEntry	Defines an entry in the neighbor table	522
EmberNetworkInitStruct	Defines the network initialization configuration that should be used when <code>emberNetworkInitExtended()</code> is called by the application	524
EmberNetworkParameters	Holds network parameters	524
EmberPrivateKey283k1Data	This data structure contains the private key data that is used for Certificate Based Key Exchange (CBKE) in SECT283k1 Elliptical Cryptography	526
EmberPrivateKeyData	This data structure contains the private key data that is used for Certificate Based Key Exchange (CBKE)	526
EmberPublicKey283k1Data	This data structure contains the public key data that is used for Certificate Based Key Exchange (CBKE) in SECT283k1 Elliptical Cryptography	527
EmberPublicKeyData	This data structure contains the public key data that is used for Certificate Based Key Exchange (CBKE)	528
EmberReleaseTypeStruct	A structure relating version types to human readable strings	528
EmberRf4ceApplicationInfo	Defines the application information block (see section 3.3.1, Figure 17)	529
EmberRf4cePairingTableEntry	The internal representation of a pairing table entry	530
EmberRf4ceVendorInfo	Defines the vendor information block (see section 3.3.1, Figure 16) . .	532
EmberRouteTableEntry	Defines an entry in the route table	533
EmberSignature283k1Data	This data structure contains a DSA signature used in SECT283k1 Elliptical Cryptography. It is the bit concatenation of the 'r' and 's' components of the signature	534
EmberSignatureData	This data structure contains a DSA signature. It is the bit concatenation of the 'r' and 's' components of the signature	534
EmberSmacData	This data structure contains the Shared Message Authentication Code (SMAC) data that is used for Certificate Based Key Exchange (CBKE)	535

EmberTaskControl	
Control structure for tasks	535
EmberTokTypeStackZllData	
.	536
EmberTokTypeStackZllSecurity	
.	537
EmberVersion	
Version struct containing all version information	538
EmberZigbeeNetwork	
Defines a ZigBee network and the associated parameters	539
EmberZllAddressAssignment	
Network and group address assignment information	540
EmberZllDeviceInfoRecord	
Information discovered during a ZLL scan about the ZLL device's endpoint information	541
EmberZllInitialSecurityState	
This describes the Initial Security features and requirements that will be used when forming or joining ZigBee Light Link networks	542
EmberZllNetwork	
Information about the ZLL network and specific device that responded to a ZLL scan request	543
EmberZllSecurityAlgorithmData	
Information about the ZLL security being and how to transmit the network key to the device securely	545
HalEepromInformationType	
This structure defines a variety of information about the attached external EEPROM device	546
InterPanHeader	
A struct for keeping track of all of the header info	547
USB_ConfigurationDescriptor_TypeDef	
USB Configuration Descriptor	548
USB_DeviceDescriptor_TypeDef	
USB Device Descriptor	550
USB_EndpointDescriptor_TypeDef	
USB Endpoint Descriptor	552
USB_InterfaceDescriptor_TypeDef	
USB Interface Descriptor	554
USB_Setup_TypeDef	
USB Setup request package	556
USB_StringDescriptor_TypeDef	
USB String Descriptor	557
USBD_Callbacks_TypeDef	
USB Device stack callback structure	558
USBD_Init_TypeDef	
USB Device stack initialization structure	559

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

_EM35x_API.top	Starting page for the Ember API documentation for the EM35x exclusively for building documentation	562
adc.h	Header for A/D converter	563
cortexm3adc.h	Header for EM35x A/D converter	565
ami-inter-pan-host.h	Utilities for sending and receiving ZigBee AMI InterPAN messages. See Sending and Receiving Messages for documentation	568
ami-inter-pan.h	Utilities for sending and receiving ZigBee AMI InterPAN messages. See Sending and Receiving Messages for documentation	569
app-bootloader.h	570
binding-table.h	571
bootload.h	572
bootloader-eeprom.h	573
bootloader-gpio.h	575
bootloader-interface-app.h	577
bootloader-interface-standalone.h	578
bootloader-interface.h	580
bootloader-serial.h	581
button.h	582
buzzer.h	584
child.h	585
command-interpreter2.h	Processes commands coming from the serial port. See Command Interpreter for documentation	588
config.h	591
crc.h	592
dev0680.h	601
diagnostic.h	610

em_usb.h	USB protocol stack library API for EFM32	619
em_usbd.c	USB protocol stack library, device API	627
em_usbd.h	USB protocol stack library API for EFM32	636
em_usbdch9.c	USB protocol stack library, USB chapter 9 command handler	639
em_usbdep.c	USB protocol stack library, USB device endpoint handlers	644
em_usbhal.c	USB protocol stack library, low level USB peripheral access	648
em_usbhal.h	USB protocol stack library, low level USB peripheral access	651
em_usbint.c	USB protocol stack library, USB device peripheral interrupt handlers . .	656
em_usbtypes.h	USB protocol stack library, internal type definitions	660
ember-configuration-defaults.h	User-configurable stack memory allocation defaults	663
ember-debug.h	668
ember-types.h	Ember data type definitions	683
ember.h	The master include file for the EmberZNet API	703
endian.h	704
error-def.h	Return-code definitions for EmberZNet stack API functions	709
error.h	Return codes for Ember API functions and module definitions	722
event.h	Scheduling events for future execution. See Event Scheduling for documentation	724
flash.h	725
form-and-join.h	Utilities for forming and joining networks	727
fragment-host.h	Fragmented message support for EZSP Hosts. Splits long messages into smaller blocks for transmission and reassembles received blocks. See Message Fragmentation for documentation	728
fragment.h	Splits long messages into smaller blocks for transmission and reassembles received blocks. See Message Fragmentation for documentation . .	729
hal.h	Generic set of HAL includes for all platforms	731
iar.h	738
led.h	746
message.h	EmberZNet API for sending and receiving messages. See Sending and Receiving Messages for documentation	748
mfplib.h	750
micro-common.h	Minimal Hal functions common across all microcontroller-specific files. See Common Microcontroller Functions for documentation	752

micro-types.h	This file handles defines and enums related to all the micros	754
micro.h	Full HAL functions common across all microcontroller-specific files. See Common Microcontroller Functions for documentation	761
cortexm3/micro.h	Utility and convenience functions for EM35x microcontroller. See Common Microcontroller Functions for documentation	764
multi-network.h	EmberZNet API for multi-network support. See Multi-Network Manager for documentation	766
network-formation.h	767
network-manager.h	Utilities for use by the ZigBee network manager. See Network Manager for documentation	769
nvic-config.h	769
packet-buffer.h	Packet buffer allocation and management routines See Packet Buffers for documentation	774
platform-common.h	779
random.h	782
ref0657.h	789
reset-def.h	Definitions for all the reset cause types	796
rf4ce-api.h	ZigBee RF4CE stack APIs and callbacks. See Ember ZigBee RF4CE APIs and Handlers for documentation	800
rf4ce-types.h	Zigbee RF4CE types and defines. See Ember ZigBee RF4CE APIs and Handlers for documentation	803
security.h	EmberZNet security API. See Security for documentation	807
hal/micro/serial.h	Serial hardware abstraction layer interfaces. See Serial UART Communication for documentation	810
app/util/serial/serial.h	High-level serial communication functions	816
sim-eeprom.h	Simulated EEPROM system for wear leveling token storage across flash. See Simulated EEPROM for documentation	818
stack-info.h	EmberZNet API for accessing and setting stack information. See Stack Information for documentation	820
standalone-bootloader.h	823
symbol-timer.h	Functions that provide access to symbol time. One symbol period is 16 microseconds	824
system-timer.h	825
token-manufacturing.h	Definitions for manufacturing tokens	831
token-stack.h	Definitions for stack tokens. See Stack Tokens for documentation	839

token.h	
Token system for storing non-volatile information. See Tokens for documentation	845
cortexm3/token.h	
Cortex-M3 Token system for storing non-volatile information. See Tokens for documentation	853
trust-center.h	
EmberZNet security API See Security for documentation	856
zigbee-device-common.h	
ZigBee Device Object (ZDO) functions available on all platforms. See ZigBee Device Object (ZDO) Information for documentation	858
zigbee-device-host.h	
ZigBee Device Object (ZDO) functions not provided by the stack. See ZigBee Device Object (ZDO) Information for documentation	861
zigbee-device-library.h	
ZigBee Device Object (ZDO) functions not provided by the stack. See ZigBee Device Object (ZDO) Information for documentation	862
zigbee-device-stack.h	
ZigBee Device Object (ZDO) functions included in the stack	863
zll-api.h	864
zll-types.h	
Ember data type definitions	866

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](#)
- [Multi-Network Manager](#)
- [Ember ZigBee Light Link \(ZLL\) APIs and Handlers](#)
- [Ember ZigBee Light Link \(ZLL\) Data Types](#)
- [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)
- bool `emberStackIsUp` (void)
- `EmberEUI64` `emberGetEui64` (void)
- bool `emberIsLocalEui64` (`EmberEUI64` eui64)
- `EmberNodeId` `emberGetNodeId` (void)
- `EmberNodeId` `emberRadioGetNodeId` (void)
- void `emberSetManufacturerCode` (`uint16_t` code)
- void `emberSetPowerDescriptor` (`uint16_t` descriptor)
- void `emberSetMaximumIncomingTransferSize` (`uint16_t` size)
- void `emberSetMaximumOutgoingTransferSize` (`uint16_t` size)
- void `emberSetDescriptorCapability` (`uint8_t` capability)
- `EmberStatus` `emberGetNetworkParameters` (`EmberNetworkParameters` *parameters)
- `EmberStatus` `emberGetNodeType` (`EmberNodeType` *resultLocation)
- `EmberStatus` `emberSetRadioChannel` (`uint8_t` channel)
- `uint8_t` `emberGetRadioChannel` (void)
- `EmberStatus` `emberSetRadioPower` (`int8_t` power)
- `int8_t` `emberGetRadioPower` (void)
- `EmberPanId` `emberGetPanId` (void)
- `EmberPanId` `emberRadioGetPanId` (void)
- void `emberGetExtendedPanId` (`uint8_t` *resultLocation)
- `uint8_t` `emberGetEndpoint` (`uint8_t` index)
- bool `emberGetEndpointDescription` (`uint8_t` endpoint, `EmberEndpointDescription` *result)
- `uint16_t` `emberGetEndpointCluster` (`uint8_t` endpoint, `EmberClusterListId` listId, `uint8_t` listIndex)
- bool `emberIsNodeIdValid` (`EmberNodeId` nodeId)

- `EmberNodeId emberLookupNodeIdByEui64 (EmberEUI64 eui64)`
- `EmberStatus emberLookupEui64ByNodeId (EmberNodeId nodeId, EmberEUI64 eui64-`
Return)
- `void emberCounterHandler (EmberCounterType type, uint8_t data)`
- `void emberStackTokenChangedHandler (uint16_t tokenAddress)`
- `EmberStatus emberGetNeighbor (uint8_t index, EmberNeighborTableEntry *result)`
- `EmberStatus emberGetRouteTableEntry (uint8_t index, EmberRouteTableEntry *result)`
- `uint8_t emberStackProfile (void)`
- `uint8_t emberTreeDepth (void)`
- `uint8_t emberNeighborCount (void)`
- `uint8_t emberRouteTableSize (void)`
- `uint8_t emberNextZigbeeSequenceNumber (void)`

Variables

- PGM `uint8_t emberStackProfileId []`
- `uint8_t emberEndpointCount`
- `EmberEndpoint emberEndpoints []`

Radio-specific Functions

- `EmberStatus emberSetTxPowerMode (uint16_t txPowerMode)`
- `uint16_t emberGetTxPowerMode (void)`
- `EmberStatus emberSetNodeId (EmberNodeId nodeId)`
- `void emberRadioNeedsCalibratingHandler (void)`
- `void emberCalibrateCurrentChannel (void)`

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 [26](#) of file [config.h](#).

6.2.2.7 #define EMBER_VERSION_TYPE

Definition at line [31](#) of file [config.h](#).

6.2.2.8 #define SOFTWARE_VERSION

Software version. High byte = release number, low byte = patch number.

Definition at line [36](#) 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_NETWORK_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 sub-optimal. 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

<i>status</i>	Stack status. One of the following: <ul style="list-style-type: none"> • 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 bool 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 bool emberIsLocalEui64 (EmberEUI64 *eui64*)

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

Parameters

<i>eui64</i>	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 (uint16_t *code*)

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

Parameters

<i>code</i>	The manufacturer code for the local node.
-------------	---

6.2.3.9 void emberSetPowerDescriptor (uint16_t *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

<i>descriptor</i>	The new power descriptor for the local node.
-------------------	--

6.2.3.10 void emberSetMaximumIncomingTransferSize (*uint16_t 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

<i>size</i>	The maximum incoming transfer size for the local node.
-------------	--

6.2.3.11 void emberSetMaximumOutgoingTransferSize (*uint16_t 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

<i>size</i>	The maximum outgoing transfer size for the local node.
-------------	--

6.2.3.12 void emberSetDescriptorCapability (*uint8_t capability*)

Sets the descriptor capability field of the node.

Parameters

<i>capability</i>	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

<i>parameters</i>	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

<i>result-Location</i>	A pointer to an <i>EmberNodeType</i> 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 (`uint8_t 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

<code>channel</code>	Desired radio channel.
----------------------	------------------------

Returns

An [EmberStatus](#) value indicating the success or failure of the command.

6.2.3.16 `uint8_t 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 (`int8_t 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

<code>power</code>	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 int8_t 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(uint8_t * resultLocation)

Fetches a node's 8 byte Extended PAN identifier. If this is called when a device is not currently on a network (see [emberNetworkState](#)), then the Extended PAN ID returned will be an invalid value.

6.2.3.22 uint8_t emberGetEndpoint(uint8_t index)

Retrieves the endpoint number for the index'th endpoint. `index` must be less than the value of `emberEndpointCount`.

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

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

Returns

The endpoint number for the index'th endpoint.

6.2.3.23 bool emberGetEndpointDescription (uint8_t *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

<i>endpoint</i>	The endpoint whose description is to be returned.
<i>result</i>	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 uint16_t emberGetEndpointCluster (uint8_t *endpoint*, EmberClusterListId *listId*, uint8_t *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

<i>endpoint</i>	The endpoint from which the cluster ID is to be read.
<i>listId</i>	The list from which the cluster ID is to be read.
<i>listIndex</i>	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 bool emberIsNodeIdValid (EmberNodeId *nodeId*)

Determines whether nodeId is valid.

Parameters

<i>nodeId</i>	A node ID.
---------------	------------

Returns

true if nodeId is valid, false otherwise.

6.2.3.26 EmberNodeId emberLookupNodIdByEui64 (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

<i>eui64</i>	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 emberLookupEui64ByNodId (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

<i>nodeId</i>	The short ID of the node to look up.
<i>eui64Return</i>	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*, uint8_t *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 CONFIGURATION_HEADER to use this. This function may be called in ISR context, so processing should be kept to a minimum.

Parameters

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

6.2.3.29 void emberStackTokenChangedHandler (uint16_t *tokenAddress*)

A callback invoked to inform the application that a stack token has changed.

Parameters

<i>tokenAddress</i>	The address of the stack token that has changed.
---------------------	--

6.2.3.30 EmberStatus emberGetNeighbor (uint8_t *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

<i>index</i>	The index of a neighbor table entry.
<i>result</i>	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.

6.2.3.31 EmberStatus emberGetRouteTableEntry (uint8_t *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

<i>index</i>	The index of a route table entry.
<i>result</i>	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_SUCCESS](#) otherwise.

6.2.3.32 uint8_t emberStackProfile (void)

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

Returns

stack profile

6.2.3.33 uint8_t emberTreeDepth (void)

Returns the depth of the node in the network.

Returns

current depth

6.2.3.34 uint8_t emberNeighborCount (void)

Returns the number of active entries in the neighbor table.

Returns

number of active entries in the neighbor table

6.2.3.35 uint8_t emberRouteTableSize (void)

Returns the size of the route table.

Returns

the size of the route table

6.2.3.36 uint8_t emberNextZigbeeSequenceNumber (void)

Increments and returns the ZigBee sequence number.

Returns

the next ZigBee sequence number

6.2.3.37 EmberStatus emberSetTxPowerMode (uint16_t 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 STANDALONE 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_CONFIG token appropriately.

Parameters

<i>txPowerMode</i>	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.

6.2.3.38 uint16_t emberGetTxPowerMode (void)

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

Returns

the current tx power mode.

6.2.3.39 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 INITIAL state.

Parameters

<i>nodeId</i>	Specifies the short ID to be assigned to the node.
---------------	--

Returns

[EMBER_SUCCESS](#) if successful; an error code otherwise.

6.2.3.40 void emberRadioNeedsCalibratingHandler (void)

The radio calibration callback function.

The Voltage Controlled Oscillator (VCO) can drift with temperature changes. During every call to [emberTick\(\)](#), 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.

6.2.3.41 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.

6.2.4 Variable Documentation

6.2.4.1 PGM uint8_t emberStackProfileId[]

The application must provide a definition for this variable.

6.2.4.2 uint8_t 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 `emberGetEndpoint()`, `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_HAS_GET_ENDPOINT in its CONFIGURATION_HEADER.

6.3 Ember Common Data Types

Data Structures

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

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

Macros

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

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

Typedefs

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

Enumerations

- enum EmberNodeType {
 EMBER_UNKNOWN_DEVICE, EMBER_COORDINATOR, EMBER_ROUTER,
 EMBER_END_DEVICE,
 EMBER_SLEEPY_END_DEVICE, EMBER_MOBILE_END_DEVICE, EMBER_RF4CE_TARGET, EMBER_RF4CE_CONTROLLER
 }
- enum EmberEndDeviceConfiguration { EMBER_END_DEVICE_CONFIG_NONE, EMBER_END_DEVICE_CONFIG_PERSIST_DATA_ON_PARENT }
- enum EmberNetworkInitBitmask { EMBER_NETWORK_INIT_NO_OPTIONS, EMBER_NETWORK_INIT_PARENT_INFO_IN_TOKEN }
- enum EmberApsOption {
 EMBER_APS_OPTION_NONE, EMBER_APS_OPTION_DSA_SIGN, EMBER_APS_OPTION_ENCRYPTION, EMBER_APS_OPTION_RETRY,
 EMBER_APS_OPTION_ENABLE_ROUTE_DISCOVERY, EMBER_APS_OPTION_FORCE_ROUTE_DISCOVERY, EMBER_APS_OPTION_SOURCE_EUI64,
 EMBER_APS_OPTION_DESTINATION_EUI64,
 EMBER_APS_OPTION_ENABLE_ADDRESS_DISCOVERY, EMBER_APS_OPTION_POLL_RESPONSE, EMBER_APS_OPTION_ZDO_RESPONSE_REQUIRED, EMBER_APS_OPTION_FRAGMENT
 }
- enum EmberIncomingMessageType {
 EMBER_INCOMING_UNICAST, EMBER_INCOMING_UNICAST_REPLY, EMBER_INCOMING_MULTICAST, EMBER_INCOMING_MULTICAST_LOOPBACK,
 EMBER_INCOMING_BROADCAST, EMBER_INCOMING_BROADCAST_LOOPBACK
 }

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

```

ST, EMBER_COUNTERAPS DATA TX BROADCAST,
EMBER_COUNTERAPS DATA RX UNICAST, EMBER_COUNTERAPS D-
ATA TX UNICAST SUCCESS, EMBER_COUNTERAPS DATA TX UNICA-
ST RETRY, EMBER_COUNTERAPS DATA TX UNICAST FAILED,
EMBER_COUNTERROUTE DISCOVERY INITIATED, EMBER_COUNTER-
_NEIGHBOR ADDED, EMBER_COUNTERNEIGHBOR REMOVED, EM-
BER_COUNTERNEIGHBOR STALE,
EMBER_COUNTERJOIN INDICATION, EMBER_COUNTERCHILD REMO-
VED, EMBER_COUNTERASH_OVERFLOW ERROR, EMBER_COUNTER-
ASH FRAMING ERROR,
EMBER_COUNTERASH_OVERRUN ERROR, EMBER_COUNTERNWK F-
RAME COUNTER FAILURE, EMBER_COUNTERAPS FRAME COUNTER-
_FAILURE, EMBER_COUNTERASH_XOFF,
EMBER_COUNTERAPS LINK KEY NOT AUTHORIZED, EMBER_COUN-
TER_NWK_DECRYPTION_FAILURE, EMBER_COUNTERAPS DECRYPTI-
ON_FAILURE, EMBER_COUNTERALLOCATE_PACKET_BUFFER FAILU-
RE,
EMBER_COUNTERRELAYED UNICAST, EMBER_COUNTERPHY TO M-
AC_QUEUE LIMIT REACHED, EMBER_COUNTERPACKET VALIDATE-
LIBRARY DROPPED COUNT, EMBER_COUNTERTYPE_NWK_RETRY_O-
VERFLOW,
EMBER_COUNTERPHYCCA FAIL COUNT, EMBER_COUNTERBROAD-
CAST_TABLE_FULL, EMBER_COUNTERTYPE COUNT }

• enum EmberInitialSecurityBitmask {
  EMBER_DISTRIBUTED_TRUST_CENTER_MODE, EMBER_TRUST_CEN-
TE_R_GLOBAL_LINK_KEY, EMBER_PRECONFIGURED_NETWORK_KEY_MO-
DE, EMBER_HAVE_TRUST_CENTER_EUI64,
EMBER_TRUST_CENTERUSES HASHED_LINK_KEY, EMBER_HAVE_PR-
ECONFIGURED_KEY, EMBER_HAVE_NETWORK_KEY, EMBER_GET LIN-
K_KEY WHEN JOINING,
EMBER_REQUIRE_ENCRYPTED_KEY, EMBER_NO_FRAME COUNTER_R-
ESET, EMBER_GET PRECONFIGURED_KEY FROM INSTALL_CODE }

• enum EmberExtendedSecurityBitmask { EMBER_JOINERGLOBALLINK KE-
Y, EMBER_EXT_NO_FRAME COUNTER_RESET, EMBER_NWK_LEAVE_R-
EQUEST NOT ALLOWED }

• enum EmberCurrentSecurityBitmask {
  EMBER_STANDARD_SECURITY_MODE_, EMBER_DISTRIBUTED_TRUST-
_CENTER_MODE_, EMBER_TRUST_CENTERGLOBALLINKKEY_, EM-
BER_HAVE_TRUST_CENTERLINKKEY,
EMBER_TRUST_CENTERUSES HASHEDLINKKEY_ }

• enum EmberKeyStructBitmask {
  EMBER_KEY HAS_SEQUENCE_NUMBER, EMBER_KEY HAS_OUTGOING-
_FRAME COUNTER, EMBER_KEY HAS_INCOMING_FRAME COUNTER, E-
MBER_KEY HAS_PARTNER_EUI64,
EMBER_KEY IS AUTHORIZED, EMBER_KEY PARTNER_IS_SLEEPY }

• enum EmberKeyType {
  EMBER_TRUST_CENTERLINKKEY, EMBER_TRUST_CENTERMASTER-
KEY, EMBER_CURRENTNETWORKKEY, EMBER_NEXTNETWORK_K-
EY,
EMBER_APPLICATIONLINKKEY, EMBER_APPLICATIONMASTERKE-
Y }

• enum EmberKeyStatus {
  EMBER_KEY_STATUS_NONE, EMBER_APPLINKKEY ESTABLISHED, E-
}

```

```

MBER_APP_MASTER_KEY_ESTABLISHED, EMBER_TRUST_CENTER_LINK_KEY_ESTABLISHED,
EMBER_KEY_ESTABLISHMENT_TIMEOUT, EMBER_KEY_TABLE_FULL, EMBER_TC_RESPONDED_TO_KEY_REQUEST, EMBER_TC_APP_KEY_SENT_TO_REQUESTER,
EMBER_TC_RESPONSE_TO_KEY_REQUEST_FAILED, EMBER_TC_REQUEST_KEY_TYPE_NOT_SUPPORTED, EMBER_TC_NO_LINK_KEY_FOR_REQUESTER, EMBER_TC_REQUESTER_EUI64_UNKNOWN,
EMBER_TC RECEIVED FIRST APP KEY REQUEST, EMBER_TC TIMEOUT_WAITING_FOR_SECOND_APP_KEY_REQUEST, EMBER_TC_NON_MATCHING_APP_KEY_REQUEST_RECEIVED, EMBER_TC FAILED_TO_SEND_APP_KEYS,
EMBER_TC FAILED_TO_STORE_APP_KEY_REQUEST, EMBER_TC REJECTED_APP_KEY_REQUEST, EMBER_TC FAILED_TO_GENERATE_NEW_KEY, EMBER_TC FAILED_TO_SEND_TC_KEY,
EMBER_TRUST_CENTER_IS_PRE_R21, EMBER_TC_REQUESTER_VERIFY_KEY_TIMEOUT, EMBER_TC_REQUESTER_VERIFY_KEY_FAILURE, EMBER_TC_REQUESTER_VERIFY_KEY_SUCCESS,
EMBER_VERIFY_LINK_KEY_FAILURE, EMBER_VERIFY_LINK_KEY_SUCCESS }

• enum EmberLinkKeyRequestPolicy { EMBER_DENY_KEY_REQUESTS, EMBER_ALLOW_KEY_REQUESTS, EMBER_GENERATE_NEW_TC_LINK_KEY }

• enum EmberKeySettings { EMBER_KEY_PERMISSIONS_NONE, EMBER_KEY_PERMISSIONS_READING_ALLOWED, EMBER_KEY_PERMISSIONS_HASHING_ALLOWED }

• enum EmberMacPassthroughType {
EMBER_MAC_PASSTHROUGH_NONE, EMBER_MAC_PASSTHROUGH_SEINTERPAN, EMBER_MAC_PASSTHROUGH_EMBERNET, EMBER_MAC_PASSTHROUGH_EMBERNET_SOURCE,
EMBER_MAC_PASSTHROUGH_APPLICATION, EMBER_MAC_PASSTHROUGH_CUSTOM }

```

Functions

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

Miscellaneous Ember Types

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

```

A_3,
EMBER_VERSION_TYPE_BETA_1, EMBER_VERSION_TYPE_BETA_2, EM-
BER_VERSION_TYPE_BETA_3, EMBER_VERSION_TYPE_GA }
• enum EmberLeaveRequestFlags { EMBER_ZIGBEE_LEAVE_AND_REJOIN, E-
MBER_ZIGBEE_LEAVE_AND_REMOVE_CHILDREN }
• enum EmberLeaveReason {
EMBER_LEAVE_REASON_NONE, EMBER_LEAVE_DUE_TO_NWK_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_A-
PP_EVENT_1 }
• typedef uint8_t EmberStatus
• typedef uint8_t EmberEUI64 [EUI64_SIZE]
• typedef uint8_t EmberMessageBuffer
• typedef uint16_t EmberNodeId
• typedef uint16_t EmberMulticastId
• typedef uint16_t EmberPanId
• const EmberVersion emberVersion
• #define EMBER_RELEASE_TYPE_TO_STRING_STRUCT_DATA
• #define EUI64_SIZE
• #define EXTENDED_PAN_ID_SIZE
• #define EMBER_ENCRYPTION_KEY_SIZE
• #define EMBER_CERTIFICATE_SIZE
• #define EMBER_PUBLIC_KEY_SIZE
• #define EMBER_PRIVATE_KEY_SIZE
• #define EMBER_SMAC_SIZE
• #define EMBER_SIGNATURE_SIZE
• #define EMBER_AES_HASH_BLOCK_SIZE
• #define EMBER_CERTIFICATE_283K1_SIZE
• #define EMBER_PUBLIC_KEY_283K1_SIZE
• #define EMBER_PRIVATE_KEY_283K1_SIZE
• #define EMBER_SIGNATURE_283K1_SIZE
• #define __EMBERSTATUS_TYPE__
• #define EMBER_MAX_802_15_4_CHANNEL_NUMBER
• #define EMBER_MIN_802_15_4_CHANNEL_NUMBER
• #define EMBER_NUM_802_15_4_CHANNELS
• #define EMBER_ALL_802_15_4_CHANNELS_MASK
• #define EMBER_ZIGBEE_COORDINATOR_ADDRESS
• #define EMBER_NULL_NODE_ID
• #define EMBER_NULL_BINDING
• #define EMBER_TABLE_ENTRY_UNUSED_NODE_ID
• #define EMBER_MULTICAST_NODE_ID
• #define EMBER_UNKNOWN_NODE_ID
• #define EMBER_DISCOVERY_ACTIVE_NODE_ID
• #define EMBER_NULL_ADDRESS_TABLE_INDEX
• #define EMBER_ZDO_ENDPOINT
• #define EMBER_BROADCAST_ENDPOINT
• #define EMBER_ZDO_PROFILE_ID
• #define EMBER_WILDCARD_PROFILE_ID
• #define EMBER_MAXIMUM_STANDARD_PROFILE_ID
• #define EMBER_BROADCAST_TABLE_TIMEOUT_QS
• #define EMBER_MANUFACTURER_ID

```

ZigBee Broadcast Addresses

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

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

Ember Concentrator Types

- #define EMBER_LOW_RAM_CONCENTRATOR
- #define EMBER_HIGH_RAM_CONCENTRATOR

txPowerModes for emberSetTxPowerMode and mfplibSetPower

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

Alarm Message and Counters Request Definitions

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

ZDO response status.

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

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

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

Network and IEEE Address Request/Response

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

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

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

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

Node Descriptor Request/Response

```
<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_DESCRIPTOR_REQUEST
- #define MATCH_DESCRIPTOR_RESPONSE

Discovery Cache Request / Response

```

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

```

- #define DISCOVERY_CACHE_REQUEST
- #define DISCOVERY_CACHE_RESPONSE

End Device Announce and End Device Announce Response

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

- #define END_DEVICE_ANNOUNCE
- #define END_DEVICE_ANNOUNCE_RESPONSE

System Server Discovery Request / Response

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

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

- #define SYSTEM_SERVER_DISCOVERY_REQUEST
- #define SYSTEM_SERVER_DISCOVERY_RESPONSE

Parent Announce and Parent Announce Response

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

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

- #define PARENT_ANNOUNCE
- #define PARENT_ANNOUNCE_RESPONSE

ZDO server mask bits

These are used in server discovery requests and responses.

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

Find Node Cache Request / Response

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

```

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

```

- #define FIND_NODE_CACHE_REQUEST
- #define FIND_NODE_CACHE_RESPONSE

End Device Bind Request / Response

```

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

```

- #define END_DEVICE_BIND_REQUEST
- #define END_DEVICE_BIND_RESPONSE

Binding types and Request / Response

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

```

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

```

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

LQI Table Request / Response

```

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

```

The device-type byte has the following fields:

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

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

The permit-joining byte has the following fields

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

- #define LQI_TABLE_REQUEST
- #define LQI_TABLE_RESPONSE

Routing Table Request / Response

```

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

```

The status byte has the following fields:

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

- #define ROUTING_TABLE_REQUEST
- #define ROUTING_TABLE_RESPONSE

Binding Table Request / Response

```

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

```

Note

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

- #define BINDING_TABLE_REQUEST
- #define BINDING_TABLE_RESPONSE

Leave Request / Response

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

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

Permit Joining Request / Response

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

- #define PERMIT_JOINING_REQUEST
- #define PERMIT_JOINING_RESPONSE

Network Update Request / Response

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

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

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

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

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

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

- #define NWK_UPDATE_REQUEST
- #define NWK_UPDATE_RESPONSE

Unsupported

Not mandatory and not supported.

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

ZDO configuration flags.

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

`EMBER_APPLICATION RECEIVES_SUPPORTED_ZDO_REQUESTS EMBER_APPLICATION_HANDLES_UNSUPPORTED_ZDO_REQUESTS EMBER_APPLICATION_HANDLES_ENDPOINT_ZDO_REQUESTS EMBER_APPLICATION_HANDLES_BINDING_ZDO_REQUESTS`

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

- enum `EmberZdoConfigurationFlags` { `EMBER_APP RECEIVES_SUPPORTED_ZDO_REQUESTS, EMBER_APP_HANDLES_UNSUPPORTED_ZDO_REQUESTS, EMBER_APP_HANDLES_ENDPOINT_ZDO_REQUESTS, EMBER_APP_HANDLES_BINDING_ZDO_REQUESTS` }

6.3.1 Detailed Description

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

6.3.2 Macro Definition Documentation

6.3.2.1 #define EMBER_RELEASE_TYPE_TO_STRING_STRUCT_DATA

`EmberReleaseTypeStruct` Data that relates release type to the correct string.

Definition at line 75 of file `ember-types.h`.

6.3.2.2 #define EUI64_SIZE

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

Definition at line 107 of file `ember-types.h`.

6.3.2.3 #define EXTENDED_PAN_ID_SIZE

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

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

6.3.2.4 #define EMBER_ENCRYPTION_KEY_SIZE

Size of an encryption key in bytes (16).

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

6.3.2.5 #define EMBER_CERTIFICATE_SIZE

Size of Implicit Certificates used for Certificate Based Key Exchange.

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

6.3.2.6 #define EMBER_PUBLIC_KEY_SIZE

Size of Public Keys used in Elliptical Cryptography ECMQV algorithms.

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

6.3.2.7 #define EMBER_PRIVATE_KEY_SIZE

Size of Private Keys used in Elliptical Cryptography ECMQV algorithms.

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

6.3.2.8 #define EMBER_SMAC_SIZE

Size of the SMAC used in Elliptical Cryptography ECMQV algorithms.

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

6.3.2.9 #define EMBER_SIGNATURE_SIZE

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

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

6.3.2.10 #define EMBER_AES_HASH_BLOCK_SIZE

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

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

6.3.2.11 #define EMBER_CERTIFICATE_283K1_SIZE

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

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

6.3.2.12 #define EMBER_PUBLIC_KEY_283K1_SIZE

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

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

6.3.2.13 #define EMBER_PRIVATE_KEY_283K1_SIZE

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

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

6.3.2.14 #define EMBER_SIGNATURE_283K1_SIZE

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

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

6.3.2.15 #define __EMBERSTATUS_TYPE__

Return type for Ember functions.

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

6.3.2.16 #define EMBER_MAX_802_15_4_CHANNEL_NUMBER

The maximum 802.15.4 channel number is 26.

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

6.3.2.17 #define EMBER_MIN_802_15_4_CHANNEL_NUMBER

The minimum 802.15.4 channel number is 11.

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

6.3.2.18 #define EMBER_NUM_802_15_4_CHANNELS

There are sixteen 802.15.4 channels.

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

6.3.2.19 #define EMBER_ALL_802_15_4_CHANNELS_MASK

Bitmask to scan all 802.15.4 channels.

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

6.3.2.20 #define EMBER_ZIGBEE_COORDINATOR_ADDRESS

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

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

6.3.2.21 #define EMBER_NULL_NODE_ID

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

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

6.3.2.22 #define EMBER_NULL_BINDING

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

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

6.3.2.23 #define EMBER_TABLE_ENTRY_UNUSED_NODE_ID

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

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

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

6.3.2.24 #define EMBER_MULTICAST_NODE_ID

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

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

6.3.2.25 #define EMBER_UNKNOWN_NODE_ID

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

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

6.3.2.26 #define EMBER_DISCOVERY_ACTIVE_NODE_ID

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

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

6.3.2.27 #define EMBER_NULL_ADDRESS_TABLE_INDEX

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

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

6.3.2.28 #define EMBER_ZDO_ENDPOINT

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

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

6.3.2.29 #define EMBER_BROADCAST_ENDPOINT

The broadcast endpoint, as defined in the ZigBee spec.

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

6.3.2.30 #define EMBER_ZDO_PROFILE_ID

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

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

6.3.2.31 #define EMBER_WILDCARD_PROFILE_ID

The profile ID used to address all the public profiles.

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

6.3.2.32 #define EMBER_MAXIMUM_STANDARD_PROFILE_ID

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

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

6.3.2.33 #define EMBER_BROADCAST_TABLE_TIMEOUT_QS

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

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

6.3.2.34 #define EMBER_MANUFACTURER_ID

Ember's Manufacturer ID.

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

6.3.2.35 #define EMBER_BROADCAST_ADDRESS

Broadcast to all routers.

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

6.3.2.36 #define EMBER_RX_ON_WHEN_IDLE_BROADCAST_ADDRESS

Broadcast to all non-sleepy devices.

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

6.3.2.37 #define EMBER_SLEEPY_BROADCAST_ADDRESS

Broadcast to all devices, including sleepy end devices.

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

6.3.2.38 #define EMBER_MIN_BROADCAST_ADDRESS

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

6.3.2.39 #define emberIsZigbeeBroadcastAddress(address)

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

6.3.2.40 #define EMBER_LOW_RAM_CONCENTRATOR

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

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

6.3.2.41 #define EMBER_HIGH_RAM_CONCENTRATOR

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

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

6.3.2.42 #define EMBER_JOIN_DECISION_STRINGS

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

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

6.3.2.43 #define EMBER_DEVICE_UPDATE_STRINGS

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

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

6.3.2.44 #define emberInitializeNetworkParameters(*parameters*)

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

6.3.2.45 #define EMBER_COUNTER_STRINGS

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

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

6.3.2.46 #define EMBER_TX_POWER_MODE_DEFAULT

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

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

6.3.2.47 #define EMBER_TX_POWER_MODE_BOOST

The application should call [emberSetTxPowerMode\(\)](#) with the txPowerMode parameter set to this value to enable boost power mode.

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

6.3.2.48 #define EMBER_TX_POWER_MODE_ALTERNATE

The application should call [emberSetTxPowerMode\(\)](#) with the txPowerMode parameter set to this value to enable the alternate transmitter output.

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

6.3.2.49 #define EMBER_TX_POWER_MODE_BOOST_AND_ALTERNATE

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

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

6.3.2.50 #define EMBER_PRIVATE_PROFILE_ID

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

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

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

6.3.2.51 #define EMBER_PRIVATE_PROFILE_ID_START

Ember's first private profile ID.

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

6.3.2.52 #define EMBER_PRIVATE_PROFILE_ID_END

Ember's last private profile ID.

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

6.3.2.53 #define EMBER_BROADCAST_ALARM_CLUSTER

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

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

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

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

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

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

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

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

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

6.3.2.54 #define EMBER_UNICAST_ALARM_CLUSTER

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

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

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

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

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

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

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

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

6.3.2.55 #define EMBER_CACHED_UNICAST_ALARM_CLUSTER

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

1. flags
2. priority
3. payload

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

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

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

6.3.2.56 #define EMBER_REPORT_COUNTERS_REQUEST

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

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

6.3.2.57 #define EMBER_REPORT_COUNTERS_RESPONSE

The cluster id used to respond to an EMBER_REPORT_COUNTERS_REQUEST.

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

6.3.2.58 #define EMBER_REPORT_AND_CLEAR_COUNTERS_REQUEST

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

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

6.3.2.59 #define EMBER_REPORT_AND_CLEAR_COUNTERS_RESPONSE

The cluster id used to respond to an EMBER_REPORT_AND_CLEAR_COUNTERS_REQUEST.

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

6.3.2.60 #define EMBER_OTA_CERTIFICATE_UPGRADE_CLUSTER

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

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

6.3.2.61 #define EMBER_STANDARD_SECURITY_MODE

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

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

6.3.2.62 #define EMBER_TRUST_CENTER_NODE_ID

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

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

6.3.2.63 #define EMBER_NO_TRUST_CENTER_MODE

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

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

6.3.2.64 #define EMBER_GLOBAL_LINK_KEY

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

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

6.3.2.65 #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 2138 of file [ember-types.h](#).

6.3.2.66 #define EMBER_MAC_FILTER_MATCH_ENABLED_MASK

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

6.3.2.67 #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_MASK

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

6.3.2.68 #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_MASK

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

6.3.2.69 #define EMBER_MAC_FILTER_MATCH_ON_DEST_MASK

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

6.3.2.70 #define EMBER_MAC_FILTER_MATCH_ON_SOURCE_MASK

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

6.3.2.71 #define EMBER_MAC_FILTER_MATCH_ENABLED

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

6.3.2.72 #define EMBER_MAC_FILTER_MATCH_DISABLED

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

6.3.2.73 #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_NONE

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

6.3.2.74 #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_LOCAL

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

6.3.2.75 #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_BROADCAST

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

6.3.2.76 #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_NONE

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

6.3.2.77 #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_NON_LOCAL

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

6.3.2.78 #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_LOCAL

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

6.3.2.79 #define EMBER_MAC_FILTER_MATCH_ON_DEST_BROADCAST_SHORT

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

6.3.2.80 #define EMBER_MAC_FILTER_MATCH_ON_DEST_UNICAST_SHORT

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

6.3.2.81 #define EMBER_MAC_FILTER_MATCH_ON_DEST_UNICAST_LONG

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

6.3.2.82 #define EMBER_MAC_FILTER_MATCH_ON_SOURCE_LONG

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

6.3.2.83 #define EMBER_MAC_FILTER_MATCH_ON_SOURCE_SHORT

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

6.3.2.84 #define EMBER_MAC_FILTER_MATCH_ON_SOURCE_NONE

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

6.3.2.85 #define EMBER_MAC_FILTER_MATCH_END

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

6.3.2.86 #define NETWORK_ADDRESS_REQUEST

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

6.3.2.87 #define NETWORK_ADDRESS_RESPONSE

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

6.3.2.88 #define IEEE_ADDRESS_REQUEST

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

6.3.2.89 #define IEEE_ADDRESS_RESPONSE

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

6.3.2.90 #define NODE_DESCRIPTOR_REQUEST

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

6.3.2.91 #define NODE_DESCRIPTOR_RESPONSE

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

6.3.2.92 #define POWER_DESCRIPTOR_REQUEST

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

6.3.2.93 #define POWER_DESCRIPTOR_RESPONSE

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

6.3.2.94 #define SIMPLE_DESCRIPTOR_REQUEST

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

6.3.2.95 #define SIMPLE_DESCRIPTOR_RESPONSE

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

6.3.2.96 #define ACTIVE_ENDPOINTS_REQUEST

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

6.3.2.97 #define ACTIVE_ENDPOINTS_RESPONSE

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

6.3.2.98 #define MATCH_DESCRIPTOR_REQUEST

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

6.3.2.99 #define MATCH_DESCRIPTOR_RESPONSE

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

6.3.2.100 #define DISCOVERY_CACHE_REQUEST

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

6.3.2.101 #define DISCOVERY_CACHE_RESPONSE

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

6.3.2.102 #define END_DEVICE_ANNOUNCE

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

6.3.2.103 #define END_DEVICE_ANNOUNCE_RESPONSE

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

6.3.2.104 #define SYSTEM_SERVER_DISCOVERY_REQUEST

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

6.3.2.105 #define SYSTEM_SERVER_DISCOVERY_RESPONSE

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

6.3.2.106 #define PARENT_ANNOUNCE

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

6.3.2.107 #define PARENT_ANNOUNCE_RESPONSE

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

6.3.2.108 #define FIND_NODE_CACHE_REQUEST

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

6.3.2.109 #define FIND_NODE_CACHE_RESPONSE

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

6.3.2.110 #define END_DEVICE_BIND_REQUEST

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

6.3.2.111 #define END_DEVICE_BIND_RESPONSE

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

6.3.2.112 #define UNICAST_BINDING

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

6.3.2.113 #define UNICAST_MANY_TO_ONE_BINDING

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

6.3.2.114 #define MULTICAST_BINDING

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

6.3.2.115 #define BIND_REQUEST

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

6.3.2.116 #define BIND_RESPONSE

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

6.3.2.117 #define UNBIND_REQUEST

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

6.3.2.118 #define UNBIND_RESPONSE

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

6.3.2.119 #define LQI_TABLE_REQUEST

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

6.3.2.120 #define LQI_TABLE_RESPONSE

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

6.3.2.121 #define ROUTING_TABLE_REQUEST

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

6.3.2.122 #define ROUTING_TABLE_RESPONSE

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

6.3.2.123 #define BINDING_TABLE_REQUEST

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

6.3.2.124 #define BINDING_TABLE_RESPONSE

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

6.3.2.125 #define LEAVE_REQUEST

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

6.3.2.126 #define LEAVE_RESPONSE

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

6.3.2.127 #define LEAVE_REQUEST_REMOVE_CHILDREN_FLAG

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

6.3.2.128 #define LEAVE_REQUEST_REJOIN_FLAG

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

6.3.2.129 #define PERMIT_JOINING_REQUEST

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

6.3.2.130 #define PERMIT_JOINING_RESPONSE

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

6.3.2.131 #define NWK_UPDATE_REQUEST

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

6.3.2.132 #define NWK_UPDATE_RESPONSE

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

6.3.2.133 #define COMPLEX_DESCRIPTOR_REQUEST

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

6.3.2.134 #define COMPLEX_DESCRIPTOR_RESPONSE

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

6.3.2.135 #define USER_DESCRIPTOR_REQUEST

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

6.3.2.136 #define USER_DESCRIPTOR_RESPONSE

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

6.3.2.137 #define DISCOVERY_REGISTER_REQUEST

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

6.3.2.138 #define DISCOVERY_REGISTER_RESPONSE

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

6.3.2.139 #define USER_DESCRIPTOR_SET

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

6.3.2.140 #define USER_DESCRIPTOR_CONFIRM

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

6.3.2.141 #define NETWORK_DISCOVERY_REQUEST

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

6.3.2.142 #define NETWORK_DISCOVERY_RESPONSE

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

6.3.2.143 #define DIRECT_JOIN_REQUEST

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

6.3.2.144 #define DIRECT_JOIN_RESPONSE

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

6.3.2.145 #define CLUSTER_ID_RESPONSE_MINIMUM

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

6.3.2.146 #define WEAK_TEST

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

6.3.3 Typedef Documentation

6.3.3.1 typedef uint8_t EmberStatus

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

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

6.3.3.2 typedef uint8_t EmberEUI64[EUI64_SIZE]

EUI 64-bit ID (an IEEE address).

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

6.3.3.3 typedef uint8_t EmberMessageBuffer

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

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

See [packet-buffer.h](#) for APIs related to stack and linked buffers.

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

6.3.3.4 `typedef uint16_t EmberNodeId`

16-bit ZigBee network address.

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

6.3.3.5 `typedef uint16_t EmberMulticastId`

16-bit ZigBee multicast group identifier.

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

6.3.3.6 `typedef uint16_t EmberPanId`

802.15.4 PAN ID.

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

6.3.3.7 `typedef uint8_t EmberTaskId`

brief An identifier for a task

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

6.3.3.8 `typedef PGM struct EmberEventData_S EmberEventData`**6.3.3.9 `typedef uint16_t EmberMacFilterMatchData`**

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

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

6.3.3.10 `typedef uint8_t EmberLibraryStatus`

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

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

6.3.4 Enumeration Type Documentation**6.3.4.1 `enum EmberVersionType`**

Type of Ember software version.

Enumerator:

EMBER_VERSION_TYPE_PRE_RELEASE

EMBER_VERSION_TYPE_ALPHA_1

EMBER_VERSION_TYPE_ALPHA_2

EMBER_VERSION_TYPE_ALPHA_3

EMBER_VERSION_TYPE_BETA_1
EMBER_VERSION_TYPE_BETA_2
EMBER_VERSION_TYPE_BETA_3
EMBER_VERSION_TYPE_GA

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

6.3.4.2 enum EmberLeaveRequestFlags

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

Enumerator:

EMBER_ZIGBEE_LEAVE_AND_REJOIN Leave and rejoin
EMBER_ZIGBEE_LEAVE_AND_REMOVE_CHILDREN Send all children leave command

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

6.3.4.3 enum EmberLeaveReason

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

Enumerator:

EMBER_LEAVE_REASON_NONE
EMBER_LEAVE_DUE_TO_NWK_LEAVE_MESSAGE
EMBER_LEAVE_DUE_TO_APS_REMOVE_MESSAGE
EMBER_LEAVE_DUE_TO_ZDO_LEAVE_MESSAGE
EMBER_LEAVE_DUE_TO_ZLL_TOUCHLINK
EMBER_LEAVE_DUE_TO_APP_EVENT_I

Definition at line 347 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.

EMBER_RF4CE_TARGET RF4CE target node.

EMBER_RF4CE_CONTROLLER RF4CE controller node.

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

6.3.4.5 enum EmberEndDeviceConfiguration

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

Enumerator:

EMBER_END_DEVICE_CONFIG_NONE

EMBER_END_DEVICE_CONFIG_PERSIST_DATA_ON_PARENT

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

6.3.4.6 enum EmberNetworkInitBitmask

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

Enumerator:

EMBER_NETWORK_INIT_NO_OPTIONS

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

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

6.3.4.7 enum EmberApsOption

Options to use when sending a message.

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

Enumerator:

EMBER_APS_OPTION_NONE No options.

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

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

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

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

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

EMBER_APS_OPTION_ZDO_RESPONSE_REQUIRED This incoming message is a valid ZDO request and the application is responsible for sending a ZDO response. This flag is used only within [emberIncomingMessageHandler\(\)](#) when `EMBER_APPLICATION RECEIVES UNSUPPORTED_ZDO_REQUESTS` is defined.

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

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

6.3.4.8 enum EmberIncomingMessageType

Defines the possible incoming message types.

Enumerator:

EMBER_INCOMING_UNICAST Unicast.

EMBER_INCOMING_UNICAST_REPLY Unicast reply.

EMBER_INCOMING_MULTICAST Multicast.

EMBER_INCOMING_MULTICAST_LOOPBACK Multicast sent by the local device.

EMBER_INCOMING_BROADCAST Broadcast.

EMBER_INCOMING_BROADCAST_LOOPBACK Broadcast sent by the local device.

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

6.3.4.9 enum EmberOutgoingMessageType

Defines the possible outgoing message types.

Enumerator:

EMBER_OUTGOING_DIRECT Unicast sent directly to an EmberNodeId.

EMBER_OUTGOING_VIA_ADDRESS_TABLE Unicast sent using an entry in the address table.

EMBER_OUTGOING_VIA_BINDING Unicast sent using an entry in the binding table.

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

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

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

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

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

6.3.4.10 enum EmberZigbeeCommandType

A type of command received by the stack.

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

Enumerator:

EMBER_ZIGBEE_COMMAND_TYPE_MAC Describes an 802.15.4 MAC layer command.

EMBER_ZIGBEE_COMMAND_TYPE_NWK Describes a ZigBee Network layer command.

EMBER_ZIGBEE_COMMAND_TYPE_APS Describes a ZigBee Application Support layer command.

EMBER_ZIGBEE_COMMAND_TYPE_ZDO Describes a ZigBee Device Object command.

EMBER_ZIGBEE_COMMAND_TYPE_ZCL Describes a ZigBee Cluster Library command.

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

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

6.3.4.11 enum EmberNetworkStatus

Defines the possible join states for a node.

Enumerator:

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

EMBER_JOINING_NETWORK The node is currently attempting to join a network.

EMBER_JOINED_NETWORK The node is joined to a network.

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

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

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

6.3.4.12 enum EmberNetworkScanType

Type for a network scan.

Enumerator:

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

EMBER_ACTIVE_SCAN An active scan scans each channel for available networks.

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

6.3.4.13 enum EmberBindingType

Defines binding types.

Enumerator:

EMBER_UNUSED_BINDING A binding that is currently not in use.

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

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

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

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

6.3.4.14 enum EmberJoinDecision

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

Enumerator:

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

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

6.3.4.15 enum EmberDeviceUpdate

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

Enumerator:

- EMBER_STANDARD_SECURITY_SECURED_REJOIN***
- EMBER_STANDARD_SECURITY_UNSECURED_JOIN***
- EMBER_DEVICE_LEFT***
- EMBER_STANDARD_SECURITY_UNSECURED_REJOIN***
- EMBER_HIGH_SECURITY_SECURED_REJOIN***
- EMBER_HIGH_SECURITY_UNSECURED_JOIN***
- EMBER_HIGH_SECURITY_UNSECURED_REJOIN***

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

6.3.4.16 enum EmberRejoinReason

Notes the last rejoin reason.

Enumerator:

- EMBER_REJOIN_REASON_NONE***
- EMBER_REJOIN_DUE_TO_NWK_KEY_UPDATE***
- EMBER_REJOIN_DUE_TO_LEAVE_MESSAGE***
- EMBER_REJOIN_DUE_TO_NO_PARENT***
- EMBER_REJOIN_DUE_TO_ZLL_TOUCHLINK***
- EMBER_REJOIN_DUE_TO_APP_EVENT_5***
- EMBER_REJOIN_DUE_TO_APP_EVENT_4***
- EMBER_REJOIN_DUE_TO_APP_EVENT_3***
- EMBER_REJOIN_DUE_TO_APP_EVENT_2***
- EMBER_REJOIN_DUE_TO_APP_EVENT_1***

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

6.3.4.17 enum EmberClusterListId

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

Enumerator:

EMBER_INPUT_CLUSTER_LIST Input clusters the endpoint will accept.

EMBER_OUTPUT_CLUSTER_LIST Output clusters the endpoint can send.

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

6.3.4.18 enum EmberEventUnits

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

Enumerator:

EMBER_EVENT_INACTIVE The event is not scheduled to run.

EMBER_EVENT_MS_TIME The execution time is in approximate milliseconds.

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

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

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

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

6.3.4.19 enum EmberJoinMethod

The type of method used for joining.

Enumerator:

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

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

EMBER_USE_NWK_REJOIN_HAVE_NWK_KEY

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

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

6.3.4.20 enum EmberCounterType

Defines the events reported to the application by the [emberCounterHandler\(\)](#).

Enumerator:

EMBER_COUNTER_MAC_RX_BROADCAST The MAC received a broadcast.

EMBER_COUNTER_MAC_TX_BROADCAST The MAC transmitted a broadcast.

EMBER_COUNTER_MAC_RX_UNICAST The MAC received a unicast.

EMBER_COUNTER_MAC_TX_UNICAST_SUCCESS The MAC successfully transmitted a unicast.

EMBER_COUNTER_MAC_TX_UNICAST_RETRY The MAC retried a unicast.

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

EMBER_COUNTER_MAC_TX_UNICAST_FAILED The MAC unsuccessfully transmitted a unicast.

EMBER_COUNTERAPS_DATA_RX_BROADCAST The APS layer received a data broadcast.

EMBER_COUNTERAPS_DATA_TX_BROADCAST The APS layer transmitted a data broadcast.

EMBER_COUNTERAPS_DATA_RX_UNICAST The APS layer received a data unicast.

EMBER_COUNTERAPS_DATA_TX_UNICAST_SUCCESS The APS layer successfully transmitted a data unicast.

EMBER_COUNTERAPS_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_COUNTERAPS_DATA_TX_UNICAST_SUCCESS](#) and [EMBER_COUNTERAPS_DATA_TX_UNICAST_FAILED](#) types.

EMBER_COUNTERAPS_DATA_TX_UNICAST_FAILED The APS layer unsuccessfully transmitted a data unicast.

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

EMBER_COUNTERNEIGHBOR_ADDED An entry was added to the neighbor table.

EMBER_COUNTERNEIGHBOR_REMOVED An entry was removed from the neighbor table.

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

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

EMBER_COUNTERCHILD_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_COUNTERAPS_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_COUNTERAPS_LINK_KEY_NOTAUTHORIZED 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_COUNTERAPS_DECRYPTION_FAILURE An APS encrypted message was received but dropped because decryption failed.

EMBER_COUNTER_ALLOCATE_PACKET_BUFFER_FAILURE The number of times we failed to allocate a set of linked packet buffers. This doesn't necessarily mean that the packet buffer count was 0 at the time, but that the number requested was greater than the number free.

EMBER_COUNTER_RELAYED_UNICAST The number of relayed unicast packets.

EMBER_COUNTER_PHY_TO_MAC_QUEUE_LIMIT_REACHED The number of times we dropped a packet due to reaching the preset PHY to MAC queue limit (emMaxPhyToMacQueueLength). The limit will determine how many messages are accepted by the PHY between calls to [emberTick\(\)](#). After that limit is hit, packets will be dropped. The number of dropped packets will be recorded in this counter.

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

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

EMBER_COUNTER_TYPE_NWK_RETRY_OVERFLOW The number of times the NWK retry queue is full and a new message failed to be added.

EMBER_COUNTERPHY_CCA_FAIL_COUNT The number of times the PHY layer was unable to transmit due to a failed CCA

EMBER_COUNTER_BROADCAST_TABLE_FULL The number of times a NWK broadcast was dropped because the broadcast table was full.

EMBER_COUNTER_TYPE_COUNT A placeholder giving the number of Ember counter types.

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

6.3.4.21 enum EmberInitialSecurityBitmask

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

Enumerator:

EMBER_DISTRIBUTED_TRUST_CENTER_MODE This enables Distributed Trust Center Mode for the device forming the network. (Previously known as [EMBER_NO_TRUST_CENTER_MODE](#))

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

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

EMBER_HAVE_TRUST_CENTER_EUI64 This denotes that the [EmberInitialSecurityState::preconfiguredTrustCenterEui64](#) has a value in it containing the trust center EUI64. The device will only join a network and accept commands from a trust center with that EUI64. Normally this bit is NOT set, and the EUI64 of the trust center is learned during the join process. When commissioning a device to join onto an existing network that is using a trust center, and without sending any messages, this bit must be set and the field [EmberInitialSecurityState::preconfiguredTrustCenterEui64](#) must be populated with the appropriate EUI64.

EMBER_TRUST_CENTERUSES_HASHED_LINK_KEY This denotes that the [EmberInitialSecurityState::preconfiguredKey](#) is not the actual Link Key but a Root Key known only to the Trust Center. It is hashed with the IEEE Address of the destination device in order to create the actual Link Key used in encryption. This is bit is only used by the Trust Center. The joining device need not set this.

EMBER_HAVE_PRECONFIGURED_KEY This denotes that the [EmberInitialSecurityState::preconfiguredKey](#) element has valid data that should be used to configure the initial security state.

EMBER_HAVE_NETWORK_KEY This denotes that the [EmberInitialSecurityState::networkKey](#) element has valid data that should be used to configure the initial security state.

EMBER_GET_LINK_KEY_WHEN_JOINING This denotes to a joining node that it should attempt to acquire a Trust Center Link Key during joining. This is necessary if the device does not have a pre-configured key, or wants to obtain a new one (since it may be using a well-known key during joining).

EMBER_REQUIRE_ENCRYPTED_KEY This denotes that a joining device should only accept an encrypted network key from the Trust Center (using its pre-configured key). A key sent in-the-clear by the Trust Center will be rejected and the join will fail. This option is only valid when utilizing a pre-configured key.

EMBER_NO_FRAME_COUNTER_RESET This denotes whether the device should NOT reset its outgoing frame counters (both NWK and APS) when [emberSetInitialSecurityState\(\)](#) is called. Normally it is advised to reset the frame counter before joining a new network. However in cases where a device is joining to the same network again (but not using [emberRejoinNetwork\(\)](#)) it should keep the NWK and APS frame counters stored in its tokens.

NOTE: The application is allowed to dynamically change the behavior via **EMBER_EXT_NO_FRAME_COUNTER_RESET** field.

EMBER_GET_PRECONFIGURED_KEY_FROM_INSTALL_CODE This denotes that the device should obtain its preconfigured key from an installation code

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

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

6.3.4.22 enum EmberExtendedSecurityBitmask

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

Enumerator:

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

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

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

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

6.3.4.23 enum EmberCurrentSecurityBitmask

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

Enumerator:

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

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

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

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

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

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

6.3.4.24 enum EmberKeyStructBitmask

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

Enumerator:

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

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

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

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

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

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

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

6.3.4.25 enum EmberKeyType

This denotes the type of security key.

Enumerator:

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

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

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

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

EMBER_APPLICATION_LINK_KEY This denotes that the key is an Application Link Key

EMBER_APPLICATION_MASTER_KEY This denotes that the key is an Application Master Key

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

6.3.4.26 enum EmberKeyStatus

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

Enumerator:

EMBER_KEY_STATUS_NONE

EMBER_APP_LINK_KEY_ESTABLISHED

EMBER_APP_MASTER_KEY_ESTABLISHED

```

EMBER_TRUST_CENTER_LINK_KEY_ESTABLISHED
EMBER_KEY_ESTABLISHMENT_TIMEOUT
EMBER_KEY_TABLE_FULL
EMBER_TC_RESPONDED_TO_KEY_REQUEST
EMBER_TC_APP_KEY_SENT_TO_REQUESTER
EMBER_TC_RESPONSE_TO_KEY_REQUEST_FAILED
EMBER_TC_REQUEST_KEY_TYPE_NOT_SUPPORTED
EMBER_TC_NO_LINK_KEY_FOR_REQUESTER
EMBER_TC_REQUESTER_EUI64_UNKNOWN
EMBER_TC RECEIVED FIRST APP KEY REQUEST
EMBER_TC_TIMEOUT_WAITING_FOR_SECOND_APP_KEY_REQUEST
EMBER_TC_NON_MATCHING_APP_KEY_REQUEST_RECEIVED
EMBER_TC FAILED TO SEND APP KEYS
EMBER_TC FAILED TO STORE_APP KEY REQUEST
EMBER_TC REJECTED_APP KEY REQUEST
EMBER_TC FAILED TO GENERATE_NEW KEY
EMBER_TC FAILED TO SEND_TC KEY
EMBER_TRUST_CENTER_IS_PRE_R21
EMBER_TC_REQUESTER_VERIFY_KEY_TIMEOUT
EMBER_TC_REQUESTER_VERIFY_KEY_FAILURE
EMBER_TC_REQUESTER_VERIFY_KEY_SUCCESS
EMBER_VERIFY_LINK_KEY_FAILURE
EMBER_VERIFY_LINK_KEY_SUCCESS

```

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

6.3.4.27 enum EmberLinkKeyRequestPolicy

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

Enumerator:

```

EMBER_DENY_KEY_REQUESTS
EMBER_ALLOW_KEY_REQUESTS
EMBER_GENERATE_NEW_TC_LINK_KEY

```

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

6.3.4.28 enum EmberKeySettings

Enumerator:

```

EMBER_KEY_PERMISSIONS_NONE
EMBER_KEY_PERMISSIONS_READING_ALLOWED
EMBER_KEY_PERMISSIONS_HASHING_ALLOWED

```

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

6.3.4.29 enum EmberMacPassthroughType

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

Enumerator:

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

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

6.3.4.30 enum EmberZdoStatus

Enumerator:

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

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

6.3.4.31 enum EmberZdoServerMask

Enumerator:

```
EMBER_ZDP_PRIMARY_TRUST_CENTER
EMBER_ZDP_SECONDARY_TRUST_CENTER
EMBER_ZDP_PRIMARY_BINDING_TABLE_CACHE
EMBER_ZDP_SECONDARY_BINDING_TABLE_CACHE
EMBER_ZDP_PRIMARY_DISCOVERY_CACHE
EMBER_ZDP_SECONDARY_DISCOVERY_CACHE
EMBER_ZDP_NETWORK_MANAGER
```

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

6.3.4.32 enum EmberZdoConfigurationFlags

Enumerator:

```
EMBER_APP RECEIVES SUPPORTED_ZDO_REQUESTS
EMBER_APP HANDLES UNSUPPORTED_ZDO_REQUESTS
EMBER_APP HANDLES ZDO_ENDPOINT_REQUESTS
EMBER_APP HANDLES ZDO_BINDING_REQUESTS
```

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

6.3.5 Function Documentation

6.3.5.1 uint8_t* emberKeyContents (EmberKeyData * key)

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

Parameters

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

Returns

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

6.3.5.2 uint8_t* emberCertificateContents (EmberCertificateData * cert)

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

Parameters

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

Returns

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

6.3.5.3 `uint8_t* emberPublicKeyContents (EmberPublicKeyData * key)`

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

Parameters

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

Returns

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

6.3.5.4 `uint8_t* emberPrivateKeyContents (EmberPrivateKeyData * key)`

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

Parameters

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

Returns

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

6.3.5.5 `uint8_t* emberSmacContents (EmberSmacData * key)`

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

6.3.5.6 `uint8_t* emberSignatureContents (EmberSignatureData * sig)`

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

6.3.5.7 `uint8_t* emberCertificate283k1Contents (EmberCertificate283k1Data * cert)`

This function allows the programmer to gain access to the actual certificate data bytes of the `Ember283k1CertificateData` struct.

Parameters

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

Returns

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

6.3.5.8 `uint8_t* emberPublicKey283k1Contents (EmberPublicKey283k1Data * key)`

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

Parameters

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

Returns

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

6.3.5.9 `uint8_t* emberPrivateKey283k1Contents (EmberPrivateKey283k1Data * key)`

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

Parameters

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

Returns

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

6.3.5.10 `uint8_t* ember283k1SignatureContents (Ember283k1SignatureData * sig)`

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

6.3.6 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 (uint8_t duration)`
- `EmberStatus emberJoinNetwork (EmberNodeType nodeType, EmberNetworkParameters *parameters)`
- `EmberStatus emberLeaveNetwork (void)`
- `EmberStatus emberSendZigbeeLeave (EmberNodeId destination, EmberLeaveRequestFlags flags)`
- `EmberStatus emberFindAndRejoinNetworkWithReason (bool haveCurrentNetworkKey, uint32_t channelMask, EmberRejoinReason reason)`
- `EmberStatus emberFindAndRejoinNetwork (bool haveCurrentNetworkKey, uint32_t channelMask)`
- `EmberStatus emberFindAndRejoinNetworkWithNodeType (bool haveCurrentNetworkKey, uint32_t channelMask, EmberNodeType nodeType)`
- `EmberRejoinReason emberGetLastRejoinReason (void)`
- `EmberStatus emberRejoinNetwork (bool haveCurrentNetworkKey)`
- `EmberStatus emberStartScan (EmberNetworkScanType scanType, uint32_t channelMask, uint8_t duration)`
- `EmberStatus emberStopScan (void)`
- `void emberScanCompleteHandler (uint8_t channel, EmberStatus status)`
- `void emberEnergyScanResultHandler (uint8_t channel, int8_t maxRssiValue)`
- `void emberNetworkFoundHandler (EmberZigbeeNetwork *networkFound)`
- `bool emberStackIsPerformingRejoin (void)`
- `EmberLeaveReason emberGetLastLeaveReason (EmberNodeId *returnNodeIdThatSentLeave)`
- `bool emberGetPermitJoining (void)`

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

<i>parameters</i>	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 (uint8_t *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

<i>duration</i>	A value of 0x00 disables joining. A value of 0xFF enables joining. Any other value enables joining for that number of seconds.
-----------------	--

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

<i>nodeType</i>	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() .
<i>parameters</i>	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

<i>destination</i>	is the node Id of the device that is being told to leave.
<i>flags</i>	is an bitmask indicating additional considerations for the leave request. See the EmberLeaveRequestFlags 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(bool haveCurrentNetworkKey, uint32_t 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

<i>haveCurrentNetworkKey</i>	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).
<i>channelMask</i>	A mask indicating the channels to be scanned. See emberStartScan() for format details.
<i>reason</i>	An enumeration 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 (bool haveCurrentNetworkKey, uint32_t channelMask)`

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

6.4.2.12 EmberStatus `emberFindAndRejoinNetworkWithNodeType (bool haveCurrentNetworkKey, uint32_t channelMask, EmberNodeType.nodeType)`

This call attempts to rejoin the network with a different device type.

Parameters

<i>haveCurrentNetworkKey</i>	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).
<i>channelMask</i>	A mask indicating the channels to be scanned. See emberStartScan() for format details.

<i>nodeType</i>	An enumeration indicating the device type to rejoin as. The stack only accepts EMBER_END_DEVICE and EMBER_SLEEPY_END_DEVICE .
-----------------	---

6.4.2.13 EmberRejoinReason `emberGetLastRejoinReason (void)`

Returns the enumeration for why a rejoin previously occurred..

6.4.2.14 EmberStatus `emberRejoinNetwork (bool haveCurrentNetworkKey)`

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

6.4.2.15 EmberStatus `emberStartScan (EmberNetworkScanType scanType, uint32_t channelMask, uint8_t 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 communicate 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

<i>scanType</i>	Indicates the type of scan to be performed. Possible values: EMBER_ENERGY_SCAN , EMBER_ACTIVE_SCAN .
<i>channelMask</i>	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.
<i>duration</i>	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 $((2^{\text{duration}}) + 1)$ 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.

6.4.2.16 EmberStatus emberStopScan (void)

Terminates a scan in progress.

Returns

Returns [EMBER_SUCCESS](#) if successful.

6.4.2.17 void emberScanCompleteHandler (uint8_t 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

<i>channel</i>	The channel on which the current error occurred. Undefined for the case of EMBER_SUCCESS .
<i>status</i>	The error condition that occurred on the current channel. Value will be EMBER_SUCCESS when the scan has completed.

6.4.2.18 void emberEnergyScanResultHandler (uint8_t channel, int8_t maxRssiValue)

Reports the maximum RSSI value measured on the channel.

Parameters

<i>channel</i>	The 802.15.4 channel number on which the RSSI value was measured.
<i>maxRssi-Value</i>	The maximum RSSI value measured (in units of dBm).

6.4.2.19 void emberNetworkFoundHandler (EmberZigbeeNetwork * networkFound)

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

Parameters

<i>network-Found</i>	A pointer to a EmberZigbeeNetwork structure that contains the discovered network and its associated parameters.
----------------------	---

6.4.2.20 bool 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.

**6.4.2.21 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.

6.4.2.22 bool emberGetPermitJoining (void)

Indicates the state of the permit joining in the MAC.

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 uint8_t * emberMessageBufferContents (EmberMessageBuffer buffer)
- void emberSetMessageBufferLength (EmberMessageBuffer buffer, uint8_t newLength)
- void emberHoldMessageBuffer (EmberMessageBuffer buffer)
- void emberReleaseMessageBuffer (EmberMessageBuffer buffer)
- uint8_t emberPacketBufferFreeCount (void)

Buffer Functions

- EmberMessageBuffer emberAllocateLinkedBuffers (uint8_t count)
- EmberMessageBuffer emberFillStackBuffer (unsigned int 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 (uint8_t *contents, uint8_t length)
- void emberCopyToLinkedBuffers (uint8_t *contents, EmberMessageBuffer buffer, uint8_t startIndex, uint8_t length)
- void emberCopyFromLinkedBuffers (EmberMessageBuffer buffer, uint8_t startIndex, uint8_t *contents, uint8_t length)
- void emberCopyBufferBytes (EmberMessageBuffer to, uint16_t toIndex, EmberMessageBuffer from, uint16_t fromIndex, uint16_t count)
- EmberStatus emberAppendToLinkedBuffers (EmberMessageBuffer buffer, uint8_t *contents, uint8_t length)
- EmberStatus emberAppendPgmToLinkedBuffers (EmberMessageBuffer buffer, PG-M_P contents, uint8_t length)
- EmberStatus emberAppendPgmStringToLinkedBuffers (EmberMessageBuffer buffer, PGM_P suffix)
- EmberStatus emberSetLinkedBuffersLength (EmberMessageBuffer buffer, uint8_t length)
- uint8_t * emberGetLinkedBuffersPointer (EmberMessageBuffer buffer, uint8_t index)
- XAP2B_PAGEZERO_ON uint8_t emberGetLinkedBuffersByte (EmberMessageBuffer buffer, uint8_t index)

- XAP2B_PAGEZERO_OFF void [emberSetLinkedBuffersByte](#) (EmberMessageBuffer buffer, uint8_t index, uint8_t byte)
- uint16_t [emberGetLinkedBuffersLowHighInt16u](#) (EmberMessageBuffer buffer, uint8_t index)
- void [emberSetLinkedBuffersLowHighInt16u](#) (EmberMessageBuffer buffer, uint8_t index, uint16_t value)
- uint32_t [emberGetLinkedBuffersLowHighInt32u](#) (EmberMessageBuffer buffer, uint8_t index)
- void [emberSetLinkedBuffersLowHighInt32u](#) (EmberMessageBuffer buffer, uint8_t index, uint32_t value)
- EmberMessageBuffer [emberCopyLinkedBuffers](#) (EmberMessageBuffer buffer)
- EmberMessageBuffer [emberMakeUnsharedLinkedBuffer](#) (EmberMessageBuffer buffer, bool isShared)

6.5.1 Detailed Description

These functions implement a fixed-block-size memory management scheme to store and manipulate EmberZNet 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

<i>buffer</i>	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

<i>buffer</i>	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_MESSAGE_BUFFER](#) as its link.

Parameters

<i>buffer</i>	The buffer whose link is to be set.
<i>newLink</i>	The buffer that is to follow <i>buffer</i> .

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 uint8_t* emberMessageBufferContents (EmberMessageBuffer *buffer*)

Gets a pointer to a buffer's contents. This pointer can be used to access only the first [PAC-KET_BUFFER_SIZE](#) bytes in the buffer. To read a message composed of multiple buffers, use [emberCopyFromLinkedBuffers\(\)](#).

Parameters

<i>buffer</i>	The buffer whose contents are desired.
---------------	--

Returns

Returns a pointer to the contents of *buffer*.

6.5.3.2 void emberSetMessageBufferLength (EmberMessageBuffer *buffer*, uint8_t *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

<i>buffer</i>	A buffer
<i>newLength</i>	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

<i>buffer</i>	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

<i>buffer</i>	A buffer.
---------------	-----------

6.5.3.5 EmberMessageBuffer emberAllocateLinkedBuffers (uint8_t *count*)

Allocates one or more linked buffers.

Parameters

<i>count</i>	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 (*unsigned int count, ...*)

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

Parameters

<i>count</i>	Buffer length.
...	<i>count</i> 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 (*uint8_t * contents, uint8_t 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

<i>contents</i>	A pointer to data to place in the allocated buffers.
<i>length</i>	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 (*uint8_t * contents, EmberMessageBuffer buffer, uint8_t startIndex, uint8_t 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

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

6.5.3.9 void emberCopyFromLinkedBuffers (EmberMessageBuffer *buffer*, uint8_t *startIndex*, uint8_t * *contents*, uint8_t *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

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

6.5.3.10 void emberCopyBufferBytes (EmberMessageBuffer *to*, uint16_t *toIndex*, EmberMessageBuffer *from*, uint16_t *fromIndex*, uint16_t *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

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

6.5.3.11 EmberStatus emberAppendToLinkedBuffers (EmberMessageBuffer *buffer*, uint8_t * *contents*, uint8_t *length*)

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

Parameters

<i>buffer</i>	The buffer to append data to.
<i>contents</i>	A pointer to data to append.
<i>length</i>	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*, uint8_t *length*)

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

Parameters

<i>buffer</i>	The buffer to append data to.
<i>contents</i>	The data to append.
<i>length</i>	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

<i>buffer</i>	The buffer to append data to.
<i>suffix</i>	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*, uint8_t *length*)

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

Parameters

<i>buffer</i>	The buffer whose length is to be set.
<i>length</i>	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 uint8_t* emberGetLinkedBuffersPointer (EmberMessageBuffer *buffer*, uint8_t *index*)

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

Parameters

<i>buffer</i>	The buffer that the requested byte must come from.
<i>index</i>	The index of the requested byte.

Returns

A pointer to the requested byte.

6.5.3.16 XAP2B_PAGEZERO_ON uint8_t emberGetLinkedBuffersByte (EmberMessageBuffer *buffer*, uint8_t *index*)

Gets a specified byte in a linked list of buffers.

Parameters

<i>buffer</i>	The buffer that the requested byte must come from.
<i>index</i>	The index of the requested byte.

Returns

A byte.

6.5.3.17 XAP2B_PAGEZERO_OFF void emberSetLinkedBuffersByte (EmberMessageBuffer *buffer*, uint8_t *index*, uint8_t *byte*)

Sets the indexed byte in a linked list of buffers.

Parameters

<i>buffer</i>	The buffer holding the byte to be set.
<i>index</i>	The index of the byte to set.
<i>byte</i>	The value to set the byte to.

6.5.3.18 uint16_t emberGetLinkedBuffersLowHighInt16u (EmberMessageBuffer *buffer*, uint8_t *index*)

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

Parameters

<i>buffer</i>	The buffer containing the 2-byte value.
<i>index</i>	The index of the low byte.

Returns

The 2-byte value.

6.5.3.19 void emberSetLinkedBuffersLowHighInt16u (EmberMessageBuffer *buffer*, uint8_t *index*, uint16_t *value*)

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

Parameters

<i>buffer</i>	The buffer to set the 2-byte value in.
<i>index</i>	The index of the low byte.
<i>value</i>	The 2-byte value to set.

6.5.3.20 `uint32_t emberGetLinkedBuffersLowHighInt32u (EmberMessageBuffer buffer, uint8_t index)`

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

Parameters

<i>buffer</i>	The buffer containing the 4-byte value.
<i>index</i>	The index of the low byte.

Returns

The 4-byte value.

6.5.3.21 `void emberSetLinkedBuffersLowHighInt32u (EmberMessageBuffer buffer, uint8_t index, uint32_t value)`

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

Parameters

<i>buffer</i>	The buffer to set the 2-byte value in.
<i>index</i>	The index of the low byte.
<i>value</i>	The 4-byte value to set.

6.5.3.22 `EmberMessageBuffer emberCopyLinkedBuffers (EmberMessageBuffer buffer)`

Copies a chain of linked buffers.

Parameters

<i>buffer</i>	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, bool 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

<i>buffer</i>	The buffer to copy.
<i>isShared</i>	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 `uint8_t 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

- uint8_t [emberMaximumApsPayloadLength](#) (void)
- [EmberStatus](#) [emberSendMulticastWithAlias](#) ([EmberApsFrame](#) *apsFrame, uint8_t radius, uint8_t nonmemberRadius, [EmberMessageBuffer](#) message, [EmberNodeId](#) alias, uint8_t sequence)
- [EmberStatus](#) [emberSendMulticast](#) ([EmberApsFrame](#) *apsFrame, uint8_t radius, uint8_t nonmemberRadius, [EmberMessageBuffer](#) message)
- [EmberStatus](#) [emberSendUnicast](#) ([EmberOutgoingMessageType](#) type, uint16_t indexOrDestination, [EmberApsFrame](#) *apsFrame, [EmberMessageBuffer](#) message)
- [EmberStatus](#) [emberSendBroadcast](#) ([EmberNodeId](#) destination, [EmberApsFrame](#) *apsFrame, uint8_t radius, [EmberMessageBuffer](#) message)
- [EmberStatus](#) [emberProxyBroadcast](#) ([EmberNodeId](#) source, [EmberNodeId](#) destination, uint8_t sequence, [EmberApsFrame](#) *apsFrame, uint8_t radius, [EmberMessageBuffer](#) message)
- [EmberStatus](#) [emberSendManyToOneRouteRequest](#) (uint16_t concentratorType, uint8_t radius)
- uint8_t [emberAppendSourceRouteHandler](#) ([EmberNodeId](#) destination, [EmberMessageBuffer](#) header)
- void [emberIncomingRouteRecordHandler](#) ([EmberNodeId](#) source, [EmberEUI64](#) sourceEui, uint8_t relayCount, [EmberMessageBuffer](#) header, uint8_t relayListIndex)

- void `emberIncomingManyToOneRouteRequestHandler` (`EmberNodeId` source, `EmberEUI64` longId, `uint8_t` cost)
- void `emberIncomingRouteErrorHandler` (`EmberStatus` status, `EmberNodeId` target)
- `EmberStatus` `emberCancelMessage` (`EmberMessageBuffer` message)
- void `emberMessageSentHandler` (`EmberOutgoingMessageType` type, `uint16_t` indexOrDestination, `EmberApsFrame` *apsFrame, `EmberMessageBuffer` message, `EmberStatus` status)
- void `emberIncomingMessageHandler` (`EmberIncomingMessageType` type, `EmberApsFrame` *apsFrame, `EmberMessageBuffer` message)
- `EmberStatus` `emberGetLastHopLqi` (`uint8_t` *lastHopLqi)
- `EmberStatus` `emberGetLastHopRssi` (`int8_t` *lastHopRssi)
- `EmberNodeId` `emberGetSender` (`void`)
- `EmberStatus` `emberGetSenderEui64` (`EmberEUI64` senderEui64)
- `EmberStatus` `emberSendReply` (`uint16_t` clusterId, `EmberMessageBuffer` reply)
- void `emberSetReplyFragmentData` (`uint16_t` fragmentData)
- bool `emberAddressTableEntryIsActive` (`uint8_t` addressTableIndex)
- `EmberStatus` `emberSetAddressTableRemoteEui64` (`uint8_t` addressTableIndex, `EmberEUI64` eui64)
- void `emberSetAddressTableRemoteNodeId` (`uint8_t` addressTableIndex, `EmberNodeId` id)
- void `emberGetAddressTableRemoteEui64` (`uint8_t` addressTableIndex, `EmberEUI64` eui64)
- `EmberNodeId` `emberGetAddressTableRemoteNodeId` (`uint8_t` addressTableIndex)
- void `emberSetExtendedTimeout` (`EmberEUI64` remoteEui64, `bool` extendedTimeout)
- `bool` `emberGetExtendedTimeout` (`EmberEUI64` remoteEui64)
- void `emberIdConflictHandler` (`EmberNodeId` conflictingId)
- `bool` `emberPendingAckedMessages` (`void`)
- void `emberIncomingCommandHandler` (`EmberZigbeeCommandType` commandType, `EmberMessageBuffer` commandBuffer, `uint8_t` indexOfCommand, `void` *data)
- `EmberMessageBuffer` `makeInterPanMessage` (`InterPanHeader` *headerData, `EmberMessageBuffer` payload)
- `uint8_t` `parseInterPanMessage` (`EmberMessageBuffer` message, `uint8_t` startOffset, `InterPanHeader` *headerData)
- `uint8_t` `makeInterPanMessage` (`InterPanHeader` *headerData, `uint8_t` *message, `uint8_t` maxLength, `uint8_t` *payload, `uint8_t` payloadLength)
- `uint8_t` `parseInterPanMessage` (`uint8_t` *message, `uint8_t` messageLength, `InterPanHeader` *headerData)

Variables

- `uint16_t` `emberApsAckTimeoutMs`
- `EmberMulticastTableEntry` * `emberMulticastTable`
- `uint8_t` `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 uint8_t 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 emberSendMulticastWithAlias (EmberApsFrame * *apsFrame*, uint8_t *radius*, uint8_t *nonmemberRadius*, EmberMessageBuffer *message*, EmberNodeId *alias*, uint8_t *sequence*)

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

Parameters

<i>apsFrame</i>	The APS frame for the message. The multicast will be sent to the groupId in this frame.
<i>radius</i>	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.
<i>nonmember-Radius</i>	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.
<i>message</i>	A message.
<i>alias</i>	The alias from which to send the multicast
<i>sequence</i>	The NWK sequence number for 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-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 emberSendMulticast (EmberApsFrame * *apsFrame*, uint8_t *radius*, uint8_t *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

<i>apsFrame</i>	The APS frame for the message. The multicast will be sent to the groupId in this frame.
<i>radius</i>	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.
<i>nonmember-Radius</i>	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.
<i>message</i>	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.

6.6.3.4 `EmberStatus emberSendUnicast (EmberOutgoingMessageType type, uint16_t 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

<i>type</i>	Specifies the outgoing message type. Must be one of EMBER_OUTGOING_DIRECT , EMBER_OUTGOING_VIA_ADDRESS_TABLE , or EMBER_OUTGOING_VIA_BINDING .
<i>indexOr-Destination</i>	Depending on the type of addressing used, this is either the EmberNodeId of the destination, an index into the address table, or an index into the binding table.
<i>apsFrame</i>	The APS frame which is to be added to the message.
<i>message</i>	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.

6.6.3.5 EmberStatus [emberSendBroadcast](#) ([EmberNodeId](#) *destination*, [EmberApsFrame](#) * *apsFrame*, [uint8_t](#) *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

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

Returns

An [EmberStatus](#) value.

6.6.3.6 EmberStatus [emberProxyBroadcast](#) ([EmberNodeId](#) *source*, [EmberNodeId](#) *destination*, [uint8_t](#) *sequence*, [EmberApsFrame](#) * *apsFrame*, [uint8_t](#) *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

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

Returns

An [EmberStatus](#) value.

6.6.3.7 EmberStatus `emberSendManyToOneRouteRequest(uint16_t concentratorType, uint8_t 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 EmberZNet 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

<i>concentrator-Type</i>	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.
--------------------------	--

<i>radius</i>	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 [EMBER_ERR_FATAL](#) otherwise.

6.6.3.8 `uint8_t 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

<i>destination</i>	The network destination of the message.
<i>header</i>	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.

6.6.3.9 `void emberIncomingRouteRecordHandler (EmberNodeId source, EmberEUI64 sourceEui, uint8_t relayCount, EmberMessageBuffer header, uint8_t 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

<i>source</i>	The id of the node that initiated the route record.
<i>sourceEui</i>	The EUI64 of the node that initiated the route record.
<i>relayCount</i>	The number of relays in the list.
<i>header</i>	The message buffer containing the route record frame.

<i>relayList-Index</i>	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.
------------------------	---

6.6.3.10 void `emberIncomingManyToOneRouteRequestHandler (EmberNodeId source, EmberEUI64 longId, uint8_t 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_REQUEST_HANDLER` in its configuration header to use this.

Parameters

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

6.6.3.11 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 `EMBERAPSOPTION_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_ROUTE_ERROR_HANDLER` in its configuration header.

Parameters

<i>status</i>	<code>EMBER_SOURCE_ROUTE_FAILURE</code> , <code>EMBER_MANY_TO_ONE_ROUTE_FAILURE</code> , <code>EMBER_MAC_INDIRECT_TIMEOUT</code>
<i>target</i>	The short id of the remote node.

6.6.3.12 EmberStatus emberCancelMessage (EmberMessageBuffer *message*)

DEPRECATED.

Parameters

<i>message</i>	A message.
----------------	------------

ReturnsAlways returns `EMBER_SUCCESS`.**6.6.3.13 void emberMessageSentHandler (EmberOutgoingMessageType *type*, uint16.t *indexOrDestination*, EmberApsFrame * *apsFrame*, EmberMessageBuffer *message*, EmberStatus *status*)**

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

Parameters

<i>type</i>	The type of message sent.
<i>indexOrDestination</i>	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.
<i>apsFrame</i>	The APS frame for the message.
<i>message</i>	The message that was sent.
<i>status</i>	An <code>EmberStatus</code> value of <code>EMBER_SUCCESS</code> if an ACK was received from the destination or <code>EMBER_DELIVERY_FAILED</code> if no ACK was received.

6.6.3.14 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

<i>type</i>	The type of the incoming message. One of the following: <ul style="list-style-type: none"> • EMBER_INCOMING_UNICAST • EMBER_INCOMING_UNICAST_REPLY • EMBER_INCOMING_MULTICAST • EMBER_INCOMING_MULTICAST_LOOPBACK • EMBER_INCOMING_BROADCAST • EMBER_INCOMING_BROADCAST_LOOPBACK
<i>apsFrame</i>	The APS frame from the incoming message.
<i>message</i>	The message that was sent.

6.6.3.15 EmberStatus [emberGetLastHopLqi \(uint8_t * lastHopLqi \)](#)

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

Note

This function may only be called from within

- [emberIncomingMessageHandler\(\)](#)
- [emberRf4ceIncomingMessageHandler\(\)](#)
- [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 handler 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

<i>lastHopLqi</i>	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.

6.6.3.16 EmberStatus `emberGetLastHopRssi (int8_t * 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()`
- `emberRf4ceIncomingMessageHandler()`
- `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 handler 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

<code>lastHopRssi</code>	The RSSI for the last incoming message processed.
--------------------------	---

Returns

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

6.6.3.17 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.

6.6.3.18 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

<i>senderEui64</i>	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.

6.6.3.19 EmberStatus emberSendReply (uint16_t *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

<i>clusterId</i>	The cluster ID to use for the reply.
<i>reply</i>	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.

6.6.3.20 void emberSetReplyFragmentData (uint16_t *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

<i>fragment-</i> <i>Data</i>	The low byte is the block number of the reply. The high byte is the ack bitfield of the reply.
---------------------------------	--

6.6.3.21 bool emberAddressTableEntryIsActive (uint8_t *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 **EMBER_TABLE_ENTRY_UNUSED_NODE_ID** when the address table entry is not in use.

Parameters

<i>address-</i> <i>TableIndex</i>	The index of an address table entry.
--------------------------------------	--------------------------------------

Returns

true if the address table entry is active, false otherwise.

6.6.3.22 EmberStatus emberSetAddressTableRemoteEui64 (uint8_t *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

<i>address-TableIndex</i>	The index of an address table entry.
<i>eui64</i>	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.

6.6.3.23 void emberSetAddressTableRemoteNodeId (*uint8_t 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

<i>address-TableIndex</i>	The index of an address table entry.
<i>id</i>	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.

6.6.3.24 void emberGetAddressTableRemoteEui64 (*uint8_t addressTableIndex*, *EmberEUI64 eui64*)

Gets the EUI64 of an address table entry.

Parameters

<i>address-TableIndex</i>	The index of an address table entry.
<i>eui64</i>	The EUI64 of the address table entry is copied to this location.

6.6.3.25 EmberNodeId emberGetAddressTableRemoteNodeId (*uint8_t addressTableIndex*)

Gets the short ID of an address table entry.

Parameters

<i>address- TableIndex</i>	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.

6.6.3.26 void emberSetExtendedTimeout (EmberEUI64 *remoteEui64*, bool *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

<i>remoteEui64</i>	The address of the node for which the timeout is to be set.
<i>extended- Timeout</i>	true if the retry interval should be increased by EMBER_INDIRECT_T- RANSMISSION_TIMEOUT . false if the normal retry interval should be used.

6.6.3.27 bool 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

<i>remoteEui64</i>	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.

6.6.3.28 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

<i>conflictingId</i>	The short ID for which a conflict was detected.
----------------------	---

6.6.3.29 bool 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.

6.6.3.30 void emberIncomingCommandHandler (EmberZigbeeCommandType *commandType*, EmberMessageBuffer *commandBuffer*, uint8_t *indexOfCommand*, void * *data*)

A protocol layer command has been received by the stack.

This is called when the stack receives a command that is meant for one of the protocol layers specified in [EmberZigbeeCommandType](#). The implementation can get a flat buffer of bytes representing the command byte plus the command payload by calling [emberCopyFromLinkedBuffers\(\)](#) with a contents array big enough to hold the command payload.

The commandType argument is one of the values of the [EmberZigbeeCommandType](#) enum. If the stack receives an 802.15.4 MAC beacon, it will call this function with the commandType argument set to [EMBER_ZIGBEE_COMMAND_TYPE_BEACON](#).

The implementation of this callback should **not** alter the data contained in the [EmberMessageBuffer](#), since the stack will still be processing the command at the time that this is called.

Parameters

<i>command-Type</i>	The type of command received. See EmberZigbeeCommandType .
<i>command-Buffer</i>	The EmberMessageBuffer for the command payload.
<i>indexOf-Command</i>	The starting index in the EmberMessageBuffer for the command. This means the first byte at this index will be the command byte itself, and the reset will be the command payload.
<i>data</i>	This is a pointer to auxillary data for the command. ZDO commands pass the EmberApsFrame associated with the packet here. Otherwise, this value is NULL.

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

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

6.6.3.32 `uint8_t parseInterPanMessage (EmberMessageBuffer message, uint8_t startOffset, InterPanHeader * headerData)`

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

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

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

6.6.3.34 `uint8_t parseInterPanMessage (uint8_t * message, uint8_t messageLength, InterPanHeader * headerData)`

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

6.6.4 Variable Documentation

6.6.4.1 `uint16_t 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 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 uint8_t emberMulticastTableSize

The number of entries in the multicast table.

6.7 End Devices

Functions

- `EmberNodeId emberChildId (uint8_t childIndex)`
- `uint8_t emberChildIndex (EmberNodeId childId)`
- `EmberStatus emberGetChildData (uint8_t index, EmberEUI64 childEui64Return, EmberNodeType *childTypeReturn)`
- `void emberChildJoinHandler (uint8_t index, bool joining)`
- `EmberStatus emberPollForData (void)`
- `void emberPollCompleteHandler (EmberStatus status)`
- `EmberStatus emberSetMessageFlag (EmberNodeId childId)`
- `EmberStatus emberClearMessageFlag (EmberNodeId childId)`
- `void emberPollHandler (EmberNodeId childId, bool transmitExpected)`
- `uint8_t emberChildCount (void)`
- `uint8_t emberRouterChildCount (void)`
- `uint8_t emberMaxChildCount (void)`
- `uint8_t 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_APSS_LAYER_ACTIVE, EMBER_ASSOCIATING, EMBER_ZLL_TOUCH_LINKING, EMBER_RF4CE_NETWORK_BUSY,
 EMBER_RF4CE_DUTY_CYCLING_ENABLED, EMBER_NETWORK_TIMOUT_REQUEST, EMBER_SEND_ORPHAN_NOTIFICATION } }`
- `uint16_t emberCurrentStackTasks (void)`
- `bool emberOkToNap (void)`
- `bool emberOkToHibernate (void)`
- `bool 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 266 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.

EMBERAPS_LAYER_ACTIVE The ZigBee APS layer has messages awaiting an ACK.

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

EMBER_ZLL_TOUCH_LINKING The node is currently touch linking.

EMBER_RF4CE_NETWORK_BUSY The node is busy performing some ZigBee RF4CE network operation.

EMBER_RF4CE_DUTY_CYCLING_ENABLED The node is in RF4CE duty-cycling mode.

EMBER_NETWORK_TIMEOUT_REQUEST Network Timeout Request Event

EMBER_SEND_ORPHAN_NOTIFICATION Sending Orphan Notification Event

Definition at line 235 of file [child.h](#).

6.7.4 Function Documentation

6.7.4.1 EmberNodeId emberChildId (uint8_t childIndex)

Converts a child index to a node ID.

Parameters

<i>childIndex</i>	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 uint8_t emberChildIndex (EmberNodeId *childId*)

Converts a node ID to a child index.

Parameters

<i>childId</i>	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(uint8_t 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 **EMBER_SUCCESS**. If there is no child at 'index' it returns **EMBER_NOT_JOINED**. Possible child indexes run from zero to `emberMaxChildCount()` - 1.

Parameters

<i>index</i>	The index of the child of interest.
<i>childEui64-Return</i>	The child's EUI64 is copied into here.
<i>childType-Return</i>	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(uint8_t index, bool 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_APPLICATION_HAS_CHILD_JOIN_HANDLER** in its CONFIGURATION_HEADER

Parameters

<i>index</i>	The index of the child of interest.
<i>joining</i>	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 `EMBER_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

<code>status</code>	<p>An EmberStatus value:</p> <ul style="list-style-type: none"> • <code>EMBER_SUCCESS</code> - Data was received in response to the poll. • <code>EMBER_MAC_NO_DATA</code> - No data was pending. • <code>EMBER_DELIVERY_FAILED</code> - The poll message could not be sent. • <code>EMBER_MAC_NO_ACK RECEIVED</code> - 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

<i>childId</i>	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

<i>childId</i>	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*, bool *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

<i>childId</i>	The ID of the child that is requesting data.
<i>transmit-Expected</i>	true if the child is expecting an application-supplied data message. false otherwise.

6.7.4.10 uint8_t emberChildCount (void)

Returns the number of children the node currently has.

Returns

number of children

6.7.4.11 uint8_t emberRouterChildCount (void)

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

Returns

number of router children

6.7.4.12 uint8_t 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 uint8_t 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 emberGetParentNodId (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 uint16_t 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 bool 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 bool 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_DEVICE](#) or [EMBER_MOBILE_END_DEVICE](#).

Returns

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

6.7.4.19 bool 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.

Referenced by [usbSuspendDsr\(\)](#).

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)
- bool emberHaveLinkKey ([EmberEUI64](#) remoteDevice)
- EmberStatus emberGenerateRandomKey ([EmberKeyData](#) *keyAddress)
- void emberSwitchNetworkKeyHandler (uint8_t sequenceNumber)
- EmberStatus emberRequestLinkKey ([EmberEUI64](#) partner)
- void emberZigbeeKeyEstablishmentHandler ([EmberEUI64](#) partner, [EmberKeyStatus](#) status)
- EmberStatus emberGetKeyTableEntry (uint8_t index, [EmberKeyStruct](#) *result)
- EmberStatus emberSetKeyTableEntry (uint8_t index, [EmberEUI64](#) address, bool linkKey, [EmberKeyData](#) *keyData)
- EmberStatus emberAddOrUpdateKeyTableEntry ([EmberEUI64](#) address, bool linkKey, [EmberKeyData](#) *keyData)
- uint8_t emberFindKeyTableEntry ([EmberEUI64](#) address, bool linkKey)
- EmberStatus emberEraseKeyTableEntry (uint8_t index)
- EmberStatus emberClearKeyTable (void)
- EmberStatus emberStopWritingStackTokens (void)
- EmberStatus emberStartWritingStackTokens (void)
- bool emberWritingStackTokensEnabled (void)
- EmberStatus emberApsCryptMessage (bool encrypt, [EmberMessageBuffer](#) buffer, uint8_t apsHeaderEndIndex, [EmberEUI64](#) remoteEui64)
- EmberStatus emberGetMfgSecurityConfig ([EmberMfgSecurityStruct](#) *settings)
- EmberStatus emberSetMfgSecurityConfig (uint32_t magicNumber, const [EmberMfgSecurityStruct](#) *settings)
- EmberStatus emberSetOutgoingNwkFrameCounter (uint32_t desiredValue)
- EmberStatus emberSetOutgoingApsFrameCounter (uint32_t desiredValue)
- EmberStatus emberAddTransientLinkKey ([EmberEUI64](#) partnerEUI64, [EmberKeyData](#) *key)
- void emberClearTransientLinkKeys (void)

Variables

- `uint16_t emberTransientKeyTimeoutS`

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 [emberNetworkInit](#) 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

<i>state</i>	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_SECURITY_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

<i>mask</i>	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

<i>mask</i>	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

<i>state</i>	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 [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

<i>type</i>	The Type of key to get (e.g. Trust Center Link or Network).
<i>keyStruct</i>	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_MASTER_KEY**.

6.8.3.6 bool emberHaveLinkKey (EmberEUI64 *remoteDevice*)

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

Parameters

<i>remote-Device</i>	The long address of some other device in the network.
----------------------	---

Returns

bool 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

<i>keyAddress</i>	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 (uint8_t *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

<i>sequence-Number</i>	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

<i>partner</i>	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

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

6.8.3.11 EmberStatus emberGetKeyTableEntry (uint8_t *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

<i>index</i>	The index in the key table of the entry to get.
<i>result</i>	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_OF_RANGE](#) if the passed index is not valid. [EMBER_SUCCESS](#) on success.

6.8.3.12 EmberStatus emberSetKeyTableEntry (uint8_t *index*, EmberEUI64 *address*, bool *linkKey*, EmberKeyData * *keyData*)

A function to set an entry in the key table.

Parameters

<i>index</i>	The index in the key table of the entry to set.
<i>address</i>	The address of the partner device associated with the key.
<i>keyData</i>	A pointer to the key data associated with the key entry.
<i>linkKey</i>	A bool 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_INDEX_OUT_OF_RANGE](#) if passed index is not valid. [EMBER_SUCCESS](#) on success.

6.8.3.13 EmberStatus emberAddOrUpdateKeyTableEntry (EmberEUI64 *address*, bool *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

<i>address</i>	The IEEE Address of the partner device that shares the key.
<i>linkKey</i>	A bool indicating whether this is a Link or Master key.
<i>keyData</i>	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 uint8_t emberFindKeyTableEntry (EmberEUI64 *address*, bool *linkKey*)

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

Parameters

<i>address</i>	The IEEE Address of the partner device that shares the key. To find the first empty entry pass in an address of all zeros.
<i>linkKey</i>	A bool 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 (uint8_t index)

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

Parameters

<i>index</i>	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_NO_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 [emberStopWritingStackTokens\(\)](#) that it now wrote to the token system. [EMBER_INVALID_CALL](#) otherwise.

6.8.3.19 bool 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 (bool encrypt, EmberMessageBuffer buffer, uint8_t 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

<i>encrypt</i>	a bool indicating whether perform encryption (true) or decryption (false).
<i>buffer</i>	An EmberMessageBuffer containing the APS frame to decrypt or encrypt.
<i>apsHeader-EndIndex</i>	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.
<i>remoteEui64</i>	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 [EmberStatus](#) 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

<i>settings</i>	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 (uint32_t 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_CONFIG_MAGIC_NUMBER`. This prevents accidental calls to this function when `emberGetMfgSecurityConfig()` 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 will 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 `EmberMfgSecurityStruct` are set to `EMBER_KEY_PERMISSIONS_READING_ALLOWED`, then the actual key data is returned. This is the default value of the token.

Parameters

<code>magicNumber</code>	A 32-bit value that must correspond to <code>EMBER_MFG_SECURITY_CONFIG_MAGIC_NUMBER</code> , otherwise <code>EMBER_INVALID_CALL</code> will be returned.
<code>settings</code>	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_CONFIGURATION_INVALID` if there was an attempt to write an unerased manufacturing token (i.e. the token has already been set).

6.8.3.23 `EmberStatus emberSetOutgoingNwkFrameCounter (uint32_t desiredValue)`

Function to set NWK layer outgoing frame counter (intended for device restoration purposes). Caveats:

- Can only be called before NetworkInit / FormNetwork / JoinNetwork, when `emberNetworkState()`==EMBER_NO_NETWORK.
- This function should be called before `emberSetInitialSecurityState`, and the EMBER_NO_FRAME_COUNTER_RESET bitmask should be added to the initial security bitmask when ::emberSetInitialSecurityState is called.
- If used in multi-network context, be sure to call `emberSetCurrentNetwork()` prior to calling this function.

Parameters

<code>desiredValue</code>	The desired outgoing NWK frame counter value. This should needs to be less than MAX_INT32U_VALUE to ensure that rollover does not occur on the next encrypted transmission.
---------------------------	---

Returns

`EMBER_SUCCESS` if calling context is valid (`emberNetworkState()` == EMBER_NO_NETWORK) and `desiredValue < MAX_INT32U_VALUE`. Otherwise, `EMBER_INVALID_CALL`.

6.8.3.24 EmberStatus `emberSetOutgoingApsFrameCounter (uint32_t desiredValue)`

Function to set APS layer outgoing frame counter for Trust Center Link Key (intended for device restoration purposes). Caveats:

- Can only be called before NetworkInit / FormNetwork / JoinNetwork, when `emberNetworkState()`==EMBER_NO_NETWORK.
- This function should be called before `emberSetInitialSecurityState`, and the EMBER_NO_FRAME_COUNTER_RESET bitmask should be added to the initial security bitmask when ::emberSetInitialSecurityState is called.
- If used in multi-network context, be sure to call `emberSetCurrentNetwork()` prior to calling this function.

Parameters

<code>desiredValue</code>	The desired outgoing APS frame counter value. This should needs to be less than MAX_INT32U_VALUE to ensure that rollover does not occur on the next encrypted transmission.
---------------------------	---

Returns

`EMBER_SUCCESS` if calling context is valid (`emberNetworkState()` == EMBER_NO_NETWORK) and `desiredValue < MAX_INT32U_VALUE`. Otherwise, `EMBER_INVALID_CALL`.

6.8.3.25 EmberStatus emberAddTransientLinkKey (EmberEUI64 *partnerEUI64*, EmberKeyData * *key*)

Add a temporary link key for a joining device. The link key will be stored for *emberTransientKeyTimeoutS* seconds. After that time, the key will be removed. The removed key will need to be added again using this API in order for it to be used by a joining device again.

Parameters

<i>partnerEUI64</i>	The EUI of the joining device. If all FF's are entered for this value, then the key will be able to be used for all joining devices that do not already have transient key table entries.
<i>key</i>	The temporary link key to the joining device.

Returns

[EMBER_SUCCESS](#) if the transient key has been added. [EMBER_INVALID_CALL](#) if the key type is invalid. [EMBER_NO_BUFFERS](#) if there are no buffers.

6.8.3.26 void emberClearTransientLinkKeys (void)

Clear all of the transient link keys from RAM.

6.8.4 Variable Documentation

6.8.4.1 uint16_t emberTransientKeyTimeoutS

The length of time, in seconds, that a trust center will store a transient link key that a device can use to join its network. A transient key is added with a call to `emberAddTransientLinkKey`. After the transient key is added, it will be removed once this amount of time has passed. A joining device will not be able to use that key to join until it is added again on the trust center. The default value is 300 seconds, i.e., 5 minutes.

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 targetLong, EmberKeyData *newKey)`
- `EmberStatus emberBroadcastNetworkKeySwitch (void)`
- `EmberJoinDecision emberTrustCenterJoinHandler (EmberNodeId newNodeId, EmberEUI64 newNodeEui64, EmberDeviceUpdate status, EmberNodeId parentOfNewNode)`
- `EmberStatus emberBecomeTrustCenter (EmberKeyData *newNetworkKey)`
- `EmberStatus emberSendRemoveDevice (EmberNodeId destShort, EmberEUI64 destLong, EmberEUI64 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

<code>key</code>	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

<i>targetShort</i>	the short node ID of the device to send a NWK key update to.
<i>targetLong</i>	the EUI64 of the node to send a key update NWK key update to.
<i>nwkKey</i>	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

<i>newNodeId</i>	The node id of the device wishing to join.
<i>newNode-Eui64</i>	The EUI64 of the device wishing to join.
<i>status</i>	The EmberUpdateDeviceStatus indicating whether the device is joining/rejoining or leaving.
<i>parentOf-NewNode</i>	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

<i>newNetwork-Key</i>	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

<i>destShort</i>	The short node ID of the destination of the command.
<i>destLong</i>	The EUI64 of the destination of the command.
<i>deviceToRemoveLong</i>	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 (uint8_t index, EmberBindingTableEntry *value)`
- `EmberStatus emberGetBinding (uint8_t index, EmberBindingTableEntry *result)`
- `EmberStatus emberDeleteBinding (uint8_t index)`
- `bool emberBindingIsActive (uint8_t index)`
- `EmberNodeId emberGetBindingRemoteNodeId (uint8_t index)`
- `void emberSetBindingRemoteNodeId (uint8_t index, EmberNodeId id)`
- `EmberStatus emberClearBindingTable (void)`
- `EmberStatus emberRemoteSetBindingHandler (EmberBindingTableEntry *entry)`
- `EmberStatus emberRemoteDeleteBindingHandler (uint8_t index)`
- `uint8_t emberGetBindingIndex (void)`
- `EmberStatus emberSetReplyBinding (uint8_t index, EmberBindingTableEntry *entry)`
- `EmberStatus emberNoteSendersBinding (uint8_t 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 (uint8_t 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

<code>index</code>	The index of a binding table entry.
<code>value</code>	A pointer to a structure.

Returns

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

6.10.2.2 `EmberStatus emberGetBinding (uint8_t index, EmberBindingTableEntry * result)`

Copies a binding table entry to the structure that `result` points to.

Parameters

<i>index</i>	The index of a binding table entry.
<i>result</i>	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 (uint8_t *index*)

Deletes a binding table entry.

Parameters

<i>index</i>	The index of a binding table entry.
--------------	-------------------------------------

Returns

An [EmberStatus](#) value that indicates the success or failure of the command.

6.10.2.4 bool emberBindingIsActive (uint8_t *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

<i>index</i>	The index of a binding table entry.
--------------	-------------------------------------

Returns

true if the binding table entry is active, false otherwise.

6.10.2.5 EmberNodeId emberGetBindingRemoteNodeId (uint8_t *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 by 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

<i>index</i>	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 (uint8_t *index*, EmberNodeId *id*)

Set the node ID for the binding's destination. See [emberGetBindingRemoteNodeId\(\)](#) for a description.

Parameters

<i>index</i>	The index of a binding table entry.
<i>id</i>	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 [emberRemoteDeleteBindingHandler\(\)](#).

Parameters

<i>entry</i>	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 (uint8_t *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_APPLICATION_HAS_REMOTE_BINDING_HANDLER in its CONFIGURATION_HEADER and also include [emberRemoteSetBindingHandler\(\)](#).

Parameters

<i>index</i>	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 uint8_t 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 (uint8_t *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

<i>index</i>	The index of the binding to set.
<i>entry</i>	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 (uint8_t *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

<i>index</i>	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_APSC_UNICAST_MESSAGE_COUNT
- #define EMBER_BINDING_TABLE_SIZE
- #define EMBER_ADDRESS_TABLE_SIZE
- #define EMBER_RESERVED_MOBILE_CHILD_ENTRIES
- #define EMBER_ROUTE_TABLE_SIZE
- #define EMBER_DISCOVERY_TABLE_SIZE
- #define EMBER_MULTICAST_TABLE_SIZE
- #define EMBER_SOURCE_ROUTE_TABLE_SIZE
- #define EMBER_DEFAULT_BROADCAST_TABLE_SIZE
- #define EMBER_BROADCAST_TABLE_SIZE
- #define EMBER_RETRY_QUEUE_SIZE
- #define EMBER_ASSERT_SERIAL_PORT
- #define EMBER_MAXIMUM_ALARM_DATA_SIZE
- #define EMBER_BROADCAST_ALARM_DATA_SIZE
- #define EMBER_UNICAST_ALARM_DATA_SIZE
- #define EMBER_FRAGMENT_DELAY_MS
- #define EMBER_FRAGMENT_MAX_WINDOW_SIZE
- #define EMBER_FRAGMENT_WINDOW_SIZE
- #define EMBER_BINDING_TABLE_TOKEN_SIZE
- #define EMBER_CHILD_TABLE_TOKEN_SIZE
- #define EMBER_KEY_TABLE_TOKEN_SIZE
- #define EMBER_REQUEST_KEY_TIMEOUT
- #define EMBER_TRANSIENT_KEY_TIMEOUT_S
- #define EMBER_END_DEVICE_BIND_TIMEOUT
- #define EMBER_PAN_ID_CONFLICT_REPORT_THRESHOLD
- #define EMBER_TASK_COUNT
- #define EMBER_MAX_SUPPORTED_NETWORKS

- #define EMBER_SUPPORTED_NETWORKS
- #define EMBER_ZLL_GROUP_ADDRESSES
- #define EMBER_ZLL_RSSI_THRESHOLD
- #define EMBER_RF4CE_PAIRING_TABLE_SIZE
- #define EMBER_RF4CE_PAIRING_TABLE_TOKEN_SIZE
- #define EMBER_RF4CE_PENDING_OUTGOING_PACKET_TABLE_SIZE
- #define EMBER_GP_PROXY_TABLE_SIZE
- #define EMBER_GP_PROXY_TABLE_TOKEN_SIZE
- #define EMBER_GP_SINK_TABLE_SIZE
- #define EMBER_GP_SINK_TABLE_TOKEN_SIZE

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_MAX_END_DEVICE_CHILDREN](#). In the mesh stack router children are not stored in the child table.

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

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

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

6.11.2.7 #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 75.

Each buffer requires 36 bytes of RAM (32 for the buffer itself plus 4 bytes of overhead).

Definition at line 218 of file [ember-configuration-defaults.h](#).

6.11.2.12 #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=true (0xFFFFD). 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_SHIFT](#), that an [EMBER_END_DEVICE](#) or [EMBER_SLEEPY_END_DEVICE](#) can wait between polls. The timeout value in seconds is [EMBER_END_DEVICE_POLL_TIMEOUT << EMBER_END_DEVICE_POLL_TIMEOUT_SHIFT](#). If no poll is heard within this time, then the parent removes the end device from its tables. Note: there is a separate [EMBER_MOBILE_NODE_POLL_TIMEOUT](#) for mobile end devices.

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

Definition at line 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_TIMEOUT](#) for an explanation of how this value is used.

The default value of 6 means gives [EMBER_END_DEVICE_POLL_TIMEOUT](#) a default unit of 64 seconds, or approximately one minute. The maximum value for [EMBER_END_DEVICE_POLL_TIMEOUT_SHIFT](#) is 14.

Definition at line 289 of file [ember-configuration-defaults.h](#).

6.11.2.19 #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 414 of file [ember-configuration-defaults.h](#).

6.11.2.29 #define EMBER_BROADCAST_TABLE_SIZE

Definition at line 417 of file [ember-configuration-defaults.h](#).

6.11.2.30 #define EMBER_RETRY_QUEUE_SIZE

Definition at line 426 of file [ember-configuration-defaults.h](#).

6.11.2.31 #define EMBER_ASSERT_SERIAL_PORT

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

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

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

Definition at line 442 of file [ember-configuration-defaults.h](#).

6.11.2.32 #define EMBER_MAXIMUM_ALARM_DATA_SIZE

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

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

`EMBER_MAXIMUM_ALARM_DATA_SIZE` is defined to be 16.

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

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

6.11.2.33 #define EMBER_BROADCAST_ALARM_DATA_SIZE

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

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

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

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

Definition at line 478 of file `ember-configuration-defaults.h`.

6.11.2.34 #define EMBER_UNICAST_ALARM_DATA_SIZE

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

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

Definition at line 492 of file `ember-configuration-defaults.h`.

6.11.2.35 #define EMBER_FRAGMENT_DELAY_MS

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

Definition at line 501 of file `ember-configuration-defaults.h`.

6.11.2.36 #define EMBER_FRAGMENT_MAX_WINDOW_SIZE

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

Definition at line 507 of file `ember-configuration-defaults.h`.

6.11.2.37 #define EMBER_FRAGMENT_WINDOW_SIZE

The number of blocks of a fragmented message that can be sent in a single window. The maximum is [EMBER_FRAGMENT_MAX_WINDOW_SIZE](#). The default value is 1.

Definition at line [514](#) of file [ember-configuration-defaults.h](#).

6.11.2.38 #define EMBER_BINDING_TABLE_TOKEN_SIZE

Definition at line [520](#) of file [ember-configuration-defaults.h](#).

6.11.2.39 #define EMBER_CHILD_TABLE_TOKEN_SIZE

Definition at line [523](#) of file [ember-configuration-defaults.h](#).

6.11.2.40 #define EMBER_KEY_TABLE_TOKEN_SIZE

Definition at line [526](#) of file [ember-configuration-defaults.h](#).

6.11.2.41 #define EMBER_REQUEST_KEY_TIMEOUT

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

The Zigbee Pro Compliant value is 0.

Definition at line [542](#) of file [ember-configuration-defaults.h](#).

6.11.2.42 #define EMBER_TRANSIENT_KEY_TIMEOUT_S

The length of time, in seconds, that a trust center will store a transient link key that a device can use to join its network. A transient key is added with a call to `emberAddTransientLinkKey`. After the transient key is added, it will be removed once this amount of time has passed. A joining device will not be able to use that key to join until it is added again on the trust center. The default value is 300 seconds, i.e., 5 minutes.

Definition at line [556](#) of file [ember-configuration-defaults.h](#).

6.11.2.43 #define EMBER_END_DEVICE_BIND_TIMEOUT

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

Definition at line [563](#) of file [ember-configuration-defaults.h](#).

6.11.2.44 #define EMBER_PAN_ID_CONFLICT_REPORT_THRESHOLD

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

Definition at line 575 of file [ember-configuration-defaults.h](#).

6.11.2.45 #define EMBER_TASK_COUNT

The number of event tasks that can be tracked for the purpose of processor idling. The Ember Zigbee Pro and Zigbee RF4CE stacks require 1 task each, an application and associated libraries may use additional tasks, though typically no more than 4 are needed for most applications.

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

6.11.2.46 #define EMBER_MAX_SUPPORTED_NETWORKS

The number of networks supported by the stack.

Definition at line 589 of file [ember-configuration-defaults.h](#).

6.11.2.47 #define EMBER_SUPPORTED_NETWORKS

Definition at line 594 of file [ember-configuration-defaults.h](#).

6.11.2.48 #define EMBER_ZLL_GROUP_ADDRESSES

The number of unique group identifiers that this device requires.

Definition at line 601 of file [ember-configuration-defaults.h](#).

6.11.2.49 #define EMBER_ZLL_RSSI_THRESHOLD

The RSSI threshold applied to incoming scan requests.

Definition at line 607 of file [ember-configuration-defaults.h](#).

6.11.2.50 #define EMBER_RF4CE_PAIRING_TABLE_SIZE

The maximum number of pairings supported by the stack.

Definition at line 613 of file [ember-configuration-defaults.h](#).

6.11.2.51 #define EMBER_RF4CE_PAIRING_TABLE_TOKEN_SIZE

The maximum number of pairings stored in non-volatile memory.

Definition at line 619 of file [ember-configuration-defaults.h](#).

6.11.2.52 #define EMBER_RF4CE_PENDING_OUTGOING_PACKET_TABLE_SIZE

The maximum number of outgoing RF4CE packets supported by the stack.

Definition at line [625](#) of file [ember-configuration-defaults.h](#).

6.11.2.53 #define EMBER_GP_PROXY_TABLE_SIZE

The number of proxy table entries supported.

Definition at line [631](#) of file [ember-configuration-defaults.h](#).

6.11.2.54 #define EMBER_GP_PROXY_TABLE_TOKEN_SIZE

The maximum number of pairings stored in non-volatile memory.

Definition at line [638](#) of file [ember-configuration-defaults.h](#).

6.11.2.55 #define EMBER_GP_SINK_TABLE_SIZE

The number of sink table entries supported.

Definition at line [644](#) of file [ember-configuration-defaults.h](#).

6.11.2.56 #define EMBER_GP_SINK_TABLE_TOKEN_SIZE

The maximum number of pairings stored in non-volatile memory.

Definition at line [650](#) of file [ember-configuration-defaults.h](#).

6.12 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_NOT_FOUND`(x03)
- #define `EMBER EEPROM_MFG_STACK_VERSION_MISMATCH`(x04)
- #define `EMBER_INCOMPATIBLE_STATIC_MEMORY_DEFINITIONS`(x05)
- #define `EMBER EEPROM_MFG_VERSION_MISMATCH`(x06)
- #define `EMBER EEPROM_STACK_VERSION_MISMATCH`(x07)

Packet Buffer Module Errors

- #define `EMBER_NO_BUFFERS`(x18)

Serial Manager Errors

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

MAC Errors

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

- #define EMBER_MAC_INVALID_CHANNEL_MASK(x35)
- #define EMBER_MAC_COMMAND_TRANSMIT_FAILURE(x36)
- #define EMBER_MAC_NO_ACK RECEIVED(x40)
- #define EMBER_MAC_RADIO_NETWORK_SWITCH_FAILED(x41)
- #define EMBER_MAC_INDIRECT_TIMEOUT(x42)

Simulated EEPROM Errors

- #define EMBER_SIM_EEPROM_ERASE_PAGE_GREEN(x43)
- #define EMBER_SIM_EEPROM_ERASE_PAGE_RED(x44)
- #define EMBER_SIM_EEPROM_FULL(x45)
- #define EMBER_SIM_EEPROM_INIT_1 FAILED(x48)
- #define EMBER_SIM_EEPROM_INIT_2 FAILED(x49)
- #define EMBER_SIM_EEPROM_INIT_3 FAILED(x4A)
- #define EMBER_SIM_EEPROM_REPAIRING(x4D)

Flash Errors

- #define EMBER_ERR_FLASH_WRITE_INHIBITED(x46)
- #define EMBER_ERR_FLASH_VERIFY FAILED(x47)
- #define EMBER_ERR_FLASH_PROG FAIL(x4B)
- #define EMBER_ERR_FLASH_ERASE FAIL(x4C)

Bootloader Errors

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

Transport Errors

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

Green Power status codes

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

HAL Module Errors

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

PHY Errors

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

Return Codes Passed to emberStackStatusHandler()

See also [emberStackStatusHandler\(\)](#).

- #define EMBER_NETWORK_UP(x90)
- #define EMBER_NETWORK_DOWN(x91)
- #define EMBER_JOIN FAILED(x94)
- #define EMBER_MOVE FAILED(x96)
- #define EMBER_CANNOT JOIN AS ROUTER(x98)
- #define EMBER_NODE_ID_CHANGED(x99)
- #define EMBER_PAN_ID_CHANGED(x9A)
- #define EMBER_CHANNEL_CHANGED(x9B)
- #define EMBER_NO_BEACONS(xAB)
- #define EMBER RECEIVED KEY IN THE CLEAR(xAC)
- #define EMBER_NO_NETWORK_KEY RECEIVED(xAD)
- #define EMBER_NO_LINK_KEY RECEIVED(xAE)
- #define EMBER_PRECONFIGURED_KEY REQUIRED(xAF)

Security Errors

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

Miscellaneous Network Errors

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

Miscellaneous Utility Errors

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

ZigBee RF4CE specific errors.

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

Application Errors

These error codes are available for application use.

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

6.12.1 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

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

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

6.12.2.2 #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_NOT_FOUND(x03)

The requested information was not found.

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

6.12.2.6 #define EMBER_EEPROM_MFG_STACK_VERSION_MISMATCH(x04)

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

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

6.12.2.7 #define EMBER_INCOMPATIBLE_STATIC_MEMORY_DEFINITIONS(x05)

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

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

6.12.2.8 #define EMBER_EEPROM_MFG_VERSION_MISMATCH(x06)

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

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

6.12.2.9 #define EMBER_EEPROM_STACK_VERSION_MISMATCH(x07)

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

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

6.12.2.10 #define EMBER_NO_BUFFERS(x18)

There are no more buffers.

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

6.12.2.11 #define EMBER_SERIAL_INVALID_BAUD_RATE(x20)

Specified an invalid baud rate.

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

6.12.2.12 #define EMBER_SERIAL_INVALID_PORT(x21)

Specified an invalid serial port.

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

6.12.2.13 #define EMBER_SERIAL_TX_OVERFLOW(x22)

Tried to send too much data.

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

6.12.2.14 #define EMBER_SERIAL_RX_OVERFLOW(x23)

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

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

6.12.2.15 #define EMBER_SERIAL_RX_FRAME_ERROR(x24)

Detected a UART framing error.

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

6.12.2.16 #define EMBER_SERIAL_RX_PARITY_ERROR(x25)

Detected a UART parity error.

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

6.12.2.17 #define EMBER_SERIAL_RX_EMPTY(x26)

There is no received data to process.

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

6.12.2.18 #define EMBER_SERIAL_RX_OVERRUN_ERROR(x27)

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

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

6.12.2.19 #define EMBER_MAC_TRANSMIT_QUEUE_FULL(x39)

The MAC transmit queue is full.

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

6.12.2.20 #define EMBER_MAC_UNKNOWN_HEADER_TYPE(x3A)

MAC header FCF error on receive.

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

6.12.2.21 #define EMBER_MAC_ACK_HEADER_TYPE(x3B)

MAC ACK header received.

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

6.12.2.22 #define EMBER_MAC_SCANNING(x3D)

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

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

6.12.2.23 #define EMBER_MAC_NO_DATA(x31)

No pending data exists for device doing a data poll.

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

6.12.2.24 #define EMBER_MAC_JOINED_NETWORK(x32)

Attempt to scan when we are joined to a network.

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

6.12.2.25 #define EMBER_MAC_BAD_SCAN_DURATION(x33)

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

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

6.12.2.26 #define EMBER_MAC_INCORRECT_SCAN_TYPE(x34)

emberStartScan was called with an incorrect scan type.

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

6.12.2.27 #define EMBER_MAC_INVALID_CHANNEL_MASK(x35)

emberStartScan was called with an invalid channel mask.

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

6.12.2.28 #define EMBER_MAC_COMMAND_TRANSMIT_FAILURE(x36)

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

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

6.12.2.29 #define EMBER_MAC_NO_ACK RECEIVED(x40)

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

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

6.12.2.30 #define EMBER_MAC_RADIO_NETWORK_SWITCH_FAILED(x41)

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

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

6.12.2.31 #define EMBER_MAC INDIRECT_TIMEOUT(x42)

Indirect data message timed out before polled.

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

6.12.2.32 #define EMBER_SIM EEPROM_ERASE_PAGE_GREEN(x43)

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

The application should call the function [halSimEepromErasePage\(\)](#) when it can to erase a page.

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

6.12.2.33 #define EMBER_SIM EEPROM_ERASE_PAGE_RED(x44)

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

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

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

6.12.2.34 #define EMBER_SIM_EEPROM_FULL(x45)

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

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

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

6.12.2.35 #define EMBER_SIM_EEPROM_INIT_1_FAILED(x48)

Attempt 1 to initialize the Simulated EEPROM has failed.

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

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

6.12.2.36 #define EMBER_SIM_EEPROM_INIT_2_FAILED(x49)

Attempt 2 to initialize the Simulated EEPROM has failed.

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

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

6.12.2.37 #define EMBER_SIM_EEPROM_INIT_3_FAILED(x4A)

Attempt 3 to initialize the Simulated EEPROM has failed.

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

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

6.12.2.38 #define EMBER_SIM_EEPROM_REPAIRING(x4D)

The Simulated EEPROM is repairing itself.

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

Note

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

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

6.12.2.39 #define EMBER_ERR_FLASH_WRITE_INHIBITED(x46)

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

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

6.12.2.40 #define EMBER_ERR_FLASH_VERIFY_FAILED(x47)

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

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

6.12.2.41 #define EMBER_ERR_FLASH_PROG_FAIL(x4B)**Description:**

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

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

6.12.2.42 #define EMBER_ERR_FLASH_ERASE_FAIL(x4C)**Description:**

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

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

6.12.2.43 #define EMBER_ERR_BOOTLOADER_TRAP_TABLE_BAD(x58)

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

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

6.12.2.44 #define EMBER_ERR_BOOTLOADER_TRAP_UNKNOWN(x59)

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

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

6.12.2.45 #define EMBER_ERR_BOOTLOADER_NO_IMAGE(x05A)

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

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

6.12.2.46 #define EMBER_DELIVERY_FAILED(x66)

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

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

6.12.2.47 #define EMBER_BINDING_INDEX_OUT_OF_RANGE(x69)

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

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

6.12.2.48 #define EMBER_ADDRESS_TABLE_INDEX_OUT_OF_RANGE(x6A)

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

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

6.12.2.49 #define EMBER_INVALID_BINDING_INDEX(x6C)

An invalid binding table index was given to a function.

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

6.12.2.50 #define EMBER_INVALID_CALL(x70)

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

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

6.12.2.51 #define EMBER_COST_NOT_KNOWN(x71)

The link cost to a node is not known.

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

6.12.2.52 #define EMBER_MAX_MESSAGE_LIMIT_REACHED(x72)

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

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

6.12.2.53 #define EMBER_MESSAGE_TOO_LONG(x74)

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

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

6.12.2.54 #define EMBER_BINDING_IS_ACTIVE(x75)

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

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

6.12.2.55 #define EMBER_ADDRESS_TABLE_ENTRY_IS_ACTIVE(x76)

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

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

6.12.2.56 #define EMBER_MATCH(x78)

security match

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

6.12.2.57 #define EMBER_DROP_FRAME(x79)

drop frame

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

6.12.2.58 #define EMBER_PASS_UNPROCESSED(x7A)

security match

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

6.12.2.59 #define EMBER_TX_THEN_DROP(x7B)

security match

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

6.12.2.60 #define EMBER_NO_SECURITY(x7C)

security match

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

6.12.2.61 #define EMBER_COUNTER_FAILURE(x7D)

security match

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

6.12.2.62 #define EMBER_AUTH_FAILURE(x7E)

security match

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

6.12.2.63 #define EMBER_UNPROCESSED(x7F)

security match

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

6.12.2.64 #define EMBER_ADC_CONVERSION_DONE(x80)

Conversion is complete.

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

6.12.2.65 #define EMBER_ADC_CONVERSION_BUSY(x81)

Conversion cannot be done because a request is being processed.

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

6.12.2.66 #define EMBER_ADC_CONVERSION_DEFERRED(x82)

Conversion is deferred until the current request has been processed.

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

6.12.2.67 #define EMBER_ADC_NO_CONVERSION_PENDING(x84)

No results are pending.

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

6.12.2.68 #define EMBER_SLEEP_INTERRUPTED(x85)

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

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

6.12.2.69 #define EMBER_PHY_TX_UNDERFLOW(x88)

The transmit hardware buffer underflowed.

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

6.12.2.70 #define EMBER_PHY_TX_INCOMPLETE(x89)

The transmit hardware did not finish transmitting a packet.

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

6.12.2.71 #define EMBER_PHY_INVALID_CHANNEL(x8A)

An unsupported channel setting was specified.

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

6.12.2.72 #define EMBER_PHY_INVALID_POWER(x8B)

An unsupported power setting was specified.

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

6.12.2.73 #define EMBER_PHY_TX_BUSY(x8C)

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

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

6.12.2.74 #define EMBER_PHY_TX_CCA_FAIL(x8D)

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

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

6.12.2.75 #define EMBER_PHY_OSCILLATOR_CHECK_FAILED(x8E)

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

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

6.12.2.76 #define EMBER_PHY_ACK RECEIVED(x8F)

The expected ACK was received after the last transmission.

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

6.12.2.77 #define EMBER_NETWORK_UP(x90)

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

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

6.12.2.78 #define EMBER_NETWORK_DOWN(x91)

The network is not operating.

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

6.12.2.79 #define EMBER_JOIN FAILED(x94)

An attempt to join a network failed.

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

6.12.2.80 #define EMBER_MOVE_FAILED(x96)

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

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

6.12.2.81 #define EMBER_CANNOT_JOIN_AS_ROUTER(x98)

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

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

6.12.2.82 #define EMBER_NODE_ID_CHANGED(x99)

The local node ID has changed. The application can obtain the new node ID by calling [emberGetNodeId\(\)](#).

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

6.12.2.83 #define EMBER_PAN_ID_CHANGED(x9A)

The local PAN ID has changed. The application can obtain the new PAN ID by calling [emberGetPanId\(\)](#).

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

6.12.2.84 #define EMBER_CHANNEL_CHANGED(x9B)

The channel has changed.

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

6.12.2.85 #define EMBER_NO_BEACONS(xAB)

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

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

6.12.2.86 #define EMBER RECEIVED KEY IN THE CLEAR(xAC)

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

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

6.12.2.87 #define EMBER_NO_NETWORK_KEY RECEIVED(xAD)

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

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

6.12.2.88 #define EMBER_NO_LINK_KEY RECEIVED(xAE)

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

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

6.12.2.89 #define EMBER_PRECONFIGURED_KEY_REQUIRED(xAF)

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

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

6.12.2.90 #define EMBER_KEY_INVALID(xB2)

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

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

6.12.2.91 #define EMBER_INVALID_SECURITY_LEVEL(x95)

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

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

6.12.2.92 #define EMBER_APS_ENCRYPTION_ERROR(xA6)

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

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

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

6.12.2.93 #define EMBER_TRUST_CENTER_MASTER_KEY_NOT_SET(xA7)

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

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

6.12.2.94 #define EMBER_SECURITY_STATE_NOT_SET(xA8)

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

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

6.12.2.95 #define EMBER_KEY_TABLE_INVALID_ADDRESS(*xB3*)

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

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

6.12.2.96 #define EMBER_SECURITY_CONFIGURATION_INVALID(*xB7*)

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

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

6.12.2.97 #define EMBER_TOO_SOON_FOR_SWITCH_KEY(*xB8*)

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

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

6.12.2.98 #define EMBER_SIGNATURE_VERIFY_FAILURE(*xB9*)

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

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

6.12.2.99 #define EMBER_KEY_NOT_AUTHORIZED(*xBB*)

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

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

6.12.2.100 #define EMBER_SECURITY_DATA_INVALID(*xBD*)

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

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

6.12.2.101 #define EMBER_NOT_JOINED(*x93*)

The node has not joined a network.

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

6.12.2.102 #define EMBER_NETWORK_BUSY(xA1)

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

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

6.12.2.103 #define EMBER_INVALID_ENDPOINT(xA3)

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

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

6.12.2.104 #define EMBER_BINDING_HAS_CHANGED(xA4)

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

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

6.12.2.105 #define EMBER_INSUFFICIENT_RANDOM_DATA(xA5)

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

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

6.12.2.106 #define EMBER_SOURCE_ROUTE_FAILURE(xA9)

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

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

6.12.2.107 #define EMBER_MANY_TO_ONE_ROUTE_FAILURE(xAA)

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

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

6.12.2.108 #define EMBER_STACK_AND_HARDWARE_MISMATCH(xB0)

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

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

6.12.2.109 #define EMBER_INDEX_OUT_OF_RANGE(xB1)

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

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

6.12.2.110 #define EMBER_TABLE_FULL(*xB4*)

There are no empty entries left in the table.

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

6.12.2.111 #define EMBER_TABLE_ENTRY_ERASED(*xB6*)

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

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

6.12.2.112 #define EMBER_LIBRARY_NOT_PRESENT(*xB5*)

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

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

6.12.2.113 #define EMBER_OPERATION_IN_PROGRESS(*xBA*)

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

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

6.12.2.114 #define EMBER_TRUST_CENTER_EUI_HAS_CHANGED(*xBC*)

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

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

6.12.2.115 #define EMBER_NO_RESPONSE(*xC0*)

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

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

6.12.2.116 #define EMBER_DUPLICATE_ENTRY(*xC1*)

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

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

6.12.2.117 #define EMBER_NOT_PERMITTED(*xC2*)

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

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

6.12.2.118 #define EMBER_DISCOVERY_TIMEOUT(XC3)

The node has timed out during auto discovery response mode.

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

6.12.2.119 #define EMBER_DISCOVERY_ERROR(XC4)

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

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

6.12.2.120 #define EMBER_SECURITY_TIMEOUT(XC5)

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

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

6.12.2.121 #define EMBER_SECURITY_FAILURE(XC6)

Generic error code indicating a security failure.

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

6.12.2.122 #define EMBER_APPLICATION_ERROR_0(XF0)

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

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

6.12.2.123 #define EMBER_APPLICATION_ERROR_1(XF1)

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

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

6.12.2.124 #define EMBER_APPLICATION_ERROR_2(XF2)

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

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

6.12.2.125 #define EMBER_APPLICATION_ERROR_3(XF3)

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

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

6.12.2.126 #define EMBER_APPLICATION_ERROR_4(xF4)

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

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

6.12.2.127 #define EMBER_APPLICATION_ERROR_5(xF5)

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

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

6.12.2.128 #define EMBER_APPLICATION_ERROR_6(xF6)

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

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

6.12.2.129 #define EMBER_APPLICATION_ERROR_7(xF7)

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

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

6.12.2.130 #define EMBER_APPLICATION_ERROR_8(xF8)

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

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

6.12.2.131 #define EMBER_APPLICATION_ERROR_9(xF9)

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

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

6.12.2.132 #define EMBER_APPLICATION_ERROR_10(xFA)

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

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

6.12.2.133 #define EMBER_APPLICATION_ERROR_11(*xFB*)

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

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

6.12.2.134 #define EMBER_APPLICATION_ERROR_12(*xFc*)

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

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

6.12.2.135 #define EMBER_APPLICATION_ERROR_13(*xFD*)

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

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

6.12.2.136 #define EMBER_APPLICATION_ERROR_14(*xFE*)

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

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

6.12.2.137 #define EMBER_APPLICATION_ERROR_15(*xFF*)

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

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

6.12.3 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_STACKAPS_FRAME_COUNTER
- #define CREATOR_STACK_TRUST_CENTER
- #define CREATOR_STACK_NETWORK_MANAGEMENT
- #define CREATOR_STACK_PARENT_INFO
- #define CREATOR_STACK_PARENT_ADDITIONAL_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_MANAGEMEN-T
- #define CREATOR_MULTI_NETWORK_STACK_PARENT_INFO

- #define CREATOR_MULTI_NETWORK_STACK_NONCE_COUNTER
- #define CREATOR_MULTI_NETWORK_STACK_PARENT_ADDITIONAL_INFO
- #define CREATOR_STACK_RF4CE_DATA
- #define CREATOR_STACK_RF4CE_PAIRING_TABLE
- #define CREATOR_STACK_GP_DATA
- #define CREATOR_STACK_GP_PROXY_TABLE
- #define CREATOR_STACK_GP_SINK_TABLE
- #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
- #define CREATOR_STACK_ADDITIONAL_CHILD_DATA

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 ::APPLICATION_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

<i>region</i>	A name for the next region being located.
<i>address</i>	The address of the beginning of the next region.

Definition at line 60 of file [token-stack.h](#).

6.13.2.2 #define DEFINE_BASIC_TOKEN(*name*, *type*, ...)

Definition at line 97 of file [token-stack.h](#).

6.13.2.3 #define DEFINE_COUNTER_TOKEN(*name*, *type*, ...)

Definition at line 100 of file [token-stack.h](#).

6.13.2.4 #define DEFINE_INDEXED_TOKEN(*name*, *type*, *arraysize*, ...)

Definition at line 103 of file [token-stack.h](#).

6.13.2.5 #define DEFINE_FIXED_BASIC_TOKEN(*name*, *type*, *address*, ...)

Definition at line 106 of file [token-stack.h](#).

6.13.2.6 #define DEFINE_FIXED_COUNTER_TOKEN(*name*, *type*, *address*, ...)

Definition at line 110 of file [token-stack.h](#).

6.13.2.7 #define DEFINE_FIXED_INDEXED_TOKEN(*name*, *type*, *arraysize*, *address*, ...)

Definition at line 114 of file [token-stack.h](#).

6.13.2.8 #define DEFINE_MFG_TOKEN(*name*, *type*, *address*, ...)

Definition at line 118 of file [token-stack.h](#).

6.13.2.9 #define CREATOR_STACK_NVDATA_VERSION

Definition at line 148 of file [token-stack.h](#).

6.13.2.10 #define CREATOR_STACK_BOOT_COUNTER

Definition at line 149 of file [token-stack.h](#).

6.13.2.11 #define CREATOR_STACK_NONCE_COUNTER

Definition at line 150 of file [token-stack.h](#).

6.13.2.12 #define CREATOR_STACK_ANALYSIS_REBOOT

Definition at line 151 of file [token-stack.h](#).

6.13.2.13 #define CREATOR_STACK_KEYS

Definition at line 152 of file [token-stack.h](#).

6.13.2.14 #define CREATOR_STACK_NODE_DATA

Definition at line 153 of file [token-stack.h](#).

6.13.2.15 #define CREATOR_STACK_CLASSIC_DATA

Definition at line 154 of file [token-stack.h](#).

6.13.2.16 #define CREATOR_STACK_ALTERNATE_KEY

Definition at line 155 of file [token-stack.h](#).

6.13.2.17 #define CREATOR_STACKAPS_FRAME_COUNTER

Definition at line 156 of file [token-stack.h](#).

6.13.2.18 #define CREATOR_STACK_TRUST_CENTER

Definition at line 157 of file [token-stack.h](#).

6.13.2.19 #define CREATOR_STACK_NETWORK_MANAGEMENT

Definition at line 158 of file [token-stack.h](#).

6.13.2.20 #define CREATOR_STACK_PARENT_INFO

Definition at line 159 of file [token-stack.h](#).

6.13.2.21 #define CREATOR_STACK_PARENT_ADDITIONAL_INFO

Definition at line 160 of file [token-stack.h](#).

6.13.2.22 #define CREATOR_MULTI_NETWORK_STACK_KEYS

Definition at line 162 of file [token-stack.h](#).

6.13.2.23 #define CREATOR_MULTI_NETWORK_STACK_NODE_DATA

Definition at line 163 of file [token-stack.h](#).

6.13.2.24 #define CREATOR_MULTI_NETWORK_STACK_ALTERNATE_KEY

Definition at line 164 of file [token-stack.h](#).

6.13.2.25 #define CREATOR_MULTI_NETWORK_STACK_TRUST_CENTER

Definition at line 165 of file [token-stack.h](#).

6.13.2.26 #define CREATOR_MULTI_NETWORK_STACK_NETWORK_MANAGEMENT

Definition at line 166 of file [token-stack.h](#).

6.13.2.27 #define CREATOR_MULTI_NETWORK_STACK_PARENT_INFO

Definition at line 167 of file [token-stack.h](#).

6.13.2.28 #define CREATOR_MULTI_NETWORK_STACK_NONCE_COUNTER

Definition at line 171 of file [token-stack.h](#).

6.13.2.29 #define CREATOR_MULTI_NETWORK_STACK_PARENT_ADDITIONAL_INFO

Definition at line 172 of file [token-stack.h](#).

6.13.2.30 #define CREATOR_STACK_RF4CE_DATA

Definition at line 175 of file [token-stack.h](#).

6.13.2.31 #define CREATOR_STACK_RF4CE_PAIRING_TABLE

Definition at line 176 of file [token-stack.h](#).

6.13.2.32 #define CREATOR_STACK_GP_DATA

Definition at line 179 of file [token-stack.h](#).

6.13.2.33 #define CREATOR_STACK_GP_PROXY_TABLE

Definition at line 180 of file [token-stack.h](#).

6.13.2.34 #define CREATOR_STACK_GP_SINK_TABLE

Definition at line 181 of file [token-stack.h](#).

6.13.2.35 #define CREATOR_STACK_BINDING_TABLE

Definition at line 184 of file [token-stack.h](#).

6.13.2.36 #define CREATOR_STACK_CHILD_TABLE

Definition at line 185 of file [token-stack.h](#).

6.13.2.37 #define CREATOR_STACK_KEY_TABLE

Definition at line 186 of file [token-stack.h](#).

6.13.2.38 #define CREATOR_STACK_CERTIFICATE_TABLE

Definition at line 187 of file [token-stack.h](#).

6.13.2.39 #define CREATOR_STACK_ZLL_DATA

Definition at line 188 of file [token-stack.h](#).

6.13.2.40 #define CREATOR_STACK_ZLL_SECURITY

Definition at line 189 of file [token-stack.h](#).

6.13.2.41 #define CREATOR_STACK_ADDITIONAL_CHILD_DATA

Definition at line 190 of file [token-stack.h](#).

6.13.2.42 #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 218 of file [token-stack.h](#).

6.14 ZigBee Device Object

Functions

- [EmberStatus emberNetworkAddressRequest \(EmberEUI64 target, bool reportKids, uint8_t childStartIndex\)](#)
- [EmberStatus emberIeeeAddressRequest \(EmberNodeId target, bool reportKids, uint8_t childStartIndex, EmberApsOption options\)](#)
- [EmberStatus emberEnergyScanRequest \(EmberNodeId target, uint32_t scanChannels, uint8_t scanDuration, uint16_t scanCount\)](#)
- [EmberStatus emberSetNetworkManagerRequest \(EmberNodeId networkManager, uint32_t activeChannels\)](#)
- [EmberStatus emberChannelChangeRequest \(uint8_t channel\)](#)
- [EmberStatus emberSendDeviceAnnouncement \(void\)](#)
- [EmberStatus emberSendParentAnnouncement \(void\)](#)
- [uint8_t 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, bool reportKids, uint8_t childStartIndex)

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

Parameters

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

Returns

An [EmberStatus](#) value.

- [EMBER_SUCCESS](#) - The request was transmitted successfully.
- [EMBER_NO_BUFFERS](#) - Insufficient message buffers were available to construct the request.
- [EMBER_NETWORK_DOWN](#) - The node is not part of a network.
- [EMBER_NETWORK_BUSY](#) - Transmission of the request failed.

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

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

Parameters

<i>target</i>	The network address of the node.
<i>reportKids</i>	true to request that the target list their children in the response.
<i>childStart-Index</i>	The index of the first child to list in the response. Ignored if reportKids is false.
<i>options</i>	The options to use when sending the request. See emberSendUnicast() for a description.

Returns

An [EmberStatus](#) value.

- [EMBER_SUCCESS](#)
- [EMBER_NO_BUFFERS](#)
- [EMBER_NETWORK_DOWN](#)
- [EMBER_NETWORK_BUSY](#)

6.14.2.3 EmberStatus [emberEnergyScanRequest](#) (EmberNodeId *target*, uint32_t *scanChannels*, uint8_t *scanDuration*, uint16_t *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

<i>target</i>	The network address of the node to perform the scan.
<i>scan-Channels</i>	A mask of the channels to be scanned.
<i>scan-Duration</i>	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).
<i>scanCount</i>	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*, uint32_t *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

<i>network-Manager</i>	The network address of the network manager.
<i>active-Channels</i>	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 (*uint8_t 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

<i>channel</i>	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 EmberStatus emberSendParentAnnouncement (void)

Sends a broadcast for a ZDO Parent Announcement. Normally it is NOT required to call this as the stack automatically sends a Parent Announce when a Zigbee Router/Coordinator reboots, is in a joined or authenticated state and has atleast one device. However if the device wishes to re-send its Parent announcement they can use this call.

Returns

An [EmberStatus](#) value.

- [EMBER_SUCCESS](#)
- [EMBER_INVALID_CALL](#)

6.14.2.8 uint8_t 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 (bool broadcast, EmberEUI64 destEui64, EmberMessageBuffer 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 (bool 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

<code>broadcast</code>	If true, the destination address and pan id are both set to the broadcast address.
<code>destEui64</code>	The EUI64 of the target node. Ignored if the broadcast field is set to true.
<code>message</code>	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_BOOTLOAD_HANDLERS` in its `CONFIGURATION_HEADER`.

Parameters

<code>longId</code>	The EUI64 of the sending node.
<code>message</code>	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_HANDLERS in its CONFIGURATION_HEADER.

Parameters

<i>message</i>	The message that was sent.
<i>status</i>	EMBER_SUCCESS if the transmission was successful, or EMBER_DELIVERY_FAILED if not.

6.16 Event Scheduling

Macros

- #define __EVENT_H__
- #define emberEventControlSetInactive(control)
- #define emberEventControlGetActive(control)
- #define emberEventControlSetActive(control)
- #define EMBER_MAX_EVENT_CONTROL_DELAY_MS
- #define emberEventControlSetDelayMS(control, delay)
- #define EMBER_MAX_EVENT_CONTROL_DELAY_QS
- #define emberEventControlSetDelayQS(control, delay)
- #define EMBER_MAX_EVENT_CONTROL_DELAY_MINUTES
- #define emberEventControlSetDelayMinutes(control, delay)
- #define emberEventControlGetRemainingMS(control)
- #define emberTaskEnableIdling(allow)
- #define emberMarkTaskActive(taskid)

Functions

- void emEventControlSetActive (EmberEventControl *event)
- void emEventControlSetDelayMS (EmberEventControl *event, uint32_t delay)
- uint32_t emEventControlGetRemainingMS (EmberEventControl *event)
- void emberRunEvents (EmberEventData *events)
- void emberRunTask (EmberTaskId taskid)
- uint32_t emberMsToNextEvent (EmberEventData *events, uint32_t maxMs)
- uint32_t emberMsToNextEventExtended (EmberEventData *events, uint32_t maxMs, uint8_t *returnIndex)
- uint32_t emberMsToNextStackEvent (void)
- EmberTaskId emberTaskInit (EmberEventData *events)
- bool emberMarkTaskIdle (EmberTaskId taskid)
- void emTaskEnableIdling (bool 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 `emberEventControlSetActive()`. 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, 0)`.

The base time units for events are ticks. Each tick is approximately equal to a millisecond, but the true duration depends on the platform. The duration of a tick is $1000 / \text{::MILLISECOND_TICKS_PER_SECOND}$, where 1000 is the number of milliseconds per second and `::MILLISECOND_TICKS_PER_SECOND` is the platform-specific number of ticks per second. For example, `::MILLISECOND_TICKS_PER_SECOND` on the EM357 SoC is 1024, so each tick is therefore $1000 / 1024 = \sim 0.98$ milliseconds. Calling `emberEventControlSetDelayMS(someEvent, 100)` on the EM357 SoC will schedule the event for 100 ticks * (1000 milliseconds / 1024 ticks) = ~ 97.7 milliseconds. Note however that the accuracy of the base tick depends on your timer source. Further, the scheduled delay is the minimum delay. If `emberRunEvents` or `emberRunTask` are not called frequently enough, the actual delay may be longer than the scheduled delay.

Additionally, the APIs for quarter second and minute delays (`emberEventControlSetDelayQS` and `emberEventControlSetDelayMinutes`) use "binary" units. One quarter second is 256 ticks and one minute is 65536 ticks. Calling `emberEventControlSetDelayMinutes(someEvent, 3)` on the EM357 SoC will schedule the event for 3 minutes * (65536 ticks / minute) * (1000 milliseconds / 1024 ticks) = ~ 3.2 minutes. It is possible to avoid these binary units by using `emberEventControlSetDelayMS` and the various `MILLISECOND_TICKS_PER_XXX` multipliers. For example, calling `emberEventControlSetDelayMS(someEvent, 3 * MILLISECOND_TICKS_PER_MINUTE)` will delay for 3 minutes on any platform. Be aware of `EMBER_MAX_EVENT_CONTROL_DELAY_MS` when using this approach.

Following are some brief usage examples.

```
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,     signalEventHandler },
    { &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 125 of file [event.h](#).

6.16.2.2 #define emberEventControlSetActive(*control*)

Sets this [EmberEventControl](#) as inactive (no pending event).

Definition at line 129 of file [event.h](#).

6.16.2.3 #define emberEventControlGetActive(*control*)

Returns true if the event is active, false otherwise.

Definition at line 134 of file [event.h](#).

6.16.2.4 #define emberEventControlSetActive(*control*)

Sets this [EmberEventControl](#) to run at the next available opportunity.

Definition at line 140 of file [event.h](#).

6.16.2.5 #define EMBER_MAX_EVENT_CONTROL_DELAY_MS

The maximum delay that may be passed to [emberEventControlSetDelayMS](#).

Definition at line 151 of file [event.h](#).

6.16.2.6 #define emberEventControlSetDelayMS(*control*, *delay*)

Sets this [EmberEventControl](#) to run "delay" milliseconds in the future. NOTE: To avoid rollover errors in event calculation, the delay must be less than [EMBER_MAX_EVENT_CONTROL_DELAY_MS](#).

Definition at line 157 of file [event.h](#).

6.16.2.7 #define EMBER_MAX_EVENT_CONTROL_DELAY_QS

The maximum delay that may be passed to [emberEventControlSetDelayQS](#).

Definition at line 169 of file [event.h](#).

6.16.2.8 #define emberEventControlSetDelayQS(*control*, *delay*)

Sets this [EmberEventControl](#) to run "delay" quarter seconds in the future. The 'quarter seconds' are actually 256 milliseconds long. NOTE: To avoid rollover errors in event calculation, the delay must be less than [EMBER_MAX_EVENT_CONTROL_DELAY_QS](#).

Definition at line 176 of file [event.h](#).

6.16.2.9 #define EMBER_MAX_EVENT_CONTROL_DELAY_MINUTES

The maximum delay that may be passed to [emberEventControlSetDelayMinutes](#).

Definition at line 182 of file [event.h](#).

6.16.2.10 #define emberEventControlSetDelayMinutes(*control*, *delay*)

Sets this [EmberEventControl](#) to run "delay" minutes in the future. The 'minutes' are actually 65536 (0x10000) milliseconds long. NOTE: To avoid rollover errors in event calculation, the delay must be less than [EMBER_MAX_EVENT_CONTROL_DELAY_MINUTES](#).

Definition at line 189 of file [event.h](#).

6.16.2.11 #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 195 of file [event.h](#).

6.16.2.12 #define emberTaskEnableIdling(*allow*)

Call this to indicate that an application supports processor idling.

Definition at line 261 of file [event.h](#).

6.16.2.13 #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 269 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*, uint32_t *delay*)

Sets this [EmberEventControl](#) to run "delay" milliseconds in the future. NOTE: To avoid rollover errors in event calculation, the delay must be less than [EMBER_MAX_EVENT_CONTROL_DELAY_MS](#).

6.16.3.3 uint32_t 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.

6.16.3.4 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.

6.16.3.5 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\(\)](#).

6.16.3.6 uint32_t emberMsToNextEvent (EmberEventData * *events*, uint32_t *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.

6.16.3.7 uint32_t emberMsToNextEventExtended (EmberEventData * *events*, uint32_t *maxMs*, uint8_t * *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.

6.16.3.8 uint32_t emberMsToNextStackEvent (void)

Returns the number of milliseconds before the next stack event is scheduled to expire.

6.16.3.9 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.

6.16.3.10 bool 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.

6.16.3.11 void emTaskEnableIdling (bool *allow*)**6.16.3.12 void emMarkTaskActive (EmberTaskId *taskid*)**

6.17 Multi-Network Manager

Functions

- `uint8_t emberGetCurrentNetwork (void)`
- `EmberStatus emberSetCurrentNetwork (uint8_t index)`
- `uint8_t emberGetCallbackNetwork (void)`

6.17.1 Detailed Description

See [multi-network.h](#) for source code.

6.17.2 Function Documentation

6.17.2.1 `uint8_t emberGetCurrentNetwork (void)`

Returns the current network index.

6.17.2.2 `EmberStatus emberSetCurrentNetwork (uint8_t index)`

Sets the current network.

Parameters

<code>index</code>	The network index.
--------------------	--------------------

Returns

`EMBER_INDEX_OUT_OF_RANGE` if the index does not correspond to a valid network, and `EMBER_SUCCESS` otherwise.

6.17.2.3 `uint8_t 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.

6.18 Ember ZigBee Light Link (ZLL) APIs and Handlers

Functions

- `EmberStatus emberZllFormNetwork (EmberZllNetwork *networkInfo, int8_t radio-TxPower)`
- `EmberStatus emberZllJoinTarget (const EmberZllNetwork *targetNetworkInfo)`
- `EmberStatus emberZllSetInitialSecurityState (const EmberKeyData *networkKey, const EmberZllInitialSecurityState *securityState)`
- `EmberStatus emberZllStartScan (uint32_t channelMask, int8_t radioPowerForScan, EmberNodeType nodeType)`
- `EmberStatus emberZllSetRxOnWhenIdle (uint16_t durationMs)`
- `void emberZllNetworkFoundHandler (const EmberZllNetwork *networkInfo, const EmberZllDeviceInfoRecord *deviceInfo)`
- `void emberZllScanCompleteHandler (EmberStatus status)`
- `void emberZllAddressAssignmentHandler (const EmberZllAddressAssignment *address-Info)`
- `void emberZllTouchLinkTargetHandler (const EmberZllNetwork *networkInfo)`
- `void emberZllGetTokenStackZllData (EmberTokTypeStackZllData *token)`
- `void emberZllGetTokenStackZllSecurity (EmberTokTypeStackZllSecurity *token)`
- `void emberZllGetTokensStackZll (EmberTokTypeStackZllData *data, EmberTok-TypeStackZllSecurity *security)`
- `void emberZllSetTokenStackZllData (EmberTokTypeStackZllData *token)`
- `bool emberIsZllNetwork (void)`
- `void emberZllSetNonZllNetwork (void)`
- `EmberZllPolicy emberZllGetPolicy (void)`
- `EmberStatus emberZllSetPolicy (EmberZllPolicy policy)`

6.18.1 Detailed Description

See [zll-api.h](#) for source code.

6.18.2 Function Documentation

6.18.2.1 `EmberStatus emberZllFormNetwork (EmberZllNetwork * networkInfo, int8_t radioTxPower)`

This will set the device type as a router or end device (depending on the passed `nodeType`) and setup a ZLL commissioning network with the passed parameters. If `panId` is `0xFFFF`, a random PAN ID will be generated. If `extendedPanId` is set to all F's, then a random extended pan ID will be generated. If `channel` is `0xFF`, then channel 11 will be used. If all F values are passed for PAN Id or Extended PAN ID then the randomly generated values will be returned in the passed structure.

Parameters

<code>networkInfo</code>	A pointer to an <code>EmberZllNetwork</code> struct indicating the network parameters to use when forming the network. If random values are requested, the stack's randomly generated values will be returned in the structure.
<code>radioTx-Power</code>	the radio output power at which a node is to operate.

Returns

An [EmberStatus](#) value indicating whether the operation succeeded, or why it failed.

6.18.2.2 EmberStatus emberZllJoinTarget (const EmberZllNetwork * targetNetworkInfo)

This call will cause the device to send a NWK start or join to the target device and cause the remote AND local device to start operating on a network together. If the local device is a factory new device then it will send a ZLL NWK start to the target requesting that the target generate new network parameters. If the device is not factory new then the local device will send a NWK join request using the current network parameters.

Parameters

<i>targetNetworkInfo</i>	A pointer to an EmberZllNetwork structure that indicates the info about what device to send the NWK start/join request to. This information must have previously been returned from a ZLL scan.
--------------------------	---

Returns

An [EmberStatus](#) value indicating whether the operation succeeded, or why it failed.

6.18.2.3 EmberStatus emberZllSetInitialSecurityState (const EmberKeyData * networkKey, const EmberZllInitialSecurityState * securityState)

This call will cause the device to setup the security information used in its network. It must be called prior to forming, starting, or joining a network.

Parameters

<i>networkKey</i>	is a pointer to an EmberKeyData structure containing the value for the network key. If the value is set to all F's, then a random network key will be generated.
<i>securityState</i>	The security configuration to be set.

Returns

An [EmberStatus](#) value indicating whether the operation succeeded, or why it failed.

6.18.2.4 EmberStatus emberZllStartScan (uint32_t channelMask, int8_t radioPowerForScan, EmberNodeType nodeType)

This call will initiate a ZLL network scan on all the specified channels. Results will be returned in [emberZllNetworkFoundHandler\(\)](#).

Parameters

<i>channelMask</i>	indicating the range of channels to scan.
<i>radioPowerForScan</i>	the radio output power used for the scan requests.
<i>nodeType</i>	the the node type of the local device.

Returns

An [EmberStatus](#) value indicating whether the operation succeeded, or why it failed.

6.18.2.5 EmberStatus emberZllSetRxOnWhenIdle (*uint16_t durationMs*)

This call will change the mode of the radio so that the receiver is on when the device is idle. Changing the idle mode permits the application to communicate with the target of a touch link before the network is established. The receiver will remain on until the duration elapses, the duration is set to zero, or [emberZllJoinTarget](#) is called.

Parameters

<i>durationMs</i>	The duration in milliseconds to leave the radio on.
-------------------	---

Returns

An [EmberStatus](#) value indicating whether the operation succeeded or why it failed.

6.18.2.6 void emberZllNetworkFoundHandler (*const EmberZllNetwork * networkInfo*, *const EmberZllDeviceInfoRecord * deviceInfo*)

This call is fired when a ZLL network scan finds a ZLL network. The network information will be returned to the application for processing.

Parameters

<i>networkInfo</i>	is a pointer to an EmberZllNetwork struct containing the Zigbee and ZLL specific information about the discovered network.
<i>deviceInfo</i>	is a pointer to an EmberZllDeviceInfoRecord struct containing the device specific info. This pointer may be NULL, indicating the device has either 0 sub-devices, or more than 1 sub-devices.

6.18.2.7 void emberZllScanCompleteHandler (*EmberStatus status*)

This call is fired when a ZLL network scan is complete.

Parameters

<i>status</i>	An EmberStatus value indicating whether the operation succeeded, or why it failed. If the status is not EMBER_SUCCESS , the application should not attempt to start or join a network returned via emberZllNetworkFoundHandler .
---------------	--

6.18.2.8 void emberZllAddressAssignmentHandler (*const EmberZllAddressAssignment * addressInfo*)

This call is fired when network and group addresses are assigned to a remote mode in a network start or network join request.

Parameters

<i>addressInfo</i>	is a pointer to an EmberZllAddressAssignment struct containing the address assignment information.
--------------------	--

6.18.2.9 void emberZllTouchLinkTargetHandler (const EmberZllNetwork * *networkInfo*)

This call is fired when the device is a target of a touch link.

Parameters

<i>networkInfo</i>	is a pointer to an EmberZllNetwork struct containing the Zigbee and ZLL specific information about the initiator.
--------------------	---

6.18.2.10 void emberZllGetTokenStackZllData (EmberTokTypeStackZllData * *token*)

This call reads the Zll Stack data token.

6.18.2.11 void emberZllGetTokenStackZllSecurity (EmberTokTypeStackZllSecurity * *token*)

This call reads the Zll Stack security token.

6.18.2.12 void emberZllGetTokensStackZll (EmberTokTypeStackZllData * *data*, EmberTokTypeStackZllSecurity * *security*)

This call reads both the Zll Stack data and security tokens.

6.18.2.13 void emberZllSetTokenStackZllData (EmberTokTypeStackZllData * *token*)

This call sets the Zll Stack data token.

6.18.2.14 bool emberIsZllNetwork (void)

This call returns whether or not the network is a ZLL network.

6.18.2.15 void emberZllSetNonZllNetwork (void)

This call will alter the ZLL data token to reflect the fact that the network is non-ZLL.

6.18.2.16 EmberZllPolicy emberZllGetPolicy (void)

This call will get the policy that enables or disables ZLL processing.

6.18.2.17 EmberStatus `emberZllSetPolicy` (`EmberZllPolicy policy`)

This call will set the policy to enable or disable ZLL processing.

6.19 Ember ZigBee Light Link (ZLL) Data Types

Data Structures

- struct [EmberZllSecurityAlgorithmData](#)
Information about the ZLL security being and how to transmit the network key to the device securely.
- struct [EmberZllNetwork](#)
Information about the ZLL network and specific device that responded to a ZLL scan request.
- struct [EmberZllDeviceInfoRecord](#)
Information discovered during a ZLL scan about the ZLL device's endpoint information.
- struct [EmberZllAddressAssignment](#)
Network and group address assignment information.
- struct [EmberZllInitialSecurityState](#)
This describes the Initial Security features and requirements that will be used when forming or joining ZigBee Light Link networks.
- struct [EmberTokTypeStackZllData](#)
- struct [EmberTokTypeStackZllSecurity](#)

ZigBee Light Link Types

- enum [EmberZllState](#) {
 EMBER_ZLL_STATE_NONE, EMBER_ZLL_STATE_FACTORY_NEW, EMBER_ZLL_STATE_ADDRESS_ASSIGNMENT_CAPABLE, EMBER_ZLL_STATE_LINK_INITIATOR,
 EMBER_ZLL_STATE_LINK_PRIORITY_REQUEST, EMBER_ZLL_STATE_PROFILE_INTEROP, EMBER_ZLL_STATE_NON_ZLL_NETWORK }
- enum [EmberZllKeyIndex](#) { EMBER_ZLL_KEY_INDEX_DEVELOPMENT, EMBER_ZLL_KEY_INDEX_MASTER, EMBER_ZLL_KEY_INDEX_CERTIFICATION }
- enum [EmberZllPolicy](#) { EMBER_ZLL_POLICY_ENABLED, EMBER_ZLL_POLICY_DISABLED }
- #define EMBER_ZLL_PRIMARY_CHANNEL_MASK
- #define EMBER_ZLL_SECONDARY_CHANNEL_MASK
- #define EMBER_ZLL_NULL_NODE_ID
- #define EMBER_ZLL_MIN_NODE_ID
- #define EMBER_ZLL_MAX_NODE_ID
- #define EMBER_ZLL_NULL_GROUP_ID
- #define EMBER_ZLL_MIN_GROUP_ID
- #define EMBER_ZLL_MAX_GROUP_ID
- #define EMBER_ZLL_CLUSTER_ID
- #define EMBER_ZLL_PROFILE_ID
- #define EMBER_ZLL_KEY_MASK_DEVELOPMENT
- #define EMBER_ZLL_KEY_MASK_MASTER
- #define EMBER_ZLL_KEY_MASK_CERTIFICATION
- #define EMBER_ZLL_CERTIFICATION_ENCRYPTION_KEY
- #define EMBER_ZLL_CERTIFICATION_PRECONFIGURED_LINK_KEY

6.19.1 Detailed Description

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

6.19.2 Macro Definition Documentation

6.19.2.1 `#define EMBER_ZLL_PRIMARY_CHANNEL_MASK`

The list of primary ZLL channels.

Definition at line [28](#) of file [zll-types.h](#).

6.19.2.2 `#define EMBER_ZLL_SECONDARY_CHANNEL_MASK`

The list of secondary ZLL channels.

Definition at line [36](#) of file [zll-types.h](#).

6.19.2.3 `#define EMBER_ZLL_NULL_NODE_ID`

A distinguished network identifier in the ZLL network address space that indicates no free network identifiers were assigned to the device.

Definition at line [53](#) of file [zll-types.h](#).

6.19.2.4 `#define EMBER_ZLL_MIN_NODE_ID`

The minimum network identifier in the ZLL network address space.

Definition at line [58](#) of file [zll-types.h](#).

6.19.2.5 `#define EMBER_ZLL_MAX_NODE_ID`

The maximum network identifier in the ZLL network address space.

Definition at line [63](#) of file [zll-types.h](#).

6.19.2.6 `#define EMBER_ZLL_NULL_GROUP_ID`

A distinguished group identifier in the ZLL group address space that indicates no free group identifiers were assigned to the device.

Definition at line [69](#) of file [zll-types.h](#).

6.19.2.7 `#define EMBER_ZLL_MIN_GROUP_ID`

The minimum group identifier in the ZLL group address space.

Definition at line [74](#) of file [zll-types.h](#).

6.19.2.8 #define EMBER_ZLL_MAX_GROUP_ID

The maximum group identifier in the ZLL group address space.

Definition at line 79 of file [zll-types.h](#).

6.19.2.9 #define EMBER_ZLL_CLUSTER_ID

The ZigBee Light Link Commissioning cluster ID.

Definition at line 163 of file [zll-types.h](#).

6.19.2.10 #define EMBER_ZLL_PROFILE_ID

The ZigBee Light Link Profie ID.

Definition at line 168 of file [zll-types.h](#).

6.19.2.11 #define EMBER_ZLL_KEY_MASK_DEVELOPMENT

Key encryption bitmask corresponding to encryption key index [EMBER_ZLL_KEY_INDEX_DEVELOPMENT](#).

Definition at line 193 of file [zll-types.h](#).

6.19.2.12 #define EMBER_ZLL_KEY_MASK_MASTER

Key encryption bitmask corresponding to encryption key index [EMBER_ZLL_KEY_INDEX_MASTER](#).

Definition at line 199 of file [zll-types.h](#).

6.19.2.13 #define EMBER_ZLL_KEY_MASK_CERTIFICATION

Key encryption bitmask corresponding to encryption key index [EMBER_ZLL_KEY_INDEX_CERTIFICATION](#).

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

6.19.2.14 #define EMBER_ZLL_CERTIFICATION_ENCRYPTION_KEY

Encryption key for use during development and certification in conjunction with [EMBER_ZLL_KEY_INDEX_CERTIFICATION](#).

Definition at line 211 of file [zll-types.h](#).

6.19.2.15 #define EMBER_ZLL_CERTIFICATION_PRECONFIGURED_LINK_KEY

Pre-configured link key for use during development and certification in conjunction with [EMBER_ZLL_KEY_INDEX_CERTIFICATION](#).

Definition at line 219 of file [zll-types.h](#).

6.19.3 Enumeration Type Documentation

6.19.3.1 enum EmberZllState

A bitmask indicating the state of the ZLL device. These map directly to the ZLL information field in the scan response.

Enumerator:

EMBER_ZLL_STATE_NONE No state.

EMBER_ZLL_STATE_FACTORY_NEW The device is factory new.

EMBER_ZLL_STATE_ADDRESS_ASSIGNMENT_CAPABLE The device is capable of assigning addresses to other devices.

EMBER_ZLL_STATE_LINK_INITIATOR The device is initiating a link operation.

EMBER_ZLL_STATE_LINK_PRIORITY_REQUEST The device is requesting link priority.

EMBER_ZLL_STATE_PROFILE_INTEROP The device is a ZigBee 3.0 device.

EMBER_ZLL_STATE_NON_ZLL_NETWORK The device is on a non-ZLL network.

Definition at line 86 of file [zll-types.h](#).

6.19.3.2 enum EmberZllKeyIndex

The key encryption algorithms supported by the stack.

Enumerator:

EMBER_ZLL_KEY_INDEX_DEVELOPMENT Key encryption algorithm for use during development.

EMBER_ZLL_KEY_INDEX_MASTER Key encryption algorithm shared by all certified devices.

EMBER_ZLL_KEY_INDEX_CERTIFICATION Key encryption algorithm for use during development and certification.

Definition at line 175 of file [zll-types.h](#).

6.19.3.3 enum EmberZllPolicy

This enumeration indicates whether or not the stack processes ZLL messages.

Enumerator:

EMBER_ZLL_POLICY_ENABLED Indicates that ZLL processing is enabled.

EMBER_ZLL_POLICY_DISABLED Indicates that ZLL processing is disabled.

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

6.20 Manufacturing and Functional Test Library

Functions

- `EmberStatus mfglibStart (void(*mfglibRxCallback)(uint8_t *packet, uint8_t linkQuality, int8_t rss))`
- `EmberStatus mfglibEnd (void)`
- `EmberStatus mfglibStartTone (void)`
- `EmberStatus mfglibStopTone (void)`
- `EmberStatus mfglibStartStream (void)`
- `EmberStatus mfglibStopStream (void)`
- `EmberStatus mfglibSendPacket (uint8_t *packet, uint16_t repeat)`
- `EmberStatus mfglibSetChannel (uint8_t chan)`
- `uint8_t mfglibGetChannel (void)`
- `EmberStatus mfglibSetPower (uint16_t txPowerMode, int8_t power)`
- `int8_t mfglibGetPower (void)`
- `void mfglibSetSynOffset (int8_t synOffset)`
- `int8_t mfglibGetSynOffset (void)`
- `void mfglibTestContModCal (uint8_t channel, uint32_t duration)`

6.20.1 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.

6.20.2 Function Documentation

6.20.2.1 `EmberStatus mfglibStart (void(*)(uint8_t *packet, uint8_t linkQuality, int8_t rss) 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

<i>mfglibRx-Callback</i>	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.

6.20.2.2 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.

6.20.2.3 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.

6.20.2.4 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.

6.20.2.5 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.

6.20.2.6 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.

6.20.2.7 EmberStatus mfglibSendPacket (`uint8_t * packet, uint16_t 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 $((((\text{uint16_t})\text{packet}) \& 1) == 0)$ 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 `uint16_t`.)

Parameters

<code>packet</code>	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 packet-Contents. 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.
<code>repeat</code>	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 STRE-AM test is running.

6.20.2.8 EmberStatus mfglibSetChannel (`uint8_t 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

<code>chan</code>	Valid values depend upon the radio used.
-------------------	--

Returns

One of the following:

- [EMBER_SUCCESS](#) if the channel has been set.
- [EMBER_PHY_INVALID_CHANNEL](#) if the channel requested is invalid.

- **EMBER_ERR_FATAL** if the mfg test mode is not available or TONE or STRE-AM test is running.

6.20.2.9 uint8_t mfglibGetChannel (void)

Returns the current radio channel, as previously set via [mfglibSetChannel\(\)](#).

Application Usage:

Use this function to get current channel.

Returns

Current channel.

6.20.2.10 EmberStatus mfglibSetPower (uint16_t txPowerMode, int8_t 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

<i>txPower- Mode</i>	boost mode or external PA.
<i>power</i>	Power in units of dBm, which can be negative.

Returns

One of the following:

- **EMBER_SUCCESS** if the power has been set.
- **EMBER_PHY_INVALID_POWER** if the power requested is invalid.
- **EMBER_ERR_FATAL** if the mfg test mode is not available or TONE or STRE-AM test is running.

6.20.2.11 int8_t 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.

6.20.2.12 void mfplibSetSynOffset (int8_t 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

<i>synOffset</i>	the number of 11.7kHz steps to offset the carrier frequency (may be negative)
------------------	---

6.20.2.13 int8_t mfplibGetSynOffset (void)

get the current synth offset in 11.7kHz steps. see [mfplibSetSynOffset\(\)](#) for details

Returns

the synth offset in 11.7kHz steps

6.20.2.14 void mfplibTestContModCal (uint8_t channel, uint32_t 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

<i>channel</i>	Selects the channel to transmit on.
<i>duration</i>	Duration in ms, 0 == infinite.

Returns

None.

6.21 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](#) (uint8_t *start, uint8_t *end)
- void [emberDebugBinaryPrintf](#) (PGM_P formatString,...)
- void [emDebugSendVuartMessage](#) (uint8_t *buff, uint8_t len)
- void [emberDebugError](#) (EmberStatus code)
- bool [emberDebugReportOff](#) (void)
- void [emberDebugReportRestore](#) (bool state)
- void [emberDebugPrintf](#) (PGM_P formatString,...)

6.21.1 Detailed Description

EmberZNet debugging utilities. See [ember-debug.h](#) for source code.

6.21.2 Macro Definition Documentation

6.21.2.1 #define NO_DEBUG

Definition at line [20](#) of file [ember-debug.h](#).

6.21.2.2 #define BASIC_DEBUG

Definition at line [21](#) of file [ember-debug.h](#).

6.21.2.3 #define FULL_DEBUG

Definition at line [22](#) of file [ember-debug.h](#).

6.21.2.4 #define emberDebugInit(port)

This function is obsolete and no longer required to initialize the debug system.

Parameters

<i>port</i>	Ignored because the port used for debug communication is automatically determined for each platform.
-------------	--

Definition at line [31](#) of file [ember-debug.h](#).

6.21.3 Function Documentation

6.21.3.1 void emberDebugAssert (PGM_P *filename*, int *linenumber*)

Prints the filename and line number to the debug serial port.

Parameters

<i>filename</i>	The name of the file where the assert occurred.
<i>linenumber</i>	The line number in the file where the assert occurred.

6.21.3.2 void emberDebugMemoryDump (uint8_t * *start*, uint8_t * *end*)

Prints the contents of RAM to the debug serial port.

Parameters

<i>start</i>	The start address of the block of RAM to dump.
<i>end</i>	The end address of the block of RAM to dump (address of the last byte).

6.21.3.3 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 [emberDebugPrintf\(\)](#). The format string must contain only these conversion specification characters:

- B - uint8_t value.
- W - uint16_t value, printed least significant byte first.
- D - uint32_t 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 uint8_t argument.
- xxxf - pointer to Flash (PGM_P), length is xxx (max 255).
- lf - pointer to Flash (PGM_P), length is uint8_t argument.
- b - EmberMessageBuffer.

Examples:

```
emberDebugBinaryPrintf("BWD", status, panId, channelMask)
;
emberDebugBinaryPrintf("F8p", "string example", eui64);
emberDebugBinaryPrintf("lp64fb", length, bytes, dataTable
, buffer);
```

Parameters

<i>formatString</i>	A string of conversion specification characters describing the arguments to be printed.
...	The arguments to be printed.

6.21.3.4 void emDebugSendVuartMessage (*uint8_t *buff, uint8_t len*)

internal debug command used by the HAL to send vuart data out the the debug channel

Parameters

<i>buff</i>	pointer to the data to send
<i>len</i>	length of the data to send

6.21.3.5 void emberDebugError (*EmberStatus code*)

Prints an [EmberStatus](#) return code to the serial port.

Parameters

<i>code</i>	The EmberStatus code to print.
-------------	--

6.21.3.6 bool emberDebugReportOff (*void*)

Turns off all debug output.

Returns

The current state (true for on, false for off).

6.21.3.7 void emberDebugReportRestore (*bool state*)

Restores the state of the debug output.

Parameters

<i>state</i>	The state returned from emberDebugReportOff() . This is done so that debug output is not blindly turned on.
--------------	---

6.21.3.8 void emberDebugPrintf (*PGM_P formatString, ...*)

Prints text debug messages.

Parameters

<i>formatString</i>	Takes the following:
---------------------	----------------------

<code>%%</code>	Percent sign
<code>%c</code>	Single-byte char
<code>%s</code>	RAM string
<code>%p</code>	Flash string (does not follow the printf standard)
<code>%u</code>	Two-byte unsigned decimal
<code>%d</code>	Two-byte signed decimal
<code>%x, %%2x, %%4x</code>	1-, 2-, 4-byte hex value (always 0 padded; does not follow the printf standard)

6.22 Ember ZigBee RF4CE Stack API Reference

Modules

- [Ember ZigBee RF4CE APIs and Handlers](#)
- [Ember ZigBee RF4CE Data Types](#)
- [Packet Buffers](#)
- [Configuration](#)
- [Status Codes](#)
- [Stack Tokens](#)
- [Bootloader](#)
- [Event Scheduling](#)
- [Multi-Network Manager](#)
- [Manufacturing and Functional Test Library](#)
- [Debugging Utilities](#)

6.22.1 Detailed Description

6.23 Ember ZigBee RF4CE APIs and Handlers

Functions

- `EmberStatus emberRf4ceSetPairingTableEntry (uint8_t pairingIndex, EmberRf4cePairingTableEntry *entry)`
- `EmberStatus emberRf4ceGetPairingTableEntry (uint8_t pairingIndex, EmberRf4cePairingTableEntry *entry)`
- `EmberStatus emberRf4ceSetApplicationInfo (EmberRf4ceApplicationInfo *appInfo)`
- `EmberStatus emberRf4ceGetApplicationInfo (EmberRf4ceApplicationInfo *appInfo)`
- `uint8_t emberRf4ceGetBaseChannel (void)`
- `EmberStatus emberRf4ceKeyUpdate (uint8_t pairingIndex, EmberKeyData *key)`
- `EmberStatus emberRf4ceSend (uint8_t pairingIndex, uint8_t profileId, uint16_t vendorId, EmberRf4ceTxOption txOptions, uint8_t messageTag, uint8_t messageLength, uint8_t *message)`
- `void emberRf4ceMessageSentHandler (EmberStatus status, uint8_t pairingIndex, EmberRf4ceTxOption txOptions, uint8_t profileId, uint16_t vendorId, uint8_t messageTag, uint8_t messageLength, uint8_t *message)`
- `void emberRf4ceIncomingMessageHandler (uint8_t pairingIndex, uint8_t profileId, uint16_t vendorId, EmberRf4ceTxOption txOptions, uint8_t messageLength, uint8_t *message)`
- `uint8_t emberRf4ceGetMaxPayload (uint8_t pairingIndex, EmberRf4ceTxOption txOptions)`
- `EmberStatus emberRf4ceStart (EmberRf4ceNodeCapabilities capabilities, EmberRf4ceVendorInfo *vendorInfo, int8_t power)`
- `EmberStatus emberRf4ceStop (void)`
- `EmberStatus emberRf4ceDiscovery (EmberPanId panId, EmberNodeId nodeId, uint8_t searchDevType, uint16_t discDuration, uint8_t maxDiscRepetitions, uint8_t discProfileIdListLength, uint8_t *discProfileIdList)`
- `void emberRf4ceDiscoveryCompleteHandler (EmberStatus status)`
- `bool emberRf4ceDiscoveryRequestHandler (EmberEUI64 srcIeeeAddr, uint8_t nodeCapabilities, EmberRf4ceVendorInfo *vendorInfo, EmberRf4ceApplicationInfo *appInfo, uint8_t searchDevType, uint8_t rxLinkQuality)`
- `bool emberRf4ceDiscoveryResponseHandler (bool atCapacity, uint8_t channel, EmberPanId panId, EmberEUI64 srcIeeeAddr, uint8_t nodeCapabilities, EmberRf4ceVendorInfo *vendorInfo, EmberRf4ceApplicationInfo *appInfo, uint8_t rxLinkQuality, uint8_t discRequestLqi)`
- `EmberStatus emberRf4ceEnableAutoDiscoveryResponse (uint16_t duration)`
- `void emberRf4ceAutoDiscoveryResponseCompleteHandler (EmberStatus status, EmberEUI64 srcIeeeAddr, uint8_t nodeCapabilities, EmberRf4ceVendorInfo *vendorInfo, EmberRf4ceApplicationInfo *appInfo, uint8_t searchDevType)`
- `EmberStatus emberRf4cePair (uint8_t channel, EmberPanId panId, EmberEUI64 ieeeAddr, uint8_t keyExchangeTransferCount)`
- `void emberRf4cePairCompleteHandler (EmberStatus status, uint8_t pairingIndex, EmberRf4ceVendorInfo *vendorInfo, EmberRf4ceApplicationInfo *appInfo)`
- `bool emberRf4cePairRequestHandler (EmberStatus status, uint8_t pairingIndex, EmberEUI64 srcIeeeAddr, uint8_t nodeCapabilities, EmberRf4ceVendorInfo *vendorInfo, EmberRf4ceApplicationInfo *appInfo, uint8_t keyExchangeTransferCount)`
- `EmberStatus emberRf4ceUnpair (uint8_t pairingIndex)`
- `void emberRf4ceUnpairHandler (uint8_t pairingIndex)`
- `void emberRf4ceUnpairCompleteHandler (uint8_t pairingIndex)`

- [EmberStatus emberRf4ceSetPowerSavingParameters](#) (uint32_t dutyCycle, uint32_t activePeriod)
- [EmberStatus emberRf4ceSetFrequencyAgilityParameters](#) (uint8_t rssiWindowSize, uint8_t channelChangeReads, int8_t rssiThreshold, uint16_t readInterval, uint8_t readDuration)
- [EmberStatus emberRf4ceSetDiscoveryLqiThreshold](#) (uint8_t threshold)

6.23.1 Detailed Description

See [rf4ce-api.h](#) for source code.

6.23.2 Function Documentation

6.23.2.1 EmberStatus emberRf4ceSetPairingTableEntry (*uint8_t pairingIndex,* *EmberRf4cePairingTableEntry * entry*)

Sets the pairing table entry corresponding to the passed index.

Parameters

<i>pairingIndex</i>	The index of the pairing table entry to be set.
<i>entry</i>	A pointer to an EmberRf4cePairingTableEntry struct to be copied into the pairing table at the passed index. If the passed pointer is NULL, the stack will delete the entry stored at the passed pairing index.

Returns

An [EmberStatus](#) value of [EMBER_SUCCESS](#) if the pairing table entry was successfully set. An [EmberStatus](#) value of [EMBER_INVALID_CALL](#) if the current network is not an RF4CE network, or the node hasn't been started, or the node is currently busy performing some discovery or pairing process. An [EmberStatus](#) value of [EMBER_INDEX_OUT_OF_RANGE](#) if the specified index is outside the valid range.

6.23.2.2 EmberStatus emberRf4ceGetPairingTableEntry (*uint8_t pairingIndex,* *EmberRf4cePairingTableEntry * entry*)

Retrieves the pairing table entry stored at the passed index.

Parameters

<i>pairingIndex</i>	The index of the requested pairing table entry.
<i>entry</i>	A pointer to an EmberRf4cePairingTableEntry struct where the requested pairing table entry will be copied.

Returns

An [EmberStatus](#) value of [EMBER_SUCCESS](#) if the requested entry was successfully copied in the passed [EmberRf4cePairingTableEntry](#) struct. An [EmberStatus](#) value of [EMBER_INVALID_CALL](#) if the current network is not an RF4CE network, or the node hasn't been started or the node is currently busy performing some discovery or

pairing process. An [EmberStatus](#) value of [EMBER_INDEX_OUT_OF_RANGE](#) if the specified index is outside the valid range.

6.23.2.3 EmberStatus emberRf4ceSetApplicationInfo (EmberRf4ceApplicationInfo * *appInfo*)

Sets the application information of the node.

Parameters

<i>appInfo</i>	A pointer to an EmberRf4ceApplicationInfo containing the application information to be set at the node.
----------------	---

Returns

An [EmberStatus](#) value of [EMBER_SUCCESS](#) if the application information was successfully set. An [EmberStatus](#) value of [EMBER_INVALID_CALL](#) if the stack failed to set the application information. This API can be called before and after the node has been started.

6.23.2.4 EmberStatus emberRf4ceGetApplicationInfo (EmberRf4ceApplicationInfo * *appInfo*)

Retrieves the application information of the node.

Parameters

<i>appInfo</i>	A pointer to an EmberRf4ceApplicationInfo where the stack will copy the application information of the node.
----------------	--

Returns

An [EmberStatus](#) value of [EMBER_SUCCESS](#) if the application information was successfully retrieved. An [EmberStatus](#) value of [EMBER_INVALID_CALL](#) if the stack failed to retrieve the application information.

6.23.2.5 uint8_t emberRf4ceGetBaseChannel (void)

Retrieves the device base channel as specified in the RF4CE specs.

Returns

The base channel of the device.

6.23.2.6 EmberStatus emberRf4ceKeyUpdate (uint8_t *pairingIndex*, EmberKeyData * *key*)

An API for updating the link key of a pairing table entry.

Parameters

<i>pairingIndex</i>	The index of the pairing table entry to be updated.
<i>key</i>	A pointer to an EmberKeyData struct containing the new key.

Returns

An [EmberStatus](#) value of EMBER_SUCCESS if the pairing table entry was successfully updated. An [EmberStatus](#) value of EMBER_INDEX_OUT_OF_RANGE if the specified index is outside the valid range. An [EmberStatus](#) value of EMBER_INVALID_CALL if the current network is not an RF4CE network, or the node hasn't been started, or the node is currently busy performing some discovery or pairing process, or if the passed pairing index does not correspond to an active secured pairing link and/or the node does not support security.

6.23.2.7 EmberStatus emberRf4ceSend (uint8_t *pairingIndex*, uint8_t *profileId*, uint16_t *vendorId*, EmberRf4ceTxOption *txOptions*, uint8_t *messageTag*, uint8_t *messageLength*, uint8_t * *message*)

Sends a message as per the ZigBee RF4CE specification.

Parameters

<i>pairingIndex</i>	The index of the entry in the pairing table to be used to transmit the packet. this parameter is ignored if broadcast bit is set in the txOptions bitmask.
<i>profileId</i>	The profile ID to be included in the RF4CE network header of the outgoing RF4CE network DATA frame.
<i>vendorId</i>	The vendor ID to be included in the RF4CE network header of the outgoing RF4CE network DATA frame. This field is meaningful only if the EMBER_RF4CE_TX_OPTIONS_VENDOR_SPECIFIC_BIT is set in the txOptions bitmask.
<i>txOptions</i>	7-bit transmission options bitmask as per ZigBee RF4CE specification (see Table 2, section 3.1.1.1).
<i>messageTag</i>	A value chosen by the application. This value will be passed in the corresponding emberRf4ceMessageSentHandler() call.
<i>messageLength</i>	The length in bytes of the message to be sent.
<i>message</i>	A pointer to the message to be sent.

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_INDEX_OUT_OF_RANGE](#) - If the application requested a unicast transmission and the passed pairingIndex parameter is an invalid index.
- :: [EMBER_MESSAGE_TOO_LONG](#) - If the passed message is too long. Notice that the maximum payload allowed for a certain combination of pairing index and TX options can be retrieved using the [emberRf4ceGetMaxPayload\(\)](#) API.
- [EMBER_INVALID_CALL](#) - If at least one of the following requirements are not met.

- The current network is an RF4CE network.
- The current network is up and running.
- The node is busy performing some discovery or pairing process.
- Unicast transmissions require that the passed pairingIndex corresponds to an active pairing entry.
- Broadcast transmissions can not be acked, encrypted or use long addressing.
- Secured transmissions can only be unicast and require a secured pairing entry.
- If channel designator is to be included in the packet, the transmission must be acked, the node must be a target and the destination must support channel normalization.
- **EMBER_NO_BUFFERS** - The message was not sent because of RAM shortage.
- **EMBER_DELIVERY_FAILED** - The message was rejected by the Network or MAC layers.

6.23.2.8 void emberRf4ceMessageSentHandler (EmberStatus *status*, uint8_t *pairingIndex*, EmberRf4ceTxOption *txOptions*, uint8_t *profileId*, uint16_t *vendorId*, uint8_t *messageTag*, uint8_t *messageLength*, uint8_t * *message*)

A callback invoked by the ZigBee RF4CE stack when it has completed sending a message.

Parameters

<i>status</i>	An EmberStatus value of: <ul style="list-style-type: none"> • EMBER_SUCCESS - The message was successfully delivered. • EMBER_DELIVERY_FAILED - The message was not delivered.
<i>pairingIndex</i>	The index of the entry in the pairing table used to transmit the message.
<i>txOptions</i>	The TX options bitmask as per ZigBee RF4CE specification used for transmitting the packet.
<i>profileId</i>	The profile ID included in the message.
<i>vendorId</i>	The vendor ID included in the message, if any.
<i>messageTag</i>	The tag value that was originally passed by the application in the emberRf4ceSend() API.
<i>messageLength</i>	The length in bytes of the message.
<i>message</i>	A pointer to the payload of the message that was sent.

6.23.2.9 void emberRf4ceIncomingMessageHandler (uint8_t *pairingIndex*, uint8_t *profileId*, uint16_t *vendorId*, EmberRf4ceTxOption *txOptions*, uint8_t *messageLength*, uint8_t * *message*)

A callback invoked by the ZigBee RF4CE stack when a message is received.

Parameters

<i>pairingIndex</i>	The index of the entry in the pairing table corresponding to the PAN on which the message was received.
<i>profileId</i>	The profile ID included in the message.

<i>vendorId</i>	The vendor ID included in the message, if any.
<i>txOptions</i>	The TX options used by the source node to transmit the received message.
<i>message-Length</i>	The length in bytes of the received message.
<i>message</i>	A pointer to the payload of the received message.

6.23.2.10 `uint8_t emberRf4ceGetMaxPayload (uint8_t pairingIndex, EmberRf4ceTxOption txOptions)`

An API that returns the maximum payload of an RF4CE network DATA frame according to the passed pairing index and TX options.

Parameters

<i>pairingIndex</i>	The index of the pairing table entry a packet shall be sent to. This parameter is meaningful only if the EMBER_RF4CE_TX_OPTIONS_BROADCAST_BIT bit in the passed TX options is not set.
<i>txOptions</i>	The TX options bitmask.

Returns

The maximum allowed payload in bytes according to the passed pairing index and TX options. Notice that if the EMBER_RF4CE_TX_OPTIONS_BROADCAST_BIT bit is not set in the TX options bitmask and the passed pairing index is not valid, 0 is returned.

6.23.2.11 `EmberStatus emberRf4ceStart (EmberRf4ceNodeCapabilities capabilities, EmberRf4ceVendorInfo * vendorInfo, int8_t power)`

The node starts the network operations. If the node is started as target, it will attempt to form a new RF4CE PAN by performing an energy scan for determining the best channel, by performing an active scan to determine an unique pan ID and unique network address. Once all these steps are successfully completed, the node starts normal network operations as RF4CE target. If the node is started as controller, no scanning will be performed.

Parameters

<i>capabilities</i>	An EmberRf4ceNodeCapabilities that specifies the node capabilities as described in section 3.4.2.4.
<i>vendorInfo</i>	A pointer to an EmberRf4ceVendorInfo struct containing the vendor information of the node.
<i>power</i>	The radio power the node should use in transmission.

Returns

An [EmberStatus](#) value of [EMBER_SUCCESS](#) if the starting network operations are successfully initiated, [EMBER_INVALID_CALL](#) if one or more passed parameters are invalid, or the current network is already up, or the node is already up as "always on" node on another network (an "always on" node is a non-sleepy ZigBee PRO node or a ZigBee RF4CE node). The application can set the application information using

the `emberRf4ceSetApplicationInfo()` API.

6.23.2.12 EmberStatus `emberRf4ceStop(void)`

6.23.2.13 EmberStatus `emberRf4ceDiscovery(EmberPanId panId, EmberNodeId nodeId, uint8_t searchDevType, uint16_t discDuration, uint8_t maxDiscRepetitions, uint8_t discProfileIdListLength, uint8_t * discProfileIdList)`

The node performs a discovery of ZigBee RF4CE nodes that matches the requirements specified in the passed parameters.

Parameters

<i>panId</i>	The PAN ID of the destination device for the discovery. This value can be set to EMBER_RF4CE_BROADCAST_PAN_ID to indicate a wildcard.
<i>nodeId</i>	The network address of the destination device for the discovery. This value can be set to EMBER_RF4CE_BROADCAST_ADDRESS to indicate a wildcard.
<i>searchDevType</i>	The device type to discover. This value can be set to 0xFF to indicate a wildcard.
<i>discDuration</i>	The time (in milliseconds) to wait for discovery responses to be sent back from potential target nodes on each channel.
<i>maxDiscRepetitions</i>	The maximum number of discovery trials. A discovery trial is defined as the transmission of a discovery request command frame on all available channels.
<i>discProfileIdListLength</i>	The length of the discovery profile ID list. The stack supports up to 7 profile ID entries.
<i>discProfileIdList</i>	The list of profile IDs against which profile IDs contained in received discovery response command frames will be matched for acceptance.

Returns

An `EmberStatus` value of `EMBER_SUCCESS` if the discovery process started successfully, `EMBER_INVALID_CALL` if the node is down (either the node was not cold started with the `emberRf4ceStart()` or warm started with `::emberRf4ceNetworkInit()`), or the current network is not an RF4CE network, or the node is already in the middle of a discovery or pairing process, or another `EmberStatus` value indicating the error occurred.

6.23.2.14 void `emberRf4ceDiscoveryCompleteHandler(EmberStatus status)`

A callback invoked by the ZigBee RF4CE stack when it has completed the discovery process.

Parameters

<i>status</i>	An <code>EmberStatus</code> value of <code>EMBER_SUCCESS</code> if discovery has been correctly performed over the three RF4CE channels and at least a valid discovery response was received. An <code>EmberStatus</code> value of <code>EMBER_DISCOVERY_TIMEOUT</code> if the discovery process completed and no valid discovery response was received. Otherwise, another <code>EmberStatus</code> value indicating the error occurred.
---------------	---

```
6.23.2.15 bool emberRf4ceDiscoveryRequestHandler ( EmberEUI64 srcIEEEAddr,
uint8_t nodeCapabilities, EmberRf4ceVendorInfo * vendorInfo,
EmberRf4ceApplicationInfo * appInfo, uint8_t searchDevType, uint8_t
rxLinkQuality )
```

A callback invoked by the ZigBee RF4CE stack when a discovery request is received. If the callback returns true, the stack shall respond with a discovery response, otherwise it will silently discard the discovery request message.

Parameters

<i>srcIEEEAddr</i>	The IEEE address of the node that issued the discovery request.
<i>node- Capabilities</i>	The node capabilities of the node that issued the discovery request.
<i>vendorInfo</i>	A pointer to an EmberRf4ceVendorInfo struct containing the vendor information of the node that issued the discovery request.
<i>appInfo</i>	A pointer to an EmberRf4ceApplicationInfo struct containing the application information of the node that issued the discovery request.
<i>searchDev- Type</i>	The device type being discovered. If this is 0xFF, any type is being requested.
<i>rxLink- Quality</i>	LQI value, as passed via the MAC sub-layer, of the discovery request command frame.

Returns

false if the discovery request should be discarded. Return true if the application wants to respond to the discovery request. The application can set the application information of the discovery response within this callback using the [emberRf4ceSetApplication-
Info\(\)](#) API.

```
6.23.2.16 bool emberRf4ceDiscoveryResponseHandler ( bool atCapacity, uint8_t channel,
EmberPanId panId, EmberEUI64 srcIEEEAddr, uint8_t nodeCapabilities,
EmberRf4ceVendorInfo * vendorInfo, EmberRf4ceApplicationInfo *
appInfo, uint8_t rxLinkQuality, uint8_t discRequestLqi )
```

A callback invoked by the ZigBee RF4CE stack when a discovery response is received.

Parameters

<i>atCapacity</i>	A bool set to true if the node sending the discovery response has no free entry in its pairing table, false otherwise.
<i>channel</i>	The channel on which the discovery response was received.
<i>panId</i>	The PAN identifier of the responding device.
<i>srcIEEEAddr</i>	The IEEE address of the responding device.
<i>node- Capabilities</i>	The capabilities of the responding node.
<i>vendorInfo</i>	The vendor information of the responding device.
<i>appInfo</i>	The application information of the responding device.
<i>rxLink- Quality</i>	LQI value, as passed via the MAC sub-layer, of the discovery response command frame.
<i>discRequest- Lqi</i>	The LQI of the discovery request command frame reported by the responding device.

Returns

If this callback returns true the stack will continue the discovery process. If this callback returns false, the discovery process will end at the end of the current discovery trial. A discovery trial is defined as the transmission of a discovery request command frame on all available channels.

6.23.2.17 EmberStatus emberRf4ceEnableAutoDiscoveryResponse (uint16_t duration)

The node automatically handles the receipt of discovery request command frames. Note that during this auto discovery response mode, the stack does not inform the application of the arrival of discovery request command frames. Furthermore, during this mode, if the node receives a command frame that is not a discovery request command frame, it will be discarded. At the end of the auto discovery response mode, the stack will call the [emberRf4ceAutoDiscoveryResponseCompleteHandler\(\)](#) callback.

Parameters

<i>duration</i>	The maximum duration, in milliseconds, while the node will be in auto discovery response mode.
-----------------	--

Returns

An [EmberStatus](#) value of EMBER_SUCCESS if the node successfully entered in auto discovery response mode. An [EmberStatus](#) value of EMBER_INVALID_CALL if the node has not been started, or the current network is not an RF4CE network, or the node is already in the middle of another discovery or pairing process. The application can set the application information used during the auto discovery process using the [emberRf4ceSetApplicationInfo\(\)](#) API.

6.23.2.18 void emberRf4ceAutoDiscoveryResponseCompleteHandler (EmberStatus status, EmberEUI64 srcIEEEAddr, uint8_t nodeCapabilities, EmberRf4ceVendorInfo * vendorInfo, EmberRf4ceApplicationInfo * appInfo, uint8_t searchDevType)

A callback invoked by the ZigBee RF4CE stack when it has completed the requested auto discovery response phase.

Parameters

<i>status</i>	An EmberStatus value of EMBER_SUCCESS indicating that it successfully received a discovery request frame twice from the same node with IEEE address specified by the srcIEEEAddr parameter. An EmberStatus value of EMBER_DISCOVERY_TIMEOUT if the node has not received the two discovery request frame within the auto discovery response duration interval. An EmberStatus value of EMBER_DISCOVERY_ERROR if the node has received two valid discovery request command frames from two different nodes. An EmberStatus value of EMBER_NO_BUFFERS if the node could not respond because of RAM shortage. An EmberStatus value of EMBER_ERR_FATAL if the MAC layer rejected the discovery response.
---------------	--

<i>srcIeeeAddr</i>	An EmberEUI64 value indicating the IEEE address from which the discovery request command frame was received. This parameter is non-NULL only if the status parameter is EMBER_SUCCESS.
<i>node-Capabilities</i>	The node capabilities of the node that issued the discovery request. This parameter is meaningful only if the status parameter is EMBER_SUCCESS.
<i>vendorInfo</i>	A pointer to an EmberRf4ceVendorInfo struct containing the vendor information of the node that issued the discovery request. This parameter is non-NULL only if the status parameter is EMBER_SUCCESS.
<i>appInfo</i>	A pointer to an EmberRf4ceApplicationInfo struct containing the application information of the node that issued the discovery request. This parameter is non-NULL only if the status parameter is EMBER_SUCCESS.
<i>searchDev-Type</i>	The device type being discovered. If this is 0xFF, any type is being requested. This parameter is meaningful only if the status parameter is EMBER_SUCCESS.

6.23.2.19 EmberStatus emberRf4cePair (uint8_t *channel*, EmberPanId *panId*, EmberEUI64 *ieeeAddr*, uint8_t *keyExchangeTransferCount*)

The node initiates the Rf4CE pairing process according to the specified parameters.

Parameters

<i>channel</i>	The logical channel of the device with which to pair.
<i>panId</i>	The PAN identifier of the device with which to pair.
<i>ieeeAddr</i>	The IEEE address of the device with which to pair.
<i>key-Exchange-Transfer-Count</i>	The number of transfers the target should use to exchange the link key with the pairing originator.

Returns

An [EmberStatus](#) value of EMBER_SUCCESS if the pairing process was successfully initiated and an unused pairing entry will be created for the new pairing. An [EmberStatus](#) value of EMBER_DUPLICATE_ENTRY if the pairing process was successfully initiated and an existing pairing entry will be updated. An [EmberStatus](#) value of EMBER_INVALID_CALL if the node has not been started, or the current network is not an RF4CE network, or the node is already in the middle of another discovery or pairing process. Another [EmberStatus](#) value indicating the specific error occurred otherwise.

6.23.2.20 void emberRf4cePairCompleteHandler (EmberStatus *status*, uint8_t *pairingIndex*, EmberRf4ceVendorInfo * *vendorInfo*, EmberRf4ceApplicationInfo * *appInfo*)

A callback invoked by the ZigBee RF4CE stack when the originator or the recipient node has completed the pairing process.

Parameters

<i>status</i>	An EmberStatus value of EMBER_SUCCESS if the pairing process succeeded and a pairing link has been established. An EmberStatus value of EMBER_NO_RESPONSE if the originator has timed out waiting for the pair response or for the ping response during the link key exchange procedure. An EmberStatus value of EMBER_TABLE_FULL if a pair response was received at the originator indicating that the recipient device has no available entry in its pairing table. An EmberStatus value of EMBER_NOT_PERMITTED if a pair response was received at the originator indicating that the recipient device did not accept the pair request. An EmberStatus value of EMBER_SECURITY_TIMEOUT if the node has timed out during the link key exchange or recovery procedures. An EmberStatus value of EMBER_SECURITY_FAILURE if some other error occurred during the link key exchange or recovery procedures. Another EmberStatus value indicating the specific reason why the originator or the recipient node failed to deliver a command frame.
<i>pairingIndex</i>	The index of the pairing table entry corresponding to the pairing link that was established during the pairing process. This field is meaningful only if the status parameter is EMBER_SUCCESS or EMBER_DUPLICATE_ENTRY.
<i>vendorInfo</i>	A pointer to an EmberRf4ceVendorInfo struct containing the vendor information of the peer node. This parameter is non-NULL only if the status parameter is EMBER_SUCCESS or EMBER_DUPLICATE_ENTRY.
<i>appInfo</i>	A pointer to an EmberRf4ceApplicationInfo struct containing the application information of the peer node. This parameter is non-NULL only if the status parameter is EMBER_SUCCESS or EMBER_DUPLICATE_ENTRY.

```
6.23.2.21 bool emberRf4cePairRequestHandler ( EmberStatus status, uint8_t
                                             pairingIndex, EmberEUI64 srcIeeeAddr, uint8_t nodeCapabilities,
                                             EmberRf4ceVendorInfo * vendorInfo, EmberRf4ceApplicationInfo *
                                             appInfo, uint8_t keyExchangeTransferCount )
```

A callback invoked by the ZigBee RF4CE stack when a pair request has been received.

Parameters

<i>status</i>	An EmberStatus value of EMBER_SUCCESS if the request pairing is not a duplicate pairing and an unused entry in the pairing table is available. An EmberStatus value of EMBER_TABLE_FULL if the request pairing is not a duplicate pairing and the pairing table is full. An EmberStatus value of EMBER_DUPLICATE_ENTRY if the request pairing is a duplicate pairing. In this case, the stack will update the entry indicated by the <i>pairingIndex</i> parameter.
<i>pairingIndex</i>	The index of the entry that will be used by the stack for the pairing link. If the status parameter is EMBER_TABLE_FULL this parameter is meaningless.
<i>srcIeeeAddr</i>	An EmberEUI64 value indicating the source IEEE address of the incoming pair request command.
<i>node- Capabilities</i>	The node capabilities of requesting device.
<i>vendorInfo</i>	The vendor information of the requesting device.

<i>appInfo</i>	The application information of the requesting device.
<i>key-Exchange-Transfer-Count</i>	The number of transfers to be used to exchange the link key with the pairing originator, indicated in the incoming pair request command.

Returns

true if the application accepts the pair, false otherwise. The application can set the application information of the pair response within this callback using the [emberRf4ceSetApplicationInfo\(\)](#) API.

6.23.2.22 EmberStatus emberRf4ceUnpair (uint8_t pairingIndex)

The node attempts to remove the specified pairing link. If successful, the node transmit an unpair request command frame to the peer node. The pairing link will be then removed regardless on whether the peer node has acknowledged the unpair request command frame.

Parameters

<i>pairingIndex</i>	The index of the pairing link to be removed.
---------------------	--

Returns

An [EmberStatus](#) value of EMBER_SUCCESS if the unpairing process has successfully started. An [EmberStatus](#) value of EMBER_INVALID_CALL if the node has not been started, or the current network is not an RF4CE network, or if the node is in the middle of a discovery or pairing process or if the passed index does not correspond to an active pairing link. An [EmberStatus](#) value of EMBER_NO_BUFFERS if the unpair request command frame was not sent because of RAM shortage. An [EmberStatus](#) value of EMBER_ERR_FATAL if the MAC layer rejected the unpair request command frame.

6.23.2.23 void emberRf4ceUnpairHandler (uint8_t pairingIndex)

A callback invoked by the ZigBee RF4CE stack when an unpair command frame has been received. The stack will remove the pairing link indicated by the passed index.

Parameters

<i>pairingIndex</i>	The index of the pairing link to be removed.
---------------------	--

Returns

An [EmberStatus](#) value of EMBER_SUCCESS if the unpairing process was successfully initiated. Another [EmberStatus](#) value indicating the specific error occurred otherwise.

6.23.2.24 void emberRf4ceUnpairCompleteHandler (*uint8_t pairingIndex*)

A callback invoked by the ZigBee RF4CE stack when the unpair procedure has been completed. According to the RF4CE specs, during the unpair procedure, the stack sends an unpair command frame. If the command is not successfully delivered, the stack tries another RF4CE channel until the frame is received or the stack already tried all the RF4CE channels. Either way, at the end of the unpair process the pairing table entry is deleted and this callback is invoked.

Parameters

<i>pairingIndex</i>	The index of the removed pairing link.
---------------------	--

6.23.2.25 EmberStatus emberRf4ceSetPowerSavingParameters (*uint32_t dutyCycle, uint32_t activePeriod*)

The node enables or disables RF4CE power saving mode according to the passed parameters. If dutyCycle is 0x000000, power saving mode is disabled and the node radio will be always on listening for incoming packets. If activePeriod is 0x000000 and dutyCycle is > 0 the radio is turned off until further notice. Otherwise, the stack will enable power saving mode and switch the radio on and off according to the passed parameters, whereas the node's radio will be on for activePeriod milliseconds every dutyCycle milliseconds.

Parameters

<i>dutyCycle</i>	The duty cycle of a device in milliseconds. A value of 0x000000 disable power saving management at the RF4CE stack and the application must provide this functionality directly. Legal values for this parameter are 0x000000 and any value in the range [nwkcMinActivePeriod=16.8ms, nwkcMaxDutyCycle=1s].
<i>activePeriod</i>	The active period of a device in milliseconds. Legal values for this parameter fall in the set [nwkcMinActivePeriod=16.8ms, dutyCycle) U {0x000000}. If the dutyCycle parameter is 0, the activePeriod parameter is ignored.

Returns

An [EmberStatus](#) value of EMBER_SUCCESS if the power saving parameters were correctly set. An [EmberStatus](#) value of EMBER_INVALID_CALL if the node has not been started yet, or the current network is not an RF4CE network, or the node is currently busy with some discovery or pairing process, or if any of the passed parameters is not in the legal range.

6.23.2.26 EmberStatus emberRf4ceSetFrequencyAgilityParameters (*uint8_t rssiWindowSize, uint8_t channelChangeReads, int8_t rssiThreshold, uint16_t readInterval, uint8_t readDuration*)

Every target node periodically performs an energy scan of the base channel to detect whether the channel becomes compromised. In particular if channelChangeReads RSSI reads out of the last rssiWindowSize RSSI reads are above the rssiThreshold, the target will switch its base channel to the next channel.

Parameters

<i>rssiWindow-Size</i>	Defines the size of the RSSI reads window, that is, the number of the most recent RSSI reads that are taken into consideration to decide whether a channel switch is required or not. Valid values for this parameter fall in the interval [1,32]. Setting this parameter to 0 disables frequency agility at the target. This parameter is set by default to 16.
------------------------	--

channelChangeReads Defines the number of RSSI reads above the RSSI threshold that will trigger a channel switch. Valid values for this parameter fall in the interval [1,*rssiWindowSize*]. This parameter is set by default to 15.

Parameters

<i>rssi-Threshold</i>	Defines the RSSI threshold. A RSSI value resulting from the periodic scan that is above this threshold is considered a 'channel congested' read. If the stack detects multiple channel congested reads, it will eventually move the base channel to the next RF4CE channel. This parameter is set by default to the CCA threshold.
<i>readInterval</i>	The interval length (in seconds) between two consecutive RSSI reads. This parameter is set by default to 10 seconds.
<i>read-Duration</i>	Sets the exponent of the number of scan periods of the RSSI read process, where a scan period is 960 symbols, and a symbol is 16 microseconds. The scan will occur for $((2^{\text{duration}}) + 1)$ 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. This parameter is set by default to 0.

Returns

An [EmberStatus](#) value of EMBER_SUCCESS if the frequency agility parameters were correctly set. An [EmberStatus](#) value of EMBER_INVALID_CALL if the node has not been started yet, or if it has been started as controller, or the current network is not an RF4CE network, or the node is currently busy with some discovery or pairing process, or some of the passed parameters are not valid.

6.23.2.27 EmberStatus emberRf4ceSetDiscoveryLqiThreshold (uint8_t *threshold*)

Set the discovery LQI threshold parameter.

Parameters

<i>threshold</i>	The LQI threshold below which discovery requests will be rejected.
------------------	--

Returns

An [EmberStatus](#) value of EMBER_SUCCESS if the discovery LQI threshold was successfully set. Another [EmberStatus](#) value indicating the specific error occurred otherwise.

6.24 Ember ZigBee RF4CE Data Types

Data Structures

- struct [EmberRf4ceVendorInfo](#)
Defines the vendor information block (see section 3.3.1, Figure 16).
- struct [EmberRf4ceApplicationInfo](#)
Defines the application information block (see section 3.3.1, Figure 17).
- struct [EmberRf4cePairingTableEntry](#)
The internal representation of a pairing table entry.

RF4CE Types

- enum [EmberRf4ceTxOption](#) {
 EMBER_RF4CE_TX_OPTIONS_NONE, EMBER_RF4CE_TX_OPTIONS_BRO-
 ADCAST_BIT, EMBER_RF4CE_TX_OPTIONS_USE_IEEE_ADDRESS_BIT, E-
 MBER_RF4CE_TX_OPTIONS_ACK_REQUESTED_BIT,
 EMBER_RF4CE_TX_OPTIONS_SECURITY_ENABLED_BIT, EMBER_RF4CE-
 _TX_OPTIONS_SINGLE_CHANNEL_BIT, EMBER_RF4CE_TX_OPTIONS_C-
 HANNEL_DESIGNATOR_BIT, EMBER_RF4CE_TX_OPTIONS_VENDOR_SP-
 ECIFIC_BIT }
- enum [EmberRf4ceNodeCapabilities](#) {
 EMBER_RF4CE_NODE_CAPABILITIES_NONE, EMBER_RF4CE_NODE_CA-
 PABILITIES_IS_TARGET_BIT, EMBER_RF4CE_NODE_CAPABILITIES_POW-
 ER_SOURCE_BIT, EMBER_RF4CE_NODE_CAPABILITIES_SECURITY_BIT,
 EMBER_RF4CE_NODE_CAPABILITIES_CHANNEL_NORM_BIT }
- enum [EmberRf4ceApplicationCapabilities](#) {
 EMBER_RF4CE_APP_CAPABILITIES_NONE, EMBER_RF4CE_APP_CAPAB-
 ILITIES_USER_STRING_BIT, EMBER_RF4CE_APP_CAPABILITIES_SUPPO-
 RTED_DEVICE_TYPES_MASK, EMBER_RF4CE_APP_CAPABILITIES_SUPP-
 ORTED_DEVICE_TYPES_OFFSET,
 EMBER_RF4CE_APP_CAPABILITIES_SUPPORTED_PROFILES_MASK, EM-
 BER_RF4CE_APP_CAPABILITIES_SUPPORTED_PROFILES_OFFSET }
- enum [EmberRf4cePairingEntryStatus](#) { EMBER_RF4CE_PAIRING_TABLE_EN-
 TRY_STATUS_UNUSED, EMBER_RF4CE_PAIRING_TABLE_ENTRY_STAT-
 US_PROVISIONAL, EMBER_RF4CE_PAIRING_TABLE_ENTRY_STATUS_A-
 CTIVE }
- #define EMBER_RF4CE_BROADCAST_ADDRESS
- #define EMBER_RF4CE_BROADCAST_PAN_ID
- #define EMBER_ALL_ZIGBEE_RF4CE_CHANNELS_MASK
- #define EMBER_RF4CE_NULL_VENDOR_ID
- #define EMBER_RF4CE_VENDOR_STRING_LENGTH
- #define EMBER_RF4CE_APPLICATION_USER_STRING_LENGTH
- #define EMBER_RF4CE_APPLICATION_DEVICE_TYPE_LIST_MAX_LENGT-
 H
- #define EMBER_RF4CE_APPLICATION_PROFILE_ID_LIST_MAX_LENGTH
- #define EMBER_RF4CE_PAIRING_TABLE_BROADCAST_INDEX
- #define EMBER_RF4CE_MIN_ACTIVE_PERIOD_MS
- #define EMBER_RF4CE_MAX_DUTY_CYCLE_MS
- #define EMBER_RF4CE_PAIRING_TABLE_ENTRY_INFO_STATUS_MASK

- #define EMBER_RF4CE_PAIRING_TABLE_ENTRY_INFO_HAS_LINK_KEY_BIT
- #define EMBER_RF4CE_PAIRING_TABLE_ENTRY_INFO_IS_PAIRING_INITIATOR_BIT

6.24.1 Detailed Description

See [rf4ce-types.h](#) for source code.

6.24.2 Macro Definition Documentation

6.24.2.1 #define EMBER_RF4CE_BROADCAST_ADDRESS

Zigbee RF4CE broadcast network address.

Definition at line [27](#) of file [rf4ce-types.h](#).

6.24.2.2 #define EMBER_RF4CE_BROADCAST_PAN_ID

Zigbee RF4CE broadcast PAN ID.

Definition at line [32](#) of file [rf4ce-types.h](#).

6.24.2.3 #define EMBER_ALL_ZIGBEE_RF4CE_CHANNELS_MASK

Bitmask to scan all the Zigbee RF4CE channels.

Definition at line [37](#) of file [rf4ce-types.h](#).

6.24.2.4 #define EMBER_RF4CE_NULL_VENDOR_ID

Zigbee RF4CE broadcast network address.

Definition at line [43](#) of file [rf4ce-types.h](#).

6.24.2.5 #define EMBER_RF4CE_VENDOR_STRING_LENGTH

The length of the vendor string stored in the [EmberRf4ceVendorInfo](#) struct.

Definition at line [49](#) of file [rf4ce-types.h](#).

6.24.2.6 #define EMBER_RF4CE_APPLICATION_USER_STRING_LENGTH

The length of the application user string stored in the [EmberRf4ceApplicationInfo](#) struct.

Definition at line [55](#) of file [rf4ce-types.h](#).

6.24.2.7 #define EMBER_RF4CE_APPLICATION_DEVICE_TYPE_LIST_MAX_LENGTH

The length of the application device type list stored in the [EmberRf4ceApplicationInfo](#) struct.

Definition at line [61](#) of file [rf4ce-types.h](#).

6.24.2.8 #define EMBER_RF4CE_APPLICATION_PROFILE_ID_LIST_MAX_LENGTH

The length of the application profile ID list stored in the [EmberRf4ceApplicationInfo](#) struct.

Definition at line [67](#) of file [rf4ce-types.h](#).

6.24.2.9 #define EMBER_RF4CE_PAIRING_TABLE_BROADCAST_INDEX

A distinguished pairing table index used to indicate a broadcast message.

Definition at line [73](#) of file [rf4ce-types.h](#).

6.24.2.10 #define EMBER_RF4CE_MIN_ACTIVE_PERIOD_MS

Minimum active period duration in milliseconds.

Definition at line [78](#) of file [rf4ce-types.h](#).

6.24.2.11 #define EMBER_RF4CE_MAX_DUTY_CYCLE_MS

Maximum duty cycle duration in milliseconds.

Definition at line [83](#) of file [rf4ce-types.h](#).

6.24.2.12 #define EMBER_RF4CE_PAIRING_TABLE_ENTRY_INFO_STATUS_MASK

The status of the pairing table entry stored in the info byte.

Definition at line [293](#) of file [rf4ce-types.h](#).

6.24.2.13 #define EMBER_RF4CE_PAIRING_TABLE_ENTRY_INFO_HAS_LINK_KEY_BIT

Bit in the info byte of a pairing table entry that indicates whether the pairing entry has a link key.

Definition at line [299](#) of file [rf4ce-types.h](#).

6.24.2.14 #define EMBER_RF4CE_PAIRING_TABLE_ENTRY_INFO_IS_PAIRING_INITIATOR_BIT

Bit in the info byte of a pairing table entry that indicates whether the node initiated the pairing process.

Definition at line [305](#) of file [rf4ce-types.h](#).

6.24.3 Enumeration Type Documentation

6.24.3.1 enum EmberRf4ceTxOption

Options to use when sending a message.

These are defined in the RF4CE specs (see Table 2, section 3.1.1.1).

Enumerator:

- EMBER_RF4CE_TX_OPTIONS_NONE*** No options.
- EMBER_RF4CE_TX_OPTIONS_BROADCAST_BIT*** Broadcast or unicast transmission (transmission mode).
- EMBER_RF4CE_TX_OPTIONS_USE_IEEE_ADDRESS_BIT*** Use destination IEEE address or destination network address (destination addressing mode).
- EMBER_RF4CE_TX_OPTIONS_ACK_REQUESTED_BIT*** MAC acknowledged transmission (acknowledgment mode).
- EMBER_RF4CE_TX_OPTIONS_SECURITY_ENABLED_BIT*** Transmit with or without security (security mode).
- EMBER_RF4CE_TX_OPTIONS_SINGLE_CHANNEL_BIT*** Use single or multiple channel operation (channel agility mode).
- EMBER_RF4CE_TX_OPTIONS_CHANNEL_DESIGNATOR_BIT*** Specify channel designator or not (channel normalization mode).
- EMBER_RF4CE_TX_OPTIONS_VENDOR_SPECIFIC_BIT*** Data is vendor specific or non-vendor specific (payload mode).

Definition at line 91 of file [rf4ce-types.h](#).

6.24.3.2 enum EmberRf4ceNodeCapabilities

RF4CE node capabilities.

These are defined in the RF4CE specs (see section 3.4.2.4).

Enumerator:

- EMBER_RF4CE_NODE_CAPABILITIES_NONE***
- EMBER_RF4CE_NODE_CAPABILITIES_IS_TARGET_BIT*** From RF4CE specs: "The node type sub-field is 1-bit in length and shall be set to one if the node is a target node. Otherwise the node type sub-field shall be set to zero indicating a controller node".
- EMBER_RF4CE_NODE_CAPABILITIES_POWER_SOURCE_BIT*** From RF4CE specs: "The power source sub-field is 1-bit in length and shall be set to one if the node is receiving power from the alternating current mains. Otherwise, the power source sub-field shall be set to zero".
- EMBER_RF4CE_NODE_CAPABILITIES_SECURITY_BIT*** From RF4CE specs: "The security capable sub-field is 1-bit in length and shall be set to one if the node is capable of sending and receiving cryptographically protected frames. Otherwise, the security capable sub-field shall be set to zero".
- EMBER_RF4CE_NODE_CAPABILITIES_CHANNEL_NORM_BIT*** From RF4CE specs: "The channel normalization sub-field is 1-bit in length and shall be set to one if the node will react to a channel change request through the channel designator sub-field of the frame control field. Otherwise, the channel normalization sub-field shall be set to zero".

Definition at line 138 of file [rf4ce-types.h](#).

6.24.3.3 enum EmberRf4ceApplicationCapabilities

RF4CE application capabilities.

These are defined in the RF4CE specs (see section 3.3.1.5, Figure 18).

Enumerator:

EMBER_RF4CE_APP_CAPABILITIES_NONE

EMBER_RF4CE_APP_CAPABILITIES_USER_STRING_BIT From RF4CE specs-

: "The user string bit shall be set to one if a user string is to be included in the frame. Otherwise, it shall be set to zero. The user string field shall be included in the frame only if the user string bit is set to 1".

EMBER_RF4CE_APP_CAPABILITIES_SUPPORTED_DEVICE_TYPES_MASK

From RF4CE specs: "The number of supported device types sub-field is 2 bits in length and shall contain the number of device types supported by the application".

EMBER_RF4CE_APP_CAPABILITIES_SUPPORTED_DEVICE_TYPES_OFFSET

The offset of the device types subfield within the application capabilities field.

EMBER_RF4CE_APP_CAPABILITIES_SUPPORTED_PROFILES_MASK From

RF4CE specs: "The number of supported profiles sub-field is 3 bits in length and shall contain the number of profiles disclosed as supported by the application".

EMBER_RF4CE_APP_CAPABILITIES_SUPPORTED_PROFILES_OFFSET The

offset of the supported profiles subfield within the application capabilities field.

Definition at line 181 of file [rf4ce-types.h](#).

6.24.3.4 enum EmberRf4cePairingEntryStatus

RF4CE pairing table entry status.

Enumerator:

EMBER_RF4CE_PAIRING_TABLE_ENTRY_STATUS_UNUSED The pairing ta-
ble entry is not in use.

EMBER_RF4CE_PAIRING_TABLE_ENTRY_STATUS_PROVISIONAL The pair-
ing table entry is marked as provisional because currently involved in some pair-
ing process.

EMBER_RF4CE_PAIRING_TABLE_ENTRY_STATUS_ACTIVE The pairing ta-
ble entry is in use.

Definition at line 269 of file [rf4ce-types.h](#).

6.25 Hardware Abstraction Layer (HAL) API Reference

Modules

- [Common Microcontroller Functions](#)
- [Token Access](#)
- [Sample APIs for Peripheral Access](#)
- [USB Device Stack Library](#)
- [System Timer Control](#)
- [Symbol Timer Control](#)
- [HAL Configuration](#)
- [HAL Utilities](#)
- [Bootloader Interfaces](#)
- [Custom Bootloader HAL](#)

6.25.1 Detailed Description

EM35x Microprocessors

HAL function names have the following prefix conventions:

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

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

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

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

See also [hal.h](#).

6.26 Common Microcontroller Functions

Macros

- #define halGetEm2xxResetInfo()
- #define MICRO_DISABLE_WATCH_DOG_KEY
- #define GPIO_MASK_SIZE
- #define GPIO_MASK
- #define WAKE_GPIO_MASK
- #define WAKE_GPIO_SIZE
- #define WAKE_MASK_INVALID
- #define WAKE_EVENT_SIZE
- #define DEBUG_TOGGLE(n)

Typedefs

- typedef uint32_t WakeEvents
- typedef uint32_t WakeMask

Enumerations

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

Functions

- void halStackProcessBootCount (void)
- uint8_t halGetResetInfo (void)
- PGM_P halGetResetString (void)
- void halInit (void)
- void halReboot (void)
- void halPowerUp (void)
- void halPowerDown (void)
- void halResume (void)
- void halSuspend (void)
- void halInternalEnableWatchDog (void)
- void halInternalDisableWatchDog (uint8_t magicKey)
- bool halInternalWatchDogEnabled (void)
- void halSleep (SleepModes sleepMode)
- void halCommonDelayMicroseconds (uint16_t us)
- void halCommonDisableVreg1v8 (void)
- void halCommonEnableVreg1v8 (void)
- void halInternalSysReset (uint16_t extendedCause)
- uint16_t halGetExtendedResetInfo (void)
- PGM_P halGetExtendedResetString (void)

- `EmberStatus halSetRadioHoldOff (bool enable)`
- `bool halGetRadioHoldOff (void)`
- `void halStackRadioPowerDownBoard (void)`
- `void halStackRadioPowerUpBoard (void)`
- `void halStackRadioPowerMainControl (bool powerUp)`
- `void halRadioPowerUpHandler (void)`
- `void halRadioPowerDownHandler (void)`

Variables

- `volatile int8_t halCommonVreg1v8EnableCount`

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 35 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 SC3_VECTOR_INDEX
- #define SC4_VECTOR_INDEX
- #define USB_VECTOR_INDEX
- #define VECTOR_TABLE_LENGTH

6.26.1 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.

Many of the supplied example applications use these microcontroller functions. See [hal/micro/micro-common.h](#) for source code.

See also [hal/micro/cortexm3/micro.h](#) for source code.

6.26.2 Macro Definition Documentation

6.26.2.1 #define halGetEm2xxResetInfo()

Calls [halGetExtendedResetInfo\(\)](#) and translates the EM35x or COBRA reset code to the corresponding value used by the EM2XX HAL. Any reset codes not present in the EM2XX-X 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-compatible reset code. If not supported by the EM2XX, return the platform-specific code with B7 set.

Definition at line [88](#) of file [micro.h](#).

6.26.2.2 #define MICRO_DISABLE_WATCH_DOG_KEY

The value that must be passed as the single parameter to [halInternalDisableWatchDog\(\)](#) in order to successfully disable the watchdog timer.

Definition at line [49](#) of file [micro-common.h](#).

6.26.2.3 #define GPIO_MASK_SIZE

Definition at line [223](#) of file [micro-common.h](#).

6.26.2.4 #define GPIO_MASK

Definition at line [224](#) of file [micro-common.h](#).

6.26.2.5 #define WAKE_GPIO_MASK

Definition at line [225](#) of file [micro-common.h](#).

6.26.2.6 #define WAKE_GPIO_SIZE

Definition at line [226](#) of file [micro-common.h](#).

6.26.2.7 #define WAKE_MASK_INVALID

Definition at line [231](#) of file [micro-common.h](#).

6.26.2.8 #define WAKE_EVENT_SIZE**Note**

The preprocessor symbol WAKE_EVENT_SIZE has been deprecated. Please use WakeMask instead.

Definition at line [236](#) of file [micro-common.h](#).

6.26.2.9 #define DEBUG_TOGGLE(n)

Definition at line [265](#) of file [micro-common.h](#).

6.26.2.10 #define STACK_VECTOR_INDEX

A numerical definition for a vector.

Definition at line [121](#) of file [cortexm3/micro.h](#).

6.26.2.11 #define RESET_VECTOR_INDEX

A numerical definition for a vector.

Definition at line [122](#) of file [cortexm3/micro.h](#).

6.26.2.12 #define NMI_VECTOR_INDEX

A numerical definition for a vector.

Definition at line [123](#) of file [cortexm3/micro.h](#).

6.26.2.13 #define HARD_FAULT_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 124 of file [cortexm3/micro.h](#).

6.26.2.14 #define MEMORY_FAULT_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 125 of file [cortexm3/micro.h](#).

6.26.2.15 #define BUS_FAULT_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 126 of file [cortexm3/micro.h](#).

6.26.2.16 #define USAGE_FAULT_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 127 of file [cortexm3/micro.h](#).

6.26.2.17 #define RESERVED07_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 128 of file [cortexm3/micro.h](#).

6.26.2.18 #define RESERVED08_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 129 of file [cortexm3/micro.h](#).

6.26.2.19 #define RESERVED09_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 130 of file [cortexm3/micro.h](#).

6.26.2.20 #define RESERVED10_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 131 of file [cortexm3/micro.h](#).

6.26.2.21 #define SVCALL_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 132 of file [cortexm3/micro.h](#).

6.26.2.22 #define DEBUG_MONITOR_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 133 of file [cortexm3/micro.h](#).

6.26.2.23 #define RESERVED13_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 134 of file [cortexm3/micro.h](#).

6.26.2.24 #define PENDSV_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 135 of file [cortexm3/micro.h](#).

6.26.2.25 #define SYSTICK_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 136 of file [cortexm3/micro.h](#).

6.26.2.26 #define TIMER1_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 137 of file [cortexm3/micro.h](#).

6.26.2.27 #define TIMER2_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 138 of file [cortexm3/micro.h](#).

6.26.2.28 #define MANAGEMENT_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 139 of file [cortexm3/micro.h](#).

6.26.2.29 #define BASEBAND_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 140 of file [cortexm3/micro.h](#).

6.26.2.30 #define SLEEP_TIMER_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 141 of file [cortexm3/micro.h](#).

6.26.2.31 #define SC1_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 142 of file [cortexm3/micro.h](#).

6.26.2.32 #define SC2_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 143 of file [cortexm3/micro.h](#).

6.26.2.33 #define SECURITY_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 144 of file [cortexm3/micro.h](#).

6.26.2.34 #define MAC_TIMER_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 145 of file [cortexm3/micro.h](#).

6.26.2.35 #define MAC_TX_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 146 of file [cortexm3/micro.h](#).

6.26.2.36 #define MAC_RX_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 147 of file [cortexm3/micro.h](#).

6.26.2.37 #define ADC_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 148 of file [cortexm3/micro.h](#).

6.26.2.38 #define IRQA_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 149 of file [cortexm3/micro.h](#).

6.26.2.39 #define IRQB_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 150 of file [cortexm3/micro.h](#).

6.26.2.40 #define IRQC_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 151 of file [cortexm3/micro.h](#).

6.26.2.41 #define IRQD_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 152 of file [cortexm3/micro.h](#).

6.26.2.42 #define DEBUG_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 153 of file [cortexm3/micro.h](#).

6.26.2.43 #define SC3_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 154 of file [cortexm3/micro.h](#).

6.26.2.44 #define SC4_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 155 of file [cortexm3/micro.h](#).

6.26.2.45 #define USB_VECTOR_INDEX

A numerical definition for a vector.

Definition at line 156 of file [cortexm3/micro.h](#).

6.26.2.46 #define VECTOR_TABLE_LENGTH

Number of vectors.

Definition at line 161 of file [cortexm3/micro.h](#).

6.26.3 Typedef Documentation

6.26.3.1 typedef uint32_t WakeEvents

Definition at line 227 of file [micro-common.h](#).

6.26.3.2 typedef uint32_t WakeMask

Definition at line 228 of file [micro-common.h](#).

6.26.4 Enumeration Type Documentation

6.26.4.1 enum SleepModes

Enumerations for the possible microcontroller sleep modes.

- SLEEPMODE_RUNNING Everything is active and running. In practice this mode is not used, but it is defined for completeness of information.
- SLEEPMODE_IDLE Only the CPU is idled. The rest of the chip continues running normally. The chip will wake from any interrupt.
- SLEEPMODE_WAKETIMER The sleep timer clock sources remain running. The RC is always running and the 32kHz XTAL depends on the board header. Wakeup is possible from both GPIO and the sleep timer. System time is maintained. The sleep timer is assumed to be configured properly for wake events.
- SLEEPMODE_MAINTAINTIMER The sleep timer clock sources remain running. The RC is always running and the 32kHz XTAL depends on the board header. Wakeup is possible from only GPIO. System time is maintained. NOTE: This mode is not available on EM2XX chips.
- SLEEPMODE_NOTIMER The sleep timer clock sources (both RC and XTAL) are turned off. Wakeup is possible from only GPIO. System time is lost.
- SLEEPMODE_HIBERNATE This maps to EM4 Hibernate on the EFM32/EFR32 devices. RAM is not retained in SLEEPMODE_HIBERNATE so waking up from this sleepmode will behave like a reset. NOTE: This mode is only available on EFM32/EFR32

Enumerator:

```
SLEEPMODE_RUNNING
SLEEPMODE_IDLE
SLEEPMODE_WAKETIMER
SLEEPMODE_MAINTAINTIMER
SLEEPMODE_NOTIMER
SLEEPMODE_HIBERNATE
SLEEPMODE_RESERVED
SLEEPMODE_POWERDOWN
SLEEPMODE_POWERSAVE
```

Definition at line 98 of file [micro-common.h](#).

6.26.5 Function Documentation

6.26.5.1 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.

6.26.5.2 uint8_t halGetResetInfo (void)

Gets information about what caused the microcontroller to reset.

Returns

A code identifying the cause of the reset.

6.26.5.3 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.

6.26.5.4 void halInit (void)

Initializes microcontroller-specific peripherals.

6.26.5.5 void halReboot (void)

Restarts the microcontroller and therefore everything else.

6.26.5.6 void halPowerUp (void)

Powers up microcontroller peripherals and board peripherals.

6.26.5.7 void halPowerDown (void)

Powers down microcontroller peripherals and board peripherals.

6.26.5.8 void halResume (void)

Resumes microcontroller peripherals and board peripherals.

6.26.5.9 void halSuspend (void)

Suspends microcontroller peripherals and board peripherals.

6.26.5.10 void hallInternalEnableWatchDog (void)

Enables the watchdog timer.

6.26.5.11 void hallInternalDisableWatchDog (uint8_t *magicKey*)

Disables the watchdog timer.

Note

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

Parameters

<i>magicKey</i>	A value (MICRO_DISABLE_WATCH_DOG_KEY) that enables the function.
-----------------	--

6.26.5.12 bool hallInternalWatchDogEnabled (void)

Determines whether the watchdog has been enabled or disabled.

Returns

A bool value indicating if the watchdog is current enabled.

6.26.5.13 void halSleep (SleepModes *sleepMode*)

Puts the microcontroller to sleep in a specified mode.

Note

This routine always enables interrupts.

Parameters

<i>sleepMode</i>	A microcontroller sleep mode
------------------	------------------------------

See Also

[SleepModes](#)

Referenced by [usbSuspendDsr\(\)](#).

6.26.5.14 void halCommonDelayMicroseconds (uint16_t us)

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

The function is implemented with cycle-counted busy loops and is intended to create the short delays required when interfacing with hardware peripherals.

The accuracy of the timing provided by this function is not specified, but a general rule is that when running off of a crystal oscillator it will be within 10us. If the micro is running off of another type of oscillator (e.g. RC) the timing accuracy will potentially be much worse.

Parameters

<i>us</i>	The specified time, in microseconds. Values should be between 1 and 65535 microseconds.
-----------	---

6.26.5.15 void halCommonDisableVreg1v8 (void)

Disable the 1.8V regulator. This function is to be used when the 1.8V supply is provided externally. Disabling the regulator saves current consumption. Disabling the regulator will cause ADC readings of external signals to be wrong. These external signals include analog sources ADC0 thru ADC5 and VDD_PADS/4.

Note

: Only used when DISABLE_INTERNAL_1V8_REGULATOR is defined.

6.26.5.16 void halCommonEnableVreg1v8 (void)

Enable the 1.8V regulator. Normally the 1.8V regulator is enabled out of reset. This function is only needed if the 1.8V regulator has been disabled and ADC conversions on external signals are needed. These external signals include analog sources ADC0 thru ADC5 and VDD_PADS/4. The state of 1v8 survives deep sleep.

Note

: Only used when DISABLE_INTERNAL_1V8_REGULATOR is defined.

6.26.5.17 void halInternalSysReset (uint16_t extendedCause)

Records the specified reset cause then forces a reboot.

6.26.5.18 uint16_t halGetExtendedResetInfo (void)

Returns the Extended Reset Cause information.

Returns

A 16-bit code identifying the base and extended cause of the reset

6.26.5.19 PGM_P halGetExtendedResetString (void)

Calls [halGetExtendedResetInfo\(\)](#) and supplies a string describing the extended cause of the reset. [halGetResetString\(\)](#) 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.

6.26.5.20 EmberStatus halSetRadioHoldOff (bool enable)

Enables or disables Radio HoldOff support.

Parameters

<i>enable</i>	When true, configures RHO_GPIO in BOARD_HEADER as an input which, when asserted, will prevent the radio from transmitting. When false, configures RHO_GPIO for its original default purpose.
---------------	--

Returns

[EMBER_SUCCESS](#) if Radio HoldOff was configured as desired or [EMBER_BAD_ARGUMENT](#) if requesting it be enabled but RHO has not been configured by the [BOARD_HEADER](#).

6.26.5.21 bool halGetRadioHoldOff (void)

Returns whether Radio HoldOff has been enabled or not.

Returns

true if Radio HoldOff has been enabled, false otherwise.

6.26.5.22 void halStackRadioPowerDownBoard (void)

To assist with saving power when the radio automatically powers down, this function allows the stack to tell the HAL to put pins specific to radio functionality in their powerdown state. The pin state used is the state used by [halInternalPowerDownBoard](#), but applied only to the pins identified in the global variable [gpioRadioPowerBoardMask](#). The stack will automatically call this function as needed, but it will only change GPIO state based on [gpioRadioPowerBoardMask](#). Most commonly, the bits set in [gpioRadioPowerBoardMask](#) pertain to using a Front End Module. This function is often called from interrupt context.

6.26.5.23 void halStackRadioPowerUpBoard (void)

To assist with saving power when the radio automatically powers up, this function allows the stack to tell the HAL to put pins specific to radio functionality in their powerup state. The pin state used is the state used by `halInternalPowerUpBoard`, but applied only to the pins identified in the global variable `gpioRadioPowerBoardMask`. The stack will automatically call this function as needed, but it will only change GPIO state based on `gpioRadioPowerBoardMask`. Most commonly, the bits set in `gpioRadioPowerBoardMask` pertain to using a Front End Module. This function can be called from interrupt context.

6.26.5.24 void halStackRadioPowerMainControl (bool powerUp)

This function is called automatically by the stack prior to Radio power-up and after Radio power-down. It can be used to prepare for the radio being powered on and to clean up after it's been powered off. Unlike [halStackRadioPowerUpBoard\(\)](#) and [halStackRadioPowerDownBoard\(\)](#), which can be called from interrupt context, this function is only called from main-line context.

6.26.5.25 void halRadioPowerUpHandler (void)

Handler called in main context prior to radio being powered on.

6.26.5.26 void halRadioPowerDownHandler (void)

Handler called in main context after radio has been powered off.

6.26.6 Variable Documentation

6.26.6.1 volatile int8_t halCommonVreg1v8EnableCount

Helper variable to track the state of 1.8V regulator.

Note

: Only used when `DISABLE_INTERNAL_1V8_REGULATOR` is defined.

6.27 Token Access

Modules

- [Tokens](#)
- [Simulated EEPROM](#)
- [Simulated EEPROM 2](#)

6.27.1 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 designed 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 manufacturing token definitions.

Note

Simulated EEPROM, version 2 is only supported on EM335x chips.

6.28 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)

6.28.1 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
#ifndef DEFINETYPES
    typedef data_type type
#endif //DEFINETYPES
#ifndef 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 CREATOR 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 in order 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 typedefs 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:

```
uint32_t myData[5]
```

This example data storage can be a token with typedef of `uint32_t` 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 cycles 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
#ifndef DEFINETYPES
    typedef uint8_t tokTypeSensorName[10];
    typedef struct {
        uint8_t initValues[5];
        uint8_t reportInterval;
        uint16_t calibrationValue;
    } tokTypeSensorParameters;
#endif //DEFINETYPES
#ifndef 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 entries 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.

6.28.2 Macro Definition Documentation

6.28.2.1 #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

<i>data</i>	A pointer to where the token data should be placed.
<i>token</i>	The token name used in DEFINE_*_TOKEN , prepended with TOKEN_ .

Definition at line [276](#) of file [token.h](#).

6.28.2.2 #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

<i>data</i>	A pointer to where the token data should be placed.
<i>token</i>	The token name used in DEFINE_*_TOKEN , prepended with TOKEN_ .

Definition at line [291](#) of file [token.h](#).

6.28.2.3 #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

<i>data</i>	A pointer to where the token data should be placed.
<i>token</i>	The token name used in <code>DEFINE_*_TOKEN</code> , prepended with <code>TOKEN_</code> .
<i>index</i>	The index to access in the indexed token.

Definition at line 307 of file [token.h](#).

6.28.2.4 #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

<i>token</i>	The token name used in <code>DEFINE_*_TOKEN</code> , prepended with <code>TOKEN_</code> .
<i>data</i>	A pointer to the data being written.

Definition at line 321 of file [token.h](#).

6.28.2.5 #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

<i>token</i>	The token name used in <code>DEFINE_*_TOKEN</code> , prepended with <code>TOKEN_</code> .
<i>index</i>	The index to access in the indexed token.
<i>data</i>	A pointer to where the token data should be placed.

Definition at line 338 of file [token.h](#).

6.28.2.6 `#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

<i>token</i>	The token name used in <code>DEFINE_*_TOKEN</code> , prepended with <code>TOKEN_</code> .
--------------	---

Definition at line 351 of file [token.h](#).

6.28.3 Function Documentation

6.28.3.1 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.

6.29 Simulated EEPROM

Functions

- void [halSimEepromCallback](#) ([EmberStatus](#) status)
- uint8_t [halSimEepromErasePage](#) (void)
- void [halSimEepromStatus](#) (uint16_t *freeWordsUntilFull, uint16_t *totalPageUseCount)

6.29.1 Detailed Description

The Simulated EEPROM system (typically referred to as SimEE) is designed to operate under the [Token Access API](#) and provide a non-volatile storage system. Since the flash write cycles are finite, the Simulated EEPROM's primary purpose is to perform wear leveling across several hardware flash pages, ultimately increasing the number of times tokens may be written before a hardware failure.

The Simulated EEPROM needs to periodically perform a page erase operation to recover storage area for future token writes. The page erase operation requires an ATOMIC block of 21ms. Since this is such a long time to not be able to service any interrupts, the page erase operation is under application control providing the application the opportunity to decide when to perform the operation and complete any special handling needed that might be needed.

Note

The best, safest, and recommended practice is for the application to regularly and always call the function [halSimEepromErasePage\(\)](#) when the application can expect and deal with the page erase delay. [halSimEepromErasePage\(\)](#) will immediately return if there is nothing to erase. If there is something that needs to be erased, doing so as regularly and as soon as possible will keep the SimEE in the healthiest state possible.

`::ERASE_CRITICAL_THRESHOLD` is the metric the freePtr is compared against. This metric is set to about 3/4 full. The freePtr is a marker used internally by the Simulated EEPROM to track where data ends and where available write space begins. If the freePtr crosses this threshold, [halSimEepromCallback\(\)](#) will be called with an EmberStatus of `EMBER_SIM_EEPROM_ERASE_PAGE_RED`, indicating a critical need for the application to call [halSimEepromErasePage\(\)](#) which will erase a hardware page and provide fresh storage for the Simulated EEPROM to write token data. If freePtr is less than the threshold, the callback will have an EmberStatus of `EMBER_SIM_EEPROM_ERASE_PAGE_GREEN` indicating the application should call [halSimEepromErasePage\(\)](#) at its earliest convenience, but doing so is not critically important at this time.

Some functions in this file return an [EmberStatus](#) value. See [error-def.h](#) for definitions of all [EmberStatus](#) return values.

See [hal/micro/sim-eeprom.h](#) for source code.

6.29.2 Function Documentation

6.29.2.1 void halSimEepromCallback (EmberStatus *status*)

The Simulated EEPROM callback function, implemented by the application.

Parameters

<i>status</i>	An EmberStatus error code indicating one of the conditions described below.
---------------	---

This callback will report an EmberStatus of [EMBER_SIM EEPROM_ERASE_PAGE_GREEN](#) whenever a token is set and a page needs to be erased. If the main application loop does not periodically call [halSimEepromErasePage\(\)](#), it is best to then erase a page in response to [EMBER_SIM EEPROM_ERASE_PAGE_GREEN](#).

This callback will report an EmberStatus of [EMBER_SIM EEPROM_ERASE_PAGE_RED](#) when the pages *must* be erased to prevent data loss. [halSimEepromErasePage\(\)](#) needs to be called until it returns 0 to indicate there are no more pages that need to be erased. Ignoring this indication and not erasing the pages will cause dropping the new data trying to be written.

This callback will report an EmberStatus of [EMBER_SIM EEPROM_FULL](#) when the new data cannot be written due to unerased pages. **Not erasing pages regularly, not erasing in response to EMBER_SIM EEPROM_ERASE_PAGE_GREEN, or not erasing in response to EMBER_SIM EEPROM_ERASE_PAGE_RED will cause EMBER_SIM EEPROM_FULL and the new data will be lost!.** Any future write attempts will be lost as well.

This callback will report an EmberStatus of [EMBER_SIM EEPROM_REPAIRING](#) when the Simulated EEPROM needs to repair 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 a new chip or changing token definitions.

If the callback ever reports the status [EMBER_ERR_FLASH_WRITE_INHIBITED](#) or [EMBER_ERR_FLASH_VERIFY_FAILED](#), this indicates a catastrophic failure in flash writing, meaning either the address being written is not empty or the write itself has failed. If [EMBER_ERR_FLASH_WRITE_INHIBITED](#) is encountered, the function `::halInternalSimEeRepair(false)` should be called and the chip should then be reset to allow proper initialization to recover. If [EMBER_ERR_FLASH_VERIFY_FAILED](#) is encountered the Simulated EEPROM (and tokens) on the specific chip with this error should not be trusted anymore.

6.29.2.2 uint8_t halSimEepromErasePage (void)

Erases a hardware flash page, if needed.

This function can be called at anytime from anywhere in the application (except ISRs) and will only take effect if needed (otherwise it will return immediately). Since this function takes 21ms to erase a hardware page during which interrupts cannot be serviced, it is preferable to call this function while in a state that can withstand being unresponsive for so long. The Simulated EEPROM will periodically request through the [halSimEepromCallback\(\)](#) that a page be erased. The Simulated EEPROM will never erase a page (which could result in data loss) and relies entirely on the application to call this function to approve a page erase (only one erase per call to this function).

The Simulated EEPROM depends on the ability to move between two Virtual Pages, which are comprised of multiple hardware pages. Before moving to the unused Virtual Page, all hardware pages comprising the unused Virtual Page must be erased first. The erase time of a hardware flash page is 21ms. During this time the chip will be unresponsive and unable to service an interrupt or execute any code (due to the flash being unavailable during the erase procedure). This function is used to trigger a page erase.

Returns

A count of how many virtual pages are left to be erased. This return value allows for calling code to easily loop over this function until the function returns 0.

6.29.2.3 void halSimEepromStatus (uint16_t * *freeWordsUntilFull*, uint16_t * *totalPageUseCount*)

Provides two basic statistics.

- The number of unused words until SimEE is full
- The total page use count

There is a lot of management and state processing involved with the Simulated EEPROM, and most of it has no practical purpose in the application. These two parameters provide a simple metric for knowing how soon the Simulated EEPROM will be full (-::*freeWordsUntilFull*) and how many times (approximatly) SimEE has rotated pysical flash pages (-::*totalPageUseCount*).

Parameters

<i>freeWords- UntilFull</i>	Number of unused words available to SimEE until the SimEE is full and would trigger an EMBER_SIM_EEPROM_ERASE_PAGE_RED then EMBER_SIM_EEPROM_FULL callback.
<i>totalPage- UseCount</i>	The value of the highest page counter indicating how many times the Simulated EEPROM has rotated physical flash pages (and approximate write cycles).

6.30 Simulated EEPROM 2

6.31 Sample APIs for Peripheral Access

Modules

- [Serial UART Communication](#)
- [Button Control](#)
- [Buzzer Control](#)
- [LED Control](#)
- [Flash Memory Control](#)

6.31.1 Detailed Description

These are sample API for accessing peripherals and can be modified as needed for your applications.

6.32 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 `SerialParity` { `DEFINE_PARITY`, `DEFINE_PARITY`, `DEFINE_PARITY` }

Functions

- void `halHostFlushBuffers` (void)
- uint16_t `halHostEnqueueTx` (const uint8_t *data, uint16_t length)
- void `halHostFlushTx` (void)
- uint16_t `serialCopyFromRx` (const uint8_t *data, uint16_t length)
- void `emLoadSerialTx` (void)

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` (uint8_t port, `SerialBaudRate` rate, `SerialParity` parity, uint8_t stopBits)
- void `halInternalPowerDownUart` (void)
- void `halInternalPowerUpUart` (void)
- void `halInternalStartUartTx` (uint8_t port)

- void [halInternalStopUartTx](#) (uint8_t port)
- [EmberStatus halInternalForceWriteUartData](#) (uint8_t port, uint8_t *data, uint8_t length)
- [EmberStatus halInternalForceReadUartByte](#) (uint8_t port, uint8_t *dataByte)
- void [halInternalWaitUartTxComplete](#) (uint8_t port)
- void [halInternalRestartUart](#) (void)
- bool [halInternalUartFlowControlRxIsEnabled](#) (uint8_t port)
- bool [halInternalUartXonRefreshDone](#) (uint8_t port)
- bool [halInternalUartTxIsIdle](#) (uint8_t port)
- bool [serialDropPacket](#) (void)
- #define [halInternalUartFlowControl](#)(port)
- #define [halInternalUartRxPump](#)(port)
- #define [halInternalUart1FlowControlRxIsEnabled\(\)](#)
- #define [halInternalUart1XonRefreshDone\(\)](#)
- #define [halInternalUart1TxIsIdle\(\)](#)

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, uint8_t port)

Virtual UART API

API used by the stack in debug builds to receive data arriving over the virtual UART.

- void [halStackReceiveVuartMessage](#) (uint8_t *data, uint8_t length)

6.32.1 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.

6.32.2 Macro Definition Documentation

6.32.2.1 #define EMBER_SERIAL_UNUSED

A numerical definition for a possible serial mode the code can test for.

Definition at line 93 of file [hal/micro/serial.h](#).

6.32.2.2 #define EMBER_SERIAL_FIFO

A numerical definition for a possible serial mode the code can test for.

Definition at line [94](#) of file [hal/micro/serial.h](#).

6.32.2.3 #define EMBER_SERIAL_BUFFER

A numerical definition for a possible serial mode the code can test for.

Definition at line [95](#) of file [hal/micro/serial.h](#).

6.32.2.4 #define EMBER_SERIAL_LOWLEVEL

A numerical definition for a possible serial mode the code can test for.

Definition at line [96](#) of file [hal/micro/serial.h](#).

6.32.2.5 #define FIFO_ENQUEUE(*queue*, *data*, *size*)

Macro that enqueues a byte of data in a FIFO queue.

Parameters

<i>queue</i>	Pointer to the FIFO queue.
<i>data</i>	Data byte to be enqueueed.
<i>size</i>	Size used to control the wrap-around of the FIFO pointers.

Definition at line [283](#) of file [hal/micro/serial.h](#).

6.32.2.6 #define FIFO_DEQUEUE(*queue*, *size*)

Macro that de-queues a byte of data from a FIFO queue.

Parameters

<i>queue</i>	Pointer to the FIFO queue.
<i>size</i>	Size used to control the wrap-around of the FIFO pointers.

Definition at line [298](#) of file [hal/micro/serial.h](#).

6.32.2.7 #define halInternalUartFlowControl(*port*)

This function is used in FIFO mode when flow control is enabled. It is called from [ember-SerialReadByte\(\)](#), and based on the number of bytes used in the uart receive queue, decides when to tell the host it may resume transmission.

Parameters

<i>port</i>	Serial port number (0 or 1). (Does nothing for port 0)
-------------	--

Definition at line [498](#) of file [hal/micro/serial.h](#).

6.32.2.8 #define hallInternalUartRxPump(port)

This function exists only in Buffer Mode on the EM2xx and in software UART (SOF-TUART) mode on the EM3xx. This function is called by [emberSerialReadByte\(\)](#). It is responsible for maintaining synchronization between the emSerialRxQueue and the UART DMA.

Parameters

<i>port</i>	Serial port number (0 or 1).
-------------	------------------------------

Definition at line [511](#) of file [hal/micro/serial.h](#).

6.32.2.9 #define hallInternalUart1FlowControlRxIsEnabled()

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

<i>port</i>	Serial port number (0 or 1). (Does nothing for port 0)
-------------	--

Definition at line [528](#) of file [hal/micro/serial.h](#).

6.32.2.10 #define hallInternalUart1XonRefreshDone()

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

<i>port</i>	Serial port number (0 or 1). (Does nothing for port 0)
-------------	--

Definition at line [535](#) of file [hal/micro/serial.h](#).

6.32.2.11 #define hallInternalUart1TxIsIdle()

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

<i>port</i>	Serial port number (0 or 1). (Does nothing for port 0)
-------------	--

Definition at line [547](#) of file [hal/micro/serial.h](#).

6.32.3 Enumeration Type Documentation

6.32.3.1 enum SerialBaudRate

Assign numerical values for variables that hold Baud Rate parameters.

Enumerator:

Definition at line 311 of file [hal/micro/serial.h](#).

6.32.3.2 enum SerialParity

CORTEXM3 EFM32 MICRO.

Assign numerical values for the types of parity. Use for variables that hold Parity parameters.

Enumerator:

DEFINE_PARITY

Definition at line 381 of file [hal/micro/serial.h](#).

6.32.4 Function Documentation

6.32.4.1 EmberStatus halInternalUartInit (uint8_t *port*, SerialBaudRate *rate*, SerialParity *parity*, uint8_t *stopBits*)

Initializes the UART to the given settings (same parameters as [emberSerialInit\(\)](#)).

Parameters

<i>port</i>	Serial port number (0 or 1).
<i>rate</i>	Baud rate (see SerialBaudRate).
<i>parity</i>	Parity value (see SerialParity).
<i>stopBits</i>	Number of stop bits.

Returns

An error code if initialization failed (such as invalid baud rate), otherwise EMBER_SUCCESS.

6.32.4.2 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.

6.32.4.3 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.

6.32.4.4 void halInternalStartUartTx (uint8_t *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 progress.

Parameters

<i>port</i>	Serial port number (0 or 1).
-------------	------------------------------

6.32.4.5 void halInternalStopUartTx (uint8_t *port*)

Called by serial code to stop any interrupt-driven serial transmission currently in progress.

Parameters

<i>port</i>	Serial port number (0 or 1).
-------------	------------------------------

6.32.4.6 EmberStatus halInternalForceWriteUartData (uint8_t port, uint8_t * data, uint8_t 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

<i>port</i>	Serial port number (0 or 1).
<i>data</i>	Pointer to the data to be transmitted.
<i>length</i>	The length of data to be transmitted

6.32.4.7 EmberStatus halInternalForceReadUartByte (uint8_t port, uint8_t * 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

<i>port</i>	Serial port number (0 or 1).
<i>dataByte</i>	The byte to receive data into.

6.32.4.8 void halInternalWaitUartTxComplete (uint8_t port)

Blocks until the UART has finished transmitting any data in its hardware registers.

Parameters

<i>port</i>	Serial port number (0 or 1).
-------------	------------------------------

6.32.4.9 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.)

6.32.4.10 bool halInternalUartFlowControlRxIsEnabled (uint8_t port)

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 if the host is able to send.

6.32.4.11 bool halInternalUartXonRefreshDone (uint8_t port)

When Xon/Xoff flow control is used, returns true if the host is not being held off and XON refreshing is complete.

6.32.4.12 `bool halInternalUartTxIsIdle (uint8_t port)`

Returns true if the uart transmitter is idle, including the transmit shift register.

6.32.4.13 `bool serialDropPacket (void)`

Testing function implemented by the upper layer. Determines whether the next packet should be dropped. Returns true if the next packet should be dropped, false otherwise.

6.32.4.14 `void emSerialBufferNextMessage (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

<code>q</code>	Pointer to the buffer queue structure for the port.
----------------	---

6.32.4.15 `void emSerialBufferNextBlock (EmSerialBufferQueue * q, uint8_t 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

<code>q</code>	Pointer to the buffer queue structure for the port.
<code>port</code>	Serial port number (0 or 1).

6.32.4.16 `void halStackReceiveVuartMessage (uint8_t * data, uint8_t 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

<code>data</code>	Pointer to the the data received
<code>length</code>	Length of the data received

6.32.4.17 `void halHostFlushBuffers (void)`

6.32.4.18 `uint16_t halHostEnqueueTx (const uint8_t * data, uint16_t length)`

6.32.4.19 `void halHostFlushTx (void)`

6.32.4.20 `uint16_t serialCopyFromRx (const uint8_t * data, uint16_t length)`

6.32.4.21 `void emLoadSerialTx (void)`

6.33 Button Control

Functions

- void [halInternalInitButton](#) (void)
- uint8_t [halButtonState](#) (uint8_t button)
- uint8_t [halButtonPinState](#) (uint8_t button)
- void [halButtonIsr](#) (uint8_t button, uint8_t state)

Button State Definitions

A set of numerical definitions for use with the button APIs indicating the state of a button.

- #define [BUTTON_PRESSED](#)
- #define [BUTTON_RELEASED](#)

6.33.1 Detailed Description

Sample API functions for using push-buttons. See [button.h](#) for source code.

6.33.2 Macro Definition Documentation

6.33.2.1 #define BUTTON_PRESSED

Button state is pressed.

Definition at line 24 of file [button.h](#).

6.33.2.2 #define BUTTON_RELEASED

Button state is released.

Definition at line 28 of file [button.h](#).

6.33.3 Function Documentation

6.33.3.1 void [halInternalInitButton](#) (void)

Initializes the buttons. This function is automatically called by [halInit\(\)](#).

6.33.3.2 uint8_t [halButtonState](#) (uint8_t button)

Returns the current state (pressed or released) of a button.

Note

This function is correlated with [halButtonIsr\(\)](#) and so returns the shadow state rather than reading the actual state of the pin.

Parameters

<i>button</i>	The button being queried, either BUTTON0 or BUTTON1 as defined in the appropriate BOARD_HEADER.
---------------	---

Returns

BUTTON_PRESSED if the button is pressed or **BUTTON_RELEASED** if the button is not pressed.

6.33.3.3 uint8_t halButtonPinState (uint8_t *button*)

Returns the current state (pressed or released) of the pin associated with a button.

This reads the actual state of the pin and can be used on startup to determine the initial position of the buttons.

Parameters

<i>button</i>	The button being queried, either BUTTON0 or BUTTON1 as defined in the appropriate BOARD_HEADER.
---------------	---

Returns

BUTTON_PRESSED if the button is pressed or **BUTTON_RELEASED** if the button is not pressed.

6.33.3.4 void halButtonIsr (uint8_t *button*, uint8_t *state*)

A callback called in interrupt context whenever a button changes its state.

Application Usage:

Must be implemented by the application. This function should contain the functionality to be executed in response to changes of state in each of the buttons, or callbacks to the appropriate functionality.

Parameters

<i>button</i>	The button which has changed state, either BUTTON0 or BUTTON1 as defined in the appropriate BOARD_HEADER.
<i>state</i>	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.

6.34 Buzzer Control

Functions

- void `halPlayTune_P` (uint8_t PGM *tune, bool 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`

6.34.1 Detailed Description

Sample API functions for playing tunes on a piezo buzzer. See [buzzer.h](#) for source code.

6.34.2 Macro Definition Documentation

6.34.2.1 `#define NOTE_C3`

A note which can be used in tune structure definitions.

Definition at line [23](#) of file [buzzer.h](#).

6.34.2.2 `#define NOTE_Db3`

A note which can be used in tune structure definitions.

Definition at line [24](#) of file [buzzer.h](#).

6.34.2.3 `#define NOTE_D3`

A note which can be used in tune structure definitions.

Definition at line [25](#) of file [buzzer.h](#).

6.34.2.4 `#define NOTE_Eb3`

A note which can be used in tune structure definitions.

Definition at line [26](#) of file [buzzer.h](#).

6.34.2.5 `#define NOTE_E3`

A note which can be used in tune structure definitions.

Definition at line [27](#) of file [buzzer.h](#).

6.34.2.6 `#define NOTE_F3`

A note which can be used in tune structure definitions.

Definition at line [28](#) of file [buzzer.h](#).

6.34.2.7 `#define NOTE_Gb3`

A note which can be used in tune structure definitions.

Definition at line [29](#) of file [buzzer.h](#).

6.34.2.8 `#define NOTE_G3`

A note which can be used in tune structure definitions.

Definition at line [30](#) of file `buzzer.h`.

6.34.2.9 #define NOTE_Ab3

A note which can be used in tune structure definitions.

Definition at line [31](#) of file `buzzer.h`.

6.34.2.10 #define NOTE_A3

A note which can be used in tune structure definitions.

Definition at line [32](#) of file `buzzer.h`.

6.34.2.11 #define NOTE_Bb3

A note which can be used in tune structure definitions.

Definition at line [33](#) of file `buzzer.h`.

6.34.2.12 #define NOTE_B3

A note which can be used in tune structure definitions.

Definition at line [34](#) of file `buzzer.h`.

6.34.2.13 #define NOTE_C4

A note which can be used in tune structure definitions.

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

6.34.2.14 #define NOTE_Db4

A note which can be used in tune structure definitions.

Definition at line [36](#) of file `buzzer.h`.

6.34.2.15 #define NOTE_D4

A note which can be used in tune structure definitions.

Definition at line [37](#) of file `buzzer.h`.

6.34.2.16 #define NOTE_Eb4

A note which can be used in tune structure definitions.

Definition at line [38](#) of file `buzzer.h`.

6.34.2.17 #define NOTE_E4

A note which can be used in tune structure definitions.

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

6.34.2.18 #define NOTE_F4

A note which can be used in tune structure definitions.

Definition at line [40](#) of file [buzzer.h](#).

6.34.2.19 #define NOTE_Gb4

A note which can be used in tune structure definitions.

Definition at line [41](#) of file [buzzer.h](#).

6.34.2.20 #define NOTE_G4

A note which can be used in tune structure definitions.

Definition at line [42](#) of file [buzzer.h](#).

6.34.2.21 #define NOTE_Ab4

A note which can be used in tune structure definitions.

Definition at line [43](#) of file [buzzer.h](#).

6.34.2.22 #define NOTE_A4

A note which can be used in tune structure definitions.

Definition at line [44](#) of file [buzzer.h](#).

6.34.2.23 #define NOTE_Bb4

A note which can be used in tune structure definitions.

Definition at line [45](#) of file [buzzer.h](#).

6.34.2.24 #define NOTE_B4

A note which can be used in tune structure definitions.

Definition at line [46](#) of file [buzzer.h](#).

6.34.2.25 #define NOTE_C5

A note which can be used in tune structure definitions.

Definition at line [47](#) of file [buzzer.h](#).

6.34.2.26 #define NOTE_Db5

A note which can be used in tune structure definitions.

Definition at line [48](#) of file [buzzer.h](#).

6.34.2.27 #define NOTE_D5

A note which can be used in tune structure definitions.

Definition at line [49](#) of file [buzzer.h](#).

6.34.2.28 #define NOTE_Eb5

A note which can be used in tune structure definitions.

Definition at line [50](#) of file [buzzer.h](#).

6.34.2.29 #define NOTE_E5

A note which can be used in tune structure definitions.

Definition at line [51](#) of file [buzzer.h](#).

6.34.2.30 #define NOTE_F5

A note which can be used in tune structure definitions.

Definition at line [52](#) of file [buzzer.h](#).

6.34.2.31 #define NOTE_Gb5

A note which can be used in tune structure definitions.

Definition at line [53](#) of file [buzzer.h](#).

6.34.2.32 #define NOTE_G5

A note which can be used in tune structure definitions.

Definition at line [54](#) of file [buzzer.h](#).

6.34.2.33 #define NOTE_Ab5

A note which can be used in tune structure definitions.

Definition at line [55](#) of file [buzzer.h](#).

6.34.2.34 #define NOTE_A5

A note which can be used in tune structure definitions.

Definition at line [56](#) of file [buzzer.h](#).

6.34.2.35 #define NOTE_Bb5

A note which can be used in tune structure definitions.

Definition at line [57](#) of file [buzzer.h](#).

6.34.2.36 #define NOTE_B5

A note which can be used in tune structure definitions.

Definition at line [58](#) of file [buzzer.h](#).

6.34.3 Function Documentation

6.34.3.1 void halPlayTune_P (uint8_t PGM * *tune*, bool *bkg*)

Plays a tune on the piezo buzzer.

The tune is played in the background if ::bkg is true. Otherwise, the API blocks until the playback of the tune is complete. [halPlayTune_P\(\)](#) is not meant to be called back-to-back.

Parameters

<i>tune</i>	A pointer to tune to play.
<i>bkg</i>	Determines whether the tune plays in the background. If true, tune plays in background; if false, tune plays in foreground.

A tune is implemented as follows:

```
uint8_t PGM hereIamTune[] = {      //All tunes are stored in flash.
    NOTE_B4, 1,                   //Plays the note B4 for 100 milliseconds.
    0, 1,                         //Pause for 100 milliseconds.
    NOTE_B5, 5,                   //Plays the note B5 for 500 milliseconds.
    0, 0                          //NULL terminates the tune.
};
```

6.34.3.2 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.

6.35 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 (bool turnOn)`

6.35.1 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.

6.35.2 Typedef Documentation

6.35.2.1 `typedef enum HalBoardLedPins HalBoardLed`

Ensures that the definitions from the BOARD_HEADER are always used as parameters to the LED functions.

Definition at line 32 of file [led.h](#).

6.35.3 Function Documentation

6.35.3.1 `void halInternalInitLed (void)`

Configures GPIOs pertaining to the control of LEDs.

6.35.3.2 `void halToggleLed (HalBoardLed led)`

Atomically wraps an XOR or similar operation for a single GPIO pin attached to an LED.

Parameters

<code>led</code>	Identifier (from BOARD_HEADER) for the LED to be toggled.
------------------	---

6.35.3.3 `void halSetLed (HalBoardLed led)`

Turns on (sets) a GPIO pin connected to an LED so that the LED turns on.

Parameters

<i>led</i>	Identifier (from BOARD_HEADER) for the LED to turn on.
------------	--

6.35.3.4 void halClearLed (HalBoardLed *led*)

Turns off (clears) a GPIO pin connected to an LED, which turns off the LED.

Parameters

<i>led</i>	Identifier (from BOARD_HEADER) for the LED to turn off.
------------	---

6.35.3.5 void halStackIndicateActivity (bool *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

<i>turnOn</i>	See Usage.
---------------	------------

6.36 Flash Memory Control

Functions

- bool [halFlashEraseIsActive](#) (void)

6.36.1 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. **Additionally, 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.

6.36.2 Function Documentation

6.36.2.1 bool halFlashEraseIsActive (void)

Tells the calling code if a Flash Erase operation is active.

This state is important 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 bool flag: true if Flash Erase is active, false otherwise.

6.37 USB Device Stack Library

Modules

- [USB_COMMON](#)
- [USB_DEVICE](#)

6.37.1 Detailed Description

The source files for the USB device stack resides in the `usb` directory and follows the naming convention: `em_usbdnnn.c/h`.

- [Introduction](#)
- [The device stack API](#)
- [Configuring the device stack](#)

6.37.2 Introduction

The USB device protocol stack provides an API which makes it possible to create USB devices with a minimum of effort. The device stack supports control, bulk and interrupt transfers.

The stack is highly configurable to suit various needs, it does also contain useful debugging features together with several demonstration projects to get you started fast.

We recommend that you read through this documentation, then proceed to build and test a few example projects before you start designing your own device.

6.37.3 The device stack API

This section contains brief descriptions of the functions in the API. You will find detailed information on input and output parameters and return values by clicking on the hyper-linked function names. It is also a good idea to study the code in the USB demonstration projects.

Your application code must include one header file: [`em_usb.h`](#).

All functions defined in the API can be called from within interrupt handlers.

Pitfalls:

The USB peripheral will fill your receive buffers in quantities of WORD's (4 bytes). Transmit and receive buffers must be WORD aligned, in addition when allocating storage for receive buffers, round size up to next WORD boundary. If it is possible that the host will send more data than your device expects, round buffer size up to the next multiple of maxpacket size for the relevant endpoint to avoid data corruption.

Transmit buffers passed to must be statically allocated because only initiates the transfer. When the host decide to actually perform the transfer, your data must be available.

[`USBD_Init\(\)`](#)

This function is called to register your device and all its properties with the device stack. The application must fill in a [`USBD_Init_TypeDef`](#) structure prior to calling. Refer to

[DeviceInitCallbacks](#) for the optional callback functions defined within this structure. When this function has been called your device is ready to be enumerated by the USB host.

[USBD_Read\(\)](#), [USBD_Write\(\)](#)

These functions initiate data transfers.

initiate a transfer of data *from* host *to* device (an *OUT* transfer in USB terminology).

initiate a transfer of data *from* device *to* host (an *IN* transfer).

When the USB host actually performs the transfer, your application will be notified by means of a callback function which you provide (optionally). Refer to [TransferCallback](#) for details of the callback functionality.

[USBD_AbortTransfer\(\)](#), [USBD_AbortAllTransfers\(\)](#)

These functions terminate transfers that are initiated, but has not yet taken place. If a transfer is initiated with but the USB host never actually perform the transfers, these functions will deactivate the transfer setup to make the USB device endpoint hardware ready for new (and potentially) different transfers.

[USBD_Connect\(\)](#), [USBD_Disconnect\(\)](#)

These functions turns the data-line (D+ or D-) pullup on or off. They can be used to force reenumeration. It's good practice to delay at least one second between to allow the USB host to unload the currently active device driver.

[USBD_EpIsBusy\(\)](#)

Check if an endpoint is busy.

[USBD_StallEp\(\)](#), [USBD_UnStallEp\(\)](#)

These functions stalls or un-stalls an endpoint. This functionality may not be needed by your application, but the USB device stack use them in response to standard setup commands SET_FEATURE and CLEAR_FEATURE. They may be useful when implementing some USB classes, e.g. a mass storage device use them extensively.

[USBD_RemoteWakeup\(\)](#)

Used in SUSPENDED state (see [USB_Status_TypeDef](#)) to signal resume to host. It's the applications responsibility to adhere to the USB standard which states that a device can not signal resume before it has been SUSPENDED for at least 5 ms. The function will also check the configuration descriptor defined by the application to see if it is legal for the device to signal resume.

[USBD_GetUsbState\(\)](#)

Returns the device USB state (see [USBD_State_TypeDef](#)). Refer to Figure 9-1. "Device State Diagram" in the USB revision 2.0 specification.

[USBD_GetUsbStateName\(\)](#)

Returns a text string naming a given USB device state.

The transfer complete callback function:

[USB_XferCompleteCb_TypeDef\(\)](#) is called when a transfer completes. It is called with three parameters, the status of the transfer, the number of bytes transferred and the number of bytes remaining. It may not always be needed to have a callback on transfer completion, but you should keep in mind that a transfer may be aborted when you least expect it. A transfer will be aborted if host stalls the endpoint, if host resets your device, if host unconfigures your device or if you unplug your device cable and the device is selfpowered. is also called if your application use calls.

Note

This callback is called from within an interrupt handler with interrupts disabled.

Optional callbacks passed to the stack via the [USBD_Init\(\)](#) function:

These callbacks are all optional, and it is up to the application programmer to decide if the application needs the functionality they provide.

Note

These callbacks are all called from within an interrupt handler with interrupts disabled.

[USBD_UsbResetCb_TypeDef\(\)](#) is called each time reset signalling is sensed on the USB wire.

[USBD_SofIntCb_TypeDef\(\)](#) is called with framenumber as a parameter on each SOF interrupt.

[USBD_DeviceStateChangeCb_TypeDef\(\)](#) is called whenever the device state change. Useful for detecting e.g. SUSPENDED state change in order to reduce current consumption of buspowered devices. The USB HID keyboard example project has a good example on how to use this callback.

[USBD_IsSelfPoweredCb_TypeDef\(\)](#) is called by the device stack when host queries the device with a standard setup GET_STATUS command to check if the device is currently selfpowered or buspowered. This feature is only applicable on selfpowered devices which also works when only buspower is available.

[USBD_SetupCmdCb_TypeDef\(\)](#) is called each time a setup command is received from host. Use this callback to override or extend the default handling of standard setup commands, and to implement class or vendor specific setup commands. The USB HID keyboard example project has a good example on how to use this callback.

6.37.4 Configuring the device stack

Your application must provide a header file named *usbconfig.h*. This file must contain the following #define:

```
#define NUM_EP_USED n      // Your application use 'n' endpoints in
                           // addition to endpoint 0.
```

6.38 USB_COMMON

Data Structures

- struct [USB_Setup_TypeDef](#)
USB Setup request package.
- struct [USB_DeviceDescriptor_TypeDef](#)
USB Device Descriptor.
- struct [USB_ConfigurationDescriptor_TypeDef](#)
USB Configuration Descriptor.
- struct [USB_InterfaceDescriptor_TypeDef](#)
USB Interface Descriptor.
- struct [USB_EndpointDescriptor_TypeDef](#)
USB Endpoint Descriptor.
- struct [USB_StringDescriptor_TypeDef](#)
USB String Descriptor.

Macros

- #define [USB_SETUP_DIR_OUT](#)
- #define [USB_SETUP_DIR_IN](#)
- #define [USB_SETUP_DIR_MASK](#)
- #define [USB_SETUP_DIR_D2H](#)
- #define [USB_SETUP_DIR_H2D](#)
- #define [USB_SETUP_TYPE_STANDARD](#)
- #define [USB_SETUP_TYPE_CLASS](#)
- #define [USB_SETUP_TYPE_VENDOR](#)
- #define [USB_SETUP_TYPE_STANDARD_MASK](#)
- #define [USB_SETUP_TYPE_CLASS_MASK](#)
- #define [USB_SETUP_TYPE_VENDOR_MASK](#)
- #define [USB_SETUP_RECIPIENT_DEVICE](#)
- #define [USB_SETUP_RECIPIENT_INTERFACE](#)
- #define [USB_SETUP_RECIPIENT_ENDPOINT](#)
- #define [USB_SETUP_RECIPIENT_OTHER](#)
- #define [GET_STATUS](#)
- #define [CLEAR_FEATURE](#)
- #define [SET_FEATURE](#)
- #define [SET_ADDRESS](#)
- #define [GET_DESCRIPTOR](#)
- #define [SET_DESCRIPTOR](#)
- #define [GET_CONFIGURATION](#)
- #define [SET_CONFIGURATION](#)
- #define [GET_INTERFACE](#)
- #define [SET_INTERFACE](#)
- #define [SYNCH_FRAME](#)
- #define [USB HID_GET_REPORT](#)
- #define [USB HID_GET_IDLE](#)
- #define [USB HID_SET_REPORT](#)
- #define [USB HID_SET_IDLE](#)

- #define USB_HID_SET_PROTOCOL
- #define USB_CDC_SETLINECODING
- #define USB_CDC_GETLINECODING
- #define USB_CDC_SETCTRLLINESSTATE
- #define USB_MSD_BOTRESET
- #define USB_MSD_GETMAXLUN
- #define USB_DEVICE_DESCRIPTOR
- #define USB_CONFIG_DESCRIPTOR
- #define USB_STRING_DESCRIPTOR
- #define USB_INTERFACE_DESCRIPTOR
- #define USB_ENDPOINT_DESCRIPTOR
- #define USB_DEVICE_QUALIFIER_DESCRIPTOR
- #define USB_OTHER_SPEED_CONFIG_DESCRIPTOR
- #define USB_INTERFACE_POWER_DESCRIPTOR
- #define USB_HUB_DESCRIPTOR
- #define USB_HID_DESCRIPTOR
- #define USB_HID_REPORT_DESCRIPTOR
- #define USB_CS_INTERFACE_DESCRIPTOR
- #define USB_DEVICE_DESCSIZE
- #define USB_CONFIG_DESCSIZE
- #define USB_INTERFACE_DESCSIZE
- #define USB_ENDPOINT_DESCSIZE
- #define USB_DEVICE_QUALIFIER_DECSIZE
- #define USB_OTHER_SPEED_CONFIG_DECSIZE
- #define USB_HID_DESCSIZE
- #define USB_CDC_HEADER_FND_DECSIZE
- #define USB_CDC_CALLMNG_FND_DECSIZE
- #define USB_CDC_ACM_FND_DECSIZE
- #define USB_EP0_SIZE
- #define USB_MAX_EP_SIZE
- #define USB_EPTYPE_CTRL
- #define USB_EPTYPE_ISOC
- #define USB_EPTYPE_BULK
- #define USB_EPTYPE_INTR
- #define USB_EP_DIR_IN
- #define USB_SETUP_PKT_SIZE
- #define USB_EPNUM_MASK
- #define USB_LANGID_ENUS
- #define USB_MAX_DEVICE_ADDRESS
- #define CONFIG_DESC_BM_REMOTEWAKEUP
- #define CONFIG_DESC_BM_SELFPOWERED
- #define CONFIG_DESC_BM_RESERVED_D7
- #define CONFIG_DESC_BM_TRANSFERTYPE
- #define CONFIG_DESC_MAXPOWER_mA(x)
- #define DEVICE_IS_SELFPOWERED
- #define REMOTE_WAKEUP_ENABLED
- #define USB_FEATURE_ENDPOINT_HALT
- #define USB_FEATURE_DEVICE_REMOTE_WAKEUP
- #define HUB_FEATURE_PORT_RESET
- #define HUB_FEATURE_PORT_POWER

- #define HUB_FEATURE_C_PORT_CONNECTION
- #define HUB_FEATURE_C_PORT_RESET
- #define HUB_FEATURE_PORT_INDICATOR
- #define USB_CLASS_CDC
- #define USB_CLASS_CDC_DATA
- #define USB_CLASS_CDC_ACM
- #define USB_CLASS_CDC_HFN
- #define USB_CLASS_CDC_CMNGFN
- #define USB_CLASS_CDC_ACMFN
- #define USB_CLASS_CDC_UNIONFN
- #define USB_CLASS_HID
- #define USB_CLASS_HID_KEYBOARD
- #define USB_CLASS_HID_MOUSE
- #define USB_CLASS_HUB
- #define USB_CLASS_MSD
- #define USB_CLASS_MSD_BOT_TRANSPORT
- #define USB_CLASS_MSD_SCSI_CMDSET
- #define USB_CLASS_MSD_CSW_CMDPASSED
- #define USB_CLASS_MSD_CSW_CMDFAILED
- #define USB_CLASS_MSD_CSW_PHASEERROR
- #define PORT_FULL_SPEED
- #define PORT_LOW_SPEED
- #define nibble2Ascii(n)
- #define STATIC_CONST_STRING_DESC(_name,...)
- #define STATIC_CONST_STRING_DESC_LANGID(_name, x, y)
- #define UBUF(x, y)
- #define STATIC_UBUF(x, y)

Typedefs

- typedef int(* **USB_XferCompleteCb_TypeDef**)(USB_Status_TypeDef status, uint32_t xferred, uint32_t remaining)
- typedef void(* **USBTIMER_Callback_TypeDef**)(void)

Enumerations

- enum **USB_Status_TypeDef** {

USB_STATUS_OK, USB_STATUS_REQ_ERR, USB_STATUS_EP_BUSY, USB_STATUS_REQ_UNHANDLED,

USB_STATUS_ILLEGAL, USB_STATUS_EP_STALLED, USB_STATUS_EP_ABORTED, USB_STATUS_EP_ERROR,

USB_STATUS_EP_NAK, USB_STATUS_DEVICE_UNCONFIGURED, USB_STATUS_DEVICE_SUSPENDED, USB_STATUS_DEVICE_RESET,

USB_STATUS_TIMEOUT, USB_STATUS_DEVICE_REMOVED, USB_STATUS_HC_BUSY, USB_STATUS_DEVICE_MALFUNCTION,

USB_STATUS_PORT_OVERCURRENT }

Functions

- void [USBTIMER_DelayMs](#) (uint32_t msec)
- void [USBTIMER_DelayUs](#) (uint32_t usec)
- void [USBTIMER_Init](#) (void)
- void [USBTIMER_Start](#) (uint32_t id, uint32_t timeout, [USBTIMER_Callback_Type-Def](#) callback)
- void [USBTIMER_Stop](#) (uint32_t id)

6.38.1 Detailed Description

Sample API functions for using USB. See [em_usb.h](#) for source code.

6.38.2 Macro Definition Documentation

6.38.2.1 #define USB_SETUP_DIR_OUT

Setup request data stage OUT direction value.

Definition at line [80](#) of file [em_usb.h](#).

6.38.2.2 #define USB_SETUP_DIR_IN

Setup request data stage IN direction value.

Definition at line [81](#) of file [em_usb.h](#).

6.38.2.3 #define USB_SETUP_DIR_MASK

Setup request data stage direction mask.

Definition at line [82](#) of file [em_usb.h](#).

Referenced by [USBD_Init\(\)](#).

6.38.2.4 #define USB_SETUP_DIR_D2H

Setup request data stage IN direction mask.

Definition at line [83](#) of file [em_usb.h](#).

6.38.2.5 #define USB_SETUP_DIR_H2D

Setup request data stage OUT direction mask.

Definition at line [84](#) of file [em_usb.h](#).

6.38.2.6 #define USB_SETUP_TYPE_STANDARD

Standard setup request value.

Definition at line [87](#) of file [em_usb.h](#).

6.38.2.7 #define USB_SETUP_TYPE_CLASS

Class setup request value.

Definition at line [88](#) of file [em_usb.h](#).

6.38.2.8 #define USB_SETUP_TYPE_VENDOR

Vendor setup request value.

Definition at line [89](#) of file [em_usb.h](#).

6.38.2.9 #define USB_SETUP_TYPE_STANDARD_MASK

Standard setup request mask.

Definition at line [90](#) of file [em_usb.h](#).

6.38.2.10 #define USB_SETUP_TYPE_CLASS_MASK

Class setup request mask.

Definition at line [91](#) of file [em_usb.h](#).

6.38.2.11 #define USB_SETUP_TYPE_VENDOR_MASK

Vendor setup request mask.

Definition at line [92](#) of file [em_usb.h](#).

6.38.2.12 #define USB_SETUP_RECIPIENT_DEVICE

Setup request device recipient value.

Definition at line [95](#) of file [em_usb.h](#).

6.38.2.13 #define USB_SETUP_RECIPIENT_INTERFACE

Setup request interface recipient value.

Definition at line [96](#) of file [em_usb.h](#).

6.38.2.14 #define USB_SETUP_RECIPIENT_ENDPOINT

Setup request endpoint recipient value.

Definition at line [97](#) of file [em_usb.h](#).

6.38.2.15 #define USB_SETUP_RECIPIENT_OTHER

Setup request other recipient value.

Definition at line [98](#) of file [em_usb.h](#).

6.38.2.16 #define GET_STATUS

Standard setup request GET_STATUS.

Definition at line 101 of file [em_usb.h](#).

6.38.2.17 #define CLEAR_FEATURE

Standard setup request CLEAR_FEATURE.

Definition at line 102 of file [em_usb.h](#).

6.38.2.18 #define SET_FEATURE

Standard setup request SET_FEATURE.

Definition at line 103 of file [em_usb.h](#).

6.38.2.19 #define SET_ADDRESS

Standard setup request SET_ADDRESS.

Definition at line 104 of file [em_usb.h](#).

6.38.2.20 #define GET_DESCRIPTOR

Standard setup request GET_DESCRIPTOR.

Definition at line 105 of file [em_usb.h](#).

6.38.2.21 #define SET_DESCRIPTOR

Standard setup request SET_DESCRIPTOR.

Definition at line 106 of file [em_usb.h](#).

6.38.2.22 #define GET_CONFIGURATION

Standard setup request GET_CONFIGURATION.

Definition at line 107 of file [em_usb.h](#).

6.38.2.23 #define SET_CONFIGURATION

Standard setup request SET_CONFIGURATION.

Definition at line 108 of file [em_usb.h](#).

6.38.2.24 #define GET_INTERFACE

Standard setup request GET_INTERFACE.

Definition at line 109 of file [em_usb.h](#).

6.38.2.25 #define SET_INTERFACE

Standard setup request SET_INTERFACE.

Definition at line 110 of file [em_usb.h](#).

6.38.2.26 #define SYNCH_FRAME

Standard setup request SYNCH_FRAME.

Definition at line 111 of file [em_usb.h](#).

6.38.2.27 #define USB_HID_GET_REPORT

HID class setup request GET_REPORT.

Definition at line 114 of file [em_usb.h](#).

6.38.2.28 #define USB_HID_GET_IDLE

HID class setup request GET_IDLE.

Definition at line 115 of file [em_usb.h](#).

6.38.2.29 #define USB_HID_SET_REPORT

HID class setup request SET_REPORT.

Definition at line 116 of file [em_usb.h](#).

6.38.2.30 #define USB_HID_SET_IDLE

HID class setup request SET_IDLE.

Definition at line 117 of file [em_usb.h](#).

6.38.2.31 #define USB_HID_SET_PROTOCOL

HID class setup request SET_PROTOCOL.

Definition at line 118 of file [em_usb.h](#).

6.38.2.32 #define USB_CDC_SETLINECODING

CDC class setup request SET_LINE_CODING.

Definition at line 119 of file [em_usb.h](#).

6.38.2.33 #define USB_CDC_GETLINECODING

CDC class setup request GET_LINE_CODING.

Definition at line 120 of file [em_usb.h](#).

6.38.2.34 #define USB_CDC_SETCTRLLINESTATE

CDC class setup request SET_CONTROL_LINE_STATE.

Definition at line 121 of file [em_usb.h](#).

6.38.2.35 #define USB_MSD_BOTRESET

MSD class setup request Bulk only transfer reset.

Definition at line 122 of file [em_usb.h](#).

6.38.2.36 #define USB_MSD_GETMAXLUN

MSD class setup request Get Max LUN.

Definition at line 123 of file [em_usb.h](#).

6.38.2.37 #define USB_DEVICE_DESCRIPTOR

DEVICE descriptor value.

Definition at line 126 of file [em_usb.h](#).

6.38.2.38 #define USB_CONFIG_DESCRIPTOR

CONFIGURATION descriptor value.

Definition at line 127 of file [em_usb.h](#).

6.38.2.39 #define USB_STRING_DESCRIPTOR

STRING descriptor value.

Definition at line 128 of file [em_usb.h](#).

6.38.2.40 #define USB_INTERFACE_DESCRIPTOR

INTERFACE descriptor value.

Definition at line 129 of file [em_usb.h](#).

6.38.2.41 #define USB_ENDPOINT_DESCRIPTOR

ENDPOINT descriptor value.

Definition at line 130 of file [em_usb.h](#).

Referenced by [USBD_Init\(\)](#).

6.38.2.42 #define USB_DEVICE_QUALIFIER_DESCRIPTOR

DEVICE_QUALIFIER descriptor value.

Definition at line 131 of file [em_usb.h](#).

6.38.2.43 #define USB_OTHER_SPEED_CONFIG_DESCRIPTOR

OTHER_SPEED_CONFIGURATION descriptor value.

Definition at line 132 of file [em_usb.h](#).

6.38.2.44 #define USB_INTERFACE_POWER_DESCRIPTOR

INTERFACE_POWER descriptor value.

Definition at line 133 of file [em_usb.h](#).

6.38.2.45 #define USB_HUB_DESCRIPTOR

HUB descriptor value.

Definition at line 134 of file [em_usb.h](#).

6.38.2.46 #define USB_HID_DESCRIPTOR

HID descriptor value.

Definition at line 135 of file [em_usb.h](#).

6.38.2.47 #define USB_HID_REPORT_DESCRIPTOR

HID REPORT descriptor value.

Definition at line 136 of file [em_usb.h](#).

6.38.2.48 #define USB_CS_INTERFACE_DESCRIPTOR

Audio Class-specific Descriptor Type.

Definition at line 137 of file [em_usb.h](#).

6.38.2.49 #define USB_DEVICE_DESCSIZE

Device descriptor size.

Definition at line 139 of file [em_usb.h](#).

6.38.2.50 #define USB_CONFIG_DESCSIZE

Configuration descriptor size.

Definition at line 140 of file [em_usb.h](#).

6.38.2.51 #define USB_INTERFACE_DECSIZE

Interface descriptor size.

Definition at line 141 of file [em_usb.h](#).

6.38.2.52 #define USB_ENDPOINT_DECSIZE

Endpoint descriptor size.

Definition at line 142 of file [em_usb.h](#).

6.38.2.53 #define USB_DEVICE_QUALIFIER_DECSIZE

Device qualifier descriptor size.

Definition at line 143 of file [em_usb.h](#).

6.38.2.54 #define USB_OTHER_SPEED_CONFIG_DECSIZE

Device other speed configuration descriptor size.

Definition at line 144 of file [em_usb.h](#).

6.38.2.55 #define USB_HID_DECSIZE

HID descriptor size.

Definition at line 145 of file [em_usb.h](#).

6.38.2.56 #define USB_CDC_HEADER_FND_DECSIZE

CDC Header functional descriptor size.

Definition at line 146 of file [em_usb.h](#).

6.38.2.57 #define USB_CDC_CALLMNG_FND_DECSIZE

CDC Call Management functional descriptor size.

Definition at line 147 of file [em_usb.h](#).

6.38.2.58 #define USB_CDC_ACM_FND_DECSIZE

CDC Abstract Control Management functional descriptor size.

Definition at line 148 of file [em_usb.h](#).

6.38.2.59 #define USB_EP0_SIZE

The size of endpoint 0.

Definition at line 151 of file [em_usb.h](#).

Referenced by [USBD_Init\(\)](#).

6.38.2.60 #define USB_MAX_EP_SIZE

The max size of any full speed endpoint.

Definition at line [152](#) of file [em_usb.h](#).

6.38.2.61 #define USB_EPTYPE_CTRL

Endpoint type control.

Definition at line [153](#) of file [em_usb.h](#).

Referenced by [USBD_Init\(\)](#).

6.38.2.62 #define USB_EPTYPE_ISOC

Endpoint type isochron.

Definition at line [154](#) of file [em_usb.h](#).

6.38.2.63 #define USB_EPTYPE_BULK

Endpoint type bulk.

Definition at line [155](#) of file [em_usb.h](#).

6.38.2.64 #define USB_EPTYPE_INTR

Endpoint type interrupt.

Definition at line [156](#) of file [em_usb.h](#).

6.38.2.65 #define USB_EP_DIR_IN

Endpoint direction mask.

Definition at line [157](#) of file [em_usb.h](#).

6.38.2.66 #define USB_SETUP_PKT_SIZE

Setup request packet size.

Definition at line [158](#) of file [em_usb.h](#).

6.38.2.67 #define USB_EPNUM_MASK

Endpoint number mask.

Definition at line [159](#) of file [em_usb.h](#).

Referenced by [USBD_Init\(\)](#).

6.38.2.68 #define USB_LANGID_ENUS

English-United States language id.

Definition at line 160 of file [em_usb.h](#).

6.38.2.69 #define USB_MAX_DEVICE_ADDRESS

Maximum allowable device address.

Definition at line 161 of file [em_usb.h](#).

6.38.2.70 #define CONFIG_DESC_BM_REMOTEWAKEUP

Configuration descriptor attribute macro.

Definition at line 163 of file [em_usb.h](#).

6.38.2.71 #define CONFIG_DESC_BM_SELFPOWERED

Configuration descriptor attribute macro.

Definition at line 164 of file [em_usb.h](#).

6.38.2.72 #define CONFIG_DESC_BM_RESERVED_D7

Configuration descriptor attribute macro.

Definition at line 165 of file [em_usb.h](#).

6.38.2.73 #define CONFIG_DESC_BM_TRANSFERTYPE

Configuration descriptor transfer type bitmask.

Definition at line 166 of file [em_usb.h](#).

Referenced by [USBD_Init\(\)](#).

6.38.2.74 #define CONFIG_DESC_MAXPOWER_mA(x)

Configuration descriptor power macro.

Definition at line 167 of file [em_usb.h](#).

6.38.2.75 #define DEVICE_IS_SELFPOWERED

Standard request GET_STATUS bitmask.

Definition at line 169 of file [em_usb.h](#).

6.38.2.76 #define REMOTE_WAKEUP_ENABLED

Standard request GET_STATUS bitmask.

Definition at line 170 of file [em_usb.h](#).

6.38.2.77 #define USB_FEATURE_ENDPOINT_HALT

Standard request CLEAR/SET_FEATURE bitmask.

Definition at line 171 of file [em_usb.h](#).

6.38.2.78 #define USB_FEATURE_DEVICE_REMOTE_WAKEUP

Standard request CLEAR/SET_FEATURE bitmask.

Definition at line 172 of file [em_usb.h](#).

6.38.2.79 #define HUB_FEATURE_PORT_RESET

HUB class request CLEAR/SET_PORT_FEATURE feature selector.

Definition at line 174 of file [em_usb.h](#).

6.38.2.80 #define HUB_FEATURE_PORT_POWER

HUB class request CLEAR/SET_PORT_FEATURE feature selector.

Definition at line 175 of file [em_usb.h](#).

6.38.2.81 #define HUB_FEATURE_C_PORT_CONNECTION

HUB class request CLEAR/SET_PORT_FEATURE feature selector.

Definition at line 176 of file [em_usb.h](#).

6.38.2.82 #define HUB_FEATURE_C_PORT_RESET

HUB class request CLEAR/SET_PORT_FEATURE feature selector.

Definition at line 177 of file [em_usb.h](#).

6.38.2.83 #define HUB_FEATURE_PORT_INDICATOR

HUB class request CLEAR/SET_PORT_FEATURE feature selector.

Definition at line 178 of file [em_usb.h](#).

6.38.2.84 #define USB_CLASS_CDC

CDC device/interface class code.

Definition at line 180 of file [em_usb.h](#).

6.38.2.85 #define USB_CLASS_CDC_DATA

CDC Data interface class code.

Definition at line 181 of file [em_usb.h](#).

6.38.2.86 #define USB_CLASS_CDC_ACM

CDC Abstract Control Model interface subclass code.

Definition at line 182 of file [em_usb.h](#).

6.38.2.87 #define USB_CLASS_CDC_HFN

CDC class Header Functional Descriptor subtype.

Definition at line 183 of file [em_usb.h](#).

6.38.2.88 #define USB_CLASS_CDC_CMNGFN

CDC class Call Management Functional Descriptor subtype.

Definition at line 184 of file [em_usb.h](#).

6.38.2.89 #define USB_CLASS_CDC_ACMFN

CDC class Abstract Control Management Functional Descriptor subtype.

Definition at line 185 of file [em_usb.h](#).

6.38.2.90 #define USB_CLASS_CDC_UNIONFN

CDC class Union Functional Descriptor subtype.

Definition at line 186 of file [em_usb.h](#).

6.38.2.91 #define USB_CLASS_HID

HID device/interface class code.

Definition at line 188 of file [em_usb.h](#).

6.38.2.92 #define USB_CLASS_HID_KEYBOARD

HID keyboard interface protocol code.

Definition at line 189 of file [em_usb.h](#).

6.38.2.93 #define USB_CLASS_HID_MOUSE

HID mouse interface protocol code.

Definition at line 190 of file [em_usb.h](#).

6.38.2.94 #define USB_CLASS_HUB

HUB device/interface class code.

Definition at line 192 of file [em_usb.h](#).

6.38.2.95 #define USB_CLASS_MSD

MSD device/interface class code.

Definition at line 194 of file [em_usb.h](#).

6.38.2.96 #define USB_CLASS_MSD_BOT_TRANSPORT

MSD Bulk Only Transport protocol.

Definition at line 195 of file [em_usb.h](#).

6.38.2.97 #define USB_CLASS_MSD_SCSI_CMDSET

MSD Subclass SCSI transparent command set.

Definition at line 196 of file [em_usb.h](#).

6.38.2.98 #define USB_CLASS_MSD_CSW_CMDPASSED

MSD BOT Command status wrapper command passed code.

Definition at line 197 of file [em_usb.h](#).

6.38.2.99 #define USB_CLASS_MSD_CSW_CMDFAILED

MSD BOT Command status wrapper command failed code.

Definition at line 198 of file [em_usb.h](#).

6.38.2.100 #define USB_CLASS_MSD_CSW_PHASEERROR

MSD BOT Command status wrapper cmd phase error code.

Definition at line 199 of file [em_usb.h](#).

6.38.2.101 #define PORT_FULL_SPEED

Full speed return value for USBH_GetPortSpeed().

Definition at line 201 of file [em_usb.h](#).

6.38.2.102 #define PORT_LOW_SPEED

Low speed return value for USBH_GetPortSpeed().

Definition at line 202 of file [em_usb.h](#).

6.38.2.103 #define nibble2Ascii(n)

Definition at line 254 of file [em_usb.h](#).

6.38.2.104 #define STATIC_CONST_STRING_DESC(_name, ...)

Definition at line 256 of file [em_usb.h](#).

6.38.2.105 #define STATIC_CONST_STRING_DESC_LANGID(_name, x, y)

Macro for creating USB compliant language string descriptors.

Example: `STATIC_CONST_STRING_DESC_LANGID(langID, 0x04, 0x09);`

Definition at line 279 of file [em_usb.h](#).

6.38.2.106 #define UBUF(x, y)

Macro for creating WORD (4 byte) aligned uint8_t array with size which is a multiple of WORD size.

Example:

`UBUF(rxBuffer, 37); => uint8_t rxBuffer[40];`

Definition at line 304 of file [em_usb.h](#).

6.38.2.107 #define STATIC_UBUF(x, y)

Definition at line 305 of file [em_usb.h](#).

6.38.3 Typedef Documentation

6.38.3.1 typedef int(* USB_XferCompleteCb_TypeDef)(USB_Status_TypeDef status, uint32_t xferred, uint32_t remaining)

USB transfer callback function.

The callback function is called when a transfer has completed. An application should check the status, xferred and optionally the remaining parameters before deciding if the transfer is usable. In the case where the transfer is part of a control request data stage, the callback function should return an appropriate [USB_Status_TypeDef](#) status.

Parameters

in	status	The transfer status. See USB_Status_TypeDef .
in	xferred	Number of bytes actually transferred.
in	remaining	Number of bytes not transferred.

Returns

[USB_STATUS_OK](#) on success, else an appropriate error code.

Definition at line 534 of file [em_usb.h](#).

6.38.3.2 `typedef void(* USBTIMER_Callback_TypeDef)(void)`

USBTIMER callback function.

The callback function is called when an USBTIMER has expired. The callback is done with interrupts disabled.

Definition at line 544 of file [em_usb.h](#).

6.38.4 Enumeration Type Documentation

6.38.4.1 `enum USB_Status_TypeDef`

USB transfer status enumerator.

Enumerator:

- `USB_STATUS_OK`** No errors detected.
- `USB_STATUS_REQ_ERR`** Setup request error.
- `USB_STATUS_EP_BUSY`** Endpoint is busy.
- `USB_STATUS_REQ_UNHANDLED`** Setup request not handled.
- `USB_STATUS_ILLEGAL`** Illegal operation attempted.
- `USB_STATUS_EP_STALLED`** Endpoint is stalled.
- `USB_STATUS_EP_ABORTED`** Endpoint transfer was aborted.
- `USB_STATUS_EP_ERROR`** Endpoint transfer error.
- `USB_STATUS_EP_NAK`** Endpoint NAK'ed transfer request.
- `USB_STATUS_DEVICE_UNCONFIGURED`** Device is unconfigured.
- `USB_STATUS_DEVICE_SUSPENDED`** Device is suspended.
- `USB_STATUS_DEVICE_RESET`** Device is/was reset.
- `USB_STATUS_TIMEOUT`** Transfer timeout.
- `USB_STATUS_DEVICE_REMOVED`** Device was removed.
- `USB_STATUS_HC_BUSY`** Host channel is busy.
- `USB_STATUS_DEVICE_MALFUNCTION`** Malfunctioning device attached.
- `USB_STATUS_PORT_OVERCURRENT`** VBUS shortcircuit/overcurrent failure.

Definition at line 317 of file [em_usb.h](#).

6.38.5 Function Documentation

6.38.5.1 `void USBTIMER_DelayMs (uint32_t msec)`

6.38.5.2 `void USBTIMER_DelayUs (uint32_t usec)`

6.38.5.3 `void USBTIMER_Init (void)`

Referenced by [USBD_Init\(\)](#).

6.38.5.4 void USBTIMER_Start (*uint32_t id*, *uint32_t timeout*,
 USBTIMER_Callback_TypeDef callback)

6.38.5.5 void USBTIMER_Stop (*uint32_t id*)

6.39 USB_DEVICE

Data Structures

- struct [USBD_Init_TypeDef](#)
USB Device stack initialization structure.
- struct [USBD_Callbacks_TypeDef](#)
USB Device stack callback structure.

Typedefs

- typedef void(* [USBD_UsbResetCb_TypeDef](#))(void)
- typedef void(* [USBD_SofIntCb_TypeDef](#))(uint16_t sofNr)
- typedef void(* [USBD_DeviceStateChangeCb_TypeDef](#))([USBD_State_TypeDef](#) oldState, [USBD_State_TypeDef](#) newState)
- typedef bool(* [USBD_IsSelfPoweredCb_TypeDef](#))(void)
- typedef int(* [USBD_SetupCmdCb_TypeDef](#))(const [USB_Setup_TypeDef](#) *setup)
- typedef struct
[USBD_Callbacks_TypeDef](#) [USBD_Callbacks_TypeDef](#)

Enumerations

- enum [USBD_State_TypeDef](#) {
 [USBD_STATE_NONE](#), [USBD_STATE_ATTACHED](#), [USBD_STATE_POWERED](#), [USBD_STATE_DEFAULT](#),
[USBD_STATE_ADDRESSED](#), [USBD_STATE_CONFIGURED](#), [USBD_STATE_SUSPENDED](#), [USBD_STATE_LASTMARKER](#) }

Functions

- void [USBD_AbortAllTransfers](#) (void)
- int [USBD_AbortTransfer](#) (int epAddr)
- void [USBD_Connect](#) (void)
- void [USBD_Disconnect](#) (void)
- bool [USBD_EpIsBusy](#) (int epAddr)
- [USBD_State_TypeDef](#) [USBD_GetUsbState](#) (void)
- const char * [USBD_GetUsbStateName](#) ([USBD_State_TypeDef](#) state)
- int [USBD_Init](#) (const [USBD_Init_TypeDef](#) *p)
- int [USBD_Read](#) (int epAddr, void *data, int byteCount, [USB_XferCompleteCb_TypeDef](#) callback)
- int [USBD_RemoteWakeup](#) (void)
- bool [USBD_SafeToEnterEM2](#) (void)
- int [USBD_StallEp](#) (int epAddr)
- void [USBD_Stop](#) (void)
- int [USBD_UnStallEp](#) (int epAddr)
- int [USBD_Write](#) (int epAddr, void *data, int byteCount, [USB_XferCompleteCb_TypeDef](#) callback)
- void [usbSuspendDsr](#) (void)

6.39.1 Detailed Description

USB DEVICE protocol stack, see [USB Device Stack Library](#) page for detailed documentation. See [em_usbd.c](#) for source code.

6.39.2 Typedef Documentation

6.39.2.1 `typedef void(* USBD_UsbResetCb_TypeDef)(void)`

USB Reset callback function.

Called whenever USB reset signalling is detected on the USB port.

Definition at line [570](#) of file [em_usb.h](#).

6.39.2.2 `typedef void(* USBD_SofIntCb_TypeDef)(uint16_t sofNr)`

USB Start Of Frame (SOF) interrupt callback function.

Called at each SOF interrupt (if enabled),

Parameters

<code>in</code>	<code>sofNr</code>	Current frame number. The value rolls over to 0 after 16383 (0x3FFF).
-----------------	--------------------	---

Definition at line [582](#) of file [em_usb.h](#).

6.39.2.3 `typedef void(* USBD_DeviceStateChangeCb_TypeDef)(USBD_State_TypeDef oldState, USBD_State_TypeDef newState)`

USB State change callback function.

Called whenever the device change state.

Parameters

<code>in</code>	<code>oldState</code>	The device USB state just leaved. See USBD_State_TypeDef .
<code>in</code>	<code>newState</code>	New (the current) USB device state. See USBD_State_TypeDef .

Definition at line [597](#) of file [em_usb.h](#).

6.39.2.4 `typedef bool(* USBD_IsSelfPoweredCb_TypeDef)(void)`

USB power mode callback function.

Called whenever the device stack needs to query if the device is currently self- or bus-powered. Typically when host has issued an [GET_STATUS](#) setup command.

Returns

True if self-powered, false otherwise.

Definition at line 611 of file [em_usb.h](#).

6.39.2.5 `typedef int(* USBD_SetupCmdCb_TypeDef)(const USB_Setup_TypeDef *setup)`

USB setup request callback function.

Called on each setup request received from host. This gives the application a possibility to extend or override standard requests, and to implement class or vendor specific requests. Return `USB_STATUS_OK` if the request is handled, return `USB_STATUS_REQ_ERR` if it is an illegal request or return `USB_STATUS_REQ_UNHANDLED` to pass the request on to the default request handler.

Parameters

<code>in</code>	<code>setup</code>	Pointer to an USB setup packet. See USB_Setup_TypeDef .
-----------------	--------------------	---

Returns

An appropriate status/error code. See [USB_Status_TypeDef](#).

Definition at line 631 of file [em_usb.h](#).

6.39.2.6 `typedef struct USBD_Callbacks_TypeDef USBD_Callbacks_TypeDef`

USB Device stack callback structure.

Callback functions used by the device stack to signal events or query status to/from the application. See [USBD_Init_TypeDef](#). Assign members to NULL if your application don't need a specific callback.

6.39.3 Enumeration Type Documentation**6.39.3.1 `enum USBD_State_TypeDef`**

USB device state enumerator.

Enumerator:

- `USBD_STATE_NONE` Device state is undefined/unknown.
- `USBD_STATE_ATTACHED` Device state is ATTACHED.
- `USBD_STATE_POWERED` Device state is POWERED.
- `USBD_STATE_DEFAULT` Device state is DEFAULT.
- `USBD_STATE_ADDRESSED` Device state is ADDRESSED.
- `USBD_STATE_CONFIGURED` Device state is CONFIGURED.
- `USBD_STATE_SUSPENDED` Device state is SUSPENDED.
- `USBD_STATE_LASTMARKER` Device state enum end marker.

Definition at line 349 of file [em_usb.h](#).

6.39.4 Function Documentation

6.39.4.1 void USBD_AbortAllTransfers (void)

Abort all pending transfers.

Aborts transfers for all endpoints currently in use. Pending transfers on the default endpoint (EP0) are not aborted.

Definition at line 56 of file [em_usbd.c](#).

References [ATOMIC](#), and [USB_STATUS_EP_ABORTED](#).

6.39.4.2 int USBD_AbortTransfer (int epAddr)

Abort a pending transfer on a specific endpoint.

Parameters

in	<i>epAddr</i>	The address of the endpoint to abort.
----	---------------	---------------------------------------

Definition at line 70 of file [em_usbd.c](#).

References [assert](#), [DISABLE_INTERRUPTS](#), [NULL](#), [RESTORE_INTERRUPTS](#), [USB_STATUS_EP_ABORTED](#), [USB_STATUS_OK](#), [USBD_STATE_ADDRESSED](#), and [USBD_STATE_CONFIGURED](#).

6.39.4.3 void USBD_Connect (void)

Start USB device operation.

Device operation is started by connecting a pullup resistor on the appropriate USB data line.

Definition at line 122 of file [em_usbd.c](#).

References [ATOMIC](#).

6.39.4.4 void USBD_Disconnect (void)

Stop USB device operation.

Device operation is stopped by disconnecting the pullup resistor from the appropriate USB data line. Often referred to as a "soft" disconnect.

Definition at line 137 of file [em_usbd.c](#).

References [ATOMIC](#), and [USBD_STATE_SUSPENDED](#).

Referenced by [USBD_Stop\(\)](#).

6.39.4.5 bool USBD_EpIsBusy (int epAddr)

Check if an endpoint is busy doing a transfer.

Parameters

in	<i>epAddr</i>	The address of the endpoint to check.
----	---------------	---------------------------------------

Returns

True if endpoint is busy, false otherwise.

Definition at line 207 of file [em_usbd.c](#).

References [assert](#), and [NULL](#).

6.39.4.6 USBD_State_TypeDef USBD_GetUsbState(void)

Get current USB device state.

Returns

Device USB state. See [USBD_State_TypeDef](#).

Definition at line 173 of file [em_usbd.c](#).

Referenced by [USBD_Read\(\)](#), and [USBD_Write\(\)](#).

6.39.4.7 const char * USBD_GetUsbStateName(USBD_State_TypeDef state)

Get a string naming a device USB state.

Parameters

in	<i>state</i>	Device USB state. See USBD_State_TypeDef .
----	--------------	--

Returns

State name string pointer.

Definition at line 188 of file [em_usbd.c](#).

References [USBD_STATE_LASTMARKER](#).

6.39.4.8 int USBD_Init(const USBD_Init_TypeDef * p)

Initializes USB device hardware and internal protocol stack data structures, then connects the data-line (D+ or D-) pullup resistor to signal host that enumeration can begin.

Note

You may later use [USBD_Disconnect\(\)](#) and [USBD_Connect\(\)](#) to force reenumeration.

Parameters

in	<i>p</i>	Pointer to device initialization struct. See USBD_Init_TypeDef .
----	----------	--

Returns

[USB_STATUS_OK](#) on success, else an appropriate error code.

Definition at line 337 of file [em_usbd.c](#).

References [assert](#), [USB_EndpointDescriptor_TypeDef::bEndpointAddress](#), [USB_EndpointDescriptor_TypeDef::bmAttributes](#), [USBD_Init_TypeDef::bufferingMultiplier](#), [USBD_Init_TypeDef::callbacks](#), [CONFIG_DESC_BM_TRANSFERTYPE](#), [USBD_Init_TypeDef::configDescriptor](#), [USBD_Init_TypeDef::deviceDescriptor](#), [DISABLE_INTERRUPTS](#), [MEMSET](#), [NULL](#), [USBD_Init_TypeDef::numberOfStrings](#), [RESTORE_INTERRUPTS](#), [USBD_Init_TypeDef::stringDescriptors](#), [USB_ENDPOINT_DESCRIPTOR](#), [USB_EP0_SIZE](#), [USB_EPNUM_MASK](#), [USB_EPTYPE_CTRL](#), [USB_REMOTEWKUPEN_STATE](#), [USB_SETUP_DIR_MASK](#), [USB_STATUS_ILLEGAL](#), [USB_STATUS_OK](#), [USBD_STATUS_LASTMARKER](#), [USBD_STATE_NONE](#), [USBTIMER_Init\(\)](#), and [USB_EndpointDescriptor_TypeDef::wMaxPacketSize](#).

6.39.4.9 int USBD_Read (int *epAddr*, void * *data*, int *byteCount*, USB_XferCompleteCb_TypeDef *callback*)

Start a read (OUT) transfer on an endpoint.

Note

The transfer buffer length must be a multiple of 4 bytes in length and WORD (4 byte) aligned. When allocating the buffer, round buffer length up. If it is possible that the host will send more data than your device expects, round buffer size up to the next multiple of maxpacket size.

Parameters

in	<i>epAddr</i>	Endpoint address.
in	<i>data</i>	Pointer to transfer data buffer.
in	<i>byteCount</i>	Transfer length.
in	<i>callback</i>	Function to be called on transfer completion. Supply NULL if no callback is needed. See USB_XferCompleteCb_TypeDef .

Returns

[USB_STATUS_OK](#) on success, else an appropriate error code.

Definition at line 658 of file [em_usbd.c](#).

References [assert](#), [DISABLE_INTERRUPTS](#), [NULL](#), [RESTORE_INTERRUPTS](#), [USB_STATUS_DEVICE_UNCONFIGURED](#), [USB_STATUS_EP_BUSY](#), [USB_STATUS_EP_STALLED](#), [USB_STATUS_OK](#), [USBD_GetUsbState\(\)](#), and [USBD_STATE_CONFIGURED](#).

6.39.4.10 int USBD_RemoteWakeup (void)

Perform a remote wakeup signalling sequence.

Note

It is the responsibility of the application to ensure that remote wakeup is not attempted before the device has been suspended for at least 5 miliseconds. This function should not be called from within an interrupt handler.

Returns

[USB_STATUS_OK](#) on success, else an appropriate error code.

Definition at line [771](#) of file [em_usbd.c](#).

References [elapsedTimeInt16u](#), [halCommonGetInt16uMillisecondTick\(\)](#), [USB_STATUS_ILLEGAL](#), [USB_STATUS_OK](#), [USB_STATUS_TIMEOUT](#), and [USBD_STATE_SUSPENDED](#).

6.39.4.11 bool USBD_SafeToEnterEM2(void)

6.39.4.12 int USBD_StallEp(int epAddr)

Set an endpoint in the stalled (halted) state.

Parameters

in	<i>epAddr</i>	The address of the endpoint to stall.
----	---------------	---------------------------------------

Returns

[USB_STATUS_OK](#) on success, else an appropriate error code.

Definition at line [230](#) of file [em_usbd.c](#).

References [assert](#), [ATOMIC](#), [NULL](#), [USB_STATUS_ILLEGAL](#), and [USB_STATUS_OK](#).

6.39.4.13 void USBD_Stop(void)

Stop USB device stack operation.

The data-line pullup resistor is turned off, USB interrupts are disabled, and finally the USB pins are disabled.

Definition at line [313](#) of file [em_usbd.c](#).

References [USBD_Disconnect\(\)](#), and [USBD_STATE_NONE](#).

6.39.4.14 int USBD_UnStallEp(int epAddr)

Reset stall state on a stalled (halted) endpoint.

Parameters

in	<i>epAddr</i>	The address of the endpoint to un-stall.
----	---------------	--

Returns

[USB_STATUS_OK](#) on success, else an appropriate error code.

Definition at line 273 of file [em_usbd.c](#).

References [assert](#), [ATOMIC](#), [NULL](#), [USB_STATUS_ILLEGAL](#), and [USB_STATUS_OK](#).

6.39.4.15 int USBD_Write (int epAddr, void * data, int byteCount, USB_XferCompleteCb_TypeDef callback)

Start a write (IN) transfer on an endpoint.

Parameters

in	<i>epAddr</i>	Endpoint address.
in	<i>data</i>	Pointer to transfer data buffer. This buffer must be WORD (4 byte) aligned.
in	<i>byteCount</i>	Transfer length.
in	<i>callback</i>	Function to be called on transfer completion. Supply NULL if no callback is needed. See USB_XferCompleteCb_TypeDef .

Returns

[USB_STATUS_OK](#) on success, else an appropriate error code.

Definition at line 539 of file [em_usbd.c](#).

References [assert](#), [DISABLE_INTERRUPTS](#), [NULL](#), [RESTORE_INTERRUPTS](#), [USB_STATUS_DEVICE_UNCONFIGURED](#), [USB_STATUS_EP_BUSY](#), [USB_STATUS_EP_STALLED](#), [USB_STATUS_ILLEGAL](#), [USB_STATUS_OK](#), [USBD_GetUsbState\(\)](#), and [USBD_STATE_CONFIGURED](#).

6.39.4.16 void usbSuspendDsr (void)

USB suspend delayed service routine.

This function keeps the device in a low power state in order to meet USB specification during USB suspend state.

Definition at line 740 of file [em_usbd.c](#).

References [emberStackPowerDown\(\)](#), [halSleep\(\)](#), [SLEEPMODE_IDLE](#), and [USBD_STATE_SUSPENDED](#).

6.40 System Timer Control

Macros

- `#define halIdleForMilliseconds(duration)`

Functions

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

6.40.1 Detailed Description

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

- When used with a 32.768kHz crystal, the system tick is 0.976 milliseconds.
- When used with a 3.6864MHz crystal, the system tick is 1.111 milliseconds.

A single quarter-second tick (as returned by `halCommonGetInt16uQuarterSecondTick()`) is approximately 0.25 seconds.

The values used by the time support functions will wrap after an interval. The length of the interval depends on the length of the tick and the number of bits in the value. However, there is no issue when comparing time deltas of less than half this interval with a subtraction, if all data types are the same.

See [system-timer.h](#) for source code.

6.40.2 Macro Definition Documentation

6.40.2.1 `#define halIdleForMilliseconds(duration)`

Definition at line 193 of file [system-timer.h](#).

6.40.3 Function Documentation

6.40.3.1 `uint16_t halInternalStartSystemTimer (void)`

Initializes the system tick.

Returns

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

6.40.3.2 uint16_t halCommonGetInt16uMillisecondTick (void)

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

Returns

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

Referenced by [USBD_RemoteWakeup\(\)](#).

6.40.3.3 uint32_t halCommonGetInt32uMillisecondTick (void)

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

EmberStack Usage:

Unused, implementation optional.

Returns

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

6.40.3.4 uint16_t halCommonGetInt16uQuarterSecondTick (void)

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

EmberStack Usage:

Unused, implementation optional.

Returns

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

6.40.3.5 EmberStatus halSleepForQuarterSeconds (uint32_t * duration)

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

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

Note

This routine always enables interrupts.

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

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

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

EmberStack Usage:

Unused, implementation optional.

Parameters

<i>duration</i>	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.

6.40.3.6 EmberStatus halSleepForMilliseconds (*uint32_t * duration*)

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

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

Note

This routine always enables interrupts.

This function is not implemented on AVR-based platforms.

Sleep durations less than 3 milliseconds are not allowed on on EM2XX-based platforms. Any attempt to sleep for less than 3 milliseconds on EM2XX-based platforms

will cause the function to immediately exit without sleeping and return [EMBER_SLEEP_INTERRUPTED](#).

The maximum sleep time of the hardware is limited on EM2XX-based platforms to 32 seconds. Any sleep duration greater than this limit will wake up briefly (e.g. 16 microseconds) to reenable another sleep cycle. Due to this limitation, this function should not be used with durations within 3 milliseconds of a multiple 32 seconds. The short sleep cycle that results from such durations is not handled reliably by the system timer on EM2XX-based platforms. If a sleep duration within 3 milliseconds of a multiple of 32 seconds is desired, `halSleepForQuarterSeconds` should be used.

EmberStack Usage:

Unused, implementation optional.

Parameters

<i>duration</i>	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.

6.40.3.7 EmberStatus halCommonIdleForMilliseconds (`uint32_t * duration`)

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

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

Note

This routine always enables interrupts.

EmberStack Usage:

Unused, implementation optional.

Parameters

<i>duration</i>	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.

6.41 Symbol Timer Control

Symbol Timer Functions

- void [halInternalStartSymbolTimer](#) (void)
- void [halInternalSfdCaptureIsr](#) (void)
- uint32_t [halStackGetInt32uSymbolTick](#) (void)

MAC Timer Support Functions

These functions are used for MAC layer timing.

Applications should not directly call these functions. They are used internally by the operation of the stack.

- void [halStackOrderInt16uSymbolDelayA](#) (uint16_t symbols)
- void [halStackOrderNextSymbolDelayA](#) (uint16_t symbols)
- void [halStackCancelSymbolDelayA](#) (void)
- void [halStackSymbolDelayAIsr](#) (void)

6.41.1 Detailed Description

See [symbol-timer.h](#) for source code.

6.41.2 Function Documentation

6.41.2.1 void [halInternalStartSymbolTimer](#) (void)

Initializes the symbol timer. This function is only implemented when a general purpose microcontroller timer peripheral is used to implement the symbol timer (e.g. AVR/EM2420) When a dedicated symbol timer peripheral exists (e.g. EM2xx, EM3xx) this initialization is performed directly by the PHY.

6.41.2.2 void [halInternalSfdCaptureIsr](#) (void)

Should be called by the radio HAL whenever an SFD (Start Frame Delimiter) is detected.

6.41.2.3 uint32_t [halStackGetInt32uSymbolTick](#) (void)

Returns the current symbol time in symbol ticks (16 microseconds).

Returns

The least significant 32 bits of the current symbol time in symbol ticks (16 microseconds).

6.41.2.4 void halStackOrderInt16uSymbolDelayA (uint16_t *symbols*)

Sets up a timer and calls an interrupt-context callback when it expires.

Used by the MAC to request an interrupt callback at a specified amount of time in the future.

Parameters

<i>symbols</i>	The delay, in symbol ticks (16 microseconds).
----------------	---

6.41.2.5 void halStackOrderNextSymbolDelayA (uint16_t *symbols*)

Sets up a timer and calls an interrupt-context callback when it expires.

Note

This is different from [halStackOrderInt16uSymbolDelayA\(\)](#) in that it sets up the interrupt a specific number of symbols after the last timer interrupt occurred, instead of from the time this function is called.

Used by the MAC to request periodic timer based interrupts.

Parameters

<i>symbols</i>	The delay, in symbol ticks (16 microseconds).
----------------	---

6.41.2.6 void halStackCancelSymbolDelayA (void)

Cancels the timer set up by [halStackOrderInt16uSymbolDelayA\(\)](#).

6.41.2.7 void halStackSymbolDelayAlsr (void)

This is the interrupt level callback into the stack that is called when the timers set by hal-StackOrder*SymbolDelayA expire.

6.42 HAL Configuration

Modules

- Sample Breakout Board Configuration
- IAR PLATFORM_HEADER Configuration
- Common PLATFORM_HEADER Configuration
- NVIC Configuration
- Reset Cause Type Definitions

6.42.1 Detailed Description

6.43 Sample Breakout Board Configuration

Macros

- `#define PWRUP_CFG_SC1_TXD`
- `#define PWRDN_OUT_SC1_nRTS`

Custom Baud Rate Definitions

Application Framework NCP Configuration Board Header

This board header (dev0680) is not supported in framework NCP applications. NCP applications must use either the dev0680spi or dev0680uart board headers when creating custom NCP applications through the framework.

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, BOARDLED0, BOA-`
- `RDLED1,`
- `BOARDLED2, BOARDLED3, BOARD_ACTIVITY_LED, BOARD_HEARTBE-`
- `AT_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

USB Power State

Define if the USB is self powered or bus powered since the configuration descriptor needs to report to the host the powered state.

Note

VBUS Monitoring is required for USB to function when the EM358 device is configured as self-powered.

- #define USB_SELPWRD_STATE

USB Remote Wakeup Enable

If the USB device needs to awake the host from suspend, then it needs to have remote wakeup enable.

Note

The host can deny remote wakeup, keeping the device in suspend.

If the device has remote wakeup enabled the configuration descriptor needs to report this fact to the host. Additionally, the USB core in the chip needs to be directly told. Set the define USB_REMOTEWKUPEN_STATE to 0 if remote wake is disabled or 1 if enabled.

- #define USB_REMOTEWKUPEN_STATE

USB Maximum Power Consumption

The USB device must report the maximum power it will draw from the bus. This is done via the bMaxPower parameter in the Configuration Descriptor reported to the host. The value used is in units of 2mA.

Self-powered devices are low power devices and must draw less than 100mA.

Systems that have components such as a FEM are likely to consume more than 100mA and are considered high power and therefore must be bus-powered.

- #define `USB_MAX_POWER`

USB Enumeration Control

The following are used to aid in the abstraction of which GPIO is used for controlling the pull-up resistor for enumeration.

The hardware setup connects the D+ signal to a GPIO via a 1.5kOhm pull-up resistor. Any GPIO can be used since it just needs to be a simple push-pull output configuration.

- #define `ENUMCTRL`
- #define `ENUMCTRL_SETCFG(cfg)`
- #define `ENUMCTRL_SET()`
- #define `ENUMCTRL_CLR()`

USB VBUS Monitoring Support

Note

VBUS Monitoring is required for USB to function when the EM358 device is configured as self-powered.

The following are used to aid in the abstraction of which GPIO and IRQ is used for VBUS Monitoring.

Remember that IRQA and IRQB are fixed to GPIO PB0 and PB6 respectively while IRQC and IRQD can be assigned to any GPIO. Since USB's D- and D+ data pins are fixed to PA0 and PA1 respectively, SC2 can't be used so it makes sense to allocate PA2 for enumeration control and PA3 for VBUS monitoring. Therefore, using PA3 for VBUS monitoring requires IRQC or IRQD.

The driver will only try to use VBUSMON functionality if `USB_SELFPWRD_STATE` is set to 1.

- #define `VBUSMON`
- #define `VBUSMON_IN`
- #define `VBUSMON_SETCFG()`
- #define `VBUSMON_SEL()`
- #define `VBUSMON_ISR`
- #define `VBUSMON_INTCFG`
- #define `VBUSMON_INT_EN_BIT`
- #define `VBUSMON_FLAG_BIT`
- #define `VBUSMON_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_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_DFL_RHO_FOR_RHO
- #define PWRUP_OUT_DFL_RHO_FOR_RHO
- #define PWRDN_CFG_DFL_RHO_FOR_RHO
- #define PWRDN_OUT_DFL_RHO_FOR_RHO
- #define PWRUP_CFG_DFL_RHO_FOR_DFL
- #define PWRUP_OUT_DFL_RHO_FOR_DFL
- #define PWRDN_CFG_DFL_RHO_FOR_DFL
- #define PWRDN_OUT_DFL_RHO_FOR_DFL
- #define PWRUP_CFG_DFL_RHO
- #define PWRUP_OUT_DFL_RHO
- #define PWRDN_CFG_DFL_RHO
- #define PWRDN_OUT_DFL_RHO
- #define halInternalInitRadioHoldOff()

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 [hal-Init\(\)](#) 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 deepsleeping. 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

USB Configuration Defines

Provide the proper set of pin configuration for when USB is not enumerated. Not enumerated primarily refers to the driver not being configured or deep sleep. The configuration used here is only for keeping the USB off the bus. The GPIO configuration used when active is controlled by the USB driver since the driver needs to control the enumeration process (which affects GPIO state.)

Note

: Using USB requires Serial port 3 to be defined and is only possible on EM3582/E-M3586/EM3588/EM359 chips.

- #define PWRUP_CFG_USBDM
- #define PWRUP_OUT_USBDM
- #define PWRUP_CFG_USBDP
- #define PWRUP_OUT_USBDP
- #define PWRUP_CFG_ENUMCTRL
- #define PWRUP_OUT_ENUMCTRL
- #define PWRUP_CFG_VBUSMON
- #define PWRUP_OUT_VBUSMON
- #define PWRDN_CFG_USBDM
- #define PWRDN_OUT_USBDM
- #define PWRDN_CFG_USBDP
- #define PWRDN_OUT_USBDP
- #define PWRDN_CFG_ENUMCTRL
- #define PWRDN_OUT_ENUMCTRL
- #define PWRDN_CFG_VBUSMON
- #define PWRDN_OUT_VBUSMON

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.

- uint16_t gpioCfgPowerUp [6]
- uint16_t gpioCfgPowerDown [6]
- uint8_t gpioOutPowerUp [3]
- uint8_t gpioOutPowerDown [3]
- GpioMaskType 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 SET_RESUME_GPIO_CFG_REGISTERS()
- #define SET_RESUME_GPIO_OUTPUT_DATA_REGISTERS()
- #define SET_SUSPEND_GPIO_CFG_REGISTERS()
- #define SET_SUSPEND_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

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, BOARDLED0, BOA-
 RDLED1,
 BOARDLED2, BOARDLED3, BOARD_ACTIVITY_LED, BOARD_HEARTBE-
 AT_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_DFL_RHO_FOR_RHO
- #define PWRUP_OUT_DFL_RHO_FOR_RHO
- #define PWRDN_CFG_DFL_RHO_FOR_RHO
- #define PWRDN_OUT_DFL_RHO_FOR_RHO
- #define PWRUP_CFG_DFL_RHO_FOR_DFL
- #define PWRUP_OUT_DFL_RHO_FOR_DFL
- #define PWRDN_CFG_DFL_RHO_FOR_DFL
- #define PWRDN_OUT_DFL_RHO_FOR_DFL
- #define PWRUP_CFG_DFL_RHO
- #define PWRUP_OUT_DFL_RHO
- #define PWRDN_CFG_DFL_RHO
- #define PWRDN_OUT_DFL_RHO
- #define halInternalInitRadioHoldOff()
- #define ADJUST_GPIO_CONFIG_DFL_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 [hal-Init\(\)](#) 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`

ENABLE_ALT_FUNCTION_TX_ACTIVE

This define does not equate to anything. It is used as a trigger to enable the REG_EN alternate function on PA7. Default is to not enable REG_EN functionality on PA7.

When ENABLE_ALT_FUNCTION_TX_ACTIVE is defined, [halInit\(\)](#) and [halPowerUp\(\)](#) will enable the TX_ACTIVE alternate functionality of PC5. [halPowerDown\(\)](#) will configure PC5 to be a low output. TX_ACTIVE can be used for external PA power management and RF switching logic. In transmit mode the Tx baseband drives TX_ACTIVE high. In receive mode the TX_ACTIVE signal is low. This define will override any settings for PC5.

- `#define ENABLE_ALT_FUNCTION_TX_ACTIVE`

EEPROM_USES_SHUTDOWN_CONTROL

This define does not equate to anything. It is used as a trigger to enable the nTX_ACTIVE alternate function on PC6. Default is to not enable nTX_ACTIVE functionality on PC6.

When EEPROM_USES_SHUTDOWN_CONTROL is defined, logic is enabled in the EEPROM driver which drives PB7 high upon EEPROM initialization. In Ember reference designs, PB7 acts as an EEPROM enable pin and therefore must be driven high in order to use the EEPROM. This option is intended to be enabled when running app-bootloader on designs based on current Ember reference designs.

- #define EEPROM_USES_SHUTDOWN_CONTROL

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 deepsleeping. 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.

- uint16_t gpioCfgPowerUp [6]
- uint16_t gpioCfgPowerDown [6]
- uint8_t gpioOutPowerUp [3]
- uint8_t gpioOutPowerDown [3]
- GpioMaskType gpioRadioPowerBoardMask
- #define FEM_CTX_BIT
- #define FEM_CRX_BIT
- #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

6.43.1 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 / TRACEDATA2
- 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) / TRACEDATA1
- PC1 - Power Amplifier antenna select control / TRACEDATA3
- PC2 - JTAG (JTDO) / SWO / TRACEDATA0
- PC3 - JTAG (JTDI) / TRACECLK
- PC4 - JTAG (JTMS) / SWDIO
- PC5 - LED (on BoB)
- PC6 - Button (IRQC pointed to PC6)
- PC7 - TEMP_EN

Note

The file [ref0657.h](#) is a special variant of [dev0680.h](#) intended only for use with the SiGe SE2432L reference design.

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 (also used for SiGe SE2432L CPS)
- 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 - TX_ACTIVE (FEM CTX)
- PC6 - Button (IRQC pointed to PC6)
- PC7 - TEMP_EN

6.43.2 Macro Definition Documentation

6.43.2.1 #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 [78](#) of file [dev0680.h](#).

6.43.2.2 #define BUTTON0

The actual GPIO BUTTON0 is connected to. This define should be used whenever referencing BUTTON0.

Definition at line [137](#) of file [dev0680.h](#).

6.43.2.3 #define BUTTON0_IN

The GPIO input register for BUTTON0.

Definition at line [141](#) of file [dev0680.h](#).

6.43.2.4 #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 [146](#) of file [dev0680.h](#).

6.43.2.5 #define BUTTON0_ISR

The interrupt service routine for BUTTON0.

Definition at line [150](#) of file [dev0680.h](#).

6.43.2.6 #define BUTTON0_INTCFG

The interrupt configuration register for BUTTON0.

Definition at line 154 of file [dev0680.h](#).

6.43.2.7 #define BUTTON0_INT_EN_BIT

The interrupt enable bit for BUTTON0.

Definition at line 158 of file [dev0680.h](#).

6.43.2.8 #define BUTTON0_FLAG_BIT

The interrupt flag bit for BUTTON0.

Definition at line 162 of file [dev0680.h](#).

6.43.2.9 #define BUTTON0_MISS_BIT

The missed interrupt bit for BUTTON0.

Definition at line 166 of file [dev0680.h](#).

6.43.2.10 #define BUTTON1

The actual GPIO BUTTON1 is connected to. This define should be used whenever referencing BUTTON1, such as controlling if pieces are compiled in. Remember there may be other things that might want to use IRQC.

Definition at line 174 of file [dev0680.h](#).

6.43.2.11 #define BUTTON1_IN

The GPIO input register for BUTTON1.

Definition at line 178 of file [dev0680.h](#).

6.43.2.12 #define BUTTON1_SEL()

Point the proper IRQ at the desired pin for BUTTON1. Remember there may be other things that might want to use IRQC.

Note

For this board, IRQC is pointed at PC6

Definition at line 184 of file [dev0680.h](#).

6.43.2.13 #define BUTTON1_ISR

The interrupt service routine for BUTTON1. Remember there may be other things that might want to use IRQC.

Definition at line 189 of file [dev0680.h](#).

6.43.2.14 #define BUTTON1_INTCFG

The interrupt configuration register for BUTTON1.

Definition at line 193 of file [dev0680.h](#).

6.43.2.15 #define BUTTON1_INT_EN_BIT

The interrupt enable bit for BUTTON1.

Definition at line 197 of file [dev0680.h](#).

6.43.2.16 #define BUTTON1_FLAG_BIT

The interrupt flag bit for BUTTON1.

Definition at line 201 of file [dev0680.h](#).

6.43.2.17 #define BUTTON1_MISS_BIT

The missed interrupt bit for BUTTON1.

Definition at line 205 of file [dev0680.h](#).

6.43.2.18 #define USB_SELPWRD_STATE

The USB power state.

Set the define USB_SELPWRD_STATE: 0 if the device is bus powered. 1 if the device self powered.

Definition at line 228 of file [dev0680.h](#).

6.43.2.19 #define USB_REMOTEWKUPEN_STATE

USB Remote Wakeup Enable.

Set the define USB_REMOTEWKUPEN_STATE: 0 remote wakeup is disabled. 1 remote wakeup is enabled.

Definition at line 253 of file [dev0680.h](#).

Referenced by [USBD_Init\(\)](#).

6.43.2.20 #define USB_MAX_POWER

USB Max Power parameter (bMaxPower) the driver will report to the host in the Configuration Descriptor.

Definition at line 273 of file [dev0680.h](#).

6.43.2.21 #define ENUMCTRL

The actual GPIO ENUMCTRL is connected to. The GPIO only needs to be a simple push-pull output or input.

Definition at line 291 of file [dev0680.h](#).

6.43.2.22 #define ENUMCTRL_SETCFG(cfg)

Set the GPIO's configuration to the provided state. The two states used are GPIOCFG_OUT when the device is enumerated and GPIOCFG_IN when the device is not enumerated.

Definition at line 297 of file [dev0680.h](#).

6.43.2.23 #define ENUMCTRL_SET()

When the GPIO used for enumeration is configured as push-pull, this macro makes it easy to set the output state high.

Definition at line 302 of file [dev0680.h](#).

6.43.2.24 #define ENUMCTRL_CLR()

When the GPIO used for enumeration is configured as push-pull, this macro makes it easy to clear the output state low.

Definition at line 307 of file [dev0680.h](#).

6.43.2.25 #define VBUSMON

The actual GPIO VBUSMON is connected to. Remember that other pieces might want to use PA3.

Leaving VBUSMON undefined will keep VBUS Monitoring functionality from being compiled in and not conflict with other pieces that might want to use the GPIO or IRQ that VBUS Monitoring needs.

Definition at line 337 of file [dev0680.h](#).

6.43.2.26 #define VBUSMON_IN

The GPIO input register for VBUSMON.

Definition at line 341 of file [dev0680.h](#).

6.43.2.27 #define VBUSMON_SETCFG()

The GPIO configuration needed for VBUSMON. The configuration needs to be a simple input that will monitor for edge transitions.

Definition at line 346 of file [dev0680.h](#).

6.43.2.28 #define VBUSMON_SEL()

Point the proper IRQ at the desired pin for VBUSMON. Remember that other pieces that might want to use IRQC.

Note

For this board, IRQC is pointed at PA3.

Definition at line 352 of file [dev0680.h](#).

6.43.2.29 #define VBUSMON_ISR

The interrupt service routine for VBUSMON. Remember that other pieces that might want to use IRQC.

Definition at line 357 of file [dev0680.h](#).

6.43.2.30 #define VBUSMON_INTCFG

The interrupt configuration register for VBUSMON.

Definition at line 361 of file [dev0680.h](#).

6.43.2.31 #define VBUSMON_INT_EN_BIT

The interrupt enable bit for VBUSMON.

Definition at line 365 of file [dev0680.h](#).

6.43.2.32 #define VBUSMON_FLAG_BIT

The interrupt flag bit for VBUSMON.

Definition at line 369 of file [dev0680.h](#).

6.43.2.33 #define VBUSMON_MISS_BIT

The missed interrupt bit for VBUSMON.

Definition at line 373 of file [dev0680.h](#).

6.43.2.34 #define RHO_ASSERTED

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. The GPIO signal level to assert Radio HoldOff (1=high, 0=low).

Definition at line 422 of file [dev0680.h](#).

6.43.2.35 #define RHO_CFG

The GPIO configuration register for Radio HoldOff.

Definition at line 426 of file [dev0680.h](#).

6.43.2.36 #define RHO_IN

The GPIO input register for Radio HoldOff.

Definition at line 430 of file [dev0680.h](#).

6.43.2.37 #define RHO_OUT

The GPIO output register for Radio HoldOff.

Definition at line 434 of file [dev0680.h](#).

6.43.2.38 #define RHO_SEL()

Point the proper IRQ at the desired pin for Radio HoldOff. Remember there may be other things that might want to use this IRQ.

Definition at line 439 of file [dev0680.h](#).

6.43.2.39 #define RHO_ISR

The interrupt service routine for Radio HoldOff. Remember there may be other things that might want to use this IRQ.

Definition at line 444 of file [dev0680.h](#).

6.43.2.40 #define RHO_INTCFG

The interrupt configuration register for Radio HoldOff.

Definition at line 448 of file [dev0680.h](#).

6.43.2.41 #define RHO_INT_EN_BIT

The interrupt enable bit for Radio HoldOff.

Definition at line 452 of file [dev0680.h](#).

6.43.2.42 #define RHO_FLAG_BIT

The interrupt flag bit for Radio HoldOff.

Definition at line [456](#) of file [dev0680.h](#).

6.43.2.43 #define RHO_MISS_BIT

The missed interrupt bit for Radio HoldOff.

Definition at line [460](#) of file [dev0680.h](#).

6.43.2.44 #define PWRUP_CFG_DFL_RHO_FOR_RHO

Configuration of GPIO for Radio HoldOff operation.

Definition at line [464](#) of file [dev0680.h](#).

6.43.2.45 #define PWRUP_OUT_DFL_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. The GPIO signal level to assert Radio HoldOff (1=high, 0=low).

Definition at line [465](#) of file [dev0680.h](#).

6.43.2.46 #define PWRDN_CFG_DFL_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. The GPIO signal level to assert Radio HoldOff (1=high, 0=low).

Definition at line [466](#) of file [dev0680.h](#).

6.43.2.47 #define PWRDN_OUT_DFL_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. The GPIO signal level to assert Radio HoldOff (1=high, 0=low).

Definition at line [467](#) of file [dev0680.h](#).

6.43.2.48 #define PWRUP_CFG_DFL_RHO_FOR_DFL

Configuration of GPIO for default behavior.

Definition at line [471](#) of file [dev0680.h](#).

6.43.2.49 #define PWRUP_OUT_DFL_RHO_FOR_DFL

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. The GPIO signal level to assert Radio HoldOff (1=high, 0=low).

Definition at line [472](#) of file [dev0680.h](#).

6.43.2.50 #define PWRDN_CFG_DFL_RHO_FOR_DFL

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. The GPIO signal level to assert Radio HoldOff (1=high, 0=low).

Definition at line [473](#) of file [dev0680.h](#).

6.43.2.51 #define PWRDN_OUT_DFL_RHO_FOR_DFL

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. The GPIO signal level to assert Radio HoldOff (1=high, 0=low).

Definition at line [474](#) of file [dev0680.h](#).

6.43.2.52 #define PWRUP_CFG_DFL_RHO

The following definitions are helpers for managing Radio HoldOff and should not be modified.

(defined(RADIO_HOLDOFF) && defined(RHO_GPIO))

Definition at line [488](#) of file [dev0680.h](#).

6.43.2.53 #define PWRUP_OUT_DFL_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. The GPIO signal level to assert Radio HoldOff (1=high, 0=low).

Definition at line [489](#) of file [dev0680.h](#).

6.43.2.54 #define PWRDN_CFG_DFL_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. The GPIO signal level to assert Radio HoldOff (1=high, 0=low).

Definition at line [490](#) of file [dev0680.h](#).

6.43.2.55 #define PWRDN_OUT_DFL_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. The GPIO signal level to assert Radio HoldOff (1=high, 0=low).

Definition at line [491](#) of file [dev0680.h](#).

6.43.2.56 #define hallInternalInitRadioHoldOff()

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. The GPIO signal level to assert Radio HoldOff (1=high, 0=low).

Definition at line [492](#) of file [dev0680.h](#).

6.43.2.57 #define TEMP_SENSOR_ADC_CHANNEL

The analog input channel to use for the temperature sensor.

Definition at line [565](#) of file [dev0680.h](#).

6.43.2.58 #define TEMP_SENSOR_SCALE_FACTOR

The scale factor to compensate for different input ranges.

Definition at line [569](#) of file [dev0680.h](#).

6.43.2.59 #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 [585](#) of file [dev0680.h](#).

6.43.2.60 #define OSC32K_STARTUP_DELAY_MS

Definition at line [623](#) of file [dev0680.h](#).

6.43.2.61 #define PWRUP_CFG_PTI_EN

Give the packet trace configuration a friendly name.

Definition at line [774](#) of file [dev0680.h](#).

6.43.2.62 #define PWRUP_OUT_PTI_EN

Give the packet trace configuration a friendly name.

Definition at line [775](#) of file [dev0680.h](#).

6.43.2.63 #define PWRDN_CFG_PTI_EN

Give the packet trace configuration a friendly name.

Definition at line [776](#) of file [dev0680.h](#).

6.43.2.64 #define PWRDN_OUT_PTI_EN

Give the packet trace configuration a friendly name.

Definition at line [777](#) of file [dev0680.h](#).

6.43.2.65 #define PWRUP_CFG_PTI_DATA

Give the packet trace configuration a friendly name.

Definition at line [778](#) of file [dev0680.h](#).

6.43.2.66 #define PWRUP_OUT_PTI_DATA

Give the packet trace configuration a friendly name.

Definition at line [779](#) of file [dev0680.h](#).

6.43.2.67 #define PWRDN_CFG_PTI_DATA

Give the packet trace configuration a friendly name.

Definition at line [780](#) of file [dev0680.h](#).

6.43.2.68 #define PWRDN_OUT_PTI_DATA

Give the packet trace configuration a friendly name.

Definition at line [781](#) of file [dev0680.h](#).

6.43.2.69 #define PWRUP_CFG_BUTTON1

Give GPIO PC6 configuration a friendly name.

Definition at line [838](#) of file [dev0680.h](#).

6.43.2.70 #define PWRUP_OUT_BUTTON1

Give GPIO PC6 configuration a friendly name.

Definition at line [839](#) of file [dev0680.h](#).

6.43.2.71 #define PWRDN_CFG_BUTTON1

Give GPIO PC6 configuration a friendly name.

Definition at line [840](#) of file [dev0680.h](#).

6.43.2.72 #define PWRDN_OUT_BUTTON1

Give GPIO PC6 configuration a friendly name.

Definition at line [841](#) of file [dev0680.h](#).

6.43.2.73 #define CFG_TEMPEN

Give GPIO PC7 configuration a friendly name.

ENABLE_OSC32K

Definition at line [851](#) of file [dev0680.h](#).

6.43.2.74 #define PWRUP_CFG_LED2

Give the TX_ACTIVE configuration a friendly name.

ENABLE_ALT_FUNCTION_TX_ACTIVE

Definition at line [872](#) of file [dev0680.h](#).

6.43.2.75 #define PWRUP_OUT_LED2

Give the TX_ACTIVE configuration a friendly name.

ENABLE_ALT_FUNCTION_TX_ACTIVE

Definition at line [873](#) of file [dev0680.h](#).

6.43.2.76 #define PWRDN_CFG_LED2

Give the TX_ACTIVE configuration a friendly name.

ENABLE_ALT_FUNCTION_TX_ACTIVE

Definition at line [874](#) of file [dev0680.h](#).

6.43.2.77 #define PWRDN_OUT_LED2

Give the TX_ACTIVE configuration a friendly name.

ENABLE_ALT_FUNCTION_TX_ACTIVE

Definition at line [875](#) of file [dev0680.h](#).

6.43.2.78 #define PWRUP_CFG_USBDM

Give the USB configuration a friendly name.

Definition at line [916](#) of file [dev0680.h](#).

6.43.2.79 #define PWRUP_OUT_USBDM

Give the USB configuration a friendly name.

Definition at line [917](#) of file [dev0680.h](#).

6.43.2.80 #define PWRUP_CFG_USBDP

Give the USB configuration a friendly name.

Definition at line [918](#) of file [dev0680.h](#).

6.43.2.81 #define PWRUP_OUT_USBDP

Give the USB configuration a friendly name.

Definition at line [919](#) of file [dev0680.h](#).

6.43.2.82 #define PWRUP_CFG_ENUMCTRL

Give the USB configuration a friendly name.

Definition at line [920](#) of file [dev0680.h](#).

6.43.2.83 #define PWRUP_OUT_ENUMCTRL

Give the USB configuration a friendly name.

Definition at line [921](#) of file [dev0680.h](#).

6.43.2.84 #define PWRUP_CFG_VBUSMON

Give the USB configuration a friendly name.

Definition at line [922](#) of file [dev0680.h](#).

6.43.2.85 #define PWRUP_OUT_VBUSMON

Give the USB configuration a friendly name.

Definition at line [923](#) of file [dev0680.h](#).

6.43.2.86 #define PWRDN_CFG_USBDM

Give the USB configuration a friendly name.

Definition at line [924](#) of file [dev0680.h](#).

6.43.2.87 #define PWRDN_OUT_USBDM

Give the USB configuration a friendly name.

Definition at line [925](#) of file [dev0680.h](#).

6.43.2.88 #define PWRDN_CFG_USBDP

Give the USB configuration a friendly name.

Definition at line [926](#) of file [dev0680.h](#).

6.43.2.89 #define PWRDN_OUT_USBDP

Give the USB configuration a friendly name.

Definition at line [927](#) of file [dev0680.h](#).

6.43.2.90 #define PWRDN_CFG_ENUMCTRL

Give the USB configuration a friendly name.

Definition at line [928](#) of file [dev0680.h](#).

6.43.2.91 #define PWRDN_OUT_ENUMCTRL

Give the USB configuration a friendly name.

Definition at line [929](#) of file [dev0680.h](#).

6.43.2.92 #define PWRDN_CFG_VBUSMON

Give the USB configuration a friendly name.

Definition at line [930](#) of file [dev0680.h](#).

6.43.2.93 #define PWRDN_OUT_VBUSMON

Give the USB configuration a friendly name.

Definition at line [931](#) of file [dev0680.h](#).

6.43.2.94 #define PWRUP_CFG_SC1_TXD

Give GPIO SC1 TXD and nRTS configurations friendly names.

SLEEPY_IP_MODEM_UART

Definition at line [942](#) of file [dev0680.h](#).

6.43.2.95 #define PWRDN_OUT_SC1_nRTS

Definition at line [943](#) of file [dev0680.h](#).

6.43.2.96 #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 [halStackRadioPowerDownBoard\(\)](#).

Definition at line [974](#) of file [dev0680.h](#).

6.43.2.97 #define DEFINE_POWERUP_GPIO_CFG_VARIABLES()

Initialize GPIO powerup configuration variables.

Definition at line [981](#) of file [dev0680.h](#).

6.43.2.98 #define DEFINE_POWERUP_GPIO_OUTPUT_DATA_VARIABLES()

Initialize GPIO powerup output variables.

Definition at line [1013](#) of file [dev0680.h](#).

6.43.2.99 #define DEFINE_POWERDOWN_GPIO_CFG_VARIABLES()

Initialize powerdown GPIO configuration variables.

Definition at line [1048](#) of file [dev0680.h](#).

6.43.2.100 #define DEFINE_POWERDOWN_GPIO_OUTPUT_DATA_VARIABLES()

Initialize powerdown GPIO output variables.

Definition at line [1082](#) of file [dev0680.h](#).

6.43.2.101 #define SET_POWERUP_GPIO_CFG_REGISTERS()

Set powerup GPIO configuration registers.

Definition at line 1121 of file [dev0680.h](#).

6.43.2.102 #define SET_POWERUP_GPIO_OUTPUT_DATA_REGISTERS()

Set powerup GPIO output registers.

Definition at line 1133 of file [dev0680.h](#).

6.43.2.103 #define SET_POWERDOWN_GPIO_CFG_REGISTERS()

Set powerdown GPIO configuration registers.

Definition at line 1142 of file [dev0680.h](#).

6.43.2.104 #define SET_POWERDOWN_GPIO_OUTPUT_DATA_REGISTERS()

Set powerdown GPIO output registers.

Definition at line 1154 of file [dev0680.h](#).

6.43.2.105 #define SET_RESUME_GPIO_CFG_REGISTERS()

Set resume GPIO configuration registers. Identical to SET_POWERUP.

Definition at line 1162 of file [dev0680.h](#).

6.43.2.106 #define SET_RESUME_GPIO_OUTPUT_DATA_REGISTERS()

Set resume GPIO output registers. Identical to SET_POWERUP.

Definition at line 1169 of file [dev0680.h](#).

6.43.2.107 #define SET_SUSPEND_GPIO_CFG_REGISTERS()

Set suspend GPIO configuration registers. SET_POWERDOWN minus USB regs.

Definition at line 1176 of file [dev0680.h](#).

6.43.2.108 #define SET_SUSPEND_GPIO_OUTPUT_DATA_REGISTERS()

Set suspend GPIO output registers. SET_POWERDOWN minus USB regs.

Definition at line 1188 of file [dev0680.h](#).

6.43.2.109 #define CONFIGURE_EXTERNAL_REGULATOR_ENABLE()

External regulator enable/disable macro.

Definition at line 1201 of file [dev0680.h](#).

6.43.2.110 #define WAKE_ON_PA0

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [1216](#) of file [dev0680.h](#).

6.43.2.111 #define WAKE_ON_PA1

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [1217](#) of file [dev0680.h](#).

6.43.2.112 #define WAKE_ON_PA2

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [1218](#) of file [dev0680.h](#).

6.43.2.113 #define WAKE_ON_PA3

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [1219](#) of file [dev0680.h](#).

6.43.2.114 #define WAKE_ON_PA4

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [1220](#) of file [dev0680.h](#).

6.43.2.115 #define WAKE_ON_PA5

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [1221](#) of file [dev0680.h](#).

6.43.2.116 #define WAKE_ON_PA6

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [1222](#) of file [dev0680.h](#).

6.43.2.117 #define WAKE_ON_PA7

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [1223](#) of file [dev0680.h](#).

6.43.2.118 #define WAKE_ON_PB0

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [1224](#) of file [dev0680.h](#).

6.43.2.119 #define WAKE_ON_PB1

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [1225](#) of file [dev0680.h](#).

6.43.2.120 #define WAKE_ON_PB2

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [1229](#) of file [dev0680.h](#).

6.43.2.121 #define WAKE_ON_PB3

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [1231](#) of file [dev0680.h](#).

6.43.2.122 #define WAKE_ON_PB4

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [1232](#) of file [dev0680.h](#).

6.43.2.123 #define WAKE_ON_PB5

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [1233](#) of file [dev0680.h](#).

6.43.2.124 #define WAKE_ON_PB6

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [1234](#) of file [dev0680.h](#).

6.43.2.125 #define WAKE_ON_PB7

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [1235](#) of file [dev0680.h](#).

6.43.2.126 #define WAKE_ON_PC0

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [1236](#) of file [dev0680.h](#).

6.43.2.127 #define WAKE_ON_PC1

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [1237](#) of file [dev0680.h](#).

6.43.2.128 #define WAKE_ON_PC2

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [1238](#) of file [dev0680.h](#).

6.43.2.129 #define WAKE_ON_PC3

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [1239](#) of file [dev0680.h](#).

6.43.2.130 #define WAKE_ON_PC4

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [1240](#) of file [dev0680.h](#).

6.43.2.131 #define WAKE_ON_PC5

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [1241](#) of file [dev0680.h](#).

6.43.2.132 #define WAKE_ON_PC6

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [1242](#) of file [dev0680.h](#).

6.43.2.133 #define WAKE_ON_PC7

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [1243](#) of file [dev0680.h](#).

6.43.2.134 #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 [ref0657.h](#).

6.43.2.135 #define BUTTON0

The actual GPIO BUTTON0 is connected to. This define should be used whenever referencing BUTTON0.

Definition at line [124](#) of file [ref0657.h](#).

6.43.2.136 #define BUTTON0_IN

The GPIO input register for BUTTON0.

Definition at line 128 of file [ref0657.h](#).

6.43.2.137 #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 [ref0657.h](#).

6.43.2.138 #define BUTTON0_ISR

The interrupt service routine for BUTTON0.

Definition at line 137 of file [ref0657.h](#).

6.43.2.139 #define BUTTON0_INTCFG

The interrupt configuration register for BUTTON0.

Definition at line 141 of file [ref0657.h](#).

6.43.2.140 #define BUTTON0_INT_EN_BIT

The interrupt enable bit for BUTTON0.

Definition at line 145 of file [ref0657.h](#).

6.43.2.141 #define BUTTON0_FLAG_BIT

The interrupt flag bit for BUTTON0.

Definition at line 149 of file [ref0657.h](#).

6.43.2.142 #define BUTTON0_MISS_BIT

The missed interrupt bit for BUTTON0.

Definition at line 153 of file [ref0657.h](#).

6.43.2.143 #define BUTTON1

The actual GPIO BUTTON1 is connected to. This define should be used whenever referencing BUTTON1.

Definition at line 159 of file [ref0657.h](#).

6.43.2.144 #define BUTTON1_IN

The GPIO input register for BUTTON1.

Definition at line [163](#) of file [ref0657.h](#).

6.43.2.145 #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 [ref0657.h](#).

6.43.2.146 #define BUTTON1_ISR

The interrupt service routine for BUTTON1.

Definition at line [172](#) of file [ref0657.h](#).

6.43.2.147 #define BUTTON1_INTCFG

The interrupt configuration register for BUTTON1.

Definition at line [176](#) of file [ref0657.h](#).

6.43.2.148 #define BUTTON1_INT_EN_BIT

The interrupt enable bit for BUTTON1.

Definition at line [180](#) of file [ref0657.h](#).

6.43.2.149 #define BUTTON1_FLAG_BIT

The interrupt flag bit for BUTTON1.

Definition at line [184](#) of file [ref0657.h](#).

6.43.2.150 #define BUTTON1_MISS_BIT

The missed interrupt bit for BUTTON1.

Definition at line [188](#) of file [ref0657.h](#).

6.43.2.151 #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 [233](#) of file [ref0657.h](#).

6.43.2.152 #define RHO_ASSERTED

The GPIO signal level to assert Radio HoldOff (1=high, 0=low).

Definition at line [237](#) of file [ref0657.h](#).

6.43.2.153 #define RHO_CFG

The GPIO configuration register for Radio HoldOff.

Definition at line [241](#) of file [ref0657.h](#).

6.43.2.154 #define RHO_IN

The GPIO input register for Radio HoldOff.

Definition at line [245](#) of file [ref0657.h](#).

6.43.2.155 #define RHO_OUT

The GPIO output register for Radio HoldOff.

Definition at line [249](#) of file [ref0657.h](#).

6.43.2.156 #define RHO_SEL()

Point the proper IRQ at the desired pin for Radio HoldOff. Remember there may be other things that might want to use this IRQ.

Definition at line [254](#) of file [ref0657.h](#).

6.43.2.157 #define RHO_ISR

The interrupt service routine for Radio HoldOff. Remember there may be other things that might want to use this IRQ.

Definition at line [259](#) of file [ref0657.h](#).

6.43.2.158 #define RHO_INTCFG

The interrupt configuration register for Radio HoldOff.

Definition at line [263](#) of file [ref0657.h](#).

6.43.2.159 #define RHO_INT_EN_BIT

The interrupt enable bit for Radio HoldOff.

Definition at line [267](#) of file [ref0657.h](#).

6.43.2.160 #define RHO_FLAG_BIT

The interrupt flag bit for Radio HoldOff.

Definition at line [271](#) of file [ref0657.h](#).

6.43.2.161 #define RHO_MISS_BIT

The missed interrupt bit for Radio HoldOff.

Definition at line [275](#) of file [ref0657.h](#).

6.43.2.162 #define PWRUP_CFG_DFL_RHO_FOR_RHO

Configuration of GPIO for Radio HoldOff operation.

Definition at line [279](#) of file [ref0657.h](#).

6.43.2.163 #define PWRUP_OUT_DFL_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 [ref0657.h](#).

6.43.2.164 #define PWRDN_CFG_DFL_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 [ref0657.h](#).

6.43.2.165 #define PWRDN_OUT_DFL_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 [ref0657.h](#).

6.43.2.166 #define PWRUP_CFG_DFL_RHO_FOR_DFL

Configuration of GPIO for default behavior.

Definition at line 286 of file [ref0657.h](#).

6.43.2.167 #define PWRUP_OUT_DFL_RHO_FOR_DFL

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 287 of file [ref0657.h](#).

6.43.2.168 #define PWRDN_CFG_DFL_RHO_FOR_DFL

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 [ref0657.h](#).

6.43.2.169 #define PWRDN_OUT_DFL_RHO_FOR_DFL

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 [ref0657.h](#).

6.43.2.170 #define PWRUP_CFG_DFL_RHO

The following definitions are helpers for managing Radio HoldOff and should not be modified.

(defined(RADIO_HOLDOFF) && defined(RHO_GPIO))

Definition at line 303 of file [ref0657.h](#).

6.43.2.171 #define PWRUP_OUT_DFL_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 304 of file [ref0657.h](#).

6.43.2.172 #define PWRDN_CFG_DFL_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 305 of file [ref0657.h](#).

6.43.2.173 #define PWRDN_OUT_DFL_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 306 of file [ref0657.h](#).

6.43.2.174 #define hallInternalInitRadioHoldOff()

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 [ref0657.h](#).

6.43.2.175 #define ADJUST_GPIO_CONFIG_DFL_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 312 of file [ref0657.h](#).

6.43.2.176 #define TEMP_SENSOR_ADC_CHANNEL

The analog input channel to use for the temperature sensor.

Definition at line 380 of file [ref0657.h](#).

6.43.2.177 #define TEMP_SENSOR_SCALE_FACTOR

The scale factor to compensate for different input ranges.

Definition at line 384 of file [ref0657.h](#).

6.43.2.178 #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 400 of file [ref0657.h](#).

6.43.2.179 #define OSC32K_STARTUP_DELAY_MS

Definition at line 438 of file [ref0657.h](#).

6.43.2.180 #define ENABLE_ALT_FUNCTION_TX_ACTIVE

This define does not equate to anything. It is used as a trigger to enable the TX_ACTIVE alternate function on PC5. It is enabled by default in this board file because the SiGe SE2432L requires the TX_ACTIVE alternate function to operate properly.

Definition at line 490 of file [ref0657.h](#).

6.43.2.181 #define EEPROM_USES_SHUTDOWN_CONTROL

This define does not equate to anything. It is used as a trigger to enable the logic that drives PB7 high in the EEPROM driver.

Definition at line 529 of file [ref0657.h](#).

6.43.2.182 #define PWRUP_CFG_PTI_EN

Give the packet trace configuration a friendly name.

Definition at line 559 of file [ref0657.h](#).

6.43.2.183 #define PWRUP_OUT_PTI_EN

Give the packet trace configuration a friendly name.

Definition at line 560 of file [ref0657.h](#).

6.43.2.184 #define PWRDN_CFG_PTI_EN

Give the packet trace configuration a friendly name.

Definition at line [561](#) of file [ref0657.h](#).

6.43.2.185 #define PWRDN_OUT_PTI_EN

Give the packet trace configuration a friendly name.

Definition at line [562](#) of file [ref0657.h](#).

6.43.2.186 #define PWRUP_CFG_PTI_DATA

Give the packet trace configuration a friendly name.

Definition at line [563](#) of file [ref0657.h](#).

6.43.2.187 #define PWRUP_OUT_PTI_DATA

Give the packet trace configuration a friendly name.

Definition at line [564](#) of file [ref0657.h](#).

6.43.2.188 #define PWRDN_CFG_PTI_DATA

Give the packet trace configuration a friendly name.

Definition at line [565](#) of file [ref0657.h](#).

6.43.2.189 #define PWRDN_OUT_PTI_DATA

Give the packet trace configuration a friendly name.

Definition at line [566](#) of file [ref0657.h](#).

6.43.2.190 #define PWRUP_CFG_BUTTON1

Give GPIO PC6 configuration a friendly name.

Definition at line [623](#) of file [ref0657.h](#).

6.43.2.191 #define PWRUP_OUT_BUTTON1

Give GPIO PC6 configuration a friendly name.

Definition at line [624](#) of file [ref0657.h](#).

6.43.2.192 #define PWRDN_CFG_BUTTON1

Give GPIO PC6 configuration a friendly name.

Definition at line [625](#) of file [ref0657.h](#).

6.43.2.193 #define PWRDN_OUT_BUTTON1

Give GPIO PC6 configuration a friendly name.

Definition at line [626](#) of file [ref0657.h](#).

6.43.2.194 #define CFG_TEMPEN

Give GPIO PC7 configuration a friendly name.

ENABLE_OSC32K

Definition at line [636](#) of file [ref0657.h](#).

6.43.2.195 #define PWRUP_CFG_LED2

Give the TX_ACTIVE configuration a friendly name.

Definition at line [652](#) of file [ref0657.h](#).

6.43.2.196 #define PWRUP_OUT_LED2

Give the TX_ACTIVE configuration a friendly name.

Definition at line [653](#) of file [ref0657.h](#).

6.43.2.197 #define PWRDN_CFG_LED2

Give the TX_ACTIVE configuration a friendly name.

Definition at line [654](#) of file [ref0657.h](#).

6.43.2.198 #define PWRDN_OUT_LED2

Give the TX_ACTIVE configuration a friendly name.

Definition at line [655](#) of file [ref0657.h](#).

6.43.2.199 #define FEM_CTX_BIT

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\(\)](#).

Definition at line [690](#) of file [ref0657.h](#).

6.43.2.200 #define FEM_CRX_BIT

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\(\)](#).

Definition at line [698](#) of file [ref0657.h](#).

6.43.2.201 #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 [halStackRadioPowerDownBoard\(\)](#).

Definition at line [707](#) of file [ref0657.h](#).

6.43.2.202 #define DEFINE_POWERUP_GPIO_CFG_VARIABLES()

Initialize GPIO powerup configuration variables.

Definition at line [718](#) of file [ref0657.h](#).

6.43.2.203 #define DEFINE_POWERUP_GPIO_OUTPUT_DATA_VARIABLES()

Initialize GPIO powerup output variables.

Definition at line [750](#) of file [ref0657.h](#).

6.43.2.204 #define DEFINE_POWERDOWN_GPIO_CFG_VARIABLES()

Initialize powerdown GPIO configuration variables.

Definition at line [787](#) of file [ref0657.h](#).

6.43.2.205 #define DEFINE_POWERDOWN_GPIO_OUTPUT_DATA_VARIABLES()

Initialize powerdown GPIO output variables.

Definition at line [820](#) of file [ref0657.h](#).

6.43.2.206 #define SET_POWERUP_GPIO_CFG_REGISTERS()

Set powerup GPIO configuration registers.

Definition at line [860](#) of file [ref0657.h](#).

6.43.2.207 #define SET_POWERUP_GPIO_OUTPUT_DATA_REGISTERS()

Set powerup GPIO output registers.

Definition at line [872](#) of file [ref0657.h](#).

6.43.2.208 #define SET_POWERDOWN_GPIO_CFG_REGISTERS()

Set powerdown GPIO configuration registers.

Definition at line [881](#) of file [ref0657.h](#).

6.43.2.209 #define SET_POWERDOWN_GPIO_OUTPUT_DATA_REGISTERS()

Set powerdown GPIO output registers.

Definition at line [893](#) of file [ref0657.h](#).

6.43.2.210 #define CONFIGURE_EXTERNAL_REGULATOR_ENABLE()

External regulator enable/disable macro.

Definition at line [906](#) of file [ref0657.h](#).

6.43.2.211 #define WAKE_ON_PA0

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [921](#) of file [ref0657.h](#).

6.43.2.212 #define WAKE_ON_PA1

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [922](#) of file [ref0657.h](#).

6.43.2.213 #define WAKE_ON_PA2

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [923](#) of file [ref0657.h](#).

6.43.2.214 #define WAKE_ON_PA3

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [924](#) of file [ref0657.h](#).

6.43.2.215 #define WAKE_ON_PA4

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [925](#) of file [ref0657.h](#).

6.43.2.216 #define WAKE_ON_PA5

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [926](#) of file [ref0657.h](#).

6.43.2.217 #define WAKE_ON_PA6

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [927](#) of file [ref0657.h](#).

6.43.2.218 #define WAKE_ON_PA7

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [928](#) of file [ref0657.h](#).

6.43.2.219 #define WAKE_ON_PB0

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [929](#) of file [ref0657.h](#).

6.43.2.220 #define WAKE_ON_PB1

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [930](#) of file [ref0657.h](#).

6.43.2.221 #define WAKE_ON_PB2

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [934](#) of file [ref0657.h](#).

6.43.2.222 #define WAKE_ON_PB3

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [936](#) of file [ref0657.h](#).

6.43.2.223 #define WAKE_ON_PB4

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [937](#) of file [ref0657.h](#).

6.43.2.224 #define WAKE_ON_PB5

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [938](#) of file [ref0657.h](#).

6.43.2.225 #define WAKE_ON_PB6

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [939](#) of file [ref0657.h](#).

6.43.2.226 #define WAKE_ON_PB7

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [940](#) of file [ref0657.h](#).

6.43.2.227 #define WAKE_ON_PC0

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [941](#) of file [ref0657.h](#).

6.43.2.228 #define WAKE_ON_PC1

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [942](#) of file [ref0657.h](#).

6.43.2.229 #define WAKE_ON_PC2

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [943](#) of file [ref0657.h](#).

6.43.2.230 #define WAKE_ON_PC3

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [944](#) of file [ref0657.h](#).

6.43.2.231 #define WAKE_ON_PC4

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [945](#) of file [ref0657.h](#).

6.43.2.232 #define WAKE_ON_PC5

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [946](#) of file [ref0657.h](#).

6.43.2.233 #define WAKE_ON_PC6

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [947](#) of file [ref0657.h](#).

6.43.2.234 #define WAKE_ON_PC7

true if this GPIO can wake the chip from deep sleep, false if not.

Definition at line [948](#) of file [ref0657.h](#).

6.43.3 Enumeration Type Documentation

6.43.3.1 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
BOARDLED0
BOARDLED1
BOARDLED2
BOARDLED3
BOARD_ACTIVITY_LED
BOARD_HEARTBEAT_LED
```

Definition at line 107 of file [dev0680.h](#).

6.43.3.2 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
BOARDLED0
BOARDLED1
BOARDLED2
BOARDLED3
BOARD_ACTIVITY_LED
BOARD_HEARTBEAT_LED
```

Definition at line 94 of file [ref0657.h](#).

6.43.4 Variable Documentation

6.43.4.1 `uint16_t 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\(\)](#).

6.43.4.2 `uint16_t 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\(\)](#).

6.43.4.3 `uint8_t 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 [halStackRadioPowerDownBoard\(\)](#).

6.43.4.4 `uint8_t 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 [halStackRadioPowerDownBoard\(\)](#).

6.43.4.5 `GpioMaskType 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 [halStackRadioPowerDownBoard\(\)](#).

6.43.4.6 `uint16_t 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\(\)](#).

6.43.4.7 `uint16_t 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\(\)](#).

6.43.4.8 `uint8_t 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 [halStackRadioPowerDownBoard\(\)](#).

6.43.4.9 `uint8_t 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 [halStackRadioPowerDownBoard\(\)](#).

6.43.4.10 `GpioMaskType 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 [halStackRadioPowerDownBoard\(\)](#).

6.44 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
- #define MAIN_FUNCTION_ARGUMENTS

Functions

- void `_executeBarrierInstructions` (void)

Master Variable Types

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

- typedef bool `boolean`
- typedef unsigned char `int8u`
- typedef signed char `int8s`
- typedef unsigned short `int16u`
- typedef signed short `int16s`
- typedef unsigned int `int32u`
- typedef signed int `int32s`
- typedef unsigned long long `int64u`
- typedef signed long long `int64s`
- typedef unsigned 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 `halResetWatchdog()`
- #define __attribute__(...)
- #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)
- #define STRINGIZE(X)
- #define ALIGNMENT(X)
- #define WEAK(__symbol)
- #define NO_INIT(__symbol)

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 __BAT_INIT__
- #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 __INTERNAL_STORAGE__
- #define __UNRETAINED_RAM__
- #define __NO_INIT_SEGMENT_BEGIN
- #define __DEBUG_CHANNEL_SEGMENT_BEGIN
- #define __INTVEC_SEGMENT_BEGIN
- #define __CSTACK_SEGMENT_BEGIN
- #define __RESETINFO_SEGMENT_BEGIN
- #define __DATA_INIT_SEGMENT_BEGIN
- #define __DATA_SEGMENT_BEGIN
- #define __BSS_SEGMENT_BEGIN
- #define __APP_RAM_SEGMENT_BEGIN
- #define __CONST_SEGMENT_BEGIN

- #define _TEXT_SEGMENT_BEGIN
- #define _TEXTRW_INIT_SEGMENT_BEGIN
- #define _TEXTRW_SEGMENT_BEGIN
- #define _AAT_SEGMENT_BEGIN
- #define _BAT_SEGMENT_BEGIN
- #define _BAT_INIT_SEGMENT_BEGIN
- #define _FAT_SEGMENT_BEGIN
- #define _RAT_SEGMENT_BEGIN
- #define _NVM_SEGMENT_BEGIN
- #define _SIMEE_SEGMENT_BEGIN
- #define _EMHEAP_SEGMENT_BEGIN
- #define _EMHEAP_OVERLAY_SEGMENT_BEGIN
- #define _GUARD_REGION_SEGMENT_BEGIN
- #define _DLIB_PERTHREAD_INIT_SEGMENT_BEGIN
- #define _DLIB_PERTHREAD_INITIALIZED_DATA_SEGMENT_BEGIN
- #define _DLIB_PERTHREAD_ZERO_DATA_SEGMENT_BEGIN
- #define _INTERNAL_STORAGE_SEGMENT_BEGIN
- #define _UNRETAINED_RAM_SEGMENT_BEGIN
- #define _NO_INIT_SEGMENT_END
- #define _DEBUG_CHANNEL_SEGMENT_END
- #define _INTVEC_SEGMENT_END
- #define _CSTACK_SEGMENT_END
- #define _RESETINFO_SEGMENT_END
- #define _DATA_INIT_SEGMENT_END
- #define _DATA_SEGMENT_END
- #define _BSS_SEGMENT_END
- #define _APP_RAM_SEGMENT_END
- #define _CONST_SEGMENT_END
- #define _TEXT_SEGMENT_END
- #define _TEXTRW_INIT_SEGMENT_END
- #define _TEXTRW_SEGMENT_END
- #define _AAT_SEGMENT_END
- #define _BAT_SEGMENT_END
- #define _BAT_INIT_SEGMENT_END
- #define _FAT_SEGMENT_END
- #define _RAT_SEGMENT_END
- #define _NVM_SEGMENT_END
- #define _SIMEE_SEGMENT_END
- #define _EMHEAP_SEGMENT_END
- #define _EMHEAP_OVERLAY_SEGMENT_END
- #define _GUARD_REGION_SEGMENT_END
- #define _DLIB_PERTHREAD_INIT_SEGMENT_END
- #define _DLIB_PERTHREAD_INITIALIZED_DATA_SEGMENT_END
- #define _DLIB_PERTHREAD_ZERO_DATA_SEGMENT_END
- #define _INTERNAL_STORAGE_SEGMENT_END
- #define _UNRETAINED_RAM_SEGMENT_END
- #define _NO_INIT_SEGMENT_SIZE
- #define _DEBUG_CHANNEL_SEGMENT_SIZE
- #define _INTVEC_SEGMENT_SIZE
- #define _CSTACK_SEGMENT_SIZE

- #define _RESETINFO_SEGMENT_SIZE
- #define _DATA_INIT_SEGMENT_SIZE
- #define _DATA_SEGMENT_SIZE
- #define _BSS_SEGMENT_SIZE
- #define _APP_RAM_SEGMENT_SIZE
- #define _CONST_SEGMENT_SIZE
- #define _TEXT_SEGMENT_SIZE
- #define _TEXTRW_INIT_SEGMENT_SIZE
- #define _TEXTRW_SEGMENT_SIZE
- #define _AAT_SEGMENT_SIZE
- #define _BAT_SEGMENT_SIZE
- #define _BAT_INIT_SEGMENT_SIZE
- #define _FAT_SEGMENT_SIZE
- #define _RAT_SEGMENT_SIZE
- #define _NVM_SEGMENT_SIZE
- #define _SIMEE_SEGMENT_SIZE
- #define _EMHEAP_SEGMENT_SIZE
- #define _EMHEAP_OVERLAY_SEGMENT_SIZE
- #define _GUARD_REGION_SEGMENT_SIZE
- #define _DLIB_PERTHREAD_INIT_SEGMENT_SIZE
- #define _DLIB_PERTHREAD_INITIALIZED_DATA_SEGMENT_SIZE
- #define _DLIB_PERTHREAD_ZERO_DATA_SEGMENT_SIZE
- #define _INTERNAL_STORAGE_SEGMENT_SIZE
- #define _UNRETAINED_RAM_SEGMENT_SIZE

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 trigger 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)

6.44.1 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 PLATFROM_HEADER to point to it. `iar.h` automatically includes `platform-common.h`.

See `iar.h` and `platform-common.h` for source code.

6.44.2 Macro Definition Documentation

6.44.2.1 #define HAL_HAS_INT64

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

Definition at line 113 of file `iar.h`.

6.44.2.2 #define _HAL_USE_COMMON_PGM_

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

Definition at line 118 of file `iar.h`.

6.44.2.3 #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 132 of file `iar.h`.

6.44.2.4 #define NTOHS(val)

Define intrinsics for NTOHL and NTOHS to save code space by making endian.c compile to nothing.

Definition at line 139 of file `iar.h`.

6.44.2.5 #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 140 of file `iar.h`.

6.44.2.6 #define NO_STIPPING

A friendlier name for the compiler's intrinsic for not stripping.

Definition at line 147 of file [iar.h](#).

6.44.2.7 #define EEPROM

A friendlier name for the compiler's intrinsic for eeprom reference.

Definition at line 154 of file [iar.h](#).

6.44.2.8 #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 163 of file [iar.h](#).

6.44.2.9 #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 180 of file [iar.h](#).

Referenced by [USBD_AbortTransfer\(\)](#), [USBD_EpIsBusy\(\)](#), [USBD_Init\(\)](#), [USBD_Read\(\)](#), [USBD_StallEp\(\)](#), [USBD_UnStallEp\(\)](#), and [USBD_Write\(\)](#).

6.44.2.10 #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 209 of file [iar.h](#).

6.44.2.11 #define __attribute__(...)

Define **attribute** to nothing since it isn't handled by IAR.

Definition at line 215 of file [iar.h](#).

6.44.2.12 #define UNUSED

Declare a variable as unused to avoid a warning. Has no effect in IAR builds.

Definition at line 222 of file [iar.h](#).

6.44.2.13 #define SIGNED_ENUM

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

Definition at line 227 of file [iar.h](#).

6.44.2.14 #define STACK_FILL_VALUE

Define the magic value that is interpreted by IAR C-SPY's Stack View.

Definition at line [233](#) of file [iar.h](#).

6.44.2.15 #define RAMFUNC

Define a generic RAM function identifier to a compiler specific one.

Definition at line [243](#) of file [iar.h](#).

6.44.2.16 #define NO_OPERATION()

Define a generic no operation identifier to a compiler specific one.

Definition at line [249](#) of file [iar.h](#).

6.44.2.17 #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 [255](#) of file [iar.h](#).

6.44.2.18 #define simulatedTimePasses()

Stub for code not running in simulation.

Definition at line [264](#) of file [iar.h](#).

6.44.2.19 #define simulatedTimePassesMs(x)

Stub for code not running in simulation.

Definition at line [268](#) of file [iar.h](#).

6.44.2.20 #define simulatedSerialTimePasses()

Stub for code not running in simulation.

Definition at line [272](#) of file [iar.h](#).

6.44.2.21 #define _HAL_USE_COMMON_DIVMOD_

Use the Divide and Modulus Operations from [platform-common.h](#).

Definition at line [278](#) of file [iar.h](#).

6.44.2.22 #define VAR_AT_SEGMENT(__variableDeclaration, __segmentName)

Provide a portable way to specify the segment where a variable lives.

Definition at line [285](#) of file [iar.h](#).

6.44.2.23 #define STRINGIZE(*X*)

Convinience macro for turning a token into a string.

Definition at line 291 of file [iar.h](#).

6.44.2.24 #define ALIGNMENT(*X*)

Provide a portable way to align data.

Definition at line 296 of file [iar.h](#).

6.44.2.25 #define WEAK(*_symbol*)

Provide a portable way to specify a symbol as weak.

Definition at line 302 of file [iar.h](#).

6.44.2.26 #define NO_INIT(*_symbol*)

Provide a portable way to specify a non initialized symbol.

Definition at line 308 of file [iar.h](#).

6.44.2.27 #define __NO_INIT__

Portable segment names.

Definition at line 321 of file [iar.h](#).

6.44.2.28 #define __DEBUG_CHANNEL__

Portable segment names.

Definition at line 322 of file [iar.h](#).

6.44.2.29 #define __INTVEC__

Portable segment names.

Definition at line 323 of file [iar.h](#).

6.44.2.30 #define __CSTACK__

Portable segment names.

Definition at line 324 of file [iar.h](#).

6.44.2.31 #define __RESETINFO__

Portable segment names.

Definition at line 325 of file [iar.h](#).

6.44.2.32 #define __DATA_INIT__

Portable segment names.

Definition at line [326](#) of file [iar.h](#).

6.44.2.33 #define __DATA__

Portable segment names.

Definition at line [327](#) of file [iar.h](#).

6.44.2.34 #define __BSS__

Portable segment names.

Definition at line [328](#) of file [iar.h](#).

6.44.2.35 #define __APP_RAM__

Portable segment names.

Definition at line [329](#) of file [iar.h](#).

6.44.2.36 #define __CONST__

Portable segment names.

Definition at line [330](#) of file [iar.h](#).

6.44.2.37 #define __TEXT__

Portable segment names.

Definition at line [331](#) of file [iar.h](#).

6.44.2.38 #define __TEXTRW_INIT__

Portable segment names.

Definition at line [332](#) of file [iar.h](#).

6.44.2.39 #define __TEXTRW__

Portable segment names.

Definition at line [333](#) of file [iar.h](#).

6.44.2.40 #define __AAT__

Portable segment names.

Definition at line [334](#) of file [iar.h](#).

6.44.2.41 #define __BAT__

Portable segment names.

Definition at line 335 of file [iar.h](#).

6.44.2.42 #define __BAT_INIT__

Portable segment names.

Definition at line 336 of file [iar.h](#).

6.44.2.43 #define __FAT__

Portable segment names.

Definition at line 337 of file [iar.h](#).

6.44.2.44 #define __RAT__

Portable segment names.

Definition at line 338 of file [iar.h](#).

6.44.2.45 #define __NVM__

Portable segment names.

Definition at line 339 of file [iar.h](#).

6.44.2.46 #define __SIMEE__

Portable segment names.

Definition at line 340 of file [iar.h](#).

6.44.2.47 #define __EMHEAP__

Portable segment names.

Definition at line 341 of file [iar.h](#).

6.44.2.48 #define __EMHEAP_OVERLAY__

Portable segment names.

Definition at line 342 of file [iar.h](#).

6.44.2.49 #define __GUARD_REGION__

Portable segment names.

Definition at line 343 of file [iar.h](#).

6.44.2.50 #define __DLIB_PERTHREAD_INIT__

Portable segment names.

Definition at line 344 of file [iar.h](#).

6.44.2.51 #define __DLIB_PERTHREAD_INITIALIZED_DATA__

Portable segment names.

Definition at line 345 of file [iar.h](#).

6.44.2.52 #define __DLIB_PERTHREAD_ZERO_DATA__

Portable segment names.

Definition at line 346 of file [iar.h](#).

6.44.2.53 #define __INTERNAL_STORAGE__

Portable segment names.

Definition at line 347 of file [iar.h](#).

6.44.2.54 #define __UNRETAINED_RAM__

Portable segment names.

Definition at line 348 of file [iar.h](#).

6.44.2.55 #define __NO_INIT_SEGMENT_BEGIN

Portable segment names.

Definition at line 387 of file [iar.h](#).

6.44.2.56 #define __DEBUG_CHANNEL_SEGMENT_BEGIN

Portable segment names.

Definition at line 388 of file [iar.h](#).

6.44.2.57 #define __INTVEC_SEGMENT_BEGIN

Portable segment names.

Definition at line 389 of file [iar.h](#).

6.44.2.58 #define __CSTACK_SEGMENT_BEGIN

Portable segment names.

Definition at line 390 of file [iar.h](#).

6.44.2.59 #define _RESETINFO_SEGMENT_BEGIN

Portable segment names.

Definition at line 391 of file [iar.h](#).

6.44.2.60 #define _DATA_INIT_SEGMENT_BEGIN

Portable segment names.

Definition at line 392 of file [iar.h](#).

6.44.2.61 #define _DATA_SEGMENT_BEGIN

Portable segment names.

Definition at line 393 of file [iar.h](#).

6.44.2.62 #define _BSS_SEGMENT_BEGIN

Portable segment names.

Definition at line 394 of file [iar.h](#).

6.44.2.63 #define _APP_RAM_SEGMENT_BEGIN

Portable segment names.

Definition at line 395 of file [iar.h](#).

6.44.2.64 #define _CONST_SEGMENT_BEGIN

Portable segment names.

Definition at line 396 of file [iar.h](#).

6.44.2.65 #define _TEXT_SEGMENT_BEGIN

Portable segment names.

Definition at line 397 of file [iar.h](#).

6.44.2.66 #define _TEXTRW_INIT_SEGMENT_BEGIN

Portable segment names.

Definition at line 398 of file [iar.h](#).

6.44.2.67 #define _TEXTRW_SEGMENT_BEGIN

Portable segment names.

Definition at line 399 of file [iar.h](#).

6.44.2.68 #define _AAT_SEGMENT_BEGIN

Portable segment names.

Definition at line 400 of file [iar.h](#).

6.44.2.69 #define _BAT_SEGMENT_BEGIN

Portable segment names.

Definition at line 401 of file [iar.h](#).

6.44.2.70 #define _BAT_INIT_SEGMENT_BEGIN

Portable segment names.

Definition at line 402 of file [iar.h](#).

6.44.2.71 #define _FAT_SEGMENT_BEGIN

Portable segment names.

Definition at line 403 of file [iar.h](#).

6.44.2.72 #define _RAT_SEGMENT_BEGIN

Portable segment names.

Definition at line 404 of file [iar.h](#).

6.44.2.73 #define _NVM_SEGMENT_BEGIN

Portable segment names.

Definition at line 405 of file [iar.h](#).

6.44.2.74 #define _SIMEE_SEGMENT_BEGIN

Portable segment names.

Definition at line 406 of file [iar.h](#).

6.44.2.75 #define _EMHEAP_SEGMENT_BEGIN

Portable segment names.

Definition at line 407 of file [iar.h](#).

6.44.2.76 #define _EMHEAP_OVERLAY_SEGMENT_BEGIN

Portable segment names.

Definition at line 408 of file [iar.h](#).

6.44.2.77 #define _GUARD_REGION_SEGMENT_BEGIN

Portable segment names.

Definition at line [409](#) of file [iar.h](#).

6.44.2.78 #define _DLIB_PERTHREAD_INIT_SEGMENT_BEGIN

Portable segment names.

Definition at line [410](#) of file [iar.h](#).

6.44.2.79 #define _DLIB_PERTHREAD_INITIALIZED_DATA_SEGMENT_BEGIN

Portable segment names.

Definition at line [411](#) of file [iar.h](#).

6.44.2.80 #define _DLIB_PERTHREAD_ZERO_DATA_SEGMENT_BEGIN

Portable segment names.

Definition at line [412](#) of file [iar.h](#).

6.44.2.81 #define _INTERNAL_STORAGE_SEGMENT_BEGIN

Portable segment names.

Definition at line [413](#) of file [iar.h](#).

6.44.2.82 #define _UNRETAINED_RAM_SEGMENT_BEGIN

Portable segment names.

Definition at line [414](#) of file [iar.h](#).

6.44.2.83 #define _NO_INIT_SEGMENT_END

Portable segment names.

Definition at line [416](#) of file [iar.h](#).

6.44.2.84 #define _DEBUG_CHANNEL_SEGMENT_END

Portable segment names.

Definition at line [417](#) of file [iar.h](#).

6.44.2.85 #define _INTVEC_SEGMENT_END

Portable segment names.

Definition at line [418](#) of file [iar.h](#).

6.44.2.86 #define _CSTACK_SEGMENT_END

Portable segment names.

Definition at line [419](#) of file [iar.h](#).

6.44.2.87 #define _RESETINFO_SEGMENT_END

Portable segment names.

Definition at line [420](#) of file [iar.h](#).

6.44.2.88 #define _DATA_INIT_SEGMENT_END

Portable segment names.

Definition at line [421](#) of file [iar.h](#).

6.44.2.89 #define _DATA_SEGMENT_END

Portable segment names.

Definition at line [422](#) of file [iar.h](#).

6.44.2.90 #define _BSS_SEGMENT_END

Portable segment names.

Definition at line [423](#) of file [iar.h](#).

6.44.2.91 #define _APP_RAM_SEGMENT_END

Portable segment names.

Definition at line [424](#) of file [iar.h](#).

6.44.2.92 #define _CONST_SEGMENT_END

Portable segment names.

Definition at line [425](#) of file [iar.h](#).

6.44.2.93 #define _TEXT_SEGMENT_END

Portable segment names.

Definition at line [426](#) of file [iar.h](#).

6.44.2.94 #define _TEXTRW_INIT_SEGMENT_END

Portable segment names.

Definition at line [427](#) of file [iar.h](#).

6.44.2.95 #define _TEXTRW_SEGMENT_END

Portable segment names.

Definition at line [428](#) of file [iar.h](#).

6.44.2.96 #define _AAT_SEGMENT_END

Portable segment names.

Definition at line [429](#) of file [iar.h](#).

6.44.2.97 #define _BAT_SEGMENT_END

Portable segment names.

Definition at line [430](#) of file [iar.h](#).

6.44.2.98 #define _BAT_INIT_SEGMENT_END

Portable segment names.

Definition at line [431](#) of file [iar.h](#).

6.44.2.99 #define _FAT_SEGMENT_END

Portable segment names.

Definition at line [432](#) of file [iar.h](#).

6.44.2.100 #define _RAT_SEGMENT_END

Portable segment names.

Definition at line [433](#) of file [iar.h](#).

6.44.2.101 #define _NVM_SEGMENT_END

Portable segment names.

Definition at line [434](#) of file [iar.h](#).

6.44.2.102 #define _SIMEE_SEGMENT_END

Portable segment names.

Definition at line [435](#) of file [iar.h](#).

6.44.2.103 #define _EMHEAP_SEGMENT_END

Portable segment names.

Definition at line [436](#) of file [iar.h](#).

6.44.2.104 #define _EMHEAP_OVERLAY_SEGMENT_END

Portable segment names.

Definition at line [437](#) of file [iar.h](#).

6.44.2.105 #define _GUARD_REGION_SEGMENT_END

Portable segment names.

Definition at line [438](#) of file [iar.h](#).

6.44.2.106 #define _DLIB_PERTHREAD_INIT_SEGMENT_END

Portable segment names.

Definition at line [439](#) of file [iar.h](#).

6.44.2.107 #define _DLIB_PERTHREAD_INITIALIZED_DATA_SEGMENT_END

Portable segment names.

Definition at line [440](#) of file [iar.h](#).

6.44.2.108 #define _DLIB_PERTHREAD_ZERO_DATA_SEGMENT_END

Portable segment names.

Definition at line [441](#) of file [iar.h](#).

6.44.2.109 #define _INTERNAL_STORAGE_SEGMENT_END

Portable segment names.

Definition at line [442](#) of file [iar.h](#).

6.44.2.110 #define _UNRETAINED_RAM_SEGMENT_END

Portable segment names.

Definition at line [443](#) of file [iar.h](#).

6.44.2.111 #define _NO_INIT_SEGMENT_SIZE

Portable segment names.

Definition at line [445](#) of file [iar.h](#).

6.44.2.112 #define _DEBUG_CHANNEL_SEGMENT_SIZE

Portable segment names.

Definition at line [446](#) of file [iar.h](#).

6.44.2.113 #define _INTVEC_SEGMENT_SIZE

Portable segment names.

Definition at line [447](#) of file [iar.h](#).

6.44.2.114 #define _CSTACK_SEGMENT_SIZE

Portable segment names.

Definition at line [448](#) of file [iar.h](#).

6.44.2.115 #define _RESETINFO_SEGMENT_SIZE

Portable segment names.

Definition at line [449](#) of file [iar.h](#).

6.44.2.116 #define _DATA_INIT_SEGMENT_SIZE

Portable segment names.

Definition at line [450](#) of file [iar.h](#).

6.44.2.117 #define _DATA_SEGMENT_SIZE

Portable segment names.

Definition at line [451](#) of file [iar.h](#).

6.44.2.118 #define _BSS_SEGMENT_SIZE

Portable segment names.

Definition at line [452](#) of file [iar.h](#).

6.44.2.119 #define _APP_RAM_SEGMENT_SIZE

Portable segment names.

Definition at line [453](#) of file [iar.h](#).

6.44.2.120 #define _CONST_SEGMENT_SIZE

Portable segment names.

Definition at line [454](#) of file [iar.h](#).

6.44.2.121 #define _TEXT_SEGMENT_SIZE

Portable segment names.

Definition at line [455](#) of file [iar.h](#).

6.44.2.122 #define _TEXTRW_INIT_SEGMENT_SIZE

Portable segment names.

Definition at line [456](#) of file [iar.h](#).

6.44.2.123 #define _TEXTRW_SEGMENT_SIZE

Portable segment names.

Definition at line [457](#) of file [iar.h](#).

6.44.2.124 #define _AAT_SEGMENT_SIZE

Portable segment names.

Definition at line [458](#) of file [iar.h](#).

6.44.2.125 #define _BAT_SEGMENT_SIZE

Portable segment names.

Definition at line [459](#) of file [iar.h](#).

6.44.2.126 #define _BAT_INIT_SEGMENT_SIZE

Portable segment names.

Definition at line [460](#) of file [iar.h](#).

6.44.2.127 #define _FAT_SEGMENT_SIZE

Portable segment names.

Definition at line [461](#) of file [iar.h](#).

6.44.2.128 #define _RAT_SEGMENT_SIZE

Portable segment names.

Definition at line [462](#) of file [iar.h](#).

6.44.2.129 #define _NVM_SEGMENT_SIZE

Portable segment names.

Definition at line [463](#) of file [iar.h](#).

6.44.2.130 #define _SIMEE_SEGMENT_SIZE

Portable segment names.

Definition at line [464](#) of file [iar.h](#).

6.44.2.131 #define _EMHEAP_SEGMENT_SIZE

Portable segment names.

Definition at line [465](#) of file [iar.h](#).

6.44.2.132 #define _EMHEAP_OVERLAY_SEGMENT_SIZE

Portable segment names.

Definition at line [466](#) of file [iar.h](#).

6.44.2.133 #define _GUARD_REGION_SEGMENT_SIZE

Portable segment names.

Definition at line [467](#) of file [iar.h](#).

6.44.2.134 #define _DLIB_PERTHREAD_INIT_SEGMENT_SIZE

Portable segment names.

Definition at line [468](#) of file [iar.h](#).

6.44.2.135 #define _DLIB_PERTHREAD_INITIALIZED_DATA_SEGMENT_SIZE

Portable segment names.

Definition at line [469](#) of file [iar.h](#).

6.44.2.136 #define _DLIB_PERTHREAD_ZERO_DATA_SEGMENT_SIZE

Portable segment names.

Definition at line [470](#) of file [iar.h](#).

6.44.2.137 #define _INTERNAL_STORAGE_SEGMENT_SIZE

Portable segment names.

Definition at line [471](#) of file [iar.h](#).

6.44.2.138 #define _UNRETAINED_RAM_SEGMENT_SIZE

Portable segment names.

Definition at line [472](#) of file [iar.h](#).

6.44.2.139 #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 [496](#) of file [iar.h](#).

6.44.2.140 #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 [497](#) of file [iar.h](#).

6.44.2.141 #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 [498](#) of file [iar.h](#).

6.44.2.142 #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 [499](#) of file [iar.h](#).

6.44.2.143 #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 [639](#) of file [iar.h](#).

Referenced by [USBD_AbortTransfer\(\)](#), [USBD_Init\(\)](#), [USBD_Read\(\)](#), and [USBD_Write\(\)](#).

6.44.2.144 #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 [656](#) of file [iar.h](#).

Referenced by [USBD_AbortTransfer\(\)](#), [USBD_Init\(\)](#), [USBD_Read\(\)](#), and [USBD_Write\(\)](#).

6.44.2.145 #define INTERRUPTS_ON()

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

Definition at line [670](#) of file [iar.h](#).

6.44.2.146 #define INTERRUPTS_OFF()

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

Definition at line [684](#) of file [iar.h](#).

6.44.2.147 #define INTERRUPTS_ARE_OFF()**Returns**

true if global interrupts are disabled.

Definition at line [696](#) of file [iar.h](#).

6.44.2.148 #define INTERRUPTS_WERE_ON()**Returns**

true if global interrupt flag was enabled when [DISABLE_INTERRUPTS\(\)](#) was called.

Definition at line [702](#) of file [iar.h](#).

6.44.2.149 #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 [708](#) of file [iar.h](#).

Referenced by [USBD_AbortAllTransfers\(\)](#), [USBD_Connect\(\)](#), [USBD_Disconnect\(\)](#), [USBD_StallEp\(\)](#), and [USBD_UnStallEp\(\)](#).

6.44.2.150 #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 [724](#) of file [iar.h](#).

6.44.2.151 #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 [746](#) of file [iar.h](#).

6.44.2.152 #define _HAL_USE_COMMON_MEMUTILS_

If the line below is uncommented we will use Ember memory APIs, otherwise, we will use the C Standard library (memset,memcpy,memmove) APIs.

Definition at line [760](#) of file [iar.h](#).

6.44.2.153 #define PLATCOMMONOKTOINCLUDE

Include [platform-common.h](#) last to pick up defaults and common definitions.

Definition at line [789](#) of file [iar.h](#).

6.44.2.154 #define MAIN_FUNCTION_PARAMETERS

The kind of arguments the main function takes.

Definition at line [796](#) of file [iar.h](#).

6.44.2.155 #define MAIN_FUNCTION_ARGUMENTS

Definition at line [797](#) of file [iar.h](#).

6.44.3 Typedef Documentation**6.44.3.1 typedef bool boolean**

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

Definition at line [98](#) of file [iar.h](#).

6.44.3.2 `typedef unsigned char int8u`

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

Definition at line [99](#) of file `iar.h`.

6.44.3.3 `typedef signed char int8s`

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

Definition at line [100](#) of file `iar.h`.

6.44.3.4 `typedef unsigned short int16u`

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

Definition at line [101](#) of file `iar.h`.

6.44.3.5 `typedef signed short int16s`

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

Definition at line [102](#) of file `iar.h`.

6.44.3.6 `typedef unsigned int int32u`

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

Definition at line [103](#) of file `iar.h`.

6.44.3.7 `typedef signed int int32s`

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

Definition at line [104](#) of file `iar.h`.

6.44.3.8 `typedef unsigned long long int64u`

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

Definition at line [105](#) of file `iar.h`.

6.44.3.9 `typedef signed long long int64s`

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

Definition at line [106](#) of file `iar.h`.

6.44.3.10 `typedef unsigned int PointerType`

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

Definition at line [107](#) of file `iar.h`.

6.44.4 Function Documentation

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

A prototype definition for use by the assert macro. (see [hal/micro/micro.h](#))

6.44.4.2 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.

6.44.4.3 void _executeBarrierInstructions (void)

6.44.4.4 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>I</i>	An integer.
----------	-------------

Returns

A nonnegative integer.

6.45 Common PLATFORM_HEADER Configuration

Macros

- #define **MEMSET**(d, v, l)
- #define **MEMCOPY**(d, s, l)
- #define **MEMMOVE**(d, s, l)
- #define **MEMPGMCOPY**(d, s, l)
- #define **MEMCOMPARE**(s0, s1, l)
- #define **MEMPGMCOMPARE**(s0, s1, l)

Generic Types

- #define **TRUE**
- #define **FALSE**
- #define **NULL**

Bit Manipulation Macros

- #define **BIT**(x)
- #define **BIT32**(x)
- #define **SETBIT**(reg, bit)
- #define **SETBITS**(reg, bits)
- #define **CLEARBIT**(reg, bit)
- #define **CLEARBITS**(reg, bits)
- #define **READBIT**(reg, bit)
- #define **READBITS**(reg, bits)

Byte Manipulation Macros

- #define **LOW_BYTEn**
- #define **HIGH_BYTEn**
- #define **HIGH_LOW_TO_INT**(high, low)
- #define **BYTE_0**(n)
- #define **BYTE_1**(n)
- #define **BYTE_2**(n)
- #define **BYTE_3**(n)
- #define **COUNTOF**(a)

Time Manipulation Macros

- #define **elapsedTimeInt8u**(oldTime, newTime)
- #define **elapsedTimeInt16u**(oldTime, newTime)
- #define **elapsedTimeInt32u**(oldTime, newTime)
- #define **MAX_INT8U_VALUE**
- #define **HALF_MAX_INT8U_VALUE**
- #define **timeGTorEqualInt8u**(t1, t2)
- #define **MAX_INT16U_VALUE**

- #define HALF_MAX_INT16U_VALUE
- #define timeGTorEqualInt16u(t1, t2)
- #define MAX_INT32U_VALUE
- #define HALF_MAX_INT32U_VALUE
- #define timeGTorEqualInt32u(t1, t2)

Miscellaneous Macros

- #define UNUSED_VAR(x)
- #define DEBUG_LEVEL

6.45.1 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.

6.45.2 Macro Definition Documentation

6.45.2.1 #define MEMSET(d, v, l)

Friendly convenience macro pointing to the C Stdlib functions.

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

Definition at line 184 of file [platform-common.h](#).

Referenced by [USBD_Init\(\)](#).

6.45.2.2 #define MEMCOPY(d, s, l)

Definition at line 185 of file [platform-common.h](#).

6.45.2.3 #define MEMMOVE(d, s, l)

Definition at line 186 of file [platform-common.h](#).

6.45.2.4 #define MEMPGMCOPY(d, s, l)

Definition at line 187 of file [platform-common.h](#).

6.45.2.5 #define MEMCOMPARE(s0, s1, l)

Definition at line 188 of file [platform-common.h](#).

6.45.2.6 #define MEMPGMCOMPARE(s0, s1, l)

Definition at line 189 of file [platform-common.h](#).

6.45.2.7 #define TRUE

An alias for one, used for clarity.

Definition at line 213 of file [platform-common.h](#).

6.45.2.8 #define FALSE

An alias for zero, used for clarity.

Definition at line 218 of file [platform-common.h](#).

6.45.2.9 #define NULL

The null pointer.

Definition at line 224 of file [platform-common.h](#).

Referenced by [USBD_AbortTransfer\(\)](#), [USBD_EpIsBusy\(\)](#), [USBD_Init\(\)](#), [USBD_Read\(\)](#), [USBD_StallEp\(\)](#), [USBD_UnStallEp\(\)](#), and [USBD_Write\(\)](#).

6.45.2.10 #define BIT(x)

Useful to reference a single bit of a byte.

Definition at line 238 of file [platform-common.h](#).

6.45.2.11 #define BIT32(x)

Useful to reference a single bit of an uint32_t type.

Definition at line 243 of file [platform-common.h](#).

6.45.2.12 #define SETBIT(reg, bit)

Sets `bit` in the `reg` register or byte.

Note

Assuming `reg` is an IO register, some platforms (such as the AVR) can implement this in a single atomic operation.

Definition at line 250 of file [platform-common.h](#).

6.45.2.13 #define SETBITS(reg, bits)

Sets the bits in the `reg` register or the byte as specified in the bitmask `bits`.

Note

This is never a single atomic operation.

Definition at line [257](#) of file [platform-common.h](#).

6.45.2.14 #define CLEARBIT(reg, bit)

Clears a bit in the `reg` register or byte.

Note

Assuming `reg` is an IO register, some platforms (such as the AVR) can implement this in a single atomic operation.

Definition at line [264](#) of file [platform-common.h](#).

6.45.2.15 #define CLEARBITS(reg, bits)

Clears the bits in the `reg` register or byte as specified in the bitmask `bits`.

Note

This is never a single atomic operation.

Definition at line [271](#) of file [platform-common.h](#).

6.45.2.16 #define READBIT(reg, bit)

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

Definition at line [276](#) of file [platform-common.h](#).

6.45.2.17 #define READBITS(reg, bits)

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

Definition at line [282](#) of file [platform-common.h](#).

6.45.2.18 #define LOW_BYTE(n)

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

Definition at line [296](#) of file [platform-common.h](#).

6.45.2.19 #define HIGH_BYTE(n)

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

Definition at line [301](#) of file [platform-common.h](#).

6.45.2.20 #define HIGH_LOW_TO_INT(*high, low*)

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

Definition at line 307 of file [platform-common.h](#).

6.45.2.21 #define BYTE_0(*n*)

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

Definition at line 315 of file [platform-common.h](#).

6.45.2.22 #define BYTE_1(*n*)

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

Definition at line 320 of file [platform-common.h](#).

6.45.2.23 #define BYTE_2(*n*)

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

Definition at line 325 of file [platform-common.h](#).

6.45.2.24 #define BYTE_3(*n*)

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

Definition at line 330 of file [platform-common.h](#).

6.45.2.25 #define COUNTOF(*a*)

Returns the number of entries in an array.

Definition at line 335 of file [platform-common.h](#).

6.45.2.26 #define elapsedTimelnt8u(*oldTime, newTime*)

Returns the elapsed time between two 8 bit values. Result may not be valid if the time samples differ by more than 127.

Definition at line 350 of file [platform-common.h](#).

6.45.2.27 #define elapsedTimelnt16u(*oldTime, newTime*)

Returns the elapsed time between two 16 bit values. Result may not be valid if the time samples differ by more than 32767.

Definition at line 357 of file [platform-common.h](#).

Referenced by [USBD_RemoteWakeup\(\)](#).

6.45.2.28 #define elapsedTimeInt32u(oldTime, newTime)

Returns the elapsed time between two 32 bit values. Result may not be valid if the time samples differ by more than 2147483647.

Definition at line 364 of file [platform-common.h](#).

6.45.2.29 #define MAX_INT8U_VALUE

Returns true if t1 is greater than t2. Can only account for 1 wrap around of the variable before it is wrong.

Definition at line 371 of file [platform-common.h](#).

6.45.2.30 #define HALF_MAX_INT8U_VALUE

Returns the elapsed time between two 8 bit values. Result may not be valid if the time samples differ by more than 127.

Definition at line 372 of file [platform-common.h](#).

6.45.2.31 #define timeGToEqualInt8u(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](#).

6.45.2.32 #define MAX_INT16U_VALUE

Returns true if t1 is greater than t2. Can only account for 1 wrap around of the variable before it is wrong.

Definition at line 380 of file [platform-common.h](#).

6.45.2.33 #define HALF_MAX_INT16U_VALUE

Returns the elapsed time between two 8 bit values. Result may not be valid if the time samples differ by more than 127.

Definition at line 381 of file [platform-common.h](#).

6.45.2.34 #define timeGToEqualInt16u(t1, t2)

Returns the elapsed time between two 8 bit values. Result may not be valid if the time samples differ by more than 127.

Definition at line 382 of file [platform-common.h](#).

6.45.2.35 #define MAX_INT32U_VALUE

Returns true if t1 is greater than t2. Can only account for 1 wrap around of the variable before it is wrong.

Definition at line [389](#) of file [platform-common.h](#).

6.45.2.36 #define HALF_MAX_INT32U_VALUE

Returns the elapsed time between two 8 bit values. Result may not be valid if the time samples differ by more than 127.

Definition at line [390](#) of file [platform-common.h](#).

6.45.2.37 #define timeGTorEqualInt32u(t1, t2)

Returns the elapsed time between two 8 bit values. Result may not be valid if the time samples differ by more than 127.

Definition at line [391](#) of file [platform-common.h](#).

6.45.2.38 #define UNUSED_VAR(x)

Description:

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

Definition at line [408](#) of file [platform-common.h](#).

6.45.2.39 #define DEBUG_LEVEL

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

Definition at line [422](#) of file [platform-common.h](#).

6.46 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 halScl1Isr(void);
- void halScl1Isr(void);
- void halAdcIsr(void);
- void halIrqAIsr(void);
- void halIrqBIsr(void);
- void halIrqCIsr(void);
- void halIrqDIsr(void);

Note

This file should **not** be modified.

6.47 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 numerical equivalents. In addition, each base and extended definition is given a corresponding 3 letter ASCII string to facilitate printing. The ASCII strings are best use with [halGetExtendedResetString](#).

For example:

```
RESET_BASE_DEF(INTERNAL,          0x03,      "EXT")
RESET_EXT_DEF(INTERNAL, UNKNOWN,   0x00,      "UNK")
RESET_EXT_DEF(INTERNAL, 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.

6.48 HAL Utilities

Modules

- Crash and Watchdog Diagnostics
- Cyclic Redundancy Code (CRC)
- Random Number Generation
- Network to Host Byte Order Conversion

6.48.1 Detailed Description

6.49 Crash and Watchdog Diagnostics

Macros

- `#define halResetWasCrash()`

Functions

- `uint32_t halGetMainStackBytesUsed (void)`
- `void halPrintCrashSummary (uint8_t port)`
- `void halPrintCrashDetails (uint8_t port)`
- `void halPrintCrashData (uint8_t port)`
- `const HalAssertInfoType * halGetAssertInfo (void)`

6.49.1 Detailed Description

Crash and watchdog diagnostic functions. See [diagnostic.h](#) for source code.

6.49.2 Macro Definition Documentation

6.49.2.1 `#define halResetWasCrash()`

Macro evaluating to true if the last reset was a crash, false otherwise.

Definition at line [359](#) of file [diagnostic.h](#).

6.49.3 Function Documentation

6.49.3.1 `uint32_t halGetMainStackBytesUsed (void)`

Returns the number of bytes used in the main stack.

Returns

The number of bytes used in the main stack.

6.49.3.2 `void halPrintCrashSummary (uint8_t port)`

Print a summary of crash details.

Parameters

<code>port</code>	Serial port number (0 or 1).
-------------------	------------------------------

6.49.3.3 `void halPrintCrashDetails (uint8_t port)`

Print the complete, decoded crash details.

Parameters

<i>port</i>	Serial port number (0 or 1).
-------------	------------------------------

6.49.3.4 void halPrintCrashData (uint8_t *port*)

Print the complete crash data.

Parameters

<i>port</i>	Serial port number (0 or 1).
-------------	------------------------------

6.49.3.5 const HalAssertInfoType* halGetAssertInfo (void)

If last reset was from an assert, return saved assert information.

Returns

Pointer to struct containing assert filename and line.

6.50 Cyclic Redundancy Code (CRC)

Macros

- #define INITIAL_CRC
- #define CRC32_START
- #define CRC32_END

Functions

- uint16_t halCommonCrc16 (uint8_t newByte, uint16_t prevResult)
- uint32_t halCommonCrc32 (uint8_t newByte, uint32_t prevResult)

6.50.1 Detailed Description

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

6.50.2 Macro Definition Documentation

6.50.2.1 #define INITIAL_CRC

Definition at line [49](#) of file [crc.h](#).

6.50.2.2 #define CRC32_START

Definition at line [50](#) of file [crc.h](#).

6.50.2.3 #define CRC32_END

Definition at line [51](#) of file [crc.h](#).

6.50.3 Function Documentation

6.50.3.1 uint16_t halCommonCrc16 (uint8_t newByte, uint16_t prevResult)

Calculates 16-bit cyclic redundancy code (CITT CRC 16).

Applies the standard CITT CRC 16 polynomial to a single byte. It should support being called first with an initial value, then repeatedly until all data is processed.

Parameters

<i>newByte</i>	The new byte to be run through CRC.
<i>prevResult</i>	The previous CRC result.

Returns

The new CRC result.

6.50.3.2 `uint32_t halCommonCrc32 (uint8_t newByte, uint32_t prevResult)`

Calculates 32-bit cyclic redundancy code.

Note

On some radios or micros, the CRC for error detection on packet data is calculated in hardware.

Applies a CRC32 polynomial to a single byte. It should support being called first with an initial value, then repeatedly until all data is processed.

Parameters

<i>newByte</i>	The new byte to be run through CRC.
<i>prevResult</i>	The previous CRC result.

Returns

The new CRC result.

6.51 Random Number Generation

Functions

- void [halStackSeedRandom](#) (uint32_t seed)
- uint16_t [halCommonGetRandom](#) (void)

6.51.1 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.

6.51.2 Function Documentation

6.51.2.1 void halStackSeedRandom (uint32_t seed)

Seeds the [halCommonGetRandom\(\)](#) pseudorandom number generator.

Called by the stack during initialization with a seed from the radio.

Parameters

<i>seed</i>	A seed for the pseudorandom number generator.
-------------	---

6.51.2.2 uint16_t 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.

6.52 Network to Host Byte Order Conversion

Macros

- `#define HTONL`
- `#define HTONS`

Functions

- `uint16_t NTOHS (uint16_t val)`
- `uint32_t NTOHL (uint32_t val)`
- `uint32_t SwapEndiannessInt32u (uint32_t val)`

6.52.1 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.

6.52.2 Macro Definition Documentation

6.52.2.1 `#define HTONL`

Definition at line [59](#) of file [endian.h](#).

6.52.2.2 `#define HTONS`

Definition at line [62](#) of file [endian.h](#).

6.52.3 Function Documentation

6.52.3.1 `uint16_t NTOHS (uint16_t val)`

Converts a short (16-bit) value from network to host byte order.

6.52.3.2 `uint32_t NTOHL (uint32_t val)`

Converts a long (32-bit) value from network to host byte order.

6.52.3.3 `uint32_t SwapEndiannessInt32u (uint32_t val)`

6.53 Bootloader Interfaces

Modules

- [Common](#)
- [Standalone](#)
- [Application](#)

6.53.1 Detailed Description

6.54 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_SERIAL_USB**
- #define **BL_EXT_TYPE_SERIAL_USB_OTA**
- #define **BL_EXT_TYPE_APP_UNKNOWN**
- #define **BL_EXT_TYPE_APP_SPI**
- #define **BL_EXT_TYPE_APP_I2C**
- #define **BL_EXT_TYPE_APP_LOCAL_STORAGE**
- #define **BOOTLOADER_INVALID_VERSION**

Functions

- **BI BaseType halBootloaderGetType (void)**
- **BI ExtendedType halBootloaderGetInstalledType (void)**
- **uint16_t halGetBootloaderVersion (void)**
- **void halGetExtendedBootloaderVersion (uint32_t *emberVersion, uint32_t *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 uint8_t BI BaseType**
- **typedef uint16_t BI ExtendedType**

6.54.1 Detailed Description

Common bootloader interface defines and functions. See [bootloader-interface.h](#) for source code.

6.54.2 Macro Definition Documentation

6.54.2.1 `#define BL_TYPE_NULL`

Numerical definition for a bootloader type.

Definition at line [27](#) of file [bootloader-interface.h](#).

6.54.2.2 `#define BL_TYPE_STANDALONE`

Numerical definition for a bootloader type.

Definition at line [28](#) of file [bootloader-interface.h](#).

6.54.2.3 `#define BL_TYPE_APPLICATION`

Numerical definition for a bootloader type.

Definition at line [29](#) of file [bootloader-interface.h](#).

6.54.2.4 `#define BL_TYPE_BOOTLOADER`

Numerical definition for a bootloader type.

Definition at line [30](#) of file [bootloader-interface.h](#).

6.54.2.5 `#define BL_TYPE_SMALL_BOOTLOADER`

Numerical definition for a bootloader type.

Definition at line [31](#) of file [bootloader-interface.h](#).

6.54.2.6 `#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](#).

6.54.2.7 `#define BOOTLOADER_MAKE_EXTENDED_TYPE(baseType, extendedSpecifier)`

Macro returning the extended type of a bootloader when given a base type and extended-Specifier.

Definition at line [64](#) of file [bootloader-interface.h](#).

6.54.2.8 #define BL_EXT_TYPE_NULL

Macro defining the extended NULL bootloader type.

Definition at line [69](#) of file [bootloader-interface.h](#).

6.54.2.9 #define BL_EXT_TYPE_STANDALONE_UNKNOWN

Macro defining the extended standalone unknown bootloader type.

Definition at line [73](#) of file [bootloader-interface.h](#).

6.54.2.10 #define BL_EXT_TYPE_SERIAL_UART

Macro defining the extended standalone UART bootloader type.

Definition at line [77](#) of file [bootloader-interface.h](#).

6.54.2.11 #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](#).

6.54.2.12 #define BL_EXT_TYPE_EZSP_SPI

Definition at line [85](#) of file [bootloader-interface.h](#).

6.54.2.13 #define BL_EXT_TYPE_EZSP_SPI_OTA

Definition at line [86](#) of file [bootloader-interface.h](#).

6.54.2.14 #define BL_EXT_TYPE_SERIAL_USB

Macro defining the extended standalone USB bootloader type.

Definition at line [90](#) of file [bootloader-interface.h](#).

6.54.2.15 #define BL_EXT_TYPE_SERIAL_USB_OTA

Macro defining the extended standalone OTA and USB bootloader type.

Definition at line [94](#) of file [bootloader-interface.h](#).

6.54.2.16 #define BL_EXT_TYPE_APP_UNKNOWN

Macro defining the extended application unknown bootloader type.

Definition at line [99](#) of file [bootloader-interface.h](#).

6.54.2.17 #define BL_EXT_TYPE_APP_SPI

Macro defining the extended application SPI bootloader type.

Definition at line 103 of file [bootloader-interface.h](#).

6.54.2.18 #define BL_EXT_TYPE_APP_I2C

Macro defining the extended application I2C bootloader type.

Definition at line 107 of file [bootloader-interface.h](#).

6.54.2.19 #define BL_EXT_TYPE_APP_LOCAL_STORAGE

Macro defining a type for the local storage app bootloader.

Definition at line 111 of file [bootloader-interface.h](#).

6.54.2.20 #define BOOTLOADER_INVALID_VERSION

Define an invalid bootloader version.

Definition at line 122 of file [bootloader-interface.h](#).

6.54.3 Typedef Documentation**6.54.3.1 typedef uint8_t BI BaseType**

Define the bootloader base type.

Definition at line 41 of file [bootloader-interface.h](#).

6.54.3.2 typedef uint16_t BI ExtendedType

Define the bootloader extended type.

Definition at line 44 of file [bootloader-interface.h](#).

6.54.4 Function Documentation**6.54.4.1 BI BaseType halBootloaderGetType (void)**

Returns the bootloader base type the application was built for.

Returns

[BL_TYPE_NULL](#), [BL_TYPE_STANDALONE](#), or [BL_TYPE_APPLICATION](#)

6.54.4.2 BI ExtendedType halBootloaderGetInstalledType (void)

Returns the extended bootloader type of the bootloader that is present on the chip.

6.54.4.3 `uint16_t 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

6.54.4.4 `void halGetExtendedBootloaderVersion (uint32_t * emberVersion, uint32_t * 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

<i>ember-Version</i>	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.
<i>customer-Version</i>	This will return the 32bit value specified in <code>CUSTOMER_BOOTLOADER_VERSION</code> at build time.

6.55 Standalone

Macros

- `#define NO_BOOTLOADER_MODE`
- `#define STANDALONE_BOOTLOADER_NORMAL_MODE`
- `#define STANDALONE_BOOTLOADER_RECOVERY_MODE`

Functions

- `uint16_t halGetStandaloneBootloaderVersion (void)`
- `EmberStatus halLaunchStandaloneBootloader (uint8_t mode)`

6.55.1 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.

6.55.2 Macro Definition Documentation

6.55.2.1 `#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`.

6.55.2.2 `#define STANDALONE_BOOTLOADER_NORMAL_MODE`

Define a numerical value for the normal bootloader mode.

Definition at line 37 of file `bootloader-interface-standalone.h`.

6.55.2.3 `#define STANDALONE_BOOTLOADER_RECOVERY_MODE`

Define a numerical value for the recovery bootloader mode.

Definition at line 41 of file `bootloader-interface-standalone.h`.

6.55.3 Function Documentation

6.55.3.1 `uint16_t 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.

6.55.3.2 EmberStatus halLaunchStandaloneBootloader (uint8_t 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

<i>mode</i>	<p>Controls the mode in which the standalone bootloader will run. See the bootloader Application Note for full details. Options are:</p> <ul style="list-style-type: none"> • 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. <p>Both modes also allow an image transfer to begin via serial xmodem.</p>
-------------	--

Returns

An [EmberStatus](#) error if the standalone bootloader is not present, or [EMBER_SUCCESS](#).

6.56 Application

Macros

- `#define BOOTLOADER_SEGMENT_SIZE_LOG2`
- `#define BOOTLOADER_SEGMENT_SIZE`
- `#define BL_IMAGE_IS_VALID_CONTINUE`

Functions

- `uint8_t halAppBootloaderInit (void)`
- `const HalEepromInformationType * halAppBootloaderInfo (void)`
- `void halAppBootloaderShutdown (void)`
- `void halAppBootloaderImageIsValidReset (void)`
- `uint16_t halAppBootloaderImageIsValid (void)`
- `EmberStatus halAppBootloaderInstallNewImage (void)`
- `uint8_t halAppBootloaderWriteRawStorage (uint32_t address, const uint8_t *data, uint16_t len)`
- `uint8_t halAppBootloaderReadRawStorage (uint32_t address, uint8_t *data, uint16_t len)`
- `uint8_t halAppBootloaderEraseRawStorage (uint32_t address, uint32_t len)`
- `bool halAppBootloaderStorageBusy (void)`
- `uint8_t halAppBootloaderReadDownloadSpace (uint16_t pageToBeRead, uint8_t *destRamBuffer)`
- `uint8_t halAppBootloaderWriteDownloadSpace (uint16_t pageToBeWritten, uint8_t *RamPtr)`
- `uint8_t halAppBootloaderGetImageData (uint32_t *timestamp, uint8_t *userData)`
- `uint16_t halAppBootloaderGetVersion (void)`
- `uint16_t halAppBootloaderGetRecoveryVersion (void)`

6.56.1 Detailed Description

Definition 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.

6.56.2 Macro Definition Documentation

6.56.2.1 `#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`.

6.56.2.2 #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](#).

6.56.2.3 #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](#).

6.56.3 Function Documentation

6.56.3.1 uint8_t 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](#)

6.56.3.2 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

6.56.3.3 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.

6.56.3.4 void halAppBootloaderImageIsValidReset (void)

Call this function once before checking for a valid image to reset the call flag.

6.56.3.5 uint16_t 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.

6.56.3.6 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!

6.56.3.7 uint8_t halAppBootloaderWriteRawStorage (uint32_t address, const uint8_t * data, uint16_t 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 [HalEepromInformationType](#) structure for more information. Note: Some storage devices require contents to be erased before new data can be written, and will return an [EEPROM_ERR_ERASE_REQUIRED](#) error if write is called on a location that is not already erased. Refer to the [HalEepromInformationType](#) structure to see if the attached storage device requires erasing.

Parameters

<i>address</i>	Address to start writing data
<i>data</i>	A pointer to the buffer of data to write.
<i>len</i>	Length of the data to write

Returns

[EEPROM_SUCCESS](#) or [EEPROM_ERR](#).

6.56.3.8 `uint8_t halAppBootloaderReadRawStorage (uint32_t address, uint8_t * data, uint16_t 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 [HalEepromInformationType](#) structure for more information.

Parameters

<code>address</code>	Address from which to start reading data
<code>data</code>	A pointer to a buffer where data should be read into
<code>len</code>	Length of the data to read

Returns

[EEPROM_SUCCESS](#) or [EEPROM_ERR](#).

6.56.3.9 `uint8_t halAppBootloaderEraseRawStorage (uint32_t address, uint32_t 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

<code>address</code>	Address to start erasing
<code>len</code>	Length of the region to be erased

Returns

[EEPROM_SUCCESS](#) or [EEPROM_ERR](#).

6.56.3.10 `bool 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.

6.56.3.11 uint8_t halAppBootloaderReadDownloadSpace (uint16_t pageToBeRead, uint8_t * destRamBuffer)

Converts pageToBeRead to an address and the calls storage read function. Note: This function is deprecated. It has been replaced by [halAppBootloaderReadRawStorage\(\)](#)

Parameters

<i>pageToBeRead</i>	pass in the page to be read. This will be converted to the appropriate address. Pages are EEPROM_PAGE_SIZE long.
<i>destRamBuffer</i>	a pointer to the buffer to write to.

Returns

[EEPROM_SUCCESS](#) or [EEPROM_ERR](#).

6.56.3.12 uint8_t halAppBootloaderWriteDownloadSpace (uint16_t pageToBeWritten, uint8_t * RamPtr)

Converts pageToBeWritten to an address and calls the storage write function. Note: This function is deprecated. It has been replaced by [halAppBootloaderWriteRawStorage\(\)](#)

Parameters

<i>pageToBeWritten</i>	pass in the page to be written. This will be converted to the appropriate address. Pages are EEPROM_PAGE_SIZE long.
<i>RamPtr</i>	a pointer to the data to be written.

Returns

[EEPROM_SUCCESS](#) or [EEPROM_ERR](#)

6.56.3.13 uint8_t halAppBootloaderGetImageData (uint32_t * timestamp, uint8_t * userData)

Read the application image data from storage.

Parameters

<i>timestamp</i>	write the image timestamp to this data pointer.
<i>userData</i>	write the user data field to this buffer.

Returns

[EEPROM_SUCCESS](#) or [EEPROM_ERR](#)

6.56.3.14 uint16_t halAppBootloaderGetVersion (void)

Returns the application bootloader version.

6.56.3.15 uint16_t halAppBootloaderGetRecoveryVersion(void)

Returns the recovery image version.

6.57 Custom Bootloader HAL

Modules

- [Common](#)
- [Standalone](#)
- [Application](#)

6.57.1 Detailed Description

6.58 Common

Modules

- [GPIO](#)
- [Serial](#)

6.58.1 Detailed Description

6.59 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)
- bool `bootloadForceActivation` (void)

State Indicator Macros

The bootloader indicates which state it is in by calling these macros. Map them to the `::hal-BootloadStateIndicator` 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()`

6.59.1 Detailed Description

EM35x bootloader GPIO definitions. See `bootloader-gpio.h` for source code.

6.59.2 Macro Definition Documentation

6.59.2.1 #define BL_STATE_UP()

Finished init sequence, ready for bootload.

Definition at line 28 of file `bootloader-gpio.h`.

6.59.2.2 #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`.

6.59.2.3 #define BL_STATE_POLLING_LOOP()

Standalone bootloader polling serial/radio interface.

Definition at line [38](#) of file [bootloader-gpio.h](#).

6.59.2.4 #define BL_STATE_DOWNLOAD_LOOP()

Processing download image.

Definition at line [42](#) of file [bootloader-gpio.h](#).

6.59.2.5 #define BL_STATE_DOWNLOAD_SUCCESS()

Download process was a success.

Definition at line [46](#) of file [bootloader-gpio.h](#).

6.59.2.6 #define BL_STATE_DOWNLOAD_FAILURE()

Download process failed.

Definition at line [50](#) of file [bootloader-gpio.h](#).

6.59.3 Enumeration Type Documentation

6.59.3.1 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](#).

6.59.4 Function Documentation

6.59.4.1 void bootloadGpioInit (void)

Initialize GPIO.

6.59.4.2 void bootloadStateIndicator (enum blState_e state)

Helper function used for displaying bootloader state (for example: with LEDs).

6.59.4.3 **bool bootloadForceActivation(void)**

Force activation of bootloader.

6.60 Serial

Functions

- void `serInit` (void)
- void `serPutFlush` (void)
- void `serPutChar` (uint8_t ch)
- void `serPutStr` (const char *str)
- void `serPutBuf` (const uint8_t buf[], uint8_t size)
- void `serPutDecimal` (uint16_t val)
- void `serPutHex` (uint8_t byte)
- void `serPutHexInt` (uint16_t word)
- bool `serCharAvailable` (void)
- uint8_t `serGetChar` (uint8_t *ch)
- void `serGetFlush` (void)

6.60.1 Detailed Description

EM35x common bootloader serial definitions. See [bootloader-serial.h](#) for source code.

6.60.2 Function Documentation

6.60.2.1 void `serInit` (void)

Initialize serial port.

6.60.2.2 void `serPutFlush` (void)

Flush the transmitter.

6.60.2.3 void `serPutChar` (uint8_t ch)

Transmit a character.

Parameters

<code>ch</code>	A character.
-----------------	--------------

6.60.2.4 void `serPutStr` (const char * str)

Transmit a string.

Parameters

<code>str</code>	A string.
------------------	-----------

6.60.2.5 void serPutBuf (const uint8_t *buf*[], uint8_t *size*)

Transmit a buffer.

Parameters

<i>buf</i>	A buffer.
<i>size</i>	Length of buffer.

6.60.2.6 void serPutDecimal (uint16_t *val*)

Transmit a 16bit value in decimal.

Parameters

<i>val</i>	The data to print.
------------	--------------------

6.60.2.7 void serPutHex (uint8_t *byte*)

Transmit a byte as hex.

Parameters

<i>byte</i>	A byte.
-------------	---------

6.60.2.8 void serPutHexInt (uint16_t *word*)

Transmit a 16bit integer as hex.

Parameters

<i>word</i>	A 16bit integer.
-------------	------------------

6.60.2.9 bool serCharAvailable (void)

Determine if a character is available.

Returns

true if a character is available, false otherwise.

6.60.2.10 uint8_t serGetChar (uint8_t * *ch*)

Get a character if available, otherwise return an error.

Parameters

<i>ch</i>	Pointer to a location where the received byte will be placed.
-----------	---

Returns

::BL_SUCCESS if a character was obtained, ::BL_ERR otherwise.

6.60.2.11 void serGetFlush (void)

Flush the receiver.

6.61 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` (uint8_t commState)
- bool `checkDebugMenuOption` (uint8_t ch)
- BL_Status `initOtaState` (void)
- BL_Status `checkOtaStart` (void)
- BL_Status `receiveOtaImage` (void)
- bool `palsPresent` (void)
- bool `halCheckIntegrity` (void)

6.61.1 Detailed Description

EM35x standalone bootloader public definitions. See `standalone-bootloader.h` for source code.

6.61.2 Function Documentation

6.61.2.1 void `bootloaderMenu` (void)

This function must be implemented, providing a bootloader menu.

6.61.2.2 BL_Status `receiveImage` (uint8_t *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)

6.61.2.3 bool `checkDebugMenuOption` (uint8_t *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.

6.61.2.4 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.

6.61.2.5 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.

6.61.2.6 BL_Status receiveOtalmage(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.

6.61.2.7 bool 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.

6.61.2.8 `bool halCheckIntegrity(void)`

Validate application integrity by running AES-MMO hash and comparing to AAT.

Returns

false if fails integrity check, true if pass

6.62 Application

Data Structures

- struct `HalEepromInformationType`

This structure defines a variety of information about the attached external EEPROM device.

Macros

- `#define EEPROM_PAGE_SIZE`
- `#define EEPROM_FIRST_PAGE`
- `#define EEPROM_IMAGE_START`
- `#define EEPROM_SUCCESS`
- `#define EEPROM_ERR`
- `#define EEPROM_ERR_MASK`
- `#define EEPROM_ERR_PG_BOUNDARY`
- `#define EEPROM_ERR_PG_SZ`
- `#define EEPROM_ERR_WRT_DATA`
- `#define EEPROM_ERR_IMG_SZ`
- `#define EEPROM_ERR_ADDR`
- `#define EEPROM_ERR_INVALID_CHIP`
- `#define EEPROM_ERR_ERASE_REQUIRED`
- `#define EEPROM_ERR_NO_ERASE_SUPPORT`

EEPROM interaction functions.

- `uint8_t halEepromInit (void)`
- `void halEepromShutdown (void)`
- `const HalEepromInformationType * halEepromInfo (void)`
- `uint32_t halEepromSize (void)`
- `bool halEepromBusy (void)`
- `uint8_t halEepromRead (uint32_t address, uint8_t *data, uint16_t len)`
- `uint8_t halEepromWrite (uint32_t address, const uint8_t *data, uint16_t len)`
- `uint8_t halEepromErase (uint32_t address, uint32_t totalLength)`
- `#define EEPROM_INFO_VERSION`
- `#define EEPROM_INFO_MAJOR_VERSION`
- `#define EEPROM_INFO_MAJOR_VERSION_MASK`
- `#define EEPROM_INFO_MIN_VERSION_WITH_WORD_SIZE_SUPPORT`
- `#define EEPROM_CAPABILITIES_ERASE_SUPPORTED`
- `#define EEPROM_CAPABILITIES_PAGE_ERASE_REQD`
- `#define EEPROM_CAPABILITIES_BLOCKING_WRITE`
- `#define EEPROM_CAPABILITIES_BLOCKING_ERASE`

Required Custom Functions

- `void bootloaderAction (bool 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](#) (bool install)

6.62.1 Detailed Description

EM35x application bootloader and generic EEPROM Interface. The file [bootloader-eeprom.h](#) defines generic EEPROM parameters.

Changing EEPROM size will change the size of the application image space without changing the size or relative location of the recovery and reserved sections. See eeprom.c for more information on modifying EEPROM functionality.

See [bootloader-eeprom.h](#) for source code.

See [app-bootloader.h](#) for source code.

6.62.2 Macro Definition Documentation

6.62.2.1 #define EEPROM_PAGE_SIZE

Definition of an EEPROM page size, in bytes. This definition is deprecated, and should no longer be used.

Definition at line 24 of file [bootloader-eeprom.h](#).

6.62.2.2 #define EEPROM_FIRST_PAGE

Define the location of the first page in EEPROM. This definition is deprecated, and should no longer be used.

Definition at line 29 of file [bootloader-eeprom.h](#).

6.62.2.3 #define EEPROM_IMAGE_START

Define the location of the image start in EEPROM as a function of the [EEPROM_FIRST_PAGE](#) and [EEPROM_PAGE_SIZE](#). This definition is deprecated, and should no longer be used.

Definition at line 35 of file [bootloader-eeprom.h](#).

6.62.2.4 #define EEPROM_SUCCESS

Define EEPROM success status.

Definition at line 39 of file [bootloader-eeprom.h](#).

6.62.2.5 #define EEPROM_ERR

Define EEPROM error status.

Definition at line [43](#) of file `bootloader-eeprom.h`.

6.62.2.6 #define EEPROM_ERR_MASK

Define EEPROM error mask.

Definition at line [47](#) of file `bootloader-eeprom.h`.

6.62.2.7 #define EEPROM_ERR_PG_BOUNDARY

Define EEPROM page boundary error.

Definition at line [51](#) of file `bootloader-eeprom.h`.

6.62.2.8 #define EEPROM_ERR_PG_SZ

Define EEPROM page size error.

Definition at line [55](#) of file `bootloader-eeprom.h`.

6.62.2.9 #define EEPROM_ERR_WRT_DATA

Define EEPROM write data error.

Definition at line [59](#) of file `bootloader-eeprom.h`.

6.62.2.10 #define EEPROM_ERR_IMG_SZ

Define EEPROM image too large error.

Definition at line [63](#) of file `bootloader-eeprom.h`.

6.62.2.11 #define EEPROM_ERR_ADDR

Define EEPROM invalid address error.

Definition at line [67](#) of file `bootloader-eeprom.h`.

6.62.2.12 #define EEPROM_ERR_INVALID_CHIP

Define EEPROM chip initialization error.

Definition at line [71](#) of file `bootloader-eeprom.h`.

6.62.2.13 #define EEPROM_ERR_ERASE_REQUIRED

Define EEPROM erase required error.

Definition at line [75](#) of file `bootloader-eeprom.h`.

6.62.2.14 #define EEPROM_ERR_NO_ERASE_SUPPORT

Define EEPROM error for no erase support.

Definition at line [79](#) of file [bootloader-eeprom.h](#).

6.62.2.15 #define EEPROM_INFO_VERSION

The current version of the [HalEepromInformationType](#) data structure.

Definition at line [121](#) of file [bootloader-eeprom.h](#).

6.62.2.16 #define EEPROM_INFO_MAJOR_VERSION

The current version of the [HalEepromInformationType](#) data structure.

Definition at line [122](#) of file [bootloader-eeprom.h](#).

6.62.2.17 #define EEPROM_INFO_MAJOR_VERSION_MASK

The current version of the [HalEepromInformationType](#) data structure.

Definition at line [123](#) of file [bootloader-eeprom.h](#).

6.62.2.18 #define EEPROM_INFO_MIN_VERSION_WITH_WORD_SIZE_SUPPORT

The current version of the [HalEepromInformationType](#) data structure.

Definition at line [129](#) of file [bootloader-eeprom.h](#).

6.62.2.19 #define EEPROM_CAPABILITIES_ERASE_SUPPORTED

Eeprom capabilites mask that indicates the erase API is supported.

Definition at line [133](#) of file [bootloader-eeprom.h](#).

6.62.2.20 #define EEPROM_CAPABILITIES_PAGE_ERASE_REQD

Eeprom capabilites mask that indicates page erasing is required before new data can be written to a device.

Definition at line [138](#) of file [bootloader-eeprom.h](#).

6.62.2.21 #define EEPROM_CAPABILITIES_BLOCKING_WRITE

Eeprom capabilites mask that indicates that the write routine is blocking on this device.

Definition at line [143](#) of file [bootloader-eeprom.h](#).

6.62.2.22 #define EEPROM_CAPABILITIES_BLOCKING_ERASE

Eeprom capabilites mask that indicates that the erase routine is blocking on this device.

Definition at line 148 of file [bootloader-eeprom.h](#).

6.62.3 Function Documentation

6.62.3.1 `uint8_t halEepromInit (void)`

Initialize EEPROM. Note: some earlier drivers may assert instead of returning an error if initialization fails.

Returns

`EEPROM_SUCCESS` or `EEPROM_ERR_INVALID_CHIP`

6.62.3.2 `void halEepromShutdown (void)`

Shutdown the EEPROM to conserve power.

6.62.3.3 `const HalEepromInformationType* halEepromInfo (void)`

Call this function to get information about the external EEPROM and its capabilities.

The format of this call must not be altered. However, the content can be changed to work with a different device.

Returns

A pointer to a `HalEepromInformationType` data structure, or NULL if the driver does not support this API

6.62.3.4 `uint32_t halEepromSize (void)`

Return the size of the EEPROM.

The format of this call must not be altered. However, the content can be changed to work with a different device.
Internal use only. No exposure to application

Returns

`int32_t size`

6.62.3.5 `bool halEepromBusy (void)`

Determine if the external EEPROM is still busy performing the last operation, such as a write or an erase.

The format of this call must not be altered. However, the content can be changed to work with a different device.

Returns

true if still busy or false if not.

6.62.3.6 `uint8_t halEepromRead (uint32_t address, uint8_t * data, uint16_t len)`

Read from the external EEPROM.

This is the standard external EEPROM read function. The format of this call must not be altered. However, the content can be changed to work with a different device. Note: Not all storage implementations support accesses that are not page aligned, refer to the [HalEepromInformationType](#) structure for more information.

Parameters

<code>address</code>	The address to start reading from.
<code>data</code>	A pointer to where read data is stored.
<code>len</code>	The length of data to read.

Returns

[EEPROM_SUCCESS](#) or [EEPROM_ERR](#)

6.62.3.7 `uint8_t halEepromWrite (uint32_t address, const uint8_t * data, uint16_t len)`

Write to the external EEPROM.

This is the standard external EEPROM write function. The format of this call must not be altered. However, the content can be changed to work with a different device. Note: Not all storage implementations support accesses that are not page aligned, refer to the [HalEepromInformationType](#) structure for more information. Note: Some storage devices require contents to be erased before new data can be written, and will return an [EEPROM_ERR_ERASE_REQUIRED](#) error if write is called on a location that is not already erased. Refer to the [HalEepromInformationType](#) structure to see if the attached storage device requires erasing.

Parameters

<code>address</code>	The address to start writing to.
<code>data</code>	A pointer to the data to write.
<code>len</code>	The length of data to write.

Returns

[EEPROM_SUCCESS](#) or [EEPROM_ERR](#)

6.62.3.8 `uint8_t halEepromErase (uint32_t address, uint32_t totalLength)`

Erases the specified region of the external EEPROM.

The format of this call must not be altered. However, the content can be changed to work with a different device.

Note: Most devices require the specified region to be page aligned, and will return an error if an unaligned region is specified.

Note: Many devices take an extremely long time to perform an erase operation. When erasing a large region, it may be preferable to make multiple calls to this API so that other application

functionality can be performed while the erase is in progress. The ::halEepromBusy() API may be used to determine when the last erase operation has completed. Erase timing information can be found in the HalEepromInformationType structure.

Parameters

<i>address</i>	Address to start erasing
<i>len</i>	Length of the region to be erased

Returns

[EEPROM_SUCCESS](#) or [EEPROM_ERR](#).

6.62.3.9 void bootloaderAction (bool *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

<i>runRecovery</i>	If true, recover mode is activated. Otherwise, normal image installation is activated.
--------------------	--

6.62.3.10 BL_Status recoveryMode (void)

Activates [recoveryMode](#) to receive a new image over xmodem.

Returns

::BL_SUCCESS if an image was successfully received.

6.62.3.11 BL_Status processImage (bool *install*)

Processes an image in the external eeprom.

Parameters

<i>install</i>	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

6.63 Application Utilities API Reference

Modules

- [Forming and Joining Networks](#)
- [ZigBee Device Object \(ZDO\) Information](#)
- [Message Fragmentation](#)
- [Network Manager](#)
- [Serial Communication](#)
- [Command Interpreter](#)

6.63.1 Detailed Description

The Application Utilities API consists of sample utilities you can modify and use in your applications.

6.64 Forming and Joining Networks

Macros

- #define NETWORK_STORAGE_SIZE
- #define NETWORK_STORAGE_SIZE_SHIFT
- #define FORM_AND_JOIN_MAX_NETWORKS

Functions

- EmberStatus emberScanForUnusedPanId (uint32_t channelMask, uint8_t duration)
- EmberStatus emberScanForJoinableNetwork (uint32_t channelMask, uint8_t *extendedPanId)
- EmberStatus emberScanForNextJoinableNetwork (void)
- bool emberFormAndJoinIsScanning (void)
- bool emberFormAndJoinCanContinueJoinableNetworkScan (void)
- void emberUnusedPanIdFoundHandler (EmberPanId panId, uint8_t channel)
- void emberJoinableNetworkFoundHandler (EmberZigbeeNetwork *networkFound, uint8_t lqi, int8_t rssi)
- void emberScanErrorHandler (EmberStatus status)
- bool emberFormAndJoinScanCompleteHandler (uint8_t channel, EmberStatus status)
- bool emberFormAndJoinNetworkFoundHandler (EmberZigbeeNetwork *networkFound, uint8_t lqi, int8_t rssi)
- bool emberFormAndJoinEnergyScanResultHandler (uint8_t channel, int8_t maxRssiValue)
- void emberFormAndJoinTick (void)
- void emberFormAndJoinTaskInit (void)
- void emberFormAndJoinRunTask (void)
- void emberFormAndJoinCleanup (EmberStatus status)

Variables

- bool emberEnableDualChannelScan

6.64.1 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.

6.64.2 Macro Definition Documentation

6.64.2.1 #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 exceed 32 (`PACKET_BUFFER_SIZE`).

Definition at line [68](#) of file `form-and-join.h`.

6.64.2.2 #define NETWORK_STORAGE_SIZE_SHIFT

`Log_base2` of `NETWORK_STORAGE_SIZE`.

Definition at line [72](#) of file `form-and-join.h`.

6.64.2.3 #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](#).

6.64.3 Function Documentation

6.64.3.1 EmberStatus emberScanForUnusedPanId (*uint32_t channelMask, uint8_t duration*)

Find an unused PAN id.

Does an energy scan on the indicated channels and randomly chooses one from amongst those with the least average energy. Then picks a short PAN id that does not appear during an active scan on the chosen channel. The chosen PAN id and channel are returned via the [emberUnusedPanIdFoundHandler\(\)](#) callback. If an error occurs, the application is informed via the [emberScanErrorHandler\(\)](#).

Parameters

<i>channelMask</i>	
<i>duration</i>	The duration of the energy scan. See the documentation for emberStartScan() in stack/include/network-formation.h for information on duration values.

Returns

EMBER_LIBRARY_NOT_PRESENT if the form and join library is not available.

6.64.3.2 EmberStatus emberScanForJoinableNetwork (*uint32_t channelMask, uint8_t * extendedPanId*)

Finds a joinable network.

Performs an active scan on the specified channels looking for networks that:

1. currently permit joining,
2. match the stack profile of the application,
3. match the extended PAN id argument if it is not NULL.

Upon finding a matching network, the application is notified via the [emberJoinableNetworkFoundHandler\(\)](#) callback, and scanning stops. If an error occurs during the scanning process, the application is informed via the [emberScanErrorHandler\(\)](#), and scanning stops.

If the application determines that the discovered network is not the correct one, it may call [emberScanForNextJoinableNetwork\(\)](#) to continue the scanning process where it was left off and find a different joinable network. If the next network is not the correct one,

the application can continue to call [emberScanForNextJoinableNetwork\(\)](#). Each call must occur within 30 seconds of the previous one, otherwise the state of the scan process is deleted to free up memory. Calling [emberScanForJoinableNetwork\(\)](#) causes any old state to be forgotten and starts scanning from the beginning.

Parameters

<i>channelMask</i>	
<i>extendedPan- Id</i>	

Returns

`EMBER_LIBRARY_NOT_PRESENT` if the form and join library is not available.

6.64.3.3 EmberStatus `emberScanForNextJoinableNetwork(void)`

See [emberScanForJoinableNetwork\(\)](#).

6.64.3.4 bool `emberFormAndJoinIsScanning(void)`

Returns true if and only if the form and join library is in the process of scanning and is therefore expecting scan results to be passed to it from the application.

6.64.3.5 bool `emberFormAndJoinCanContinueJoinableNetworkScan(void)`

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

6.64.3.6 void `emberUnusedPanIdFoundHandler(EmberPanId panId, uint8_t channel)`

A callback the application needs to implement.

Notifies the application of the PAN id and channel found following a call to [emberScanForUnusedPanId\(\)](#).

Parameters

<i>panId</i>	
<i>channel</i>	

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

A callback the application needs to implement.

Notifies the application of the network found after a call to [emberScanForJoinableNetwork\(\)](#) or [emberScanForNextJoinableNetwork\(\)](#).

Parameters

<i>network-Found</i>	
<i>lqi</i>	The lqi value of the received beacon.
<i>rssi</i>	The rssi value of the received beacon.

6.64.3.8 void emberScanErrorHandler (EmberStatus *status*)

A callback the application needs to implement.

If an error occurs while scanning, this function is called and the scan effort is aborted.

Possible return status values are:

- EMBER_INVALID_CALL: if [emberScanForNextJoinableNetwork\(\)](#) is called more than 30 seconds after a previous call to [emberScanForJoinableNetwork\(\)](#) or [emberScanForNextJoinableNetwork\(\)](#).
- EMBER_NO_BUFFERS: if there is not enough memory to start a scan.
- EMBER_NO_BEACONS: if no joinable beacons are found.
- EMBER_MAC_SCANNING: if a scan is already in progress.

Parameters

<i>status</i>

6.64.3.9 bool emberFormAndJoinScanCompleteHandler (uint8_t *channel*, EmberStatus *status*)

The application must call this function from within its [emberScanCompleteHandler\(\)](#) (on the node) or [ezspScanCompleteHandler\(\)](#) (on an EZSP host). Default callback implementations are provided in the form-and-join-* callbacks.c files.

Returns

true iff the library made use of the call.

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

The application must call this function from within its [emberNetworkFoundHandler\(\)](#) (on the node) or [ezspNetworkFoundHandler\(\)](#) (on an EZSP host). Default callback implementations are provided in the form-and-join-* callbacks.c files.

Returns

true iff the library made use of the call.

6.64.3.11 `bool emberFormAndJoinEnergyScanResultHandler(uint8_t channel, int8_t maxRSSiValue)`

The application must call this function from within its [emberEnergyScanResultHandler\(\)](#) (on the node) or [ezspEnergyScanResultHandler\(\)](#) (on an EZSP host). Default callback implementations are provided in the form-and-join-*.callbacks.c files.

Returns

true iff the library made use of the call.

6.64.3.12 `void emberFormAndJoinTick(void)`

Used by the form and join code on the node to time out a joinable scan after 30 seconds of inactivity. The application must call [emberFormAndJoinTick\(\)](#) regularly. This function does not exist for the EZSP host library.

6.64.3.13 `void emberFormAndJoinTaskInit(void)`

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

6.64.3.14 `void emberFormAndJoinRunTask(void)`

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

6.64.3.15 `void emberFormAndJoinCleanup(EmberStatus status)`

When form-and-join state is no longer needed, the application can call this routine to cleanup and free resources. On the SOC platforms this will free the allocated message buffer.

6.64.4 Variable Documentation

6.64.4.1 `bool emberEnableDualChannelScan`

With some board layouts, the EM250 and EM260 are susceptible to a dual channel issue in which packets from 12 channels above or below can sometimes be heard faintly. This affects channels 11 - 14 and 23 - 26. Hardware reference designs EM250_REF_DES_LAT, version C0 and EM250_REF_DES_CER, version B0 solve the problem.

Setting the `emberEnableDualChannelScan` variable to true enables a software workaround to the dual channel issue which can be used with vulnerable boards. After [emberScanForJoinableNetwork\(\)](#) discovers a network on one of the susceptible channels, the channel number that differs by 12 is also scanned. If the same network can be heard there, the true channel is determined by comparing the link quality of the received beacons. The default value of `emberEnableDualChannelScan` is true for the EM250 and EM260. It is not used on other platforms.

6.65 ZigBee Device Object (ZDO) Information

Macros

- #define ZDO_MESSAGE_OVERHEAD

Service Discovery Functions

- EmberStatus emberMatchDescriptorsRequest (EmberNodeId target, uint16_t profile, EmberMessageBuffer inClusters, EmberMessageBuffer outClusters, EmberApsOption options)

Binding Manager Functions

- EmberStatus emberEndDeviceBindRequest (uint8_t endpoint, EmberApsOption options)

Function to Decode Address Response Messages

- EmberNodeId emberDecodeAddressResponse (EmberMessageBuffer response, EmberEUI64 eui64Return)

Service Discovery Functions

- EmberStatus emberNodeDescriptorRequest (EmberNodeId target, EmberApsOption options)
- EmberStatus emberPowerDescriptorRequest (EmberNodeId target, EmberApsOption options)
- EmberStatus emberSimpleDescriptorRequest (EmberNodeId target, uint8_t targetEndpoint, EmberApsOption options)
- EmberStatus emberActiveEndpointsRequest (EmberNodeId target, EmberApsOption options)

Binding Manager Functions

- EmberStatus emberBindRequest (EmberNodeId target, EmberEUI64 source, uint8_t sourceEndpoint, uint16_t clusterId, uint8_t type, EmberEUI64 destination, EmberMulticastId groupAddress, uint8_t destinationEndpoint, EmberApsOption options)
- EmberStatus emberUnbindRequest (EmberNodeId target, EmberEUI64 source, uint8_t sourceEndpoint, uint16_t clusterId, uint8_t type, EmberEUI64 destination, EmberMulticastId groupAddress, uint8_t destinationEndpoint, EmberApsOption options)

Node Manager Functions

- EmberStatus emberLqiTableRequest (EmberNodeId target, uint8_t startIndex, EmberApsOption options)
- EmberStatus emberRoutingTableRequest (EmberNodeId target, uint8_t startIndex, EmberApsOption options)

- `EmberStatus emberBindingTableRequest (EmberNodeId target, uint8_t startIndex, EmberApsOption options)`
- `EmberStatus emberLeaveRequest (EmberNodeId target, EmberEUI64 deviceAddress, uint8_t leaveRequestFlags, EmberApsOption options)`
- `EmberStatus emberPermitJoiningRequest (EmberNodeId target, uint8_t duration, uint8_t authentication, EmberApsOption options)`
- `void emberSetZigDevRequestRadius (uint8_t radius)`
- `uint8_t emberGetZigDevRequestRadius (void)`
- `uint8_t emberGetLastZigDevRequestSequence (void)`
- `uint8_t emberGetLastAppZigDevRequestSequence (void)`

Device Discovery Functions

- `EmberStatus emberNetworkAddressRequest (EmberEUI64 target, bool reportKids, uint8_t childstartIndex)`
- `EmberStatus emberIeeeAddressRequest (EmberNodeId target, bool reportKids, uint8_t childstartIndex, EmberApsOption options)`

Service Discovery Functions

- `EmberStatus ezspMatchDescriptorsRequest (EmberNodeId target, uint16_t profile, uint8_t inCount, uint8_t outCount, uint16_t *inClusters, uint16_t *outClusters, EmberApsOption options)`

Binding Manager Functions

- `EmberStatus ezspEndDeviceBindRequest (EmberNodeId localNodeId, EmberEUI64 localEui64, uint8_t endpoint, uint16_t profile, uint8_t inCount, uint8_t outCount, uint16_t *inClusters, uint16_t *outClusters, EmberApsOption options)`

Function to Decode Address Response Messages

- `EmberNodeId ezspDecodeAddressResponse (uint8_t *response, EmberEUI64 eui64Return)`

6.65.1 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);
    uint8_t targetEndpoint = emberUnsignedCommandArgument
        (state, 1);
    if (emberSimpleDescriptorRequest(target,
                                    targetEndpoint,
                                    EMBER_APS_OPTION_NONE)
        != EMBER_SUCCESS) {
        emberSerialPrintf(SERIAL_PORT, "
            emberSimpleDescriptorRequest failed\r\n");
    }
}
```

The following code shows an example of an application's use of [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);
    uint8_t targetEndpoint = emberUnsignedCommandArgument
        (state, 1);
    if (emberSimpleDescriptorRequest(target,
                                    targetEndpoint,
                                    EMBER_APS_OPTION_NONE)
        != EMBER_SUCCESS) {
        emberSerialPrintf(SERIAL_PORT, "
            emberSimpleDescriptorRequest failed\r\n");
    }
}
```

The following code shows an example of an application's use of [ezspDecodeAddressResponse\(\)](#).

```
void ezspIncomingMessageHandler(EmberIncomingMessageType
    type,
                                EmberApsFrame *apsFrame,
                                uint8_t lastHopLqi,
                                int8_t lastHopRssi,
                                EmberNodeId sender,
                                uint8_t bindingIndex,
                                uint8_t addressIndex,
                                uint8_t messageLength,
                                uint8_t *messageContents)
{
    if (apsFrame->profileId == EMBER_ZDO_PROFILE_ID)
    {
        switch (apsFrame->clusterId) {
        case NETWORK_ADDRESS_RESPONSE:
        case IEEE_ADDRESS_RESPONSE:
        {
            EmberEUI64 eui64;
            EmberNodeId nodeId = ezspDecodeAddressResponse
                (messageContents,
                 eui64);
            // Use nodeId and eui64 here.
            break;
        }
        default:
            // Handle other incoming ZDO responses here.
        }
    } else {
        // Handle incoming application messages here.
    }
}
```

6.65.2 Macro Definition Documentation

6.65.2.1 #define ZDO_MESSAGE_OVERHEAD

ZDO messages start with a sequence number.

Definition at line 16 of file [zigbee-device-common.h](#).

6.65.3 Function Documentation

6.65.3.1 EmberStatus emberMatchDescriptorsRequest (EmberNodeId *target*, uint16_t *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

<i>target</i>	The node whose matching endpoints are desired. The request can be sent unicast or broadcast ONLY to the "RX-on-when-idle-address" (0xFFFFD) If sent as a broadcast, any node that has matching endpoints will send a response.
<i>profile</i>	The application profile to match.
<i>inClusters</i>	The list of input clusters. To not match any input clusters, use EMBER_NULL_MESSAGE_BUFFER .
<i>outClusters</i>	The list of output clusters. To not match any output clusters, use EMBER_NULL_MESSAGE_BUFFER .
<i>options</i>	The options to use when sending the unicast request. See emberSendUnicast() for a description. This parameter is ignored if the target is a broadcast address.

Returns

An EmberStatus value. [EMBER_SUCCESS](#), [EMBER_NO_BUFFERS](#), [EMBER_NETWORK_DOWN](#) or [EMBER_NETWORK_BUSY](#).

6.65.3.2 EmberStatus emberEndDeviceBindRequest (uint8_t *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

<i>endpoint</i>	The endpoint on the local device whose simple descriptor will be used to create the request.
<i>options</i>	The options to use when sending the request. See emberSendUnicast() for a description.

Returns

An EmberStatus value. [EMBER_SUCCESS](#), [EMBER_NO_BUFFERS](#), [EMBER_NETWORK_DOWN](#) or [EMBER_NETWORK_BUSY](#).

6.65.3.3 EmberNodeId emberDecodeAddressResponse (EmberMessageBuffer *response*, EmberEUI64 *eui64Return*)

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

Parameters

<i>response</i>	The received ZDO message with cluster ID NETWORK_ADDRESS_RESPONSE or IEEE_ADDRESS_RESPONSE .
<i>eui64Return</i>	The EUI64 from the response is copied here.

Returns

Returns the node ID from the response if the response status was [EMBER_ZDP_SUCCESS](#). Otherwise, returns [EMBER_NULL_NODE_ID](#).

6.65.3.4 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

<i>target</i>	The node whose node descriptor is desired.
<i>options</i>	The options to use when sending the request. See emberSendUnicast() for a description.

Returns

An [EmberStatus](#) value. [EMBER_SUCCESS](#), [EMBER_NO_BUFFERS](#), [EMBER_NETWORK_DOWN](#) or [EMBER_NETWORK_BUSY](#).

6.65.3.5 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

<i>target</i>	The node whose power descriptor is desired.
<i>options</i>	The options to use when sending the request. See emberSendUnicast() for a description.

Returns

An [EmberStatus](#) value. [EMBER_SUCCESS](#), [EMBER_NO_BUFFERS](#), [EMBER_NETWORK_DOWN](#) or [EMBER_NETWORK_BUSY](#).

6.65.3.6 EmberStatus emberSimpleDescriptorRequest (EmberNodeId *target*, uint8_t *targetEndpoint*, EmberApsOption *options*)

Request the specified node to send the simple descriptor for the specified endpoint. The simple descriptor contains information specific to a single endpoint. It describes the application profile identifier, application device identifier, application device version, application flags, application input clusters and application output clusters. It is defined in the ZigBee Application Framework Specification.

Parameters

<i>target</i>	The node of interest.
<i>target-Endpoint</i>	The endpoint on the target node whose simple descriptor is desired.
<i>options</i>	The options to use when sending the request. See emberSendUnicast() for a description.

Returns

An EmberStatus value. [EMBER_SUCCESS](#), [EMBER_NO_BUFFERS](#), [EMBER_NETWORK_DOWN](#) or [EMBER_NETWORK_BUSY](#).

6.65.3.7 EmberStatus emberActiveEndpointsRequest (EmberNodeId *target*, EmberApsOption *options*)

Request the specified node to send a list of its active endpoints. An active endpoint is one for which a simple descriptor is available.

Parameters

<i>target</i>	The node whose active endpoints are desired.
<i>options</i>	The options to use when sending the request. See emberSendUnicast() for a description.

Returns

An EmberStatus value. [EMBER_SUCCESS](#), [EMBER_NO_BUFFERS](#), [EMBER_NETWORK_DOWN](#) or [EMBER_NETWORK_BUSY](#).

6.65.3.8 EmberStatus emberBindRequest (EmberNodeId *target*, EmberEUI64 *source*, uint8_t *sourceEndpoint*, uint16_t *clusterId*, uint8_t *type*, EmberEUI64 *destination*, EmberMulticastId *groupAddress*, uint8_t *destinationEndpoint*, EmberApsOption *options*)

Send a request to create a binding entry with the specified contents on the specified node.

Parameters

<i>target</i>	The node on which the binding will be created.
<i>source</i>	The source EUI64 in the binding entry.
<i>source-Endpoint</i>	The source endpoint in the binding entry.

<i>clusterId</i>	The cluster ID in the binding entry.
<i>type</i>	The type of binding, either UNICAST_BINDING , MULTICAST_BINDING , or UNICAST_MANY_TO_ONE_BINDING . UNICAST_MANY_TO_ONE_BINDING is an Ember-specific extension and should be used only when the target is an Ember device.
<i>destination</i>	The destination EUI64 in the binding entry for UNICAST_BINDING or UNICAST_MANY_TO_ONE_BINDING .
<i>group-Address</i>	The group address for the MULTICAST_BINDING .
<i>destination-Endpoint</i>	The destination endpoint in the binding entry for the UNICAST_BINDING or UNICAST_MANY_TO_ONE_BINDING .
<i>options</i>	The options to use when sending the request. See emberSendUnicast() for a description.

Returns

An EmberStatus value. [EMBER_SUCCESS](#), [EMBER_NO_BUFFERS](#), [EMBER_NETWORK_DOWN](#) or [EMBER_NETWORK_BUSY](#).

6.65.3.9 EmberStatus emberUnbindRequest (EmberNodeId *target*, EmberEUI64 *source*, uint8_t *sourceEndpoint*, uint16_t *clusterId*, uint8_t *type*, EmberEUI64 *destination*, EmberMulticastId *groupAddress*, uint8_t *destinationEndpoint*, EmberApsOption *options*)

Send a request to remove a binding entry with the specified contents from the specified node.

Parameters

<i>target</i>	The node on which the binding will be removed.
<i>source</i>	The source EUI64 in the binding entry.
<i>source-Endpoint</i>	The source endpoint in the binding entry.
<i>clusterId</i>	The cluster ID in the binding entry.
<i>type</i>	The type of binding, either UNICAST_BINDING , MULTICAST_BINDING , or UNICAST_MANY_TO_ONE_BINDING . UNICAST_MANY_TO_ONE_BINDING is an Ember-specific extension and should be used only when the target is an Ember device.
<i>destination</i>	The destination EUI64 in the binding entry for the UNICAST_BINDING or UNICAST_MANY_TO_ONE_BINDING .
<i>group-Address</i>	The group address for the MULTICAST_BINDING .
<i>destination-Endpoint</i>	The destination endpoint in the binding entry for the UNICAST_BINDING or UNICAST_MANY_TO_ONE_BINDING .
<i>options</i>	The options to use when sending the request. See emberSendUnicast() for a description.

Returns

An EmberStatus value.

- [EMBER_SUCCESS](#)

- [EMBER_NO_BUFFERS](#) _ [EMBER_NETWORK_DOWN](#)
- [EMBER_NETWORK_BUSY](#)

6.65.3.10 EmberStatus `emberLqiTableRequest (EmberNodeId target, uint8_t startIndex, EmberApsOption options)`

Request the specified node to send its LQI (neighbor) table. The response gives PAN ID, EUI64, node ID and cost for each neighbor. The EUI64 is only available if security is enabled. The other fields in the response are set to zero. The response format is defined in the ZigBee Device Profile Specification.

Parameters

<i>target</i>	The node whose LQI table is desired.
<i>startIndex</i>	The index of the first neighbor to include in the response.
<i>options</i>	The options to use when sending the request. See emberSendUnicast() for a description.

Returns

An EmberStatus value. [EMBER_SUCCESS](#), [EMBER_NO_BUFFERS](#), [EMBER_NETWORK_DOWN](#) or [EMBER_NETWORK_BUSY](#).

6.65.3.11 EmberStatus `emberRoutingTableRequest (EmberNodeId target, uint8_t startIndex, EmberApsOption options)`

Request the specified node to send its routing table. The response gives destination node ID, status and many-to-one flags, and the next hop node ID. The response format is defined in the ZigBee Device Profile Specification.

Parameters

<i>target</i>	The node whose routing table is desired.
<i>startIndex</i>	The index of the first route entry to include in the response.
<i>options</i>	The options to use when sending the request. See emberSendUnicast() for a description.

Returns

An EmberStatus value. [EMBER_SUCCESS](#), [EMBER_NO_BUFFERS](#), [EMBER_NETWORK_DOWN](#) or [EMBER_NETWORK_BUSY](#).

6.65.3.12 EmberStatus `emberBindingTableRequest (EmberNodeId target, uint8_t startIndex, EmberApsOption options)`

Request the specified node to send its nonvolatile bindings. The response gives source address, source endpoint, cluster ID, destination address and destination endpoint for each binding entry. The response format is defined in the ZigBee Device Profile Specification. Note that bindings that have the Ember-specific [UNICAST_MANY_TO_ONE_BINDING](#) type are reported as having the standard [UNICAST_BINDING](#) type.

Parameters

<i>target</i>	The node whose binding table is desired.
<i>startIndex</i>	The index of the first binding entry to include in the response.
<i>options</i>	The options to use when sending the request. See emberSendUnicast() for a description.

Returns

An EmberStatus value. [EMBER_SUCCESS](#), [EMBER_NO_BUFFERS](#), [EMBER_NETWORK_DOWN](#) or [EMBER_NETWORK_BUSY](#).

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

Request the specified node to remove the specified device from the network. The device to be removed must be the node to which the request is sent or one of its children.

Parameters

<i>target</i>	The node which will remove the device.
<i>device-Address</i>	All zeros if the target is to remove itself from the network or the EUI64 of a child of the target device to remove that child.
<i>leave-Request-Flags</i>	A bitmask of leave options. Include LEAVE_REQUEST_REMOVE_CHILDREN_FLAG if the target is to remove their children and/or LEAVE_REQUEST_REJOIN_FLAG if the target is to rejoin the network immediately after leaving.
<i>options</i>	The options to use when sending the request. See emberSendUnicast() for a description.

Returns

An EmberStatus value. [EMBER_SUCCESS](#), [EMBER_NO_BUFFERS](#), [EMBER_NETWORK_DOWN](#) or [EMBER_NETWORK_BUSY](#).

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

Request the specified node to allow or disallow association.

Parameters

<i>target</i>	The node which will allow or disallow association. The request can be broadcast by using a broadcast address (0xFFFFC/0xFFFFD/0xFFFF). No response is sent if the request is broadcast.
<i>duration</i>	A value of 0x00 disables joining. A value of 0xFF enables joining. Any other value enables joining for that number of seconds.
<i>authentication</i>	Controls Trust Center authentication behavior.
<i>options</i>	The options to use when sending the request. See emberSendUnicast() for a description. This parameter is ignored if the target is a broadcast address.

Returns

An EmberStatus value. [EMBER_SUCCESS](#), [EMBER_NO_BUFFERS](#), [EMBER_NETWORK_DOWN](#) or [EMBER_NETWORK_BUSY](#).

6.65.3.15 void emberSetZigDevRequestRadius (uint8_t radius)

Change the default radius for broadcast ZDO requests.

Parameters

<i>radius</i>	The radius to be used for future ZDO request broadcasts.
---------------	--

6.65.3.16 uint8_t emberGetZigDevRequestRadius (void)

Retrieve the default radius for broadcast ZDO requests.

Returns

The radius to be used for future ZDO request broadcasts.

6.65.3.17 uint8_t emberGetLastZigDevRequestSequence (void)

Provide access to the application ZDO transaction sequence number for last request. This function has been deprecated and replaced by [emberGetLastAppZigDevRequestSequence\(\)](#).

Returns

Last application ZDO transaction sequence number used

6.65.3.18 uint8_t emberGetLastAppZigDevRequestSequence (void)

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

Returns

Last application ZDO transaction sequence number used

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

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

Parameters

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

Returns

An [EmberStatus](#) value.

- [EMBER_SUCCESS](#) - The request was transmitted successfully.
- [EMBER_NO_BUFFERS](#) - Insufficient message buffers were available to construct the request.
- [EMBER_NETWORK_DOWN](#) - The node is not part of a network.
- [EMBER_NETWORK_BUSY](#) - Transmission of the request failed.

6.65.3.20 EmberStatus emberleeeAddressRequest (EmberNodeId *target*, bool *reportKids*, uint8_t *childStartIndex*, EmberApsOption *options*)

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

Parameters

<i>target</i>	The network address of the node.
<i>reportKids</i>	true to request that the target list their children in the response.
<i>childStartIndex</i>	The index of the first child to list in the response. Ignored if reportKids is false.
<i>options</i>	The options to use when sending the request. See emberSendUnicast() for a description.

Returns

An [EmberStatus](#) value.

- [EMBER_SUCCESS](#)
- [EMBER_NO_BUFFERS](#)
- [EMBER_NETWORK_DOWN](#)
- [EMBER_NETWORK_BUSY](#)

6.65.3.21 EmberStatus ezspMatchDescriptorsRequest (EmberNodeId *target*, uint16_t *profile*, uint8_t *inCount*, uint8_t *outCount*, uint16_t * *inClusters*, uint16_t * *outClusters*, EmberApsOption *options*)

Request the specified node to send a list of its endpoints that match the specified application profile and, optionally, lists of input and/or output clusters.

Parameters

<i>target</i>	The node whose matching endpoints are desired. The request can be sent unicast or broadcast ONLY to the "RX-on-when-idle-address" (0xFFFFD) If sent as a broadcast, any node that has matching endpoints will send a response.
<i>profile</i>	The application profile to match.
<i>inCount</i>	The number of input clusters. To not match any input clusters, set this value to 0.
<i>outCount</i>	The number of output clusters. To not match any output clusters, set this value to 0.

<i>inClusters</i>	The list of input clusters.
<i>outClusters</i>	The list of output clusters.
<i>options</i>	The options to use when sending the unicast request. See emberSendUnicast() for a description. This parameter is ignored if the target is a broadcast address.

Returns

An EmberStatus value. EMBER_SUCCESS, EMBER_NO_BUFFERS, EMBER_NETWORK_DOWN or EMBER_NETWORK_BUSY.

6.65.3.22 EmberStatus ezspEndDeviceBindRequest (EmberNodeId *localNodeId*, EmberEUI64 *localEui64*, uint8_t *endpoint*, uint16_t *profile*, uint8_t *inCount*, uint8_t *outCount*, uint16_t * *inClusters*, uint16_t * *outClusters*, EmberApsOption *options*)

An end device bind request to the coordinator. If the coordinator receives a second end device bind request then a binding is created for every matching cluster.

Parameters

<i>localNodeId</i>	The node ID of the local device.
<i>localEui64</i>	The EUI64 of the local device.
<i>endpoint</i>	The endpoint to be bound.
<i>profile</i>	The application profile of the endpoint.
<i>inCount</i>	The number of input clusters.
<i>outCount</i>	The number of output clusters.
<i>inClusters</i>	The list of input clusters.
<i>outClusters</i>	The list of output clusters.
<i>options</i>	The options to use when sending the request. See emberSendUnicast() for a description.

Returns

An EmberStatus value. EMBER_SUCCESS, EMBER_NO_BUFFERS, EMBER_NETWORK_DOWN or EMBER_NETWORK_BUSY.

6.65.3.23 EmberNodeId ezspDecodeAddressResponse (uint8_t * *response*, EmberEUI64 *eui64Return*)

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

Parameters

<i>response</i>	The received ZDO message with cluster ID NETWORK_ADDRESS_RESPONSE or IEEE_ADDRESS_RESPONSE.
<i>eui64Return</i>	The EUI64 from the response is copied here.

Returns

Returns the node ID from the response if the response status was EMBER_ZDP_SUCCESS. Otherwise, returns EMBER_NULL_NODE_ID.

6.66 Message Fragmentation

Transmitting

- `EmberStatus emberFragmentSendUnicast (EmberOutgoingMessageType type, uint16_t indexOrDestination, EmberApsFrame *apsFrame, EmberMessageBuffer payload, uint8_t maxFragmentSize)`
- `bool emberFragmentMessageSent (EmberApsFrame *apsFrame, EmberMessageBuffer buffer, EmberStatus status)`
- `void emberFragmentMessageSentHandler (EmberStatus status)`

Receiving

- `bool emberFragmentIncomingMessage (EmberApsFrame *apsFrame, EmberMessageBuffer payload)`
- `void emberFragmentTick (void)`

Initialization

- `void ezspFragmentInit (uint16_t receiveBufferLength, uint8_t *receiveBuffer)`

Transmitting

- `EmberStatus ezspFragmentSendUnicast (EmberOutgoingMessageType type, uint16_t indexOrDestination, EmberApsFrame *apsFrame, uint8_t maxFragmentSize, uint16_t messageLength, uint8_t *messageContents)`
- `EmberStatus ezspFragmentSourceRouteHandler (void)`
- `bool ezspFragmentMessageSent (EmberApsFrame *apsFrame, EmberStatus status)`
- `void ezspFragmentMessageSentHandler (EmberStatus status)`

Receiving

- `bool ezspFragmentIncomingMessage (EmberApsFrame *apsFrame, EmberNodeId sender, uint16_t *messageLength, uint8_t **messageContents)`
- `void ezspFragmentTick (void)`

6.66.1 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_FRAGMENT_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 `emberFragmentMessageSent()` returns true, the fragmentation code has handled the event and the application must not process it further. The fragmentation code calls the application-defined `emberFragmentMessageSentHandler()` 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 `ezspFragmentMessageSent()` returns true, the fragmentation code has handled the event and the application must not process it further. The fragmentation code calls the application-defined `ezspFragmentMessageSentHandler()` when it has finished sending the long message.

To receive a long message, the application must add a call to `ezspFragmentIncomingMessage()` at the start of its `ezspIncomingMessageHandler()`. If `ezspFragmentIncomingMessage()` returns true, the fragmentation code has handled the message and the application must not process it further. The application must also call `ezspFragmentTick()` regularly.

6.66.2 Function Documentation

6.66.2.1 EmberStatus `emberFragmentSendUnicast (EmberOutgoingMessageType type, uint16_t indexOrDestination, EmberApsFrame * apsFrame, EmberMessageBuffer payload, uint8_t 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

<code>type</code>	Specifies the outgoing message type. Must be one of <code>EMBER_OUTGOING_DIRECT</code> , <code>EMBER_OUTGOING_VIA_ADDRESS_TABLE</code> , or <code>EMBER_OUTGOING_VIA_BINDING</code> .
<code>indexOrDestination</code>	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.
<code>apsFrame</code>	The APS frame for the message.
<code>payload</code>	The long message to be sent.
<code>maxFragmentSize</code>	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 (**EMBER_FRAGMENT_WINDOW_SIZE**) is zero.

6.66.2.2 bool 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

<i>apsFrame</i>	The APS frame passed to emberMessageSentHandler() .
<i>buffer</i>	The buffer passed to emberMessageSentHandler() .
<i>status</i>	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.

6.66.2.3 void emberFragmentMessageSentHandler (EmberStatus *status*)

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

Parameters

<i>status</i>	EMBER_SUCCESS if all the blocks of the long message were delivered to the destination, otherwise EMBER_DELIVERY_FAILED , EMBER_NO_BUFFERS , EMBER_NETWORK_DOWN or EMBER_NETWORK_BUSY .
---------------	---

6.66.2.4 bool 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

<i>apsFrame</i>	The APS frame passed to emberIncomingMessageHandler() .
<i>payload</i>	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.

6.66.2.5 void emberFragmentTick (void)

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

6.66.2.6 void ezspFragmentInit (uint16_t *receiveBufferLength*, uint8_t * *receiveBuffer*)

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

Parameters

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

6.66.2.7 EmberStatus ezspFragmentSendUnicast (EmberOutgoingMessageType *type*, uint16_t *indexOrDestination*, EmberApsFrame * *apsFrame*, uint8_t *maxFragmentSize*, uint16_t *messageLength*, uint8_t * *messageContents*)

Sends a long message by splitting it into blocks. Only one long message can be sent at a time. Calling this function a second time aborts the first message.

Parameters

<i>type</i>	Specifies the outgoing message type. Must be one of EMBER_OUTGOING_DIRECT , EMBER_OUTGOING_VIA_ADDRESS_TABLE , or EMBER_OUTGOING_VIA_BINDING .
<i>indexOr-Destination</i>	Depending on the type of addressing used, this is either the EmberNodeId of the destination, an index into the address table, or an index into the binding table.
<i>apsFrame</i>	The APS frame for the message.

<i>max-Fragment-Size</i>	The message will be broken into blocks no larger than this.
<i>message-Length</i>	The length of the messageContents parameter in bytes.
<i>message-Contents</i>	The long message to be sent.

Returns

An EmberStatus value.

- [EMBER_SUCCESS](#)
- [EMBER_MESSAGE_TOO_LONG](#)
- [EMBER_NETWORK_DOWN](#)
- [EMBER_NETWORK_BUSY](#)
- [EMBER_INVALID_CALL](#) is returned if messageLength is zero or if the window size ([::EZSP_CONFIG_FRAGMENT_WINDOW_SIZE](#)) is zero.

6.66.2.8 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_HANDLER](#) in its configuration header if it defines this callback.

Returns

[EMBER_SUCCESS](#) if the source route has been set. Any other value will abort transmission of the current long message.

6.66.2.9 bool ezspFragmentMessageSent(EmberApsFrame * *apsFrame*, EmberStatus *status*)

The application must call this function at the start of its [ezspMessageSentHandler\(\)](#). If it returns true, the fragmentation code has handled the event and the application must not process it further.

Parameters

<i>apsFrame</i>	The APS frame passed to ezspMessageSentHandler() .
<i>status</i>	The status passed to ezspMessageSentHandler() .

Returns

true if the sent message was a block of a long message. The fragmentation code has handled the event so the application must return immediately from its ezspMessageSentHandler(). Returns false otherwise. The fragmentation code has not handled the event so the application must continue to process it.

6.66.2.10 void ezspFragmentMessageSentHandler (EmberStatus status)

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

Parameters

<i>status</i>	EMBER_SUCCESS if all the blocks of the long message were delivered to the destination, otherwise EMBER_DELIVERY_FAILED , EMBER_NETWORK_DOWN or EMBER_NETWORK_BUSY .
---------------	---

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

The application must call this function at the start of its ezspIncomingMessageHandler(). If it returns true, the fragmentation code has handled the message and the application must not process it further. When the final block of a long message is received, this function replaces the message with the reassembled long message and returns false so that the application processes it.

Parameters

<i>apsFrame</i>	The APS frame passed to ezspIncomingMessageHandler().
<i>sender</i>	The sender passed to ezspIncomingMessageHandler().
<i>messageLength</i>	A pointer to the message length passed to ezspIncomingMessageHandler().
<i>messageContents</i>	A pointer to the message contents passed to ezspIncomingMessageHandler().

Returns

true if the incoming message was a block of an incomplete long message. The fragmentation code has handled the message so the application must return immediately from its ezspIncomingMessageHandler(). Returns false if the incoming message was not part of a long message. The fragmentation code has not handled the message so the application must continue to process it. Returns false if the incoming message was a block that completed a long message. The fragmentation code replaces the message with the reassembled long message so the application must continue to process it.

6.66.2.12 void ezspFragmentTick (void)

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

6.67 Network Manager

Macros

- #define NM_WARNING_LIMIT
- #define NM_WINDOW_SIZE
- #define NM_CHANNEL_MASK
- #define NM_WATCHLIST_SIZE

Functions

- void nmUtilWarningHandler (void)
- bool nmUtilProcessIncoming ([EmberApsFrame](#) *apsFrame, uint8_t messageLength, uint8_t *message)
- [EmberStatus nmUtilChangeChannelRequest](#) (void)

6.67.1 Detailed Description

The network manager is an optional function of one device in the ZigBee network. Devices on the network send unsolicited ZDO energy scan reports to the network manager when more than 25% of unicasts fail within a rolling window, but no more than once every 15 minutes.

See [network-manager.h](#) for source code.

The network manager is the coordinator by default but can be changed via [emberSetNetworkManagerRequest\(\)](#). It processes the energy scan reports from the devices on the network, and is responsible for determining if the network should change channels in an attempt to resolve reliability problems that might be caused by RF interference.

Note that EmberZNet networks are quite robust to many interferers such as 802.11 (WiFi), and the presence of interferers does not necessarily degrade application performance or require a channel change. Because changing channels is disruptive to network operation, channel changes should not be done solely because of observed higher noise levels, as the noise may not be causing any problem.

Also note that receipt of unsolicited scan reports is only an indication of unicast failures in the network. These might be caused by RF interference, or for some other reason such as a device failure. In addition, only the application can tell whether the delivery failures caused an actual problem for the application. In general, it is difficult to automatically determine with certainty that network problems are caused by RF interference. Channel changes should therefore be done sparingly and with careful application design.

The stack provides three APIs in [include/zigbee-device-stack.h](#):

- [emberEnergyScanRequest](#)
- [emberSetNetworkManagerRequest](#)
- [emberChannelChangeRequest](#)

This library provides some additional functions:

- [nmUtilProcessIncomingMessage](#)

- nmUtilWarningHandler
- nmUtilChangeChannelRequest

An application implementing network manager functionality using this library should pass all incoming messages to nmUtilProcessIncomingMessage, which will return true if the message was processed as a ZDO energy scan report. The application should not make any calls to [emberEnergyScanRequest\(\)](#), as the library assumes all incoming scan reports are unsolicited and indicate unicast failures.

When NM_WARNING_LIMIT reports have been processed within NM_WINDOW_SIZE minutes, the nmUtilWarningHandler callback, which must be implemented by the application, is invoked. The default values for these parameters are set in [network-manager.h](#) and may be modified using #defines within the application configuration header.

The application may use the nmUtilWarningHandler callback, along with other application-specific information, to decide if and when to change the channel by calling nmUtilChangeChannelRequest. This function chooses a new channel from the NM_CHANNEL_MASK parameter using information gathered over time.

In the event of a network-wide channel change, it is possible that some devices, especially sleepy end devices, do not receive the broadcast and remain on the old channel. Devices should use the API [emberFindAndRejoinNetwork](#) to get back to the right channel.

Two implementations of this library are provided: network-manager.c, and network-manager-lite.c. The former keeps track of the mean and deviation of the energy on each channel and uses these stats to choose the channel to change to. This consumes a fair amount of RAM. The latter takes the simpler (and possibly more effective) approach of just avoiding past bad channels. Application developers are encouraged to use and modify either of these solutions to take into account their own application-specific needs.

6.67.2 Macro Definition Documentation

6.67.2.1 #define NM_WARNING_LIMIT

Definition at line [97](#) of file [network-manager.h](#).

6.67.2.2 #define NM_WINDOW_SIZE

Definition at line [101](#) of file [network-manager.h](#).

6.67.2.3 #define NM_CHANNEL_MASK

Definition at line [107](#) of file [network-manager.h](#).

6.67.2.4 #define NM_WATCHLIST_SIZE

Definition at line [113](#) of file [network-manager.h](#).

6.67.3 Function Documentation

6.67.3.1 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.

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

Called from the app in emberIncomingMessageHandler. Returns true if and only if the library processed the message.

Parameters

<i>apsFrame</i>	
<i>message- Length</i>	
<i>message</i>	

6.67.3.3 EmberStatus nmUtilChangeChannelRequest (void)

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

6.68 Serial Communication

Functions

- `EmberStatus emberSerialInit (uint8_t port, SerialBaudRate rate, SerialParity parity, uint8_t stopBits)`
- `uint16_t emberSerialReadAvailable (uint8_t port)`
- `EmberStatus emberSerialReadByte (uint8_t port, uint8_t *dataByte)`
- `EmberStatus emberSerialReadData (uint8_t port, uint8_t *data, uint16_t length, uint16_t *bytesRead)`
- `EmberStatus emberSerialReadDataTimeout (uint8_t port, uint8_t *data, uint16_t length, uint16_t *bytesRead, uint16_t firstByteTimeout, uint16_t subsequentByteTimeout)`
- `EmberStatus emberSerialReadLine (uint8_t port, char *data, uint8_t max)`
- `EmberStatus emberSerialReadPartialLine (uint8_t port, char *data, uint8_t max, uint8_t *index)`
- `uint16_t emberSerialWriteAvailable (uint8_t port)`
- `uint16_t emberSerialWriteUsed (uint8_t port)`
- `EmberStatus emberSerialWriteByte (uint8_t port, uint8_t dataByte)`
- `EmberStatus emberSerialWriteHex (uint8_t port, uint8_t dataByte)`
- `EmberStatus emberSerialWriteString (uint8_t port, PGM_P string)`
- `XAP2B_PAGEZERO_ON EmberStatus emberSerialPrintf (uint8_t port, PGM_P formatString,...)`
- `XAP2B_PAGEZERO_OFF XAP2B_PAGEZERO_ON EmberStatus emberSerialPrintfLine (uint8_t port, PGM_P formatString,...)`
- `XAP2B_PAGEZERO_OFF XAP2B_PAGEZERO_ON EmberStatus emberSerialPrintCarriageReturn (uint8_t port)`
- `XAP2B_PAGEZERO_OFF EmberStatus emberSerialPrintVarArg (uint8_t port, PGM_P formatString, va_list ap)`
- `EmberStatus emberSerialWriteData (uint8_t port, uint8_t *data, uint8_t length)`
- `EmberStatus emberSerialWriteBuffer (uint8_t port, EmberMessageBuffer buffer, uint8_t start, uint8_t length)`
- `XAP2B_PAGEZERO_ON EmberStatus emberSerialWaitSend (uint8_t port)`
- `XAP2B_PAGEZERO_OFF EmberStatus emberSerialGuaranteedPrintf (uint8_t port, PGM_P formatString,...)`
- `void emberSerialBufferTick (void)`
- `void emberSerialFlushRx (uint8_t port)`
- `bool emberSerialUnused (uint8_t port)`

6.68.1 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.

6.68.2 Function Documentation

6.68.2.1 EmberStatus emberSerialInit (uint8_t port, SerialBaudRate rate, SerialParity parity, uint8_t stopBits)

Initializes a serial port to a specific baud rate, parity, and number of stop bits. Eight data bits are always used.

Parameters

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

Returns

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

6.68.2.2 `uint16_t emberSerialReadAvailable (uint8_t port)`

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

Parameters

<i>port</i>	A serial port number (0 or 1).
-------------	--------------------------------

Returns

The number of bytes available.

6.68.2.3 `EmberStatus emberSerialReadByte (uint8_t port, uint8_t * dataByte)`

Reads a byte from the specified RX queue. If an error is returned, the dataByte should be ignored. For errors other than `EMBER_SERIAL_RX_EMPTY` multiple bytes of data may have been lost and serial protocols should attempt to resynchronize.

Parameters

<i>port</i>	A serial port number (0 or 1).
<i>dataByte</i>	A pointer to storage location for the byte.

Returns

One of the following (see the Main Page):

- `EMBER_SERIAL_RX_EMPTY` if no data is available
- `EMBER_SERIAL_RX_OVERFLOW` if the serial receive fifo was out of space
- `EMBER_SERIAL_RX_FRAME_ERROR` if a framing error was received
- `EMBER_SERIAL_RX_PARITY_ERROR` if a parity error was received
- `EMBER_SERIAL_RX_OVERRUN_ERROR` if the hardware fifo was out of space
- `EMBER_SUCCESS` if a data byte is returned

6.68.2.4 `EmberStatus emberSerialReadData (uint8_t port, uint8_t * data, uint16_t length, uint16_t * bytesRead)`

Reads bytes from the specified RX queue. Blocks until the full length has been read or an error occurs. In the event of an error, some valid data may have already been read before the error occurred, in which case that data will be in the buffer pointed to by `data` and the number of bytes successfully read will be placed in `bytesRead`.

Parameters

<i>port</i>	A serial port number (0 or 1).
<i>data</i>	A pointer to storage location for the data. It must be at least <code>length</code> in size.
<i>length</i>	The number of bytes to read.
<i>bytesRead</i>	A pointer to a location that will receive the number of bytes read. If the function returns early due to an error, this value may be less than <code>length</code> . This parameter may be <code>NULL</code> , 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

6.68.2.5 EmberStatus emberSerialReadDataTimeout (*uint8_t port*, *uint8_t * data*, *uint16_t length*, *uint16_t * bytesRead*, *uint16_t firstByteTimeout*, *uint16_t subsequentByteTimeout*)

Reads bytes from the specified RX queue, up to a maximum of *length* bytes. The function may return before *length* bytes is read if a timeout is reached or an error occurs. Returns [EMBER_SERIAL_RX_EMPTY](#) if a timeout occurs.

Parameters

<i>port</i>	A serial port number (0 or 1).
<i>data</i>	A pointer to storage location for the data. It must be at least <i>length</i> in size.
<i>length</i>	The maximum number of bytes to read.
<i>bytesRead</i>	A pointer to a location that will receive the number of bytes read. If the function returns early due to an error or timeout, this value may be less than <i>length</i> . This parameter may be NULL, in which case it is ignored.
<i>firstByte-Timeout</i>	The amount of time, in milliseconds, to wait for the first byte to arrive (if the queue is empty when the function is called). This value must be a minimum of 2 due to the timer resolution.
<i>subsequent-ByteTimeout</i>	The amount of time, in milliseconds, to wait after the previous byte was received for the next byte to arrive. This value must be a minimum of 2 due to the timer resolution.

Returns

One of the following (see the Main Page):

- [EMBER_SERIAL_RX_EMPTY](#) if the timeout was exceeded before the requested amount of data was read
- [EMBER_SERIAL_RX_OVERFLOW](#) if the serial receive fifo was out of space
- [EMBER_SERIAL_RX_FRAME_ERROR](#) if a framing error was received
- [EMBER_SERIAL_RX_PARITY_ERROR](#) if a parity error was received
- [EMBER_SERIAL_RX_OVERRUN_ERROR](#) if the hardware fifo was out of space
- [EMBER_SUCCESS](#) if all the data requested is returned

6.68.2.6 EmberStatus emberSerialReadLine (*uint8_t port, char * data, uint8_t max*)

Simulates a terminal interface, reading a line of characters at a time. Supports backspace. Always converts to uppercase. Blocks until a line has been read or max has been exceeded. Calls on [halResetWatchdog\(\)](#).

Parameters

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

Returns

[EMBER_SUCCESS](#)

6.68.2.7 EmberStatus emberSerialReadPartialLine (*uint8_t port, char * data, uint8_t max, uint8_t * index*)

Simulates a partial terminal interface, reading a line of characters at a time. Supports backspace. Always converts to uppercase. Returns [EMBER_SUCCESS](#) when a line has been read or max has been exceeded. Must initialize the index variable to 0 to start a line.

Parameters

<i>port</i>	A serial port number (0 or 1).
<i>data</i>	A pointer to storage location for the read line. There must be <i>max</i> contiguous bytes available at this location.
<i>max</i>	The maximum number of bytes to read.
<i>index</i>	The address of a variable that holds the place in the <i>data</i> to continue. Set to 0 to start a line read.

Returns

One of the following (see the Main Page):

- [EMBER_SERIAL_RX_EMPTY](#) if a partial line is in progress.
- [EMBER_SERIAL_RX_OVERFLOW](#) if the serial receive fifo was out of space.
- [EMBER_SERIAL_RX_FRAME_ERROR](#) if a framing error was received.
- [EMBER_SERIAL_RX_PARITY_ERROR](#) if a parity error was received.
- [EMBER_SERIAL_RX_OVERRUN_ERROR](#) if the hardware fifo was out of space.
- [EMBER_SUCCESS](#) if a full line is ready.

6.68.2.8 uint16_t emberSerialWriteAvailable (*uint8_t port*)

Returns the number of bytes (in FIFO mode) or messages (in buffered mode) that can currently be queued to send without blocking or dropping.

Parameters

<i>port</i>	A serial port number (0 or 1).
-------------	--------------------------------

Returns

The number of bytes or messages available for queueing.

6.68.2.9 uint16_t emberSerialWriteUsed (uint8_t port)

Returns the number of bytes (in FIFO mode) or messages (in buffered mode) that are currently queued and still being sent.

Parameters

<i>port</i>	A serial port number (0 or 1).
-------------	--------------------------------

Returns

The number of bytes or messages available for queueing.

6.68.2.10 EmberStatus emberSerialWriteByte (uint8_t port, uint8_t dataByte)

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

Parameters

<i>port</i>	A serial port number (0 or 1).
<i>dataByte</i>	The byte to be queued.

Returns

One of the following (see the Main Page):

- [EMBER_SERIAL_TX_OVERFLOW](#) indicates that data was dropped.
- [EMBER_NO_BUFFERS](#) indicates that there was an insufficient number of available stack buffers.
- [EMBER_SUCCESS](#).

6.68.2.11 EmberStatus emberSerialWriteHex (uint8_t port, uint8_t dataByte)

Converts a given byte of data to its two-character ASCII hex representation and queues it for transmission on the specified port. Values less than 0xF are always zero padded and queued as "0F".

Parameters

<i>port</i>	A serial port number (0 or 1).
<i>dataByte</i>	The byte to be converted.

Returns

One of the following (see the Main Page):

- [EMBER_SERIAL_TX_OVERFLOW](#) indicates that data was dropped.
- [EMBER_NO_BUFFERS](#) indicates that there was an insufficient number of available stack buffers.
- [EMBER_SUCCESS](#).

6.68.2.12 EmberStatus emberSerialWriteString (uint8_t port, PGM_P string)

Queues a string for transmission on the specified port.

Parameters

<i>port</i>	A serial port number (0 or 1).
<i>string</i>	The string to be queued.

Returns

One of the following (see the Main Page):

- [EMBER_SERIAL_TX_OVERFLOW](#) indicates that data was dropped.
- [EMBER_NO_BUFFERS](#) indicates that there was an insufficient number of available stack buffers.
- [EMBER_SUCCESS](#).

6.68.2.13 XAP2B_PAGEZERO_ON EmberStatus emberSerialPrintf (uint8_t port, PGM_P formatString, ...)

Printf for printing on a specified port. Supports the following format specifiers:

- %% percent sign
- c single-byte character
- s RAM string
- p flash string (nonstandard specifier)
- u 2-byte unsigned decimal
- d 2-byte signed decimal
- l 4-byte signed decimal
- x %2x %4x 1-, 2-, 4-byte hex value (always 0 padded) (nonstandard specifier)

Parameters

<i>port</i>	A serial port number (0 or 1).
<i>formatString</i>	The string to print.
...	Format specifiers.

Returns

One of the following (see the Main Page):

- [EMBER_SERIAL_TX_OVERFLOW](#) indicates that data was dropped.
- [EMBER_NO_BUFFERS](#) indicates that there was an insufficient number of available stack buffers.
- [EMBER_SUCCESS](#).

6.68.2.14 XAP2B_PAGEZERO_OFF XAP2B_PAGEZERO_ON EmberStatus emberSerialPrintfLine (*uint8_t port*, *PGM_P formatString*, ...)

Printf for printing on a specified port. Same as [emberSerialPrintf\(\)](#) except it prints a carriage return at the end of the text.

Parameters

<i>port</i>	A serial port number (0 or 1).
<i>formatString</i>	The string to print.
...	Format specifiers.

Returns

One of the following (see the Main Page):

- [EMBER_SERIAL_TX_OVERFLOW](#) indicates that data was dropped.
- [EMBER_NO_BUFFERS](#) indicates that there was an insufficient number of available stack buffers.
- [EMBER_SUCCESS](#).

6.68.2.15 XAP2B_PAGEZERO_OFF XAP2B_PAGEZERO_ON EmberStatus emberSerialPrintCarriageReturn (*uint8_t port*)

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

Parameters

<i>port</i>	A serial port number (0 or 1).
-------------	--------------------------------

Returns

One of the following (see the Main Page):

- [EMBER_SERIAL_TX_OVERFLOW](#) indicates that data was dropped.
- [EMBER_NO_BUFFERS](#) indicates that there was an insufficient number of available stack buffers.
- [EMBER_SUCCESS](#).

6.68.2.16 XAP2B_PAGEZERO_OFF EmberStatus emberSerialPrintVarArg (*uint8_t port*, *PGM_P formatString*, *va_list ap*)

Prints a format string with a variable argument list.

Parameters

<i>port</i>	A serial port number (0 or 1).
<i>formatString</i>	A printf style format string.
<i>ap</i>	A variable argument list.

Returns

One of the following (see the Main Page):

- [EMBER_SERIAL_TX_OVERFLOW](#) indicates that data was dropped.
- [EMBER_NO_BUFFERS](#) indicates that there was an insufficient number of available stack buffers.
- [EMBER_SUCCESS](#).

6.68.2.17 EmberStatus emberSerialWriteData (*uint8_t port*, *uint8_t * data*, *uint8_t length*)

Queues an arbitrary chunk of data for transmission on a specified port.

Parameters

<i>port</i>	A serial port number (0 or 1).
<i>data</i>	A pointer to data.
<i>length</i>	The number of bytes to queue.

Returns

One of the following (see the Main Page):

- [EMBER_SERIAL_TX_OVERFLOW](#) indicates that data was dropped.
- [EMBER_NO_BUFFERS](#) indicates that there was an insufficient number of available stack buffers.
- [EMBER_SUCCESS](#).

6.68.2.18 EmberStatus emberSerialWriteBuffer (*uint8_t port*, *EmberMessageBuffer buffer*, *uint8_t start*, *uint8_t length*)

Queues data contained in linked stack buffers for transmission on a specified port. Can specify an arbitrary initial offset within the linked buffer chain.

Parameters

<i>port</i>	A serial port number (0 or 1).
<i>buffer</i>	The starting buffer in linked buffer chain.
<i>start</i>	The offset from first buffer in chain.
<i>length</i>	The number of bytes to queue.

Returns

One of the following (see the Main Page):

- [EMBER_SERIAL_TX_OVERFLOW](#) indicates that data was dropped.
- [EMBER_NO_BUFFERS](#) indicates that there was an insufficient number of available stack buffers.
- [EMBER_SUCCESS](#).

6.68.2.19 XAP2B_PAGEZERO_ON EmberStatus emberSerialWaitSend (uint8_t port)

Waits for all data currently queued on the specified port to be transmitted before returning. **Note:** Call this function before serial reinitialization to ensure that transmission is complete.

Parameters

<i>port</i>	A serial port number (0 or 1).
-------------	--------------------------------

Returns

One of the following (see the Main Page):

- [EMBER_SERIAL_TX_OVERFLOW](#) indicates that data was dropped.
- [EMBER_NO_BUFFERS](#) indicates that there was an insufficient number of available stack buffers.
- [EMBER_SUCCESS](#).

6.68.2.20 XAP2B_PAGEZERO_OFF EmberStatus emberSerialGuaranteedPrintf (uint8_t port, PGM_P formatString, ...)

A printf routine that takes over the specified serial port and immediately transmits the given data regardless of what is currently queued. Does not return until the transmission is complete.

Application Usage:

Useful for fatal situations (such as asserts) where the node will be reset, but information on the cause for the reset needs to be transmitted first.

Parameters

<i>port</i>	A serial port number (0 or 1).
<i>formatString</i>	The string to print.
...	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**.

6.68.2.21 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.

6.68.2.22 void emberSerialFlushRx (uint8_t port)

Flushes the receive buffer in case none of the incoming serial data is wanted.

Parameters

<i>port</i>	A serial port number (0 or 1).
-------------	--------------------------------

6.68.2.23 bool emberSerialUnused (uint8_t port)

Indicates whether the port is unused or invalid.

Parameters

<i>port</i>	A serial port number.
-------------	-----------------------

Returns

true if the port is unused or invalid.

6.69 Command Interpreter

Data Structures

- struct `EmberCommandEntry`
Command entry for a command table.

Macros

- `#define MAX_TOKEN_COUNT`
- `#define emberCommandEntryAction(name, action, argumentTypes, description)`
- `#define emberCommandEntryActionWithDetails(name, action, argumentTypes, description, argumentDescriptionArray)`
- `#define emberCommandEntrySubMenu(name, subMenu, description)`
- `#define emberCommandEntryTerminator()`
- `#define EMBER_COMMAND_INTERPRETER_CONFIGURATION_ECHO`
- `#define emberProcessCommandInput(port)`
- `#define emberCommandInterpreterEchoOn()`
- `#define emberCommandInterpreterEchoOff()`
- `#define emberCommandInterpreterIsEchoOn()`

Typedefs

- `typedef void(* CommandAction)(void)`

Enumerations

- enum `EmberCommandStatus` {

`EMBER_CMD_SUCCESS, EMBER_CMD_ERR_PORT_PROBLEM, EMBER_CMD_ERR_NO_SUCH_COMMAND, EMBER_CMD_ERR_WRONG_NUMBER_OF_ARGUMENTS,`

`EMBER_CMD_ERR_ARGUMENT_OUT_OF_RANGE, EMBER_CMD_ERR_ARGUMENT_SYNTAX_ERROR, EMBER_CMD_ERR_STRING_TOO_LONG, EMBER_CMD_ERR_INVALID_ARGUMENT_TYPE }`

Functions

- `void emberCommandReaderSetDefaultBase (uint8_t base)`
- `void emberCommandActionHandler (const CommandAction action)`
- `void emberCommandErrorHandler (EmberCommandStatus status)`
- `void emberPrintCommandUsage (EmberCommandEntry *entry)`
- `void emberPrintCommandUsageNotes (void)`
- `void emberPrintCommandTable (void)`
- `void emberCommandClearBuffer (void)`
- `void emberCommandReaderInit (void)`
- `bool emberProcessCommandString (uint8_t *input, uint8_t sizeOrPort)`

Variables

- `EmberCommandEntry * emberCurrentCommand`
- `EmberCommandEntry emberCommandTable []`
- `uint8_t emberCommandInterpreter2Configuration`

Command Table Settings

- `#define EMBER_MAX_COMMAND_ARGUMENTS`
- `#define EMBER_COMMAND_BUFFER_LENGTH`
- `#define EMBER_COMMAND_INTEPRETER_HAS_DESCRIPTION_FIELD`

Functions to Retrieve Arguments

Use the following functions in your functions that process commands to retrieve arguments from the command interpreter. These functions pull out unsigned integers, signed integers, and strings, and hex strings. Index 0 is the first command argument.

- `uint8_t emberCommandArgumentCount (void)`
- `uint32_t emberUnsignedCommandArgument (uint8_t argNum)`
- `int32_t emberSignedCommandArgument (uint8_t argNum)`
- `bool emberStringToHostOrderIpv4Address (const uint8_t *string, uint32_t *hostOrderIpv4Address)`
- `bool emberStringArgumentToHostOrderIpv4Address (uint8_t argNum, uint32_t *hostOrderIpv4Address)`
- `uint8_t * emberStringCommandArgument (int8_t argNum, uint8_t *length)`
- `const char * emberCommandName (void)`
- `uint8_t emberCopyStringArgument (int8_t argNum, uint8_t *destination, uint8_t maxLength, bool leftPad)`
- `uint8_t emberCopyBigEndianEui64Argument (int8_t index, EmberEUI64 destination)`
- `#define emberCopyKeyArgument(index, keyDataPointer)`
- `#define emberCopyEui64Argument(index, eui64)`
- `#define emberGetEui64Argument(index, eui64)`

6.69.1 Detailed Description

Interpret serial port commands. See `command-interpreter2.c` for source code.

See the following application usage example followed by a brief explanation.

```
// Usage: network form 22 0xAB12 -3 { 00 01 02 A3 A4 A5 A6 A7 }
void formCommand(void)
{
    uint8_t channel = emberUnsignedCommandArgument(0)
    ;
    uint16_t panId = emberUnsignedCommandArgument(1)
    ;
    int8_t power = emberSignedCommandArgument(2);
    uint8_t length;
    uint8_t *eui64 = emberStringCommandArgument(3, &
        length);
    ...
    ... call emberFormNetwork() etc
    ...
}
```

```

}

// The main command table.
EmberCommandEntry emberCommandTable[] = {
    emberCommandEntrySubMenu("network",    networkCommands,
                           "Network form/join commands"),
    emberCommandEntryAction("status",      statusCommand,
                           "Prints application status"),
    ...
    emberCommandEntryTerminator()
};

// The table of network commands.
EmberCommandEntry networkCommands[] = {
    emberCommandEntryAction("form",        formCommand, "Form a network"),
    emberCommandEntryAction("join",        joinCommand, "Join a network"),
    ...
    emberCommandEntryTerminator()
};

void main(void)
{
    emberCommandReaderInit();
    while(0) {
        ...
        // Process input and print prompt if it returns true.
        if (emberProcessCommandInput(serialPort)) {
            emberSerialPrintf(1, "%p>", PROMPT);
        }
        ...
    }
}

```

1. Applications specify the commands that can be interpreted by defining the `emberCommandTable` array of type `EmberCommandEntry`. The table includes the following information for each command:
 - (a) The full command name.
 - (b) Your application's function name that implements the command.
 - (c) An `EmberCommandEntry::argumentTypes` string specifies the number and types of arguments the command accepts. See `::argumentTypes` for details.
 - (d) A description string explains the command.
2. A default error handler `emberCommandErrorHandler()` is provided to deal with incorrect command input. Applications may override it.
3. The application calls `emberCommandReaderInit()` to initialize, and `emberProcessCommandInput()` in its main loop.
4. Within the application's command functions, use `emberXXXCommandArgument()` functions to retrieve command arguments.

The command interpreter does extensive processing and validation of the command input before calling the function that implements the command. It checks that the number, type, syntax, and range of all arguments are correct. It performs any conversions necessary (for example, converting integers and strings input in hexadecimal notation into the corresponding bytes), so that no additional parsing is necessary within command functions. If there is an error in the command input, `emberCommandErrorHandler()` is called rather than a command function.

The command interpreter allows inexact matches of command names. The input command may be either shorter or longer than the actual command. However, if more than one inexact match is found and there is no exact match, an error of type `EMBER_CMD_ERROR_NO_SUCH_COMMAND` will be generated. To disable this feature, define `EMBER_REQUIRE_EXACT_COMMAND_NAME` in the application configuration header.

6.69.2 Macro Definition Documentation

6.69.2.1 #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](#).

6.69.2.2 #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](#).

6.69.2.3 #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](#).

6.69.2.4 #define MAX_TOKEN_COUNT

Definition at line 122 of file [command-interpreter2.h](#).

6.69.2.5 #define emberCommandEntryAction(*name*, *action*, *argumentTypes*, *description*)

Definition at line 187 of file [command-interpreter2.h](#).

6.69.2.6 #define emberCommandEntryActionWithDetails(*name*, *action*, *argumentTypes*, *description*, *argumentDescriptionArray*)

Definition at line 190 of file [command-interpreter2.h](#).

6.69.2.7 #define emberCommandEntrySubMenu(*name*, *subMenu*, *description*)

Definition at line 198 of file [command-interpreter2.h](#).

6.69.2.8 #define emberCommandEntryTerminator()

Definition at line 202 of file [command-interpreter2.h](#).

6.69.2.9 #define EMBER_COMMAND_INTERPRETER_CONFIGURATION_ECHO

Definition at line 243 of file [command-interpreter2.h](#).

6.69.2.10 #define emberCopyKeyArgument(*index*, *keyDataPointer*)

A convenience macro for copying security key arguments to an [EmberKeyData](#) pointer.

Definition at line [329](#) of file [command-interpreter2.h](#).

6.69.2.11 #define emberCopyEui64Argument(*index*, *eui64*)

A convenience macro for copying eui64 arguments to an EmberEUI64.

Definition at line [336](#) of file [command-interpreter2.h](#).

6.69.2.12 #define emberGetEui64Argument(*index*, *eui64*)

A convenience macro for copying security key arguments to an [EmberKeyData](#) pointer.

Definition at line [338](#) of file [command-interpreter2.h](#).

6.69.2.13 #define emberProcessCommandInput(*port*)

Process input coming in on the given serial port.

Returns

true if an end of line character was read. If the application uses a command line prompt, this indicates it is time to print the prompt.

```
void emberProcessCommandInput(uint8_t port);
```

Definition at line [384](#) of file [command-interpreter2.h](#).

6.69.2.14 #define emberCommandInterpreterEchoOn()

Turn echo of command line on.

Definition at line [389](#) of file [command-interpreter2.h](#).

6.69.2.15 #define emberCommandInterpreterEchoOff()

Turn echo of command line off.

Definition at line [395](#) of file [command-interpreter2.h](#).

6.69.2.16 #define emberCommandInterpreterIsEchoOn()

Returns true if echo is on, false otherwise.

Definition at line [401](#) of file [command-interpreter2.h](#).

6.69.3 Typedef Documentation

6.69.3.1 typedef void(* CommandAction)(void)

Definition at line [124](#) of file [command-interpreter2.h](#).

6.69.4 Enumeration Type Documentation

6.69.4.1 enum EmberCommandStatus

Command error states.

If you change this list, ensure you also change the strings that describe these errors in the array emberCommandErrorNames[] in command-interpreter.c.

Enumerator:

```
EMBER_CMD_SUCCESS
EMBER_CMD_ERR_PORT_PROBLEM
EMBER_CMD_ERR_NO SUCH_COMMAND
EMBER_CMD_ERR_WRONG_NUMBER_OF_ARGUMENTS
EMBER_CMD_ERR_ARGUMENT_OUT_OF_RANGE
EMBER_CMD_ERR_ARGUMENT_SYNTAX_ERROR
EMBER_CMD_ERR_STRING_TOO_LONG
EMBER_CMD_ERR_INVALID_ARGUMENT_TYPE
```

Definition at line 251 of file [command-interpreter2.h](#).

6.69.5 Function Documentation

6.69.5.1 uint8_t emberCommandArgumentCount (void)

Returns the number of arguments for the current command.

6.69.5.2 uint32_t emberUnsignedCommandArgument (uint8_t argNum)

Retrieves unsigned integer arguments.

6.69.5.3 int32_t emberSignedCommandArgument (uint8_t argNum)

Retrieves signed integer arguments.

6.69.5.4 bool emberStringToHostOrderIpv4Address (const uint8_t * string, uint32_t * hostOrderIpv4Address)

Parses an IPv4 address string and returns a host order uint32_t. Returns true if address is valid dotted quad notation (A.B.C.D), false otherwise.

6.69.5.5 bool emberStringArgumentToHostOrderIpv4Address (uint8_t argNum, uint32_t * hostOrderIpv4Address)

Parses an IPv4 address string from a command argument and returns host order uint32_t. Returns true if address is valid dotted quad notation (A.B.C.D), false otherwise.

6.69.5.6 `uint8_t* emberStringCommandArgument (int8_t argNum, uint8_t * length)`

Retrieve quoted string or hex string arguments. Hex strings have already been converted into binary. To retrieve the name of the command itself, use an argNum of -1. For example, to retrieve the first character of the command, do: `uint8_t firstChar = emberStringCommandArgument(-1, NULL)[0]`. If the command is nested, an index of -2, -3, etc will work to retrieve the higher level command names. Note that [-1] only returns the text entered. If an abbreviated command name is entered only the text entered will be returned with [-1].

6.69.5.7 `const char* emberCommandName (void)`

A convenience macro for copying security key arguments to an [EmberKeyData](#) pointer.

6.69.5.8 `uint8_t emberCopyStringArgument (int8_t argNum, uint8_t * destination, uint8_t maxLength, bool leftPad)`

Copies the string argument to the given destination up to maxLength. If the argument length is nonzero but less than maxLength and leftPad is true, leading zeroes are prepended to bring the total length of the target up to maxLength. If the argument is longer than the maxLength, it is truncated to maxLength. Returns the minimum of the argument length and maxLength.

This function is commonly used for reading in hex strings such as EUI64 or key data and left padding them with zeroes. See [emberCopyKeyArgument](#) and [emberCopyEui64Argument](#) for convenience macros for this purpose.

6.69.5.9 `uint8_t emberCopyBigEndianEui64Argument (int8_t index, EmberEUI64 destination)`

Copies eui64 arguments in big-endian format to an EmberEUI64. This is useful because eui64s are often presented to users in big-endian format even though they are used in software in little-endian format.

6.69.5.10 `void emberCommandReaderSetDefaultBase (uint8_t base)`

6.69.5.11 `void emberCommandActionHandler (const CommandAction action)`

The application may implement this handler. To override the default handler, define EMBER_APPLICATION_HAS_COMMAND_ACTION_HANDLER in the CONFIGURATION_HEADER.

6.69.5.12 `void emberCommandErrorHandler (EmberCommandStatus status)`

The application may implement this handler. To override the default handler, define EMBER_APPLICATION_HAS_COMMAND_ERROR_HANDLER in the CONFIGURATION_HEADER. Defining this will also remove the help functions [emberPrintCommandUsage\(\)](#), [emberPrintCommandUsageNotes\(\)](#), and [emberPrintCommandTable\(\)](#).

6.69.5.13 void emberPrintCommandUsage (EmberCommandEntry * *entry*)

6.69.5.14 void emberPrintCommandUsageNotes (void)

6.69.5.15 void emberPrintCommandTable (void)

6.69.5.16 void emberCommandClearBuffer (void)

6.69.5.17 void emberCommandReaderInit (void)

Initialize the command interpreter.

6.69.5.18 bool emberProcessCommandString (uint8_t * *input*, uint8_t *sizeOrPort*)

Process the given string as a command.

6.69.6 Variable Documentation

6.69.6.1 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.

6.69.6.2 EmberCommandEntry emberCommandTable[]

6.69.6.3 uint8_t emberCommandInterpreter2Configuration

Configuration byte.

6.70 Adc

Macros

- #define NUM_ADC_USERS
- #define ADC_MUX_ADC0
- #define ADC_MUX_ADC1
- #define ADC_MUX_ADC2
- #define ADC_MUX_ADC3
- #define ADC_MUX_ADC4
- #define ADC_MUX_ADC5
- #define ADC_MUX_GND
- #define ADC_MUX_VREF2
- #define ADC_MUX_VREF
- #define ADC_MUX_VREG2

Typedefs

- typedef uint8_t ADCChannelType
- typedef uint8_t ADCReferenceType
- typedef uint8_t ADCRateType

Enumerations

- enum ADCUser { ADC_USER_LQI, ADC_USER_APP, ADC_USER_APP2 }
- enum { ADC_REF_INT }
- enum {
 ADC_SAMPLE_CLOCKS_32, ADC_SAMPLE_CLOCKS_64, ADC_SAMPLE_CLOCKS_128, ADC_SAMPLE_CLOCKS_256,
 ADC_SAMPLE_CLOCKS_512, ADC_SAMPLE_CLOCKS_1024, ADC_SAMPLE_CLOCKS_2048, ADC_SAMPLE_CLOCKS_4096
 }
- enum {
 ADC_CONVERSION_TIME_US_32, ADC_CONVERSION_TIME_US_64, ADC_CONVERSION_TIME_US_128, ADC_CONVERSION_TIME_US_256,
 ADC_CONVERSION_TIME_US_512, ADC_CONVERSION_TIME_US_1024, ADC_CONVERSION_TIME_US_2048, ADC_CONVERSION_TIME_US_4096
 }
- enum {
 ADC_SOURCE_ADC0_VREF2, ADC_SOURCE_ADC0_GND, ADC_SOURCE_ADC1_VREF2, ADC_SOURCE_ADC1_GND,
 ADC_SOURCE_ADC2_VREF2, ADC_SOURCE_ADC2_GND, ADC_SOURCE_ADC3_VREF2, ADC_SOURCE_ADC3_GND,
 ADC_SOURCE_ADC4_VREF2, ADC_SOURCE_ADC4_GND, ADC_SOURCE_ADC5_VREF2, ADC_SOURCE_ADC5_GND,
 ADC_SOURCE_ADC1_ADC0, ADC_SOURCE_ADC0_ADC1, ADC_SOURCE_ADC3_ADC2, ADC_SOURCE_ADC2_ADC3,
 ADC_SOURCE_ADC5_ADC4, ADC_SOURCE_GND_VREF2, ADC_SOURCE_VGND, ADC_SOURCE_VREF_VREF2,
 ADC_SOURCE_VREF, ADC_SOURCE_VREG2_VREF2, ADC_SOURCE_VDD_GND
 }

Functions

- void `halInternalInitAdc` (void)
- void `halInternalSleepAdc` (void)
- `EmberStatus halStartAdcConversion (ADCUser id, ADCReferenceType reference, ADCChannelType channel, ADCRateType rate)`
- `EmberStatus halRequestAdcData (ADCUser id, uint16_t *value)`
- `EmberStatus halReadAdcBlocking (ADCUser id, uint16_t *value)`
- `EmberStatus halAdcCalibrate (ADCUser id)`
- `int32_t halConvertValueToVolts (uint16_t value)`
- void `emberCalibrateVref` (void)
- void `halAdcSetClock` (bool slow)
- bool `halAdcGetClock` (void)
- `uint16_t halMeasureVdd (ADCRateType rate)`

6.70.1 Detailed Description

Sample A/D converter driver.

See `adc.h` for source code.

Note

EM35x ADC driver support is preliminary and may change in a future release.
The EmberZNet stack does use these functions.

To use the ADC system, include this file and ensure that `halInternalInitAdc()` is called whenever the microcontroller is started. Call `halInternalSleepAdc()` to sleep the module and `halInternalInitAdc()` to wake up the module.

A "user" is a separate thread of execution and usage. That is, internal Ember code is one user and clients are a different user. But a client that is calling the ADC in two different functions constitutes only one user, as long as the ADC access is not interleaved.

Note

This code does not allow access to the continuous reading mode of the ADC, which some clients may require.

Many functions in this file return an `EmberStatus` value. See `error-def.h` for definitions of all `EmberStatus` return values.

Sample A/D converter driver.

Note

EM35x ADC driver support is preliminary and may be changed in a future release

6.70.2 Macro Definition Documentation

6.70.2.1 `#define NUM_ADC_USERS`

Be sure to update `NUM_ADC_USERS` if additional users are added to the `ADCUser` list.

Definition at line 69 of file `adc.h`.

6.70.2.2 #define ADC_MUX_ADC0

Channel 0 : ADC0 on PB5

Definition at line [93](#) of file [cortexm3/adc.h](#).

6.70.2.3 #define ADC_MUX_ADC1

Channel 1 : ADC0 on PB6

Definition at line [95](#) of file [cortexm3/adc.h](#).

6.70.2.4 #define ADC_MUX_ADC2

Channel 2 : ADC0 on PB7

Definition at line [97](#) of file [cortexm3/adc.h](#).

6.70.2.5 #define ADC_MUX_ADC3

Channel 3 : ADC0 on PC1

Definition at line [99](#) of file [cortexm3/adc.h](#).

6.70.2.6 #define ADC_MUX_ADC4

Channel 4 : ADC0 on PA4

Definition at line [101](#) of file [cortexm3/adc.h](#).

6.70.2.7 #define ADC_MUX_ADC5

Channel 5 : ADC0 on PA5

Definition at line [103](#) of file [cortexm3/adc.h](#).

6.70.2.8 #define ADC_MUX_GND

Channel 8 : VSS (0V) - not for high voltage range

Definition at line [105](#) of file [cortexm3/adc.h](#).

6.70.2.9 #define ADC_MUX_VREF2

Channel 9 : VREF/2 (0.6V)

Definition at line [107](#) of file [cortexm3/adc.h](#).

6.70.2.10 #define ADC_MUX_VREF

Channel A : VREF (1.2V)

Definition at line [109](#) of file [cortexm3/adc.h](#).

6.70.2.11 #define ADC_MUX_VREG2

Channel B : Regulator/2 (0.9V) - not for high voltage range

Definition at line 111 of file [cortexm3/adc.h](#).

6.70.3 Typedef Documentation

6.70.3.1 typedef uint8_t ADCChannelType

A type for the channel enumeration.

Definition at line 73 of file [adc.h](#).

6.70.3.2 typedef uint8_t ADCReferenceType

A type for the reference voltage enumeration.

Definition at line 77 of file [adc.h](#).

6.70.3.3 typedef uint8_t ADCRateType

A type for the sample rate enumeration.

Definition at line 81 of file [adc.h](#).

6.70.4 Enumeration Type Documentation

6.70.4.1 enum ADCUser

ADC functions employ a user ID to keep different users separate.

Avoid many users because each user requires some amount of state storage.

See Also

[NUM_ADC_USERS](#)

Enumerator:

ADC_USER_LQI LQI User ID.

ADC_USER_APP Application User ID

ADC_USER_APP2 Application User ID

Definition at line 51 of file [adc.h](#).

6.70.4.2 anonymous enum

ADCReferenceType Enumeration. Note: EM35x only supports one reference type: Internal.

Enumerator:

ADC_REF_INT Internal reference.

Definition at line 33 of file [cortexm3/adc.h](#).

6.70.4.3 anonymous enum

ADCRateType enumeration. These rates are specified in the number of clock cycles That a conversion takes. The actual time taken will depend on the selected ADC clock rate. (Default is 6MHz)

Enumerator:

- ADC_SAMPLE_CLOCKS_32*** Rate 32 cycles, 5 effective bits in ADC_DATA[15:11]
- ADC_SAMPLE_CLOCKS_64*** Rate 64 cycles, 6 effective bits in ADC_DATA[15:10]
- ADC_SAMPLE_CLOCKS_128*** Rate 128 cycles, 7 effective bits in ADC_DATA[15:9]
- ADC_SAMPLE_CLOCKS_256*** Rate 256 cycles, 8 effective bits in ADC_DATA[15:8]
- ADC_SAMPLE_CLOCKS_512*** Rate 512 cycles, 9 effective bits in ADC_DATA[15:7]
- ADC_SAMPLE_CLOCKS_1024*** Rate 1024 cycles, 10 effective bits in ADC_DATA[15:6]
- ADC_SAMPLE_CLOCKS_2048*** Rate 2048 cycles, 11 effective bits in ADC_DATA[15:5]
- ADC_SAMPLE_CLOCKS_4096*** Rate 4096 cycles, 12 effective bits in ADC_DATA[15:4]

Definition at line 44 of file [cortexm3/adc.h](#).

6.70.4.4 anonymous enum

ADC rates for compatibility with EM2xx applications. For EM35x only applications, the ADC_SAMPLE_CLOCKS_nnn definitions should be used instead.

Enumerator:

- ADC_CONVERSION_TIME_US_32*** Rate 32 us, 5 effective bits in ADC_DATA[15:11]
- ADC_CONVERSION_TIME_US_64*** Rate 64 us, 6 effective bits in ADC_DATA[15:10]
- ADC_CONVERSION_TIME_US_128*** Rate 128 us, 7 effective bits in ADC_DATA[15:9]
- ADC_CONVERSION_TIME_US_256*** Rate 256 us, 8 effective bits in ADC_DATA[15:8]
- ADC_CONVERSION_TIME_US_512*** Rate 512 us, 9 effective bits in ADC_DATA[15:7]
- ADC_CONVERSION_TIME_US_1024*** Rate 1024 us, 10 effective bits in ADC_DATA[15:6]

ADC_CONVERSION_TIME_US_2048 Rate 2048 us, 11 effective bits in ADC_D-
ATA[15:5]

ADC_CONVERSION_TIME_US_4096 Rate 4096 us, 12 effective bits in ADC_D-
ATA[15:4]

Definition at line 67 of file [cortexm3/adc.h](#).

6.70.4.5 anonymous enum

Enumerator:

```
ADC_SOURCE_ADC0_VREF2
ADC_SOURCE_ADC0_GND
ADC_SOURCE_ADC1_VREF2
ADC_SOURCE_ADC1_GND
ADC_SOURCE_ADC2_VREF2
ADC_SOURCE_ADC2_GND
ADC_SOURCE_ADC3_VREF2
ADC_SOURCE_ADC3_GND
ADC_SOURCE_ADC4_VREF2
ADC_SOURCE_ADC4_GND
ADC_SOURCE_ADC5_VREF2
ADC_SOURCE_ADC5_GND
ADC_SOURCE_ADC1_ADC0
ADC_SOURCE_ADC0_ADC1
ADC_SOURCE_ADC3_ADC2
ADC_SOURCE_ADC2_ADC3
ADC_SOURCE_ADC5_ADC4
ADC_SOURCE_GND_VREF2
ADC_SOURCE_VGND
ADC_SOURCE_VREF_VREF2
ADC_SOURCE_VREF
ADC_SOURCE_VREG2_VREF2
ADC_SOURCE_VDD_GND
```

Definition at line 115 of file [cortexm3/adc.h](#).

6.70.5 Function Documentation

6.70.5.1 void halInternalInitAdc (void)

Initializes and powers-up the ADC. Should also be called to wake from sleep. The ADC is required for EM250 stack operation so this function must be called from halInit.

6.70.5.2 void halInternalSleepAdc(void)

Shuts down the voltage reference and ADC system to minimize power consumption in sleep.

6.70.5.3 EmberStatus halStartAdcConversion(ADCUser *id*, ADCReferenceType *reference*, ADCChannelType *channel*, ADCRateType *rate*)

Starts an ADC conversion for the user specified by *id*.

Application Usage:

The application must set *reference* to the voltage reference desired (see the ADC references enum, fixed at [ADC_REF_INT](#) for the em250), set *channel* to the channel number required (see the ADC channel enum), and set *rate* to reflect the number of bits of accuracy desired (see the ADC rates enum, fixed at [ADC_CONVERSION_TIME_US_256](#) for the Atmega).

Parameters

<i>id</i>	An ADC user.
<i>reference</i>	Voltage reference to use, chosen from enum ADCReferenceType (fixed at ADC_REF_INT for the EM250).
<i>channel</i>	Microprocessor channel number. For EM250 channels, see the EM250 ADC channels enum. For basic, single-ended Atmel channels, see the Atmega single-ended ADC channels enum. For more complex measurements on Atmels (differential and amped channel numbers), see the Atmel datasheet for your micro.
<i>rate</i>	EM250 rate number (see the ADC EM250 rate enum).

Returns

One of the following:

- EMBER_ADC_CONVERSION_DEFERRED if the conversion is still waiting to start.
- EMBER_ADC_CONVERSION_BUSY if the conversion is currently taking place.
- EMBER_ERR_FATAL if a passed parameter is invalid.

6.70.5.4 EmberStatus halRequestAdcData(ADCUser *id*, uint16_t * *value*)

Returns the status of a pending conversion previously started by [halStartAdcConversion\(\)](#). If the conversion is complete, writes the raw register value of the conversion (the unaltered value taken directly from the ADC's data register) into *value*.

Parameters

<i>id</i>	An ADC user.
<i>value</i>	Pointer to an uint16_t to be loaded with the new value. Take note that the Atmel's ADC only generates 8-bit values which are loaded into the lower 8 bits of <i>value</i> .

Returns

One of the following:

- [EMBER_ADC_CONVERSION_DONE](#) if the conversion is complete.
- [EMBER_ADC_CONVERSION_DEFERRED](#) if the conversion is still waiting to start.
- [EMBER_ADC_CONVERSION_BUSY](#) if the conversion is currently taking place.
- [EMBER_ADC_NO_CONVERSION_PENDING](#) if `id` does not have a pending conversion.

6.70.5.5 EmberStatus halReadAdcBlocking (ADCUser *id*, uint16_t * *value*)

Waits for the user's request to complete and then, if a conversion was done, writes the raw register value of the conversion (the unaltered value taken directly from the ADC's data register) into `value` and returns [EMBER_ADC_CONVERSION_DONE](#), or immediately returns [EMBER_ADC_NO_CONVERSION_PENDING](#).

Parameters

<i>id</i>	An ADC user.
<i>value</i>	Pointer to an <code>uint16_t</code> to be loaded with the new value. Take note that the Atmel's ADC only generates 8-bit values which are loaded into the lower 8 bits of <code>value</code> .

Returns

One of the following:

- [EMBER_ADC_CONVERSION_DONE](#) if the conversion is complete.
- [EMBER_ADC_NO_CONVERSION_PENDING](#) if `id` does not have a pending conversion.

6.70.5.6 EmberStatus halAdcCalibrate (ADCUser *id*)

Calibrates or recalibrates the ADC system.

Application Usage:

Use this function to (re)calibrate as needed. This function is intended for the EM250 microcontroller, which requires proper calibration to calculate a human readable value (a value in volts). If the app does not call this function, the first time (and only the first time) the function [halConvertValueToVolts\(\)](#) is called, this function is invoked. To maintain accurate volt calculations, the application should call this whenever it expects the temperature of the micro to change.

Parameters

<i>id</i>	An ADC user.
-----------	--------------

Returns

One of the following:

- [EMBER_ADC_CONVERSION_DONE](#) if the calibration is complete.
- [EMBER_ERR_FATAL](#) if the calibration failed.

6.70.5.7 int32_t halConvertValueToVolts (uint16_t value)

Convert the raw register value (the unaltered value taken directly from the ADC's data register) into a signed fixed point value with units 10^{-4} Volts. The returned value will be in the range -12000 to +12000 (-1.2000 volts to +1.2000 volts).

Application Usage:

Use this function to get a human useful value.

Parameters

<i>value</i>	An uint16_t to be converted.
--------------	------------------------------

Returns

Volts as signed fixed point with units 10^{-4} Volts.

6.70.5.8 void emberCalibrateVref (void)

Calibrates Vref to be 1.2V +/-10mV.

Application Usage:

This function must be called from [halInternalInitAdc\(\)](#) before making ADC readings. This function is not intended to be called from any function other than [halInternalInitAdc\(\)](#). This function ensures that the master cell voltage and current bias values are calibrated before calibrating Vref.

6.70.5.9 void halAdcSetClock (bool slow)

Set ADC clock mode.

Parameters

<i>slow</i>	A bool to select slow or normal clock.
-------------	--

6.70.5.10 bool halAdcGetClock (void)

Get ADC clock mode.

Returns

A true if the slow clock is selected.

6.70.5.11 uint16_t halMeasureVdd (ADCRateType *rate*)

Measures VDD_PADS in millivolts at the specified sample rate Due to the conversions performed, this function takes slightly under 250us with a variation across successive conversions approximately +/-20mv of the average conversion.

Returns

A measurement of VDD_PADS in millivolts.

Chapter 7

Data Structure Documentation

7.1 EmberAesMmoHashContext Struct Reference

```
#include <ember-types.h>
```

Data Fields

- `uint8_t result [EMBER_AES_HASH_BLOCK_SIZE]`
- `uint32_t length`

7.1.1 Detailed Description

This data structure contains the context data when calculating an AES MMO hash (message digest).

Definition at line 1537 of file [ember-types.h](#).

7.1.2 Field Documentation

7.1.2.1 `uint8_t EmberAesMmoHashContext::result[EMBER_AES_HASH_BLOCK_SIZE]`

Definition at line 1538 of file [ember-types.h](#).

7.1.2.2 `uint32_t EmberAesMmoHashContext::length`

Definition at line 1539 of file [ember-types.h](#).

The documentation for this struct was generated from the following file:

- [ember-types.h](#)

7.2 EmberApsFrame Struct Reference

```
#include <ember-types.h>
```

Data Fields

- `uint16_t profileId`
- `uint16_t clusterId`
- `uint8_t sourceEndpoint`
- `uint8_t destinationEndpoint`
- `EmberApsOption options`
- `uint16_t groupId`
- `uint8_t sequence`

7.2.1 Detailed Description

An in-memory representation of a ZigBee APS frame of an incoming or outgoing message.

Definition at line [960](#) of file `ember-types.h`.

7.2.2 Field Documentation

7.2.2.1 `uint16_t EmberApsFrame::profileId`

The application profile ID that describes the format of the message.

Definition at line [962](#) of file `ember-types.h`.

7.2.2.2 `uint16_t EmberApsFrame::clusterId`

The cluster ID for this message.

Definition at line [964](#) of file `ember-types.h`.

7.2.2.3 `uint8_t EmberApsFrame::sourceEndpoint`

The source endpoint.

Definition at line [966](#) of file `ember-types.h`.

7.2.2.4 `uint8_t EmberApsFrame::destinationEndpoint`

The destination endpoint.

Definition at line [968](#) of file `ember-types.h`.

7.2.2.5 `EmberApsOption EmberApsFrame::options`

A bitmask of options from the enumeration above.

Definition at line [970](#) of file `ember-types.h`.

7.2.2.6 `uint16_t EmberApsFrame::groupId`

The group ID for this message, if it is multicast mode.

Definition at line 972 of file [ember-types.h](#).

7.2.2.7 `uint8_t EmberApsFrame::sequence`

The sequence number.

Definition at line 974 of file [ember-types.h](#).

The documentation for this struct was generated from the following file:

- [ember-types.h](#)

7.3 EmberBindingTableEntry Struct Reference

```
#include <ember-types.h>
```

Data Fields

- [EmberBindingType type](#)
- `uint8_t local`
- `uint16_t clusterId`
- `uint8_t remote`
- [EmberEUI64 identifier](#)
- `uint8_t networkIndex`

7.3.1 Detailed Description

Defines an entry in the binding table.

A binding entry specifies a local endpoint, a remote endpoint, a cluster ID and either the destination EUI64 (for unicast bindings) or the 64-bit group address (for multicast bindings).

Definition at line 984 of file [ember-types.h](#).

7.3.2 Field Documentation

7.3.2.1 `EmberBindingType EmberBindingTableEntry::type`

The type of binding.

Definition at line 986 of file [ember-types.h](#).

7.3.2.2 `uint8_t EmberBindingTableEntry::local`

The endpoint on the local node.

Definition at line 988 of file [ember-types.h](#).

7.3.2.3 `uint16_t EmberBindingTableEntry::clusterId`

A cluster ID that matches one from the local endpoint's simple descriptor. This cluster ID is set by the provisioning application to indicate which part an endpoint's functionality is bound to this particular remote node and is used to distinguish between unicast and multicast bindings. Note that a binding can be used to send messages with any cluster ID, not just that listed in the binding.

Definition at line 996 of file [ember-types.h](#).

7.3.2.4 `uint8_t EmberBindingTableEntry::remote`

The endpoint on the remote node (specified by `identifier`).

Definition at line 998 of file [ember-types.h](#).

7.3.2.5 `EmberEUI64 EmberBindingTableEntry::identifier`

A 64-bit identifier. This is either:

- The destination EUI64, for unicasts
- A 16-bit multicast group address, for multicasts

Definition at line 1003 of file [ember-types.h](#).

7.3.2.6 `uint8_t EmberBindingTableEntry::networkIndex`

The index of the network the binding belongs to.

Definition at line 1005 of file [ember-types.h](#).

The documentation for this struct was generated from the following file:

- [ember-types.h](#)

7.4 EmberCertificate283k1Data Struct Reference

```
#include <ember-types.h>
```

Data Fields

- `uint8_t contents [EMBER_CERTIFICATE_283K1_SIZE]`

7.4.1 Detailed Description

This data structure contains the certificate data that is used for Certificate Based Key Exchange (CBKE) in SECT283k1 Elliptical Cryptography.

Definition at line 1544 of file [ember-types.h](#).

7.4.2 Field Documentation

7.4.2.1 uint8_t EmberCertificate283k1Data::contents[EMBER_CERTIFICATE_283K1_SIZE]

Definition at line 1546 of file [ember-types.h](#).

The documentation for this struct was generated from the following file:

- [ember-types.h](#)

7.5 EmberCertificateData Struct Reference

```
#include <ember-types.h>
```

Data Fields

- [uint8_t contents \[EMBER_CERTIFICATE_SIZE\]](#)

7.5.1 Detailed Description

This data structure contains the certificate data that is used for Certificate Based Key Exchange (CBKE).

Definition at line 1499 of file [ember-types.h](#).

7.5.2 Field Documentation

7.5.2.1 uint8_t EmberCertificateData::contents[EMBER_CERTIFICATE_SIZE]

Definition at line 1500 of file [ember-types.h](#).

The documentation for this struct was generated from the following file:

- [ember-types.h](#)

7.6 EmberCommandEntry Struct Reference

```
#include <command-interpreter2.h>
```

Data Fields

- [PGM_P name](#)
- [CommandAction action](#)
- [PGM_P argumentTypes](#)
- [PGM_P description](#)
- [PGM_P const * argumentDescriptions](#)

7.6.1 Detailed Description

Command entry for a command table.

Definition at line 129 of file [command-interpreter2.h](#).

7.6.2 Field Documentation

7.6.2.1 PGM_P EmberCommandEntry::name

Use letters, digits, and underscores, '_', for the command name. Command names are case-sensitive.

Definition at line 136 of file [command-interpreter2.h](#).

7.6.2.2 CommandAction EmberCommandEntry::action

A reference to a function in the application that implements the command. If this entry refers to a nested command, then action field has to be set to NULL.

Definition at line 142 of file [command-interpreter2.h](#).

7.6.2.3 PGM_P EmberCommandEntry::argumentTypes

In case of normal (non-nested) commands, argumentTypes is a string that specifies the number and types of arguments the command accepts. The argument specifiers are:

- u: one-byte unsigned integer.
- v: two-byte unsigned integer
- w: four-byte unsigned integer
- s: one-byte signed integer
- r: two-byte signed integer
- q: four-byte signed integer
- b: string. The argument can be entered in ascii by using quotes, for example: "foo". Or it may be entered in hex by using curly braces, for example: { 08 A1 f2 }. There must be an even number of hex digits, and spaces are ignored.
- *: zero or more of the previous type. If used, this must be the last specifier.
- ?: Unknown number of arguments. If used this must be the only character. This means, that command interpreter will not perform any validation of arguments, and will call the action directly, trusting it that it will handle with whatever arguments are passed in. Integer arguments can be either decimal or hexadecimal. A 0x prefix indicates a hexadecimal integer. Example: 0x3ed.

In case of a nested command (action is NULL), then this field contains a pointer to the nested [EmberCommandEntry](#) array.

Definition at line 171 of file [command-interpreter2.h](#).

7.6.2.4 PGM_P EmberCommandEntry::description

A description of the command.

Definition at line 176 of file [command-interpreter2.h](#).

7.6.2.5 PGM_P const* EmberCommandEntry::argumentDescriptions

An array of strings, with a NULL terminator, indicating what each argument is.

Definition at line 180 of file [command-interpreter2.h](#).

The documentation for this struct was generated from the following file:

- [command-interpreter2.h](#)

7.7 EmberCurrentSecurityState Struct Reference

```
#include <ember-types.h>
```

Data Fields

- [EmberCurrentSecurityBitmask bitmask](#)
- [EmberEUI64 trustCenterLongAddress](#)

7.7.1 Detailed Description

This describes the security features used by the stack for a joined device.

Definition at line 1828 of file [ember-types.h](#).

7.7.2 Field Documentation

7.7.2.1 EmberCurrentSecurityBitmask EmberCurrentSecurityState::bitmask

This bitmask indicates the security features currently in use on this node.

Definition at line 1831 of file [ember-types.h](#).

7.7.2.2 EmberEUI64 EmberCurrentSecurityState::trustCenterLongAddress

This indicates the EUI64 of the Trust Center. It will be all zeroes if the Trust Center Address is not known (i.e. the device is in a Distributed Trust Center network).

Definition at line 1835 of file [ember-types.h](#).

The documentation for this struct was generated from the following file:

- [ember-types.h](#)

7.8 EmberEndpoint Struct Reference

```
#include <stack-info.h>
```

Data Fields

- `uint8_t endpoint`
- `EmberEndpointDescription PGM * description`
- `uint16_t PGM * inputClusterList`
- `uint16_t PGM * outputClusterList`

7.8.1 Detailed Description

Gives the endpoint information for a particular endpoint.

Definition at line 286 of file [stack-info.h](#).

7.8.2 Field Documentation

7.8.2.1 `uint8_t EmberEndpoint::endpoint`

An endpoint of the application on this node.

Definition at line 288 of file [stack-info.h](#).

7.8.2.2 `EmberEndpointDescription PGM* EmberEndpoint::description`

The endpoint's description.

Definition at line 290 of file [stack-info.h](#).

7.8.2.3 `uint16_t PGM* EmberEndpoint::inputClusterList`

Input clusters the endpoint will accept.

Definition at line 292 of file [stack-info.h](#).

7.8.2.4 `uint16_t PGM* EmberEndpoint::outputClusterList`

Output clusters the endpoint may send.

Definition at line 294 of file [stack-info.h](#).

The documentation for this struct was generated from the following file:

- [stack-info.h](#)

7.9 EmberEndpointDescription Struct Reference

```
#include <stack-info.h>
```

Data Fields

- `uint16_t profileId`
- `uint16_t deviceId`
- `uint8_t deviceVersion`
- `uint8_t inputClusterCount`
- `uint8_t outputClusterCount`

7.9.1 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 270 of file [stack-info.h](#).

7.9.2 Field Documentation

7.9.2.1 `uint16_t EmberEndpointDescription::profileId`

Identifies the endpoint's application profile.

Definition at line 272 of file [stack-info.h](#).

7.9.2.2 `uint16_t EmberEndpointDescription::deviceId`

The endpoint's device ID within the application profile.

Definition at line 274 of file [stack-info.h](#).

7.9.2.3 `uint8_t EmberEndpointDescription::deviceVersion`

The endpoint's device version.

Definition at line 276 of file [stack-info.h](#).

7.9.2.4 `uint8_t EmberEndpointDescription::inputClusterCount`

The number of input clusters.

Definition at line 278 of file [stack-info.h](#).

7.9.2.5 `uint8_t EmberEndpointDescription::outputClusterCount`

The number of output clusters.

Definition at line 280 of file [stack-info.h](#).

The documentation for this struct was generated from the following file:

- [stack-info.h](#)

7.10 EmberEventControl Struct Reference

```
#include <ember-types.h>
```

Data Fields

- [EmberEventUnits status](#)
- [EmberTaskId taskid](#)
- [uint32_t timeToExecute](#)

7.10.1 Detailed Description

Control structure for events.

This structure should not be accessed directly. This holds the event status (one of the *EMBER_EVENT_* values) and the time left before the event fires.

Definition at line [1272](#) of file [ember-types.h](#).

7.10.2 Field Documentation

7.10.2.1 EmberEventUnits EmberEventControl::status

The event's status, either inactive or the units for timeToExecute.

Definition at line [1274](#) of file [ember-types.h](#).

7.10.2.2 EmberTaskId EmberEventControl::taskid

The id of the task this event belongs to.

Definition at line [1276](#) of file [ember-types.h](#).

7.10.2.3 uint32_t EmberEventControl::timeToExecute

How long before the event fires. Units are always in milliseconds

Definition at line [1280](#) of file [ember-types.h](#).

The documentation for this struct was generated from the following file:

- [ember-types.h](#)

7.11 EmberEventData_S Struct Reference

```
#include <ember-types.h>
```

Data Fields

- [EmberEventControl * control](#)
- [void\(* handler \)\(void\)](#)

7.11.1 Detailed Description

Complete events with a control and a handler procedure.

An application typically creates an array of events along with their handlers. The main loop passes the array to [emberRunEvents\(\)](#) in order to call the handlers of any events whose time has arrived.

Definition at line 1290 of file [ember-types.h](#).

7.11.2 Field Documentation

7.11.2.1 EmberEventControl* EmberEventData_S::control

The control structure for the event.

Definition at line 1292 of file [ember-types.h](#).

7.11.2.2 void(* EmberEventData_S::handler)(void)

The procedure to call when the event fires.

Definition at line 1294 of file [ember-types.h](#).

The documentation for this struct was generated from the following file:

- [ember-types.h](#)

7.12 EmberInitialSecurityState Struct Reference

```
#include <ember-types.h>
```

Data Fields

- [uint16_t bitmask](#)
- [EmberKeyData preconfiguredKey](#)
- [EmberKeyData networkKey](#)
- [uint8_t networkKeySequenceNumber](#)
- [EmberEUI64 preconfiguredTrustCenterEui64](#)

7.12.1 Detailed Description

This describes the Initial Security features and requirements that will be used when forming or joining the network.

Definition at line 1748 of file [ember-types.h](#).

7.12.2 Field Documentation

7.12.2.1 `uint16_t EmberInitialSecurityState::bitmask`

This bitmask enumerates which security features should be used, as well as the presence of valid data within other elements of the `EmberInitialSecurityState` data structure. For more details see the `EmberInitialSecurityBitmask`.

Definition at line 1753 of file `ember-types.h`.

7.12.2.2 `EmberKeyData EmberInitialSecurityState::preconfiguredKey`

This is the pre-configured key that can be used by devices when joining the network if the Trust Center does not send the initial security data in-the-clear. For the Trust Center, it will be the global link key and **must** be set regardless of whether joining devices are expected to have a pre-configured Link Key. This parameter will only be used if the `EmberInitialSecurityState::bitmask` sets the bit indicating `EMBER_HAVE_PRECONFIGURED_KEY`.

Definition at line 1762 of file `ember-types.h`.

7.12.2.3 `EmberKeyData EmberInitialSecurityState::networkKey`

This is the Network Key used when initially forming the network. This must be set on the Trust Center. It is not needed for devices joining the network. This parameter will only be used if the `EmberInitialSecurityState::bitmask` sets the bit indicating `EMBER_HAVE_NETWORK_KEY`.

Definition at line 1768 of file `ember-types.h`.

7.12.2.4 `uint8_t EmberInitialSecurityState::networkKeySequenceNumber`

This is the sequence number associated with the network key. It must be set if the Network Key is set. It is used to indicate a particular of the network key for updating and switching. This parameter will only be used if the `EMBER_HAVE_NETWORK_KEY` is set. Generally it should be set to 0 when forming the network; joining devices can ignore this value.

Definition at line 1775 of file `ember-types.h`.

7.12.2.5 `EmberEUI64 EmberInitialSecurityState::preconfiguredTrustCenterEui64`

This is the long address of the trust center on the network that will be joined. It is usually NOT set prior to joining the network and instead it is learned during the joining message exchange. This field is only examined if `EMBER_HAVE_TRUST_CENTER_EUI64` is set in the `EmberInitialSecurityState::bitmask`. Most devices should clear that bit and leave this field alone. This field must be set when using commissioning mode. It is required to be in little-endian format.

Definition at line 1783 of file `ember-types.h`.

The documentation for this struct was generated from the following file:

- `ember-types.h`

7.13 EmberKeyData Struct Reference

```
#include <ember-types.h>
```

Data Fields

- `uint8_t contents [EMBER_ENCRYPTION_KEY_SIZE]`

7.13.1 Detailed Description

This data structure contains the key data that is passed into various other functions.

Definition at line 1492 of file `ember-types.h`.

7.13.2 Field Documentation

7.13.2.1 `uint8_t EmberKeyData::contents[EMBER_ENCRYPTION_KEY_SIZE]`

This is the key byte data.

Definition at line 1494 of file `ember-types.h`.

The documentation for this struct was generated from the following file:

- `ember-types.h`

7.14 EmberKeyStruct Struct Reference

```
#include <ember-types.h>
```

Data Fields

- `EmberKeyStructBitmask bitmask`
- `EmberKeyType type`
- `EmberKeyData key`
- `uint32_t outgoingFrameCounter`
- `uint32_t incomingFrameCounter`
- `uint8_t sequenceNumber`
- `EmberEUI64 partnerEUI64`

7.14.1 Detailed Description

This describes a one of several different types of keys and its associated data.

Definition at line 1901 of file `ember-types.h`.

7.14.2 Field Documentation

7.14.2.1 EmberKeyStructBitmask EmberKeyStruct::bitmask

This bitmask indicates whether various fields in the structure contain valid data. It also contains the index of the network the key belongs to.

Definition at line 1905 of file [ember-types.h](#).

7.14.2.2 EmberKeyType EmberKeyStruct::type

This indicates the type of the security key.

Definition at line 1907 of file [ember-types.h](#).

7.14.2.3 EmberKeyData EmberKeyStruct::key

This is the actual key data.

Definition at line 1909 of file [ember-types.h](#).

7.14.2.4 uint32_t EmberKeyStruct::outgoingFrameCounter

This is the outgoing frame counter associated with the key. It will contain valid data based on the [EmberKeyStructBitmask](#).

Definition at line 1912 of file [ember-types.h](#).

7.14.2.5 uint32_t EmberKeyStruct::incomingFrameCounter

This is the incoming frame counter associated with the key. It will contain valid data based on the [EmberKeyStructBitmask](#).

Definition at line 1915 of file [ember-types.h](#).

7.14.2.6 uint8_t EmberKeyStruct::sequenceNumber

This is the sequence number associated with the key. It will contain valid data based on the [EmberKeyStructBitmask](#).

Definition at line 1918 of file [ember-types.h](#).

7.14.2.7 EmberEUI64 EmberKeyStruct::partnerEUI64

This is the Partner EUI64 associated with the key. It will contain valid data based on the [EmberKeyStructBitmask](#).

Definition at line 1921 of file [ember-types.h](#).

The documentation for this struct was generated from the following file:

- [ember-types.h](#)

7.15 EmberMacFilterMatchStruct Struct Reference

```
#include <ember-types.h>
```

Data Fields

- `uint8_t filterIndexMatch`
- `EmberMacPassthroughType legacyPassthroughType`
- `EmberMessageBuffer message`

7.15.1 Detailed Description

This structure indicates a matching raw MAC message has been received by the application configured MAC filters.

Definition at line [2214](#) of file `ember-types.h`.

7.15.2 Field Documentation

7.15.2.1 `uint8_t EmberMacFilterMatchStruct::filterIndexMatch`

Definition at line [2215](#) of file `ember-types.h`.

7.15.2.2 `EmberMacPassthroughType EmberMacFilterMatchStruct::legacyPassthroughType`

Definition at line [2216](#) of file `ember-types.h`.

7.15.2.3 `EmberMessageBuffer EmberMacFilterMatchStruct::message`

Definition at line [2217](#) of file `ember-types.h`.

The documentation for this struct was generated from the following file:

- `ember-types.h`

7.16 EmberMessageDigest Struct Reference

```
#include <ember-types.h>
```

Data Fields

- `uint8_t contents [EMBER_AES_HASH_BLOCK_SIZE]`

7.16.1 Detailed Description

This data structure contains an AES-MMO Hash (the message digest).

Definition at line 1530 of file [ember-types.h](#).

7.16.2 Field Documentation

7.16.2.1 uint8_t EmberMessageDigest::contents[EMBER_AES_HASH_BLOCK_SIZE]

Definition at line 1531 of file [ember-types.h](#).

The documentation for this struct was generated from the following file:

- [ember-types.h](#)

7.17 EmberMfgSecurityStruct Struct Reference

```
#include <ember-types.h>
```

Data Fields

- [EmberKeySettings keySettings](#)

7.17.1 Detailed Description

This structure is used to get/set the security config that is stored in manufacturing tokens.

Definition at line 2129 of file [ember-types.h](#).

7.17.2 Field Documentation

7.17.2.1 EmberKeySettings EmberMfgSecurityStruct::keySettings

Definition at line 2130 of file [ember-types.h](#).

The documentation for this struct was generated from the following file:

- [ember-types.h](#)

7.18 EmberMulticastTableEntry Struct Reference

```
#include <ember-types.h>
```

Data Fields

- [EmberMulticastId multicastId](#)
- [uint8_t endpoint](#)
- [uint8_t networkIndex](#)

7.18.1 Detailed Description

Defines an entry in the multicast table.

A multicast table entry indicates that a particular endpoint is a member of a particular multicast group. Only devices with an endpoint in a multicast group will receive messages sent to that multicast group.

Definition at line 1073 of file [ember-types.h](#).

7.18.2 Field Documentation

7.18.2.1 EmberMulticastId EmberMulticastTableEntry::multicastId

The multicast group ID.

Definition at line 1075 of file [ember-types.h](#).

7.18.2.2 uint8_t EmberMulticastTableEntry::endpoint

The endpoint that is a member, or 0 if this entry is not in use (the ZDO is not a member of any multicast groups).

Definition at line 1079 of file [ember-types.h](#).

7.18.2.3 uint8_t EmberMulticastTableEntry::networkIndex

The network index of the network the entry is related to.

Definition at line 1081 of file [ember-types.h](#).

The documentation for this struct was generated from the following file:

- [ember-types.h](#)

7.19 EmberNeighborTableEntry Struct Reference

```
#include <ember-types.h>
```

Data Fields

- `uint16_t shortId`
- `uint8_t averageLqi`
- `uint8_t inCost`
- `uint8_t outCost`
- `uint8_t age`
- `EmberEUI64 longId`

7.19.1 Detailed Description

Defines an entry in the neighbor table.

A neighbor table entry stores information about the reliability of RF links to and from neighboring nodes.

Definition at line 1014 of file [ember-types.h](#).

7.19.2 Field Documentation

7.19.2.1 uint16_t EmberNeighborTableEntry::shortId

The neighbor's two byte network id.

Definition at line 1016 of file [ember-types.h](#).

7.19.2.2 uint8_t EmberNeighborTableEntry::averageLqi

An exponentially weighted moving average of the link quality values of incoming packets from this neighbor as reported by the PHY.

Definition at line 1019 of file [ember-types.h](#).

7.19.2.3 uint8_t EmberNeighborTableEntry::inCost

The incoming cost for this neighbor, computed from the average LQI. Values range from 1 for a good link to 7 for a bad link.

Definition at line 1022 of file [ember-types.h](#).

7.19.2.4 uint8_t EmberNeighborTableEntry::outCost

The outgoing cost for this neighbor, obtained from the most recently received neighbor exchange message from the neighbor. A value of zero means that a neighbor exchange message from the neighbor has not been received recently enough, or that our id was not present in the most recently received one. EmberZNet Pro only.

Definition at line 1029 of file [ember-types.h](#).

7.19.2.5 uint8_t EmberNeighborTableEntry::age

In EmberZNet Pro, the number of aging periods elapsed since a neighbor exchange message was last received from this neighbor. In stack profile 1, the number of aging periods since any packet was received. An entry with an age greater than 3 is considered stale and may be reclaimed. The aging period is 16 seconds.

Definition at line 1035 of file [ember-types.h](#).

7.19.2.6 EmberEUI64 EmberNeighborTableEntry::longId

The 8 byte EUI64 of the neighbor.

Definition at line 1037 of file [ember-types.h](#).

The documentation for this struct was generated from the following file:

- [ember-types.h](#)

7.20 EmberNetworkInitStruct Struct Reference

```
#include <ember-types.h>
```

Data Fields

- [EmberNetworkInitBitmask bitmask](#)

7.20.1 Detailed Description

Defines the network initialization configuration that should be used when [emberNetworkInitExtended\(\)](#) is called by the application.

Definition at line 474 of file [ember-types.h](#).

7.20.2 Field Documentation

7.20.2.1 EmberNetworkInitBitmask EmberNetworkInitStruct::bitmask

Definition at line 475 of file [ember-types.h](#).

The documentation for this struct was generated from the following file:

- [ember-types.h](#)

7.21 EmberNetworkParameters Struct Reference

```
#include <ember-types.h>
```

Data Fields

- [uint8_t extendedPanId \[8\]](#)
- [uint16_t panId](#)
- [int8_t radioTxPower](#)
- [uint8_t radioChannel](#)
- [EmberJoinMethod joinMethod](#)
- [EmberNodeId nwkManagerId](#)
- [uint8_t nwkUpdateId](#)
- [uint32_t channels](#)

7.21.1 Detailed Description

Holds network parameters.

For information about power settings and radio channels, see the technical specification for the RF communication module in your Developer Kit.

Definition at line [915](#) of file [ember-types.h](#).

7.21.2 Field Documentation

7.21.2.1 `uint8_t EmberNetworkParameters::extendedPanId[8]`

The network's extended PAN identifier.

Definition at line [917](#) of file [ember-types.h](#).

7.21.2.2 `uint16_t EmberNetworkParameters::panId`

The network's PAN identifier.

Definition at line [919](#) of file [ember-types.h](#).

7.21.2.3 `int8_t EmberNetworkParameters::radioTxPower`

A power setting, in dBm.

Definition at line [921](#) of file [ember-types.h](#).

7.21.2.4 `uint8_t EmberNetworkParameters::radioChannel`

A radio channel. Be sure to specify a channel supported by the radio.

Definition at line [923](#) of file [ember-types.h](#).

7.21.2.5 `EmberJoinMethod EmberNetworkParameters::joinMethod`

Join method: The protocol messages used to establish an initial parent. It is ignored when forming a ZigBee network, or when querying the stack for its network parameters.

Definition at line [928](#) of file [ember-types.h](#).

7.21.2.6 `EmberNodeId EmberNetworkParameters::nwkManagerId`

NWK Manager ID. The ID of the network manager in the current network. This may only be set at joining when using EMBER_USE_NWK_COMMISSIONING as the join method.

Definition at line [934](#) of file [ember-types.h](#).

7.21.2.7 `uint8_t EmberNetworkParameters::nwkUpdateId`

NWK Update ID. The value of the ZigBee nwkUpdateId known by the stack. This is used to determine the newest instance of the network after a PAN ID or channel change. This may only be set at joining when using EMBER_USE_NWK_COMMISSIONING as the join method.

Definition at line 940 of file [ember-types.h](#).

7.21.2.8 `uint32_t EmberNetworkParameters::channels`

NWK channel mask. The list of preferred channels that the NWK manager has told this device to use when searching for the network. This may only be set at joining when using EMBER_USE_NWK_COMMISSIONING as the join method.

Definition at line 946 of file [ember-types.h](#).

The documentation for this struct was generated from the following file:

- [ember-types.h](#)

7.22 `EmberPrivateKey283k1Data` Struct Reference

```
#include <ember-types.h>
```

Data Fields

- `uint8_t contents [EMBER_PRIVATE_KEY_283K1_SIZE]`

7.22.1 Detailed Description

This data structure contains the private key data that is used for Certificate Based Key Exchange (CBKE) in SECT283k1 Elliptical Cryptography.

Definition at line 1557 of file [ember-types.h](#).

7.22.2 Field Documentation

7.22.2.1 `uint8_t EmberPrivateKey283k1Data::contents[EMBER_PRIVATE_KEY_283-K1_SIZE]`

Definition at line 1558 of file [ember-types.h](#).

The documentation for this struct was generated from the following file:

- [ember-types.h](#)

7.23 `EmberPrivateKeyData` Struct Reference

```
#include <ember-types.h>
```

Data Fields

- `uint8_t contents [EMBER_PRIVATE_KEY_SIZE]`

7.23.1 Detailed Description

This data structure contains the private key data that is used for Certificate Based Key Exchange (CBKE).

Definition at line 1511 of file [ember-types.h](#).

7.23.2 Field Documentation

7.23.2.1 `uint8_t EmberPrivateKeyData::contents[EMBER_PRIVATE_KEY_SIZE]`

Definition at line 1512 of file [ember-types.h](#).

The documentation for this struct was generated from the following file:

- [ember-types.h](#)

7.24 EmberPublicKey283k1Data Struct Reference

```
#include <ember-types.h>
```

Data Fields

- `uint8_t contents [EMBER_PUBLIC_KEY_283K1_SIZE]`

7.24.1 Detailed Description

This data structure contains the public key data that is used for Certificate Based Key Exchange (CBKE) in SECT283k1 Elliptical Cryptography.

Definition at line 1551 of file [ember-types.h](#).

7.24.2 Field Documentation

7.24.2.1 `uint8_t EmberPublicKey283k1Data::contents[EMBER_PUBLIC_KEY_283K1_SIZE]`

Definition at line 1552 of file [ember-types.h](#).

The documentation for this struct was generated from the following file:

- [ember-types.h](#)

7.25 EmberPublicKeyData Struct Reference

```
#include <ember-types.h>
```

Data Fields

- uint8_t contents [EMBER_PUBLIC_KEY_SIZE]

7.25.1 Detailed Description

This data structure contains the public key data that is used for Certificate Based Key Exchange (CBKE).

Definition at line 1505 of file [ember-types.h](#).

7.25.2 Field Documentation

7.25.2.1 uint8_t EmberPublicKeyData::contents[EMBER_PUBLIC_KEY_SIZE]

Definition at line 1506 of file [ember-types.h](#).

The documentation for this struct was generated from the following file:

- [ember-types.h](#)

7.26 EmberReleaseTypeStruct Struct Reference

```
#include <ember-types.h>
```

Data Fields

- [EmberVersionType](#) typeNum
- PGM_P typeString

7.26.1 Detailed Description

A structure relating version types to human readable strings.

Definition at line 67 of file [ember-types.h](#).

7.26.2 Field Documentation

7.26.2.1 EmberVersionType EmberReleaseTypeStruct::typeNum

Definition at line 68 of file [ember-types.h](#).

7.26.2.2 PGM_P EmberReleaseTypeStruct::typeString

Definition at line 69 of file [ember-types.h](#).

The documentation for this struct was generated from the following file:

- [ember-types.h](#)

7.27 EmberRf4ceApplicationInfo Struct Reference

```
#include <rf4ce-types.h>
```

Data Fields

- [EmberRf4ceApplicationCapabilities](#) capabilities
- [uint8_t userString \[EMBER_RF4CE_APPLICATION_USER_STRING_LENGTH\]](#)
- [uint8_t deviceTypeList \[EMBER_RF4CE_APPLICATION_DEVICE_TYPE_LIST_MAX_LENGTH\]](#)
- [uint8_t profileIdList \[EMBER_RF4CE_APPLICATION_PROFILE_ID_LIST_MAX_LENGTH\]](#)

7.27.1 Detailed Description

Defines the application information block (see section 3.3.1, Figure 17).

Definition at line 242 of file [rf4ce-types.h](#).

7.27.2 Field Documentation

7.27.2.1 EmberRf4ceApplicationCapabilities EmberRf4ceApplicationInfo::capabilities

The application capabilities field shall contain information relating to the capabilities of the application of the node.

Definition at line 247 of file [rf4ce-types.h](#).

7.27.2.2 uint8_t EmberRf4ceApplicationInfo::userString[EMBER_RF4CE_APPLICATION_USER_STRING_LENGTH]

The user string field shall contain the user specified identification string.

Definition at line 252 of file [rf4ce-types.h](#).

7.27.2.3 uint8_t EmberRf4ceApplicationInfo::deviceTypeList[EMBER_RF4CE_APPLICATION_DEVICE_TYPE_LIST_MAX_LENGTH]

The device type list field shall contain the list of device types supported by the node.

Definition at line 257 of file [rf4ce-types.h](#).

7.27.2.4 uint8_t EmberRf4ceApplicationInfo::profileIdList[EMBER_RF4CE_APPLICATION_PROFILE_ID_LIST_MAX_LENGTH]

The profile ID list field shall contain the list of profile identifiers disclosed as supported by the node.

Definition at line 262 of file [rf4ce-types.h](#).

The documentation for this struct was generated from the following file:

- [rf4ce-types.h](#)

7.28 EmberRf4cePairingTableEntry Struct Reference

```
#include <rf4ce-types.h>
```

Data Fields

- [EmberKeyData securityLinkKey](#)
- [EmberEUI64 destLongId](#)
- [uint32_t frameCounter](#)
- [EmberNodeId sourceNodeId](#)
- [EmberPanId destPanId](#)
- [EmberNodeId destNodeId](#)
- [uint16_t destVendorId](#)
- [uint8_t destProfileIdList \[EMBER_RF4CE_APPLICATION_PROFILE_ID_LIST_MAX_LENGTH\]](#)
- [uint8_t destProfileIdListLength](#)
- [uint8_t info](#)
- [uint8_t channel](#)
- [uint8_t capabilities](#)
- [uint8_t lastSeqn](#)

7.28.1 Detailed Description

The internal representation of a pairing table entry.

Definition at line 309 of file [rf4ce-types.h](#).

7.28.2 Field Documentation

7.28.2.1 EmberKeyData EmberRf4cePairingTableEntry::securityLinkKey

The link key to be used to secure this pairing link.

Definition at line 313 of file [rf4ce-types.h](#).

7.28.2.2 EmberEUI64 EmberRf4cePairingTableEntry::destLongId

The IEEE address of the destination device.

Definition at line 317 of file [rf4ce-types.h](#).

7.28.2.3 uint32_t EmberRf4cePairingTableEntry::frameCounter

The frame counter last received from the recipient node.

Definition at line 321 of file [rf4ce-types.h](#).

7.28.2.4 EmberNodeId EmberRf4cePairingTableEntry::sourceNodeId

The network address to be assumed by the source device.

Definition at line 325 of file [rf4ce-types.h](#).

7.28.2.5 EmberPanId EmberRf4cePairingTableEntry::destPanId

The PAN identifier of the destination device.

Definition at line 329 of file [rf4ce-types.h](#).

7.28.2.6 EmberNodeId EmberRf4cePairingTableEntry::destNodeId

The network address of the destination device.

Definition at line 333 of file [rf4ce-types.h](#).

7.28.2.7 uint16_t EmberRf4cePairingTableEntry::destVendorId

The vendor ID of the destination device.

Definition at line 337 of file [rf4ce-types.h](#).

7.28.2.8 uint8_t EmberRf4cePairingTableEntry::destProfileIdList[EMBER_RF4CE_APPLICATION_PROFILE_ID_LIST_MAX_LENGTH]

The list of profiles supported by the destination device.

Definition at line 341 of file [rf4ce-types.h](#).

7.28.2.9 uint8_t EmberRf4cePairingTableEntry::destProfileIdListLength

The length of the list of supported profiles by the destination device.

Definition at line 345 of file [rf4ce-types.h](#).

7.28.2.10 uint8_t EmberRf4cePairingTableEntry::info

Info byte (bits [3-7] are reserved for internal use).

Definition at line 349 of file [rf4ce-types.h](#).

7.28.2.11 uint8_t EmberRf4cePairingTableEntry::channel

The expected channel of the destination device.

Definition at line 353 of file [rf4ce-types.h](#).

7.28.2.12 uint8_t EmberRf4cePairingTableEntry::capabilities

The node capabilities of the recipient node.

Definition at line 357 of file [rf4ce-types.h](#).

7.28.2.13 uint8_t EmberRf4cePairingTableEntry::lastSeqn

Last MAC sequence number seen on this pairing link.

Definition at line 361 of file [rf4ce-types.h](#).

The documentation for this struct was generated from the following file:

- [rf4ce-types.h](#)

7.29 EmberRf4ceVendorInfo Struct Reference

```
#include <rf4ce-types.h>
```

Data Fields

- `uint16_t vendorId`
- `uint8_t vendorString [EMBER_RF4CE_VENDOR_STRING_LENGTH]`

7.29.1 Detailed Description

Defines the vendor information block (see section 3.3.1, Figure 16).

Definition at line 227 of file [rf4ce-types.h](#).

7.29.2 Field Documentation

7.29.2.1 uint16_t EmberRf4ceVendorInfo::vendorId

The vendor identifier field shall contain the vendor identifier of the node.

Definition at line 232 of file [rf4ce-types.h](#).

7.29.2.2 uint8_t EmberRf4ceVendorInfo::vendorString[EMBER_RF4CE_VENDOR_STRING_LENGTH]

The vendor string field shall contain the vendor string of the node.

Definition at line 236 of file [rf4ce-types.h](#).

The documentation for this struct was generated from the following file:

- [rf4ce-types.h](#)

7.30 EmberRouteTableEntry Struct Reference

```
#include <ember-types.h>
```

Data Fields

- uint16_t destination
- uint16_t nextHop
- uint8_t status
- uint8_t age
- uint8_t concentratorType
- uint8_t routeRecordState

7.30.1 Detailed Description

Defines an entry in the route table.

A route table entry stores information about the next hop along the route to the destination.

Definition at line 1045 of file [ember-types.h](#).

7.30.2 Field Documentation

7.30.2.1 uint16_t EmberRouteTableEntry::destination

The short id of the destination.

Definition at line 1047 of file [ember-types.h](#).

7.30.2.2 uint16_t EmberRouteTableEntry::nextHop

The short id of the next hop to this destination.

Definition at line 1049 of file [ember-types.h](#).

7.30.2.3 uint8_t EmberRouteTableEntry::status

Indicates whether this entry is active (0), being discovered (1), or unused (3).

Definition at line 1052 of file [ember-types.h](#).

7.30.2.4 uint8_t EmberRouteTableEntry::age

The number of seconds since this route entry was last used to send a packet.

Definition at line 1055 of file [ember-types.h](#).

7.30.2.5 uint8_t EmberRouteTableEntry::concentratorType

Indicates whether this destination is a High RAM Concentrator (2), a Low RAM Concentrator (1), or not a concentrator (0).

Definition at line 1058 of file [ember-types.h](#).

7.30.2.6 uint8_t EmberRouteTableEntry::routeRecordState

For a High RAM Concentrator, indicates whether a route record is needed (2), has been sent (1), or is no longer needed (0) because a source routed message from the concentrator has been received.

Definition at line 1063 of file [ember-types.h](#).

The documentation for this struct was generated from the following file:

- [ember-types.h](#)

7.31 EmberSignature283k1Data Struct Reference

```
#include <ember-types.h>
```

Data Fields

- `uint8_t contents [EMBER_SIGNATURE_283K1_SIZE]`

7.31.1 Detailed Description

This data structure contains a DSA signature used in SECT283k1 Elliptical Cryptography. It is the bit concatenation of the 'r' and 's' components of the signature.

Definition at line 1565 of file [ember-types.h](#).

7.31.2 Field Documentation

7.31.2.1 uint8_t EmberSignature283k1Data::contents[EMBER_SIGNATURE_283K1_SIZE]

Definition at line 1566 of file [ember-types.h](#).

The documentation for this struct was generated from the following file:

- [ember-types.h](#)

7.32 EmberSignatureData Struct Reference

```
#include <ember-types.h>
```

Data Fields

- `uint8_t contents [EMBER_SIGNATURE_SIZE]`

7.32.1 Detailed Description

This data structure contains a DSA signature. It is the bit concatenation of the 'r' and 's' components of the signature.

Definition at line 1524 of file [ember-types.h](#).

7.32.2 Field Documentation

7.32.2.1 uint8_t EmberSignatureData::contents[EMBER_SIGNATURE_SIZE]

Definition at line 1525 of file [ember-types.h](#).

The documentation for this struct was generated from the following file:

- [ember-types.h](#)

7.33 EmberSmacData Struct Reference

```
#include <ember-types.h>
```

Data Fields

- [uint8_t contents \[EMBER_SMAC_SIZE\]](#)

7.33.1 Detailed Description

This data structure contains the Shared Message Authentication Code (SMAC) data that is used for Certificate Based Key Exchange (CBKE).

Definition at line 1517 of file [ember-types.h](#).

7.33.2 Field Documentation

7.33.2.1 uint8_t EmberSmacData::contents[EMBER_SMAC_SIZE]

Definition at line 1518 of file [ember-types.h](#).

The documentation for this struct was generated from the following file:

- [ember-types.h](#)

7.34 EmberTaskControl Struct Reference

```
#include <ember-types.h>
```

Data Fields

- `uint32_t nextEventTime`
- `EmberEventData * events`
- `bool busy`

7.34.1 Detailed Description

Control structure for tasks.

This structure should not be accessed directly.

Definition at line 1301 of file `ember-types.h`.

7.34.2 Field Documentation

7.34.2.1 `uint32_t EmberTaskControl::nextEventTime`

Definition at line 1303 of file `ember-types.h`.

7.34.2.2 `EmberEventData* EmberTaskControl::events`

Definition at line 1305 of file `ember-types.h`.

7.34.2.3 `bool EmberTaskControl::busy`

Definition at line 1307 of file `ember-types.h`.

The documentation for this struct was generated from the following file:

- `ember-types.h`

7.35 EmberTokTypeStackZllData Struct Reference

```
#include <zll-types.h>
```

Data Fields

- `uint32_t bitmask`
- `uint16_t freeNodeIdMin`
- `uint16_t freeNodeIdMax`
- `uint16_t myGroupIdMin`
- `uint16_t freeGroupIdMin`
- `uint16_t freeGroupIdMax`
- `uint8_t rssiCorrection`

7.35.1 Detailed Description

Definition at line 256 of file `zll-types.h`.

7.35.2 Field Documentation

7.35.2.1 `uint32_t EmberTokTypeStackZllData::bitmask`

Definition at line 257 of file [zll-types.h](#).

7.35.2.2 `uint16_t EmberTokTypeStackZllData::freeNodeIdMin`

Definition at line 258 of file [zll-types.h](#).

7.35.2.3 `uint16_t EmberTokTypeStackZllData::freeNodeIdMax`

Definition at line 259 of file [zll-types.h](#).

7.35.2.4 `uint16_t EmberTokTypeStackZllData::myGroupIdMin`

Definition at line 260 of file [zll-types.h](#).

7.35.2.5 `uint16_t EmberTokTypeStackZllData::freeGroupIdMin`

Definition at line 261 of file [zll-types.h](#).

7.35.2.6 `uint16_t EmberTokTypeStackZllData::freeGroupIdMax`

Definition at line 262 of file [zll-types.h](#).

7.35.2.7 `uint8_t EmberTokTypeStackZllData::rssiCorrection`

Definition at line 263 of file [zll-types.h](#).

The documentation for this struct was generated from the following file:

- [zll-types.h](#)

7.36 EmberTokTypeStackZllSecurity Struct Reference

```
#include <zll-types.h>
```

Data Fields

- `uint32_t bitmask`
- `uint8_t keyIndex`
- `uint8_t encryptionKey [16]`
- `uint8_t preconfiguredKey [16]`

7.36.1 Detailed Description

Definition at line 266 of file [zll-types.h](#).

7.36.2 Field Documentation

7.36.2.1 `uint32_t EmberTokTypeStackZllSecurity::bitmask`

Definition at line 267 of file [zll-types.h](#).

7.36.2.2 `uint8_t EmberTokTypeStackZllSecurity::keyIndex`

Definition at line 268 of file [zll-types.h](#).

7.36.2.3 `uint8_t EmberTokTypeStackZllSecurity::encryptionKey[16]`

Definition at line 269 of file [zll-types.h](#).

7.36.2.4 `uint8_t EmberTokTypeStackZllSecurity::preconfiguredKey[16]`

Definition at line 270 of file [zll-types.h](#).

The documentation for this struct was generated from the following file:

- [zll-types.h](#)

7.37 EmberVersion Struct Reference

```
#include <ember-types.h>
```

Data Fields

- `uint16_t build`
- `uint8_t major`
- `uint8_t minor`
- `uint8_t patch`
- `uint8_t special`
- `EmberVersionType type`

7.37.1 Detailed Description

Version struct containing all version information.

Definition at line 90 of file [ember-types.h](#).

7.37.2 Field Documentation

7.37.2.1 uint16_t EmberVersion::build

Definition at line 91 of file [ember-types.h](#).

7.37.2.2 uint8_t EmberVersion::major

Definition at line 92 of file [ember-types.h](#).

7.37.2.3 uint8_t EmberVersion::minor

Definition at line 93 of file [ember-types.h](#).

7.37.2.4 uint8_t EmberVersion::patch

Definition at line 94 of file [ember-types.h](#).

7.37.2.5 uint8_t EmberVersion::special

Definition at line 95 of file [ember-types.h](#).

7.37.2.6 EmberVersionType EmberVersion::type

Definition at line 96 of file [ember-types.h](#).

The documentation for this struct was generated from the following file:

- [ember-types.h](#)

7.38 EmberZigbeeNetwork Struct Reference

```
#include <ember-types.h>
```

Data Fields

- uint16_t panId
- uint8_t channel
- bool allowingJoin
- uint8_t extendedPanId [8]
- uint8_t stackProfile
- uint8_t nwkUpdateId

7.38.1 Detailed Description

Defines a ZigBee network and the associated parameters.

Definition at line 441 of file [ember-types.h](#).

7.38.2 Field Documentation

7.38.2.1 `uint16_t EmberZigbeeNetwork::panId`

Definition at line 442 of file [ember-types.h](#).

7.38.2.2 `uint8_t EmberZigbeeNetwork::channel`

Definition at line 443 of file [ember-types.h](#).

7.38.2.3 `bool EmberZigbeeNetwork::allowingJoin`

Definition at line 444 of file [ember-types.h](#).

7.38.2.4 `uint8_t EmberZigbeeNetwork::extendedPanId[8]`

Definition at line 445 of file [ember-types.h](#).

7.38.2.5 `uint8_t EmberZigbeeNetwork::stackProfile`

Definition at line 446 of file [ember-types.h](#).

7.38.2.6 `uint8_t EmberZigbeeNetwork::nwkUpdateId`

Definition at line 447 of file [ember-types.h](#).

The documentation for this struct was generated from the following file:

- [ember-types.h](#)

7.39 EmberZllAddressAssignment Struct Reference

```
#include <zll-types.h>
```

Data Fields

- [EmberNodeId nodeId](#)
- [EmberNodeId freeNodeIdMin](#)
- [EmberNodeId freeNodeIdMax](#)
- [EmberMulticastId groupIdMin](#)
- [EmberMulticastId groupIdMax](#)
- [EmberMulticastId freeGroupIdMin](#)
- [EmberMulticastId freeGroupIdMax](#)

7.39.1 Detailed Description

Network and group address assignment information.

Definition at line 150 of file [zll-types.h](#).

7.39.2 Field Documentation

7.39.2.1 EmberNodeId EmberZllAddressAssignment::nodeId

Definition at line 151 of file [zll-types.h](#).

7.39.2.2 EmberNodeId EmberZllAddressAssignment::freeNodeIdMin

Definition at line 152 of file [zll-types.h](#).

7.39.2.3 EmberNodeId EmberZllAddressAssignment::freeNodeIdMax

Definition at line 153 of file [zll-types.h](#).

7.39.2.4 EmberMulticastId EmberZllAddressAssignment::groupIdMin

Definition at line 154 of file [zll-types.h](#).

7.39.2.5 EmberMulticastId EmberZllAddressAssignment::groupIdMax

Definition at line 155 of file [zll-types.h](#).

7.39.2.6 EmberMulticastId EmberZllAddressAssignment::freeGroupIdMin

Definition at line 156 of file [zll-types.h](#).

7.39.2.7 EmberMulticastId EmberZllAddressAssignment::freeGroupIdMax

Definition at line 157 of file [zll-types.h](#).

The documentation for this struct was generated from the following file:

- [zll-types.h](#)

7.40 EmberZllDeviceInfoRecord Struct Reference

```
#include <zll-types.h>
```

Data Fields

- [EmberEUI64 ieeeAddress](#)
- [uint8_t endpointId](#)
- [uint16_t profileId](#)
- [uint16_t deviceId](#)
- [uint8_t version](#)
- [uint8_t groupIdCount](#)

7.40.1 Detailed Description

Information discovered during a ZLL scan about the ZLL device's endpoint information.

Definition at line 138 of file [zll-types.h](#).

7.40.2 Field Documentation

7.40.2.1 EmberEUI64 EmberZllDeviceInfoRecord::ieeeAddress

Definition at line 139 of file [zll-types.h](#).

7.40.2.2 uint8_t EmberZllDeviceInfoRecord::endpointId

Definition at line 140 of file [zll-types.h](#).

7.40.2.3 uint16_t EmberZllDeviceInfoRecord::profileId

Definition at line 141 of file [zll-types.h](#).

7.40.2.4 uint16_t EmberZllDeviceInfoRecord::deviceId

Definition at line 142 of file [zll-types.h](#).

7.40.2.5 uint8_t EmberZllDeviceInfoRecord::version

Definition at line 143 of file [zll-types.h](#).

7.40.2.6 uint8_t EmberZllDeviceInfoRecord::groupIdCount

Definition at line 144 of file [zll-types.h](#).

The documentation for this struct was generated from the following file:

- [zll-types.h](#)

7.41 EmberZllInitialSecurityState Struct Reference

```
#include <zll-types.h>
```

Data Fields

- `uint32_t bitmask`
- `EmberZllKeyIndex keyIndex`
- `EmberKeyData encryptionKey`
- `EmberKeyData preconfiguredKey`

7.41.1 Detailed Description

This describes the Initial Security features and requirements that will be used when forming or joining ZigBee Light Link networks.

Definition at line 227 of file [zll-types.h](#).

7.41.2 Field Documentation

7.41.2.1 `uint32_t EmberZllInitialSecurityState::bitmask`

This bitmask is unused. All values are reserved for future use.

Definition at line 229 of file [zll-types.h](#).

7.41.2.2 `EmberZllKeyIndex EmberZllInitialSecurityState::keyIndex`

The key encryption algorithm advertised by the application.

Definition at line 231 of file [zll-types.h](#).

7.41.2.3 `EmberKeyData EmberZllInitialSecurityState::encryptionKey`

The encryption key for use by algorithms that require it.

Definition at line 233 of file [zll-types.h](#).

7.41.2.4 `EmberKeyData EmberZllInitialSecurityState::preconfiguredKey`

The pre-configured link key used during classical ZigBee commissioning.

Definition at line 235 of file [zll-types.h](#).

The documentation for this struct was generated from the following file:

- [zll-types.h](#)

7.42 EmberZllNetwork Struct Reference

```
#include <zll-types.h>
```

Data Fields

- `EmberZigbeeNetwork zigbeeNetwork`
- `EmberZllSecurityAlgorithmData securityAlgorithm`
- `EmberEUI64 eui64`
- `EmberNodeId nodeId`
- `EmberZllState state`
- `EmberNodeType nodeType`
- `uint8_t numberSubDevices`
- `uint8_t totalGroupIdentifiers`
- `uint8_t rssiCorrection`

7.42.1 Detailed Description

Information about the ZLL network and specific device that responded to a ZLL scan request.

Definition at line 122 of file [zll-types.h](#).

7.42.2 Field Documentation

7.42.2.1 EmberZigbeeNetwork `EmberZllNetwork::zigbeeNetwork`

Definition at line 123 of file [zll-types.h](#).

7.42.2.2 EmberZllSecurityAlgorithmData `EmberZllNetwork::securityAlgorithm`

Definition at line 124 of file [zll-types.h](#).

7.42.2.3 EmberEUI64 `EmberZllNetwork::eui64`

Definition at line 125 of file [zll-types.h](#).

7.42.2.4 EmberNodeId `EmberZllNetwork::nodeId`

Definition at line 126 of file [zll-types.h](#).

7.42.2.5 EmberZllState `EmberZllNetwork::state`

Definition at line 127 of file [zll-types.h](#).

7.42.2.6 EmberNodeType `EmberZllNetwork::nodeType`

Definition at line 128 of file [zll-types.h](#).

7.42.2.7 `uint8_t EmberZllNetwork::numberSubDevices`

Definition at line 129 of file [zll-types.h](#).

7.42.2.8 `uint8_t EmberZllNetwork::totalGroupIdentifiers`

Definition at line 130 of file [zll-types.h](#).

7.42.2.9 `uint8_t EmberZllNetwork::rssicorrection`

Definition at line 131 of file [zll-types.h](#).

The documentation for this struct was generated from the following file:

- [zll-types.h](#)

7.43 EmberZllSecurityAlgorithmData Struct Reference

```
#include <zll-types.h>
```

Data Fields

- `uint32_t transactionId`
- `uint32_t responseId`
- `uint16_t bitmask`

7.43.1 Detailed Description

Information about the ZLL security being and how to transmit the network key to the device securely.

Definition at line 112 of file [zll-types.h](#).

7.43.2 Field Documentation

7.43.2.1 `uint32_t EmberZllSecurityAlgorithmData::transactionId`

Definition at line 113 of file [zll-types.h](#).

7.43.2.2 `uint32_t EmberZllSecurityAlgorithmData::responseId`

Definition at line 114 of file [zll-types.h](#).

7.43.2.3 `uint16_t EmberZllSecurityAlgorithmData::bitmask`

Definition at line 115 of file [zll-types.h](#).

The documentation for this struct was generated from the following file:

- [zll-types.h](#)

7.44 HalEepromInformationType Struct Reference

```
#include <bootloader-eeprom.h>
```

Data Fields

- uint16_t version
- uint16_t capabilitiesMask
- uint16_t pageEraseMs
- uint16_t partEraseMs
- uint32_t pageSize
- uint32_t partSize
- const char *const partDescription
- uint8_t wordSizeBytes

7.44.1 Detailed Description

This structure defines a variety of information about the attached external EEPROM device.

Definition at line 100 of file [bootloader-eeprom.h](#).

7.44.2 Field Documentation

7.44.2.1 uint16_t HalEepromInformationType::version

The version of this data structure

Definition at line 102 of file [bootloader-eeprom.h](#).

7.44.2.2 uint16_t HalEepromInformationType::capabilitiesMask

A bitmask describing the capabilities of this particular external EEPROM

Definition at line 104 of file [bootloader-eeprom.h](#).

7.44.2.3 uint16_t HalEepromInformationType::pageEraseMs

Maximum time it takes to erase a page. (in 1024Hz Milliseconds)

Definition at line 106 of file [bootloader-eeprom.h](#).

7.44.2.4 uint16_t HalEepromInformationType::partEraseMs

Maximum time it takes to erase the entire part. (in 1024Hz Milliseconds)

Definition at line 108 of file [bootloader-eeprom.h](#).

7.44.2.5 `uint32_t HalEepromInformationType::pageSize`

The size of a single erasable page in bytes

Definition at line 110 of file [bootloader-eeprom.h](#).

7.44.2.6 `uint32_t HalEepromInformationType::partSize`

The total size of the external EEPROM in bytes

Definition at line 112 of file [bootloader-eeprom.h](#).

7.44.2.7 `const char* const HalEepromInformationType::partDescription`

Pointer to a string describing the attached external EEPROM

Definition at line 114 of file [bootloader-eeprom.h](#).

7.44.2.8 `uint8_t HalEepromInformationType::wordSizeBytes`

The number of bytes in a word for the external EEPROM

Definition at line 116 of file [bootloader-eeprom.h](#).

The documentation for this struct was generated from the following file:

- [bootloader-eeprom.h](#)

7.45 InterPanHeader Struct Reference

```
#include <ami-inter-pan.h>
```

Data Fields

- `uint8_t messageType`
- `uint16_t panId`
- `bool hasLongAddress`
- `EmberNodeId shortAddress`
- `EmberEUI64 longAddress`
- `uint16_t profileId`
- `uint16_t clusterId`
- `uint16_t groupId`

7.45.1 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](#).

7.45.2 Field Documentation

7.45.2.1 `uint8_t InterPanHeader::messageType`

Definition at line 52 of file [ami-inter-pan.h](#).

7.45.2.2 `uint16_t InterPanHeader::panId`

Definition at line 57 of file [ami-inter-pan.h](#).

7.45.2.3 `bool InterPanHeader::hasLongAddress`

Definition at line 58 of file [ami-inter-pan.h](#).

7.45.2.4 `EmberNodeId InterPanHeader::shortAddress`

Definition at line 59 of file [ami-inter-pan.h](#).

7.45.2.5 `EmberEUI64 InterPanHeader::longAddress`

Definition at line 60 of file [ami-inter-pan.h](#).

7.45.2.6 `uint16_t InterPanHeader::profileId`

Definition at line 63 of file [ami-inter-pan.h](#).

7.45.2.7 `uint16_t InterPanHeader::clusterId`

Definition at line 64 of file [ami-inter-pan.h](#).

7.45.2.8 `uint16_t InterPanHeader::groupId`

Definition at line 65 of file [ami-inter-pan.h](#).

The documentation for this struct was generated from the following files:

- [ami-inter-pan.h](#)
- [ami-inter-pan-host.h](#)

7.46 USB_ConfigurationDescriptor_TypeDef Struct Reference

```
#include <em_usb.h>
```

Data Fields

- `uint8_t bLength`
- `uint8_t bDescriptorType`
- `uint16_t wTotalLength`
- `uint8_t bNumInterfaces`
- `uint8_t bConfigurationValue`
- `uint8_t iConfiguration`
- `uint8_t bmAttributes`
- `uint8_t bMaxPower`

7.46.1 Detailed Description

USB Configuration Descriptor.

Definition at line 422 of file [em_usb.h](#).

7.46.2 Field Documentation

7.46.2.1 `uint8_t USB_ConfigurationDescriptor_TypeDef::bLength`

Size of this descriptor in bytes

Definition at line 424 of file [em_usb.h](#).

7.46.2.2 `uint8_t USB_ConfigurationDescriptor_TypeDef::bDescriptorType`

Constant CONFIGURATION Descriptor Type

Definition at line 425 of file [em_usb.h](#).

7.46.2.3 `uint16_t USB_ConfigurationDescriptor_TypeDef::wTotalLength`

Total length of data returned for this configuration. Includes the combined length of all descriptors (configuration, interface, endpoint, and class- or vendor-specific) returned for this configuration.

Definition at line 426 of file [em_usb.h](#).

7.46.2.4 `uint8_t USB_ConfigurationDescriptor_TypeDef::bNumInterfaces`

Number of interfaces supported by this configuration

Definition at line 431 of file [em_usb.h](#).

7.46.2.5 `uint8_t USB_ConfigurationDescriptor_TypeDef::bConfigurationValue`

Value to use as an argument to the SetConfiguration request to select this configuration.

Definition at line 433 of file [em_usb.h](#).

7.46.2.6 `uint8_t USB_ConfigurationDescriptor_TypeDef::iConfiguration`

Index of string descriptor describing this configuration.

Definition at line 436 of file [em_usb.h](#).

7.46.2.7 `uint8_t USB_ConfigurationDescriptor_TypeDef::bmAttributes`

Configuration characteristics.

D7: Reserved (set to one)

D6: Self-powered

D5: Remote Wakeup

D4...0: Reserved (reset to zero)

Definition at line 438 of file [em_usb.h](#).

7.46.2.8 `uint8_t USB_ConfigurationDescriptor_TypeDef::bMaxPower`

Maximum power consumption of the USB device, unit is 2mA per LSB

Definition at line 443 of file [em_usb.h](#).

The documentation for this struct was generated from the following file:

- [em_usb.h](#)

7.47 `USB_DeviceDescriptor_TypeDef` Struct Reference

```
#include <em_usb.h>
```

Data Fields

- `uint8_t bLength`
- `uint8_t bDescriptorType`
- `uint16_t bcdUSB`
- `uint8_t bDeviceClass`
- `uint8_t bDeviceSubClass`
- `uint8_t bDeviceProtocol`
- `uint8_t bMaxPacketSize0`
- `uint16_t idVendor`
- `uint16_t idProduct`
- `uint16_t bcdDevice`
- `uint8_t iManufacturer`
- `uint8_t iProduct`
- `uint8_t iSerialNumber`
- `uint8_t bNumConfigurations`

7.47.1 Detailed Description

USB Device Descriptor.

Definition at line 398 of file [em_usb.h](#).

7.47.2 Field Documentation

7.47.2.1 uint8_t USB_DeviceDescriptor_TypeDef::bLength

Size of this descriptor in bytes

Definition at line 400 of file [em_usb.h](#).

7.47.2.2 uint8_t USB_DeviceDescriptor_TypeDef::bDescriptorType

Constant DEVICE Descriptor Type

Definition at line 401 of file [em_usb.h](#).

7.47.2.3 uint16_t USB_DeviceDescriptor_TypeDef::bcdUSB

USB Specification Release Number in Binary-Coded Decimal

Definition at line 402 of file [em_usb.h](#).

7.47.2.4 uint8_t USB_DeviceDescriptor_TypeDef::bDeviceClass

Class code (assigned by the USB-IF)

Definition at line 404 of file [em_usb.h](#).

7.47.2.5 uint8_t USB_DeviceDescriptor_TypeDef::bDeviceSubClass

Subclass code (assigned by the USB-IF)

Definition at line 405 of file [em_usb.h](#).

7.47.2.6 uint8_t USB_DeviceDescriptor_TypeDef::bDeviceProtocol

Protocol code (assigned by the USB-IF)

Definition at line 406 of file [em_usb.h](#).

7.47.2.7 uint8_t USB_DeviceDescriptor_TypeDef::bMaxPacketSize0

Maximum packet size for endpoint zero

Definition at line 407 of file [em_usb.h](#).

7.47.2.8 uint16_t USB_DeviceDescriptor_TypeDef::idVendor

Vendor ID (assigned by the USB-IF)

Definition at line 408 of file [em_usb.h](#).

7.47.2.9 uint16_t USB_DeviceDescriptor_TypeDef::idProduct

Product ID (assigned by the manufacturer)

Definition at line 409 of file [em_usb.h](#).

7.47.2.10 uint16_t USB_DeviceDescriptor_TypeDef::bcdDevice

Device release number in binary-coded decimal

Definition at line 410 of file [em_usb.h](#).

7.47.2.11 uint8_t USB_DeviceDescriptor_TypeDef::iManufacturer

Index of string descriptor describing manufacturer

Definition at line 411 of file [em_usb.h](#).

7.47.2.12 uint8_t USB_DeviceDescriptor_TypeDef::iProduct

Index of string descriptor describing product

Definition at line 412 of file [em_usb.h](#).

7.47.2.13 uint8_t USB_DeviceDescriptor_TypeDef::iSerialNumber

Index of string descriptor describing the device serialnumber

Definition at line 413 of file [em_usb.h](#).

7.47.2.14 uint8_t USB_DeviceDescriptor_TypeDef::bNumConfigurations

Number of possible configurations

Definition at line 415 of file [em_usb.h](#).

The documentation for this struct was generated from the following file:

- [em_usb.h](#)

7.48 USB_EndpointDescriptor_TypeDef Struct Reference

```
#include <em_usb.h>
```

Data Fields

- `uint8_t bLength`
- `uint8_t bDescriptorType`
- `uint8_t bEndpointAddress`
- `uint8_t bmAttributes`
- `uint16_t wMaxPacketSize`
- `uint8_t bInterval`

7.48.1 Detailed Description

USB Endpoint Descriptor.

Definition at line 490 of file [em_usb.h](#).

7.48.2 Field Documentation

7.48.2.1 `uint8_t USB_EndpointDescriptor_TypeDef::bLength`

Size of this descriptor in bytes

Definition at line 492 of file [em_usb.h](#).

7.48.2.2 `uint8_t USB_EndpointDescriptor_TypeDef::bDescriptorType`

Constant ENDPOINT Descriptor Type

Definition at line 493 of file [em_usb.h](#).

7.48.2.3 `uint8_t USB_EndpointDescriptor_TypeDef::bEndpointAddress`

The address of the endpoint

Definition at line 494 of file [em_usb.h](#).

Referenced by [USBD_Init\(\)](#).

7.48.2.4 `uint8_t USB_EndpointDescriptor_TypeDef::bmAttributes`

This field describes the endpoint attributes

Definition at line 495 of file [em_usb.h](#).

Referenced by [USBD_Init\(\)](#).

7.48.2.5 `uint16_t USB_EndpointDescriptor_TypeDef::wMaxPacketSize`

Maximum packet size for the endpoint

Definition at line 496 of file [em_usb.h](#).

Referenced by [USBD_Init\(\)](#).

7.48.2.6 uint8_t USB_EndpointDescriptor_TypeDef::bInterval

Interval for polling EP for data transfers

Definition at line 497 of file [em_usb.h](#).

The documentation for this struct was generated from the following file:

- [em_usb.h](#)

7.49 USB_InterfaceDescriptor_TypeDef Struct Reference

```
#include <em_usb.h>
```

Data Fields

- uint8_t **bLength**
- uint8_t **bDescriptorType**
- uint8_t **bInterfaceNumber**
- uint8_t **bAlternateSetting**
- uint8_t **bNumEndpoints**
- uint8_t **bInterfaceClass**
- uint8_t **bInterfaceSubClass**
- uint8_t **bInterfaceProtocol**
- uint8_t **iInterface**

7.49.1 Detailed Description

USB Interface Descriptor.

Definition at line 450 of file [em_usb.h](#).

7.49.2 Field Documentation

7.49.2.1 uint8_t USB_InterfaceDescriptor_TypeDef::bLength

Size of this descriptor in bytes.

Definition at line 452 of file [em_usb.h](#).

7.49.2.2 uint8_t USB_InterfaceDescriptor_TypeDef::bDescriptorType

Constant INTERFACE Descriptor Type.

Definition at line 453 of file [em_usb.h](#).

7.49.2.3 uint8_t USB_InterfaceDescriptor_TypeDef::bInterfaceNumber

Number of this interface. Zero-based value identifying the index in the array of concurrent interfaces supported by this configuration.

Definition at line 454 of file [em_usb.h](#).

7.49.2.4 uint8_t USB_InterfaceDescriptor_TypeDef::bAlternateSetting

Value used to select this alternate setting for the interface identified in the prior field.

Definition at line [457](#) of file [em_usb.h](#).

7.49.2.5 uint8_t USB_InterfaceDescriptor_TypeDef::bNumEndpoints

Number of endpoints used by this interface (excluding endpoint zero). If this value is zero, this interface only uses the Default Control Pipe.

Definition at line [459](#) of file [em_usb.h](#).

7.49.2.6 uint8_t USB_InterfaceDescriptor_TypeDef::bInterfaceClass

Class code (assigned by the USB-IF). A value of zero is reserved for future standardization. If this field is set to FFH, the interface class is vendor-specific. All other values are reserved for assignment by the USB-IF.

Definition at line [462](#) of file [em_usb.h](#).

7.49.2.7 uint8_t USB_InterfaceDescriptor_TypeDef::bInterfaceSubClass

Subclass code (assigned by the USB-IF). These codes are qualified by the value of the b-InterfaceClass field. If the bInterfaceClass field is reset to zero, this field must also be reset to zero. If the bInterfaceClass field is not set to FFH, all values are reserved for assignment by the USB-IF.

Definition at line [467](#) of file [em_usb.h](#).

7.49.2.8 uint8_t USB_InterfaceDescriptor_TypeDef::bInterfaceProtocol

Protocol code (assigned by the USB). These codes are qualified by the value of the b-InterfaceClass and the bInterfaceSubClass fields. If an interface supports class-specific requests, this code identifies the protocols that the device uses as defined by the specification of the device class. If this field is reset to zero, the device does not use a class-specific protocol on this interface. If this field is set to FFH, the device uses a vendor-specific protocol for this interface

Definition at line [473](#) of file [em_usb.h](#).

7.49.2.9 uint8_t USB_InterfaceDescriptor_TypeDef::iInterface

Index of string descriptor describing this interface.

Definition at line [483](#) of file [em_usb.h](#).

The documentation for this struct was generated from the following file:

- [em_usb.h](#)

7.50 USB_Setup_TypeDef Struct Reference

```
#include <em_usb.h>
```

Data Fields

- union {
 struct {
 union {
 struct {
 uint8_t **Recipient**: 5
 uint8_t **Type**: 2
 uint8_t **Direction**: 1
 }
 uint8_t **bmRequestType**
 }
 uint8_t **bRequest**
 uint16_t **wValue**
 uint16_t **wIndex**
 uint16_t **wLength**
 }
 uint32_t **dw** [2]
 };

7.50.1 Detailed Description

USB Setup request package.

Definition at line 370 of file [em_usb.h](#).

7.50.2 Field Documentation

7.50.2.1 uint8_t USB_Setup_TypeDef::Recipient

Request recipient (device, interface, endpoint or other).

Definition at line 380 of file [em_usb.h](#).

7.50.2.2 uint8_t USB_Setup_TypeDef::Type

Request type (standard, class or vendor).

Definition at line 381 of file [em_usb.h](#).

7.50.2.3 uint8_t USB_Setup_TypeDef::Direction

Transfer direction of SETUP data phase.

Definition at line 382 of file [em_usb.h](#).

7.50.2.4 `uint8_t USB_Setup_TypeDef::bmRequestType`

Request characteristics.

Definition at line 384 of file [em_usb.h](#).

7.50.2.5 `uint8_t USB_Setup_TypeDef::bRequest`

Request code.

Definition at line 386 of file [em_usb.h](#).

7.50.2.6 `uint16_t USB_Setup_TypeDef::wValue`

Varies according to request.

Definition at line 387 of file [em_usb.h](#).

7.50.2.7 `uint16_t USB_Setup_TypeDef::wIndex`

Index or offset, varies according to request.

Definition at line 388 of file [em_usb.h](#).

7.50.2.8 `uint16_t USB_Setup_TypeDef::wLength`

Number of bytes to transfer if there is a data stage.

Definition at line 389 of file [em_usb.h](#).

7.50.2.9 `uint32_t USB_Setup_TypeDef::dw[2]`

Definition at line 391 of file [em_usb.h](#).

7.50.2.10 `union { ... }`

The documentation for this struct was generated from the following file:

- [em_usb.h](#)

7.51 `USB_StringDescriptor_TypeDef Struct Reference`

```
#include <em_usb.h>
```

Data Fields

- `uint8_t len`
- `uint8_t type`
- `char16_t name []`

7.51.1 Detailed Description

USB String Descriptor.

Definition at line 503 of file [em_usb.h](#).

7.51.2 Field Documentation

7.51.2.1 uint8_t USB_StringDescriptor_TypeDef::len

Size of this descriptor in bytes.

Definition at line 505 of file [em_usb.h](#).

7.51.2.2 uint8_t USB_StringDescriptor_TypeDef::type

Constant STRING Descriptor Type.

Definition at line 506 of file [em_usb.h](#).

7.51.2.3 char16_t USB_StringDescriptor_TypeDef::name[]

The string encoded with UTF-16LE UNICODE charset.

Definition at line 507 of file [em_usb.h](#).

The documentation for this struct was generated from the following file:

- [em_usb.h](#)

7.52 USBD_Callbacks_TypeDef Struct Reference

```
#include <em_usb.h>
```

Data Fields

- const [USBD_UsbResetCb_TypeDef](#) usbReset
- const [USBD_DeviceStateChangeCb_TypeDef](#) usbStateChange
- const [USBD_SetupCmdCb_TypeDef](#) setupCmd
- const [USBD_IsSelfPoweredCb_TypeDef](#) isSelfPowered
- const [USBD_SofIntCb_TypeDef](#) sofInt

7.52.1 Detailed Description

USB Device stack callback structure.

Callback functions used by the device stack to signal events or query status to/from the application. See [USBD_Init_TypeDef](#). Assign members to NULL if your application don't need a specific callback.

Definition at line 664 of file [em_usb.h](#).

7.52.2 Field Documentation

7.52.2.1 const USBD_UsbResetCb_TypeDef USBD_Callbacks_TypeDef::usbReset

Called whenever USB reset signalling is detected on the USB port.

Definition at line 666 of file [em_usb.h](#).

7.52.2.2 const USBD_DeviceStateChangeCb_TypeDef USBD_Callbacks_TypeDef::usbStateChange

Called whenever the device change state.

Definition at line 668 of file [em_usb.h](#).

7.52.2.3 const USBD_SetupCmdCb_TypeDef USBD_Callbacks_TypeDef::setupCmd

Called on each setup request received from host.

Definition at line 669 of file [em_usb.h](#).

7.52.2.4 const USBD_IsSelfPoweredCb_TypeDef USBD_Callbacks_TypeDef::isSelfPowered

Called whenever the device stack needs to query if the device is currently self- or bus-powered. Applies to devices which can operate in both modes.

Definition at line 670 of file [em_usb.h](#).

7.52.2.5 const USBD_SofIntCb_TypeDef USBD_Callbacks_TypeDef::sofInt

Called at each SOF interrupt. If NULL, the device stack will not enable the SOF interrupt.

Definition at line 673 of file [em_usb.h](#).

The documentation for this struct was generated from the following file:

- [em_usb.h](#)

7.53 USBD_Init_TypeDef Struct Reference

```
#include <em_usb.h>
```

Data Fields

- const [USB_DeviceDescriptor_TypeDef](#) * deviceDescriptor
- const uint8_t * configDescriptor
- const void *const * stringDescriptors
- const uint8_t [numberOfStrings](#)
- const uint8_t * [bufferingMultiplier](#)

- USBD_Callbacks_TypeDef* [callbacks](#)
- const uint32_t [reserved](#)

7.53.1 Detailed Description

USB Device stack initialization structure.

This structure is passed to [USBD_Init\(\)](#) when starting up the device.

Definition at line [642](#) of file [em_usb.h](#).

7.53.2 Field Documentation

7.53.2.1 const USB_DeviceDescriptor_TypeDef* USBD_Init_TypeDef::deviceDescriptor

Pointer to a device descriptor.

Definition at line [644](#) of file [em_usb.h](#).

Referenced by [USBD_Init\(\)](#).

7.53.2.2 const uint8_t* USBD_Init_TypeDef::configDescriptor

Pointer to a configuration descriptor.

Definition at line [645](#) of file [em_usb.h](#).

Referenced by [USBD_Init\(\)](#).

7.53.2.3 const void* const* USBD_Init_TypeDef::stringDescriptors

Pointer to an array of string descriptor pointers.

Definition at line [646](#) of file [em_usb.h](#).

Referenced by [USBD_Init\(\)](#).

7.53.2.4 const uint8_t USBD_Init_TypeDef::numberOfStrings

Number of strings in string descriptor array.

Definition at line [647](#) of file [em_usb.h](#).

Referenced by [USBD_Init\(\)](#).

7.53.2.5 const uint8_t* USBD_Init_TypeDef::bufferingMultiplier

Pointer to an array defining the size of the endpoint buffers. The size is given in multiples of endpoint size. Generally a value of 1 (single) or 2 (double) buffering should be used.

Definition at line [648](#) of file [em_usb.h](#).

Referenced by [USBD_Init\(\)](#).

7.53.2.6 `USBD_Callbacks_TypeDef* USBD_Init_TypeDef::callbacks`

Pointer to struct with callbacks ([USBD_Callbacks_TypeDef](#)). These callbacks are used by the device stack to signal events to or query the application.

Definition at line [653](#) of file [em_usb.h](#).

Referenced by [USBD_Init\(\)](#).

7.53.2.7 `const uint32_t USBD_Init_TypeDef::reserved`

Reserved for future use.

Definition at line [657](#) of file [em_usb.h](#).

The documentation for this struct was generated from the following file:

- [em_usb.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 adc.h File Reference

Macros

- [#define NUM_ADC_USERS](#)

TypeDefs

- [typedef uint8_t ADCChannelType](#)
- [typedef uint8_t ADCReferenceType](#)
- [typedef uint8_t ADCRateType](#)

Enumerations

- [enum ADCUser { ADC_USER_LQI, ADC_USER_APP, ADC_USER_APP2 }](#)

Functions

- void `halInternalInitAdc` (void)
- void `halInternalSleepAdc` (void)
- EmberStatus `halStartAdcConversion` (ADCUser id, ADCReferenceType reference, ADCChannelType channel, ADCRateType rate)
- EmberStatus `halRequestAdcData` (ADCUser id, uint16_t *value)
- EmberStatus `halReadAdcBlocking` (ADCUser id, uint16_t *value)
- EmberStatus `halAdcCalibrate` (ADCUser id)
- int32_t `halConvertValueToVolts` (uint16_t value)
- void `emberCalibrateVref` (void)
- void `halAdcSetClock` (bool slow)
- bool `halAdcGetClock` (void)

8.3.1 Detailed Description

Header for A/D converter. See [Adc](#) for documentation.

Definition in file `adc.h`.

8.4 adc.h

```

00001
00039 #ifndef __ADC_H__
00040 #define __ADC_H__
00041
00042 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00043
00051 enum ADCUser
00052 #else
00053 // A type for the ADC User enumeration.
00054 typedef uint8_t ADCUser;
00055 enum
00056 #endif //DOXYGEN_SHOULD_SKIP_THIS
00057 {
00059     ADC_USER_LQI = 0,
00061     ADC_USER_APP = 1,
00063     ADC_USER_APP2 = 2
00064 };
00065
00069 #define NUM_ADC_USERS 3 // make sure to update if the above is adjusted
00070
00073 typedef uint8_t ADCChannelType;
00074
00077 typedef uint8_t ADCReferenceType;
00078
00081 typedef uint8_t ADCRateType;
00082
00083
00084 #if defined(EMBER_TEST)
00085     #include "unix/simulation/adc.h"
00086 #elif defined(CORTEXM3_EFM32_MICRO)
00087     // disabling until valid. search for CORTEXM3_EFM32_MICRO for more instances
00088     // #include "cortexm3/efm32/adc.h"
00089 #elif defined(CORTEXM3)
00090     #include "cortexm3/adc.h"
00091 #else
00092     // platform that doesn't have ADC support
00093 #endif
00094
00095
00100 void halInternalInitAdc(void);
00101
00102
00106 void halInternalSleepAdc(void);
00107

```

```

00108
00138 EmberStatus halStartAdcConversion(ADCUser
00139     id,
00140             ADCReferenceType reference,
00141             ADCChannelType channel,
00142             ADCRateType rate);
00142
00163 EmberStatus halRequestAdcData(ADCUser id,
00164     uint16_t *value);
00164
00165
00183 EmberStatus halReadAdcBlocking(ADCUser id,
00184     uint16_t *value);
00184
00185
00202 EmberStatus halAdcCalibrate(ADCUser id);
00203
00204
00216 int32_t halConvertValueToVolts(uint16_t value);
00217
00218
00227 void emberCalibrateVref(void);
00228
00233 void halAdcSetClock(bool slow);
00234
00239 bool halAdcGetClock(void);
00240
00241
00242 #endif // __ADC_H__
00243

```

8.5 adc.h File Reference

Macros

- #define ADC_MUX_ADC0
- #define ADC_MUX_ADC1
- #define ADC_MUX_ADC2
- #define ADC_MUX_ADC3
- #define ADC_MUX_ADC4
- #define ADC_MUX_ADC5
- #define ADC_MUX_GND
- #define ADC_MUX_VREF2
- #define ADC_MUX_VREF
- #define ADC_MUX_VREG2

Enumerations

- enum { ADC_REF_INT }
- enum {
 ADC_SAMPLE_CLOCKS_32, ADC_SAMPLE_CLOCKS_64, ADC_SAMPLE_CLOCKS_128, ADC_SAMPLE_CLOCKS_256, ADC_SAMPLE_CLOCKS_512, ADC_SAMPLE_CLOCKS_1024, ADC_SAMPLE_CLOCKS_2048, ADC_SAMPLE_CLOCKS_4096 }
- enum {
 ADC_CONVERSION_TIME_US_32, ADC_CONVERSION_TIME_US_64, ADC_CONVERSION_TIME_US_128, ADC_CONVERSION_TIME_US_256, ADC_CONVERSION_TIME_US_512, ADC_CONVERSION_TIME_US_1024, ADC_CONVERSION_TIME_US_2048, ADC_CONVERSION_TIME_US_4096 }

- enum {
 ADC_SOURCE_ADC0_VREF2, ADC_SOURCE_ADC0_GND, ADC_SOURCE_ADC1_VREF2,
 ADC_SOURCE_ADC1_GND, ADC_SOURCE_ADC2_VREF2, ADC_SOURCE_ADC2_GND,
 ADC_SOURCE_ADC3_VREF2, ADC_SOURCE_ADC3_GND, ADC_SOURCE_ADC4_VREF2,
 ADC_SOURCE_ADC4_GND, ADC_SOURCE_ADC5_VREF2, ADC_SOURCE_ADC5_GND,
 ADC_SOURCE_ADC1_ADC0, ADC_SOURCE_ADC0_ADC1, ADC_SOURCE_ADC3_ADC2,
 ADC_SOURCE_ADC2_ADC3, ADC_SOURCE_ADC5_ADC4, ADC_SOURCE_GND_VREF2,
 ADC_SOURCE_VGND, ADC_SOURCE_VREF_VREF2, ADC_SOURCE_VREF, ADC_SOURCE_VREG2_VREF2,
 ADC_SOURCE_VDD_GND }

Functions

- void [halAdcSetClock](#) (bool slow)
- bool [halAdcGetClock](#) (void)
- uint16_t [halMeasureVdd](#) (ADCRateType rate)

8.5.1 Detailed Description

Header for EM35x A/D converter. See [Adc](#) for documentation.

Definition in file [cortexm3/adc.h](#).

8.6 cortexm3/adc.h

```

00001
00020 #ifndef __EM3XX_ADC_H__
00021 #define __EM3XX_ADC_H__
00022
00023 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00024     #ifndef __ADC_H__
00025         #error do not include this file directly - include micro/adc.h
00026     #endif
00027 #endif
00028
00029
00033 enum {
00035     ADC_REF_INT = 42
00036 };
00037
00038
00044 enum {
00046     ADC_SAMPLE_CLOCKS_32 = 0x0,
00048     ADC_SAMPLE_CLOCKS_64 = 0x1,
00050     ADC_SAMPLE_CLOCKS_128 = 0x2,
00052     ADC_SAMPLE_CLOCKS_256 = 0x3,
00054     ADC_SAMPLE_CLOCKS_512 = 0x4,
00056     ADC_SAMPLE_CLOCKS_1024 = 0x5,
00058     ADC_SAMPLE_CLOCKS_2048 = 0x6,
00060     ADC_SAMPLE_CLOCKS_4096 = 0x7,
00061 };
00062
00063
00067 enum
00068 {
00070     ADC_CONVERSION_TIME_US_32 = 0x0,
00072     ADC_CONVERSION_TIME_US_64 = 0x1,
00074     ADC_CONVERSION_TIME_US_128 = 0x2,
00076     ADC_CONVERSION_TIME_US_256 = 0x3,
```

```

00078     ADC_CONVERSION_TIME_US_512 = 0x4,
00080     ADC_CONVERSION_TIME_US_1024 = 0x5,
00082     ADC_CONVERSION_TIME_US_2048 = 0x6,
00084     ADC_CONVERSION_TIME_US_4096 = 0x7,
00085 };
00086
00087
00088 #if defined(EMBER_TEST) && !defined(ADC_MUXN_BITS)
00089     #define ADC_MUXN_BITS (4)
00090 #endif
00091
00093 #define ADC_MUX_ADC0 0x0
00094
00095 #define ADC_MUX_ADC1 0x1
00096
00097 #define ADC_MUX_ADC2 0x2
00098
00099 #define ADC_MUX_ADC3 0x3
00100
00101 #define ADC_MUX_ADC4 0x4
00102
00103 #define ADC_MUX_ADC5 0x5
00104
00105 #define ADC_MUX_GND 0x8
00106
00107 #define ADC_MUX_VREF2 0x9
00108
00109 #define ADC_MUX_VREF 0xA
00110
00111 #define ADC_MUX_VREG2 0xB
00112
00113 // ADC_SOURCE_<pos>_<neg> selects <pos> as the positive input and <neg> as
00114 // the negative input.
00115 enum
00116 {
00117     ADC_SOURCE_ADC0_VREF2 = ((ADC_MUX_ADC0 <<
00118         ADC_MUXN_BITS) + ADC_MUX_VREF2),
00119     ADC_SOURCE_ADC0_GND = ((ADC_MUX_ADC0 <<
00120         ADC_MUXN_BITS) + ADC_MUX_GND),
00121     ADC_SOURCE_ADC1_VREF2 = ((ADC_MUX_ADC1 <<
00122         ADC_MUXN_BITS) + ADC_MUX_VREF2),
00123     ADC_SOURCE_ADC1_GND = ((ADC_MUX_ADC1 <<
00124         ADC_MUXN_BITS) + ADC_MUX_GND),
00125
00126     ADC_SOURCE_ADC2_VREF2 = ((ADC_MUX_ADC2 <<
00127         ADC_MUXN_BITS) + ADC_MUX_VREF2),
00128     ADC_SOURCE_ADC2_GND = ((ADC_MUX_ADC2 <<
00129         ADC_MUXN_BITS) + ADC_MUX_GND),
00130     ADC_SOURCE_ADC3_VREF2 = ((ADC_MUX_ADC3 <<
00131         ADC_MUXN_BITS) + ADC_MUX_VREF2),
00132     ADC_SOURCE_ADC3_GND = ((ADC_MUX_ADC3 <<
00133         ADC_MUXN_BITS) + ADC_MUX_GND),
00134
00135     ADC_SOURCE_ADC4_VREF2 = ((ADC_MUX_ADC4 <<
00136         ADC_MUXN_BITS) + ADC_MUX_VREF2),
00137     ADC_SOURCE_ADC4_GND = ((ADC_MUX_ADC4 <<
00138         ADC_MUXN_BITS) + ADC_MUX_GND),
00139     ADC_SOURCE_ADC5_VREF2 = ((ADC_MUX_ADC5 <<
00140         ADC_MUXN_BITS) + ADC_MUX_VREF2),
00141     ADC_SOURCE_ADC5_GND = ((ADC_MUX_ADC5 <<
00142         ADC_MUXN_BITS) + ADC_MUX_GND),
00143     ADC_SOURCE_GND_VREF2 = ((ADC_MUX_GND <<

```

```

00144     ADC_SOURCE_VGND      = ((ADC_MUX_GND    <<
00145         ADC_MUXN_BITS) + ADC_MUX_GND),
00146     ADC_SOURCE_VREF2     = ((ADC_MUX_VREF   <<
00147         ADC_MUXN_BITS) + ADC_MUX_VREF2),
00147     ADC_SOURCE_VREF      = ((ADC_MUX_VREF   <<
00148         ADC_MUXN_BITS) + ADC_MUX_GND),
00149     ADC_SOURCE_VREG2_VREF2 = ((ADC_MUX_VREG2 <
00150         <ADC_MUXN_BITS) + ADC_MUX_VREF2),
00150     ADC_SOURCE_VDD_GND   = ((ADC_MUX_VREG2 <<
00151         ADC_MUXN_BITS) + ADC_MUX_GND)
00151 };
00152
00153
00154 void halAdcSetClock(bool slow);
00155 bool halAdcGetClock(void);
00156
00157
00165 uint16_t halMeasureVdd(ADCRateType rate);
00166
00167
00172 #endif //__EM3XX_ADC_H__

```

8.7 ami-inter-pan-host.h File Reference

Data Structures

- struct [InterPanHeader](#)
A struct for keeping track of all of the header info.

Macros

- #define INTER_PAN_UNICAST
- #define INTER_PAN_BROADCAST
- #define INTER_PAN_MULTICAST
- #define MAX_INTER_PAN_MAC_SIZE
- #define STUB_NWK_SIZE
- #define STUB_NWK_FRAME_CONTROL
- #define MAX_STUB_APS_SIZE
- #define MAX_INTER_PAN_HEADER_SIZE

Functions

- uint8_t [makeInterPanMessage](#) (InterPanHeader *headerData, uint8_t *message, uint8_t maxLength, uint8_t *payload, uint8_t payloadLength)
- uint8_t [parseInterPanMessage](#) (uint8_t *message, uint8_t messageLength, InterPanHeader *headerData)

8.7.1 Detailed Description

Utilities for sending and receiving ZigBee AMI InterPAN messages. See [Sending and Receiving Messages](#) for documentation.

Deprecated The ami-inter-pan library is deprecated and will be removed in a future release. Similar functionality is available in the Inter-PAN plugin in Application Framework.

Definition in file [ami-inter-pan-host.h](#).

8.8 ami-inter-pan-host.h

```

00001
00019 #ifndef AMI_INTER_PAN_HOST_H
00020 #define AMI_INTER_PAN_HOST_H
00021
00028 #define INTER_PAN_UNICAST 0x03
00029 #define INTER_PAN_BROADCAST 0x0B
00030 #define INTER_PAN_MULTICAST 0x0F
00031
00032
00033 // Frame control, sequence, dest PAN ID, dest, source PAN ID, source.
00034 #define MAX_INTER_PAN_MAC_SIZE (2 + 1 + 2 + 8 + 2 + 8)
00035 //Short form has a short destination.
00036
00037 // NWK stub frame has two control bytes.
00038 #define STUB_NWK_SIZE 2
00039 #define STUB_NWK_FRAME_CONTROL 0x000B
00040
00041 // APS frame control, group ID, cluster ID, profile ID
00042 #define MAX_STUB_APS_SIZE (1 + 2 + 2 + 2)
00043
00044 // Short form has no group ID.
00045 #define MAX_INTER_PAN_HEADER_SIZE \
00046 (MAX_INTER_PAN_MAC_SIZE + STUB_NWK_SIZE + MAX_STUB_APS_SIZE)
00047
00052 typedef struct {
00053     uint8_t messageType;           // one of the INTER_PAN_...CAST values
00054
00055     // MAC addressing
00056     // For outgoing messages this is the destination. For incoming messages
00057     // it is the source, which always has a long address.
00058     uint16_t panId;
00059     bool hasLongAddress;          // always true for incoming messages
00060     EmberNodeId shortAddress;
00061     EmberEUI64 longAddress;
00062
00063     // APS data
00064     uint16_t profileId;
00065     uint16_t clusterId;
00066     uint16_t groupId;            // only used for INTER_PAN_MULTICAST
00067 } InterPanHeader;
00068
00075 uint8_t makeInterPanMessage(InterPanHeader *
    headerData,
00076                               uint8_t *message,
00077                               uint8_t maxLength,
00078                               uint8_t *payload,
00079                               uint8_t payloadLength);
00080
00088 uint8_t parseInterPanMessage(uint8_t *message,
00089                               uint8_t messageLength,
00090                               InterPanHeader *headerData);
00091
00092 #endif // AMI_INTER_PAN_HOST_H
00093

```

8.9 ami-inter-pan.h File Reference

Data Structures

- **struct InterPanHeader**

A struct for keeping track of all of the header info.

Macros

- #define INTER_PAN_UNICAST
- #define INTER_PAN_BROADCAST
- #define INTER_PAN_MULTICAST
- #define MAX_INTER_PAN_MAC_SIZE
- #define STUB_NWK_SIZE
- #define STUB_NWK_FRAME_CONTROL
- #define MAX_STUB_APS_SIZE
- #define MAX_INTER_PAN_HEADER_SIZE

Functions

- EmberMessageBuffer makeInterPanMessage (InterPanHeader *headerData, EmberMessageBuffer payload)
- uint8_t parseInterPanMessage (EmberMessageBuffer message, uint8_t startOffset, InterPanHeader *headerData)

8.9.1 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](#).

8.10 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_STUBAPS_SIZE)
00047
00051 typedef struct {
00052     uint8_t messageType;           // one of the INTER_PAN_...CAST
00053     values
00054     // MAC addressing
00055     // For outgoing messages this is the destination. For incoming messages
00056     // it is the source, which always has a long address.
00057     uint16_t panId;
00058     bool hasLongAddress;         // always true for incoming messages
00059     EmberNodeId shortAddress;
00060     EmberEUI64 longAddress;
00061
00062     // APS data
00063     uint16_t profileId;
00064     uint16_t clusterId;
00065     uint16_t groupId;           // only used for INTER_PAN_MULTICAST
00066 } InterPanHeader;
00067
00068
00072 EmberMessageBuffer makeInterPanMessage(
00073     InterPanHeader *headerData,
00074                                     EmberMessageBuffer
00075     payload);
00074
00082 uint8_t parseInterPanMessage(EmberMessageBuffer
00083     message,
00084                                     uint8_t startOffset,
00085                                     InterPanHeader *headerData);
00086 #endif // AMI_INTER_PAN_H
00087

```

8.11 app-bootloader.h File Reference

Required Custom Functions

- void [bootloaderAction](#) (bool 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](#) (bool install)

8.11.1 Detailed Description

See [Application](#) for detailed documentation.

Definition in file [app-bootloader.h](#).

8.12 app-bootloader.h

```

00001
00014 #ifndef __APP_BOOTLOADER_H__
00015 #define __APP_BOOTLOADER_H__
00016
00018
00029 void bootloaderAction(bool runRecovery);
00032

```

```

00033
00042 BL_Status recoveryMode(void);
00043
00049 BL_Status processImage(bool install);
00053 #endif //__APP_BOOTLOADER_H__
00054

```

8.13 binding-table.h File Reference

Functions

- EmberStatus emberSetBinding (uint8_t index, EmberBindingTableEntry *value)
- EmberStatus emberGetBinding (uint8_t index, EmberBindingTableEntry *result)
- EmberStatus emberDeleteBinding (uint8_t index)
- bool emberBindingIsActive (uint8_t index)
- EmberNodeId emberGetBindingRemoteNodeId (uint8_t index)
- void emberSetBindingRemoteNodeId (uint8_t index, EmberNodeId id)
- EmberStatus emberClearBindingTable (void)
- EmberStatus emberRemoteSetBindingHandler (EmberBindingTableEntry *entry)
- EmberStatus emberRemoteDeleteBindingHandler (uint8_t index)
- uint8_t emberGetBindingIndex (void)
- EmberStatus emberSetReplyBinding (uint8_t index, EmberBindingTableEntry *entry)
- EmberStatus emberNoteSendersBinding (uint8_t index)

8.13.1 Detailed Description

See [Binding Table](#) for documentation.

Definition in file [binding-table.h](#).

8.14 binding-table.h

```

00001
00027 EmberStatus emberSetBinding(uint8_t index,
                                EmberBindingTableEntry *value);
00028
00040 EmberStatus emberGetBinding(uint8_t index,
                                 EmberBindingTableEntry *result);
00041
00049 EmberStatus emberDeleteBinding(uint8_t index);
00050
00063 bool emberBindingIsActive(uint8_t index);
00064
00086 EmberNodeId emberGetBindingRemoteNodeId(
    uint8_t index);
00087
00095 void emberSetBindingRemoteNodeId(uint8_t index,
                                     EmberNodeId id);
00096
00102 EmberStatus emberClearBindingTable(void);
00103
00126 EmberStatus emberRemoteSetBindingHandler
    (EmberBindingTableEntry *entry);
00127
00145 EmberStatus emberRemoteDeleteBindingHandler
    (uint8_t index);
00146
00167 uint8_t emberGetBindingIndex(void);
00168
00186 EmberStatus emberSetReplyBinding(uint8_t index,

```

```

        EmberBindingTableEntry *entry);
00187
00199 EmberStatus emberNoteSendersBinding(uint8_t
index);
00200
00201 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00202 // This is defined in hal/ember-configuration.c.
00203 extern uint8_t emberBindingTableSize;
00204 #endif
00205

```

8.15 bootload.h File Reference

Functions

- `EmberStatus emberSendBootloadMessage (bool broadcast, EmberEUI64 destEui64, EmberMessageBuffer message)`
- `void emberIncomingBootloadMessageHandler (EmberEUI64 longId, EmberMessageBuffer message)`
- `void emberBootloadTransmitCompleteHandler (EmberMessageBuffer message, EmberStatus status)`

8.15.1 Detailed Description

See [Bootloader](#) for documentation.

Definition in file [bootload.h](#).

8.16 bootload.h

```

00001
00030 EmberStatus emberSendBootloadMessage(bool
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

```

8.17 bootloader-eeprom.h File Reference

Data Structures

- struct [HalEepromInformationType](#)

This structure defines a variety of information about the attached external EEPROM device.

Macros

- #define EEPROM_PAGE_SIZE
- #define EEPROM_FIRST_PAGE
- #define EEPROM_IMAGE_START
- #define EEPROM_SUCCESS
- #define EEPROM_ERR
- #define EEPROM_ERR_MASK
- #define EEPROM_ERR_PG_BOUNDARY
- #define EEPROM_ERR_PG_SZ
- #define EEPROM_ERR_WRT_DATA
- #define EEPROM_ERR_IMG_SZ
- #define EEPROM_ERR_ADDR
- #define EEPROM_ERR_INVALID_CHIP
- #define EEPROM_ERR_ERASE_REQUIRED
- #define EEPROM_ERR_NO_ERASE_SUPPORT

EEPROM interaction functions.

- #define EEPROM_INFO_VERSION
- #define EEPROM_INFO_MAJOR_VERSION
- #define EEPROM_INFO_MAJOR_VERSION_MASK
- #define EEPROM_INFO_MIN_VERSION_WITH_WORD_SIZE_SUPPORT
- #define EEPROM_CAPABILITIES_ERASE_SUPPORTED
- #define EEPROM_CAPABILITIES_PAGE_ERASE_REQD
- #define EEPROM_CAPABILITIES_BLOCKING_WRITE
- #define EEPROM_CAPABILITIES_BLOCKING_ERASE
- uint8_t halEepromInit (void)
- void halEepromShutdown (void)
- const HalEepromInformationType * halEepromInfo (void)
- uint32_t halEepromSize (void)
- bool halEepromBusy (void)
- uint8_t halEepromRead (uint32_t address, uint8_t *data, uint16_t len)
- uint8_t halEepromWrite (uint32_t address, const uint8_t *data, uint16_t len)
- uint8_t halEepromErase (uint32_t address, uint32_t totalLength)

8.17.1 Detailed Description

See [Application](#) for detailed documentation.

Definition in file [bootloader-eeprom.h](#).

8.18 bootloader-eeprom.h

```

00001
00018 #ifndef __BOOTLOADER_EEPROM_H__
00019 #define __BOOTLOADER_EEPROM_H__
00020
00024 #define EEPROM_PAGE_SIZE      (128ul)
00025
00029 #define EEPROM_FIRST_PAGE    (0)
00030

```

```

00035 #define EEPROM_IMAGE_START  (EEPROM_FIRST_PAGE*EEPROM_PAGE_SIZE)
00036
00039 #define EEPROM_SUCCESS 0
00040
00043 #define EEPROM_ERR 1
00044
00047 #define EEPROM_ERR_MASK 0x80
00048
00051 #define EEPROM_ERR_PG_BOUNDARY 0x81
00052
00055 #define EEPROM_ERR_PG_SZ 0x82
00056
00059 #define EEPROM_ERR_WRT_DATA 0x83
00060
00063 #define EEPROM_ERR_IMG_SZ 0x84
00064
00067 #define EEPROM_ERR_ADDR 0x85
00068
00071 #define EEPROM_ERR_INVALID_CHIP 0x86
00072
00075 #define EEPROM_ERR_ERASE_REQUIRED 0x87
00076
00079 #define EEPROM_ERR_NO_ERASE_SUPPORT 0x88
00080
00091 uint8_t halEepromInit(void);
00092
00095 void halEepromShutdown(void);
00096
00100 typedef struct {
00101     uint16_t version;
00104     uint16_t capabilitiesMask;
00106     uint16_t pageEraseMs;
00108     uint16_t partEraseMs;
00110     uint32_t pageSize;
00112     uint32_t partSize;
00114     const char * const partDescription;
00116     uint8_t wordSizeBytes;
00117 } HalEepromInformationType;
00118
00121 #define EEPROM_INFO_VERSION          (0x0102)
00122 #define EEPROM_INFO_MAJOR_VERSION    (0x0100)
00123 #define EEPROM_INFO_MAJOR_VERSION_MASK (0xFF00)
00124 // *** Eeprom info version history: ***
00125 // 0x0102 - Added a word size field to specify the number of bytes per flash
00126 //           word in the EEPROM. Writes should always be aligned to the word
00127 //           size and have a length that is a multiple of the word size.
00128 // 0x0101 - Initial version
00129 #define EEPROM_INFO_MIN_VERSION_WITH_WORD_SIZE_SUPPORT 0x0102
00130
00133 #define EEPROM_CAPABILITIES_ERASE_SUPPORTED (0x0001)
00134
00138 #define EEPROM_CAPABILITIES_PAGE_ERASE_REQD (0x0002)
00139
00143 #define EEPROM_CAPABILITIES_BLOCKING_WRITE (0x0004)
00144
00148 #define EEPROM_CAPABILITIES_BLOCKING_ERASE (0x0008)
00149
00159 const HalEepromInformationType *halEepromInfo
00160 (void);
00160
00169 uint32_t halEepromSize(void);
00170
00179 bool halEepromBusy(void);
00180
00196 uint8_t halEepromRead(uint32_t address, uint8_t *data, uint16_t
00197 len);
00197
00219 uint8_t halEepromWrite(uint32_t address, const uint8_t *data,
00220 uint16_t len);
00220
00241 uint8_t halEepromErase(uint32_t address, uint32_t totalLength);
00242
00243
00244 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00245
00246 // This structure holds information about where we left off when last accessing
00247 // the eeprom.
00248 typedef struct {
00249     uint32_t address;
00250     uint16_t pages;

```

```

00251     uint16_t pageBufFinger;
00252     uint16_t pageBufLen;
00253     uint8_t pageBuf[EEPROM_PAGE_SIZE];
00254 } EepromStateType;
00255
00256 #endif //DOXYGEN_SHOULD_SKIP_THIS
00257
00260 #endif //__BOOTLOADER_EEPROM_H__
00261

```

8.19 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)
- bool [bootloadForceActivation](#) (void)

State Indicator Macros

The bootloader indicates which state it is in by calling these macros. Map them to the ::hal-BootloadStateIndicator 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()

8.19.1 Detailed Description

See [GPIO](#) for detailed documentation.

Definition in file [bootloader-gpio.h](#).

8.20 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 { bootloaderStateIndicator(BL_ST_DOWN); } while(0)
00035
00038 #define BL_STATE_POLLING_LOOP() do {
00039     bootloaderStateIndicator(BL_ST_POLLING_LOOP); } while(0)
00040
00042 #define BL_STATE_DOWNLOAD_LOOP() do {
00043     bootloaderStateIndicator(BL_ST_DOWNLOAD_LOOP); } while(0)
00044
00046 #define BL_STATE_DOWNLOAD_SUCCESS() do {
00047     bootloaderStateIndicator(BL_ST_DOWNLOAD_SUCCESS); } while(0)
00048
00050 #define BL_STATE_DOWNLOAD_FAILURE() do {
00051     bootloaderStateIndicator(BL_ST_DOWNLOAD_FAILURE); } while(0)
00052
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 bootloaderGpioInit(void);
00074
00078 void bootloaderStateIndicator(enum blState_e state
00079 );
00082 bool bootloaderForceActivation( void );
00083
00084 #endif // __BOOTLOADER_GPIO_H__
00085

```

8.21 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

- uint8_t halAppBootloaderInit (void)
- const HalEepromInformationType * halAppBootloaderInfo (void)
- void halAppBootloaderShutdown (void)
- void halAppBootloaderImageIsValidReset (void)
- uint16_t halAppBootloaderImageIsValid (void)
- EmberStatus halAppBootloaderInstallNewImage (void)
- uint8_t halAppBootloaderWriteRawStorage (uint32_t address, const uint8_t *data, uint16_t len)
- uint8_t halAppBootloaderReadRawStorage (uint32_t address, uint8_t *data, uint16_t len)
- uint8_t halAppBootloaderEraseRawStorage (uint32_t address, uint32_t len)
- bool halAppBootloaderStorageBusy (void)
- uint8_t halAppBootloaderReadDownloadSpace (uint16_t pageToBeRead, uint8_t *dest-RamBuffer)

- `uint8_t halAppBootloaderWriteDownloadSpace (uint16_t pageToBeWritten, uint8_t *RamPtr)`
- `uint8_t halAppBootloaderGetImageData (uint32_t *timestamp, uint8_t *userData)`
- `uint16_t halAppBootloaderGetVersion (void)`
- `uint16_t halAppBootloaderGetRecoveryVersion (void)`

8.21.1 Detailed Description

See [Application](#) for documentation.

Definition in file [bootloader-interface-app.h](#).

8.22 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 uint8_t halAppBootloaderInit(void);
00043
00050 const HalEepromInformationType *halAppBootloaderInfo
00051   (void);
00055 void halAppBootloaderShutdown(void);
00056
00060 void halAppBootloaderImageIsValidReset(void);
00061
00065 #define BL_IMAGE_IS_VALID_CONTINUE ((uint16_t)0xFFFF)
00066
00092 uint16_t halAppBootloaderImageIsValid(void);
00093
00094
00099 EmberStatus halAppBootloaderInstallNewImage
00100   (void);
00101
00102
00123 uint8_t halAppBootloaderWriteRawStorage(uint32_t
00124   address,
00125           const uint8_t *data,
00126           uint16_t len);
00141 uint8_t halAppBootloaderReadRawStorage(uint32_t
00142   address, uint8_t *data, uint16_t len);
00160 uint8_t halAppBootloaderEraseRawStorage(uint32_t
00161   address, uint32_t len);
00167 bool halAppBootloaderStorageBusy(void);
00168
00181 uint8_t halAppBootloaderReadDownloadSpace(
00182   uint16_t pageToBeRead,
00183           uint8_t* destRamBuffer);
00184
00197 uint8_t halAppBootloaderWriteDownloadSpace(
00198   uint16_t pageToBeWritten,
00199           uint8_t* RamPtr);
00208 uint8_t halAppBootloaderGetImageData(uint32_t *
00209   timestamp, uint8_t *userData);
00209
00210

```

```

00213 uint16_t halAppBootloaderGetVersion(void);
00214
00215
00216 uint16_t halAppBootloaderGetRecoveryVersion(
00217     void);
00218
00219
00220
00221 #endif //__BOOTLOADER_INTERFACE_APP_H__
00222

```

8.23 bootloader-interface-standalone.h File Reference

Macros

- `#define NO_BOOTLOADER_MODE`
- `#define STANDALONE_BOOTLOADER_NORMAL_MODE`
- `#define STANDALONE_BOOTLOADER_RECOVERY_MODE`

Functions

- `uint16_t halGetStandaloneBootloaderVersion (void)`
- `EmberStatus halLaunchStandaloneBootloader (uint8_t mode)`

8.23.1 Detailed Description

See [Standalone](#) for documentation.

Definition in file [bootloader-interface-standalone.h](#).

8.24 bootloader-interface-standalone.h

```

00001
00002 #ifndef __BOOTLOADER_STANDALONE_H__
00003 #define __BOOTLOADER_STANDALONE_H__
00004
00005
00006 uint16_t halGetStandaloneBootloaderVersion(
00007     void);
00008
00009
00010 #define NO_BOOTLOADER_MODE          0xFF
00011
00012 #define STANDALONE_BOOTLOADER_NORMAL_MODE    1
00013
00014 #define STANDALONE_BOOTLOADER_RECOVERY_MODE 0
00015
00016 EmberStatus halLaunchStandaloneBootloader
00017     (uint8_t mode);
00018
00019
00020
00021 #endif //__BOOTLOADER_STANDALONE_H__
00022

```

8.25 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_SERIAL_USB
- #define BL_EXT_TYPE_SERIAL_USB_OTA
- #define BL_EXT_TYPE_APP_UNKNOWN
- #define BL_EXT_TYPE_APP_SPI
- #define BL_EXT_TYPE_APP_I2C
- #define BL_EXT_TYPE_APP_LOCAL_STORAGE
- #define BOOTLOADER_INVALID_VERSION

Functions

- BI BaseType halBootloaderGetType (void)
- BI ExtendedType halBootloaderGetInstalledType (void)
- uint16_t halGetBootloaderVersion (void)
- void halGetExtendedBootloaderVersion (uint32_t *emberVersion, uint32_t *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 uint8_t BI BaseType
- typedef uint16_t BI ExtendedType

8.25.1 Detailed Description

See [Common](#) for detailed documentation.

Definition in file [bootloader-interface.h](#).

8.26 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 uint8_t BlBaseType;
00044 typedef uint16_t BlExtendedType;
00052 BlBaseType halBootloaderGetType(void);
00053
00054
00058 #define BOOTLOADER_BASE_TYPE(extendedType) \
00059             ((uint8_t)((extendedType) >> 8) & 0xFF)
00060
00064 #define BOOTLOADER_MAKE_EXTENDED_TYPE(baseType, extendedSpecifier) \
00065             ((uint16_t)((uint16_t)baseType) << 8) | \
00066             ((uint16_t)extendedSpecifier)&0xFF)
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_SERIAL_USB   ((BL_TYPE_STANDALONE << 8) | 0x07)
00091
00094 #define BL_EXT_TYPE_SERIAL_USB_OTA ((BL_TYPE_STANDALONE << 8) | 0x08)
00095
00096
00099 #define BL_EXT_TYPE_APP_UNKNOWN  ((BL_TYPE_APPLICATION << 8) | 0x00)
00100
00103 #define BL_EXT_TYPE_APP_SPI     ((BL_TYPE_APPLICATION << 8) | 0x01)
00104
00107 #define BL_EXT_TYPE_APP_I2C     ((BL_TYPE_APPLICATION << 8) | 0x02)
00108
00111 #define BL_EXT_TYPE_APP_LOCAL_STORAGE ((BL_TYPE_APPLICATION << 8) | 0x03)
00112
00116 BlExtendedType halBootloaderGetInstalledType
00117 (void);
00118
00119
00122 #define BOOTLOADER_INVALID_VERSION 0xFFFF
00123
00130 uint16_t halGetBootloaderVersion(void);
00131
00132
00143 void halGetExtendedBootloaderVersion(uint32_t*
00144     emberVersion, uint32_t* customerVersion);
00145 #endif // __BOOTLOADER_INTERFACE_H__
00146

```

8.27 bootloader-serial.h File Reference

Functions

- void `serInit` (void)

- void [serPutFlush](#) (void)
- void [serPutChar](#) (uint8_t ch)
- void [serPutStr](#) (const char *str)
- void [serPutBuf](#) (const uint8_t buf[], uint8_t size)
- void [serPutDecimal](#) (uint16_t val)
- void [serPutHex](#) (uint8_t byte)
- void [serPutHexInt](#) (uint16_t word)
- bool [serCharAvailable](#) (void)
- uint8_t [serGetChar](#) (uint8_t *ch)
- void [serGetFlush](#) (void)

8.27.1 Detailed Description

See [Serial](#) for detailed documentation.

Definition in file [bootloader-serial.h](#).

8.28 bootloader-serial.h

```

00001
00002 //[[ Author(s): David Iacobone, diacobone@ember.com
00003 //                  Lee Taylor, lee@ember.com
00004 //]]
00010
00018 #ifndef __BOOTLOADER_SERIAL_H__
00019 #define __BOOTLOADER_SERIAL_H__
00020
00021
00022 #ifndef MSD_BOOTLOADER
00023
00025 void serInit(void);
00026
00029 void serPutFlush(void);
00033 void serPutChar(uint8_t ch);
00037 void serPutStr(const char *str);
00042 void serPutBuf(const uint8_t buf[], uint8_t size);
00046 void serPutDecimal(uint16_t val);
00050 void serPutHex(uint8_t byte);
00054 void serPutHexInt(uint16_t word);
00055
00059 bool serCharAvailable(void);
00064 uint8_t serGetChar(uint8_t* ch);
00065
00068 void serGetFlush(void);
00069
00070 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00071 #ifdef BTL_HAS_EZSP_SPI
00072 extern uint8_t preSpiComBufIndex;
00073 #define serResetCharInput() preSpiComBufIndex = 0;
00074 #endif
00075 #endif
00076
00077 #else //MSD_BOOTLOADER
00078
00079 // clear serial calls
00080 #define serInit(x) {}
00081 #define serPutFlush(x)
00082 #define serPutChar(x)
00083 #define serPutStr(x)
00084 #define serPutBuf(x)
00085 #define serPutDecimal(x)
00086 #define serPutHex(x) x
00087 #define serPutHexInt(x)
00088 #define serCharAvailable(x)
00089 #define serGetChar(x)
00090 #define serGetFlush(x)
00091

```

```

00092 #endif //MSD_BOOTLOADER
00093
00094 #endif //__BOOTLOADER_SERIAL_H__
00095

```

8.29 button.h File Reference

Functions

- void [halInternalInitButton](#) (void)
- uint8_t [halButtonState](#) (uint8_t button)
- uint8_t [halButtonPinState](#) (uint8_t button)
- void [halButtonIsr](#) (uint8_t button, uint8_t state)

Button State Definitions

A set of numerical definitions for use with the button APIs indicating the state of a button.

- #define [BUTTON_PRESSED](#)
- #define [BUTTON_RELEASED](#)

8.29.1 Detailed Description

See [Button Control](#) for documentation.

Definition in file [button.h](#).

8.30 button.h

```

00001
00024 #define BUTTON_PRESSED 1
00025
00028 #define BUTTON_RELEASED 0
00029
00036 void halInternalInitButton(void);
00037
00049 uint8_t halButtonState(uint8_t button);
00050
00063 uint8_t halButtonPinState(uint8_t button);
00064
00079 void halButtonIsr(uint8_t button, uint8_t state);
00080

```

8.31 buzzer.h File Reference

Functions

- void [halPlayTune_P](#) (uint8_t PGM *tune, bool 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

8.31.1 Detailed Description

See [Buzzer Control](#) for documentation.

Definition in file [buzzer.h](#).

8.32 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
00083 void halPlayTune_P(uint8_t PGM *tune, bool bkg);
00084
00085
00090 void halStackIndicatePresence(void);
00091

```

8.33 child.h File Reference

Functions

- `EmberNodeId emberChildId (uint8_t childIndex)`
- `uint8_t emberChildIndex (EmberNodeId childId)`
- `EmberStatus emberGetChildData (uint8_t index, EmberEUI64 childEui64Return, EmberNodeType *childTypeReturn)`
- `void emberChildJoinHandler (uint8_t index, bool joining)`
- `EmberStatus emberPollForData (void)`
- `void emberPollCompleteHandler (EmberStatus status)`
- `EmberStatus emberSetMessageFlag (EmberNodeId childId)`
- `EmberStatus emberClearMessageFlag (EmberNodeId childId)`
- `void emberPollHandler (EmberNodeId childId, bool transmitExpected)`
- `uint8_t emberChildCount (void)`
- `uint8_t emberRouterChildCount (void)`
- `uint8_t emberMaxChildCount (void)`
- `uint8_t 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_APSS_LAYER_ACTIVE, EMBER_ASSOCIATING, EMBER_ZLL_TOUCH_LINKING, EMBER_RF4CE_NETWORK_BUSY,
 EMBER_RF4CE_DUTY_CYCLING_ENABLED, EMBER_NETWORK_TIMEOUT_REQUEST, EMBER_SEND_ORPHAN_NOTIFICATION
 }
- uint16_t emberCurrentStackTasks (void)
- bool emberOkToNap (void)
- bool emberOkToHibernate (void)
- bool emberOkToLongPoll (void)
- void emberStackPowerDown (void)
- void emberStackPowerUp (void)

8.33.1 Detailed Description

See [End Devices](#) for documentation.

Definition in file [child.h](#).

8.34 child.h

```

00001
00023 EmberNodeId emberChildId(uint8_t childIndex);
00024
00031 uint8_t emberChildIndex(EmberNodeId childId);
00032
00047 EmberStatus emberGetChildData(uint8_t index,
00048                                     EmberEUI64 childEui64Return,
00049                                     EmberNodeType *childTypeReturn);
00050
00066 void emberChildJoinHandler(uint8_t index, bool 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, bool
transmitExpected);
00163
00164
00165 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00166
00170 uint8_t emberChildCount(void);
00171
00177 uint8_t emberRouterChildCount(void);
00178
00184 uint8_t emberMaxChildCount(void);
00185
00191 uint8_t emberMaxRouterChildCount(void);
00192
00199 EmberNodeId emberGetParentNodeId(void);
00200
00207 EmberEUI64 emberGetParentEui64(void);
00208

```

```

00209 #else // Doxygen ignores the following
00210
00211 extern uint8_t emMaxEndDeviceChildren; // maximum for this node
00212 extern uint8_t emEndDeviceChildCount; // how many we have
00213 extern bool emEnableR21StackBehavior; //Enabling R21 behavior.
00214 extern uint8_t emDefaultTimeoutValue; // Default Timeout Enumeration.
00215
00216 // The '+ 0' prevents anyone from accidentally assigning to these.
00217 #define emberChildCount() (emEndDeviceChildCount + 0)
00218 #define emberRouterChildCount() 0
00219 #define emberMaxChildCount() (emMaxEndDeviceChildren + 0)
00220 #define emberMaxRouterChildCount() 0
00221
00222 // Implemented in ember-stack-common.c
00223 EmberNodeId emberGetParentNodeId(void);
00224 uint8_t* emberGetParentEui64(void);
00225
00226
00227 #endif
00228
00229
00230
00231
00232
00233 enum
00234 {
00235     EMBER_OUTGOING_MESSAGES = 0x0001,
00236     EMBER_INCOMING_MESSAGES = 0x0002,
00237     EMBER_RADIO_IS_ON = 0x0004,
00238     EMBER_TRANSPORT_ACTIVE = 0x0008,
00239     EMBERAPS_LAYER_ACTIVE = 0x0010,
00240     EMBER_ASSOCIATING = 0x0020,
00241     EMBER_ZLL_TOUCH_LINKING = 0x0040,
00242     EMBER_RF4CE_NETWORK_BUSY = 0x0080,
00243     EMBER_RF4CE_DUTY_CYCLING_ENABLED = 0x0100,
00244     EMBER_NETWORK_TIMEOUT_REQUEST = 0x0200,
00245     EMBER_SEND_ORPHAN_NOTIFICATION = 0x0400,
00246 };
00247
00248 #define EMBER_HIGH_PRIORITY_TASKS
00249 \
00250 (EMBER_OUTGOING_MESSAGES | EMBER_INCOMING_MESSAGES | EMBER_RADIO_IS_ON
00251 \
00252 | EMBER_RF4CE_NETWORK_BUSY)
00253
00254 uint16_t emberCurrentStackTasks(void);
00255
00256 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00257 bool emberOkToNap(void);
00258 #else
00259 #define emberOkToNap() \
00260 (! (emberCurrentStackTasks() & EMBER_HIGH_PRIORITY_TASKS))
00261 #endif
00262
00263 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00264 bool emberOkToHibernate(void);
00265 #else
00266 #define emberOkToHibernate() (! emberCurrentStackTasks())
00267 #endif
00268
00269 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00270 bool emberOkToLongPoll(void);
00271 #else
00272 #define emberOkToLongPoll() (! emberPendingAckedMessages())
00273 #endif
00274
00275 void emberStackPowerDown(void);
00276
00277 void emberStackPowerUp(void);
00278
00279 // @} END Power Management
00280
00281 // @} END addtogroup
00282

```

8.35 command-interpreter2.h File Reference

Data Structures

- struct `EmberCommandEntry`
Command entry for a command table.

Macros

- #define MAX_TOKEN_COUNT
- #define emberCommandEntryAction(name, action, argumentTypes, description)
- #define emberCommandEntryActionWithDetails(name, action, argumentTypes, description, argumentDescriptionArray)
- #define emberCommandEntrySubMenu(name, subMenu, description)
- #define emberCommandEntryTerminator()
- #define EMBER_COMMAND_INTERPRETER_CONFIGURATION_ECHO
- #define emberProcessCommandInput(port)
- #define emberCommandInterpreterEchoOn()
- #define emberCommandInterpreterEchoOff()
- #define emberCommandInterpreterIsEchoOn()

TypeDefs

- typedef void(* `CommandAction`)(void)

Enumerations

- enum `EmberCommandStatus` {

`EMBER_CMD_SUCCESS, EMBER_CMD_ERR_PORT_PROBLEM, EMBER_CMD_ERR_NO_SUCH_COMMAND, EMBER_CMD_ERR_WRONG_NUMBER_OF_ARGUMENTS,`

`EMBER_CMD_ERR_ARGUMENT_OUT_OF_RANGE, EMBER_CMD_ERR_ARGUMENT_SYNTAX_ERROR, EMBER_CMD_ERR_STRING_TOO_LONG, EMBER_CMD_ERR_INVALID_ARGUMENT_TYPE }`

Functions

- void `emberCommandReaderSetDefaultBase` (uint8_t base)
- void `emberCommandActionHandler` (const `CommandAction` action)
- void `emberCommandErrorHandler` (`EmberCommandStatus` status)
- void `emberPrintCommandUsage` (`EmberCommandEntry` *entry)
- void `emberPrintCommandUsageNotes` (void)
- void `emberPrintCommandTable` (void)
- void `emberCommandClearBuffer` (void)
- void `emberCommandReaderInit` (void)
- bool `emberProcessCommandString` (uint8_t *input, uint8_t sizeOrPort)

Variables

- `EmberCommandEntry` * `emberCurrentCommand`
- `EmberCommandEntry` `emberCommandTable` []
- uint8_t `emberCommandInterpreter2Configuration`

Command Table Settings

- #define EMBER_MAX_COMMAND_ARGUMENTS
- #define EMBER_COMMAND_BUFFER_LENGTH
- #define EMBER_COMMAND_INTERPRETER_HAS_DESCRIPTION_FIELD

Functions to Retrieve Arguments

Use the following functions in your functions that process commands to retrieve arguments from the command interpreter. These functions pull out unsigned integers, signed integers, and strings, and hex strings. Index 0 is the first command argument.

- #define emberCopyKeyArgument(index, keyDataPointer)
- #define emberCopyEui64Argument(index, eui64)
- #define emberGetEui64Argument(index, eui64)
- uint8_t emberCommandArgumentCount (void)
- uint32_t emberUnsignedCommandArgument (uint8_t argNum)
- int32_t emberSignedCommandArgument (uint8_t argNum)
- bool emberStringToHostOrderIpv4Address (const uint8_t *string, uint32_t *hostOrderIpv4Address)
- bool emberStringArgumentToHostOrderIpv4Address (uint8_t argNum, uint32_t *hostOrderIpv4Address)
- uint8_t * emberStringCommandArgument (int8_t argNum, uint8_t *length)
- const char * emberCommandName (void)
- uint8_t emberCopyStringArgument (int8_t argNum, uint8_t *destination, uint8_t maxLength, bool leftPad)
- uint8_t emberCopyBigEndianEui64Argument (int8_t index, EmberEUI64 destination)

8.35.1 Detailed Description

Processes commands coming from the serial port. See [Command Interpreter](#) for documentation.

Definition in file [command-interpreter2.h](#).

8.36 command-interpreter2.h

```

00001
00010 #ifndef __COMMAND_INTERPRETER2_H__
00011 #define __COMMAND_INTERPRETER2_H__
00012
00100 #ifndef EMBER_MAX_COMMAND_ARGUMENTS
00101
00104 #define EMBER_MAX_COMMAND_ARGUMENTS 16
00105 #endif
00106
00107 #ifndef EMBER_COMMAND_BUFFER_LENGTH
00108 #define EMBER_COMMAND_BUFFER_LENGTH 100
00109 #endif
00110
00115 #if defined(DOXYGEN_SHOULD_SKIP_THIS)
00116 #define EMBER_COMMAND_INTERPRETER_HAS_DESCRIPTION_FIELD
00117 #endif
00118
00122 // The (+ 1) takes into account the leading command.

```

```

00123 #define MAX_TOKEN_COUNT (EMBER_MAX_COMMAND_ARGUMENTS + 1)
00124
00125 typedef void (*CommandAction)(void);
00126
00127 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00128
00129 typedef struct {
00130 #else
00131     typedef PGM struct {
00132 #endif
00133
00134     PGM_P name;
00135     CommandAction action;
00136     PGM_P argumentTypes;
00137 #if defined(EMBER_COMMAND_INTEPRETER_HAS_DESCRIPTION_FIELD)
00138     PGM_P description;
00139
00140     PGM_P const * argumentDescriptions;
00141 #endif
00142 } EmberCommandEntry;
00143
00144
00145 #if defined(EMBER_COMMAND_INTEPRETER_HAS_DESCRIPTION_FIELD)
00146 /* @brief Macro to define a CLI action */
00147 #define emberCommandEntryAction(name, action, argumentTypes, description) \
00148     { (name), (action), (argumentTypes), (description), NULL }
00149
00150 #define emberCommandEntryActionWithDetails(name, \
00151                                         action, \
00152                                         argumentTypes, \
00153                                         description, \
00154                                         argumentDescriptionArray) \
00155     { (name), (action), (argumentTypes), (description), \
00156      (argumentDescriptionArray) }
00157
00158 /* @brief Macro to define a CLI sub-menu (nested command) */
00159 #define emberCommandEntrySubMenu(name, subMenu, description) \
00160     { (name), NULL, (PGM_P)(subMenu), (description), NULL }
00161
00162 /* @brief Macro to define a command entry array terminator.*/
00163 #define emberCommandEntryTerminator() \
00164     { NULL, NULL, NULL, NULL, NULL }
00165
00166 #else // Don't include description data in struct
00167
00168 /* @brief Macro to define a CLI action */
00169 #define emberCommandEntryAction(name, action, argumentTypes, description) \
00170     { (name), (action), (argumentTypes) }
00171
00172 #define emberCommandEntryActionWithDetails(name, \
00173                                         action, \
00174                                         argumentTypes, \
00175                                         description, \
00176                                         argumentDescriptionArray) \
00177     { (name), (action), (argumentTypes) }
00178
00179 /* @brief Macro to define a CLI sub-menu (nested command) */
00180 #define emberCommandEntrySubMenu(name, subMenu, description) \
00181     { (name), NULL, (PGM_P)(subMenu) }
00182
00183 /* @brief Macro to define a command entry array terminator.*/
00184 #define emberCommandEntryTerminator() \
00185     { NULL, NULL, NULL }
00186
00187 #endif
00188
00189 extern EmberCommandEntry *emberCurrentCommand
00190 ;
00191
00192 extern EmberCommandEntry emberCommandTable[];
00193
00194 extern uint8_t emberCommandInterpreter2Configuration
00195 ;
00196
00197 #define EMBER_COMMAND_INTERPRETER_CONFIGURATION_ECHO (0x01)
00198
00199 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00200
00201 enum EmberCommandStatus
00202
00203 #else

```

```

00254 typedef uint8_t EmberCommandStatus;
00255 enum
00256 #endif
00257 {
00258     EMBER_CMD_SUCCESS,
00259     EMBER_CMD_ERR_PORT_PROBLEM,
00260     EMBER_CMD_ERR_NO_SUCH_COMMAND,
00261     EMBER_CMD_ERR_WRONG_NUMBER_OF_ARGUMENTS
00262     ,
00263     EMBER_CMD_ERR_ARGUMENT_OUT_OF_RANGE,
00264     EMBER_CMD_ERR_STRING_TOO_LONG,
00265     EMBER_CMD_ERR_INVALID_ARGUMENT_TYPE
00266 };
00267
00277 uint8_t emberCommandArgumentCount(void);
00278
00280 uint32_t emberUnsignedCommandArgument(uint8_t
00281     argNum);
00283 int32_t emberSignedCommandArgument(uint8_t argNum);
00284
00288 bool emberStringToHostOrderIpv4Address(const
00289     uint8_t* string, uint32_t* hostOrderIpv4Address);
00293 bool emberStringArgumentToHostOrderIpv4Address
00294     (uint8_t argNum, uint32_t* hostOrderIpv4Address);
00295
00306 uint8_t *emberStringCommandArgument(int8_t argNum,
00307     uint8_t *length);
00308 const char *emberCommandName(void);
00309
00322 uint8_t emberCopyStringArgument(int8_t argNum,
00323             uint8_t *destination,
00324             uint8_t maxLength,
00325             bool leftPad);
00326
00330 #define emberCopyKeyArgument(index, keyDataPointer) \
00331     (emberCopyStringArgument((index),
00332             emberKeyContents((keyDataPointer)), \
00333             EMBER_ENCRYPTION_KEY_SIZE, \
00334             true))
00335
00337 #define emberCopyEui64Argument(index, eui64) \
00338     (emberCopyStringArgument((index), (eui64), EUI64_SIZE, true))
00339 #define emberGetEui64Argument(index, eui64) \
00340     (emberCopyStringArgument((index), (eui64), EUI64_SIZE, true))
00341
00346 uint8_t emberCopyBigEndianEui64Argument(int8_t
00347     index, EmberEUI64 destination);
00347
00351 void emberCommandReaderSetDefaultBase(uint8_t
00352     base);
00352
00357 void emberCommandActionHandler(const CommandAction
00358     action);
00364 void emberCommandErrorHandler(EmberCommandStatus status
00365 );
00365 void emberPrintCommandUsage(EmberCommandEntry
00366     *entry);
00366 void emberPrintCommandUsageNotes(void);
00367 void emberPrintCommandTable(void);
00368 void emberCommandClearBuffer(void);
00369
00372 void emberCommandReaderInit(void);
00373
00376 bool emberProcessCommandString(uint8_t *input, uint8_t
00377     sizeOrPort);
00377
00386 #define emberProcessCommandInput(port) \
00387     emberProcessCommandString(NULL, port)
00388
00391 #define emberCommandInterpreterEchoOn()
00392     (emberCommandInterpreter2Configuration
00393         |= EMBER_COMMAND_INTERPRETER_CONFIGURATION_ECHO)
00394

```

```

00397 #define emberCommandInterpreterEchoOff()          \
00398     (emberCommandInterpreter2Configuration        \
00399     &= (~EMBER_COMMAND_INTERPRETER_CONFIGURATION_ECHO)) \
00400 \
00403 #define emberCommandInterpreterIsEchoOn()         \
00404     (emberCommandInterpreter2Configuration        \
00405     & EMBER_COMMAND_INTERPRETER_CONFIGURATION_ECHO) \
00406 \
00409 #endif // __COMMAND_INTERPRETER2_H__

```

8.37 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`

8.37.1 Detailed Description

See [Stack Information](#) for documentation.

Definition in file [config.h](#).

8.38 config.h

```

00001
00018 // The 4 digit version: A.B.C.D
00019 #define EMBER_MAJOR_VERSION 5
00020 #define EMBER_MINOR_VERSION 7
00021 #define EMBER_PATCH_VERSION 0
00022 #define EMBER_SPECIAL_VERSION 0
00023
00024 // 2 bytes
00025 #define EMBER_BUILD_NUMBER 411
00026 #define EMBER_FULL_VERSION ( ((uint16_t)EMBER_MAJOR_VERSION << 12) \
00027           | ((uint16_t)EMBER_MINOR_VERSION << 8) \
00028           | ((uint16_t)EMBER_PATCH_VERSION << 4) \
00029           | ((uint16_t)EMBER_SPECIAL_VERSION))
00030
00031 #define EMBER_VERSION_TYPE EMBER_VERSION_TYPE_GA
00032
00036 #define SOFTWARE_VERSION EMBER_FULL_VERSION
00037

```

8.39 crc.h File Reference

Macros

- `#define INITIAL_CRC`
- `#define CRC32_START`
- `#define CRC32_END`

Functions

- `uint16_t halCommonCrc16 (uint8_t newByte, uint16_t prevResult)`
- `uint32_t halCommonCrc32 (uint8_t newByte, uint32_t prevResult)`

8.39.1 Detailed Description

See [Cyclic Redundancy Code \(CRC\)](#) for detailed documentation.

Definition in file [crc.h](#).

8.40 crc.h

```

00001
00007 #ifndef __CRC_H__
00008 #define __CRC_H__
00009
00028 uint16_t halCommonCrc16(uint8_t newByte, uint16_t prevResult);
00029
00030
00046 uint32_t halCommonCrc32(uint8_t newByte, uint32_t prevResult);
00047
00048 // 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
00052           LSB-MSB
00053
00057 #endif //__CRC_H__
00058

```

8.41 dev0680.h File Reference

Macros

- `#define PWRUP_CFG_SC1_TXD`
- `#define PWRDN_OUT_SC1_nRTS`

Custom Baud Rate Definitions

Application Framework NCP Configuration Board Header

This board header (dev0680) is not supported in framework NCP applications. NCP applications must use either the dev0680spi or dev0680uart board headers when creating custom NCP applications through the framework.

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, BOARDLED0, BOA-

 RDLED1,

 BOARDLED2, BOARDLED3, BOARD_ACTIVITY_LED, BOARD_HEARTBE-

 AT_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](#)

USB Power State

Define if the USB is self powered or bus powered since the configuration descriptor needs to report to the host the powered state.

Note

VBUS Monitoring is required for USB to function when the EM358 device is configured as self-powered.

- #define `USB_SELPWRD_STATE`

USB Remote Wakeup Enable

If the USB device needs to awake the host from suspend, then it needs to have remote wakeup enable.

Note

The host can deny remote wakeup, keeping the device in suspend.

If the device has remote wakeup enabled the configuration descriptor needs to report this fact to the host. Additionally, the USB core in the chip needs to be directly told. Set the define `USB_REMOTEWKUPEN_STATE` to 0 if remote wake is disabled or 1 if enabled.

- #define `USB_REMOTEWKUPEN_STATE`

USB Maximum Power Consumption

The USB device must report the maximum power it will draw from the bus. This is done via the `bMaxPower` parameter in the Configuration Descriptor reported to the host. The value used is in units of 2mA.

Self-powered devices are low power devices and must draw less than 100mA.

Systems that have components such as a FEM are likely to consume more than 100mA and are considered high power and therefore must be bus-powered.

- #define `USB_MAX_POWER`

USB Enumeration Control

The following are used to aid in the abstraction of which GPIO is used for controlling the pull-up resistor for enumeration.

The hardware setup connects the D+ signal to a GPIO via a 1.5kOhm pull-up resistor. Any GPIO can be used since it just needs to be a simple push-pull output configuration.

- #define `ENUMCTRL`
- #define `ENUMCTRL_SETCFG(cfg)`
- #define `ENUMCTRL_SET()`
- #define `ENUMCTRL_CLR()`

USB VBUS Monitoring Support

Note

VBUS Monitoring is required for USB to function when the EM358 device is configured as self-powered.

The following are used to aid in the abstraction of which GPIO and IRQ is used for VBUS Monitoring.

Remember that IRQA and IRQB are fixed to GPIO PB0 and PB6 respectively while IRQC and IRQD can be assigned to any GPIO. Since USB's D- and D+ data pins are fixed to PA0 and PA1 respectively, SC2 can't be used so it makes sense to allocate PA2 for enumeration control and PA3 for VBUS monitoring. Therefore, using PA3 for VBUS monitoring requires IRQC or IRQD.

The driver will only try to use VBUSMON functionality if USB_SELPWRD_STATE is set to 1.

- #define VBUSMON
- #define VBUSMON_IN
- #define VBUSMON_SETCFG()
- #define VBUSMON_SEL()
- #define VBUSMON_ISR
- #define VBUSMON_INTCFG
- #define VBUSMON_INT_EN_BIT
- #define VBUSMON_FLAG_BIT
- #define VBUSMON_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_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_DFL_RHO_FOR_RHO
- #define PWRUP_OUT_DFL_RHO_FOR_RHO

- #define PWRDN_CFG_DFL_RHO_FOR_RHO
- #define PWRDN_OUT_DFL_RHO_FOR_RHO
- #define PWRUP_CFG_DFL_RHO_FOR_DFL
- #define PWRUP_OUT_DFL_RHO_FOR_DFL
- #define PWRDN_CFG_DFL_RHO_FOR_DFL
- #define PWRDN_OUT_DFL_RHO_FOR_DFL
- #define PWRUP_CFG_DFL_RHO
- #define PWRUP_OUT_DFL_RHO
- #define PWRDN_CFG_DFL_RHO
- #define PWRDN_OUT_DFL_RHO
- #define halInternalInitRadioHoldOff()

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 deepsleeping. 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

USB Configuration Defines

Provide the proper set of pin configuration for when USB is not enumerated. Not enumerated primarily refers to the driver not being configured or deep sleep. The configuration used here is only for keeping the USB off the bus. The GPIO configuration used when active is controlled by the USB driver since the driver needs to control the enumeration process (which affects GPIO state.)

Note

: Using USB requires Serial port 3 to be defined and is only possible on EM3582/E-M3586/EM3588/EM359 chips.

- #define PWRUP_CFG_USBDM
- #define PWRUP_OUT_USBDM
- #define PWRUP_CFG_USBDP
- #define PWRUP_OUT_USBDP
- #define PWRUP_CFG_ENUMCTRL
- #define PWRUP_OUT_ENUMCTRL
- #define PWRUP_CFG_VBUSMON
- #define PWRUP_OUT_VBUSMON
- #define PWRDN_CFG_USBDM
- #define PWRDN_OUT_USBDM
- #define PWRDN_CFG_USBDP
- #define PWRDN_OUT_USBDP
- #define PWRDN_CFG_ENUMCTRL
- #define PWRDN_OUT_ENUMCTRL
- #define PWRDN_CFG_VBUSMON
- #define PWRDN_OUT_VBUSMON

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 SET_RESUME_GPIO_CFG_REGISTERS()
- #define SET_RESUME_GPIO_OUTPUT_DATA_REGISTERS()
- #define SET_SUSPEND_GPIO_CFG_REGISTERS()
- #define SET_SUSPEND_GPIO_OUTPUT_DATA_REGISTERS()
- #define CONFIGURE_EXTERNAL_REGULATOR_ENABLE()
- uint16_t gpioCfgPowerUp [6]
- uint16_t gpioCfgPowerDown [6]
- uint8_t gpioOutPowerUp [3]
- uint8_t gpioOutPowerDown [3]
- GpioMaskType 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 `halInternalSuspendBoard()`
- #define `halInternalPowerUpBoard()`
- #define `halInternalResumeBoard()`

8.41.1 Detailed Description

See [Sample Breakout Board Configuration](#) for detailed documentation.

Definition in file `dev0680.h`.

8.41.2 Macro Definition Documentation

8.41.2.1 #define `halInternalInitBoard()`

Initialize the board. This function is called from `halInit()`.

Definition at line 1262 of file `dev0680.h`.

8.41.2.2 #define `halInternalPowerDownBoard()`

Power down the board. This function is called from `halPowerDown()`.

Definition at line 1274 of file `dev0680.h`.

8.41.2.3 #define `halInternalSuspendBoard()`

Suspend the board. This function is called from `halSuspend()`.

Definition at line 1286 of file `dev0680.h`.

8.41.2.4 #define `halInternalPowerUpBoard()`

Power up the board. This function is called from `halPowerUp()`.

Definition at line 1298 of file `dev0680.h`.

8.41.2.5 #define halInternalResumeBoard()

Resume the board. This function is called from [halResume\(\)](#).

Definition at line 1314 of file [dev0680.h](#).

8.42 dev0680.h

```

00001
00046 #ifndef __BOARD_H__
00047 #define __BOARD_H__
00048
00055 #if defined(EMBER_AF_NCP)           \
00056   || defined(EZSP_SPI)             \
00057   || defined(EZSP_ASH)             \
00058   || defined(SLEEPY_EZSP_ASH)
00059   #error "NCP applications must use either the dev0680spi or dev0680uart board
00060   header."
00060 #endif
00061
00078 #define EMBER_SERIAL_BAUD_CUSTOM 13
00079
00080
00101
00107 enum HalBoardLedPins {
00108     BOARDLED0 = PORTA_PIN(6),
00109     BOARDLED1 = PORTA_PIN(7),
00110     BOARDLED2 = PORTC_PIN(5),
00111     BOARDLED3 = BOARDLED2,
00112     BOARD_ACTIVITY_LED = BOARDLED0,
00113     BOARD_HEARTBEAT_LED = BOARDLED1
00114 };
00115
00137 #define BUTTON0           PORTB_PIN(6)
00138
00141 #define BUTTON0_IN          GPIO_PBIN
00142
00146 #define BUTTON0_SEL()       do { } while(0)
00147
00150 #define BUTTON0_ISR         halIrqBIsr
00151
00154 #define BUTTON0_INTCFG      GPIO_INTCFGB
00155
00158 #define BUTTON0_INT_EN_BIT  INT IRQB
00159
00162 #define BUTTON0_FLAG_BIT    INT IRQBFLAG
00163
00166 #define BUTTON0_MISS_BIT    INT MISSIRQB
00167
00174 #define BUTTON1           PORTC_PIN(6)
00175
00178 #define BUTTON1_IN          GPIO_PCIN
00179
00184 #define BUTTON1_SEL()       do { GPIO_IRQCSEL = PORTC_PIN(6); } while(0)
00185
00189 #define BUTTON1_ISR         halIrqCIsr
00190
00193 #define BUTTON1_INTCFG      GPIO_INTCFGC
00194
00197 #define BUTTON1_INT_EN_BIT  INT IRQC
00198
00201 #define BUTTON1_FLAG_BIT    INT IRQCFLAG
00202
00205 #define BUTTON1_MISS_BIT    INT MISSIRQC
00206
00207
00208
00225 #ifdef USB_BUSPWRD
00226 #define USB_SELFPPWRD_STATE (0)
00227 #else
00228 #define USB_SELFPPWRD_STATE (1)
00229#endif
00230
00231
00232

```

```

00253 #define USB_REMOTEWKUPEN_STATE (1)
00254
00255
00256
00273 #define USB_MAX_POWER (50)
00274
00275
00276
00291 #define ENUMCTRL PORTA_PIN(2)
00292
00297 #define ENUMCTRL_SETCFG(cfg) do { SET_REG_FIELD(GPIO_PACFGL, PA2_CFG, cfg); }
00298     while(0)
00302 #define ENUMCTRL_SET()          do { GPIO_PASET = PA2; } while(0)
00303
00307 #define ENUMCTRL_CLR()          do { GPIO_PACL = PA2; } while(0)
00308
00309
00310
00337 #define VBUSMON PA3
00338
00341 #define VBUSMON_IN           GPIO_PAINT
00342
00346 #define VBUSMON_SETCFG() do { SET_REG_FIELD(GPIO_PACFGL, PA3_CFG, GPIOCFG_IN);
00347     } while(0)
00352 #define VBUSMON_SEL()          do { GPIO_IRQDSEL = PORTA_PIN(3); } while(0)
00353
00357 #define VBUSMON_ISR            halIrqDISr
00358
00361 #define VBUSMON_INTCFG        GPIO_INTCFGD
00362
00365 #define VBUSMON_INT_EN_BIT    INT IRQD
00366
00369 #define VBUSMON_FLAG_BIT      INT IRQDFLAG
00370
00373 #define VBUSMON_MISS_BIT      INT MISSIRQD
00374
00375
00391
00396 //##define RADIO_HOLDOFF // Configure Radio HoldOff at bootup
00398
00411
00418 //##define RHO_GPIO           PORTA_PIN(6)
00422 #define RHO_ASSERTED          1
00423
00426 #define RHO_CFG                GPIO_PACFGH
00427
00430 #define RHO_IN                 GPIO_PAINT
00431
00434 #define RHO_OUT                GPIO_PAOUT
00435
00439 #define RHO_SEL()              do { GPIO_IRQDSEL = RHO_GPIO; } while(0)
00440
00444 #define RHO_ISR                halIrqDISr
00445
00448 #define RHO_INTCFG            GPIO_INTCFGD
00449
00452 #define RHO_INT_EN_BIT        INT IRQD
00453
00456 #define RHO_FLAG_BIT          INT IRQDFLAG
00457
00460 #define RHO_MISS_BIT          INT MISSIRQD
00461
00464 #define PWRUP_CFG_DFL_RHO_FOR_RHO GPIOCFG_IN_PUD
00465 #define PWRUP_OUT_DFL_RHO_FOR_RHO GPIOOUT_PULLDOWN /* Deassert */
00466 #define PWRDN_CFG_DFL_RHO_FOR_RHO GPIOCFG_IN_PUD
00467 #define PWRDN_OUT_DFL_RHO_FOR_RHO GPIOOUT_PULLDOWN /* Deassert */
00468
00471 #define PWRUP_CFG_DFL_RHO_FOR_DFL GPIOCFG_OUT
00472 #define PWRUP_OUT_DFL_RHO_FOR_DFL 1 /* LED default off */
00473 #define PWRDN_CFG_DFL_RHO_FOR_DFL GPIOCFG_OUT
00474 #define PWRDN_OUT_DFL_RHO_FOR_DFL 1 /* LED off */
00475
00479 #if      (defined(RADIO_HOLDOFF) && defined(RHO_GPIO))
00480 // Initial bootup configuration is for Radio HoldOff
00481 #define PWRUP_CFG_DFL_RHO          PWRUP_CFG_DFL_RHO_FOR_RHO
00482 #define PWRUP_OUT_DFL_RHO         PWRUP_OUT_DFL_RHO_FOR_RHO
00483 #define PWRDN_CFG_DFL_RHO         PWRDN_CFG_DFL_RHO_FOR_RHO
00484 #define PWRDN_OUT_DFL_RHO         PWRDN_OUT_DFL_RHO_FOR_RHO

```

```

00485 #define halInternalInitRadioHoldOff() halSetRadioHoldOff(true)
00486 #else
00487 // Initial bootup configuration is for default
00488 #define PWRUP_CFG_DFL_RHO PWRUP_CFG_DFL_RHO_FOR_DFL
00489 #define PWRUP_OUT_DFL_RHO PWRUP_OUT_DFL_RHO_FOR_DFL
00490 #define PWRDN_CFG_DFL_RHO PWRDN_CFG_DFL_RHO_FOR_DFL
00491 #define PWRDN_OUT_DFL_RHO PWRDN_OUT_DFL_RHO_FOR_DFL
00492 #define halInternalInitRadioHoldOff() /* no-op */
00493 #endif//(defined(RADIO_HOLDOFF) && defined(RHO_GPIO))
00494
00495 #ifdef RHO_GPIO
00496
00497 #define ADJUST_GPIO_CONFIG_DFL_RHO(enableRadioHoldOff) do {
00498     ATOMIC( /* Must read-modify-write so to be safe, use ATOMIC() */ \
00499         if (enableRadioHoldOff) { /* Radio HoldOff */ \
00500             /* Actual register state */ \
00501             /*halGpioSetConfig(RHO_CFG, PWRUP_CFG_DFL_RHO_FOR_RHO);*/ \
00502             RHO_CFG = RHO_CFG \
00503             & ~ (0x000F << ((RHO_GPIO&3)*4)) \
00504             | (PWRUP_CFG_DFL_RHO_FOR_RHO << ((RHO_GPIO&3)*4)); \
00505             RHO_OUT = RHO_OUT \
00506             & ~ (0x0001 << ((RHO_GPIO&7) )) \
00507             | (PWRUP_OUT_DFL_RHO_FOR_RHO << ((RHO_GPIO&7) )); \
00508             /* Shadow register state */ \
00509             gpioCfgPowerUp[RHO_GPIO>>2] = gpioCfgPowerUp[RHO_GPIO>>2] \
00510             & ~ (0x000F << ((RHO_GPIO&3)*4)) \
00511             | (PWRUP_CFG_DFL_RHO_FOR_RHO << ((RHO_GPIO&3)*4)); \
00512             gpioOutPowerUp[RHO_GPIO>>3] = gpioOutPowerUp[RHO_GPIO>>3] \
00513             & ~ (0x0001 << ((RHO_GPIO&7) )) \
00514             | (PWRUP_OUT_DFL_RHO_FOR_RHO << ((RHO_GPIO&7) )); \
00515             gpioCfgPowerDown[RHO_GPIO>>2] = gpioCfgPowerDown[RHO_GPIO>>2] \
00516             & ~ (0x000F << ((RHO_GPIO&3)*4)) \
00517             | (PWRDN_CFG_DFL_RHO_FOR_RHO << ((RHO_GPIO&3)*4)); \
00518             gpioOutPowerDown[RHO_GPIO>>3] = gpioOutPowerDown[RHO_GPIO>>3] \
00519             & ~ (0x0001 << ((RHO_GPIO&7) )) \
00520             | (PWRDN_OUT_DFL_RHO_FOR_RHO << ((RHO_GPIO&7) )); \
00521             RHO_INTCFG = (0 << GPIO_INTFILT_BIT) /* 0 = no filter */ \
00522             | (3 << GPIO_INTMOD_BIT); /* 3 = both edges */ \
00523         } else { /* default */ \
00524             /* Actual register state */ \
00525             /*halGpioSetConfig(RHO_CFG, PWRUP_CFG_DFL_RHO_FOR_DFL);*/ \
00526             RHO_CFG = RHO_CFG \
00527             & ~ (0x000F << ((RHO_GPIO&3)*4)) \
00528             | (PWRUP_CFG_DFL_RHO_FOR_DFL << ((RHO_GPIO&3)*4)); \
00529             RHO_OUT = RHO_OUT \
00530             & ~ (0x0001 << ((RHO_GPIO&7) )) \
00531             | (PWRUP_OUT_DFL_RHO_FOR_DFL << ((RHO_GPIO&7) )); \
00532             /* Shadow register state */ \
00533             gpioCfgPowerUp[RHO_GPIO>>2] = gpioCfgPowerUp[RHO_GPIO>>2] \
00534             & ~ (0x000F << ((RHO_GPIO&3)*4)) \
00535             | (PWRUP_CFG_DFL_RHO_FOR_DFL << ((RHO_GPIO&3)*4)); \
00536             gpioOutPowerUp[RHO_GPIO>>3] = gpioOutPowerUp[RHO_GPIO>>3] \
00537             & ~ (0x0001 << ((RHO_GPIO&7) )) \
00538             | (PWRUP_OUT_DFL_RHO_FOR_DFL << ((RHO_GPIO&7) )); \
00539             gpioCfgPowerDown[RHO_GPIO>>2] = gpioCfgPowerDown[RHO_GPIO>>2] \
00540             & ~ (0x000F << ((RHO_GPIO&3)*4)) \
00541             | (PWRDN_CFG_DFL_RHO_FOR_DFL << ((RHO_GPIO&3)*4)); \
00542             gpioOutPowerDown[RHO_GPIO>>3] = gpioOutPowerDown[RHO_GPIO>>3] \
00543             & ~ (0x0001 << ((RHO_GPIO&7) )) \
00544             | (PWRDN_OUT_DFL_RHO_FOR_DFL << ((RHO_GPIO&7) )); \
00545             RHO_INTCFG = 0; /* disabled */ \
00546         } \
00547         RHO_SEL(); /* Point IRQ at the desired pin */ \
00548     } while (0) \
00549
00550 #endif//RHO_GPIO
00551
00552
00565 #define TEMP_SENSOR_ADC_CHANNEL ADC_SOURCE_ADC0_VREF2
00566
00569 #define TEMP_SENSOR_SCALE_FACTOR 1
00570
00585 #define PACKET_TRACE // We do have PACKET_TRACE support
00586
00587
00623 #define OSC32K_STARTUP_DELAY_MS (0)
00624
00625 #if OSC32K_STARTUP_DELAY_MS > MAX_INT16U_VALUE
00626 #error "OSC32K_STARTUP_DELAY_MS must fit in 16 bits."
00627 #endif

```

```

00628
00635 // #define ENABLE_OSC32K // Enable 32.768 kHz osc instead of 1 kHz RC osc
00637
00638
00655 // #define ENABLE_ALT_FUNCTION_REG_EN
00657
00658
00685 // #define DISABLE_INTERNAL_1V8_REGULATOR
00687
00688
00704 // #define ENABLE_ALT_FUNCTION_TX_ACTIVE
00706
00707
00726 // #define ENABLE_ALT_FUNCTION_NTX_ACTIVE
00728
00744 // #define EEPROM_USES_SHUTDOWN_CONTROL
00746
00747
00759
00760
00773 #ifdef PACKET_TRACE
00774     #define PWRUP_CFG_PTI_EN    GPIOCFG_OUT_ALT
00775     #define PWRUP_OUT_PTI_EN    0
00776     #define PWRDN_CFG_PTI_EN    GPIOCFG_IN_PUD
00777     #define PWRDN_OUT_PTI_EN    GPIOOUT_PULLDOWN
00778     #define PWRUP_CFG_PTI_DATA  GPIOCFG_OUT_ALT
00779     #define PWRUP_OUT_PTI_DATA  1
00780     #define PWRDN_CFG_PTI_DATA  GPIOCFG_IN_PUD
00781     #define PWRDN_OUT_PTI_DATA  GPIOOUT_PULLUP
00782 #else
00783     #define PWRUP_CFG_PTI_EN    GPIOCFG_IN
00784     #define PWRUP_OUT_PTI_EN    0
00785     #define PWRDN_CFG_PTI_EN    GPIOCFG_IN
00786     #define PWRDN_OUT_PTI_EN    0
00787     #define PWRUP_CFG_PTI_DATA  GPIOCFG_IN
00788     #define PWRUP_OUT_PTI_DATA  0
00789     #define PWRDN_CFG_PTI_DATA  GPIOCFG_IN
00790     #define PWRDN_OUT_PTI_DATA  0
00791 #endif//PACKET_TRACE
00792
00793
00794
00817 #if defined(ENABLE_OSC32K) && defined(ENABLE_ALT_FUNCTION_NTX_ACTIVE)
00818     //Oops! Only one of these can be used at a time!
00819     #error ENABLE_OSC32K and ENABLE_ALT_FUNCTION_NTX_ACTIVE are mutually\
00820 exclusive. They define conflicting usage for GPIO PC6.
00821
00822 #elif defined(ENABLE_OSC32K) && !defined(ENABLE_ALT_FUNCTION_NTX_ACTIVE)
00823     //Use OCS32K configuration
00824     #define PWRUP_CFG_BUTTON1  GPIOCFG_ANALOG
00825     #define PWRUP_OUT_BUTTON1  0
00826     #define PWRDN_CFG_BUTTON1  GPIOCFG_ANALOG
00827     #define PWRDN_OUT_BUTTON1  0
00828
00829 #elif !defined(ENABLE_OSC32K) && defined(ENABLE_ALT_FUNCTION_NTX_ACTIVE)
00830     //Use nTX_ACTIVE configuration
00831     #define PWRUP_CFG_BUTTON1  GPIOCFG_OUT_ALT
00832     #define PWRUP_OUT_BUTTON1  0
00833     #define PWRDN_CFG_BUTTON1  GPIOCFG_OUT
00834     #define PWRDN_OUT_BUTTON1  0
00835
00836 #else
00837     //Use Button1 configuration
00838     #define PWRUP_CFG_BUTTON1  GPIOCFG_IN_PUD
00839     #define PWRUP_OUT_BUTTON1  GPIOOUT_PULLUP /* Button needs a pullup */
00840     #define PWRDN_CFG_BUTTON1  GPIOCFG_IN_PUD
00841     #define PWRDN_OUT_BUTTON1  GPIOOUT_PULLUP /* Button needs a pullup */
00842
00843 #endif
00844
00848 #ifdef ENABLE_OSC32K
00849     #define CFG_TEMPEN        GPIOCFG_ANALOG
00850 #else
00851     #define CFG_TEMPEN        GPIOCFG_OUT
00852 #endif//ENABLE_OSC32K
00853
00854
00855
00866 #ifdef ENABLE_ALT_FUNCTION_TX_ACTIVE
00867     #define PWRUP_CFG_LED2    GPIOCFG_OUT_ALT

```

```

00868     #define PWRUP_OUT_LED2 0
00869     #define PWRDN_CFG_LED2 GPIOCFG_OUT
00870     #define PWRDN_OUT_LED2 0
00871 #else
00872     #define PWRUP_CFG_LED2 GPIOCFG_OUT
00873     #define PWRUP_OUT_LED2 1 /* LED default off */
00874     #define PWRDN_CFG_LED2 GPIOCFG_OUT
00875     #define PWRDN_OUT_LED2 1 /* LED default off */
00876 #endif//ENABLE_ALT_FUNCTION_TX_ACTIVE
00877
00878
00879
00896 #if (!defined(EM_SERIAL3_DISABLED)) && \
00897     (defined(CORTEXM3_EM3582) || defined(CORTEXM3_EM3586) || \
00898     defined(CORTEXM3_EM3588) || defined(CORTEXM3_EM359))
00899     #define PWRUP_CFG_USBDM GPIOCFG_IN
00900     #define PWRUP_OUT_USBDM 0
00901     #define PWRUP_CFG_USBDP GPIOCFG_IN
00902     #define PWRUP_OUT_USBDP 0
00903     #define PWRUP_CFG_ENUMCTRL GPIOCFG_OUT
00904     #define PWRUP_OUT_ENUMCTRL 0
00905     #define PWRUP_CFG_VBUSMON GPIOCFG_IN
00906     #define PWRUP_OUT_VBUSMON 0
00907     #define PWRDN_CFG_USBDM GPIOCFG_IN
00908     #define PWRDN_OUT_USBDM 0
00909     #define PWRDN_CFG_USBDP GPIOCFG_IN
00910     #define PWRDN_OUT_USBDP 0
00911     #define PWRDN_CFG_ENUMCTRL GPIOCFG_OUT
00912     #define PWRDN_OUT_ENUMCTRL 0
00913     #define PWRDN_CFG_VBUSMON GPIOCFG_IN
00914     #define PWRDN_OUT_VBUSMON 0
00915 #else
00916     #define PWRUP_CFG_USBDM GPIOCFG_OUT_ALT
00917     #define PWRUP_OUT_USBDM 0
00918     #define PWRUP_CFG_USBDP GPIOCFG_IN
00919     #define PWRUP_OUT_USBDP 0
00920     #define PWRUP_CFG_ENUMCTRL GPIOCFG_OUT_ALT
00921     #define PWRUP_OUT_ENUMCTRL 0
00922     #define PWRUP_CFG_VBUSMON GPIOCFG_OUT
00923     #define PWRUP_OUT_VBUSMON 1
00924     #define PWRDN_CFG_USBDM GPIOCFG_IN_PUD
00925     #define PWRDN_OUT_USBDM GPIOOUT_PULLUP
00926     #define PWRDN_CFG_USBDP GPIOCFG_IN_PUD
00927     #define PWRDN_OUT_USBDP GPIOOUT_PULLUP
00928     #define PWRDN_CFG_ENUMCTRL GPIOCFG_IN_PUD
00929     #define PWRDN_OUT_ENUMCTRL GPIOOUT_PULLUP
00930     #define PWRDN_CFG_VBUSMON GPIOCFG_OUT
00931     #define PWRDN_OUT_VBUSMON 1
00932 #endif//
00933
00934
00938 #if      SLEEPY_IP_MODEM_UART
00939     #define PWRUP_CFG_SC1_TXD GPIOCFG_OUT      // Let UART driver manage OUT_ALT
00940     #define PWRDN_OUT_SC1_nRTS 0                 // Let peer send data to wake
00941 #else
00942     #define PWRUP_CFG_SC1_TXD GPIOCFG_OUT_ALT // Pre-set for UART operation
00943     #define PWRDN_OUT_SC1_nRTS 1                 // Deassert nRTS when sleeping
00944 #endif//SLEEPY_IP_MODEM_UART
00945
00946
00955 //Each pin has 4 cfg bits. There are 3 ports with 2 cfg registers per
00956 //port since the cfg register only holds 2 pins (16bits). Therefore,
00957 //the cfg arrays need to be 6 entries of 16bits.
00958 extern uint16_t gpioCfgPowerUp[6];
00959 extern uint16_t gpioCfgPowerDown[6];
00960 //Each pin has 1 out bit. There are 3 ports with 1 out register per
00961 //port (8bits). Therefore, the out arrays need to be 3 entries of 8bits.
00962 extern uint8_t gpioOutPowerUp[3];
00963 extern uint8_t gpioOutPowerDown[3];
00964 //A single mask variable covers all GPIO.
00965 extern GpioMaskType gpioRadioPowerBoardMask;
00966
00967
00974 #define DEFINE_GPIO_RADIO_POWER_BOARD_MASK_VARIABLE() \
00975 GpioMaskType gpioRadioPowerBoardMask = 0
00976
00977
00981 #define DEFINE_POWERUP_GPIO_CFG_VARIABLES() \
00982 uint16_t gpioCfgPowerUp[6] = {
00983

```

```

00983     ((PWRUP_CFG_USBDM    <<PA0_CFG_BIT) | \
00984     (PWRUP_CFG_USBDP   <<PA1_CFG_BIT) | \
00985     (PWRUP_CFG_ENUMCTRL <<PA2_CFG_BIT) | \
00986     (PWRUP_CFG_VBUSMON <<PA3_CFG_BIT)), \
00987     ((PWRUP_CFG_PTI_EN   <<PA4_CFG_BIT) | \
00988     (PWRUP_CFG_PTI_DATA <<PA5_CFG_BIT) | \
00989     (PWRUP_CFG_DFL_RHO  <<PA6_CFG_BIT)) | \
00990     (GPIOCFG_OUT        <<PA7_CFG_BIT)), \
00991     ((GPIOCFG_OUT        <<PB0_CFG_BIT) | \
00992     (PWRUP_CFG_SC1_TXD  <<PB1_CFG_BIT) /* SC1TXD */ | \
00993     (GPIOCFG_IN_PUD     <<PB2_CFG_BIT) /* SC1RXD */ | \
00994     (GPIOCFG_IN_PUD     <<PB3_CFG_BIT) /* SC1nCTS */ | \
00995     ((GPIOCFG_OUT_ALT   <<PB4_CFG_BIT) /* SC1nRTS */ | \
00996     (GPIOCFG_ANALOG     <<PB5_CFG_BIT) | \
00997     (GPIOCFG_IN_PUD     <<PB6_CFG_BIT) | \
00998     (GPIOCFG_OUT_ALT    <<PB7_CFG_BIT)), \
00999     ((GPIOCFG_IN         <<PC0_CFG_BIT) | \
01000     (GPIOCFG_OUT         <<PC1_CFG_BIT) | \
01001     (GPIOCFG_OUT_ALT    <<PC2_CFG_BIT) | \
01002     (GPIOCFG_IN         <<PC3_CFG_BIT)), \
01003     ((GPIOCFG_IN         <<PC4_CFG_BIT) | \
01004     (PWRUP_CFG_LED2     <<PC5_CFG_BIT) | \
01005     (PWRUP_CFG_BUTTON1  <<PC6_CFG_BIT) | \
01006     (CFG_TEMPEN         <<PC7_CFG_BIT)) | \
01007   } \
01008 \
01009 \
01013 #define DEFINE_POWERUP_GPIO_OUTPUT_DATA_VARIABLES()
01014 uint8_t gpioOutPowerUp[3] = {
01015     ((PWRUP_OUT_USBDM    <<PA0_BIT) | \
01016     (PWRUP_OUT_USBDP   <<PA1_BIT) | \
01017     (PWRUP_OUT_ENUMCTRL <<PA2_BIT) | \
01018     (PWRUP_OUT_VBUSMON <<PA3_BIT) | \
01019     (PWRUP_OUT_PTI_EN   <<PA4_BIT) | \
01020     (PWRUP_OUT_PTI_DATA <<PA5_BIT) | \
01021     (PWRUP_OUT_DFL_RHO  <<PA6_BIT)) | \
01022     /* LED default off */ \
01023     (1                <<PA7_BIT)), \
01024     (1                <<PB0_BIT) | \
01025     (1                <<PB1_BIT) /* SC1TXD */ | \
01026     (1                <<PB2_BIT) /* SC1RXD */ | \
01027     (1                <<PB3_BIT) /* SC1nCTS */ | \
01028     (0                <<PB4_BIT) /* SC1nRTS */ | \
01029     (0                <<PB5_BIT)) | \
01030     /* PB6 has button needing a pullup */ \
01031     (GPIOOUT_PULLUP    <<PB6_BIT) | \
01032     (0                <<PB7_BIT)), \
01033     (0                <<PC0_BIT) | \
01034     (0                <<PC1_BIT) | \
01035     (1                <<PC2_BIT) | \
01036     (0                <<PC3_BIT) | \
01037     (0                <<PC4_BIT)), \
01038     (PWRUP_OUT_LED2    <<PC5_BIT) | \
01039     (PWRUP_OUT_BUTTON1 <<PC6_BIT)) | \
01040     /* Temp Sensor default on */ \
01041     (1                <<PC7_BIT)) | \
01042   } \
01043 \
01044 \
01048 #define DEFINE_POWERDOWN_GPIO_CFG_VARIABLES()
01049 uint16_t gpioCfgPowerDown[6] = {
01050     ((PWRDN_CFG_USBDM   <<PA0_CFG_BIT) | \
01051     (PWRDN_CFG_USBDP   <<PA1_CFG_BIT) | \
01052     (PWRDN_CFG_ENUMCTRL <<PA2_CFG_BIT)) | \
01053     ((PWRDN_CFG_VBUSMON <<PA3_CFG_BIT)), \
01054     ((PWRDN_CFG_PTI_EN   <<PA4_CFG_BIT) | \
01055     (PWRDN_CFG_PTI_DATA <<PA5_CFG_BIT)) | \
01056     (PWRDN_CFG_DFL_RHO  <<PA6_CFG_BIT)) | \
01057     (GPIOCFG_OUT        <<PA7_CFG_BIT)), \
01058     ((GPIOCFG_OUT        <<PB0_CFG_BIT) | \

```

```

01059 \
01060 /*\ (GPIOCFG_OUT <<PB1_CFG_BIT) | /* SC1TXD
01061 /*\ (GPIOCFG_IN_PUD <<PB2_CFG_BIT) | /* SC1RXD
01062 /*\ ((GPIOCFG_OUT <<PB3_CFG_BIT)), /* SC1nCTS
01063 /*\ /* disable analog for sleep */
01064 \
01065 \
01066 /* need to use pulldown for sleep */
01067 \
01068 ((GPIOCFG_IN_PUD <<PB7_CFG_BIT)),
01069 \
01070 \
01071 \
01072 \
01073 \
01074 \
01075 \
01076 \
01077 \
01078 \
01082 #define DEFINE_POWERDOWN_GPIO_OUTPUT_DATA_VARIABLES()
01083 uint8_t gpioOutPowerDown[3] = {
01084 ((PWRDN_OUT_USBDM <<PA0_BIT) |
01085 (PWRDN_OUT_USBDP <<PA1_BIT) |
01086 (PWRDN_OUT_ENUMCTRL <<PA2_BIT) |
01087 (PWRDN_OUT_VBUSMON <<PA3_BIT) |
01088 /* enable is idle low */
01089 (PWRDN_OUT_PT1_EN <<PA4_BIT) |
01090 /* data is idle high */
01091 (PWRDN_OUT_PT1_DATA <<PA5_BIT) |
01092 (PWRDN_OUT_DFL_RHO <<PA6_BIT) |
01093 /* LED off */
01094 (1 <<PA7_BIT)),
01095 ((0 <<PB0_BIT) |
01096 (1 <<PB1_BIT) | /* SC1TXD */ |
01097 (GPIOOUT_PULLUP <<PB2_BIT) | /* SC1RXD */ |
01098 (GPIOOUT_PULLUP <<PB3_BIT) | /* SC1nCTS */ |
01099 (PWRDN_OUT_SC1_nRTS <<PB4_BIT) | /* SC1nRTS */ |
01100 /* tempsense needs pulldown */
01101 (GPIOOUT_PULLDOWN <<PB5_BIT) |
01102 /* PB6 has button needing a pullup */
01103 (GPIOOUT_PULLUP <<PB6_BIT) |
01104 /* buzzer needs pulldown for sleep */
01105 (GPIOOUT_PULLDOWN <<PB7_BIT)),
01106 ((GPIOOUT_PULLUP <<PC0_BIT) |
01107 (0 <<PC1_BIT) |
01108 (1 <<PC2_BIT) |
01109 (GPIOOUT_PULLDOWN <<PC3_BIT) |
01110 (GPIOOUT_PULLDOWN <<PC4_BIT) |
01111 (PWRDN_OUT_LED2 <<PC5_BIT) |
01112 (PWRDN_OUT_BUTTON1 <<PC6_BIT) |
01113 /* Temp Sensor off */
01114 (0 <<PC7_BIT))
01115 \
01116 \
01117 \
01121 #define SET_POWERUP_GPIO_CFG_REGISTERS() \
01122 GPIO_PACFGL = gpioCfgPowerUp[0]; \
01123 GPIO_PACFGH = gpioCfgPowerUp[1]; \
01124 GPIO_PBCFGL = gpioCfgPowerUp[2]; \
01125 GPIO_PBCFGH = gpioCfgPowerUp[3]; \
01126 GPIO_PCCFGL = gpioCfgPowerUp[4];

```

```

01127     GPIO_PCCFGH = gpioCfgPowerUp[5];
01128
01129
01130 #define SET_POWERUP_GPIO_OUTPUT_DATA_REGISTERS() \
01131     GPIO_PAOUT = gpioOutPowerUp[0]; \
01132     GPIO_PBOUT = gpioOutPowerUp[1]; \
01133     GPIO_PCOUT = gpioOutPowerUp[2];
01134
01135
01136
01137
01138 #define SET_POWERDOWN_GPIO_CFG_REGISTERS() \
01139     GPIO_PACFGL = gpioCfgPowerDown[0]; \
01140     GPIO_PACFGH = gpioCfgPowerDown[1]; \
01141     GPIO_PBCFGL = gpioCfgPowerDown[2]; \
01142     GPIO_PBCFGH = gpioCfgPowerDown[3]; \
01143     GPIO_PCCFGL = gpioCfgPowerDown[4]; \
01144     GPIO_PCCFGH = gpioCfgPowerDown[5];
01145
01146
01147
01148
01149
01150
01151 #define SET_POWERDOWN_GPIO_OUTPUT_DATA_REGISTERS() \
01152     GPIO_PAOUT = gpioOutPowerDown[0]; \
01153     GPIO_PBOUT = gpioOutPowerDown[1]; \
01154     GPIO_PCOUT = gpioOutPowerDown[2];
01155
01156
01157
01158
01159 #define SET_RESUME_GPIO_CFG_REGISTERS() \
01160     SET_POWERUP_GPIO_CFG_REGISTERS()
01161
01162
01163
01164
01165
01166 #define SET_RESUME_GPIO_OUTPUT_DATA_REGISTERS() \
01167     SET_POWERUP_GPIO_OUTPUT_DATA_REGISTERS()
01168
01169
01170
01171
01172
01173 #define SET_SUSPEND_GPIO_CFG_REGISTERS() \
01174     /* GPIO_PACFGL USB untouched! */ \
01175     GPIO_PACFGH = gpioCfgPowerDown[1]; \
01176     GPIO_PBCFGL = gpioCfgPowerDown[2]; \
01177     GPIO_PBCFGH = gpioCfgPowerDown[3]; \
01178     GPIO_PCCFGL = gpioCfgPowerDown[4]; \
01179     GPIO_PCCFGH = gpioCfgPowerDown[5];
01180
01181
01182
01183
01184
01185 #define SET_SUSPEND_GPIO_OUTPUT_DATA_REGISTERS() \
01186     GPIO_PAOUT = (GPIO_PAOUT & 0x0F) /*USB untouched*/ \
01187         | (gpioOutPowerDown[0] & 0xF0); \
01188     GPIO_PBOUT = gpioOutPowerDown[1]; \
01189     GPIO_PCOUT = gpioOutPowerDown[2];
01190
01191
01192
01193
01194
01195 #ifdef ENABLE_ALT_FUNCTION_REG_EN
01196     #define CONFIGURE_EXTERNAL_REGULATOR_ENABLE() GPIO_DBGCFG |= GPIO_EXTREGEN;
01197 #else
01198     #define CONFIGURE_EXTERNAL_REGULATOR_ENABLE() GPIO_DBGCFG &= ~GPIO_EXTREGEN;
01199 #endif
01200
01201
01202
01203
01204
01205
01206 #define WAKE_ON_PA0 false
01207 #define WAKE_ON_PA1 false
01208 #define WAKE_ON_PA2 false
01209 #define WAKE_ON_PA3 false
01210 #define WAKE_ON_PA4 false
01211 #define WAKE_ON_PA5 false
01212 #define WAKE_ON_PA6 false
01213 #define WAKE_ON_PA7 false
01214 #define WAKE_ON_PB0 false
01215 #define WAKE_ON_PB1 false
01216 #if SLEEPY_IP_MODEM_UART // SC1RXD
01217     #define WAKE_ON_PB2 true
01218 #else
01219     #define WAKE_ON_PB2 false
01220 #endif
01221 #define WAKE_ON_PB3 false
01222 #define WAKE_ON_PB4 false
01223 #define WAKE_ON_PB5 false
01224 #define WAKE_ON_PB6 true //BUTTON0
01225 #define WAKE_ON_PB7 false
01226 #define WAKE_ON_PC0 false
01227 #define WAKE_ON_PC1 false
01228 #define WAKE_ON_PC2 false
01229 #define WAKE_ON_PC3 false
01230 #define WAKE_ON_PC4 false

```

```

01241 #define WAKE_ON_PC5    false
01242 #define WAKE_ON_PC6    true     //BUTTON1
01243 #define WAKE_ON_PC7    false
01244
01245
01246
01248
01249
01262 #define halInternalInitBoard()
01263     do {
01264         halInternalPowerUpBoard();
01265         halInternalInitRadioHoldOff();
01266         halInternalRestartUart();
01267         halInternalInitButton();
01268     } while(0)
01269
01274 #define halInternalPowerDownBoard()
01275     do {
01276         /* Board peripheral deactivation */
01277         /* halInternalSleepAdc(); */
01278         SET_POWERDOWN_GPIO_OUTPUT_DATA_REGISTERS()
01279         SET_POWERDOWN_GPIO_CFG_REGISTERS()
01280     } while(0)
01281
01286 #define halInternalSuspendBoard()
01287     do {
01288         /* Board peripheral deactivation */
01289         /* halInternalSleepAdc(); */
01290         SET_SUSPEND_GPIO_OUTPUT_DATA_REGISTERS()
01291         SET_SUSPEND_GPIO_CFG_REGISTERS()
01292     } while(0)
01293
01298 #define halInternalPowerUpBoard()
01299     do {
01300         SET_POWERUP_GPIO_OUTPUT_DATA_REGISTERS()
01301         SET_POWERUP_GPIO_CFG_REGISTERS()
01302         /*The radio GPIO should remain in the powerdown state */
01303         /*until the stack specifically powers them up. */
01304         halStackRadioPowerDownBoard();
01305         CONFIGURE_EXTERNAL_REGULATOR_ENABLE()
01306         /* Board peripheral reactivation */
01307         halInternalInitAdc();
01308     } while(0)
01309
01314 #define halInternalResumeBoard()
01315     do {
01316         SET_RESUME_GPIO_OUTPUT_DATA_REGISTERS()
01317         SET_RESUME_GPIO_CFG_REGISTERS()
01318         /*The radio GPIO should remain in the powerdown state */
01319         /*until the stack specifically powers them up. */
01320         halStackRadioPowerDownBoard();
01321         CONFIGURE_EXTERNAL_REGULATOR_ENABLE()
01322         /* Board peripheral reactivation */
01323         halInternalInitAdc();
01324     } while(0)
01325
01326
01327 #endif //__BOARD_H__
01328

```

8.43 diagnostic.h File Reference

Macros

- `#define halResetWasCrash()`

Functions

- `uint32_t halGetMainStackBytesUsed (void)`
- `void halPrintCrashSummary (uint8_t port)`

- void [halPrintCrashDetails](#) (uint8_t port)
- void [halPrintCrashData](#) (uint8_t port)
- const HalAssertInfoType * [halGetAssertInfo](#) (void)

8.43.1 Detailed Description

See [Crash and Watchdog Diagnostics](#) for detailed documentation.

Definition in file [diagnostic.h](#).

8.44 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     uint16_t resetReason;        // reason written out just before forcing a reset
00031     uint16_t resetSignature;
00032     uint16_t panId;            // PanId that the bootloader will use
00033     uint8_t  radioChannel;      // emberGetRadioChannel() - 11
00034     uint8_t  radioPower;        // emberGetRadioPower()
00035     uint8_t  radioLnaCal;       // Low Noise Amplifier calibration
00036     uint8_t  reserved[22];     // (reserved for future use)
00037 } HalBootParamType;
00038
00039 typedef struct
00040 {
00041     PGM_P file;
00042     int line;
00043 } HalAssertInfoType;
00044
00045 // note that assertInfo and dmaProt are written just before a forced reboot
00046 typedef union
00047 {
00048     HalAssertInfoType assertInfo;
00049     struct { uint32_t channel; uint32_t address; } dmaProt;
00050 } HalCrashSpecificDataType;
00051
00052 // Define crash registers as structs so a debugger can display their bit fields
00053 typedef union {
00054     struct
00055     {
00056         uint32_t EXCPT          : 9;    // B0-8
00057         uint32_t ICIIT_LOW       : 7;    // B9-15
00058         uint32_t                 : 8;    // B16-23
00059         uint32_t T              : 1;    // B24
00060         uint32_t ICIIT_HIGH      : 2;    // B25-26
00061         uint32_t Q              : 1;    // B27
00062         uint32_t V              : 1;    // B28
00063         uint32_t C              : 1;    // B29
00064         uint32_t Z              : 1;    // B30
00065         uint32_t N              : 1;    // B31
00066     } bits;
00067     uint32_t word;
00068 } HalCrashxPsrType;
00069
00070 typedef union {

```

```

00071 struct
00072 {
00073     uint32_t VECTACTIVE : 9; // B0-8
00074     uint32_t : 2; // B9-10
00075     uint32_t RETTOBASE : 1; // B11
00076     uint32_t VECTPENDING : 9; // B12-20
00077     uint32_t : 1; // B21
00078     uint32_t ISR_PENDING : 1; // B22
00079     uint32_t ISR_PREAMPT : 1; // B23
00080     uint32_t : 1; // B24
00081     uint32_t PENDSTCLR : 1; // B25
00082     uint32_t PENDSTSET : 1; // B26
00083     uint32_t PENDSVCLR : 1; // B27
00084     uint32_t PENDSVSET : 1; // B28
00085     uint32_t : 2; // B29-30
00086     uint32_t NMIPENDSET : 1; // B31
00087 } bits;
00088 uint32_t word;
00089 } HalCrashIcsrType;
00090
00091 typedef union {
00092     struct
00093     {
00094 #ifdef _EZR_DEVICE
00095         uint32_t DMA IRQn : 1; // B0
00096         uint32_t GPIO EVEN IRQn : 1; // B1
00097         uint32_t TIMER0 IRQn : 1; // B2
00098         uint32_t USART0_RX IRQn : 1; // B3
00099         uint32_t USART0_TX IRQn : 1; // B4
00100         uint32_t USB IRQn : 1; // B5
00101         uint32_t ACMP0 IRQn : 1; // B6
00102         uint32_t ADC0 IRQn : 1; // B7
00103         uint32_t DAC0 IRQn : 1; // B8
00104         uint32_t I2C0 IRQn : 1; // B9
00105         uint32_t I2C1 IRQn : 1; // B10
00106         uint32_t GPIO ODD IRQn : 1; // B11
00107         uint32_t TIMER1 IRQn : 1; // B12
00108         uint32_t TIMER2 IRQn : 1; // B13
00109         uint32_t TIMER3 IRQn : 1; // B14
00110         uint32_t USART1_RX IRQn : 1; // B15
00111         uint32_t USART1_TX IRQn : 1; // B16
00112         uint32_t LESENSE IRQn : 1; // B17
00113         uint32_t USART2_RX IRQn : 1; // B18
00114         uint32_t USART2_TX IRQn : 1; // B19
00115         uint32_t UART0_RX IRQn : 1; // B20
00116         uint32_t UART0_TX IRQn : 1; // B21
00117         uint32_t UART1_RX IRQn : 1; // B22
00118         uint32_t UART1_TX IRQn : 1; // B23
00119         uint32_t LEUART0 IRQn : 1; // B24
00120         uint32_t LEUART1 IRQn : 1; // B25
00121         uint32_t LETIMER0 IRQn : 1; // B26
00122         uint32_t PCNT0 IRQn : 1; // B27
00123         uint32_t PCNT1 IRQn : 1; // B28
00124         uint32_t PCNT2 IRQn : 1; // B29
00125         uint32_t RTC IRQn : 1; // B30
00126         uint32_t BURTC IRQn : 1; // B31
00127         uint32_t CMU IRQn : 1; // B32
00128         uint32_t VCMP IRQn : 1; // B33
00129         uint32_t LCD IRQn : 1; // B34
00130         uint32_t MSC IRQn : 1; // B35
00131         uint32_t AES IRQn : 1; // B36
00132         uint32_t EBI IRQn : 1; // B37
00133         uint32_t EMU IRQn : 1; // B38
00134         uint32_t : 25; // B39-63
00135     } bits;
00136     uint32_t word[2];
00137 #elif defined (_EFR_DEVICE)
00138     uint32_t EMU IRQn : 1; // B0
00139     uint32_t FRC_PRI IRQn : 1; // B1
00140     uint32_t FRC IRQn : 1; // B2
00141     uint32_t MODEM IRQn : 1; // B3
00142     uint32_t RAC_SEQ IRQn : 1; // B4
00143     uint32_t RAC_RSM IRQn : 1; // B5
00144     uint32_t BUFC IRQn : 1; // B6
00145     uint32_t LDMA IRQn : 1; // B7
00146     uint32_t GPIO EVEN IRQn : 1; // B8
00147     uint32_t TIMER0 IRQn : 1; // B9
00148     uint32_t USART0_RX IRQn : 1; // B10
00149     uint32_t USART0_TX IRQn : 1; // B11
00150     uint32_t ACMP0 IRQn : 1; // B12

```

```

00151     uint32_t ADC0_IRQHandler      : 1; // B13
00152     uint32_t IDAC0_IRQHandler    : 1; // B14
00153     uint32_t I2C0_IRQHandler     : 1; // B15
00154     uint32_t GPIO_ODD_IRQHandler : 1; // B16
00155     uint32_t TIMER1_IRQHandler   : 1; // B17
00156     uint32_t USART1_RX_IRQHandler: 1; // B18
00157     uint32_t USART1_TX_IRQHandler: 1; // B19
00158     uint32_t LEUART0_IRQHandler  : 1; // B20
00159     uint32_t PCNT0_IRQHandler    : 1; // B21
00160     uint32_t CMU_IRQHandler     : 1; // B22
00161     uint32_t MSC_IRQHandler     : 1; // B23
00162     uint32_t CRYPTO_IRQHandler   : 1; // B24
00163     uint32_t LETIMER0_IRQHandler : 1; // B25
00164     uint32_t AGC_IRQHandler     : 1; // B26
00165     uint32_t PROTIMER_IRQHandler: 1; // B27
00166     uint32_t RTCC_IRQHandler    : 1; // B28
00167     uint32_t SYNTH_IRQHandler   : 1; // B29
00168     uint32_t CRYOTIMER_IRQHandler: 1; // B30
00169     uint32_t RFSENSE_IRQHandler  : 1; // B31
00170     uint32_t FPUEH_IRQHandler   : 1; // B32
00171     uint32_t WDOG_IRQHandler    : 1; // B33
00172     uint32_t                      : 30; // B34-63
00173 } bits;
00174 uint32_t word[2];
00175 #elif CORTEXM3_EMBER_MICRO
00176     uint32_t Timer1           : 1; // B0
00177     uint32_t Timer2           : 1; // B1
00178     uint32_t Management       : 1; // B2
00179     uint32_t Baseband         : 1; // B3
00180     uint32_t Sleep_Timer     : 1; // B4
00181     uint32_t SC1              : 1; // B5
00182     uint32_t SC2              : 1; // B6
00183     uint32_t Security         : 1; // B7
00184     uint32_t MAC_Timer        : 1; // B8
00185     uint32_t MAC_TX           : 1; // B9
00186     uint32_t MAC_RX           : 1; // B10
00187     uint32_t ADC               : 1; // B11
00188     uint32_t IRQ_A            : 1; // B12
00189     uint32_t IRQ_B            : 1; // B13
00190     uint32_t IRQ_C            : 1; // B14
00191     uint32_t IRQ_D            : 1; // B15
00192     uint32_t Debug             : 1; // B16
00193     uint32_t                      : 15; // B17-31
00194 } bits;
00195 uint32_t word;
00196 #else
00197 #error micro not recognized
00198 #endif
00199 } HalCrashIntActiveType;
00200
00201 typedef union {
00202     struct
00203     {
00204         uint32_t MEMFAULTACT      : 1; // B0
00205         uint32_t BUSFAULTACT      : 1; // B1
00206         uint32_t                  : 1; // B2
00207         uint32_t USGFAULTACT      : 1; // B3
00208         uint32_t                  : 3; // B4-6
00209         uint32_t SVCALLACT        : 1; // B7
00210         uint32_t MONITORACT       : 1; // B8
00211         uint32_t                  : 1; // B9
00212         uint32_t PENDSVACT        : 1; // B10
00213         uint32_t SYSTICKACT       : 1; // B11
00214         uint32_t USGFAULTPENDED   : 1; // B12
00215         uint32_t MEMFAULTPENDED   : 1; // B13
00216         uint32_t BUSFAULTPENDED   : 1; // B14
00217         uint32_t SVCALLPENDED     : 1; // B15
00218         uint32_t MEMFAULTENA      : 1; // B16
00219         uint32_t BUSFAULTENA      : 1; // B17
00220         uint32_t USGFAULTENA      : 1; // B18
00221         uint32_t                      : 13; // B19-31
00222     } bits;
00223     uint32_t word;
00224 } HalCrashShcsrType;
00225
00226 typedef union {
00227     struct
00228     {
00229         uint32_t IACCVIOL          : 1; // B0
00230         uint32_t DACCVIOL          : 1; // B1

```

```

00231     uint32_t          : 1; // B2
00232     uint32_t MUNSTKERR : 1; // B3
00233     uint32_t MSTKERR  : 1; // B4
00234     uint32_t          : 2; // B5-6
00235     uint32_t MMARVALID : 1; // B7
00236     uint32_t IBUSERR   : 1; // B8
00237     uint32_t PRECISERR : 1; // B9
00238     uint32_t IMPRECISERR: 1; // B10
00239     uint32_t UNSTKERR  : 1; // B11
00240     uint32_t STKERR    : 1; // B12
00241     uint32_t          : 2; // B13-14
00242     uint32_t BFARVALID : 1; // B15
00243     uint32_t UNDEFINSTR: 1; // B16
00244     uint32_t INVSTATE   : 1; // B17
00245     uint32_t INVPC     : 1; // B18
00246     uint32_t NOCP      : 1; // B19
00247     uint32_t          : 4; // B20-23
00248     uint32_t UNALIGNED  : 1; // B24
00249     uint32_t DIVBYZERO  : 1; // B25
00250     uint32_t          : 6; // B26-31
00251 } bits;
00252     uint32_t word;
00253 } HalCrashCfsrType;
00254
00255 typedef union {
00256     struct
00257     {
00258         uint32_t          : 1; // B0
00259         uint32_t VECTTBL   : 1; // B1
00260         uint32_t          : 28; // B2-29
00261         uint32_t FORCED   : 1; // B30
00262         uint32_t DEBUGEVT  : 1; // B31
00263     } bits;
00264     uint32_t word;
00265 } HalCrashHfsrType;
00266
00267 typedef union {
00268     struct
00269     {
00270         uint32_t HALTED   : 1; // B0
00271         uint32_t BKPT     : 1; // B1
00272         uint32_t DWTRAP    : 1; // B2
00273         uint32_t VCATCH    : 1; // B3
00274         uint32_t EXTERNAL  : 1; // B4
00275         uint32_t          : 27; // B5-31
00276     } bits;
00277     uint32_t word;
00278 } HalCrashDfsrType;
00279
00280 typedef union {
00281     struct
00282     {
00283         uint32_t MISSED   : 1; // B0
00284         uint32_t RESERVED  : 1; // B1
00285         uint32_t PROTECTED : 1; // B2
00286         uint32_t WRONGSIZE : 1; // B3
00287         uint32_t          : 28; // B4-31
00288     } bits;
00289     uint32_t word;
00290 } HalCrashAfsrType;
00291
00292 #define NUM RETURNS 6
00293
00294 // Define the crash data structure
00295 typedef struct
00296 {
00297     //
00298     // The components within this first block are written by the assembly
00299     // language common fault handler, and position and order is critical.
00300     // cstartup-iar-boot-entry.s79 also relies on the position/order here.
00301     // Do not edit without also modifying that code.
00302     //
00303     uint16_t resetReason; // reason written out just before forcing a reset
00304     uint16_t resetSignature;
00305     uint32_t R0;           // processor registers
00306     uint32_t R1;
00307     uint32_t R2;
00308     uint32_t R3;

```

```

00309     uint32_t R4;
00310     uint32_t R5;
00311     uint32_t R6;
00312     uint32_t R7;
00313     uint32_t R8;
00314     uint32_t R9;
00315     uint32_t R10;
00316     uint32_t R11;
00317     uint32_t R12;
00318     uint32_t LR;
00319     uint32_t mainSP;           // main and process stack pointers
00320     uint32_t processSP;
00321     //
*****// End of the block written by the common fault handler.
00323     //
*****00324
00325     uint32_t PC;             // stacked return value (if it could be read)
00326     HalCrashxPsrType xPSR; // stacked processor status reg (if it could be read)
00327     uint32_t mainSPUsed;    // bytes used in main stack
00328     uint32_t processSPUsed; // bytes used in process stack
00329     uint32_t mainStackBottom; // address of the bottom of the stack
00330     HalCrashIcsrType icsr; // interrupt control state register
00331     HalCrashShcsrType shcsr; // system handlers control and state register
00332     HalCrashIntActiveType intActive; // irq active bit register
00333     HalCrashCfsrType cfsr; // configurable fault status register
00334     HalCrashHfsrType hfsr; // hard fault status register
00335     HalCrashDfsrType dfsr; // debug fault status register
00336     uint32_t faultAddress; // fault address register (MMAR or BFAR)
00337     HalCrashAfsrType afsr; // auxiliary fault status register
00338     uint32_t returns[NUM RETURNS]; // probable return addresses found on the
00339     stack
00340     HalCrashSpecificDataType data; // additional data specific to the crash type
00341 } HalCrashInfoType;
00342 typedef union
00343 {
00344     HalCrashInfoType crash;
00345     HalBootParamType boot;
00346 } HalResetInfoType;
00347
00348 #define RESETINFO_WORDS ((sizeof(HalResetInfoType)+3)/4)
00349
00350 extern HalResetInfoType halResetInfo;
00351
00352 void halInternalSaveAssertInfo(void);
00353
00354 #endif // DOXYGEN_SHOULD_SKIP_THIS
00355
00356 #define halResetWasCrash() \
00357         ( ( 1 << halGetResetInfo() ) & RESET_CRASH_REASON_MASK ) != 0
00358
00359 uint32_t halGetMainStackBytesUsed(void);
00360
00361 void halPrintCrashSummary(uint8_t port);
00362
00363 void halPrintCrashDetails(uint8_t port);
00364
00365 void halPrintCrashData(uint8_t port);
00366
00367 const HalAssertInfoType *halGetAssertInfo(void);
00368
00369 #endif // __EM3XX_DIAGNOSTIC_H__
00370

```

8.45 em_usb.h File Reference

```
#include <string.h>
```

```
#include <stddef.h>
#include "hal/micro/cortexm3/usb/usbconfig.h"
#include <stdint.h>
#include <stdbool.h>
#include <uchar.h>
```

Data Structures

- struct [USB_Setup_TypeDef](#)
USB Setup request package.
- struct [USB_DeviceDescriptor_TypeDef](#)
USB Device Descriptor.
- struct [USB_ConfigurationDescriptor_TypeDef](#)
USB Configuration Descriptor.
- struct [USB_InterfaceDescriptor_TypeDef](#)
USB Interface Descriptor.
- struct [USB_EndpointDescriptor_TypeDef](#)
USB Endpoint Descriptor.
- struct [USB_StringDescriptor_TypeDef](#)
USB String Descriptor.
- struct [USBD_Init_TypeDef](#)
USB Device stack initialization structure.
- struct [USBD_Callbacks_TypeDef](#)
USB Device stack callback structure.

Macros

- #define [USB_SETUP_DIR_OUT](#)
- #define [USB_SETUP_DIR_IN](#)
- #define [USB_SETUP_DIR_MASK](#)
- #define [USB_SETUP_DIR_D2H](#)
- #define [USB_SETUP_DIR_H2D](#)
- #define [USB_SETUP_TYPE_STANDARD](#)
- #define [USB_SETUP_TYPE_CLASS](#)
- #define [USB_SETUP_TYPE_VENDOR](#)
- #define [USB_SETUP_TYPE_STANDARD_MASK](#)
- #define [USB_SETUP_TYPE_CLASS_MASK](#)
- #define [USB_SETUP_TYPE_VENDOR_MASK](#)
- #define [USB_SETUP_RECIPIENT_DEVICE](#)
- #define [USB_SETUP_RECIPIENT_INTERFACE](#)
- #define [USB_SETUP_RECIPIENT_ENDPOINT](#)
- #define [USB_SETUP_RECIPIENT_OTHER](#)
- #define [GET_STATUS](#)
- #define [CLEAR_FEATURE](#)
- #define [SET_FEATURE](#)
- #define [SET_ADDRESS](#)
- #define [GET_DESCRIPTOR](#)

- #define SET_DESCRIPTOR
- #define GET_CONFIGURATION
- #define SET_CONFIGURATION
- #define GET_INTERFACE
- #define SET_INTERFACE
- #define SYNCH_FRAME
- #define USB_HID_GET_REPORT
- #define USB_HID_GET_IDLE
- #define USB_HID_SET_REPORT
- #define USB_HID_SET_IDLE
- #define USB_HID_SET_PROTOCOL
- #define USB_CDC_SETLINECODING
- #define USB_CDC_GETLINECODING
- #define USB_CDC_SETCTRLLINESTATE
- #define USB_MSD_BOTRESET
- #define USB_MSD_GETMAXLUN
- #define USB_DEVICE_DESCRIPTOR
- #define USB_CONFIG_DESCRIPTOR
- #define USB_STRING_DESCRIPTOR
- #define USB_INTERFACE_DESCRIPTOR
- #define USB_ENDPOINT_DESCRIPTOR
- #define USB_DEVICE_QUALIFIER_DESCRIPTOR
- #define USB_OTHER_SPEED_CONFIG_DESCRIPTOR
- #define USB_INTERFACE_POWER_DESCRIPTOR
- #define USB_HUB_DESCRIPTOR
- #define USB_HID_DESCRIPTOR
- #define USB_HID_REPORT_DESCRIPTOR
- #define USB_CS_INTERFACE_DESCRIPTOR
- #define USB_DEVICE_DESCSIZE
- #define USB_CONFIG_DESCSIZE
- #define USB_INTERFACE_DESCSIZE
- #define USB_ENDPOINT_DESCSIZE
- #define USB_DEVICE_QUALIFIER_DESCSIZE
- #define USB_OTHER_SPEED_CONFIG_DESCSIZE
- #define USB_HID_DESCSIZE
- #define USB_CDC_HEADER_FND_DESCSIZE
- #define USB_CDC_CALLMNG_FND_DESCSIZE
- #define USB_CDC_ACM_FND_DESCSIZE
- #define USB_EP0_SIZE
- #define USB_MAX_EP_SIZE
- #define USB_EPTYPE_CTRL
- #define USB_EPTYPE_ISOC
- #define USB_EPTYPE_BULK
- #define USB_EPTYPE_INTR
- #define USB_EP_DIR_IN
- #define USB_SETUP_PKT_SIZE
- #define USB_EPNUM_MASK
- #define USB_LANGID_ENUS
- #define USB_MAX_DEVICE_ADDRESS
- #define CONFIG_DESC_BM_REMOTEWAKEUP

- #define CONFIG_DESC_BM_SELFPOWERED
- #define CONFIG_DESC_BM_RESERVED_D7
- #define CONFIG_DESC_BM_TRANSFERTYPE
- #define CONFIG_DESC_MAXPOWER_mA(x)
- #define DEVICE_IS_SELFPOWERED
- #define REMOTE_WAKEUP_ENABLED
- #define USB_FEATURE_ENDPOINT_HALT
- #define USB_FEATURE_DEVICE_REMOTE_WAKEUP
- #define HUB_FEATURE_PORT_RESET
- #define HUB_FEATURE_PORT_POWER
- #define HUB_FEATURE_C_PORT_CONNECTION
- #define HUB_FEATURE_C_PORT_RESET
- #define HUB_FEATURE_PORT_INDICATOR
- #define USB_CLASS_CDC
- #define USB_CLASS_CDC_DATA
- #define USB_CLASS_CDC_ACM
- #define USB_CLASS_CDC_HFN
- #define USB_CLASS_CDC_CMNGFN
- #define USB_CLASS_CDC_ACMFN
- #define USB_CLASS_CDC_UNIONFN
- #define USB_CLASS_HID
- #define USB_CLASS_HID_KEYBOARD
- #define USB_CLASS_HID_MOUSE
- #define USB_CLASS_HUB
- #define USB_CLASS_MSD
- #define USB_CLASS_MSD_BOT_TRANSPORT
- #define USB_CLASS_MSD_SCSI_CMDSET
- #define USB_CLASS_MSD_CSW_CMDPASSED
- #define USB_CLASS_MSD_CSW_CMDFAILED
- #define USB_CLASS_MSD_CSW_PHASEERROR
- #define PORT_FULL_SPEED
- #define PORT_LOW_SPEED
- #define nibble2Ascii(n)
- #define STATIC_CONST_STRING_DESC(_name,...)
- #define STATIC_CONST_STRING_DESC_LANGID(_name, x, y)
- #define UBUF(x, y)
- #define STATIC_UBUF(x, y)
- #define NUM_QTIMERS

Typedefs

- typedef int(* **USB_XferCompleteCb_TypeDef**)(USB_Status_TypeDef status, uint32_t xferred, uint32_t remaining)
- typedef void(* **USBTIMER_Callback_TypeDef**)(void)
- typedef void(* **USBD_UsbResetCb_TypeDef**)(void)
- typedef void(* **USBD_SofIntCb_TypeDef**)(uint16_t sofNr)
- typedef void(* **USBD_DeviceStateChangeCb_TypeDef**)(USBD_State_TypeDef oldState, **USBD_State_TypeDef** newState)
- typedef bool(* **USBD_IsSelfPoweredCb_TypeDef**)(void)
- typedef int(* **USBD_SetupCmdCb_TypeDef**)(const **USB_Setup_TypeDef** *setup)
- typedef struct **USBD_Callbacks_TypeDef** **USBD_Callbacks_TypeDef**

Enumerations

- enum `USB_Status_TypeDef` {
 `USB_STATUS_OK`, `USB_STATUS_REQ_ERR`, `USB_STATUS_EP_BUSY`, `USB_STATUS_REQ_UNHANDLED`,
 `USB_STATUS_ILLEGAL`, `USB_STATUS_EP_STALLED`, `USB_STATUS_EP_ABORTED`, `USB_STATUS_EP_ERROR`,
 `USB_STATUS_EP_NAK`, `USB_STATUS_DEVICE_UNCONFIGURED`, `USB_STATUS_DEVICE_SUSPENDED`, `USB_STATUS_DEVICE_RESET`,
 `USB_STATUS_TIMEOUT`, `USB_STATUS_DEVICE_REMOVED`, `USB_STATUS_HC_BUSY`, `USB_STATUS_DEVICE_MALFUNCTION`,
 `USB_STATUS_PORT_OVERCURRENT` }
- enum `USBD_State_TypeDef` {
 `USBD_STATE_NONE`, `USBD_STATE_ATTACHED`, `USBD_STATE_POWERED`, `USBD_STATE_DEFAULT`,
 `USBD_STATE_ADDRESSED`, `USBD_STATE_CONFIGURED`, `USBD_STATE_SUSPENDED`, `USBD_STATE_LASTMARKER` }

Functions

- void `USBTIMER_DelayMs` (uint32_t msec)
- void `USBTIMER_DelayUs` (uint32_t usec)
- void `USBTIMER_Init` (void)
- void `USBTIMER_Start` (uint32_t id, uint32_t timeout, `USBTIMER_Callback_TypeDef` callback)
- void `USBTIMER_Stop` (uint32_t id)
- void `USBD_AbortAllTransfers` (void)
- int `USBD_AbortTransfer` (int epAddr)
- void `USBD_Connect` (void)
- void `USBD_Disconnect` (void)
- bool `USBD_EpIsBusy` (int epAddr)
- `USBD_State_TypeDef USBD_GetUsbState` (void)
- const char * `USBD_GetUsbStateName` (`USBD_State_TypeDef` state)
- int `USBD_Init` (const `USBD_Init_TypeDef` *p)
- int `USBD_Read` (int epAddr, void *data, int byteCount, `USB_XferCompleteCb_TypeDef` callback)
- int `USBD_RemoteWakeup` (void)
- bool `USBD_SafeToEnterEM2` (void)
- int `USBD_StallEp` (int epAddr)
- void `USBD_Stop` (void)
- int `USBD_UnStallEp` (int epAddr)
- int `USBD_Write` (int epAddr, void *data, int byteCount, `USB_XferCompleteCb_TypeDef` callback)
- void `usbSuspendDsr` (void)
- void `usbTxData` (void)
- void `usbForceTxData` (uint8_t *data, uint8_t length)
- void `halInternalUart3RxIsr` (uint8_t *rxData, uint8_t length)

8.45.1 Detailed Description

USB protocol stack library API for EFM32.

Author

Nathaniel Ting

Version

3.20.3

Definition in file [em_usb.h](#).

8.45.2 Macro Definition Documentation

8.45.2.1 #define NUM_QTIMERS

Definition at line [556](#) of file [em_usb.h](#).

8.45.3 Function Documentation

8.45.3.1 void usbTxData(void)

8.45.3.2 void usbForceTxData(uint8_t * data, uint8_t length)

8.45.3.3 void halInternalUart3RxIsr(uint8_t * rxData, uint8_t length)

8.46 em_usb.h

```

00001 /*****
00012 #ifndef __EM_USB_H
00013 #define __EM_USB_H
00014
00015 #include <string.h>
00016 #include <stdbool.h>
00017
00018
00019 #if !defined(CORTEXM3_EM35X_USB) \
00020   || (EMBER_SERIAL3_MODE == EMBER_SERIAL_UNUSED) \
00021   || defined(USB_MSD) \
00022   || defined(USB_HID)
00023 #define EM_SERIAL3_ENABLED 0
00024 #else
00025 #define EM_SERIAL3_ENABLED 1
00026 #endif
00027
00028
00029 #if EM_SERIAL3_ENABLED
00030   #include "hal/micro/cortexm3/usb/cdc/usbconfig.h"
00031 #elif defined(USB_MSD)
00032   #include "hal/micro/cortexm3/usb/msd/usbconfig.h"
00033 #elif defined(USB_HID)
00034   #include "hal/micro/cortexm3/usb/hid/usbconfig.h"
00035 #else
00036   #include "hal/micro/cortexm3/usb/usbconfig.h"
00037 #endif
00038
00039 #if defined( USB_USE_PRINTF )
00040 #include <stdio.h>
00041 #endif
00042

```

```

00043
00044 #if USB_SELFPWD_STATE==0
00045     #define USB_SUSPEND
00046 #endif
00047
00048
00049 #ifdef __CC_ARM
00050 #pragma anon_unions
00051 #endif
00052
00053 #include <stdint.h>
00054 #include <stdbool.h>
00055 typedef unsigned short char16_t;
00056 // #define false 0;
00057 // #define true 1;
00058
00059
00060 /* Debug buffer */
00061 #if defined (USB_DEBUG)
00062     #include "app/util/serial/serial.h"
00063     #include <stdio.h>
00064     extern char *DEBUG_BUFFER;
00065     void USBD_PrintDebug(void);
00066     #define USB_PRINTDEBUG(a,b,c) DEBUG_BUFFER += sprintf(a,b,c)
00067 #endif
00068
00069
00070 /* SETUP request, direction of data stage */
00081 #define USB_SETUP_DIR_OUT      0
00082 #define USB_SETUP_DIR_IN       1
00083 #define USB_SETUP_DIR_MASK    0x80
00084 #define USB_SETUP_DIR_D2H     0x80
00085 #define USB_SETUP_DIR_H2D     0x00
00086
00087 /* SETUP request type */
00088 #define USB_SETUP_TYPE_STANDARD 0
00089 #define USB_SETUP_TYPE_CLASS   1
00090 #define USB_SETUP_TYPE_VENDOR  2
00091 #define USB_SETUP_TYPE_STANDARD_MASK 0x00
00092 #define USB_SETUP_TYPE_CLASS_MASK 0x20
00093 #define USB_SETUP_TYPE_VENDOR_MASK 0x40
00094
00095 /* SETUP request recipient */
00096 #define USB_SETUP_RECIPIENT_DEVICE 0
00097 #define USB_SETUP_RECIPIENT_INTERFACE 1
00098 #define USB_SETUP_RECIPIENT_ENDPOINT 2
00099 #define USB_SETUP_RECIPIENT_OTHER   3
00100
00101 /* SETUP standard request codes for Full Speed devices */
00102 #define GET_STATUS            0
00103 #define CLEAR_FEATURE         1
00104 #define SET_FEATURE           3
00105 #define SET_ADDRESS           5
00106 #define GET_DESCRIPTOR        6
00107 #define SET_DESCRIPTOR        7
00108 #define GET_CONFIGURATION     8
00109 #define SET_CONFIGURATION      9
00110 #define GET_INTERFACE          10
00111 #define SET_INTERFACE          11
00112 #define SYNCH_FRAME           12
00113
00114 /* SETUP class request codes */
00115 #define USB_HID_GET_REPORT    0x01
00116 #define USB_HID_GET_IDLE      0x02
00117 #define USB_HID_SET_REPORT    0x09
00118 #define USB_HID_SET_IDLE      0x0A
00119 #define USB_HID_SET_PROTOCOL  0x0B
00120 #define USB_CDC_SETLINECODING 0x20
00121 #define USB_CDC_GETLINECODING 0x21
00122 #define USB_CDC_SETCTRLLINESTATE 0x22
00123 #define USB_MSD_BOTRESET      0xFF
00124 #define USB_MSD_GETMAXLUN     0xFE
00125
00126 /* SETUP command GET/SET_DESCRIPTOR descriptor types */
00127 #define USB_DEVICE_DESCRIPTOR 1
00128 #define USB_CONFIG_DESCRIPTOR 2
00129 #define USB_STRING_DESCRIPTOR 3
00130 #define USB_INTERFACE_DESCRIPTOR 4
00131 #define USB_ENDPOINT_DESCRIPTOR 5
00132 #define USB_DEVICE_QUALIFIER_DESCRIPTOR 6
00133 #define USB_OTHER_SPEED_CONFIG_DESCRIPTOR 7
00134 #define USB_INTERFACE_POWER_DESCRIPTOR 8
00135 #define USB_HUB_DESCRIPTOR    0x29
00136 #define USB_HID_DESCRIPTOR   0x21
00137 #define USB_HID_REPORT_DESCRIPTOR 0x22
00138 #define USB_CS_INTERFACE_DESCRIPTOR 0x24

```

```

00140 #define USB_DEVICE_DESCSIZE          18
00141 #define USB_CONFIG_DESCSIZE        9
00142 #define USB_INTERFACE_DESCSIZE      9
00143 #define USB_ENDPOINT_DESCSIZE       7
00144 #define USB_DEVICE_QUALIFIER_DESCSIZE 10
00145 #define USB_OTHER_SPEED_CONFIG_DESCSIZE 9
00146 #define USB_HID_DESCSIZE           9
00147 #define USB_CDC_HEADER_FND_DESCSIZE 5
00148 #define USB_CDC_CALLMNG_FND_DESCSIZE 5
00149 #define USB_CDC_ACN_FND_DESCSIZE    4
00151 /* Misc. USB definitions */
00152 #define USB_EP0_SIZE                8
00153 #define USB_MAX_EP_SIZE             64
00154 #define USB_EPTYPE_CTRL             0
00155 #define USB_EPTYPE_ISOC              1
00156 #define USB_EPTYPE_BULK              2
00157 #define USB_EPTYPE_INTR              3
00158 #define USB_EP_DIR_IN              0x80
00159 #define USB_SETUP_PKT_SIZE          8
00160 #define USB_EPNUM_MASK              0x0F
00161 #define USB_LANGID_ENUS            0x0409
00162 #define USB_MAX_DEVICE_ADDRESS     127
00164 #define CONFIG_DESC_BM_REMOTEWAKEUP 0x20
00165 #define CONFIG_DESC_BM_SELFPOWERED   0x40
00166 #define CONFIG_DESC_BM_RESERVED_D7  0x80
00167 #define CONFIG_DESC_BM_TRANSFERTYPE 0x03
00168 #define CONFIG_DESC_MAXPOWER_mA(x) (((x)+1)/2)
00170 #define DEVICE_IS_SELFPOWERED        0x0001
00171 #define REMOTE_WAKEUP_ENABLED        0x0002
00172 #define USB_FEATURE_ENDPOINT_HALT   0
00173 #define USB_FEATURE_DEVICE_REMOTE_WAKEUP 1
00175 #define HUB_FEATURE_PORT_RESET      4
00176 #define HUB_FEATURE_PORT_POWER      8
00177 #define HUB_FEATURE_C_PORT_CONNECTION 16
00178 #define HUB_FEATURE_C_PORT_RESET    20
00179 #define HUB_FEATURE_PORT_INDICATOR 22
00181 #define USB_CLASS_CDC               2
00182 #define USB_CLASS_CDC_DATA          0x0A
00183 #define USB_CLASS_CDC_ACM          2
00184 #define USB_CLASS_CDC_HFN          0
00185 #define USB_CLASS_CDC_CMNGFN       1
00186 #define USB_CLASS_CDC_ACDFN        2
00187 #define USB_CLASS_CDC_UNIONFN      6
00189 #define USB_CLASS_HID              3
00190 #define USB_CLASS_HID_KEYBOARD     1
00191 #define USB_CLASS_HID_MOUSE        2
00193 #define USB_CLASS_HUB              9
00195 #define USB_CLASS_MSD              8
00196 #define USB_CLASS_MSD_BOT_TRANSPORT 0x50
00197 #define USB_CLASS_MSD_SCSI_CMDSET   6
00198 #define USB_CLASS_MSD_CS_CMPDFAILED 0
00199 #define USB_CLASS_MSD_CS_CSW_FAILED 1
00200 #define USB_CLASS_MSD_CS_PHASEERROR 2
00202 #define PORT_FULL_SPEED            1
00203 #define PORT_LOW_SPEED             2
00208 /**** ----- Helper Macros ----- ****/
00209
00211 #define EFM32_MIN(a, b)    ((a) < (b) ? (a) : (b))
00212
00213 #define EFM32_MAX(a, b)    ((a) > (b) ? (a) : (b))
00214
00215 #if !defined(__GNUC__)
00216
00217 #define STRINGIZE(X) #X
00218 #define EFM32_PACK_START(X) __Pragma( STRINGIZE( pack( X ) ) )
00219 #define EFM32_PACK_END()    __Pragma( "pack()" )
00220 #define __attribute__(...)
00221
00222 #ifdef __CC_ARM
00223
00224 #define EFM32_ALIGN(X) __align(X)
00225 #endif
00226 #ifdef __ICCARM__
00227
00228 #define EFM32_ALIGN(X) __Pragma( STRINGIZE( data_alignment=X ) )
00229 #endif
00230
00231 #else // !defined(__GNUC__)
00232
00233 #define EFM32_PACK_START( x )

```

```

00234 #define EFM32_PACK_END()
00235 #define EFM32_ALIGN(X)
00236
00237 #endif // !defined(__GNUC__)
00238
00241 #if defined( __GNUC__ )                                /* GCC compilers */
00242 #if defined( __CHAR16_TYPE__ )
00243     typedef __CHAR16_TYPE__ char16_t;
00244 #else
00245     typedef unsigned short char16_t;
00246 #endif
00247
00248 #elif defined( __ICCARM__ )                            /* IAR compiler */
00249 #include <uchar.h>
00250
00251 #elif defined( __CC_ARM )                             /* MDK-ARM compiler */
00252     typedef unsigned short char16_t;
00253 #endif
00254
00255 #define nibble2Ascii(n) ((n) + (((n) < 10) ? '0' : 'A' - 10));
00256
00257 #define STATIC_CONST_STRING_DESC( _name, ... )
00258 EFM32_PACK_START( 1 )
00259     typedef struct
00260 {
00261     uint8_t len;
00262     uint8_t type;
00263     char16_t name[ 1 + sizeof( (char16_t[]) {__VA_ARGS__} ) / 2];
00264 } __attribute__ ((packed)) _##_name;
00265 EFM32_PACK_END()
00266 EFM32_ALIGN( 4 )
00267 EFM32_PACK_START( 1 )
00268 static const _##_name _name =
00269 {
00270     .len = sizeof( _##_name ) - 4,
00271     .type = USB_STRING_DESCRIPTOR,
00272     .name = {__VA_ARGS__},
00273     .name[ ( ( sizeof( _##_name ) - 2 ) / 2 ) - 1 ] = '\0'
00274 }
00275 EFM32_PACK_END()
00276
00278 #define STATIC_CONST_STRING_DESC_LANGID( _name, x, y )
00279 EFM32_PACK_START( 1 )
00280     typedef struct
00281 {
00282     uint8_t len;
00283     uint8_t type;
00284     uint8_t name[ 2 ];
00285 } _##_name;
00286 EFM32_PACK_END()
00287 EFM32_ALIGN( 4 )
00288 EFM32_PACK_START( 1 )
00289 static const _##_name _name __attribute__ ((aligned(4)))=
00290 {
00291     .len = 4,
00292     .type = USB_STRING_DESCRIPTOR,
00293     .name = { y, x }
00294 }
00295 EFM32_PACK_END()
00296
00297 EFM32_PACK_END()
00298
00299
00300 #if !defined(__GNUC__)
00301 #define UBUF( x, y ) EFM32_ALIGN( 4 )      uint8_t x[((y)+3)&~3]
00302 #define STATIC_UBUF( x, y ) EFM32_ALIGN( 4 ) static uint8_t x[((y)+3)&~3]
00303 #else
00304 #define UBUF( x, y )      uint8_t x[((y)+3)&~3] __attribute__
00305             ((aligned(4)))
00306 #define STATIC_UBUF( x, y ) static uint8_t x[((y)+3)&~3] __attribute__
00307             ((aligned(4)))
00308 #endif
00309
00310 #define STATIC_UBUF( x, y ) static uint8_t x[((y)+3)&~3] __attribute__
00311             ((aligned(4)))
00312
00313 #endif
00314
00315 #endif
00316
00317 typedef enum
00318 {
00319     /* NOTE: Please keep in sync with table errMsg[] in em_usbhal.c */
00320     USB_STATUS_OK          = 0,
00321     USB_STATUS_REQ_ERR     = -1,
00322     USB_STATUS_EP_BUSY     = -2,
00323     USB_STATUS_REQ_UNHANDLED = -3,
00324     USB_STATUS_ILLEGAL    = -4,
00325 }
```

```

00326     USB_STATUS_EP_STALLED      = -5,
00327     USB_STATUS_EP_ABORTED      = -6,
00328     USB_STATUS_EP_ERROR        = -7,
00329     USB_STATUS_EP_NAK          = -8,
00330     USB_STATUS_DEVICE_UNCONFIGURED = -9,
00331     USB_STATUS_DEVICE_SUSPENDED   = -10,
00332     USB_STATUS_DEVICE_RESET      = -11,
00333     USB_STATUS_TIMEOUT          = -12,
00334     USB_STATUS_DEVICE_REMOVED    = -13,
00335     USB_STATUS_HC_BUSY          = -14,
00336     USB_STATUS_DEVICE_MALFUNCTION = -15,
00337     USB_STATUS_PORT_OVERCURRENT = -16,
00338 } USB_Status_TypeDef;
00343 /***** */
00350 typedef enum
00351 {
00352     USBD_STATE_NONE      = 0,
00353     USBD_STATE_ATTACHED   = 1,
00354     USBD_STATE_POWERED    = 2,
00355     USBD_STATE_DEFAULT     = 3,
00356     USBD_STATE_ADDRESSED   = 4,
00357     USBD_STATE_CONFIGURED  = 4,
00358     USBD_STATE_SUSPENDED    = 6,
00359     USBD_STATE_LASTMARKER  = 7,
00360 } USBD_State_TypeDef;
00370 EFM32_PACK_START( 1 )
00371 typedef struct
00372 {
00373     union
00374     {
00375         struct
00376         {
00377             union
00378             {
00379                 struct
00380                 {
00381                     uint8_t Recipient : 5;
00382                     uint8_t Type      : 2;
00383                     uint8_t Direction : 1;
00384                 };
00385                 uint8_t bmRequestType;
00386             };
00387             uint8_t bRequest;
00388             uint16_t wValue;
00389             uint16_t wIndex;
00390             uint16_t wLength;
00391         };
00392         uint32_t dw[2];
00393     };
00394 } __attribute__ ((packed)) USB_Setup_TypeDef;
00395 EFM32_PACK_END()
00396
00398 EFM32_PACK_START( 1 )
00399 typedef struct
00400 {
00401     uint8_t bLength;
00402     uint8_t bDescriptorType;
00403     uint16_t bcdUSB;
00405     uint8_t bDeviceClass;
00406     uint8_t bDeviceSubClass;
00407     uint8_t bDeviceProtocol;
00408     uint8_t bMaxPacketSize0;
00409     uint16_t idVendor;
00410     uint16_t idProduct;
00411     uint16_t bcdDevice;
00412     uint8_t iManufacturer;
00413     uint8_t iProduct;
00414     uint8_t iSerialNumber;
00416     uint8_t bNumConfigurations;
00417 } __attribute__ ((packed)) USB_DeviceDescriptor_TypeDef
00418 EFM32_PACK_START( 1 )
00419
00420
00422 EFM32_PACK_START( 1 )
00423 typedef struct
00424 {
00425     uint8_t bLength;
00426     uint8_t bDescriptorType;
00427     uint16_t wTotalLength;

```

```

00432     uint8_t bNumInterfaces;
00434     uint8_t bConfigurationValue;
00437     uint8_t iConfiguration;
00439     uint8_t bmAttributes;
00444     uint8_t bMaxPower;
00446 } __attribute__ ((packed)) USB_ConfigurationDescriptor_TypeDef
00447 ;
00447 EFM32_PACK_END()
00448
00450 EFM32_PACK_START( 1 )
00451 typedef struct
00452 {
00453     uint8_t bLength;
00454     uint8_t bDescriptorType;
00455     uint8_t bInterfaceNumber;
00458     uint8_t bAlternateSetting;
00460     uint8_t bNumEndpoints;
00463     uint8_t bInterfaceClass;
00468     uint8_t bInterfaceSubClass;
00474     uint8_t bInterfaceProtocol;
00484     uint8_t iInterface;
00486 } __attribute__ ((packed)) USB_InterfaceDescriptor_TypeDef
00487 ;
00487 EFM32_PACK_END()
00488
00490 EFM32_PACK_START( 1 )
00491 typedef struct
00492 {
00493     uint8_t bLength;
00494     uint8_t bDescriptorType;
00495     uint8_t bEndpointAddress;
00496     uint8_t bmAttributes;
00497     uint16_t wMaxPacketSize;
00498     uint8_t bInterval;
00499 } __attribute__ ((packed)) USB_EndpointDescriptor_TypeDef
00500 ;
00500 EFM32_PACK_END()
00501
00503 EFM32_PACK_START( 1 )
00504 typedef struct
00505 {
00506     uint8_t len;
00507     uint8_t type;
00508     char16_t name[];
00509 } __attribute__ ((packed)) USB_StringDescriptor_TypeDef
00510 ;
00511 EFM32_PACK_END()
00512 *****
00535 typedef int (*USB_XferCompleteCb_TypeDef)(
00536     USB_Status_TypeDef status, uint32_t xferred, uint32_t
00537     remaining );
00536
00537 *****
00545 typedef void (*USBTIMER_Callback_TypeDef)( void );
00546
00547 void USBTIMER_DelayMs( uint32_t msec );
00548 void USBTIMER_DelayUs( uint32_t usec );
00549 void USBTIMER_Init( void );
00550 void USBTIMER_Start( uint32_t id, uint32_t
00551     timeout, USBTIMER_Callback_TypeDef callback );
00551 void USBTIMER_Stop( uint32_t id );
00554 #if defined( NUM_APP_TIMERS )
00555     #define NUM_QTIMERS ( NUM_APP_TIMERS )
00556 #else
00557     #define NUM_QTIMERS 0
00558 #endif
00559
00560
00563 *** ----- DEVICE mode API definitions ----- ***
00564
00565 *****
00571 typedef void (*USBD_UsbResetCb_TypeDef)( void );
00572
00573 *****
00583 typedef void (*USBD_SofIntCb_TypeDef)( uint16_t sofNr );
00584
00585 *****
00598 typedef void (*USBD_DeviceStateChangeCb_TypeDef
00599     ( USBD_State_TypeDef oldState, USBD_State_TypeDef

```

```

        newState );
00599 /***** */
00600 typedef bool (*USBD_IsSelfPoweredCb_TypeDef) ( void
00612 );
00613 ;
00614 /***** */
00615 typedef int (*USBD_SetupCmdCb_TypeDef) ( const
00616 USB_Setup_TypeDef *setup );
00617 ;
00618 struct USBD_Callbacks_TypeDef;
00619 typedef struct USBD_Callbacks_TypeDef const *
00620 USBD_Callbacks_TypeDef_Pointer;
00621 typedef struct
00622 {
00623     const USB_DeviceDescriptor_TypeDef *
00624     deviceDescriptor;
00625     const uint8_t                                *configDescriptor;
00626     const void * const                         *stringDescriptors;
00627     const uint8_t                                numberofStrings;
00628     const uint8_t                                *bufferingMultiplier;
00629     USBD_Callbacks_TypeDef_Pointer            callbacks;
00630     const uint32_t                                reserved;
00631 } USBD_Init_TypeDef;
00632 ;
00633 typedef struct USBD_Callbacks_TypeDef
00634 {
00635     const USBD_UsbResetCb_TypeDef           usbReset
00636     ;
00637     const USBD_DeviceStateChangeCb_TypeDef
00638     usbStateChange;
00639     const USBD_SetupCmdCb_TypeDef           setupCmd
00640     ;
00641     const USBD_IsSelfPoweredCb_TypeDef
00642     isSelfPowered;
00643     const USBD_SofIntCb_TypeDef           sofInt;
00644 };
00645 } USBD_Callbacks_TypeDef;
00646 ;
00647 /* --- DEVICE mode API --- */
00648
00649 void          USBD_AbortAllTransfers( void );
00650 int           USBD_AbortTransfer(      int epAddr );
00651 void           USBD_Connect(          void );
00652 void           USBD_Disconnect(         void );
00653 bool          USBD_EpIsBusy(         int epAddr );
00654 USBD_State_TypeDef USBD_GetUsbState(void);
00655 const char * USBD_GetUsbStateName(USBD_State_TypeDef state );
00656 int           USBD_Init(          const USBD_Init_TypeDef
00657     *p );
00658 int           USBD_Read(          int epAddr, void *data,
00659     int byteCount, USB_XferCompleteCb_TypeDef callback );
00660 int           USBD_RemoteWakeUp(       void );
00661 bool          USBD_SafeToEnterEM2(    void );
00662 int           USBD_StallEp(         int epAddr );
00663 void           USBD_Stop(          void );
00664 int           USBD_UnStallEp(       int epAddr );
00665 int           USBD_Write(          int epAddr, void *data,
00666     int byteCount, USB_XferCompleteCb_TypeDef callback );
00667 ;
00668 /* --- UART Access --- */
00669 void usbSuspendDsr(void);
00670 ;
00671 void usbTxData (void);
00672 void usbForceTxData (uint8_t *data, uint8_t length);
00673 void halInternalUart3RxIsr(uint8_t *rxData, uint8_t length
00674 );
00675 ;
00676 #endif /* __EM_USB_H */
00677

```

8.47 em_usbd.c File Reference

```
#include <PLATFORM_HEADER>
#include "stack/include/ember.h"
#include "hal/hal.h"
#include "em_usb.h"
#include "em_usbhal.h"
#include "em_usbtypes.h"
#include "em_usbd.h"
#include "app/util/serial/serial.h"
```

Functions

- void [USBD_AbortAllTransfers](#) (void)
- int [USBD_AbortTransfer](#) (int epAddr)
- void [USBD_Connect](#) (void)
- void [USBD_Disconnect](#) (void)
- [USBD_State_TypeDef USBD_GetUsbState](#) (void)
- const char * [USBD_GetUsbStateName](#) ([USBD_State_TypeDef](#) state)
- bool [USBD_EpIsBusy](#) (int epAddr)
- int [USBD_StallEp](#) (int epAddr)
- int [USBD_UnStallEp](#) (int epAddr)
- void [USBD_Stop](#) (void)
- int [USBD_Init](#) (const [USBD_Init_TypeDef](#) *p)
- int [USBD_Write](#) (int epAddr, void *data, int byteCount, [USB_XferCompleteCb_TypeDef](#) callback)
- int [USBD_Read](#) (int epAddr, void *data, int byteCount, [USB_XferCompleteCb_TypeDef](#) callback)
- void [usbSuspendDsr](#) (void)
- int [USBD_RemoteWakeup](#) (void)

8.47.1 Detailed Description

USB protocol stack library, device API.

Author

Nathaniel Ting

Version

3.20.3

Definition in file [em_usbd.c](#).

8.48 em_usbd.c

```

00001 /*****
00002 #include PLATFORM_HEADER
00003 #include "stack/include/ember.h"
00004 #include "hal/hal.h"
00011
00012 #if CORTEXM3_EM35X_USB
00013 #include "em_usb.h"
00014 #include "em_usbdhal.h"
00015
00016 #include "em_usbtypes.h"
00017 #include "em_usbd.h"
00018
00019 #include "app/util/serial/serial.h"
00020
00021
00022
00025 static USBD_Device_TypeDef device;
00026 USBD_Device_TypeDef *dev = &device;
00027
00028 static const char *stateNames[] =
00029 {
00030     [ USBD_STATE_NONE ] = "NONE",
00031     [ USBD_STATE_ATTACHED ] = "ATTACHED",
00032     [ USBD_STATE_POWERED ] = "POWERED",
00033     [ USBD_STATE_DEFAULT ] = "DEFAULT",
00034     [ USBD_STATE_ADDRESSED ] = "ADDRESSED",
00035     [ USBD_STATE_CONFIGURED ] = "CONFIGURED",
00036     [ USBD_STATE_SUSPENDED ] = "SUSPENDED",
00037     [ USBD_STATE_LASTMARKER ] = "UNDEFINED"
00038 };
00039
00040 /*****
00056 void USBD_AbortAllTransfers( void )
00057 {
00058     ATOMIC(
00059         USBDHAL_AbortAllTransfers( USB_STATUS_EP_ABORTED );
00060     )
00061 }
00062
00063 /*****
00070 int USBD_AbortTransfer( int epAddr )
00071 {
00072     USB_XferCompleteCb_TypeDef callback;
00073     USBD_Ep_TypeDef *ep = USBD_GetEpFromAddr( epAddr );
00074
00075
00076
00077     // nUSBD_AbortTransfer(), Illegal request
00078     assert (ep!=NULL);
00079
00080     // nUSBD_AbortTransfer(), Illegal endpoint
00081     assert (ep->num!=0);
00082
00083
00084     DECLARE_INTERRUPT_STATE;
00085     DISABLE_INTERRUPTS();
00086     if ( ep->state == D_EP_IDLE )
00087     {
00088         RESTORE_INTERRUPTS();
00089         return USB_STATUS_OK;
00090     }
00091
00092     // USBD_AbortEp( ep );
00093
00094     ep->state = D_EP_IDLE;
00095     if ( ep->xferCompleteCb )
00096     {
00097         callback = ep->xferCompleteCb;
00098         ep->xferCompleteCb = NULL;
00099
00100         if ( ( dev->lastState == USBD_STATE_CONFIGURED ) &&
00101             ( dev->state == USBD_STATE_ADDRESSED ) )
00102         {
00103             USBDHAL_DeactivateEp( ep );
00104         }
00105
00106     // DEBUG_TRACE_ABORT( USB_STATUS_EP_ABORTED );

```

```

00107     callback( USB_STATUS_EP_ABORTED, ep->xferred, ep->
00108     remaining );
00109
00110     RESTORE_INTERRUPTS();
00111     return USB_STATUS_OK;
00112 }
00113
00114 /*****
00115 ****
00116 ****
00117 ****
00118 ****
00119 ****
00120 ****
00121 ****
00122 ****
00123 ****
00124 ****
00125 ****
00126 ****
00127 ****
00128 ****
00129 ****
00130 ****
00131 ****
00132 ****
00133 ****
00134 ****
00135 ****
00136 ****
00137 ****
00138 ****
00139 ****
00140 ****
00141 ****
00142 ****
00143 ****
00144 void USBD_SetUsbState( USBD_State_TypeDef newState )
00145 {
00146     USBD_State_TypeDef currentState;
00147
00148     currentState = dev->state;
00149     if ( newState == USBD_STATE_SUSPENDED )
00150     {
00151         dev->savedState = currentState;
00152     }
00153
00154     dev->lastState = dev->state;
00155     dev->state = newState;
00156
00157     if ( ( dev->callbacks->usbStateChange ) &&
00158         ( currentState != newState ) )
00159     {
00160         dev->callbacks->usbStateChange( currentState, newState );
00161     }
00162 }
00163
00164 /*****
00165 ****
00166 ****
00167 ****
00168 ****
00169 ****
00170 ****
00171 ****
00172 ****
00173 ****
00174 ****
00175 ****
00176 ****
00177 ****
00178 ****
00179 ****
00180 ****
00181 ****
00182 ****
00183 ****
00184 ****
00185 ****
00186 ****
00187 ****
00188 const char *USBD_GetUsbStateName( USBD_State_TypeDef
00189 state )
00190 {
00191     if ( state > USBD_STATE_LASTMARKER )
00192         state = USBD_STATE_LASTMARKER;
00193
00194     return stateNames[ state ];
00195 }
00196
00197 /*****
00198 ****
00199 ****
00200 ****
00201 ****
00202 ****
00203 ****
00204 ****
00205 ****
00206 ****
00207 ****
00208 ****
00209     USBD_Ep_TypeDef *ep = USBD_GetEpFromAddr( epAddr );
00210
00211 // USBD_EpIsBusy(), Illegal endpoint
00212 assert (ep!=NULL);
00213
00214 if ( ep->state == D_EP_IDLE )
00215     return false;
00216
00217 return true;
00218 }
00219
00220 /*****
00221 ****
00222 ****
00223 ****
00224 ****
00225 ****
00226 ****
00227 ****
00228 ****
00229 ****
00230 ****
00231 ****
00232 ****
00233 ****
00234 ****

```

```

00235
00236 #ifdef USB_DEBUG_STALL
00237     DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "Stalling EP%d", ep->num);
00238     if (ep->in)
00239         DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "in\r\n");
00240     else
00241         DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "out\r\n");
00242 #endif
00243
00244 // USBD_StallEp(), Illegal request
00245 assert (ep!=NULL);
00246
00247 // USBD_StallEp(), Illegal endpoint
00248 // assert (ep->num!=0);
00249
00250 ATOMIC(
00251     retVal = USBDHAL_StallEp( ep );
00252 )
00253
00254 if ( retVal != USB_STATUS_OK )
00255 {
00256     retVal = USB_STATUS_ILLEGAL;
00257 }
00258
00259 return retVal;
00260 }
00261
00262
00263 /*****
00273 int USBD_UnStallEp( int epAddr )
00274 {
00275     USB_Status_TypeDef retVal;
00276     USBD_Ep_TypeDef *ep = USBD_GetEpFromAddr( epAddr );
00277
00278     #ifdef USB_DEBUG_STALL
00279         DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "Unstalling EP%d", ep->num);
00280         if (ep->in)
00281             DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "in\r\n");
00282         else
00283             DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "out\r\n");
00284 #endif
00285
00286
00287 // USBD_StallEp(), Illegal request
00288 assert (ep!=NULL);
00289 // USBD_StallEp(), Illegal endpoint
00290 // assert (ep->num!=0);
00291
00292
00293 ATOMIC(
00294     retVal = USBDHAL_UnStallEp( ep );
00295 )
00296
00297 if ( retVal != USB_STATUS_OK )
00298 {
00299     retVal = USB_STATUS_ILLEGAL;
00300 }
00301
00302 return retVal;
00303 }
00304
00305 /*****
00313 void USBD_Stop( void )
00314 {
00315     USBD_Disconnect();
00316     USBHAL_DisableGlobalInt();
00317     USBD_SetUsbState( USBD_STATE_NONE );
00318 }
00319
00320
00321 /*****
00337 int USBD_Init( const USBD_Init_TypeDef *p )
00338 {
00339
00340     int numEps;
00341     USBD_Ep_TypeDef *ep;
00342     uint8_t txFifoNum;
00343     uint8_t *conf, *confEnd;
00344     USB_EndpointDescriptor_TypeDef *epd;
00345     uint32_t totalRxFifoSize, totalTxFifoSize, numInEps, numOutEps;

```

```

00346
00347     USBTIMER_Init();
00348
00349     MEMSET( dev, 0, sizeof( USBD_Device_TypeDef ) );
00350
00351     dev->setup          = dev->setupPkt;
00352     dev->deviceDescriptor = p->deviceDescriptor;
00353     dev->configDescriptor = (USB_ConfigurationDescriptor_TypeDef
00354                                         *)
00355     dev->stringDescriptors = p->stringDescriptors;
00356     dev->numberOfStrings   = p->numberOfStrings;
00357     dev->state             = USBD_STATE_LASTMARKER;
00358     dev->savedState        = USBD_STATE_NONE;
00359     dev->lastState          = USBD_STATE_NONE;
00360     dev->callbacks         = p->callbacks;
00361 #if USB_REMOTEWKUPEN_STATE
00362     dev->remoteWakeupEnabled = true;
00363 #else
00364     dev->remoteWakeupEnabled = false;
00365 #endif
00366
00367
00368 /* Initialize EP0 */
00369
00370     ep           = &dev->ep[ 0 ];
00371     ep->in       = false;
00372     ep->buf      = NULL;
00373     ep->num      = 0;
00374     ep->mask     = 1;
00375     ep->addr     = 0;
00376     ep->type     = USB_EPTYPE_CTRL;
00377     ep->txFifoNum = 0;
00378     ep->packetSize = USB_EP0_SIZE;
00379     ep->remaining = 0;
00380     ep->xferred   = 0;
00381     ep->state     = D_EP_IDLE;
00382     ep->xferCompleteCb = NULL;
00383     ep->fifoSize  = USB_EP0_SIZE / 4;
00384
00385     totalTxFifoSize = ep->fifoSize * p->bufferingMultiplier[ 0
00386 ];
00386     totalRxFifoSize = (ep->fifoSize + 1) * p->bufferingMultiplier
00387 [ 0 ];
00388 /* Parse configuration descriptor */
00389     numEps = 0;
00390     numInEps = 0;
00391     numOutEps = 0;
00392     conf = (uint8_t*)dev->configDescriptor;
00393     confEnd = conf + dev->configDescriptor->wTotalLength;
00394
00395     txFifoNum = 1;
00396
00397
00398 #ifdef USB_DEBUG
00399 // DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "ep-->\tin\ttype\tpacketSize\r\n");
00400 #endif
00401
00402     while ( conf < confEnd )
00403     {
00404         // USBD_Init(), Illegal configuration descriptor
00405         assert( *conf );
00406
00407         if ( *(conf + 1) == USB_ENDPOINT_DESCRIPTOR )
00408         {
00409             numEps++;
00410             epd = (USB_EndpointDescriptor_TypeDef*)conf
00411 ;
00412
00413             ep           = &dev->ep[ numEps ];
00414             ep->in       = ( epd->bEndpointAddress &
00415                               USB_SETUP_DIR_MASK ) != 0;
00415             ep->buf      = NULL;
00416             ep->addr     = epd->bEndpointAddress;
00417             ep->num      = epd->addr & USB_EPNUM_MASK;
00418             ep->mask     = 1 << ep->num;
00419             ep->type     = epd->bmAttributes &
00420 CONFIG_DESC_BM_TRANSFERTYPE;

```

```

00420     ep->packetSize      = epd->wMaxPacketSize;
00421     ep->remaining       = 0;
00422     ep->xferred         = 0;
00423     ep->state           = D_EP_IDLE;
00424     ep->xferCompleteCb = NULL;
00425
00426
00427     if ( ep->in )
00428     {
00429         numInEps++;
00430         ep->txFifoNum = txFifoNum++;
00431         ep->fifoSize = (ep->packetSize/4) * p->bufferingMultiplier
00432         [ numEps ];
00433         dev->inEpAddr2EpIndex[ ep->num ] = numEps;
00434         totalTx_fifoSize += ep->fifoSize;
00435
00436         // USBD_Init(), Illegal IN EP address
00437         assert (ep->num < MAX_NUM_IN_EPS);
00438     }
00439     else
00440     {
00441         numOutEps++;
00442         ep->fifoSize = (ep->packetSize/4 + 1) * p->bufferingMultiplier
00443         [ numEps ];
00444         dev->outEpAddr2EpIndex[ ep->num ] = numEps;
00445         totalRx_fifoSize += ep->fifoSize;
00446
00447         // USBD_Init(), Illegal OUT EP address
00448         assert (ep->num < MAX_NUM_OUT_EPS);
00449     }
00450     conf += *conf;
00451
00452 }
00453
00454 /* Rx-FIFO size: SETUP packets : 4*n + 6      n=#CTRL EP's
00455 *          GOTNAK        : 1
00456 *          Status info   : 2*n      n=#OUT EP's (EPO included) in HW
00457 */
00458 totalRx_fifoSize += 10 + 1 + ( 2 * (MAX_NUM_OUT_EPS + 1) );
00459
00460 // USBD_Init(), Illegal EP count
00461 assert (numEps == NUM_EP_USED);
00462
00463 // USBD_Init(), Illegal IN EP count
00464 assert (numInEps < MAX_NUM_IN_EPS);
00465
00466 // USBD_Init(), Illegal OUT EP count
00467 assert (numOutEps < MAX_NUM_OUT_EPS);
00468
00469 DECLARE_INTERRUPT_STATE;
00470 DISABLE_INTERRUPTS();
00471
00472 USBHAL_DisableGlobalInt();
00473
00474 if ( USBDHAL_CoreInit( 1, 1 ) == USB_STATUS_OK )
00475 {
00476     USBDHAL_EnableUsbResetInt();
00477     USBHAL_EnableGlobalInt();
00478
00479     // NVIC_ClearPendingIRQ( USB IRQn );
00480     // NVIC_EnableIRQ( USB IRQn );
00481 }
00482 else
00483 {
00484     RESTORE_INTERRUPTS();
00485 //    USBD_Init(), FIFO setup error
00486     assert(0);
00487     return USB_STATUS_ILLEGAL;
00488 }
00489
00490 /* Enable EPs */
00491 uint8_t i;
00492 for (i=1;i<=numEps;i++)
00493 {
00495 #ifdef USB_DEBUG
00496     // DEBUG_BUFFER +=

```

```

00498     sprintf(DEBUG_BUFFER, "EP%d\t%d\t%d\t%d\r\n", i, ep->in, ep->type, ep->packetSize);
00499     #endif
00500     ep = &dev->ep[ i ];
00501     if (ep->in)
00502     {
00503         USB_ENABLEIN |= USB_ENABLEINEP0 << ep->num;
00504         INT_USBCFG |= INT_USBTXACTIVEEP0 << ep->num;
00505     }
00506     else
00507     {
00508         USB_ENABLEOUT |= USB_ENABLEOUTEP0 << ep->num;
00509         INT_USBCFG |= INT_USBRXVALIDEP0 << ep->num;
00510     }
00511
00512 /* Connect USB */
00513 USBDHAL_Connect();
00514
00515 RESTORE_INTERRUPTS();
00516 return USB_STATUS_OK;
00517 }
00518
00519 /*****
00539 int USBD_Write( int epAddr, void *data, int byteCount,
00540                   USB_XferCompleteCb_TypeDef callback )
00541 {
00542     #ifdef USB_DEBUG_WRITE
00543     DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "USBD_Write:%d\r\n", byteCount);
00544     #endif
00545
00546     USBD_Ep_TypeDef *ep = USBD_GetEpFromAddr( epAddr );
00547
00548     if ( ep == NULL )
00549     {
00550         #ifdef USB_DEBUG_WRITE
00551         DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "USBD_Write(), Illegal endpoint\r\n");
00552         #endif
00553         // USBD_Write(), Illegal endpoint
00554         assert( 0 );
00555         return USB_STATUS_ILLEGAL;
00556     }
00557
00558     // USBD_Write(), Illegal transfer size
00559     // assert ((byteCount < MAX_XFER_LEN) && ((byteCount / ep->packetSize) <
00560     // MAX_PACKETS_PR_XFER));
00561
00562     // USBD_Write(), Misaligned data buffer
00563     if (data!=NULL)
00564     {
00565         assert(!((uint32_t)data & 3));
00566
00567         DECLARE_INTERRUPT_STATE;
00568         DISABLE_INTERRUPTS();
00569
00570         if ( USBDHAL_EpIsStalled( ep ) )
00571         {
00572             // INT_Enable();
00573             RESTORE_INTERRUPTS();
00574             // USBD_Write(), Endpoint is halted
00575             #ifdef USB_DEBUG_WRITE
00576             DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "USBD_Write(), Endpoint is halted\r\n");
00577             #endif
00578         }
00579
00580         if ( ep->state != D_EP_IDLE )
00581         {
00582             RESTORE_INTERRUPTS();
00583             #ifdef USB_DEBUG_WRITE
00584             DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "USBD_Write(), Endpoint is busy\r\n");
00585             DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "ep->state = %d\r\n", ep->state);
00586             #endif
00587             return USB_STATUS_EP_BUSY;
00588         }
00589
00590         if ( ( ep->num > 0 ) && ( USBD_GetUsbState() !=
00591             USBD_STATE_CONFIGURED ) )

```

```

00591    {
00592        #ifdef USB_DEBUG_WRITE
00593            DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "USBD_Write(), Device not configured
00594 \r\n");
00594        #endif
00595        RESTORE_INTERRUPTS();
00596        return USB_STATUS_DEVICE_UNCONFIGURED;
00597    }
00598
00599
00600 //if data is null, this is a zero length packet
00601 if (data == NULL)
00602 {
00603     ep->buf = NULL;
00604 }
00605 else
00606 {
00607     ep->buf = (uint8_t*)data;
00608 }
00609
00610 ep->remaining = byteCount;
00611 ep->xferred = 0;
00612
00613
00614 if (ep->num == 0 )
00615 {
00616     ep->in = true;
00617 }
00618 // USBD_Write(), Illegal EP direction
00619 assert (ep->in == true);
00620
00621 ep->state = D_EP_TRANSMITTING;
00622 ep->xferCompleteCb = callback;
00623
00624 // kickoff USB transfer
00625 USBD_ArmEp( ep );
00626 RESTORE_INTERRUPTS();
00627
00628 return USB_STATUS_OK;
00629
00630 }
00631
00632 /*****
00658 int USBD_Read( int epAddr, void *data, int byteCount,
00659                 USB_XferCompleteCb_TypeDef callback )
00660 {
00661
00662     USBD_Ep_TypeDef *ep = USBD_GetEpFromAddr( epAddr );
00663
00664     #ifdef USB_DEBUG_READ
00665         DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "USBD_Read:%X - %d\r\nep->in=%d\r\n",
00666         epAddr, ep->num, ep->in);
00667     #endif
00668
00669     assert (ep!=NULL);
00670
00671     // assert ((byteCount < MAX_XFER_LEN) && ((byteCount/ep->packetSize) <
00672     // MAX_PACKETS_PR_XFER));
00673
00674     if (data!=NULL)
00675         assert (!((uint32_t)data & 3));
00676
00677
00678     DECLARE_INTERRUPT_STATE;
00679     DISABLE_INTERRUPTS();
00680
00681     if ( USBDHAL_EpIsStalled( ep ) )
00682     {
00683         RESTORE_INTERRUPTS();
00684         #ifdef USB_DEBUG_READ
00685             DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "USBD_Read(), Endpoint is halted\r\n"
00686         );
00687         #endif
00688         return USB_STATUS_EP_STALLED;
00689     }
00690     if ( ep->state == D_EP_TRANSMITTING )
00691     {

```

```

00692     RESTORE_INTERRUPTS();
00693     #ifdef USB_DEBUG_READ
00694     DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "USBD_Read(), Endpoint is busy\r\n" )
00695     ;
00696     #endif
00697     }
00698
00699     if ( ( ep->num > 0 ) && ( USBD_GetUsbState() !=
00700     USBD_STATE_CONFIGURED ) )
00701     {
00702         RESTORE_INTERRUPTS();
00703         #ifdef USB_DEBUG_READ
00704         DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "\nUSBD_Read(), Device not configured
00705     " );
00706         #endif
00707         return USB_STATUS_DEVICE_UNCONFIGURED;
00708     }
00709     ep->buf      = (uint8_t*)data;
00710     ep->remaining = byteCount;
00711     ep->xferred   = 0;
00712
00713     if ( ep->num == 0 )
00714     {
00715         ep->in = false;
00716     }
00717
00718 // USBD_Read(), Illegal EP direction
00719 assert(ep->in == false);
00720
00721     ep->state      = D_EP RECEIVING;
00722     ep->xferCompleteCb = callback;
00723
00724 // kickoff USB transfer
00725     USBD_ArmEp( ep );
00726     RESTORE_INTERRUPTS();
00727     return USB_STATUS_OK;
00728 }
00729
00730
00731
00732 /*****
00733 void usbSuspendDsr(void)
00734 {
00735     if(dev->state == USBD_STATE_SUSPENDED) {
00736
00737         #ifndef EMBER_NO_STACK
00738         emberStackPowerDown();
00739         #endif // EMBER_NO_STACK
00740         // halPowerDown();
00741         halSuspendCallback();
00742         //Turn idle sleep into USB sleep which divides down all the chip clocks,
00743         //by 4, except system timer.
00744         CPU_CLKSEL |= USBSUSP_CLKSEL_FIELD;
00745         halSleep(SLEEPMODE_IDLE);
00746     }
00747
00748
00749
00750
00751
00752
00753
00754 }
00755
00756
00757
00758 ****/
00759 int USBD_RemoteWakeup( void )
00760 {
00761     #ifdef USB_DEBUG_SUSPEND
00762     DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "RemoteWakeup..." );
00763     #endif
00764
00765     if ( ( dev->state != USBD_STATE_SUSPENDED ) ||
00766         ( dev->remoteWakeupEnabled == false ) )
00767     {
00768         // Not suspend or remote wakeup not enabled
00769         #ifdef USB_DEBUG_SUSPEND
00770             DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "illegal\r\n");
00771             #endif
00772             return USB_STATUS_ILLEGAL;
00773     }
00774     USBDHAL_SetRemoteWakeup();

```

```

00788     uint16_t timeout = 4; // Set for 4 ms. Host should take over within 1 ms
00789     //store start time, compare difference with current time with timeout period
00790     uint16_t startTime = halCommonGetInt16uMillisecondTick
00791     ();
00792     uint16_t curTime = halCommonGetInt16uMillisecondTick
00793     ();
00794     while ((elapsedTimeInt16u(startTime, curTime) <= timeout)
00795         && (dev->state == USBD_STATE_SUSPENDED)) { //exit
00796     if USB resumes
00797         curTime = halCommonGetInt16uMillisecondTick
00798     ();
00799     }
00800     if (dev->state == USBD_STATE_SUSPENDED) { //record
00801         failure if USB fails to resume
00802         #ifdef USB_DEBUG_SUSPEND
00803             DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "timed out\r\n");
00804         #endif
00805         return USB_STATUS_TIMEOUT;
00806     }
00807     #ifdef USB_DEBUG_SUSPEND
00808         DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "succeeded\r\n");
00809     #endif
00810     return USB_STATUS_OK;
00811 }
00812
00813 // DEBUG buffer with printout function, note buffer has no overflow protection
00814 #ifndef USB_DEBUG
00815 static char debugbuffer[4000];
00816 char *DEBUG_BUFFER = debugbuffer;
00817
00818 void USBD_PrintDebug(void)
00819 {
00820     debugbuffer[3999] = '\0';
00821     emberSerialGuaranteedPrintf(SER232, debugbuffer);
00822     MEMSET(debugbuffer, 0, 4000);
00823     DEBUG_BUFFER = debugbuffer;
00824 }
00825 #endif
00826 // interface with serial/uart queues. Enables emberserialprintf
00827 uint8_t dequeueTxIntoBuffer(uint8_t *data)
00828 {
00829     EmSerialFifoQueue *q = (EmSerialFifoQueue *)emSerialTxQueues[3];
00830     uint8_t txSize=0;
00831
00832     //If there are bytes in the Q and txSize hasn't maxed out, pull more
00833     //bytes off the Q into the DMA buffer
00834     while((q->used > 0) && (txSize < EP5_SIZE)) {
00835         *data = FIFO_DEQUEUE(q, emSerialTxQueueWraps[3]);
00836         data++;
00837         txSize++;
00838     }
00839
00840     return txSize;
00841 }
00842
00843 // interface with serial/uart queues. Enables emberserialprintf.
00844 // EP_IN must be defined. Assumes EP1 size, 8 bytes, which may be larger
00845 // depending on endpoint.
00846 void usbTxData ()
00847 {
00848     if (!USBD_EpIsBusy(CDC_EP_IN))
00849     {
00850         uint8_t data[EP5_SIZE];
00851         uint8_t txSize = dequeueTxIntoBuffer(data);
00852         if (txSize>0)
00853         {
00854             USBD_Write(CDC_EP_IN, data, txSize, NULL );
00855         }
00856     }
00857 }
00858
00859 // interface with serial/uart library. Enables emberSerialGuaranteedPrintf
00860 void usbForceTxData (uint8_t *data, uint8_t length)
00861 {
00862     while (USBD_EpIsBusy(CDC_EP_IN)) {}

```

```

00863     USBD_Write(CDC_EP_IN, data, length, NULL );
00864     while (USBD_EpIsBusy(CDC_EP_IN)) {}
00865 }
00866
00867 #endif //EM_SERIAL3_ENABLED
00868
00871 /***** THE REST OF THE FILE IS DOCUMENTATION ONLY !*****
01032 #endif //CORTEXM3_EM35X_USB
01033

```

8.49 em_usbd.h File Reference

```

#include "em_usb.h"
#include <PLATFORM_HEADER>
#include "stack/include/ember.h"
#include "hal/hal.h"
#include "em_usbhal.h"

```

8.49.1 Detailed Description

USB protocol stack library API for EFM32.

Author

Nathaniel Ting

Version

3.20.3

Definition in file [em_usbd.h](#).

8.50 em_usbd.h

```

00001 ****
00008 #ifndef __EM_USBD_H
00009 #define __EM_USBD_H
00010
00011 #include "em_usb.h"
00012
00013 #include PLATFORM_HEADER
00014 #include "stack/include/ember.h"
00015 #include "hal/hal.h"
00016 #include "em_usbhal.h"
00017
00018 #ifdef __cplusplus
00019 extern "C" {
00020 #endif
00021
00025 extern USBD_Device_TypeDef *dev;
00026 extern volatile bool USBDPoweredDown;
00027
00028
00029 static inline void USBD_ArmEp0( USBD_Ep_TypeDef *ep );
00030 static inline void USBD_ArmEpN( USBD_Ep_TypeDef *ep );
00031 static inline void USBD_AbortEp( USBD_Ep_TypeDef *ep );
00032
00033 void USBD_SetUsbState( USBD_State_TypeDef newState );
00034
00035 int USBDC9_SetupCmd( USBD_Device_TypeDef *device );

```

```

00036
00037 void USBDEP_Ep0Handler( USBD_Device_TypeDef *device );
00038 void USBDEP_EpHandler( uint8_t epAddr );
00039
00040 static inline void USBD_ActivateAllEps( bool forceIdle )
00041 {
00042     // int i;
00043
00044     // for ( i = 1; i <= NUM_EP_USED; i++ )
00045     // {
00046     //     USBDHAL_ActivateEp( &dev->ep[ i ], forceIdle );
00047     // }
00048 }
00049
00050 static inline void USBD_ArmEp( USBD_Ep_TypeDef *ep )
00051 {
00052     if ( ep->num == 0 )
00053     {
00054         USBD_ArmEp0( ep );
00055     }
00056     else
00057     {
00058         USBD_ArmEpN( ep );
00059     }
00060 }
00061
00062 static inline void USBD_ArmEp0( USBD_Ep_TypeDef *ep )
00063 {
00064     if ( ep->in )
00065     {
00066         uint8_t txSize = (ep->remaining > ep->packetSize) ? ep->packetSize : ep->
remaining;
00067
00068         #ifdef USB_DEBUG_EPO
00069             #ifdef USB_DEBUG_VERBOSE
00070                 DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "in-->");
00071                 uint8_t i;
00072                 for(i=0;i<txSize;i++)
00073                 {
00074                     DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "%02X", ep->buf[i]);
00075                 }
00076                 DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "\r\n");
00077             #else
00078                 DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "in\r\n");
00079             #endif
00080         #endif
00081     }
00082
00083     if ( txSize>0 )
00084     {
00085         MEMMOVE(EPINBUF[0], ep->buf, txSize);
00086     }
00087     ep->buf += txSize;
00088
00089     USB_TXBUFSIZEPOA = txSize;
00090     USB_TXLOAD = USB_TXLOADPOA;
00091 }
00092 else
00093 {
00094     // assert (0);
00095 }
00096 }
00097
00098 static inline void USBD_ArmEpN( USBD_Ep_TypeDef *ep )
00099 {
00100     #if defined(USB_DEBUG_READ) || defined(USB_DEBUG_WRITE)
00101     DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "armEp%d",ep->num);
00102     #endif
00103
00104     if ( ep->in )
00105     {
00106         #ifdef USB_DEBUG_WRITE
00107             DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "in\r\n");
00108         #endif
00109         USBDHAL_StartEpIn( ep );
00110     }
00111 else
00112 {
00113     #ifdef USB_DEBUG_READ
00114         DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "out\r\n");

```

```

00115     #endif
00116     USBDHAL_StartEpOut( ep );
00117 }
00118
00119 #ifdef USB_DEBUG_EP
00120     #ifdef USB_DEBUG_VERBOSE
00121         DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "armEp%d-->\r\n",ep->num);
00122         DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "USB_ENABLEIN\tUSB_ENABLEOUT\t"
00123             USB_CTRL\tINT_USBCFG\r\n");
00124         DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "%X          \t%X          \t%X
00125             \t%X\r\n\r\n",USB_ENABLEIN,USB_ENABLEOUT,USB_CTRL,INT_USBCFG);
00126     #else
00127         DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "armEp%d\r\n",ep->num);
00128     #endif
00129 }
00130 static inline void USBD_DeactivateAllEps( USB_Status_TypeDef
00131 reason )
00132 {
00133     int i;
00134     USBD_Ep_TypeDef *ep;
00135     for ( i = 1; i <= NUM_EP_USED; i++ )
00136     {
00137         ep = &dev->ep[ i ];
00138
00139         if ( ep->state == D_EP_IDLE )
00140         {
00141             USBDHAL_DeactivateEp( ep );
00142         }
00143     }
00144
00145     USBDHAL_AbortAllTransfers( reason );
00146 }
00147
00148 static inline USBD_Ep_TypeDef *USBD_GetEpFromAddr( uint8_t epAddr )
00149 {
00150     int epIndex;
00151     USBD_Ep_TypeDef *ep = NULL;
00152
00153     if ( epAddr & USB_SETUP_DIR_MASK )
00154     {
00155         epIndex = dev->inEpAddr2EpIndex[ epAddr & USB_EPNUM_MASK ];
00156     }
00157     else
00158     {
00159         epIndex = dev->outEpAddr2EpIndex[ epAddr & USB_EPNUM_MASK ];
00160     }
00161
00162     if ( epIndex )
00163     {
00164         ep = &dev->ep[ epIndex ];
00165     }
00166     else if ( ( epAddr & USB_EPNUM_MASK ) == 0 )
00167     {
00168         ep = &dev->ep[ 0 ];
00169     }
00170
00171     return ep;
00172 }
00173
00174 static inline USBD_Ep_TypeDef *USBD_GetEpFromNum( uint8_t eppnum )
00175 {
00176     USBD_Ep_TypeDef *ep = NULL;
00177     ep = &dev->ep[ eppnum ];
00178     return ep;
00179 }
00180
00181 static inline void USBD_ReArmEp0( USBD_Ep_TypeDef *ep )
00182 {
00183     USBD_ArmEp0(ep);
00184 }
00185
00186 static inline void USBD_AbortEp( USBD_Ep_TypeDef *ep )
00187 {
00188     if ( ep->state == D_EP_IDLE )
00189     {
00190         return;
00191     }

```

```

00192     if ( ep->in )
00193     {
00194         // Unsupported
00195         // USBDHAL_AbortEpIn( ep );
00196     }
00197     else
00198     {
00199         // Unsupported
00200         // USBDHAL_AbortEpOut( ep );
00201     }
00202 }
00203 }
00204
00207 #ifdef __cplusplus
00208 }
00209#endif
00210#endif /* __EM_USBD_H */

```

8.51 em_usbdch9.c File Reference

```

#include <PLATFORM_HEADER>
#include "stack/include/ember.h"
#include "hal/hal.h"
#include "em_usb.h"
#include "em_usbhal.h"
#include "em_usbtypes.h"
#include "em_usbd.h"

```

8.51.1 Detailed Description

USB protocol stack library, USB chapter 9 command handler.

Author

Nathaniel Ting

Version

3.20.3

Definition in file [em_usbdch9.c](#).

8.52 em_usbdch9.c

```

00001 /*****
00010 #include PLATFORM_HEADER
00011 #include "stack/include/ember.h"
00012 #include "hal/hal.h"
00013
00014 #if defined(CORTEXM3_EM35X_USB)
00015 #include "em_usb.h"
00016 #include "em_usbhal.h"
00017 #include "em_usbtypes.h"
00018 #include "em_usbd.h"
00019
00022 static USB_Status_TypeDef ClearFeature      (
00023             USBD_Device_TypeDef *pDev );
00023 static USB_Status_TypeDef GetConfiguration (

```

```

        USBD_Device_TypeDef *pDev );
00024 static USB_Status_TypeDef GetDescriptor      (
00025     USBD_Device_TypeDef *pDev );
00026 static USB_Status_TypeDef GetInterface      (
00027     USBD_Device_TypeDef *pDev );
00028 static USB_Status_TypeDef GetStatus         (
00029     USBD_Device_TypeDef *pDev );
00030 static USB_Status_TypeDef SetAddress        (
00031     USBD_Device_TypeDef *pDev );
00032
00033 int USBDCH9_SetupCmd( USBD_Device_TypeDef *device )
00034 {
00035     int status;
00036     device->setup = (USB_Setup_TypeDef *)usbBufferA.eps.ep0OUT;
00037     USB_Setup_TypeDef *p = device->setup;
00038
00039 #ifdef USB_DEBUG_CH9
00040     uint8_t i;
00041     for(i=0;i<8;i++) {
00042         DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "%02X", usbBufferA.eps.ep0OUT[i]);
00043     }
00044     DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "\t");
00045 #endif
00046
00047 // DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "bmRequestType = %X\r\n" ,
00048 p->bmRequestType);
00049 // DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "bRequest =      %X\r\n" ,
00050 p->bRequest);
00051 // DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "wValue =      %X\r\n" ,
00052 p->wValue);
00053 // DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "wIndex =      %X\r\n" ,
00054 p->wIndex);
00055 // DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "wLength =      %X\r\n" ,
00056 p->wLength);
00057
00058 /* Vendor unique, Class or Standard setup commands override ? */
00059 if ( device->callbacks->setupCmd )
00060 {
00061     status = device->callbacks->setupCmd( p );
00062 }
00063
00064 status = USB_STATUS_REQ_ERR;
00065
00066 if ( p->Type == USB_SETUP_TYPE_STANDARD )
00067 {
00068     switch ( p->bRequest )
00069     {
00070         case GET_DESCRIPTOR:
00071             #ifdef USB_DEBUG_CH9
00072                 DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "GET_DESCRIPTOR -> \r\n");
00073             #endif
00074             status = GetDescriptor( device );
00075             break;
00076
00077         /*Hardware Handled Standard Commands. Can be disabled.*/
00078         case GET_STATUS:
00079             #ifdef USB_DEBUG_CH9
00080                 DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "GET_STATUS -> \r\n");
00081             #endif
00082             status = GetStatus( device );
00083             break;
00084
00085         case CLEAR_FEATURE:
00086             #ifdef USB_DEBUG_CH9
00087                 DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "CLEAR_FEATURE -> \r\n");
00088             #endif
00089             status = ClearFeature( device );
00090             break;

```



```

00166      #endif
00167      if ( index != 0 )
00168      {
00169          break;
00170      }
00171
00172      data    = pDev->deviceDescriptor;
00173      length  = pDev->deviceDescriptor->bLength;
00174      break;
00175
00176      case USB_CONFIG_DESCRIPTOR :
00177          #ifdef USB_DEBUG3
00178          DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "ConfigDescriptor \r\n");
00179          #endif
00180          if ( index != 0 )
00181          {
00182              break;
00183          }
00184          data    = pDev->configDescriptor;
00185          length  = pDev->configDescriptor->wTotalLength;
00186          USBD_SetUsbState(USBD_STATE_CONFIGURED);
00187          break;
00188
00189      case USB_STRING_DESCRIPTOR :
00190          #ifdef USB_DEBUG3
00191          DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "StringDescriptor:%d \r\n",index);
00192          #endif
00193          if ( index < pDev->numberOfStrings )
00194          {
00195              USB_StringDescriptor_TypeDef *s;
00196              s = ((USB_StringDescriptor_TypeDef**)pDev->
00197                  stringDescriptors)[index];
00198
00199              data    = s;
00200              length  = s->len;
00201          }
00202          break;
00203
00204      // call USBD_Write to send data back over EPO, which will also update EP
00205      state
00206      {
00207          USBD_Write( 0, (void*)data, EFM32_MIN( length, p->wLength
00208 ), NULL );
00209          retVal = USB_STATUS_OK;
00210
00211      return retVal;
00212  }
00213
00214
00215 /* Hardware Handled Standard Commands */
00216 static USB_Status_TypeDef ClearFeature( USBD_Device_TypeDef *
00217 pDev )
00218 {
00219     USB_Status_TypeDef retVal = USB_STATUS_REQ_ERR
00220 ;
00221     return retVal;
00222 }
00223
00224 static USB_Status_TypeDef GetConfiguration(
00225 USBD_Device_TypeDef *pDev )
00226 {
00227     USB_Status_TypeDef retVal = USB_STATUS_REQ_ERR
00228 ;
00229     return retVal;
00230 }
00231
00232 static USB_Status_TypeDef GetInterface( USBD_Device_TypeDef *
00233 pDev )
00234 {
00235     USB_Status_TypeDef retVal = USB_STATUS_REQ_ERR
00236 ;
00237     return retVal;
00238 }
00239
00240 static USB_Status_TypeDef GetStatus( USBD_Device_TypeDef *
00241 pDev )
00242 {

```

```

00236     USB_Status_TypeDef retVal = USB_STATUS_REQ_ERR
00237     ;
00238     return retVal;
00239 }
00240 static USB_Status_TypeDef SetAddress( USBD_Device_TypeDef *
00241     pDev )
00242 {
00243     USB_Status_TypeDef retVal = USB_STATUS_REQ_ERR
00244     ;
00245     return retVal;
00246 }
00247 static USB_Status_TypeDef SetConfiguration(
00248     USBD_Device_TypeDef *pDev )
00249 {
00250     USB_Status_TypeDef retVal = USB_STATUS_REQ_ERR
00251     ;
00252     return retVal;
00253 }
00254 static USB_Status_TypeDef SetFeature( USBD_Device_TypeDef *
00255     pDev )
00256 {
00257     USB_Status_TypeDef retVal = USB_STATUS_REQ_ERR
00258     ;
00259     return retVal;
00260 }
00261 static USB_Status_TypeDef SetInterface( USBD_Device_TypeDef *
00262     pDev )
00263 {
00264     USB_Status_TypeDef retVal = USB_STATUS_REQ_ERR
00265     ;
00266 #endif //CORTEXM3_EM35X_USB
00267

```

8.53 em_usbdep.c File Reference

```

#include <PLATFORM_HEADER>
#include "stack/include/ember.h"
#include "hal/hal.h"
#include "em_usb.h"
#include "em_usbhal.h"
#include "em_usbtypes.h"
#include "em_usbd.h"

```

8.53.1 Detailed Description

USB protocol stack library, USB device endpoint handlers.

Author

Nathaniel Ting

Version

3.20.3

Definition in file [em_usbdep.c](#).

8.54 em_usbdep.c

```

00001 /*****
00002  *include PLATFORM_HEADER
00010 #include "stack/include/ember.h"
00011 #include "hal/hal.h"
00012
00013 #if defined(CORTEXM3_EM35X_USB)
00014 #include "em_usb.h"
00015 #include "em_usbhal.h"
00016 #include "em_usbtypes.h"
00017 #include "em_usbd.h"
00018
00019 */
00020 * USBDEP_Ep0Handler() is called each time a packet has been transmitted
00021 * or received on the default endpoint.
00022 * A state machine navigate us through the phases of a control transfer
00023 * according to "chapter 9" in the USB spec.
00024 */
00025 void USBDEP_Ep0Handler( USBD_Device_TypeDef *device )
00026 {
00027     int status;
00028     USBD_Ep_TypeDef *ep;
00029     static bool statusIn;
00030     static uint32_t xferred;
00031     static USB_XferCompleteCb_TypeDef callback;
00032     ep = &device->ep[ 0 ];
00033
00034     #ifdef USB_DEBUG_EPO
00035     DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "ep->state = ");
00036     #endif
00037
00038
00039
00040
00041     switch ( ep->state )
00042     {
00043         case D_EP_IDLE:
00044             ep->remaining = 0;
00045             ep->zlp = 0;
00046             callback = NULL;
00047             statusIn = false;
00048             #ifdef USB_DEBUG_EPO
00049             DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "IDLE\r\n");
00050             #endif
00051
00052             if (*EPOUTBUFSIZE[0] == 0)
00053             {
00054
00055                 #ifdef USB_DEBUG_EPO
00056                 DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "0 byte ack\r\n");
00057                 #endif
00058                 break;
00059             }
00060             status = USBDCH9_SetupCmd( device );
00061
00062             if ( status == USB_STATUS_REQ_ERR )
00063             {
00064                 // ignore error for now since this can be triggered by requests we
00065                 // don't
00066                 // need to handle
00067                 ep->in = true;
00068                 USBDHAL_StallEp( ep ); /* Stall Ep0 IN */
00069                 ep->in = false; /* OUT for next SETUP */
00070                 // USBDHAL_StallEp( ep ); /* Stall Ep0 OUT */
00071             */
00072             USBDHAL_ReenableEp0Setup( device ); /* Prepare for next SETUP packet */
00073         }
00074     else /* ( Status == USB_STATUS_OK ) */
00075     {
00076         if ( (ep->state == D_EP RECEIVING) || (ep->state == D_EP TRANSMITTING)
00077         )
00078         {
00079             callback = ep->xferCompleteCb;
00080         }
00081
00082         if ( ep->state != D_EP RECEIVING )
00083         {
00084             // enable with care. since setup involves lots of transfers, the
00085             debug
00086             // buffer will fill up very quickly.
00087         }
00088     }
00089
00090     }
00091
00092     #endif
00093
00094     if ( ep->remaining > 0 )
00095     {
00096         if ( ep->zlp == 0 )
00097         {
00098             ep->zlp = 1;
00099             ep->remaining--;
00100             ep->remaining -= ep->zlp;
00101             ep->zlp = 0;
00102
00103             if ( ep->remaining == 0 )
00104             {
00105                 if ( callback != NULL )
00106                 {
00107                     callback( ep );
00108                 }
00109             }
00110         }
00111     }
00112
00113     if ( ep->remaining == 0 )
00114     {
00115         if ( ep->zlp == 0 )
00116         {
00117             ep->zlp = 1;
00118             ep->remaining--;
00119             ep->remaining -= ep->zlp;
00120             ep->zlp = 0;
00121
00122             if ( ep->remaining == 0 )
00123             {
00124                 if ( callback != NULL )
00125                 {
00126                     callback( ep );
00127                 }
00128             }
00129         }
00130     }
00131
00132     if ( ep->remaining == 0 )
00133     {
00134         if ( ep->zlp == 0 )
00135         {
00136             ep->zlp = 1;
00137             ep->remaining--;
00138             ep->remaining -= ep->zlp;
00139             ep->zlp = 0;
00140
00141             if ( ep->remaining == 0 )
00142             {
00143                 if ( callback != NULL )
00144                 {
00145                     callback( ep );
00146                 }
00147             }
00148         }
00149     }
00150
00151     if ( ep->remaining == 0 )
00152     {
00153         if ( ep->zlp == 0 )
00154         {
00155             ep->zlp = 1;
00156             ep->remaining--;
00157             ep->remaining -= ep->zlp;
00158             ep->zlp = 0;
00159
00160             if ( ep->remaining == 0 )
00161             {
00162                 if ( callback != NULL )
00163                 {
00164                     callback( ep );
00165                 }
00166             }
00167         }
00168     }
00169
00170     if ( ep->remaining == 0 )
00171     {
00172         if ( ep->zlp == 0 )
00173         {
00174             ep->zlp = 1;
00175             ep->remaining--;
00176             ep->remaining -= ep->zlp;
00177             ep->zlp = 0;
00178
00179             if ( ep->remaining == 0 )
00180             {
00181                 if ( callback != NULL )
00182                 {
00183                     callback( ep );
00184                 }
00185             }
00186         }
00187     }
00188
00189     if ( ep->remaining == 0 )
00190     {
00191         if ( ep->zlp == 0 )
00192         {
00193             ep->zlp = 1;
00194             ep->remaining--;
00195             ep->remaining -= ep->zlp;
00196             ep->zlp = 0;
00197
00198             if ( ep->remaining == 0 )
00199             {
00200                 if ( callback != NULL )
00201                 {
00202                     callback( ep );
00203                 }
00204             }
00205         }
00206     }
00207
00208     if ( ep->remaining == 0 )
00209     {
00210         if ( ep->zlp == 0 )
00211         {
00212             ep->zlp = 1;
00213             ep->remaining--;
00214             ep->remaining -= ep->zlp;
00215             ep->zlp = 0;
00216
00217             if ( ep->remaining == 0 )
00218             {
00219                 if ( callback != NULL )
00220                 {
00221                     callback( ep );
00222                 }
00223             }
00224         }
00225     }
00226
00227     if ( ep->remaining == 0 )
00228     {
00229         if ( ep->zlp == 0 )
00230         {
00231             ep->zlp = 1;
00232             ep->remaining--;
00233             ep->remaining -= ep->zlp;
00234             ep->zlp = 0;
00235
00236             if ( ep->remaining == 0 )
00237             {
00238                 if ( callback != NULL )
00239                 {
00240                     callback( ep );
00241                 }
00242             }
00243         }
00244     }
00245
00246     if ( ep->remaining == 0 )
00247     {
00248         if ( ep->zlp == 0 )
00249         {
00250             ep->zlp = 1;
00251             ep->remaining--;
00252             ep->remaining -= ep->zlp;
00253             ep->zlp = 0;
00254
00255             if ( ep->remaining == 0 )
00256             {
00257                 if ( callback != NULL )
00258                 {
00259                     callback( ep );
00260                 }
00261             }
00262         }
00263     }
00264
00265     if ( ep->remaining == 0 )
00266     {
00267         if ( ep->zlp == 0 )
00268         {
00269             ep->zlp = 1;
00270             ep->remaining--;
00271             ep->remaining -= ep->zlp;
00272             ep->zlp = 0;
00273
00274             if ( ep->remaining == 0 )
00275             {
00276                 if ( callback != NULL )
00277                 {
00278                     callback( ep );
00279                 }
00280             }
00281         }
00282     }
00283
00284     if ( ep->remaining == 0 )
00285     {
00286         if ( ep->zlp == 0 )
00287         {
00288             ep->zlp = 1;
00289             ep->remaining--;
00290             ep->remaining -= ep->zlp;
00291             ep->zlp = 0;
00292
00293             if ( ep->remaining == 0 )
00294             {
00295                 if ( callback != NULL )
00296                 {
00297                     callback( ep );
00298                 }
00299             }
00300         }
00301     }
00302
00303     if ( ep->remaining == 0 )
00304     {
00305         if ( ep->zlp == 0 )
00306         {
00307             ep->zlp = 1;
00308             ep->remaining--;
00309             ep->remaining -= ep->zlp;
00310             ep->zlp = 0;
00311
00312             if ( ep->remaining == 0 )
00313             {
00314                 if ( callback != NULL )
00315                 {
00316                     callback( ep );
00317                 }
00318             }
00319         }
00320     }
00321
00322     if ( ep->remaining == 0 )
00323     {
00324         if ( ep->zlp == 0 )
00325         {
00326             ep->zlp = 1;
00327             ep->remaining--;
00328             ep->remaining -= ep->zlp;
00329             ep->zlp = 0;
00330
00331             if ( ep->remaining == 0 )
00332             {
00333                 if ( callback != NULL )
00334                 {
00335                     callback( ep );
00336                 }
00337             }
00338         }
00339     }
00340
00341     if ( ep->remaining == 0 )
00342     {
00343         if ( ep->zlp == 0 )
00344         {
00345             ep->zlp = 1;
00346             ep->remaining--;
00347             ep->remaining -= ep->zlp;
00348             ep->zlp = 0;
00349
00350             if ( ep->remaining == 0 )
00351             {
00352                 if ( callback != NULL )
00353                 {
00354                     callback( ep );
00355                 }
00356             }
00357         }
00358     }
00359
00360     if ( ep->remaining == 0 )
00361     {
00362         if ( ep->zlp == 0 )
00363         {
00364             ep->zlp = 1;
00365             ep->remaining--;
00366             ep->remaining -= ep->zlp;
00367             ep->zlp = 0;
00368
00369             if ( ep->remaining == 0 )
00370             {
00371                 if ( callback != NULL )
00372                 {
00373                     callback( ep );
00374                 }
00375             }
00376         }
00377     }
00378
00379     if ( ep->remaining == 0 )
00380     {
00381         if ( ep->zlp == 0 )
00382         {
00383             ep->zlp = 1;
00384             ep->remaining--;
00385             ep->remaining -= ep->zlp;
00386             ep->zlp = 0;
00387
00388             if ( ep->remaining == 0 )
00389             {
00390                 if ( callback != NULL )
00391                 {
00392                     callback( ep );
00393                 }
00394             }
00395         }
00396     }
00397
00398     if ( ep->remaining == 0 )
00399     {
00400         if ( ep->zlp == 0 )
00401         {
00402             ep->zlp = 1;
00403             ep->remaining--;
00404             ep->remaining -= ep->zlp;
00405             ep->zlp = 0;
00406
00407             if ( ep->remaining == 0 )
00408             {
00409                 if ( callback != NULL )
00410                 {
00411                     callback( ep );
00412                 }
00413             }
00414         }
00415     }
00416
00417     if ( ep->remaining == 0 )
00418     {
00419         if ( ep->zlp == 0 )
00420         {
00421             ep->zlp = 1;
00422             ep->remaining--;
00423             ep->remaining -= ep->zlp;
00424             ep->zlp = 0;
00425
00426             if ( ep->remaining == 0 )
00427             {
00428                 if ( callback != NULL )
00429                 {
00430                     callback( ep );
00431                 }
00432             }
00433         }
00434     }
00435
00436     if ( ep->remaining == 0 )
00437     {
00438         if ( ep->zlp == 0 )
00439         {
00440             ep->zlp = 1;
00441             ep->remaining--;
00442             ep->remaining -= ep->zlp;
00443             ep->zlp = 0;
00444
00445             if ( ep->remaining == 0 )
00446             {
00447                 if ( callback != NULL )
00448                 {
00449                     callback( ep );
00450                 }
00451             }
00452         }
00453     }
00454
00455     if ( ep->remaining == 0 )
00456     {
00457         if ( ep->zlp == 0 )
00458         {
00459             ep->zlp = 1;
00460             ep->remaining--;
00461             ep->remaining -= ep->zlp;
00462             ep->zlp = 0;
00463
00464             if ( ep->remaining == 0 )
00465             {
00466                 if ( callback != NULL )
00467                 {
00468                     callback( ep );
00469                 }
00470             }
00471         }
00472     }
00473
00474     if ( ep->remaining == 0 )
00475     {
00476         if ( ep->zlp == 0 )
00477         {
00478             ep->zlp = 1;
00479             ep->remaining--;
00480             ep->remaining -= ep->zlp;
00481             ep->zlp = 0;
00482
00483             if ( ep->remaining == 0 )
00484             {
00485                 if ( callback != NULL )
00486                 {
00487                     callback( ep );
00488                 }
00489             }
00490         }
00491     }
00492
00493     if ( ep->remaining == 0 )
00494     {
00495         if ( ep->zlp == 0 )
00496         {
00497             ep->zlp = 1;
00498             ep->remaining--;
00499             ep->remaining -= ep->zlp;
00500             ep->zlp = 0;
00501
00502             if ( ep->remaining == 0 )
00503             {
00504                 if ( callback != NULL )
00505                 {
00506                     callback( ep );
00507                 }
00508             }
00509         }
00510     }
00511
00512     if ( ep->remaining == 0 )
00513     {
00514         if ( ep->zlp == 0 )
00515         {
00516             ep->zlp = 1;
00517             ep->remaining--;
00518             ep->remaining -= ep->zlp;
00519             ep->zlp = 0;
00520
00521             if ( ep->remaining == 0 )
00522             {
00523                 if ( callback != NULL )
00524                 {
00525                     callback( ep );
00526                 }
00527             }
00528         }
00529     }
00530
00531     if ( ep->remaining == 0 )
00532     {
00533         if ( ep->zlp == 0 )
00534         {
00535             ep->zlp = 1;
00536             ep->remaining--;
00537             ep->remaining -= ep->zlp;
00538             ep->zlp = 0;
00539
00540             if ( ep->remaining == 0 )
00541             {
00542                 if ( callback != NULL )
00543                 {
00544                     callback( ep );
00545                 }
00546             }
00547         }
00548     }
00549
00550     if ( ep->remaining == 0 )
00551     {
00552         if ( ep->zlp == 0 )
00553         {
00554             ep->zlp = 1;
00555             ep->remaining--;
00556             ep->remaining -= ep->zlp;
00557             ep->zlp = 0;
00558
00559             if ( ep->remaining == 0 )
00560             {
00561                 if ( callback != NULL )
00562                 {
00563                     callback( ep );
00564                 }
00565             }
00566         }
00567     }
00568
00569     if ( ep->remaining == 0 )
00570     {
00571         if ( ep->zlp == 0 )
00572         {
00573             ep->zlp = 1;
00574             ep->remaining--;
00575             ep->remaining -= ep->zlp;
00576             ep->zlp = 0;
00577
00578             if ( ep->remaining == 0 )
00579             {
00580                 if ( callback != NULL )
00581                 {
00582                     callback( ep );
00583                 }
00584             }
00585         }
00586     }
00587
00588     if ( ep->remaining == 0 )
00589     {
00590         if ( ep->zlp == 0 )
00591         {
00592             ep->zlp = 1;
00593             ep->remaining--;
00594             ep->remaining -= ep->zlp;
00595             ep->zlp = 0;
00596
00597             if ( ep->remaining == 0 )
00598             {
00599                 if ( callback != NULL )
00600                 {
00601                     callback( ep );
00602                 }
00603             }
00604         }
00605     }
00606
00607     if ( ep->remaining == 0 )
00608     {
00609         if ( ep->zlp == 0 )
00610         {
00611             ep->zlp = 1;
00612             ep->remaining--;
00613             ep->remaining -= ep->zlp;
00614             ep->zlp = 0;
00615
00616             if ( ep->remaining == 0 )
00617             {
00618                 if ( callback != NULL )
00619                 {
00620                     callback( ep );
00621                 }
00622             }
00623         }
00624     }
00625
00626     if ( ep->remaining == 0 )
00627     {
00628         if ( ep->zlp == 0 )
00629         {
00630             ep->zlp = 1;
00631             ep->remaining--;
00632             ep->remaining -= ep->zlp;
00633             ep->zlp = 0;
00634
00635             if ( ep->remaining == 0 )
00636             {
00637                 if ( callback != NULL )
00638                 {
00639                     callback( ep );
00640                 }
00641             }
00642         }
00643     }
00644
00645     if ( ep->remaining == 0 )
00646     {
00647         if ( ep->zlp == 0 )
00648         {
00649             ep->zlp = 1;
00650             ep->remaining--;
00651             ep->remaining -= ep->zlp;
00652             ep->zlp = 0;
00653
00654             if ( ep->remaining == 0 )
00655             {
00656                 if ( callback != NULL )
00657                 {
00658                     callback( ep );
00659                 }
00660             }
00661         }
00662     }
00663
00664     if ( ep->remaining == 0 )
00665     {
00666         if ( ep->zlp == 0 )
00667         {
00668             ep->zlp = 1;
00669             ep->remaining--;
00670             ep->remaining -= ep->zlp;
00671             ep->zlp = 0;
00672
00673             if ( ep->remaining == 0 )
00674             {
00675                 if ( callback != NULL )
00676                 {
00677                     callback( ep );
00678                 }
00679             }
00680         }
00681     }
00682
00683     if ( ep->remaining == 0 )
00684     {
00685         if ( ep->zlp == 0 )
00686         {
00687             ep->zlp = 1;
00688             ep->remaining--;
00689             ep->remaining -= ep->zlp;
00690             ep->zlp = 0;
00691
00692             if ( ep->remaining == 0 )
00693             {
00694                 if ( callback != NULL )
00695                 {
00696                     callback( ep );
00697                 }
00698             }
00699         }
00700     }
00701
00702     if ( ep->remaining == 0 )
00703     {
00704         if ( ep->zlp == 0 )
00705         {
00706             ep->zlp = 1;
00707             ep->remaining--;
00708             ep->remaining -= ep->zlp;
00709             ep->zlp = 0;
00710
00711             if ( ep->remaining == 0 )
00712             {
00713                 if ( callback != NULL )
00714                 {
00715                     callback( ep );
00716                 }
00717             }
00718         }
00719     }
00720
00721     if ( ep->remaining == 0 )
00722     {
00723         if ( ep->zlp == 0 )
00724         {
00725             ep->zlp = 1;
00726             ep->remaining--;
00727             ep->remaining -= ep->zlp;
00728             ep->zlp = 0;
00729
00730             if ( ep->remaining == 0 )
00731             {
00732                 if ( callback != NULL )
00733                 {
00734                     callback( ep );
00735                 }
00736             }
00737         }
00738     }
00739
00740     if ( ep->remaining == 0 )
00741     {
00742         if ( ep->zlp == 0 )
00743         {
00744             ep->zlp = 1;
00745             ep->remaining--;
00746             ep->remaining -= ep->zlp;
00747             ep->zlp = 0;
00748
00749             if ( ep->remaining == 0 )
00750             {
00751                 if ( callback != NULL )
00752                 {
00753                     callback( ep );
00754                 }
00755             }
00756         }
00757     }
00758
00759     if ( ep->remaining == 0 )
00760     {
00761         if ( ep->zlp == 0 )
00762         {
00763             ep->zlp = 1;
00764             ep->remaining--;
00765             ep->remaining -= ep->zlp;
00766             ep->zlp = 0;
00767
00768             if ( ep->remaining == 0 )
00769             {
00770                 if ( callback != NULL )
00771                 {
00772                     callback( ep );
00773                 }
00774             }
00775         }
00776     }
00777
00778     if ( ep->remaining == 0 )
00779     {
00780         if ( ep->zlp == 0 )
00781         {
00782             ep->zlp = 1;
00783             ep->remaining--;
00784             ep->remaining -= ep->zlp;
00785             ep->zlp = 0;
00786
00787             if ( ep->remaining == 0 )
00788             {
00789                 if ( callback != NULL )
00790                 {
00791                     callback( ep );
00792                 }
00793             }
00794         }
00795     }
00796
00797     if ( ep->remaining == 0 )
00798     {
00799         if ( ep->zlp == 0 )
00800         {
00801             ep->zlp = 1;
00802             ep->remaining--;
00803             ep->remaining -= ep->zlp;
00804             ep->zlp = 0;
00805
00806             if ( ep->remaining == 0 )
00807             {
00808                 if ( callback != NULL )
00809                 {
00810                     callback( ep );
00811                 }
00812             }
00813         }
00814     }
00815
00816     if ( ep->remaining == 0 )
00817     {
00818         if ( ep->zlp == 0 )
00819         {
00820             ep->zlp = 1;
00821             ep->remaining--;
00822             ep->remaining -= ep->zlp;
00823             ep->zlp = 0;
00824
00825             if ( ep->remaining == 0 )
00826             {
00827                 if ( callback != NULL )
00828                 {
00829                     callback( ep );
00830                 }
00831             }
00832         }
00833     }
00834
00835     if ( ep->remaining == 0 )
00836     {
00837         if ( ep->zlp == 0 )
00838         {
00839             ep->zlp = 1;
00840             ep->remaining--;
00841             ep->remaining -= ep->zlp;
00842             ep->zlp = 0;
00843
00844             if ( ep->remaining == 0 )
00845             {
00846                 if ( callback != NULL )
00847                 {
00848                     callback( ep );
00849                 }
00850             }
00851         }
00852     }
00853
00854     if ( ep->remaining == 0 )
00855     {
00856         if ( ep->zlp == 0 )
00857         {
00858             ep->zlp = 1;
00859             ep->remaining--;
00860             ep->remaining -= ep->zlp;
00861             ep->zlp = 0;
00862
00863             if ( ep->remaining == 0 )
00864             {
00865                 if ( callback != NULL )
00866                 {
00867                     callback( ep );
00868                 }
00869             }
00870         }
00871     }
00872
00873     if ( ep->remaining == 0 )
00874     {
00875         if ( ep->zlp == 0 )
00876         {
00877             ep->zlp = 1;
00878             ep->remaining--;
00879             ep->remaining -= ep->zlp;
00880             ep->zlp = 0;
00881
00882             if ( ep->remaining == 0 )
00883             {
00884                 if ( callback != NULL )
00885                 {
00886                     callback( ep );
00887                 }
00888             }
00889         }
00890     }
00891
00892     if ( ep->remaining == 0 )
00893     {
00894         if ( ep->zlp == 0 )
00895         {
00896             ep->zlp = 1;
00897             ep->remaining--;
00898             ep->remaining -= ep->zlp;
00899             ep->zlp = 0;
00900
00901             if ( ep->remaining == 0 )
00902             {
00903                 if ( callback != NULL )
00904                 {
00905                     callback( ep );
00906                 }
00907             }
00908         }
00909     }
00910
00911     if ( ep->remaining == 0 )
00912     {
00913         if ( ep->zlp == 0 )
00914         {
00915             ep->zlp = 1;
00916             ep->remaining--;
00917             ep->remaining -= ep->zlp;
00918             ep->zlp = 0;
00919
00920             if ( ep->remaining == 0 )
00921             {
00922                 if ( callback != NULL )
00923                 {
00924                     callback( ep );
00925                 }
00926             }
00927         }
00928     }
00929
00930     if ( ep->remaining == 0 )
00931     {
00932         if ( ep->zlp == 0 )
00933         {
00934             ep->zlp = 1;
00935             ep->remaining--;
00936             ep->remaining -= ep->zlp;
00937             ep->zlp = 0;
00938
00939             if ( ep->remaining == 0 )
00940             {
00941                 if ( callback != NULL )
00942                 {
00943                     callback( ep );
00944                 }
00945             }
00946         }
00947     }
00948
00949     if ( ep->remaining == 0 )
00950     {
00951         if ( ep->zlp == 0 )
00952         {
00953             ep->zlp = 1;
00954             ep->remaining--;
00955             ep->remaining -= ep->zlp;
00956             ep->zlp = 0;
00957
00958             if ( ep->remaining == 0 )
00959             {
00960                 if ( callback != NULL )
00961                 {
00962                     callback( ep );
00963                 }
00964             }
00965         }
00966     }
00967
00968     if ( ep->remaining == 0 )
00969     {
00970         if ( ep->zlp == 0 )
00971         {
00972             ep->zlp = 1;
00973             ep->remaining--;
00974             ep->remaining -= ep->zlp;
00975             ep->zlp = 0;
00976
00977             if ( ep->remaining == 0 )
00978             {
00979                 if ( callback != NULL )
00980                 {
00981                     callback( ep );
00982                 }
00983             }
00984         }
00985     }
00986
00987     if ( ep->remaining == 0 )
00988     {
00989         if ( ep->zlp == 0 )
00990         {
00991             ep->zlp = 1;
00992             ep->remaining--;
00993             ep->remaining -= ep->zlp;
00994             ep->zlp = 0;
00995
00996             if ( ep->remaining == 0 )
00997             {
00998                 if ( callback != NULL )
00999                 {
01000                     callback( ep );
01001                 }
01002             }
01003         }
01004     }
01005
01006     if ( ep->remaining == 0 )
01007     {
01008         if ( ep->zlp == 0 )
01009         {
01010             ep->zlp = 1;
01011             ep->remaining--;
01012             ep->remaining -= ep->zlp;
01013             ep->zlp = 0;
01014
01015             if ( ep->remaining == 0 )
01016             {
01017                 if ( callback != NULL )
01018                 {
01019                     callback( ep );
01020                 }
01021             }
01022         }
01023     }
01024
01025     if ( ep->remaining == 0 )
01026     {
01027         if ( ep->zlp == 0 )
01028         {
01029             ep->zlp = 1;
01030             ep->remaining--;
01031             ep->remaining -= ep->zlp;
01032             ep->zlp = 0;
01033
01034             if ( ep->remaining == 0 )
01035             {
01036                 if ( callback != NULL )
01037                 {
01038                     callback( ep );
01039                 }
01040             }
01041         }
01042     }
01043
01044     if ( ep->remaining == 0 )
01045     {
01046         if ( ep->zlp == 0 )
01047         {
01048             ep->zlp = 1;
01049             ep->remaining--;
01050             ep->remaining -= ep->zlp;
01051             ep->zlp = 0;
01052
01053             if ( ep->remaining == 0 )
01054             {
01055                 if ( callback != NULL )
01056                 {
01057                     callback( ep );
01058                 }
01059             }
01060         }
01061     }
01062
01063     if ( ep->remaining == 0 )
01064     {
01065         if ( ep->zlp == 0 )
01066         {
01067             ep->zlp = 1;
01068             ep->remaining--;
01069             ep->remaining -= ep->zlp;
01070             ep->zlp = 0;
01071
01072             if ( ep->remaining == 0 )
01073             {
01074                 if ( callback != NULL )
01075                 {
01076                     callback( ep );
01077                 }
01078             }
01079         }
01080     }
01081
01082     if ( ep->remaining == 0 )
01083     {
01084         if ( ep->zlp == 0 )
01085         {
01086             ep->zlp = 1;
01087             ep->remaining--;
01088             ep->remaining -= ep->zlp;
01089             ep->zlp = 0;
01090
01091             if ( ep->remaining == 0 )
01092             {
01093                 if ( callback != NULL )
01094                 {
01095                     callback( ep );
01096                 }
01097             }
01098         }
01099     }
01100
01101     if ( ep->remaining == 0 )
01102     {
01103         if ( ep->zlp == 0 )
01104         {
01105             ep->zlp = 1;
01106             ep->remaining--;
01107             ep->remaining -= ep->zlp;
01108             ep->zlp = 0;
01109
01110             if ( ep->remaining == 0 )
01111             {
01112                 if ( callback != NULL )
01113                 {
01114                     callback( ep );
01115                 }
01116             }
01117         }
01118     }
01119
01120     if ( ep->remaining == 0 )
01121     {
01122         if ( ep->zlp == 0 )
01123         {
01124             ep->zlp = 1;
01125             ep->remaining--;
01126             ep->remaining -= ep->zlp;
01127             ep->zlp = 0;
01128
01129             if ( ep->remaining == 0 )
01130             {
01131                 if ( callback != NULL )
01132                 {
01133                     callback( ep );
01134                 }
01135             }
01136         }
01137     }
01138
01139     if ( ep->remaining == 0 )
01140     {
01141         if ( ep->zlp == 0 )
01142         {
01143             ep->zlp = 1;
01144             ep->remaining--;
01145             ep->remaining -= ep->zlp;
01146             ep->zlp = 0;
01147
01148             if ( ep->remaining == 0 )
01149             {
01150                 if ( callback != NULL )
01151                 {
01152                     callback( ep );
01153                 }
01154             }
01155         }
01156     }
01157
01158     if ( ep->remaining == 0 )
01159     {
01160         if ( ep->zlp == 0 )
01161         {
01162             ep->zlp = 
```

```

00083         // #ifdef USB_DEBUG_EPO
00084         // DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "ep->remaining =
00085 %d\r\n", ep->remaining);
00086         // #endif
00087
00088         if ( ep->remaining )
00089         {
00090             /* Data will be sent to host, check if a ZLP must be appended */
00091             if ( ( ep->remaining < device->setup->wLength ) &&
00092                 ( ep->remaining % ep->packetSize == 0      ) )
00093             {
00094                 ep->zlp = 1;
00095             }
00096         else
00097         {
00098             /* Prepare for next SETUP packet*/
00099             USBDHAL_ReenableEp0Setup( device );
00100
00101             /* No data stage, a ZLP may have been sent. If not, send one */
00102             if ( ep->zlp == 0 )
00103             {
00104                 #ifdef USB_DEBUG_EPO
00105                 DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "no ack\r\n");
00106                 #endif
00107                 ep->state = D_EP_IDLE;
00108                 USBD_Write( 0, NULL, 0, NULL );           /* ACK to host */
00109                 ep->state = D_EP_STATUS;
00110             }
00111         else
00112         {
00113             ep->state = D_EP_IDLE;
00114             ep->in = false;                         /* OUT for next SETUP */
00115         }
00116     }
00117 }
00118 }
00119 break; //D_EP_IDLE
00120 case D_EP RECEIVING:
00121     if ( ep->remaining )
00122     {
00123         ep->in = false;
00124         USBD_ReArmEp0( ep );
00125     }
00126 else
00127 {
00128     status = USB_STATUS_OK;
00129     if ( callback != NULL )
00130     {
00131         status = callback( USB_STATUS_OK, ep->xferred, 0 );
00132         callback = NULL;
00133     }
00134
00135     if ( status != USB_STATUS_OK )
00136     {
00137         ep->in = true;
00138         USBDHAL_StallEp( ep );                  /* Stall Ep0 IN          */
00139         ep->in = false;                        /* OUT for next SETUP   */
00140         USBDHAL_StallEp( ep );                  /* Stall Ep0 OUT         */
00141         USBDHAL_StartEp0Setup( dev );          /* Prepare for next SETUP pkt. */
00142         ep->state = D_EP_IDLE;
00143     }
00144 else
00145 {
00146     USBDHAL_StartEp0Setup( dev );          /* Prepare for next SETUP packet*/
00147     ep->state = D_EP_IDLE;                /* USBD_Write() sets state back */
00148                                         /* to EP_TRANSMITTING      */
00149     USBD_Write( 0, NULL, 0, NULL );
00150     ep->state = D_EP_STATUS;
00151 }
00152 }
00153 }
00154 break;
00155 case D_EP TRANSMITTING:
00156     #ifdef USB_DEBUG_EPO
00157     DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "TX\r\n");
00158     #endif
00159     if ( ep->remaining )
00160     {

```

```

00161     /* There is more data to transmit */
00162     USBD_ReArmEp0( ep );
00163
00164
00165     // enable with care. since setup involves lots of transfers, the debug
00166     // buffer will fill up very quickly.
00167     // #ifdef USB_DEBUG_EP0
00168     // DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "ep->remaining =
00169     %d\r\n",ep->remaining);
00170     // #endif
00171 }
00172 else
00173 {
00174     #ifdef USB_DEBUG_EP0
00175     DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "TX done\r\n");
00176     #endif
00177     /* All data transferred, is a ZLP packet needed ? */
00178     if ( ep->zlp == 1 )
00179     {
00180         #ifdef USB_DEBUG_EP0
00181         DEBUG_BUFFER += sprintf(DEBUG_BUFFER,"ack\r\n");
00182         #endif
00183         xferred = ep->xferred;
00184         ep->state = D_EP_IDLE;           /* USBD_Write() sets state back */
00185                                         /* to EP_TRANSMITTING          */
00186         USBD_Write( 0, NULL, 0, NULL ); /* Send ZLP
00187                                         */
00188         ep->zlp = 2;
00189     }
00190     else
00191     {
00192         if ( ep->zlp == 0 )
00193         {
00194             xferred = ep->xferred;
00195             ep->zlp = 1;
00196         }
00197
00198         // host will send ack, which will be handled by hardware
00199         // ep->state = D_EP_IDLE;
00200         // USBD_Read( 0, NULL, 0, NULL ); /* Get ZLP packet (ACK) from host
00201
00202         statusIn = true;
00203         ep->state = D_EP_STATUS;
00204     }
00205     break;
00206 case D_EP_STATUS:
00207     #ifdef USB_DEBUG_EP0
00208     DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "STATUS\r\n");
00209     #endif
00210     if ( statusIn )
00211     {
00212         USBDHAL_ReenableEp0Setup( device );
00213     }
00214     if ( callback != NULL )
00215     {
00216         callback( USB_STATUS_OK, xferred, 0 );
00217     }
00218
00219     ep->state = D_EP_IDLE;
00220     ep->in = false;                  /* OUT for next SETUP */
00221     break;
00222 }
00223 }
00224
00225 void USBDEP_EpHandler( uint8_t epAddr )
00226 {
00227     USB_XferCompleteCb_TypeDef callback;
00228
00229     USBD_Ep_TypeDef *ep = USBD_GetEpFromAddr( epAddr );
00230
00231     #if defined(USB_DEBUG_READ) || defined(USB_DEBUG_WRITE) ||
00232     defined(USB_DEBUG_EP)
00233     DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "EP%dHandler\tep->state = %d\r\n",
00234     num,ep->state);
00235     #endif
00236 }
00237
00238
00239
00240
00241
00242
00243
00244
00245
00246
00247
00248
00249
00250
00251
00252
00253
00254
00255
00256
00257
00258
00259
00260
00261
00262
00263
00264
00265
00266
00267
00268
00269
00270
00271
00272
00273
00274
00275
00276
00277
00278
00279
00280
00281
00282
00283
00284
00285
00286
00287
00288
00289
00290
00291
00292
00293
00294
00295
00296
00297
00298
00299
00300
00301
00302
00303
00304
00305
00306
00307
00308
00309
00310
00311
00312
00313
00314
00315
00316
00317
00318
00319
00320
00321
00322
00323
00324
00325
00326
00327
00328
00329
00330
00331
00332
00333
00334
00335
00336
00337
00338
00339
00340
00341
00342
00343
00344
00345
00346
00347
00348
00349
00350
00351
00352
00353
00354
00355
00356
00357
00358
00359
00360
00361
00362
00363
00364
00365
00366
00367
00368
00369
00370
00371
00372
00373
00374
00375
00376
00377
00378
00379
00380
00381
00382
00383
00384
00385
00386
00387
00388
00389
00390
00391
00392
00393
00394
00395
00396
00397
00398
00399
00400
00401
00402
00403
00404
00405
00406
00407
00408
00409
00410
00411
00412
00413
00414
00415
00416
00417
00418
00419
00420
00421
00422
00423
00424
00425
00426
00427
00428
00429
00430
00431
00432
00433
00434
00435
00436
00437
00438
00439
00440
00441
00442
00443
00444
00445
00446
00447
00448
00449
00450
00451
00452
00453
00454
00455
00456
00457
00458
00459
00460
00461
00462
00463
00464
00465
00466
00467
00468
00469
00470
00471
00472
00473
00474
00475
00476
00477
00478
00479
00480
00481
00482
00483
00484
00485
00486
00487
00488
00489
00490
00491
00492
00493
00494
00495
00496
00497
00498
00499
00500
00501
00502
00503
00504
00505
00506
00507
00508
00509
00510
00511
00512
00513
00514
00515
00516
00517
00518
00519
00520
00521
00522
00523
00524
00525
00526
00527
00528
00529
00530
00531
00532
00533
00534
00535
00536
00537
00538
00539
00540
00541
00542
00543
00544
00545
00546
00547
00548
00549
00550
00551
00552
00553
00554
00555
00556
00557
00558
00559
00560
00561
00562
00563
00564
00565
00566
00567
00568
00569
00570
00571
00572
00573
00574
00575
00576
00577
00578
00579
00580
00581
00582
00583
00584
00585
00586
00587
00588
00589
00590
00591
00592
00593
00594
00595
00596
00597
00598
00599
00600
00601
00602
00603
00604
00605
00606
00607
00608
00609
00610
00611
00612
00613
00614
00615
00616
00617
00618
00619
00620
00621
00622
00623
00624
00625
00626
00627
00628
00629
00630
00631
00632
00633
00634
00635
00636
00637
00638
00639
00640
00641
00642
00643
00644
00645
00646
00647
00648
00649
00650
00651
00652
00653
00654
00655
00656
00657
00658
00659
00660
00661
00662
00663
00664
00665
00666
00667
00668
00669
00670
00671
00672
00673
00674
00675
00676
00677
00678
00679
00680
00681
00682
00683
00684
00685
00686
00687
00688
00689
00690
00691
00692
00693
00694
00695
00696
00697
00698
00699
00700
00701
00702
00703
00704
00705
00706
00707
00708
00709
00710
00711
00712
00713
00714
00715
00716
00717
00718
00719
00720
00721
00722
00723
00724
00725
00726
00727
00728
00729
00730
00731
00732
00733
00734
00735
00736
00737
00738
00739
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749
00750
00751
00752
00753
00754
00755
00756
00757
00758
00759
00760
00761
00762
00763
00764
00765
00766
00767
00768
00769
00770
00771
00772
00773
00774
00775
00776
00777
00778
00779
00780
00781
00782
00783
00784
00785
00786
00787
00788
00789
00790
00791
00792
00793
00794
00795
00796
00797
00798
00799
00800
00801
00802
00803
00804
00805
00806
00807
00808
00809
00810
00811
00812
00813
00814
00815
00816
00817
00818
00819
00820
00821
00822
00823
00824
00825
00826
00827
00828
00829
00830
00831
00832
00833
00834
00835
00836
00837
00838
00839
00840
00841
00842
00843
00844
00845
00846
00847
00848
00849
00850
00851
00852
00853
00854
00855
00856
00857
00858
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00879
00880
00881
00882
00883
00884
00885
00886
00887
00888
00889
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999
01000
01001
01002
01003
01004
01005
01006
01007
01008
01009
01010
01011
01012
01013
01014
01015
01016
01017
01018
01019
01020
01021
01022
01023
01024
01025
01026
01027
01028
01029
01030
01031
01032
01033
01034
01035
01036
01037
01038
01039
01040
01041
01042
01043
01044
01045
01046
01047
01048
01049
01050
01051
01052
01053
01054
01055
01056
01057
01058
01059
01060
01061
01062
01063
01064
01065
01066
01067
01068
01069
01070
01071
01072
01073
01074
01075
01076
01077
01078
01079
01080
01081
01082
01083
01084
01085
01086
01087
01088
01089
01090
01091
01092
01093
01094
01095
01096
01097
01098
01099
01100
01101
01102
01103
01104
01105
01106
01107
01108
01109
01110
01111
01112
01113
01114
01115
01116
01117
01118
01119
01120
01121
01122
01123
01124
01125
01126
01127
01128
01129
01130
01131
01132
01133
01134
01135
01136
01137
01138
01139
01140
01141
01142
01143
01144
01145
01146
01147
01148
01149
01150
01151
01152
01153
01154
01155
01156
01157
01158
01159
01160
01161
01162
01163
01164
01165
01166
01167
01168
01169
01170
01171
01172
01173
01174
01175
01176
01177
01178
01179
01180
01181
01182
01183
01184
01185
01186
01187
01188
01189
01190
01191
01192
01193
01194
01195
01196
01197
01198
01199
01200
01201
01202
01203
01204
01205
01206
01207
01208
01209
01210
01211
01212
01213
01214
01215
01216
01217
01218
01219
01220
01221
01222
01223
01224
01225
01226
01227
01228
01229
01230
01231
01232
01233
01234
01235
01236
01237
01238
01239
01240
01241
01242
01243
01244
01245
01246
01247
01248
01249
01250
01251
01252
01253
01254
01255
01256
01257
01258
01259
01260
01261
01262
01263
01264
01265
01266
01267
01268
01269
01270
01271
01272
01273
01274
01275
01276
01277
01278
01279
01280
01281
01282
01283
01284
01285
01286
01287
01288
01289
01290
01291
01292
01293
01294
01295
01296
01297
01298
01299
01300
01301
01302
01303
01304
01305
01306
01307
01308
01309
01310
01311
01312
01313
01314
01315
01316
01317
01318
01319
01320
01321
01322
01323
01324
01325
01326
01327
01328
01329
01330
01331
01332
01333
01334
01335
01336
01337
01338
01339
01340
01341
01342
01343
01344
01345
01346
01347
01348
01349
01350
01351
01352
01353
01354
01355
01356
01357
01358
01359
01360
01361
01362
01363
01364
01365
01366
01367
01368
01369
01370
01371
01372
01373
01374
01375
01376
01377
01378
01379
01380
01381
01382
01383
01384
01385
01386
01387
01388
01389
01390
01391
01392
01393
01394
01395
01396
01397
01398
01399
01400
01401
01402
01403
01404
01405
01406
01407
01408
01409
01410
01411
01412
01413
01414
01415
01416
01417
01418
01419
01420
01421
01422
01423
01424
01425
01426
01427
01428
01429
01430
01431
01432
01433
01434
01435
01436
01437
01438
01439
01440
01441
01442
01443
01444
01445
01446
01447
01448
01449
01450
01451
01452
01453
01454
01455
01456
01457
01458
01459
01460
01461
01462
01463
01464
01465
01466
01467
01468
01469
01470
01471
01472
01473
01474
01475
01476
01477
01478
01479
01480
01481
01482
01483
01484
01485
01486
01487
01488
01489
01490
01491
01492
01493
01494
01495
01496
01497
01498
01499
01500
01501
01502
01503
01504
01505
01506
01507
01508
01509
01510
01511
01512
01513
01514
01515
01516
01517
01518
01519
01520
01521
01522
01523
01524
01525
01526
01527
01528
01529
01530
01531
01532
01533
01534
01535
01536
01537
01538
01539
01540
01541
01542
01543
01544
01545
01546
01547
01548
01549
01550
01551
01552
01553
01554
01555
01556
01557
01558
01559
01560
01561
01562
01563
01564
01565
01566
01567
01568
01569
01570
01571
01572
01573
01574
01575
01576
01577
01578
01579
01580
01581
01582
01583
01584
01585
01586
01587
01588
01589
01590
01591
01592
01593
01594
01595
01596
01597
01598
01599
01600
01601
01602
01603
01604
01605
01606
01607
01608
01609
01610
01611
01612
01613
01614
01615
01616
01617
01618
01619
01620
01621
01622
01623
01624
01625
01626
01627
01628
01629
01630
01631
01632
01633
01634
01635
01636
01637
01638
01639
01640
01641
01642
01643
01644
01645
01646
01647
01648
01649
01650
01651
01652
01653
01654
01655
01656
01657
01658
01659
01660
01661
01662
01663
01664
01665
01666
01667
01668
01669
01670
01671
01672
01673
01674
01675
01676
01677
01678
01679
01680
01681
01682
01683
01684
01685
01686
01687
01688
01689
01690
01691
01692
01693
01694
01695
01696
01697
01698
01699
01700
01701
01702
01703
01704
01705
01706
01707
01708
01709
01710
01711
01712
01713
01714
01715
01716
01717
01718
01719
01720
01721
01722
01723
01724
01725
01726
01727
01728
01729
01730
01731
01732
01733
01734
01735
01736
01737
01738
01739
01740
01741
01742
01743
01744
01745
01746
01747
01748
01749
01750
01751
01752
01753
01754
01755
01756
01757
01758
01759
01760
01761
01762
01763
01764
01765
01766
01767
01768
01769
01770
01771
01772
01773
01774
01775
01776
01777
01778
01779
01780
01781
01782
01783
01784
01785
01786
01787
01788
01789
01790
01791
01792
01793
01794
01795
01796
01797
01798
01799
01800
01801
01802
01803
01804
01805
01806
01807
01808
01809
01810
01811
01812
01813
01814
01815
01816
01817
01818
01819
01820
01821
01822
01823
01824
01825
01826
01827
01828
01829
01830
01831
01832
01833
01834
01835
01836
01837
01838
01839
01840
01841
01842
01843
01844
01845
01846
01847
01848
01849
01850
01851
01852
01853
01854
01855
01856
01857
01858
01859
01860
01861
01862
01863
01864
01865
01866
01867
01868
01869
01870
01871
01872
01873
01874
01875
01876
01877
01878
01879
01880
01881
01882
01883
01884
01885
01886
01887
01888
01889
01890
01891
01892
01893
01894
01895
01896
01897
01898
01899
01900
01901
01902
01903
01904
01905
01906
01907
01908
01909
01910
01911
01912
01913
01914
01915
01916
01917
01918
01919
01920
01921
01922
01923
01924
01925
01926
01927
01928
01929
01930
01931
01932
01933
01934
01935
01936
01937
01938
01939
01940
01941
01942
01943
01944
01945
01946
01947
01948
01949
01950
01951
01952
01953
01954
01955
01956
01957
01958
01959
01960
01961
01962
01963
01964
0196
```

```

00236     if ( ep->state == D_EP_TRANSMITTING )
00237     {
00238         #ifdef USB_DEBUG_EP
00239             DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "ep->xferred=%d \tep->remaining = %d
00240             \r\n",ep->xferred, ep->remaining);
00241         #endif
00242         if ( ep->remaining )
00243         {
00244             // uint8_t txSize = (ep->remaining > ep->packetSize) ? ep->packetSize :
00245             ep->remaining;
00246             /* There is more data to transmit */
00247             USBD_ArmEp( ep );
00248             // ep->buf += ep->xferred;
00249             // ep->buf += txSize;
00250         }
00251         else // Done transmitting
00252         {
00253             // Put ep back into idle state
00254             ep->state = D_EP_IDLE;
00255             // If there's a callback, call it
00256             if ( ep->xferCompleteCb )
00257             {
00258                 callback = ep->xferCompleteCb;
00259                 ep->xferCompleteCb = NULL;
00260                 callback( USB_STATUS_OK, ep->xferred, ep->remaining );
00261             }
00262             #if EM_SERIAL3_ENABLED
00263                 halInternalStartUartTx(3);
00264             #endif
00265         }
00266     else if ( ep->state == D_EP RECEIVING )
00267     {
00268         #ifdef USB_DEBUG_EP
00269             DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "ep->remaining = %d\r\n",ep->
00270             remaining);
00271         #endif
00272         if ( ep->remaining )
00273         {
00274             /* Short Packet */
00275             if (ep->packetSize>(ep->remaining + *EPOUTBUFSIZE[ep->num]))
00276             {
00277
00278                 if ( ep->xferCompleteCb )
00279                 {
00280                     callback = ep->xferCompleteCb;
00281                     ep->xferCompleteCb = NULL;
00282                     callback( USB_STATUS_TIMEOUT, ep->xferred, ep->
00283                     remaining );
00284                 }
00285             }
00286             else
00287             {
00288                 /* There is more data to transmit */
00289                 USBD_ArmEp( ep );
00290                 ep->buf += *EPOUTBUFSIZE[ep->num]; // move buffer pointer
00291             }
00292         }
00293     }
00294
00295     else
00296     {
00297         // Put ep back into idle state
00298         ep->state = D_EP_IDLE;
00299         // If there's a callback, call it
00300         if ( ep->xferCompleteCb )
00301         {
00302             callback = ep->xferCompleteCb;
00303             ep->xferCompleteCb = NULL;
00304             callback( USB_STATUS_OK, ep->xferred, ep->remaining );
00305         }
00306     }
00307 }
00308 else
00309 {
00310     // EpHandler should only be called if the endpoint is in use
00311     // ep->state set in read/write

```

```

00312     // assert( 0 );
00313 }
00314 }
00315
00318 #endif //CORTEXM3_EM35X_USB
00319

```

8.55 em_usbhal.c File Reference

```

#include <PLATFORM_HEADER>
#include "stack/include/ember.h"
#include "hal/hal.h"
#include "em_usb.h"
#include "em_usbtypes.h"
#include "em_usbhal.h"

```

8.55.1 Detailed Description

USB protocol stack library, low level USB peripheral access.

Author

Nathaniel Ting

Version

3.20.3

Definition in file [em_usbhal.c](#).

8.56 em_usbhal.c

```

00001 ****
00009 #include PLATFORM_HEADER
00010 #include "stack/include/ember.h"
00011 #include "hal/hal.h"
00012
00013 #if defined(CORTEXM3_EM35X_USB)
00014 #include "em_usb.h"
00015 #include "em_usbtypes.h"
00016 #include "em_usbhal.h"
00017
00018
00022 bool usbVbusPresent;
00023 extern USBD_Device_TypeDef *dev;
00024
00025 ALIGNMENT(8)
00026 EndPointUnion usbBufferA = {0,};
00027 uint8_t *EPOUTBUF[7] = {usbBufferA.eps.ep0OUT,
00028                           usbBufferA.eps.ep1OUT,
00029                           usbBufferA.eps.ep2OUT,
00030                           usbBufferA.eps.ep3OUT,
00031                           usbBufferA.eps.ep4OUT,
00032                           usbBufferA.eps.ep5OUT,
00033                           usbBufferA.eps.ep6OUT};
00034 uint8_t *EPINBUF[7] = {usbBufferA.eps.ep0IN,
00035                           usbBufferA.eps.ep1IN,
00036                           usbBufferA.eps.ep2IN,
00037                           usbBufferA.eps.ep3IN,

```

```

00038             usbBufferA.eps.ep4IN,
00039             usbBufferA.eps.ep5IN,
00040             usbBufferA.eps.ep6IN};
00041 volatile uint32_t *EPOUTBUFSIZE[7] = {&USB_RXBUFSIZEEP0A,
00042                                         &USB_RXBUFSIZEEP1A,
00043                                         &USB_RXBUFSIZEEP2A,
00044                                         &USB_RXBUFSIZEEP3A,
00045                                         &USB_RXBUFSIZEEP4A,
00046                                         &USB_RXBUFSIZEEP5A,
00047                                         &USB_RXBUFSIZEEP6A};
00048
00049 volatile uint32_t *EPINBUFSIZE[7] = {&USB_TXBUFSIZEEP0A,
00050                                         &USB_TXBUFSIZEEP1A,
00051                                         &USB_TXBUFSIZEEP2A,
00052                                         &USB_TXBUFSIZEEP3A,
00053                                         &USB_TXBUFSIZEEP4A,
00054                                         &USB_TXBUFSIZEEP5A,
00055                                         &USB_TXBUFSIZEEP6A};
00056 void USBDHAL_Connect( void )
00057 {
00058     #if USB_SELFPWD_STATE==1 && defined(VBUSMON)
00059
00060     //For self-powered devices, The USB specification requires that the pull-up
00061     //resistor is disconnected if VBUS is not connected.
00062     vbusMonCfg();
00063     if(usbVbusPresent) {
00064         //Set ENUMCTRL oput-high to enumerate.
00065         ENUMCTRL_SETCFG(GPIOCFG_OUT);
00066         ENUMCTRL_SET();
00067     }
00068     #else // USB_SELFPWD_STATE==1 && defined(VBUSMON)
00069         //Set ENUMCTRL oput-high to enumerate.
00070         ENUMCTRL_SETCFG(GPIOCFG_OUT);
00071         ENUMCTRL_SET();
00072     #endif // USB_SELFPWD_STATE==1 && defined(VBUSMON)
00073
00074 }
00075
00076 void USBDHAL_Disconnect( void )
00077 {
00078     ENUMCTRL_SETCFG(GPIOCFG_IN);
00079     ENUMCTRL_CLR();
00080 }
00081
00082 USB_Status_TypeDef USBDHAL_CoreInit( uint32_t totalRxFifoSize
00083                                     ,
00084                                     uint32_t totalTxFifoSize )
00085
00086     //Configure PA0 and PA1 in analog mode for USB
00087     SET_REG_FIELD(GPIO_PACFGL, PA0_CFG, GPIOCFG_ANALOG);
00088     SET_REG_FIELD(GPIO_PACFGL, PA1_CFG, GPIOCFG_ANALOG);
00089
00090     USB_BUFBASEA = (uint32_t)usbBufferA.allEps;
00091     MEMSET(usbBufferA.allEps, 0, USB_BUFFER_SIZE);
00092
00093     // Double buffering currently not supported
00094     // #ifdef USB_DOUBLE_BUFFER
00095     //     USB_BUFBASEB = (uint32_t)usbBufferB.allEps;
00096     //     MEMSET(usbBufferB.allEps, 0, USB_BUFFER_SIZE);
00097     //     //Refer to "handle buffer B" in the driveEp3Tx() function above
00098     //     //to learn why double buffering is not enabled by default.
00099     //
00100    //     //Enable double buffering bulk EP3.
00101    //     USB_CTRL |= (USB_ENBUFOUTEP3B | USB_ENBUFINEP3B);
00102    // #endif // USB_DOUBLE_BUFFER
00103
00104    //Use the SELFPWD and REMOTEWKUPEN state defines to set the appropriate
00105    //bits in USB_CTRL
00106    USB_CTRL = (USB_CTRL & (~(USB_SELFPWD_MASK | USB_REMOTEWKUPEN_MASK))) |
00107        ((USB_SELFPWD_STATE<<USB_SELFPWD_BIT) |
00108         (USB_REMOTEWKUPEN_STATE<<
00109             USB_REMOTEWKUPEN_BIT));
00110
00110 #ifdef EMBER_EMU_TEST
00111     //Select which GPIO output is used for connect/disconnect. Any
00112     //value > 47, selects disconnected state.
00113     EMU_USB_DISCONNECT = PORTA_PIN(2);
00114 #endif //EMBER_EMU_TEST
00115

```

```

00116
00117     return USB_STATUS_OK;
00118 }
00119
00120
00121 void USBDHAL_AbortAllTransfers( USB_Status_TypeDef reason )
00122 {
00123     int i;
00124     USBD_Ep_TypeDef *ep;
00125     USB_XferCompleteCb_TypeDef callback;
00126
00127     if ( reason != USB_STATUS_DEVICE_RESET )
00128     {
00129         // unsupported
00130         // USBDHAL_AbortAllEps();
00131     }
00132
00133     for ( i = 1; i <= NUM_EP_USED; i++ )
00134     {
00135         ep = &(dev->ep[i]);
00136         if ( ep->state != D_EP_IDLE )
00137         {
00138             ep->state = D_EP_IDLE;
00139             if ( ep->xferCompleteCb )
00140             {
00141                 callback = ep->xferCompleteCb;
00142                 ep->xferCompleteCb = NULL;
00143
00144                 if ( ( dev->lastState == USBD_STATE_CONFIGURED ) &
00145                     ( dev->state == USBD_STATE_ADDRESSED ) )
00146                 {
00147                     USBDHAL_DeactivateEp( ep );
00148                 }
00149                 callback( reason, ep->xferred, ep->remaining );
00150             }
00151         }
00152     }
00153 }
00154
00155 #ifdef VBUSMON
00156 //Use a selectable IRQ for monitoring VBUS on VBUSMON.
00157 void vbusMonCfg(void)
00158 {
00159     //VBUSMON just needs to be a simple input.
00160     VBUSMON_SETCFG();
00161
00162     //start from a fresh state, just in case
00163     VBUSMON_INTCFG = 0;           //disable triggering
00164     INT_CFGCLR = VBUSMON_INT_EN_BIT; //clear top level int
enable
00165     INT_GPIOFLAG = VBUSMON_FLAG_BIT; //clear stale interrupt
00166     INT_MISS = VBUSMON_MISS_BIT;   //clear stale missed
interrupt
00167     //configure
00168     VBUSMON_SEL();              //point IRQ at the desired pin
00169     VBUSMON_INTCFG = ((0 << GPIO_INTFILT_BIT) | //no filter
00170                       (3 << GPIO_INTMOD_BIT)); //3 = both edges
00171
00172     usbVbusPresent = ((VBUSMON_IN & VBUSMON) == VBUSMON);
00173
00174     INT_CFGSET = VBUSMON_INT_EN_BIT; //set top level interrupt
enable
00175 }
00176 #endif //VBUSMON
00177
00178 #ifndef EMBER_APPLICATION_HAS_CUSTOM_SUSPEND_CALLBACK
00179     uint32_t savedPeripheralReg;
00180     void halSuspendCallback(void)
00181     {
00182         halSuspend();
00183         savedPeripheralReg = PERIPHERAL_DISABLE;
00184         PERIPHERAL_DISABLE |= ((1 << PERIDIS_ADC_BIT) | \
00185                               (1 << PERIDIS_TIM2_BIT) | \
00186                               (1 << PERIDIS_TIM1_BIT) | \
00187                               (1 << PERIDIS_SC1_BIT) | \
00188                               (1 << PERIDIS_SC2_BIT));
00189     }
00190

```

```

00191 void halResumeCallback(void)
00192 {
00193     PERIPHERAL_DISABLE = savedPeripheralReg;
00194     halResume();
00195 }
00196 #endif // EMBER_APPLICATION_HAS_CUSTOM_SUSPEND_CALLBACK
00197
00198
00200 #endif //CORTEXM3_EM35X_USB
00201

```

8.57 em_usbhal.h File Reference

```
#include "em_usbtypes.h"
```

8.57.1 Detailed Description

USB protocol stack library, low level USB peripheral access.

Author

Nathaniel Ting

Version

3.20.3

Definition in file [em_usbhal.h](#).

8.58 em_usbhal.h

```

00001 /*********************************************************************
00008 #ifndef __EM_USBHAL_H
00009 #define __EM_USBHAL_H
00010
00011 #include "em_usbtypes.h"
00012
00015 #ifdef __cplusplus
00016 extern "C" {
00017 #endif
00018
00019 #ifdef VBUSMON
00020 void vbusMonCfg(void);
00021 #endif //VBUSMON
00022 void halSuspendCallback(void);
00023 void halResumeCallback(void);
00024
00025 #define INT_USBTXACTIVE ( INT_USBTXACTIVEEPO | \
00026                         INT_USBTXACTIVEEEP1 | \
00027                         INT_USBTXACTIVEEEP2 | \
00028                         INT_USBTXACTIVEEEP3 | \
00029                         INT_USBTXACTIVEEEP4 | \
00030                         INT_USBTXACTIVEEEP5 | \
00031                         INT_USBTXACTIVEEEP6 )
00032
00033 #define INT_USBRXVALID ( INT_USBRXVALIDEPO | \
00034                         INT_USBRXVALIDEPI | \
00035                         INT_USBRXVALIDEPII | \
00036                         INT_USBRXVALIDEPIII | \
00037                         INT_USBRXVALIDEPIV | \
00038                         INT_USBRXVALIDEPIV | \
00039                         INT_USBRXVALIDEPIVI )

```

```

00040
00041 //Define the size of all the endpoints.
00042 #define EP0_SIZE (8)    //CTRL endpoint
00043 #define EP1_SIZE (8)    //BULK\INTERRUPT
00044 #define EP2_SIZE (8)    //BULK\INTERRUPT
00045 #define EP3_SIZE (64)   //BULK\INTERRUPT
00046 #define EP4_SIZE (32)   //BULK\INTERRUPT
00047 #define EP5_SIZE (64)   //BULK\INTERRUPT
00048 #define EP6_SIZE (512)  //ISOCHRONOUS endpoint
00049 //The USB core in the chip does not distinguish between INTERRUPT and BULK.
00050 //The BULK/INTERRUPT endpoints can be morphed between each other's type by
00051 //simply updating the endpoint descriptor.
00052
00053 //All physical endpoints are packed together and accessed through
00054 //a contiguous block of RAM. Therefore, define the overall size of the
00055 //buffer by adding all the endpoints together. The multiply by 2 is
00056 //because each endpoint is bidirection with IN and OUT.
00057 #define USB_BUFFER_SIZE ( (EP0_SIZE*2) + \
00058             (EP1_SIZE*2) + \
00059             (EP2_SIZE*2) + \
00060             (EP3_SIZE*2) + \
00061             (EP4_SIZE*2) + \
00062             (EP5_SIZE*2) + \
00063             (EP6_SIZE*2) )
00064
00065 //Define all of the endpoints as byte arrays.
00066 typedef struct
00067 {
00068     uint8_t ep0IN[EP0_SIZE];
00069     uint8_t ep0OUT[EP0_SIZE];
00070
00071     uint8_t ep1IN[EP1_SIZE];
00072     uint8_t ep1OUT[EP1_SIZE];
00073
00074     uint8_t ep2IN[EP2_SIZE];
00075     uint8_t ep2OUT[EP2_SIZE];
00076
00077     uint8_t ep3IN[EP3_SIZE];
00078     uint8_t ep3OUT[EP3_SIZE];
00079
00080     uint8_t ep4IN[EP4_SIZE];
00081     uint8_t ep4OUT[EP4_SIZE];
00082
00083     uint8_t ep5IN[EP5_SIZE];
00084     uint8_t ep5OUT[EP5_SIZE];
00085
00086     uint8_t ep6IN[EP6_SIZE];
00087     uint8_t ep6OUT[EP6_SIZE];
00088 } EndPointStruct;
00089
00090 //Overlay the endpoint structure on top of a flat byte array of the entire
00091 //memory block that USB will be accessing. This way an endpoint can be
00092 //accessed specifically by name or the entire block can be accessed.
00093 typedef union
00094 {
00095     uint8_t allEps[USB_BUFFER_SIZE];
00096     EndPointStruct eps;
00097 } EndPointUnion;
00098
00099 extern EndPointUnion usbBufferA;
00100 extern EndPointUnion usbBufferB;
00101
00102 extern uint8_t *EPOUTBUF[7];
00103 extern uint8_t *EPINBUF[7];
00104 extern volatile uint32_t *EPOUTBUFSIZE[7];
00105 extern volatile uint32_t *EPINBUFSIZE[7];
00106
00107 extern bool usbVbusPresent;
00108
00109 static inline void USBHAL_DisableGlobalInt( void )
00110 {
00111     INT_CFGCLR = INT_USB;
00112     /* Disable all interrupts. */
00113     INT_USBCFG = 0;
00114
00115     /* Clear pending interrupts */
00116     INT_USBFLAG = 0xFFFFFFFF;
00117 }
00118
00119 static inline void USBHAL_EnableGlobalInt( void )

```

```

00120 {
00121     INT_CFGSET = INT_USB;
00122 }
00123
00124 static inline void USBDHAL_EnableUsbResetInt( void )
00125 {
00126     /* Disable all interrupts. */
00127     INT_USBCFG = 0;
00128
00129     /* Clear pending interrupts */
00130     INT_USBFLAG = 0xFFFFFFFF;
00131
00132 #ifndef BOOTLOADER
00133     INT_USBCFG = (INT_USBRESET | 
00134                 INT_USBRXVALIDEPO0 | 
00135                 INT_USBTXACTIVEEPO0 | 
00136                 INT_USBRESUME | 
00137                 INT_USBWAKEUP | 
00138                 INT_USBSUSPEND );
00139 #else
00140     INT_USBCFG = (INT_USBRESET | 
00141                 INT_USBRXVALIDEPO0 | 
00142                 INT_USBTXACTIVEEPO0 );
00143
00144 #endif
00145 }
00146
00147 static inline void USBDHAL_StartEp0Setup( USBD_Device_TypeDef *dev )
00148 {
00149     dev->ep[ 0 ].in = false;
00150     dev->setup = dev->setupPkt;
00151 }
00152
00153 static inline void USBDHAL_ReenableEp0Setup( USBD_Device_TypeDef *dev )
00154 {
00155     dev->setup = dev->setupPkt;
00156 }
00157
00158 static inline void USBDHAL_DeactivateEp( USBD_Ep_TypeDef *ep )
00159 {
00160     #ifdef USB_DEBUG_EP
00161         DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "Deactivating EP%d", ep->num);
00162     #endif
00163     if (ep->in)
00164     {
00165         #ifdef USB_DEBUG_EP
00166             #ifdef USB_DEBUG_VERBOSE
00167                 DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "in-->\r\n");
00168             #else
00169                 DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "in\r\n");
00170             #endif
00171         #endif
00172         USB_ENABLEIN &= ~(USB_ENABLEINEPO0      << ep->num);
00173         USB_CTRL      &= ~(USB_ENBUFINEPOB     << ep->num);
00174         INT_USBCFG    &= ~(INT_USBTXACTIVEEPO0 << ep->num);
00175     }
00176     else
00177     {
00178         #ifdef USB_DEBUG_EP
00179             #ifdef USB_DEBUG_VERBOSE
00180                 DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "out-->\r\n");
00181             #else
00182                 DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "out\r\n");
00183             #endif
00184         #endif
00185         USB_ENABLEOUT &= ~ (USB_ENABLEOUTEPO0 << ep->num);
00186         USB_CTRL      &= ~ (USB_ENBUFOUTEPOB   << ep->num);
00187         INT_USBCFG    &= ~ (INT_USBRXVALIDEPO0 << ep->num);
00188     }
00189
00190     #ifdef USB_DEBUG_EP
00191         #ifdef USB_DEBUG_VERBOSE
00192             DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "USB_ENABLEIN\tUSB_ENABLEOUT\t"
00193             USB_CTRL\tINT_USBCFG\r\n");
00193             DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "%X \t%X \t%X\r\n", USB_ENABLEIN, USB_ENABLEOUT, USB_CTRL, INT_USBCFG);
00194         #endif
00195     #endif
00196 }
00197

```

```

00198 static inline USB_Status_TypeDef USBDHAL_StallEp(
00199     USBD_Ep_TypeDef *ep )
00200 {
00201     USB_Status_TypeDef retVal = USB_STATUS_REQ_ERR
00202 ;
00203     if ( ep->in == true )
00204     {
00205         USB_STALLIN |= USB_STALLINEPO << ep->num;
00206         retVal = USB_STATUS_OK;
00207     }
00208     else
00209     {
00210         USB_STALLOUT |= USB_STALLOUTEPO << ep->num;
00211         retVal = USB_STATUS_OK;
00212     }
00213     return retVal;
00214 }
00215
00216 static inline USB_Status_TypeDef USBDHAL_UnStallEp(
00217     USBD_Ep_TypeDef *ep )
00218 {
00219     USB_Status_TypeDef retVal = USB_STATUS_REQ_ERR
00220 ;
00221     if ( ep->in == true )
00222     {
00223         USB_STALLIN &= ~USB_STALLINEPO << ep->num;
00224         retVal = USB_STATUS_OK;
00225     }
00226     else
00227     {
00228         USB_STALLOUT &= ~USB_STALLOUTEPO << ep->num;
00229         retVal = USB_STATUS_OK;
00230     }
00231     return retVal;
00232 }
00233
00234 static inline USB_Status_TypeDef USBDHAL_GetStallStatusEp(
00235             USBD_Ep_TypeDef *ep, uint16_t *halt )
00236 {
00237     USB_Status_TypeDef retVal = USB_STATUS_REQ_ERR
00238 ;
00239     if ( ep->in == true )
00240     {
00241         *halt = USB_STALLIN & (USB_STALLINEPO << ep->num) ? 1: 0;
00242         retVal = USB_STATUS_OK;
00243     }
00244     else
00245     {
00246         *halt = USB_STALLOUT & (USB_STALLOUTEPO << ep->num) ? 1: 0;
00247         retVal = USB_STATUS_OK;
00248     }
00249     return retVal;
00250 }
00251 }
00252
00253
00254 static inline bool USBDHAL_EpIsStalled( USBD_Ep_TypeDef *ep )
00255 {
00256     bool retVal = false;
00257     uint16_t stallStatus;
00258
00259     if ( USBDHAL_GetStallStatusEp( ep, &stallStatus ) == USB_STATUS_OK
00260     )
00261     {
00262         retVal = stallStatus & 1 ? true : false;
00263     }
00264     return retVal;
00265 }
00266
00267 static inline void USBDHAL_StartEpIn( USBD_Ep_TypeDef *ep )
00268 {
00269     uint8_t txSize = (ep->remaining > ep->packetSize) ? ep->packetSize : ep->
00270     remaining;
00271     if(txSize > 0)

```

```

00271  {
00272    if (ep->buf!=NULL)
00273    {
00274      MEMMOVE(EPINBUF[ep->num], ep->buf, txSize);
00275    }
00276  }
00277  ep->buf += txSize;
00278
00279
00280 #if (defined (USB_DEBUG_WRITE) && defined(USB_DEBUG_VERBOSE))
00281   uint8_t i;
00282   for(i=0;i<txSize;i++)
00283   {
00284     DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "%02X ", EPINBUF[ep->num][i]);
00285   }
00286   DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "\r\n");
00287 #endif
00288
00289 *EPINBUFSIZE[ep->num] = txSize;
00290
00291 USB_TXLOAD = USB_TXLOADEP0A << ep->num;
00292 }
00293
00294 static void USBDHAL_StartEpOut( USBD_Ep_TypeDef *ep )
00295 {
00296
00297 #if defined(USB_DEBUG_EP) && defined(USB_DEBUG_VERBOSE)
00298   uint8_t i;
00299   for(i=0;i<8;i++)
00300   {
00301     DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "%c", EPOUTBUF[ep->num][i]);
00302   }
00303   DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "\r\n");
00304 #endif
00305
00306 // USB_ENABLEOUT |= EPOUT[ep->num].ENABLE;
00307 // INT_USBCFG |= EPOUT[ep->num].INT;
00308 // USB_RXVALID |= (EPOUT[ep->num].RXVALIDA | EPOUT[ep->num].RXVALIDB);
00309 }
00310
00311
00312 static inline void USBDHAL_SetRemoteWakeup( void )
00313 {
00314   USB_RESUME = 1;
00315 }
00316
00317 // not supported
00318 void USBDHAL_AbortAllEps( void );
00319
00320 void USBDHAL_Connect( void );
00321 void USBDHAL_Disconnect( void );
00322 USB_Status_TypeDef USBDHAL_CoreInit( uint32_t totalRxFifoSize
00323 , uint32_t totalTxFifoSize );
00324 void USBDHAL_AbortAllTransfers( USB_Status_TypeDef reason );
00325
00326 #ifdef __cplusplus
00327 }
00328#endif
00329
00330#endif /* __EM_USBHAL_H */

```

8.59 em_usbint.c File Reference

```

#include <PLATFORM_HEADER>
#include "stack/include/ember.h"
#include "hal/hal.h"
#include "em_usb.h"
#include "em_usbdhal.h"
#include "em_usbtypes.h"
#include "em_usbd.h"
#include "app/util/serial/serial.h"

```

8.59.1 Detailed Description

USB protocol stack library, USB device peripheral interrupt handlers.

Author

Nathaniel Ting

Version

3.20.3

Definition in file [em_usbint.c](#).

8.60 em_usbint.c

```

00001 /***** *****
00002 #include PLATFORM_HEADER
00010 #include "stack/include/ember.h"
00011 #include "hal/hal.h"
00012
00013 #if defined(CORTEXM3_EM35X_USB)
00014 // Only define this ISR if USB is on and EnergyMicro USB is enabled. When this
00015 // function is defined it will cause extra code to be pulled in even if the ISR
00016 // will never fire.
00017
00018 #include "em_usb.h"
00019 #include "em_usbhal.h"
00020 #include "em_usbtypes.h"
00021 #include "em_usbd.h"
00022 #include "app/util/serial/serial.h"
00023
00026 USBD_Ep_TypeDef *ep;
00027
00028 void halUsbIsr(void)
00029 {
00030     uint32_t byteCount;
00031
00032     #ifdef BOOTLOADER
00033         VBUSMON_ISR();
00034     #endif
00035
00036     uint32_t status = INT_USBFLAG;
00037     INT_USBFLAG = status;
00038
00039     if (status == 0 )
00040     {
00041         return;
00042     }
00043     //Ensure EP0 IN is not stalled. Refer to receiving getQualifierDescriptor
00044     //for more information.
00045     USB_STALLIN &= ~USB_STALLINEP0;
00046
00047     if(status & INT_USBWAKEUP & INT_USBCFG)
00048     {
00049         #ifdef USB_DEBUG_SUSPEND
00050             DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "INT_USBWAKEUP\r\n");
00051         #endif
00052
00053         //The INT_USBWAKEUP interrupt triggers for a successful remote wakeup
00054         //This is essentially a resume only initiated by the device instead of host
00055         //need INT_USBCFG mask to enable turning on/off remote wakeup interrupt
00056
00057         #ifndef BOOTLOADER
00058             USBD_SetUsbState( USBD_STATE_POWERED );

```

```

00059     halResumeCallback();
00060     #ifndef EMBER_NO_STACK
00061         emberStackPowerUp();
00062     #endif // EMBER_NO_STACK
00063     #endif
00064 }
00065
00066 if(status & INT_USBRESUME)
00067 {
00068     #ifdef USB_DEBUG_SUSPEND
00069         DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "INT_USBRESUME\r\n");
00070     #endif
00071     //The INT_USBRESUME interrupt pulls us out of suspend while the USB
00072     //core auto clears the USBSUSP_CLKSEL bit (used when entering suspend).
00073
00074     #ifndef BOOTLOADER
00075         USBD_SetUsbState( USBD_STATE_POWERED );
00076         halResumeCallback();
00077     #ifndef EMBER_NO_STACK
00078         emberStackPowerUp();
00079     #endif // EMBER_NO_STACK
00080     #endif
00081 }
00082
00083 if(status & INT_USBSUSPEND)
00084 {
00085     #ifdef USB_DEBUG_SUSPEND
00086         DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "INT_USBSUSPEND\r\n");
00087     #endif
00088     //If suspend if being supported, we now have 7ms to get down to
00089     //suspend current. Setting usb state to USBD_STATE_SUSPENDED will allow
00090     //the usbSuspendDsr() function to place us in the appropriate low
00091     //power clocking and idle mode.
00092
00093     //NOTE: The stack and HAL should be shutdown like deep sleeping before
00094     //suspend is entered. USB sleep involves slowing down peripheral
00095     //clocking, but it does not stop the clocks. This means
00096     //peripherals will keep running if they're enabled. Being at
00097     //a slower clock speed can severely disrupt application behavior
00098     //that depends on clean peripheral behavior at a desired clock
00099     //rate. For example, problems with the UART would be very obvious.
00100
00101     #ifndef BOOTLOADER
00102         USBD_SetUsbState( USBD_STATE_SUSPENDED );
00103     #endif
00104     //USBTODO- Software needs to remember when INT_USBSUSPEND fires so that
00105     //it can return to suspend if a non-USB event pulls us out of
00106     //the USB sleep mode. Only INT_USBRESUME is allowed to keep us
00107     //out of suspend.
00108 }
00109
00110 if(status & INT_USBRESET)
00111 {
00112     #ifdef USB_DEBUG_INT
00113         DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "\r\n\r\nINT_USBRESET\r\n");
00114     #endif
00115     //When a USB reset occurs it resets the core but not the DMA. To ensure
00116     //transactions after a reset are fresh all buffers (DMA) need to be
00117     //cleared.
00118     USB_BUFCLEAR = (USB_BUFCLEARINEP6 |
00119                     USB_BUFCLEARINEP5 |
00120                     USB_BUFCLEARINEP4 |
00121                     USB_BUFCLEARINEP3 |
00122                     USB_BUFCLEARINEP2 |
00123                     USB_BUFCLEARINEP1 |
00124                     USB_BUFCLEARINEP0 );
00125
00126     /* Setup EP0 to receive SETUP packets */
00127     USBDHAL_StartEp0Setup( dev );
00128     USBD_Ep_TypeDef *ep;
00129     ep = &dev->ep[ 0 ];
00130     ep->state = D_EP_IDLE;
00131     ep->remaining = 0;
00132
00133     ep->in = true;
00134     USBDHAL_UnStallEp( ep );           /* Stall Ep0 IN */          */
00135     ep->in = false;                  /* OUT for next SETUP */   */
00136     USBDHAL_UnStallEp( ep );           /* Stall Ep0 OUT */        */
00137
00138     if ( dev->callbacks->usbReset )

```

```

00139     {
00140         dev->callbacks->usbReset ();
00141     }
00142
00143     // USBD_SetUsbState( USBD_STATE_DEFAULT );
00144     // USBDHAL_AbortAllTransfers( USB_STATUS_DEVICE_RESET );
00145 }
00146
00147 if(USB_RXBUFSIZEPOA == 0)
00148 {
00149     USB_BUFCLEAR = USB_BUFCLEARINEP0;
00150
00151     USBDHAL_StartEp0Setup( dev );
00152     USBD_Ep_TypeDef *ep;
00153     ep = &dev->ep[ 0 ];
00154     ep->state = D_EP_IDLE;
00155     ep->remaining = 0;
00156 }
00157
00158 //==== RX functionality ====
00159 if (status & INT_USBRXVALID)
00160 {
00161     int epnum;
00162     uint16_t epmask;
00163
00164     // sweep through eps to determine which EP the interrupt is on
00165     for ( epnum = 0,                      epmask = INT_USBRXVALIDEPO;
00166           epnum <= MAX_NUM_OUT_EPS;
00167           epnum++,                      epmask <= 1 )
00168     {
00169         if (status & epmask)
00170         {
00171             // determine how many bytes were received
00172             volatile uint32_t *bufsize = EPOUTBUFSIZE[epnum];
00173
00174             #ifdef USB_DEBUG_INT
00175                 #ifdef USB_DEBUG_VERBOSE
00176                     DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "RX%d-->",epnum);
00177                     uint8_t i;
00178                     for(i=0;i<*bufsize;i++)
00179                     {
00180                         DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "%02X", EPOUTBUF[epnum][i]);
00181                     }
00182                     DEBUG_BUFFER += sprintf(DEBUG_BUFFER, " %d bytes\r\n", *bufsize);
00183                 #else
00184                     DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "RX%d-->%d bytes\r\n",epnum, *
00185                     bufsize);
00186                 #endif
00187             #endif
00188
00189             // update endpoint state
00190             ep = USBD_GetEpFromAddr( epnum );
00191             byteCount = EFM32_MIN(ep->remaining,*bufsize);
00192             ep->remaining -= byteCount;
00193             ep->xferred += byteCount;
00194
00195             // #ifdef USB_DEBUG_INT
00196             //     DEBUG_BUFFER += sprintf(DEBUG_BUFFER, " %d remaining, %d
00197             xferred\r\n", ep->remaining, ep->xferred);
00198             // #endif
00199
00200             // copy buffer to endpoint specified buffer location before erasing
00201             if (ep->state == D_EP RECEIVING)
00202             {
00203                 MEMMOVE(ep->buf,EPOUTBUF[ep->num], *bufsize);
00204             }
00205
00206             // dispatch endpoint handler, erase buffer, reenable endpoint RX
00207             if (epnum ==0)
00208             {
00209                 USBDEP_Ep0Handler( dev );
00210             }
00211             else
00212             {
00213                 #if EM_SERIAL3_ENABLED
00214                     halInternalUart3RxIsr(EPOUTBUF[ep->num], *
00215                     bufsize);
00216                 #endif //EM_SERIAL3_ENABLED
00217             }
00218         }
00219     }
00220 }
```

```

00215         USBDEP_EpHandler(ep->addr);
00216     }
00217
00218     MEMSET(EPOUTBUF[epnum], 0, *bufsize);
00219     USB_RXVALID = ((USB_RXVALIDEP0A<<epnum) | (USB_RXVALIDEP0B<<epnum));
00220 }
00221 }
00222 }
00223
00224 //==== TX functionality ====
00225 //NOTE: INT_USBTXACTIVEEEPx interrupts fire on USB_TXACTIVEEPxy falling edge.
00226 if (status & INT_USBTXACTIVE)
00227 {
00228     int epnum;
00229     uint16_t epmask;
00230
00231     // sweep through eps to determine which EP the interrupt is on
00232     for ( epnum = 0,           epmask = INT_USBTXACTIVEEP0;
00233           epnum <= MAX_NUM_OUT_EPS;   epmask <= 1 )
00234         epnum++,                 epmask <<= 1 )
00235
00236     if (status & epmask)
00237     {
00238         // determine how many bytes were transmitted
00239         volatile int32u *bufsize = EPINBUFSIZE[epnum];
00240
00241         #ifdef USB_DEBUG_INT
00242             DEBUG_BUFFER += sprintf(DEBUG_BUFFER, "TX%d\r\n",epnum);
00243         #endif
00244
00245         // update endpoint state
00246         ep = USBD_GetEpFromAddr( USB_SETUP_DIR_MASK | epnum )
00247 ;
00248         byteCount = EFM32_MIN(ep->remaining,*bufsize);
00249         ep->remaining -= byteCount;
00250         ep->xferred += byteCount;
00251
00252         // dispatch endpoint handler
00253         if (epnum == 0)
00254         {
00255             USBDEP_Ep0Handler( dev );
00256         }
00257         else
00258         {
00259             USBDEP_EpHandler(ep->addr);
00260         }
00261     }
00262 }
00263 }
00264
00265 #ifdef VBUSMON
00266 void VBUSMON_ISR(void)
00267 {
00268     //VBUSMON is used for sensing VBUS indicating if USB is physically connected.
00269     //An IRQ is used to monitor for any edge change on VBUSMON.
00270     //For self-powered devices, VBUS monitoring is required for the EM358 device
00271     //to function cleanly across physical connect/disconnects of the USB.
00272     //The USB specification requires that the pull-up resistor for
00273     //enumerating is disconnected if VBUS is not connected.
00274
00275     //clear int before read to avoid potential of missing interrupt
00276     INT_MISS = VBUSMON_MISS_BIT;      //clear missed interrupt
00277     flag
00278     INT_GPIOFLAG = VBUSMON_FLAG_BIT; //clear top level interrupt
00279     flag
00280
00281     //Because this ISR/IRQ triggers on both edges, usbVbusPresent is used
00282     //to track the state of VBUS.
00283     //Always directly read VBUSMON to ensure this ISR is changing to the correct,
00284     //current state.
00285     //Falling edge: VBUS removed - Must set ENUMCTRL to input-low.
00286     //Rising edge: VBUS applied - If usbConfigEnumerate() has been called and
00287     //this device is ready to enumerate, set ENUMCTRL to output-high and
00288     //perform enumeration.
00289     usbVbusPresent = ((VBUSMON_IN & VBUSMON) == VBUSMON);
00290
00291     if(usbVbusPresent)
00292     {
00293         if (USBD_GetUsbState() == USBD_STATE_CONFIGURED

```

```

        )
00292     return;
00293 //Only attempt enumerate if usbConfigEnumerate has already been called.
00294 //Set ENUMCTRL ouput-high to enumerate.
00295 ENUMCTRL_SETCFG(GPIOCFG_OUT);
00296 ENUMCTRL_SET();
00297
00298 // halSetLed(BOARDLED0);
00299 USBD_SetUsbState(USBD_STATE_DEFAULT);
00300 }
00301 else
00302 {
00303 //Configure ENUMCTRL as an input so the device de-enumerates and
00304 //the pin does not put any load on the bus. (The spec says the
00305 //pull-up resistor used for enumeration should be tri-state. On
00306 //EM358 devices the input configuration is the best available choice
00307 //since tri-state isn't possible.)
00308 //Clear down ENUMCTRL to be ready for the next time enumerating.
00309 ENUMCTRL_SETCFG(GPIOCFG_IN);
00310 ENUMCTRL_CLR();
00311 // halClearLed(BOARDLED0);
00312 USBD_SetUsbState(USBD_STATE_NONE);
00313 }
00314 }
00315 #endif // VBUSMON
00316
00319 #endif // CORTEXM3_EM35X_USB

```

8.61 em_usotypes.h File Reference

8.61.1 Detailed Description

USB protocol stack library, internal type definitions.

Author

Nathaniel Ting

Version

3.20.3

Definition in file [em_usotypes.h](#).

8.62 em_usotypes.h

```

00001 /***** *****/
00008 #ifndef __EM_USBTYPES_H
00009 #define __EM_USBTYPES_H
0010
00011 // #include "em_device.h"
00012 // #include "em_usb.h"
00013
00014 #ifdef __cplusplus
00015 extern "C" {
00016 #endif
00017
00020 /* Limits imposed by the USB peripheral */
00021 #define NP_RX_QUE_DEPTH          8
00022 #define HP_RX_QUE_DEPTH          8
00023 // #define MAX_XFER_LEN           524287L      /* 2^19 - 1 bytes
00024 */ // #define MAX_PACKETS_PR_XFER    1023       /* 2^10 - 1 packets
00025 */ // #define MAX_NUM_TX_FIFOS       6          /* In addition to EP0 Tx FIFO */

```

```

00026 #define MAX_NUM_IN_EPS      6          /* In addition to EP0      */
00027 #define MAX_NUM_OUT_EPS     6          /* In addition to EP0      */
00028
00029 /* Limit imposed by the USB standard */
00030 #define MAX_USB_EP_NUM    12
00031
00032
00033
00034
00035 /* Macros for selecting a hardware timer. */
00036 #define USB_TIMER0 0
00037 #define USB_TIMER1 1
00038
00039 // #ifndef NUM_EP_USED
00040 // #define NUM_EP_USED 0
00041 // #endif
00042
00043
00044 typedef enum
00045 {
00046     D_EP_IDLE           = 0,
00047     D_EP_TRANSMITTING   = 1,
00048     D_EP RECEIVING      = 2,
00049     D_EP_STATUS         = 3
00050 } USBD_EpState_TypeDef;
00051
00052 typedef struct
00053 {
00054     bool                 in;
00055     uint8_t              zlp;
00056     uint8_t              num;
00057     uint8_t              addr;
00058     uint8_t              type;
00059     uint8_t              txFifoNum;
00060     uint8_t              *buf;
00061     uint16_t             packetSize;
00062     uint16_t             mask;
00063     uint32_t             remaining;
00064     uint32_t             xferred;
00065     uint32_t             hwXferSize;
00066     uint32_t             fifoSize;
00067     USBD_EpState_TypeDef state;
00068     USB_XferCompleteCb_TypeDef xferCompleteCb;
00069 } USBD_Ep_TypeDef;
00070
00071 typedef struct
00072 {
00073     USB_Setup_TypeDef          *setup;
00074     USB_Setup_TypeDef          setupPkt[3];
00075     uint8_t                     configurationValue; /* Must be DWORD
00076     aligned */
00077     bool                        remoteWakeupEnabled;
00078     uint8_t                     numberofStrings;
00079     USBD_State_TypeDef          state;
00080     USBD_State_TypeDef          savedState;
00081     USBD_State_TypeDef          lastState;
00082     const USB_DeviceDescriptor_TypeDef *deviceDescriptor;
00083     const USB_ConfigurationDescriptor_TypeDef *configDescriptor;
00084     const void * const          stringDescriptors;
00085     const USBD_Callbacks_TypeDef *callbacks;
00086     USBD_Ep_TypeDef             ep[ NUM_EP_USED + 1 ];
00087     uint8_t                     inEpAddr2EpIndex[ MAX_USB_EP_NUM + 1 ];
00088     uint8_t                     outEpAddr2EpIndex[ MAX_USB_EP_NUM + 1 ];
00089
00090
00093 #ifdef __cplusplus
00094 }
00095 #endif
00096
00097 #endif /* __EM_USBTYPES_H */

```

8.63 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_APSS_UNICAST_MESSAGE_COUNT
- #define EMBER_BINDING_TABLE_SIZE
- #define EMBER_ADDRESS_TABLE_SIZE
- #define EMBER_RESERVED_MOBILE_CHILD_ENTRIES
- #define EMBER_ROUTE_TABLE_SIZE
- #define EMBER_DISCOVERY_TABLE_SIZE
- #define EMBER_MULTICAST_TABLE_SIZE
- #define EMBER_SOURCE_ROUTE_TABLE_SIZE
- #define EMBER_DEFAULT_BROADCAST_TABLE_SIZE
- #define EMBER_BROADCAST_TABLE_SIZE
- #define EMBER_RETRY_QUEUE_SIZE
- #define EMBER_ASSERT_SERIAL_PORT
- #define EMBER_MAXIMUM_ALARM_DATA_SIZE
- #define EMBER_BROADCAST_ALARM_DATA_SIZE
- #define EMBER_UNICAST_ALARM_DATA_SIZE
- #define EMBER_FRAGMENT_DELAY_MS
- #define EMBER_FRAGMENT_MAX_WINDOW_SIZE
- #define EMBER_FRAGMENT_WINDOW_SIZE
- #define EMBER_BINDING_TABLE_TOKEN_SIZE
- #define EMBER_CHILD_TABLE_TOKEN_SIZE
- #define EMBER_KEY_TABLE_TOKEN_SIZE
- #define EMBER_REQUEST_KEY_TIMEOUT
- #define EMBER_TRANSIENT_KEY_TIMEOUT_S
- #define EMBER_END_DEVICE_BIND_TIMEOUT
- #define EMBER_PAN_ID_CONFLICT_REPORT_THRESHOLD
- #define EMBER_TASK_COUNT
- #define EMBER_MAX_SUPPORTED_NETWORKS

- #define EMBER_SUPPORTED_NETWORKS
- #define EMBER_ZLL_GROUP_ADDRESSES
- #define EMBER_ZLL_RSSI_THRESHOLD
- #define EMBER_RF4CE_PAIRING_TABLE_SIZE
- #define EMBER_RF4CE_PAIRING_TABLE_TOKEN_SIZE
- #define EMBER_RF4CE_PENDING_OUTGOING_PACKET_TABLE_SIZE
- #define EMBER_GP_PROXY_TABLE_SIZE
- #define EMBER_GP_PROXY_TABLE_TOKEN_SIZE
- #define EMBER_GP_SINK_TABLE_SIZE
- #define EMBER_GP_SINK_TABLE_TOKEN_SIZE

8.63.1 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](#).

8.64 ember-configuration-defaults.h

```

00001 // Todo:
00014 // - explain how to use a configuration header
00015 // - the documentation of the custom handlers should
00016 //   go in hal/ember-configuration.c, not here
00017 // - the stack profile documentation is out of date
00019
00047 #ifndef __EMBER_CONFIGURATION_DEFAULTS_H__
00048 #define __EMBER_CONFIGURATION_DEFAULTS_H__
00049
00050 #ifdef CONFIGURATION_HEADER
00051     #include CONFIGURATION_HEADER
00052 #endif
00053
00054 #ifndef EMBER_API_MAJOR_VERSION
00055
00058     #define EMBER_API_MAJOR_VERSION 2
00059 #endif
00060
00061 #ifndef EMBER_API_MINOR_VERSION
00062
00065     #define EMBER_API_MINOR_VERSION 0
00066 #endif
00067
00080 #ifndef EMBER_STACK_PROFILE
00081     #define EMBER_STACK_PROFILE 0
00082 #endif
00083
00084 #if (EMBER_STACK_PROFILE == 2)
00085     #define EMBER_MAX_DEPTH          15
00086     #define EMBER_SECURITY_LEVEL      5
00087     #define EMBER_MIN_ROUTE_TABLE_SIZE 10
00088     #define EMBER_MIN_DISCOVERY_TABLE_SIZE 4
00089     #define EMBER_INDIRECT_TRANSMISSION_TIMEOUT 7680
00090     #define EMBER_SEND_MULTICASTS_TO_SLEEPY_ADDRESS false
00091 #endif

```

```

00092
00093 #ifndef EMBER_MAX_END_DEVICE_CHILDREN
00094
00098 #define EMBER_MAX_END_DEVICE_CHILDREN 6
00099 #endif
00100
00101 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00102 /* Need to put in a compile time check to make sure that we aren't specifying
00103 * too many child devices. The NCP may or may not support 64 end devices. But
00104 * the host code doesn't matter.
00105 */
00106 #if defined(HAL_HAS_INT64) || defined(EZSP_HOST)
00107     #if EMBER_MAX_END_DEVICE_CHILDREN > 64
00108         #error "EMBER_MAX_END_DEVICE_CHILDREN can not exceed 64."
00109     #endif
00110 #else
00111     #if EMBER_MAX_END_DEVICE_CHILDREN > 32
00112         #error "EMBER_MAX_END_DEVICE_CHILDREN can not exceed 32."
00113     #endif
00114 #endif
00115
00116 #endif // DOXYGEN_SHOULD_SKIP_THIS
00117
00118 #ifndef EMBER_SECURITY_LEVEL
00119
00123 #define EMBER_SECURITY_LEVEL 5
00124 #endif
00125
00126 #if ! (EMBER_SECURITY_LEVEL == 0           \
00127       || EMBER_SECURITY_LEVEL == 5)
00128     #error "Unsupported security level"
00129 #endif
00130
00131 #ifdef EMBER_CHILD_TABLE_SIZE
00132     #if (EMBER_MAX_END_DEVICE_CHILDREN < EMBER_CHILD_TABLE_SIZE)
00133         #undef EMBER_CHILD_TABLE_SIZE
00134     #endif
00135 #endif
00136
00137 #ifndef EMBER_CHILD_TABLE_SIZE
00138
00152 #define EMBER_CHILD_TABLE_SIZE EMBER_MAX_END_DEVICE_CHILDREN
00153 #endif
00154
00168 #ifndef EMBER_KEY_TABLE_SIZE
00169     #define EMBER_KEY_TABLE_SIZE 0
00170 #endif
00171
00181 #ifndef EMBER_CERTIFICATE_TABLE_SIZE
00182     #define EMBER_CERTIFICATE_TABLE_SIZE 0
00183 #else
00184     #if EMBER_CERTIFICATE_TABLE_SIZE > 1
00185         #error "EMBER_CERTIFICATE_TABLE_SIZE > 1 is not supported!"
00186     #endif
00187 #endif
00188
00194 #ifndef EMBER_MAX_DEPTH
00195     #define EMBER_MAX_DEPTH 15
00196 #elif (EMBER_MAX_DEPTH > 15)
00197     // Depth is a 4-bit field
00198     #error "EMBER_MAX_DEPTH cannot be greater than 15"
00199 #endif
00200
00207 #ifndef EMBER_MAX_HOPS
00208     #define EMBER_MAX_HOPS (2 * EMBER_MAX_DEPTH)
00209 #endif
00210
00217 #ifndef EMBER_PACKET_BUFFER_COUNT
00218     #define EMBER_PACKET_BUFFER_COUNT 75
00219 #endif
00220
00232 #define EMBER_MAX_NEIGHBOR_TABLE_SIZE 16
00233 #ifndef EMBER_NEIGHBOR_TABLE_SIZE
00234     #define EMBER_NEIGHBOR_TABLE_SIZE 16
00235 #endif
00236
00243 #ifndef EMBER INDIRECT TRANSMISSION TIMEOUT
00244     #define EMBER INDIRECT TRANSMISSION TIMEOUT 3000
00245 #endif
00246 #define EMBER_MAX INDIRECT TRANSMISSION TIMEOUT 30000

```

```

00247 #if (EMBER_INDIRECT_TRANSMISSION_TIMEOUT
00248     > EMBER_MAX_INDIRECT_TRANSMISSION_TIMEOUT)
00249     #error "Indirect transmission timeout too large."
00250 #endiff
00251
00258 #ifndef EMBER_SEND_MULTICASTS_TO_SLEEPY_ADDRESS
00259     #define EMBER_SEND_MULTICASTS_TO_SLEEPY_ADDRESS false
00260 #endiff
00261
00262
00277 #ifndef EMBER_END_DEVICE_POLL_TIMEOUT
00278     #define EMBER_END_DEVICE_POLL_TIMEOUT 5
00279 #endiff
00280
00288 #ifndef EMBER_END_DEVICE_POLL_TIMEOUT_SHIFT
00289     #define EMBER_END_DEVICE_POLL_TIMEOUT_SHIFT 6
00290 #endiff
00291
00298 #ifndef EMBER_MOBILE_NODE_POLL_TIMEOUT
00299     #define EMBER_MOBILE_NODE_POLL_TIMEOUT 20
00300 #endiff
00301
00314 #ifndef EMBERAPS_UNICAST_MESSAGE_COUNT
00315     #define EMBERAPS_UNICAST_MESSAGE_COUNT 10
00316 #endiff
00317
00320 #ifndef EMBER_BINDING_TABLE_SIZE
00321     #define EMBER_BINDING_TABLE_SIZE 0
00322 #endiff
00323
00328 #ifndef EMBER_ADDRESS_TABLE_SIZE
00329     #define EMBER_ADDRESS_TABLE_SIZE 8
00330 #endiff
00331
00338 #ifndef EMBER_RESERVED_MOBILE_CHILD_ENTRIES
00339     #define EMBER_RESERVED_MOBILE_CHILD_ENTRIES 0
00340 #endiff
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     #endiff
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 #endiff
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     #endiff
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 #endiff
00374
00380 #ifndef EMBER_MULTICAST_TABLE_SIZE
00381     #define EMBER_MULTICAST_TABLE_SIZE 8
00382 #endiff
00383
00390 #ifndef EMBER_SOURCE_ROUTE_TABLE_SIZE
00391     #define EMBER_SOURCE_ROUTE_TABLE_SIZE 32
00392 #endiff
00393
00406 // TODO: we don't have one stack profile anymore, we might have two networks
00407 // the first one is Zigbee Pro, the second one is RF4CE.
00408 #if (EMBER_STACK_PROFILE == 2) && !defined(EMBER_TEST)
00409     #if defined(EMBER_BROADCAST_TABLE_SIZE)
00410         #error "Cannot override broadcast table size unless (EMBER_STACK_PROFILE != 2)"
00411     #endiff
00412 #endiff
00413
00414 #define EMBER_DEFAULT_BROADCAST_TABLE_SIZE 15
00415

```

```

00416 #ifndef EMBER_BROADCAST_TABLE_SIZE
00417     #define EMBER_BROADCAST_TABLE_SIZE EMBER_DEFAULT_BROADCAST_TABLE_SIZE
00418 #elif EMBER_BROADCAST_TABLE_SIZE < EMBER_DEFAULT_BROADCAST_TABLE_SIZE
00419     #error "EMBER_BROADCAST_TABLE_SIZE is less than the minimum value of 15."
00420 #elif 254 < EMBER_BROADCAST_TABLE_SIZE
00421     #error "EMBER_BROADCAST_TABLE_SIZE is larger than the maximum value of 254."
00422 #endif
00423
00424
00425 #ifndef EMBER_RETRY_QUEUE_SIZE
00426     #define EMBER_RETRY_QUEUE_SIZE 8
00427 #endif
00428
00429
00430
00440 #if !defined(EMBER_ASSERT_OUTPUT_DISABLED) \
00441     && !defined(EMBER_ASSERT_SERIAL_PORT)
00442     #define EMBER_ASSERT_SERIAL_PORT 1
00443 #endif
00444
00458 #define EMBER_MAXIMUM_ALARM_DATA_SIZE 16
00459
00477 #ifndef EMBER_BROADCAST_ALARM_DATA_SIZE
00478     #define EMBER_BROADCAST_ALARM_DATA_SIZE 0
00479 #elif EMBER_MAXIMUM_ALARM_DATA_SIZE < EMBER_BROADCAST_ALARM_DATA_SIZE
00480     #error "EMBER_BROADCAST_ALARM_DATA_SIZE is too large."
00481 #endif
00482
00491 #ifndef EMBER_UNICAST_ALARM_DATA_SIZE
00492     #define EMBER_UNICAST_ALARM_DATA_SIZE 0
00493 #elif EMBER_MAXIMUM_ALARM_DATA_SIZE < EMBER_UNICAST_ALARM_DATA_SIZE
00494     #error "EMBER_UNICAST_ALARM_DATA_SIZE is too large."
00495 #endif
00496
00500 #ifndef EMBER_FRAGMENT_DELAY_MS
00501     #define EMBER_FRAGMENT_DELAY_MS 0
00502 #endif
00503
00507 #define EMBER_FRAGMENT_MAX_WINDOW_SIZE 8
00508
00513 #ifndef EMBER_FRAGMENT_WINDOW_SIZE
00514     #define EMBER_FRAGMENT_WINDOW_SIZE 1
00515 #elif EMBER_FRAGMENT_MAX_WINDOW_SIZE < EMBER_FRAGMENT_WINDOW_SIZE
00516     #error "EMBER_FRAGMENT_WINDOW_SIZE is too large."
00517 #endif
00518
00519 #ifndef EMBER_BINDING_TABLE_TOKEN_SIZE
00520     #define EMBER_BINDING_TABLE_TOKEN_SIZE EMBER_BINDING_TABLE_SIZE
00521 #endif
00522 #ifndef EMBER_CHILD_TABLE_TOKEN_SIZE
00523     #define EMBER_CHILD_TABLE_TOKEN_SIZE EMBER_CHILD_TABLE_SIZE
00524 #endif
00525 #ifndef EMBER_KEY_TABLE_TOKEN_SIZE
00526     #define EMBER_KEY_TABLE_TOKEN_SIZE EMBER_KEY_TABLE_SIZE
00527 #endif
00528
00541 #ifndef EMBER_REQUEST_KEY_TIMEOUT
00542     #define EMBER_REQUEST_KEY_TIMEOUT 0
00543 #elif EMBER_REQUEST_KEY_TIMEOUT > 10
00544     #error "EMBER_REQUEST_KEY_TIMEOUT is too large."
00545 #endif
00546
00555 #ifndef EMBER_TRANSIENT_KEY_TIMEOUT_S
00556     #define EMBER_TRANSIENT_KEY_TIMEOUT_S (300)
00557 #endif
00558
00562 #ifndef EMBER_END_DEVICE_BIND_TIMEOUT
00563     #define EMBER_END_DEVICE_BIND_TIMEOUT 60
00564 #endif
00565
00574 #ifndef EMBER_PAN_ID_CONFLICT_REPORT_THRESHOLD
00575     #define EMBER_PAN_ID_CONFLICT_REPORT_THRESHOLD 1
00576 #endif
00577
00583 #ifndef EMBER_TASK_COUNT
00584     #define EMBER_TASK_COUNT (4)
00585 #endif
00586
00589 #define EMBER_MAX_SUPPORTED_NETWORKS 2
00590 #ifndef EMBER_SUPPORTED_NETWORKS

```

```

00591 #ifdef EMBER_TEST
00592     #define EMBER_SUPPORTED_NETWORKS 2
00593 #else
00594     #define EMBER_SUPPORTED_NETWORKS 1
00595 #endif
00596 #endif
00597
00598 #ifndef EMBER_ZLL_GROUP_ADDRESSES
00599     #define EMBER_ZLL_GROUP_ADDRESSES 1
00600 #endif
00603
00604 #ifndef EMBER_ZLL_RSSI_THRESHOLD
00605     #define EMBER_ZLL_RSSI_THRESHOLD -128
00608 #endif
00609
00610 #ifndef EMBER_RF4CE_PAIRING_TABLE_SIZE
00611
00613     #define EMBER_RF4CE_PAIRING_TABLE_SIZE      0
00614 #endif
00615
00616 #ifndef EMBER_RF4CE_PAIRING_TABLE_TOKEN_SIZE
00617
00619     #define EMBER_RF4CE_PAIRING_TABLE_TOKEN_SIZE EMBER_RF4CE_PAIRING_TABLE_SIZE
00620 #endif
00621
00622 #ifndef EMBER_RF4CE_PENDING_OUTGOING_PACKET_TABLE_SIZE
00623
00625     #define EMBER_RF4CE_PENDING_OUTGOING_PACKET_TABLE_SIZE 0
00626 #endif
00627
00628 #ifndef EMBER_GP_PROXY_TABLE_SIZE
00629
00631     #define EMBER_GP_PROXY_TABLE_SIZE 5
00632 //XXZEZSP
00633 #endif
00634
00635 #ifndef EMBER_GP_PROXY_TABLE_TOKEN_SIZE
00636
00638     #define EMBER_GP_PROXY_TABLE_TOKEN_SIZE EMBER_GP_PROXY_TABLE_SIZE
00639 #endif
00640
00641 #ifndef EMBER_GP_SINK_TABLE_SIZE
00642
00644     #define EMBER_GP_SINK_TABLE_SIZE 0
00645 #endif
00646
00647 #ifndef EMBER_GP_SINK_TABLE_TOKEN_SIZE
00648
00650     #define EMBER_GP_SINK_TABLE_TOKEN_SIZE EMBER_GP_SINK_TABLE_SIZE
00651 #endif
00652
00653
00656 #endif //__EMBER_CONFIGURATION_DEFAULTS_H__

```

8.65 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 (uint8_t *start, uint8_t *end)

- void `emberDebugBinaryPrintf` (PGM_P formatString,...)
- void `emDebugSendVuartMessage` (uint8_t *buff, uint8_t len)
- void `emberDebugError` (EmberStatus code)
- bool `emberDebugReportOff` (void)
- void `emberDebugReportRestore` (bool state)
- void `emberDebugPrintf` (PGM_P formatString,...)

8.65.1 Detailed Description

See [Debugging Utilities](#) for documentation.

Definition in file `ember-debug.h`.

8.66 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
00024
00031 #define emberDebugInit(port) do {} while(false)
00032
00033
00034
00035 #if (DEBUG_LEVEL >= BASIC_DEBUG) || defined(DOXYGEN_SHOULD_SKIP_THIS)
00036
00042 void emberDebugAssert(PGM_P filename, int linenumber);
00043
00044
00052 void emberDebugMemoryDump(uint8_t *start, uint8_t *end);
00053
00080 void emberDebugBinaryPrintf(PGM_P formatString, ...);
00081
00089 void emDebugSendVuartMessage(uint8_t *buff, uint8_t len
    );
00090
00091 //##if (DEBUG_LEVEL == FULL_DEBUG) || defined(DOXYGEN_SHOULD_SKIP_THIS)
00096 void emberDebugError(EmberStatus code);
00097
00102 bool emberDebugReportOff(void);
00103
00109 void emberDebugReportRestore(bool state);
00110
00111 // Format: Same as emberSerialPrintf
00112 // emberDebugPrintf("format string"[, parameters ...])
00128 void emberDebugPrintf(PGM_P formatString, ...);
00129
00130 #else // (DEBUG_LEVEL >= BASIC_DEBUG) || defined(DOXYGEN_SHOULD_SKIP_THIS)
00131     #define emberDebugAssert(filename, linenumber) do {} while(false)
00132     #define emberDebugMemoryDump(start, end) do {} while(false)
00133     #define emberDebugBinaryPrintf(formatString, ...) do {} while(false)
00134     #define emDebugSendVuartMessage(buff, len) do {} while(false)
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 >= BASIC_DEBUG) || defined(DOXYGEN_SHOULD_SKIP_THIS)
00142
00143
00144
00145 //##else // (DEBUG_LEVEL == FULL_DEBUG) || defined(DOXYGEN_SHOULD_SKIP_THIS)
00146
00147 //##endif // (DEBUG_LEVEL == FULL_DEBUG) || defined(DOXYGEN_SHOULD_SKIP_THIS)

```

```

00148
00152 #endif // __EMBER_DEBUG_H__
00153

```

8.67 ember-types.h File Reference

```

#include "stack/include/error.h"
#include "stack/include/zll-types.h"
#include "stack/include/rf4ce-types.h"
#include "stack/include/gp-types.h"

```

Data Structures

- struct [EmberReleaseTypeStruct](#)
A structure relating version types to human readable strings.
- struct [EmberVersion](#)
Version struct containing all version information.
- struct [EmberZigbeeNetwork](#)
Defines a ZigBee network and the associated parameters.
- struct [EmberNetworkInitStruct](#)
Defines the network initialization configuration that should be used when [emberNetwork-InitExtended\(\)](#) is called by the application.
- struct [EmberNetworkParameters](#)
Holds network parameters.
- struct [EmberApsFrame](#)
An in-memory representation of a ZigBee APS frame of an incoming or outgoing message.
- struct [EmberBindingTableEntry](#)
Defines an entry in the binding table.
- struct [EmberNeighborTableEntry](#)
Defines an entry in the neighbor table.
- struct [EmberRouteTableEntry](#)
Defines an entry in the route table.
- struct [EmberMulticastTableEntry](#)
Defines an entry in the multicast table.
- struct [EmberEventControl](#)
Control structure for events.
- struct [EmberEventData_S](#)
Complete events with a control and a handler procedure.
- struct [EmberTaskControl](#)
Control structure for tasks.
- struct [EmberKeyData](#)
This data structure contains the key data that is passed into various other functions.
- struct [EmberCertificateData](#)
This data structure contains the certificate data that is used for Certificate Based Key Exchange (CBKE).
- struct [EmberPublicKeyData](#)

This data structure contains the public key data that is used for Certificate Based Key Exchange (CBKE).

- struct [EmberPrivateKeyData](#)

This data structure contains the private key data that is used for Certificate Based Key Exchange (CBKE).

- struct [EmberSmacData](#)

This data structure contains the Shared Message Authentication Code (SMAC) data that is used for Certificate Based Key Exchange (CBKE).

- struct [EmberSignatureData](#)

This data structure contains a DSA signature. It is the bit concatenation of the 'r' and 's' components of the signature.

- struct [EmberMessageDigest](#)

This data structure contains an AES-MMO Hash (the message digest).

- struct [EmberAesMmoHashContext](#)

This data structure contains the context data when calculating an AES MMO hash (message digest).

- struct [EmberCertificate283k1Data](#)

This data structure contains the certificate data that is used for Certificate Based Key Exchange (CBKE) in SECT283k1 Elliptical Cryptography.

- struct [EmberPublicKey283k1Data](#)

This data structure contains the public key data that is used for Certificate Based Key Exchange (CBKE) in SECT283k1 Elliptical Cryptography.

- struct [EmberPrivateKey283k1Data](#)

This data structure contains the private key data that is used for Certificate Based Key Exchange (CBKE) in SECT283k1 Elliptical Cryptography.

- struct [EmberSignature283k1Data](#)

This data structure contains a DSA signature used in SECT283k1 Elliptical Cryptography. It is the bit concatenation of the 'r' and 's' components of the signature.

- struct [EmberInitialSecurityState](#)

This describes the Initial Security features and requirements that will be used when forming or joining the network.

- struct [EmberCurrentSecurityState](#)

This describes the security features used by the stack for a joined device.

- struct [EmberKeyStruct](#)

This describes a one of several different types of keys and its associated data.

- struct [EmberMfgSecurityStruct](#)

This structure is used to get/set the security config that is stored in manufacturing tokens.

- struct [EmberMacFilterMatchStruct](#)

This structure indicates a matching raw MAC message has been received by the application configured MAC filters.

Macros

- #define EMBER_MIN_BROADCAST_ADDRESS
- #define emberIsZigbeeBroadcastAddress(address)
- #define EMBER_JOIN_DECISION_STRINGS
- #define EMBER_DEVICE_UPDATE_STRINGS
- #define emberInitializeNetworkParameters(parameters)
- #define EMBER_COUNTER_STRINGS

- #define EMBER_STANDARD_SECURITY_MODE
- #define EMBER_TRUST_CENTER_NODE_ID
- #define EMBER_NO_TRUST_CENTER_MODE
- #define EMBER_GLOBAL_LINK_KEY
- #define EMBER_MFG_SECURITY_CONFIG_MAGIC_NUMBER
- #define EMBER_MAC_FILTER_MATCH_ENABLED_MASK
- #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_MASK
- #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_MASK
- #define EMBER_MAC_FILTER_MATCH_ON_DEST_MASK
- #define EMBER_MAC_FILTER_MATCH_ON_SOURCE_MASK
- #define EMBER_MAC_FILTER_MATCH_ENABLED
- #define EMBER_MAC_FILTER_MATCH_DISABLED
- #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_NONE
- #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_LOCAL
- #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_BROADCAST
- #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_NONE
- #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_NON_LOCAL
- #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_LOCAL
- #define EMBER_MAC_FILTER_MATCH_ON_DEST_BROADCAST_SHORT
- #define EMBER_MAC_FILTER_MATCH_ON_DEST_UNICAST_SHORT
- #define EMBER_MAC_FILTER_MATCH_ON_DEST_UNICAST_LONG
- #define EMBER_MAC_FILTER_MATCH_ON_SOURCE_LONG
- #define EMBER_MAC_FILTER_MATCH_ON_SOURCE_SHORT
- #define EMBER_MAC_FILTER_MATCH_ON_SOURCE_NONE
- #define EMBER_MAC_FILTER_MATCH_END
- #define WEAK_TEST

Typedefs

- typedef uint8_t EmberTaskId
- typedef PGM struct EmberEventData_S EmberEventData
- typedef uint16_t EmberMacFilterMatchData
- typedef uint8_t EmberLibraryStatus

Enumerations

- enum EmberNodeType {
 EMBER_UNKNOWN_DEVICE, EMBER_COORDINATOR, EMBER_ROUTER,
 EMBER_END_DEVICE,
 EMBER_SLEEPY_END_DEVICE, EMBER_MOBILE_END_DEVICE, EMBER_RF4CE_TARGET, EMBER_RF4CE_CONTROLLER
 }
- enum EmberEndDeviceConfiguration { EMBER_END_DEVICE_CONFIG_NONE, EMBER_END_DEVICE_CONFIG_PERSIST_DATA_ON_PARENT }
- enum EmberNetworkInitBitmask { EMBER_NETWORK_INIT_NO_OPTIONS, EMBER_NETWORK_INIT_PARENT_INFO_IN_TOKEN }
- enum EmberApsOption {
 EMBER_APS_OPTION_NONE, EMBER_APS_OPTION_DSA_SIGN, EMBER_APS_OPTION_ENCRYPTION, EMBER_APS_OPTION_RETRY,
 EMBER_APS_OPTION_ENABLE_ROUTE_DISCOVERY, EMBER_APS_OPTION_FORCE_ROUTE_DISCOVERY, EMBER_APS_OPTION_SOURCE_EUI64,
 }

```

EMBER_APS_OPTION_DESTINATION_EUI64,
EMBER_APS_OPTION_ENABLE_ADDRESS_DISCOVERY, EMBER_APS_O-
PTION_POLL_RESPONSE, EMBER_APS_OPTION_ZDO_RESPONSE_REQU-
RED, EMBER_APS_OPTION_FRAGMENT }

• enum EmberIncomingMessageType {
EMBER_INCOMING_UNICAST, EMBER_INCOMING_UNICAST_REPLY, E-
MBER_INCOMING_MULTICAST, EMBER_INCOMING_MULTICAST_LOOP-
BACK,
EMBER_INCOMING_BROADCAST, EMBER_INCOMING_BROADCAST_LO-
OPBACK }

• enum EmberOutgoingMessageType {
EMBER_OUTGOING_DIRECT, EMBER_OUTGOING_VIA_ADDRESS_TABL-
E, EMBER_OUTGOING_VIA_BINDING, EMBER_OUTGOING_MULTICAST,
EMBER_OUTGOING_MULTICAST_WITH_ALIAS, EMBER_OUTGOING_BR-
OADCAST_WITH_ALIAS, EMBER_OUTGOING_BROADCAST }

• enum EmberZigbeeCommandType {
EMBER_ZIGBEE_COMMAND_TYPE_MAC, EMBER_ZIGBEE_COMMAND_-_
TYPE_NWK, EMBER_ZIGBEE_COMMAND_TYPE_APS, EMBER_ZIGBEE_-_
COMMAND_TYPE_ZDO,
EMBER_ZIGBEE_COMMAND_TYPE_ZCL, EMBER_ZIGBEE_COMMAND_T-
YPE_BEACON }

• enum EmberNetworkStatus {
EMBER_NO_NETWORK, EMBER_JOINING_NETWORK, EMBER_JOINED_-_
NETWORK, EMBER_JOINED_NETWORK_NO_PARENT,
EMBER_LEAVING_NETWORK }

• enum EmberNetworkScanType { EMBER_ENERGY_SCAN, EMBER_ACTIVE_-_
SCAN }

• enum EmberBindingType { EMBER_UNUSED_BINDING, EMBER_UNICAST_-_
BINDING, EMBER_MANY_TO_ONE_BINDING, EMBER_MULTICAST_BIN-
DING }

• enum EmberJoinDecision { EMBER_USE_PRECONFIGURED_KEY, EMBER_S-
END_KEY_IN_THE_CLEAR, EMBER_DENY_JOIN, EMBER_NO_ACTION }

• enum EmberDeviceUpdate {
EMBER_STANDARD_SECURITY_SECURED_REJOIN, EMBER_STANDARD-
_SECURITY_UNSECURED_JOIN, EMBER_DEVICE_LEFT, EMBER_STAND-
ARD_SECURITY_UNSECURED_REJOIN,
EMBER_HIGH_SECURITY_SECURED_REJOIN, EMBER_HIGH_SECURITY-
_UNSECURED_JOIN, EMBER_HIGH_SECURITY_UNSECURED_REJOIN }

• enum EmberRejoinReason {
EMBER_REJOIN_REASON_NONE, EMBER_REJOIN_DUE_TO_NWK_KEY_-_
UPDATE, EMBER_REJOIN_DUE_TO_LEAVE_MESSAGE, EMBER_REJOIN-
_DUE_TO_NO_PARENT,
EMBER_REJOIN_DUE_TO_ZLL_TOUCHLINK, EMBER_REJOIN_DUE_TO_-
APP_EVENT_5, EMBER_REJOIN_DUE_TO_APP_EVENT_4, EMBER_REJOI-
N_DUE_TO_APP_EVENT_3,
EMBER_REJOIN_DUE_TO_APP_EVENT_2, EMBER_REJOIN_DUE_TO_APP-
_EVENT_1 }

• enum EmberClusterListId { EMBER_INPUT_CLUSTER_LIST, EMBER_OUTP-
UT_CLUSTER_LIST }

• enum EmberEventUnits {
EMBER_EVENT_INACTIVE, EMBER_EVENT_MS_TIME, EMBER_EVENT_-_
QS_TIME, EMBER_EVENT_MINUTE_TIME,
EMBER_EVENT_ZERO_DELAY }

```

- enum EmberJoinMethod { EMBER_USE_MAC_ASSOCIATION, EMBER_USE_NWK_REJOIN, EMBER_USE_NWK_REJOIN_HAVE_NWK_KEY, EMBER_USE_NWK_COMMISSIONING }
- enum EmberCounterType {
 EMBER_COUNTER_MAC_RX_BROADCAST, EMBER_COUNTER_MAC_TX_BROADCAST, EMBER_COUNTER_MAC_RX_UNICAST, EMBER_COUNTER_MAC_TX_UNICAST_SUCCESS,
 EMBER_COUNTER_MAC_TX_UNICAST_RETRY, EMBER_COUNTER_MAC_TX_UNICAST_FAILED, EMBER_COUNTER_APSS_DATA_RX_BROADCAST, EMBER_COUNTER_APSS_DATA_TX_BROADCAST,
 EMBER_COUNTER_APSS_DATA_RX_UNICAST, EMBER_COUNTER_APSS_DATA_RX_UNICAST_SUCCESS, EMBER_COUNTER_APSS_DATA_TX_UNICAST_RETRY, EMBER_COUNTER_APSS_DATA_TX_UNICAST_FAILED,
 EMBER_COUNTER_ROUTE_DISCOVERY_INITIATED, EMBER_COUNTER_NEIGHBOR_ADDED, EMBER_COUNTER_NEIGHBOR_REMOVED, EMBER_COUNTER_NEIGHBOR_STALE,
 EMBER_COUNTER_JOIN_INDICATION, EMBER_COUNTER_CHILD_REMOVED, EMBER_COUNTER_ASH_OVERFLOW_ERROR, EMBER_COUNTER_ASH_FRAMING_ERROR,
 EMBER_COUNTER_ASH_OVERRUN_ERROR, EMBER_COUNTER_NWK_FRAME_COUNTER_FAILURE, EMBER_COUNTER_APSS_FRAME_COUNTER_FAILURE, EMBER_COUNTER_ASH_XOFF,
 EMBER_COUNTER_APSS_LINK_KEY_NOTAUTHORIZED, EMBER_COUNTER_NWK_DECRYPTION_FAILURE, EMBER_COUNTER_APSS_DECRYPTION_FAILURE, EMBER_COUNTER_ALLOCATE_PACKET_BUFFER_FAILURE,
 EMBER_COUNTER_RELAYED_UNICAST, EMBER_COUNTER_PHY_TO_MAC_QUEUE_LIMIT_REACHED, EMBER_COUNTER_PACKET_VALIDATE_LIBRARY_DROPPED_COUNT, EMBER_COUNTER_TYPE_NWK_RETRY_OVERFLOW,
 EMBER_COUNTER_PHY_CCA_FAIL_COUNT, EMBER_COUNTER_BROADCAST_TABLE_FULL, EMBER_COUNTER_TYPE_COUNT }
- enum EmberInitialSecurityBitmask {
 EMBER_DISTRIBUTED_TRUST_CENTER_MODE, EMBER_TRUST_CENTER_GLOBAL_LINK_KEY, EMBER_PRECONFIGURED_NETWORK_KEY_MODE, EMBER_HAVE_TRUST_CENTER_EUI64,
 EMBER_TRUST_CENTERUSES_HASHEDLINKKEY, EMBER_HAVE_PRECONFIGUREDKEY, EMBER_HAVE_NETWORKKEY, EMBER_GETLINKKEYWHENJOINING,
 EMBER_REQUIRE_ENCRYPTEDKEY, EMBER_NO_FRAME_COUNTER_RESET, EMBER_GET_PRECONFIGUREDKEYFROMINSTALLCODE }
- enum EmberExtendedSecurityBitmask { EMBER_JOINER_GLOBAL_LINK_KEY, EMBER_EXT_NO_FRAME_COUNTER_RESET, EMBER_NWK_LEAVE_REQUEST_NOT_ALLOWED }
- enum EmberCurrentSecurityBitmask {
 EMBER_STANDARD_SECURITY_MODE_, EMBER_DISTRIBUTED_TRUST_CENTER_MODE_, EMBER_TRUST_CENTER_GLOBAL_LINK_KEY_, EMBER_HAVE_TRUST_CENTER_LINK_KEY,
 EMBER_TRUST_CENTERUSES_HASHEDLINKKEY_ }
- enum EmberKeyStructBitmask {
 EMBER_KEY_HAS_SEQUENCE_NUMBER, EMBER_KEY_HAS_OUTGOING_FRAME_COUNTER, EMBER_KEY_HAS_INCOMING_FRAME_COUNTER, EMBER_KEY_HAS_PARTNER_EUI64,
 EMBER_KEY_IS_AUTHORIZED, EMBER_KEY_PARTNER_IS_SLEEPY }

- enum `EmberKeyType` {
 `EMBER_TRUST_CENTER_LINK_KEY, EMBER_TRUST_CENTER_MASTER_KEY, EMBER_CURRENT_NETWORK_KEY, EMBER_NEXT_NETWORK_KEY,`
`EMBER_APPLICATION_LINK_KEY, EMBER_APPLICATION_MASTER_KEY }`
- enum `EmberKeyStatus` {
 `EMBER_KEY_STATUS_NONE, EMBER_APP_LINK_KEY_ESTABLISHED, EMBER_APP_MASTER_KEY_ESTABLISHED, EMBER_TRUST_CENTER_LINK_KEY_ESTABLISHED,`
`EMBER_KEY_ESTABLISHMENT_TIMEOUT, EMBER_KEY_TABLE_FULL, EMBER_TC_RESPONDED_TO_KEY_REQUEST, EMBER_TC_APP_KEY_SEND_TO_REQUESTER,`
`EMBER_TC_RESPONSE_TO_KEY_REQUEST_FAILED, EMBER_TC_REQUEST_KEY_TYPE_NOT_SUPPORTED, EMBER_TC_NO_LINK_KEY_FOR_REQUESTER, EMBER_TC_REQUESTER_EUI64_UNKNOWN,`
`EMBER_TC RECEIVED FIRST APP KEY REQUEST, EMBER_TC TIMEOUT_WAITING FOR SECOND APP KEY REQUEST, EMBER_TC NON_MATCHING_APP_KEY_REQUEST RECEIVED, EMBER_TC FAILED TO SEND_APP KEYS,`
`EMBER_TC FAILED TO STORE_APP KEY REQUEST, EMBER_TC REJECTED_APP KEY REQUEST, EMBER_TC FAILED TO GENERATE_NEW_KEY, EMBER_TC FAILED TO SEND_TC_KEY,`
`EMBER_TRUST_CENTER_IS_PRE_R21, EMBER_TC_REQUESTER_VERIFY_KEY_TIMEOUT, EMBER_TC_REQUESTER_VERIFY_KEY_FAILURE, EMBER_TC_REQUESTER_VERIFY_KEY_SUCCESS,`
`EMBER_VERIFY_LINK_KEY_FAILURE, EMBER_VERIFY_LINK_KEY_SUCCESS }`
- enum `EmberLinkKeyRequestPolicy` { `EMBER_DENY_KEY_REQUESTS, EMBER_ALLOW_KEY_REQUESTS, EMBER_GENERATE_NEW_TC_LINK_KEY` }
- enum `EmberKeySettings` { `EMBER_KEY_PERMISSIONS_NONE, EMBER_KEY_PERMISSIONS_READING_ALLOWED, EMBER_KEY_PERMISSIONS_HASHING_ALLOWED` }
- enum `EmberMacPassthroughType` {
 `EMBER_MAC_PASSTHROUGH_NONE, EMBER_MAC_PASSTHROUGH_SEINTERPAN, EMBER_MAC_PASSTHROUGH_EMBERNET, EMBER_MAC_PASSTHROUGH_EMBERNET_SOURCE,`
`EMBER_MAC_PASSTHROUGH_APPLICATION, EMBER_MAC_PASSTHROUGH_CUSTOM }`

Functions

- `uint8_t * emberKeyContents (EmberKeyData *key)`
- `uint8_t * emberCertificateContents (EmberCertificateData *cert)`
- `uint8_t * emberPublicKeyContents (EmberPublicKeyData *key)`
- `uint8_t * emberPrivateKeyContents (EmberPrivateKeyData *key)`
- `uint8_t * emberSmacContents (EmberSmacData *key)`
- `uint8_t * emberSignatureContents (EmberSignatureData *sig)`
- `uint8_t * emberCertificate283k1Contents (EmberCertificate283k1Data *cert)`
- `uint8_t * emberPublicKey283k1Contents (EmberPublicKey283k1Data *key)`
- `uint8_t * emberPrivateKey283k1Contents (EmberPrivateKey283k1Data *key)`
- `uint8_t * ember283k1SignatureContents (Ember283k1SignatureData *sig)`

Miscellaneous Ember Types

- #define EMBER_RELEASE_TYPE_TO_STRING_STRUCT_DATA
- #define EUI64_SIZE
- #define EXTENDED_PAN_ID_SIZE
- #define EMBER_ENCRYPTION_KEY_SIZE
- #define EMBER_CERTIFICATE_SIZE
- #define EMBER_PUBLIC_KEY_SIZE
- #define EMBER_PRIVATE_KEY_SIZE
- #define EMBER_SMAC_SIZE
- #define EMBER_SIGNATURE_SIZE
- #define EMBER_AES_HASH_BLOCK_SIZE
- #define EMBER_CERTIFICATE_283K1_SIZE
- #define EMBER_PUBLIC_KEY_283K1_SIZE
- #define EMBER_PRIVATE_KEY_283K1_SIZE
- #define EMBER_SIGNATURE_283K1_SIZE
- #define __EMBERSTATUS_TYPE__
- #define EMBER_MAX_802_15_4_CHANNEL_NUMBER
- #define EMBER_MIN_802_15_4_CHANNEL_NUMBER
- #define EMBER_NUM_802_15_4_CHANNELS
- #define EMBER_ALL_802_15_4_CHANNELS_MASK
- #define EMBER_ZIGBEE_COORDINATOR_ADDRESS
- #define EMBER_NULL_NODE_ID
- #define EMBER_NULL_BINDING
- #define EMBER_TABLE_ENTRY_UNUSED_NODE_ID
- #define EMBER_MULTICAST_NODE_ID
- #define EMBER_UNKNOWN_NODE_ID
- #define EMBER_DISCOVERY_ACTIVE_NODE_ID
- #define EMBER_NULL_ADDRESS_TABLE_INDEX
- #define EMBER_ZDO_ENDPOINT
- #define EMBER_BROADCAST_ENDPOINT
- #define EMBER_ZDO_PROFILE_ID
- #define EMBER_WILDCARD_PROFILE_ID
- #define EMBER_MAXIMUM_STANDARD_PROFILE_ID
- #define EMBER_BROADCAST_TABLE_TIMEOUT_QS
- #define EMBER_MANUFACTURER_ID
- enum EmberVersionType {
 EMBER_VERSION_TYPE_PRE_RELEASE, EMBER_VERSION_TYPE_ALPHA_1,
 EMBER_VERSION_TYPE_ALPHA_2, EMBER_VERSION_TYPE_ALPHA_3,
 EMBER_VERSION_TYPE_BETA_1, EMBER_VERSION_TYPE_BETA_2, EMBER_VERSION_TYPE_BETA_3,
 EMBER_VERSION_TYPE_GA
 }
- enum EmberLeaveRequestFlags { EMBER_ZIGBEE_LEAVE_AND_REJOIN, EMBER_ZIGBEE_LEAVE_AND_REMOVE_CHILDREN }
- enum EmberLeaveReason {
 EMBER_LEAVE_REASON_NONE, EMBER_LEAVE_DUE_TO_NWK_LEAVE_MESSAGE,
 EMBER_LEAVE_DUE_TO_APS_REMOVE_MESSAGE, EMBER_LEAVE_DUE_TO_ZDO_LEAVE_MESSAGE,
 EMBER_LEAVE_DUE_TO_ZLL_TOUCHLINK, EMBER_LEAVE_DUE_TO_APP_EVENT_1
 }
- typedef uint8_t EmberStatus

- `typedef uint8_t EmberEUI64 [EUI64_SIZE]`
- `typedef uint8_t EmberMessageBuffer`
- `typedef uint16_t EmberNodeId`
- `typedef uint16_t EmberMulticastId`
- `typedef uint16_t EmberPanId`
- `const EmberVersion emberVersion`

ZigBee Broadcast Addresses

ZigBee specifies three different broadcast addresses that reach different collections of nodes. Broadcasts are normally sent only to routers. Broadcasts can also be forwarded to end devices, either all of them or only those that do not sleep. Broadcasting to end devices is both significantly more resource-intensive and significantly less reliable than broadcasting to routers.

- `#define EMBER_BROADCAST_ADDRESS`
- `#define EMBER_RX_ON_WHEN_IDLE_BROADCAST_ADDRESS`
- `#define EMBER_SLEEPY_BROADCAST_ADDRESS`

Ember Concentrator Types

- `#define EMBER_LOW_RAM_CONCENTRATOR`
- `#define EMBER_HIGH_RAM_CONCENTRATOR`

txPowerModes for emberSetTxPowerMode and mfplibSetPower

- `#define EMBER_TX_POWER_MODE_DEFAULT`
- `#define EMBER_TX_POWER_MODE_BOOST`
- `#define EMBER_TX_POWER_MODE_ALTERNATE`
- `#define EMBER_TX_POWER_MODE_BOOST_AND_ALTERNATE`

Alarm Message and Counters Request Definitions

- `#define EMBER_PRIVATE_PROFILE_ID`
- `#define EMBER_PRIVATE_PROFILE_ID_START`
- `#define EMBER_PRIVATE_PROFILE_ID_END`
- `#define EMBER_BROADCAST_ALARM_CLUSTER`
- `#define EMBER_UNICAST_ALARM_CLUSTER`
- `#define EMBER_CACHED_UNICAST_ALARM_CLUSTER`
- `#define EMBER_REPORT_COUNTERS_REQUEST`
- `#define EMBER_REPORT_COUNTERS_RESPONSE`
- `#define EMBER_REPORT_AND_CLEAR_COUNTERS_REQUEST`
- `#define EMBER_REPORT_AND_CLEAR_COUNTERS_RESPONSE`
- `#define EMBER_OTA_CERTIFICATE_UPGRADE_CLUSTER`

ZDO response status.

Most responses to ZDO commands contain a status byte. The meaning of this byte is defined by the ZigBee Device Profile.

- enum EmberZdoStatus {
 EMBER_ZDP_SUCCESS, EMBER_ZDP_INVALID_REQUEST_TYPE, EMBER_ZDP_DEVICE_NOT_FOUND, EMBER_ZDP_INVALID_ENDPOINT, EMBER_ZDP_NOT_ACTIVE, EMBER_ZDP_NOT_SUPPORTED, EMBER_ZDP_TIMEOUT, EMBER_ZDP_NO_MATCH, EMBER_ZDP_NO_ENTRY, EMBER_ZDP_NO_DESCRIPTOR, EMBER_ZDP_INSUFFICIENT_SPACE, EMBER_ZDP_NOT_PERMITTED, EMBER_ZDP_TABLE_FULL, EMBER_ZDP_NOT_AUTHORIZED, EMBER_NWK_ALREADY_PRESENT, EMBER_NWK_TABLE_FULL, EMBER_NWK_UNKNOWN_DEVICE }

Network and IEEE Address Request/Response

Defines for ZigBee device profile cluster IDs follow. These include descriptions of the formats of the messages.

Note that each message starts with a 1-byte transaction sequence number. This sequence number is used to match a response command frame to the request frame that it is replying to. The application shall maintain a 1-byte counter that is copied into this field and incremented by one for each command sent. When a value of 0xff is reached, the next command shall re-start the counter with a value of 0x00

```
Network request: <transaction sequence number: 1>
                  <EUI64:8>   <type:1> <start index:1>
IEEE request:    <transaction sequence number: 1>
                  <node ID:2> <type:1> <start index:1>
                  <type> = 0x00 single address response, ignore the start index
                  = 0x01 extended response -> sends kid's IDs as well
Response:        <transaction sequence number: 1>
                  <status:1> <EUI64:8> <node ID:2>
                  <ID count:1> <start index:1> <child ID:2>*

```

- #define NETWORK_ADDRESS_REQUEST
- #define NETWORK_ADDRESS_RESPONSE
- #define IEEE_ADDRESS_REQUEST
- #define IEEE_ADDRESS_RESPONSE

Node Descriptor Request/Response

```
<br>

@code
Request:  <transaction sequence number: 1> <node ID:2>
Response: <transaction sequence number: 1> <status:1> <node ID:2>

// <node descriptor: 13> // // Node Descriptor field is divided into subfields of bitmasks
// as follows: // (Note: All lengths below are given in bits rather than bytes.) // Logical Type:
3 // Complex Descriptor Available: 1 // User Descriptor Available: 1 // (reserved/unused):
3 // APS Flags: 3 // Frequency Band: 5 // MAC capability flags: 8 // Manufacturer Code-
: 16 // Maximum buffer size: 8 // Maximum incoming transfer size: 16 // Server mask:
16 // Maximum outgoing transfer size: 16 // Descriptor Capability Flags: 8 // See ZigBee
document 053474, Section 2.3.2.3 for more details.
```

- #define NODE_DESCRIPTOR_REQUEST
- #define NODE_DESCRIPTOR_RESPONSE

Power Descriptor Request / Response

```
<br>

@code
Request: <transaction sequence number: 1> <node ID:2>
Response: <transaction sequence number: 1> <status:1> <node ID:2>
           <current power mode, available power sources:1>
           <current power source, current power source level:1>
```

// See ZigBee document 053474, Section 2.3.2.4 for more details.

- #define POWER_DESCRIPTOR_REQUEST
- #define POWER_DESCRIPTOR_RESPONSE

Simple Descriptor Request / Response

```
Request: <transaction sequence number: 1>
          <node ID:2> <endpoint:1>
Response: <transaction sequence number: 1>
          <status:1> <node ID:2> <length:1> <endpoint:1>
          <app profile ID:2> <app device ID:2>
          <app device version, app flags:1>
          <input cluster count:1> <input cluster:2>*
          <output cluster count:1> <output cluster:2>*
```

- #define SIMPLE_DESCRIPTOR_REQUEST
- #define SIMPLE_DESCRIPTOR_RESPONSE

Active Endpoints Request / Response

```
Request: <transaction sequence number: 1> <node ID:2>
Response: <transaction sequence number: 1>
          <status:1> <node ID:2> <endpoint count:1> <endpoint:1>*
```

- #define ACTIVE_ENDPOINTS_REQUEST
- #define ACTIVE_ENDPOINTS_RESPONSE

Match Descriptors Request / Response

```
Request: <transaction sequence number: 1>
          <node ID:2> <app profile ID:2>
          <input cluster count:1> <input cluster:2>*
          <output cluster count:1> <output cluster:2>*
Response: <transaction sequence number: 1>
          <status:1> <node ID:2> <endpoint count:1> <endpoint:1>*
```

- #define MATCH_DESCRIPTORS_REQUEST
- #define MATCH_DESCRIPTORS_RESPONSE

Discovery Cache Request / Response

```
Request: <transaction sequence number: 1>
         <source node ID:2> <source EUI64:8>
Response: <transaction sequence number: 1>
          <status (== EMBER_ZDP_SUCCESS):1>

• #define DISCOVERY_CACHE_REQUEST
• #define DISCOVERY_CACHE_RESPONSE
```

End Device Announce and End Device Announce Response

```
Request: <transaction sequence number: 1>
         <node ID:2> <EUI64:8> <capabilities:1>
No response is sent.
```

- #define END_DEVICE_ANNOUNCE
- #define END_DEVICE_ANNOUNCE_RESPONSE

System Server Discovery Request / Response

This is broadcast and only servers which have matching services respond. The response contains the request services that the recipient provides.

```
Request: <transaction sequence number: 1> <server mask:2>
Response: <transaction sequence number: 1>
          <status (== EMBER_ZDP_SUCCESS):1> <server mask:2>

• #define SYSTEM_SERVER_DISCOVERY_REQUEST
• #define SYSTEM_SERVER_DISCOVERY_RESPONSE
```

Parent Announce and Parent Announce Response

This is broadcast and only servers which have matching children respond. The response contains the list of children that the recipient now holds.

```
Request: <transaction sequence number: 1>
         <number of children:1> <child EUI64:8> <child Age:4>*
Response: <transaction sequence number: 1>
          <number of children:1> <child EUI64:8> <child Age:4>*

• #define PARENT_ANNOUNCE
• #define PARENT_ANNOUNCE_RESPONSE
```

ZDO server mask bits

These are used in server discovery requests and responses.

- enum EmberZdoServerMask {
 EMBER_ZDP_PRIMARY_TRUST_CENTER, EMBER_ZDP_SECONDARY_TRUST_CENTER, EMBER_ZDP_PRIMARY_BINDING_TABLE_CACHE, EMBER_ZDP_SECONDARY_BINDING_TABLE_CACHE,
 EMBER_ZDP_PRIMARY_DISCOVERY_CACHE, EMBER_ZDP_SECONDARY_DISCOVERY_CACHE, EMBER_ZDP_NETWORK_MANAGER }

Find Node Cache Request / Response

This is broadcast and only discovery servers which have the information for the device of interest, or the device of interest itself, respond. The requesting device can then direct any service discovery requests to the responder.

```
Request: <transaction sequence number: 1>
         <device of interest ID:2> <d-of-i EUI64:8>
Response: <transaction sequence number: 1>
          <responder ID:2> <device of interest ID:2> <d-of-i EUI64:8>

• #define FIND_NODE_CACHE_REQUEST
• #define FIND_NODE_CACHE_RESPONSE
```

End Device Bind Request / Response

```
Request: <transaction sequence number: 1>
         <node ID:2> <EUI64:8> <endpoint:1> <app profile ID:2>
         <input cluster count:1> <input cluster:2>*
         <output cluster count:1> <output cluster:2>*
Response: <transaction sequence number: 1> <status:1>

• #define END_DEVICE_BIND_REQUEST
• #define END_DEVICE_BIND_RESPONSE
```

Binding types and Request / Response

Bind and unbind have the same formats. There are two possible formats, depending on whether the destination is a group address or a device address. Device addresses include an endpoint, groups don't.

```
Request: <transaction sequence number: 1>
         <source EUI64:8> <source endpoint:1>
         <cluster ID:2> <destination address:3 or 10>
Destination address:
         <0x01:1> <destination group:2>
Or:
         <0x03:1> <destination EUI64:8> <destination endpoint:1>
Response: <transaction sequence number: 1> <status:1>

• #define UNICAST_BINDING
• #define UNICAST_MANY_TO_ONE_BINDING
• #define MULTICAST_BINDING
• #define BIND_REQUEST
• #define BIND_RESPONSE
• #define UNBIND_REQUEST
• #define UNBIND_RESPONSE
```

LQI Table Request / Response

```
Request: <transaction sequence number: 1> <start index:1>
Response: <transaction sequence number: 1> <status:1>
          <neighbor table entries:1> <start index:1>
          <entry count:1> <entry:22>*
<entry> = <extended PAN ID:8> <EUI64:8> <node ID:2>
          <device type, rx on when idle, relationship:1>
          <permit joining:1> <depth:1> <LQI:1>
```

The device-type byte has the following fields:

Name	Mask	Values
device type	0x03	0x00 coordinator 0x01 router 0x02 end device 0x03 unknown
rx mode	0x0C	0x00 off when idle 0x04 on when idle 0x08 unknown
relationship	0x70	0x00 parent 0x10 child 0x20 sibling 0x30 other 0x40 previous child
reserved	0x10	

The permit-joining byte has the following fields

Name	Mask	Values
permit joining	0x03	0x00 not accepting join requests 0x01 accepting join requests 0x02 unknown
reserved	0xFC	

- #define LQI_TABLE_REQUEST
- #define LQI_TABLE_RESPONSE

Routing Table Request / Response

```
Request: <transaction sequence number: 1> <start index:1>
Response: <transaction sequence number: 1> <status:1>
           <routing table entries:1> <start index:1>
           <entry count:1> <entry:5>*
           <entry> = <destination address:2>
                     <status:1>
                     <next hop:2>
```

The status byte has the following fields:

Name	Mask	Values
status	0x07	0x00 active 0x01 discovery underway 0x02 discovery failed 0x03 inactive 0x04 validation underway
flags	0x38	0x08 memory constrained 0x10 many-to-one 0x20 route record required
reserved	0xC0	

- #define ROUTING_TABLE_REQUEST
- #define ROUTING_TABLE_RESPONSE

Binding Table Request / Response

```
Request: <transaction sequence number: 1> <start index:1>
```

```
Response: <transaction sequence number: 1>
    <status:1> <binding table entries:1> <start index:1>
    <entry count:1> <entry:14/21>*
<entry> = <source EUI64:8> <source endpoint:1> <cluster ID:2>
    <dest addr mode:1> <dest:2/8> <dest endpoint:0/1>
```

Note

If Dest. Address Mode = 0x03, then the Long Dest. Address will be used and Dest. endpoint will be included. If Dest. Address Mode = 0x01, then the Short Dest. Address will be used and there will be no Dest. endpoint.

- #define BINDING_TABLE_REQUEST
- #define BINDING_TABLE_RESPONSE

Leave Request / Response

```
Request: <transaction sequence number: 1> <EUI64:8> <flags:1>
    The flag bits are:
    0x40 remove children
    0x80 rejoin
Response: <transaction sequence number: 1> <status:1>
```

- #define LEAVE_REQUEST
- #define LEAVE_RESPONSE
- #define LEAVE_REQUEST_REMOVE_CHILDREN_FLAG
- #define LEAVE_REQUEST_REJOIN_FLAG

Permit Joining Request / Response

```
Request: <transaction sequence number: 1>
    <duration:1> <permit authentication:1>
Response: <transaction sequence number: 1> <status:1>
```

- #define PERMIT_JOINING_REQUEST
- #define PERMIT_JOINING_RESPONSE

Network Update Request / Response

```
Request: <transaction sequence number: 1>
    <scan channels:4> <duration:1> <count:0/1> <manager:0/2>

    If the duration is in 0x00 ... 0x05, then 'count' is present but
    not 'manager'. Perform 'count' scans of the given duration on the
    given channels.

    If duration is 0xFE, then 'channels' should have a single channel
    and 'count' and 'manager' are not present. Switch to the indicated
    channel.

    If duration is 0xFF, then 'count' is not present. Set the active
    channels and the network manager ID to the values given.

    Unicast requests always get a response, which is INVALID_REQUEST if the
    duration is not a legal value.
```

```
Response: <transaction sequence number: 1> <status:1>
    <scanned channels:4> <transmissions:2> <failures:2>
    <energy count:1> <energy:1>*
```

- #define NWK_UPDATE_REQUEST
- #define NWK_UPDATE_RESPONSE

Unsupported

Not mandatory and not supported.

- #define COMPLEX_DESCRIPTOR_REQUEST
- #define COMPLEX_DESCRIPTOR_RESPONSE
- #define USER_DESCRIPTOR_REQUEST
- #define USER_DESCRIPTOR_RESPONSE
- #define DISCOVERY_REGISTER_REQUEST
- #define DISCOVERY_REGISTER_RESPONSE
- #define USER_DESCRIPTOR_SET
- #define USER_DESCRIPTOR_CONFIRM
- #define NETWORK_DISCOVERY_REQUEST
- #define NETWORK_DISCOVERY_RESPONSE
- #define DIRECT_JOIN_REQUEST
- #define DIRECT_JOIN_RESPONSE
- #define CLUSTER_ID_RESPONSE_MINIMUM

ZDO configuration flags.

For controlling which ZDO requests are passed to the application. These are normally controlled via the following configuration definitions:

EMBER_APPLICATION RECEIVES_SUPPORTED_ZDO_REQUESTS EMBER_APPLICATION_HANDLES_UNSUPPORTED_ZDO_REQUESTS EMBER_APPLICATION_HANDLES_ENDPOINT_ZDO_REQUESTS EMBER_APPLICATION_HANDLES_BINDING_ZDO_REQUESTS

See ember-configuration.h for more information.

- enum EmberZdoConfigurationFlags { EMBER_APP_RECEIVES_SUPPORTED_ZDO_REQUESTS, EMBER_APP_HANDLES_UNSUPPORTED_ZDO_REQUESTS, EMBER_APP_HANDLES_ENDPOINT_REQUESTS, EMBER_APP_HANDLES_BINDING_REQUESTS }

8.67.1 Detailed Description

Ember data type definitions. See [Ember Common Data Types](#) for details.

Definition in file [ember-types.h](#).

8.68 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 uint8_t EmberVersionType;
00040 enum
00041 #endif
00042 {
00043     EMBER_VERSION_TYPE_PRE_RELEASE = 0x00,
00044
00045     //Alpha, should be used rarely
00046     EMBER_VERSION_TYPE_ALPHA_1      = 0x11,
00047     EMBER_VERSION_TYPE_ALPHA_2      = 0x12,
00048     EMBER_VERSION_TYPE_ALPHA_3      = 0x13,
00049     // Leave space in case we decide to add other types in the future.
00050     EMBER_VERSION_TYPE_BETA_1       = 0x21,
00051     EMBER_VERSION_TYPE_BETA_2       = 0x22,
00052     EMBER_VERSION_TYPE_BETA_3       = 0x23,
00053
00054
00055
00056     // Anything other than 0xAA is considered pre-release
00057     // We may define other types in the future (e.g. beta, alpha)
00058     // We chose an arbitrary number (0xAA) to allow for expansion, but
00059     // to prevent ambiguity in case 0x00 or 0xFF is accidentally retrieved
00060     // as the version type.
00061     EMBER_VERSION_TYPE_GA          = 0xAA,
00062 };
00063
00064 typedef struct {
00065     EmberVersionType typeNum;
00066     PGM_P typeString;
00067 } EmberReleaseTypeStruct;
00068
00069 #define EMBER_RELEASE_TYPE_TO_STRING_STRUCT_DATA \
00070 { {EMBER_VERSION_TYPE_PRE_RELEASE, "Pre-Release"}, \
00071 {EMBER_VERSION_TYPE_ALPHA_1, "Alpha 1"}, \
00072 {EMBER_VERSION_TYPE_ALPHA_2, "Alpha 2"}, \
00073 {EMBER_VERSION_TYPE_ALPHA_3, "Alpha 3"}, \
00074 {EMBER_VERSION_TYPE_BETA_1, "Beta 1"}, \
00075 {EMBER_VERSION_TYPE_BETA_2, "Beta 2"}, \
00076 {EMBER_VERSION_TYPE_BETA_3, "Beta 3"}, \
00077 {EMBER_VERSION_TYPE_GA, "GA"}, \
00078 {0xFF, NULL}, \
00079
00080
00081
00082
00083
00084
00085
00086
00087
00088
00089
00090 typedef struct {
00091     uint16_t build;
00092     uint8_t major;
00093     uint8_t minor;
00094     uint8_t patch;
00095     uint8_t special;
00096     EmberVersionType type;
00097 } EmberVersion;
00098
00099 extern const EmberVersion emberVersion;
00100
00101 #define EUI64_SIZE 8
00102
00103 #define EXTENDED_PAN_ID_SIZE 8
00104
00105 #define EMBER_ENCRYPTION_KEY_SIZE 16
00106
00107 #define EMBER_CERTIFICATE_SIZE 48
00108
00109 #define EMBER_PUBLIC_KEY_SIZE 22
00110
00111 #define EMBER_PRIVATE_KEY_SIZE 21
00112
00113 #define EMBER_SMAC_SIZE 16
00114
00115 #define EMBER_SIGNATURE_SIZE 42
00116
00117 #define EMBER_AES_HASH_BLOCK_SIZE 16
00118
00119 #define EMBER_CERTIFICATE_283K1_SIZE 74
00120
00121 #define EMBER_PUBLIC_KEY_283K1_SIZE 37
00122
00123 #define EMBER_PRIVATE_KEY_283K1_SIZE 36
00124
00125 #define EMBER_SIGNATURE_283K1_SIZE 72
00126
00127
00128
00129
00130
00131
00132
00133
00134
00135
00136
00137
00138
00139
00140
00141
00142
00143
00144
00145
00146
00147
00148
00149
00150
00151
00152
00153
00154
00155
00156
00157
00158
00159
00160
00161
00162
00163
00164
00165
00166
00167
00168
00169
00170
00171
00172
00173
00174
00175
00176
00177
00178
00179
00180
00181
00182
00183
00184
00185
00186
00187
00188
00189
00190
00191
00192
00193
00194
00195
00196
00197
00198
00199
00200
00201
00202
00203
00204
00205
00206
00207
00208
00209
00210
00211
00212
00213
00214
00215
00216
00217
00218
00219
00220
00221
00222
00223
00224
00225
00226
00227
00228
00229
00230
00231
00232
00233
00234
00235
00236
00237
00238
00239
00240
00241
00242
00243
00244
00245
00246
00247
00248
00249
00250
00251
00252
00253
00254
00255
00256
00257
00258
00259
00260
00261
00262
00263
00264
00265
00266
00267
00268
00269
00270
00271
00272
00273
00274
00275
00276
00277
00278
00279
00280
00281
00282
00283
00284
00285
00286
00287
00288
00289
00290
00291
00292
00293
00294
00295
00296
00297
00298
00299
00300
00301
00302
00303
00304
00305
00306
00307
00308
00309
00310
00311
00312
00313
00314
00315
00316
00317
00318
00319
00320
00321
00322
00323
00324
00325
00326
00327
00328
00329
00330
00331
00332
00333
00334
00335
00336
00337
00338
00339
00340
00341
00342
00343
00344
00345
00346
00347
00348
00349
00350
00351
00352
00353
00354
00355
00356
00357
00358
00359
00360
00361
00362
00363
00364
00365
00366
00367
00368
00369
00370
00371
00372
00373
00374
00375
00376
00377
00378
00379
00380
00381
00382
00383
00384
00385
00386
00387
00388
00389
00390
00391
00392
00393
00394
00395
00396
00397
00398
00399
00400
00401
00402
00403
00404
00405
00406
00407
00408
00409
00410
00411
00412
00413
00414
00415
00416
00417
00418
00419
00420
00421
00422
00423
00424
00425
00426
00427
00428
00429
00430
00431
00432
00433
00434
00435
00436
00437
00438
00439
00440
00441
00442
00443
00444
00445
00446
00447
00448
00449
00450
00451
00452
00453
00454
00455
00456
00457
00458
00459
00460
00461
00462
00463
00464
00465
00466
00467
00468
00469
00470
00471
00472
00473
00474
00475
00476
00477
00478
00479
00480
00481
00482
00483
00484
00485
00486
00487
00488
00489
00490
00491
00492
00493
00494
00495
00496
00497
00498
00499
00500
00501
00502
00503
00504
00505
00506
00507
00508
00509
00510
00511
00512
00513
00514
00515
00516
00517
00518
00519
00520
00521
00522
00523
00524
00525
00526
00527
00528
00529
00530
00531
00532
00533
00534
00535
00536
00537
00538
00539
00540
00541
00542
00543
00544
00545
00546
00547
00548
00549
00550
00551
00552
00553
00554
00555
00556
00557
00558
00559
00560
00561
00562
00563
00564
00565
00566
00567
00568
00569
00570
00571
00572
00573
00574
00575
00576
00577
00578
00579
00580
00581
00582
00583
00584
00585
00586
00587
00588
00589
00590
00591
00592
00593
00594
00595
00596
00597
00598
00599
00600
00601
00602
00603
00604
00605
00606
00607
00608
00609
00610
00611
00612
00613
00614
00615
00616
00617
00618
00619
00620
00621
00622
00623
00624
00625
00626
00627
00628
00629
00630
00631
00632
00633
00634
00635
00636
00637
00638
00639
00640
00641
00642
00643
00644
00645
00646
00647
00648
00649
00650
00651
00652
00653
00654
00655
00656
00657
00658
00659
00660
00661
00662
00663
00664
00665
00666
00667
00668
00669
00670
00671
00672
00673
00674
00675
00676
00677
00678
00679
00680
00681
00682
00683
00684
00685
00686
00687
00688
00689
00690
00691
00692
00693
00694
00695
00696
00697
00698
00699
00700
00701
00702
00703
00704
00705
00706
00707
00708
00709
00710
00711
00712
00713
00714
00715
00716
00717
00718
00719
00720
00721
00722
00723
00724
00725
00726
00727
00728
00729
00730
00731
00732
00733
00734
00735
00736
00737
00738
00739
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749
00750
00751
00752
00753
00754
00755
00756
00757
00758
00759
00760
00761
00762
00763
00764
00765
00766
00767
00768
00769
00770
00771
00772
00773
00774
00775
00776
00777
00778
00779
00780
00781
00782
00783
00784
00785
00786
00787
00788
00789
00790
00791
00792
00793
00794
00795
00796
00797
00798
00799
00800
00801
00802
00803
00804
00805
00806
00807
00808
00809
00810
00811
00812
00813
00814
00815
00816
00817
00818
00819
00820
00821
00822
00823
00824
00825
00826
00827
00828
00829
00830
00831
00832
00833
00834
00835
00836
00837
00838
00839
00840
00841
00842
00843
00844
00845
00846
00847
00848
00849
00850
00851
00852
00853
00854
00855
00856
00857
00858
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00879
00880
00881
00882
00883
00884
00885
00886
00887
00888
00889
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999
00999
01000
01001
01002
01003
01004
01005
01006
01007
01008
01009
010010
010011
010012
010013
010014
010015
010016
010017
010018
010019
010020
010021
010022
010023
010024
010025
010026
010027
010028
010029
010030
010031
010032
010033
010034
010035
010036
010037
010038
010039
010040
010041
010042
010043
010044
010045
010046
010047
010048
010049
010050
010051
010052
010053
010054
010055
010056
010057
010058
010059
010060
010061
010062
010063
010064
010065
010066
010067
010068
010069
010070
010071
010072
010073
010074
010075
010076
010077
010078
010079
010080
010081
010082
010083
010084
010085
010086
010087
010088
010089
010090
010091
010092
010093
010094
010095
010096
010097
010098
010099
0100100
0100101
0100102
0100103
0100104
0100105
0100106
0100107
0100108
0100109
0100110
0100111
0100112
0100113
0100114
0100115
0100116
0100117
0100118
0100119
0100120
0100121
0100122
0100123
0100124
0100125
0100126
0100127
0100128
0100129
0100130
0100131
0100132
0100133
0100134
0100135
0100136
0100137
0100138
0100139
0100140
0100141
0100142
0100143
0100144
0100145
0100146
0100147
0100148
0100149
0100150
0100151
0100152
0100153
0100154
0100155
0100156
0100157
0100158
0100159
0100160
0100161
0100162
0100163
0100164
0100165
0100166
0100167
0100168
0100169
0100170
0100171
0100172
0100173
0100174
0100175
0100176
0100177
0100178
0100179
0100180
0100181
0100182
0100183
0100184
0100185
0100186
0100187
0100188
0100189
0100190
0100191
0100192
0100193
0100194
0100195
0100196
0100197
0100198
0100199
0100200
0100201
0100202
0100203
0100204
0100205
0100206
0100207
0100208
0100209
0100210
0100211
0100212
0100213
0100214
0100215
0100216
0100217
0100218
0100219
0100220
0100221
0100222
0100223
0100224
0100225
0100226
0100227
0100228
0100229
0100230
0100231
0100232
0100233
0100234
0100235
0100236
0100237
0100238
0100239
0100240
0100241
0100242
0100243
0100244
0100245
0100246
0100247
0100248
0100249
0100250
0100251
0100252
0100253
0100254
0100255
0100256
0100257
0100258
0100259
0100260
0100261
0100262
0100263
0100264
0100265
0100266
0100267
0100268
0100269
0100270
0100271
0100272
0100273
0100274
0100275
0100276
0100277
0100278
0100279
0100280
0100281
0100282
0100283
0100284
0100285
0100286
0100287
0100288
0100289
0100290
0100291
0100292
0100293
0100294
0100295
0100296
0100297
0100298
0100299
0100300
0100301
0100302
0100303
0100304
0100305
0100306
0100307
0100308
0100309
0100310
0100311
0100312
0100313
0100314
0100315
0100316
0100317
0100318
0100319
0100320
0100321
0100322
0100323
0100324
0100325
0100326
0100327
0100328
0100329
0100330
0100331
0100332
0100333
0100334
0100335
0100336
0100337
0100338
0100339
0100340
0100341
0100342
0100343
0100344
0100345
0100346
0100347
0100348
0100349
0100350
0100351
0100352
0100353
0100354
0100355
0100356
0100357
0100358
0100359
0100360
0100361
0100362
0100363
0100364
0100365
0100366
0100367
0100368
0100369
0100370
0100371
0100372
0100373
0100374
0100375
0100376
0100377
0100378
0100379
0100380
0100381
0100382
0100383
0100384
0100385
0100386
0100387
0100388
0100389
0100390
0100391
0100392
0100393
0100394
0100395
0100396
0100397
0100398
0100399
0100400
0100401
0100402
0100403
0100404
0100405
0100406
0100407
0100408
0100409
0100410
0100411
0100412
0100413
0100414
0100415
0100416
0100417
0100418
0100419
0100420
0100421
0100422
0100423
0100424
0100425
0100426
0100427
0100428
0100429
0100430
0100431
0100432
0100433
0100434
0100435
0100436
0100437
0100438
0100439
0100440
0100441
0100442
0100443
0100444
0100445
0100446
0100447
0100448
0100449
0100450
0100451
0100452
0100453
0100454
0100455
0100456
0100457
0100458
0100459
0100460
0100461
0100462
0100463
0100464
0100465
0100466
0100467
0100468
0100469
0100470
0100471
0100472
0100473
0100474
0100475
0100476
0100477
0100478
0100479
0100480
0100481
0100482
0100483
0100484
0100485
0100486
0100487
0100488
0100489
0100490
0100491
0100492
0100493
0100494
0100495
0100496
0100497
0100498
0100499
0100500
0100501
0100502
0100503
0100504
0100505
0100506
0100507
0100508
0100509
0100510
0100511
0100512
0100513
0100514
0100515
0100516
0100517
0100518
0100519
0100520
0100521
0100522
0100523
0100524
0100525
0100526
0100527
0100528
0100529
0100530
0100531
0100532
0100533
0100534
0100535
0100536
0100537
0100538
0100539
0100540
0100541
0100542
0100543
0100544
0100545
0100546
0100547
0100548
0100549
0100550
0100551
0100552
0100553
0100554
0100555
0100556
0100557
0100558
0100559
0100560
0100561
0100562
0100563
0100564
0100565
0100566
0100567
0100568
0100569
0100570
0100571
0100572
0100573
0100574
0100575
0100576
0100577
0100578
0100579
0100580
0100581
0100582
0100583
0100584
0100585
0100586
0100587
0100588
0100589
0100590
0100591
0100592
0100593
0100594
0100595
0100596
0100597
0100598
0100599
0100600
0100601
0100602
0100603
0100604
0100605
0100606
0100607
0100608
0100609
0100610
0100611
0100612
0100613
0100614
0100615
0100616
0100617
0100618
0100619
0100620
0100621
0100622
0100623
0100624
0100625
0100626
0100627
0100628
0100629
0100630
0100631
0100632
0100633
0100634
0100635
010063
```

```

00172
00176 #ifndef __EMBERSTATUS_TYPE__
00177 #define __EMBERSTATUS_TYPE__
00178     typedef uint8_t EmberStatus;
00179 #endif //__EMBERSTATUS_TYPE__
00180
00181 #include "stack/include/error.h"
00182
00186 typedef uint8_t EmberEUI64[EUI64_SIZE];
00187
00197 typedef uint8_t EmberMessageBuffer;
00198
00202 typedef uint16_t EmberNodeId;
00203
00205 typedef uint16_t EmberMulticastId;
00206
00210 typedef uint16_t EmberPanId;
00211
00215 #define EMBER_MAX_802_15_4_CHANNEL_NUMBER 26
00216
00220 #define EMBER_MIN_802_15_4_CHANNEL_NUMBER 11
00221
00225 #define EMBER_NUM_802_15_4_CHANNELS \
00226     (EMBER_MAX_802_15_4_CHANNEL_NUMBER - EMBER_MIN_802_15_4_CHANNEL_NUMBER + 1)
00227
00231 #define EMBER_ALL_802_15_4_CHANNELS_MASK 0x07FFF800UL
00232
00236 #define EMBER_ZIGBEE_COORDINATOR_ADDRESS 0x0000
00237
00242 #define EMBER_NULL_NODE_ID 0xFFFF
00243
00248 #define EMBER_NULL_BINDING 0xFF
00249
00259 #define EMBER_TABLE_ENTRY_UNUSED_NODE_ID 0xFFFF
00260
00267 #define EMBER_MULTICAST_NODE_ID          0xFFFFE
00268
00276 #define EMBER_UNKNOWN_NODE_ID           0xFFFFD
00277
00285 #define EMBER_DISCOVERY_ACTIVE_NODE_ID 0xFFFFC
00286
00291 #define EMBER_NULL_ADDRESS_TABLE_INDEX 0xFF
00292
00296 #define EMBER_ZDO_ENDPOINT 0
00297
00301 #define EMBER_BROADCAST_ENDPOINT 0xFF
00302
00306 #define EMBER_ZDO_PROFILE_ID 0x0000
00307
00311 #define EMBER_WILDCARD_PROFILE_ID 0xFFFF
00312
00316 #define EMBER_MAXIMUM_STANDARD_PROFILE_ID 0x7FFF
00317
00323 #define EMBER_BROADCAST_TABLE_TIMEOUT_QS (20 * 4)
00324
00325
00329 #define EMBER_MANUFACTURER_ID 0x1002
00330
00331
00332 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00333 enum EmberLeaveRequestFlags
00334 #else
00335 typedef uint8_t EmberLeaveRequestFlags;
00336 enum
00337 #endif
00338 {
00340     EMBER_ZIGBEE_LEAVE_AND_REJOIN      = 0x80,
00341
00343     EMBER_ZIGBEE_LEAVE_AND_REMOVE_CHILDREN
00344     = 0x40,
00345
00346 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00347 enum EmberLeaveReason
00348 #else
00349 typedef uint8_t EmberLeaveReason;
00350 enum
00351 #endif
00352 {
00353     EMBER_LEAVE_REASON_NONE           = 0,

```

```

00354     EMBER_LEAVE_DUE_TO_NWK_LEAVE_MESSAGE = 1
00355     EMBER_LEAVE_DUE_TOAPS_REMOVE_MESSAGE =
00356     2,
00357     // Currently, the stack does not process the ZDO leave message since it is
00358     optional
00359     EMBER_LEAVE_DUE_TO_ZDO_LEAVE_MESSAGE = 3
00360     ,
00361     EMBER_LEAVE_DUE_TO_ZLL_TOUCHLINK = 4,
00362     00364
00365     00365
00366     #define EMBER_BROADCAST_ADDRESS 0xFFFFC
00367
00368     #define EMBER_RX_ON_WHEN_IDLE_BROADCAST_ADDRESS 0xFFFFD
00369
00370     #define EMBER_SLEEPY_BROADCAST_ADDRESS 0xFFFFF
00371
00372     // From table 3.51 of 053474r14
00373     #define EMBER_MIN_BROADCAST_ADDRESS 0xFFFF8
00374
00375     #define emberIsZigbeeBroadcastAddress(address) \
00376     (EMBER_MIN_BROADCAST_ADDRESS <= ((uint16_t) (address)))
00377
00378
00379
00380
00381
00382
00383
00384
00385
00386
00387
00388
00389
00390
00391
00392
00393
00394
00395
00396
00397
00398
00399
00400
00401
00402
00403
00404
00405
00406
00407
00408
00409
00410
00411
00412
00413
00414
00415
00416
00417
00418
00419
00420
00421
00422
00423
00424
00425
00426
00427
00428
00429
00430
00431
00432
00433
00434
00435
00436
00437
00438
00439
00440
00441
00442
00443
00444
00445
00446
00447
00448
00449
00450
00451
00452
00453
00454
00455
00456
00457
00458
00459
00460
00461
00462
00463
00464
00465
00466
00467

```

```

00468
00469
00474 typedef struct {
00475     EmberNetworkInitBitmask bitmask;
00476 } EmberNetworkInitStruct;
00477
00478
00485 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00486 enum EmberApsOption
00487 #else
00488 typedef uint16_t EmberApsOption;
00489 enum
00490 #endif
00491 {
00493     EMBER_APS_OPTION_NONE             = 0x0000,
00494
00495 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00496     EMBER_APS_OPTION_ENCRYPT_WITH_TRANSIENT_KEY = 0x0001,
00497     EMBER_APS_OPTION_USE_ALIAS_SEQUENCE_NUMBER = 0x0002,
00498 #endif
00499
00511     EMBER_APS_OPTION_DSA_SIGN          = 0x0010,
00514     EMBER_APS_OPTION_ENCRYPTION        = 0x0020
00518     EMBER_APS_OPTION_RETRY            = 0x0040,
00524     EMBER_APS_OPTION_ENABLE_ROUTE_DISCOVERY
00525         = 0x0100,
00527     EMBER_APS_OPTION_FORCE_ROUTE_DISCOVERY
00528         = 0x0200,
00529     EMBER_APS_OPTION_SOURCE_EUI64      =
0x0400,
00531     EMBER_APS_OPTION_DESTINATION_EUI64 =
0x0800,
00534     EMBER_APS_OPTION_ENABLE_ADDRESS_DISCOVERY
00535         = 0x1000,
00539     EMBER_APS_OPTION_POLL_RESPONSE    =
0x2000,
00544     EMBER_APS_OPTION_ZDO_RESPONSE_REQUIRED
00545         = 0x4000,
00550     EMBER_APS_OPTION_FRAGMENT          =
SIGNED_ENUM 0x8000
00551 };
00552
00553
00554
00558 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00559 enum EmberIncomingMessageType
00560 #else
00561 typedef uint8_t EmberIncomingMessageType;
00562 enum
00563 #endif
00564 {
00566     EMBER_INCOMING_UNICAST,
00568     EMBER_INCOMING_UNICAST_REPLY,
00570     EMBER_INCOMING_MULTICAST,
00572     EMBER_INCOMING_MULTICAST_LOOPBACK,
00574     EMBER_INCOMING_BROADCAST,
00576     EMBER_INCOMING_BROADCAST_LOOPBACK
00577 };
00578
00579
00583 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00584 enum EmberOutgoingMessageType
00585 #else
00586 typedef uint8_t EmberOutgoingMessageType;
00587 enum
00588 #endif
00589 {
00591     EMBER_OUTGOING_DIRECT,
00593     EMBER_OUTGOING_VIA_ADDRESS_TABLE,
00595     EMBER_OUTGOING_VIA_BINDING,
00598     EMBER_OUTGOING_MULTICAST,
00601     EMBER_OUTGOING_MULTICAST_WITH_ALIAS,
00604     EMBER_OUTGOING_BROADCAST_WITH_ALIAS,
00607     EMBER_OUTGOING_BROADCAST
00608 };
00609
00615 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00616 enum EmberZigbeeCommandType
00617 #else

```

```

00618 typedef uint8_t EmberZigbeeCommandType;
00619 enum
00620 #endif
00621 {
00623     EMBER_ZIGBEE_COMMAND_TYPE_MAC,
00625     EMBER_ZIGBEE_COMMAND_TYPE_NWK,
00627     EMBER_ZIGBEE_COMMAND_TYPE_APS,
00629     EMBER_ZIGBEE_COMMAND_TYPE_ZDO,
00631     EMBER_ZIGBEE_COMMAND_TYPE_ZCL,
00632
00634     EMBER_ZIGBEE_COMMAND_TYPE_BEACON,
00635 };
00636
00640 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00641 enum EmberNetworkStatus
00642 #else
00643 typedef uint8_t EmberNetworkStatus;
00644 enum
00645 #endif
00646 {
00648     EMBER_NO_NETWORK,
00650     EMBER_JOINING_NETWORK,
00652     EMBER_JOINED_NETWORK,
00655     EMBER_JOINED_NETWORK_NO_PARENT,
00657     EMBER_LEAVING_NETWORK
00658 };
00659
00660
00664 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00665 enum EmberNetworkScanType
00666 #else
00667 typedef uint8_t EmberNetworkScanType;
00668 enum
00669 #endif
00670 {
00672     EMBER_ENERGY_SCAN,
00674     EMBER_ACTIVE_SCAN
00675 };
00676
00677
00681 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00682 enum EmberBindingType
00683 #else
00684 typedef uint8_t EmberBindingType;
00685 enum
00686 #endif
00687 {
00689     EMBER_UNUSED_BINDING      = 0,
00691     EMBER_UNICAST_BINDING    = 1,
00695     EMBER_MANY_TO_ONE_BINDING = 2,
00699     EMBER_MULTICAST_BINDING  = 3,
00700 };
00701
00702
00711 #define EMBER_LOW_RAM_CONCENTRATOR 0xFFFF8
00712
00716 #define EMBER_HIGH_RAM_CONCENTRATOR 0xFFFF9
00717
00719
00720
00724 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00725 enum EmberJoinDecision
00726 #else
00727 typedef uint8_t EmberJoinDecision;
00728 enum
00729 #endif
00730 {
00732     EMBER_USE_PRECONFIGURED_KEY = 0,
00734     EMBER_SEND_KEY_IN_THE_CLEAR,
00736     EMBER_DENY_JOIN,
00738     EMBER_NO_ACTION
00739 };
00740
00744 #define EMBER_JOIN_DECISION_STRINGS \
00745     "use preconfigured key",           \
00746     "send key in the clear",          \
00747     "deny join",                   \
00748     "no action",
00749
00750

```

```

00756 // These map to the actual values within the APS Command frame so they cannot
00757 // be arbitrarily changed.
00758 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00759 enum EmberDeviceUpdate
00760 #else
00761 typedef uint8_t EmberDeviceUpdate;
00762 enum
00763#endif
00764 {
00765     EMBER_STANDARD_SECURITY_SECURED_REJOIN
00766     = 0,
00767     EMBER_STANDARD_SECURITY_UNSECURED_JOIN
00768     = 1,
00769     EMBER_DEVICE_LEFT
00770     = 2,
00771     EMBER_STANDARD_SECURITY_UNSECURED_REJOIN
00772     = 3,
00773     EMBER_HIGH_SECURITY_SECURED_REJOIN
00774     =
00775     4,
00776     EMBER_HIGH_SECURITY_UNSECURED_JOIN
00777     =
00778     5,
00779     /* 6 Reserved */
00780     EMBER_HIGH_SECURITY_UNSECURED_REJOIN
00781     = 7,
00782     /* 8 - 15 Reserved */
00783 };
00784
00785 #define EMBER_DEVICE_UPDATE_STRINGS
00786     "secured rejoin",
00787     "UNsecured join",
00788     "device left",
00789     "UNsecured rejoin",
00790     "high secured rejoin",
00791     "high UNsecured join",
00792     "RESERVED",
00793     /* reserved status code, per the spec. */
00794     "high UNsecured rejoin",
00795
00796 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00797 enum EmberRejoinReason
00798 #else
00799 typedef uint8_t EmberRejoinReason;
00800 enum
00801#endif
00802 {
00803     EMBER_REJOIN_REASON_NONE
00804     = 0,
00805     EMBER_REJOIN_DUE_TO_NWK_KEY_UPDATE
00806     = 1,
00807     EMBER_REJOIN_DUE_TO_LEAVE_MESSAGE
00808     = 2,
00809     EMBER_REJOIN_DUE_TO_NO_PARENT
00810     = 3,
00811     EMBER_REJOIN_DUE_TO_ZLL_TOUCHLINK
00812     = 4,
00813
00814     // App. Framework events
00815     // 0xA0 - 0xE0
00816
00817     // Customer Defined Events
00818     // I numbered these backwards in case there is ever request
00819     // for more application events. We can expand them
00820     // without renumbering the previous ones.
00821     EMBER_REJOIN_DUE_TO_APP_EVENT_5
00822     = 0xFB,
00823     EMBER_REJOIN_DUE_TO_APP_EVENT_4
00824     = 0xFC,
00825     EMBER_REJOIN_DUE_TO_APP_EVENT_3
00826     = 0xFD,
00827     EMBER_REJOIN_DUE_TO_APP_EVENT_2
00828     = 0xFE,
00829     EMBER_REJOIN_DUE_TO_APP_EVENT_1
00830     = 0xFF,
00831 };
00832
00833 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00834 enum EmberClusterListId
00835 #else
00836 typedef uint8_t EmberClusterListId;
00837 enum
00838#endif
00839 {
00840     EMBER_INPUT_CLUSTER_LIST
00841     = 0,
00842     EMBER_OUTPUT_CLUSTER_LIST
00843     = 1
00844 };
00845
00846 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00847 enum EmberEventUnits
00848 #else
00849 typedef uint8_t EmberEventUnits;
00850 enum
00851
00852
00853
00854
00855
00856
00857
00858
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00879
00880
00881
00882
00883
00884
00885
00886
00887
00888
00889
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999
01000
01001
01002
01003
01004
01005
01006
01007
01008
01009
01010
01011
01012
01013
01014
01015
01016
01017
01018
01019
01020
01021
01022
01023
01024
01025
01026
01027
01028
01029
01030
01031
01032
01033
01034
01035
01036
01037
01038
01039
01040
01041
01042
01043
01044
01045
01046
01047
01048
01049
01050
01051
01052
01053
01054
01055
01056
01057
01058
01059
01060
01061
01062
01063
01064
01065
01066
01067
01068
01069
01070
01071
01072
01073
01074
01075
01076
01077
01078
01079
01080
01081
01082
01083
01084
01085
01086
01087
01088
01089
01090
01091
01092
01093
01094
01095
01096
01097
01098
01099
01100
01101
01102
01103
01104
01105
01106
01107
01108
01109
01110
01111
01112
01113
01114
01115
01116
01117
01118
01119
01120
01121
01122
01123
01124
01125
01126
01127
01128
01129
01130
01131
01132
01133
01134
01135
01136
01137
01138
01139
01140
01141
01142
01143
01144
01145
01146
01147
01148
01149
01150
01151
01152
01153
01154
01155
01156
01157
01158
01159
01160
01161
01162
01163
01164
01165
01166
01167
01168
01169
01170
01171
01172
01173
01174
01175
01176
01177
01178
01179
01180
01181
01182
01183
01184
01185
01186
01187
01188
01189
01190
01191
01192
01193
01194
01195
01196
01197
01198
01199
01200
01201
01202
01203
01204
01205
01206
01207
01208
01209
01210
01211
01212
01213
01214
01215
01216
01217
01218
01219
01220
01221
01222
01223
01224
01225
01226
01227
01228
01229
01230
01231
01232
01233
01234
01235
01236
01237
01238
01239
01240
01241
01242
01243
01244
01245
01246
01247
01248
01249
01250
01251
01252
01253
01254
01255
01256
01257
01258
01259
01260
01261
01262
01263
01264
01265
01266
01267
01268
01269
01270
01271
01272
01273
01274
01275
01276
01277
01278
01279
01280
01281
01282
01283
01284
01285
01286
01287
01288
01289
01290
01291
01292
01293
01294
01295
01296
01297
01298
01299
01300
01301
01302
01303
01304
01305
01306
01307
01308
01309
01310
01311
01312
01313
01314
01315
01316
01317
01318
01319
01320
01321
01322
01323
01324
01325
01326
01327
01328
01329
01330
01331
01332
01333
01334
01335
01336
01337
01338
01339
01340
01341
01342
01343
01344
01345
01346
01347
01348
01349
01350
01351
01352
01353
01354
01355
01356
01357
01358
01359
01360
01361
01362
01363
01364
01365
01366
01367
01368
01369
01370
01371
01372
01373
01374
01375
01376
01377
01378
01379
01380
01381
01382
01383
01384
01385
01386
01387
01388
01389
01390
01391
01392
01393
01394
01395
01396
01397
01398
01399
01400
01401
01402
01403
01404
01405
01406
01407
01408
01409
01410
01411
01412
01413
01414
01415
01416
01417
01418
01419
01420
01421
01422
01423
01424
01425
01426
01427
01428
01429
01430
01431
01432
01433
01434
01435
01436
01437
01438
01439
01440
01441
01442
01443
01444
01445
01446
01447
01448
01449
01450
01451
01452
01453
01454
01455
01456
01457
01458
01459
01460
01461
01462
01463
01464
01465
01466
01467
01468
01469
01470
01471
01472
01473
01474
01475
01476
01477
01478
01479
01480
01481
01482
01483
01484
01485
01486
01487
01488
01489
01490
01491
01492
01493
01494
01495
01496
01497
01498
01499
01500
01501
01502
01503
01504
01505
01506
01507
01508
01509
01510
01511
01512
01513
01514
01515
01516
01517
01518
01519
01520
01521
01522
01523
01524
01525
01526
01527
01528
01529
01530
01531
01532
01533
01534
01535
01536
01537
01538
01539
01540
01541
01542
01543
01544
01545
01546
01547
01548
01549
01550
01551
01552
01553
01554
01555
01556
01557
01558
01559
01560
01561
01562
01563
01564
01565
01566
01567
01568
01569
01570
01571
01572
01573
01574
01575
01576
01577
01578
01579
01580
01581
01582
01583
01584
01585
01586
01587
01588
01589
01590
01591
01592
01593
01594
01595
01596
01597
01598
01599
01600
01601
01602
01603
01604
01605
01606
01607
01608
01609
01610
01611
01612
01613
01614
01615
01616
01617
01618
01619
01620
01621
01622
01623
01624
01625
01626
01627
01628
01629
01630
01631
01632
01633
01634
01635
01636
01637
01638
01639
01640
01641
01642
01643
01644
01645
01646
01647
01648
01649
01650
01651
01652
01653
01654
01655
01656
01657
01658
01659
01660
01661
01662
01663
01664
01665
01666
01667
01668
01669
01670
01671
01672
01673
01674
01675
01676
01677
01678
01679
01680
01681
01682
01683
01684
01685
01686
01687
01688
01689
01690
01691
01692
01693
01694
01695
01696
01697
01698
01699
01700
01701
01702
01703
01704
01705
01706
01707
01708
01709
01710
01711
01712
01713
01714
01715
01716
01717
01718
01719
01720
01721
01722
01723
01724
01725
01726
01727
01728
01729
01729
01730
01731
01732
01733
01734
01735
01736
01737
01738
01739
01739
01740
01741
01742
01743
01744
01745
01746
01747
01748
01749
01749
01750
01751
01752
01753
01754
01755
01756
01757
01758
01759
01759
01760
01761
01762
01763
01764
01765
01766
01767
01768
01769
01769
01770
01771
01772
01773
01774
01775
01776
01777
01778
01779
01779
01780
01781
01782
01783
01784
01785
01786
01787
01788
01789
01789
01790
01791
01792
01793
01794
01795
01796
01797
01798
01799
01799
01800
01801
01802
01803
01804
01805
01806
01807
01808
01809
01809
01810
01811
01812
01813
01814
01815
01816
01817
01818
01819
01819
01820
01821
01822
01823
01824
01825
01826
01827
01828
01829
01829
01830
01831
01832
01833
01834
01835
01836
01837
01838
01839
01839
01840
01841
01842
01843
01844
01845
01846
01847
01848
01849
01849
01850
01851
01852
01853
01854
01855
01856
01857
01858
01859
01859
01860
01861
01862
01863
01864
01865
01866
01867
01868
01869
01869
01870
01871
01872
01873
01874
01875
01876
01877
01878
01879
01879
01880
01881
01882
01883
01884
01885
01886
01887
01888
01889
01889
01890
01891
01892
01893
01894
01895
01896
01897
01898
01899
01899
01900
01901
01902
01903
01904
01905
01906
01907
01908
01909
01909
01910
01911
01912
01913
01914
01915
01916
01917
01918
01919
01919
01920
01921
01922
01923
01924
01925
01926
01927
01928
01929
01929
01930
01931
01932
01933
01934
01935
01936
01937
01938
01939
01939
01940
01941
01942
01943
01944
01945
01946
01947
01948
01949
01949
01950
01951
01952
01953
01954
01955
01956
01957
01958
01959
01959
01960
01961
01962
01963
01964
01965
01966
01967
01968
01969
01969
01970
01971
01972
01973
01974
01975
01976
01977
01978
01979
01979
01980
01981
01982
01983
01984
01985
01986
01987
01988
01989
01989
01990
01991
01992
01993
01994
01995
01996
01997
01998
01999
01999
02000
02001
02002
02003
02004
02005
02006
02007
02008
02009
02009
02010
02011
02012
02013
02014
02015
02016
02017
02018
02019
02019
02020
02021
02022
02023
02024
02025
02026
02027
02028
02029
02029
02030
02031
02032
02033
02034
02035
02036
02037
02038
02039
02039
02040
02041
02042
02043
02044
02045
02046
02047
02048
02049
02049
02050
02051
02052
02053
02054
02055
02056
02057
02058
02059
02059
02060
02061
02062
02063
02064
02065
02066
02067
02068
02069
02069
02070
02071
02072
02073
02074
02075
02076
02077
02078
02079
02079
02080
02081
02082
02083
02084
02085
02086
02087
02088
02089
02089
02090
02091
02092
02093
02094
02095
02096
02097
02098
02099
02099
02100
02101
02102
02103
02104
02105
02106
02107
02108
02109
02109
02110
02111
02112
02113
02114
02115
02116
02117
02118
02119
02119
02120
02121
02122
02123
02124
02125
02126
02127
02128
02129
02129
02130
02131
02132
02133
02134
02135
02136
02137
02138
02139
02139
02140
02141
02142
02143
02144
02145
02146
02147
02148
02149
02149
02150
02151
02152
02153
02154
02155
02156
02157
02158
02159
02159
02160
02161
02162
02163
02164
02165
02166
02167
02168
02169
02169
02170
02171
02172
02173
02174
02175
02176
02177
02178
02179
02179
02180
02181
02182
02183
02184
02185
02186
02187
02188
02189
02189
02190
02191
02192
02193
02194
02195
02196
02197
02198
02199
02199
02200
02201
02202
02203
02204
02205
02206
02207
02208
02209
02209
02210
02211
02212
02213
02214
02215
02216
02217
02218
02219
02219
02220
02221
02222
02223
02224
02225
02226
02227
02228
02229
02229
02230
02231
02232
02233
02234
02235
02236
02237
02238
02239
02239
02240
02241
02242
02243
02244
02245
02246
02247
02248
02249
02249
02250
02251
02252
02253
02254
02255
02256
02257
02258
02259
02259
02260
02261
02262
02263
02264
02265
02266
02267
02268
02269
02269
02270
02271
02272
02273
02274
02275
02276
02277
02278
02279
02279
02280
02281
02282
02283
02284
02285
02286
02287
02288
02289
02289
02290
02291
02292
02293
02294
02295
02296
02297
02298
02299
02299
02300
02301
02302
02303
02304
02305
02306
02307
02308
02309
02309
02310
02311
02312
02313
02314
02315
02316
02317
02318
02319
02319
02320
02321
02322
02323
02324
02325
02326
02327
02328
02329
02329
02330
02331
02332
02333
02334
02335
02336
02337
02338
02339
02339
02340
02341
02342
02343
02344
02345
02346
02347
02348
02349
02349
02350
02351
02352
02353
02354
02355
02356
02357
02358
02359
02359
02360
02361
02362
02363
02364
02365
02366
02367
02368
02369
02369
02370
02371
02372
02373
02374
02375
02376
02377
02378
02379
02379
02380
02381
02382
02383
02384
02385
02386
02387
02388
02389
02389
02390
02391
02392
02393
02394
02395
02396
02397
02398
02399
02399
02400
02401
02402
02403
02404
02405
02406
02407
02408
02409
02409
02410
02411
02412
02413
02414
02415
02416
02417
02418
02419
02419
02420
02421
02422
02423
02424
02425
02426
02427
02428
02429
02429
02430
02431
02432
02433
02434
02435
02436
02437
02438
02439
02439
02440
02441
02442
02443
02444
02445
02446
02447
02448
02449
02449
02450
02451
02452
02453
02454
02455
02456
02457
02458
02459
02459
02460
02461
02462
02463
02464
02465
02466
02467
02468
02469
02469
02470
02471
02472
02473
02474
02475
02476
02477
02478
02479
02479
02480
02481
02482
0
```

```

00845 #endif
00846 {
00848     EMBER_EVENT_INACTIVE = 0,
00850     EMBER_EVENT_MS_TIME,
00853     EMBER_EVENT_QS_TIME,
00856     EMBER_EVENT_MINUTE_TIME,
00858     EMBER_EVENT_ZERO_DELAY
00859 };
00860
00861
00865 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00866 enum EmberJoinMethod
00867 #else
00868 typedef uint8_t EmberJoinMethod;
00869 enum
00870 #endif
00871 {
00877     EMBER_USE_MAC_ASSOCIATION      = 0,
00878
00889     EMBER_USE_NWK_REJOIN          = 1,
00890
00891
00892     /* For those networks where the "permit joining" flag is never turned
00893     * on, they will need to use a NWK Rejoin. If those devices have been
00894     * preconfigured with the NWK key (including sequence number) they can use
00895     * a secured rejoin. This is only necessary for end devices since they need
00896     * a parent. Routers can simply use the ::EMBER_USE_NWK_COMMISIONING
00897     * join method below.
00898     */
00899     EMBER_USE_NWK_REJOIN_HAVE_NWK_KEY = 2,
00900
00905     EMBER_USE_NWK_COMMISIONING    = 3,
00906 };
00907
00908
00915 typedef struct {
00917     uint8_t   extendedPanId[8];
00919     uint16_t  panId;
00921     int8_t    radioTxPower;
00923     uint8_t   radioChannel;
00928     EmberJoinMethod joinMethod;
00929
00934     EmberNodeId nwkManagerId;
00940     uint8_t   nwkUpdateId;
00946     uint32_t  channels;
00947 } EmberNetworkParameters;
00948
00949
00950 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00951 #define emberInitializeNetworkParameters(parameters) \
00952     (MEMSET(parameters, 0, sizeof(EmberNetworkParameters)))
00953 #else
00954 void emberInitializeNetworkParameters(
00955     EmberNetworkParameters* parameters);
00956 #endif
00956
00960 typedef struct {
00962     uint16_t  profileId;
00964     uint16_t  clusterId;
00966     uint8_t   sourceEndpoint;
00968     uint8_t   destinationEndpoint;
00970     EmberApsOption options;
00972     uint16_t  groupId;
00974     uint8_t   sequence;
00975 } EmberApsFrame;
00976
00977
00984 typedef struct {
00986     EmberBindingType type;
00988     uint8_t   local;
00996     uint16_t  clusterId;
00998     uint8_t   remote;
01003     EmberEUI64 identifier;
01005     uint8_t   networkIndex;
01006 } EmberBindingTableEntry;
01007
01008
01014 typedef struct {
01016     uint16_t  shortId;
01019     uint8_t   averageLqi;

```

```

01022     uint8_t  inCost;
01029     uint8_t  outCost;
01035     uint8_t  age;
01037     EmberEUI64 longId;
01038 } EmberNeighborTableEntry;
01039
01045 typedef struct {
01047     uint16_t destination;
01049     uint16_t nextHop;
01052     uint8_t  status;
01055     uint8_t  age;
01058     uint8_t  concentratorType;
01063     uint8_t  routeRecordState;
01064 } EmberRouteTableEntry;
01065
01073 typedef struct {
01075     EmberMulticastId multicastId;
01079     uint8_t  endpoint;
01081     uint8_t  networkIndex;
01082 } EmberMulticastTableEntry;
01083
01088 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01089 enum EmberCounterType
01090#else
01091 typedef uint8_t EmberCounterType;
01092 enum
01093#endif
01094 {
01096     EMBER_COUNTER_MAC_RX_BROADCAST = 0,
01098     EMBER_COUNTER_MAC_TX_BROADCAST = 1,
01100     EMBER_COUNTER_MAC_RX_UNICAST = 2,
01102     EMBER_COUNTER_MAC_TX_UNICAST_SUCCESS = 3,
01108     EMBER_COUNTER_MAC_TX_UNICAST_RETRY = 4,
01110     EMBER_COUNTER_MAC_TX_UNICAST_FAILED = 5,
01111
01113     EMBER_COUNTERAPS_DATA_RX_BROADCAST = 6,
01115     EMBER_COUNTERAPS_DATA_TX_BROADCAST = 7,
01117     EMBER_COUNTERAPS_DATA_RX_UNICAST = 8,
01119     EMBER_COUNTERAPS_DATA_TX_UNICAST_SUCCESS
01120     = 9,
01125     EMBER_COUNTERAPS_DATA_TX_UNICAST_RETRY
01126     = 10,
01127     EMBER_COUNTERAPS_DATA_TX_UNICAST_FAILED
01128     = 11,
01131     EMBER_COUNTER_ROUTE_DISCOVERY_INITIATED
01132     = 12,
01134     EMBER_COUNTER_NEIGHBOR_ADDED = 13,
01136     EMBER_COUNTER_NEIGHBOR_REMOVED = 14,
01138     EMBER_COUNTER_NEIGHBOR_STALE = 15,
01139
01141     EMBER_COUNTER_JOIN_INDICATION = 16,
01143     EMBER_COUNTER_CHILD_REMOVED = 17,
01144
01146     EMBER_COUNTER_ASH_OVERFLOW_ERROR = 18,
01148     EMBER_COUNTER_ASH_FRAMING_ERROR = 19,
01150     EMBER_COUNTER_ASH_OVERRUN_ERROR = 20,
01151
01154     EMBER_COUNTER_NWK_FRAME_COUNTER_FAILURE
01155     = 21,
01158     EMBER_COUNTERAPS_FRAME_COUNTER_FAILURE
01159     = 22,
01161     EMBER_COUNTER_ASH_XOFF = 23,
01162
01166     EMBER_COUNTERAPS_LINK_KEY_NOT_AUTHORIZED
01167     = 24,
01170     EMBER_COUNTER_NWK_DECRYPTION_FAILURE = 25
01171
01174     EMBER_COUNTERAPS_DECRYPTION_FAILURE = 26
01175
01180     EMBER_COUNTER_ALLOCATE_PACKET_BUFFER_FAILURE
01181     = 27,
01183     EMBER_COUNTER_RELAYED_UNICAST = 28,

```

```

01184
01196     EMBER_COUNTER_PHY_TO_MAC_QUEUE_LIMIT_REACHED
01197     = 29,
01198
01202     EMBER_COUNTER_PACKET_VALIDATE_LIBRARY_DROPPED_COUNT
01203     = 30,
01204
01207     EMBER_COUNTER_TYPE_NWK_RETRY_OVERFLOW =
01208     31,
01209
01212     EMBER_COUNTER_PHY_CCA_FAIL_COUNT = 32,
01213
01217     EMBER_COUNTER_BROADCAST_TABLE_FULL = 33,
01218
01220     EMBER_COUNTER_TYPE_COUNT = 34,
01221 };
01222
01226 #define EMBER_COUNTER_STRINGS \
01227     "Mac Rx Bcast", \
01228     "Mac Tx Bcast", \
01229     "Mac Rx Ucast", \
01230     "Mac Tx Ucast", \
01231     "Mac Tx Ucast Retry", \
01232     "Mac Tx Ucast Fail", \
01233     "APS Rx Bcast", \
01234     "APS Tx Bcast", \
01235     "APS Rx Ucast", \
01236     "APS Tx Ucast Success", \
01237     "APS Tx Ucast Retry", \
01238     "APS Tx Ucast Fail", \
01239     "Route Disc Initiated", \
01240     "Neighbor Added", \
01241     "Neighbor Removed", \
01242     "Neighbor Stale", \
01243     "Join Indication", \
01244     "Child Moved", \
01245     "ASH Overflow", \
01246     "ASH Frame Error", \
01247     "ASH Overrun Error", \
01248     "NWK FC Failure", \
01249     "APS FC Failure", \
01250     "ASH XOff", \
01251     "APS Unauthorized Key", \
01252     "NWK Decrypt Failures", \
01253     "APS Decrypt Failures", \
01254     "Packet Buffer Allocate Failures", \
01255     "Relayed Ucast", \
01256     "Phy to MAC queue limit reached", \
01257     "Packet Validate drop count", \
01258     "NWK retry overflow", \
01259     "CCA Failures", \
01260     "Broadcast table full", \
01261     NULL \
01262
01264 typedef uint8_t EmberTaskId;
01265
01272 typedef struct {
01274     EmberEventUnits status;
01276     EmberTaskId taskid;
01280     uint32_t timeToExecute;
01281 } EmberEventControl;
01282
01290 typedef PGM struct EmberEventData_S {
01292     EmberEventControl *control;
01294     void (*handler)(void);
01295 } EmberEventData;
01296
01301 typedef struct {
01302     // The time when the next event associated with this task will fire
01303     uint32_t nextEventTime;
01304     // The list of events associated with this task
01305     EmberEventData *events;
01306     // A flag that indicates the task has something to do other than events
01307     bool busy;
01308 } EmberTaskControl;
01309
01314
01319 #define EMBER_TX_POWER_MODE_DEFAULT          0x0000
01320
01323 #define EMBER_TX_POWER_MODE_BOOST            0x0001

```

```

01324
01328 #define EMBER_TX_POWER_MODE_ALTERNATE           0x0002
01329
01333 #define EMBER_TX_POWER_MODE_BOOST_AND_ALTERNATE (EMBER_TX_POWER_MODE_BOOST
01334   \
01334           | EMBER_TX_POWER_MODE_ALTERNATE)
01335 #ifndef DOXYGEN_SHOULD_SKIP_THIS
01336 // The application does not ever need to call emberSetTxPowerMode() with the
01337 // txPowerMode parameter set to this value. This value is used internally by
01338 // the stack to indicate that the default token configuration has not been
01339 // overridden by a prior call to emberSetTxPowerMode().
01340 #define EMBER_TX_POWER_MODE_USE_TOKEN               0x8000
01341 #endif//DOXYGEN_SHOULD_SKIP_THIS
01342
01344
01349
01357 #define EMBER_PRIVATE_PROFILE_ID    0xC00E
01358
01362 #define EMBER_PRIVATE_PROFILE_ID_START 0xC00D
01363
01367 #define EMBER_PRIVATE_PROFILE_ID_END   0xC016
01368
01407 #define EMBER_BROADCAST_ALARM_CLUSTER      0x0000
01408
01445 #define EMBER_UNICAST_ALARM_CLUSTER        0x0001
01446
01462 #define EMBER_CACHED_UNICAST_ALARM_CLUSTER 0x0002
01463
01467 #define EMBER_REPORT_COUNTERS_REQUEST 0x0003
01468
01470 #define EMBER_REPORT_COUNTERS_RESPONSE 0x8003
01471
01476 #define EMBER_REPORT_AND_CLEAR_COUNTERS_REQUEST 0x0004
01477
01479 #define EMBER_REPORT_AND_CLEAR_COUNTERS_RESPONSE 0x8004
01480
01485 #define EMBER_OTA_CERTIFICATE_UPGRADE_CLUSTER 0x0005
01486
01488
01489
01492 typedef struct {
01494     uint8_t contents[EMBER_ENCRYPTION_KEY_SIZE];
01495 } EmberKeyData;
01496
01499 typedef struct {
01500     uint8_t contents[EMBER_CERTIFICATE_SIZE];
01501 } EmberCertificateData;
01502
01505 typedef struct {
01506     uint8_t contents[EMBER_PUBLIC_KEY_SIZE];
01507 } EmberPublicKeyData;
01508
01511 typedef struct {
01512     uint8_t contents[EMBER_PRIVATE_KEY_SIZE];
01513 } EmberPrivateKeyData;
01514
01517 typedef struct {
01518     uint8_t contents[EMBER_SMAC_SIZE];
01519 } EmberSmacData;
01520
01524 typedef struct {
01525     uint8_t contents[EMBER_SIGNATURE_SIZE];
01526 } EmberSignatureData;
01527
01530 typedef struct {
01531     uint8_t contents[EMBER_AES_HASH_BLOCK_SIZE];
01532 } EmberMessageDigest;
01533
01537 typedef struct {
01538     uint8_t result[EMBER_AES_HASH_BLOCK_SIZE];
01539     uint32_t length;
01540 } EmberAesMmoHashContext;
01541
01544 typedef struct {
01545     /* This is the certificate byte data. */
01546     uint8_t contents[EMBER_CERTIFICATE_283K1_SIZE];
01547 } EmberCertificate283k1Data;
01548
01551 typedef struct {
01552     uint8_t contents[EMBER_PUBLIC_KEY_283K1_SIZE];

```

```

0153 } EmberPublicKey283k1Data;
0154
01557 typedef struct {
01558     uint8_t contents[EMBER_PRIVATE_KEY_283K1_SIZE];
01559 } EmberPrivateKey283k1Data;
01560
01565 typedef struct {
01566     uint8_t contents[EMBER_SIGNATURE_283K1_SIZE];
01567 } EmberSignature283k1Data;
01568
01574 #define EMBER_STANDARD_SECURITY_MODE 0x0000
01575
01579 #define EMBER_TRUST_CENTER_NODE_ID 0x0000
01580
01581
01585 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01586 enum EmberInitialSecurityBitmask
01587 #else
01588 typedef uint16_t EmberInitialSecurityBitmask;
01589 enum
01590 #endif
01591 {
01594     EMBER_DISTRIBUTED_TRUST_CENTER_MODE
01595     = 0x0002,
01597     EMBER_TRUST_CENTER_GLOBAL_LINK_KEY
01598     = 0x0004,
01600     EMBER_PRECONFIGURED_NETWORK_KEY_MODE
01601     = 0x0008,
01602 #if !defined DOXYGEN_SHOULD_SKIP_THIS
01603     // Hidden fields used internally.
01604     EMBER_HAVE_TRUST_CENTER_UNKNOWN_KEY_TOKEN = 0x0010,
01605     EMBER_HAVE_TRUST_CENTER_LINK_KEY_TOKEN = 0x0020,
01606     EMBER_HAVE_TRUST_CENTER_MASTER_KEY_TOKEN = 0x0030,
01607 #endif
01608
01618     EMBER_HAVE_TRUST_CENTER_EUI64
01619     = 0x0040,
01626     EMBER_TRUST_CENTERUSES_HASHED_LINK_KEY
01627     = 0x0084,
01631     EMBER_HAVE_PRECONFIGURED_KEY
01632     = 0x0100,
01635     EMBER_HAVE_NETWORK_KEY
01641     EMBER_GET_LINK_KEY_WHEN_JOINING
01642     = 0x0400,
01647     EMBER_REQUIRE_ENCRYPTED_KEY
01648     = 0x0800
01658     EMBER_NO_FRAME_COUNTER_RESET
01659     = 0x1000,
01664     EMBER_GET_PRECONFIGURED_KEY_FROM_INSTALL_CODE
01665     = 0x2000,
01666 #if !defined DOXYGEN_SHOULD_SKIP_THIS
01667     // Internal data
01668     EM_SAVED_IN_TOKEN
01669     #define EM_SECURITY_INITIALIZED
01670
01671     // This is only used internally. High security is not released or supported
01672     // except for golden unit compliance.
01673     #define EMBER_HIGH_SECURITY_MODE
01674 #else
01675     /* All other bits are reserved and must be zero. */
01676 #endif
01677 };
01678
01682 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01683 enum EmberExtendedSecurityBitmask
01684 #else
01685 typedef uint16_t EmberExtendedSecurityBitmask;
01686 enum
01687 #endif
01688 {
01689 #ifndef DOXYGEN_SHOULD_SKIP_THIS
01690     // If this bit is set, we set the 'key token data' field in the Initial
01691     // Security Bitmask to 0 (No Preconfig Key token), otherwise we leave the
01692     // field as it is.
01693     EMBER_PRECONFIG_KEY_NOT_VALID
01694     = 0x0001,

```

```

01695 // bits 1-3 are unused.
01696 #define EMBER_JOINER_GLOBAL_LINK_KEY          = 0x0010,
01697 #define EMBER_EXT_NO_FRAME_COUNTER_RESET      = 0x0020,
01698 // bit 6-7 reserved for future use (stored in TOKEN).
01699 #define EMBER_NWK_LEAVE_REQUEST_NOT_ALLOWED =
01700 0x0100,
01701 #ifndef DOXYGEN_SHOULD_SKIP_THIS
01702 #define EMBER_R18_STACK_BEHAVIOR             = 0x0200,
01703#endif
01704 // bit 10 and 11 are stored in RAM only.
01705 // bit 11 is reserved for future use.
01706 // bits 12-15 are unused.
01707 };
01708 #define EMBER_NO_TRUST_CENTER_MODE   EMBER_DISTRIBUTED_TRUST_CENTER_MODE
01709 #define EMBER_GLOBAL_LINK_KEY      EMBER_TRUST_CENTER_GLOBAL_LINK_KEY
01710
01711 #if !defined DOXYGEN_SHOULD_SKIP_THIS
01712 #define NO_TRUST_CENTER_KEY_TOKEN    0x0000
01713 #define TRUST_CENTER_KEY_TOKEN_MASK 0x0030
01714 #define SECURITY_BIT_TOKEN_MASK    0x71FF
01715
01716 #define SECURITY_LOWER_BIT_MASK    0x000000FF // ""
01717 #define SECURITY_UPPER_BIT_MASK    0x00FF0000L // ""
01718#endif
01719
01720 typedef struct {
01721     uint16_t bitmask;
01722     EmberKeyData preconfiguredKey;
01723     EmberKeyData networkKey;
01724     uint8_t networkKeySequenceNumber;
01725     EmberEUI64 preconfiguredTrustCenterEui64
01726 };
01727 } EmberInitialSecurityState;
01728
01729 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01730 enum EmberCurrentSecurityBitmask
01731#else
01732 typedef uint16_t EmberCurrentSecurityBitmask;
01733 enum
01734#endif
01735 {
01736     #if defined DOXYGEN_SHOULD_SKIP_THIS
01737     // These options are the same for Initial and Current Security state
01738     #else
01739     #endif
01740     EMBER_STANDARD_SECURITY_MODE_           =
01741     0x0000,
01742     EMBER_DISTRIBUTED_TRUST_CENTER_MODE_  =
01743     0x0002,
01744     EMBER_TRUST_CENTER_GLOBAL_LINK_KEY_   =
01745     0x0004,
01746     #else
01747     // Bit 3 reserved
01748     #endif
01749     EMBER_HAVE_TRUST_CENTER_LINK_KEY_     =
01750     0x0010,
01751     EMBER_TRUST_CENTERUSES_HASHED_LINK_KEY_ =
01752     0x0084,
01753     // Bits 1,5,6, 8-15 reserved
01754 };
01755 #if !defined DOXYGEN_SHOULD_SKIP_THIS
01756 #define INITIAL_AND_CURRENT_BITMASK      0x00FF
01757#endif
01758
01759
01760
01761
01762
01763
01764
01765
01766
01767
01768
01769
01770
01771
01772
01773
01774
01775
01776
01777
01778
01779
01780
01781
01782
01783
01784
01785
01786
01787
01788
01789
01790
01791
01792
01793
01794
01795
01796
01797
01798
01799
01800
01801
01802
01803
01804
01805
01806
01807
01808
01809
01810
01811
01812
01813
01814
01815
01816
01817
01818
01819
01820
01821
01822
01823
01824
01825

```

```

01828 typedef struct {
01829     EmberCurrentSecurityBitmask bitmask;
01830     EmberEUI64 trustCenterLongAddress;
01831 } EmberCurrentSecurityState;
01832
01833
01842 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01843 enum EmberKeyStructBitmask
01844 {
01845     typedef uint16_t EmberKeyStructBitmask;
01846     enum
01847     #endif
01848     {
01849         EMBER_KEY_HAS_SEQUENCE_NUMBER = 0x0001,
01850         EMBER_KEY_HAS_OUTGOING_FRAME_COUNTER =
0x0002,
01851         EMBER_KEY_HAS_INCOMING_FRAME_COUNTER =
0x0004,
01852         EMBER_KEY_HAS_PARTNER_EUI64 = 0x0008,
01853         EMBER_KEY_IS_AUTHORIZED = 0x0010,
01854         EMBER_KEY_PARTNER_IS_SLEEPY = 0x0020,
01855     };
01856
01877 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01878 enum EmberKeyType
01879 {
01880     typedef uint8_t EmberKeyType;
01881     enum
01882     #endif
01883     {
01884         EMBER_TRUST_CENTER_LINK_KEY = 1,
01885         EMBER_TRUST_CENTER_MASTER_KEY = 2,
01886         EMBER_CURRENT_NETWORK_KEY = 3,
01887         EMBER_NEXT_NETWORK_KEY = 4,
01888         EMBER_APPLICATION_LINK_KEY = 5,
01889         EMBER_APPLICATION_MASTER_KEY = 6,
01890     },
01891
01901 typedef struct {
01902     EmberKeyStructBitmask bitmask;
01903     EmberKeyType type;
01904     EmberKeyData key;
01905     uint32_t outgoingFrameCounter;
01906     uint32_t incomingFrameCounter;
01907     uint8_t sequenceNumber;
01908     EmberEUI64 partnerEUI64;
01909 } EmberKeyStruct;
01910
01924
01928 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01929 enum EmberKeyStatus
01930 {
01931     typedef uint8_t EmberKeyStatus;
01932     enum
01933     #endif
01934     {
01935         EMBER_KEY_STATUS_NONE = 0,
01936         EMBER_APP_LINK_KEY_ESTABLISHED = 1,
01937         EMBER_APP_MASTER_KEY_ESTABLISHED = 2,
01938         EMBER_TRUST_CENTER_LINK_KEY_ESTABLISHED
01939             = 3,
01940         EMBER_KEY_ESTABLISHMENT_TIMEOUT = 4,
01941         EMBER_KEY_TABLE_FULL = 5,
01942
01943         // These are success status values applying only to the
01944         // Trust Center answering key requests
01945         EMBER_TC_RESPONDED_TO_KEY_REQUEST = 6
01946
01947         EMBER_TC_APP_KEY_SENT_TO_REQUESTER = 7,
01948
01949         // These are failure status values applying only to the
01950         // Trust Center answering key requests
01951         EMBER_TC_RESPONSE_TO_KEY_REQUEST_FAILED
01952             = 8,
01953         EMBER_TC_REQUEST_KEY_TYPE_NOT_SUPPORTED
01954             = 9,
01955         EMBER_TC_NO_LINK_KEY_FOR_REQUESTER = 10
01956     }
01957 }

```

```

10,
01953   EMBER_TC_REQUESTER_EUI64_UNKNOWN          = 11
01954   ,
01954   EMBER_TC_RECEIVED_FIRST_APP_KEY_REQUEST    = 12,
01955   EMBER_TC_TIMEOUT_WAITING_FOR_SECOND_APP_KEY_REQUEST
01955   = 13,
01956   EMBER_TC_NON_MATCHING_APP_KEY_REQUEST_RECEIVED
01956   = 14,
01957   EMBER_TC_FAILED_TO_SEND_APP_KEYS           = 15
01958   ,
01958   EMBER_TC_FAILED_TO_STORE_APP_KEY_REQUEST    = 16,
01959   EMBER_TC_REJECTED_APP_KEY_REQUEST          =
01959   17,
01960   EMBER_TC_FAILED_TO_GENERATE_NEW_KEY         =
01960   18,
01961   EMBER_TC_FAILED_TO_SEND_TC_KEY              = 19,
01962
01963 // These are generic status values for a key requester.
01964   EMBER_TRUST_CENTER_IS_PRE_R21             = 30,
01965
01966 // These are status values applying only to the Trust Center
01967 // verifying link keys.
01968   EMBER_TC_REQUESTER_VERIFY_KEY_TIMEOUT      =
01968   50,
01969   EMBER_TC_REQUESTER_VERIFY_KEY_FAILURE       =
01969   51,
01970   EMBER_TC_REQUESTER_VERIFY_KEY_SUCCESS        =
01970   52,
01971
01972 // These are status values applying only to the key requester
01973 // verifying link keys.
01974   EMBER_VERIFY_LINK_KEY_FAILURE               = 100,
01975   EMBER_VERIFY_LINK_KEY_SUCCESS              = 101,
01976 };
01977
01981 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01982 enum EmberLinkKeyRequestPolicy
01983 #else
01984 typedef uint8_t EmberLinkKeyRequestPolicy;
01985 enum
01986 #endif
01987 {
01988   EMBER_DENY_KEY_REQUESTS  = 0x00,
01989   EMBER_ALLOW_KEY_REQUESTS = 0x01,
01990   EMBER_GENERATE_NEW_TC_LINK_KEY = 0x02,
01991 };
01992
01993
02001 #if defined DOXYGEN_SHOULD_SKIP_THIS
02002 uint8_t* emberKeyContents(EmberKeyData* key);
02003 #else
02004 #define emberKeyContents(key) ((key)->contents)
02005 #endif
02006
02014 #if defined DOXYGEN_SHOULD_SKIP_THIS
02015 uint8_t* emberCertificateContents(EmberCertificateData
02015 * cert);
02016 #else
02017 #define emberCertificateContents(cert) ((cert)->contents)
02018 #endif
02019
02027 #if defined DOXYGEN_SHOULD_SKIP_THIS
02028 uint8_t* emberPublicKeyContents(EmberPublicKeyData
02028 * key);
02029 #else
02030 #define emberPublicKeyContents(key) ((key)->contents)
02031 #endif
02032
02040 #if defined DOXYGEN_SHOULD_SKIP_THIS
02041 uint8_t* emberPrivateKeyContents(EmberPrivateKeyData
02041 * key);
02042 #else
02043 #define emberPrivateKeyContents(key) ((key)->contents)
02044 #endif
02045
02050 #if defined DOXYGEN_SHOULD_SKIP_THIS
02051 uint8_t* emberSmacContents(EmberSmacData* key);
02052 #else

```

```

02053 #define emberSmacContents(key) ((key)->contents)
02054 #endif
02055
02059 #if defined DOXYGEN_SHOULD_SKIP_THIS
02060 uint8_t* emberSignatureContents(EmberSignatureData
02061 * sig);
02061 #else
02062 #define emberSignatureContents(sig) ((sig)->contents)
02063 #endif
02064
02072 #if defined DOXYGEN_SHOULD_SKIP_THIS
02073 uint8_t* emberCertificate283k1Contents(
02074 EmberCertificate283k1Data* cert);
02074 #else
02075 #define emberCertificate283k1Contents(cert) ((cert)->contents)
02076 #endif
02077
02085 #if defined DOXYGEN_SHOULD_SKIP_THIS
02086 uint8_t* emberPublicKey283k1Contents(
02087 EmberPublicKey283k1Data* key);
02087 #else
02088 #define emberPublicKey283k1Contents(key) ((key)->contents)
02089 #endif
02090
02098 #if defined DOXYGEN_SHOULD_SKIP_THIS
02099 uint8_t* emberPrivateKey283k1Contents(
02100 EmberPrivateKey283k1Data* key);
02100 #else
02101 #define emberPrivateKey283k1Contents(key) ((key)->contents)
02102 #endif
02103
02107 #if defined DOXYGEN_SHOULD_SKIP_THIS
02108 uint8_t* ember283k1SignatureContents(
02109 Ember283k1SignatureData* sig);
02109 #else
02110 #define ember283k1SignatureContents(sig) ((sig)->contents)
02111 #endif
02112
02113 #ifdef DOXYGEN_SHOULD_SKIP_THIS
02114 enum EmberKeySettings
02115 #else
02116 typedef uint16_t EmberKeySettings;
02117 enum
02118 #endif
02119 {
02120   EMBER_KEY_PERMISSIONS_NONE          = 0x0000,
02121   EMBER_KEY_PERMISSIONS_READING_ALLOWED =
0x0001,
02122   EMBER_KEY_PERMISSIONS_HASHING_ALLOWED =
0x0002,
02123 };
02124
02125
02129 typedef struct {
02130   EmberKeySettings keySettings;
02131 } EmberMfgSecurityStruct;
02132
02133
02138 #define EMBER_MFG_SECURITY_CONFIG_MAGIC_NUMBER 0xCABAD11FUL
02139
02140
02145 #ifdef DOXYGEN_SHOULD_SKIP_THIS
02146 enum EmberMacPassthroughType
02147 #else
02148 typedef uint8_t EmberMacPassthroughType;
02149 enum
02150 #endif
02151 {
02153   EMBER_MAC_PASSTHROUGH_NONE          = 0x00,
02155   EMBER_MAC_PASSTHROUGH_SE_INTERPAN    =
0x01,
02157   EMBER_MAC_PASSTHROUGH_EMBERNET      = 0x02,
02159   EMBER_MAC_PASSTHROUGH_EMBERNET_SOURCE =
0x04,
02161   EMBER_MAC_PASSTHROUGH_APPLICATION    =
0x08,
02163   EMBER_MAC_PASSTHROUGH_CUSTOM         = 0x10,
02164
02165 #if !defined DOXYGEN_SHOULD_SKIP_THIS
02166

```

```

02167     EM_MAC_PASSTHROUGH_INTERNAL_ZLL      = 0x80,
02168     EM_MAC_PASSTHROUGH_INTERNAL_GP       = 0x40
02169 #endif
02170 };
02171
02176 typedef uint16_t EmberMacFilterMatchData;
02177
02178 #define EMBER_MAC_FILTER_MATCH_ENABLED_MASK      0x0001
02179 #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_MASK   0x0003
02180 #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_MASK  0x000C
02181 #define EMBER_MAC_FILTER_MATCH_ON_DEST_MASK        0x0030
02182 #define EMBER_MAC_FILTER_MATCH_ON_SOURCE_MASK      0x0080
02183
02184 // Globally turn on/off this filter
02185 #define EMBER_MAC_FILTER_MATCH_ENABLED            0x0000
02186 #define EMBER_MAC_FILTER_MATCH_DISABLED           0x0001
02187
02188 // Pick either one of these
02189 #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_NONE   0x0000
02190 #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_LOCAL   0x0001
02191 #define EMBER_MAC_FILTER_MATCH_ON_PAN_DEST_BROADCAST 0x0002
02192
02193 // and one of these
02194 #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_NONE  0x0000
02195 #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_NON_LOCAL 0x0004
02196 #define EMBER_MAC_FILTER_MATCH_ON_PAN_SOURCE_LOCAL   0x0008
02197
02198 // and one of these
02199 #define EMBER_MAC_FILTER_MATCH_ON_DEST_BROADCAST_SHORT 0x0000
02200 #define EMBER_MAC_FILTER_MATCH_ON_DEST_UNICAST_SHORT  0x0010
02201 #define EMBER_MAC_FILTER_MATCH_ON_DEST_UNICAST_LONG   0x0020
02202
02203 // and one of these
02204 #define EMBER_MAC_FILTER_MATCH_ON_SOURCE_LONG      0x0000
02205 #define EMBER_MAC_FILTER_MATCH_ON_SOURCE_SHORT     0x0080
02206 #define EMBER_MAC_FILTER_MATCH_ON_SOURCE_NONE       0x0100
02207
02208 // Last entry should set this and nothing else. No other bits will be
02209 // examined.
02209 #define EMBER_MAC_FILTER_MATCH_END                 0x8000
02210
02214 typedef struct {
02215     uint8_t filterIndexMatch;
02216     EmberMacPassthroughType legacyPassthroughType
02217     ;
02218 } EmberMacFilterMatchStruct;
02219
02220
02224 typedef uint8_t EmberLibraryStatus;
02225
02230
02236 #ifdef DOXYGEN_SHOULD_SKIP_THIS
02237 enum EmberZdoStatus
02238 #else
02239 typedef uint8_t EmberZdoStatus;
02240 enum
02241 #endif
02242 {
02243     // These values are taken from Table 48 of ZDP Errata 043238r003 and Table 2
02244     // of NWK 02130r10.
02245     EMBER_ZDP_SUCCESS          = 0x00,
02246     // 0x01 to 0x7F are reserved
02247     EMBER_ZDP_INVALID_REQUEST_TYPE = 0x80,
02248     EMBER_ZDP_DEVICE_NOT_FOUND   = 0x81,
02249     EMBER_ZDP_INVALID_ENDPOINT   = 0x82,
02250     EMBER_ZDP_NOT_ACTIVE        = 0x83,
02251     EMBER_ZDP_NOT_SUPPORTED     = 0x84,
02252     EMBER_ZDP_TIMEOUT           = 0x85,
02253     EMBER_ZDP_NO_MATCH          = 0x86,
02254     // 0x87 is reserved          = 0x87,
02255     EMBER_ZDP_NO_ENTRY          = 0x88,
02256     EMBER_ZDP_NO_DESCRIPTOR     = 0x89,
02257     EMBER_ZDP_INSUFFICIENT_SPACE = 0x8a,
02258     EMBER_ZDP_NOT_PERMITTED    = 0x8b,
02259     EMBER_ZDP_TABLE_FULL        = 0x8c,
02260     EMBER_ZDP_NOT_AUTHORIZED   = 0x8d,
02261
02262     EMBER_NWK_ALREADY_PRESENT   = 0xC5,
02263     EMBER_NWK_TABLE_FULL        = 0xC7,

```

```

02264     EMBER_NWK_UNKNOWN_DEVICE      = 0xC8
02265 };
02266
02279
02280
02281
02282
02283
02284
02285
02286
02287
02288
02289
02290
02291
02292
02293 #define NETWORK_ADDRESS_REQUEST      0x0000
02294 #define NETWORK_ADDRESS_RESPONSE    0x8000
02295 #define IEEE_ADDRESS_REQUEST        0x0001
02296 #define IEEE_ADDRESS_RESPONSE       0x8001
02297
02298
02305 //           <node descriptor: 13>
02306 //
02307 //   Node Descriptor field is divided into subfields of bitmasks as follows:
02308 //   (Note: All lengths below are given in bits rather than bytes.)
02309 //       Logical Type:            3
02310 //       Complex Descriptor Available: 1
02311 //       User Descriptor Available: 1
02312 //       (reserved/unused):       3
02313 //       APS Flags:              3
02314 //       Frequency Band:         5
02315 //       MAC capability flags:   8
02316 //       Manufacturer Code:      16
02317 //       Maximum buffer size:    8
02318 //       Maximum incoming transfer size: 16
02319 //       Server mask:             16
02320 //       Maximum outgoing transfer size: 16
02321 //       Descriptor Capability Flags: 8
02322 //       See ZigBee document 053474, Section 2.3.2.3 for more details.
02324 #define NODE_DESCRIPTOR_REQUEST      0x0002
02325 #define NODE_DESCRIPTOR_RESPONSE     0x8002
02326
02327
02336 //       See ZigBee document 053474, Section 2.3.2.4 for more details.
02338 #define POWER_DESCRIPTOR_REQUEST     0x0003
02339 #define POWER_DESCRIPTOR_RESPONSE    0x8003
02340
02341
02355 #define SIMPLE_DESCRIPTOR_REQUEST    0x0004
02356 #define SIMPLE_DESCRIPTOR_RESPONSE   0x8004
02357
02358
02367 #define ACTIVE_ENDPOINTS_REQUEST     0x0005
02368 #define ACTIVE_ENDPOINTS_RESPONSE    0x8005
02369
02370
02382 #define MATCH_DESCRIPTORS_REQUEST    0x0006
02383 #define MATCH_DESCRIPTORS_RESPONSE   0x8006
02384
02385
02395 #define DISCOVERY_CACHE_REQUEST      0x0012
02396 #define DISCOVERY_CACHE_RESPONSE    0x8012
02397
02398
02407 #define END_DEVICE_ANNOUNCE          0x0013
02408 #define END_DEVICE_ANNOUNCE_RESPONSE 0x8013
02409
02410
02422 #define SYSTEM_SERVER_DISCOVERY_REQUEST 0x0015
02423 #define SYSTEM_SERVER_DISCOVERY_RESPONSE 0x8015
02424
02425
02438 #define PARENT_ANNOUNCE               0x001F
02439 #define PARENT_ANNOUNCE_RESPONSE     0x801F
02440
02441
02446 #ifdef DOXYGEN_SHOULD_SKIP_THIS
02447 enum EmberZdoServerMask

```

```

02448 #else
02449 typedef uint16_t EmberZdoServerMask;
02450 enum
02451 #endif
02452 {
02453     EMBER_ZDP_PRIMARY_TRUST_CENTER      =
02454     0x0001,
02455     EMBER_ZDP_SECONDARY_TRUST_CENTER    =
02456     0x0002,
02457     EMBER_ZDP_PRIMARY_BINDING_TABLE_CACHE
= 0x0004,
02458     EMBER_ZDP_SECONDARY_BINDING_TABLE_CACHE
= 0x0008,
02459     EMBER_ZDP_PRIMARY_DISCOVERY_CACHE   =
02460     0x0010,
02461     EMBER_ZDP_SECONDARY_DISCOVERY_CACHE =
02462     0x0020,
02463     EMBER_ZDP_NETWORK_MANAGER           = 0x0040,
02464     // Bits 0x0080 to 0x8000 are reserved.
02465 };
02466
02467 #define FIND_NODE_CACHE_REQUEST      0x001C
02468 #define FIND_NODE_CACHE_RESPONSE      0x801C
02469
02470
02471 #define END_DEVICE_BIND_REQUEST        0x0020
02472 #define END_DEVICE_BIND_RESPONSE        0x8020
02473
02474
02475 #define UNICAST_BINDING               0x03
02476 #define UNICAST_MANY_TO_ONE_BINDING    0x83
02477 #define MULTICAST_BINDING              0x01
02478
02479
02480 #define BIND_REQUEST                  0x0021
02481 #define BIND_RESPONSE                 0x8021
02482 #define UNBIND_REQUEST                0x0022
02483 #define UNBIND_RESPONSE               0x8022
02484
02485
02486 #define LQI_TABLE_REQUEST              0x0031
02487 #define LQI_TABLE_RESPONSE             0x8031
02488
02489
02490 #define ROUTING_TABLE_REQUEST         0x0032
02491 #define ROUTING_TABLE_RESPONSE        0x8032
02492
02493
02494 #define BINDING_TABLE_REQUEST          0x0033
02495 #define BINDING_TABLE_RESPONSE         0x8033
02496
02497
02498 #define LEAVE_REQUEST                 0x0034
02499 #define LEAVE_RESPONSE                0x8034
02500
02501
02502 #define LEAVE_REQUEST_REMOVE_CHILDREN_FLAG 0x40
02503 #define LEAVE_REQUEST_REJOIN_FLAG       0x80
02504
02505
02506 #define PERMIT_JOINING_REQUEST        0x0036
02507 #define PERMIT_JOINING_RESPONSE       0x8036
02508
02509
02510 #define NWK_UPDATE_REQUEST             0x0038
02511 #define NWK_UPDATE_RESPONSE            0x8038
02512
02513
02514 #define COMPLEX_DESCRIPTOR_REQUEST     0x0010
02515 #define COMPLEX_DESCRIPTOR_RESPONSE    0x8010
02516 #define USER_DESCRIPTOR_REQUEST        0x0011
02517 #define USER_DESCRIPTOR_RESPONSE       0x8011
02518 #define DISCOVERY_REGISTER_REQUEST     0x0012
02519 #define DISCOVERY_REGISTER_RESPONSE    0x8012
02520
02521
02522 #define USER_DESCRIPTOR_SET            0x0014
02523 #define USER_DESCRIPTOR_CONFIRM         0x8014
02524 #define NETWORK_DISCOVERY_REQUEST      0x0030
02525 #define NETWORK_DISCOVERY_RESPONSE     0x8030
02526
02527
02528 #define DIRECT_JOIN_REQUEST            0x0035
02529 #define DIRECT_JOIN_RESPONSE           0x8035
02530
02531

```

```

02705 #define CLUSTER_ID_RESPONSE_MINIMUM 0x8000
02706
02707
02720 #ifdef DOXYGEN_SHOULD_SKIP_THIS
02721 enum EmberZdoConfigurationFlags
02722 #else
02723 typedef uint8_t EmberZdoConfigurationFlags;
02724 enum
02725 #endif
02726
02727 {
02728 EMBER_APP RECEIVES SUPPORTED_ZDO_REQUESTS
02729 = 0x01,
02730 EMBER_APP HANDLES UNSUPPORTED_ZDO_REQUESTS
02731 = 0x02,
02732 EMBER_APP HANDLES ZDO_ENDPOINT_REQUESTS
02733 = 0x04,
02734 EMBER_APP HANDLES ZDO_BINDING_REQUESTS
02735 = 0x08
02736 };
02737
02738
02739
02740
02741
02742
02743
02744 #endif // EMBER_TYPES_H
02745
02746 #include "stack/include/zll-types.h"
02747 #include "stack/include/rf4ce-types.h"
02748 #include "stack/include/gp-types.h"
02749
02750
02751
02752

```

8.69 ember.h File Reference

```

#include "ember-types.h"
#include "byte-utilities.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 "zigbee-device-stack.h"
#include "event.h"
#include "ember-debug.h"
#include "library.h"
#include "multi-network.h"
#include "zll-api.h"
#include "rf4ce-api.h"

```

PHY Information

Bit masks for TOKEN_MFG_RADIO_BANDS_SUPPORTED.

- #define RADIO_BANDS_SUPPORTED_2400

8.69.1 Detailed Description

The master include file for the EmberZNet API. See [EmberZNet Stack API Reference](#) for documentation.

Definition in file [ember.h](#).

8.69.2 Macro Definition Documentation

8.69.2.1 #define RADIO_BANDS_SUPPORTED_2400

2.4GHz band

Definition at line [63](#) of file [ember.h](#).

8.70 ember.h

```

00001
00021 #ifndef __EMBER_H__
00022 #define __EMBER_H__
00023
00024 #include "ember-types.h"
00025 #include "byte-utilities.h"
00026 #include "stack-info.h"
00027 #include "network-formation.h"
00028 #include "packet-buffer.h"
00029 #include "message.h"
00030 #include "raw-message.h"
00031 #include "child.h"
00032 #include "security.h"
00033 #include "aes-mmo.h"
00034 #include "binding-table.h"
00035 #include "bootload.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 #include "zll-api.h"
00042 #include "rf4ce-api.h"
00043
00044 // We strip the multi-network code for flash saving purposes on the EM2xx and
00045 // EM351 platforms.
00046 #ifndef EMBER_MULTI_NETWORK_STRIPPED
00047 #if defined(XAP2B) || defined(CORTEXM3_EM351)
00048 #define EMBER_MULTI_NETWORK_STRIPPED
00049 #endif // defined(XAP2B) || defined(CORTEXM3_EM351)
00050 #endif // EMBER_MULTI_NETWORK_STRIPPED
00051
00056 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00057 #define RADIO_BANDS_SUPPORTED_868 BIT(0)
00058 #define RADIO_BANDS_SUPPORTED_915 BIT(1)
00059 #define RADIO_BANDS_SUPPORTED_433 BIT(2)
00060 #endif // DOXYGEN_SHOULD_SKIP_THIS
00061
00063 #define RADIO_BANDS_SUPPORTED_2400 BIT(3)
00064
00065 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00066 #define RADIO_BANDS_SUPPORTED_408 BIT(4)
00067 #endif // DOXYGEN_SHOULD_SKIP_THIS
00068
00070
00071 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00072
00075 #include "config/config.h"
00077

```

```

00078 #ifdef DEBUG_ASSERT
00079 extern bool enableFailure;
00080 extern uint8_t rateOfFailure;
00081 extern uint8_t failuresInARow;
00082 static uint8_t bufferFailure;
00083 bool generateFailure(void);
00084 void dumpFailure(void);
00085 #endif
00086
00087 #endif //DOXYGEN_SHOULD_SKIP_THIS
00088
00089 #endif // __EMBER_H__
00090

```

8.71 endian.h File Reference

Macros

- `#define HTONL`
- `#define HTONS`

Functions

- `uint16_t NTOHS (uint16_t val)`
- `uint32_t NTOHL (uint32_t val)`
- `uint32_t SwapEndiannessInt32u (uint32_t val)`

8.71.1 Detailed Description

See [Network to Host Byte Order Conversion](#) for detailed documentation.

Definition in file [endian.h](#).

8.72 endian.h

```

00001
00007 #ifndef __ENDIAN_H__
00008 #define __ENDIAN_H__
00009
00019 // If this is being compiled on a Mac then we need to disable the defines
00020 // from its own internal _endian.h file since they conflict with our
00021 // defines. This allows us to compile the 3xx tools for Mac.
00022 #ifdef __APPLE__
00023     #undef NTOHL
00024     #undef NTOHS
00025     #undef HTONL
00026     #undef HTONS
00027 #endif
00028
00029 #if BIGENDIAN_CPU == false
00030
00034     #ifndef NTOHS // some platforms already define this
00035         uint16_t NTOHS(uint16_t val);
00036     #endif
00037
00041     #ifndef NTOHL // some platforms already define this
00042         uint32_t NTOHL(uint32_t val);
00043     #endif
00044
00045 #else // BIGENDIAN_CPU == true
00046
00047     #ifndef NTOHS // some platforms already define this

```

```

00048     #define NTOHS(val)  (val)
00049     #endif
00050     #ifndef NTOHL // some platforms already define this
00051         #define NTOHL(val)  (val)
00052     #endif
00053
00054 #endif
00055
00056 // The HTON and NTOH operations are the same for sane architectures (eg. big
00057 // and little endian) so define the inverse functions if they don't exist
00058 #ifndef HTONL
00059     #define HTONL NTOHL
00060 #endif
00061 #ifndef HTONS
00062     #define HTONS NTOHS
00063 #endif
00064
00065
00066 /* Swap byte order, e.g. LE to BE or BE to LE.
00067 * This function is used when working with 802.15.4 frames on 8051 MCUs. */
00068 uint32_t SwapEndiannessInt32u(uint32_t val);
00069
00070
00074 #endif //__ENDIAN_H__
00075

```

8.73 error-def.h File Reference

Generic Messages

These messages are system wide.

- #define EMBER_SUCCESS(x00)
- #define EMBER_ERR_FATAL(x01)
- #define EMBER_BAD_ARGUMENT(x02)
- #define EMBER_NOT_FOUND(x03)
- #define EMBER_EEPROM_MFG_STACK_VERSION_MISMATCH(x04)
- #define EMBER_INCOMPATIBLE_STATIC_MEMORY_DEFINITIONS(x05)
- #define EMBER_EEPROM_MFG_VERSION_MISMATCH(x06)
- #define EMBER_EEPROM_STACK_VERSION_MISMATCH(x07)

Packet Buffer Module Errors

- #define EMBER_NO_BUFFERS(x18)

Serial Manager Errors

- #define EMBER_SERIAL_INVALID_BAUD_RATE(x20)
- #define EMBER_SERIAL_INVALID_PORT(x21)
- #define EMBER_SERIAL_TX_OVERFLOW(x22)
- #define EMBER_SERIAL_RX_OVERFLOW(x23)
- #define EMBER_SERIAL_RX_FRAME_ERROR(x24)
- #define EMBER_SERIAL_RX_PARITY_ERROR(x25)
- #define EMBER_SERIAL_RX_EMPTY(x26)
- #define EMBER_SERIAL_RX_OVERRUN_ERROR(x27)

MAC Errors

- #define EMBER_MAC_TRANSMIT_QUEUE_FULL(x39)
- #define EMBER_MAC_UNKNOWN_HEADER_TYPE(x3A)
- #define EMBER_MAC_ACK_HEADER_TYPE(x3B)
- #define EMBER_MAC_SCANNING(x3D)
- #define EMBER_MAC_NO_DATA(x31)
- #define EMBER_MAC_JOINED_NETWORK(x32)
- #define EMBER_MAC_BAD_SCAN_DURATION(x33)
- #define EMBER_MAC_INCORRECT_SCAN_TYPE(x34)
- #define EMBER_MAC_INVALID_CHANNEL_MASK(x35)
- #define EMBER_MAC_COMMAND_TRANSMIT_FAILURE(x36)
- #define EMBER_MAC_NO_ACK RECEIVED(x40)
- #define EMBER_MAC_RADIO_NETWORK_SWITCH_FAILED(x41)
- #define EMBER_MAC_INDIRECT_TIMEOUT(x42)

Simulated EEPROM Errors

- #define EMBER_SIM_EEPROM_ERASE_PAGE_GREEN(x43)
- #define EMBER_SIM_EEPROM_ERASE_PAGE_RED(x44)
- #define EMBER_SIM_EEPROM_FULL(x45)
- #define EMBER_SIM_EEPROM_INIT_1 FAILED(x48)
- #define EMBER_SIM_EEPROM_INIT_2 FAILED(x49)
- #define EMBER_SIM_EEPROM_INIT_3 FAILED(x4A)
- #define EMBER_SIM_EEPROM_REPAIRING(x4D)

Flash Errors

- #define EMBER_ERR_FLASH_WRITE_INHIBITED(x46)
- #define EMBER_ERR_FLASH_VERIFY FAILED(x47)
- #define EMBER_ERR_FLASH_PROG FAIL(x4B)
- #define EMBER_ERR_FLASH_ERASE FAIL(x4C)

Bootloader Errors

- #define EMBER_ERR_BOOTLOADER_TRAP_TABLE_BAD(x58)
- #define EMBER_ERR_BOOTLOADER_TRAP_UNKNOWN(x59)
- #define EMBER_ERR_BOOTLOADER_NO_IMAGE(x05A)

Transport Errors

- #define EMBER_DELIVERY FAILED(x66)
- #define EMBER_BINDING_INDEX_OUT_OF_RANGE(x69)
- #define EMBER_ADDRESS_TABLE_INDEX_OUT_OF_RANGE(x6A)
- #define EMBER_INVALID_BINDING_INDEX(x6C)
- #define EMBER_INVALID_CALL(x70)
- #define EMBER_COST_NOT_KNOWN(x71)
- #define EMBER_MAX_MESSAGE_LIMIT_REACHED(x72)
- #define EMBER_MESSAGE_TOO_LONG(x74)
- #define EMBER_BINDING_IS_ACTIVE(x75)
- #define EMBER_ADDRESS_TABLE_ENTRY_IS_ACTIVE(x76)

Green Power status codes

- #define EMBER_MATCH(x78)
- #define EMBER_DROP_FRAME(x79)
- #define EMBER_PASS_UNPROCESSED(x7A)
- #define EMBER_TX_THEN_DROP(x7B)
- #define EMBER_NO_SECURITY(x7C)
- #define EMBER_COUNTER_FAILURE(x7D)
- #define EMBER_AUTH_FAILURE(x7E)
- #define EMBER_UNPROCESSED(x7F)

HAL Module Errors

- #define EMBER_ADC_CONVERSION_DONE(x80)
- #define EMBER_ADC_CONVERSION_BUSY(x81)
- #define EMBER_ADC_CONVERSION_DEFERRED(x82)
- #define EMBER_ADC_NO_CONVERSION_PENDING(x84)
- #define EMBER_SLEEP_INTERRUPTED(x85)

PHY Errors

- #define EMBER_PHY_TX_UNDERFLOW(x88)
- #define EMBER_PHY_TX_INCOMPLETE(x89)
- #define EMBER_PHY_INVALID_CHANNEL(x8A)
- #define EMBER_PHY_INVALID_POWER(x8B)
- #define EMBER_PHY_TX_BUSY(x8C)
- #define EMBER_PHY_TX_CCA_FAIL(x8D)
- #define EMBER_PHY_OSCILLATOR_CHECK_FAILED(x8E)
- #define EMBER_PHY_ACK RECEIVED(x8F)

Return Codes Passed to emberStackStatusHandler()

See also [emberStackStatusHandler\(\)](#).

- #define EMBER_NETWORK_UP(x90)
- #define EMBER_NETWORK_DOWN(x91)
- #define EMBER_JOIN FAILED(x94)
- #define EMBER_MOVE FAILED(x96)
- #define EMBER_CANNOT JOIN AS ROUTER(x98)
- #define EMBER_NODE_ID_CHANGED(x99)
- #define EMBER_PAN_ID_CHANGED(x9A)
- #define EMBER_CHANNEL_CHANGED(x9B)
- #define EMBER_NO_BEACONS(xAB)
- #define EMBER RECEIVED KEY IN THE CLEAR(xAC)
- #define EMBER_NO_NETWORK_KEY RECEIVED(xAD)
- #define EMBER_NO_LINK_KEY RECEIVED(xAE)
- #define EMBER_PRECONFIGURED_KEY REQUIRED(xAF)

Security Errors

- #define EMBER_KEY_INVALID(xB2)
- #define EMBER_INVALID_SECURITY_LEVEL(x95)
- #define EMBER_APS_ENCRYPTION_ERROR(xA6)
- #define EMBER_TRUST_CENTER_MASTER_KEY_NOT_SET(xA7)
- #define EMBER_SECURITY_STATE_NOT_SET(xA8)
- #define EMBER_KEY_TABLE_INVALID_ADDRESS(xB3)
- #define EMBER_SECURITY_CONFIGURATION_INVALID(xB7)
- #define EMBER_TOO_SOON_FOR_SWITCH_KEY(xB8)
- #define EMBER_SIGNATURE_VERIFY_FAILURE(xB9)
- #define EMBER_KEY_NOTAUTHORIZED(xBB)
- #define EMBER_SECURITY_DATA_INVALID(xBD)

Miscellaneous Network Errors

- #define EMBER_NOT_JOINED(x93)
- #define EMBER_NETWORK_BUSY(xA1)
- #define EMBER_INVALID_ENDPOINT(xA3)
- #define EMBER_BINDING_HAS_CHANGED(xA4)
- #define EMBER_INSUFFICIENT_RANDOM_DATA(xA5)
- #define EMBER_SOURCE_ROUTE_FAILURE(xA9)
- #define EMBER_MANY_TO_ONE_ROUTE_FAILURE(xAA)

Miscellaneous Utility Errors

- #define EMBER_STACK_AND_HARDWARE_MISMATCH(xB0)
- #define EMBER_INDEX_OUT_OF_RANGE(xB1)
- #define EMBER_TABLE_FULL(xB4)
- #define EMBER_TABLE_ENTRY_ERASED(xB6)
- #define EMBER_LIBRARY_NOT_PRESENT(xB5)
- #define EMBER_OPERATION_IN_PROGRESS(xBA)
- #define EMBER_TRUST_CENTER_EUI_HAS_CHANGED(xBC)

ZigBee RF4CE specific errors.

- #define EMBER_NO_RESPONSE(xC0)
- #define EMBER_DUPLICATE_ENTRY(xC1)
- #define EMBER_NOT_PERMITTED(xC2)
- #define EMBER_DISCOVERY_TIMEOUT(xC3)
- #define EMBER_DISCOVERY_ERROR(xC4)
- #define EMBER_SECURITY_TIMEOUT(xC5)
- #define EMBER_SECURITY_FAILURE(xC6)

Application Errors

These error codes are available for application use.

- #define EMBER_APPLICATION_ERROR_0(xF0)
- #define EMBER_APPLICATION_ERROR_1(xF1)
- #define EMBER_APPLICATION_ERROR_2(xF2)
- #define EMBER_APPLICATION_ERROR_3(xF3)
- #define EMBER_APPLICATION_ERROR_4(xF4)
- #define EMBER_APPLICATION_ERROR_5(xF5)
- #define EMBER_APPLICATION_ERROR_6(xF6)
- #define EMBER_APPLICATION_ERROR_7(xF7)
- #define EMBER_APPLICATION_ERROR_8(xF8)
- #define EMBER_APPLICATION_ERROR_9(xF9)
- #define EMBER_APPLICATION_ERROR_10(xFA)
- #define EMBER_APPLICATION_ERROR_11(xFB)
- #define EMBER_APPLICATION_ERROR_12(xFC)
- #define EMBER_APPLICATION_ERROR_13(xFD)
- #define EMBER_APPLICATION_ERROR_14(xFE)
- #define EMBER_APPLICATION_ERROR_15(xFF)

8.73.1 Detailed Description

Return-code definitions for EmberZNet stack API functions. See [Status Codes](#) for documentation.

Definition in file [error-def.h](#).

8.74 error-def.h

```

00001
00038
00039 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00040
00043 #define EMBER_SUCCESS(0x00)
00044 #else
00045 DEFINE_ERROR(SUCCESS, 0)
00046 #endif //DOXYGEN_SHOULD_SKIP_THIS
00047
00048
00049 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00050
00053 #define EMBER_ERR_FATAL(0x01)
00054 #else
00055 DEFINE_ERROR(ERR_FATAL, 0x01)
00056 #endif //DOXYGEN_SHOULD_SKIP_THIS
00057
00058
00059 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00060
00063 #define EMBER_BAD_ARGUMENT(0x02)
00064 #else
00065 DEFINE_ERROR(BAD_ARGUMENT, 0x02)
00066 #endif //DOXYGEN_SHOULD_SKIP_THIS
00067
00068
00069 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00070
00073 #define EMBER_NOT_FOUND(0x03)
00074 #else
00075 DEFINE_ERROR(NOT_FOUND, 0x03)

```

```

00076 #endif //DOXYGEN_SHOULD_SKIP_THIS
00077
00078
00079 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00080
00084 #define EMBER_EEPROM_MFG_STACK_VERSION_MISMATCH(0x04)
00085 #else
00086 DEFINE_ERROR(EEPROM_MFG_STACK_VERSION_MISMATCH, 0x04)
00087 #endif //DOXYGEN_SHOULD_SKIP_THIS
00088
00089
00090 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00091
00095 #define EMBER_INCOMPATIBLE_STATIC_MEMORY_DEFINITIONS(0x05)
00096 #else
00097 DEFINE_ERROR(INCOMPATIBLE_STATIC_MEMORY_DEFINITIONS, 0x05)
00098 #endif //DOXYGEN_SHOULD_SKIP_THIS
00099
00100
00101 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00102
00106 #define EMBER_EEPROM_MFG_VERSION_MISMATCH(0x06)
00107 #else
00108 DEFINE_ERROR(EEPROM_MFG_VERSION_MISMATCH, 0x06)
00109 #endif //DOXYGEN_SHOULD_SKIP_THIS
00110
00111
00112 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00113
00117 #define EMBER_EEPROM_STACK_VERSION_MISMATCH(0x07)
00118 #else
00119 DEFINE_ERROR(EEPROM_STACK_VERSION_MISMATCH, 0x07)
00120 #endif //DOXYGEN_SHOULD_SKIP_THIS
00121
00123
00124
00129
00130 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00131
00134 #define EMBER_NO_BUFFERS(0x18)
00135 #else
00136 DEFINE_ERROR(NO_BUFFERS, 0x18)
00137 #endif //DOXYGEN_SHOULD_SKIP_THIS
00138
00140
00145
00146 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00147
00150 #define EMBER_SERIAL_INVALID_BAUD_RATE(0x20)
00151 #else
00152 DEFINE_ERROR(SERIAL_INVALID_BAUD_RATE, 0x20)
00153 #endif //DOXYGEN_SHOULD_SKIP_THIS
00154
00155
00156 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00157
00160 #define EMBER_SERIAL_INVALID_PORT(0x21)
00161 #else
00162 DEFINE_ERROR(SERIAL_INVALID_PORT, 0x21)
00163 #endif //DOXYGEN_SHOULD_SKIP_THIS
00164
00165
00166 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00167
00170 #define EMBER_SERIAL_TX_OVERFLOW(0x22)
00171 #else
00172 DEFINE_ERROR(SERIAL_TX_OVERFLOW, 0x22)
00173 #endif //DOXYGEN_SHOULD_SKIP_THIS
00174
00175
00176 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00177
00181 #define EMBER_SERIAL_RX_OVERFLOW(0x23)
00182 #else
00183 DEFINE_ERROR(SERIAL_RX_OVERFLOW, 0x23)
00184 #endif //DOXYGEN_SHOULD_SKIP_THIS
00185
00186
00187 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00188

```

```

00191 #define EMBER_SERIAL_RX_FRAME_ERROR(0x24)
00192 #else
00193 DEFINE_ERROR(SERIAL_RX_FRAME_ERROR, 0x24)
00194 #endif //DOXYGEN_SHOULD_SKIP_THIS
00195
00196
00197 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00198
00201 #define EMBER_SERIAL_RX_PARITY_ERROR(0x25)
00202 #else
00203 DEFINE_ERROR(SERIAL_RX_PARITY_ERROR, 0x25)
00204 #endif //DOXYGEN_SHOULD_SKIP_THIS
00205
00206
00207 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00208
00211 #define EMBER_SERIAL_RX_EMPTY(0x26)
00212 #else
00213 DEFINE_ERROR(SERIAL_RX_EMPTY, 0x26)
00214 #endif //DOXYGEN_SHOULD_SKIP_THIS
00215
00216
00217 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00218
00222 #define EMBER_SERIAL_RX_OVERRUN_ERROR(0x27)
00223 #else
00224 DEFINE_ERROR(SERIAL_RX_OVERRUN_ERROR, 0x27)
00225 #endif //DOXYGEN_SHOULD_SKIP_THIS
00226
00228
00233
00234 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00235
00238 #define EMBER_MAC_TRANSMIT_QUEUE_FULL(0x39)
00239 #else
00240 // Internal
00241 DEFINE_ERROR(MAC_TRANSMIT_QUEUE_FULL, 0x39)
00242 #endif //DOXYGEN_SHOULD_SKIP_THIS
00243
00244
00245 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00246
00249 #define EMBER_MAC_UNKNOWN_HEADER_TYPE(0x3A)
00250 #else
00251 DEFINE_ERROR(MAC_UNKNOWN_HEADER_TYPE, 0x3A)
00252 #endif //DOXYGEN_SHOULD_SKIP_THIS
00253
00254 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00255
00258 #define EMBER_MAC_ACK_HEADER_TYPE(0x3B)
00259 #else
00260 DEFINE_ERROR(MAC_ACK_HEADER_TYPE, 0x3B)
00261 #endif //DOXYGEN_SHOULD_SKIP_THIS
00262
00263
00264
00265 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00266
00269 #define EMBER_MAC_SCANNING(0x3D)
00270 #else
00271 DEFINE_ERROR(MAC_SCANNING, 0x3D)
00272 #endif //DOXYGEN_SHOULD_SKIP_THIS
00273
00274
00275 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00276
00279 #define EMBER_MAC_NO_DATA(0x31)
00280 #else
00281 DEFINE_ERROR(MAC_NO_DATA, 0x31)
00282 #endif //DOXYGEN_SHOULD_SKIP_THIS
00283
00284
00285 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00286
00289 #define EMBER_MAC_JOINED_NETWORK(0x32)
00290 #else
00291 DEFINE_ERROR(MAC_JOINED_NETWORK, 0x32)
00292 #endif //DOXYGEN_SHOULD_SKIP_THIS
00293
00294

```

```

00295 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00296
00300 #define EMBER_MAC_BAD_SCAN_DURATION(0x33)
00301 #else
00302 DEFINE_ERROR(MAC_BAD_SCAN_DURATION, 0x33)
00303 #endif //DOXYGEN_SHOULD_SKIP_THIS
00304
00305
00306 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00307
00310 #define EMBER_MAC_INCORRECT_SCAN_TYPE(0x34)
00311 #else
00312 DEFINE_ERROR(MAC_INCORRECT_SCAN_TYPE, 0x34)
00313 #endif //DOXYGEN_SHOULD_SKIP_THIS
00314
00315
00316 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00317
00320 #define EMBER_MAC_INVALID_CHANNEL_MASK(0x35)
00321 #else
00322 DEFINE_ERROR(MAC_INVALID_CHANNEL_MASK, 0x35)
00323 #endif //DOXYGEN_SHOULD_SKIP_THIS
00324
00325
00326 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00327
00331 #define EMBER_MAC_COMMAND_TRANSMIT_FAILURE(0x36)
00332 #else
00333 DEFINE_ERROR(MAC_COMMAND_TRANSMIT_FAILURE, 0x36)
00334 #endif //DOXYGEN_SHOULD_SKIP_THIS
00335
00336
00337 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00338
00342 #define EMBER_MAC_NO_ACK RECEIVED(0x40)
00343 #else
00344 DEFINE_ERROR(MAC_NO_ACK RECEIVED, 0x40)
00345 #endif //DOXYGEN_SHOULD_SKIP_THIS
00346
00347
00348 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00349
00353 #define EMBER_MAC_RADIO_NETWORK_SWITCH_FAILED(0x41)
00354 #else
00355 DEFINE_ERROR(MAC_RADIO_NETWORK_SWITCH_FAILED, 0x41)
00356 #endif //DOXYGEN_SHOULD_SKIP_THIS
00357
00358
00359 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00360
00363 #define EMBER_MAC_INDIRECT_TIMEOUT(0x42)
00364 #else
00365 DEFINE_ERROR(MAC_INDIRECT_TIMEOUT, 0x42)
00366 #endif //DOXYGEN_SHOULD_SKIP_THIS
00367
00369
00370
00375
00376
00377 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00378
00386 #define EMBER_SIM_EEPROM_ERASE_PAGE_GREEN(0x43)
00387 #else
00388 DEFINE_ERROR(SIM EEPROM_ERASE_PAGE_GREEN, 0x43)
00389 #endif //DOXYGEN_SHOULD_SKIP_THIS
00390
00391
00392 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00393
00402 #define EMBER_SIM_EEPROM_ERASE_PAGE_RED(0x44)
00403 #else
00404 DEFINE_ERROR(SIM EEPROM_ERASE_PAGE_RED, 0x44)
00405 #endif //DOXYGEN_SHOULD_SKIP_THIS
00406
00407
00408 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00409
00417 #define EMBER_SIM_EEPROM_FULL(0x45)
00418 #else
00419 DEFINE_ERROR(SIM EEPROM_FULL, 0x45)

```

```

00420 #endif //DOXYGEN_SHOULD_SKIP_THIS
00421
00422
00423 // Errors 46 and 47 are now defined below in the
00424 // flash error block (was attempting to prevent renumbering)
00425
00426
00427 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00428
00429 #define EMBER_SIM_EEPROM_INIT_1_FAILED(0x48)
00430 #else
00431 DEFINE_ERROR(SIM EEPROM INIT 1 FAILED, 0x48)
00432 #endif //DOXYGEN_SHOULD_SKIP_THIS
00433
00434
00435 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00436
00437 #define EMBER_SIM_EEPROM_INIT_2_FAILED(0x49)
00438 #else
00439 DEFINE_ERROR(SIM EEPROM INIT 2 FAILED, 0x49)
00440 #endif //DOXYGEN_SHOULD_SKIP_THIS
00441
00442
00443 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00444
00445 #define EMBER_SIM_EEPROM_INIT_3_FAILED(0x4A)
00446 #else
00447 DEFINE_ERROR(SIM EEPROM INIT 3 FAILED, 0x4A)
00448 #endif //DOXYGEN_SHOULD_SKIP_THIS
00449
00450
00451 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00452
00453
00454 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00455
00456 #define EMBER_SIM_EEPROM_REPAIRING(0x4D)
00457 #else
00458 DEFINE_ERROR(SIM EEPROM REPAIRING, 0x4D)
00459 #endif //DOXYGEN_SHOULD_SKIP_THIS
00460
00461
00462
00463
00464 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00465
00466
00467 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00468
00469
00470 #define EMBER_ERR_FLASH_WRITE_INHIBITED(0x46)
00471 #else
00472 DEFINE_ERROR(ERR FLASH WRITE INHIBITED, 0x46)
00473 #endif //DOXYGEN_SHOULD_SKIP_THIS
00474
00475
00476
00477 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00478
00479 #define EMBER_ERR_FLASH_VERIFY_FAILED(0x47)
00480 #else
00481 DEFINE_ERROR(ERR FLASH VERIFY FAILED, 0x47)
00482 #endif //DOXYGEN_SHOULD_SKIP_THIS
00483
00484
00485
00486
00487
00488
00489
00490
00491 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00492
00493
00494
00495 #define EMBER_ERR_FLASH_PROG_FAIL(0x4B)
00496 #else
00497 DEFINE_ERROR(ERR FLASH PROG FAIL, 0x4B)
00498 #endif //DOXYGEN_SHOULD_SKIP_THIS
00499
00500
00501
00502
00503
00504
00505
00506
00507
00508
00509
00510
00511
00512
00513
00514
00515
00516
00517
00518
00519
00520
00521
00522
00523
00524
00525
00526
00527
00528
00529
00530
00531
00532
00533
00534
00535
00536
00537
00538
00539
00540
00541
00542
00543
00544
00545
00546
00547
00548
00549
00550
00551
00552
00553
00554
00555
00556
00557
00558
00559
00560 #else

```

```

00561 #define _ERROR_(ERR_BOOTLOADER_TRAP_TABLE_BAD, 0x58)
00562 #endif //DOXYGEN_SHOULD_SKIP_THIS
00563
00564
00565 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00566
00570 #define EMBER_ERR_BOOTLOADER_TRAP_UNKNOWN (0x59)
00571 #else
00572 #define _ERROR_(ERR_BOOTLOADER_TRAP_UNKNOWN, 0x59)
00573 #endif //DOXYGEN_SHOULD_SKIP_THIS
00574
00575
00576 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00577
00581 #define EMBER_ERR_BOOTLOADER_NO_IMAGE (0x05A)
00582 #else
00583 #define _ERROR_(ERR_BOOTLOADER_NO_IMAGE, 0x5A)
00584 #endif //DOXYGEN_SHOULD_SKIP_THIS
00585
00587
00588
00593
00594 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00595
00599 #define EMBER_DELIVERY_FAILED (0x66)
00600 #else
00601 #define _ERROR_(DELIVERY_FAILED, 0x66)
00602 #endif //DOXYGEN_SHOULD_SKIP_THIS
00603
00604
00605 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00606
00609 #define EMBER_BINDING_INDEX_OUT_OF_RANGE (0x69)
00610 #else
00611 #define _ERROR_(BINDING_INDEX_OUT_OF_RANGE, 0x69)
00612 #endif //DOXYGEN_SHOULD_SKIP_THIS
00613
00614
00615 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00616
00620 #define EMBER_ADDRESS_TABLE_INDEX_OUT_OF_RANGE (0x6A)
00621 #else
00622 #define _ERROR_(ADDRESS_TABLE_INDEX_OUT_OF_RANGE, 0x6A)
00623 #endif //DOXYGEN_SHOULD_SKIP_THIS
00624
00625
00626 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00627
00630 #define EMBER_INVALID_BINDING_INDEX (0x6C)
00631 #else
00632 #define _ERROR_(INVALID_BINDING_INDEX, 0x6C)
00633 #endif //DOXYGEN_SHOULD_SKIP_THIS
00634
00635
00636 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00637
00641 #define EMBER_INVALID_CALL (0x70)
00642 #else
00643 #define _ERROR_(INVALID_CALL, 0x70)
00644 #endif //DOXYGEN_SHOULD_SKIP_THIS
00645
00646
00647 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00648
00651 #define EMBER_COST_NOT_KNOWN (0x71)
00652 #else
00653 #define _ERROR_(COST_NOT_KNOWN, 0x71)
00654 #endif //DOXYGEN_SHOULD_SKIP_THIS
00655
00656
00657 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00658
00662 #define EMBER_MAX_MESSAGE_LIMIT_REACHED (0x72)
00663 #else
00664 #define _ERROR_(MAX_MESSAGE_LIMIT_REACHED, 0x72)
00665 #endif //DOXYGEN_SHOULD_SKIP_THIS
00666
00667 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00668
00672 #define EMBER_MESSAGE_TOO_LONG (0x74)

```

```

00673 #else
00674 #define _ERROR (MESSAGE_TOO_LONG, 0x74)
00675 #endif //DOXYGEN_SHOULD_SKIP_THIS
00676
00677
00678 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00679
00683 #define EMBER_BINDING_IS_ACTIVE (0x75)
00684 #else
00685 #define _ERROR (BINDING_IS_ACTIVE, 0x75)
00686 #endif //DOXYGEN_SHOULD_SKIP_THIS
00687
00688 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00689
00693 #define EMBER_ADDRESS_TABLE_ENTRY_IS_ACTIVE (0x76)
00694 #else
00695 #define _ERROR (ADDRESS_TABLE_ENTRY_IS_ACTIVE, 0x76)
00696 #endif //DOXYGEN_SHOULD_SKIP_THIS
00697
00699 //
00700
00705
00706 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00707
00710 #define EMBER_MATCH (0x78)
00711 #else
00712 #define _ERROR (MATCH, 0x78)
00713 #endif //DOXYGEN_SHOULD_SKIP_THIS
00714 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00715
00718 #define EMBER_DROP_FRAME (0x79)
00719 #else
00720 #define _ERROR (DROP_FRAME, 0x79)
00721 #endif //DOXYGEN_SHOULD_SKIP_THIS
00722
00725 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00726 #define EMBER_PASS_UNPROCESSED (0x7A)
00727 #else
00728 #define _ERROR (PASS_UNPROCESSED, 0x7A)
00729 #endif //DOXYGEN_SHOULD_SKIP_THIS
00730
00733 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00734 #define EMBER_TX_THEN_DROP (0x7B)
00735 #else
00736 #define _ERROR (TX_THEN_DROP, 0x7B)
00737 #endif //DOXYGEN_SHOULD_SKIP_THIS
00738
00741 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00742 #define EMBER_NO_SECURITY (0x7C)
00743 #else
00744 #define _ERROR (NO_SECURITY, 0x7C)
00745 #endif //DOXYGEN_SHOULD_SKIP_THIS
00746
00749 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00750 #define EMBER_COUNTER_FAILURE (0x7D)
00751 #else
00752 #define _ERROR (COUNTER_FAILURE, 0x7D)
00753 #endif //DOXYGEN_SHOULD_SKIP_THIS
00754
00757 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00758 #define EMBER_AUTH_FAILURE (0x7E)
00759 #else
00760 #define _ERROR (AUTH_FAILURE, 0x7E)
00761 #endif //DOXYGEN_SHOULD_SKIP_THIS
00762
00765 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00766 #define EMBER_UNPROCESSED (0x7F)
00767 #else
00768 #define _ERROR (UNPROCESSED, 0x7F)
00769 #endif //DOXYGEN_SHOULD_SKIP_THIS
00770
00772 //
00773
00778
00779
00780 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00781
00784 #define EMBER_ADC_CONVERSION_DONE (0x80)
00785 #else
00786 #define _ERROR (ADC_CONVERSION_DONE, 0x80)

```

```

00787 #endif //DOXYGEN_SHOULD_SKIP_THIS
00788
00789
00790 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00791
00795 #define EMBER_ADC_CONVERSION_BUSY(0x81)
00796 #else
00797 DEFINE_ERROR(ADC_CONVERSION_BUSY, 0x81)
00798#endif //DOXYGEN_SHOULD_SKIP_THIS
00799
00800
00801 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00802
00806 #define EMBER_ADC_CONVERSION_DEFERRED(0x82)
00807 #else
00808 DEFINE_ERROR(ADC_CONVERSION_DEFERRED, 0x82)
00809#endif //DOXYGEN_SHOULD_SKIP_THIS
00810
00811
00812 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00813
00816 #define EMBER_ADC_NO_CONVERSION_PENDING(0x84)
00817 #else
00818 DEFINE_ERROR(ADC_NO_CONVERSION_PENDING, 0x84)
00819#endif //DOXYGEN_SHOULD_SKIP_THIS
00820
00821
00822 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00823
00827 #define EMBER_SLEEP_INTERRUPTED(0x85)
00828 #else
00829 DEFINE_ERROR(SLEEP_INTERRUPTED, 0x85)
00830#endif //DOXYGEN_SHOULD_SKIP_THIS
00831
00833
00838
00839
00840 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00841
00844 #define EMBER_PHY_TX_UNDERFLOW(0x88)
00845 #else
00846 DEFINE_ERROR(PHY_TX_UNDERFLOW, 0x88)
00847#endif //DOXYGEN_SHOULD_SKIP_THIS
00848
00849
00850 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00851
00854 #define EMBER_PHY_TX_INCOMPLETE(0x89)
00855 #else
00856 DEFINE_ERROR(PHY_TX_INCOMPLETE, 0x89)
00857#endif //DOXYGEN_SHOULD_SKIP_THIS
00858
00859
00860 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00861
00864 #define EMBER_PHY_INVALID_CHANNEL(0x8A)
00865 #else
00866 DEFINE_ERROR(PHY_INVALID_CHANNEL, 0x8A)
00867#endif //DOXYGEN_SHOULD_SKIP_THIS
00868
00869
00870 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00871
00874 #define EMBER_PHY_INVALID_POWER(0x8B)
00875 #else
00876 DEFINE_ERROR(PHY_INVALID_POWER, 0x8B)
00877#endif //DOXYGEN_SHOULD_SKIP_THIS
00878
00879
00880 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00881
00885 #define EMBER_PHY_TX_BUSY(0x8C)
00886 #else
00887 DEFINE_ERROR(PHY_TX_BUSY, 0x8C)
00888#endif //DOXYGEN_SHOULD_SKIP_THIS
00889
00890
00891 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00892
00896 #define EMBER_PHY_TX_CCA_FAIL(0x8D)

```

```

00897 #else
00898 DEFINE_ERROR(PHY_TX_CCA_FAIL, 0x8D)
00899 #endif //DOXYGEN_SHOULD_SKIP_THIS
00900
00901
00902 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00903
00907 #define EMBER_PHY_OSCILLATOR_CHECK_FAILED(0x8E)
00908 #else
00909 DEFINE_ERROR(PHY_OSCILLATOR_CHECK_FAILED, 0x8E)
00910 #endif //DOXYGEN_SHOULD_SKIP_THIS
00911
00912
00913 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00914
00917 #define EMBER_PHY_ACK RECEIVED(0x8F)
00918 #else
00919 DEFINE_ERROR(PHY_ACK RECEIVED, 0x8F)
00920 #endif //DOXYGEN_SHOULD_SKIP_THIS
00921
00923
00929
00930
00931 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00932
00936 #define EMBER_NETWORK_UP(0x90)
00937 #else
00938 DEFINE_ERROR(NETWORK_UP, 0x90)
00939 #endif //DOXYGEN_SHOULD_SKIP_THIS
00940
00941
00942 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00943
00946 #define EMBER_NETWORK_DOWN(0x91)
00947 #else
00948 DEFINE_ERROR(NETWORK_DOWN, 0x91)
00949 #endif //DOXYGEN_SHOULD_SKIP_THIS
00950
00951
00952 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00953
00956 #define EMBER_JOIN FAILED(0x94)
00957 #else
00958 DEFINE_ERROR(JOIN FAILED, 0x94)
00959 #endif //DOXYGEN_SHOULD_SKIP_THIS
00960
00961
00962 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00963
00967 #define EMBER_MOVE FAILED(0x96)
00968 #else
00969 DEFINE_ERROR(MOVE FAILED, 0x96)
00970 #endif //DOXYGEN_SHOULD_SKIP_THIS
00971
00972
00973 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00974
00979 #define EMBER_CANNOT JOIN AS ROUTER(0x98)
00980 #else
00981 DEFINE_ERROR(CANNOT JOIN AS ROUTER, 0x98)
00982 #endif //DOXYGEN_SHOULD_SKIP_THIS
00983
00984
00985 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00986
00989 #define EMBER_NODE_ID_CHANGED(0x99)
00990 #else
00991 DEFINE_ERROR(NODE_ID_CHANGED, 0x99)
00992 #endif
00993
00994
00995 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00996
00999 #define EMBER_PAN_ID_CHANGED(0x9A)
01000 #else
01001 DEFINE_ERROR(PAN_ID_CHANGED, 0x9A)
01002 #endif
01003
01004 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01005

```

```

01007 #define EMBER_CHANNEL_CHANGED (0x9B)
01008 #else
01009     DEFINE_ERROR(CHANNEL_CHANGED, 0x9B)
01010 #endif
01011
01012 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01013
01016 #define EMBER_NO_BEACONS (0xAB)
01017 #else
01018     DEFINE_ERROR(NO_BEACONS, 0xAB)
01019 #endif
01020
01021
01022 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01023
01027 #define EMBER RECEIVED_KEY_IN_THE_CLEAR (0xAC)
01028 #else
01029     DEFINE_ERROR(RECEIVED_KEY_IN_THE_CLEAR, 0xAC)
01030 #endif
01031
01032
01033 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01034
01037 #define EMBER_NO_NETWORK_KEY_RECEIVED (0xAD)
01038 #else
01039     DEFINE_ERROR(NO_NETWORK_KEY_RECEIVED, 0xAD)
01040 #endif
01041
01042
01043 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01044
01047 #define EMBER_NO_LINK_KEY_RECEIVED (0xAE)
01048 #else
01049     DEFINE_ERROR(NO_LINK_KEY_RECEIVED, 0xAE)
01050 #endif
01051
01052
01053 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01054
01058 #define EMBER_PRECONFIGURED_KEY_REQUIRED (0xAF)
01059 #else
01060     DEFINE_ERROR(PRECONFIGURED_KEY_REQUIRED, 0xAF)
01061 #endif
01062
01063
01065
01069 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01070
01074 #define EMBER_KEY_INVALID (0xB2)
01075 #else
01076     DEFINE_ERROR(KEY_INVALID, 0xB2)
01077 #endif // DOXYGEN_SHOULD_SKIP_THIS
01078
01079 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01080
01084 #define EMBER_INVALID_SECURITY_LEVEL (0x95)
01085 #else
01086     DEFINE_ERROR(INVALID_SECURITY_LEVEL, 0x95)
01087 #endif //DOXYGEN_SHOULD_SKIP_THIS
01088
01089 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01090
01098 #define EMBER_APS_ENCRYPTION_ERROR (0xA6)
01099 #else
01100     DEFINE_ERROR(APS_ENCRYPTION_ERROR, 0xA6)
01101 #endif //DOXYGEN_SHOULD_SKIP_THIS
01102
01103 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01104
01107 #define EMBER_TRUST_CENTER_MASTER_KEY_NOT_SET (0xA7)
01108 #else
01109     DEFINE_ERROR(TRUST_CENTER_MASTER_KEY_NOT_SET, 0xA7)
01110 #endif //DOXYGEN_SHOULD_SKIP_THIS
01111
01112 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01113
01116 #define EMBER_SECURITY_STATE_NOT_SET (0xA8)
01117 #else
01118     DEFINE_ERROR(SECURITY_STATE_NOT_SET, 0xA8)
01119 #endif //DOXYGEN_SHOULD_SKIP_THIS

```

```

01120
01121 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01122
01129 #define EMBER_KEY_TABLE_INVALID_ADDRESS(0xB3)
01130 #else
01131 DEFINE_ERROR(KEY_TABLE_INVALID_ADDRESS, 0xB3)
01132 #endif //DOXYGEN_SHOULD_SKIP_THIS
01133
01134 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01135
01138 #define EMBER_SECURITY_CONFIGURATION_INVALID(0xB7)
01139 #else
01140 DEFINE_ERROR(SECRETARY_CONFIGURATION_INVALID, 0xB7)
01141 #endif //DOXYGEN_SHOULD_SKIP_THIS
01142
01143 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01144
01149 #define EMBER_TOO_SOON_FOR_SWITCH_KEY(0xB8)
01150 #else
01151     DEFINE_ERROR(TOO_SOON_FOR_SWITCH_KEY, 0xB8)
01152 #endif
01153
01154 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01155
01158 #define EMBER_SIGNATURE_VERIFY_FAILURE(0xB9)
01159 #else
01160     DEFINE_ERROR(SIGNATURE_VERIFY_FAILURE, 0xB9)
01161 #endif
01162
01163 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01164
01170 #define EMBER_KEY_NOTAUTHORIZED(0xBB)
01171 #else
01172     DEFINE_ERROR(KEY_NOTAUTHORIZED, 0xBB)
01173 #endif
01174
01175
01176 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01177
01180 #define EMBER_SECURITY_DATA_INVALID(0xBD)
01181 #else
01182     DEFINE_ERROR(SECRETARY_DATA_INVALID, 0xBD)
01183 #endif
01184
01186
01187
01192
01193
01194 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01195
01198 #define EMBER_NOT_JOINED(0x93)
01199 #else
01200 DEFINE_ERROR(NOT_JOINED, 0x93)
01201 #endif //DOXYGEN_SHOULD_SKIP_THIS
01202
01203 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01204
01208 #define EMBER_NETWORK_BUSY(0xA1)
01209 #else
01210 DEFINE_ERROR(NETWORK_BUSY, 0xA1)
01211 #endif //DOXYGEN_SHOULD_SKIP_THIS
01212
01213
01214 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01215
01219 #define EMBER_INVALID_ENDPOINT(0xA3)
01220 #else
01221 DEFINE_ERROR(INVALID_ENDPOINT, 0xA3)
01222 #endif //DOXYGEN_SHOULD_SKIP_THIS
01223
01224
01225 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01226
01230 #define EMBER_BINDING_HAS_CHANGED(0xA4)
01231 #else
01232 DEFINE_ERROR(BINDING_HAS_CHANGED, 0xA4)
01233 #endif //DOXYGEN_SHOULD_SKIP_THIS
01234
01235 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01236

```

```

01240 #define EMBER_INSUFFICIENT_RANDOM_DATA(0xA5)
01241 #else
01242     DEFINE_ERROR(INSUFFICIENT_RANDOM_DATA, 0xA5)
01243 #endif //DOXYGEN_SHOULD_SKIP_THIS
01244
01245
01246 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01247
01248 #define EMBER_SOURCE_ROUTE_FAILURE(0xA9)
01249 #else
01250     DEFINE_ERROR(SOURCE_ROUTE_FAILURE, 0xA9)
01251 #endif
01252
01253 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01254
01255 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01256
01257 #define EMBER_MANY_TO_ONE_ROUTE_FAILURE(0xAA)
01258 #else
01259     DEFINE_ERROR(MANY_TO_ONE_ROUTE_FAILURE, 0xAA)
01260 #endif
01261
01262 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01263
01264 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01265
01266
01267
01268
01269
01270
01271
01272
01273
01274
01275 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01276
01277 #define EMBER_STACK_AND_HARDWARE_MISMATCH(0xB0)
01278 #else
01279     DEFINE_ERROR(STACK_AND_HARDWARE_MISMATCH, 0xB0)
01280 #endif //DOXYGEN_SHOULD_SKIP_THIS
01281
01282 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01283
01284 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01285
01286 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01287
01288 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01289
01290 #define EMBER_INDEX_OUT_OF_RANGE(0xB1)
01291 #else
01292     DEFINE_ERROR(INDEX_OUT_OF_RANGE, 0xB1)
01293 #endif
01294
01295 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01296
01297
01298 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01299
01300 #define EMBER_TABLE_FULL(0xB4)
01301 #else
01302     DEFINE_ERROR(TABLE_FULL, 0xB4)
01303 #endif //DOXYGEN_SHOULD_SKIP_THIS
01304
01305 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01306
01307 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01308
01309 #define EMBER_TABLE_ENTRY_ERASED(0xB6)
01310 #else
01311     DEFINE_ERROR(TABLE_ENTRY_ERASED, 0xB6)
01312 #endif
01313
01314 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01315
01316
01317 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01318
01319 #define EMBER_LIBRARY_NOT_PRESENT(0xB5)
01320 #else
01321     DEFINE_ERROR(LIBRARY_NOT_PRESENT, 0xB5)
01322 #endif
01323
01324 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01325
01326
01327 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01328
01329 #define EMBER_OPERATION_IN_PROGRESS(0xBA)
01330 #else
01331     DEFINE_ERROR(OPERATION_IN_PROGRESS, 0xBA)
01332 #endif
01333
01334 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01335
01336
01337 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01338
01339 #define EMBER_TRUST_CENTER_EUI_HAS_CHANGED(0xBC)
01340 #else
01341     DEFINE_ERROR(TRUST_CENTER_EUI_HAS_CHANGED, 0xBC)
01342 #endif
01343
01344 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01345
01346 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01347
01348
01349
01350
01351
01352
01353 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01354
01355 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01356
01357 #define EMBER_NO_RESPONSE(0xC0)
01358 #else
01359 #endif
01360
01361 #endif

```

```

01362     DEFINE_ERROR(NO_RESPONSE, 0xC0)
01363 #endif
01364
01365 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01366
01370 #define EMBER_DUPLICATE_ENTRY(0xC1)
01371 #else
01372     DEFINE_ERROR(DUPLICATE_ENTRY, 0xC1)
01373 #endif
01374
01375 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01376
01381 #define EMBER_NOT_PERMITTED(0xC2)
01382 #else
01383     DEFINE_ERROR(NOT_PERMITTED, 0xC2)
01384 #endif
01385
01386 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01387
01390 #define EMBER_DISCOVERY_TIMEOUT(0xC3)
01391 #else
01392     DEFINE_ERROR(DISCOVERY_TIMEOUT, 0xC3)
01393 #endif
01394
01395
01396 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01397
01401 #define EMBER_DISCOVERY_ERROR(0xC4)
01402 #else
01403     DEFINE_ERROR(DISCOVERY_ERROR, 0xC4)
01404 #endif
01405
01406
01407 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01408
01412 #define EMBER_SECURITY_TIMEOUT(0xC5)
01413 #else
01414     DEFINE_ERROR(SECURITY_TIMEOUT, 0xC5)
01415 #endif
01416
01417
01418 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01419
01422 #define EMBER_SECURITY_FAILURE(0xC6)
01423 #else
01424     DEFINE_ERROR(SECURITY_FAILURE, 0xC6)
01425 #endif
01426
01428
01434
01435 #ifdef DOXYGEN_SHOULD_SKIP_THIS
01436
01440 #define EMBER_APPLICATION_ERROR_0(0xF0)
01441 #define EMBER_APPLICATION_ERROR_1(0xF1)
01442 #define EMBER_APPLICATION_ERROR_2(0xF2)
01443 #define EMBER_APPLICATION_ERROR_3(0xF3)
01444 #define EMBER_APPLICATION_ERROR_4(0xF4)
01445 #define EMBER_APPLICATION_ERROR_5(0xF5)
01446 #define EMBER_APPLICATION_ERROR_6(0xF6)
01447 #define EMBER_APPLICATION_ERROR_7(0xF7)
01448 #define EMBER_APPLICATION_ERROR_8(0xF8)
01449 #define EMBER_APPLICATION_ERROR_9(0xF9)
01450 #define EMBER_APPLICATION_ERROR_10(0xFA)
01451 #define EMBER_APPLICATION_ERROR_11(0xFB)
01452 #define EMBER_APPLICATION_ERROR_12(0xFC)
01453 #define EMBER_APPLICATION_ERROR_13(0xFD)
01454 #define EMBER_APPLICATION_ERROR_14(0xFE)
01455 #define EMBER_APPLICATION_ERROR_15(0xFF)
01456 #else
01457     DEFINE_ERROR(APPLICATION_ERROR_0, 0xF0)
01458     DEFINE_ERROR(APPLICATION_ERROR_1, 0xF1)
01459     DEFINE_ERROR(APPLICATION_ERROR_2, 0xF2)
01460     DEFINE_ERROR(APPLICATION_ERROR_3, 0xF3)
01461     DEFINE_ERROR(APPLICATION_ERROR_4, 0xF4)
01462     DEFINE_ERROR(APPLICATION_ERROR_5, 0xF5)
01463     DEFINE_ERROR(APPLICATION_ERROR_6, 0xF6)
01464     DEFINE_ERROR(APPLICATION_ERROR_7, 0xF7)
01465     DEFINE_ERROR(APPLICATION_ERROR_8, 0xF8)
01466     DEFINE_ERROR(APPLICATION_ERROR_9, 0xF9)
01467     DEFINE_ERROR(APPLICATION_ERROR_10, 0xFA)

```

```

01468 DEFINE_ERROR( APPLICATION_ERROR_11, 0xFB)
01469 DEFINE_ERROR( APPLICATION_ERROR_12, 0xFC)
01470 DEFINE_ERROR( APPLICATION_ERROR_13, 0xFD)
01471 DEFINE_ERROR( APPLICATION_ERROR_14, 0xFE)
01472 DEFINE_ERROR( APPLICATION_ERROR_15, 0xFF)
01473 #endif //DOXYGEN_SHOULD_SKIP_THIS
01474
01476

```

8.75 error.h File Reference

Macros

- `#define __EMBERSTATUS_TYPE__`
- `#define DEFINE_ERROR(symbol, value)`

TypeDefs

- `typedef uint8_t EmberStatus`

Enumerations

- `enum { EMBER_ERROR_CODE_COUNT }`

8.75.1 Detailed Description

Return codes for Ember API functions and module definitions. See [Status Codes](#) for documentation.

Definition in file [error.h](#).

8.75.2 Macro Definition Documentation

8.75.2.1 `#define __EMBERSTATUS_TYPE__`

Return type for Ember functions.

Definition at line [18](#) of file [error.h](#).

8.75.3 TypeDef Documentation

8.75.3.1 `typedef uint8_t EmberStatus`

Definition at line [19](#) of file [error.h](#).

8.76 error.h

```

00001
00011 #ifndef __ERRORS_H__
00012 #define __ERRORS_H__
00013

```

```

00017 #ifndef __EMBERSTATUS_TYPE__
00018 #define __EMBERSTATUS_TYPE__
00019   typedef uint8_t EmberStatus;
00020 #endif //__EMBERSTATUS_TYPE__
00021
00035 #define DEFINE_ERROR(symbol, value) \
00036   EMBER_ ## symbol = value,
00037
00038
00039 enum {
00040 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00041 #include "include/error-def.h"
00042 #endif //DOXYGEN_SHOULD_SKIP_THIS
00043
00044   EMBER_ERROR_CODE_COUNT
00045
00046 };
00047
00048 };
00049
00050 #undef DEFINE_ERROR
00051
00052 #endif // __ERRORS_H__
00053

```

8.77 event.h File Reference

Macros

- #define __EVENT_H__
- #define emberEventControlSetInactive(control)
- #define emberEventControlGetActive(control)
- #define emberEventControlSetActive(control)
- #define EMBER_MAX_EVENT_CONTROL_DELAY_MS
- #define emberEventControlSetDelayMS(control, delay)
- #define EMBER_MAX_EVENT_CONTROL_DELAY_QS
- #define emberEventControlSetDelayQS(control, delay)
- #define EMBER_MAX_EVENT_CONTROL_DELAY_MINUTES
- #define emberEventControlSetDelayMinutes(control, delay)
- #define emberEventControlGetRemainingMS(control)
- #define emberTaskEnableIdling(allow)
- #define emberMarkTaskActive(taskid)

Functions

- void emEventControlSetActive (EmberEventControl *event)
- void emEventControlSetDelayMS (EmberEventControl *event, uint32_t delay)
- uint32_t emEventControlGetRemainingMS (EmberEventControl *event)
- void emberRunEvents (EmberEventData *events)
- void emberRunTask (EmberTaskId taskid)
- uint32_t emberMsToNextEvent (EmberEventData *events, uint32_t maxMs)
- uint32_t emberMsToNextEventExtended (EmberEventData *events, uint32_t maxMs, uint8_t *returnIndex)
- uint32_t emberMsToNextStackEvent (void)
- EmberTaskId emberTaskInit (EmberEventData *events)
- bool emberMarkTaskIdle (EmberTaskId taskid)
- void emTaskEnableIdling (bool allow)
- void emMarkTaskActive (EmberTaskId taskid)

8.77.1 Detailed Description

Scheduling events for future execution. See [Event Scheduling](#) for documentation.

Definition in file [event.h](#).

8.78 event.h

```

00001
00009 #if defined(XAP2B) || defined(EZSP_HOST)
0010  // The xap2b platform and hosts do not support processor idling
0011  #define EMBER_NO_IDLE_SUPPORT
0012 #endif
0013
0019 // Controlling events
0020
0021 // Possible event status values. Having zero as the 'inactive' value
0022 // causes events to initially be inactive.
0023 //
0024 #ifndef __EVENT_H__
0025 #define __EVENT_H__
0026
0029 #define emberEventControlSetInactive(control) \
0030   do { (control).status = EMBER_EVENT_INACTIVE; } while(0)
0031
0034 #define emberEventControlGetActive(control) \
0035   ((control).status != EMBER_EVENT_INACTIVE)
0036
0040 #define emberEventControlSetActive(control) \
0041   do { emEventControlSetActive(&(control)); } while(0)
0042
0046 void emEventControlSetActive(EmberEventControl
    *event);
0047
0051 #define EMBER_MAX_EVENT_CONTROL_DELAY_MS (HALF_MAX_INT32U_VALUE - 1)
0052
0057 #define emberEventControlSetDelayMS(control, delay) \
0058   do { emEventControlSetDelayMS(&(control), (delay)); } while(0)
0059
0064 void emEventControlSetDelayMS(EmberEventControl
    *event, uint32_t delay);
0065
0069 #define EMBER_MAX_EVENT_CONTROL_DELAY_QS (EMBER_MAX_EVENT_CONTROL_DELAY_MS >>
    8)
0070
0076 #define emberEventControlSetDelayQS(control, delay) \
0077   do { emEventControlSetDelayMS(&(control), (delay) << 8); } while(0)
0078
0082 #define EMBER_MAX_EVENT_CONTROL_DELAY_MINUTES (EMBER_MAX_EVENT_CONTROL_DELAY_MS
    >> 16)
0083
0089 #define emberEventControlSetDelayMinutes(control, delay) \
0090   do { emEventControlSetDelayMS(&(control), (delay) << 16); } while(0)
0091
0095 #define emberEventControlGetRemainingMS(control) \
0096   (emEventControlGetRemainingMS(&(control)))
0097
0099 uint32_t emEventControlGetRemainingMS(
    EmberEventControl *event);
00202
00203
00204 // Running events
00205
00212 void emberRunEvents(EmberEventData *events);
00213
00218 void emberRunTask(EmberTaskId taskid);
00219
00227 uint32_t emberMsToNextEvent(EmberEventData *
    events, uint32_t maxMs);
00228
00234 uint32_t emberMsToNextEventExtended(EmberEventData
    *events, uint32_t maxMs, uint8_t* returnIndex);
00235
00239 uint32_t emberMsToNextStackEvent(void);
00240

```

```

00241
00246 EmberTaskId emberTaskInit(EmberEventData
    *events);
00247
00256 bool emberMarkTaskIdle(EmberTaskId taskid);
00257
00258 #ifndef EMBER_NO_IDLE_SUPPORT
00259
00261 #define emberTaskEnableIdling(allow) \
00262     do { emTaskEnableIdling((allow)); } while(0)
00263
00264 void emTaskEnableIdling(bool allow);
00265
00269 #define emberMarkTaskActive(taskid) \
00270     do { emMarkTaskActive((taskid)); } while(0)
00271
00272 void emMarkTaskActive(EmberTaskId taskid);
00273 #else
00274 #define emberTaskEnableIdling(allow) do {} while(0)
00275 #define emberMarkTaskActive(taskid) do {} while(0)
00276 #endif // EMBER_NO_IDLE_SUPPORT
00277
00278#endif // __EVENT_H__
00279
00280// @} END addtogroup
00281
00282

```

8.79 flash.h File Reference

```
#include "memmap.h"
```

Functions

- bool halFlashEraseIsActive (void)

8.79.1 Detailed Description

See [Flash Memory Control](#) for documentation.

Definition in file [flash.h](#).

8.80 flash.h

```

00001
00022 #ifndef __FLASH_H__
00023 #define __FLASH_H__
00024
00025 #include "memmap.h"
00026
00027
00037 bool 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(uint8_t eraseType, uint32_t
    address);
00069 EmberStatus halInternalFlashWrite(uint32_t address, uint16_t * data,

```

```

    uint32_t length);
00100 EmberStatus halInternalCibOptionByteWrite(uint8_t byte, uint8_t data
00101 );
00120 #endif //DOXYGEN_SHOULD_SKIP_THIS
00122 #endif //__FLASH_H__
00123
00124

```

8.81 form-and-join.h File Reference

Macros

- `#define NETWORK_STORAGE_SIZE`
- `#define NETWORK_STORAGE_SIZE_SHIFT`
- `#define FORM_AND_JOIN_MAX_NETWORKS`

Functions

- `EmberStatus emberScanForUnusedPanId (uint32_t channelMask, uint8_t duration)`
- `EmberStatus emberScanForJoinableNetwork (uint32_t channelMask, uint8_t *extendedPanId)`
- `EmberStatus emberScanForNextJoinableNetwork (void)`
- `bool emberFormAndJoinIsScanning (void)`
- `bool emberFormAndJoinCanContinueJoinableNetworkScan (void)`
- `void emberUnusedPanIdFoundHandler (EmberPanId panId, uint8_t channel)`
- `void emberJoinableNetworkFoundHandler (EmberZigbeeNetwork *networkFound, uint8_t lqi, int8_t rssi)`
- `void emberScanErrorHandler (EmberStatus status)`
- `bool emberFormAndJoinScanCompleteHandler (uint8_t channel, EmberStatus status)`
- `bool emberFormAndJoinNetworkFoundHandler (EmberZigbeeNetwork *networkFound, uint8_t lqi, int8_t rssi)`
- `bool emberFormAndJoinEnergyScanResultHandler (uint8_t channel, int8_t maxRssiValue)`
- `void emberFormAndJoinTick (void)`
- `void emberFormAndJoinTaskInit (void)`
- `void emberFormAndJoinRunTask (void)`
- `void emberFormAndJoinCleanup (EmberStatus status)`

Variables

- `bool emberEnableDualChannelScan`

8.81.1 Detailed Description

Utilities for forming and joining networks. See [Forming and Joining Networks](#) for documentation.

Definition in file [form-and-join.h](#).

8.82 form-and-join.h

```

00001 #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(uint32_t
    channelMask, uint8_t duration);
00122
00149 EmberStatus emberScanForJoinableNetwork(
    uint32_t channelMask, uint8_t* extendedPanId);
00150
00152 EmberStatus emberScanForNextJoinableNetwork
    (void);
00153
00169 extern bool emberEnableDualChannelScan;
00170
00175 bool emberFormAndJoinIsScanning(void);
00176
00181 bool emberFormAndJoinCanContinueJoinableNetworkScan
    (void);
00182
00183 //
-----00184 // Callbacks the application needs to implement.
00185
00194 void emberUnusedPanIdFoundHandler(EmberPanId
    panId, uint8_t channel);
00195
00206 void emberJoinableNetworkFoundHandler(
    EmberZigbeeNetwork *networkFound,
00207                                     uint8_t lqi,
00208                                     int8_t rssi);
00209
00227 void emberScanErrorHandler(EmberStatus status);
00228
00229 //
-----00230 // Library functions the application must call from within the
00231 // corresponding EmberZNet or EZSP callback.
00232
00240 bool emberFormAndJoinScanCompleteHandler(
    uint8_t channel, EmberStatus status);
00241
00249 bool emberFormAndJoinNetworkFoundHandler(
    EmberZigbeeNetwork *networkFound,
00250                                     uint8_t lqi,
00251                                     int8_t rssi);
00252
00260 bool emberFormAndJoinEnergyScanResultHandler
    (uint8_t channel, int8_t maxRssiValue);
00261
00266 void emberFormAndJoinTick(void);
00267
00271 void emberFormAndJoinTaskInit(void);
00272
00276 void emberFormAndJoinRunTask(void);
00277
00282 void emberFormAndJoinCleanup(EmberStatus
    status);

```

```
00283
00284
00285
```

8.83 fragment-host.h File Reference

Initialization

- void `ezspFragmentInit` (uint16_t receiveBufferLength, uint8_t *receiveBuffer)

Transmitting

- `EmberStatus ezspFragmentSendUnicast` (`EmberOutgoingMessageType` type, uint16_t indexOrDestination, `EmberApsFrame` *apsFrame, uint8_t maxFragmentSize, uint16_t messageLength, uint8_t *messageContents)
- `EmberStatus ezspFragmentSourceRouteHandler` (void)
- bool `ezspFragmentMessageSent` (`EmberApsFrame` *apsFrame, `EmberStatus` status)
- void `ezspFragmentMessageSentHandler` (`EmberStatus` status)

Receiving

- bool `ezspFragmentIncomingMessage` (`EmberApsFrame` *apsFrame, `EmberNodeId` sender, uint16_t *messageLength, uint8_t **messageContents)
- void `ezspFragmentTick` (void)

8.83.1 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](#).

8.84 fragment-host.h

```
00001
00059 void ezspFragmentInit(uint16_t receiveBufferLength, uint8_t *
    receiveBuffer);
00060
00095 EmberStatus ezspFragmentSendUnicast(
    EmberOutgoingMessageType type,
    uint16_t indexOrDestination,
    EmberApsFrame *apsFrame,
    uint8_t maxFragmentSize,
    uint16_t messageLength,
    uint8_t *messageContents);
00100
00114 EmberStatus ezspFragmentSourceRouteHandler
    (void);
```

```

00115
00130 bool ezspFragmentMessageSent(EmberApsFrame
    *apsFrame, EmberStatus status);
00131
00140 void ezspFragmentMessageSentHandler(EmberStatus
    status);
00141
00173 bool ezspFragmentIncomingMessage(EmberApsFrame
    *apsFrame,
                                EmberNodeId sender,
00175                                uint16_t *messageLength,
00176                                uint8_t **messageContents);
00177
00182 void ezspFragmentTick(void);
00183

```

8.85 fragment.h File Reference

Transmitting

- `EmberStatus emberFragmentSendUnicast (EmberOutgoingMessageType type, uint16_t indexOrDestination, EmberApsFrame *apsFrame, EmberMessageBuffer payload, uint8_t maxFragmentSize)`
- `bool emberFragmentMessageSent (EmberApsFrame *apsFrame, EmberMessageBuffer buffer, EmberStatus status)`
- `void emberFragmentMessageSentHandler (EmberStatus status)`

Receiving

- `bool emberFragmentIncomingMessage (EmberApsFrame *apsFrame, EmberMessageBuffer payload)`
- `void emberFragmentTick (void)`

8.85.1 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](#).

8.86 fragment.h

```

00001
00068 EmberStatus emberFragmentSendUnicast(
    EmberOutgoingMessageType type,
00069                                uint16_t indexOrDestination,
00070                                EmberApsFrame *apsFrame,
00071                                EmberMessageBuffer
    payload,
00072                                uint8_t maxFragmentSize);
00073
00088 bool emberFragmentMessageSent (EmberApsFrame
    *apsFrame,

```

```

00089                               EmberMessageBuffer buffer,
00090                               EmberStatus status);
00091
00099 void emberFragmentMessageSentHandler(EmberStatus
     status);
00100
00127 bool emberFragmentIncomingMessage(EmberApsFrame
     *apsFrame,
00128                               EmberMessageBuffer
     payload);
00129
00133 void emberFragmentTick(void);
00134

```

8.87 hal.h File Reference

```

#include "micro/micro.h"
#include "micro/antenna.h"
#include "micro/adc.h"
#include "micro/button.h"
#include "micro/buzzer.h"
#include "micro/crc.h"
#include "micro/endian.h"
#include "micro/led.h"
#include "micro/random.h"
#include "micro/serial.h"
#include "micro/spi.h"
#include "micro/system-timer.h"
#include "micro/bootloader-eeprom.h"
#include "micro/bootloader-interface.h"
#include "micro/diagnostic.h"
#include "micro/token.h"

```

Macros

- `#define emAmHost()`

8.87.1 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](#).

8.87.2 Macro Definition Documentation

8.87.2.1 `#define emAmHost()`

Definition at line [250](#) of file [hal.h](#).

8.88 hal.h

```

00001
00051 #ifndef __HAL_H__
00052 #define __HAL_H__
00053
00054 #ifdef HAL_HOST
00055
00056 #include "host/button-common.h"
00057 #include "host/crc.h"
00058 #include "host/led-common.h"
00059 #include "host/micro-common.h"
00060 #include "host/serial.h"
00061 #include "host/system-timer.h"
00062 #include "host/bootloader-eeprom.h"
00063 //Pull in the micro specific ADC, buzzer, and clocks headers. The
00064 //specific header is chosen by the build include path pointing at
00065 //the appropriate directory.
00066 #include "adc.h"
00067 #include "buzzer.h"
00068
00069 #else //HAL_MICRO
00070
00071 // Keep micro and board first for specifics used by other headers
00072 #include "micro/micro.h"
00073 #include "micro/antenna.h"
00074 #if !defined(STACK) && defined(BOARD_HEADER)
00075 #include BOARD_HEADER
00076 #endif
00077
00078 #if (defined(EMBER_STACK_CONNECT))
00079     #if (defined(UNIX_HOST) && !defined(EMBER_TEST))
00080         #include "micro/adc.h"
00081         #include "micro/button.h"
00082         #include "micro/buzzer.h"
00083         #include "micro/crc.h"
00084         #include "micro/endian.h"
00085         #include "micro/led.h"
00086         #include "micro/random.h"
00087         #include "micro/serial.h"
00088         #include "micro/spi.h"
00089         #include "micro/system-timer.h"
00090     #else
00091         // TODO: here we include only the functionalities that we will have on
00092         mustang
00093             #if (defined(CORTEXM3))
00094                 #include "micro/adc.h"
00095                 #include "micro/bootloader-eeprom.h"
00096                 #if ((defined _EFR_DEVICE) || (defined CORTEXM3_EMBER_MICRO))
00097                     #include "micro/bootloader-interface.h"
00098                 #endif
00099                 #include "micro/button.h"
00100                 #include "micro/led.h"
00101                 #include "micro/buzzer.h"
00102                 #include "micro/crc.h"
00103                 #include "micro/diagnostic.h"
00104             #endif //CORTEXM3
00105             #include "micro/flash.h"
00106             #include "micro/sim-eeprom.h"
00107             #include "micro/system-timer.h"
00108             #include "micro/symbol-timer.h"
00109             #include "micro/spi.h"
00110             #if (defined(CORTEXM3) || defined(EMBER_TEST))
00111                 #include "micro/serial.h"
00112             #else
00113                 #include "micro/serial-minimal.h"
00114             #endif
00115             #include "micro/random.h"
00116             #include "micro/token.h"
00117             #ifdef EMBER_TEST
00118                 #include "micro/adc.h"
00119                 #include "micro/bootloader-interface.h"
00120                 #include "micro/button.h"
00121                 #include "micro/led.h"
00122             #endif
00123         #endif // UNIX_HOST && !EMBER_TEST
00124     #elif (defined(EMBER_STACK_OWL_RX))
00125         // TODO: here we include only the functionalities that we will have on OWL-RX

```

```

00126 #include "micro/button.h"
00127 #include "micro/flash.h"
00128 #include "micro/led.h"
00129 #include "micro/dog_glcd.h"
00130 #include "micro/system-timer.h"
00131 #include "micro/symbol-timer.h"
00132 #include "micro/spi.h"
00133 #include "micro/serial-minimal.h"
00134 #include "micro/endian.h"
00135 // #include "micro/random.h"
00136 // #include "micro/token.h"
00137 // #ifdef EMBER_TEST
00138 // #include "micro/adc.h"
00139 // #include "micro/bootloader-interface.h"
00140 // #include "micro/button.h"
00141 // #include "micro/led.h"
00142 // #endif
00143 #elif (defined(EMBER_STACK_OWL_TX))
00144 // TODO: here we include only the functionalities that we will have on OWL-TX
00145 // #include "micro/button.h"
00146 // #include "micro/flash.h"
00147 // #include "micro/led.h"
00148 // #include "micro/dog_glcd.h"
00149 // #include "micro/system-timer.h"
00150 // #include "micro/symbol-timer.h"
00151 // #include "micro/spi.h"
00152 // #include "micro/serial-minimal.h"
00153 // #include "micro/random.h"
00154 // #include "micro/token.h"
00155 // #ifdef EMBER_TEST
00156 // #include "micro/adc.h"
00157 // #include "micro/bootloader-interface.h"
00158 // #include "micro/button.h"
00159 // #include "micro/led.h"
00160 // #endif
00161 #elif (defined(EMBER_STACK_WASP))
00162 // TODO: here we include only the functionalities that we will have on
mustang
00163 #if (defined(CORTEXM3))
00164 // #include "micro/adc.h"
00165 // #include "micro/bootloader-eeprom.h"
00166 #include "micro/button.h"
00167 #include "micro/buzzer.h"
00168 #include "micro/led.h"
00169 #include "micro/diagnostic.h"
00170 #endif //CORTEXM3
00171 #include "micro/flash.h"
00172 #include "micro/system-timer.h"
00173 #include "micro/symbol-timer.h"
00174 #include "micro/spi.h"
00175 #if (defined(CORTEXM3))
00176 #include "micro/serial.h"
00177 #else
00178 #include "micro/serial-minimal.h"
00179 #endif
00180 #include "micro/random.h"
00181 #include "micro/token.h"
00182 #ifdef EMBER_TEST
00183 #include "micro/adc.h"
00184 #include "micro/bootloader-interface.h"
00185 #include "micro/button.h"
00186 #include "micro/led.h"
00187 #endif
00188 #elif (! defined(EMBER_STACK_IP))
00189 // Pro Stack
00190 #include "micro/adc.h"
00191 #include "micro/button.h"
00192 #include "micro/buzzer.h"
00193 #include "micro/crc.h"
00194 #include "micro/endian.h"
00195 #include "micro/led.h"
00196 #include "micro/random.h"
00197 #include "micro/serial.h"
00198 #include "micro/spi.h"
00199 #include "micro/system-timer.h"
00200 #include "micro/bootloader-eeprom.h"
00201
00202 //Host processors do not use the following modules, therefore the header
00203 //files should be ignored.
00204 #ifndef EZSP_HOST

```

```

00205      #include "micro/bootloader-interface.h"
00206      #include "micro/diagnostic.h"
00207      #include "micro/token.h"
00208      //No public HAL code in release 4.0 uses the symbol timer,
00209      //therefore it should not be in doxygen.
00210      #ifndef DOXYGEN_SHOULD_SKIP_THIS
00211          #include "micro/symbol-timer.h"
00212      #endif // DOXYGEN_SHOULD_SKIP_THIS
00213  #endif //EZSP_HOST
00214
00215 #else
00216     // IP Stack
00217     #include "micro/adc.h"
00218     #include "micro/button.h"
00219     #include "micro/buzzer.h"
00220     #include "micro/crc.h"
00221     #include "micro/endian.h"
00222     #include "micro/led.h"
00223     #include "micro/random.h"
00224     #include "micro/serial.h"
00225     #include "micro/spi.h"
00226     #include "micro/system-timer.h"
00227     //Host processors do not use the following modules, therefore the header
00228     //files should be ignored.
00229     #ifndef UNIX_HOST
00230         #include "micro/bootloader-interface.h"
00231         #include "micro/diagnostic.h"
00232         #include "micro/token.h"
00233         //No public HAL code in release 4.0 uses the symbol timer,
00234         //therefore it should not be in doxygen.
00235         #ifndef DOXYGEN_SHOULD_SKIP_THIS
00236             #include "micro/symbol-timer.h"
00237         #endif // DOXYGEN_SHOULD_SKIP_THIS
00238     #endif //UNIX_HOST
00239
00240 #endif // !EMBER_STACK_IP
00241
00242 #endif // !HAL_HOST
00243
00244 #if ((defined(RTOS) && !defined(IP_MODEM_LIBRARY)) \
00245     || (defined(UNIX_HOST) \
00246         || defined(UNIX_HOST_SIM)))
00247     #define EMBER_HOST
00248     #define emAmHost() true
00249 #else
00250     #define emAmHost() false
00251 #endif
00252
00253 #endif //__HAL_H__
00254

```

8.89 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
- #define MAIN_FUNCTION_ARGUMENTS

Functions

- void `_executeBarrierInstructions` (void)

Master Variable Types

These are a set of typedefs to make the size of all variable declarations explicitly known.

- `typedef bool boolean`
- `typedef unsigned char int8u`
- `typedef signed char int8s`
- `typedef unsigned short int16u`
- `typedef signed short int16s`
- `typedef unsigned int int32u`
- `typedef signed int int32s`
- `typedef unsigned long long int64u`
- `typedef signed long long int64s`
- `typedef unsigned int PointerType`

Miscellaneous Macros

- `#define BIGENDIAN_CPU`
- `#define NTOHS(val)`
- `#define NTOHL(val)`
- `#define NO_STIPPING`
- `#define EEPROM`
- `#define __SOURCEFILE__`
- `#define assert(condition)`
- `#define halResetWatchdog()`
- `#define __attribute__(...)`
- `#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)`
- `#define STRINGIZE(X)`
- `#define ALIGNMENT(X)`
- `#define WEAK(__symbol)`
- `#define NO_INIT(__symbol)`
- `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 __BAT_INIT__
- #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 __INTERNAL_STORAGE__
- #define __UNRETAINED_RAM__
- #define __NO_INIT_SEGMENT_BEGIN
- #define __DEBUG_CHANNEL_SEGMENT_BEGIN
- #define __INTVEC_SEGMENT_BEGIN
- #define __CSTACK_SEGMENT_BEGIN
- #define __RESETINFO_SEGMENT_BEGIN
- #define __DATA_INIT_SEGMENT_BEGIN
- #define __DATA_SEGMENT_BEGIN
- #define __BSS_SEGMENT_BEGIN
- #define __APP_RAM_SEGMENT_BEGIN
- #define __CONST_SEGMENT_BEGIN
- #define __TEXT_SEGMENT_BEGIN
- #define __TEXTRW_INIT_SEGMENT_BEGIN
- #define __TEXTRW_SEGMENT_BEGIN
- #define __AAT_SEGMENT_BEGIN
- #define __BAT_SEGMENT_BEGIN
- #define __BAT_INIT_SEGMENT_BEGIN
- #define __FAT_SEGMENT_BEGIN
- #define __RAT_SEGMENT_BEGIN
- #define __NVM_SEGMENT_BEGIN
- #define __SIMEE_SEGMENT_BEGIN

- #define _EMHEAP_SEGMENT_BEGIN
- #define _EMHEAP_OVERLAY_SEGMENT_BEGIN
- #define _GUARD_REGION_SEGMENT_BEGIN
- #define _DLIB_PERTHREAD_INIT_SEGMENT_BEGIN
- #define _DLIB_PERTHREAD_INITIALIZED_DATA_SEGMENT_BEGIN
- #define _DLIB_PERTHREAD_ZERO_DATA_SEGMENT_BEGIN
- #define _INTERNAL_STORAGE_SEGMENT_BEGIN
- #define _UNRETAINED_RAM_SEGMENT_BEGIN
- #define _NO_INIT_SEGMENT_END
- #define _DEBUG_CHANNEL_SEGMENT_END
- #define _INTVEC_SEGMENT_END
- #define _CSTACK_SEGMENT_END
- #define _RESETINFO_SEGMENT_END
- #define _DATA_INIT_SEGMENT_END
- #define _DATA_SEGMENT_END
- #define _BSS_SEGMENT_END
- #define _APP_RAM_SEGMENT_END
- #define _CONST_SEGMENT_END
- #define _TEXT_SEGMENT_END
- #define _TEXTRW_INIT_SEGMENT_END
- #define _TEXTRW_SEGMENT_END
- #define _AAT_SEGMENT_END
- #define _BAT_SEGMENT_END
- #define _BAT_INIT_SEGMENT_END
- #define _FAT_SEGMENT_END
- #define _RAT_SEGMENT_END
- #define _NVM_SEGMENT_END
- #define _SIMEE_SEGMENT_END
- #define _EMHEAP_SEGMENT_END
- #define _EMHEAP_OVERLAY_SEGMENT_END
- #define _GUARD_REGION_SEGMENT_END
- #define _DLIB_PERTHREAD_INIT_SEGMENT_END
- #define _DLIB_PERTHREAD_INITIALIZED_DATA_SEGMENT_END
- #define _DLIB_PERTHREAD_ZERO_DATA_SEGMENT_END
- #define _INTERNAL_STORAGE_SEGMENT_END
- #define _UNRETAINED_RAM_SEGMENT_END
- #define _NO_INIT_SEGMENT_SIZE
- #define _DEBUG_CHANNEL_SEGMENT_SIZE
- #define _INTVEC_SEGMENT_SIZE
- #define _CSTACK_SEGMENT_SIZE
- #define _RESETINFO_SEGMENT_SIZE
- #define _DATA_INIT_SEGMENT_SIZE
- #define _DATA_SEGMENT_SIZE
- #define _BSS_SEGMENT_SIZE
- #define _APP_RAM_SEGMENT_SIZE
- #define _CONST_SEGMENT_SIZE
- #define _TEXT_SEGMENT_SIZE
- #define _TEXTRW_INIT_SEGMENT_SIZE
- #define _TEXTRW_SEGMENT_SIZE
- #define _AAT_SEGMENT_SIZE

- #define _BAT_SEGMENT_SIZE
- #define _BAT_INIT_SEGMENT_SIZE
- #define _FAT_SEGMENT_SIZE
- #define _RAT_SEGMENT_SIZE
- #define _NVM_SEGMENT_SIZE
- #define _SIMEE_SEGMENT_SIZE
- #define _EMHEAP_SEGMENT_SIZE
- #define _EMHEAP_OVERLAY_SEGMENT_SIZE
- #define _GUARD_REGION_SEGMENT_SIZE
- #define _DLIB_PERTHREAD_INIT_SEGMENT_SIZE
- #define _DLIB_PERTHREAD_INITIALIZED_DATA_SEGMENT_SIZE
- #define _DLIB_PERTHREAD_ZERO_DATA_SEGMENT_SIZE
- #define _INTERNAL_STORAGE_SEGMENT_SIZE
- #define _UNRETAINED_RAM_SEGMENT_SIZE

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 trigger 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)

8.89.1 Detailed Description

See [IAR PLATFORM_HEADER Configuration](#) for detailed documentation.

Definition in file [iar.h](#).

8.90 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     #include <stdint.h>
00035     #include <stdbool.h>
00036     #include <string.h>
00037     #if defined (CORTEXM3_EMBER_MICRO)
00038         #include "micro/cortexm3/em35x/regs.h"
00039         #include "micro/cortexm3/micro-features.h"
00040     #elif defined (CORTEXM3_EFM32_MICRO)
00041         // EFR32, EZR32
00042         #include "em_device.h"
00043         #include "micro/cortexm3/efm32/regs.h"
00044         #include "micro/cortexm3/micro-features.h"
00045         #define NVIC_CONFIG "hal/micro/cortexm3/efm32/nvic-config.h"
00046     #else
00047         #error Unknown CORTEXM3 micro
00048     #endif
00049 //Provide a default NVIC configuration file. The build process can
00050 //override this if it needs to.
00051 #ifndef NVIC_CONFIG
00052     #define NVIC_CONFIG "hal/micro/cortexm3/nvic-config.h"
00053 #endif
00054 //[]
00055 #ifdef EMBER_EMU_TEST
00056     #ifdef I_AM_AN_EMULATOR
00057         // This register is defined for both the chip and the emulator with
00058         // with distinct reset values. Need to undefine to avoid preprocessor
00059         // collision.
00060         #undef DATA_EMU_REGS_BASE
00061         #undef DATA_EMU_REGS_END
00062         #undef DATA_EMU_REGS_SIZE
00063         #undef I_AM_AN_EMULATOR
00064         #undef I_AM_AN_EMULATOR_REG
00065         #undef I_AM_AN_EMULATOR_ADDR
00066         #undef I_AM_AN_EMULATOR_RESET
00067         #undef I_AM_AN_EMULATOR_I_AM_AN_EMULATOR
00068         #undef I_AM_AN_EMULATOR_I_AM_AN_EMULATOR_MASK
00069         #undef I_AM_AN_EMULATOR_I_AM_AN_EMULATOR_BIT
00070         #undef I_AM_AN_EMULATOR_I_AM_AN_EMULATOR_BITS
00071     #endif//I_AM_AN_EMULATOR
00072     #if defined (CORTEXM3_EMBER_MICRO)
00073         #include "micro/cortexm3/em35x/regs-emu.h"
00074         #include "micro/cortexm3/micro-features.h"
00075     #else
00076         #error MICRO currently not supported for emulator builds.
00077     #endif
00078 #endif//EMBER_EMU_TEST
00079 //]
00080
00081 // suppress warnings about unknown pragmas
00082 // (as they may be pragmas known to other platforms)
00083 #pragma diag_suppress = pe161
00084
00085 #endif // DOXYGEN_SHOULD_SKIP_THIS
00086
00087
00088
00089
00098 typedef bool boolean; /*To ease adoption of bool instead of boolean.*/
00099 typedef unsigned char int8u;
00100 typedef signed char int8s;
00101 typedef unsigned short int16u;
00102 typedef signed short int16s;

```

```

00103 typedef unsigned int    int32u;
00104 typedef signed   int    int32s;
00105 typedef unsigned long long int64u;
00106 typedef signed   long long int64s;
00107 typedef unsigned int    PointerType;
00109
00113 #define HAL_HAS_INT64
00114
00118 #define _HAL_USE_COMMON_PGM_
00119
00120
00121
00123
00125
00126
00127
00132 #define BIGENDIAN_CPU  false
00133
00134
00139 #define NTOHS(val)  (__REV16(val))
00140 #define NTOHL(val)  (__REV(val))
00141
00142
00147 #define NO_STRIPPING  __root
00148
00149
00154 #define EEPROM  errorerror
00155
00156
00157 #ifndef __SOURCEFILE__
00158
00163 #define __SOURCEFILE__ __FILE__
00164 #endif
00165
00166
00167 #undef assert
00168
00172 void halInternalAssertFailed(const char *filename, int
linenumber);
00173
00179 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00180 #define assert(condition)
00181 #else //DOXYGEN_SHOULD_SKIP_THIS
00182 // Don't define PUSH_REGS_BEFORE_ASSERT if it causes problems with the
// compiler.
00183 // For example, in some compilers any inline assembly disables all
// optimization.
00184 //
00185 // For IAR V5.30, inline assembly apparently does not affect compiler output.
00186 // #define PUSH_REGS_BEFORE_ASSERT
00187 #ifdef PUSH_REGS_BEFORE_ASSERT
00188 #define assert(condition) \
00189     do { if (! (condition)) { \
00190         asm("PUSH {R0,R1,R2,LR}"); \
00191         halInternalAssertFailed(__SOURCEFILE__, __LINE__); } } while(0)
00192 #else
00193 #define assert(condition) \
00194     do { if (! (condition)) { \
00195         halInternalAssertFailed(__SOURCEFILE__, __LINE__); } } while(0)
00196 #endif
00197 #endif //DOXYGEN_SHOULD_SKIP_THIS
00198
00204 void halInternalResetWatchDog(void);
00205 #ifdef RTOS
00206     void rtosResetWatchdog(void);
00207     #define halResetWatchdog()  rtosResetWatchdog()
00208 #else
00209     #define halResetWatchdog()  halInternalResetWatchDog()
00210 #endif //RTOS
00211
00215 #define __attribute__(...)
00216
00217
00222 #define UNUSED
00223
00227 #define SIGNED_ENUM
00228
00229
00233 #define STACK_FILL_VALUE  0xCDCDCDCD
00234

```

```

00238 #ifdef RAMEXE
00239     //If the whole build is running out of RAM, as chosen by the RAMEXE build
00240     //define, then define RAMFUNC to nothing since it's not needed.
00241     #define RAMFUNC
00242 #else //RAMEXE
00243     #define RAMFUNC __ramfunc
00244 #endif //RAMEXE
00245
00246 #define NO_OPERATION() __no_operation()
00247
00248 #define SET_REG_FIELD(reg, field, value)
00249     do{
00250         reg = ((reg & (~field##_MASK)) |
00251                 (uint32_t) ((uint32_t) value) << field##_BIT));
00252     }while(0)
00253
00254 #define simulatedTimePasses()
00255
00256 #define simulatedTimePassesMs(x)
00257
00258 #define simulatedSerialTimePasses()
00259
00260 #define _HAL_USE_COMMON_DIVMOD_
00261
00262
00263 #define VAR_AT_SEGMENT(__variableDeclaration, __segmentName) \
00264     __variableDeclaration @ __segmentName \
00265
00266 #define STRINGIZE(X) #X
00267
00268 #define ALIGNMENT(X) \
00269     _Pragma( STRINGIZE( data_alignment=X ) )
00270
00271 #define WEAK(__symbol) \
00272     __weak __symbol
00273
00274 #define NO_INIT(__symbol) \
00275     __no_init __symbol
00276
00277
00278 #define __NO_INIT__ ".noinit"
00279 #define __DEBUG_CHANNEL__ "DEBUG_CHANNEL"
00280 #define __INTVEC__ ".intvec"
00281 #define __CSTACK__ "CSTACK"
00282 #define __RESETINFO__ "RESETINFO"
00283 #define __DATA_INIT__ ".data_init"
00284 #define __DATA__ ".data"
00285 #define __BSS__ ".bss"
00286 #define __APP_RAM__ "APP_RAM"
00287 #define __CONST__ ".rodata"
00288 #define __TEXT__ ".text"
00289 #define __TEXTRW_INIT__ ".textrw_init"
00290 #define __TEXTRW__ ".textrw"
00291 #define __AAT__ "AAT" // Application address table
00292 #define __BAT__ "BAT" // Bootloader address table
00293 #define __BAT_INIT__ "BAT" // Bootloader address table
00294 #define __FAT__ "FAT" // Fixed address table
00295 #define __RAT__ "RAT" // Ramexe address table
00296 #define __NVM__ "NVM" //Non-Volatile Memory data storage
00297 #define __SIMEE__ "SIMEE" //Simulated EEPROM storage
00298 #define __EMHEAP__ "EMHEAP" // Heap region for extra memory
00299 #define __EMHEAP_OVERLAY__ "EMHEAP_overlay" // Heap and reset info combined
00300 #define __GUARD_REGION__ "GUARD_REGION" // Guard page between heap and stack
00301 #define __DLIB_PERTHREAD_INIT__ "__DLIB_PERTHREAD_init" // DLIB_PERTHREAD flash
00302     initialization data
00303 #define __DLIB_PERTHREAD_INITIALIZED_DATA__ "DLIB_PERTHREAD_INITIALIZED_DATA"
00304 // DLIB_PERTHREAD RAM region to init
00305 #define __DLIB_PERTHREAD_ZERO_DATA__ "DLIB_PERTHREAD_ZERO_DATA" //
00306     DLIB_PERTHREAD RAM region to zero out
00307 #define __INTERNAL_STORAGE__ "INTERNAL_STORAGE" //Internal storage region
00308 #define __UNRETAINED_RAM__ "UNRETAINED_RAM" //Region of RAM not retained during
00309     deepsleep
00310
00311
00312 //=====
00313 // The '#pragma segment=' declaration must be used before attempting to access
00314 // the segments so the compiler properly handles the __segment_*() functions.
00315
00316
00317
00318
00319
00320
00321
00322
00323
00324
00325
00326
00327
00328
00329
00330
00331
00332
00333
00334
00335
00336
00337
00338
00339
00340
00341
00342
00343
00344
00345
00346
00347
00348
00349
00350
00351 //=====
00352 // The '#pragma segment=' declaration must be used before attempting to access
00353 // the segments so the compiler properly handles the __segment_*() functions.

```

```

00354 //
00355 // The segment names used here are the default segment names used by IAR. Refer
00356 // to the IAR Compiler Reference Guide for a proper description of these
00357 // segments.
00358 //=====
00359 #pragma segment=__NO_INIT__
00360 #pragma segment=__DEBUG_CHANNEL__
00361 #pragma segment=__INTVEC__
00362 #pragma segment=__CSTACK__
00363 #pragma segment=__RESETINFO__
00364 #pragma segment=__DATA_INIT__
00365 #pragma segment=__DATA__
00366 #pragma segment=__BSS__
00367 #pragma segment=__APP_RAM__
00368 #pragma segment=__CONST__
00369 #pragma segment=__TEXT__
00370 #pragma segment=__TEXTRW_INIT__
00371 #pragma segment=__TEXTRW__
00372 #pragma segment=__AAT__
00373 #pragma segment=__BAT__
00374 #pragma segment=__FAT__
00375 #pragma segment=__RAT__
00376 #pragma segment=__NVM__
00377 #pragma segment=__SIMEE__
00378 #pragma segment=__EMHEAP__
00379 #pragma segment=__EMHEAP_OVERLAY__
00380 #pragma segment=__GUARD_REGION__
00381 #pragma segment=__DLIB_PERTHREAD_INIT__
00382 #pragma segment=__DLIB_PERTHREAD_INITIALIZED_DATA__
00383 #pragma segment=__DLIB_PERTHREAD_ZERO_DATA__
00384 #pragma segment=__INTERNAL_STORAGE__
00385 #pragma segment=__UNRETAINED_RAM__
00386
00387 #define __NO_INIT_SEGMENT_BEGIN
    __segment_begin(__NO_INIT__)
00388 #define __DEBUG_CHANNEL_SEGMENT_BEGIN
    __segment_begin(__DEBUG_CHANNEL__)
00389 #define __INTVEC_SEGMENT_BEGIN
    __segment_begin(__INTVEC__)
00390 #define __CSTACK_SEGMENT_BEGIN
    __segment_begin(__CSTACK__)
00391 #define __RESETINFO_SEGMENT_BEGIN
    __segment_begin(__RESETINFO__)
00392 #define __DATA_INIT_SEGMENT_BEGIN
    __segment_begin(__DATA_INIT__)
00393 #define __DATA_SEGMENT_BEGIN
    __segment_begin(__DATA__)
00394 #define __BSS_SEGMENT_BEGIN
    __segment_begin(__BSS__)
00395 #define __APP_RAM_SEGMENT_BEGIN
    __segment_begin(__APP_RAM__)
00396 #define __CONST_SEGMENT_BEGIN
    __segment_begin(__CONST__)
00397 #define __TEXT_SEGMENT_BEGIN
    __segment_begin(__TEXT__)
00398 #define __TEXTRW_INIT_SEGMENT_BEGIN
    __segment_begin(__TEXTRW_INIT__)
00399 #define __TEXTRW_SEGMENT_BEGIN
    __segment_begin(__TEXTRW__)
00400 #define __AAT_SEGMENT_BEGIN
    __segment_begin(__AAT__)
00401 #define __BAT_SEGMENT_BEGIN
    __segment_begin(__BAT__)
00402 #define __BAT_INIT_SEGMENT_BEGIN
    __segment_begin(__BAT_INIT__)
00403 #define __FAT_SEGMENT_BEGIN
    __segment_begin(__FAT__)
00404 #define __RAT_SEGMENT_BEGIN
    __segment_begin(__RAT__)
00405 #define __NVM_SEGMENT_BEGIN
    __segment_begin(__NVM__)
00406 #define __SIMEE_SEGMENT_BEGIN
    __segment_begin(__SIMEE__)
00407 #define __EMHEAP_SEGMENT_BEGIN
    __segment_begin(__EMHEAP__)
00408 #define __EMHEAP_OVERLAY_SEGMENT_BEGIN
    __segment_begin(__EMHEAP_OVERLAY__)
00409 #define __GUARD_REGION_SEGMENT_BEGIN
    __segment_begin(__GUARD_REGION__)
00410 #define __DLIB_PERTHREAD_INIT_SEGMENT_BEGIN
    __segment_begin(__DLIB_PERTHREAD_INIT__)
00411 #define __DLIB_PERTHREAD_INITIALIZED_DATA_SEGMENT_BEGIN
    __segment_begin(__DLIB_PERTHREAD_INITIALIZED_DATA__)
00412 #define __DLIB_PERTHREAD_ZERO_DATA_SEGMENT_BEGIN
    __segment_begin(__DLIB_PERTHREAD_ZERO_DATA__)
00413 #define __INTERNAL_STORAGE_SEGMENT_BEGIN
    __segment_begin(__INTERNAL_STORAGE__)

```

```

    __segment_begin(__INTERNAL_STORAGE__)
00414 #define _UNRETAINED_RAM_SEGMENT_BEGIN
        __segment_begin(__UNRETAINED_RAM__)
00415
00416 #define _NO_INIT_SEGMENT_END
00417 #define _DEBUG_CHANNEL_SEGMENT_END
        __segment_end(__DEBUG_CHANNEL__)
00418 #define _INTVEC_SEGMENT_END
00419 #define _CSTACK_SEGMENT_END
00420 #define _RESETINFO_SEGMENT_END
        __segment_end(__RESETINFO__)
00421 #define _DATA_INIT_SEGMENT_END
        __segment_end(__DATA_INIT__)
00422 #define _DATA_SEGMENT_END
00423 #define _BSS_SEGMENT_END
00424 #define _APP_RAM_SEGMENT_END
00425 #define _CONST_SEGMENT_END
00426 #define _TEXT_SEGMENT_END
00427 #define _TEXTRW_INIT_SEGMENT_END
        __segment_end(__TEXTRW_INIT__)
00428 #define _TEXTRW_SEGMENT_END
00429 #define _AAT_SEGMENT_END
00430 #define _BAT_SEGMENT_END
00431 #define _BAT_INIT_SEGMENT_END
        __segment_end(__BAT_INIT__)
00432 #define _FAT_SEGMENT_END
00433 #define _RAT_SEGMENT_END
00434 #define _NVM_SEGMENT_END
00435 #define _SIMEE_SEGMENT_END
00436 #define _EMHEAP_SEGMENT_END
00437 #define _EMHEAP_OVERLAY_SEGMENT_END
        __segment_end(__EMHEAP_OVERLAY__)
00438 #define _GUARD_REGION_SEGMENT_END
        __segment_end(__GUARD_REGION__)
00439 #define _DLIB_PERTHREAD_INIT_SEGMENT_END
        __segment_end(__DLIB_PERTHREAD_INIT__)
00440 #define _DLIB_PERTHREAD_INITIALIZED_DATA_SEGMENT_END
        __segment_end(__DLIB_PERTHREAD_INITIALIZED_DATA__)
00441 #define _DLIB_PERTHREAD_ZERO_DATA_SEGMENT_END
        __segment_end(__DLIB_PERTHREAD_ZERO_DATA__)
00442 #define _INTERNAL_STORAGE_SEGMENT_END
        __segment_end(__INTERNAL_STORAGE__)
00443 #define _UNRETAINED_RAM_SEGMENT_END
        __segment_end(__UNRETAINED_RAM__)
00444
00445 #define _NO_INIT_SEGMENT_SIZE
        __segment_size(__NO_INIT__)
00446 #define _DEBUG_CHANNEL_SEGMENT_SIZE
        __segment_size(__DEBUG_CHANNEL__)
00447 #define _INTVEC_SEGMENT_SIZE
        __segment_size(__INTVEC__)
00448 #define _CSTACK_SEGMENT_SIZE
        __segment_size(__CSTACK__)
00449 #define _RESETINFO_SEGMENT_SIZE
        __segment_size(__RESETINFO__)
00450 #define _DATA_INIT_SEGMENT_SIZE
        __segment_size(__DATA_INIT__)
00451 #define _DATA_SEGMENT_SIZE
00452 #define _BSS_SEGMENT_SIZE
00453 #define _APP_RAM_SEGMENT_SIZE
        __segment_size(__APP_RAM__)
00454 #define _CONST_SEGMENT_SIZE
00455 #define _TEXT_SEGMENT_SIZE
00456 #define _TEXTRW_INIT_SEGMENT_SIZE
        __segment_size(__TEXTRW_INIT__)
00457 #define _TEXTRW_SEGMENT_SIZE
        __segment_size(__TEXTRW__)
00458 #define _AAT_SEGMENT_SIZE
00459 #define _BAT_SEGMENT_SIZE
00460 #define _BAT_INIT_SEGMENT_SIZE
        __segment_size(__BAT_INIT__)
00461 #define _FAT_SEGMENT_SIZE
00462 #define _RAT_SEGMENT_SIZE
00463 #define _NVM_SEGMENT_SIZE
00464 #define _SIMEE_SEGMENT_SIZE
00465 #define _EMHEAP_SEGMENT_SIZE
        __segment_size(__EMHEAP__)
00466 #define _EMHEAP_OVERLAY_SEGMENT_SIZE
        __segment_size(__EMHEAP_OVERLAY__)
00467 #define _GUARD_REGION_SEGMENT_SIZE
        __segment_size(__GUARD_REGION__)

```

```

    __segment_size(__GUARD_REGION__)
00468 #define _DLIB_PERTHREAD_INIT_SEGMENT_SIZE
    __segment_size(__DLIB_PERTHREAD_INIT__)
00469 #define _DLIB_PERTHREAD_INITIALIZED_DATA_SEGMENT_SIZE
    __segment_size(__DLIB_PERTHREAD_INITIALIZED_DATA__)
00470 #define _DLIB_PERTHREAD_ZERO_DATA_SEGMENT_SIZE
    __segment_size(__DLIB_PERTHREAD_ZERO_DATA__)
00471 #define _INTERNAL_STORAGE_SEGMENT_SIZE
    __segment_size(__INTERNAL_STORAGE__)
00472 #define _UNRETAINED_RAM_SEGMENT_SIZE
    __segment_size(__UNRETAINED_RAM__)

00473
00476 //A utility function for inserting barrier instructions. These
00477 //instructions should be used whenever the MPU is enabled or disabled so
00478 //that all memory/instruction accesses can complete before the MPU changes
00479 //state.
00480 void _executeBarrierInstructions(void);
00481
00483
00493
00494
00495
00496 #define ATOMIC_LITE(blah) ATOMIC(blah)
00497 #define DECLARE_INTERRUPT_STATE_LITE DECLARE_INTERRUPT_STATE
00498 #define DISABLE_INTERRUPTS_LITE() DISABLE_INTERRUPTS()
00499 #define RESTORE_INTERRUPTS_LITE() RESTORE_INTERRUPTS()
00500
00501 #ifdef BOOTLOADER
00502     #ifndef DOXYGEN_SHOULD_SKIP_THIS
00503         // The bootloader does not use interrupts
00504         #define DECLARE_INTERRUPT_STATE
00505         #define DISABLE_INTERRUPTS() do { } while(0)
00506         #define RESTORE_INTERRUPTS() do { } while(0)
00507         #define INTERRUPTS_ON() do { } while(0)
00508         #define INTERRUPTS_OFF() do { } while(0)
00509         #define INTERRUPTS_ARE_OFF() (false)
00510         #define ATOMIC(blah) { blah }
00511         #define HANDLE_PENDING_INTERRUPTS() do { } while(0)
00512         #define SET_BASE_PRIORITY_LEVEL(basepri) do { } while(0)
00513     #endif // DOXYGEN_SHOULD_SKIP_THIS
00514 #else // BOOTLOADER
00515
00516     #ifndef DOXYGEN_SHOULD_SKIP_THIS
00517         // A series of macros for the diagnostics of the global interrupt state.
00518         // These macros either enable or disable the debugging code in the
00519         // Interrupt Manipulation Macros as well as define the two pins used for
00520         // indicating the entry and exit of critical sections.
00521         //UNCOMMENT the below #define to enable interrupt debugging.
00522         //#define INTERRUPT_DEBUGGING
00523         #ifdef INTERRUPT_DEBUGGING
00524             // Designed to use the breakout board LED (PC5)
00525             #define ATOMIC_DEBUG(x) x
00526             #define I_PIN      5
00527             #define I_PORT     C
00528             #define I_CFG_HL   H
00529             // For the concatenation to work, we need to define the regs via their
00530             // own macros:
00531             #define I_SET_REG(port)      GPIO_P ## port ## SET
00532             #define I_CLR_REG(port)      GPIO_P ## port ## CLR
00533             #define I_OUT_REG(port)      GPIO_P ## port ## OUT
00534             #define I_CFG_MSK(port, pin) P ## port ## pin ## _CFG_MASK
00535             #define I_CFG_BIT(port, pin) P ## port ## pin ## _CFG_BIT
00536             #define I_CFG_REG(port, hl)  GPIO_P ## port ## CFG ## hl
00537             // Finally, the macros to actually manipulate the interrupt status IO
00538             #define I_OUT(port, pin, hl) \
00539                 do { I_CFG_REG(port, hl) &= ~(I_CFG_MSK(port, pin));      \
00540                     I_CFG_REG(port, hl) |= (GPIOCFG_OUT << I_CFG_BIT(port, pin)); \
00541                 } while (0)
00542             #define I_SET(port, pin)   do { I_SET_REG(port) = (BIT(pin)); } while (0)
00543             #define I_CLR(port, pin)   do { I_CLR_REG(port) = (BIT(pin)); } while (0)
00544             #define I_STATE(port, pin) ((I_OUT_REG(port) & BIT(pin)) == BIT(pin))
00545             #if !defined(RTOS)
00546                 #define DECLARE_INTERRUPT_STATE uint8_t _emIsrState, _emIsrDbgIoState
00547             #else
00548                 #define DECLARE_INTERRUPT_STATE uint8_t _emIsrDbgIoState
00549             #endif
00550             #define ATOMIC_DEBUG(x)
00551             #if !defined(RTOS)
00552

```

```

00557     #define DECLARE_INTERRUPT_STATE uint8_t _emIsrState
00558 #else
00559     #define DECLARE_INTERRUPT_STATE
00560 #endif
00561
00562 #endif//INTERRUPT_DEBUGGING
00563
00564 // Prototypes for the BASEPRI and PRIMASK access functions. They are very
00565 // basic and instantiated in assembly code in the file spmr.s37 (since
00566 // there are no C functions that cause the compiler to emit code to access
00567 // the BASEPRI/PRIMASK). This will inhibit the core from taking interrupts
00568 // with a priority equal to or less than the BASEPRI value.
00569 // Note that the priority values used by these functions are 5 bits and
00570 // right-aligned
00571 extern uint8_t _readBasePri(void);
00572 extern void _writeBasePri(uint8_t priority);
00573
00574 // Prototypes for BASEPRI functions used to disable and enable interrupts
00575 // while still allowing enabled faults to trigger.
00576 extern void _enableBasePri(void);
00577 extern uint8_t _disableBasePri(void);
00578 extern bool _basePriIsDisabled(void);
00579
00580 // Prototypes for setting and clearing PRIMASK for global interrupt
00581 // enable/disable.
00582 extern void _setPriMask(void);
00583 extern void _clearPriMask(void);
00584
00585 #if defined(RTOS)
00586     // Prototypes for the RTOS critical functions we will need to do ATOMIC()
00587     uint8_t rtosEnterCritical(void);
00588     uint8_t rtosExitCritical(void);
00589 #endif //defined(RTOS)
00590
00591 #endif // DOXYGEN_SHOULD_SKIP_THIS
00592
00593 //The core Global Interrupt Manipulation Macros start here.
00594
00595 // When building for an RTOS be sure to use compatible interrupt routines
00596 #if defined(RTOS)
00597
00598     #define DISABLE_INTERRUPTS()
00599         do {
00600             rtosEnterCritical();
00601             ATOMIC_DEBUG(
00602                 _emIsrDbgIoState = I_STATE(I_PORT, I_PIN);
00603                 I_SET(I_PORT, I_PIN);
00604             )
00605         } while(0)
00606
00607     #define RESTORE_INTERRUPTS()
00608         do {
00609             ATOMIC_DEBUG(
00610                 if(!_emIsrDbgIoState)
00611                     I_CLR(I_PORT, I_PIN);
00612             )
00613             rtosExitCritical();
00614         } while(0)
00615
00616 #else // defined(RTOS)
00617
00618     #define DISABLE_INTERRUPTS()
00619         do {
00620             _emIsrState = _disableBasePri();
00621             ATOMIC_DEBUG(
00622                 _emIsrDbgIoState = I_STATE(I_PORT, I_PIN);
00623                 I_SET(I_PORT, I_PIN);
00624             )
00625         } while(0)
00626
00627     #define RESTORE_INTERRUPTS()
00628         do {
00629             ATOMIC_DEBUG(
00630                 if(!_emIsrDbgIoState)
00631                     I_CLR(I_PORT, I_PIN);
00632             )
00633             _writeBasePri(_emIsrState); \
00634         } while(0)
00635
00636 #endif // defined(RTOS)

```

```

00665
00670 #define INTERRUPTS_ON() \
00671     do { \
00672         ATOMIC_DEBUG( \
00673             I_OUT(I_PORT, I_PIN, I_CFG_HL); \
00674             I_CLR(I_PORT, I_PIN); \
00675         ) \
00676         _enableBasePri(); \
00677     } while(0)
00678
00679
00684 #define INTERRUPTS_OFF() \
00685     do { \
00686         (void)_disableBasePri(); \
00687         ATOMIC_DEBUG( \
00688             I_SET(I_PORT, I_PIN); \
00689         ) \
00690     } while(0)
00691
00692
00696 #define INTERRUPTS_ARE_OFF() ( _basePriIsDisabled() )
00697
00702 #define INTERRUPTS_WERE_ON() (_emIsrState == 0)
00703
00708 #define ATOMIC(blah) \
00709 { \
00710     DECLARE_INTERRUPT_STATE; \
00711     DISABLE_INTERRUPTS(); \
00712     { blah } \
00713     RESTORE_INTERRUPTS(); \
00714 }
00715
00716
00724 #define HANDLE_PENDING_INTERRUPTS() \
00725     do { \
00726         if (INTERRUPTS_ARE_OFF()) { \
00727             INTERRUPTS_ON(); \
00728             INTERRUPTS_OFF(); \
00729         } \
00730     } while (0)
00731
00732
00746 #define SET_BASE_PRIORITY_LEVEL(basepri) \
00747     do { \
00748         _writeBasePri(basepri); \
00749     } while(0)
00750
00751 #endif // BOOTLOADER
00752
00753
00754
00755
00760 #define _HAL_USE_COMMON_MEMUTILS_
00761
00763
00767
00768
00769
00779 int abs(int I);
00780
00782
00783
00784
00785
00789 #define PLATCOMMONOKTOINCLUDE
00790     #include "hal/micro/generic/compiler/platform-common.h"
00791 #undef PLATCOMMONOKTOINCLUDE
00792
00796 #define MAIN_FUNCTION_PARAMETERS void
00797 #define MAIN_FUNCTION_ARGUMENTS
00798 #endif // __IAR_H__
00799

```

8.91 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 (bool turnOn)`

8.91.1 Detailed Description

See [LED Control](#) for documentation.

Definition in file `led.h`.

8.92 led.h

```

00001 #ifndef __LED_H__
00002 #define __LED_H__
00003
00024 void halInternalInitLed(void);
00025
00029 #if defined(STACK) || defined(MINIMAL_HAL)
00030     typedef uint8_t HalBoardLed;
00031 #else
00032     typedef enum HalBoardLedPins HalBoardLed;
00033 #endif
00034 // Note: Even though many compilers will use 16 bits for an enum instead of 8,
00035 // we choose to use an enum here. The possible compiler inefficiency does not
00036 // affect stack-based parameters and local variables, which is the
00037 // general case for led parameters.
00038
00044 void halToggleLed(HalBoardLed led);
00045
00051 void halSetLed(HalBoardLed led);
00052
00058 void halClearLed(HalBoardLed led);
00059
00069 void halStackIndicateActivity(bool turnOn);
00070
00074 #endif // __LED_H__

```

8.93 message.h File Reference

Macros

- `#define EMBER_APSC_MAX_ACK_WAIT_HOPS_MULTIPLIER_MS`
- `#define EMBER_APSC_MAX_ACK_WAIT_TERMINAL_SECURITY_MS`

Functions

- `uint8_t emberMaximumApsPayloadLength (void)`

- `EmberStatus emberSendMulticastWithAlias (EmberApsFrame *apsFrame, uint8_t radius, uint8_t nonmemberRadius, EmberMessageBuffer message, EmberNodeId alias, uint8_t sequence)`
- `EmberStatus emberSendMulticast (EmberApsFrame *apsFrame, uint8_t radius, uint8_t nonmemberRadius, EmberMessageBuffer message)`
- `EmberStatus emberSendUnicast (EmberOutgoingMessageType type, uint16_t indexOrDestination, EmberApsFrame *apsFrame, EmberMessageBuffer message)`
- `EmberStatus emberSendBroadcast (EmberNodeId destination, EmberApsFrame *apsFrame, uint8_t radius, EmberMessageBuffer message)`
- `EmberStatus emberProxyBroadcast (EmberNodeId source, EmberNodeId destination, uint8_t sequence, EmberApsFrame *apsFrame, uint8_t radius, EmberMessageBuffer message)`
- `EmberStatus emberSendManyToOneRouteRequest (uint16_t concentratorType, uint8_t radius)`
- `uint8_t emberAppendSourceRouteHandler (EmberNodeId destination, EmberMessageBuffer header)`
- `void emberIncomingRouteRecordHandler (EmberNodeId source, EmberEUI64 sourceEui, uint8_t relayCount, EmberMessageBuffer header, uint8_t relayListIndex)`
- `void emberIncomingManyToOneRouteRequestHandler (EmberNodeId source, EmberEUI64 longId, uint8_t cost)`
- `void emberIncomingRouteErrorHandler (EmberStatus status, EmberNodeId target)`
- `EmberStatus emberCancelMessage (EmberMessageBuffer message)`
- `void emberMessageSentHandler (EmberOutgoingMessageType type, uint16_t indexOrDestination, EmberApsFrame *apsFrame, EmberMessageBuffer message, EmberStatus status)`
- `void emberIncomingMessageHandler (EmberIncomingMessageType type, EmberApsFrame *apsFrame, EmberMessageBuffer message)`
- `EmberStatus emberGetLastHopLqi (uint8_t *lastHopLqi)`
- `EmberStatus emberGetLastHopRssi (int8_t *lastHopRssi)`
- `EmberNodeId emberGetSender (void)`
- `EmberStatus emberGetSenderEui64 (EmberEUI64 senderEui64)`
- `EmberStatus emberSendReply (uint16_t clusterId, EmberMessageBuffer reply)`
- `void emberSetReplyFragmentData (uint16_t fragmentData)`
- `bool emberAddressTableEntryIsActive (uint8_t addressTableIndex)`
- `EmberStatus emberSetAddressTableRemoteEui64 (uint8_t addressTableIndex, EmberEUI64 eui64)`
- `void emberSetAddressTableRemoteNodeId (uint8_t addressTableIndex, EmberNodeId id)`
- `void emberGetAddressTableRemoteEui64 (uint8_t addressTableIndex, EmberEUI64 eui64)`
- `EmberNodeId emberGetAddressTableRemoteNodeId (uint8_t addressTableIndex)`
- `void emberSetExtendedTimeout (EmberEUI64 remoteEui64, bool extendedTimeout)`
- `bool emberGetExtendedTimeout (EmberEUI64 remoteEui64)`
- `void emberIdConflictHandler (EmberNodeId conflictingId)`
- `bool emberPendingAckedMessages (void)`
- `void emberIncomingCommandHandler (EmberZigbeeCommandType commandType, EmberMessageBuffer commandBuffer, uint8_t indexOfCommand, void *data)`

Variables

- `uint16_t emberApsAckTimeoutMs`
- `EmberMulticastTableEntry * emberMulticastTable`
- `uint8_t emberMulticastTableSize`

8.93.1 Detailed Description

EmberZNet API for sending and receiving messages. See [Sending and Receiving Messages](#) for documentation.

Definition in file [message.h](#).

8.94 message.h

```

00001
00024 uint8_t 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 uint16_t emberApsAckTimeoutMs;
00049
00083 EmberStatus emberSendMulticastWithAlias(
    EmberApsFrame *apsFrame,
00084                               uint8_t radius,
00085                               uint8_t nonmemberRadius,
00086                               EmberMessageBuffer
    message,
00087                               EmberNodeId alias,
00088                               uint8_t sequence);
00089
00119 EmberStatus emberSendMulticast(EmberApsFrame
    *apsFrame,
00120                               uint8_t radius,
00121                               uint8_t nonmemberRadius,
00122                               EmberMessageBuffer message);
00123
00181 EmberStatus emberSendUnicast(
    EmberOutgoingMessageType type,
00182                               uint16_t indexOrDestination,
00183                               EmberApsFrame *apsFrame,
00184                               EmberMessageBuffer message);
00185
00202 EmberStatus emberSendBroadcast(EmberNodeId
    destination,
00203                               EmberApsFrame *apsFrame,
00204                               uint8_t radius,
00205                               EmberMessageBuffer message);
00206
00227 EmberStatus emberProxyBroadcast(EmberNodeId
    source,
00228                               EmberNodeId destination,
00229                               uint8_t sequence,
00230                               EmberApsFrame *apsFrame,
00231                               uint8_t radius,
00232                               EmberMessageBuffer message);
00233
00286 EmberStatus emberSendManyToOneRouteRequest
    (uint16_t concentratorType,
00287                               uint8_t radius);
00288
00313 uint8_t emberAppendSourceRouteHandler(EmberNodeId
    destination,
00314                               EmberMessageBuffer header
    );
00315
00335 void emberIncomingRouteRecordHandler(EmberNodeId
    source,
00336                               EmberEUI64 sourceEui,
00337                               uint8_t relayCount,
00338                               EmberMessageBuffer
    header,
00339                               uint8_t relayListIndex);
00340
00354 void emberIncomingManyToOneRouteRequestHandler
    (EmberNodeId source,
00355                               EmberEUI64 longId,
00356                               uint8_t cost);

```

```

00357
00403 void emberIncomingRouteErrorHandler(EmberStatus
00404     status,
00405             EmberNodeId target);
00412 EmberStatus emberCancelMessage(EmberMessageBuffer
00413     message);
00413 void emberMessageSentHandler(EmberOutgoingMessageType
00414     type,
00415             uint16_t indexOrDestination,
00416             EmberApsFrame *apsFrame,
00417             EmberMessageBuffer message,
00418             EmberStatus status);
00419
00420 void emberIncomingMessageHandler(
00421     EmberIncomingMessageType type,
00422             EmberApsFrame *apsFrame,
00423             EmberMessageBuffer message);
00424
00425 EmberStatus emberGetLastHopLqi(uint8_t *lastHopLqi
00426 );
00427
00428 EmberStatus emberGetLastHopRssi(int8_t *
00429     lastHopRssi);
00430
00431 EmberNodeId emberGetSender(void);
00432
00433 EmberStatus emberGetSenderEui64(EmberEUI64
00434     senderEui64);
00435
00436 EmberStatus emberSendReply(uint16_t clusterId,
00437     EmberMessageBuffer reply);
00438
00439 void emberSetReplyFragmentData(uint16_t fragmentData);
00440
00441
00442 bool emberAddressTableEntryIsActive(uint8_t
00443     addressTableIndex);
00444
00445 EmberStatus emberSetAddressTableRemoteEui64
00446     (uint8_t addressTableIndex,
00447             EmberEUI64 eui64);
00448
00449
00450 void emberSetAddressTableRemoteNodeId(uint8_t
00451     addressTableIndex,
00452             EmberNodeId id);
00453
00454 void emberGetAddressTableRemoteEui64(uint8_t
00455     addressTableIndex,
00456             EmberEUI64 eui64);
00457
00458 EmberNodeId emberGetAddressTableRemoteNodeId
00459     (uint8_t addressTableIndex);
00460
00461 void emberSetExtendedTimeout(EmberEUI64
00462     remoteEui64, bool extendedTimeout);
00463
00464 bool emberGetExtendedTimeout(EmberEUI64
00465     remoteEui64);
00466
00467 void emberIdConflictHandler(EmberNodeId
00468     conflictingId);
00469
00470 bool emberPendingAckedMessages(void);
00471
00472 extern EmberMulticastTableEntry *emberMulticastTable
00473 ;
00474
00475 extern uint8_t emberMulticastTableSize;
00476
00477 void emberIncomingCommandHandler(
00478     EmberZigbeeCommandType commandType,
00479     EmberMessageBuffer
00480     commandBuffer,
00481             uint8_t indexOfCommand,
00482             void *data);
00483
00484

```

8.95 mfglib.h File Reference

Functions

- [EmberStatus mfglibStart](#) (void(*mfglibRxCallback)(uint8_t *packet, uint8_t link-Quality, int8_t rssi))
- [EmberStatus mfglibEnd](#) (void)
- [EmberStatus mfglibStartTone](#) (void)
- [EmberStatus mfglibStopTone](#) (void)
- [EmberStatus mfglibStartStream](#) (void)
- [EmberStatus mfglibStopStream](#) (void)
- [EmberStatus mfglibSendPacket](#) (uint8_t *packet, uint16_t repeat)
- [EmberStatus mfglibSetChannel](#) (uint8_t chan)
- [uint8_t mfglibGetChannel](#) (void)
- [EmberStatus mfglibSetPower](#) (uint16_t txPowerMode, int8_t power)
- [int8_t mfglibGetPower](#) (void)
- [void mfglibSetSynOffset](#) (int8_t synOffset)
- [int8_t mfglibGetSynOffset](#) (void)
- [void mfglibTestContModCal](#) (uint8_t channel, uint32_t duration)

8.95.1 Detailed Description

See [Manufacturing and Functional Test Library](#) for documentation.

Definition in file [mfglib.h](#).

8.96 mfglib.h

```

00001
00030 #ifndef __MFGLIB_H__
00031 #define __MFGLIB_H__
00032
00055 EmberStatus mfglibStart(void (*mfglibRxCallback)(uint8_t
    *packet, uint8_t linkQuality, int8_t 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(uint8_t * packet,
    uint16_t repeat);
00144
00158 EmberStatus mfglibSetChannel(uint8_t chan);
00159
00166 uint8_t mfglibGetChannel(void);
00167
00186 EmberStatus mfglibSetPower(uint16_t txPowerMode,int8_t
    power);
00187
00194 int8_t mfglibGetPower(void);
00195
00212 void mfglibSetSynOffset(int8_t synOffset);
00213
00218 int8_t mfglibGetSynOffset(void);
00219
00235 void mfglibTestContModCal(uint8_t channel, uint32_t

```

```

        duration);
00236 #endif // __MFGLIB_H__
00238

```

8.97 micro-common.h File Reference

Macros

- `#define MICRO_DISABLE_WATCH_DOG_KEY`
- `#define GPIO_MASK_SIZE`
- `#define GPIO_MASK`
- `#define WAKE_GPIO_MASK`
- `#define WAKE_GPIO_SIZE`
- `#define WAKE_MASK_INVALID`
- `#define WAKE_EVENT_SIZE`
- `#define DEBUG_TOGGLE(n)`

Typedefs

- `typedef uint32_t WakeEvents`
- `typedef uint32_t WakeMask`

Enumerations

- `enum SleepModes {
 SLEEPMODE_RUNNING, SLEEPMODE_IDLE, SLEEPMODE_WAKETIMER,
 SLEEPMODE_MAINTAINTIMER,
 SLEEPMODE_NOTIMER, SLEEPMODE_HIBERNATE, SLEEPMODE_RESERVED,
 SLEEPMODE_POWERDOWN,
 SLEEPMODE_POWERSAVE }`

Functions

- `void halInit (void)`
- `void halReboot (void)`
- `void halPowerUp (void)`
- `void halPowerDown (void)`
- `void halResume (void)`
- `void halSuspend (void)`
- `void halInternalEnableWatchDog (void)`
- `void halInternalDisableWatchDog (uint8_t magicKey)`
- `bool halInternalWatchDogEnabled (void)`
- `void halSleep (SleepModes sleepMode)`
- `void halCommonDelayMicroseconds (uint16_t us)`
- `void halCommonDisableVreg1v8 (void)`
- `void halCommonEnableVreg1v8 (void)`

Variables

- volatile int8_t `halCommonVreg1v8EnableCount`

8.97.1 Detailed Description

Minimal Hal functions common across all microcontroller-specific files. See [Common Microcontroller Functions](#) for documentation.

Definition in file [micro-common.h](#).

8.98 micro-common.h

```

00001
00016 #ifndef __MICRO_COMMON_H__
00017 #define __MICRO_COMMON_H__
00018
00019
00022 void halInit(void);
00023
00026 void halReboot(void);
00027
00030 void halPowerUp(void);
00031
00034 void halPowerDown(void);
00035
00038 void halResume(void);
00039
00042 void halSuspend(void);
00043
00044
00049 #define MICRO_DISABLE_WATCH_DOG_KEY 0xA5
00050
00053 void halInternalEnableWatchDog(void);
00054
00062 void halInternalDisableWatchDog(uint8_t magicKey);
00063
00068 bool halInternalWatchDogEnabled( void );
00069
00070 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00071
00098 enum SleepModes
00099 #else
00100 typedef uint8_t SleepModes;
00101 enum
00102 #endif
00103 {
00104     SLEEPMODE_RUNNING = 0,
00105     SLEEPMODE_IDLE = 1,
00106     SLEEPMODE_WAKETIMER = 2,
00107     SLEEPMODE_MAINTAINTIMER = 3,
00108     SLEEPMODE_NOTIMER = 4,
00109     SLEEPMODE_HIBERNATE = 5,
00110
00111     //The following SleepModes are deprecated on EM2xx and EM3xx chips.  Each
00112     //micro's halSleep() function will remap these modes to the appropriate
00113     //replacement, as necessary.
00114     SLEEPMODE_RESERVED = 6,
00115     SLEEPMODE_POWERDOWN = 7,
00116     SLEEPMODE_POWERSAVE = 8,
00117
00118 };
00119
00120
00121
00122 #ifdef CORTEXM3_EMBER_MICRO
00123     #ifdef EMBER_MICRO_PORT_F_GPIO
00124         #define WAKEPORTD      uint8_t portD;
00125         #define WAKEPORTE     uint8_t portE;
00126         #define WAKEPORTF     uint8_t portF;
00127         #define GPIO_MASK_SIZE 48

```

```

00128     #define GPIO_MASK          0xFFFFFFFFFFFFFF
00129     typedef uint64_t GpioMaskType;
00130     typedef uint64_t WakeMask;
00131 #elif defined (EMBER_MICRO_PORT_E_GPIO)
00132     #define WAKEPORTD          uint8_t portD;
00133     #define WAKEPORTE          uint8_t portE;
00134     #define WAKEPORTF          -
00135     #define GPIO_MASK_SIZE     40
00136     #define GPIO_MASK          0xFFFFFFFFFFF
00137     typedef uint64_t GpioMaskType;
00138     typedef uint64_t WakeMask;
00139 #elif defined (EMBER_MICRO_PORT_D_GPIO)
00140     #define WAKEPORTD          uint8_t portD;
00141     #define WAKEPORTE          -
00142     #define WAKEPORTF          -
00143     #define GPIO_MASK_SIZE     32
00144     #define GPIO_MASK          0xFFFFFFFF
00145     typedef uint32_t GpioMaskType;
00146     typedef uint64_t WakeMask;
00147 #else
00148     #define WAKEPORTD
00149     #define WAKEPORTE
00150     #define WAKEPORTF
00151     #define GPIO_MASK_SIZE     24
00152     #define GPIO_MASK          0xFFFFFFF
00153     typedef uint32_t GpioMaskType;
00154     typedef uint32_t WakeMask;
00155 #endif
00156
00157 #define WAKE_GPIO_MASK        GPIO_MASK
00158 #define WAKE_GPIO_SIZE        GPIO_MASK_SIZE
00159
00160 //We don't have a real register to hold this composite information.
00161 //Pretend we do so halGetWakeInfo can operate like halGetResetInfo.
00162 //This "register" is only ever set by halSleep.
00163 // [31] = WakeInfoValid
00164 // [30] = SleepSkipped
00165 // [29] = CSYSPWRUPREQ
00166 // [28] = CDBGPWRUPREQ
00167 // [27] = WAKE_CORE
00168 // [26] = TIMER_WAKE_WRAP
00169 // [25] = TIMER_WAKE_COMPB
00170 // [24] = TIMER_WAKE_COMPA
00171 // [23:0] = corresponding GPIO activity
00172 #define WAKEINFOINVALID_INTERNAL_WAKE_EVENT_BIT (GPIO_MASK_SIZE+7)
00173 #define SLEEP_SKIPPED_INTERNAL_WAKE_EVENT_BIT (GPIO_MASK_SIZE+6)
00174 #define CSYSPWRUPREQ_INTERNAL_WAKE_EVENT_BIT (GPIO_MASK_SIZE+5)
00175 #define CDBGPWRUPREQ_INTERNAL_WAKE_EVENT_BIT (GPIO_MASK_SIZE+4)
00176 #define WAKE_CORE_INTERNAL_WAKE_EVENT_BIT (GPIO_MASK_SIZE+3)
00177 #define WRAP_INTERNAL_WAKE_EVENT_BIT (GPIO_MASK_SIZE+2)
00178 #define CMPB_INTERNAL_WAKE_EVENT_BIT (GPIO_MASK_SIZE+1)
00179 #define CMPA_INTERNAL_WAKE_EVENT_BIT (GPIO_MASK_SIZE+0)
00180 //This define shifts events from the PWRUP_EVENT register into the proper
00181 //place in the halInternalWakeEvent variable
00182 #define INTERNAL_WAKE_EVENT_BIT_SHIFT (GPIO_MASK_SIZE-4)
00183 typedef union
00184 {
00185     struct
00186     {
00187         uint8_t portA;
00188         uint8_t portB;
00189         uint8_t portC;
00190         WAKEPORTD
00191         WAKEPORTE
00192         WAKEPORTF
00193         union {
00194             struct
00195             {
00196                 uint8_t TIMER_WAKE_COMPA : 1;
00197                 uint8_t TIMER_WAKE_COMPB : 1;
00198                 uint8_t TIMER_WAKE_WRAP : 1;
00199                 uint8_t WAKE_CORE_B : 1;
00200                 uint8_t CDBGPWRUPREQ : 1;
00201                 uint8_t CSYSPWRUPREQ : 1;
00202                 uint8_t SleepSkipped : 1;
00203                 uint8_t WakeInfoValid : 1;
00204             } bits;
00205             uint8_t byte;
00206         } internal;
00207     } events;

```

```

00208     WakeMask eventflags;
00209 } WakeEvents;
00210 #undef WAKEPORTD
00211 #undef WAKEPORTE
00212 #undef WAKEPORTF
00213 #elif defined (CORTEXM3_EFM32_MICRO)
00214 #define WAKE_GPIO_SIZE    16
00215 #define WAKE_GPIO_MASK     0x0000FFFF
00216 #define WAKE_IRQ_GPIO      (1 << 16)
00217 #define WAKE_IRQ_SYSTIMER   (1 << 17)
00218 #define WAKE_IRQ_RFSENSE    (1 << 18)
00219
00220 typedef uint32_t WakeEvents;
00221 typedef uint32_t WakeMask;
00222 #else // to cover simulation targets
00223 #define GPIO_MASK_SIZE    24
00224 #define GPIO_MASK          0xFFFFFFFF
00225 #define WAKE_GPIO_MASK      GPIO_MASK
00226 #define WAKE_GPIO_SIZE       GPIO_MASK_SIZE
00227 typedef uint32_t WakeEvents;
00228 typedef uint32_t WakeMask;
00229 #endif // CORTEXM3_EMBER_MICRO
00230
00231 #define WAKE_MASK_INVALID (-1)
00232
00233 #define WAKE_EVENT_SIZE     WakeMask
00234
00235 void halSleep(SleepModes sleepMode);
00236
00237 void halCommonDelayMicroseconds(uint16_t us);
00238
00239 #define DEBUG_TOGGLE(n)
00240
00241
00242
00243 extern volatile int8_t halCommonVreg1v8EnableCount;
00244
00245 void halCommonDisableVreg1v8(void);
00246
00247 void halCommonEnableVreg1v8(void);
00248
00249 #endif // __MICRO_COMMON_H__
00300

```

8.99 micro-types.h File Reference

8.99.1 Detailed Description

This file handles defines and enums related to all the micros. THIS IS A GENERATED FILE. DO NOT EDIT.

Definition in file [micro-types.h](#).

8.100 micro-types.h

```

00001
00002 #ifndef __MICRO_DEFINES_H__
00003 #define __MICRO_DEFINES_H__
00004
00005 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00006
00007 // Below is a list of PLATFORM and MICRO values that are used to define the
00008 // chips our code runs on. These values are used in EBL headers, bootloader
00009 // query response messages, and possibly other places in the code
00010 // -----
00011 // PLAT 1 was the AVR ATmega, no longer supported (still used for EMBER_TEST)
00012
00013 // PLAT 2 is the XAP2b

```

```

00023 // for PLAT 2, MICRO 1 is the em250
00024 // for PLAT 2, MICRO 2 is the em260
00025
00026 // PLAT 4 is the CortexM3
00027 // for PLAT 4, MICRO 1 is the em350
00028 // for PLAT 4, MICRO 2 is the em360
00029 // for PLAT 4, MICRO 3 is the em357
00030 // for PLAT 4, MICRO 4 is the em367
00031 // for PLAT 4, MICRO 5 is the em351
00032 // for PLAT 4, MICRO 6 is the em35x
00033 // for PLAT 4, MICRO 8 is the em3588
00034 // for PLAT 4, MICRO 9 is the em359
00035 // for PLAT 4, MICRO 10 is the em3581
00036 // for PLAT 4, MICRO 11 is the em3582
00037 // for PLAT 4, MICRO 12 is the em3585
00038 // for PLAT 4, MICRO 13 is the em3586
00039 // for PLAT 4, MICRO 14 is the em3587
00040 // for PLAT 4, MICRO 15 is the sky66107
00041 // for PLAT 4, MICRO 16 is the em3597
00042 // for PLAT 4, MICRO 17 is the em356
00043 // for PLAT 4, MICRO 18 is the em3598
00044 // for PLAT 4, MICRO 19 is the em3591
00045 // for PLAT 4, MICRO 20 is the em3592
00046 // for PLAT 4, MICRO 21 is the em3595
00047 // for PLAT 4, MICRO 22 is the em3596
00048 // for PLAT 4, MICRO 23 is the em317
00049 // for PLAT 4, MICRO 24 is the efr32
00050 // for PLAT 4, MICRO 25 is the em341
00051 // for PLAT 4, MICRO 26 is the em342
00052 // for PLAT 4, MICRO 27 is the em346
00053 // for PLAT 4, MICRO 30 is the em355
00054 // for PLAT 4, MICRO 31 is the em3555
00055 // for PLAT 4, MICRO 32 is the ezs32hg
00056 // for PLAT 4, MICRO 33 is the ezs32lg
00057 // for PLAT 4, MICRO 34 is the ezs32wg
00058
00059 // PLAT 5 is the C8051
00060 // for PLAT 5, MICRO 1 is the siF930
00061 // for PLAT 5, MICRO 2 is the siF960
00062 // for PLAT 5, MICRO 3 is the cobra
00063 // for PLAT 5, MICRO 4 is the siF370
00064 // for PLAT 5, MICRO 5 is the siF393
00065
00066 // PLAT 6 is the FFD
00067 // for PLAT 6, MICRO 1 is the si4010
00068
00069 //

-----
00070 // FAMILY values are used in EBL headers and bootloaders as well
00071 // FAMILY 16 is the EFR32MG1P
00072 // FAMILY 17 is the EFR32MG1B
00073 // FAMILY 18 is the EFR32MG1V
00074 // FAMILY 19 is the EFR32BG1P
00075 // FAMILY 20 is the EFR32BG1B
00076 // FAMILY 21 is the EFR32BG1V
00077 // FAMILY 22 is the EFR32ZG1P
00078 // FAMILY 23 is the EFR32ZG1B
00079 // FAMILY 24 is the EFR32ZG1V
00080 // FAMILY 25 is the EFR32FG1P
00081 // FAMILY 27 is the EFR32FG1V
00082 //

-----
00083
00084 // Create an enum for the platform types
00085 enum {
00086     EMBER_PLATFORM_UNKNOWN      = 0,
00087     EMBER_PLATFORM_AVR_ATMEGA = 1,
00088     EMBER_PLATFORM_XAP2B       = 2,
00089     EMBER_PLATFORM_CORTEXM3    = 4,
00090     EMBER_PLATFORM_C8051      = 5,
00091     EMBER_PLATFORM_FFD        = 6,
00092     EMBER_PLATFORM_MAX_VALUE  = 7
00093 };
00094 typedef uint16_t EmberPlatformEnum;
00095
00096 #define EMBER_PLATFORM_STRINGS \
00097     "Unknown", \
00098     "Test", \
00099     "XAP2B", \
0100     "CortexM3",

```

```

00101 "C8051",
00102 "FFD",
00103 NULL,
00104
00105 // Create an enum for all of the different AVR ATMega micros we support
00106 enum {
00107     EMBER_MICRO_AVR_ATMEGA_UNKNOWN    = 0,
00108     EMBER_MICRO_AVR_ATMEGA_MAX_VALUE
00109 };
00110 typedef uint16_t EmberMicroAvrAtmegaEnum;
00111
00112 #define EMBER_MICRO_AVR_ATMEGA_STRINGS \
00113     "Unknown", \
00114     NULL,
00115
00116 // Create an enum for all of the different XAP2b micros we support
00117 enum {
00118     EMBER_MICRO_XAP2B_UNKNOWN    = 0,
00119     EMBER_MICRO_XAP2B_EM250      = 1,
00120     EMBER_MICRO_XAP2B_EM260      = 2,
00121     EMBER_MICRO_XAP2B_MAX_VALUE
00122 };
00123 typedef uint16_t EmberMicroXap2bEnum;
00124
00125 #define EMBER_MICRO_XAP2B_STRINGS \
00126     "Unknown", \
00127     "em250", \
00128     "em260", \
00129     NULL,
00130
00131 // Create an enum for all of the different CortexM3 micros we support
00132 enum {
00133     EMBER_MICRO_CORTEXM3_UNKNOWN    = 0,
00134     EMBER_MICRO_CORTEXM3_EM350      = 1,
00135     EMBER_MICRO_CORTEXM3_EM360      = 2,
00136     EMBER_MICRO_CORTEXM3_EM357      = 3,
00137     EMBER_MICRO_CORTEXM3_EM367      = 4,
00138     EMBER_MICRO_CORTEXM3_EM351      = 5,
00139     EMBER_MICRO_CORTEXM3_EM35X      = 6,
00140     EMBER_MICRO_CORTEXM3_EM3588     = 8,
00141     EMBER_MICRO_CORTEXM3_EM359      = 9,
00142     EMBER_MICRO_CORTEXM3_EM3581     = 10,
00143     EMBER_MICRO_CORTEXM3_EM3582     = 11,
00144     EMBER_MICRO_CORTEXM3_EM3585     = 12,
00145     EMBER_MICRO_CORTEXM3_EM3586     = 13,
00146     EMBER_MICRO_CORTEXM3_EM3587     = 14,
00147     EMBER_MICRO_CORTEXM3_SKY66107   = 15,
00148     EMBER_MICRO_CORTEXM3_EM3597     = 16,
00149     EMBER_MICRO_CORTEXM3_EM356      = 17,
00150     EMBER_MICRO_CORTEXM3_EM3598     = 18,
00151     EMBER_MICRO_CORTEXM3_EM3591     = 19,
00152     EMBER_MICRO_CORTEXM3_EM3592     = 20,
00153     EMBER_MICRO_CORTEXM3_EM3595     = 21,
00154     EMBER_MICRO_CORTEXM3_EM3596     = 22,
00155     EMBER_MICRO_CORTEXM3_EM317      = 23,
00156     EMBER_MICRO_CORTEXM3_EFR32      = 24,
00157     EMBER_MICRO_CORTEXM3_EM341      = 25,
00158     EMBER_MICRO_CORTEXM3_EM342      = 26,
00159     EMBER_MICRO_CORTEXM3_EM346      = 27,
00160     EMBER_MICRO_CORTEXM3_EM355      = 30,
00161     EMBER_MICRO_CORTEXM3_EM3555     = 31,
00162     EMBER_MICRO_CORTEXM3_EZR32HG    = 32,
00163     EMBER_MICRO_CORTEXM3_EZR32LG    = 33,
00164     EMBER_MICRO_CORTEXM3_EZR32WG    = 34,
00165     EMBER_MICRO_CORTEXM3_MAX_VALUE
00166 };
00167 typedef uint16_t EmberMicroCortexM3Enum;
00168
00169 #define EMBER_MICRO_CORTEXM3_STRINGS \
00170     "Unknown", \
00171     "em350", \
00172     "em360", \
00173     "em357", \
00174     "em367", \
00175     "em351", \
00176     "em35x", \
00177     "em3588", \
00178     "em359", \
00179     "em3581", \
00180     "em3582",

```

```

00181     "em3585",
00182     "em3586",
00183     "em3587",
00184     "sky66107",
00185     "em3597",
00186     "em356",
00187     "em3598",
00188     "em3591",
00189     "em3592",
00190     "em3595",
00191     "em3596",
00192     "em317",
00193     "efr32",
00194     "em341",
00195     "em342",
00196     "em346",
00197     "em355",
00198     "em3555",
00199     "ezr32hg",
00200     "ezr32lg",
00201     "ezr32wg",
00202     NULL,
00203
00204 // Create an enum for all of the different C8051 micros we support
00205 enum {
00206     EMBER_MICRO_C8051_UNKNOWN    = 0,
00207     EMBER_MICRO_C8051_SIF930    = 1,
00208     EMBER_MICRO_C8051_SIF960    = 2,
00209     EMBER_MICRO_C8051_COBRA     = 3,
00210     EMBER_MICRO_C8051_SIF370    = 4,
00211     EMBER_MICRO_C8051_SIF393    = 5,
00212     EMBER_MICRO_C8051_MAX_VALUE
00213 };
00214 typedef uint16_t EmberMicroC8051Enum;
00215
00216 #define EMBER_MICRO_C8051_STRINGS \
00217     "Unknown", \
00218     "sif930", \
00219     "sif960", \
00220     "cobra", \
00221     "sif370", \
00222     "sif393", \
00223     NULL,
00224
00225 // Create an enum for all of the different FFD micros we support
00226 enum {
00227     EMBER_MICRO_FFD_UNKNOWN    = 0,
00228     EMBER_MICRO_FFD_SI4010    = 1,
00229     EMBER_MICRO_FFD_MAX_VALUE
00230 };
00231 typedef uint16_t EmberMicroFfdEnum;
00232
00233 #define EMBER_MICRO_FFD_STRINGS \
00234     "Unknown", \
00235     "si4010", \
00236     NULL,
00237
00238 // Create an enum for the platform types
00239 enum {
00240     EMBER_FAMILY_UNKNOWN    = 0,
00241     EMBER_FAMILY_EFR32MG1P   = 16,
00242     EMBER_FAMILY_EFR32MG1B   = 17,
00243     EMBER_FAMILY_EFR32MG1V   = 18,
00244     EMBER_FAMILY_EFR32BG1P   = 19,
00245     EMBER_FAMILY_EFR32BG1B   = 20,
00246     EMBER_FAMILY_EFR32BG1V   = 21,
00247     EMBER_FAMILY_EFR32ZG1P   = 22,
00248     EMBER_FAMILY_EFR32ZG1B   = 23,
00249     EMBER_FAMILY_EFR32ZG1V   = 24,
00250     EMBER_FAMILY_EFR32FG1P   = 25,
00251     EMBER_FAMILY_EFR32FG1V   = 27,
00252     EMBER_FAMILY_MAX_VALUE
00253 };
00254 typedef uint16_t EmberFamilyEnum;
00255
00256 #define EMBER_FAMILY_STRINGS \
00257     "Unknown", \
00258     "EFR32MG1P", \
00259     "EFR32MG1B", \
00260     "EFR32MG1V"

```

```

00261 "EFR32BG1P",
00262 "EFR32BG1B",
00263 "EFR32BG1V",
00264 "EFR32ZG1P",
00265 "EFR32ZG1B",
00266 "EFR32ZG1V",
00267 "EFR32FG1P",
00268 "EFR32FG1V",
00269 NULL,
00270
00271 // A dummy type declared to allow generically passing around this
00272 // data type. Micro specific enums (such as EmberMicroCortexM3Enum)
00273 // are required to be the same size as this.
00274 typedef uint16_t EmberMicroEnum;
00275
00276 // Determine what micro and platform that we're supposed to target using the
00277 // defines passed in at build time. Then set the PLAT and MICRO defines based
00278 // on what was passed in
00279 #if (!(defined(EZSP_HOST)) && (!defined(UNIX_HOST)))
00280
00281 #if defined(EMBER_TEST)
00282     #define PLAT EMBER_PLATFORM_AVR_ATMEGA
00283     #define MICRO 2
00284 #elif defined(AVR_ATMEGA)
00285     #define PLAT EMBER_PLATFORM_AVR_ATMEGA
00286 #elif defined(XAP2B)
00287     #define PLAT EMBER_PLATFORM_XAP2B
00288     #if defined(XAP2B_EM250)
00289         #define MICRO EMBER_MICRO_XAP2B_EM250
00290     #elif defined(XAP2B_EM260)
00291         #define MICRO EMBER_MICRO_XAP2B_EM260
00292     #endif
00293 #elif defined(CORTEXM3)
00294     #define PLAT EMBER_PLATFORM_CORTEXM3
00295     #if defined(CORTEXM3_EM350)
00296         #define MICRO EMBER_MICRO_CORTEXM3_EM350
00297     #elif defined(CORTEXM3_EM360)
00298         #define MICRO EMBER_MICRO_CORTEXM3_EM360
00299     #elif defined(CORTEXM3_EM357)
00300         #define MICRO EMBER_MICRO_CORTEXM3_EM357
00301     #elif defined(CORTEXM3_EM367)
00302         #define MICRO EMBER_MICRO_CORTEXM3_EM367
00303     #elif defined(CORTEXM3_EM351)
00304         #define MICRO EMBER_MICRO_CORTEXM3_EM351
00305     #elif defined(CORTEXM3_EM35X)
00306         #define MICRO EMBER_MICRO_CORTEXM3_EM35X
00307     #elif defined(CORTEXM3_EM3588)
00308         #define MICRO EMBER_MICRO_CORTEXM3_EM3588
00309     #elif defined(CORTEXM3_EM359)
00310         #define MICRO EMBER_MICRO_CORTEXM3_EM359
00311     #elif defined(CORTEXM3_EM3581)
00312         #define MICRO EMBER_MICRO_CORTEXM3_EM3581
00313     #elif defined(CORTEXM3_EM3582)
00314         #define MICRO EMBER_MICRO_CORTEXM3_EM3582
00315     #elif defined(CORTEXM3_EM3585)
00316         #define MICRO EMBER_MICRO_CORTEXM3_EM3585
00317     #elif defined(CORTEXM3_EM3586)
00318         #define MICRO EMBER_MICRO_CORTEXM3_EM3586
00319     #elif defined(CORTEXM3_EM3587)
00320         #define MICRO EMBER_MICRO_CORTEXM3_EM3587
00321     #elif defined(CORTEXM3_SKY66107)
00322         #define MICRO EMBER_MICRO_CORTEXM3_SKY66107
00323     #elif defined(CORTEXM3_EM3597)
00324         #define MICRO EMBER_MICRO_CORTEXM3_EM3597
00325     #elif defined(CORTEXM3_EM356)
00326         #define MICRO EMBER_MICRO_CORTEXM3_EM356
00327     #elif defined(CORTEXM3_EM3598)
00328         #define MICRO EMBER_MICRO_CORTEXM3_EM3598
00329     #elif defined(CORTEXM3_EM3591)
00330         #define MICRO EMBER_MICRO_CORTEXM3_EM3591
00331     #elif defined(CORTEXM3_EM3592)
00332         #define MICRO EMBER_MICRO_CORTEXM3_EM3592
00333     #elif defined(CORTEXM3_EM3595)
00334         #define MICRO EMBER_MICRO_CORTEXM3_EM3595
00335     #elif defined(CORTEXM3_EM3596)
00336         #define MICRO EMBER_MICRO_CORTEXM3_EM3596
00337     #elif defined(CORTEXM3_EM317)
00338         #define MICRO EMBER_MICRO_CORTEXM3_EM317
00339     #elif defined(CORTEXM3_EFR32)
00340         #define MICRO EMBER_MICRO_CORTEXM3_EFR32

```

```

00341 #elif defined(CORTEXM3_EM341)
00342     #define MICRO EMBER_MICRO_CORTEXM3_EM341
00343 #elif defined(CORTEXM3_EM342)
00344     #define MICRO EMBER_MICRO_CORTEXM3_EM342
00345 #elif defined(CORTEXM3_EM346)
00346     #define MICRO EMBER_MICRO_CORTEXM3_EM346
00347 #elif defined(CORTEXM3_EM355)
00348     #define MICRO EMBER_MICRO_CORTEXM3_EM355
00349 #elif defined(CORTEXM3_EM3555)
00350     #define MICRO EMBER_MICRO_CORTEXM3_EM3555
00351 #elif defined(CORTEXM3_EZR32HG)
00352     #define MICRO EMBER_MICRO_CORTEXM3_EZR32HG
00353 #elif defined(CORTEXM3_EZR32LG)
00354     #define MICRO EMBER_MICRO_CORTEXM3_EZR32LG
00355 #elif defined(CORTEXM3_EZR32WG)
00356     #define MICRO EMBER_MICRO_CORTEXM3_EZR32WG
00357 #endif
00358 #elif defined(C8051)
00359     #define PLAT EMBER_PLATFORM_C8051
00360 #if defined(C8051_SIF930)
00361     #define MICRO EMBER_MICRO_C8051_SIF930
00362 #elif defined(C8051_SIF960)
00363     #define MICRO EMBER_MICRO_C8051_SIF960
00364 #elif defined(C8051_COBRA)
00365     #define MICRO EMBER_MICRO_C8051_COBRA
00366 #elif defined(C8051_SIF370)
00367     #define MICRO EMBER_MICRO_C8051_SIF370
00368 #elif defined(C8051_SIF393)
00369     #define MICRO EMBER_MICRO_C8051_SIF393
00370 #endif
00371 #elif defined(FFD)
00372     #define PLAT EMBER_PLATFORM_FFD
00373 #if defined(FFD_SI4010)
00374     #define MICRO EMBER_MICRO_FFD_SI4010
00375 #endif
00376 #endif
00377
00378 #endif // ((! defined(EZSP_HOST)) && (! defined(UNIX_HOST)))
00379
00380 // Determine what family we're supposed to target using the defines passed in
00381 // at
00382 // build time. Then set the FAMILY define based on what was passed in.
00383 // Parts without defined family will be assigned id 0.
00384 #if defined(EFR32MG1P)
00385     #define FAMILY EMBER_FAMILY_EFR32MG1P
00386 #elif defined(EFR32MG1B)
00387     #define FAMILY EMBER_FAMILY_EFR32MG1B
00388 #elif defined(EFR32MG1V)
00389 #elif defined(EFR32BG1P)
00390     #define FAMILY EMBER_FAMILY_EFR32BG1P
00391 #elif defined(EFR32BG1B)
00392     #define FAMILY EMBER_FAMILY_EFR32BG1B
00393 #elif defined(EFR32BG1V)
00394     #define FAMILY EMBER_FAMILY_EFR32BG1V
00395 #elif defined(EFR32ZG1P)
00396     #define FAMILY EMBER_FAMILY_EFR32ZG1P
00397 #elif defined(EFR32ZG1B)
00398     #define FAMILY EMBER_FAMILY_EFR32ZG1B
00399 #elif defined(EFR32ZG1V)
00400     #define FAMILY EMBER_FAMILY_EFR32ZG1V
00401 #elif defined(EFR32FG1P)
00402     #define FAMILY EMBER_FAMILY_EFR32FG1P
00403 #elif defined(EFR32FG1V)
00404     #define FAMILY EMBER_FAMILY_EFR32FG1V
00405 #else
00406     #define FAMILY 0
00407 #endif
00408
00409 // Define distinct literal values for each phy. These values are used in the
00410 // bootloader query response message.
00411 // PHY 0 is NULL
00412 // PHY 1 is em2420 (no longer supported)
00413 // PHY 2 is em250
00414 // PHY 3 is em3XX
00415 // PHY 4 is si446x
00416 // PHY 5 is cobra
00417 // PHY 6 is PRO2+
00418 // PHY 7 is si4x55
00419 // PHY 8 is si4010

```

```

00420 // PHY 9 is efr32
00421 // PHY 10 is pro2
00422 // PHY 11 is ezs2
00423
00424 enum {
00425     EMBER_PHY_NULL      = 0,
00426     EMBER_PHY_EM2420     = 1,
00427     EMBER_PHY_EM250      = 2,
00428     EMBER_PHY_EM3XX      = 3,
00429     EMBER_PHY_SI446X     = 4,
00430     EMBER_PHY_COBRA      = 5,
00431     EMBER_PHY_PRO2PLUS    = 6,
00432     EMBER_PHY_SI4X55      = 7,
00433     EMBER_PHY_SI4010      = 8,
00434     EMBER_PHY_EFR32      = 9,
00435     EMBER_PHY_PRO2        = 10,
00436     EMBER_PHY_EZR2        = 11,
00437     EMBER_PHY_MAX_VALUE
00438 };
00439 typedef uint16_t EmberPhyEnum;
00440
00441 #define EMBER_PHY_STRINGS \
00442     "NULL", \
00443     "em2420", \
00444     "em250", \
00445     "em3XX", \
00446     "si446x", \
00447     "cobra", \
00448     "PRO2+", \
00449     "si4x55", \
00450     "si4010", \
00451     "efr32", \
00452     "pro2", \
00453     "ezr2", \
00454     NULL, \
00455
00456 #if defined(PHY_NULL)
00457     #define PHY EMBER_PHY_NULL
00458 #elif defined(PHY_EM2420) || defined(PHY_EM2420B)
00459     #error em2420 PHY is no longer supported
00460 #elif defined(PHY_EM250) || defined(PHY_EM250B)
00461     #define PHY EMBER_PHY_EM250
00462 #elif defined(PHY_EM3XX)
00463     #define PHY EMBER_PHY_EM3XX
00464 #elif defined(PHY_SI446X_US) || defined(PHY_SI446X_EU) || \
00465     defined(PHY_SI446X_CN) || defined(PHY_SI446X_JP)
00466     #define PHY EMBER_PHY_SI446X
00466 #elif defined(PHY_COBRA)
00467     #define PHY EMBER_PHY_COBRA
00468 #elif defined(PHY_PRO2PLUS)
00469     #define PHY EMBER_PHY_PRO2PLUS
00470 #elif defined(PHY_SI4X55)
00471     #define PHY EMBER_PHY_SI4X55
00472 #elif defined(PHY_SI4010)
00473     #define PHY EMBER_PHY_SI4010
00474 #elif defined(PHY_EFR32)
00475     #define PHY EMBER_PHY_EFR32
00476 #elif defined(PHY_PRO2)
00477     #define PHY EMBER_PHY_PRO2
00478 #elif defined(PHY_EZR2)
00479     #define PHY EMBER_PHY_EZR2
00480 #endif
00481
00482 typedef struct {
00483     EmberPlatformEnum platform;
00484     EmberMicroEnum     micro;
00485     EmberPhyEnum       phy;
00486 } EmberChipTypeStruct;
00487
00488 // load up any chip-specific feature defines
00489 #if defined(PLAT) && defined(MICRO) && PLAT == EMBER_PLATFORM_CORTEXM3
00490     #include "cortexm3/micro-features.h"
00491 #endif
00492
00493 #endif // DOXYGEN_SHOULD_SKIP_THIS
00494
00495 #endif // __MICRO_DEFINES_H__

```

8.101 micro.h File Reference

```
#include "hal/micro/generic/em2xx-reset-defs.h"
#include "hal/micro/micro-types.h"
```

Macros

- #define halGetEm2xxResetInfo()

Functions

- void halStackProcessBootCount (void)
- uint8_t halGetResetInfo (void)
- PGM_P halGetResetString (void)

8.101.1 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](#).

8.102 micro.h

```
00001
00020 #ifndef __MICRO_H__
00021 #define __MICRO_H__
00022
00023 #include "hal/micro/generic/em2xx-reset-defs.h"
00024 #include "hal/micro/micro-types.h"
00025
00026 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00027
00028 // Make sure that a proper plat/micro combination was selected if we aren't
00029 // building for a host processor
00030 #if (!(defined(EZSP_HOST)) && (!defined(UNIX_HOST)))
00031
00032 #ifndef PLAT
00033     #error no platform defined, or unsupported
00034 #endif
00035 #ifndef MICRO
00036     #error no micro defined, or unsupported
00037 #endif
00038 #ifndef PHY
00039     #error no phy defined, or unsupported
00040 #endif
00041
00042 #endif // ((!defined(EZSP_HOST)) && (!defined(UNIX_HOST)))
00043
00044 #endif // DOXYGEN_SHOULD_SKIP_THIS
00045
00059 void halStackProcessBootCount(void);
00060
00065 uint8_t halGetResetInfo(void);
00066
00074 PGM_P halGetResetString(void);
00075
00085 #if defined(CORTEXM3) || defined(EMBER_STACK_COBRA)
00086     uint8_t halGetEm2xxResetInfo(void);
00087 #else
```

```

00088     #define halGetEm2xxResetInfo() halGetResetInfo()
00089 #endif
00090
00091 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00092
00093 #include "micro-common.h"
00094
00095 #if defined( EMBER_TEST )
00096     #include "hal/micro/unix/simulation/micro.h"
00097     #include "hal/micro/unix/simulation/bootloader.h"
00098 #elif defined(CORTEXM3_EMBER_MICRO)
00099     #include "cortexm3/micro.h"
00100 #elif defined(CORTEXM3_EFM32_MICRO)
00101     #include "cortexm3/efm32/micro.h"
00102 #elif defined(C8051)
00103     #include "c8051/micro.h"
00104 #elif defined(FFD)
00105 // #include "ffd/micro.h"
00106 #elif ((defined(EZSP_HOST) || defined(UNIX_HOST)))
00107     #include "hal/micro/unix/host/micro.h"
00108 #else
00109     #error no platform or micro defined
00110 #endif
00111
00112 // the number of ticks (as returned from halCommonGetInt32uMillisecondTick)
00113 // that represent an actual second. This can vary on different platforms.
00114 // It must be defined by the host system.
00115 #ifndef MILLISECOND_TICKS_PER_SECOND
00116     #define MILLISECOND_TICKS_PER_SECOND 1024UL
00117 // See bug 10232
00118 // #error "MILLISECOND_TICKS_PER_SECOND is not defined in micro.h!"
00119 #endif
00120
00121 #ifndef MILLISECOND_TICKS_PER_DECISECOND
00122     #define MILLISECOND_TICKS_PER_DECISECOND (MILLISECOND_TICKS_PER_SECOND / 10)
00123 #endif
00124
00125 #ifndef MILLISECOND_TICKS_PER_QUARTERSECOND
00126     #define MILLISECOND_TICKS_PER_QUARTERSECOND (MILLISECOND_TICKS_PER_SECOND >> 2)
00127 #endif
00128
00129 #ifndef MILLISECOND_TICKS_PER_MINUTE
00130     #define MILLISECOND_TICKS_PER_MINUTE (60UL * MILLISECOND_TICKS_PER_SECOND)
00131 #endif
00132
00133 #ifndef MILLISECOND_TICKS_PER_HOUR
00134     #define MILLISECOND_TICKS_PER_HOUR (60UL * MILLISECOND_TICKS_PER_MINUTE)
00135 #endif
00136
00137 #ifndef MILLISECOND_TICKS_PER_DAY
00138     #define MILLISECOND_TICKS_PER_DAY (24UL * MILLISECOND_TICKS_PER_HOUR)
00139 #endif
00140
00141 #endif // DOXYGEN_SHOULD_SKIP_THIS
00142
00143 #endif // __MICRO_H__
00144

```

8.103 micro.h File Reference

```
#include "micro-common.h"
```

Functions

- void **halInternalSysReset** (uint16_t extendedCause)
- uint16_t **halGetExtendedResetInfo** (void)
- PGM_P **halGetExtendedResetString** (void)
- EmberStatus **halSetRadioHoldOff** (bool enable)

- bool `halGetRadioHoldOff` (void)
- void `halStackRadioPowerDownBoard` (void)
- void `halStackRadioPowerUpBoard` (void)
- void `halStackRadioPowerMainControl` (bool powerUp)
- void `halRadioPowerUpHandler` (void)
- void `halRadioPowerDownHandler` (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 35 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 SC3_VECTOR_INDEX
- #define SC4_VECTOR_INDEX
- #define USB_VECTOR_INDEX
- #define VECTOR_TABLE_LENGTH

8.103.1 Detailed Description

Utility and convenience functions for EM35x microcontroller. See [Common Microcontroller Functions](#) for documentation.

Definition in file `cortexm3/micro.h`.

8.104 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 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00023 #ifndef __EMBERSTATUS_TYPE__
00024 #define __EMBERSTATUS_TYPE__
00025 //This is necessary here because halSetRadioHoldOff returns an
00026 //EmberStatus and not adding this typedef to this file breaks a
00027 //whole lot of builds due to include mis-ordering
00028 typedef uint8_t EmberStatus;
00029 #endif //__EMBERSTATUS_TYPE__
00030 #endif // DOXYGEN_SHOULD_SKIP_THIS
00031
00032 // Micro specific serial defines
00033 #define EM_NUM_SERIAL_PORTS 4
00034 #define EMBER_SPI_MASTER 4
00035 #define EMBER_SPI_SLAVE 5
00036 #define EMBER_I2C 6
00037
00038 // Define the priority registers of system handlers and interrupts.
00039 // This example shows how to save the current ADC interrupt priority and
00040 // then set it to 24:
00041 //     uint8_t oldAdcPriority = INTERRUPT_PRIORITY_REGISTER(ADC);
00042 //     INTERRUPT_PRIORITY_REGISTER(ADC) = 24;
00043
00044 // For Cortex-M3 faults and exceptions
00045 #define HANDLER_PRIORITY_REGISTER(handler) \
00046     (*( (uint8_t *)SCS_SHPR_7to4_ADDR ) + handler##_VECTOR_INDEX - 4 )
00047
00048 // For EM3XX-specific interrupts
00049 #define INTERRUPT_PRIORITY_REGISTER(interrupt) \
00050     (*( (uint8_t *)NVIC_IPR_3to0_ADDR ) + interrupt##_VECTOR_INDEX - 16 )
00051
00052
00053 // The reset types of the EM300 series have both a base type and an
00054 // extended type. The extended type is a 16-bit value which has the base
00055 // type in the upper 8-bits, and the extended classification in the
00056 // lower 8-bits
00057 // For backwards compatibility with other platforms, only the base type is
00058 // returned by the halGetResetInfo() API. For the full extended type, the
00059 // halGetExtendedResetInfo() API should be called.
00060
00061 #define RESET_BASE_TYPE(extendedType)    ((uint8_t)((extendedType) >> 8) &
00062     0xFF)
00062 #define RESET_EXTENDED_FIELD(extendedType) ((uint8_t)(extendedType & 0xFF))
00063 #define RESET_VALID_SIGNATURE           (0xF00F)
00064 #define RESET_INVALID_SIGNATURE        (0xC33C)
00065
00066 // Define the base reset cause types
00067 #define RESET_BASE_DEF(basename, value, string) RESET_##basename = value,
00068 #define RESET_EXT_DEF(basename, extname, extvalue, string) /*nothing*/
00069 enum {
00070     #include "reset-def.h"
00071     NUM_RESET_BASE_TYPES
00072 };
00073 #undef RESET_BASE_DEF
00074 #undef RESET_EXT_DEF
00075

```

```

00076 // Define the extended reset cause types
00077 #define RESET_EXT_VALUE(basename, extvalue) \
00078   (((RESET_##basename)<<8) + extvalue)
00079 #define RESET_BASE_DEF(basename, value, string) /*nothing*/
00080 #define RESET_EXT_DEF(basename, extname, extvalue, string) \
00081   RESET_##basename##_##extname = RESET_EXT_VALUE(basename, extvalue),
00082 enum {
00083   #include "reset-def.h"
00084 };
00085 #undef RESET_EXT_VALUE
00086 #undef RESET_BASE_DEF
00087 #undef RESET_EXT_DEF
00088
00089 // These define the size of the GUARD region configured in the MPU that
00090 // sits between the heap and the stack
00091 #define HEAP_GUARD_REGION_SIZE      (SIZE_32B)
00092 #define HEAP_GUARD_REGION_SIZE_BYTES (1<<(HEAP_GUARD_REGION_SIZE+1))
00093
00094 // Define a value to fill the guard region between the heap and stack.
00095 #define HEAP_GUARD_FILL_VALUE (0xE2E2E2E2)
00096
00097 // Resize the CSTACK to add space to the 'heap' that exists below it
00098 uint32_t halStackModifyCStackSize(int32_t stackSizeDeltaWords);
00099
00100 // Initialize the CSTACK/Heap region and the guard page in between them
00101 void halInternalInitCStackRegion(void);
00102
00103 // Helper functions to get the location of the stack/heap
00104 uint32_t halInternalGetCStackBottom(void);
00105 uint32_t halInternalGetHeapTop(void);
00106 uint32_t halInternalGetHeapBottom(void);
00107
00108 #endif // DOXYGEN_SHOULD_SKIP_THIS
00109
00110 #define STACK_VECTOR_INDEX          0 // special case: stack pointer at reset
00111 #define RESET_VECTOR_INDEX          1
00112 #define NMI_VECTOR_INDEX            2
00113 #define HARD_FAULT_VECTOR_INDEX    3
00114 #define MEMORY_FAULT_VECTOR_INDEX  4
00115 #define BUS_FAULT_VECTOR_INDEX     5
00116 #define USAGE_FAULT_VECTOR_INDEX   6
00117 #define RESERVED07_VECTOR_INDEX   7
00118 #define RESERVED08_VECTOR_INDEX   8
00119 #define RESERVED09_VECTOR_INDEX   9
00120 #define RESERVED10_VECTOR_INDEX   10
00121 #define SVCALL_VECTOR_INDEX        11
00122 #define DEBUG_MONITOR_VECTOR_INDEX 12
00123 #define RESERVED13_VECTOR_INDEX   13
00124 #define PENDSV_VECTOR_INDEX        14
00125 #define SYSTICK_VECTOR_INDEX       15
00126 #define TIMER1_VECTOR_INDEX        16
00127 #define TIMER2_VECTOR_INDEX        17
00128 #define MANAGEMENT_VECTOR_INDEX  18
00129 #define BASEBAND_VECTOR_INDEX     19
00130 #define SLEEP_TIMER_VECTOR_INDEX  20
00131 #define SC1_VECTOR_INDEX           21
00132 #define SC2_VECTOR_INDEX           22
00133 #define SECURITY_VECTOR_INDEX    23
00134 #define MAC_TIMER_VECTOR_INDEX   24
00135 #define MAC_TX_VECTOR_INDEX      25
00136 #define MAC_RX_VECTOR_INDEX      26
00137 #define ADC_VECTOR_INDEX          27
00138 #define IRQA_VECTOR_INDEX         28
00139 #define IRQB_VECTOR_INDEX        29
00140 #define IRQC_VECTOR_INDEX        30
00141 #define IRQD_VECTOR_INDEX        31
00142 #define DEBUG_VECTOR_INDEX        32
00143 #define SC3_VECTOR_INDEX          33
00144 #define SC4_VECTOR_INDEX          34
00145 #define USB_VECTOR_INDEX         35
00146
00147 #define VECTOR_TABLE_LENGTH      36
00148
00149 void halInternalSysReset(uint16_t extendedCause);
00150
00151 uint16_t halGetExtendedResetInfo(void);
00152
00153 PGM_P halGetExtendedResetString(void);
00154
00155
00156
00157
00158
00159
00160
00161
00162
00163
00164
00165
00166
00167
00168
00169
00170
00171
00172
00173
00174
00175
00176
00177
00178
00179
00180
00181
00182
00183
00184
00185
00186
00187

```

```

00188 //[[ ram vectors are not public
00189 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00190
00208 uint32_t halRegisterRamVector(uint8_t vectorNumber, uint32_t newVector);
00209
00210
00223 uint32_t halUnRegisterRamVector(uint8_t vectorNumber);
00224
00225 #endif // DOXYGEN_SHOULD_SKIP_THIS
00226 //]]
00227
00228
00240 EmberStatus halSetRadioHoldOff(bool enable);
00241
00242
00247 bool halGetRadioHoldOff(void);
00248
00249
00260 void halStackRadioPowerDownBoard(void);
00261
00262
00273 void halStackRadioPowerUpBoard(void);
00274
00282 void halStackRadioPowerMainControl(bool powerUp);
00283
00287 void halRadioPowerUpHandler(void);
00288
00292 void halRadioPowerDownHandler(void);
00293
00294 #include "micro-common.h"
00295
00296 #endif //__EM3XX_MICRO_H__
00297

```

8.105 multi-network.h File Reference

Functions

- `uint8_t emberGetCurrentNetwork (void)`
- `EmberStatus emberSetCurrentNetwork (uint8_t index)`
- `uint8_t emberGetCallbackNetwork (void)`

8.105.1 Detailed Description

EmberZNet API for multi-network support. See [Multi-Network Manager](#) for documentation.

Definition in file [multi-network.h](#).

8.106 multi-network.h

```

00001
00018 uint8_t emberGetCurrentNetwork(void);
00019
00027 EmberStatus emberSetCurrentNetwork(uint8_t
index);
00028
00034 uint8_t emberGetCallbackNetwork(void);
00035

```

8.107 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 (uint8_t duration)`
- `EmberStatus emberJoinNetwork (EmberNodeType nodeType, EmberNetworkParameters *parameters)`
- `EmberStatus emberLeaveNetwork (void)`
- `EmberStatus emberSendZigbeeLeave (EmberNodeId destination, EmberLeaveRequestFlags flags)`
- `EmberStatus emberFindAndRejoinNetworkWithReason (bool haveCurrentNetworkKey, uint32_t channelMask, EmberRejoinReason reason)`
- `EmberStatus emberFindAndRejoinNetwork (bool haveCurrentNetworkKey, uint32_t channelMask)`
- `EmberStatus emberFindAndRejoinNetworkWithNodeType (bool haveCurrentNetworkKey, uint32_t channelMask, EmberNodeType nodeType)`
- `EmberRejoinReason emberGetLastRejoinReason (void)`
- `EmberStatus emberRejoinNetwork (bool haveCurrentNetworkKey)`
- `EmberStatus emberStartScan (EmberNetworkScanType scanType, uint32_t channelMask, uint8_t duration)`
- `EmberStatus emberStopScan (void)`
- `void emberScanCompleteHandler (uint8_t channel, EmberStatus status)`
- `void emberEnergyScanResultHandler (uint8_t channel, int8_t maxRssiValue)`
- `void emberNetworkFoundHandler (EmberZigbeeNetwork *networkFound)`
- `bool emberStackIsPerformingRejoin (void)`
- `EmberLeaveReason emberGetLastLeaveReason (EmberNodeId *returnNodeIdThatSentLeave)`
- `bool emberGetPermitJoining (void)`

8.107.1 Detailed Description

See [Network Formation](#) for documentation.

Definition in file [network-formation.h](#).

8.108 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 (uint8_t duration);
00094
00115 EmberStatus emberJoinNetwork (EmberNodeType

```

```

0016     nodeType,
0017     parameters);           EmberNetworkParameters *
0018
0019 EmberStatus emberLeaveNetwork(void);
0020
0021 EmberStatus emberSendZigbeeLeave(EmberNodeId
0022     destination,           EmberLeaveRequestFlags
0023     flags);
0024
0025
0026 EmberStatus emberFindAndRejoinNetworkWithReason
0027     (bool haveCurrentNetworkKey,
0028      uint32_t channelMask,
0029      EmberRejoinReason
0030      reason);
0031
0032 EmberStatus emberFindAndRejoinNetwork(bool
0033     haveCurrentNetworkKey,
0034      uint32_t channelMask);
0035
0036 EmberStatus emberFindAndRejoinNetworkWithNodeType
0037     (bool haveCurrentNetworkKey,
0038      uint32_t channelMask,
0039      EmberNodeType
0040      nodeType);
0041
0042 EmberRejoinReason emberGetLastRejoinReason
0043     (void);
0044
0045 #ifdef DOXYGEN_SHOULD_SKIP_THIS
0046 EmberStatus emberRejoinNetwork(bool
0047     haveCurrentNetworkKey);
0048 #else
0049 #define emberRejoinNetwork(haveKey) emberFindAndRejoinNetwork((haveKey), 0)
0050 #endif
0051
0052 EmberStatus emberStartScan(EmberNetworkScanType
0053     scanType,
0054      uint32_t channelMask,
0055      uint8_t duration);
0056
0057 EmberStatus emberStopScan(void);
0058
0059 void emberScanCompleteHandler( uint8_t channel,
0060     EmberStatus status );
0061
0062 void emberEnergyScanResultHandler(uint8_t channel,
0063     int8_t maxRssiValue);
0064
0065 void emberNetworkFoundHandler(EmberZigbeeNetwork
0066     *networkFound);
0067
0068 bool emberStackIsPerformingRejoin(void);
0069
0070
0071 EmberLeaveReason emberGetLastLeaveReason
0072     (EmberNodeId* returnNodeIdThatSentLeave);
0073
0074 bool emberGetPermitJoining(void);
0075
0076

```

8.109 network-manager.h File Reference

```
#include <CONFIGURATION_HEADER>
```

Macros

- #define NM_WARNING_LIMIT
- #define NM_WINDOW_SIZE

- #define NM_CHANNEL_MASK
- #define NM_WATCHLIST_SIZE

Functions

- void nmUtilWarningHandler (void)
- bool nmUtilProcessIncoming (EmberApsFrame *apsFrame, uint8_t messageLength, uint8_t *message)
- EmberStatus nmUtilChangeChannelRequest (void)

8.109.1 Detailed Description

Utilities for use by the ZigBee network manager. See [Network Manager](#) for documentation.

Definition in file [network-manager.h](#).

8.110 network-manager.h

```

00001
00090 #include CONFIGURATION_HEADER
00091
00092 // The application is notified via nmUtilWarningHandler
00093 // if NM_WARNING_LIMIT unsolicited scan reports are received
00094 // within NM_WINDOW_SIZE minutes. To save flash and RAM,
00095 // the actual timing is approximate.
00096 #ifndef NM_WARNING_LIMIT
00097     #define NM_WARNING_LIMIT 16
00098 #endif
00099
00100 #ifndef NM_WINDOW_SIZE
00101     #define NM_WINDOW_SIZE 4
00102 #endif
00103
00104 // The channels that should be used by the network manager.
00105
00106 #ifndef NM_CHANNEL_MASK
00107     #define NM_CHANNEL_MASK EMBER_ALL_802_15_4_CHANNELS_MASK
00108 #endif
00109
00110 // The number of channels used in the NM_CHANNEL_MASK.
00111
00112 #ifndef NM_WATCHLIST_SIZE
00113     #define NM_WATCHLIST_SIZE 16
00114 #endif
00115
00122 void nmUtilWarningHandler(void);
00123
00132 bool nmUtilProcessIncoming(EmberApsFrame *
    apsFrame,
                                uint8_t messageLength,
                                uint8_t* message);
00135
00139 EmberStatus nmUtilChangeChannelRequest(
    void);
00140

```

8.111 nvic-config.h File Reference

8.112 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'd 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 // \b NOTE Our chips only implement 5 bits for priority and subpriority
00047 // (bits [7:3]). This means that the lowest 3 bits in any priority field will
00048 // be forced to 0 and not used by the hardware
00049 //
00050 // We establish 8 levels of priority (3 bits), with 4 (2 bits) of tie break
00051 // subpriority. Lower priority values are higher priorities. This means that
00052 // 0 is the highest and 7 is the lowest. The NVIC mapping is detailed below.
00053 //
00054
00055 //The 'PRIGROUP' field is 3 bits within the AIRCR register and indicates the
00056 //bit position within a given exception's 8-bit priority field to the left of
00057 //which is the "binary point" separating the preemptive priority level (in the
00058 //most-significant bits) from the subpriority (in the least-significant bits).
00059 //The preemptive priority determines whether an interrupt can preempt an
00060 //executing interrupt. The subpriority helps choose which interrupt to run if
00061 //multiple interrupts with the same preemptive priority are active at the same
00062 //time. If no subpriority is given or there is a tie then the hardware defined
00063 //exception number is used to break the tie.
00064 //
00065 //The table below shows for each PRIGROUP value (the PRIGROUP value is the
00066 //number on the far left) how the priority bits are split and how this maps
00067 //to the priorities on chip. A 'P' is preemption bit, 'S' is a subpriority bit
00068 //and an 'X' indicates that that bit is not implemented.
00069 //0=7:1= PPPPPXXX.X
00070 //1=6:2= PPPPPX.XX
00071 //2=5:3= PPPPP.XXX
00072 //3=4:4= PPPP.SXXX
00073 //4=3:5= PPP.SXXXX
00074 //5=2:6= PP.SSSXXX
00075 //6=1:7= P.SSSSSXXX
00076 //7=0:8= SSSSSSSXXX
00077 //
00078 //Below is the default priority configuration for the chip's interrupts
00079 // Pri.Sub Purpose
00080 // 0.0 faults (highest)
00081 // 1.0 not used
00082 // 2.0 SysTick for idling, management interrupt for XTAL biasing,
00083 // and the SVCcall
00084 // 3.0 DISABLE_INTERRUPTS(), INTERRUPTS_OFF(), ATOMIC()
00085 // 4.0 normal interrupts
00086 // 5.0 not used
00087 // 6.0 not used
00088 // 7.0 debug backchannel, PendSV
00089 // 7.31 reserved (lowest)
00090 #if defined(FREE_RTOS)
00091 // FreeRTOS recommends this type of configuration for CortexM3 and asserts
00092 // in some places if it finds a different one.
00093 #define PRIGROUP_POSITION 2 // PPPPP.XXX
00094 #else
00095 #define PRIGROUP_POSITION 4 // PPP.SXXXX
00096 #endif
00097
00098 // Priority level used by DISABLE_INTERRUPTS() and INTERRUPTS_OFF()
00099 // Must be lower priority than pendsv
00100 // NOTE!! INTERRUPTS_DISABLED_PRIORITY AFFECTS SPRM.S79
00101 // NOTE!! FreeRTOSConfig.h configMAX_SYSCALL_INTERRUPT_PRIORITY must match
00102 // INTERRUPTS_DISABLED_PRIORITY!
00103 #define NVIC_ATOMIC 3
00104 #define INTERRUPTS_DISABLED_PRIORITY (NVIC_ATOMIC << (PRIGROUP_POSITION+1))
00105
00106
00107 //Exceptions with fixed priorities cannot be changed by software. Simply make
00108 //them 0 since they are high priorities anyways.

```

```

00109 #define NVIC_FIXED 0
00110 //Reserved exceptions are not instantiated in the hardware. Therefore
00111 //exception priorities don't exist so just default them to lowest level.
00112 #define NVIC_NONE 0xFF
00113
00114 #ifndef SEGMENT
00115     #define SEGMENT()
00116 #endif
00117 #ifndef SEGMENT2
00118     #define SEGMENT2()
00119 #endif
00120 #ifndef PERM_EXCEPTION
00121     #define PERM_EXCEPTION(vectorNumber, functionName, priority) \
00122         EXCEPTION(vectorNumber, functionName, priority, 0)
00123 #endif
00124
00125     // SEGMENT()
00126     // **Place holder required by isr-stubs.s79 to define __CODE__**
00127     // SEGMENT2()
00128     // **Place holder required by isr-stubs.s79 to define __THUMB__**
00129     // EXCEPTION(vectorNumber, functionName, priorityLevel, subpriority)
00130     // vectorNumber = exception number defined by hardware (not used
anywhere)
00131     // functionName = name of the function that the exception should trigger
00132     // priorityLevel = priority level of the function
00133     // supriority = Used to break ties between exceptions at the same level
00134     // PERM_EXCEPTION
00135     // is used to define an exception that should not be intercepted by the
00136     // interrupt debugging logic, or that not should have a weak stub
defined.
00137     // Otherwise the definition is the same as EXCEPTION
00138
00139 //INTENTIONALLY INDENTED AND SPACED APART! Keep it that way! See comment
above!
00140     SEGMENT()
00141     SEGMENT2()
00142     PERM_EXCEPTION( 1, halEntryPoint,      NVIC_FIXED ) //Reset
00143
00144     SEGMENT()
00145     SEGMENT2()
00146     EXCEPTION(      2, halNmiIsr,        NVIC_FIXED,       0)
00147
00148     SEGMENT()
00149     SEGMENT2()
00150     EXCEPTION(      3, halHardFaultIsr,  NVIC_FIXED,       0)
00151
00152     SEGMENT()
00153     SEGMENT2()
00154     EXCEPTION(      4, halMemoryFaultIsr, 0,           0)
00155
00156     SEGMENT()
00157     SEGMENT2()
00158     EXCEPTION(      5, halBusFaultIsr,   0,           0)
00159
00160     SEGMENT()
00161     SEGMENT2()
00162     EXCEPTION(      6, halUsageFaultIsr, 0,           0)
00163
00164     SEGMENT()
00165     SEGMENT2()
00166     EXCEPTION(      7, halReserved07Isr,  NVIC_NONE, NVIC_NONE)
00167
00168     SEGMENT()
00169     SEGMENT2()
00170     EXCEPTION(      8, halReserved08Isr,  NVIC_NONE, NVIC_NONE)
00171
00172     SEGMENT()
00173     SEGMENT2()
00174     EXCEPTION(      9, halReserved09Isr,  NVIC_NONE, NVIC_NONE)
00175
00176     SEGMENT()
00177     SEGMENT2()
00178     EXCEPTION(     10, halReserved10Isr,  NVIC_NONE, NVIC_NONE)
00179
00180     SEGMENT()
00181     SEGMENT2() // Above ATOMIC for FREE_RTOS task startup from ATOMIC level
00182     EXCEPTION(    11, halSvCallIsr,   NVIC_ATOMIC-1,      0)
00183
00184     SEGMENT()
00185     SEGMENT2()

```

```

00186    EXCEPTION( 12, halDebugMonitorIsr,      4,      0)
00187    SEGMENT()
00188    SEGMENT2()
00189    EXCEPTION( 13, halReserved13Isr, NVIC_NONE, NVIC_NONE)
00190
00191    SEGMENT()
00192    SEGMENT2() // Should be lowest priority
00193    EXCEPTION( 14, halPendSvIsr,      7,      0)
00194
00195    SEGMENT()
00196    SEGMENT2() // SysTick is *ABOVE* ATOMIC for Xtal startup
00197    EXCEPTION( 15, halInternalSysTickIsr,      2,      0)
00198
00199 //The following handlers map to "External Interrupts 16 and above"
00200 //In the NVIC Interrupt registers, this corresponds to bits 19:0 with bit
00201 //0 being TIMER1 (exception 16), bit 19 being USB (exception 35), etc.
00202    SEGMENT()
00203    SEGMENT2()
00204    EXCEPTION( 16, halTimer1Isr,      4,      0)
00205
00206    SEGMENT()
00207    SEGMENT2()
00208    EXCEPTION( 17, halTimer2Isr,      4,      0)
00209
00210    SEGMENT()
00211    SEGMENT2() // Mgmt is *ABOVE* ATOMIC for DMA exceptions & Xtal startup
00212    EXCEPTION( 18, halManagementIsr,      2,      0)
00213
00214    SEGMENT()
00215    SEGMENT2()
00216    EXCEPTION( 19, halBaseBandIsr,      4,      0)
00217
00218    SEGMENT()
00219    SEGMENT2()
00220    EXCEPTION( 20, halSleepTimerIsr,      4,      0)
00221
00222    SEGMENT()
00223    SEGMENT2()
00224    EXCEPTION( 21, halSc1Isr,      4,      0)
00225
00226    SEGMENT()
00227    SEGMENT2()
00228    EXCEPTION( 22, halSc2Isr,      4,      0)
00229
00230    SEGMENT()
00231    SEGMENT2()
00232    EXCEPTION( 23, halSecurityIsr,      4,      0)
00233
00234 //MAC Timer Handler must be higher priority than emRadioTransmitIsr
00235 // for idling during managed TX->RX turnaround to function correctly.
00236 // But it is >= 3 so it doesn't run when ATOMIC.
00237    SEGMENT()
00238    SEGMENT2()
00239    EXCEPTION( 24, halStackMacTimerIsr,      3,      0)
00240
00241    SEGMENT()
00242    SEGMENT2()
00243    EXCEPTION( 25, emRadioTransmitIsr,      4,      0)
00244
00245    SEGMENT()
00246    SEGMENT2()
00247    EXCEPTION( 26, emRadioReceiveIsr,      4,      0)
00248
00249    SEGMENT()
00250    SEGMENT2()
00251    EXCEPTION( 27, halAdcIsr,      4,      0)
00252
00253    SEGMENT()
00254    SEGMENT2()
00255    EXCEPTION( 28, halIrqAIsr,      4,      0)
00256
00257    SEGMENT()
00258    SEGMENT2()
00259    EXCEPTION( 29, halIrqBIsr,      4,      0)
00260
00261    SEGMENT()
00262    SEGMENT2()
00263    EXCEPTION( 30, halIrqCIsr,      4,      0)
00264
00265

```

```

00266 SEGMENT()
00267 SEGMENT2()
00268 EXCEPTION( 31, halIrqDISr,           4,      0)
00269 SEGMENT()
00270 SEGMENT2() // Virtual UART input lowest priority
00271 EXCEPTION( 32, halDebugISR,           7,      0)
00272 SEGMENT()
00273 SEGMENT2()
00274 EXCEPTION( 33, halSc3ISR,            4,      0)
00275 SEGMENT()
00276 SEGMENT2()
00277 EXCEPTION( 34, halSc4ISR,            4,      0)
00278 SEGMENT()
00279 SEGMENT2()
00280 EXCEPTION( 35, halUsbISR,             4,      0)
00281 SEGMENT()
00282 SEGMENT2()
00283 EXCEPTION( 35, halUsbISR,             4,      0)
00284 #undef SEGMENT
00285 #undef SEGMENT2
00286 #undef FERM_EXCEPTION
00287 #endif //DOXYGEN_SHOULD_SKIP_THIS
00288
00289
00290

```

8.113 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 uint8_t * emberMessageBufferContents (EmberMessageBuffer buffer)`
- `void emberSetMessageBufferLength (EmberMessageBuffer buffer, uint8_t newLength)`
- `void emberHoldMessageBuffer (EmberMessageBuffer buffer)`
- `void emberReleaseMessageBuffer (EmberMessageBuffer buffer)`
- `uint8_t emberPacketBufferFreeCount (void)`

Buffer Functions

- `#define emberStackBufferLink(buffer)`
- `#define emberSetStackBufferLink(buffer, newLink)`
- `#define emberAllocateStackBuffer()`
- `EmberMessageBuffer emberAllocateLinkedBuffers (uint8_t count)`
- `EmberMessageBuffer emberFillStackBuffer (unsigned int 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 (uint8_t *contents, uint8_t length)`
- `void emberCopyToLinkedBuffers (uint8_t *contents, EmberMessageBuffer buffer, uint8_t startIndex, uint8_t length)`
- `void emberCopyFromLinkedBuffers (EmberMessageBuffer buffer, uint8_t startIndex, uint8_t *contents, uint8_t length)`
- `void emberCopyBufferBytes (EmberMessageBuffer to, uint16_t toIndex, EmberMessageBuffer from, uint16_t fromIndex, uint16_t count)`
- `EmberStatus emberAppendToLinkedBuffers (EmberMessageBuffer buffer, uint8_t *contents, uint8_t length)`
- `EmberStatus emberAppendPgmToLinkedBuffers (EmberMessageBuffer buffer, PG_M_P contents, uint8_t length)`
- `EmberStatus emberAppendPgmStringToLinkedBuffers (EmberMessageBuffer buffer, PGM_P suffix)`
- `EmberStatus emberSetLinkedBuffersLength (EmberMessageBuffer buffer, uint8_t length)`
- `uint8_t * emberGetLinkedBuffersPointer (EmberMessageBuffer buffer, uint8_t index)`
- `XAP2B_PAGEZERO_ON uint8_t emberGetLinkedBuffersByte (EmberMessageBuffer buffer, uint8_t index)`
- `XAP2B_PAGEZERO_OFF void emberSetLinkedBuffersByte (EmberMessageBuffer buffer, uint8_t index, uint8_t byte)`
- `uint16_t emberGetLinkedBuffersLowHighInt16u (EmberMessageBuffer buffer, uint8_t index)`
- `void emberSetLinkedBuffersLowHighInt16u (EmberMessageBuffer buffer, uint8_t index, uint16_t value)`
- `uint32_t emberGetLinkedBuffersLowHighInt32u (EmberMessageBuffer buffer, uint8_t index)`
- `void emberSetLinkedBuffersLowHighInt32u (EmberMessageBuffer buffer, uint8_t index, uint32_t value)`
- `EmberMessageBuffer emberCopyLinkedBuffers (EmberMessageBuffer buffer)`
- `EmberMessageBuffer emberMakeUnsharedLinkedBuffer (EmberMessageBuffer buffer, bool isShared)`

8.113.1 Detailed Description

Packet buffer allocation and management routines See [Packet Buffers](#) for documentation.

Definition in file [packet-buffer.h](#).

8.114 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 uint8_t *emberMessageBufferContents(
00059     EmberMessageBuffer buffer);
00060 XAP2B_PAGEZERO_OFF
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
00086     buffer, uint8_t newLength);
00087
00088 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00089 #ifdef EM_DEBUG_BUFFER_USE
00090 #define EM_BUFFER_DEBUG(X) X
00091 #define EM_BUFFER_FILE_LOC , __FILE__, __LINE__
00092 #define EM_BUFFER_FILE_DECL , char *file, int 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
00118     buffer);
00119 #else
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
00134     EM_BUFFER_FILE_DECL);
00135
00141 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00142 void emberReleaseMessageBuffer(EmberMessageBuffer
00143     buffer);
00144 #else
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
00159     EM_BUFFER_FILE_DECL);

```

```

00159 XAP2B_PAGEZERO_OFF
00160 #endif //DOXYGEN_SHOULD_SKIP_THIS
00161
00162 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00163 extern uint8_t emPacketBufferCount;
00164 extern uint8_t *emPacketBufferData;
00165 extern uint8_t *emMessageBufferLengths;
00166 extern uint8_t *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
00216     (uint8_t count);
00216
00228 EmberMessageBuffer emberFillStackBuffer(
00229     unsigned int count, ...);
00230
00237
00251 EmberMessageBuffer emberFillLinkedBuffers
00252     (uint8_t *contents, uint8_t length);
00252
00266 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00267 void
00268 emberCopyToLinkedBuffers(uint8_t *contents,
00269                             EmberMessageBuffer buffer,
00270                             uint8_t startIndex,
00271                             uint8_t length);
00272 #else
00273 #define emberCopyToLinkedBuffers(contents, buffer, startIndex, length) \
00274 emReallyCopyToLinkedBuffers((PGM_P) (contents), (buffer), (startIndex), \
00275     (length), 1)
00275 XAP2B_PAGEZERO_ON
00276 void
00277 emReallyCopyToLinkedBuffers(PGM_P contents,
00278                             EmberMessageBuffer buffer,
00279                             uint8_t startIndex,
00280                             uint8_t length,
00281                             uint8_t 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
00302     buffer,
00303             uint8_t startIndex,
00304             uint8_t *contents,
00305             uint8_t length);
00305 #else
00306 #define emberCopyFromLinkedBuffers(buffer, startIndex, contents, length) \
00307 emReallyCopyToLinkedBuffers((PGM_P) (contents), (buffer), (startIndex), \
00308     (length), 0)
00308 #endif
00309
00323 void emberCopyBufferBytes(EmberMessageBuffer
00324     to,
00325             uint16_t toIndex,
00326             EmberMessageBuffer from,
00327             uint16_t fromIndex,
00328             uint16_t count);
00328
00343 EmberStatus emberAppendToLinkedBuffers(
00344     EmberMessageBuffer buffer,
00345             uint8_t *contents,
00346             uint8_t length);
00360 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00361 // To save flash this shares an implementation with
00362 // emberAppendToLinkedBuffers().

```

```

00363 EmberStatus emberAppendPgmToLinkedBuffers
00364     (EmberMessageBuffer buffer,
00365                     PGM_P contents,
00366             uint8_t length);
00367 #define emberAppendPgmToLinkedBuffers(buffer, contents, length) \
00368     (emAppendToLinkedBuffers((buffer), (PGM_P) (contents), (length), 2))
00369
00370 EmberStatus emAppendToLinkedBuffers(EmberMessageBuffer
00371     buffer,
00372                     PGM_P contents,
00373             uint8_t length,
00374             uint8_t direction);
00375 #endif
00376
00377 EmberStatus emberAppendPgmStringToLinkedBuffers
00378     (EmberMessageBuffer buffer, PGM_P suffix);
00379
00380 EmberStatus emberSetLinkedBuffersLength(
00381     EmberMessageBuffer buffer, uint8_t length);
00382
00383 uint8_t *emberGetLinkedBuffersPointer(
00384     EmberMessageBuffer buffer, uint8_t index);
00385
00386 XAP2B_PAGEZERO_ON
00387 uint8_t emberGetLinkedBuffersByte(EmberMessageBuffer
00388     buffer, uint8_t index);
00389 XAP2B_PAGEZERO_OFF
00390
00391 void emberSetLinkedBuffersByte(EmberMessageBuffer
00392     buffer, uint8_t index, uint8_t byte);
00393
00394 uint16_t emberGetLinkedBuffersLowHighInt16u(
00395     EmberMessageBuffer buffer,
00396                     uint8_t index);
00397
00398 void emberSetLinkedBuffersLowHighInt16u(
00399     EmberMessageBuffer buffer,
00400                     uint8_t index,
00401             uint16_t value);
00402
00403 uint32_t emberGetLinkedBuffersLowHighInt32u(
00404     EmberMessageBuffer buffer,
00405                     uint8_t index);
00406
00407 void emberSetLinkedBuffersLowHighInt32u(
00408     EmberMessageBuffer buffer,
00409                     uint8_t index,
00410             uint32_t value);
00411
00412 EmberMessageBuffer emberCopyLinkedBuffers
00413     (EmberMessageBuffer buffer);
00414
00415 EmberMessageBuffer emberMakeUnsharedLinkedBuffer
00416     (EmberMessageBuffer buffer, bool isShared);
00417
00418 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00419 extern uint8_t emPacketBufferFreeCount;
00420 #define emberPacketBufferFreeCount() (emPacketBufferFreeCount)
00421 #else
00422 uint8_t emberPacketBufferFreeCount(void);
00423 #endif
00424
00425 #endif // __PACKET_BUFFER_H__

```

8.115 platform-common.h File Reference

Macros

- #define **MEMSET**(d, v, l)
- #define **MEMCOPY**(d, s, l)
- #define **MEMMOVE**(d, s, l)

- #define **MEMPGMCOPY**(d, s, l)
- #define **MEMCOMPARE**(s0, s1, l)
- #define **MEMPGMCOMPARE**(s0, s1, l)

Generic Types

- #define **TRUE**
- #define **FALSE**
- #define **NULL**

Bit Manipulation Macros

- #define **BIT**(x)
- #define **BIT32**(x)
- #define **SETBIT**(reg, bit)
- #define **SETBITS**(reg, bits)
- #define **CLEARBIT**(reg, bit)
- #define **CLEARBITS**(reg, bits)
- #define **READBIT**(reg, bit)
- #define **READBITS**(reg, bits)

Byte Manipulation Macros

- #define **LOW_BYTE**(n)
- #define **HIGH_BYTE**(n)
- #define **HIGH_LOW_TO_INT**(high, low)
- #define **BYTE_0**(n)
- #define **BYTE_1**(n)
- #define **BYTE_2**(n)
- #define **BYTE_3**(n)
- #define **COUNTOF**(a)

Time Manipulation Macros

- #define **elapsedTimeInt8u**(oldTime, newTime)
- #define **elapsedTimeInt16u**(oldTime, newTime)
- #define **elapsedTimeInt32u**(oldTime, newTime)
- #define **MAX_INT8U_VALUE**
- #define **HALF_MAX_INT8U_VALUE**
- #define **timeGTorEqualInt8u**(t1, t2)
- #define **MAX_INT16U_VALUE**
- #define **HALF_MAX_INT16U_VALUE**
- #define **timeGTorEqualInt16u**(t1, t2)
- #define **MAX_INT32U_VALUE**
- #define **HALF_MAX_INT32U_VALUE**
- #define **timeGTorEqualInt32u**(t1, t2)

Miscellaneous Macros

- `#define UNUSED_VAR(x)`
- `#define DEBUG_LEVEL`

8.115.1 Detailed Description

See [Common PLATFORM_HEADER Configuration](#) for detailed documentation.

Definition in file [platform-common.h](#).

8.116 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
00029
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
00045 #ifdef _HAL_USE_COMMON_PGM_
00046
00047     #define PGM      const
00048
00049     #define PGM_P    const char *
00050
00051     #define PGM_PU   const unsigned char *
00052
00053
00054     #define PGM_NO_CONST
00055
00056
00057 #endif //HAL_USE_COMMON_PGM_
00058
00059
00060
00061
00062
00063
00064
00065
00066
00067
00068
00069
00070
00071
00072
00073
00074
00075
00076
00077
00078 #ifdef _HAL_USE_COMMON_DIVMOD_
00079
00080     #define halCommonUDiv32By16(x, y) (((uint16_t) ((uint32_t) (x)) / ((uint16_t)
00081     (y))))
00082
00083     #define halCommonSDiv32By16(x, y) (((int16_t) (((int32_t) (x)) / ((int16_t)
00084     (y))))
00085
00086     #define halCommonUMod32By16(x, y) (((uint16_t) ((uint32_t) (x)) % ((uint16_t)
00087     (y))))
00088
00089     #define halCommonSMod32By16(x, y) (((int16_t) (((int32_t) (x)) % ((int16_t)
00090     (y))))
00091
00092 #endif //HAL_USE_COMMON_DIVMOD_
00093
00094
00095
00096
00097
00098
00099
00100
00101
00102
00103
00104
00105
00106
00107
00108
00109
00110
00111
00112
00113
00114
00115 #endif //HAL_USE_COMMON_DIVMOD_
00116
00117
00118
00119 #ifdef _HAL_USE_COMMON_MEMUTILS_

```

```

00120
00132
00136 void halCommonMemMove(void *dest, const void *src, uint16_t bytes);
00137
00138
00142 void halCommonMemSet(void *dest, uint8_t val, uint16_t bytes);
00143
00144
00148 int16_t halCommonMemCompare(const void *source0, const void *source1,
00149 uint16_t bytes);
00150
00155 int8_t halCommonMemPGMCompare(const void *source0, const void PGM_NO_CONST *
00156 source1, uint16_t bytes);
00157
00161 void halCommonMemPGMCopy(void* dest, const void PGM_NO_CONST *source,
00162 uint16_t bytes);
00163
00166 #define MEMSET(d,v,l) halCommonMemSet(d,v,l)
00167 #define MEMCOPY(d,s,l) halCommonMemMove(d,s,l)
00168 #define MEMMOVE(d,s,l) halCommonMemMove(d,s,l)
00169 #define MEMPGMCOPY(d,s,l) halCommonMemPGMCopy(d,s,l)
00170 #define MEMCOMPARE(s0,s1,l) halCommonMemCompare(s0, s1, l)
00171 #define MEMPGMCOMPARE(s0,s1,l) halCommonMemPGMCompare(s0, s1, l)
00172
00174 #else
00175
00184 #define MEMSET(d,v,l) memset((void*)(d),v,l)
00185 #define MEMCOPY(d,s,l) memcpy((void*)(d),(void*)(s),l)
00186 #define MEMMOVE(d,s,l) memmove((void*)(d),(void*)(s),l)
00187 #define MEMPGMCOPY(d,s,l) memcpy((void*)(d),(void*)(s),l)
00188 #define MEMCOMPARE(s0,s1,l) memcmp(s0, s1, l)
00189 #define MEMPGMCOMPARE(s0,s1,l) memcmp(s0, s1, l)
00190 #endif /* _HAL_USE_COMMON_MEMUTILS_ */
00191
00192
00193
00194
00195
00196
00197
00198
00199
00201 // The following sections are common on all platforms
00203
00205
00213 #define TRUE 1
00214
00218 #define FALSE 0
00219
00220 #ifndef NULL
00221
00224 #define NULL ((void *)0)
00225 #endif
00226
00228
00229
00234
00238 #define BIT(x) (1U << (x)) // Unsigned avoids compiler warnings re BIT(15)
00239
00243 #define BIT32(x) (((uint32_t) 1) << (x))
00244
00250 #define SETBIT(reg, bit) reg |= BIT(bit)
00251
00257 #define SETBITS(reg, bits) reg |= (bits)
00258
00264 #define CLEARBIT(reg, bit) reg &= ~(BIT(bit))
00265
00271 #define CLEARBITS(reg, bits) reg &= ~(bits)
00272
00276 #define READBIT(reg, bit) (reg & (BIT(bit)))
00277
00282 #define READBITS(reg, bits) (reg & (bits))
00283
00285
00286
00288
00292
00296 #define LOW_BYTE(n) ((uint8_t)((n) & 0xFF))
00297

```

```

00301 #define HIGH_BYTE(n)           ((uint8_t)(LOW_BYTE((n) >> 8)))
00302
00307 #define HIGH_LOW_TO_INT(high, low) \
00308     ((uint16_t)((uint16_t)(high) << 8)) + \
00309     ((uint16_t)((low) & 0xFF))
00310
00311
00315 #define BYTE_0(n)           ((uint8_t)((n) & 0xFF))
00316
00320 #define BYTE_1(n)           ((uint8_t)(BYTE_0((n) >> 8)))
00321
00325 #define BYTE_2(n)           ((uint8_t)(BYTE_0((n) >> 16)))
00326
00330 #define BYTE_3(n)           ((uint8_t)(BYTE_0((n) >> 24)))
00331
00335 #define COUNTOF(a) (sizeof(a)/sizeof(a[0]))
00336
00338
00339
00341
00345
00350 #define elapsedTimeInt8u(oldTime, newTime) \
00351     ((uint8_t)((newTime) - (uint8_t)(oldTime)))
00352
00357 #define elapsedTimeInt16u(oldTime, newTime) \
00358     ((uint16_t)((uint16_t)(newTime) - (uint16_t)(oldTime)))
00359
00364 #define elapsedTimeInt32u(oldTime, newTime) \
00365     ((uint32_t)((uint32_t)(newTime) - (uint32_t)(oldTime)))
00366
00371 #define MAX_INT8U_VALUE      (0xFF)
00372 #define HALF_MAX_INT8U_VALUE (0x80)
00373 #define timeGTorEqualInt8u(t1, t2) \
00374     (elapsedTimeInt8u(t2, t1) <= (HALF_MAX_INT8U_VALUE))
00375
00380 #define MAX_INT16U_VALUE      (0xFFFF)
00381 #define HALF_MAX_INT16U_VALUE (0x8000)
00382 #define timeGTorEqualInt16u(t1, t2) \
00383     (elapsedTimeInt16u(t2, t1) <= (HALF_MAX_INT16U_VALUE))
00384
00389 #define MAX_INT32U_VALUE      (0xFFFFFFFFL)
00390 #define HALF_MAX_INT32U_VALUE (0x80000000L)
00391 #define timeGTorEqualInt32u(t1, t2) \
00392     (elapsedTimeInt32u(t2, t1) <= (HALF_MAX_INT32U_VALUE))
00393
00395
00396
00398
00402
00403 #ifndef UNUSED_VAR
00404
00408 #define UNUSED_VAR(x) (void)(x)
00409 #endif
00410
00414 #ifndef DEBUG_LEVEL
00415     #if defined(DEBUG) && defined(DEBUG_STRIPPED)
00416         #error "DEBUG and DEBUG_OFF cannot be both be defined!"
00417     #elif defined(DEBUG)
00418         #define DEBUG_LEVEL FULL_DEBUG
00419     #elif defined(DEBUG_STRIPPED)
00420         #define DEBUG_LEVEL NO_DEBUG
00421     #else
00422         #define DEBUG_LEVEL BASIC_DEBUG
00423     #endif
00424 #endif
00425
00427
00428 #endif //__PLATFORMCOMMON_H__
00429

```

8.117 random.h File Reference

Functions

- void [halStackSeedRandom](#) (uint32_t seed)
- uint16_t [halCommonGetRandom](#) (void)

8.117.1 Detailed Description

See [Random Number Generation](#) for detailed documentation.

Definition in file [random.h](#).

8.118 random.h

```

00001
00016 #ifndef __RANDOM_H__
00017 #define __RANDOM_H__
00018
00026 void halStackSeedRandom(uint32_t seed);
00027
00036 #if defined( EMBER_TEST )
00037 #define halCommonGetRandom halCommonGetRandomTraced(__FILE__, __LINE__)
00038 uint16_t halCommonGetRandomTraced(char *file, int line);
00039 #else
00040 uint16_t halCommonGetRandom(void);
00041 #endif
00042
00046 #endif //__RANDOM_H__
00047
00048

```

8.119 ref0657.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, BOARDLED0, BOA-
 RDLED1,
 BOARDLED2, BOARDLED3, BOARD_ACTIVITY_LED, BOARD_HEARTBE-
 AT_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_DFL_RHO_FOR_RHO
- #define PWRUP_OUT_DFL_RHO_FOR_RHO
- #define PWRDN_CFG_DFL_RHO_FOR_RHO
- #define PWRDN_OUT_DFL_RHO_FOR_RHO
- #define PWRUP_CFG_DFL_RHO_FOR_DFL
- #define PWRUP_OUT_DFL_RHO_FOR_DFL
- #define PWRDN_CFG_DFL_RHO_FOR_DFL
- #define PWRDN_OUT_DFL_RHO_FOR_DFL
- #define PWRUP_CFG_DFL_RHO
- #define PWRUP_OUT_DFL_RHO
- #define PWRDN_CFG_DFL_RHO
- #define PWRDN_OUT_DFL_RHO
- #define halInternalInitRadioHoldOff()
- #define ADJUST_GPIO_CONFIG_DFL_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 `hal-Init()` 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`

ENABLE_ALT_FUNCTION_TX_ACTIVE

This define does not equate to anything. It is used as a trigger to enable the REG_EN alternate function on PA7. Default is to not enable REG_EN functionality on PA7.

When ENABLE_ALT_FUNCTION_TX_ACTIVE is defined, [halInit\(\)](#) and [halPowerUp\(\)](#) will enable the TX_ACTIVE alternate functionality of PC5. [halPowerDown\(\)](#) will configure PC5 to be a low output. TX_ACTIVE can be used for external PA power management and RF switching logic. In transmit mode the Tx baseband drives TX_ACTIVE high. In receive mode the TX_ACTIVE signal is low. This define will override any settings for PC5.

- `#define ENABLE_ALT_FUNCTION_TX_ACTIVE`

EEPROM_USES_SHUTDOWN_CONTROL

This define does not equate to anything. It is used as a trigger to enable the nTX_ACTIVE alternate function on PC6. Default is to not enable nTX_ACTIVE functionality on PC6.

When EEPROM_USES_SHUTDOWN_CONTROL is defined, logic is enabled in the EEPROM driver which drives PB7 high upon EEPROM initialization. In Ember reference designs, PB7 acts as an EEPROM enable pin and therefore must be driven high in order to use the EEPROM. This option is intended to be enabled when running app-bootloader on designs based on current Ember reference designs.

- #define EEPROM_USES_SHUTDOWN_CONTROL

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 deepsleeping. 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 FEM_CTX_BIT
- #define FEM_CRX_BIT
- #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()
- uint16_t gpioCfgPowerUp [6]
- uint16_t gpioCfgPowerDown [6]
- uint8_t gpioOutPowerUp [3]
- uint8_t gpioOutPowerDown [3]
- GpioMaskType 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()`

8.119.1 Detailed Description

See [Sample Breakout Board Configuration](#) for detailed documentation.

Definition in file [ref0657.h](#).

8.119.2 Macro Definition Documentation

8.119.2.1 #define `halInternalInitBoard()`

Initialize the board. This function is called from `halInit()`.

Definition at line [968](#) of file [ref0657.h](#).

8.119.2.2 #define `halInternalPowerDownBoard()`

Power down the board. This function is called from `halPowerDown()`.

Definition at line [988](#) of file [ref0657.h](#).

8.119.2.3 #define `halInternalPowerUpBoard()`

Power up the board. This function is called from `halPowerUp()`.

Definition at line [1000](#) of file [ref0657.h](#).

8.120 ref0657.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
00213
00226
00233 #define RHO_GPIO PORTA_PIN(6)
00234
00237 #define RHO_ASSERTED 1
00238
00241 #define RHO_CFG GPIO_PACFGH
00242
00245 #define RHO_IN GPIO_PAINT
00246
00249 #define RHO_OUT GPIO_PAOUT
00250
00254 #define RHO_SEL() do { GPIO IRQDSEL = RHO_GPIO; } while(0)
00255
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_DFL_RHO_FOR_RHO GPIOCFG_IN_PUD

```

```

00280 #define PWRUP_OUT_DFL_RHO_FOR_RHO    GPIOOUT_PULLDOWN /* Deassert */
00281 #define PWRDN_CFG_DFL_RHO_FOR_RHO    GPIOCFG_IN_PUD
00282 #define PWRDN_OUT_DFL_RHO_FOR_RHO   GPIOOUT_PULLDOWN /* Deassert */
00283
00284 #define PWRUP_CFG_DFL_RHO_FOR_DFL    GPIOCFG_OUT
00285 #define PWRUP_OUT_DFL_RHO_FOR_DFL    1 /* LED default off */
00286 #define PWRDN_CFG_DFL_RHO_FOR_DFL    GPIOCFG_OUT
00287 #define PWRDN_OUT_DFL_RHO_FOR_DFL    1 /* LED off */
00288
00289
00290
00291 #if      (defined(RADIO_HOLDOFF) && defined(RHO_GPIO))
00292 // Initial bootup configuration is for Radio HoldOff
00293 #define PWRUP_CFG_DFL_RHO          PWRUP_CFG_DFL_RHO_FOR_RHO
00294 #define PWRUP_OUT_DFL_RHO          PWRUP_OUT_DFL_RHO_FOR_RHO
00295 #define PWRDN_CFG_DFL_RHO          PWRDN_CFG_DFL_RHO_FOR_RHO
00296 #define PWRDN_OUT_DFL_RHO          PWRDN_OUT_DFL_RHO_FOR_RHO
00297
00298 #define halInternalInitRadioHoldOff() halSetRadioHoldOff(true)
00299
00300
00301 #else
00302 // Initial bootup configuration is for default
00303 #define PWRUP_CFG_DFL_RHO          PWRUP_CFG_DFL_RHO_FOR_DFL
00304 #define PWRUP_OUT_DFL_RHO          PWRUP_OUT_DFL_RHO_FOR_DFL
00305 #define PWRDN_CFG_DFL_RHO          PWRDN_CFG_DFL_RHO_FOR_DFL
00306 #define PWRDN_OUT_DFL_RHO          PWRDN_OUT_DFL_RHO_FOR_DFL
00307 #define halInternalInitRadioHoldOff() /* no-op */
00308#endif//((defined(RADIO_HOLDOFF) && defined(RHO_GPIO)))
00309
00310 #ifdef RHO_GPIO
00311
00312 #define ADJUST_GPIO_CONFIG_DFL_RHO(enableRadioHoldOff) do {
00313     ATOMIC( /* Must read-modify-write so to be safe, use ATOMIC() */ \
00314     if (enableRadioHoldOff) { /* Radio HoldOff */ \
00315         /* Actual register state */ \
00316         /*halGpioSetConfig(RHO_CFG, PWRUP_CFG_DFL_RHO_FOR_RHO);*/ \
00317         RHO_CFG = RHO_CFG \
00318             & ~ (0x000F           << ((RHO_GPIO&3)*4)) \
00319             | (PWRUP_CFG_DFL_RHO_FOR_RHO << ((RHO_GPIO&3)*4)); \
00320         RHO_OUT = RHO_OUT \
00321             & ~ (0x0001           << ((RHO_GPIO&7) )) \
00322             | (PWRUP_OUT_DFL_RHO_FOR_RHO << ((RHO_GPIO&7) )); \
00323         /* Shadow register state */ \
00324         gpioCfgPowerUp[RHO_GPIO>>2] = gpioCfgPowerUp[RHO_GPIO>>2] \
00325             & ~ (0x000F           << ((RHO_GPIO&3)*4)) \
00326             | (PWRUP_CFG_DFL_RHO_FOR_RHO << ((RHO_GPIO&3)*4)); \
00327         gpioOutPowerUp[RHO_GPIO>>3] = gpioOutPowerUp[RHO_GPIO>>3] \
00328             & ~ (0x0001           << ((RHO_GPIO&7) )) \
00329             | (PWRUP_OUT_DFL_RHO_FOR_RHO << ((RHO_GPIO&7) )); \
00330         gpioCfgPowerDown[RHO_GPIO>>2] = gpioCfgPowerDown[RHO_GPIO>>2] \
00331             & ~ (0x000F           << ((RHO_GPIO&3)*4)) \
00332             | (PWRDN_CFG_DFL_RHO_FOR_RHO << ((RHO_GPIO&3)*4)); \
00333         gpioOutPowerDown[RHO_GPIO>>3] = gpioOutPowerDown[RHO_GPIO>>3] \
00334             & ~ (0x0001           << ((RHO_GPIO&7) )) \
00335             | (PWRDN_OUT_DFL_RHO_FOR_RHO << ((RHO_GPIO&7) )); \
00336         RHO_INTCFG = (0 << GPIO_INTFILT_BIT) /* 0 = no filter */ \
00337             | (3 << GPIO_INTMOD_BIT); /* 3 = both edges */ \
00338     } else { /* default */ \
00339         /* Actual register state */ \
00340         /*halGpioSetConfig(RHO_CFG, PWRUP_CFG_DFL_RHO_FOR_DFL);*/ \
00341         RHO_CFG = RHO_CFG \
00342             & ~ (0x000F           << ((RHO_GPIO&3)*4)) \
00343             | (PWRUP_CFG_DFL_RHO_FOR_DFL << ((RHO_GPIO&3)*4)); \
00344         RHO_OUT = RHO_OUT \
00345             & ~ (0x0001           << ((RHO_GPIO&7) )) \
00346             | (PWRUP_OUT_DFL_RHO_FOR_DFL << ((RHO_GPIO&7) )); \
00347         /* Shadow register state */ \
00348         gpioCfgPowerUp[RHO_GPIO>>2] = gpioCfgPowerUp[RHO_GPIO>>2] \
00349             & ~ (0x000F           << ((RHO_GPIO&3)*4)) \
00350             | (PWRUP_CFG_DFL_RHO_FOR_DFL << ((RHO_GPIO&3)*4)); \
00351         gpioOutPowerUp[RHO_GPIO>>3] = gpioOutPowerUp[RHO_GPIO>>3] \
00352             & ~ (0x0001           << ((RHO_GPIO&7) )) \
00353             | (PWRUP_OUT_DFL_RHO_FOR_DFL << ((RHO_GPIO&7) )); \
00354         gpioCfgPowerDown[RHO_GPIO>>2] = gpioCfgPowerDown[RHO_GPIO>>2] \
00355             & ~ (0x000F           << ((RHO_GPIO&3)*4)) \
00356             | (PWRDN_CFG_DFL_RHO_FOR_DFL << ((RHO_GPIO&3)*4)); \
00357         gpioOutPowerDown[RHO_GPIO>>3] = gpioOutPowerDown[RHO_GPIO>>3] \
00358             & ~ (0x0001           << ((RHO_GPIO&7) )) \
00359             | (PWRDN_OUT_DFL_RHO_FOR_DFL << ((RHO_GPIO&7) )); \
00360         RHO_INTCFG = 0; /* disabled */ \
00361     } \
00362     RHO_SEL(); /* Point IRQ at the desired pin */ \
00363 } while (0) \
00364

```

```

00365 #endif//RHO_GPIO
00366
00367
00380 #define TEMP_SENSOR_ADC_CHANNEL ADC_SOURCE_ADC0_VREF2
00381
00384 #define TEMP_SENSOR_SCALE_FACTOR 1
00385
00400 #define PACKET_TRACE // We do have PACKET_TRACE support
00401
00402
00438 #define OSC32K_STARTUP_DELAY_MS (0)
00439
00440 #if OSC32K_STARTUP_DELAY_MS > MAX_INT16U_VALUE
00441 #error "OSC32K_STARTUP_DELAY_MS must fit in 16 bits."
00442 #endif
00443
00450 //">#define ENABLE_OSC32K // Enable 32.768 kHz osc instead of 1 kHz RC osc
00452
00453
00470 //">#define ENABLE_ALT_FUNCTION_REG_EN
00472
00473
00490 #define ENABLE_ALT_FUNCTION_TX_ACTIVE
00491
00492
00493
00512 //">#define ENABLE_ALT_FUNCTION_NTX_ACTIVE
00514
00529 #define EEPROM_USES_SHUTDOWN_CONTROL
00530
00531
00532
00544
00545
00558 #ifdef PACKET_TRACE
00559     #define PWRUP_CFG_PTI_EN    GPIOCFG_OUT_ALT
00560     #define PWRUP_OUT_PTI_EN    0
00561     #define PWRDN_CFG_PTI_EN    GPIOCFG_IN_PUD
00562     #define PWRDN_OUT_PTI_EN    GPIOOUT_PULLDOWN
00563     #define PWRUP_CFG_PTI_DATA  GPIOCFG_OUT_ALT
00564     #define PWRUP_OUT_PTI_DATA  1
00565     #define PWRDN_CFG_PTI_DATA  GPIOCFG_IN_PUD
00566     #define PWRDN_OUT_PTI_DATA  GPIOOUT_PULLUP
00567 #else
00568     #define PWRUP_CFG_PTI_EN    GPIOCFG_IN
00569     #define PWRUP_OUT_PTI_EN    0
00570     #define PWRDN_CFG_PTI_EN    GPIOCFG_IN
00571     #define PWRDN_OUT_PTI_EN    0
00572     #define PWRUP_CFG_PTI_DATA  GPIOCFG_IN
00573     #define PWRUP_OUT_PTI_DATA  0
00574     #define PWRDN_CFG_PTI_DATA  GPIOCFG_IN
00575     #define PWRDN_OUT_PTI_DATA  0
00576 #endif//PACKET_TRACE
00577
00578
00579
00602 #if defined(ENABLE_OSC32K) && defined(ENABLE_ALT_FUNCTION_NTX_ACTIVE)
00603     //Oops! Only one of these can be used at a time!
00604     #error ENABLE_OSC32K and ENABLE_ALT_FUNCTION_NTX_ACTIVE are mutually\
00605 exclusive. They define conflicting usage for GPIO PC6.
00606
00607 #elif defined(ENABLE_OSC32K) && !defined(ENABLE_ALT_FUNCTION_NTX_ACTIVE)
00608     //Use OCS32K configuration
00609     #define PWRUP_CFG_BUTTON1  GPIOCFG_ANALOG
00610     #define PWRUP_OUT_BUTTON1  0
00611     #define PWRDN_CFG_BUTTON1  GPIOCFG_ANALOG
00612     #define PWRDN_OUT_BUTTON1  0
00613
00614 #elif !defined(ENABLE_OSC32K) && defined(ENABLE_ALT_FUNCTION_NTX_ACTIVE)
00615     //Use nTX_ACTIVE configuration
00616     #define PWRUP_CFG_BUTTON1  GPIOCFG_OUT_ALT
00617     #define PWRUP_OUT_BUTTON1  0
00618     #define PWRDN_CFG_BUTTON1  GPIOCFG_OUT
00619     #define PWRDN_OUT_BUTTON1  0
00620
00621 #else
00622     //Use Button1 configuration
00623     #define PWRUP_CFG_BUTTON1  GPIOCFG_IN_PUD
00624     #define PWRUP_OUT_BUTTON1  GPIOOUT_PULLUP /* Button needs a pullup */
00625     #define PWRDN_CFG_BUTTON1  GPIOCFG_IN_PUD

```

```

00626     #define PWRDN_OUT_BUTTON1  GPIOOUT_PULLUP /* Button needs a pullup */
00627
00628 #endif
00629
00633 #ifdef ENABLE_OSC32K
00634     #define CFG_TEMPEN      GPIOCFG_ANALOG
00635 #else
00636     #define CFG_TEMPEN      GPIOCFG_OUT
00637 #endif//ENABLE_OSC32K
00638
00639
00640
00651 #ifdef ENABLE_ALT_FUNCTION_TX_ACTIVE
00652     #define PWRUP_CFG_LED2  GPIOCFG_OUT_ALT
00653     #define PWRUP_OUT_LED2  0
00654     #define PWRDN_CFG_LED2  GPIOCFG_OUT
00655     #define PWRDN_OUT_LED2  0
00656 #else
00657     #define PWRUP_CFG_LED2  GPIOCFG_OUT
00658     #define PWRUP_OUT_LED2  1 /* LED default off */
00659     #define PWRDN_CFG_LED2  GPIOCFG_OUT
00660     #define PWRDN_OUT_LED2  1 /* LED default off */
00661 #endif//ENABLE_ALT_FUNCTION_TX_ACTIVE
00662
00663
00664
00673 //Each pin has 4 cfg bits. There are 3 ports with 2 cfg registers per
00674 //port since the cfg register only holds 2 pins (16bits). Therefore,
00675 //the cfg arrays need to be 6 entries of 16bits.
00676 extern uint16_t gpioCfgPowerUp[6];
00677 extern uint16_t gpioCfgPowerDown[6];
00678 //Each pin has 1 out bit. There are 3 ports with 1 out register per
00679 //port (8bits). Therefore, the out arrays need to be 3 entries of 8bits.
00680 extern uint8_t gpioOutPowerUp[3];
00681 extern uint8_t gpioOutPowerDown[3];
00682 //A single mask variable covers all GPIO.
00683 extern GpioMaskType gpioRadioPowerBoardMask;
00684
00685
00686 // Set up FEM control signals based on ENABLE_ALT_FUNCTION_xxx defines above
00687 // in preparation for GPIO Radio Power Board Mask below:
00688 // -PA control line (Tx_Active)
00689 #ifdef ENABLE_ALT_FUNCTION_TX_ACTIVE
00690     #define FEM_CTX_BIT (BIT32(PORTC_PIN(5)))
00691 #else
00692     #define FEM_CTX_BIT (0)
00693 #endif
00694 // -LNA control line (!Tx_Active)
00695 #ifdef ENABLE_ALT_FUNCTION_NTX_ACTIVE
00696     #define FEM_CRX_BIT (BIT32(PORTC_PIN(6)))
00697 #else
00698     #define FEM_CRX_BIT (0)
00699 #endif
00700
00707 #define DEFINE_GPIO_RADIO_POWER_BOARD_MASK_VARIABLE() \
00708 GpioMaskType gpioRadioPowerBoardMask = { ( BIT32(PORTB_PIN(0)) /* FEM_CSD */ \
00709 \
00710 \
00711 \
00712 \
00713 \
00714 \
00718 #define DEFINE_POWERUP_GPIO_CFG_VARIABLES() \
00719 uint16_t gpioCfgPowerUp[6] = { \
00720 \
00721 \
00722 \
00723 \
00724 \
00725 \
00726 \
00727 \
00728 \
00729 \
00730

```



```

00802           (GPIOCFG_OUT      <<PB5_CFG_BIT) |
00803           (GPIOCFG_IN_PUD   <<PB6_CFG_BIT) |
00804           /* need to use pulldown for sleep */
00805           (GPIOCFG_IN_PUD   <<PB7_CFG_BIT)), ,
00806           ((GPIOCFG_IN_PUD   <<PC0_CFG_BIT) |
00807           (GPIOCFG_OUT      <<PC1_CFG_BIT) |
00808           (GPIOCFG_OUT      <<PC2_CFG_BIT) |
00809           (GPIOCFG_IN_PUD   <<PC3_CFG_BIT)), ,
00810           ((GPIOCFG_IN_PUD   <<PC4_CFG_BIT) |
00811           (PWRDN_CFG_LED2   <<PC5_CFG_BIT) |
00812           (PWRDN_CFG_BUTTON1 <<PC6_CFG_BIT) |
00813           (CFG_TEMPEN      <<PC7_CFG_BIT)) |
00814       }
00815
00816
00820 #define DEFINE_POWERDOWN_GPIO_OUTPUT_DATA_VARIABLES()
00821 uint8_t gpioOutPowerDown[3] = {
00822     ((GPIOOUT_PULLUP    <<PA0_BIT) |
00823     (GPIOOUT_PULLUP    <<PA1_BIT) |
00824     (GPIOOUT_PULLUP    <<PA2_BIT) |
00825     /* nSEL is idle high */
00826     (1                  <<PA3_BIT) |
00827     /* enable is idle low */
00828     (PWRDN_OUT_PTI_EN  <<PA4_BIT) |
00829     /* data is idle high */
00830     (PWRDN_OUT_PTI_DATA <<PA5_BIT) |
00831     (PWRDN_OUT_DFL_RHO <<PA6_BIT) |
00832     /* LED off */
00833     (1                  <<PA7_BIT)), ,
00834     ((0                <<PB0_BIT) |
00835     (GPIOOUT_PULLUP    <<PB1_BIT) | /* SC1TXD */ |
00836     (GPIOOUT_PULLUP    <<PB2_BIT) | /* SC1RXD */ |
00837     (GPIOOUT_PULLDOWN <<PB3_BIT) | /* SC1nCTS */ |
00838     (GPIOOUT_PULLUP    <<PB4_BIT) | /* SC1nRTS */ |
00839     /* SiGe SE2432L CPS pin needs to be driven low */
00840     (0                  <<PB5_BIT) |
00841     /* PB6 has button needing a pullup */
00842     (GPIOOUT_PULLUP    <<PB6_BIT) |
00843     /* buzzer needs pulldown for sleep */
00844     (GPIOOUT_PULLDOWN <<PB7_BIT)), ,
00845     ((GPIOOUT_PULLUP    <<PC0_BIT) |
00846     (0                  <<PC1_BIT) |
00847     (1                  <<PC2_BIT) |
00848     (GPIOOUT_PULLDOWN <<PC3_BIT) |
00849     (GPIOOUT_PULLDOWN <<PC4_BIT) |
00850     (PWRDN_OUT_LED2   <<PC5_BIT) |
00851     (PWRDN_OUT_BUTTON1 <<PC6_BIT) |
00852     /* Temp Sensor off */
00853     (0                  <<PC7_BIT))
00854   }
00855
00856
00860 #define SET_POWERUP_GPIO_CFG_REGISTERS() \
00861   GPIO_PACFGL = gpioCfgPowerUp[0]; \
00862   GPIO_PACFGH = gpioCfgPowerUp[1]; \
00863   GPIO_PBCFGL = gpioCfgPowerUp[2]; \
00864   GPIO_PBCFGH = gpioCfgPowerUp[3]; \
00865   GPIO_PCCFGL = gpioCfgPowerUp[4]; \
00866   GPIO_PCCFGH = gpioCfgPowerUp[5];
00867
00868
00872 #define SET_POWERUP_GPIO_OUTPUT_DATA_REGISTERS() \
00873   GPIO_PAOUT = gpioOutPowerUp[0];
00874   GPIO_PBOUT = gpioOutPowerUp[1];
00875   GPIO_PCOUT = gpioOutPowerUp[2];
00876
00877

```

```

0081 #define SET_POWERDOWN_GPIO_CFG_REGISTERS() \
0082     GPIO_PACFGL = gpioCfgPowerDown[0]; \
0083     GPIO_PACFGH = gpioCfgPowerDown[1]; \
0084     GPIO_PBCFGL = gpioCfgPowerDown[2]; \
0085     GPIO_PBCFGH = gpioCfgPowerDown[3]; \
0086     GPIO_PCCFGL = gpioCfgPowerDown[4]; \
0087     GPIO_PCCFGH = gpioCfgPowerDown[5];
0088
0089
0093 #define SET_POWERDOWN_GPIO_OUTPUT_DATA_REGISTERS() \
0094     GPIO_PAOUT = gpioOutPowerDown[0]; \
0095     GPIO_PBOUT = gpioOutPowerDown[1]; \
0096     GPIO_PCOUT = gpioOutPowerDown[2];
0097
0098
0099
00903 #ifdef ENABLE_ALT_FUNCTION_REG_EN
00904     #define CONFIGURE_EXTERNAL_REGULATOR_ENABLE()    GPIO_DBGCFG |= GPIO_EXTREGEN;
00905 #else
00906     #define CONFIGURE_EXTERNAL_REGULATOR_ENABLE()    GPIO_DBGCFG &= ~GPIO_EXTREGEN;
00907 #endif
00908
00909
00910
00921 #define WAKE_ON_PA0    false
00922 #define WAKE_ON_PA1    false
00923 #define WAKE_ON_PA2    false
00924 #define WAKE_ON_PA3    false
00925 #define WAKE_ON_PA4    false
00926 #define WAKE_ON_PA5    false
00927 #define WAKE_ON_PA6    false
00928 #define WAKE_ON_PA7    false
00929 #define WAKE_ON_PB0    false
00930 #define WAKE_ON_PB1    false
00931 #ifdef SLEEPY_EZSP_UART // SC1RXD
00932     #define WAKE_ON_PB2    true
00933 #else
00934     #define WAKE_ON_PB2    false
00935 #endif
00936 #define WAKE_ON_PB3    false
00937 #define WAKE_ON_PB4    false
00938 #define WAKE_ON_PB5    false
00939 #define WAKE_ON_PB6    true   //BUTTON0
00940 #define WAKE_ON_PB7    false
00941 #define WAKE_ON_PC0    false
00942 #define WAKE_ON_PC1    false
00943 #define WAKE_ON_PC2    false
00944 #define WAKE_ON_PC3    false
00945 #define WAKE_ON_PC4    false
00946 #define WAKE_ON_PC5    false
00947 #define WAKE_ON_PC6    true   //BUTTON1
00948 #define WAKE_ON_PC7    false
00949
00950
00951
00953
00954
00967 #ifndef EZSP_ASH
00968     #define halInternalInitBoard()
00969         do {
00970             halInternalPowerUpBoard();
00971             halInternalRestartUart();
00972             halInternalInitButton();
00973             halInternalInitRadioHoldOff();
00974         } while(0)
00975 #else
00976     #define halInternalInitBoard()
00977         do {
00978             halInternalPowerUpBoard();
00979             halInternalRestartUart();
00980             halInternalInitRadioHoldOff();
00981         } while(0)
00982 #endif
00983
00988 #define halInternalPowerDownBoard()
00989     do {
00990         /* Board peripheral deactivation */
00991         /* halInternalSleepAdc(); */
00992         SET_POWERDOWN_GPIO_OUTPUT_DATA_REGISTERS()
00993         SET_POWERDOWN_GPIO_CFG_REGISTERS()

```

```

0094         } while(0)
0095
0100 #define halInternalPowerUpBoard()
0101     do {
0102         SET_POWERUP_GPIO_OUTPUT_DATA_REGISTERS()
0103         SET_POWERUP_GPIO_CFG_REGISTERS()
0104         /*The radio GPIO should remain in the powerdown state */
0105         /*until the stack specifically powers them up. */
0106         halStackRadioPowerDownBoard();
0107         CONFIGURE_EXTERNAL_REGULATOR_ENABLE()
0108         /* Board peripheral reactivation */
0109         halInternalInitAdc();
0110     } while(0)
0111
0112
0113 #endif //__BOARD_H__
0114

```

8.121 reset-def.h File Reference

8.121.1 Detailed Description

Definitions for all the reset cause types.

Definition in file [reset-def.h](#).

8.122 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
00039 //   information
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
00054 //           RESET_basename
00055 //           value - the value of the enum definition
00056 //           string - the reset string, must be 3 characters
00057 //           description - a comment describing the cause
00058
00059 // RESET_EXT_DEF macros take the following parameters:
00060 //   RESET_EXT_DEF(basename, extname, extvalue, string) // description
00061 //       basename - the base name used in the enum definition
00062 //       extname - the extended name used in the enum definition, expanded as
00063 //           RESET_basename_extname
00064 //           extvalue - the LSB value of the enum definition, combined with the
00065 //             value of the base value in the MSB
00066 //           string - the reset string, must be 3 characters
00067 //           description - a comment describing the cause
00068
00069 // ****
00070 // This file is specifically kept wider than 80 columns in order to keep the
00071 //   information
00072 //   well organized and easy to read by glancing down the columns
00073 //   (Yes, this comment is mentioned multiple times in the file)
00074 // ****

```

```

00071
00072 // This file wants to use the bareword BOOTLOADER, which is sometimes a symbol.
00073 // To allow this we will undefine the symbol here and restore it at the end of
00074 // this file.
00075 #ifdef BOOTLOADER
00076     #define SAVED_BOOTLOADER BOOTLOADER
00077     #undef BOOTLOADER
00078 #endif
00079
00080 //[[[
00081 //      The first 3 (unknown, fib, and bootloader) reset base types and their
00082 //      extended values should never be changed, as they are used by
00083 //      bootloaders in the field which can never be changed.
00084 //      The later reset base and extended types may be changed over time
00085 //]]
00086
00087 RESET_BASE_DEF(UNKNOWN,          0x00,    "UNK")    // Underterminable cause
00088 RESET_EXT_DEF(UNKNOWN, UNKNOWN, 0x00,    "UNK")    // Undeterminable cause
00089
00090 RESET_BASE_DEF(FIB,             0x01,    "FIB")    // Reset originated
00091     from the FIB bootloader
00092     RESET_EXT_DEF(FIB, GO,        0x00,    "GO ")    // FIB bootloader
00093     caused a reset to main flash
00094     RESET_EXT_DEF(FIB, BOOTLOADER, 0x01,    "BTL")    // FIB bootloader is
00095     instructing ember bootloader to run
00096     RESET_EXT_DEF(FIB, GO2,       0x02,    "G02")    // G02 (unused)
00097     RESET_EXT_DEF(FIB, GO3,       0x03,    "G03")    // G03 (unused)
00098     RESET_EXT_DEF(FIB, GO4,       0x04,    "G04")    // G04 (unused)
00099     RESET_EXT_DEF(FIB, GO5,       0x05,    "G05")    // G05 (unused)
00100    RESET_EXT_DEF(FIB, GO6,       0x06,    "G06")    // G06 (unused)
00101    RESET_EXT_DEF(FIB, GO7,       0x07,    "G07")    // G07 (unused)
00102    RESET_EXT_DEF(FIB, GO8,       0x08,    "G08")    // G08 (unused)
00103    RESET_EXT_DEF(FIB, GO9,       0x09,    "G09")    // G09 (unused)
00104    RESET_EXT_DEF(FIB, GOA,       0x0A,    "GOA")    // GOA (unused)
00105    RESET_EXT_DEF(FIB, GOB,       0x0B,    "GOB")    // GOB (unused)
00106    RESET_EXT_DEF(FIB, GOC,       0x0C,    "GOC")    // GOC (unused)
00107    RESET_EXT_DEF(FIB, GOD,       0x0D,    "GOD")    // GOD (unused)
00108    RESET_EXT_DEF(FIB, GOE,       0x0E,    "GOE")    // GOE (unused)
00109    RESET_EXT_DEF(FIB, GOF,       0x0F,    "GOF")    // GOF (unused)
00110    RESET_EXT_DEF(FIB, JUMP,      0x10,    "JMP")    // FIB bootloader is
00111     jumping to a specific flash address
00112     RESET_EXT_DEF(FIB, BAUDRATE, 0x11,    "BDR")    // FIB bootloader
00113     detected a high baud rate, causes ember bootloader to run
00114     RESET_EXT_DEF(FIB, UNPROTECT, 0x12,    "UPR")    // Read protection
00115     disabled, flash should be erased
00116     RESET_EXT_DEF(FIB, BOOTMODE, 0x13,    "BTM")    // BOOTMODE was not
00117     held long enough
00118     RESET_EXT_DEF(FIB, MISMATCH, 0x14,    "MSM")    // MISMATCHED FIB
00119     Bootloader & Part Data
00120     RESET_EXT_DEF(FIB, FATAL,    0x15,    "FTL")    // FIB Fatal Error
00121
00122 RESET_BASE_DEF(BOOTLOADER,        0x02,    "BTL")    // Reset relates to an
00123     Ember bootloader
00124     RESET_EXT_DEF(BOOTLOADER, UNKNOWN, 0x00,    "UNK")    // Unknown
00125     bootloader cause (should never occur)
00126     RESET_EXT_DEF(BOOTLOADER, GO,     0x01,    "GO ")    // Bootloader caused
00127     reset telling app to run
00128     RESET_EXT_DEF(BOOTLOADER, BOOTLOAD, 0x02,    "BTL")    // Application
00129     requested that bootloader runs
00130     RESET_EXT_DEF(BOOTLOADER, BADIMAGE, 0x03,    "BAD")    // Application
00131     bootloader detect bad external upgrade image
00132     RESET_EXT_DEF(BOOTLOADER, FATAL,   0x04,    "FTL")    // Fatal Error or
00133     assert in bootloader
00134     RESET_EXT_DEF(BOOTLOADER, FORCE,   0x05,    "FRC")    // Forced bootloader
00135     activation
00136     RESET_EXT_DEF(BOOTLOADER, OTAVALID, 0x06,    "OTA")    // OTA Bootloader
00137     mode activation
00138     RESET_EXT_DEF(BOOTLOADER, DEEPSLEEP, 0x07,    "DSL")    // Bootloader
00139     initiated deep sleep
00140
00141 //[[[ -- Reset types below here may be changed in the future if absolutely
00142     necessary -- ]]]
00143
00144 RESET_BASE_DEF(EXTERNAL,          0x03,    "EXT")    // External reset
00145     trigger
00146     RESET_EXT_DEF(EXTERNAL, UNKNOWN, 0x00,    "UNK")    // Unknown external
00147     cause (should never occur)
00148     RESET_EXT_DEF(EXTERNAL, PIN,     0x01,    "PIN")    // External pin reset
00149
00150 RESET_BASE_DEF(POWERON,           0x04,    "PWR")    // Poweron reset type,

```

```

    supply voltage < power-on threshold
00131 RESET_EXT_DEF(POWERON, UNKNOWN,          0x00,      "UNK") // Unknown poweron
00132     reset (should never occur)
00133 RESET_EXT_DEF(POWERON, HV,                0x01,      "HV ") // High voltage
00134     poweron
00135 RESET_EXT_DEF(POWERON, LV,                0x02,      "LV ") // Low voltage
00136     poweron
00137 RESET_BASE_DEF(WATCHDOG,                  0x05,      "WDG") // Watchdog reset
00138     occurred
00139 RESET_EXT_DEF(WATCHDOG, UNKNWON,          0x00,      "UNK") // Unknown watchdog
00140     reset (should never occur)
00141 RESET_EXT_DEF(WATCHDOG, EXPIRED,          0x01,      "EXP") // Watchdog timer
00142     expired
00143 RESET_EXT_DEF(WATCHDOG, CAUGHT,           0x02,      "LWM") // Watchdog low
00144     watermark expired and caught extended info
00145 RESET_BASE_DEF(SOFTWARE,                  0x06,      "SW") // Software triggered
00146     reset
00147 RESET_EXT_DEF(SOFTWARE, UNKNOWN,          0x00,      "UNK") // Unknown software
00148     cause
00149 RESET_EXT_DEF(SOFTWARE, REBOOT,           0x01,      "RBT") // General software
00150     reboot
00151 RESET_EXT_DEF(SOFTWARE, DEEPSLEEP,         0x02,      "DSL") // App initiated deep
00152     sleep
00153 RESET_BASE_DEF(CRASH,                     0x07,      "CRS") // Software crash
00154 RESET_EXT_DEF(CRASH, UNKNOWN,             0x00,      "UNK") // Unknown crash
00155     assert in the code failed
00156 RESET_EXT_DEF(CRASH, ASSERT,              0x01,      "AST") // a self-check
00157     assert in the code failed
00158 RESET_BASE_DEF(FLASH,                      0x08,      "FSH") // Flash failure cause
00159     reset
00160 RESET_EXT_DEF(FLASH, UNKNWON,             0x00,      "UNK") // Unknown flash
00161     failure
00162 RESET_EXT_DEF(FLASH, VERIFY,              0x01,      "VFY") // Flash write verify
00163     failure
00164 RESET_EXT_DEF(FLASH, INHIBIT,             0x02,      "INH") // Flash write
00165     inhibited: already written
00166 RESET_BASE_DEF(FATAL,                     0x09,      "BAD") // A non-recoverable
00167     fatal error occurred
00168 RESET_EXT_DEF(FATAL, UNKNOWN,             0x00,      "UNK") // Unknown fatal
00169     error (should never occur)
00170 RESET_EXT_DEF(FATAL, LOCKUP,              0x01,      "LCK") // CPU Core locked up
00171 RESET_EXT_DEF(FATAL, CRYSTAL,             0x02,      "XTL") // 24MHz crystal
00172     failure
00173 RESET_EXT_DEF(FATAL, OPTIONBYTE,          0x03,      "OBF") // option byte
00174     complement error
00175 RESET_BASE_DEF(FAULT,                    0x0A,      "FLT") // A access fault
00176     occurred
00177 RESET_EXT_DEF(FAULT, UNKNOWN,             0x00,      "UNK") // An unknown fault
00178     occurred
00179 RESET_EXT_DEF(FAULT, HARD,                0x01,      "HRD") // Hard fault
00180 RESET_EXT_DEF(FAULT, MEM,                 0x02,      "MEM") // Memory protection
00181     violation
00182 RESET_EXT_DEF(FAULT, BUS,                 0x03,      "BUS") // Bus fault
00183 RESET_EXT_DEF(FAULT, USAGE,               0x04,      "USG") // Usage fault
00184 RESET_EXT_DEF(FAULT, DBGMON,              0x05,      "DBG") // Debug monitor
00185     fault
00186 RESET_EXT_DEF(FAULT, PROTDMA,             0x06,      "DMA") // DMA RAM protection
00187     violation
00188 RESET_EXT_DEF(FAULT, BADVECTOR,           0x07,      "VCT") // Uninitialized
00189     interrupt vector
00190 // Restore the value of the BOOTLOADER symbol if we had to save it off in this
00191 // file so that the word BOOTLOADER could be used.
00192 #ifdef SAVED_BOOTLOADER
00193     #define BOOTLOADER SAVED_BOOTLOADER
00194     #undef SAVED_BOOTLOADER
00195 #endif
00196
00197 #endif //DOXYGEN_SHOULD_SKIP_THIS
00198

```

8.123 rf4ce-api.h File Reference

Functions

- `EmberStatus emberRf4ceSetPairingTableEntry (uint8_t pairingIndex, EmberRf4cePairingTableEntry *entry)`
- `EmberStatus emberRf4ceGetPairingTableEntry (uint8_t pairingIndex, EmberRf4cePairingTableEntry *entry)`
- `EmberStatus emberRf4ceSetApplicationInfo (EmberRf4ceApplicationInfo *appInfo)`
- `EmberStatus emberRf4ceGetApplicationInfo (EmberRf4ceApplicationInfo *appInfo)`
- `uint8_t emberRf4ceGetBaseChannel (void)`
- `EmberStatus emberRf4ceKeyUpdate (uint8_t pairingIndex, EmberKeyData *key)`
- `EmberStatus emberRf4ceSend (uint8_t pairingIndex, uint8_t profileId, uint16_t vendorId, EmberRf4ceTxOption txOptions, uint8_t messageTag, uint8_t messageLength, uint8_t *message)`
- `void emberRf4ceMessageSentHandler (EmberStatus status, uint8_t pairingIndex, EmberRf4ceTxOption txOptions, uint8_t profileId, uint16_t vendorId, uint8_t messageTag, uint8_t messageLength, uint8_t *message)`
- `void emberRf4ceIncomingMessageHandler (uint8_t pairingIndex, uint8_t profileId, uint16_t vendorId, EmberRf4ceTxOption txOptions, uint8_t messageLength, uint8_t *message)`
- `uint8_t emberRf4ceGetMaxPayload (uint8_t pairingIndex, EmberRf4ceTxOption txOptions)`
- `EmberStatus emberRf4ceStart (EmberRf4ceNodeCapabilities capabilities, EmberRf4ceVendorInfo *vendorInfo, int8_t power)`
- `EmberStatus emberRf4ceStop (void)`
- `EmberStatus emberRf4ceDiscovery (EmberPanId panId, EmberNodeId nodeId, uint8_t searchDevType, uint16_t discDuration, uint8_t maxDiscRepetitions, uint8_t discProfileIdListLength, uint8_t *discProfileIdList)`
- `void emberRf4ceDiscoveryCompleteHandler (EmberStatus status)`
- `bool emberRf4ceDiscoveryRequestHandler (EmberEUI64 srcIeeeAddr, uint8_t nodeCapabilities, EmberRf4ceVendorInfo *vendorInfo, EmberRf4ceApplicationInfo *appInfo, uint8_t searchDevType, uint8_t rxLinkQuality)`
- `bool emberRf4ceDiscoveryResponseHandler (bool atCapacity, uint8_t channel, EmberPanId panId, EmberEUI64 srcIeeeAddr, uint8_t nodeCapabilities, EmberRf4ceVendorInfo *vendorInfo, EmberRf4ceApplicationInfo *appInfo, uint8_t rxLinkQuality, uint8_t discRequestLqi)`
- `EmberStatus emberRf4ceEnableAutoDiscoveryResponse (uint16_t duration)`
- `void emberRf4ceAutoDiscoveryResponseCompleteHandler (EmberStatus status, EmberEUI64 srcIeeeAddr, uint8_t nodeCapabilities, EmberRf4ceVendorInfo *vendorInfo, EmberRf4ceApplicationInfo *appInfo, uint8_t searchDevType)`
- `EmberStatus emberRf4cePair (uint8_t channel, EmberPanId panId, EmberEUI64 ieeeAddr, uint8_t keyExchangeTransferCount)`
- `void emberRf4cePairCompleteHandler (EmberStatus status, uint8_t pairingIndex, EmberRf4ceVendorInfo *vendorInfo, EmberRf4ceApplicationInfo *appInfo)`
- `bool emberRf4cePairRequestHandler (EmberStatus status, uint8_t pairingIndex, EmberEUI64 srcIeeeAddr, uint8_t nodeCapabilities, EmberRf4ceVendorInfo *vendorInfo, EmberRf4ceApplicationInfo *appInfo, uint8_t keyExchangeTransferCount)`
- `EmberStatus emberRf4ceUnpair (uint8_t pairingIndex)`
- `void emberRf4ceUnpairHandler (uint8_t pairingIndex)`
- `void emberRf4ceUnpairCompleteHandler (uint8_t pairingIndex)`

- `EmberStatus emberRf4ceSetPowerSavingParameters` (`uint32_t` `dutyCycle`, `uint32_t` `activePeriod`)
- `EmberStatus emberRf4ceSetFrequencyAgilityParameters` (`uint8_t` `rssiWindowSize`, `uint8_t` `channelChangeReads`, `int8_t` `rssiThreshold`, `uint16_t` `readInterval`, `uint8_t` `readDuration`)
- `EmberStatus emberRf4ceSetDiscoveryLqiThreshold` (`uint8_t` `threshold`)

8.123.1 Detailed Description

ZigBee RF4CE stack APIs and callbacks. See [Ember ZigBee RF4CE APIs and Handlers](#) for documentation.

Definition in file `rf4ce-api.h`.

8.124 rf4ce-api.h

```

00001
00015 #ifndef __RF4CE_API_H__
00016 #define __RF4CE_API_H__
00017
00034 EmberStatus emberRf4ceSetPairingTableEntry
00035     (uint8_t pairingIndex,
00036      EmberRf4cePairingTableEntry
00037      *entry);
00036
00051 EmberStatus emberRf4ceGetPairingTableEntry
00052     (uint8_t pairingIndex,
00053      EmberRf4cePairingTableEntry
00054      *entry);
00053
00064 EmberStatus emberRf4ceSetApplicationInfo
00065     (EmberRf4ceApplicationInfo *appInfo);
00065
00076 EmberStatus emberRf4ceGetApplicationInfo
00077     (EmberRf4ceApplicationInfo *appinfo);
00077
00078
00083 uint8_t emberRf4ceGetBaseChannel(void);
00084
00101 EmberStatus emberRf4ceKeyUpdate(uint8_t
00102     pairingIndex, EmberKeyData *key);
00102
00154 EmberStatus emberRf4ceSend(uint8_t pairingIndex,
00155     uint8_t profileId,
00156     uint16_t vendorId,
00157     EmberRf4ceTxOption txOptions,
00158     uint8_t messageTag,
00159     uint8_t messageLength,
00160     uint8_t *message);
00161
00186 void emberRf4ceMessageSentHandler(EmberStatus
00187     status,
00188     uint8_t pairingIndex,
00189     EmberRf4ceTxOption
00190     txOptions,
00191     uint8_t profileId,
00192     uint16_t vendorId,
00193     uint8_t messageTag,
00194     uint8_t messageLength,
00195     uint8_t *message);
00194
00212 void emberRf4ceIncomingMessageHandler(uint8_t
00213     pairingIndex,
00214     uint8_t profileId,
00215     uint16_t vendorId,
00216     EmberRf4ceTxOption
00217     txOptions,
00218     uint8_t messageLength,
00219     uint8_t *message);
00219

```

```

00218
00234 uint8_t emberRf4ceGetMaxPayload(uint8_t pairingIndex,
00235                                     EmberRf4ceTxOption txOptions);
00236
00261 EmberStatus emberRf4ceStart(
00262     EmberRf4ceNodeCapabilities capabilities,
00263     EmberRf4ceVendorInfo * vendorInfo,
00264     int8_t power);
00265 /* @brief The node stops the network operations and clears the network state
00266 * and the pairing table stored in persistent memory.
00267 */
00268 * @return An ::EmberStatus value of ::EMBER_SUCCESS if the network
00269 * operations
00270 * are successfully terminated, ::EMBER_INVALID_CALL if the current network is
00271 * not an RF4CE network or the current network is not up or the stack is
00272 * currently handling some discovery or pairing process.
00273 */
00273 EmberStatus emberRf4ceStop(void);
00274
00313 EmberStatus emberRf4ceDiscovery(EmberPanId
00314     panId,
00315             EmberNodeId nodeId,
00316             uint8_t searchDevType,
00317             uint16_t discDuration,
00318             uint8_t maxDiscRepetitions,
00319             uint8_t discProfileIdListLength,
00320             uint8_t *discProfileIdList);
00333 void emberRf4ceDiscoveryCompleteHandler(
00334     EmberStatus status);
00365 bool emberRf4ceDiscoveryRequestHandler(
00366     EmberEUI64 srcIeeeAddr,
00367             uint8_t nodeCapabilities,
00368             EmberRf4ceVendorInfo
00369             *vendorInfo,
00370             EmberRf4ceApplicationInfo
00371             *appInfo,
00372             uint8_t searchDevType,
00373             uint8_t rxLinkQuality);
00404 bool emberRf4ceDiscoveryResponseHandler(bool
00405     atCapacity,
00406             uint8_t channel,
00407             EmberPanId panId,
00408             EmberEUI64 srcIeeeAddr,
00409             uint8_t nodeCapabilities,
00410             EmberRf4ceVendorInfo
00411             *vendorInfo,
00412             EmberRf4ceApplicationInfo
00413             *appInfo,
00414             uint8_t rxLinkQuality,
00415             uint8_t discRequestLqi);
00433 EmberStatus emberRf4ceEnableAutoDiscoveryResponse
00434     (uint16_t duration);
00478 void emberRf4ceAutoDiscoveryResponseCompleteHandler
00479     (EmberStatus status,
00480         EmberEUI64
00481         srcIeeeAddr,
00482             uint8_t nodeCapabilities,
00483             EmberRf4ceVendorInfo
00484             *vendorInfo,
00485             EmberRf4ceApplicationInfo
00486             *appInfo,
00487             uint8_t searchDevType);
00507 EmberStatus emberRf4cePair(uint8_t channel,
00508             EmberPanId panId,
00509             EmberEUI64 ieeeAddr,
00510             uint8_t keyExchangeTransferCount);
00511
00552 void emberRf4cePairCompleteHandler(EmberStatus
00553     status,
00554         uint8_t pairingIndex,
00555         EmberRf4ceVendorInfo *
00556         vendorInfo,

```

```

00555             *appInfo);
00556
00592     bool emberRf4cePairRequestHandler(EmberStatus
00593         status,
00594             uint8_t pairingIndex,
00595             EmberEUI64 srcIEEEAddr,
00596             uint8_t nodeCapabilities,
00597             EmberRf4ceVendorInfo *
00598             vendorInfo,
00599             *appInfo,
00600             uint8_t keyExchangeTransferCount);
00616     EmberStatus emberRf4ceUnpair(uint8_t pairingIndex);
00617
00628     void emberRf4ceUnpairHandler(uint8_t pairingIndex);
00629
00640     void emberRf4ceUnpairCompleteHandler(uint8_t
00641         pairingIndex);
00641
00670     EmberStatus emberRf4ceSetPowerSavingParameters
00671         (uint32_t dutyCycle,
00672             uint32_t activePeriod);
00672
00722     EmberStatus emberRf4ceSetFrequencyAgilityParameters
00723         (uint8_t rssiWindowSize,
00724             uint8_t channelChangeReads,
00725             int8_t rssiThreshold,
00726             uint16_t readInterval,
00727             uint8_t readDuration);
00727
00737     EmberStatus emberRf4ceSetDiscoveryLqiThreshold
00738         (uint8_t threshold);
00739 #endif //__RF4CE_API_H__
00740

```

8.125 rf4ce-types.h File Reference

Data Structures

- struct [EmberRf4ceVendorInfo](#)
Defines the vendor information block (see section 3.3.1, Figure 16).
- struct [EmberRf4ceApplicationInfo](#)
Defines the application information block (see section 3.3.1, Figure 17).
- struct [EmberRf4cePairingTableEntry](#)
The internal representation of a pairing table entry.

RF4CE Types

- #define EMBER_RF4CE_BROADCAST_ADDRESS
- #define EMBER_RF4CE_BROADCAST_PAN_ID
- #define EMBER_ALL_ZIGBEE_RF4CE_CHANNELS_MASK
- #define EMBER_RF4CE_NULL_VENDOR_ID
- #define EMBER_RF4CE_VENDOR_STRING_LENGTH
- #define EMBER_RF4CE_APPLICATION_USER_STRING_LENGTH
- #define EMBER_RF4CE_APPLICATION_DEVICE_TYPE_LIST_MAX_LENGTH
- #define EMBER_RF4CE_APPLICATION_PROFILE_ID_LIST_MAX_LENGTH
- #define EMBER_RF4CE_PAIRING_TABLE_BROADCAST_INDEX
- #define EMBER_RF4CE_MIN_ACTIVE_PERIOD_MS

- #define EMBER_RF4CE_MAX_DUTY_CYCLE_MS
- #define EMBER_RF4CE_PAIRING_TABLE_ENTRY_INFO_STATUS_MASK
- #define EMBER_RF4CE_PAIRING_TABLE_ENTRY_INFO_HAS_LINK_KEY_BIT
- #define EMBER_RF4CE_PAIRING_TABLE_ENTRY_INFO_IS_PAIRING_INITIATOR_BIT
- enum EmberRf4ceTxOption {
 EMBER_RF4CE_TX_OPTIONS_NONE, EMBER_RF4CE_TX_OPTIONS_BROADCAST_BIT, EMBER_RF4CE_TX_OPTIONS_USE_IEEE_ADDRESS_BIT, EMBER_RF4CE_TX_OPTIONS_ACK_REQUESTED_BIT,
 EMBER_RF4CE_TX_OPTIONS_SECURITY_ENABLED_BIT, EMBER_RF4CE_TX_OPTIONS_SINGLE_CHANNEL_BIT, EMBER_RF4CE_TX_OPTIONS_CHANNEL_DESIGNATOR_BIT, EMBER_RF4CE_TX_OPTIONS_VENDOR_SPECIFIC_BIT
 }
- enum EmberRf4ceNodeCapabilities {
 EMBER_RF4CE_NODE_CAPABILITIES_NONE, EMBER_RF4CE_NODE_CAPABILITIES_IS_TARGET_BIT, EMBER_RF4CE_NODE_CAPABILITIES_POWER_SOURCE_BIT, EMBER_RF4CE_NODE_CAPABILITIES_SECURITY_BIT, EMBER_RF4CE_NODE_CAPABILITIES_CHANNEL_NORM_BIT
 }
- enum EmberRf4ceApplicationCapabilities {
 EMBER_RF4CE_APP_CAPABILITIES_NONE, EMBER_RF4CE_APP_CAPABILITIES_USER_STRING_BIT, EMBER_RF4CE_APP_CAPABILITIES_SUPPORTED_DEVICE_TYPES_MASK, EMBER_RF4CE_APP_CAPABILITIES_SUPPORTED_DEVICE_TYPES_OFFSET,
 EMBER_RF4CE_APP_CAPABILITIES_SUPPORTED_PROFILES_MASK, EMBER_RF4CE_APP_CAPABILITIES_SUPPORTED_PROFILES_OFFSET
 }
- enum EmberRf4cePairingEntryStatus { EMBER_RF4CE_PAIRING_TABLE_ENTRY_STATUS_UNUSED, EMBER_RF4CE_PAIRING_TABLE_ENTRY_STATUS_PROVISIONAL, EMBER_RF4CE_PAIRING_TABLE_ENTRY_STATUS_ACTIVE }

8.125.1 Detailed Description

Zigbee RF4CE types and defines. See [Ember ZigBee RF4CE APIs and Handlers](#) for documentation.

Definition in file [rf4ce-types.h](#).

8.126 rf4ce-types.h

```

00001
00016 #ifndef __RF4CE_TYPES_H__
00017 #define __RF4CE_TYPES_H__
00018
00023
00027 #define EMBER_RF4CE_BROADCAST_ADDRESS          0xFFFF
00028
00032 #define EMBER_RF4CE_BROADCAST_PAN_ID           0xFFFF
00033
00037 #define EMBER_ALL_ZIGBEE_RF4CE_CHANNELS_MASK   0x02108000UL
00038
00039 /*
00040 * @brief A distinguished vendor id that is used to indicate the absence of a
00041 * vendor-specific command.
00042 */
00043 #define EMBER_RF4CE_NULL_VENDOR_ID 0x0000
00044

```

```

00049 #define EMBER_RF4CE_VENDOR_STRING_LENGTH 7
00050
00055 #define EMBER_RF4CE_APPLICATION_USER_STRING_LENGTH 15
00056
00061 #define EMBER_RF4CE_APPLICATION_DEVICE_TYPE_LIST_MAX_LENGTH 3
00062
00067 #define EMBER_RF4CE_APPLICATION_PROFILE_ID_LIST_MAX_LENGTH 7
00068
00073 #define EMBER_RF4CE_PAIRING_TABLE_BROADCAST_INDEX 0xFF
00074
00078 #define EMBER_RF4CE_MIN_ACTIVE_PERIOD_MS 17
00079
00083 #define EMBER_RF4CE_MAX_DUTY_CYCLE_MS 1000
00084
00090 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00091 enum EmberRf4ceTxOption
00092 #else
00093 typedef uint8_t EmberRf4ceTxOption;
00094 enum
00095 #endif
00096 {
00100     EMBER_RF4CE_TX_OPTIONS_NONE
00101         = 0x00,
00104     EMBER_RF4CE_TX_OPTIONS_BROADCAST_BIT
00105         = 0x01,
00109     EMBER_RF4CE_TX_OPTIONS_USE_IEEE_ADDRESS_BIT
00110         = 0x02,
00113     EMBER_RF4CE_TX_OPTIONS_ACK_REQUESTED_BIT
00114         = 0x04,
00117     EMBER_RF4CE_TX_OPTIONS_SECURITY_ENABLED_BIT
00118         = 0x08,
00121     EMBER_RF4CE_TX_OPTIONS_SINGLE_CHANNEL_BIT
00122         = 0x10,
00125     EMBER_RF4CE_TX_OPTIONS_CHANNEL_DESIGNATOR_BIT
00126         = 0x20,
00129     EMBER_RF4CE_TX_OPTIONS_VENDOR_SPECIFIC_BIT
00130         = 0x40
00131 };
00131
00137 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00138 enum EmberRf4ceNodeCapabilities
00139 #else
00140 typedef uint8_t EmberRf4ceNodeCapabilities;
00141 enum
00142 #endif
00143 {
00144     EMBER_RF4CE_NODE_CAPABILITIES_NONE
00145         = 0x00,
00150     EMBER_RF4CE_NODE_CAPABILITIES_IS_TARGET_BIT
00151         = 0x01,
00156     EMBER_RF4CE_NODE_CAPABILITIES_POWER_SOURCE_BIT
00157         = 0x02,
00163     EMBER_RF4CE_NODE_CAPABILITIES_SECURITY_BIT
00164         = 0x04,
00170     EMBER_RF4CE_NODE_CAPABILITIES_CHANNEL_NORM_BIT
00171         = 0x08
00171     // Bits 4-7 are reserved
00173 };
00174
00180 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00181 enum EmberRf4ceApplicationCapabilities
00182 #else
00183 typedef uint8_t EmberRf4ceApplicationCapabilities
00184 ;
00185 enum
00186 #endif
00186 {
00187     EMBER_RF4CE_APP_CAPABILITIES_NONE
00188         = 0x00,
00194     EMBER_RF4CE_APP_CAPABILITIES_USER_STRING_BIT
00195         = 0x01,
00200     EMBER_RF4CE_APP_CAPABILITIES_SUPPORTED_DEVICE_TYPES_MASK
00201         = 0x06,
00205     EMBER_RF4CE_APP_CAPABILITIES_SUPPORTED_DEVICE_TYPES_OFFSET
00206         = 1,
00206
00207     // Bit 3 is reserved.
00208
00214     EMBER_RF4CE_APP_CAPABILITIES_SUPPORTED_PROFILES_MASK

```

```

        = 0x70,
00219 EMBER_RF4CE_APP_CAPABILITIES_SUPPORTED_PROFILES_OFFSET
        = 4,
00220 // Bit 7 is reserved.
00221 };
00223
00227 typedef struct {
00232     uint16_t vendorId;
00236     uint8_t vendorString[EMBER_RF4CE_VENDOR_STRING_LENGTH
    ];
00237 } EmberRf4ceVendorInfo;
00238
00242 typedef struct {
00247     EmberRf4ceApplicationCapabilities
        capabilities;
00252     uint8_t userString[EMBER_RF4CE_APPLICATION_USER_STRING_LENGTH
    ];
00257     uint8_t deviceTypeList[EMBER_RF4CE_APPLICATION_DEVICE_TYPE_LIST_MAX_LENGTH
    ];
00262     uint8_t profileIdList[EMBER_RF4CE_APPLICATION_PROFILE_ID_LIST_MAX_LENGTH
    ];
00263 } EmberRf4ceApplicationInfo;
00264
00268 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00269 enum EmberRf4cePairingEntryStatus
00270 #else
00271 typedef uint8_t EmberRf4cePairingEntryStatus;
00272 enum
00273 #endif
00274 {
00278     EMBER_RF4CE_PAIRING_TABLE_ENTRY_STATUS_UNUSED
        = 0x00,
00283     EMBER_RF4CE_PAIRING_TABLE_ENTRY_STATUS_PROVISIONAL
        = 0x01,
00287     EMBER_RF4CE_PAIRING_TABLE_ENTRY_STATUS_ACTIVE
        = 0x02
00288 };
00289
00293 #define EMBER_RF4CE_PAIRING_TABLE_ENTRY_INFO_STATUS_MASK
0x03
00294
00299 #define EMBER_RF4CE_PAIRING_TABLE_ENTRY_INFO_HAS_LINK_KEY_BIT
0x04
00300
00305 #define EMBER_RF4CE_PAIRING_TABLE_ENTRY_INFO_IS_PAIRING_INITIATOR_BIT
0x08
00306
00309 typedef struct {
00313     EmberKeyData securityLinkKey;
00317     EmberEUI64 destLongId;
00321     uint32_t frameCounter;
00325     EmberNodeId sourceNodeId;
00329     EmberPanId destPanId;
00333     EmberNodeId destNodeId;
00337     uint16_t destVendorId;
00341     uint8_t destProfileIdList[EMBER_RF4CE_APPLICATION_PROFILE_ID_LIST_MAX_LENGTH
    ];
00345     uint8_t destProfileIdListLength;
00349     uint8_t info;
00353     uint8_t channel;
00357     uint8_t capabilities;
00361     uint8_t lastSeqn;
00362 } EmberRf4cePairingTableEntry;
00363
00365
00366 #endif // __RF4CE_TYPES_H__
00367

```

8.127 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)
- bool [emberHaveLinkKey](#) ([EmberEUI64](#) remoteDevice)
- EmberStatus [emberGenerateRandomKey](#) ([EmberKeyData](#) *keyAddress)
- void [emberSwitchNetworkKeyHandler](#) (uint8_t sequenceNumber)
- EmberStatus [emberRequestLinkKey](#) ([EmberEUI64](#) partner)
- void [emberZigbeeKeyEstablishmentHandler](#) ([EmberEUI64](#) partner, [EmberKeyStatus](#) status)
- EmberStatus [emberGetKeyTableEntry](#) (uint8_t index, [EmberKeyStruct](#) *result)
- EmberStatus [emberSetKeyTableEntry](#) (uint8_t index, [EmberEUI64](#) address, bool linkKey, [EmberKeyData](#) *keyData)
- EmberStatus [emberAddOrUpdateKeyTableEntry](#) ([EmberEUI64](#) address, bool linkKey, [EmberKeyData](#) *keyData)
- uint8_t [emberFindKeyTableEntry](#) ([EmberEUI64](#) address, bool linkKey)
- EmberStatus [emberEraseKeyTableEntry](#) (uint8_t index)
- EmberStatus [emberClearKeyTable](#) (void)
- EmberStatus [emberStopWritingStackTokens](#) (void)
- bool [emberWritingStackTokensEnabled](#) (void)
- EmberStatus [emberApsCryptMessage](#) (bool encrypt, [EmberMessageBuffer](#) buffer, uint8_t apsHeaderEndIndex, [EmberEUI64](#) remoteEui64)
- EmberStatus [emberGetMfgSecurityConfig](#) ([EmberMfgSecurityStruct](#) *settings)
- EmberStatus [emberSetMfgSecurityConfig](#) (uint32_t magicNumber, const [EmberMfgSecurityStruct](#) *settings)
- EmberStatus [emberSetOutgoingNwkFrameCounter](#) (uint32_t desiredValue)
- EmberStatus [emberSetOutgoingApsFrameCounter](#) (uint32_t desiredValue)
- EmberStatus [emberAddTransientLinkKey](#) ([EmberEUI64](#) partnerEUI64, [EmberKeyData](#) *key)
- void [emberClearTransientLinkKeys](#) (void)

Variables

- uint16_t [emberTransientKeyTimeoutS](#)

8.127.1 Detailed Description

EmberZNet security API. See [Security](#) for documentation.

Definition in file [security.h](#).

8.128 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,
                           EmberKeyStruct* keyStruct);
00143
00144
00151 bool emberHaveLinkKey(EmberEUI64 remoteDevice);
00152
00153 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00154 extern uint8_t emberKeyTableSize;
00155 extern uint32_t* emIncomingApsFrameCounters;
00156 #endif
00157
00171 EmberStatus emberGenerateRandomKey(
    EmberKeyData* keyAddress);
00172
00180 void emberSwitchNetworkKeyHandler(uint8_t
    sequenceNumber);
00181
00211 EmberStatus emberRequestLinkKey(EmberEUI64
    partner);
00212
00222 void emberZigbeeKeyEstablishmentHandler(
    EmberEUI64 partner, EmberKeyStatus status);
00223
00224
00241 EmberStatus emberGetKeyTableEntry(uint8_t index
    ,
    EmberKeyStruct* result);
00243
00255 #if defined DOXYGEN_SHOULD_SKIP_THIS
00256 EmberStatus emberSetKeyTableEntry(uint8_t index
    ,
    EmberEUI64 address,
    bool linkKey,
    EmberKeyData* keyData);
00258
00259
00260 #else
00261 EmberStatus emSetKeyTableEntry(bool erase,
00262             uint8_t index,
00263             EmberEUI64 address,
00264             bool 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,
     bool linkKey,
     EmberKeyData* keyData);
00293
00294
00295
00307 uint8_t emberFindKeyTableEntry(EmberEUI64
    address,

```

```

00308                     bool linkKey);
00309
00318 #if defined DOXYGEN_SHOULD_SKIP_THIS
00319 EmberStatus emberEraseKeyTableEntry(uint8_t
00320 index);
00320 #else
00321 #define emberEraseKeyTableEntry(index) \
00322     emSetKeyTableEntry(true, (index), NULL, false, NULL)
00323 #endif
00324
00325
00326 EmberStatus emberClearKeyTable(void);
00327
00328 EmberStatus emberStopWritingStackTokens(
00329 void);
00330
00331 EmberStatus emberStartWritingStackTokens
00332 (void);
00333
00334 bool emberWritingStackTokensEnabled(void);
00335
00336 EmberStatus emberApsCryptMessage(bool encrypt,
00337                                     EmberMessageBuffer buffer,
00338                                     uint8_t apsHeaderEndIndex,
00339                                     EmberEUI64 remoteEui64);
00340
00341
00342 EmberStatus emberGetMfgSecurityConfig(
00343     EmberMfgSecurityStruct* settings);
00344
00345
00346 EmberStatus emberSetMfgSecurityConfig(
00347     uint32_t magicNumber,
00348                 const EmberMfgSecurityStruct
00349 * settings);
00350
00351 #if defined DOXYGEN_SHOULD_SKIP_THIS
00352 EmberStatus emberSetOutgoingNwkFrameCounter
00353 (uint32_t desiredValue);
00354 #else
00355     #define emberSetOutgoingNwkFrameCounter(desiredValue) \
00356         emSetFrameCounter(EMBER_CURRENT_NETWORK_KEY, desiredValue)
00357 #endif
00358
00359 #if defined DOXYGEN_SHOULD_SKIP_THIS
00360 EmberStatus emberSetOutgoingApsFrameCounter
00361 (uint32_t desiredValue);
00362 #else
00363     #define emberSetOutgoingApsFrameCounter(desiredValue) \
00364         emSetFrameCounter(EMBER_TRUST_CENTER_LINK_KEY, desiredValue)
00365 #endif
00366
00367 EmberStatus emSetFrameCounter(EmberKeyType keyType,
00368
00369     uint32_t desiredValue);
00370 #endif
00371
00372 EmberStatus emberAddTransientLinkKey(
00373     EmberEUI64 partnerEUI64, EmberKeyData *key);
00374
00375 void emberClearTransientLinkKeys(void);
00376
00377 extern uint16_t emberTransientKeyTimeouts;
00378
00379 // @} END addtogroup

```

8.129 serial.h File Reference

```
#include "stack/include/ember-types.h"
```

Enumerations

- enum `SerialBaudRate` {
 `DEFINE_BAUD`, `DEFINE_BAUD`, `DEFINE_BAUD`, `DEFINE_BAUD`,
 `DEFINE_BAUD`, `DEFINE_BAUD`, `DEFINE_BAUD`, `DEFINE_BAUD`,
 `DEFINE_BAUD`, `DEFINE_BAUD`, `DEFINE_BAUD`, `DEFINE_BAUD`,
 `DEFINE_BAUD`, `DEFINE_BAUD`, `DEFINE_BAUD`, `DEFINE_BAUD`,
 `DEFINE_BAUD`, `DEFINE_BAUD`, `DEFINE_BAUD`
}
- enum `SerialParity` { `DEFINE_PARITY`, `DEFINE_PARITY`, `DEFINE_PARITY` }

Functions

- void `halHostFlushBuffers` (void)
- uint16_t `halHostEnqueueTx` (const uint8_t *data, uint16_t length)
- void `halHostFlushTx` (void)
- uint16_t `serialCopyFromRx` (const uint8_t *data, uint16_t length)
- void `emLoadSerialTx` (void)

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)
- #define `halInternalUart1FlowControlRxIsEnabled()`
- #define `halInternalUart1XonRefreshDone()`
- #define `halInternalUart1TxIsIdle()`
- EmberStatus `halInternalUartInit` (uint8_t port, `SerialBaudRate` rate, `SerialParity` parity, uint8_t stopBits)

- void [halInternalPowerDownUart](#) (void)
- void [halInternalPowerUpUart](#) (void)
- void [halInternalStartUartTx](#) (uint8_t port)
- void [halInternalStopUartTx](#) (uint8_t port)
- EmberStatus [halInternalForceWriteUartData](#) (uint8_t port, uint8_t *data, uint8_t length)
- EmberStatus [halInternalForceReadUartByte](#) (uint8_t port, uint8_t *dataByte)
- void [halInternalWaitUartTxComplete](#) (uint8_t port)
- void [halInternalRestartUart](#) (void)
- bool [halInternalUartFlowControlRxEnabled](#) (uint8_t port)
- bool [halInternalUartXonRefreshDone](#) (uint8_t port)
- bool [halInternalUartTxIsIdle](#) (uint8_t port)
- bool [serialDropPacket](#) (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, uint8_t port)

Virtual UART API

API used by the stack in debug builds to receive data arriving over the virtual UART.

- void [halStackReceiveVuartMessage](#) (uint8_t *data, uint8_t length)

8.129.1 Detailed Description

Serial hardware abstraction layer interfaces. See [Serial UART Communication](#) for documentation.

Definition in file [hal/micro/serial.h](#).

8.130 hal/micro/serial.h

```

00001
00023 #ifndef __HAL_SERIAL_H__
00024 #define __HAL_SERIAL_H__
00025
00026 //[[[
00027 // Include ember.h to get EmberMessageBuffer definition.
00028 // Why is this necessary for ZIP but not for ZNet? In fact, including it in
00029 // ZNet breaks things. I can't figure out how to untangle the
00030 // dependencies, but I'm thinking this is not the right place to include
00031 // this header. Since it's breaking things for other people, I'll just
00032 // apply the band aid and worry about it later. -- ryan
00033 //]]
00034 // If only EmberMessageBuffer is needed, ember-types.h should suffice.
00035 // Some znet targets also now need this, so I'm removing the restriction below.
00036 // -- Vignesh.
00037 // #if defined(EMBER_STACK_IP) || defined(EMBER_STACK_CONNECT) ||
00038 //     defined(EMBER_STACK_WASP)
00039 #include "stack/include/ember-types.h"

```

```

00039 // #endif
00040
00041 #ifdef CORTEXM3_EFM32_MICRO
00042     #include "em_usart.h"
00043 #endif
00044
00045 //[[[
00046 // Crude method of mashing together com and serial layers before all the
00047 // drivers
00048 //]]]
00049 #ifndef CORTEXM3_EFM32_MICRO
00050 // EM_NUM_SERIAL_PORTS is inherited from the micro specific micro.h
00051 #if (EM_NUM_SERIAL_PORTS == 1)
00052     #define FOR_EACH_PORT(cast,prefix,suffix) \
00053         cast(prefix##0##suffix)
00054 #elif (EM_NUM_SERIAL_PORTS == 2)
00055     #define FOR_EACH_PORT(cast,prefix,suffix) \
00056         cast(prefix##0##suffix), \
00057         cast(prefix##1##suffix)
00058 #elif (EM_NUM_SERIAL_PORTS == 3)
00059     #define FOR_EACH_PORT(cast,prefix,suffix) \
00060         cast(prefix##0##suffix), \
00061         cast(prefix##1##suffix), \
00062         cast(prefix##2##suffix)
00063 #elif (EM_NUM_SERIAL_PORTS == 4)
00064     #define FOR_EACH_PORT(cast,prefix,suffix) \
00065         cast(prefix##0##suffix), \
00066         cast(prefix##1##suffix), \
00067         cast(prefix##2##suffix), \
00068         cast(prefix##3##suffix),
00069 #elif (EM_NUM_SERIAL_PORTS == 5)
00070     #define FOR_EACH_PORT(cast,prefix,suffix) \
00071         cast(prefix##0##suffix), \
00072         cast(prefix##1##suffix), \
00073         cast(prefix##2##suffix), \
00074         cast(prefix##3##suffix), \
00075         cast(prefix##4##suffix),
00076 #else
00077     #error unsupported number of serial ports
00078 #endif
00079 #endif //CORTEXM3_EFM32_MICRO
00080
00093 #define EMBER_SERIAL_UNUSED 0
00094 #define EMBER_SERIAL_FIFO 1
00095 #define EMBER_SERIAL_BUFFER 2
00096 #define EMBER_SERIAL_LOWLEVEL 3
00097
00099 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00100
00101 // The following tests for setting of an invalid mode
00102 #ifdef EMBER_SERIAL0_MODE
00103 #if (EMBER_SERIAL0_MODE != EMBER_SERIAL_FIFO) && \
00104     (EMBER_SERIAL0_MODE != EMBER_SERIAL_BUFFER) && \
00105     (EMBER_SERIAL0_MODE != EMBER_SERIAL_LOWLEVEL) && \
00106     (EMBER_SERIAL0_MODE != EMBER_SERIAL_UNUSED)
00107     #error Invalid Serial 0 Mode
00108 #endif
00109 #else
00110     #define EMBER_SERIAL0_MODE EMBER_SERIAL_UNUSED
00111 #endif
00112 #ifdef EMBER_SERIAL1_MODE
00113 #if (EMBER_SERIAL1_MODE != EMBER_SERIAL_FIFO) && \
00114     (EMBER_SERIAL1_MODE != EMBER_SERIAL_BUFFER) && \
00115     (EMBER_SERIAL1_MODE != EMBER_SERIAL_LOWLEVEL) && \
00116     (EMBER_SERIAL1_MODE != EMBER_SERIAL_UNUSED)
00117     #error Invalid Serial 1 Mode
00118 #endif
00119 #else
00120     #define EMBER_SERIAL1_MODE EMBER_SERIAL_UNUSED
00121 #endif
00122 #ifdef EMBER_SERIAL2_MODE
00123 #if (EMBER_SERIAL2_MODE != EMBER_SERIAL_FIFO) && \
00124     (EMBER_SERIAL2_MODE != EMBER_SERIAL_BUFFER) && \
00125     (EMBER_SERIAL2_MODE != EMBER_SERIAL_LOWLEVEL) && \
00126     (EMBER_SERIAL2_MODE != EMBER_SERIAL_UNUSED)
00127     #error Invalid Serial 2 Mode
00128 #endif
00129 #else
00130     #define EMBER_SERIAL2_MODE EMBER_SERIAL_UNUSED

```

```

00131 #endif
00132 #ifdef EMBER_SERIAL3_MODE
00133 #if (EMBER_SERIAL3_MODE != EMBER_SERIAL_FIFO) && \
00134     (EMBER_SERIAL3_MODE != EMBER_SERIAL_BUFFER) && \
00135     (EMBER_SERIAL3_MODE != EMBER_SERIAL_LOWLEVEL) && \
00136     (EMBER_SERIAL3_MODE != EMBER_SERIAL_UNUSED)
00137     #error Invalid Serial 3 Mode
00138 #endif
00139 #else
00140     #define EMBER_SERIAL3_MODE EMBER_SERIAL_UNUSED
00141 #endif
00142 #ifdef EMBER_SERIAL4_MODE
00143 #if (EMBER_SERIAL4_MODE != EMBER_SERIAL_FIFO) && \
00144     (EMBER_SERIAL4_MODE != EMBER_SERIAL_BUFFER) && \
00145     (EMBER_SERIAL4_MODE != EMBER_SERIAL_LOWLEVEL) && \
00146     (EMBER_SERIAL4_MODE != EMBER_SERIAL_UNUSED)
00147     #error Invalid Serial 4 Mode
00148 #endif
00149 #else
00150     #define EMBER_SERIAL4_MODE EMBER_SERIAL_UNUSED
00151 #endif
00152
00153 // Determine if FIFO and/or Buffer modes are being used, so those sections of
00154 // code may be disabled if not
00155 #if (EMBER_SERIAL0_MODE == EMBER_SERIAL_FIFO) || \
00156     (EMBER_SERIAL1_MODE == EMBER_SERIAL_FIFO) || \
00157     (EMBER_SERIAL2_MODE == EMBER_SERIAL_FIFO) || \
00158     (EMBER_SERIAL3_MODE == EMBER_SERIAL_FIFO) || \
00159     (EMBER_SERIAL4_MODE == EMBER_SERIAL_FIFO)
00160     #define EM_ENABLE_SERIAL_FIFO
00161 #endif
00162 #if (EMBER_SERIAL0_MODE == EMBER_SERIAL_BUFFER) || \
00163     (EMBER_SERIAL1_MODE == EMBER_SERIAL_BUFFER) || \
00164     (EMBER_SERIAL2_MODE == EMBER_SERIAL_BUFFER) || \
00165     (EMBER_SERIAL3_MODE == EMBER_SERIAL_BUFFER) || \
00166     (EMBER_SERIAL4_MODE == EMBER_SERIAL_BUFFER)
00167     #define EM_ENABLE_SERIAL_BUFFER
00168 #endif
00169
00177 typedef struct {
00179     uint16_t head;
00181     uint16_t tail;
00183     volatile uint16_t used;
00185     uint8_t fifo[];
00186 } EmSerialFifoQueue;
00187
00188
00192 typedef struct {
00194     uint8_t length;
00196     EmberMessageBuffer buffer;
00198     uint8_t startIndex;
00199 } EmSerialBufferQueueEntry;
00200
00204 typedef struct {
00206     uint8_t head;
00208     uint8_t tail;
00210     volatile uint8_t used;
00212     volatile uint8_t dead;
00215     EmberMessageBuffer currentBuffer;
00217     uint8_t *nextByte;
00219     uint8_t *lastByte;
00221     EmSerialBufferQueueEntry fifo[];
00222 } EmSerialBufferQueue;
00223
00232 #ifndef CORTEXM3_EFM32_MICRO
00233
00234 extern void *emSerialTxQueues[EM_NUM_SERIAL_PORTS];
00236 extern EmSerialFifoQueue *emSerialRxQueues[EM_NUM_SERIAL_PORTS];
00238 extern uint16_t PGM emSerialTxQueueMasks[EM_NUM_SERIAL_PORTS];
00240 extern uint16_t PGM emSerialTxQueueSizes[EM_NUM_SERIAL_PORTS];
00242 extern uint16_t PGM emSerialRxQueueSizes[EM_NUM_SERIAL_PORTS];
00244 extern uint8_t emSerialRxError[EM_NUM_SERIAL_PORTS];
00246 extern uint16_t emSerialRxErrorHandlerIndex[EM_NUM_SERIAL_PORTS];
00248 extern uint8_t PGM emSerialPortModes[EM_NUM_SERIAL_PORTS];
00249
00250 extern uint16_t PGM emSerialTxQueueWraps[EM_NUM_SERIAL_PORTS];
00251 extern uint16_t PGM emSerialRxQueueWraps[EM_NUM_SERIAL_PORTS];
00252
00253 #endif // CORTEXM3_EFM32_MICRO
00254

```

```

00256 #ifdef EZSP_UART
00257     void emCallCounterHandler(EmberCounterType type, uint8_t data
00258     );
00259     #define HANDLE_ASH_ERROR(type) emCallCounterHandler(type, 0)
00260     #else
00261     #define HANDLE_ASH_ERROR(type)
00262     #endif
00263 #endif //DOXYGEN_SHOULD_SKIP_THIS
00264
00265
00282 #undef FIFO_ENQUEUE // Avoid possible warning, replace other definition
00283 #define FIFO_ENQUEUE(queue,data,size) \
00284     do { \
00285         (queue)->fifo[(queue)->head] = (data); \
00286         (queue)->head = (((queue)->head + 1) % (size)); \
00287         (queue)->used++; \
00288     } while(0)
00289
00297 #undef FIFO_DEQUEUE // Avoid possible warning, replace other definition
00298 #define FIFO_DEQUEUE(queue,size) \
00299     (queue)->fifo[(queue)->tail]; \
00300     (queue)->tail = (((queue)->tail + 1) % (size)); \
00301     (queue)->used--
00302
00306 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00307
00311 enum SerialBaudRate
00312 #else
00313     #ifndef DEFINE_BAUD
00314     #define DEFINE_BAUD(num) BAUD_##num
00315     #endif
00316     #ifdef CORTEXM3_EFM32_MICRO
00317         typedef uint32_t SerialBaudRate;
00318     #else
00319         typedef uint8_t SerialBaudRate;
00320     #endif
00321 enum
00322 #endif //DOXYGEN_SHOULD_SKIP_THIS
00323 #ifdef CORTEXM3_EFM32_MICRO
00324 {
00325     DEFINE_BAUD(300) = 300, // BAUD_300
00326     DEFINE_BAUD(600) = 600, // BAUD_600
00327     DEFINE_BAUD(900) = 900, // etc...
00328     DEFINE_BAUD(1200) = 1200,
00329     DEFINE_BAUD(2400) = 2400,
00330     DEFINE_BAUD(4800) = 4800,
00331     DEFINE_BAUD(9600) = 9600,
00332     DEFINE_BAUD(14400) = 14400,
00333     DEFINE_BAUD(19200) = 19200,
00334     DEFINE_BAUD(28800) = 28800,
00335     DEFINE_BAUD(38400) = 38400,
00336     DEFINE_BAUD(50000) = 50000,
00337     DEFINE_BAUD(57600) = 57600,
00338     DEFINE_BAUD(76800) = 76800,
00339     DEFINE_BAUD(100000) = 100000,
00340     DEFINE_BAUD(115200) = 115200
00341 };
00342 #else //CORTEXM3_EFM32_MICRO
00343 {
00344     DEFINE_BAUD(300) = 0, // BAUD_300
00345     DEFINE_BAUD(600) = 1, // BAUD_600
00346     DEFINE_BAUD(900) = 2, // etc...
00347     DEFINE_BAUD(1200) = 3,
00348     DEFINE_BAUD(2400) = 4,
00349     DEFINE_BAUD(4800) = 5,
00350     DEFINE_BAUD(9600) = 6,
00351     DEFINE_BAUD(14400) = 7,
00352     DEFINE_BAUD(19200) = 8,
00353     DEFINE_BAUD(28800) = 9,
00354     DEFINE_BAUD(38400) = 10,
00355     DEFINE_BAUD(50000) = 11,
00356     DEFINE_BAUD(57600) = 12,
00357     DEFINE_BAUD(76800) = 13,
00358     DEFINE_BAUD(100000) = 14,
00359     DEFINE_BAUD(115200) = 15,
00360     DEFINE_BAUD(230400) = 16, /*<! define higher baud rates for the
EM2XX and EM3XX */
00361     DEFINE_BAUD(460800) = 17, /*<! Note: receiving data at baud
rates > 115200 */

```

```

00362     DEFINE_BAUD(CUSTOM) = 18 /*<! may not be reliable due to
00363     interrupt latency */
00364 #endif //CORTEXM3_EFM32_MICRO
00365
00366
00367 #ifdef CORTEXM3_EFM32_MICRO
00368
00369     typedef USART_Parity_TypeDef SerialParity;
00370     #define PARITY_NONE usartNoParity
00371     #define PARITY_ODD usartOddParity
00372     #define PARITY EVEN usartEvenParity
00373
00374 #else
00375
00376 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00377
00378     enum SerialParity
00379     #else
00380         #ifndef DEFINE_PARITY
00381             #define DEFINE_PARITY(val) PARITY_##val
00382         #endif
00383         typedef uint8_t SerialParity;
00384     enum
00385     #endif //DOXYGEN_SHOULD_SKIP_THIS
00386
00387     {
00388         DEFINE_PARITY(NONE) = 0, // PARITY_NONE
00389         DEFINE_PARITY(ODD) = 1, // PARITY_ODD
00390         DEFINE_PARITY(EVEN) = 2 // PARITY_EVEN
00391     };
00392
00393 #endif//CORTEXM3_EFM32_MICRO
00394
00395     EmberStatus halInternalUartInit(uint8_t port,
00396                                         SerialBaudRate rate,
00397                                         SerialParity parity,
00398                                         uint8_t stopBits);
00399
00400
00401     void halInternalPowerDownUart(void);
00402
00403     void halInternalPowerUpUart(void);
00404
00405     void halInternalStartUartTx(uint8_t port);
00406
00407
00408     void halInternalStopUartTx(uint8_t port);
00409
00410
00411     EmberStatus halInternalForceWriteUartData
00412         (uint8_t port, uint8_t *data, uint8_t length);
00413
00414     EmberStatus halInternalForceReadUartByte
00415         (uint8_t port, uint8_t *dataByte);
00416
00417
00418     void halInternalWaitUartTxComplete(uint8_t port);
00419
00420
00421     #if (EMBER_SERIAL1_MODE == EMBER_SERIAL_FIFO) &&
00422         ( defined(EMBER_SERIAL1_XONXOFF) ||
00423         (defined(XAP2B) && defined(EMBER_SERIAL1_RTSCTS) ) )
00424         void halInternalUartFlowControl(uint8_t port);
00425
00426     #else
00427         #define halInternalUartFlowControl(port) do {} while(false)
00428     #endif
00429
00430
00431     #if (defined(XAP2B)&&(EMBER_SERIAL1_MODE == EMBER_SERIAL_BUFFER)) ||
00432         defined(CORTEXM3)
00433         void halInternalUartRxPump(uint8_t port);
00434     #else
00435         #define halInternalUartRxPump(port) do {} while(false)
00436     #endif
00437
00438
00439     void halInternalRestartUart(void);
00440
00441
00442     bool halInternalUartFlowControlRxIsEnabled
00443         (uint8_t port);
00444     // define the old name for backwards compatibility
00445     #define halInternalUart1FlowControlRxIsEnabled()
00446         halInternalUartFlowControlRxIsEnabled(1)
00447
00448
00449     bool halInternalUartXonRefreshDone(uint8_t port);

```

```

00535 #define halInternalUart1XonRefreshDone() halInternalUartXonRefreshDone(1)
00536
00541 #ifdef CORTEXM3_EFM32_MICRO
00542     #define halInternalUartTxIsIdle(port) COM_InternalTxIsIdle(port)
00543 #else
00544     bool halInternalUartTxIsIdle(uint8_t port);
00545 #endif
00546 // define the old name for backwards compatibility
00547 #define halInternalUart1TxIsIdle() halInternalUartTxIsIdle(1)
00548
00553 bool serialDropPacket(void);
00554
00569 void emSerialBufferNextMessageIsr(
    EmSerialBufferQueue *q);
00570
00580 void emSerialBufferNextBlockIsr(EmSerialBufferQueue *
    q, uint8_t port);
00581
00598 #if defined(CORTEXM3_EFM32_MICRO)
00599 #define halStackReceiveVuartMessage(data, length)
00600 #else
00601 void halStackReceiveVuartMessage(uint8_t* data,
    uint8_t length);
00602 #endif
00603
00606 void halHostFlushBuffers(void);
00607 uint16_t halHostEnqueueTx(const uint8_t *data, uint16_t length)
    ;
00608 void halHostFlushTx(void);
00609
00610 // 'data' points to the next 'length' bytes of input.
00611 uint16_t serialCopyFromRx(const uint8_t *data, uint16_t length)
    ;
00612
00613 // tell the upper layer to load serial data
00614 void emLoadSerialTx(void);
00615
00616 #endif //__HAL_SERIAL_H__
00617

```

8.131 serial.h File Reference

Functions

- EmberStatus emberSerialInit (uint8_t port, SerialBaudRate rate, SerialParity parity, uint8_t stopBits)
- uint16_t emberSerialReadAvailable (uint8_t port)
- EmberStatus emberSerial.ReadByte (uint8_t port, uint8_t *dataByte)
- EmberStatus emberSerialReadData (uint8_t port, uint8_t *data, uint16_t length, uint16_t *bytesRead)
- EmberStatus emberSerialReadDataTimeout (uint8_t port, uint8_t *data, uint16_t length, uint16_t *bytesRead, uint16_t firstByteTimeout, uint16_t subsequentByteTimeout)
- EmberStatus emberSerialReadLine (uint8_t port, char *data, uint8_t max)
- EmberStatus emberSerialReadPartialLine (uint8_t port, char *data, uint8_t max, uint8_t *index)
- uint16_t emberSerialWriteAvailable (uint8_t port)
- uint16_t emberSerialWriteUsed (uint8_t port)
- EmberStatus emberSerialWriteByte (uint8_t port, uint8_t dataByte)
- EmberStatus emberSerialWriteHex (uint8_t port, uint8_t dataByte)
- EmberStatus emberSerialWriteString (uint8_t port, PGM_P string)
- XAP2B_PAGEZERO_ON EmberStatus emberSerialPrintf (uint8_t port, PGM_P formatString,...)

- XAP2B_PAGEZERO_OFF
XAP2B_PAGEZERO_ON EmberStatus emberSerialPrintfLine (uint8_t port, PGM_P formatString,...)
- XAP2B_PAGEZERO_OFF
XAP2B_PAGEZERO_ON EmberStatus emberSerialPrintCarriageReturn (uint8_t port)
- XAP2B_PAGEZERO_OFF EmberStatus emberSerialPrintVarArg (uint8_t port, P-GM_P formatString, va_list ap)
- EmberStatus emberSerialWriteData (uint8_t port, uint8_t *data, uint8_t length)
- EmberStatus emberSerialWriteBuffer (uint8_t port, EmberMessageBuffer buffer, uint8_t start, uint8_t length)
- XAP2B_PAGEZERO_ON EmberStatus emberSerialWaitSend (uint8_t port)
- XAP2B_PAGEZERO_OFF EmberStatus emberSerialGuaranteedPrintf (uint8_t port, PGM_P formatString,...)
- void emberSerialBufferTick (void)
- void emberSerialFlushRx (uint8_t port)
- bool emberSerialUnused (uint8_t port)

8.131.1 Detailed Description

High-level serial communication functions. See [Serial Communication](#) for documentation.

Definition in file [app/util/serial/serial.h](#).

8.132 app/util/serial/serial.h

```

00001
00012 #ifndef __SERIAL_H__
00013 #define __SERIAL_H__
00014
00015 #ifndef __HAL_H__
00016     #error hal/hal.h should be included first
00017 #endif
00018
00019 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00020 #include <stdarg.h>
00021
00022 //Rx FIFO Full indicator
00023 #define RX_FIFO_FULL (0xFFFF)
00024
00025 #endif // DOXYGEN_SHOULD_SKIP_THIS
00026
00136 EmberStatus emberSerialInit(uint8_t port,
00137                             SerialBaudRate rate,
00138                             SerialParity parity,
00139                             uint8_t stopBits);
00140
00148 uint16_t emberSerialReadAvailable(uint8_t port);
00149
00167 EmberStatus emberSerial.ReadByte(uint8_t port,
00168                                     uint8_t *dataByte);
00169
00193 EmberStatus emberSerialReadData(uint8_t port,
00194                                     uint8_t *data,
00195                                     uint16_t length,
00196                                     uint16_t *bytesRead);
00197
00232 EmberStatus emberSerialReadDataTimeout(
00233     uint8_t port,
00234                                     uint8_t *data,
00235                                     uint16_t length,
00236                                     uint16_t *bytesRead,
00237                                     uint16_t firstByteTimeout,
00238                                     uint16_t subsequentByteTimeout);

```

```

00253 EmberStatus emberSerialReadLine(uint8_t port,
00254     char *data, uint8_t max);
00258 EmberStatus emberSerialReadPartialLine(
00259     uint8_t port, char *data, uint8_t max, uint8_t *index);
00279 uint16_t emberSerialWriteAvailable(uint8_t port);
00289 uint16_t emberSerialWriteUsed(uint8_t port);
00298
00312 EmberStatus emberSerialWriteByte(uint8_t port,
00313     uint8_t dataByte);
00313
00328 EmberStatus emberSerialWriteHex(uint8_t port,
00329     uint8_t dataByte);
00329
00342 EmberStatus emberSerialWriteString(uint8_t
00343     port, PGM_P string);
00343
00368 XAP2B_PAGEZERO_ON
00369 EmberStatus emberSerialPrintf(uint8_t port, PGM_P
00370     formatString, ...);
00370 XAP2B_PAGEZERO_OFF
00371
00387 XAP2B_PAGEZERO_ON
00388 EmberStatus emberSerialPrintfLine(uint8_t port,
00389     PGM_P formatString, ...);
00389 XAP2B_PAGEZERO_OFF
00390
00401 XAP2B_PAGEZERO_ON
00402 EmberStatus emberSerialPrintCarriageReturn
00403     (uint8_t port);
00403 XAP2B_PAGEZERO_OFF
00404
00405
00418 EmberStatus emberSerialPrintfVarArg(uint8_t
00419     port, PGM_P formatString, va_list ap);
00419
00435 EmberStatus emberSerialWriteData(uint8_t port,
00436     uint8_t *data, uint8_t length);
00436
00437 //Host HALs do not use stack buffers.
00438 #ifndef HAL_HOST
00439
00457 EmberStatus emberSerialWriteBuffer(uint8_t
00458     port, EmberMessageBuffer buffer, uint8_t start, uint8_t length);
00459 #endif //HAL_HOST
00459
00472 XAP2B_PAGEZERO_ON
00473 EmberStatus emberSerialWaitSend(uint8_t port);
00474 XAP2B_PAGEZERO_OFF
00475
00496 EmberStatus emberSerialGuaranteedPrintf(
00497     uint8_t port, PGM_P formatString, ...);
00497
00503 void emberSerialBufferTick(void);
00504
00510 void emberSerialFlushRx(uint8_t port);
00511
00518 bool emberSerialUnused(uint8_t port);
00519
00520
00521
00522
00523
00526 #endif // __SERIAL_H__
00527

```

8.133 sim-eeprom.h File Reference

Functions

- void **halSimEepromCallback** (**EmberStatus** status)
- uint8_t **halSimEepromErasePage** (void)

- void [halSimEepromStatus](#) (uint16_t *freeWordsUntilFull, uint16_t *totalPageUseCount)

8.133.1 Detailed Description

Simulated EEPROM system for wear leveling token storage across flash. See [Simulated EEPROM](#) for documentation.

Definition in file [sim-eeprom.h](#).

8.134 sim-eeprom.h

```

00001
00007 #ifndef __SIM_EEPROM_H__
00008 #define __SIM_EEPROM_H__
00009
00010
00011 //pull in the platform specific information here
00012 #if defined(CORTEXM3) || defined(EMBER_TEST)
00013     #include "cortexm3/sim-eeprom.h"
00014 #elif defined (C8051_COBRA)
00015     #include "c8051/cobra/sim-eeprom.h"
00016 #else
00017     #error invalid sim-eeprom platform
00018 #endif
00019
00064 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00065 //Prototype for internal Startup for the public Init and Repair macros
00066 EmberStatus halInternalSimEeStartup(bool forceRebuildAll);
00067
00068 //Properly round values when converting bytes to words
00069 #define BYTES_TO_WORDS(x) (((x) + 1) / 2)
00070
00071 #endif
00072
00073
00074 //application functions
00075
00123 void halSimEepromCallback(EmberStatus status);
00124
00151 uint8_t halSimEepromErasePage(void);
00152
00173 void halSimEepromStatus(uint16_t *freeWordsUntilFull,
00174     uint16_t *totalPageUseCount);
00175
00176 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00177
00178 //NOTE: halInternal functions must not be called directly.
00179
00180
00205 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00206 EmberStatus halInternalSimEeInit(void);
00207 #else
00208 #define halInternalSimEeInit() (halInternalSimEeStartup(false))
00209 #endif
00210
00238 void halInternalSimEeGetData(void *vdata,
00239             uint8_t compileId,
00240             uint8_t index,
00241             uint8_t len);
00242
00283 void halInternalSimEeSetData(uint8_t compileId,
00284             void *vdata,
00285             uint8_t index,
00286             uint8_t len);
00287
00313 void halInternalSimEeIncrementCounter(uint8_t compileId);
00314
00345 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00346 EmberStatus halInternalSimEeRepair(bool forceRebuildAll);
00347 #else

```

```

00348 #define halInternalSimEeRepair(rebuild) (halInternalSimEeStartup(rebuild))
00349 #endif
00350
00351 #endif //DOXYGEN_SHOULD_SKIP_THIS
00352
00353
00354 #endif //__SIM_EEPROM_H__
00355

```

8.135 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)
- bool [emberStackIsUp](#) (void)
- [EmberEUI64](#) [emberGetEui64](#) (void)
- bool [emberIsLocalEui64](#) ([EmberEUI64](#) eui64)
- [EmberNodeId](#) [emberGetNodeId](#) (void)
- [EmberNodeId](#) [emberRadioGetNodeId](#) (void)
- void [emberSetManufacturerCode](#) (uint16_t code)
- void [emberSetPowerDescriptor](#) (uint16_t descriptor)
- void [emberSetMaximumIncomingTransferSize](#) (uint16_t size)
- void [emberSetMaximumOutgoingTransferSize](#) (uint16_t size)
- void [emberSetDescriptorCapability](#) (uint8_t capability)
- [EmberStatus](#) [emberGetNetworkParameters](#) ([EmberNetworkParameters](#) *parameters)
- [EmberStatus](#) [emberGetNodeType](#) ([EmberNodeType](#) *resultLocation)
- [EmberStatus](#) [emberSetRadioChannel](#) (uint8_t channel)
- uint8_t [emberGetRadioChannel](#) (void)
- [EmberStatus](#) [emberSetRadioPower](#) (int8_t power)
- int8_t [emberGetRadioPower](#) (void)
- [EmberPanId](#) [emberGetPanId](#) (void)
- [EmberPanId](#) [emberRadioGetPanId](#) (void)
- void [emberGetExtendedPanId](#) (uint8_t *resultLocation)
- uint8_t [emberGetEndpoint](#) (uint8_t index)
- bool [emberGetEndpointDescription](#) (uint8_t endpoint, [EmberEndpointDescription](#) *result)
- uint16_t [emberGetEndpointCluster](#) (uint8_t endpoint, [EmberClusterListId](#) listId, uint8_t listIndex)
- bool [emberIsNodeIdValid](#) ([EmberNodeId](#) nodeId)
- [EmberNodeId](#) [emberLookupNodeIdByEui64](#) ([EmberEUI64](#) eui64)
- [EmberStatus](#) [emberLookupEui64ByNodeId](#) ([EmberNodeId](#) nodeId, [EmberEUI64](#) eui64-Return)
- void [emberCounterHandler](#) ([EmberCounterType](#) type, uint8_t data)

- void `emberStackTokenChangedHandler` (uint16_t tokenAddress)
- `EmberStatus emberGetNeighbor` (uint8_t index, `EmberNeighborTableEntry` *result)
- `EmberStatus emberGetRouteTableEntry` (uint8_t index, `EmberRouteTableEntry` *result)
- uint8_t `emberStackProfile` (void)
- uint8_t `emberTreeDepth` (void)
- uint8_t `emberNeighborCount` (void)
- uint8_t `emberRouteTableSize` (void)
- uint8_t `emberNextZigbeeSequenceNumber` (void)

Variables

- PGM uint8_t `emberStackProfileId` []
- uint8_t `emberEndpointCount`
- `EmberEndpoint emberEndpoints` []

Radio-specific Functions

- `EmberStatus emberSetTxPowerMode` (uint16_t txPowerMode)
- uint16_t `emberGetTxPowerMode` (void)
- `EmberStatus emberSetNodeId` (`EmberNodeId` nodeId)
- void `emberRadioNeedsCalibratingHandler` (void)
- void `emberCalibrateCurrentChannel` (void)

8.135.1 Detailed Description

EmberZNet API for accessing and setting stack information. See [Stack Information](#) for documentation.

Definition in file [stack-info.h](#).

8.136 stack-info.h

```

00001
00051 void emberStackStatusHandler(EmberStatus
00052     status);
00052
00060 EmberNetworkStatus emberNetworkState(void);
00061
00071 bool emberStackIsUp(void);
00072
00073 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00074
00078 EmberEUI64 emberGetEui64(void);
00079
00086 bool emberIsLocalEui64(EmberEUI64 eui64);
00087
00093 EmberNodeId emberGetNodeId(void);
00094
00100 EmberNodeId emberRadioGetNodeId(void);
00101
00107 void emberSetManufacturerCode(uint16_t code);
00108
00115 void emberSetPowerDescriptor(uint16_t descriptor);
00116
00123 void emberSetMaximumIncomingTransferSize(
00124     uint16_t size);
00124
00131 void emberSetMaximumOutgoingTransferSize(
```

```

        uint16_t size);
00132 void emberSetDescriptorCapability(uint8_t
00133     capability);
00138
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 uint16_t emManufacturerCode;
00152 extern uint16_t emPowerDescriptor;
00153 extern uint16_t emMaximumIncomingTransferSize;
00154 extern uint16_t emMaximumOutgoingTransferSize;
00155 extern uint8_t 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(
00178     EmberNetworkParameters *parameters);
00188 EmberStatus emberGetNodeType(EmberNodeType
00189     *resultLocation);
00203 EmberStatus emberSetRadioChannel(uint8_t channel
00204 );
00212 uint8_t emberGetRadioChannel(void);
00213
00231 EmberStatus emberSetRadioPower(int8_t power);
00232
00240 int8_t emberGetRadioPower(void);
00241
00246 EmberPanId emberGetPanId(void);
00247
00252 EmberPanId emberRadioGetPanId(void);
00253
00258 void emberGetExtendedPanId(uint8_t *resultLocation);
00259
00261 extern PGM uint8_t emberStackProfileId[];
00262
00270 typedef struct {
00272     uint16_t profileId;
00274     uint16_t deviceId;
00276     uint8_t deviceVersion;
00278     uint8_t inputClusterCount;
00280     uint8_t outputClusterCount;
00281 } EmberEndpointDescription;
00282
00286 typedef struct {
00288     uint8_t endpoint;
00290     EmberEndpointDescription PGM * description
00291 ;
00292     uint16_t PGM * inputClusterList;
00294     uint16_t PGM * outputClusterList;
00295 } EmberEndpoint;
00296
00298 extern uint8_t emberEndpointCount;
00299
00313 extern EmberEndpoint emberEndpoints[];
00314
00328 uint8_t emberGetEndpoint(uint8_t index);
00329
00345 bool emberGetEndpointDescription(uint8_t endpoint,

```

```

00346             *result);
00347
00366 uint16_t emberGetEndpointCluster(uint8_t endpoint,
00367                                         EmberClusterListId listId,
00368                                         uint8_t listIndex);
00369
00376 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00377 bool emberIsNodeIdValid(EmberNodeId nodeId);
00378 #else
00379 #define emberIsNodeIdValid(nodeId) ((nodeId) < EMBER_DISCOVERY_ACTIVE_NODE_ID)
00380 #endif
00381
00391 EmberNodeId emberLookupNodeIdByEui64(
00392     EmberEUI64 eui64);
00393
00405 EmberStatus emberLookupEui64ByNodeId(
00406     EmberNodeId nodeId,
00407                                         EmberEUI64 eui64Return);
00408
00421 void emberCounterHandler(EmberCounterType
00422     type, uint8_t data);
00423
00428 void emberStackTokenChangedHandler(uint16_t
00429     tokenAddress);
00430
00445 EmberStatus emberGetNeighbor(uint8_t index,
00446     EmberNeighborTableEntry *result);
00447
00458 EmberStatus emberGetRouteTableEntry(uint8_t
00459     index, EmberRouteTableEntry *result);
00460
00461 uint8_t emberStackProfile(void);
00462
00469 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00470
00474 uint8_t emberTreeDepth(void);
00475 #endif
00476
00480 uint8_t emberNeighborCount(void);
00481 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00482
00486 uint8_t emberRouteTableSize(void);
00487
00488 #else // Doxygen ignores the following
00489 // The '+ 0' prevents anyone from accidentally assigning to these.
00490 #define emberTreeDepth() (emTreeDepth + 0)
00491 #define emberMaxDepth() (emMaxDepth + 0)
00492 #define emberRouteTableSize() (emRouteTableSize + 0)
00493
00494 extern uint8_t emDefaultSecurityLevel;
00495 extern uint8_t emMaxHops;
00496 extern uint8_t emRouteTableSize;
00497 extern uint8_t emMaxDepth; // maximum tree depth
00498 extern uint8_t emTreeDepth; // our depth
00499 extern uint8_t emEndDeviceConfiguration;
00500
00501 // This was originally a #define declared here. I moved
00502 // emZigbeeNetworkSecurityLevel to the networkInfo struct. This function is now
00503 // defined in ember-stack-common.c
00504 uint8_t emberSecurityLevel(void);
00505 // New function (defined in ember-stack-common.c)
00506 void emberSetSecurityLevel(uint8_t securityLevel);
00507 #endif
00508
00513 uint8_t emberNextZigbeeSequenceNumber(void);
00514
00517
00551 EmberStatus emberSetTxPowerMode(uint16_t
00552     txPowerMode);
00553
00558 uint16_t emberGetTxPowerMode(void);
00559
00567 EmberStatus emberSetNodeId(EmberNodeId
00568     nodeId);
00569
00586 void emberRadioNeedsCalibratingHandler(void);
00587
00596 void emberCalibrateCurrentChannel(void);

```

00598

8.137 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](#) (uint8_t commState)
- bool [checkDebugMenuOption](#) (uint8_t ch)
- BL_Status [initOtaState](#) (void)
- BL_Status [checkOtaStart](#) (void)
- BL_Status [receiveOtaImage](#) (void)
- bool [paIsPresent](#) (void)
- bool [halCheckIntegrity](#) (void)

8.137.1 Detailed Description

See [Standalone](#) for detailed documentation.

Definition in file [standalone-bootloader.h](#).

8.138 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(uint8_t commState);
00044
00050 bool checkDebugMenuOption(uint8_t ch);
00051
00052
00058 BL_Status initOtaState(void);
00059
00065 BL_Status checkOtaStart(void);
00066
00075 BL_Status receiveOtaImage(void);
00076
00084 bool paIsPresent(void);
00085
00091 bool halCheckIntegrity(void);
00092
00096 #endif //__STANDALONE_BOOTLOADER_H__
00097

```

8.139 symbol-timer.h File Reference

Symbol Timer Functions

- void [halInternalStartSymbolTimer](#) (void)
- void [halInternalSfdCaptureIsr](#) (void)
- uint32_t [halStackGetInt32uSymbolTick](#) (void)

MAC Timer Support Functions

These functions are used for MAC layer timing.

Applications should not directly call these functions. They are used internally by the operation of the stack.

- void [halStackOrderInt16uSymbolDelayA](#) (uint16_t symbols)
- void [halStackOrderNextSymbolDelayA](#) (uint16_t symbols)
- void [halStackCancelSymbolDelayA](#) (void)
- void [halStackSymbolDelayAIsr](#) (void)

8.139.1 Detailed Description

Functions that provide access to symbol time. One symbol period is 16 microseconds. See [Symbol Timer Control](#) for documentation.

Definition in file [symbol-timer.h](#).

8.140 symbol-timer.h

```

00001
00017 #ifndef __SYMBOL_TIMER_H__
00018 #define __SYMBOL_TIMER_H__
00019
00023
00030 void halInternalStartSymbolTimer(void);
00031
00035 void halInternalSfdCaptureIsr(void);
00036
00042 uint32_t halStackGetInt32uSymbolTick(void);
00043
00054
00063 void halStackOrderInt16uSymbolDelayA(uint16_t
    symbols);
00064
00075 void halStackOrderNextSymbolDelayA(uint16_t
    symbols);
00076
00079 void halStackCancelSymbolDelayA(void);
00080
00084 void halStackSymbolDelayAIsr(void);
00085
00088 #endif //__SYMBOL_TIMER_H__
00089

```

8.141 system-timer.h File Reference

Macros

- #define halIdleForMilliseconds(duration)

Functions

- uint16_t halInternalStartSystemTimer (void)
- uint16_t halCommonGetInt16uMillisecondTick (void)
- uint32_t halCommonGetInt32uMillisecondTick (void)
- uint16_t halCommonGetInt16uQuarterSecondTick (void)
- EmberStatus halSleepForQuarterSeconds (uint32_t *duration)
- EmberStatus halSleepForMilliseconds (uint32_t *duration)
- EmberStatus halCommonIdleForMilliseconds (uint32_t *duration)

8.141.1 Detailed Description

See [System Timer Control](#) for documentation.

Definition in file [system-timer.h](#).

8.142 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 #endif
00039
00040 #endif // DOXYGEN_SHOULD_SKIP_THIS
00041
00042
00049 uint16_t halInternalStartSystemTimer(void);
00050
00051
00059 uint16_t halCommonGetInt16uMillisecondTick(
    void);
00060
00070 uint32_t halCommonGetInt32uMillisecondTick(
    void);
00071
00081 uint16_t halCommonGetInt16uQuarterSecondTick(
    void);
00082
00124 EmberStatus halSleepForQuarterSeconds(
    uint32_t *duration);
00125
00167 EmberStatus halSleepForMilliseconds(uint32_t
    *duration);
00168
00191 EmberStatus halCommonIdleForMilliseconds
    (uint32_t *duration);
00192 // Maintain the previous API for backwards compatibility
00193 #define halIdleForMilliseconds(duration)
    halCommonIdleForMilliseconds((duration))
00194
00195 #ifdef EMBER_STACK_COBRA
00196 EmberStatus halCobraIdleForMicroseconds(uint32_t *duration);
00197 #endif
00198
00199 #endif // __SYSTEM_TIMER_H__
00200

```

8.143 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_TEST_TEMP
- #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_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 CREATOR_MFG_SECURE_BOOTLOADER_KEY
- #define CREATOR_MFG_ETHERNET_ADDRESS
- #define CREATOR_MFG_CBKE_283K1_DATA
- #define CREATOR_MFG_XO_TUNE
- #define CREATOR_MFG_EUI_64
- #define CURRENT_MFG_TOKEN_VERSION
- #define VALID_MFG_TOKEN VERSIONS
- #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,...)

8.143.1 Detailed Description

Definitions for manufacturing tokens. CAUTION: This file is generated by gen_all.py which invokes gen_em3xx_token_mfg_h.py.

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
#ifndef DEFINETYPES
typedef uint8_t tokTypeMfgExample[8];
#endif //DEFINETYPES
#ifndef DEFINETOKENS
#define MFG_EXAMPLE_LOCATION 0x0980
#define 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](#).

8.143.2 Macro Definition Documentation

8.143.2.1 #define [DEFINE_MFG_TOKEN\(name, type, address, ... \)](#)

Definition at line 60 of file [token-manufacturing.h](#).

8.143.2.2 #define TOKEN_NEXT_ADDRESS(*region*, *address*)

Definition at line [66](#) of file [token-manufacturing.h](#).

8.143.2.3 #define CREATOR_MFG_CHIP_DATA

Definition at line [85](#) of file [token-manufacturing.h](#).

8.143.2.4 #define CREATOR_MFG_PART_DATA

Definition at line [86](#) of file [token-manufacturing.h](#).

8.143.2.5 #define CREATOR_MFG_TESTER_DATA

Definition at line [87](#) of file [token-manufacturing.h](#).

8.143.2.6 #define CREATOR_MFG_EMBER_EUI_64

Definition at line [88](#) of file [token-manufacturing.h](#).

8.143.2.7 #define CREATOR_MFG_ANALOG_TRIM_NORMAL

Definition at line [89](#) of file [token-manufacturing.h](#).

8.143.2.8 #define CREATOR_MFG_ANALOG_TRIM_BOOST

Definition at line [90](#) of file [token-manufacturing.h](#).

8.143.2.9 #define CREATOR_MFG_ANALOG_TRIM_BOTH

Definition at line [91](#) of file [token-manufacturing.h](#).

8.143.2.10 #define CREATOR_MFG_REG_TRIM

Definition at line [92](#) of file [token-manufacturing.h](#).

8.143.2.11 #define CREATOR_MFG_1V8_REG_VOLTAGE

Definition at line [93](#) of file [token-manufacturing.h](#).

8.143.2.12 #define CREATOR_MFG_VREF_VOLTAGE

Definition at line [94](#) of file [token-manufacturing.h](#).

8.143.2.13 #define CREATOR_MFG_TEMP_CAL

Definition at line 95 of file [token-manufacturing.h](#).

8.143.2.14 #define CREATOR_MFG_TEST_TEMP

Definition at line 96 of file [token-manufacturing.h](#).

8.143.2.15 #define CREATOR_MFG_FIB_VERSION

Definition at line 97 of file [token-manufacturing.h](#).

8.143.2.16 #define CREATOR_MFG_FIB_CHECKSUM

Definition at line 98 of file [token-manufacturing.h](#).

8.143.2.17 #define CREATOR_MFG_FIB_OBS

Definition at line 99 of file [token-manufacturing.h](#).

8.143.2.18 #define CREATOR_MFG_CIB_OBS

Definition at line 102 of file [token-manufacturing.h](#).

8.143.2.19 #define CREATOR_MFG_CUSTOM_VERSION

Definition at line 103 of file [token-manufacturing.h](#).

8.143.2.20 #define CREATOR_MFG_CUSTOM_EUI_64

Definition at line 104 of file [token-manufacturing.h](#).

8.143.2.21 #define CREATOR_MFG_STRING

Definition at line 105 of file [token-manufacturing.h](#).

8.143.2.22 #define CREATOR_MFG_BOARD_NAME

Definition at line 106 of file [token-manufacturing.h](#).

8.143.2.23 #define CREATOR_MFG_MANUF_ID

Definition at line 107 of file [token-manufacturing.h](#).

8.143.2.24 #define CREATOR_MFG_PHY_CONFIG

Definition at line 108 of file [token-manufacturing.h](#).

8.143.2.25 #define CREATOR_MFG_BOOTLOAD_AES_KEY

Definition at line 109 of file [token-manufacturing.h](#).

8.143.2.26 #define CREATOR_MFG_EZSP_STORAGE

Definition at line 110 of file [token-manufacturing.h](#).

8.143.2.27 #define CREATOR_MFG_ASH_CONFIG

Definition at line 111 of file [token-manufacturing.h](#).

8.143.2.28 #define CREATOR_MFG_CBKE_DATA

Definition at line 112 of file [token-manufacturing.h](#).

8.143.2.29 #define CREATOR_MFG_INSTALLATION_CODE

Definition at line 113 of file [token-manufacturing.h](#).

8.143.2.30 #define CREATOR_MFG_OSC24M_BIAS_TRIM

Definition at line 114 of file [token-manufacturing.h](#).

8.143.2.31 #define CREATOR_MFG_SYNTH_FREQ_OFFSET

Definition at line 115 of file [token-manufacturing.h](#).

8.143.2.32 #define CREATOR_MFG_OSC24M_SETTLE_DELAY

Definition at line 116 of file [token-manufacturing.h](#).

8.143.2.33 #define CREATOR_MFG_SECURITY_CONFIG

Definition at line 117 of file [token-manufacturing.h](#).

8.143.2.34 #define CREATOR_MFG_CCA_THRESHOLD

Definition at line 118 of file [token-manufacturing.h](#).

8.143.2.35 #define CREATOR_MFG_SECURE_BOOTLOADER_KEY

Definition at line 119 of file [token-manufacturing.h](#).

8.143.2.36 #define CREATOR_MFG_ETHERNET_ADDRESS

Definition at line 120 of file [token-manufacturing.h](#).

8.143.2.37 #define CREATOR_MFG_CBKE_283K1_DATA

Definition at line 121 of file [token-manufacturing.h](#).

8.143.2.38 #define CREATOR_MFG_XO_TUNE

Definition at line 122 of file [token-manufacturing.h](#).

8.143.2.39 #define CREATOR_MFG_EUI_64

Definition at line 123 of file [token-manufacturing.h](#).

8.143.2.40 #define CURRENT_MFG_TOKEN_VERSION

Definition at line 127 of file [token-manufacturing.h](#).

8.143.2.41 #define VALID_MFG_TOKEN VERSIONS

Definition at line 128 of file [token-manufacturing.h](#).

8.143.2.42 #define CURRENT_MFG_CUSTOM_VERSION

Definition at line 129 of file [token-manufacturing.h](#).

8.144 token-manufacturing.h

```

0001
0002 #define DEFINE_MFG_TOKEN(name, type, address, ...) \
0003     TOKEN_NEXT_ADDRESS(name, (address)) \
0004     TOKEN_MFG(name, CREATOR_##name, 0, 0, type, 1, __VA_ARGS__)
0005
0006 #ifndef TOKEN_NEXT_ADDRESS
0007     #define TOKEN_NEXT_ADDRESS(region, address)
0008 #endif
0009
0010 // MANUFACTURING DATA
0011 // *the addresses of these tokens must not change*
0012
0013
0014 // MANUFACTURING CREATORS
0015 // The creator codes are here in one list instead of next to their token
0016 // definitions so comparison of the codes is easier. The only requirement
0017 // on these creator definitions is that they all must be unique. A favorite
0018 // method for picking creator codes is to use two ASCII characters in order
0019 // to make the codes more memorable. Also, the msb of Stack and Manufacturing
0020 // token creator codes is always 1, while the msb of Application token creator

```

```

00081 // codes is always 0. This distinction allows Application tokens
00082 // to freely use the lower 15 bits for creator codes without the risk of
00083 // duplicating a Stack or Manufacturing token creator code.
00084 //--- Fixed Information Block ---
00085 #define CREATOR_MFG_CHIP_DATA          0xC344 // msb+'C'+'D'
00086 #define CREATOR_MFG_PART_DATA         0xF064 // msb+'p'+'d'
00087 #define CREATOR_MFG_TESTER_DATA        0xF464 // msb+'t'+'d'
00088 #define CREATOR_MFG_EMBER_EUI_64      0xE545 // msb+'e'+'E'
00089 #define CREATOR_MFG_ANALOG_TRIM_NORMAL 0xF46E // msb+'t'+'n'
00090 #define CREATOR_MFG_ANALOG_TRIM_BOOST   0xF442 // msb+'t'+'B'
00091 #define CREATOR_MFG_ANALOG_TRIM_BOTH    0xF462 // msb+'t'+'b'
00092 #define CREATOR_MFG_REG_TRIM          0xF274 // msb+'r'+'t'
00093 #define CREATOR_MFG_1V8_REG_VOLTAGE    0xF276 // msb+'r'+'v'
00094 #define CREATOR_MFG_VREF_VOLTAGE      0xF676 // msb+'v'+'v'
00095 #define CREATOR_MFG_TEMP_CAL          0xF463 // msb+'t'+'c'
00096 #define CREATOR_MFG_TEST_TEMP         0xF474 // msb+'t'+'t'
00097 #define CREATOR_MFG_FIB_VERSION       0xFF09
00098 #define CREATOR_MFG_FIB_CHECKSUM     0xE663 // msb+'f'+'c'
00099 #define CREATOR_MFG_FIB_OBS          0xE66F // msb+'f'+'o'
00100
00101 //--- Customer Information Block ---
00102 #define CREATOR_MFG_C1B_OBS          0xE36F // msb+'c'+'o'
00103 #define CREATOR_MFG_CUSTOM_VERSION    0xC356
00104 #define CREATOR_MFG_CUSTOM_EUI_64      0xE345
00105 #define CREATOR_MFG_STRING           0xED73
00106 #define CREATOR_MFG_BOARD_NAME        0xC24E // msb+'B'+'N' (Board Name)
00107 #define CREATOR_MFG_MANUF_ID          0xC944 // msb+'I'+'D' (Id)
00108 #define CREATOR_MFG_PHY_CONFIG        0xD043 // msb+'P'+'C' (Phy Config)
00109 #define CREATOR_MFG_BOOTLOAD_AES_KEY   0xC24B // msb+'B'+'K' (Bootloader Key)
00110 #define CREATOR_MFG_EZSP_STORAGE       0xCD53
00111 #define CREATOR_MFG_ASH_CONFIG         0xC143 // msb+'A'+'C' (ASH Config)
00112 #define CREATOR_MFG_CBKE_DATA          0xC342 // msb+'C'+'B' (CBke)
00113 #define CREATOR_MFG_INSTALLATION_CODE   0xC943 // msb+'I'+'C' (Installation Code)
00114 #define CREATOR_MFG_OSC24M_BIAS_TRIM    0xB254 // msb+'2'+'T' (2[4mHz] Trim)
00115 #define CREATOR_MFG_SYNTH_FREQ_OFFSET   0xD346 // msb+'S'+'F' (Synth Freq)
00116 #define CREATOR_MFG_OSC24M_SETTLE_DELAY  0xB253 // msb+'2'+'S' (2[4mHz] Settle)
00117 #define CREATOR_MFG_SECURITY_CONFIG     0xD343 // msb+'S'+'C' (Security Config)
00118 #define CREATOR_MFG_CCA_THRESHOLD       0xC343 // msb+'C'+'C' (Clear Channel)
00119 #define CREATOR_MFG_SECURE_BOOTLOADER_KEY 0xD342 // msb+'S'+'B' (Secure Bootloader)
00120 #define CREATOR_MFG_ETHERNET_ADDRESS     0xC554 // msb + 'E' + 'T'
00121 #define CREATOR_MFG_CBKE_283K1_DATA      0xC345 // msb+'C'+'B' (CBke) 283k1 ECC
(a.k.a. SE 1.2)
00122 #define CREATOR_MFG_XO_TUNE            0xD854 // msb+'X'+'T' (XO TUNE Value)
00123 #define CREATOR_MFG_EUI_64             0xB634
00124
00125
00126 // The master defines indicating the version number these definitions work
with.
00127 #define CURRENT_MFG_TOKEN_VERSION 0x01FE //MSB is version, LSB is complement
00128 #define VALID_MFG_TOKEN_VERSIONS {0x01FE, 0x02FD}
00129 #define CURRENT_MFG_CUSTOM_VERSION 0x01FE //MSB is version, LSB is complement
00130
00131
00132 #ifdef DEFINETYPES
00133 //--- Fixed Information Block ---
00134 typedef uint8_t tokTypeMfgChipData[24];
00135 typedef uint8_t tokTypeMfgPartData[6];
00136 typedef uint8_t tokTypeMfgTesterData[6];
00137 typedef uint8_t tokTypeMfgEmberEui64[8];
00138 typedef struct {
00139     uint16_t ifffilterL;
00140     uint16_t lna;
00141     uint16_t ifamp;
00142     uint16_t rxadch;
00143     uint16_t prescalar;
00144     uint16_t phdet;
00145     uint16_t vco;
00146     uint16_t loopfilter;
00147     uint16_t pa;
00148     uint16_t iqmixer;
00149 } tokTypeMfgAnalogueTrim;
00150 typedef struct {
00151     uint16_t ifffilterH;
00152     uint16_t biasmaster;
00153     uint16_t moddac;
00154     uint16_t auxadc;
00155     uint16_t caladc;
00156 } tokTypeMfgAnalogueTrimBoth;

```

```

00157 typedef struct {
00158     uint8_t regTrim1V2;
00159     uint8_t regTrim1V8;
00160 } tokTypeMfgRegTrim;
00161 typedef uint16_t tokTypeMfgRegVoltage1V8;
00162 typedef uint16_t tokTypeMfgAdcVrefVoltage;
00163 typedef uint16_t tokTypeMfgTempCal;
00164 typedef struct {
00165     int8_t temp1;
00166     int8_t temp2;
00167 } tokTypeMfgTestTemp;
00168 typedef uint16_t tokTypeMfgFibVersion;
00169 typedef uint16_t tokTypeMfgFibChecksum;
00170 typedef struct {
00171     uint16_t ob2;
00172     uint16_t ob3;
00173     uint16_t ob0;
00174     uint16_t ob1;
00175 } tokTypeMfgFibObs;
00176
00177 //--- Customer Information Block ---
00178 typedef struct {
00179     uint16_t ob0;
00180     uint16_t ob1;
00181     uint16_t ob2;
00182     uint16_t ob3;
00183     uint16_t ob4;
00184     uint16_t ob5;
00185     uint16_t ob6;
00186     uint16_t ob7;
00187 } tokTypeMfgCibObs;
00188 typedef uint16_t tokTypeMfgCustomVersion;
00189 typedef uint8_t tokTypeMfgCustomEui64[8];
00190 typedef uint8_t tokTypeMfgString[16];
00191 typedef uint8_t tokTypeMfgBoardName[16];
00192 typedef uint16_t tokTypeMfgManufId;
00193 typedef uint16_t tokTypeMfgPhyConfig;
00194 typedef uint8_t tokTypeMfgBootloadAesKey[16];
00195 typedef uint8_t tokTypeMfgEzspStorage[8];
00196 typedef uint16_t tokTypeMfgAshConfig;
00197 // Smart Energy 1.0 (ECC 163k1 based crypto - 80-bit symmetric equivalent)
00198 typedef struct {
00199     uint8_t certificate[48];
00200     uint8_t caPublicKey[22];
00201     uint8_t privateKey[21];
00202     // The bottom flag bit is 1 for uninitialized, 0 for initialized.
00203     // The other flag bits should be set to 0 at initialization.
00204     uint8_t flags;
00205 } tokTypeMfgCbkeData;
00206 typedef struct {
00207     // The bottom flag bit is 1 for uninitialized, 0 for initialized.
00208     // Bits 1 and 2 give the size of the value string:
00209     // 0 = 6 bytes, 1 = 8 bytes, 2 = 12 bytes, 3 = 16 bytes.
00210     // The other flag bits should be set to 0 at initialization.
00211     // Special flags support. Due to a bug in the way some customers
00212     // had programmed the flags field, we will also examine the upper
00213     // bits 9 and 10 for the size field. Those bits are also reserved.
00214     uint16_t flags;
00215     uint8_t value[16];
00216     uint16_t crc;
00217 } tokTypeMfgInstallationCode;
00218 typedef uint16_t tokTypeMfgOsc24mBiasTrim;
00219 typedef uint16_t tokTypeMfgSynthFreqOffset;
00220 typedef uint16_t tokTypeMfgOsc24mSettleDelay;
00221 typedef uint16_t tokTypeMfgSecurityConfig;
00222 typedef uint16_t tokTypeMfgCcaThreshold;
00223 typedef struct {
00224     uint8_t data[16]; // AES-128 key
00225 } tokTypeMfgSecureBootloaderKey;
00226 typedef struct {
00227     uint8_t data[6];
00228 } tokTypeMfgEthernetAddress;
00229 // Smart Energy 1.2 (ECC 283k1 based crypto - 128-bit symmetric equivalent)
00230 typedef struct {
00231     uint8_t certificate[74];
00232     uint8_t caPublicKey[37];
00233     uint8_t privateKey[36];
00234     // The bottom flag bit is 1 for uninitialized, 0 for initialized.
00235     // The other flag bits should be set to 0 at initialization.
00236     uint8_t flags;

```

```

00237 } tokTypeMfgCbke283k1Data;
00238 typedef uint16_t tokTypeMfgXoTune;
00239 typedef uint8_t tokTypeMfgEui64[8];
00240
00241 #endif //DEFINETYPES
00242
00243
00244 #ifdef DEFINETOKENS
00245 //The Manufacturing tokens need to be stored at well-defined locations.
00246 //None of these addresses should ever change without extremely great care.
00247 //All locations are OR'ed with DATA_BIG_INFO_BASE to make a full 32bit address.
00248 //--- Fixed Information Block ---
00249 // FIB Bootloader          0x0000 //1918 bytes
00250 #define MFG_CHIP_DATA_LOCATION      0x077E // 24 bytes
00251 #define MFG_PART_DATA_LOCATION      0x0796 // 6 bytes
00252 #define MFG_TESTER_DATA_LOCATION     0x079C // 6 bytes
00253 #define MFG_EMBER_EUI_64_LOCATION    0x07A2 // 8 bytes
00254 #define MFG_ANALOG_TRIM_NORMAL_LOCATION 0x07AA // 20 bytes
00255 #define MFG_ANALOG_TRIM_BOOST_LOCATION 0x07BE // 20 bytes
00256 #define MFG_ANALOG_TRIM_BOTH_LOCATION 0x07D2 // 10 bytes
00257 #define MFG_REG_TRIM_LOCATION        0x07DC // 2 bytes
00258 #define MFG_1V8_REG_VOLTAGE_LOCATION 0x07DE // 2 bytes
00259 #define MFG_VREF_VOLTAGE_LOCATION    0x07E0 // 2 bytes
00260 #define MFG_TEMP_CAL_LOCATION        0x07E2 // 2 bytes
00261 #define MFG_TEST_TEMP_LOCATION       0x07E4 // 2 bytes
00262 #define MFG_FIB_VERSION_LOCATION    0x07F4 // 2 bytes
00263 #define MFG_FIB_CHECKSUM_LOCATION   0x07F6 // 2 bytes
00264 #define MFG_FIB_OBS_LOCATION        0x07F8 // 8 bytes
00265
00266 //--- Customer Information Block ---
00267 // The CIB is a 2KB block starting at 0x08040800.
00268 #define MFG_CIB_OBS_LOCATION        0x0800 // 128 bytes
00269 #define MFG_CUSTOM_VERSION_LOCATION 0x0810 // 2 bytes
00270 #define MFG_CUSTOM_EUI_64_LOCATION  0x0812 // 8 bytes
00271 #define MFG_STRING_LOCATION         0x081A // 16 bytes
00272 #define MFG_BOARD_NAME_LOCATION    0x082A // 16 bytes
00273 #define MFG_MANUF_ID_LOCATION      0x083A // 2 bytes
00274 #define MFG_PHY_CONFIG_LOCATION    0x083C // 2 bytes
00275 #define MFG_BOOTLOAD_AES_KEY_LOCATION 0x083E // 16 bytes
00276 #define MFG_EZSP_STORAGE_LOCATION   0x084E // 8 bytes
00277 #define MFG_ASH_CONFIG_LOCATION    0x0856 // 40 bytes
00278 #define MFG_CBKE_DATA_LOCATION     0x087E // 92 bytes
00279 #define MFG_INSTALLATION_CODE_LOCATION 0x08DA // 20 bytes
00280 #define MFG_OSC24M_BIAS_TRIM_LOCATION 0x08EE // 2 bytes
00281 #define MFG_SYNTH_FREQ_OFFSET_LOCATION 0x08F0 // 2 bytes
00282 #define MFG_OSC24M_SETTLE_DELAY_LOCATION 0x08F2 // 2 bytes
00283 #define MFG_SECURITY_CONFIG_LOCATION 0x08F4 // 2 bytes
00284 #define MFG_CCA_THRESHOLD_LOCATION   0x08F6 // 2 bytes
00285 #define MFG_SECURE_BOOTLOADER_KEY_LOCATION 0x08F8 // 16 bytes
00286 #define MFG_ETHERNET_ADDRESS_LOCATION 0x0908 // 6 bytes
00287 #define MFG_CBKE_283K1_DATA_LOCATION 0x090E // 148 bytes
00288 #define MFG_XO_TUNE_LOCATION        0x09A2 // 2 bytes
00289
00290 //--- Virtual MFG Tokens ---
00291 #define MFG_EUI_64_LOCATION        0xB634 // 8 bytes
00292
00293
00294 // Define the size of indexed token array
00295 #define MFG_ASH_CONFIG_ARRAY_SIZE   20
00296
00297
00298 //--- Fixed Information Block ---
00299 TOKEN_NEXT_ADDRESS(MFG_CHIP_DATA_ADDR, MFG_CHIP_DATA_LOCATION)
00300 TOKEN_MFG(MFG_CHIP_DATA, CREATOR_MFG_CHIP_DATA,
00301           0, 0, tokTypeMfgChipData, 1,
00302           {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
00303           0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF})
00304 TOKEN_NEXT_ADDRESS(MFG_PART_DATA_ADDR, MFG_PART_DATA_LOCATION)
00305 TOKEN_MFG(MFG_PART_DATA, CREATOR_MFG_PART_DATA,
00306           0, 0, tokTypeMfgPartData, 1,
00307           {0xFF,0xFF,0xFF,0xFF,0xFF})
00308
00309 TOKEN_NEXT_ADDRESS(MFG_TESTER_DATA_ADDR,
00310                      MFG_TESTER_DATA_LOCATION)
00311 TOKEN_MFG(MFG_TESTER_DATA, CREATOR_MFG_TESTER_DATA,
00312           0, 0, tokTypeMfgTesterData, 1,
00313           {0xFF,0xFF,0xFF,0xFF,0xFF})

```

```

00313
00314 TOKEN_NEXT_ADDRESS (MFG_EMBER_EUI_64_ADDR,
00315     MFG_EMBER_EUI_64_LOCATION)
00315 TOKEN_MFG(MFG_EMBER_EUI_64, CREATOR_MFG_EMBER_EUI_64,
00316     0, 0, tokTypeMfgEmberEui64, 1,
00317     {3,0,0,0,0,0,3})
00318
00319 TOKEN_NEXT_ADDRESS (MFG_ANALOG_TRIM_NORMAL_ADDR,
00320     MFG_ANALOG_TRIM_NORMAL_LOCATION)
00320 TOKEN_MFG(MFG_ANALOG_TRIM_NORMAL, CREATOR_MFG_ANALOG_TRIM_NORMAL
00321     ,
00322     0, 0, tokTypeMfgAnalogueTrim, 1,
00323     {0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
00324 TOKEN_NEXT_ADDRESS (MFG_ANALOG_TRIM_BOOST_ADDR,
00325     MFG_ANALOG_TRIM_BOOST_LOCATION)
00325 TOKEN_MFG(MFG_ANALOG_TRIM_BOOST, CREATOR_MFG_ANALOG_TRIM_BOOST
00326     ,
00327     0, 0, tokTypeMfgAnalogueTrim, 1,
00328     {0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
00329 TOKEN_NEXT_ADDRESS (MFG_ANALOG_TRIM_BOTH_ADDR,
00330     MFG_ANALOG_TRIM_BOTH_LOCATION)
00330 TOKEN_MFG(MFG_ANALOG_TRIM_BOTH, CREATOR_MFG_ANALOG_TRIM_BOTH
00331     ,
00332     0, 0, tokTypeMfgAnalogueTrimBoth, 1,
00333     {0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF})
00334 TOKEN_NEXT_ADDRESS (MFG_REG_TRIM_ADDR, MFG_REG_TRIM_LOCATION)
00335 TOKEN_MFG(MFG_REG_TRIM, CREATOR_MFG_REG_TRIM,
00336     0, 0, tokTypeMfgRegTrim, 1,
00337     {0xFF, 0xFF})
00338
00339 TOKEN_NEXT_ADDRESS (MFG_1V8_REG_VOLTAGE_ADDR,
00340     MFG_1V8_REG_VOLTAGE_LOCATION)
00340 TOKEN_MFG(MFG_1V8_REG_VOLTAGE, CREATOR_MFG_1V8_REG_VOLTAGE
00341     ,
00342     0, 0, tokTypeMfgRegVoltage1V8, 1,
00343     0xFFFF)
00344 TOKEN_NEXT_ADDRESS (MFG_VREF_VOLTAGE_ADDR,
00345     MFG_VREF_VOLTAGE_LOCATION)
00345 TOKEN_MFG(MFG_VREF_VOLTAGE, CREATOR_MFG_VREF_VOLTAGE,
00346     0, 0, tokTypeMfgAdcVrefVoltage, 1,
00347     0xFFFF)
00348
00349 TOKEN_NEXT_ADDRESS (MFG_TEMP_CAL_ADDR, MFG_TEMP_CAL_LOCATION)
00350 TOKEN_MFG(MFG_TEMP_CAL, CREATOR_MFG_TEMP_CAL,
00351     0, 0, tokTypeMfgTempCal, 1,
00352     0xFFFF)
00353
00354 TOKEN_NEXT_ADDRESS (MFG_TEST_TEMP_ADDR, MFG_TEST_TEMP_LOCATION
00355 )
00355 TOKEN_MFG(MFG_TEST_TEMP, CREATOR_MFG_TEST_TEMP,
00356     0, 0, tokTypeMfgTestTemp, 1,
00357     {0xFF, 0xFF})
00358
00359 TOKEN_NEXT_ADDRESS (MFG_FIB_VERSION_ADDR,
00360     MFG_FIB_VERSION_LOCATION)
00360 TOKEN_MFG(MFG_FIB_VERSION, CREATOR_MFG_FIB_VERSION,
00361     0, 0, tokTypeMfgFibVersion, 1,
00362     CURRENT_MFG_TOKEN_VERSION)
00363
00364 TOKEN_NEXT_ADDRESS (MFG_FIB_CHECKSUM_ADDR,
00365     MFG_FIB_CHECKSUM_LOCATION)
00365 TOKEN_MFG(MFG_FIB_CHECKSUM, CREATOR_MFG_FIB_CHECKSUM,
00366     0, 0, tokTypeMfgFibChecksum, 1,
00367     0xFFFF)
00368
00369 TOKEN_NEXT_ADDRESS (MFG_FIB_OBS_ADDR, MFG_FIB_OBS_LOCATION)
00370 TOKEN_MFG(MFG_FIB_OBS, CREATOR_MFG_FIB_OBS,
00371     0, 0, tokTypeMfgFibObs, 1,
00372     {0xFFFF, 0x03FC, 0xAA55, 0xFFFF})
00373
00374
00375 //--- Customer Information Block ---
00376 TOKEN_NEXT_ADDRESS (MFG_CIB_OBS_ADDR, MFG_CIB_OBS_LOCATION)
00377 TOKEN_MFG(MFG_CIB_OBS, CREATOR_MFG_CIB_OBS,

```

```

00378      0, 0, tokTypeMfgCibObs, 1,
00379      {0x5AA5, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF}
00380
00381 TOKEN_NEXT_ADDRESS (MFG_CUSTOM_VERSION_ADDR,
00382     MFG_CUSTOM_VERSION_LOCATION)
00382 TOKEN_MFG(MFG_CUSTOM_VERSION, CREATOR_MFG_CUSTOM_VERSION
00383
00384      0, 0, tokTypeMfgCustomVersion, 1,
00384      CURRENT_MFG_CUSTOM_VERSION)
00385
00386 TOKEN_NEXT_ADDRESS (MFG_CUSTOM_EUI_64_ADDR,
00386     MFG_CUSTOM_EUI_64_LOCATION)
00387 TOKEN_MFG(MFG_CUSTOM_EUI_64, CREATOR_MFG_CUSTOM_EUI_64
00388
00389      0, 0, tokTypeMfgCustomEui64, 1,
00389      {0,3,3,3,3,3,3,0})
00390
00391 TOKEN_NEXT_ADDRESS (MFG_STRING_ADDR, MFG_STRING_LOCATION)
00392 TOKEN_MFG(MFG_STRING, CREATOR_MFG_STRING,
00393      0, 0, tokTypeMfgString, 1,
00394      {0,})
00395
00396 TOKEN_NEXT_ADDRESS (MFG_BOARD_NAME_ADDR,
00396     MFG_BOARD_NAME_LOCATION)
00397 TOKEN_MFG(MFG_BOARD_NAME, CREATOR_MFG_BOARD_NAME,
00398      0, 0, tokTypeMfgBoardName, 1,
00399      {0,})
00400
00401 TOKEN_NEXT_ADDRESS (MFG_MANUF_ID_ADDR, MFG_MANUF_ID_LOCATION)
00402 TOKEN_MFG(MFG_MANUF_ID, CREATOR_MFG_MANUF_ID,
00403      0, 0, tokTypeMfgManufId, 1,
00404      {0x00,0x00,}) // default to 0 for ember
00405
00406 TOKEN_NEXT_ADDRESS (MFG_PHY_CONFIG_ADDR,
00406     MFG_PHY_CONFIG_LOCATION)
00407 TOKEN_MFG(MFG_PHY_CONFIG, CREATOR_MFG_PHY_CONFIG,
00408      0, 0, tokTypeMfgPhyConfig, 1,
00409      {0x00,0x00,}) // default to non-boost mode, internal pa.
00410
00411 TOKEN_NEXT_ADDRESS (MFG_BOOTLOAD_AES_KEY_ADDR,
00411     MFG_BOOTLOAD_AES_KEY_LOCATION)
00412 TOKEN_MFG(MFG_BOOTLOAD_AES_KEY, CREATOR_MFG_BOOTLOAD_AES_KEY
00413
00414      0, 0, tokTypeMfgBootloadAesKey, 1,
00414      {0xFF,}) // default key is all f's
00415
00416 TOKEN_NEXT_ADDRESS (MFG_EZSP_STORAGE_ADDR,
00416     MFG_EZSP_STORAGE_LOCATION)
00417 TOKEN_MFG(MFG_EZSP_STORAGE, CREATOR_MFG_EZSP_STORAGE,
00418      0, 0, tokTypeMfgEzspStorage, 1,
00419      { 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF })
00420
00421 TOKEN_NEXT_ADDRESS (MFG_ASH_CONFIG_ADDR,
00421     MFG_ASH_CONFIG_LOCATION)
00422 TOKEN_MFG(MFG_ASH_CONFIG, CREATOR_MFG_ASH_CONFIG,
00423      0, 1, tokTypeMfgAshConfig, MFG_ASH_CONFIG_ARRAY_SIZE,
00424      { 0xFFFF, })
00425
00426 TOKEN_NEXT_ADDRESS (MFG_CBKE_DATA_ADDR, MFG_CBKE_DATA_LOCATION
00427 )
00427 TOKEN_MFG(MFG_CBKE_DATA, CREATOR_MFG_CBKE_DATA,
00428      0, 0, tokTypeMfgCbkeData, 1,
00429      {0xFF,})
00430
00431 TOKEN_NEXT_ADDRESS (MFG_INSTALLATION_CODE_ADDR,
00431     MFG_INSTALLATION_CODE_LOCATION)
00432 TOKEN_MFG(MFG_INSTALLATION_CODE, CREATOR_MFG_INSTALLATION_CODE
00433
00434      0, 0, tokTypeMfgInstallationCode, 1,
00434      {0xFF,})
00435
00436 TOKEN_NEXT_ADDRESS (MFG_OSC24M_BIAS_TRIM_ADDR,
00436     MFG_OSC24M_BIAS_TRIM_LOCATION)
00437 TOKEN_MFG(MFG_OSC24M_BIAS_TRIM, CREATOR_MFG_OSC24M_BIAS_TRIM
00438
00439      0, 0, tokTypeMfgOsc24mBiasTrim, 1,
00439      {0xFF,})
00440
00441 TOKEN_NEXT_ADDRESS (MFG_SYNTH_FREQ_OFFSET_ADDR,
00441     MFG_SYNTH_FREQ_OFFSET_LOCATION)

```

```

00442 TOKEN_MFG(MFG_SYNTH_FREQ_OFFSET, CREATOR_MFG_SYNTH_FREQ_OFFSET
00443     , 0, 0, tokTypeMfgSynthFreqOffset, 1,
00444     {0xFF,0xFF,})
00445
00446 TOKEN_NEXT_ADDRESS(MFG_OSC24M_SETTLE_DELAY_ADDR,
00447 MFG_OSC24M_SETTLE_DELAY_LOCATION)
00448 TOKEN_MFG(MFG_OSC24M_SETTLE_DELAY, CREATOR_MFG_OSC24M_SETTLE_DELAY
00449     , 0, 0, tokTypeMfgOsc24mSettleDelay, 1,
00450     100)
00451 TOKEN_NEXT_ADDRESS(MFG_SECURITY_CONFIG_ADDR,
00452 MFG_SECURITY_CONFIG_LOCATION)
00453 TOKEN_MFG(MFG_SECURITY_CONFIG, CREATOR_MFG_SECURITY_CONFIG
00454     , 0, 0, tokTypeMfgSecurityConfig, 1,
00455     { 0xFF, 0xFF })
00456 TOKEN_NEXT_ADDRESS(MFG_CCA_THRESHOLD_ADDR,
00457 MFG_CCA_THRESHOLD_LOCATION)
00458 TOKEN_MFG(MFG_CCA_THRESHOLD, CREATOR_MFG_CCA_THRESHOLD
00459     , 0, 0, tokTypeMfgCcaThreshold, 1,
00460     {0xFF, 0xFF,})
00461 TOKEN_NEXT_ADDRESS(MFG_SECURE_BOOTLOADER_KEY_ADDR,
00462 MFG_SECURE_BOOTLOADER_KEY_LOCATION)
00463 TOKEN_MFG(MFG_SECURE_BOOTLOADER_KEY, CREATOR_MFG_SECURE_BOOTLOADER_KEY
00464     , 0, 0, tokTypeMfgSecureBootloaderKey, 1,
00465     {0xFF,}) // default key is all f's
00466 TOKEN_NEXT_ADDRESS(MFG_ETHERNET_ADDRESS_ADDR,
00467 MFG_ETHERNET_ADDRESS_LOCATION)
00468 TOKEN_MFG(MFG_ETHERNET_ADDRESS, CREATOR_MFG_ETHERNET_ADDRESS
00469     , 0, 0, tokTypeMfgEthernetAddress, 1,
00470     {0xFF,}) // default address is unset (all F's)
00471 TOKEN_NEXT_ADDRESS(MFG_CBKE_283K1_DATA_ADDR,
00472 MFG_CBKE_283K1_DATA_LOCATION)
00473 TOKEN_MFG(MFG_CBKE_283K1_DATA, CREATOR_MFG_CBKE_283K1_DATA
00474     , 0, 0, tokTypeMfgCbke283k1Data, 1,
00475     {0xFF,})
00476 TOKEN_NEXT_ADDRESS(MFG_XO_TUNE_ADDR, MFG_XO_TUNE_LOCATION)
00477 TOKEN_MFG(MFG_XO_TUNE, CREATOR_MFG_XO_TUNE,
00478     0, 0, tokTypeMfgXoTune, 1,
00479     {0xFFFF})
00480
00481 TOKEN_NEXT_ADDRESS(MFG_EUI_64_ADDR, MFG_EUI_64_LOCATION)
00482 TOKEN_MFG(MFG_EUI_64, CREATOR_MFG_EUI_64,
00483     0, 0, tokTypeMfgEui64, 1,
00484     {3,3,3,3,0,0,0,0})
00485
00486
00487 #endif //DEFINETOKENS
00488
00489
00490 #ifdef APPLICATION_MFG_TOKEN_HEADER
00491     #include APPLICATION_MFG_TOKEN_HEADER
00492 #endif
00493
00494 #undef TOKEN_NEXT_ADDRESS
00495

```

8.145 token-stack.h File Reference

```

#include "token-phy.h"
#include "stack/zll/zll-token-config.h"
#include "stack/rf4ce/rf4ce-token-config.h"
#include "stack/gp/gp-token-config.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_STACKAPS_FRAME_COUNTER
- #define CREATOR_STACK_TRUST_CENTER
- #define CREATOR_STACK_NETWORK_MANAGEMENT
- #define CREATOR_STACK_PARENT_INFO
- #define CREATOR_STACK_PARENT_ADDITIONAL_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_MANAGEMEN-T
- #define CREATOR_MULTI_NETWORK_STACK_PARENT_INFO

- #define CREATOR_MULTI_NETWORK_STACK_NONCE_COUNTER
- #define CREATOR_MULTI_NETWORK_STACK_PARENT_ADDITIONAL_INFO
- #define CREATOR_STACK_RF4CE_DATA
- #define CREATOR_STACK_RF4CE_PAIRING_TABLE
- #define CREATOR_STACK_GP_DATA
- #define CREATOR_STACK_GP_PROXY_TABLE
- #define CREATOR_STACK_GP_SINK_TABLE
- #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
- #define CREATOR_STACK_ADDITIONAL_CHILD_DATA

8.145.1 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`.

8.146 token-stack.h

```

00001
00059     #ifndef DEFINEADDRESSES
00060     #define TOKEN_NEXT_ADDRESS(region, address)
00061     #endif
00062
00063
00064
00065 // The basic TOKEN_DEF macro should not be used directly since the simplified
00066 // definitions are safer to use. For completeness of information, the basic
00067 // macro has the following format:
00068 //
00069 // TOKEN_DEF(name,creator,iscnt,isidx,type,arraysize,...)
00070 // name - The root name used for the token
00071 // creator - a "creator code" used to uniquely identify the token
00072 // iscnt - a bool flag that is set to identify a counter token
00073 // isidx - a bool flag that is set to identify an indexed token
00074 // type - the basic type or typdef of the token
00075 // arraysize - the number of elements making up an indexed token
00076 // ... - initializers used when resetting the tokens to default values
00077 //
00078 //
00079 // The following convenience macros are used to simplify the definition
00080 // process for commonly specified parameters to the basic TOKEN_DEF macro
00081 // DEFINE_BASIC_TOKEN(name, type, ...)
00082 // DEFINE_INDEXED_TOKEN(name, type, arraysize, ...)
00083 // DEFINE_COUNTER_TOKEN(name, type, ...)
00084 // DEFINE_FIXED_BASIC_TOKEN(name, type, address, ...)
00085 // DEFINE_FIXED_INDEXED_TOKEN(name, type, arraysize, address, ...)
00086 // DEFINE_FIXED_COUNTER_TOKEN(name, type, address, ...)
00087 // DEFINE_MFG_TOKEN(name, type, address, ...)
00088 //
00089
00097 #define DEFINE_BASIC_TOKEN(name, type, ...) \
00098     TOKEN_DEF(name, CREATOR_##name, 0, 0, type, 1, __VA_ARGS__)
00099
00100 #define DEFINE_COUNTER_TOKEN(name, type, ...) \
00101     TOKEN_DEF(name, CREATOR_##name, 1, 0, type, 1, __VA_ARGS__)
00102
00103 #define DEFINE_INDEXED_TOKEN(name, type, arraysize, ...) \

```

```

00104     TOKEN_DEF(name, CREATOR_##name, 0, 1, type, (arraysize), __VA_ARGS__)
00105
00106 #define DEFINE_FIXED_BASIC_TOKEN(name, type, address, ...) \
00107     TOKEN_NEXT_ADDRESS(name, (address)) \
00108     TOKEN_DEF(name, CREATOR_##name, 0, 0, type, 1, __VA_ARGS__)
00109
00110 #define DEFINE_FIXED_COUNTER_TOKEN(name, type, address, ...) \
00111     TOKEN_NEXT_ADDRESS(name, (address)) \
00112     TOKEN_DEF(name, CREATOR_##name, 1, 0, type, 1, __VA_ARGS__)
00113
00114 #define DEFINE_FIXED_INDEXED_TOKEN(name, type, arraysize, address, ...) \
00115     TOKEN_NEXT_ADDRESS(name, (address)) \
00116     TOKEN_DEF(name, CREATOR_##name, 0, 1, type, (arraysize), __VA_ARGS__)
00117
00118 #define DEFINE_MFG_TOKEN(name, type, address, ...) \
00119     TOKEN_NEXT_ADDRESS(name, (address)) \
00120     TOKEN_MFG(name, CREATOR_##name, 0, 0, type, 1, __VA_ARGS__)
00121
00125 // The Simulated EEPROM unit tests define all of their own tokens.
00126 #ifndef SIM_EEPROM_TEST
00127
00128 // The creator codes are here in one list instead of next to their token
00129 // definitions so comparision of the codes is easier. The only requirement
00130 // on these creator definitions is that they all must be unique. A favorite
00131 // method for picking creator codes is to use two ASCII characters inorder
00132 // to make the codes more memorable.
00133
00147 // STACK CREATORS
00148 #define CREATOR_STACK_NVDATA_VERSION 0xFF01
00149 #define CREATOR_STACK_BOOT_COUNTER 0xE263
00150 #define CREATOR_STACK_NONCE_COUNTER 0xE563
00151 #define CREATOR_STACK_ANALYSIS_REBOOT 0xE162
00152 #define CREATOR_STACK_KEYS 0xEB79
00153 #define CREATOR_STACK_NODE_DATA 0xEE64
00154 #define CREATOR_STACK_CLASSIC_DATA 0xE364
00155 #define CREATOR_STACK_ALTERNATE_KEY 0xE475
00156 #define CREATOR_STACKAPS_FRAME_COUNTER 0xE123
00157 #define CREATOR_STACK_TRUST_CENTER 0xE124
00158 #define CREATOR_STACK_NETWORK_MANAGEMENT 0xE125
00159 #define CREATOR_STACK_PARENT_INFO 0xE126
00160 #define CREATOR_STACK_PARENT_ADDITIONAL_INFO 0xE127
00161 // MULTI-NETWORK STACK CREATORS
00162 #define CREATOR_MULTI_NETWORK_STACK_KEYS 0xE210
00163 #define CREATOR_MULTI_NETWORK_STACK_NODE_DATA 0xE211
00164 #define CREATOR_MULTI_NETWORK_STACK_ALTERNATE_KEY 0xE212
00165 #define CREATOR_MULTI_NETWORK_STACK_TRUST_CENTER 0xE213
00166 #define CREATOR_MULTI_NETWORK_STACK_NETWORK_MANAGEMENT 0xE214
00167 #define CREATOR_MULTI_NETWORK_STACK_PARENT_INFO 0xE215
00168
00169 // Temporary solution for multi-network nwk counters: for now we define
00170 // the following counter which will be used on the network with index 1.
00171 #define CREATOR_MULTI_NETWORK_STACK_NONCE_COUNTER 0xE220
00172 #define CREATOR_MULTI_NETWORK_STACK_PARENT_ADDITIONAL_INFO 0xE221
00173
00174 // RF4CE stack tokens.
00175 #define CREATOR_STACK_RF4CE_DATA 0xE250
00176 #define CREATOR_STACK_RF4CE_PAIRING_TABLE 0xE251
00177
00178 // GP stack tokens.
00179 #define CREATOR_STACK_GP_DATA 0xE258
00180 #define CREATOR_STACK_GP_PROXY_TABLE 0xE259
00181 #define CREATOR_STACK_GP_SINK_TABLE 0xE25A
00182
00183 // APP CREATORS
00184 #define CREATOR_STACK_BINDING_TABLE 0xE274
00185 #define CREATOR_STACK_CHILD_TABLE 0xFF0D
00186 #define CREATOR_STACK_KEY_TABLE 0xE456
00187 #define CREATOR_STACK_CERTIFICATE_TABLE 0xE500
00188 #define CREATOR_STACK_ZLL_DATA 0xE501
00189 #define CREATOR_STACK_ZLL_SECURITY 0xE502
00190 #define CREATOR_STACK_ADDITIONAL_CHILD_DATA 0xE503
00191
00195
00196 // MANUFACTURING DATA
00197 // Since the manufacturing data is platform specific, we pull in the proper
00198 // file here.
00199 #if defined(CORTEXM3)
00200     // cortexm3 handles mfg tokens separately via mfg-token.h
00201 #elif defined(EMBER_TEST)
00202     #include "hal/micro/unix/simulation/token-manufacturing.h"

```

```

00203 #else
00204     #error no platform defined
00205 #endif
00206
00207
00209 // STACK DATA
00210 // *the addresses of these tokens must not change*
00211
00218 #define CURRENT_STACK_TOKEN_VERSION 0x03FC //MSB is version, LSB is complement
00219
00220 #ifdef DEFINETYPES
00221     typedef uint16_t tokTypeStackNvdataVersion;
00222     typedef uint32_t tokTypeStackBootCounter;
00223 //This was introduced to save the type of keep alives supported by the parent.
00224 //Bits 0 and 1 are currently in use. Bits 2-16 are reserved.
00225     typedef uint16_t tokTypeStackParentAdditionalInfo;
00226     typedef uint16_t tokTypeStackAnalysisReboot;
00227     typedef uint32_t tokTypeStackNonceCounter;
00228     typedef struct {
00229         uint8_t networkKey[16];
00230         uint8_t activeKeySeqNum;
00231     } tokTypeStackKeys;
00232     typedef struct {
00233         uint16_t panId;
00234         int8_t radioTxPower;
00235         uint8_t radioFreqChannel;
00236         uint8_t stackProfile;
00237         uint8_t nodeType;
00238         uint16_t zigbeeNodeId;
00239         uint8_t extendedPanId[8];
00240     } tokTypeStackNodeData;
00241     typedef struct {
00242         uint16_t mode;
00243         uint8_t eui64[8];
00244         uint8_t key[16];
00245     } tokTypeStackTrustCenter;
00246     typedef struct {
00247         uint32_t activeChannels;
00248         uint16_t managerNodeId;
00249         uint8_t updateId;
00250     } tokTypeStackNetworkManagement;
00251     typedef struct {
00252         uint8_t parentEui[8];
00253         uint16_t parentNodeId;
00254     } tokTypeStackParentInfo;
00255 #endif //DEFINETYPES
00256
00257 #ifdef DEFINETOKENS
00258 // The Stack tokens also need to be stored at well-defined locations
00259 // None of these addresses should ever change without extremely great care
00260 #define STACK_VERSION_LOCATION    128 // 2 bytes
00261 #define STACKAPS_NONCE_LOCATION   130 // 4 bytes
00262 #define STACK_ALT_NWK_KEY_LOCATION 134 // 17 bytes (key + sequence number)
00263 // reserved                      151   1 bytes
00264 #define STACK_BOOT_COUNT_LOCATION 152 // 2 bytes
00265 // reserved                      154   2 bytes
00266 #define STACK_NONCE_LOCATION      156 // 4 bytes
00267 // reserved                      160   1 bytes
00268 #define STACK_REBOOT_LOCATION    161 // 2 bytes
00269 // reserved                      163   7 bytes
00270 #define STACK_KEYS_LOCATION       170 // 17 bytes
00271 // reserved                      187   5 bytes
00272 #define STACK_NODE_DATA_LOCATION 192 // 16 bytes
00273 #define STACK_CLASSIC_LOCATION    208 // 26 bytes
00274 #define STACK_TRUST_CENTER_LOCATION 234 // 26 bytes
00275 // reserved                      260   8 bytes
00276 #define STACK_NETWORK_MANAGEMENT_LOCATION 268
00277                                         // 7 bytes
00278 #define STACK_PARENT_INFO_LOCATION 276 // 10bytes
00279 #define STACK_PARENT_ADDITIONAL_INFO_LOCATION 286
00280                                         // 2 bytes
00281 // reserved                      288   109 bytes
00282
00283 DEFINE_FIXED_BASIC_TOKEN(STACK_NVDATA_VERSION,
00284                             tokTypeStackNvdataVersion,
00285                             STACK_VERSION_LOCATION,
00286                             CURRENT_STACK_TOKEN_VERSION
00287 )
00287 DEFINE_FIXED_COUNTER_TOKEN(STACKAPS_FRAME_COUNTER,
00288                             tokTypeStackNonceCounter,

```

```

00289             STACK_APSS_NONCE_LOCATION,
00290             0x00000000)
00291 DEFINE_FIXED_BASIC_TOKEN(STACK_ALTERNATE_KEY,
00292             tokTypeStackKeys,
00293             STACK_ALT_NWK_KEY_LOCATION,
00294             {{0,}})
00295 DEFINE_FIXED_COUNTER_TOKEN(STACK_BOOT_COUNTER,
00296             tokTypeStackBootCounter,
00297             STACK_BOOT_COUNT_LOCATION,
00298             0x0000)
00299 DEFINE_FIXED_COUNTER_TOKEN(STACK_NONCE_COUNTER,
00300             tokTypeStackNonceCounter,
00301             STACK_NONCE_LOCATION,
00302             0x00000000)
00303 DEFINE_FIXED_BASIC_TOKEN(STACK_ANALYSIS_REBOOT,
00304             tokTypeStackAnalysisReboot,
00305             STACK_REBOOT_LOCATION,
00306             0x0000)
00307 DEFINE_FIXED_BASIC_TOKEN(STACK_KEYS,
00308             tokTypeStackKeys,
00309             STACK_KEYS_LOCATION,
00310             {{0,}})
00311 DEFINE_FIXED_BASIC_TOKEN(STACK_NODE_DATA,
00312             tokTypeStackNodeData,
00313             STACK_NODE_DATA_LOCATION,
00314             {0xFFFF, -1, 0, 0x00, 0x00, 0x0000})
00315 DEFINE_FIXED_BASIC_TOKEN(STACK_TRUST_CENTER,
00316             tokTypeStackTrustCenter,
00317             STACK_TRUST_CENTER_LOCATION,
00318             {0,})
00319 DEFINE_FIXED_BASIC_TOKEN(STACK_NETWORK_MANAGEMENT,
00320             tokTypeStackNetworkManagement,
00321             STACK_NETWORK_MANAGEMENT_LOCATION,
00322             {0, 0xFFFF, 0})
00323 DEFINE_FIXED_BASIC_TOKEN(STACK_PARENT_INFO,
00324             tokTypeStackParentInfo,
00325             STACK_PARENT_INFO_LOCATION,
00326             { {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF}, 0xFFFF})
00327 )
00327 DEFINE_FIXED_BASIC_TOKEN(STACK_PARENT_ADDITIONAL_INFO,
00328             tokTypeStackParentAdditionalInfo,
00329             STACK_PARENT_ADDITIONAL_INFO_LOCATION,
00330             0x0000)
00331
00332 #endif //DEFINETOKENS
00333
00334
00336 // PHY DATA
00337 #include "token-phy.h"
00338
00340 // MULTI-NETWORK STACK TOKENS: Tokens for the networks with index > 0.
00341 // The 0-index network info is stored in the usual tokens.
00342
00343 #ifdef DEFINETOKENS
00344 #if !defined(EMBER_MULTI_NETWORK_STRIPPED)
00345 #define EXTRA_NETWORKS_NUMBER (EMBER_SUPPORTED_NETWORKS - 1)
00346 DEFINE_INDEXED_TOKEN(MULTI_NETWORK_STACK_KEYS,
00347             tokTypeStackKeys,
00348             EXTRA_NETWORKS_NUMBER,
00349             {{0,}})
00350 DEFINE_INDEXED_TOKEN(MULTI_NETWORK_STACK_NODE_DATA,
00351             tokTypeStackNodeData,
00352             EXTRA_NETWORKS_NUMBER,
00353             {0,})
00354 DEFINE_INDEXED_TOKEN(MULTI_NETWORK_STACK_ALTERNATE_KEY,
00355             tokTypeStackKeys,
00356             EXTRA_NETWORKS_NUMBER,
00357             {{0,}})
00358 DEFINE_INDEXED_TOKEN(MULTI_NETWORK_STACK_TRUST_CENTER,
00359             tokTypeStackTrustCenter,
00360             EXTRA_NETWORKS_NUMBER,
00361             {0,})
00362 DEFINE_INDEXED_TOKEN(MULTI_NETWORK_STACK_NETWORK_MANAGEMENT
00363             tokTypeStackNetworkManagement,
00364             EXTRA_NETWORKS_NUMBER,
00365             {0,})
00366 DEFINE_INDEXED_TOKEN(MULTI_NETWORK_STACK_PARENT_INFO,
00367             tokTypeStackParentInfo,
00368             EXTRA_NETWORKS_NUMBER,

```

```

00369                               {{0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF}, 0xFFFF}
00370     )
00371 // Temporary solution for NWK counter token: the following is used for 1-index
00372 // network.
00372 DEFINE_COUNTER_TOKEN(MULTI_NETWORK_STACK_NONCE_COUNTER,
00373     tokTypeStackNonceCounter,
00374     0x00000000)
00375 DEFINE_INDEXED_TOKEN(
00376     MULTI_NETWORK_STACK_PARENT_ADDITIONAL_INFO,
00377     tokTypeStackParentAdditionalInfo,
00378     EXTRA_NETWORKS_NUMBER,
00379     {0,})
00379 #endif // EMBER_MULTI_NETWORK_STRIPPED
00380 #endif // DEFINETOKENS
00381
00382
00384 // APPLICATION DATA
00385 // *If a fixed application token is desired, its address must be above 384./*
00386
00387 #ifdef DEFINETYPES
00388     typedef uint8_t tokTypeStackBindingTable[13];
00389     typedef uint8_t tokTypeStackChildTable[11];
00390     typedef uint8_t tokTypeStackKeyTable[25];
00391     typedef uint8_t tokTypeStackAdditionalChildData;
00392 // Certificate Table Entry
00393 //   Certificate: 48-bytes
00394 //   CA Public Key: 22-bytes
00395 //   Private Key: 21-bytes
00396 //   Flags: 1-byte
00397 #define TOKEN_CERTIFICATE_TABLE_ENTRY_SIZE (48 + 22 + 21 + 1)
00398 #define TOKEN_CERTIFICATE_TABLE_ENTRY_FLAGS_INDEX
    (TOKEN_CERTIFICATE_TABLE_ENTRY_SIZE - 1)
00399     typedef uint8_t tokTypeStackCertificateTable[TOKEN_CERTIFICATE_TABLE_ENTRY_SIZE
];
00400 #endif //DEFINETYPES
00401
00402 // The following application tokens are required by the stack, but are sized by
00403 // the application via its CONFIGURATION_HEADER, which is why they are present
00404 // within the application data section. Any special application defined
00405 // tokens will follow.
00406 // NOTE: changing the size of these tokens within the CONFIGURATION_HEADER
00407 // WILL move automatically move any custom application tokens that are defined
00408 // in the APPLICATION_TOKEN_HEADER
00409 #ifdef DEFINETOKENS
00410 // Application tokens start at location 384 and are automatically positioned.
00411 TOKEN_NEXT_ADDRESS(APP, 384)
00412 DEFINE_INDEXED_TOKEN(STACK_BINDING_TABLE,
00413     tokTypeStackBindingTable,
00414     EMBER_BINDING_TABLE_TOKEN_SIZE
00415     ,
00416     {0,})
00416 DEFINE_INDEXED_TOKEN(STACK_CHILD_TABLE,
00417     tokTypeStackChildTable,
00418     EMBER_CHILD_TABLE_TOKEN_SIZE,
00419     {0,})
00420 DEFINE_INDEXED_TOKEN(STACK_KEY_TABLE,
00421     tokTypeStackKeyTable,
00422     EMBER_KEY_TABLE_TOKEN_SIZE,
00423     {0,})
00424 DEFINE_INDEXED_TOKEN(STACK_CERTIFICATE_TABLE,
00425     tokTypeStackCertificateTable,
00426     EMBER_CERTIFICATE_TABLE_SIZE,
00427     {0,})
00428 DEFINE_INDEXED_TOKEN(STACK_ADDITIONAL_CHILD_DATA,
00429     tokTypeStackAdditionalChildData,
00430     EMBER_CHILD_TABLE_TOKEN_SIZE,
00431     {0x0F,})
00432 #endif //DEFINETOKENS
00433
00434 // These must appear before the application header so that the token
00435 // numbering is consistent regardless of whether application tokens are
00436 // defined.
00437 // #if defined(EMBER_AF_PLUGIN_ZLL_LIBRARY) || defined(EMBER_TEST)
00438 //   #include "stack/zll/zll-token-config.h"
00439 // #endif
00440 // #if defined(EMBER_AF_PLUGIN_RF4CE_STACK_LIBRARY) || defined(EMBER_TEST)
00441 //   #include "stack/rf4ce/rf4ce-token-config.h"
00442 // #endif
00443 // #if defined(EMBER_AF_PLUGIN_GP_LIBRARY) || defined(EMBER_TEST)

```

```

00444 #include "stack/gp/gp-token-config.h"
00445 //endif
00446
00447 #ifdef APPLICATION_TOKEN_HEADER
00448     #include APPLICATION_TOKEN_HEADER
00449 #endif
00450
00451 //The tokens defined below are test tokens. They are normally not used by
00452 //anything but are left here as a convenience so test tokens do not have to
00453 //be recreated. If test code needs temporary, non-volatile storage, simply
00454 //uncomment and alter the set below as needed.
00455 //define CREATOR_TT01 1
00456 //define CREATOR_TT02 2
00457 //define CREATOR_TT03 3
00458 //define CREATOR_TT04 4
00459 //define CREATOR_TT05 5
00460 //define CREATOR_TT06 6
00461 //ifndef DEFINETYPES
00462 //typedef uint32_t tokTypeTT01;
00463 //typedef uint32_t tokTypeTT02;
00464 //typedef uint32_t tokTypeTT03;
00465 //typedef uint32_t tokTypeTT04;
00466 //typedef uint16_t tokTypeTT05;
00467 //typedef uint16_t tokTypeTT06;
00468 //endif //DEFINETYPES
00469 //ifndef DEFINETOKENS
00470 //define TT01_LOCATION 1
00471 //define TT02_LOCATION 2
00472 //define TT03_LOCATION 3
00473 //define TT04_LOCATION 4
00474 //define TT05_LOCATION 5
00475 //define TT06_LOCATION 6
00476 //DEFINE_FIXED_BASIC_TOKEN(TT01, tokTypeTT01, TT01_LOCATION, 0x0000)
00477 //DEFINE_FIXED_BASIC_TOKEN(TT02, tokTypeTT02, TT02_LOCATION, 0x0000)
00478 //DEFINE_FIXED_BASIC_TOKEN(TT03, tokTypeTT03, TT03_LOCATION, 0x0000)
00479 //DEFINE_FIXED_BASIC_TOKEN(TT04, tokTypeTT04, TT04_LOCATION, 0x0000)
00480 //DEFINE_FIXED_BASIC_TOKEN(TT05, tokTypeTT05, TT05_LOCATION, 0x0000)
00481 //DEFINE_FIXED_BASIC_TOKEN(TT06, tokTypeTT06, TT06_LOCATION, 0x0000)
00482 //endif //DEFINETOKENS
00483
00484
00485
00486 #else //SIM_EEPROM_TEST
00487
00488 //The Simulated EEPROM unit tests define all of their tokens via the
00489 //APPLICATION_TOKEN_HEADER macro.
00490 #ifdef APPLICATION_TOKEN_HEADER
00491     #include APPLICATION_TOKEN_HEADER
00492 #endif
00493
00494 #endif //SIM_EEPROM_TEST
00495
00496 #ifndef DEFINEADDRESSES
00497     #undef TOKEN_NEXT_ADDRESS
00498 #endif
00499

```

8.147 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)`

8.147.1 Detailed Description

Token system for storing non-volatile information. See [Tokens](#) for documentation.

Definition in file `token.h`.

8.148 token.h

```

00001
00222 #ifndef __TOKEN_H__
00223 #define __TOKEN_H__
00224
00225 #if defined(CORTEXM3)
00226     #ifdef MINIMAL_HAL
00227         #include "cortexm3/nvm-token.h"
00228         #include "cortexm3/mfg-token.h"
00229     #else //MINIMAL_HAL
00230         #include "cortexm3/token.h"
00231     #endif //MINIMAL_HAL
00232 #elif defined(C8051)
00233     #if defined(C8051_COBRA)
00234         #include "c8051/cobra/token.h"
00235     #else
00236         #include "c8051/silabs/token.h"
00237     #endif
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     uint16_t getTokenAddress(uint16_t creator);
00362     uint8_t getTokenSize(uint16_t creator );
00363     uint8_t getTokenArraySize(uint16_t creator);
00364 #endif //DOXYGEN_SHOULD_SKIP_THIS
00365

```

```

00366
00367 #endif // __TOKEN_H__
00368

```

8.149 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, uint16_t token, uint8_t index, uint8_t len)
- void **halInternalSetTokenData** (uint16_t token, uint8_t index, void *data, uint8_t len)
- void **halInternalIncrementCounterToken** (uint8_t token)
- void **halInternalGetIdxTokenPtr** (void *ptr, uint16_t ID, uint8_t index, uint8_t len)

Variables

- const uint16_t **tokenCreators** []
- const bool **tokenIsCnt** []
- const uint8_t **tokenSize** []
- const uint8_t **tokenArraySize** []
- const void *const **tokenDefaults** []

8.149.1 Detailed Description

Cortex-M3 Token system for storing non-volatile information. See [Tokens](#) for documentation. DOXYGEN 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](#).

8.149.2 Macro Definition Documentation

8.149.2.1 #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 [39](#) of file [cortexm3/token.h](#).

8.149.2.2 #define DEFINETOKENS

Definition at line [46](#) of file [cortexm3/token.h](#).

8.149.2.3 #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

<i>name,:</i>	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

<i>name,:</i>	The name of the token.
<i>type,:</i>	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

<i>name,:</i>	The name of the token.
<i>type,:</i>	The token type. The types are found in token-stack.h .

Definition at line 163 of file [cortexm3/token.h](#).

8.149.2.4 #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

<i>name,:</i>	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

<i>name,:</i>	The name of the token.
<i>type,:</i>	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

<i>name,:</i>	The name of the token.
<i>type,:</i>	The token type. The types are found in token-stack.h .

Definition at line 163 of file [cortexm3/token.h](#).

8.149.2.5 #define COUNTER_TOKEN_PAD**Description:**

A define for the token and Simulated EEPROM system that specifies, in bytes, the space allocated to a counter token for +1 marks. The number of +1 marks varies between chips based on the minimum write granularity for a chip's flash. EM35x chips can use 8bit per +1 while EFM32/EZM32/EZR32 chips use 16bit per +1.

Definition at line 149 of file [cortexm3/token.h](#).

8.149.2.6 #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

<i>name,:</i>	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

<i>name,:</i>	The name of the token.
<i>type,:</i>	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

<i>name,:</i>	The name of the token.
<i>type,:</i>	The token type. The types are found in token-stack.h .

Definition at line 163 of file [cortexm3/token.h](#).

8.149.2.7 #define halCommonGetToken(data, token)

Definition at line 236 of file [cortexm3/token.h](#).

8.149.2.8 #define halCommonGetIndexedToken(data, token, index)

Definition at line 239 of file [cortexm3/token.h](#).

8.149.2.9 #define halStackGetIndexedToken(data, token, index, size)

Definition at line 242 of file [cortexm3/token.h](#).

8.149.2.10 #define halStackGetIdxTokenPtrOrData(*ptr*, *token*, *index*)

Definition at line [245](#) of file [cortexm3/token.h](#).

8.149.2.11 #define halCommonSetToken(*token*, *data*)

Definition at line [249](#) of file [cortexm3/token.h](#).

8.149.2.12 #define halCommonSetIndexedToken(*token*, *index*, *data*)

Definition at line [252](#) of file [cortexm3/token.h](#).

8.149.2.13 #define halStackSetIndexedToken(*token*, *index*, *data*, *size*)

Definition at line [255](#) of file [cortexm3/token.h](#).

8.149.2.14 #define halCommonIncrementCounterToken(*token*)

Definition at line [258](#) of file [cortexm3/token.h](#).

8.149.3 Enumeration Type Documentation

8.149.3.1 anonymous enum

Enumerator:

TOKEN_COUNT

Definition at line [62](#) of file [cortexm3/token.h](#).

8.149.3.2 anonymous enum

Definition at line [79](#) of file [cortexm3/token.h](#).

8.149.4 Function Documentation

8.149.4.1 void halInternalGetTokenData (void * *data*, uint16_t *token*, uint8_t *index*, uint8_t *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

<i>data,:</i>	A pointer to where the data being read should be placed.
<i>token,:</i>	The name of the token to get data from. On this platform that name is defined as an address.
<i>index,:</i>	The index to access. If the token being accessed is not an indexed token, this parameter is set by the API to be 0.
<i>len,:</i>	The length of the token being worked on. This value is automatically set by the API to be the size of the token.

8.149.4.2 void halInternalSetTokenData (uint16_t *token*, uint8_t *index*, void * *data*, uint8_t *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

<i>token,:</i>	The name of the token to get data from. On this platform that name is defined as an address.
<i>index,:</i>	The index to access. If the token being accessed is not an indexed token, this parameter is set by the API to be 0.
<i>data,:</i>	A pointer to the data being written.
<i>len,:</i>	The length of the token being worked on. This value is automatically set by the API to be the size of the token.

8.149.4.3 void halInternalIncrementCounterToken (uint8_t *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

<i>token,:</i>	The name of the token.
----------------	------------------------

8.149.4.4 void hallInternalGetIdxTokenPtr (void * *ptr*, uint16_t *ID*, uint8_t *index*, uint8_t *len*)

8.149.5 Variable Documentation

8.149.5.1 const uint16_t 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

<i>creator,:</i>	The creator code type. The codes are found in token-stack.h .
------------------	---

8.149.5.2 const bool 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

<i>iscnt,:</i>	The flag indicating if the token is a counter. The iscnt's are found in token-stack.h .
----------------	---

8.149.5.3 const uint8_t 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

<i>type,::</i>	The token type. The types are found in token-stack.h .
----------------	--

8.149.5.4 const uint8_t 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

<i>arraysize,::</i>	The array size.
---------------------	-----------------

8.149.5.5 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: `uint8_t byte = *((((uint8_t *)tokenDefaults[i])+(n)`

Parameters

TOKE- N_	#name##_D- EFAULTS,::	A constant declaration of the token default values, generated for all tokens.
---------------------	----------------------------------	--

8.150 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 #if CORTEXM3_EFM32_MICRO
00024 // The manufacturing tokens live outside the Simulated EEPROM. This means
00025 // they are defined differently which is covered in mfg-token.h
00026 #include "efm32/mfg-token.h"
00027 #else
00028 // The manufacturing tokens live in the Info Blocks, while all other tokens
00029 // live in the Simulated EEPROM. This means they are defined differently,
00030 // which is covered in mfg-token.h
00031 #include "mfg-token.h"
00032 #endif
00033
00034 //-- Build structure defines
00035 #define DEFINETYPES
00036 #include "stack/config/token-stack.h"

```

```

00041 #undef DEFINETYPES
00042
00043
00044
00045 //-- Build parameter links
00046 #define DEFINETOKENS
00047
00048 #undef TOKEN_DEF
00049
00060 #define TOKEN_DEF(name,creator,iscnt,isisdx,type,arraysize,...) \
00061     TOKEN_##name,
00062     enum{
00063         #include "stack/config/token-stack.h"
00064         TOKEN_COUNT
00065     };
00066 #undef TOKEN_DEF
00067
00068
00077 #define TOKEN_DEF(name,creator,iscnt,isisdx,type,arraysize,...) \
00078     TOKEN_##name##_SIZE = sizeof(type),
00079     enum {
00080         #include "stack/config/token-stack.h"
00081     };
00082 #undef TOKEN_DEF
00083
00084
00094 extern const uint16_t tokenCreators[];
00095
00105 extern const bool tokenIsCnt[];
00106
00115 extern const uint8_t tokenSize[];
00116
00125 extern const uint8_t tokenArraySize[];
00126
00140 extern const void * const tokenDefaults[];
00141
00149 #define COUNTER_TOKEN_PAD      50
00150
00151
00152
00163 #define TOKEN_DEF(name,creator,iscnt,isisdx,type,arraysize,...) \
00164     typedef type TOKEN_##name##_TYPE;
00165     #include "stack/config/token-stack.h"
00166 #undef TOKEN_DEF
00167
00168 #undef DEFINETOKENS
00169
00170
00192 void halInternalGetTokenData(void *data, uint16_t token,
00193     uint8_t index, uint8_t len);
00193
00215 void halInternalSetTokenData(uint16_t token, uint8_t
00216     index, void *data, uint8_t len);
00228 void halInternalIncrementCounterToken(uint8_t
00229     token);
00230
00231 // See hal/micro/token.h for the full explanation of the token API as
00232 // instantiated below.
00233
00234 //These defines Link the public API to the private internal instance.
00235
00236 #define halCommonGetToken( data, token )           \
00237     halInternalGetTokenData(data, token, 0x7F, token##_SIZE)
00238
00239 #define halCommonGetIndexedToken( data, token, index ) \
00240     halInternalGetTokenData(data, token, index, token##_SIZE)
00241
00242 #define halStackGetIndexedToken( data, token, index, size ) \
00243     halInternalGetTokenData(data, token, index, size)
00244
00245 #define halStackGetIdxTokenPtrOrData( ptr, token, index ) \
00246     halInternalGetIdxTokenPtr(ptr, token, index, token##_SIZE)
00247 void halInternalGetIdxTokenPtr(void *ptr, uint16_t ID,
00248     uint8_t index, uint8_t len);
00249
00249 #define halCommonSetToken( token, data )           \
00250     halInternalSetTokenData(token, 0x7F, data, token##_SIZE)
00251

```

```

00252 #define halCommonSetIndexedToken( token, index, data )           \
00253     halInternalSetTokenData(token, index, data, token##_SIZE)
00254
00255 #define halStackSetIndexedToken( token, index, data, size ) \
00256     halInternalSetTokenData(token, index, data, size)
00257
00258 #define halCommonIncrementCounterToken( token )                  \
00259     halInternalIncrementCounterToken(token);
00260
00261 // For use only by the EZSP UART protocol
00262 #ifdef EZSP_UART
00263     #ifdef CORTEXM3_EMBER_MICRO
00264         #define halInternalMfgTokenPointer( address ) \
00265             ((const void *) (address + DATA_BIG_INFO_BASE))
00266         #define halInternalMfgIndexedToken( type, address, index ) \
00267             (*((const type *) (address + DATA_BIG_INFO_BASE) + index))
00268     #endif
00269     #ifdef CORTEXM3_EFM32_MICRO
00270         #define halInternalMfgTokenPointer( address ) \
00271             ((const void *) (USERDATA_BASE | (address&0x0FFF)))
00272         #define halInternalMfgIndexedToken( type, address, index ) \
00273             (*((const type *) (USERDATA_BASE | (address&0x0FFF)) + index))
00274     #endif
00275 #endif
00276
00277
00278 #undef TOKEN_MFG
00279
00280 #endif // __PLAT_TOKEN_H__
00281

```

8.151 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 targetLong, EmberKeyData *newKey)
- EmberStatus emberBroadcastNetworkKeySwitch (void)
- EmberJoinDecision emberTrustCenterJoinHandler (EmberNodeId newNodeId, EmberEUI64 newNodeEui64, EmberDeviceUpdate status, EmberNodeId parentOfNewNode)
- EmberStatus emberBecomeTrustCenter (EmberKeyData *newNetworkKey)
- EmberStatus emberSendRemoveDevice (EmberNodeId destShort, EmberEUI64 destLong, EmberEUI64 deviceToRemoveLong)

Variables

- EmberLinkKeyRequestPolicy emberTrustCenterLinkKeyRequestPolicy
- EmberLinkKeyRequestPolicy emberAppLinkKeyRequestPolicy

8.151.1 Detailed Description

EmberZNet security API See [Security](#) for documentation.

Definition in file [trust-center.h](#).

8.152 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 EmberStatus emberSendRemoveDevice(EmberNodeId
    destShort,

```

```

00243     EmberEUI64 destLong,
00244     EmberEUI64 deviceToRemoveLong);
00245
00246 // @} END addtogroup
00248

```

8.153 zigbee-device-common.h File Reference

Macros

- #define ZDO_MESSAGE_OVERHEAD

Service Discovery Functions

- EmberStatus emberNodeDescriptorRequest (EmberNodeId target, EmberApsOption options)
- EmberStatus emberPowerDescriptorRequest (EmberNodeId target, EmberApsOption options)
- EmberStatus emberSimpleDescriptorRequest (EmberNodeId target, uint8_t targetEndpoint, EmberApsOption options)
- EmberStatus emberActiveEndpointsRequest (EmberNodeId target, EmberApsOption options)

Binding Manager Functions

- EmberStatus emberBindRequest (EmberNodeId target, EmberEUI64 source, uint8_t sourceEndpoint, uint16_t clusterId, uint8_t type, EmberEUI64 destination, EmberMulticastId groupAddress, uint8_t destinationEndpoint, EmberApsOption options)
- EmberStatus emberUnbindRequest (EmberNodeId target, EmberEUI64 source, uint8_t sourceEndpoint, uint16_t clusterId, uint8_t type, EmberEUI64 destination, EmberMulticastId groupAddress, uint8_t destinationEndpoint, EmberApsOption options)

Node Manager Functions

- EmberStatus emberLqiTableRequest (EmberNodeId target, uint8_t startIndex, EmberApsOption options)
- EmberStatus emberRoutingTableRequest (EmberNodeId target, uint8_t startIndex, EmberApsOption options)
- EmberStatus emberBindingTableRequest (EmberNodeId target, uint8_t startIndex, EmberApsOption options)
- EmberStatus emberLeaveRequest (EmberNodeId target, EmberEUI64 deviceAddress, uint8_t leaveRequestFlags, EmberApsOption options)
- EmberStatus emberPermitJoiningRequest (EmberNodeId target, uint8_t duration, uint8_t authentication, EmberApsOption options)
- void emberSetZigDevRequestRadius (uint8_t radius)
- uint8_t emberGetZigDevRequestRadius (void)
- uint8_t emberGetLastZigDevRequestSequence (void)
- uint8_t emberGetLastAppZigDevRequestSequence (void)

8.153.1 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`.

8.154 zigbee-device-common.h

```

00001
00016 #define ZDO_MESSAGE_OVERHEAD 1
00017
00036 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00037 EmberStatus emberNodeDescriptorRequest(
00038     EmberNodeId target,
00039                             EmberApsOption options);
00039 #else
00040 // Macroized to save code space.
00041 EmberStatus emberSendZigDevRequestTarget(EmberNodeId
00042     target,
00043                             uint16_t clusterId,
00044                             EmberApsOption options);
00044 #define emberNodeDescriptorRequest(target, opts) \
00045 (emberSendZigDevRequestTarget((target), NODE_DESCRIPTOR_REQUEST, (opts)))
00046 #endif
00047
00063 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00064 EmberStatus emberPowerDescriptorRequest(
00065     EmberNodeId target,
00066                             EmberApsOption options);
00066 #else
00067 // Macroized to save code space.
00068 #define emberPowerDescriptorRequest(target, opts) \
00069 (emberSendZigDevRequestTarget((target), POWER_DESCRIPTOR_REQUEST, (opts)))
00070 #endif
00071
00090 EmberStatus emberSimpleDescriptorRequest
00091     (EmberNodeId target,
00092      uint8_t targetEndpoint,
00093      EmberApsOption options);
00093
00106 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00107 EmberStatus emberActiveEndpointsRequest(
00108     EmberNodeId target,
00109                             EmberApsOption options);
00109 #else
00110 // Macroized to save code space.
00111 #define emberActiveEndpointsRequest(target, opts) \
00112 (emberSendZigDevRequestTarget((target), ACTIVE_ENDPOINTS_REQUEST, (opts)))
00113 #endif
00114
00144 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00145 EmberStatus emberBindRequest(EmberNodeId
00146     target,
00147         EmberEUI64 source,
00148         uint8_t sourceEndpoint,
00149         uint16_t clusterId,
00149         uint8_t type,
00150         EmberEUI64 destination,
00151         EmberMulticastId groupAddress,
00152         uint8_t 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 ,
00174             uint16_t bindClusterId,
00175             EmberEUI64 source,
00176             uint8_t sourceEndpoint,
00177             uint16_t clusterId,
00178             uint8_t type,
00179             EmberEUI64 destination,
00180             EmberMulticastId
00181             groupAddress,
00182             uint8_t destinationEndpoint,
00183             EmberApsOption options);
00184 #endif
00185
00186 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00187 EmberStatus emberUnbindRequest(EmberNodeId
00188             target,
00189             EmberEUI64 source,
00190             uint8_t sourceEndpoint,
00191             uint16_t clusterId,
00192             uint8_t type,
00193             EmberEUI64 destination,
00194             EmberMulticastId groupAddress,
00195             uint8_t destinationEndpoint,
00196             EmberApsOption options);
00197
00198 #else
00199 // Macroized to save code space.
00200 #define emberUnbindRequest(target,
00201             src,
00202             srcEndpt,
00203             cluster,
00204             type,
00205             dest,
00206             groupAddress,
00207             destEndpt,
00208             opts)
00209
00210 (emberSendZigDevBindRequest((target),
00211             UNBIND_REQUEST,
00212             (src), (srcEndpt), (cluster),
00213             (type), (dest), (groupAddress),
00214             (destEndpt), (opts)))
00215
00216 #endif
00217
00218 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00219 EmberStatus emberLqiTableRequest(EmberNodeId
00220             target,
00221             uint8_t startIndex,
00222             EmberApsOption options);
00223
00224 #else
00225 #define emberLqiTableRequest(target, startIndex, options) \
00226     (emberTableRequest(LQI_TABLE_REQUEST, (target), (startIndex), (options)))
00227
00228 EmberStatus emberTableRequest(uint16_t clusterId,
00229             EmberNodeId target,
00230             uint8_t startIndex,
00231             EmberApsOption options);
00232
00233 #endif
00234
00235 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00236 EmberStatus emberRoutingTableRequest(
00237             EmberNodeId target,
00238             uint8_t startIndex,
00239             EmberApsOption options);
00240
00241 #else
00242 #define emberRoutingTableRequest(target, startIndex, options) \
00243     (emberTableRequest(ROUTING_TABLE_REQUEST, (target), (startIndex), (options)))
00244
00245 #endif
00246
00247 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00248 EmberStatus emberBindingTableRequest(
00249             EmberNodeId target,
00250             uint8_t startIndex,
00251             EmberApsOption options);
00252
00253
00254
00255
00256
00257
00258
00259
00260
00261
00262
00263
00264
00265
00266
00267
00268
00269
00270
00271
00272
00273
00274
00275
00276
00277
00278
00279
00280
00281
00282
00283
00284
00285
00286
00287
00288
00289
00290
00291
00292
00293
00294
00295
00296
00297
00298
00299
00300
00301
00302
00303
00304
00305
00306
00307
00308
00309
00310
00311
00312
00313
00314
00315
00316
00317
00318
00319
00320
00321
00322
00323
00324
00325
00326
00327
00328
00329
00330
00331
00332
00333
00334
00335
00336
00337
00338
00339
00340
00341
00342
00343
00344
00345
00346
00347
00348
00349
00350
00351
00352
00353
00354
00355
00356
00357
00358
00359
00360
00361
00362
00363
00364
00365
00366
00367
00368
00369
00370
00371
00372
00373
00374
00375
00376
00377
00378
00379
00380
00381
00382
00383
00384
00385
00386
00387
00388
00389
00390
00391
00392
00393
00394
00395
00396
00397
00398
00399
00400
00401
00402
00403
00404
00405
00406
00407
00408
00409
00410
00411
00412
00413
00414
00415
00416
00417
00418
00419
00420
00421
00422
00423
00424
00425
00426
00427
00428
00429
00430
00431
00432
00433
00434
00435
00436
00437
00438
00439
00440
00441
00442
00443
00444
00445
00446
00447
00448
00449
00450
00451
00452
00453
00454
00455
00456
00457
00458
00459
00460
00461
00462
00463
00464
00465
00466
00467
00468
00469
00470
00471
00472
00473
00474
00475
00476
00477
00478
00479
00480
00481
00482
00483
00484
00485
00486
00487
00488
00489
00490
00491
00492
00493
00494
00495
00496
00497
00498
00499
00500
00501
00502
00503
00504
00505
00506
00507
00508
00509
00510
00511
00512
00513
00514
00515
00516
00517
00518
00519
00520
00521
00522
00523
00524
00525
00526
00527
00528
00529
00530
00531
00532
00533
00534
00535
00536
00537
00538
00539
00540
00541
00542
00543
00544
00545
00546
00547
00548
00549
00550
00551
00552
00553
00554
00555
00556
00557
00558
00559
00560
00561
00562
00563
00564
00565
00566
00567
00568
00569
00570
00571
00572
00573
00574
00575
00576
00577
00578
00579
00580
00581
00582
00583
00584
00585
00586
00587
00588
00589
00590
00591
00592
00593
00594
00595
00596
00597
00598
00599
00600
00601
00602
00603
00604
00605
00606
00607
00608
00609
00610
00611
00612
00613
00614
00615
00616
00617
00618
00619
00620
00621
00622
00623
00624
00625
00626
00627
00628
00629
00630
00631
00632
00633
00634
00635
00636
00637
00638
00639
00640
00641
00642
00643
00644
00645
00646
00647
00648
00649
00650
00651
00652
00653
00654
00655
00656
00657
00658
00659
00660
00661
00662
00663
00664
00665
00666
00667
00668
00669
00670
00671
00672
00673
00674
00675
00676
00677
00678
00679
00680
00681
00682
00683
00684
00685
00686
00687
00688
00689
00690
00691
00692
00693
00694
00695
00696
00697
00698
00699
00700
00701
00702
00703
00704
00705
00706
00707
00708
00709
00710
00711
00712
00713
00714
00715
00716
00717
00718
00719
00720
00721
00722
00723
00724
00725
00726
00727
00728
00729
00730
00731
00732
00733
00734
00735
00736
00737
00738
00739
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749
00750
00751
00752
00753
00754
00755
00756
00757
00758
00759
00760
00761
00762
00763
00764
00765
00766
00767
00768
00769
00770
00771
00772
00773
00774
00775
00776
00777
00778
00779
00780
00781
00782
00783
00784
00785
00786
00787
00788
00789
00790
00791
00792
00793
00794
00795
00796
00797
00798
00799
00800
00801
00802
00803
00804
00805
00806
00807
00808
00809
00810
00811
00812
00813
00814
00815
00816
00817
00818
00819
00820
00821
00822
00823
00824
00825
00826
00827
00828
00829
00830
00831
00832
00833
00834
00835
00836
00837
00838
00839
00840
00841
00842
00843
00844
00845
00846
00847
00848
00849
00850
00851
00852
00853
00854
00855
00856
00857
00858
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00879
00880
00881
00882
00883
00884
00885
00886
00887
00888
00889
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999
01000
01001
01002
01003
01004
01005
01006
01007
01008
01009
01010
01011
01012
01013
01014
01015
01016
01017
01018
01019
01020
01021
01022
01023
01024
01025
01026
01027
01028
01029
01030
01031
01032
01033
01034
01035
01036
01037
01038
01039
01040
01041
01042
01043
01044
01045
01046
01047
01048
01049
01050
01051
01052
01053
01054
01055
01056
01057
01058
01059
01060
01061
01062
01063
01064
01065
01066
01067
01068
01069
01070
01071
01072
01073
01074
01075
01076
01077
01078
01079
01080
01081
01082
01083
01084
01085
01086
01087
01088
01089
01090
01091
01092
01093
01094
01095
01096
01097
01098
01099
01100
01101
01102
01103
01104
01105
01106
01107
01108
01109
01110
01111
01112
01113
01114
01115
01116
01117
01118
01119
01120
01121
01122
01123
01124
01125
01126
01127
01128
01129
01130
01131
01132
01133
01134
01135
01136
01137
01138
01139
01140
01141
01142
01143
01144
01145
01146
01147
01148
01149
01150
01151
01152
01153
01154
01155
01156
01157
01158
01159
01160
01161
01162
01163
01164
01165
01166
01167
01168
01169
01170
01171
01172
01173
01174
01175
01176
01177
01178
01179
01180
01181
01182
01183
01184
01185
01186
01187
01188
01189
01190
01191
01192
01193
01194
01195
01196
01197
01198
01199
01200
01201
01202
01203
01204
01205
01206
01207
01208
01209
01210
01211
01212
01213
01214
01215
01216
01217
01218
01219
01220
01221
01222
01223
01224
01225
01226
01227
01228
01229
01230
01231
01232
01233
01234
01235
01236
01237
01238
01239
01240
01241
01242
01243
01244
01245
01246
01247
01248
01249
01250
01251
01252
01253
01254
01255
01256
01257
01258
01259
01260
01261
01262
01263
01264
01265
01266
01267
01268
01269
01270
01271
01272
01273
01274
01275
01276
01277
01278
01279
01280
01281
01282
01283
01284
01285
01286
01287
01288
01289
01290
01291
01292
01293
01294
01295
01296
01297
01298
01299
01300
01301
01302
01303
01304
01305
01306
01307
01308
01309
01310
01311
01312
01313
01314
01315
01316
01317
01318
01319
01320
01321
01322
01323
01324
01325
01326
01327
01328
01329
01330
01331
01332
01333
01334
01335
01336
01337
01338
01339
01340
01341
01342
01343
01344
01345
01346
01347
01348
01349
01350
01351
01352
01353
01354
01355
01356
01357
01358
01359
01360
01361
01362
01363
01364
01365
01366
01367
01368
01369
01370
01371
01372
01373
01374
01375
01376
01377
01378
01379
01380
01381
01382
01383
01384
01385
01386
01387
01388
01389
01390
01391
01392
01393
01394
01395
01396
01397
01398
01399
01400
01401
01402
01403
01404
01405
01406
01407
01408
01409
01410
01411
01412
01413
01414
01415
01416
01417
01418
01419
01420
01421
01422
01423
01424
01425
01426
01427
01428
01429
01430
01431
01432
01433
01434
01435
01436
01437
01438
01439
01440
01441
01442
01443
01444
01445
01446
01447
01448
01449
01450
01451
01452
01453
01454
01455
01456
01457
01458
01459
01460
01461
01462
01463
01464
01465
01466
01467
01468
01469
01470
01471
01472
01473
01474
01475
01476
01477
01478
01479
01480
01481
01482
01483
01484
01485
01486
01487
01488
01489
01490
01491
01492
01493
01494
01495
01496
01497
01498
01499
01500
01501
01502
01503
01504
01505
01506
01507
01508
01509
01510
01511
01512
01513
01514
01515
01516
01517
01518
01519
01520
01521
01522
01523
01524
01525
01526
01527
01528
01529
01530
01531
01532
01533
01534
01535
01536
01537
01538
01539
01540
01541
01542
01543
01544
01545
01546
01547
01548
01549
01550
01551
01552
01553
01554
01555
01556
01557
01558
01559
01560
01561
01562
01563
01564
01565
01566
01567
01568
01569
01570
01571
01572
01573
01574
01575
01576
01577
01578
01579
01580
01581
01582
01583
01584
01585
01586
01587
01588
01589
01590
01591
01592
01593
01594
01595
01596
01597
01598
01599
01600
01601
01602
01603
01604
01605
01606
01607
01608
01609
01610
01611
01612
01613
01614
01615
01616
01617
01618
01619
01620
01621
01622
01623
01624
01625
01626
01627
01628
01629
01630
01631
01632
01633
01634
01635
01636
01637
01638
01639
01640
01641
01642
01643
01644
01645
01646
01647
01648
01649
01650
01651
01652
01653
01654
01655
01656
01657
01658
01659
01660
01661
01662
01663
01664
01665
01666
01667
01668
01669
01670
01671
01672
01673
01674
01675
01676
01677
01678
01679
01680
01681
01682
01683
01684
01685
01686
01687
01688
01689
01690
01691
01692
01693
01694
01695
01696
01697
01698
01699
01700
01701
01702
01703
01704
01705
01706
01707
01708
01709
01710
01711
01712
01713
01714
01715
01716
01717
01718
01719
01720
01721
01722
01723
01724
01725
01726
01727
01728
01729
01730
01731
01732
01733
01734
01735
01736
01737
01738
01739
01740
01741
01742
01743
01744
01745
01746
01747
01748
01749
01750
01751
01752
01753
01754
01755
01756
01757
01758
01759
01760
01761
01762
01763
01764
01765
01766
01767
01768
01769
01770
01771
01772
01773
01774
01775
01776
01777
01778
01779
01780
01781
01782
01783
01784
01785
01786
01787
01788
01789
01790
01791
01792
01793
01794
01795
01796
01797
01798
01799
01800
01801
01802
01803
01804
01805
01806
01807
01808
01809
01810
01811
01812
01813
01814
01815
01816
01817
01818
01819
01820
01821
01822
01823
01824
01825
01826
01827
01828
01829
01830
01831
01832
01833
01834
01835
01836
01837
01838
01839
01840
01841
01842
01843
01844
01845
01846
01847
01848
01849
01850
01851
01852
01853
01854
01855
01856
01857
01858
01859
01860
01861
01862
01863
01864
01865
01866
01867
01868
01869
01870
01871
01872
01873
01874
01875
01876
01877
01878
01879
01880
01881
01882
01883
01884
01885
01886
01887
01888
01889
01890
01891
01892
01893
01894
01895
01896
01897
01898
01899
01900
01901
01902
01903
01904
01905
01906
01907
01908
01909
01910
01911
01912
01913
01914
01915
01916
01917
01918
01919
01920
01921
01922
01923
01924
01925
01926
01927
01928
01929
01930
01931
01932
01933
01934
01935
01936
01937
01938
01939
01940
01941
01942
01943
01944
01945
01946
01947
01948
01949
01950
01951
01952
01953
01954
01955
01956
01957
01958
01959
01960
01961
01962
01963
01964
01965
01966
01967
01968
01969
01970
01971
01972
01973
01974
01975
01976
01
```

```

00321 #else
00322 #define emberBindingTableRequest(target, startIndex, options) \
00323     (emberTableRequest(BINDING_TABLE_REQUEST, (target), (startIndex), (options)))
00324 #endif
00325
00345 EmberStatus emberLeaveRequest(EmberNodeId
00346             target,
00347                     EmberEUI64 deviceAddress,
00348                     uint8_t leaveRequestFlags,
00349                     EmberApsOption options);
00349
00366 EmberStatus emberPermitJoiningRequest(
00367             EmberNodeId target,
00368                     uint8_t duration,
00369                     uint8_t authentication,
00370                     EmberApsOption options);
00370
00371 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00372
00377 void emberSetZigDevRequestRadius(uint8_t radius);
00378
00384 uint8_t emberGetZigDevRequestRadius(void);
00392 uint8_t emberGetLastZigDevRequestSequence(void
00393 );
00393 #else
00394 extern uint8_t zigDevRequestRadius;
00395 #define emberGetZigDevRequestRadius() (zigDevRequestRadius)
00396 #define emberSetZigDevRequestRadius(x) (zigDevRequestRadius=x)
00397 #define emberGetLastZigDevRequestSequence() \
00398     (emberGetLastAppZigDevRequestSequence())
00399 #endif
00400
00407 uint8_t emberGetLastAppZigDevRequestSequence
00408     (void);
00408
00411 #ifndef DOXYGEN_SHOULD_SKIP_THIS
00412 // -----
00413 // Utility functions used by the library code.
00414
00415 EmberStatus emberSendZigDevRequest(EmberNodeId
00416             destination,
00417                     uint16_t clusterId,
00418                     EmberApsOption options,
00419                     uint8_t *contents,
00420                     uint8_t length);
00420
00430 uint8_t emberNextZigDevRequestSequence(void);
00431
00432 #endif // DOXYGEN_SHOULD_SKIP_THIS
00433

```

8.155 zigbee-device-host.h File Reference

Device Discovery Functions

- `EmberStatus emberNetworkAddressRequest (EmberEUI64 target, bool reportKids, uint8_t childStartIndex)`
- `EmberStatus emberIeeeAddressRequest (EmberNodeId target, bool reportKids, uint8_t childStartIndex, EmberApsOption options)`

Service Discovery Functions

- `EmberStatus ezspMatchDescriptorsRequest (EmberNodeId target, uint16_t profile, uint8_t inCount, uint8_t outCount, uint16_t *inClusters, uint16_t *outClusters, EmberApsOption options)`

Binding Manager Functions

- `EmberStatus ezspEndDeviceBindRequest (EmberNodeId localNodeId, EmberEUI64 localEui64, uint8_t endpoint, uint16_t profile, uint8_t inCount, uint8_t outCount, uint16_t *inClusters, uint16_t *outClusters, EmberApsOption options)`

Function to Decode Address Response Messages

- `EmberNodeId ezspDecodeAddressResponse (uint8_t *response, EmberEUI64 eui64Return)`

8.155.1 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](#).

8.156 zigbee-device-host.h

```

00001
00104 EmberStatus emberNetworkAddressRequest(
00105     EmberEUI64 target,
00106             bool reportKids,
00107             uint8_t childStartIndex);
00107
00125 EmberStatus emberIeeeAddressRequest (
00126     EmberNodeId target,
00127             bool reportKids,
00128             uint8_t childStartIndex,
00129             EmberApsOption options);
00157 EmberStatus ezspMatchDescriptorsRequest (
00158     EmberNodeId target,
00159             uint16_t profile,
00160             uint8_t inCount,
00161             uint8_t outCount,
00162             uint16_t *inClusters,
00163             uint16_t *outClusters,
00164             EmberApsOption options);
00189 EmberStatus ezspEndDeviceBindRequest (
00190     EmberNodeId localNodeId,
00191             EmberEUI64 localEui64,
00192             uint8_t endpoint,
00193             uint16_t profile,
00194             uint8_t inCount,
00195             uint8_t outCount,
00196             uint16_t *inClusters,
00197             uint16_t *outClusters,
00198             EmberApsOption options);
00216 EmberNodeId ezspDecodeAddressResponse (
00217     uint8_t *response,
00218             EmberEUI64 eui64Return);
00218

```

8.157 zigbee-device-library.h File Reference

Service Discovery Functions

- `EmberStatus emberMatchDescriptorsRequest (EmberNodeId target, uint16_t profile, EmberMessageBuffer inClusters, EmberMessageBuffer outClusters, EmberApsOption options)`

Binding Manager Functions

- `EmberStatus emberEndDeviceBindRequest (uint8_t endpoint, EmberApsOption options)`

Function to Decode Address Response Messages

- `EmberNodeId emberDecodeAddressResponse (EmberMessageBuffer response, EmberEUI64 eui64Return)`

8.157.1 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](#).

8.158 zigbee-device-library.h

```

00001
00101 EmberStatus emberMatchDescriptorsRequest
00102     (EmberNodeId target,
00103      uint16_t profile,
00104      EmberMessageBuffer
00105      inClusters,
00106      EmberMessageBuffer
00107      outClusters,
00108      EmberApsOption options);
00125 EmberStatus emberEndDeviceBindRequest (
00126     uint8_t endpoint,
00127     EmberApsOption options);
00143 EmberNodeId emberDecodeAddressResponse (
00144     EmberMessageBuffer response,
00145     EmberEUI64 eui64Return);

```

8.159 zigbee-device-stack.h File Reference

Functions

- `EmberStatus emberNetworkAddressRequest (EmberEUI64 target, bool reportKids, uint8_t childStartIndex)`
- `EmberStatus emberIeeeAddressRequest (EmberNodeId target, bool reportKids, uint8_t childStartIndex, EmberApsOption options)`
- `EmberStatus emberEnergyScanRequest (EmberNodeId target, uint32_t scanChannels, uint8_t scanDuration, uint16_t scanCount)`
- `EmberStatus emberSetNetworkManagerRequest (EmberNodeId networkManager, uint32_t activeChannels)`
- `EmberStatus emberChannelChangeRequest (uint8_t channel)`
- `EmberStatus emberSendDeviceAnnouncement (void)`
- `EmberStatus emberSendParentAnnouncement (void)`
- `uint8_t emberGetLastStackZigDevRequestSequence (void)`

8.159.1 Detailed Description

ZigBee Device Object (ZDO) functions included in the stack. See [ZigBee Device Object](#) for documentation.

Definition in file [zigbee-device-stack.h](#).

8.160 zigbee-device-stack.h

```

00001
00010 #ifndef __ZIGBEE_DEVICE_STACK_H__
00011 #define __ZIGBEE_DEVICE_STACK_H__
00012
00013 EmberStatus emberNetworkAddressRequest(
00014     EmberEUI64 target,
00015                         bool reportKids,
00016                         uint8_t childStartIndex);
00017
00018 EmberStatus emberIeeeAddressRequest (
00019     EmberNodeId target,
00020                         bool reportKids,
00021                         uint8_t childStartIndex,
00022                         EmberApsOption options);
00023
00024 EmberStatus emberEnergyScanRequest(EmberNodeId
00025     target,
00026                         uint32_t scanChannels,
00027                         uint8_t scanDuration,
00028                         uint16_t scanCount);
00029
00030 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00031 EmberStatus emberSetNetworkManagerRequest
00032     (EmberNodeId networkManager,
00033                         uint32_t activeChannels);
00034 #else
00035 #define emberSetNetworkManagerRequest (manager, channels)
00036     (emberEnergyScanRequest(EMBER_SLEEPY_BROADCAST_ADDRESS,
00037         (channels),
00038         0xFF,
00039         (manager)))
00040 #endif
00041
00042 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00043 EmberStatus emberChannelChangeRequest (
00044     uint8_t channel);
00045 #else
00046 #define emberChannelChangeRequest (channel)
00047     (emberEnergyScanRequest(EMBER_SLEEPY_BROADCAST_ADDRESS,
00048         BIT32(channel),
00049         0xFE,
00050         0))
00051 #endif
00052
00053 EmberStatus emberSendDeviceAnnouncement (
00054     void);
00055
00056 EmberStatus emberSendParentAnnouncement (
00057     void);
00058
00059 uint8_t emberGetLastStackZigDevRequestSequence
00060     (void);
00061
00062 #endif // __ZIGBEE_DEVICE_STACK_H__

```

8.161 zll-api.h File Reference

Functions

- `EmberStatus emberZllFormNetwork (EmberZllNetwork *networkInfo, int8_t radioTxPower)`
- `EmberStatus emberZllJoinTarget (const EmberZllNetwork *targetNetworkInfo)`
- `EmberStatus emberZllSetInitialSecurityState (const EmberKeyData *networkKey, const EmberZllInitialSecurityState *securityState)`
- `EmberStatus emberZllStartScan (uint32_t channelMask, int8_t radioPowerForScan, EmberNodeType nodeType)`
- `EmberStatus emberZllSetRxOnWhenIdle (uint16_t durationMs)`
- `void emberZllNetworkFoundHandler (const EmberZllNetwork *networkInfo, const EmberZllDeviceInfoRecord *deviceInfo)`
- `void emberZllScanCompleteHandler (EmberStatus status)`
- `void emberZllAddressAssignmentHandler (const EmberZllAddressAssignment *addressInfo)`
- `void emberZllTouchLinkTargetHandler (const EmberZllNetwork *networkInfo)`
- `void emberZllGetTokenStackZllData (EmberTokTypeStackZllData *token)`
- `void emberZllGetTokenStackZllSecurity (EmberTokTypeStackZllSecurity *token)`
- `void emberZllGetTokensStackZll (EmberTokTypeStackZllData *data, EmberTokTypeStackZllSecurity *security)`
- `void emberZllSetTokenStackZllData (EmberTokTypeStackZllData *token)`
- `bool emberIsZllNetwork (void)`
- `void emberZllSetNonZllNetwork (void)`
- `EmberZllPolicy emberZllGetPolicy (void)`
- `EmberStatus emberZllSetPolicy (EmberZllPolicy policy)`

8.162 zll-api.h

```

00001 /*
00002  * @file zll-api.h
00003 *
00004  * @brief API for sending and receiving ZigBee Light Link (ZLL) messages.
00005 *
00006  * <!--Copyright 2010 by Ember Corporation. All rights reserved.          *80*
-->
00007 */
00008
00033 EmberStatus emberZllFormNetwork(EmberZllNetwork
00034           * networkInfo,
00035           int8_t radioTxPower);
00052 EmberStatus emberZllJoinTarget(const
00053           EmberZllNetwork* targetNetworkInfo);
00054
00067 EmberStatus emberZllSetInitialSecurityState
00068           (const EmberKeyData *networkKey,
00069           const EmberZllInitialSecurityState
00070           *securityState);
00082 EmberStatus emberZllStartScan(uint32_t channelMask,
00083           int8_t radioPowerForScan,
00084           EmberNodeType nodeType);
00085
00098 EmberStatus emberZllSetRxOnWhenIdle(uint16_t
00099           durationMs);
00112 void emberZllNetworkFoundHandler(const
00113           EmberZllNetwork* networkInfo,
00114           const EmberZllDeviceInfoRecord
00115           * deviceInfo);
00116

```

```

00123 void emberZllScanCompleteHandler(EmberStatus
    status);
00124
00132 void emberZllAddressAssignmentHandler(const
    EmberZllAddressAssignment* addressInfo);
00133
00140 void emberZllTouchLinkTargetHandler(const
    EmberZllNetwork *networkInfo);
00141
00145 void emberZllGetTokenStackZllData(
    EmberTokTypeStackZllData *token);
00146
00150 void emberZllGetTokenStackZllSecurity(
    EmberTokTypeStackZllSecurity *token);
00151
00155 void emberZllGetTokensStackZll(
    EmberTokTypeStackZllData *data,
00156                                         EmberTokTypeStackZllSecurity
    *security);
00157
00161 void emberZllSetTokenStackZllData(
    EmberTokTypeStackZllData *token);
00162
00166 bool emberIsZllNetwork(void);
00167
00172 void emberZllSetNonZllNetwork(void);
00173
00177 EmberZllPolicy emberZllGetPolicy(void);
00178
00182 EmberStatus emberZllSetPolicy(EmberZllPolicy
    policy);
00183

```

8.163 zll-types.h File Reference

Data Structures

- struct [EmberZllSecurityAlgorithmData](#)
Information about the ZLL security being and how to transmit the network key to the device securely.
- struct [EmberZllNetwork](#)
Information about the ZLL network and specific device that responded to a ZLL scan request.
- struct [EmberZllDeviceInfoRecord](#)
Information discovered during a ZLL scan about the ZLL device's endpoint information.
- struct [EmberZllAddressAssignment](#)
Network and group address assignment information.
- struct [EmberZllInitialSecurityState](#)
This describes the Initial Security features and requirements that will be used when forming or joining ZigBee Light Link networks.
- struct [EmberTokTypeStackZllData](#)
- struct [EmberTokTypeStackZllSecurity](#)

ZigBee Light Link Types

- #define EMBER_ZLL_PRIMARY_CHANNEL_MASK
- #define EMBER_ZLL_SECONDARY_CHANNEL_MASK
- #define EMBER_ZLL_NULL_NODE_ID
- #define EMBER_ZLL_MIN_NODE_ID
- #define EMBER_ZLL_MAX_NODE_ID

- #define EMBER_ZLL_NULL_GROUP_ID
- #define EMBER_ZLL_MIN_GROUP_ID
- #define EMBER_ZLL_MAX_GROUP_ID
- #define EMBER_ZLL_CLUSTER_ID
- #define EMBER_ZLL_PROFILE_ID
- #define EMBER_ZLL_KEY_MASK_DEVELOPMENT
- #define EMBER_ZLL_KEY_MASK_MASTER
- #define EMBER_ZLL_KEY_MASK_CERTIFICATION
- #define EMBER_ZLL_CERTIFICATION_ENCRYPTION_KEY
- #define EMBER_ZLL_CERTIFICATION_PRECONFIGURED_LINK_KEY
- enum EmberZllState {
 EMBER_ZLL_STATE_NONE, EMBER_ZLL_STATE_FACTORY_NEW, EMBER_ZLL_STATE_ADDRESS_ASSIGNMENT_CAPABLE, EMBER_ZLL_STATE_LINK_INITIATOR,
 EMBER_ZLL_STATE_LINK_PRIORITY_REQUEST, EMBER_ZLL_STATE_PROFILE_INTEROP, EMBER_ZLL_STATE_NON_ZLL_NETWORK
 }
- enum EmberZllKeyIndex { EMBER_ZLL_KEY_INDEX_DEVELOPMENT, EMBER_ZLL_KEY_INDEX_MASTER, EMBER_ZLL_KEY_INDEX_CERTIFICATION }
- enum EmberZllPolicy { EMBER_ZLL_POLICY_ENABLED, EMBER_ZLL_POLICY_DISABLED }

8.163.1 Detailed Description

Ember data type definitions. See [Ember ZigBee Light Link \(ZLL\) Data Types](#) for details.

Definition in file [zll-types.h](#).

8.164 zll-types.h

```

00001
00017 #ifndef ZLL_TYPES_H
00018 #define ZLL_TYPES_H
00019
00024
00028 #define EMBER_ZLL_PRIMARY_CHANNEL_MASK ((uint32_t)(BIT32(11) \
00029 | BIT32(15) \
00030 | BIT32(20) \
00031 | BIT32(25)))
00032
00036 #define EMBER_ZLL_SECONDARY_CHANNEL_MASK ((uint32_t)(BIT32(12) \
00037 | BIT32(13) \
00038 | BIT32(14) \
00039 | BIT32(16) \
00040 | BIT32(17) \
00041 | BIT32(18) \
00042 | BIT32(19) \
00043 | BIT32(21) \
00044 | BIT32(22) \
00045 | BIT32(23) \
00046 | BIT32(24) \
00047 | BIT32(26)))
00048
00053 #define EMBER_ZLL_NULL_NODE_ID 0x0000
00054
00058 #define EMBER_ZLL_MIN_NODE_ID 0x0001
00059
00063 #define EMBER_ZLL_MAX_NODE_ID 0xFFFF7
00064
00069 #define EMBER_ZLL_NULL_GROUP_ID 0x0000
00070

```

```

00074 #define EMBER_ZLL_MIN_GROUP_ID 0x0001
00075
00079 #define EMBER_ZLL_MAX_GROUP_ID 0xFEFF
00080
00085 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00086 enum EmberZllState
00087 #else
00088 typedef uint16_t EmberZllState;
00089 enum
00090 #endif
00091 {
00093   EMBER_ZLL_STATE_NONE          = 0x0000,
00095   EMBER_ZLL_STATE_FACTORY_NEW   =
0x0001,
00097   EMBER_ZLL_STATE_ADDRESS_ASSIGNMENT_CAPABLE
= 0x0002,
00099   EMBER_ZLL_STATE_LINK_INITIATOR
= 0x0010,
00101   EMBER_ZLL_STATE_LINK_PRIORITY_REQUEST
= 0x0020,
00103   EMBER_ZLL_STATE_PROFILE_INTEROP
0x0080,
00105   EMBER_ZLL_STATE_NON_ZLL_NETWORK
= 0x0100,
00106 };
00107
00112 typedef struct {
00113   uint32_t transactionId;
00114   uint32_t responseId;
00115   uint16_t bitmask;
00116 } EmberZllSecurityAlgorithmData;
00117
00122 typedef struct {
00123   EmberZigbeeNetwork zigbeeNetwork;
00124   EmberZllSecurityAlgorithmData securityAlgorithm
;
00125   EmberEUI64 eui64;
00126   EmberNodeId nodeId;
00127   EmberZllState state;
00128   EmberNodeType nodeType;
00129   uint8_t numberSubDevices;
00130   uint8_t totalGroupIdentifiers;
00131   uint8_t rssiCorrection;
00132 } EmberZllNetwork;
00133
00138 typedef struct {
00139   EmberEUI64 ieeeAddress;
00140   uint8_t endpointId;
00141   uint16_t profileId;
00142   uint16_t deviceId;
00143   uint8_t version;
00144   uint8_t groupIdCount;
00145 } EmberZllDeviceInfoRecord;
00146
00150 typedef struct {
00151   EmberNodeId nodeId;
00152   EmberNodeId freeNodeIdMin;
00153   EmberNodeId freeNodeIdMax;
00154   EmberMulticastId groupIdMin;
00155   EmberMulticastId groupIdMax;
00156   EmberMulticastId freeGroupIdMin;
00157   EmberMulticastId freeGroupIdMax;
00158 } EmberZllAddressAssignment;
00159
00163 #define EMBER_ZLL_CLUSTER_ID 0x1000
00164
00168 #define EMBER_ZLL_PROFILE_ID 0xC05E
00169
00170
00174 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00175 enum EmberZllKeyIndex
00176 #else
00177 typedef uint8_t EmberZllKeyIndex;
00178 enum
00179 #endif
00180 {
00182   EMBER_ZLL_KEY_INDEX_DEVELOPMENT    = 0x00,
00184   EMBER_ZLL_KEY_INDEX_MASTER        = 0x04,
00186   EMBER_ZLL_KEY_INDEX_CERTIFICATION = 0x0F,
00187 };

```

```

00188
00193 #define EMBER_ZLL_KEY_MASK_DEVELOPMENT (1 << EMBER_ZLL_KEY_INDEX_DEVELOPMENT)
00194
00199 #define EMBER_ZLL_KEY_MASK_MASTER (1 << EMBER_ZLL_KEY_INDEX_MASTER)
00200
00205 #define EMBER_ZLL_KEY_MASK_CERTIFICATION (1 <<
    EMBER_ZLL_KEY_INDEX_CERTIFICATION)
00206
00211 #define EMBER_ZLL_CERTIFICATION_ENCRYPTION_KEY \
00212     {0xC0, 0x1, 0xC2, 0xC3, 0xC4, 0xC5, 0xC6, 0xC7, \
00213         0xC8, 0xC9, 0xCA, 0xCB, 0xCC, 0xCD, 0xCE, 0xCF}
00214
00219 #define EMBER_ZLL_CERTIFICATION_PRECONFIGURED_LINK_KEY \
00220     {0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xD6, 0xD7, \
00221         0xD8, 0xD9, 0xDA, 0xDB, 0xDC, 0xDD, 0xDE, 0xDF}
00222
00227 typedef struct {
00229     uint32_t bitmask;
00231     EmberZllKeyIndex keyIndex;
00233     EmberKeyData encryptionKey;
00235     EmberKeyData preconfiguredKey;
00236 } EmberZllInitialSecurityState;
00237
00241 #ifdef DOXYGEN_SHOULD_SKIP_THIS
00242 enum EmberZllPolicy
00243 #else
00244 typedef uint8_t EmberZllPolicy;
00245 enum
00246 #endif
00247 {
00249     EMBER_ZLL_POLICY_ENABLED = 0x00,
00251     EMBER_ZLL_POLICY_DISABLED = 0x01,
00252 };
00253
00255
00256 typedef struct {
00257     uint32_t bitmask;
00258     uint16_t freeNodeIdMin;
00259     uint16_t freeNodeIdMax;
00260     uint16_t myGroupIdMin;
00261     uint16_t freeGroupIdMin;
00262     uint16_t freeGroupIdMax;
00263     uint8_t rssiCorrection;
00264 } EmberTokTypeStackZllData;
00265
00266 typedef struct {
00267     uint32_t bitmask;
00268     uint8_t keyIndex;
00269     uint8_t encryptionKey[16];
00270     uint8_t preconfiguredKey[16];
00271 } EmberTokTypeStackZllSecurity;
00272
00273 #endif // ZLL_TYPES_H
00274

```