Connect

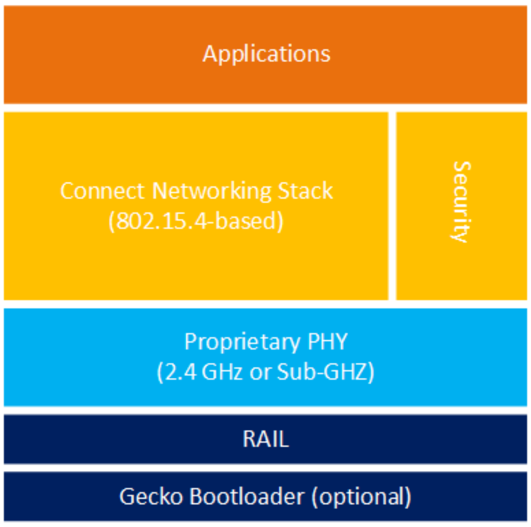
_DATA_INIT_SEGMENT_BEGIN
_DATA_SEGMENT_BEGIN
_BSS_SEGMENT_BEGIN
_CONST_SEGMENT_BEGIN
_TEXT_SEGMENT_BEGIN
_TEXTRW_INIT_SEGMENT_BEGIN
_TEXTRW_SEGMENT_BEGIN
_AAT_SEGMENT_BEGIN
_BAT_SEGMENT_BEGIN
_BAT_INIT_SEGMENT_BEGIN
_FAT_SEGMENT_BEGIN
_RAT_SEGMENT_BEGIN
_SIMEE_SEGMENT_BEGIN
_PSSTORE_SEGMENT_BEGIN
_LONGTOKEN_SEGMENT_BEGIN
_EMHEAP_SEGMENT_BEGIN
_GUARD_REGION_SEGMENT_BEGIN
_DLIB_PERTHREAD_INIT_SEGMENT_BEGIN
_DLIB_PERTHREAD_INITIALIZED_DATA_SEGMENT_BEGIN
_DLIB_PERTHREAD_ZERO_DATA_SEGMENT_BEGIN
_INTERNAL_STORAGE_SEGMENT_BEGIN
_LOCKBITS_IN_MAINFLASH_SEGMENT_BEGIN
_UNRETAINED_RAM_SEGMENT_BEGIN
_NO_INIT_SEGMENT_END
_DEBUG_CHANNEL_SEGMENT_END
_INTVEC_SEGMENT_END
_CSTACK_SEGMENT_END
_RESETINFO_SEGMENT_END
_DATA_INIT_SEGMENT_END
_DATA_SEGMENT_END
_BSS_SEGMENT_END
_CONST_SEGMENT_END
_TEXT_SEGMENT_END
_TEXTRW_INIT_SEGMENT_END
_TEXTRW_SEGMENT_END
_AAT_SEGMENT_END
_BAT_SEGMENT_END
_BAT_INIT_SEGMENT_END
_FAT_SEGMENT_END
_RAT_SEGMENT_END
_SIMEE_SEGMENT_END
_PSSTORE_SEGMENT_END
_LONGTOKEN_SEGMENT_END
_EMHEAP_SEGMENT_END

Developing with Connect

Developing Proprietary Connect Applications

Silicon Labs Connect is an IEEE 802.15.4 MAC-based wireless networking stack for a variety of proprietary applications optimized for low-power devices. This full-featured, easily customizable networking stack is designed for compliance with regulatory specifications across worldwide geographic regions and supports both Sub-GHz and 2.4 GHz frequency bands.

The Silicon Labs Connect stack supports many combinations of radio modulation, frequency and data rates. The stack provides support for end nodes, coordinators, and range extenders. It includes all wireless MAC (Medium Access Control) layer functions such as scanning and joining, setting up a point-to-point or star network, and managing device types such as sleepy end devices, routers, and coordinators. With all this functionality already implemented in the stack, users can focus on their end application development and not worry about the lower-level radio and network details.


Applications
Connect Networking Stack (802.15.4-based)
Security
Proprietary PHY (2.4 GHz or Sub-GHZ)
RAIL
Gecko Bootloader (optional)

The Connect stack is part of the Silicon Labs Flex SDK (Software Development Kit), installed through Simplicity Studio. Connect runs on top of RAIL (Radio Abstraction Interface Layer), also included with the Flex SDK. RAIL provides an intuitive, easily-customizable radio interface layer that is designed to support proprietary or standards-based wireless protocols.

The content on these pages is intended for those who want to experiment with or are already developing an application using the Silicon Labs Connect Stack.

For details about this release: Links to the Flex SDK release notes are available on the silabs.com Gecko SDK page.

For Silicon Labs' Connect product information: See the product pages on silabs.com.

For background about the Connect stack and other wireless networking topics: The Fundamentals section is a good place to start.

To get started with development: See the Getting Started section to get started working with example applications.

If you are already in development: See the Developer's Guide for details or go directly to the API Reference.

Overview

Getting Started with Silicon Labs Connect Development

To get started with Silicon Labs Connect development, download the Simplicity Studio Development environment as described in the Simplicity Studio 5 User's Guide.

This will also prompt you to install the Gecko SDK (GSDK). The GSDK combines Silicon Labs wireless software development kits (SDKs) such as the Flex SDK and Gecko Platform into a single, integrated package. The GSDK is your primary tool for developing in the Silicon Labs IoT Software ecosystem. All of Silicon Labs' stacks are written in-house to provide a seamless experience from silicon to tools, allowing you to unlock powerful features with ease, including:

- Abstraction of complex requirements like multiprotocol and pre-certification
- Industry-leading ability to support a large number of nodes
- Ultra-low power consumption
- Strong network reliability

Silicon Labs also helps future-proof your devices with over-the-air software and security updates, helping to minimize maintenance cost and improve your end user product experience!

The Flex SDK includes both the Connect library and examples and the RAIL (Radio Abstraction Interface Layer) Library and examples. Once you have downloaded Simplicity Studio and the GSDK, detailed instructions for using the Connect examples and configuration tools are provided in the **Proprietary Flex SDK Quick-Start Guide (QSG168)**.

Note: The recommended method to get started with the GSDK is to first install Simplicity Studio 5, which will set up your development environment and walk you through the GSDK installation. Alternatively, GSDK and other required tools may be installed manually from the GitHub GSDK site.

Overview

Connect Fundamentals

Silicon Labs has produced a series of documents on topics that provide useful background for Silicon Labs Connect developers.

- Silicon Labs Connect Fundamentals (PDF): Describes the features and functions of the Silicon Labs Connect stack, including its device types, network topologies, and its 'building block' development methodology using components.
- Using Silicon Labs Connect v3.x with IEEE 802.15.4 (PDF): Introduces the IEEE 802.15.4 standard on which Connect v3.x is based.
- Wireless Networking Application Development Fundamentals (PDF): For those new to wireless networking, introduces some fundamental concepts of wireless networking.

Overview

Connect Developers Guide

The Developer's Guide content is organized in the following groups:

- Developing and Debugging: A description of development resources as well as detailed information on a variety of topics.
- Bootloading: Information on using the Gecko Bootloader with Connect applications.
- Multiprotocol: Background on implementing multiprotocol applications and information on different multiprotocol models.
- Non-Volatile Data Storage: Background on managing device memory.
- Security: Describes Silicon Labs security resources and how to manage Connect security.

Overview

Developing and Debugging Connect Applications

These pages provide details on developing and debugging applications using the Connect stack. Content includes:

- **Architecture of the Silicon Labs Connect Stack v3.x (PDF)**: Describes the architecture of the Silicon Labs Connect stack v3.x an how it implements IEEE 802.15.4.
- **Customizing Applications with Silicon Labs Connect v3.x (PDF)**: Describes how to use components, callbacks, and events on top of the Gecko Platform application framework to configure features and application behavior.
- **Network Co-Processor Applications with Silicon Labs Connect v3.x (PDF)**: Describes how to run the Silicon Labs Connect stack in Network Co-Processor (NCP) mode, where the NCP runs on the EFR32 while the Host application and the Co-processor Communication daemon (CPCd) run on the Host device.
- **Using Real Time Operating Systems with Silicon Labs Connect v3.x (PDF)**: Describes the process to implement a Connect-based application on top of one of the supported Real Time Operating Systems (RTOS).
- **Energy Saving with Silicon Labs Connect v3.x (PDF)**: Describes the features available in Connect v3.x to reduce power consumption. Using those features is described in AN1252: Building Low Power Networks with the Silicon Labs Connect Stack v3.x.
- **Building Low Power Networks with the Silicon Labs Connect Stack v3.x (PDF)**: Illustrates reducing power consumption in a Connect v3.x application using the sensor example.
- **EFR32 Radio Configurator Guide for Simplicity Studio 5 (PDF)**: Documents the Radio Configurator tool, which can be used to create customizations at the PHY level.
- **PHY Limitations and Timing Optimization**: Connect specific recommendations on radio config customizations.

Development Tools

Simplicity Studio and the Simplicity IDE: Simplicity Studio is the unified development environment for all Silicon Labs technologies, SoCs, and modules. It provides you with access to the target device-specific web and SDK resources, software and hardware configuration tools, and an integrated development environment (IDE) featuring industry-standard code editors, compilers, and debuggers. See the silabs.com Simplicity Studio page to download the tools and for more information.

Network Analyzer: Simplicity Studio® 5 (SSv5)'s Network Analyzer enables debugging of complex wireless systems. This tool captures a trace of wireless network activity that can be examined in detail live or at a later time. See the Network Analyzer section of the Simplicity Studio 5 User's Guide for more information.

Wireshark: Wireshark is the recommended network protocol analyzer for the use with Wi-SUN networks. Download instructions are provided for Windows/Mac users or Linux users. Simplicity Studio® 5 supports live interaction between the application running on a Silicon Labs device and Wireshark.

Energy Profiler: Simplicity Studio® 5 (SSv5)'s Energy Profiler enables you to visualize the energy consumption of individual devices, multiple devices on one target system, or a network of interacting wireless devices to analyze and improve the power performance of these systems. Real-time information on current consumption is correlated with the program counter providing advanced energy software monitoring capabilities. It also provides a basic level of integration with the Network Analyzer network analysis tool. See the Energy Profiler section of the Simplicity Studio 5 User's Guide for more information.

Simplicity Commander: Simplicity Commander is a single, all-purpose tool to be used in a production environment. It is invoked using a simple Command Line Interface (CLI) that is also scriptable. Simplicity Commander enables customers to complete essential tasks such as configuring and building applications and bootloaders and flashing images to their devices. Simplicity Commander is available through Simplicity Studio or can be downloaded through system-specific installers. The **Simplicity Commander User's Guide** provides more information.

Silicon Labs Configurator (SLC): SLC offers command-line access to application configuration and generation functions. Software Project Generation and Configuration with SLC-CLI provides instructions on downloading and using the SLC-CLI tool.

PHY Limitations and Timing Optimization

PHY Limitations and Timing Optimization

Supported profiles

Connect supports a wide range of radio configurations, including pre-configured and customized PHYs. The pre-configured PHYs are under the following profiles:

- Connect Profile
- Connect OFDM Profile
- Long Range Profile

Under these profiles, any customization is supported by Connect. Note, however, that using the radio configurator requires basic knowledge of digital modulations. It is possible to misconfigure modulation parameters to the level where communication might be impossible.

Base Profile

The Base profile allows the widest configuration options. However, Connect is only compatible with the IEEE 802.15.4 frame configuration. For the modulations available in the Base Profile, Connect only supports 1 Byte PHR frame configurations. To set this up, follow these settings:

- Check Header Enable
- Set Header Size to 1
- Set Frame Length Encoding to VARIABLE_LENGTH
- Set Frame Bit Endian to LSB_FIRST
- Check Length Includes CRC Bytes
- Set Minimum Length to 0
- Set Maximum Length to 127
- Set Variable Length Bit Size to 7
- Set Variable Frame Length Adjust to 0
- Set Variable Length Bit Endian to LSB_FIRST
- Set Variable Length Bit Location to 0

The frame setting also requires a 2 Byte long CRC, so make sure to select a 2 Byte long CRC Polynomial.

Although Connect can handle 2 Byte PHR and the PHR required for OFDM, those settings can only be activated by selecting a PHY from the Connect profiles.

OFDM Settings

The Connect OFDM PHYs are essentially same as Wi-SUN OFDM PHYs. However, without the limitation of the Wi-SUN spec, we allow changing the channel map and carrier frequency.

OFDM bitrate depends on bandwidth and MCS (modulation coding scheme). The bandwidth can be configured on the radio configurator, while MCS can be changed run-time, with the `emberOfdmSetMcs()` API. The default MCS is 0.

The available bitrates in kb/s, depending on bandwidth and MCS:

	0.2 MHz	0.4 MHz	0.8 MHz	1.2 MHz
MCS=0	12.5	25	50	100
MCS=1	25	50	100	200

	0.2 MHz	0.4 MHz	0.8 MHz	1.2 MHz
MCS=2	50	100	200	400
MCS=3	100	200	400	800
MCS=4	150	300	600	1200
MCS=5	200	400	800	1600
MCS=6	300	600	1200	2400

Multi-PHY Considerations

Connect has limited support for the Multi-PHY capabilities of EFR32. Channel-based Multi-PHY (i.e., configuration changes are applied by changing the channel) is fully supported. However, Connect always loads the first protocol in a protocol-based Multi-PHY configuration. For more details on this term, see AN1253 (PDF).

Optimizing Connect for a PHY

Although Connect is set up to be usable with a wide range of PHYs, there are certain features that depend on the PHY. These are usually configured conservatively, so it will probably work with most configurations, but not all. Furthermore, even if it works, it might be necessary to optimize these parameters for more effective operation.

CSMA/CA

The default configuration in Connect follows the recommendation from IEEE 802.15.4. All timing parameters are symbolrate dependent, while the threshold is -65dBm. All of the parameters can be changed via `emberSetMacParams()`. These parameters very rarely need modification. Perhaps, on a very low symbolrate, it is worth setting an overall timout via `csmaTimeout`.

Turnaround Time and Acknowledgement Timeout

Turnaround time (delay between received packet and transmitted acknowledgement) in Connect is not configurable; it is always 12 symbol time.

Acknowledgement timeout by default is 25ms. This is much more than most PHY needs. Turnaround time + acknowledgement frame transmission time should be enough in theory. This can be tuned via `emberSetMacParams()`, with the `ackTimeout` argument.

Note though that in practice, optimizing the timeout doesn't improve efficiency much. The timeout only blocks the stack to transmit another packet while waiting on the acknowledgement.

Active Scan Duration

Active scan is used when a device joins a network. The joining device sends a beacon request, and waits a predefined duration for beacons of the joinable devices. This is the active scan duration, which is 960*2^5=30720 symbol time by default. This is however way too much for low bitrate; e.g. at 9.6kbps, this results in 3.2s.

This can be configured via `emberSetActiveScanDuration()`. The beacons are transmitted with a 50ms jitter after the beacon request frame (with additional delay caused by CSMA/CA).

Overview

Bootloading Embedded Applications

Bootloading allows you to update application firmware images on your devices. This section provides background information about bootloading using the Silicon Labs Gecko Bootloader.

- Bootloader Fundamentals (PDF): Bootloader Fundamentals - Introduces bootloading for Silicon Labs networking devices. Discusses the Gecko Bootloader as well as legacy Ember and Bluetooth bootloaders, and describes the file formats used by each.
- Bootloading and OTA with Silicon Labs Connect v3.x (PDF): Explains standalone (serial) and application (OTA) bootloader options available for use within Connect v3.x-based applications.
- Using the Gecko Bootloader with Silicon Labs Connect (PDF): Includes detailed information on using the Silicon Labs Gecko Bootloader with Connect. It supplements the general Gecko Bootloader implementation information provided in UG489: Silicon Labs Gecko Bootloader User's Guide for GSDK 4.0 and Higher.
- Gecko Bootloader User's Guide for GSDK 4.0 and Higher (PDF): Describes the high-level implementation of the Silicon Labs Gecko Bootloader for EFR32 SoCs and NCPs, and provides information on how to get started using the Gecko Bootloader with Silicon Labs wireless protocol stacks in GSDK 4.0 and higher.
- Series 2 Secure Boot with RTSL (PDF): Contains detailed information on configuring and using the Secure Boot with hardware Root of Trust and Secure Loader on Series 2 devices, including how to provision the signing key. This is a companion document to UG266: Silicon Labs Gecko Bootloader User's Guide.
- Transitioning to the Updated Gecko Bootloader in GSDK 4.0 and Higher (PDF): Gecko Bootloader v2.x, introduced in GSDK 4.0, contains a number of changes compared to Gecko Bootloader v1.x. This document describes the differences between the versions, including how to configure the new Gecko Bootloader in Simplicity Studio 5.

Overview

Multiprotocol

This section provides background information on multiprotocol applications.

- Multiprotocol Fundamentals (PDF): Describes the four multiprotocol modes, discusses considerations when selecting protocols for multiprotocol implementations, and reviews the Radio Scheduler, a required component of a dynamic multiprotocol solution.
- Dynamic Multiprotocol User's Guide (PDF): Describes how to implement a dynamic multiprotocol solution.

Overview

Non-Volatile Data Storage

This section offers an introduction to non-volatile data storage and describes how to use NVM3 data storage.

- Non-Volatile Data Storage Fundamentals (PDF): Introduces non-volatile data storage using flash and the three different storage implementations offered for Silicon Labs microcontrollers and SoCs: Simulated EEPROM, PS Store, and NVM3.
- Using Third Generation Non-Volatile Memory (NVM3) Data Storage (PDF): Explains how NVM3 can be used as non-volatile data storage in various protocol implementations.
- Bringing Up Custom Devices for the EFR32MG and EFR32FG Families (PDF): Describes how to initialize a piece of custom hardware (a 'device') based on the EFR32MG and EFR32FG families so that it interfaces correctly with a network stack. The same procedures can be used to restore devices whose settings have been corrupted or erased.
- Using Tokens for Non-Volatile Data Storage (PDF): Describes tokens and shows how to use them for non-volatile data storage in EmberZNet PRO and Silicon Labs Flex applications.

Overview

Security

Silicon Labs offers a range of security features depending on the part you are using and your application and production needs. This section provides background on security and how to use the available security features.

- IoT Security Fundamentals (PDF): Introduces the security concepts that must be considered when implementing an Internet of Things (IoT) system. Using the ioXt Alliance's eight security principles as a structure, it clearly delineates the solutions Silicon Labs provides to support endpoint security and what you must do outside of the Silicon Labs framework.
- Series 2 Secure Debug (PDF): Describes how to lock and unlock the debug access of EFR32 Gecko Series 2 devices. Many aspects of the debug access, including the secure debug unlock are described. The Debug Challenge Interface (DCI) and Secure Engine (SE) Mailbox Interface for locking and unlocking debug access are also included.
- Production Programming of Series 2 Devices (PDF): Provides details on programming, provisioning, and configuring Series 2 devices in production environments. Covers Secure Engine Subsystem of Series 2 devices, which runs easily upgradeable Secure Engine (SE) or Virtual Secure Engine (VSE) firmware.
- Anti-Tamper Protection Configuration and Use (PDF): Anti-Tamper Protection Configuration and Use - Shows how to program, provision, and configure the anti-tamper module on EFR32 Series 2 devices with Secure Vault.
- Authenticating Silicon Labs Devices using Device Certificates (PDF): How to authenticate an EFR32 Series 2 device with Secure Vault, using secure device certificates and signatures.
- Secure Key Storage (PDF): Explains how to securely "wrap" keys in EFR32 Series 2 devices with Secure Vault, so they can be stored in non-volatile storage.
- Programming Series 2 Devices Using the Debug Challenge Interface (DCI) and Serial Wire Debug (SWD) (PDF): Describes how to provision and configure Series 2 devices through the DCI and SWD.
- Integrating Crypto Functionality Using PSA Crypto Compared to Mbed TLS (PDF): Describes how to integrate crypto functionality into applications using PSA Crypto compared to Mbed TLS.

Connect

Connect

Silicon Labs Connect Stack API

Connect stack API is the primary Application Programming Interface (API) for applications running on Silicon Labs EFR32 Wireless Gecko SoCs to interact with the Silicon Labs Connect wireless stack.

Silicon Labs is developing products designed to meet the demands of customers moving to a connected world of devices in the home, often referred to as the IoT (Internet of Things). At a high level, the IoT goals for Silicon Labs are as follows:

- Connect all devices in the home with best-in-class mesh networking, either with Ember ZigBee PRO or other emerging standards.
- Leverage the company expertise in low-power, constrained devices.
- Enhance established low-power, mixed-signal chips.
- Provide low-cost bridging to existing Ethernet and Wi-Fi devices.
- Enable cloud services and connectivity to smartphones and tablets that promote ease of use and a common user experience for customers.

Connect Stack API Reference

Connect Stack API Reference

The primary API towards the Connect radio stack.

Modules

Connect Stack Version

Connect Data Types

Stack Information

Network Management

Radio Stream

Configuration

Status Codes

Stack Tokens

Event Scheduling

Memory Buffer

Connect Stack Version

Connect Stack Version

Macros to determine the stack version.

Note that the Connect Stack version might not match the version of Flex SDK.

See config.h for source code.

Macros

#define	EMBER_MAJOR_VERSION 4 The major version of the release. First digit of A.B.C.D.
#define	EMBER_MINOR_VERSION 0 The minor version of the release. Second digit of A.B.C.D.
#define	EMBER_PATCH_VERSION 1 The patch version of the release. Third digit of A.B.C.D.
#define	EMBER_SPECIAL_VERSION 0 Special version of the release. Fourth digit of A.B.C.D.
#define	EMBER_BUILD_NUMBER 0 Build number of the release. Should be stored on 2 bytes.
#define	EMBER_FULL_VERSION undefined Full version number stored on 2 bytes, with each of the four digits stored on 4 bits.
#define	EMBER_VERSION_TYPE EMBER_VERSION_TYPE_GA Version type of the release. EMBER_VERSION_TYPE_GA means generally available.
#define	SOFTWARE_VERSION EMBER_FULL_VERSION Full version number stored on 2 bytes, with each of the four digits stored on 4 bits.

Macro Definition Documentation

EMBER_MAJOR_VERSION

```
#define EMBER_MAJOR_VERSION
```

Value:

```
4
```

The major version of the release. First digit of A.B.C.D.

Definition at line `43` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/config.h`

EMBER_MINOR_VERSION

```
#define EMBER_MINOR_VERSION
```

Value:

```
0
```

The minor version of the release. Second digit of A.B.C.D.

Definition at line `48` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/config.h`

EMBER_PATCH_VERSION

```
#define EMBER_PATCH_VERSION
```

Value:

```
1
```

The patch version of the release. Third digit of A.B.C.D.

Patch versions are fully backwards compatible as long as the major and minor version matches.

Definition at line `56` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/config.h`

EMBER_SPECIAL_VERSION

```
#define EMBER_SPECIAL_VERSION
```

Value:

```
0
```

Special version of the release. Fourth digit of A.B.C.D.

Definition at line `61` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/config.h`

EMBER_BUILD_NUMBER

```
#define EMBER_BUILD_NUMBER
```

Value:

```
0
```

Build number of the release. Should be stored on 2 bytes.

Definition at line `66` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/config.h`

EMBER_FULL_VERSION

```
#define EMBER_FULL_VERSION
```

Value:

```
                          (  ((uint16_t)EMBER_MAJOR_VERSION << 12)   \
                          | ((uint16_t)EMBER_MINOR_VERSION <<  8) \
                          | ((uint16_t)EMBER_PATCH_VERSION <<  4) \
                          | ((uint16_t)EMBER_SPECIAL_VERSION))
```

Full version number stored on 2 bytes, with each of the four digits stored on 4 bits.

Definition at line `72` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/config.h`

EMBER_VERSION_TYPE

```
#define EMBER_VERSION_TYPE
```

Value:

```
EMBER_VERSION_TYPE_GA
```

Version type of the release. EMBER_VERSION_TYPE_GA means generally available.

Definition at line `81` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/config.h`

SOFTWARE_VERSION

```
#define SOFTWARE_VERSION
```

Value:

```
EMBER_FULL_VERSION
```

Full version number stored on 2 bytes, with each of the four digits stored on 4 bits.

Definition at line `86` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/config.h`

Connect Data Types

Connect Data Types

Definitions of Connect data types used by various Connect API functions.

See ember-types.h for source code.

Modules

EmberNetworkParameters

EmberIncomingMessage

EmberOutgoingMessage

EmberMacAddress

EmberMacFrame

EmberIncomingMacMessage

EmberOutgoingMacMessage

EmberKeyData

EventActions

Event_s

EventQueue_s

EmberEventControl

EmberEventData_S

EmberTaskControl

Enumerations

```
enum    EmberNodeType {
          EMBER_UNKNOWN_DEVICE = 0
          EMBER_STAR_COORDINATOR = 1
          EMBER_STAR_RANGE_EXTENDER = 2
          EMBER_STAR_END_DEVICE = 3
          EMBER_STAR_SLEEPY_END_DEVICE = 4
          EMBER_DIRECT_DEVICE = 5
          EMBER_MAC_MODE_DEVICE = 6
          EMBER_MAC_MODE_SLEEPY_DEVICE = 7
        }
```

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

```
enum    EmberNetworkStatus {
          EMBER_NO_NETWORK
          EMBER_JOINING_NETWORK
          EMBER_JOINED_NETWORK
          EMBER_RADIO_TEST
```

}
Defines the possible join states for a node.

enum EmberChildFlags {

```
EMBER_CHILD_FLAGS_DEVICE_IS_RANGE_EXTENDER_BIT = 0x02
EMBER_CHILD_FLAGS_DEVICE_IS_SLEEPY_BIT = 0x04
EMBER_CHILD_FLAGS_HAVE_PENDING_DATA_BIT = 0x08
EMBER_CHILD_FLAGS_AES_SECURITY_CAPABLE_BIT = 0x10
EMBER_CHILD_FLAG_DEVICE_IS_EXTENDED_BIT = 0x20
```

}
Child flags.

enum EmberMessageOptions {

```
EMBER_OPTIONS_NONE = 0x00
EMBER_OPTIONS_SECURITY_ENABLED = 0x01
EMBER_OPTIONS_ACK_REQUESTED = 0x02
EMBER_OPTIONS_HIGH_PRIORITY = 0x04
EMBER_OPTIONS_INDIRECT = 0x08
```

}
Message options.

enum EmberMacAddressMode {

```
EMBER_MAC_ADDRESS_MODE_NONE = 0x00
EMBER_MAC_ADDRESS_MODE_SHORT = 0x02
EMBER_MAC_ADDRESS_MODE_LONG = 0x03
```

}
802.15.4 addressing mode.

enum EmberEventUnits {

```
EMBER_EVENT_INACTIVE = 0
EMBER_EVENT_MS_TIME
EMBER_EVENT_QS_TIME
EMBER_EVENT_MINUTE_TIME
EMBER_EVENT_ZERO_DELAY
```

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

enum @2 {

```
EMBER_OUTGOING_MESSAGES = 0x0001
EMBER_INCOMING_MESSAGES = 0x0002
EMBER_RADIO_IS_ON = 0x0004
EMBER_ASSOCIATING = 0x0008
EMBER_SCANNING = 0x0010
```

}
Define tasks that prevent the stack from sleeping.

enum EmberCounterType {

EMBER_COUNTER_PHY_IN_PACKETS
EMBER_COUNTER_PHY_OUT_PACKETS
EMBER_COUNTER_MAC_IN_UNICAST
EMBER_COUNTER_MAC_IN_BROADCAST
EMBER_COUNTER_MAC_OUT_UNICAST_NO_ACK
EMBER_COUNTER_MAC_OUT_UNICAST_ACK_SUCCESS
EMBER_COUNTER_MAC_OUT_UNICAST_ACK_FAIL
EMBER_COUNTER_MAC_OUT_UNICAST_CCA_FAIL
EMBER_COUNTER_MAC_OUT_UNICAST_RETRY
EMBER_COUNTER_MAC_OUT_BROADCAST
EMBER_COUNTER_MAC_OUT_BROADCAST_CCA_FAIL
EMBER_COUNTER_MAC_OUT_ENCRYPT_FAIL
EMBER_COUNTER_MAC_DROP_IN_MEMORY
EMBER_COUNTER_MAC_DROP_IN_FRAME_COUNTER
EMBER_COUNTER_MAC_DROP_IN_DECRYPT
EMBER_COUNTER_NWK_OUT_FORWARDING
EMBER_COUNTER_NWK_IN_SUCCESS
EMBER_COUNTER_NWK_DROP_IN_WRONG_SOURCE
EMBER_COUNTER_NWK_DROP_IN_FORWARDING
EMBER_COUNTER_UART_IN_DATA
EMBER_COUNTER_UART_IN_MANAGEMENT
EMBER_COUNTER_UART_IN_FAIL
EMBER_COUNTER_UART_OUT_DATA
EMBER_COUNTER_UART_OUT_MANAGEMENT
EMBER_COUNTER_UART_OUT_FAIL
EMBER_COUNTER_ROUTE_2_HOP_LOOP
EMBER_COUNTER_BUFFER_ALLOCATION_FAIL
EMBER_ASH_V3_ACK_SENT
EMBER_ASH_V3_ACK_RECEIVED
EMBER_ASH_V3_NACK_SENT
EMBER_ASH_V3_NACK_RECEIVED
EMBER_ASH_V3_RESEND
EMBER_ASH_V3_BYTES_SENT
EMBER_ASH_V3_TOTAL_BYTES_RECEIVED
EMBER_ASH_V3_VALID_BYTES_RECEIVED
EMBER_ASH_V3_PAYLOAD_BYTES_SENT
EMBER_COUNTER_TYPE_COUNT

}
Define the event counters that can be requested from the application using emberGetCounter()

enum EmberPhyType {

EMBER_RADIO_CONFIGURATOR
EMBER_STANDARD_PHY_2_4GHZ
EMBER_STANDARD_PHY_915MHZ
EMBER_STANDARD_PHY_863MHZ

}
Define the PHY configuration of connect stack.

enum EmberCalType {

EMBER_CAL_TEMP_VCO = 0x00000001
EMBER_CAL_IRCAL = 0x00010000
EMBER_CAL_ALL = 0x00010001

}
Define the type of calibration requested.

enum EmberTxStreamParameters {

TX_STREAM_PN9
TX_STREAM_CW

}
Radio Stream mode.

Typedefs

typedef uint8_t	EmberEUI64[EUI64_SIZE] EUI 64-bit ID (IEEE 802.15.4 long address).
typedef uint16_t	EmberNodeId IEEE 802.15.4 node ID. Also known as short address.
typedef uint16_t	EmberPanId IEEE 802.15.4 PAN ID.
typedef uint16_t	EmberMessageLength Message length in bytes.
typedef uint8_t	EmberTaskId An identifier for a task.
typedef struct Event_s	Event
typedef struct EventQueue_s	EventQueue An event queue is currently just a list of events ordered by execution time.
typedef const struct EmberEventData_S	EmberEventData Complete events with a control and a handler procedure.
typedef uint16_t	EmberBuffer Buffers used by the memory buffer system.

Functions

uint8_t *	emberKeyContents(EmberKeyData *key) This macro allows the programmer to gain access to the key data bytes of the EmberKeyData structure.

Macros

#define	EXTENDED_PAN_ID_SIZE 8 Size of an extended PAN identifier in bytes (8).
#define	EUI64_SIZE 8 Size of EUI64 (an IEEE address) in bytes (8).
#define	EMBER_ENCRYPTION_KEY_SIZE 16 Size of an encryption key in bytes (16).
#define	EMBER_NULL_NODE_ID 0xFFFFu A distinguished network ID that will never be assigned to any node. Used to indicate the absence of a node ID.
#define	EMBER_BROADCAST_ADDRESS 0xFFFF Broadcast address.
#define	EMBER_USE_LONG_ADDRESS 0xFFFE Special short address indicating the node should use long addressing as source address.
#define	EMBER_COORDINATOR_ADDRESS 0x0000 The coordinator short address.
#define	EMBER_CAL_INVALID_VALUE (0xFFFFFFFF)

Enumeration Documentation

EmberNodeType

```
EmberNodeType
```

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

Enumerator	
EMBER_UNKNOWN_DEVICE	
EMBER_STAR_COORDINATOR	
EMBER_STAR_RANGE_EXTENDER	
EMBER_STAR_END_DEVICE	
EMBER_STAR_SLEEPY_END_DEVICE	
EMBER_DIRECT_DEVICE	
EMBER_MAC_MODE_DEVICE	
EMBER_MAC_MODE_SLEEPY_DEVICE	

Definition at line `106` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

EmberNetworkStatus

```
EmberNetworkStatus
```

Defines the possible join states for a node.

Enumerator	
EMBER_NO_NETWORK	
EMBER_JOINING_NETWORK	
EMBER_JOINED_NETWORK	
EMBER_RADIO_TEST	

Definition at line `162` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

EmberChildFlags

```
EmberChildFlags
```

Child flags.

Enumerator	
EMBER_CHILD_FLAGS_DEVICE_IS_RANGE_EXTENDER_BIT	
EMBER_CHILD_FLAGS_DEVICE_IS_SLEEPY_BIT	
EMBER_CHILD_FLAGS_HAVE_PENDING_DATA_BIT	
EMBER_CHILD_FLAGS_AES_SECURITY_CAPABLE_BIT	
EMBER_CHILD_FLAG_DEVICE_IS_EXTENDED_BIT	

Definition at line `198` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

EmberMessageOptions

> EmberMessageOptions

Message options.

Enumerator	
EMBER_OPTIONS_NONE	
EMBER_OPTIONS_SECURITY_ENABLED	
EMBER_OPTIONS_ACK_REQUESTED	
EMBER_OPTIONS_HIGH_PRIORITY	
EMBER_OPTIONS_INDIRECT	

Definition at line `225` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

EmberMacAddressMode

> EmberMacAddressMode

802.15.4 addressing mode.

Enumerator	
EMBER_MAC_ADDRESS_MODE_NONE	
EMBER_MAC_ADDRESS_MODE_SHORT	
EMBER_MAC_ADDRESS_MODE_LONG	

Definition at line `338` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

EmberEventUnits

> EmberEventUnits

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

Enumerator	
EMBER_EVENT_INACTIVE	
EMBER_EVENT_MS_TIME	
EMBER_EVENT_QS_TIME	
EMBER_EVENT_MINUTE_TIME	
EMBER_EVENT_ZERO_DELAY	

Definition at line `521` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

@2

> @2

Define tasks that prevent the stack from sleeping.

Enumerator	
EMBER_OUTGOING_MESSAGES	
EMBER_INCOMING_MESSAGES	
EMBER_RADIO_IS_ON	

EMBER_ASSOCIATING	
EMBER_SCANNING	

Definition at line `623` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

EmberCounterType

> EmberCounterType

Define the event counters that can be requested from the application using emberGetCounter()

Enumerator	
EMBER_COUNTER_PHY_IN_PACKETS	
EMBER_COUNTER_PHY_OUT_PACKETS	
EMBER_COUNTER_MAC_IN_UNICAST	
EMBER_COUNTER_MAC_IN_BROADCAST	
EMBER_COUNTER_MAC_OUT_UNICAST_NO_ACK	
EMBER_COUNTER_MAC_OUT_UNICAST_ACK_SUCCESS	
EMBER_COUNTER_MAC_OUT_UNICAST_ACK_FAIL	
EMBER_COUNTER_MAC_OUT_UNICAST_CCA_FAIL	
EMBER_COUNTER_MAC_OUT_UNICAST_RETRY	
EMBER_COUNTER_MAC_OUT_BROADCAST	
EMBER_COUNTER_MAC_OUT_BROADCAST_CCA_FAIL	
EMBER_COUNTER_MAC_OUT_ENCRYPT_FAIL	
EMBER_COUNTER_MAC_DROP_IN_MEMORY	
EMBER_COUNTER_MAC_DROP_IN_FRAME_COUNTER	
EMBER_COUNTER_MAC_DROP_IN_DECRYPT	
EMBER_COUNTER_NWK_OUT_FORWARDING	
EMBER_COUNTER_NWK_IN_SUCCESS	
EMBER_COUNTER_NWK_DROP_IN_WRONG_SOURCE	
EMBER_COUNTER_NWK_DROP_IN_FORWARDING	
EMBER_COUNTER_UART_IN_DATA	
EMBER_COUNTER_UART_IN_MANAGEMENT	
EMBER_COUNTER_UART_IN_FAIL	
EMBER_COUNTER_UART_OUT_DATA	
EMBER_COUNTER_UART_OUT_MANAGEMENT	
EMBER_COUNTER_UART_OUT_FAIL	
EMBER_COUNTER_ROUTE_2_HOP_LOOP	
EMBER_COUNTER_BUFFER_ALLOCATION_FAIL	
EMBER_ASH_V3_ACK_SENT	
EMBER_ASH_V3_ACK_RECEIVED	
EMBER_ASH_V3_NACK_SENT	
EMBER_ASH_V3_NACK_RECEIVED	
EMBER_ASH_V3_RESEND	
EMBER_ASH_V3_BYTES_SENT	
EMBER_ASH_V3_TOTAL_BYTES_RECEIVED	
EMBER_ASH_V3_VALID_BYTES_RECEIVED	
EMBER_ASH_V3_PAYLOAD_BYTES_SENT	
EMBER_COUNTER_TYPE_COUNT	

Definition at line 646 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h

EmberPhyType

```
EmberPhyType
```

Define the PHY configuration of connect stack.

Enumerator	
EMBER_RADIO_CONFIGURATOR	
EMBER_STANDARD_PHY_2_4GHZ	
EMBER_STANDARD_PHY_915MHZ	
EMBER_STANDARD_PHY_863MHZ	

Definition at line 765 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h

EmberCalType

```
EmberCalType
```

Define the type of calibration requested.

Enumerator	
EMBER_CAL_TEMP_VCO	
EMBER_CAL_IRCAL	
EMBER_CAL_ALL	

Definition at line 788 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h

EmberTxStreamParameters

```
EmberTxStreamParameters
```

Radio Stream mode.

Enumerator	
TX_STREAM_PN9	
TX_STREAM_CW	

Definition at line 874 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h

Typedef Documentation

EmberEUI64

```
typedef uint8_t EmberEUI64[EUI64_SIZE] [EUI64_SIZE]
```

EUI 64-bit ID (IEEE 802.15.4 long address).

Definition at line 73 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h

EmberNodeId

```
typedef uint16_t EmberNodeId
```

IEEE 802.15.4 node ID. Also known as short address.

Definition at line `78` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

EmberPanId

```
typedef uint16_t EmberPanId
```

IEEE 802.15.4 PAN ID.

Definition at line `83` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

EmberMessageLength

```
typedef uint16_t EmberMessageLength
```

Message length in bytes.

Definition at line `219` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

EmberTaskId

```
typedef uint8_t EmberTaskId
```

An identifier for a task.

Definition at line `540` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

Event

```
typedef struct Event_s Event
```

Definition at line `567` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

EventQueue

```
typedef struct EventQueue_s EventQueue
```

An event queue is currently just a list of events ordered by execution time.

Definition at line `575` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

EmberEventData

```
typedef const struct EmberEventData_S EmberEventData
```

Complete events with a control and a handler procedure.

An application typically creates an array of events along with their handlers. The main loop passes the array to emberRunEvents() to call the handlers of any events whose time has arrived.

Definition at line `606` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

EmberBuffer

```
typedef uint16_t EmberBuffer
```

Buffers used by the memory buffer system.

Definition at line `759` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

Function Documentation

emberKeyContents

```
uint8_t * emberKeyContents (EmberKeyData *key)
```

This macro allows the programmer to gain access to the key data bytes of the EmberKeyData structure.

Parameters

[in]	key	A Pointer to an EmberKeyData structure.

Returns

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

Definition at line `511` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

Macro Definition Documentation

EXTENDED_PAN_ID_SIZE

```
#define EXTENDED_PAN_ID_SIZE
```

Value:

```
8
```

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

Definition at line `58` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

EUI64_SIZE

```
#define EUI64_SIZE
```

Value:

```
8
```

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

Definition at line `63` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

EMBER_ENCRYPTION_KEY_SIZE

```
#define EMBER_ENCRYPTION_KEY_SIZE
```

Value:

```
16
```

Size of an encryption key in bytes (16).

Definition at line `68` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

EMBER_NULL_NODE_ID

```
#define EMBER_NULL_NODE_ID
```

Value:

```
0xFFFFu
```

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

Definition at line `89` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

EMBER_BROADCAST_ADDRESS

```
#define EMBER_BROADCAST_ADDRESS
```

Value:

```
0xFFFF
```

Broadcast address.

Definition at line `92` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

EMBER_USE_LONG_ADDRESS

```
#define EMBER_USE_LONG_ADDRESS
```

Value:

```
0xFFFE
```

Special short address indicating the node should use long addressing as source address.

Definition at line `96` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

EMBER_COORDINATOR_ADDRESS

```
#define EMBER_COORDINATOR_ADDRESS
```

Value:

```
0x0000
```

The coordinator short address.

Definition at line `99` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

EMBER_CAL_INVALID_VALUE

```
#define EMBER_CAL_INVALID_VALUE
```

Value:

```
(0xFFFFFFFF)
```

Definition at line `804` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

EmberNetworkParameters

Hold network parameters.

For information about power settings and radio channels, see the technical specification for the RF communication module in your Developer Kit and the Radio Configurator Guide (AN971).

Public Attributes

uint16_t	panId
int16_t	radioTxPower
uint16_t	radioChannel

Public Attribute Documentation

panId

```
uint16_t EmberNetworkParameters::panId
```

The network's PAN identifier.

Definition at line `187` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

radioTxPower

```
int16_t EmberNetworkParameters::radioTxPower
```

The transmit power setting, in deci-dBm.

Definition at line `189` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

radioChannel

```
uint16_t EmberNetworkParameters::radioChannel
```

The radio channel. Be sure to specify a channel supported by the radio.

Definition at line `191` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

EmberIncomingMessage

An instance of this structure is passed to emberIncomingMessageHandler(). It describes the incoming message.

Public Attributes

EmberMessageOptions	options
EmberNodeId	source
uint8_t	endpoint
int8_t	rssi
EmberMessageLength	length
uint8_t *	payload
uint32_t	timestamp
uint8_t	lqi

Public Attribute Documentation

options

```
EmberMessageOptions EmberIncomingMessage::options
```

An EmberMessageOptions value indicating the options used for the incoming packet.

Definition at line `256` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

source

```
EmberNodeId EmberIncomingMessage::source
```

An EmberNodeId value indicating source node ID.

Definition at line `260` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

endpoint

```
uint8_t EmberIncomingMessage::endpoint
```

The endpoint the message is destined to.

Definition at line `264` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

```
int8_t EmberIncomingMessage::rssi
```

The RSSI in dBm the packet was received with.

Definition at line `268` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

length

```
EmberMessageLength EmberIncomingMessage::length
```

An EmberMessageLength value indicating the length in bytes of the incoming message.

Definition at line `273` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

payload

```
uint8_t* EmberIncomingMessage::payload
```

A pointer to the message payload.

Definition at line `277` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

timestamp

```
uint32_t EmberIncomingMessage::timestamp
```

The millisecond system time returned by emberGetInt32uMillisecondTick() at the time the sync word was detected.

Definition at line `282` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

lqi

```
uint8_t EmberIncomingMessage::lqi
```

The LQI the packet was received with.

Definition at line `286` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

EmberOutgoingMessage

An instance of this structure is passed to emberMessageSentHandler(). It describes the outgoing packet.

Public Attributes

EmberMessageOptions	options
EmberNodeId	destination
uint8_t	endpoint
uint8_t	tag
EmberMessageLength	length
uint8_t *	payload
int8_t	ackRssi
uint32_t	timestamp

Public Attribute Documentation

options

```
EmberMessageOptions EmberOutgoingMessage::options
```

An EmberMessageOptions value indicating the options used for transmitting the outgoing message.

Definition at line `298` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

destination

```
EmberNodeId EmberOutgoingMessage::destination
```

An EmberNodeId value indicating the destination short ID.

Definition at line `302` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

endpoint

```
uint8_t EmberOutgoingMessage::endpoint
```

The endpoint the message is destined to.

Definition at line `306` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

```
uint8_t EmberOutgoingMessage::tag
```

A tag value the application can use to match emberMessageSend() calls to the corresponding emberMessageSentHandler() calls.

Definition at line `311` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

length

```
EmberMessageLength EmberOutgoingMessage::length
```

An EmberMessageLength value indicating the length in bytes of the incoming message.

Definition at line `316` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

payload

```
uint8_t* EmberOutgoingMessage::payload
```

A pointer to the message payload.

Definition at line `320` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

ackRssi

```
int8_t EmberOutgoingMessage::ackRssi
```

The RSSI in dBm of the ACK corresponding to this message. This field is meaningful only if EMBER_OPTIONS_ACK_REQUESTED flag is set in the options field.

Definition at line `326` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

timestamp

```
uint32_t EmberOutgoingMessage::timestamp
```

The millisecond system time returned by ::sl_sleeptimer at the time the sync word was transmitted.

Definition at line `331` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

EmberMacAddress

A structure that stores an 802.15.4 address.

Public Attributes

uint8_t	longAddress Long (EUI-64) address. Valid if `mode` is EMBER_MAC_ADDRESS_MODE_LONG.
uint16_t	shortAddress Short address (node ID). Valid if `mode` is EMBER_MAC_ADDRESS_MODE_SHORT.
union EmberMacAddress::@3	addr
EmberMacAddressMode	mode

Public Attribute Documentation

longAddress

```
uint8_t EmberMacAddress::longAddress[EUI64_SIZE]
```

Long (EUI-64) address. Valid if `mode` is EMBER_MAC_ADDRESS_MODE_LONG.

Definition at line `361` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

shortAddress

```
uint16_t EmberMacAddress::shortAddress
```

Short address (node ID). Valid if `mode` is EMBER_MAC_ADDRESS_MODE_SHORT.

Definition at line `366` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

addr

```
union EmberMacAddress::@3 EmberMacAddress::addr
```

Definition at line `367` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

mode

```
EmberMacAddressMode EmberMacAddress::mode
```

Addressing mode

Definition at line `369` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

EmberMacFrame

A structure that describes the addressing fields of a 802.15.4 frame.

Public Attributes

EmberMacAddress	srcAddress
EmberMacAddress	dstAddress
EmberPanId	srcPanId
EmberPanId	dstPanId
bool	srcPanIdSpecified
bool	dstPanIdSpecified

Public Attribute Documentation

srcAddress

```
EmberMacAddress EmberMacFrame::srcAddress
```

An EmberMacAddress structure indicating the source address of a MAC frame.

Definition at line `380` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

dstAddress

```
EmberMacAddress EmberMacFrame::dstAddress
```

An EmberMacAddress structure indicating the destination address of a MAC frame.

Definition at line `385` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

srcPanId

```
EmberPanId EmberMacFrame::srcPanId
```

An EmberPanId struct indicating the source PAN ID of a MAC frame. This field is meaningful only if srcPanIdSpecified is set to true.

Definition at line `390` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

dstPanId

```
EmberPanId EmberMacFrame::dstPanId
```

An EmberPanId struct indicating the destination PAN ID of a MAC frame. This field is meaningful only if dstPanIdSpecified is set to true.

Definition at line `395` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

srcPanIdSpecified

```
bool EmberMacFrame::srcPanIdSpecified
```

True if the srcPanId field is set, false otherwise.

Definition at line `399` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

dstPanIdSpecified

```
bool EmberMacFrame::dstPanIdSpecified
```

True if the dstPanId field is set, false otherwise.

Definition at line `403` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

EmberIncomingMacMessage

An instance of this structure is passed to emberIncomingMacMessageHandler(). It describes the incoming MAC frame.

Public Attributes

EmberMessageOptions	options
EmberMacFrame	macFrame
int8_t	rssi
uint8_t	lqi
uint32_t	frameCounter
EmberMessageLength	length
uint8_t *	payload
uint32_t	timestamp

Public Attribute Documentation

options

```
EmberMessageOptions EmberIncomingMacMessage::options
```

An EmberMessageOptions value indicating the options used for the incoming packet.

Definition at line `415` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

macFrame

```
EmberMacFrame EmberIncomingMacMessage::macFrame
```

An EmberMacFrame structure indicating the source and destination addresses and source and destination PAN IDs.

Definition at line `420` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

rssi

```
int8_t EmberIncomingMacMessage::rssi
```

The RSSI in dBm the packet was received with.

Definition at line `424` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

```
uint8_t EmberIncomingMacMessage::lqi
```

The LQI the packet was received with.

Definition at line `428` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

frameCounter

```
uint32_t EmberIncomingMacMessage::frameCounter
```

The security MAC frame counter (if any).

Definition at line `432` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

length

```
EmberMessageLength EmberIncomingMacMessage::length
```

An EmberMessageLength value indicating the length in bytes of the MAC payload of the incoming message.

Definition at line `437` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

payload

```
uint8_t* EmberIncomingMacMessage::payload
```

A pointer to the message MAC payload.

Definition at line `441` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

timestamp

```
uint32_t EmberIncomingMacMessage::timestamp
```

The millisecond system time returned by ::sl_sleeptimer at the time the sync word was detected.

Definition at line `446` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

EmberOutgoingMacMessage

An instance of this structure is passed to emberMacMessageSentHandler(). It describes the outgoing MAC frame.

Public Attributes

EmberMessageOptions	options
EmberMacFrame	macFrame
uint8_t	tag
uint32_t	frameCounter
EmberMessageLength	length
uint8_t *	payload
int8_t	ackRssi
uint32_t	timestamp

Public Attribute Documentation

options

```
EmberMessageOptions EmberOutgoingMacMessage::options
```

An EmberMessageOptions value indicating the options used for transmitting the outgoing message.

Definition at line `458` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

macFrame

```
EmberMacFrame EmberOutgoingMacMessage::macFrame
```

An EmberMacFrame struct indicating the source and destination addresses and source and destination PAN IDs of the outgoing MAC frame.

Definition at line `463` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

tag

```
uint8_t EmberOutgoingMacMessage::tag
```

A tag value the application can use to match emberMacMessageSend() calls to the corresponding emberMacMessageSentHandler() calls.

Definition at line `468` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

frameCounter

```
uint32_t EmberOutgoingMacMessage::frameCounter
```

The security frame counter of the outgoing MAC frame (if any).

Definition at line `472` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

length

```
EmberMessageLength EmberOutgoingMacMessage::length
```

An EmberMessageLength value indicating the length in bytes of the incoming message.

Definition at line `477` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

payload

```
uint8_t* EmberOutgoingMacMessage::payload
```

A pointer to the message payload.

Definition at line `481` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

ackRssi

```
int8_t EmberOutgoingMacMessage::ackRssi
```

The RSSI in dBm of the ACK corresponding to this message. This field is meaningful only if EMBER_OPTIONS_ACK_REQUESTED flag is set in the options field.

Definition at line `487` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

timestamp

```
uint32_t EmberOutgoingMacMessage::timestamp
```

The millisecond system time returned by ::sl_sleeptimer at the time the sync word was transmitted.

Definition at line `492` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

EmberKeyData

This data structure contains the security key, most prominently used by emberSetSecurityKey.

Public Attributes

uint8_t	contents

Public Attribute Documentation

contents

```
uint8_t EmberKeyData::contents[EMBER_ENCRYPTION_KEY_SIZE]
```

This is the key byte data.

Definition at line `500` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

EventActions

The static part of an event. Each event can be used with only one event queue.

Public Attributes

struct EventQueue_s *	queue
void(*	handler
void(*	marker
const char *	name

Public Attribute Documentation

queue

```
struct EventQueue_s* EventActions::queue
```

Definition at line `554` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

handler

```
void(* EventActions::handler) (struct Event_s *)
```

Definition at line `555` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

marker

```
void(* EventActions::marker) (struct Event_s *)
```

Definition at line `556` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

name

```
const char* EventActions::name
```

Definition at line `557` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

Event_s

Public Attributes

EventActions *	actions
struct Event_s *	next
uint32_t	timeToExecute

Public Attribute Documentation

actions

```
EventActions* Event_s::actions
```

Definition at line `561` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

next

```
struct Event_s* Event_s::next
```

Definition at line `565` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

timeToExecute

```
uint32_t Event_s::timeToExecute
```

Definition at line `566` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

EventQueue_s

An event queue is currently just a list of events ordered by execution time.

Public Attributes

Event *	isrEvents
Event *	events

Public Attribute Documentation

isrEvents

```
Event* EventQueue_s::isrEvents
```

Definition at line `573` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

events

```
Event* EventQueue_s::events
```

Definition at line `574` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

EmberEventControl

Control structure for events.

This structure should not be accessed directly. It holds the event status (one of the **EMBER_EVENT_** values) and the time left before the event fires.

Public Attributes

EmberEventUnits	status
EmberTaskId	taskid
uint32_t	timeToExecute

Public Attribute Documentation

status

```
EmberEventUnits EmberEventControl::status
```

The event's status, either inactive or the units for timeToExecute.

Definition at line `585` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

taskid

```
EmberTaskId EmberEventControl::taskid
```

The task ID this event belongs to.

Definition at line `587` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

timeToExecute

```
uint32_t EmberEventControl::timeToExecute
```

How long before the event fires. Units are always in milliseconds.

Definition at line `591` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

EmberEventData_S

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() to call the handlers of any events whose time has arrived.

Public Attributes

EmberEventControl *	control
void(*	handler

Public Attribute Documentation

control

```
EmberEventControl* EmberEventData_S::control
```

The control structure for the event.

Definition at line `603` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

handler

```
void(* EmberEventData_S::handler) (void)
```

The procedure to call when the event fires.

Definition at line `605` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

EmberTaskControl

Control structure for tasks.

This structure should not be accessed directly.

Public Attributes

uint32_t	nextEventTime
EmberEventData *	events
bool	busy

Public Attribute Documentation

nextEventTime

```
uint32_t EmberTaskControl::nextEventTime
```

The time when the next event associated with this task will fire

Definition at line `614` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

events

```
EmberEventData* EmberTaskControl::events
```

The list of events associated with this task

Definition at line `616` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

busy

```
bool EmberTaskControl::busy
```

A flag that indicates the task has something to do other than events

Definition at line `618` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/ember-types.h`

Stack Information

Stack Information

Connect API for accessing and modifying stack states and behaviors.

See stack-info.h for source code.

Modules

Stack Counters

Handlers

void	emberStackStatusHandler(EmberStatus status) A callback invoked when the status of the stack changes.
void	emberStackIsrHandler(void) This handler is invoked in ISR context when certain stack-related ISR routines fire.
bool	emberStackIdleHandler(uint32_t *idleTimeMs) A callback to allow the application to manage idling the MCU.
void	emberRadioNeedsCalibratingHandler(void) The radio calibration callback function.
void	emberChildJoinHandler(EmberNodeType nodeType, EmberNodeId nodeId) Invoked at coordinator, range extender, or mac mode nodes when a new child has joined the device.

APIs

void	emberStackPowerDown(void) Immediately turns the radio power completely off.
void	emberStackPowerUp(void) Power up the radio. Typically called coming out of sleep.
EmberNetworkStatus	emberNetworkState(void) Return the current join status.
bool	emberStackIsUp(void) Indicate whether the stack is currently up.
EmberStatus	emberSetSecurityKey(EmberKeyData *key) Write a key at the address of the formerly used security key. The key set by this function will not be used by the stack. The API is meant to be used to erase a key as it is now managed by PSA Crypto API.
EmberStatus	emberGetSecurityKey(EmberKeyData *key) Get the legacy security key. This function does not return the value set with the PSA Crypto API.
EmberStatus	emberSetRadioChannelExtended(uint16_t channel, bool persistent) Set the channel for sending and receiving messages on the current network. The available channels depend on the radio config you use. Channels can differ more than the frequency if it's a multi-PHY config.

EmberStatus	emberSetRadioChannel(uint16_t channel) Set the channel for sending and receiving messages on the current network. The available channels depend on the radio config you use. Channels can differ more than the frequency if it's a multi-PHY config.
uint16_t	emberGetRadioChannel(void) Get the radio channel, to which a node is set, on the current network. The available channels depend on the radio config you use. Channels can differ more than the frequency if it's a multi-PHY config.
uint16_t	emberGetDefaultChannel(void) Get the first available channel in the current radio configuration.
EmberStatus	emberPhyConfigInit(EmberPhyType phyType) Indicate if the PHY configuration of the stack. Currently only supporting EMBER_RADIO_CONFIGURATOR and EMBER_STANDARD_PHY_2_4GHZ. It must be called before initializing the stack.
EmberStatus	emberCalibrateCurrentChannelExtended(uint32_t calValueIn, uint32_t *calValueOut) Perform image rejection calibration on the current channel. The stack will notify the application that it needs channel calibration via the emberRadioNeedsCalibratingHandler() callback function during emberTick(). This function should only be called from within the context of the emberRadioNeedsCalibratingHandler() callback function. Note if this function is called when the radio is off, it will turn the radio on and leave it on.
EmberStatus	emberCalibrateCurrentChannel(void) Perform image rejection calibration on the current channel. The stack will notify the application that it needs channel calibration via the emberRadioNeedsCalibratingHandler() callback function during emberTick(). This function should only be called from within the context of the emberRadioNeedsCalibratingHandler() callback function. Note if this function is called when the radio is off, it will turn the radio on and leave it on.
EmberStatus	emberApplyIrCalibration(uint32_t calValue) Apply Image Rejection calibration on the current channel. The stack will notify the application that it needs channel calibration via the emberRadioNeedsCalibratingHandler() callback function during emberTick(). This function should only be called from within the context of the emberRadioNeedsCalibratingHandler() callback function. Note if this function is called when the radio is off, it will turn the radio on and leave it on.
EmberStatus	emberTempCalibration(void) Perform Temperature VCO calibration calibration on the current channel. The stack will notify the application that it needs channel calibration via the emberRadioNeedsCalibratingHandler() callback function during emberTick(). This function should only be called from within the context of the emberRadioNeedsCalibratingHandler() callback function. Note if this function is called when the radio is off, it will turn the radio on and leave it on.
EmberCalType	emberGetCalType(void) Fetch calibration type associated to the latest emberRadioNeedsCalibratingHandler() callback.
EmberStatus	emberSetRadioPower(int16_t power, bool persistent) Set the radio output power at which a node is to operate for the current network. The radio has a finite power resolution, so it will approximate the requested power with the closest possible value at or below the requested value.
int16_t	emberGetRadioPower(void) Get the radio output power of the current network at which a node is operating. This might be different to what you set using emberSetRadioPower because the radio has a finite power resolution, and emberSetRadioPower must approximate to the closest possible value at or below the requested value. This API however returns with the actual setting.
EmberStatus	emberSetRadioPowerMode(bool radioOn) Allow the application to turn the radio on/off. This API is intended for use with direct devices only.
EmberStatus	emberSetMacParams(int8_t ccaThreshold, uint8_t maxCcaAttempts, uint8_t minBackoffExp, uint8_t maxBackoffExp, uint16_t ccaBackoff, uint16_t ccaDuration, uint8_t maxRetries, uint32_t csmaTimeout, uint16_t ackTimeout) Set the MAC layer transmission parameters.
EmberStatus	emberMacGetParentAddress(EmberMacAddress *parentAddress) Retrieve the parent address. This API can be invoked only for nodes of EMBER_MAC_MODE_DEVICE or EMBER_MAC_MODE_SLEEPY_DEVICE type.

uint32_t	emberStackIdleTimeMs(uint16_t *currentStackTasks) Return the time in milliseconds the stack could idle for.
uint32_t	emberGetInt32uMillisecondTick(void) Return the current time in milliseconds.
uint16_t	emberCurrentStackTasks(void) Return a bitmask indicating the stack's current tasks.
bool	emberOkToNap(void) Indicate whether the stack is currently in a state with no high-priority tasks and may sleep.
bool	emberOkToHibernate(void) Indicate whether the stack currently has any pending tasks.
uint8_t *	emberGetEui64(void) Return the EUI64 ID of the local node.
bool	emberIsLocalEui64(EmberEUI64 eui64) Determine whether `eui64` is the local node's EUI64 ID. EUI64 is easily accessible in SoC mode, but in Host-NCP, the address is stored on the NCP. This API can be used on the Host to compare a value with the locally stored one.
EmberNodeId	emberGetNodeId(void) Return the 16-bit node ID of local node on the current network.
EmberPanId	emberGetPanId(void) Return the local node's PAN ID of the current network.
EmberNodeType	emberGetNodeType(void) Return an EmberNodeType value indicating the type of the node.
EmberNodeId	emberGetParentId(void) Return the parent's node ID.
EmberStatus	emberGetVersionInfo(uint16_t *gsdk_version, uint16_t *connect_stack_version, uint32_t *bootloader_version) Get the GSDK, Stack and bootloader versions all at once. The version format are not all the same. Please refer to the corresponding documentation to handle the information correctly.

Macros

#define	EMBER_HIGH_PRIORITY_TASKS (EMBER_OUTGOING_MESSAGES | EMBER_INCOMING_MESSAGES | EMBER_RADIO_IS_ON) A mask of the tasks that prevent a device from sleeping.
#define	EMBER_INVALID_CHANNEL 65535 Invalid channel number.

Handlers Documentation

emberStackStatusHandler

```
void emberStackStatusHandler (EmberStatus status)
```

A callback invoked when the status of the stack changes.

Parameters

[in]	status	Stack status. One of the following: • EMBER_NETWORK_UP • EMBER_NETWORK_DOWN • EMBER_NO_VALID_BEACONS • EMBER_JOIN_SCAN_FAILED • EMBER_JOIN_FAILED • EMBER_JOIN_DENIED • EMBER_JOIN_TIMEOUT • EMBER_MAC_SYNC_TIMEOUT

The application is free to begin messaging once it receives the EMBER_NETWORK_UP status.

Definition at line `72` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h`

emberStackIsrHandler

```
void emberStackIsrHandler (void)
```

This handler is invoked in ISR context when certain stack-related ISR routines fire.

Parameters

N/A		

Definition at line `77` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h`

emberStackIdleHandler

```
bool emberStackIdleHandler (uint32_t *idleTimeMs)
```

A callback to allow the application to manage idling the MCU.

Parameters

[inout]	idleTimeMs	A pointer to the time in millisecond the stack is allowed to idle. If the application decides to manage idling the MCU, it should update the passed value with the actual time the MCU was idled.

Returns

- true if the application is managing idling the MCU, false otherwise. If this function returns false, the stack will manage idling the MCU.

Definition at line `88` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h`

emberRadioNeedsCalibratingHandler

```
void emberRadioNeedsCalibratingHandler (void)
```

The radio calibration callback function.

Parameters

N/A		

This handler is invoked by the stack upon receiving a "calibration needed" event from the radio to inform the application that it should perform calibration of the current channel as soon as possible using the emberCalibrateCurrentChannel() API.

While calibration only takes tens of microseconds, the application can failsafe any critical processes or peripherals before calling emberCalibrateCurrentChannel(). The application must call emberCalibrateCurrentChannel() in response to this callback to maintain expected radio performance.

Definition at line `102` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h`

emberChildJoinHandler

```
void emberChildJoinHandler (EmberNodeType nodeType, EmberNodeId nodeId)
```

Invoked at coordinator, range extender, or mac mode nodes when a new child has joined the device.

Parameters

[in]	nodeType	The role of the joining device (EMBER_STAR_RANGE_EXTENDER, EMBER_STAR_END_DEVICE, EMBER_STAR_SLEEPY_END_DEVICE, EMBER_MAC_MODE_DEVICE or EMBER_MAC_MODE_SLEEPY_DEVICE).
[in]	nodeId	The node ID of the joining device.

Definition at line `601` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h`

APIs Documentation

emberStackPowerDown

```
void emberStackPowerDown (void)
```

Immediately turns the radio power completely off.

Parameters

N/A		

After calling this function, do not call any other stack function except emberStackPowerUp() because all other stack functions require that the radio is powered to operate properly.

Definition at line `118` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h`

emberStackPowerUp

```
void emberStackPowerUp (void)
```

Power up the radio. Typically called coming out of sleep.

Parameters

N/A		

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

Definition at line `124` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h`

emberNetworkState

```
EmberNetworkStatus emberNetworkState (void)
```

Return the current join status.

Parameters

N/A		

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

Returns

- An EmberNetworkStatus value indicating the current join status.

Definition at line `133` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h`

emberStackIsUp

```
bool emberStackIsUp (void)
```

Indicate whether the stack is currently up.

Parameters

N/A		

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 and does not indicate whether or not other nodes are able to communicate with this node.

Returns

- **true** if the stack is up, **false** otherwise.

Definition at line `144` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h`

emberSetSecurityKey

```
EmberStatus emberSetSecurityKey (EmberKeyData *key)
```

Write a key at the address of the formerly used security key. The key set by this function will not be used by the stack. The API is meant to be used to erase a key as it is now managed by PSA Crypto API.

Parameters

[in]	key	An EmberKeyData value containing the security key to be set.

Returns

- An EmberStatus value of EMBER_SUCCESS if the key was successfully set. Otherwise, it returns an EmberStatus value of EMBER_INVALID_CALL.

Definition at line `156` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h`

emberGetSecurityKey

```
EmberStatus emberGetSecurityKey (EmberKeyData *key)
```

Get the legacy security key. This function does not return the value set with the PSA Crypto API.

Parameters

[in]	key	An EmberKeyData where the legcay security key will be stored

Returns

- An EmberStatus value of EMBER_SUCCESS if the key was successfully read.

Definition at line `166` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h`

emberSetRadioChannelExtended

```
EmberStatus emberSetRadioChannelExtended (uint16_t channel, bool persistent)
```

Set the channel for sending and receiving messages on the current network. The available channels depend on the radio config you use. Channels can differ more than the frequency if it's a multi-PHY config.

Parameters

[in]	channel	A desired radio channel.
[in]	persistent	A flag to instruct the stack to save the channel setting in persistent or not. Each persistent call triggers a token write. Excessive usage might cause flash to wear-out.

Note

- Care should be taken when using this API. All devices on a network must use the same channel.

Returns

- An EmberStatus value of:
  - EMBER_SUCCESS if the stack accepted the channel change.
  - EMBER_INVALID_CALL if the node is currently performing frequency hopping.
  - EMBER_PHY_INVALID_CHANNEL if the passed channel is invalid.
  - EMBER_MAC_BUSY if the MAC is currently performing a high priority task.

Definition at line `222` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h`

emberSetRadioChannel

```
EmberStatus emberSetRadioChannel (uint16_t channel)
```

Set the channel for sending and receiving messages on the current network. The available channels depend on the radio config you use. Channels can differ more than the frequency if it's a multi-PHY config.

Parameters

[in]	channel	A desired radio channel.

Note

- Care should be taken when using this API. All devices on a network must use the same channel. Each call triggers a token write. Excessive usage might cause flash to wear-out.

Returns

- An EmberStatus value of:
  - EMBER_SUCCESS if the stack accepted the channel change.
  - EMBER_INVALID_CALL if the node is currently performing frequency hopping.
  - EMBER_PHY_INVALID_CHANNEL if the passed channel is invalid.
  - EMBER_MAC_BUSY if the MAC is currently performing a high priority task.

Definition at line 242 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h

emberGetRadioChannel

```
uint16_t emberGetRadioChannel (void)
```

Get the radio channel, to which a node is set, on the current network. The available channels depend on the radio config you use. Channels can differ more than the frequency if it's a multi-PHY config.

Parameters

N/A		

Returns

- The current radio channel.

Definition at line 250 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h

emberGetDefaultChannel

```
uint16_t emberGetDefaultChannel (void)
```

Get the first available channel in the current radio configuration.

Parameters

N/A		

Returns

- The first available channel in the radio configuration. 0xffff if error

Definition at line 257 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h

emberPhyConfigInit

```
EmberStatus emberPhyConfigInit (EmberPhyType phyType)
```

Indicate if the PHY configuration of the stack. Currently only supporting EMBER_RADIO_CONFIGURATOR and EMBER_STANDARD_PHY_2_4GHZ. It must be called before initializing the stack.

Parameters

N/A	phyType	

Returns

- EMBER_BAD_ARGUMENT if phyType is incorrect EMBER_INVALID_CALL if API is called after stack initialization EMBER_SUCCESS otherwise

Definition at line 267 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h

emberCalibrateCurrentChannelExtended

```
EmberStatus emberCalibrateCurrentChannelExtended (uint32_t calValueIn, uint32_t *calValueOut)
```

Perform image rejection calibration on the current channel. The stack will notify the application that it needs channel calibration via the emberRadioNeedsCalibratingHandler() callback function during emberTick(). This function should only be called from within the context of the emberRadioNeedsCalibratingHandler() callback function. Note if this function is called when the radio is off, it will turn the radio on and leave it on.

Parameters

[in]	calValueIn	the calibration value to use. Set to EMBER_CAL_INVALID_VALUE to perform automatic calibration.
[out]	calValueOut	a pointer to the calibration value that was used. This parameter is ignored when set to NULL.

Returns

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

Definition at line 286 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h

emberCalibrateCurrentChannel

```
EmberStatus emberCalibrateCurrentChannel (void)
```

Perform image rejection calibration on the current channel. The stack will notify the application that it needs channel calibration via the emberRadioNeedsCalibratingHandler() callback function during emberTick(). This function should only be called from within the context of the emberRadioNeedsCalibratingHandler() callback function. Note if this function is called when the radio is off, it will turn the radio on and leave it on.

Parameters

N/A		

Definition at line 297 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h

emberApplyIrCalibration

```
EmberStatus emberApplyIrCalibration (uint32_t calValue)
```

Apply Image Rejection calibration on the current channel. The stack will notify the application that it needs channel calibration via the emberRadioNeedsCalibratingHandler() callback function during emberTick(). This function should only be called from within the context of the emberRadioNeedsCalibratingHandler() callback function. Note if this function is called when the radio is off, it will turn the radio on and leave it on.

Parameters

[in]	calValue	the calibration value to apply. Should not be set to EMBER_CAL_INVALID_VALUE.

Returns

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

Definition at line 313 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h

emberTempCalibration

```
EmberStatus emberTempCalibration (void)
```

Perform Temperature VCO calibration calibration on the current channel. The stack will notify the application that it needs channel calibration via the emberRadioNeedsCalibratingHandler() callback function during emberTick(). This function should only be called from within the context of the emberRadioNeedsCalibratingHandler() callback function. Note if this function is called when the radio is off, it will turn the radio on and leave it on.

Parameters

N/A		

Returns

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

Definition at line `326` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h`

emberGetCalType

```
EmberCalType emberGetCalType (void)
```

Fetch calibration type associated to the latest emberRadioNeedsCalibratingHandler() callback.

Parameters

N/A		

Returns

- An EmberCalType value indicating which type of calibration should be performed.

Definition at line `334` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h`

emberSetRadioPower

```
EmberStatus emberSetRadioPower (int16_t power, bool persistent)
```

Set the radio output power at which a node is to operate for the current network. The radio has a finite power resolution, so it will approximate the requested power with the closest possible value at or below the requested value.

Parameters

[in]	power	Desired radio output power, in deci-dBm.
[in]	persistent	A flag to instruct the stack to save the power setting in persistent or not.

Note

- Care should be taken when using this API on a running network, because it directly impacts 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.

Returns

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

Definition at line `353` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h`

emberGetRadioPower

```
int16_t emberGetRadioPower (void)
```

Get the radio output power of the current network at which a node is operating. This might be different to what you set using emberSetRadioPower because the radio has a finite power resolution, and emberSetRadioPower must approximate to the closest possible value at or below the requested value. This API however returns with the actual setting.

Parameters

N/A		

Returns

- Current radio output power, in deci-dBm.

Definition at line `363` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h`

emberSetRadioPowerMode

```
EmberStatus emberSetRadioPowerMode (bool radioOn)
```

Allow the application to turn the radio on/off. This API is intended for use with direct devices only.

Parameters

[in]	radioOn	If this parameter is **true**, the radio is turned on, otherwise it's turned off.

Returns

- An EmberStatus value indicating the success or failure of the command. Failure indicates that the node type is a type other than EMBER_DIRECT_DEVICE.

Definition at line `375` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h`

emberSetMacParams

```
EmberStatus emberSetMacParams (int8_t ccaThreshold, uint8_t maxCcaAttempts, uint8_t minBackoffExp, uint8_t
maxBackoffExp, uint16_t ccaBackoff, uint16_t ccaDuration, uint8_t maxRetries, uint32_t csmaTimeout, uint16_t ackTimeout)
```

Set the MAC layer transmission parameters.

Parameters

[in]	ccaThreshold	The CCA RSSI threshold, in dBm, above which the channel is considered 'busy'. This parameter is by default set to **EMBER_RADIO_CCA_THRESHOLD**.
[in]	maxCcaAttempts	The maximum number of clear channel assessment attempts that are performed prior to fail to transmit a packet with EMBER_PHY_TX_CCA_FAIL status. This parameter is set by default to 4. If this parameter is set to 0, the CCA assessment shall not be performed.
[in]	minBackoffExp	The backoff exponent used if the initial channel clear assessment fails. This parameter is set by default to 3. Note: this is meaningful only if the checkCca parameter is set to **true**.
[in]	maxBackoffExp	The backoff exponent used if the final channel clear assessment fails. This parameter is set by default to 5. Note: this is meaningful only if the checkCca parameter is set to **true**.
[in]	ccaBackoff	The backoff unit period in microsecond. It is multiplied by the random backoff exponential controlled by minBackoffExp and maxBackoffExp to determine the overall backoff period. This parameter is set by default to the PHY symbol time in microseconds multiplied by 20.
[in]	ccaDuration	The minimum desired CCA check duration in microseconds. This parameter is set by default to the PHY symbol time in microseconds multiplied by 8.

[in]	maxRetries	The number of transmission retries that is performed if no acknowledgment was received. This parameter is set by default to 3 (which means that a total of 4 transmission attempts will be performed).
[in]	csmaTimeout	An overall timeout in microsecond time base for the the CSMA operations. This value is set by default to 0 which means that no timeout is imposed.
[in]	ackTimeout	The ack timeout in microseconds after which the transmitting gives up waiting for an acknowledgment. This parameter is set by default to (EMBER_MAC_ACK_TIMEOUT_MS * 1000).

Note

- The CSMA/CA (CCA) values are directly used in RAIL's `RAIL_CsmaConfig_t` and further information can be found in the RAIL API documentation.

Returns

- An EmberStatus value indicating whether the MAC parameters were successfully set or the reason of failure.

Definition at line `432` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h`

emberMacGetParentAddress

```
EmberStatus emberMacGetParentAddress (EmberMacAddress *parentAddress)
```

Retrieve the parent address. This API can be invoked only for nodes of EMBER_MAC_MODE_DEVICE or EMBER_MAC_MODE_SLEEPY_DEVICE type.

Parameters

N/A	parentAddress	

Returns

- An EmberStatus value of EMBER_SUCCESS if the parent address was successfully retrieved, otherwise an EmberStatus value indicating the reason of failure.

Definition at line `449` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h`

emberStackIdleTimeMs

```
uint32_t emberStackIdleTimeMs (uint16_t *currentStackTasks)
```

Return the time in milliseconds the stack could idle for.

Parameters

[in]	currentStackTasks	A pointer to an integer that is written with the active stack tasks at the time of the API call.

Returns

- Allowed idle time in milliseconds.

Definition at line `458` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h`

emberGetInt32uMillisecondTick

```
uint32_t emberGetInt32uMillisecondTick (void)
```

Return the current time in milliseconds.

Parameters

N/A		

Returns

- Current time in milliseconds.

Definition at line `464` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h`

emberCurrentStackTasks

```
uint16_t emberCurrentStackTasks (void)
```

Return a bitmask indicating the stack's current tasks.

Parameters

N/A		

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() to receive messages. Parents will hold messages for EMBER_INDIRECT_TRANSMISSION_TIMEOUT_MS milliseconds before discarding them.

Returns

- A bitmask of the stack's active tasks.

Definition at line `479` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h`

emberOkToNap

```
bool emberOkToNap (void)
```

Indicate whether the stack is currently in a state with no high-priority tasks and may sleep.

Parameters

N/A		

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

Returns

- **true** if the application may sleep but the stack may be expecting incoming messages.

Definition at line `492` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h`

emberOkToHibernate

```
bool emberOkToHibernate (void)
```

Indicate whether the stack currently has any pending tasks.

Parameters

N/A		

If no tasks are pending , emberTick() does not need to be called until next stack API function is called. This function can only be called when the node type is EMBER_STAR_SLEEPY_END_DEVICE.

Returns

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

Definition at line `502` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h`

emberGetEui64

```
uint8_t * emberGetEui64 (void)
```

Return the EUI64 ID of the local node.

Parameters

N/A		

Returns

- The 64-bit ID.

Definition at line `508` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h`

emberIsLocalEui64

```
bool emberIsLocalEui64 (EmberEUI64 eui64)
```

Determine whether `eui64` is the local node's EUI64 ID. EUI64 is easily accessible in SoC mode, but in Host-NCP, the address is stored on the NCP. This API can be used on the Host to compare a value with the locally stored one.

Parameters

[in]	eui64	An EUI64 ID.

Returns

- **true** if `eui64` is the local node's ID, otherwise **false**.

Definition at line `519` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h`

emberGetNodeId

```
EmberNodeId emberGetNodeId (void)
```

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

Parameters

N/A		

Returns

- The 16-bit ID. Byte order is little endian.

Definition at line `525` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h`

emberGetPanId

```
EmberPanId emberGetPanId (void)
```

Return the local node's PAN ID of the current network.

Parameters

N/A		

Returns

- The PAN ID.

Definition at line `531` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h`

emberGetNodeType

```
EmberNodeType emberGetNodeType (void)
```

Return an EmberNodeType value indicating the type of the node.

Parameters

N/A		

Returns

- The node type.

Definition at line `537` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h`

emberGetParentId

```
EmberNodeId emberGetParentId (void)
```

Return the parent's node ID.

Parameters

N/A		

Returns

- The parent's node ID.

Definition at line `543` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h`

emberGetVersionInfo

```
EmberStatus emberGetVersionInfo (uint16_t *gsdk_version, uint16_t *connect_stack_version, uint32_t *bootloader_version)
```

Get the GSDK, Stack and bootloader versions all at once. The version format are not all the same. Please refer to the corresponding documentation to handle the information correctly.

Parameters

N/A	gsdk_version	
N/A	connect_stack_version	
N/A	bootloader_version	

Returns

- EMBER_SUCCESS if successful

Definition at line `550` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h`

Macro Definition Documentation

EMBER_HIGH_PRIORITY_TASKS

```
#define EMBER_HIGH_PRIORITY_TASKS
```

Value:

```
(EMBER_OUTGOING_MESSAGES | EMBER_INCOMING_MESSAGES | EMBER_RADIO_IS_ON)
```

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

Definition at line `44` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h`

EMBER_INVALID_CHANNEL

```
#define EMBER_INVALID_CHANNEL
```

Value:

```
65535
```

Invalid channel number.

Definition at line `51` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h`

Stack Counters

Stack Counters

Stack counters API.

Functions

EmberStatus	emberGetCounter(EmberCounterType counterType, uint32_t *count) Retrieve the stack counter corresponding to the passed counter type.

Function Documentation

emberGetCounter

```
EmberStatus emberGetCounter (EmberCounterType counterType, uint32_t *count)
```

Retrieve the stack counter corresponding to the passed counter type.

Parameters

[in]	counterType	An EmberCounterType value indicating the stack counter to be retrieved.
[out]	count	The counter of the requested `counterType` is returned here

Returns

- An EmberStatus value of EMBER_SUCCESS if the stack counter was successfully retrieved. An EmberStatus value of EMBER_INVALID_CALL if the passed counterType is invalid. An EmberStatus value of EMBER_LIBRARY_NOT_PRESENT if the stack counter library is not present.

Definition at line `578` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h`

Network Management

Network Management

Connect API for finding, forming, joining, and leaving Connect networks.

See network-management.h for source code.

Modules

Frequency Hopping

Parent Support

Handlers

void	emberIncomingBeaconHandler(EmberPanId panId, EmberMacAddress *source, int8_t rssi, bool permitJoining, uint8_t beaconFieldsLength, uint8_t *beaconFields, uint8_t beaconPayloadLength, uint8_t *beaconPayload) Invoked if a beacon is received during the scanning procedure if the handler was initiated by the application with the emberStartActiveScan() stack APIs.
void	emberActiveScanCompleteHandler(void) Invoked after the application calls the emberStartActiveScan() stack API to inform the application that the scanning procedure is complete.
void	emberEnergyScanCompleteHandler(int8_t mean, int8_t min, int8_t max, uint16_t variance) Invoked after the application calls the emberStartEnergyScan() stack API to inform the application that the energy scan procedure is complete and to provide statistics.

Functions

EmberStatus	emberInit(void) Initialize the radio and the Ember stack.
void	emberTick(void) A periodic tick routine that should be called in the main loop in the application.
EmberStatus	emberNetworkInit(void) Resume the network operation after a reboot.
EmberStatus	emberStartActiveScan(uint16_t channel) Start an active scan. EMBER_SUCCESS signals that the scan successfully started. Upon receiving a beacon, the emberIncomingBeaconHandler() stack handler is called. At the end of the scanning procedure, the emberActiveScanCompleteHandler() stack handler is called. Note that, while a scan can be initiated when the node is currently joined to a network, the node will generally be unable to communicate with its PAN during the scan period. In particular, time-sensitive network operations might be affected because a scan operation will prevent any network operation for the duration of the scan.
EmberStatus	emberSetActiveScanDuration(uint16_t durationMs) Set the time in milliseconds the node will spend listening for incoming beacons during an active scan. The default value is set based on the symbol time of the current PHY configuration according to the 802.15.4 specs.
uint16_t	emberGetActiveScanDuration(void) Get the current active scan duration in milliseconds.

EmberStatus	emberStartEnergyScan(uint16_t channel, uint8_t samples) Start an energy scan. EMBER_SUCCESS signals that the scan successfully started. At the end of the scanning procedure, the emberEnergyScanCompleteHandler() stack handler is called. Note that, while a scan can be initiated when the node is currently joined to a network, the node is generally unable to communicate with its PAN during the scan period. In particular, time-sensitive network operations might be affected because a scan operation will prevent any network operation for the duration of the scan.
EmberStatus	emberSetApplicationBeaconPayload(uint8_t payloadLength, uint8_t *payload) Allow the application to set the application portion of the beacon payload. It's by default set to the empty string.
EmberStatus	emberJoinNetworkExtended(EmberNodeType nodeType, EmberNodeId nodeId, EmberNetworkParameters *parameters) Cause 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. Notice that forming a network causes the node's security frame counter to be reset.
EmberStatus	emberJoinNetwork(EmberNodeType nodeType, EmberNetworkParameters *parameters) Cause the stack to associate with the network using the specified network parameters. The network ID is assigned by the network coordinator. 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. Notice that joining a network causes the node's security frame counter to be reset.
EmberStatus	emberPermitJoining(uint8_t duration) Tell 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.
EmberStatus	emberJoinCommissioned(EmberNodeType nodeType, EmberNodeId nodeId, EmberNetworkParameters *parameters) Cause the stack to go up with the passed network parameters without performing any over-the-air message exchange. Notice that commissioning a network causes the node's security frame counter to be reset.
EmberStatus	emberSetSelectiveJoinPayload(uint8_t payloadLength, uint8_t *payload) When invoked at a EMBER_STAR_COORDINATOR or a EMBER_STAR_RANGE_EXTENDER, it causes the stack to only accept subsequent joining nodes with matching joining payload. When invoked at a node that has not yet joined a network, it sets the joining payload that will be included in the joining process. Notice, the join payload is included in a non-standard 802.15.4 command, therefore this feature is not available for nodes operating as EMBER_MAC_MODE_DEVICE or EMBER_MAC_MODE_SLEEPY_DEVICE.
EmberStatus	emberClearSelectiveJoinPayload(void) Clear the join payload previously set with the emberSetSelectiveJoinPayload() API. When invoked at an EMBER_STAR_COORDINATOR or an EMBER_STAR_RANGE_EXTENDER it causes the stack to accept joining nodes with any join payload pattern. When invoked at a node that has not yet joined a network, it clears the join payload. Subsequent joining attempts will not include any join payload in the over-the-air joining handshake.
EmberStatus	emberSetAuxiliaryAddressFilteringEntry(EmberNodeId nodeId, uint8_t entryIndex) Set an entry in the auxiliary address filtering table at a given address. Nodes of EMBER_DIRECT_DEVICE device type can receive incoming messages destined to any of the node IDs in the auxiliary address filtering table (while also receiving messages destined to actual node ID). If the passed node ID is EMBER_NULL_NODE_ID, the entry is cleared.
EmberNodeId	emberGetAuxiliaryAddressFilteringEntry(uint8_t entryIndex) Retrieve the content of the auxiliary address filtering table at a given address. See emberSetAuxiliaryAddressFilteringEntry() for details.
void	emberResetNetworkState(void) Forget the current network and reverts to a network status of EMBER_NO_NETWORK.
EmberStatus	emberMacFormNetwork(EmberNetworkParameters *parameters) Form a new network as an EMBER_MAC_MODE_DEVICE by becoming the coordinator. This API should be used to form a compliant 802.15.4 PAN and to inter-operate with other 802.15.4 devices. Notice that forming a network causes the node's security frame counter to be reset.

EmberStatus	emberMacSetPanCoordinator(bool isCoordinator) Configure a EMBER_MAC_MODE_DEVICE node to be a PAN coordinator. Note, this only applies to nodes that have been commissioned as EMBER_MAC_MODE_DEVICE.
EmberStatus	emberNetworkLeave(void) Allow a star topology node that previously joined a network to leave the network. The node will notify the parent node and eventually leave the network. The application is notified that the leave procedure completed via the emberStackStatusHandler() handler.
EmberStatus	emberMacAddShortToLongAddressMapping(EmberNodeId shortId, EmberEUI64 longId) Populate the short-to-long address mapping table at the MAC layer. The table is meaningful only when running as EMBER_MAC_MODE_DEVICE or EMBER_MAC_MODE_SLEEPY_DEVICE. The standard 802.15.4 encryption and authentication process requires the security nonce to be populated with the source node long ID. A receiver must do the same to decrypt a secured incoming message. This short-to-long mapping table is used to decrypt a secured incoming packet from a node using short source addressing. If no entry is found in this table, the incoming message will be dropped. This table is also used to encrypt secured outgoing messages with short source addressing in case the node is sending out a secured message with a short source address other than its own.
EmberStatus	emberMacClearShortToLongAddressMappings(void) Clear the short-to-long address mapping table at the MAC layer.
EmberStatus	emberOfdmSetMcs(uint8_t mcs) Set the MCS in case of an OFDM PHY. MCS can range from 0 to 6.
EmberStatus	emberOfdmGetMcs(uint8_t *mcs) Get the MCS in case of an OFDM PHY.

Macros

#define	EMBER_MAC_MAX_APP_BEACON_PAYLOAD_LENGTH 16 The maximum length in bytes of the application beacon payload.
#define	EMBER_MAC_STACK_BEACON_PAYLOAD "silabs-connect"
#define	EMBER_MAC_STACK_BEACON_PAYLOAD_LENGTH 14 The length in bytes of the stack beacon payload.
#define	EMBER_MAC_MAX_BEACON_FIELDS_LENGTH 84 The maximum length in bytes of the beacon fields (superframe, GTS, pending address) as per 802.15.4 specs.
#define	EMBER_CHILD_TABLE_AGING_DISABLED 0×20C400 A special timeout value that disables aging of the child table.
#define	EMBER_CHILD_TABLE_MAX_TIMEOUT_S (EMBER_CHILD_TABLE_AGING_DISABLED - 1) The maximum timeout in seconds after which a stale entry may be removed from the child table.
#define	EMBER_MAX_SELECTIVE_JOIN_PAYLOAD_LENGTH 50 The maximum length in bytes of the join payload.
#define	EMBER_MAX_AUXILIARY_ADDRESS_FILTERING_TABLE_LENGTH 2 The maximum number of entries the auxiliary address filtering table can hold.

Handlers Documentation

emberIncomingBeaconHandler

```
void emberIncomingBeaconHandler (EmberPanId panId, EmberMacAddress *source, int8_t rssi, bool permitJoining, uint8_t
beaconFieldsLength, uint8_t *beaconFields, uint8_t beaconPayloadLength, uint8_t *beaconPayload)
```

Invoked if a beacon is received during the scanning procedure if the handler was initiated by the application with the emberStartActiveScan() stack APIs.

Parameters

[in]	panId	The source pan ID of the received beacon.
[in]	source	The source node address of the received beacon.
[in]	rssi	The RSSI the beacon was received with.
[in]	permitJoining	The permit joining flag in the received beacon.
[in]	beaconFieldsLength	The length in bytes of the beacon fields defined as per 802.15.4 specs (superframe, GTS fields and pending address fields) of the received beacon.
[in]	beaconFields	A pointer to the beacon fields defined as per 802.15.4 specs (superframe, GTS fields and pending address fields) of the received beacon.
[in]	beaconPayloadLength	The length in bytes of the application beacon payload of the received beacon.
[in]	beaconPayload	A pointer to the application beacon payload of the received beacon.

Definition at line `94` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h`

emberActiveScanCompleteHandler

```
void emberActiveScanCompleteHandler (void)
```

Invoked after the application calls the emberStartActiveScan() stack API to inform the application that the scanning procedure is complete.

Parameters

N/A		

Definition at line `106` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h`

emberEnergyScanCompleteHandler

```
void emberEnergyScanCompleteHandler (int8_t mean, int8_t min, int8_t max, uint16_t variance)
```

Invoked after the application calls the emberStartEnergyScan() stack API to inform the application that the energy scan procedure is complete and to provide statistics.

Parameters

[in]	mean	The average energy detected in dBm.
[in]	min	The minimum energy detected in dBm.
[in]	max	The maximum energy detected in dBm.
[in]	variance	The variance of the energy detected in dBm.

Definition at line `117` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h`

Function Documentation

emberInit

```
EmberStatus emberInit (void)
```

Initialize the radio and the Ember stack.

Parameters

N/A		

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 are not available. Some failure modes are not fatal, but the application must follow certain procedures to permit recovery. Ignoring the return code results 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.

Definition at line `141` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h`

emberTick

```
void emberTick (void)
```

A periodic tick routine that should be called in the main loop in the application.

Parameters

N/A		

Definition at line `146` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h`

emberNetworkInit

```
EmberStatus emberNetworkInit (void)
```

Resume the network operation after a reboot.

Parameters

N/A		

This API must be called on boot prior to ANY network operations. It initializes the networking system and attempts to resume the previous network identity and configuration. If the node was not previously joined, 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, and so on.)

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.

Definition at line `165` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h`

emberStartActiveScan

```
EmberStatus emberStartActiveScan (uint16_t channel)
```

Start an active scan. EMBER_SUCCESS signals that the scan successfully started. Upon receiving a beacon, the emberIncomingBeaconHandler() stack handler is called. At the end of the scanning procedure, the emberActiveScanCompleteHandler() stack handler is called. Note that, while a scan can be initiated when the node is currently joined to a network, the node will generally be unable to communicate with its PAN during the scan period. In particular, time-sensitive network operations might be affected because a scan operation will prevent any network operation for the duration of the scan.

Parameters

[in]	channel	The channel to scan.

Possible error responses and their meanings:

- EMBER_INVALID_CALL, the node is currently frequency hopping.
- EMBER_MAC_SCANNING, indicates an ongoing scan.
- EMBER_PHY_INVALID_CHANNEL, the specified channel is not a valid channel on the current platform.

Definition at line `186` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h`

emberSetActiveScanDuration

```
EmberStatus emberSetActiveScanDuration (uint16_t durationMs)
```

Set the time in milliseconds the node will spend listening for incoming beacons during an active scan. The default value is set based on the symbol time of the current PHY configuration according to the 802.15.4 specs.

Parameters

[in]	durationMs	The active scan duration in milliseconds. A value of 0xFFFF restores the default value.

Returns

- an EmberStatus value of EMBER_SUCCESS if the active scan duration was successfully set, otherwise an EmberStatus value indicating the reason of failure.

Definition at line `199` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h`

emberGetActiveScanDuration

```
uint16_t emberGetActiveScanDuration (void)
```

Get the current active scan duration in milliseconds.

Parameters

N/A		

Returns

- a 16-bit integer indicating the current duration in millisecond of the active scan.

Definition at line `206` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h`

emberStartEnergyScan

```
EmberStatus emberStartEnergyScan (uint16_t channel, uint8_t samples)
```

Start an energy scan. EMBER_SUCCESS signals that the scan successfully started. At the end of the scanning procedure, the emberEnergyScanCompleteHandler() stack handler is called. Note that, while a scan can be initiated when the node is currently joined to a network, the node is generally unable to communicate with its PAN during the scan period. In particular, time-sensitive network operations might be affected because a scan operation will prevent any network operation for the duration of the scan.

Parameters

[in]	channel	The channel to scan.
[in]	samples	The number of energy samples to be produced. Each sample is performed averaging the detected energy over X symbols time, whereas X depends on the selected PHY configuration and set by default to 8. The symbol time duration also depends on the selected PHY configuration.

Possible error responses and their meanings:

- EMBER_INVALID_CALL, the node is currently frequency hopping.
- EMBER_BAD_ARGUMENT, the samples parameter is invalid.
- EMBER_MAC_SCANNING, indicates an ongoing scan.
- EMBER_PHY_INVALID_CHANNEL, the specified channel is not a valid channel on the current platform.
- EMBER_NO_BUFFERS, the stack doesn't have enough memory at the moment to perform the requested scan.

Definition at line `233` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h`

emberSetApplicationBeaconPayload

```
EmberStatus emberSetApplicationBeaconPayload (uint8_t payloadLength, uint8_t *payload)
```

Allow the application to set the application portion of the beacon payload. It's by default set to the empty string.

Parameters

[in]	payloadLength	The length in bytes of the application beacon payload to be set. This value can not exceed EMBER_MAC_MAX_APP_BEACON_PAYLOAD_LENGTH.
[out]	payload	A pointer to the application beacon payload to be set.

Returns

- an EmberStatus value of EMBER_SUCCESS if the application beacon payload was successfully set, otherwise an EmberStatus value indicating the reason of failure.

Definition at line `248` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h`

emberJoinNetworkExtended

```
EmberStatus emberJoinNetworkExtended (EmberNodeType nodeType, EmberNodeId nodeId, EmberNetworkParameters
*parameters)
```

Cause 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. Notice that forming a network causes the node's security frame counter to be reset.

Parameters

[in]	nodeType	Specification of the role that this node will have in the network. This role can be EMBER_STAR_RANGE_EXTENDER, EMBER_STAR_END_DEVICE, EMBER_STAR_SLEEPY_END_DEVICE, EMBER_MAC_MODE_DEVICE or EMBER_MAC_MODE_SLEEPY_DEVICE. If the node is frequency hopping, the role can not be EMBER_STAR_RANGE_EXTENDER.
[in]	nodeId	An EmberNodeId value indicating the short ID the node intends to use for addressing purposes. If this value is EMBER_NULL_NODE_ID, the network coordinator will allocate a new short address. Addresses should be allocated by the coordinator unless there is a specific need to join a network with a specific ID. If a specific ID is used, uniqueness should be guaranteed across the entire network by the application, via some out of band means. Notice that nodes of EMBER_MAC_MODE_DEVICE or EMBER_MAC_MODE_SLEEPY_DEVICE require this parameter to be set to EMBER_NULL_NODE_ID.
[in]	parameters	An EmberNetworkParameters value that specifies the network parameters 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.

Definition at line `295` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h`

emberJoinNetwork

```
EmberStatus emberJoinNetwork (EmberNodeType nodeType, EmberNetworkParameters *parameters)
```

Cause the stack to associate with the network using the specified network parameters. The network ID is assigned by the network coordinator. 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. Notice that joining a network causes the node's security frame counter to be reset.

Parameters

[in]	nodeType	Specification of the role that this node will have in the network. This role can be EMBER_STAR_RANGE_EXTENDER, EMBER_STAR_END_DEVICE, EMBER_STAR_SLEEPY_END_DEVICE, EMBER_MAC_MODE_DEVICE or EMBER_MAC_MODE_SLEEPY_DEVICE. If the node is frequency hopping, the role can not be EMBER_STAR_RANGE_EXTENDER.
[in]	parameters	An EmberNetworkParameters value that specifies the network parameters 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.

Definition at line `321` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h`

emberPermitJoining

```
EmberStatus emberPermitJoining (uint8_t duration)
```

Tell 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

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

Returns

- an EmberStatus value of EMBER_SUCCESS if the permit joining was successfully set, otherwise an EmberStatus value indicating the reason of failure.

Definition at line `337` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h`

emberJoinCommissioned

```
EmberStatus emberJoinCommissioned (EmberNodeType nodeType, EmberNodeId nodeId, EmberNetworkParameters
*parameters)
```

Cause the stack to go up with the passed network parameters without performing any over-the-air message exchange. Notice that commissioning a network causes the node's security frame counter to be reset.

Parameters

[in]	nodeType	Specifies the role that this node will have in the network. The only device types allowed in the commissioning API are EMBER_DIRECT_DEVICE, EMBER_MAC_MODE_DEVICE and EMBER_MAC_MODE_SLEEPY_DEVICE.
[in]	nodeId	An EmberNodeId value that specifies the short ID the node will have. The passed node ID must be a valid short address (any value other than EMBER_NULL_NODE_ID or EMBER_BROADCAST_ADDRESS). A value of EMBER_USE_LONG_ADDRESS is allowed only when commissioning the node as EMBER_MAC_MODE_DEVICE or EMBER_MAC_MODE_SLEEPY_DEVICE and will cause the node to send MAC level control messages such as data polls or beacons using long source addressing.
[in]	parameters	An EmberNetworkParameters value that specifies the network parameters of the network the node should participate in.

Returns

- An EmberStatus value that indicates either:
  - that the node successfully commissioned the passed network parameters
  - the reason for failure.

Definition at line `364` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h`

emberSetSelectiveJoinPayload

```
EmberStatus emberSetSelectiveJoinPayload (uint8_t payloadLength, uint8_t *payload)
```

When invoked at a EMBER_STAR_COORDINATOR or a EMBER_STAR_RANGE_EXTENDER, it causes the stack to only accept subsequent joining nodes with matching joining payload. When invoked at a node that has not yet joined a network, it sets the joining payload that will be included in the joining process. Notice, the join payload is included in a non-standard 802.15.4 command, therefore this feature is not available for nodes operating as EMBER_MAC_MODE_DEVICE or EMBER_MAC_MODE_SLEEPY_DEVICE.

Parameters

[in]	payloadLength	The length in bytes of the passed joining payload. This can not exceed EMBER_MAX_SELECTIVE_JOIN_PAYLOAD_LENGTH.
[in]	payload	A pointer to the payload to be set.

Returns

- An EmberStatus value that indicates either:

  - that the node successfully set the join payload.
  - the reason for failure.

Definition at line `389` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h`

emberClearSelectiveJoinPayload

```
EmberStatus emberClearSelectiveJoinPayload (void)
```

Clear the join payload previously set with the emberSetSelectiveJoinPayload() API. When invoked at an EMBER_STAR_COORDINATOR or an EMBER_STAR_RANGE_EXTENDER it causes the stack to accept joining nodes with any join payload pattern. When invoked at a node that has not yet joined a network, it clears the join payload. Subsequent joining attempts will not include any join payload in the over-the-air joining handshake.

Parameters

N/A		

Returns

- An EmberStatus value that indicates either:
  - that the node successfully cleared the join payload.
  - the reason for failure.

Definition at line `404` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h`

emberSetAuxiliaryAddressFilteringEntry

```
EmberStatus emberSetAuxiliaryAddressFilteringEntry (EmberNodeId nodeId, uint8_t entryIndex)
```

Set an entry in the auxiliary address filtering table at a given address. Nodes of EMBER_DIRECT_DEVICE device type can receive incoming messages destined to any of the node IDs in the auxiliary address filtering table (while also receiving messages destined to actual node ID). If the passed node ID is EMBER_NULL_NODE_ID, the entry is cleared.

Parameters

[in]	nodeId	An EmberNodeId value to be added to the auxiliary address filtering table at the passed entry index.
[in]	entryIndex	The index of the auxiliary address filtering table entry to be set.

Returns

- An EmberStatus value of EMBER_SUCCESS if auxiliary address filtering table entry was successfully set. An EmberStatus value of EMBER_INVALID_CALL if the passed entry index is invalid.

Definition at line `427` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h`

emberGetAuxiliaryAddressFilteringEntry

```
EmberNodeId emberGetAuxiliaryAddressFilteringEntry (uint8_t entryIndex)
```

Retrieve the content of the auxiliary address filtering table at a given address. See emberSetAuxiliaryAddressFilteringEntry() for details.

Parameters

[in]	entryIndex	The index in the auxiliary address filtering table entry to be retrieved.

Returns

- An EmberNodeId value of EMBER_NULL_NODE_ID if the passed entry index is invalid or if the passed entry index refers to an unused entry. Otherwise, it returns the content of the auxiliary address filtering table entry corresponding to the passed entry index.

Definition at line `442` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h`

emberResetNetworkState

```
void emberResetNetworkState (void)
```

Forget the current network and reverts to a network status of EMBER_NO_NETWORK.

Parameters

N/A		

Definition at line `447` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h`

emberMacFormNetwork

```
EmberStatus emberMacFormNetwork (EmberNetworkParameters *parameters)
```

Form a new network as an EMBER_MAC_MODE_DEVICE by becoming the coordinator. This API should be used to form a compliant 802.15.4 PAN and to inter-operate with other 802.15.4 devices. Notice that forming a network causes the node's security frame counter to be reset.

Parameters

[in]	parameters	An EmberNetworkParameters value that specifies the network parameters of the network to be formed.

Returns

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

Definition at line `461` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h`

emberMacSetPanCoordinator

```
EmberStatus emberMacSetPanCoordinator (bool isCoordinator)
```

Configure a EMBER_MAC_MODE_DEVICE node to be a PAN coordinator. Note, this only applies to nodes that have been commissioned as EMBER_MAC_MODE_DEVICE.

Parameters

[in]	isCoordinator	If set to true, the node will identify itself as the PAN coordinator.

Returns

- An EmberStatus value of EMBER_SUCCESS if the coordinator flag was successfully set, or another EmberStatus value indicating the reason of failure.

Definition at line `474` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h`

emberNetworkLeave

```
EmberStatus emberNetworkLeave (void)
```

Allow a star topology node that previously joined a network to leave the network. The node will notify the parent node and eventually leave the network. The application is notified that the leave procedure completed via the emberStackStatusHandler() handler.

Parameters

N/A		

Returns

- An EmberStatus value of EMBER_SUCCESS if the node successfully initiated the network leave procedure, or another EmberStatus value indicating the reason of failure.

Definition at line `548` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h`

emberMacAddShortToLongAddressMapping

```
EmberStatus emberMacAddShortToLongAddressMapping (EmberNodeId shortId, EmberEUI64 longId)
```

Populate the short-to-long address mapping table at the MAC layer. The table is meaningful only when running as EMBER_MAC_MODE_DEVICE or EMBER_MAC_MODE_SLEEPY_DEVICE. The standard 802.15.4 encryption and authentication process requires the security nonce to be populated with the source node long ID. A receiver must do the same to decrypt a secured incoming message. This short-to-long mapping table is used to decrypt a secured incoming packet from a node using short source addressing. If no entry is found in this table, the incoming message will be dropped. This table is also used to encrypt secured outgoing messages with short source addressing in case the node is sending out a secured message with a short source address other than its own.

Parameters

[in]	shortId	The short address of the [short, long] entry to be added to the table.
[in]	longId	The long address of the [short, long] entry to be added to the table.

Note

- Because the table is stored in RAM, the application should ensure it gets correctly re-populated upon reboot.
- Adding a new entry will cause the removal of existing entries matching the passed short ID or long ID.

Returns

- an EmberStatus value of:
  - EMBER_SUCCESS if the mapping was successfully added to the table.
  - EMBER_INVALID_CALL if the node is not running as EMBER_MAC_MODE_DEVICE or as EMBER_MAC_MODE_SLEEPY_DEVICE.
  - EMBER_TABLE_FULL if the table is currently full.
  - EMBER_NO_BUFFERS if the heap does not currently have enough space for the new entry. The size of the table is controlled by the EMBER_SECURITY_SHORT_TO_LONG_MAPPING_TABLE_SIZE.

Definition at line `584` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h`

emberMacClearShortToLongAddressMappings

```
EmberStatus emberMacClearShortToLongAddressMappings (void)
```

Clear the short-to-long address mapping table at the MAC layer.

Parameters

N/A		

Returns

- an EmberStatus value of EMBER_SUCCESS if table was cleared, or another EmberStatus value indicating the reason of failure.

Definition at line 593 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h

emberOfdmSetMcs

```
EmberStatus emberOfdmSetMcs (uint8_t mcs)
```

Set the MCS in case of an OFDM PHY. MCS can range from 0 to 6.

Parameters

[in]	mcs	The MCS value to set.

Returns

- an EmberStatus value of EMBER_SUCCESS if the MCS is valid, an EmberStatus value of EMBER_INVALID_CALL if the current PHY is not OFDM or an EmberStatus value of EMBER_BAD_ARGUMENT if the MCS value is not valid.

Definition at line 605 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h

emberOfdmGetMcs

```
EmberStatus emberOfdmGetMcs (uint8_t *mcs)
```

Get the MCS in case of an OFDM PHY.

Parameters

[out]	mcs	A pointer to the uint8_t that should hold the current MCS value.

Returns

- an EmberStatus value of EMBER_INVALID_CALL if the current PHY is not OFDM or an EmberStatus value of EMBER_SUCCESS if the current PHY is OFDM.

Definition at line 617 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h

Macro Definition Documentation

EMBER_MAC_MAX_APP_BEACON_PAYLOAD_LENGTH

```
#define EMBER_MAC_MAX_APP_BEACON_PAYLOAD_LENGTH
```

Value:

```
16
```

The maximum length in bytes of the application beacon payload.

Definition at line 44 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h

EMBER_MAC_STACK_BEACON_PAYLOAD

```
#define EMBER_MAC_STACK_BEACON_PAYLOAD
```

Value:

```
"silabs-connect"
```

Definition at line 45 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h

EMBER_MAC_STACK_BEACON_PAYLOAD_LENGTH

```
#define EMBER_MAC_STACK_BEACON_PAYLOAD_LENGTH
```

Value:

```
14
```

The length in bytes of the stack beacon payload.

Definition at line 48 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h

EMBER_MAC_MAX_BEACON_FIELDS_LENGTH

```
#define EMBER_MAC_MAX_BEACON_FIELDS_LENGTH
```

Value:

```
84
```

The maximum length in bytes of the beacon fields (superframe, GTS, pending address) as per 802.15.4 specs.

Definition at line 53 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h

EMBER_CHILD_TABLE_AGING_DISABLED

```
#define EMBER_CHILD_TABLE_AGING_DISABLED
```

Value:

```
0x20C400
```

A special timeout value that disables aging of the child table.

Definition at line 57 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h

EMBER_CHILD_TABLE_MAX_TIMEOUT_S

```
#define EMBER_CHILD_TABLE_MAX_TIMEOUT_S
```

Value:

```
(EMBER_CHILD_TABLE_AGING_DISABLED - 1)
```

The maximum timeout in seconds after which a stale entry may be removed from the child table.

Definition at line `62` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h`

EMBER_MAX_SELECTIVE_JOIN_PAYLOAD_LENGTH

```
#define EMBER_MAX_SELECTIVE_JOIN_PAYLOAD_LENGTH
```

Value:

```
50
```

The maximum length in bytes of the join payload.

Definition at line `370` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h`

EMBER_MAX_AUXILIARY_ADDRESS_FILTERING_TABLE_LENGTH

```
#define EMBER_MAX_AUXILIARY_ADDRESS_FILTERING_TABLE_LENGTH
```

Value:

```
2
```

The maximum number of entries the auxiliary address filtering table can hold.

Definition at line `409` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h`

Frequency Hopping

Frequency Hopping

API and callbacks for frequency hopping configuration.

See network-management.h for source code.

Callbacks

void	emberFrequencyHoppingStartClientCompleteHandler(EmberStatus status) This stack handler is invoked after the application calls the emberFrequencyHoppingStartClient() stack API to inform the application that the synchronization process with the server is complete. See emberFrequencyHoppingStartClient() for details.

Functions

EmberStatus	emberFrequencyHoppingSetChannelMask(uint8_t channelMaskLength, uint8_t *channelMask) Set the channel mask for frequency hopping. This API can only be invoked when the node is not frequency hopping.
EmberStatus	emberFrequencyHoppingStartServer(void) Start the device operating as a frequency hopping server. This API can only be invoked when the node is joined to a network. Notice that the server upon starting hopping shall perform an initial advertisement across the entire channel hopping sequence. This is done to resynchronize clients in case the server was started as result of a reboot.
EmberStatus	emberFrequencyHoppingStartClient(EmberNodeId serverNodeId, EmberPanId serverPanId) Start operating as a frequency hopping client and synchronize with the specified server. This API can be invoked on nodes that are already joined to a network (with the exception of nodes started as EMBER_MAC_MODE_DEVICE or EMBER_MAC_MODE_SLEEPY_DEVICE) and nodes that are not joined to a network yet. If the node is already performing frequency hopping, this API returns EMBER_INVALID_CALL. If this API returns EMBER_SUCCESS, the emberFrequencyHoppingStartClientCompleteHandler() is invoked asynchronously to inform the application whether the node successfully synchronized with the specified server or to inform the application of the reason of failure. After the client is synced to a server, it may seamlessly perform the resynchronization process if needed. Sleepy devices in particular periodically perform the resynchronization process. If the client fails a resynchronization process, it informs the application by invoking the emberStackStatusHandler() handler with EMBER_MAC_SYNC_TIMEOUT status. When this occurs, the client will no longer be synced to the server. The application may elect to attempt a new synchronization process by invoking this API again.
EmberStatus	emberFrequencyHoppingStop(void) Stop frequency hopping. This API can only be invoked when the node is frequency hopping. Applicable for both server and client.

Callbacks Documentation

emberFrequencyHoppingStartClientCompleteHandler

```
void emberFrequencyHoppingStartClientCompleteHandler (EmberStatus status)
```

This stack handler is invoked after the application calls the emberFrequencyHoppingStartClient() stack API to inform the application that the synchronization process with the server is complete. See emberFrequencyHoppingStartClient() for details.

Parameters

[in]	status	An EmberStatus value indicating whether the synchronization process with the server was completed successfully or the reason for failure.

Definition at line `733` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h`

Function Documentation

emberFrequencyHoppingSetChannelMask

```
EmberStatus emberFrequencyHoppingSetChannelMask (uint8_t channelMaskLength, uint8_t *channelMask)
```

Set the channel mask for frequency hopping. This API can only be invoked when the node is not frequency hopping.

Parameters

[in]	channelMaskLength	Length of the bitmap in bytes
[in]	channelMask	A pointer to a bitmap representing allowed channels for frequency hopping.

Note

- The application is responsible for applying this setting to both the server and clients.

Note

- The bitmap size needs to be at least (EMBER_FREQUENCY_HOPPING_END_CHANNEL + 8) >> 3 or an error is thrown.
- The bitmap needs to be set again after stopping frequency hopping.

Returns

- An EmberStatus value of EMBER_SUCCESS if the node successfully set the bitmask. An EmberStatus value of EMBER_INVALID_CALL if the node is currently performing frequency hopping. An EmberStatus value of EMBER_BAD_ARGUMENT if the resulting channel list is empty, or if channelMaskLength is shorter than expected.

Definition at line `661` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h`

emberFrequencyHoppingStartServer

```
EmberStatus emberFrequencyHoppingStartServer (void)
```

Start the device operating as a frequency hopping server. This API can only be invoked when the node is joined to a network. Notice that the server upon starting hopping shall perform an initial advertisement across the entire channel hopping sequence. This is done to resynchronize clients in case the server was started as result of a reboot.

Parameters

N/A		

Returns

- An EmberStatus value of EMBER_SUCCESS if the node successfully initiated frequency hopping server operations. An EmberStatus value of EMBER_INVALID_CALL if the node is not currently joined to a network or if the node is already performing frequency hopping.

Definition at line `675` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h`

emberFrequencyHoppingStartClient

```
EmberStatus emberFrequencyHoppingStartClient (EmberNodeId serverNodeId, EmberPanId serverPanId)
```

Start operating as a frequency hopping client and synchronize with the specified server. This API can be invoked on nodes that are already joined to a network (with the exception of nodes started as EMBER_MAC_MODE_DEVICE or EMBER_MAC_MODE_SLEEPY_DEVICE) and nodes that are not joined to a network yet. If the node is already performing frequency hopping, this API returns EMBER_INVALID_CALL. If this API returns EMBER_SUCCESS, the emberFrequencyHoppingStartClientCompleteHandler() is invoked asynchronously to inform the application whether the node successfully synchronized with the specified server or to inform the application of the reason of failure. After the client is synced to a server, it may seamlessly perform the resynchronization process if needed. Sleepy devices in particular periodically perform the resynchronization process. If the client fails a resynchronization process, it informs the application by invoking the emberStackStatusHandler() handler with EMBER_MAC_SYNC_TIMEOUT status. When this occurs, the client will no longer be synced to the server. The application may elect to attempt a new synchronization process by invoking this API again.

Parameters

[in]	serverNodeId	An EmberNodeId value indicating the node ID of the server to synchronize with.
[in]	serverPanId	An EmberPanId value indicating the PAN ID of the server to synchronize with. Note that this parameter is meaningful only if the node is not currently joined to any network.

Returns

- An EmberStatus value of EMBER_SUCCESS indicating that the node successfully initiated the synchronization process with the server, otherwise an EmberStatus value indicating the reason of failure.

Definition at line `708` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h`

emberFrequencyHoppingStop

```
EmberStatus emberFrequencyHoppingStop (void)
```

Stop frequency hopping. This API can only be invoked when the node is frequency hopping. Applicable for both server and client.

Parameters

N/A		

Returns

- An EmberStatus value of EMBER_SUCCESS indicating that the node successfully stopped frequency hopping. An EmberStatus value of EMBER_INVALID_CALL if the node is not currently frequency hopping.

Definition at line `718` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h`

Parent Support

Parent Support

Handlers

void	emberChildJoinHandler(EmberNodeType nodeType, EmberNodeId nodeId) Invoked at coordinator, range extender, or mac mode nodes when a new child has joined the device.

Handlers

The Application Framework implements all handlers, directly calling their associated callbacks. By default, Connect projects declare such callbacks as stubs in flex-callbacks-stubs.c. Hence, to use an enabled Connect feature, applications should replace the stub with their own implementation of the associated callback (typically in flex-callbacks.c). See UG235.04 for more info.

EmberStatus	emberPurgeIndirectMessages(void) Purge all indirect transmissions from the indirect message queue.
EmberStatus	emberSetIndirectQueueTimeout(uint32_t timeoutMs) Set indirect queue timeout value. The indirect queue timeout is set by default to EMBER_INDIRECT_TRANSMISSION_TIMEOUT_MS.

Functions

EmberStatus	emberFormNetwork(EmberNetworkParameters *parameters) Form a new network by becoming the coordinator. This API requires the parent-support library to be present.
EmberStatus	emberGetChildFlags(EmberMacAddress *address, EmberChildFlags *flags) Return an EmberChildFlags bitmask indicating the child flags of the child corresponding to the passed MAC address.
EmberStatus	emberGetChildInfo(EmberMacAddress *address, EmberMacAddress *addressResp, EmberChildFlags *flags) Return info on the child corresponding to the passed MAC address.
EmberStatus	emberRemoveChild(EmberMacAddress *address) Remove the node corresponding to the passed MAC address from the child table.

Handlers Documentation

emberChildJoinHandler

```
void emberChildJoinHandler (EmberNodeType nodeType, EmberNodeId nodeId)
```

Invoked at coordinator, range extender, or mac mode nodes when a new child has joined the device.

Parameters

[in]	nodeType	The role of the joining device (EMBER_STAR_RANGE_EXTENDER, EMBER_STAR_END_DEVICE, EMBER_STAR_SLEEPY_END_DEVICE, EMBER_MAC_MODE_DEVICE or EMBER_MAC_MODE_SLEEPY_DEVICE).
[in]	nodeId	The node ID of the joining device.

Definition at line 601 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/stack-info.h

Handlers Documentation

emberPurgeIndirectMessages

```
EmberStatus emberPurgeIndirectMessages (void)
```

Purge all indirect transmissions from the indirect message queue.

Parameters

N/A		

Returns

- an EmberStatus value of EMBER_SUCCESS if all indirect messages were purged, or another EmberStatus value indicating the reason of failure.

Definition at line 299 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/message.h

emberSetIndirectQueueTimeout

```
EmberStatus emberSetIndirectQueueTimeout (uint32_t timeoutMs)
```

Set indirect queue timeout value. The indirect queue timeout is set by default to EMBER_INDIRECT_TRANSMISSION_TIMEOUT_MS.

Parameters

N/A	timeoutMs	The timeout in milliseconds to be set.

Returns

- an EmberStatus value of EMBER_SUCCESS if the passed timeout was successfully set, or a value of EMBER_BAD_ARGUMENT if the passed value is invalid.

Definition at line 313 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/message.h

Function Documentation

emberFormNetwork

```
EmberStatus emberFormNetwork (EmberNetworkParameters *parameters)
```

Form a new network by becoming the coordinator. This API requires the parent-support library to be present.

Parameters

[in]	parameters	An EmberNetworkParameters value that specifies the network parameters of the network to be formed.

Returns

- An EmberStatus value that indicates either the successful formation of the new network or an EmberStatus value indicating the reason of failure.

Definition at line 263 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h

emberGetChildFlags

```
EmberStatus emberGetChildFlags (EmberMacAddress *address, EmberChildFlags *flags)
```

Return an EmberChildFlags bitmask indicating the child flags of the child corresponding to the passed MAC address.

Parameters

[in]	address	A pointer to an EmberMacAddress that specifies the MAC address of the child.
[out]	flags	A pointer to an EmberChildFlags containing the child flags of the child corresponding to the passed MAC address.

Note

- Deprecated, use emberGetChildInfo instead

Returns

- An EmberStatus value of EMBER_SUCCESS if the child was found in the child table, or another EmberStatus value indicating the reason of failure.

Definition at line `493` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h`

emberGetChildInfo

```
EmberStatus emberGetChildInfo (EmberMacAddress *address, EmberMacAddress *addressResp, EmberChildFlags *flags)
```

Return info on the child corresponding to the passed MAC address.

Parameters

[in]	address	A pointer to an EmberMacAddress that specifies the MAC address of the child (short or long).
[out]	addressResp	A pointer to an EmberMacAddress that returns the other address (respectively long or short).
[out]	flags	A pointer to an EmberChildFlags containing the child flags of the child corresponding to the passed MAC address.

Note

- For star coordinators, if the input address is short, the corresponding child will also be searched in the list of devices connected through range extender. Long address and additional flags for these devices are not available to the coordinator.

Note

- Both out parameters are optional. If set to NULL, the API will at least indicate if the child was found in the network.

Returns

- An EmberStatus value of EMBER_SUCCESS if the child was found in the child table, EMBER_CHILD_NOT_FOUND if it was not, or another EmberStatus value indicating the reason of failure.

Definition at line `521` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h`

emberRemoveChild

```
EmberStatus emberRemoveChild (EmberMacAddress *address)
```

Remove the node corresponding to the passed MAC address from the child table.

Parameters

[in]	address	A pointer to an EmberMacAddress that specifies the MAC address of the child to be removed.

Returns

- An EmberStatus value of EMBER_SUCCESS if the node was successfully removed from the child table, or another EmberStatus value indicating the reason of failure.

Definition at line `537` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/network-management.h`

Radio Stream

Radio Stream

Radio stream API.

License

Connect API managing radio stream for RF testing purpose

See radio-stream.h for source code.

Functions

EmberStatus	emberStartTxStream(EmberTxStreamParameters parameters, uint16_t channel) Start a continuous TX stream to test RF.
EmberStatus	emberStopTxStream(void) Stop an RF stream in progress.

Function Documentation

emberStartTxStream

```
EmberStatus emberStartTxStream (EmberTxStreamParameters parameters, uint16_t channel)
```

Start a continuous TX stream to test RF.

Parameters

[in]	parameters	Stream mode. See EmberTxStreamParameters.
[in]	channel	RF channel.

Returns

- EMBER_INVALID_CALL if the stack can not process the request EMBER_BAD_ARGUMENT if the parameters are wrong EMBER_SUCCESS if the stream can be started

Definition at line `50` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/radio-stream.h`

emberStopTxStream

```
EmberStatus emberStopTxStream (void)
```

Stop an RF stream in progress.

Parameters

N/A		

Returns

- EMBER_INVALID_CALL if no stream is in progress EMBER_SUCCESS otherwise

Definition at line `58` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/radio-stream.h`

Configuration

Configuration

User-configurable stack configuration macros and defaults.

Connect stack provides various interfaces to apply configuration:

- APIs to change configuration run-time
- Manufacturing tokens to change configuration during device flashing
- Macros to change configuration at compile-time

The compile time configuration macros and their default values are listed here.

The default values are always chosen to make it usable for most applications, but in some cases, you might need to change it. The recommended way is to either define the macro you need to change in the compilation command (e.g. gcc -D) or you by manipulating the configuration options in the configurator GUI.

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

Macros

#define EMBER_HEAP_SIZE 2000

The size in bytes of the Ember heap. See Memory Buffer for more details.

#define EMBER_CHILD_TABLE_SIZE 0

The maximum number of children supported by the device. Can be configured from 0 to 64. 11B of token space is allocated for each child in the table.

#define EMBER_CHILD_TABLE_TOKEN_SIZE EMBER_CHILD_TABLE_SIZE

#define EMBER_CHILD_TIMEOUT_SEC 3600

Every child should exchange regularly some sort of traffic with the parent. Eventually, if traffic is not exchanged for a prolonged period of time, the parent may remove the child from the child table. In particular the parent shall remove the oldest stale child whenever the child table is full and there is the need of making room for a new child. Range extenders periodically exchange network-level commands with the coordinator. End devices and sleepy end devices can use emberPollForData() as keep alive mechanism, or use the Poll Plugin plugin. The maximum allowed timeout value is EMBER_CHILD_TABLE_MAX_TIMEOUT_S. Setting the timeout value to EMBER_CHILD_TABLE_AGING_DISABLED disables aging of the child table.

#define EMBER_INDIRECT_QUEUE_SIZE 0

Indirect queue is used on a parent to store a message intended for a sleepy end device, this configures the size of that queue. Configure it to 0 if parent support plugin is not used.

#define EMBER_MAC_OUTGOING_QUEUE_SIZE 0

MAC Outgoing paclet queue is to store messages until the radio is available to send it (In most cases, the radio is unavailable because it's already transmitting). The configures the size of that queue.

#define EMBER_INDIRECT_TRANSMISSION_TIMEOUT_MS 8000

The maximum amount of time (in milliseconds) that the MAC will hold a message for indirect transmission to a child. The maximum value is 30 seconds (30000 milliseconds).

#define EMBER_NWK_RANGE_EXTENDER_UPDATE_PERIOD_SEC 60

The period in seconds a range extender sends an update command to the coordinator containing the list of its children.

#define	EMBER_MAC_ACK_TIMEOUT_MS 25 The ACK timeout in milliseconds. This parameter should be fine-tuned to reduce energy consumption for sleepy devices and depends on the data rate of the PHY configuration used. The maximum allowed value is 65.
#define	EMBER_RADIO_CCA_THRESHOLD -65 The CCA threshold used at the MAC layer for CSMA/CA, in dBm.
#define	EMBER_FREQUENCY_HOPPING_SEED 0 The frequency hopping channel sequence generation seed. Can be configured between 0 and 65535. See the Frequency Hopping chapter of UG235.03 for more details.
#define	EMBER_FREQUENCY_HOPPING_START_CHANNEL 0 The lowest channel on the frequency hopping list. See the Frequency Hopping chapter of UG235.03 for more details.
#define	EMBER_FREQUENCY_HOPPING_END_CHANNEL 24 The highest channel on the frequency hopping list. See the Frequency Hopping chapter of UG235.03 for more details.
#define	EMBER_FREQUENCY_HOPPING_CHANNEL_DURATION_MS 400 The time in milliseconds to stay on each channel for frequency hopping. See the Frequency Hopping chapter of UG235.03 for more details.
#define	EMBER_FREQUENCY_HOPPING_CHANNEL_GUARD_DURATION_MS 20 The time in milliseconds to guard each channel while frequency hopping. No MAC activity is allowed when entering or exiting the slot. See the Frequency Hopping chapter of UG235.03 for more details.
#define	EMBER_FREQUENCY_HOPPING_SERVER_FREQ_INFO_BROADCAST_PERIOD_S 15 The duration in seconds after which the server should broadcast its frequency hopping information to allow clients to realign. See the Frequency Hopping chapter of UG235.03 for more details.
#define	EMBER_FREQUENCY_HOPPING_SLEEPY_CLIENT_RESYNC_PERIOD_S 60 The duration in seconds after which a sleepy client should resync with the server if the last resync happened more than this duration ago. See the Frequency Hopping chapter of UG235.03 for more details.
#define	EMBER_FREQUENCY_HOPPING_ALWAYS_ON_CLIENT_SYNC_TIMEOUT_S (100) The maximum duration in seconds a non sleepy client would keep hopping without receiving frequency hopping information from the server, after which the synchronization with the server is deemed lost. A special value of EMBER_FREQUENCY_HOPPING_ALWAYS_ON_CLIENT_SYNC_DISABLE_TIMEOUT disables this timeout.
#define	EMBER_FREQUENCY_HOPPING_SERVER_ADVERTISING_ITERATION_COUNT (3) When a node is started a frequency hopping server, it will first advertise on all channels to resynchronize all existing clients in case the server was started as result of a reboot. This parameter defines the number of iterations over the entire hopping sequence.
#define	EMBER_COORDINATOR_FIRST_SHORT_ID_TO_BE_ASSIGNED 1 An EMBER_STAR_COORDINATOR assigns short IDs to other nodes in the star network sequentially starting from this short ID. This option provides a simple effective way to reserve an pool of short addresses for commissioning.
#define	EMBER_SECURITY_SHORT_TO_LONG_MAPPING_TABLE_SIZE 10 The size of the short-to-long address mapping table. See emberMacAddShortToLongAddressMapping for more details.
#define	EMBER_CSP_CALLBACK_MESSAGE_BUFFER_SIZE 127 The size of the the receiving buffer in CSP Callbacks.

Macro Definition Documentation

EMBER_HEAP_SIZE

```
#define EMBER_HEAP_SIZE
```

Value:

```
2000
```

The size in bytes of the Ember heap. See Memory Buffer for more details.

Warnings

- This should be configured from the Parent Support plugin options.

Definition at line `90` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/ember-configuration-defaults.h`

EMBER_CHILD_TABLE_SIZE

```
#define EMBER_CHILD_TABLE_SIZE
```

Value:

```
0
```

The maximum number of children supported by the device. Can be configured from 0 to 64. 11B of token space is allocated for each child in the table.

Note

- It's recommended to set it to 64 for EMBER_STAR_COORDINATOR, 32 for EMBER_STAR_RANGE_EXTENDER and 0 for anything else.

Warnings

- This should be configured from the Parent Support plugin options.

Definition at line `102` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/ember-configuration-defaults.h`

EMBER_CHILD_TABLE_TOKEN_SIZE

```
#define EMBER_CHILD_TABLE_TOKEN_SIZE
```

Value:

```
EMBER_CHILD_TABLE_SIZE
```

Definition at line `105` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/ember-configuration-defaults.h`

EMBER_CHILD_TIMEOUT_SEC

```
#define EMBER_CHILD_TIMEOUT_SEC
```

Value:

```
3600
```

Every child should exchange regularly some sort of traffic with the parent. Eventually, if traffic is not exchanged for a prolonged period of time, the parent may remove the child from the child table. In particular the parent shall remove the oldest stale child whenever the child table is full and there is the need of making room for a new child. Range extenders periodically exchange network-level commands with the coordinator. End devices and sleepy end devices can use emberPollForData() as keep alive mechanism, or use the Poll Plugin plugin. The maximum allowed timeout value is

EMBER_CHILD_TABLE_MAX_TIMEOUT_S. Setting the timeout value to EMBER_CHILD_TABLE_AGING_DISABLED disables aging of the child table.

Warnings

- This should be configured from the Parent Support plugin options.

Definition at line `123` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/ember-configuration-defaults.h`

EMBER_INDIRECT_QUEUE_SIZE

```
#define EMBER_INDIRECT_QUEUE_SIZE
```

Value:

```
0
```

Indirect queue is used on a parent to store a message intended for a sleepy end device, this configures the size of that queue. Configure it to 0 if parent support plugin is not used.

Warnings

- This should be configured from the Parent Support plugin options.

Definition at line `139` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/ember-configuration-defaults.h`

EMBER_MAC_OUTGOING_QUEUE_SIZE

```
#define EMBER_MAC_OUTGOING_QUEUE_SIZE
```

Value:

```
0
```

MAC Outgoing paclet queue is to store messages until the radio is available to send it (In most cases, the radio is unavailable because it's already transmitting). The configures the size of that queue.

Warnings

- This should be configured from the MAC Packet Queue plugin options.

Definition at line `149` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/ember-configuration-defaults.h`

EMBER_INDIRECT_TRANSMISSION_TIMEOUT_MS

```
#define EMBER_INDIRECT_TRANSMISSION_TIMEOUT_MS
```

Value:

```
8000
```

The maximum amount of time (in milliseconds) that the MAC will hold a message for indirect transmission to a child. The maximum value is 30 seconds (30000 milliseconds).

Warnings

- This should be configured from the Parent Support plugin options.

Definition at line 159 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/ember-configuration-defaults.h

EMBER_NWK_RANGE_EXTENDER_UPDATE_PERIOD_SEC

```
#define EMBER_NWK_RANGE_EXTENDER_UPDATE_PERIOD_SEC
```

Value:

```
60
```

The period in seconds a range extender sends an update command to the coordinator containing the list of its children.

- This option is only used on EMBER_STAR_RANGE_EXTENDER device.

Definition at line 168 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/ember-configuration-defaults.h

EMBER_MAC_ACK_TIMEOUT_MS

```
#define EMBER_MAC_ACK_TIMEOUT_MS
```

Value:

```
25
```

The ACK timeout in milliseconds. This parameter should be fine-tuned to reduce energy consumption for sleepy devices and depends on the data rate of the PHY configuration used. The maximum allowed value is 65.

Definition at line 177 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/ember-configuration-defaults.h

EMBER_RADIO_CCA_THRESHOLD

```
#define EMBER_RADIO_CCA_THRESHOLD
```

Value:

```
-65
```

The CCA threshold used at the MAC layer for CSMA/CA, in dBm.

Warnings

- This should be configured from the Connect Stack plugin options.

Definition at line 185 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/ember-configuration-defaults.h

EMBER_FREQUENCY_HOPPING_SEED

```
#define EMBER_FREQUENCY_HOPPING_SEED
```

Value:

```
0
```

The frequency hopping channel sequence generation seed. Can be configured between 0 and 65535. See the Frequency Hopping chapter of UG235.03 for more details.

Definition at line `194` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/ember-configuration-defaults.h`

EMBER_FREQUENCY_HOPPING_START_CHANNEL

```
#define EMBER_FREQUENCY_HOPPING_START_CHANNEL
```

Value:

```
0
```

The lowest channel on the frequency hopping list. See the Frequency Hopping chapter of UG235.03 for more details.

Definition at line `202` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/ember-configuration-defaults.h`

EMBER_FREQUENCY_HOPPING_END_CHANNEL

```
#define EMBER_FREQUENCY_HOPPING_END_CHANNEL
```

Value:

```
24
```

The highest channel on the frequency hopping list. See the Frequency Hopping chapter of UG235.03 for more details.

Definition at line `210` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/ember-configuration-defaults.h`

EMBER_FREQUENCY_HOPPING_CHANNEL_DURATION_MS

```
#define EMBER_FREQUENCY_HOPPING_CHANNEL_DURATION_MS
```

Value:

```
400
```

The time in milliseconds to stay on each channel for frequency hopping. See the Frequency Hopping chapter of UG235.03 for more details.

Definition at line `218` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/ember-configuration-defaults.h`

EMBER_FREQUENCY_HOPPING_CHANNEL_GUARD_DURATION_MS

```
#define EMBER_FREQUENCY_HOPPING_CHANNEL_GUARD_DURATION_MS
```

Value:

```
20
```

The time in milliseconds to guard each channel while frequency hopping. No MAC activity is allowed when entering or exiting the slot. See the Frequency Hopping chapter of UG235.03 for more details.

Definition at line 227 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/ember-configuration-defaults.h

EMBER_FREQUENCY_HOPPING_SERVER_FREQ_INFO_BROADCAST_PERIOD_S

```
#define EMBER_FREQUENCY_HOPPING_SERVER_FREQ_INFO_BROADCAST_PERIOD_S
```

Value:

```
15
```

The duration in seconds after which the server should broadcast its frequency hopping information to allow clients to realign. See the Frequency Hopping chapter of UG235.03 for more details.

Definition at line 236 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/ember-configuration-defaults.h

EMBER_FREQUENCY_HOPPING_SLEEPY_CLIENT_RESYNC_PERIOD_S

```
#define EMBER_FREQUENCY_HOPPING_SLEEPY_CLIENT_RESYNC_PERIOD_S
```

Value:

```
60
```

The duration in seconds after which a sleepy client should resync with the server if the last resync happened more than this duration ago. See the Frequency Hopping chapter of UG235.03 for more details.

Definition at line 245 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/ember-configuration-defaults.h

EMBER_FREQUENCY_HOPPING_ALWAYS_ON_CLIENT_SYNC_TIMEOUT_S

```
#define EMBER_FREQUENCY_HOPPING_ALWAYS_ON_CLIENT_SYNC_TIMEOUT_S
```

Value:

```
(100)
```

The maximum duration in seconds a non sleepy client would keep hopping without receiving frequency hopping information from the server, after which the synchronization with the server is deemed lost. A special value of EMBER_FREQUENCY_HOPPING_ALWAYS_ON_CLIENT_SYNC_DISABLE_TIMEOUT disables this timeout.

Definition at line 256 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/ember-configuration-defaults.h

EMBER_FREQUENCY_HOPPING_SERVER_ADVERTISING_ITERATION_COUNT

```
#define EMBER_FREQUENCY_HOPPING_SERVER_ADVERTISING_ITERATION_COUNT
```

Value:

```
(3)
```

When a node is started a frequency hopping server, it will first advertise on all channels to resynchronize all existing clients in case the server was started as result of a reboot. This parameter defines the number of iterations over the entire hopping sequence.

Definition at line `266` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/ember-configuration-defaults.h`

EMBER_COORDINATOR_FIRST_SHORT_ID_TO_BE_ASSIGNED

```
#define EMBER_COORDINATOR_FIRST_SHORT_ID_TO_BE_ASSIGNED
```

Value:

```
1
```

An EMBER_STAR_COORDINATOR assigns short IDs to other nodes in the star network sequentially starting from this short ID. This option provides a simple effective way to reserve an pool of short addresses for commissioning.

Definition at line `276` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/ember-configuration-defaults.h`

EMBER_SECURITY_SHORT_TO_LONG_MAPPING_TABLE_SIZE

```
#define EMBER_SECURITY_SHORT_TO_LONG_MAPPING_TABLE_SIZE
```

Value:

```
10
```

The size of the short-to-long address mapping table. See emberMacAddShortToLongAddressMapping for more details.

- This table is only used for EMBER_MAC_MODE_DEVICE and EMBER_MAC_MODE_SLEEPY_DEVICE, but the memory will be allocated on all device types.

Definition at line `287` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/ember-configuration-defaults.h`

EMBER_CSP_CALLBACK_MESSAGE_BUFFER_SIZE

```
#define EMBER_CSP_CALLBACK_MESSAGE_BUFFER_SIZE
```

Value:

```
127
```

The size of the the receiving buffer in CSP Callbacks.

- This must be set high enough depending on the PHY to prevent buffer overflow. In SUN PHYs (OFDM and FSK), the max message length is 2048. Considering the impact of this parameter on the stack size, this needs to be fine tuned according the need of the application

Definition at line `298` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/ember-configuration-defaults.h`

Status Codes

Status Codes

Return-code definitions for Connect stack API functions.

Many Connect 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.

Enumerations

```
enum    EmberStatus {
            EMBER_SUCCESS = 0×00
            EMBER_ERR_FATAL = 0×01
            EMBER_BAD_ARGUMENT = 0×02
            EMBER_EEPROM_MFG_STACK_VERSION_MISMATCH = 0×04
            EMBER_INVALID_CALL = 0×70
            EMBER_EEPROM_MFG_VERSION_MISMATCH = 0×06
            EMBER_EEPROM_STACK_VERSION_MISMATCH = 0×07
            EMBER_NO_BUFFERS = 0×18
            EMBER_SERIAL_INVALID_BAUD_RATE = 0×20
            EMBER_SERIAL_INVALID_PORT = 0×21
            EMBER_SERIAL_TX_OVERFLOW = 0×22
            EMBER_SERIAL_RX_OVERFLOW = 0×23
            EMBER_SERIAL_RX_FRAME_ERROR = 0×24
            EMBER_SERIAL_RX_PARITY_ERROR = 0×25
            EMBER_SERIAL_RX_EMPTY = 0×26
            EMBER_SERIAL_RX_OVERRUN_ERROR = 0×27
            EMBER_MAC_NO_DATA = 0×31
            EMBER_MAC_SYNC_TIMEOUT = 0×33
            EMBER_MAC_SYNC_WRONG_SEED = 0×34
            EMBER_MAC_SECURITY_FAILED = 0×35
            EMBER_MAC_UNKNOWN_DESTINATION = 0×37
            EMBER_MAC_SECURITY_NOT_SUPPORTED = 0×38
            EMBER_MAC_TRANSMIT_QUEUE_FULL = 0×39
            EMBER_MAC_ACK_HEADER_TYPE = 0×3B
            EMBER_MAC_SCANNING = 0×3D
            EMBER_MAC_BUSY = 0×3E
            EMBER_MAC_NO_ACK_RECEIVED = 0×40
            EMBER_MAC_INDIRECT_TIMEOUT = 0×41
            EMBER_MAC_INDIRECT_MESSAGE_PURGED = 0×42
            EMBER_SIM_EEPROM_ERASE_PAGE_GREEN = 0×43
            EMBER_SIM_EEPROM_ERASE_PAGE_RED = 0×44
            EMBER_SIM_EEPROM_FULL = 0×45
            EMBER_SIM_EEPROM_INIT_2_FAILED = 0×49
            EMBER_SIM_EEPROM_INIT_3_FAILED = 0×4A
            EMBER_SIM_EEPROM_REPAIRING = 0×4D
            EMBER_ERR_FLASH_WRITE_INHIBITED = 0×46
            EMBER_ERR_FLASH_VERIFY_FAILED = 0×47
            EMBER_ERR_FLASH_PROG_FAIL = 0×4B
            EMBER_MESSAGE_TOO_LONG = 0×74
            EMBER_ADC_CONVERSION_DONE = 0×80
            EMBER_ADC_CONVERSION_BUSY = 0×81
            EMBER_ADC_CONVERSION_DEFERRED = 0×82
            EMBER_ADC_NO_CONVERSION_PENDING = 0×84
            EMBER_SLEEP_INTERRUPTED = 0×85
            EMBER_PHY_TX_INCOMPLETE = 0×89
            EMBER_PHY_INVALID_CHANNEL = 0×8A
            EMBER_PHY_INVALID_POWER = 0×8B
            EMBER_PHY_TX_BUSY = 0×8C
            EMBER_PHY_TX_CCA_FAIL = 0×8D
            EMBER_PHY_CALIBRATING = 0×8E
            EMBER_PHY_ACK_RECEIVED = 0×8F
            EMBER_NETWORK_UP = 0×90
            EMBER_NETWORK_DOWN = 0×91
            EMBER_JOIN_SCAN_FAILED = 0×92
            EMBER_JOIN_FAILED = 0×94
            EMBER_JOIN_DENIED = 0×95
            EMBER_JOIN_TIMEOUT = 0×96
            EMBER_NO_VALID_BEACONS = 0xAB
            EMBER_SECURITY_DATA_INVALID = 0xBD
            EMBER_NOT_JOINED = 0×93
            EMBER_TABLE_FULL = 0xB4
            EMBER_LIBRARY_NOT_PRESENT = 0xB5
            EMBER_CHILD_NOT_FOUND = 0xB6
            EMBER_NVM3_ERR_OPENED_WITH_OTHER_PARAMETERS = 0xC1
            EMBER_NVM3_ERR_ALIGNMENT_INVALID = 0xC2
            EMBER_NVM3_ERR_SIZE_TOO_SMALL = 0xC3
            EMBER_NVM3_ERR_PAGE_SIZE_NOT_SUPPORTED = 0xC4
            EMBER_NVM3_ERR_TOKEN_INIT = 0xC5
            EMBER_NVM3_ERR_UPGRADE = 0xC6
```

EMBER_NVM3_ERR_UNKNOWN = 0xC7
EMBER_NCP_UNKNOWN_COMMAND_ID
= 0xD0
EMBER_APPLICATION_ERROR_0 = 0xF0
EMBER_APPLICATION_ERROR_1 = 0xF1
EMBER_APPLICATION_ERROR_2 = 0xF2
EMBER_APPLICATION_ERROR_3 = 0xF3
EMBER_APPLICATION_ERROR_4 = 0xF4
EMBER_APPLICATION_ERROR_5 = 0xF5
EMBER_APPLICATION_ERROR_6 = 0xF6
EMBER_APPLICATION_ERROR_7 = 0xF7
EMBER_APPLICATION_ERROR_8 = 0xF8
EMBER_APPLICATION_ERROR_9 = 0xF9
EMBER_APPLICATION_ERROR_10 =
0xFA
EMBER_APPLICATION_ERROR_11 =
0xFB
EMBER_APPLICATION_ERROR_12 =
0xFC
EMBER_APPLICATION_ERROR_13 =
0xFD
EMBER_APPLICATION_ERROR_14 =
0xFE
EMBER_APPLICATION_ERROR_15 =
0xFF

}

Enumeration Documentation

EmberStatus

```
EmberStatus
```

Enumerator	
EMBER_SUCCESS	The generic "no error" message.
EMBER_ERR_FATAL	The generic "fatal error" message.
EMBER_BAD_ARGUMENT	An invalid value was passed as an argument to a function.
EMBER_EEPROM_MFG_STACK_VERSION_MISMATCH	The manufacturing and stack token format in non-volatile memory is different than what the stack expects (returned at initialization).
EMBER_INVALID_CALL	The API call is not allowed given the current state of the stack.
EMBER_EEPROM_MFG_VERSION_MISMATCH	The manufacturing token format in non-volatile memory is different than what the stack expects (returned at initialization).
EMBER_EEPROM_STACK_VERSION_MISMATCH	The stack token format in non-volatile memory is different than what the stack expects (returned at initialization).
EMBER_NO_BUFFERS	There are no more buffers (either in the stack heap or the queue used by the associated module, such as indirect queue).
EMBER_SERIAL_INVALID_BAUD_RATE	Specified an invalid baud rate.
EMBER_SERIAL_INVALID_PORT	Specified an invalid serial port.
EMBER_SERIAL_TX_OVERFLOW	Tried to send too much data.
EMBER_SERIAL_RX_OVERFLOW	There was not enough space to store a received character and some characters were dropped.
EMBER_SERIAL_RX_FRAME_ERROR	Detected a UART framing error.
EMBER_SERIAL_RX_PARITY_ERROR	Detected a UART parity error.
EMBER_SERIAL_RX_EMPTY	There is no received data to process.

EMBER_SERIAL_RX_OVERRUN_ERROR	The receive interrupt was not handled in time and some characters were dropped.
EMBER_MAC_NO_DATA	No pending data exists for device doing a data poll.
EMBER_MAC_SYNC_TIMEOUT	The frequency hopping client failed the frequency hopping synchronization procedure. It timed out trying to reach the frequency hopping server.
EMBER_MAC_SYNC_WRONG_SEED	The frequency hopping client failed the frequency hopping synchronization procedure. The server is currently using a different seed.
EMBER_MAC_SECURITY_FAILED	MAC security operation failed.
EMBER_MAC_UNKNOWN_DESTINATION	Transmission failed: the destination node does not appear in the neighbor or child tables.
EMBER_MAC_SECURITY_NOT_SUPPORTED	Transmission failed: the local node does not support security or a secured transmission has been requested to a child that does not support security.
EMBER_MAC_TRANSMIT_QUEUE_FULL	The MAC transmit queue is full.
EMBER_MAC_ACK_HEADER_TYPE	MAC ACK header received.
EMBER_MAC_SCANNING	The MAC can't complete this task because it is scanning.
EMBER_MAC_BUSY	The requested operation cannot be completed because MAC is currently busy performing a high-priority task.
EMBER_MAC_NO_ACK_RECEIVED	Expected to receive an ACK following the transmission, but the MAC level ACK was never received.
EMBER_MAC_INDIRECT_TIMEOUT	Indirect data message timed out before polled.
EMBER_MAC_INDIRECT_MESSAGE_PURGED	Transmission failed: the indirect message was purged because the destination child has been removed or updated, or because emberPurgeIndirectMessages() was called.
EMBER_SIM_EEPROM_ERASE_PAGE_GREEN	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.
EMBER_SIM_EEPROM_ERASE_PAGE_RED	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.
EMBER_SIM_EEPROM_FULL	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 EMBER_SIM_EEPROM_ERASE_PAGE_RED error code.
EMBER_SIM_EEPROM_INIT_2_FAILED	Attempt 2 to initialize the Simulated EEPROM has failed.
EMBER_SIM_EEPROM_INIT_3_FAILED	Attempt 3 to initialize the Simulated EEPROM has failed.
EMBER_SIM_EEPROM_REPAIRING	The Simulated EEPROM is repairing itself.
EMBER_ERR_FLASH_WRITE_INHIBITED	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.
EMBER_ERR_FLASH_VERIFY_FAILED	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.
EMBER_ERR_FLASH_PROG_FAIL	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.
EMBER_MESSAGE_TOO_LONG	The message to be transmitted is too big to fit into a single over-the-air packet.

EMBER_ADC_CONVERSION_DONE	Conversion is complete.
EMBER_ADC_CONVERSION_BUSY	Conversion cannot be done because a request is being processed.
EMBER_ADC_CONVERSION_DEFERRED	Conversion is deferred until the current request has been processed.
EMBER_ADC_NO_CONVERSION_PENDING	No results are pending.
EMBER_SLEEP_INTERRUPTED	Sleeping (for a duration) has been abnormally interrupted and exited prematurely.
EMBER_PHY_TX_INCOMPLETE	The transmit hardware did not finish transmitting a packet.
EMBER_PHY_INVALID_CHANNEL	An unsupported channel setting was specified.
EMBER_PHY_INVALID_POWER	An unsupported power setting was specified.
EMBER_PHY_TX_BUSY	The requested operation cannot be completed because the radio is currently busy, either transmitting a packet or performing calibration.
EMBER_PHY_TX_CCA_FAIL	The transmit attempt failed because all CCA attempts indicated that the channel was busy.
EMBER_PHY_CALIBRATING	The requested operation cannot be completed because the stack has triggered a radio calibration. Wait for about one second for it to complete.
EMBER_PHY_ACK_RECEIVED	The expected ACK was received after the last transmission.
EMBER_NETWORK_UP	The stack software has completed initialization required to be in a network and is ready to send and receive packets over the air.
EMBER_NETWORK_DOWN	The network is not operating.
EMBER_JOIN_SCAN_FAILED	The node failed to initiate the scanning process during the joining process.
EMBER_JOIN_FAILED	An attempt to join a network failed.
EMBER_JOIN_DENIED	An attempt to join a network was rejected.
EMBER_JOIN_TIMEOUT	The node timed out waiting for a response during the joining process.
EMBER_NO_VALID_BEACONS	An attempt to join or rejoin the network failed because no valid beacons was received by the joining node.
EMBER_SECURITY_DATA_INVALID	The security data provided was not valid, or an integrity check failed.
EMBER_NOT_JOINED	The node has not joined a network. Returned by emberNetworkInit() if there was no connection data saved in tokens.
EMBER_TABLE_FULL	There are no empty entries left in the table.
EMBER_LIBRARY_NOT_PRESENT	The requested function cannot be executed because the library (plugin) that contains the necessary functionality is not present.
EMBER_CHILD_NOT_FOUND	The requested NodeId has not been found in the child list or grandchildren list (start network only).
EMBER_NVM3_ERR_OPENED_WITH_OTHER_PARAMETERS	NVM3 is telling the application that the initialization was aborted as the NVM3 instance was already opened with other parameters.
EMBER_NVM3_ERR_ALIGNMENT_INVALID	NVM3 is telling the application that the initialization was aborted as the NVM3 instance is not aligned properly in memory.

EMBER_NVM3_ERR_SIZE_TOO_SMALL	NVM3 is telling the application that the initialization was aborted as the size of the NVM3 instance is too small.
EMBER_NVM3_ERR_PAGE_SIZE_NOT_SUPPORTED	NVM3 is telling the application that the initialization was aborted as the NVM3 page size is not supported.
EMBER_NVM3_ERR_TOKEN_INIT	NVM3 is telling the application that there was an error initializing some of the tokens.
EMBER_NVM3_ERR_UPGRADE	NVM3 is telling the application there has been an error when attempting to upgrade SimEE tokens.
EMBER_NVM3_ERR_UNKNOWN	NVM3 is telling the application that there has been an unknown error.
EMBER_NCP_UNKNOWN_COMMAND_ID	The NCP does not know the command ID that the host sent. This can correspond to a version mismatch, where an API is available on the host but not on the NCP.
EMBER_APPLICATION_ERROR_0	This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.
EMBER_APPLICATION_ERROR_1	This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.
EMBER_APPLICATION_ERROR_2	This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.
EMBER_APPLICATION_ERROR_3	This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.
EMBER_APPLICATION_ERROR_4	This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.
EMBER_APPLICATION_ERROR_5	This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.
EMBER_APPLICATION_ERROR_6	This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.
EMBER_APPLICATION_ERROR_7	This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.
EMBER_APPLICATION_ERROR_8	This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.
EMBER_APPLICATION_ERROR_9	This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.
EMBER_APPLICATION_ERROR_10	This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.
EMBER_APPLICATION_ERROR_11	This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.
EMBER_APPLICATION_ERROR_12	This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.
EMBER_APPLICATION_ERROR_13	This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.
EMBER_APPLICATION_ERROR_14	This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.
EMBER_APPLICATION_ERROR_15	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 `58` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/error-def.h`

Stack Tokens

Stack Tokens

Definitions for stack tokens.

Stack tokens are used by the stack to store information in non-volatile memory. A typical use case is to store the network information, so after an accidental reset it can be part of the network without going through the association process again.

Note

- For the application tokens, refer to Token Access and AN1154.

Warnings

- While stack tokens can be accessed through the Token Access API, they **must not be written directly**. Most stack tokens have APIs to read/write them. This documentation is intended for those who need more information on the internal details of the connect stack.

See token-stack.h for source code.

Modules

tokTypeStackKey

tokTypeStackNodeData

tokTypeStackChildTableEntry

Token types

The types used for each stack token.

typedef uint16_t	tokTypeStackNvdataVersion Type for TOKEN_STACK_NVDATA_VERSION. Keeps the version number of stack tokens.
typedef uint32_t	tokTypeStackNonceCounter Type for TOKEN_STACK_NONCE_COUNTER. Used to make sure that Nonce used for security is not repeated even after unexpected reboot.
typedef uint32_t	tokTypeStackKeyID Type for TOKEN_STACK_SECURITY_KEY_ID. Used to make sure that Nonce used for security is not repeated even after unexpected reboot.
typedef uint16_t	tokTypeStackLastAllocatedId Type for TOKEN_STACK_LAST_ASSIGNED_ID. Stores the last assigned NodeId if the device is EMBER_STAR_COORDINATOR.
typedef uint32_t	tokTypeStackBootCounter Type for TOKEN_STACK_BOOT_COUNTER. Increments at boot (during emberInit()).
typedef EmberEUI64	tokTypeParentLongId Type for TOKEN_STACK_PARENT_LONG_ID. Stores the Long Id of the parent of this device. Only used for EMBER_MAC_MODE_DEVICE and EMBER_MAC_MODE_SLEEPY_DEVICE device types.

Macros

#define CURRENT_STACK_TOKEN_VERSION 0x03FC
The current version number of the stack tokens. MSB is the version. LSB is a complement.

Token types Documentation

tokTypeStackNvdataVersion

```
typedef uint16_t tokTypeStackNvdataVersion
```

Type for TOKEN_STACK_NVDATA_VERSION. Keeps the version number of stack tokens.

Definition at line `220` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/token-stack.h`

tokTypeStackNonceCounter

```
typedef uint32_t tokTypeStackNonceCounter
```

Type for TOKEN_STACK_NONCE_COUNTER. Used to make sure that Nonce used for security is not repeated even after unexpected reboot.

Definition at line `226` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/token-stack.h`

tokTypeStackKeyID

```
typedef uint32_t tokTypeStackKeyID
```

Type for TOKEN_STACK_SECURITY_KEY_ID. Used to make sure that Nonce used for security is not repeated even after unexpected reboot.

Definition at line `241` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/token-stack.h`

tokTypeStackLastAllocatedId

```
typedef uint16_t tokTypeStackLastAllocatedId
```

Type for TOKEN_STACK_LAST_ASSIGNED_ID. Stores the last assigned NodeId if the device is EMBER_STAR_COORDINATOR.

Definition at line `272` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/token-stack.h`

tokTypeStackBootCounter

```
typedef uint32_t tokTypeStackBootCounter
```

Type for TOKEN_STACK_BOOT_COUNTER. Increments at boot (during emberInit()).

Definition at line `278` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/token-stack.h`

tokTypeParentLongId

```
typedef EmberEUI64 tokTypeParentLongId
```

Type for TOKEN_STACK_PARENT_LONG_ID. Stores the Long Id of the parent of this device. Only used for EMBER_MAC_MODE_DEVICE and EMBER_MAC_MODE_SLEEPY_DEVICE device types.

Definition at line `285` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/token-stack.h`

Macro Definition Documentation

CURRENT_STACK_TOKEN_VERSION

```
#define CURRENT_STACK_TOKEN_VERSION
```

Value:

```
0x03FC
```

The current version number of the stack tokens. MSB is the version. LSB is a complement.

See hal/micro/token.h for a more complete explanation.

Definition at line `206` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/token-stack.h`

tokTypeStackKey

Type for TOKEN_STACK_SECURITY_KEY. Keeps the security key for MAC layer security.

Public Attributes

uint8_t	networkKey

Public Attribute Documentation

networkKey

```
uint8_t tokTypeStackKey::networkKey[16]
```

The key itself

Definition at line `234` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/token-stack.h`

tokTypeStackNodeData

Type for TOKEN_STACK_NODE_DATA. Generic information of the node is stored in this token.

Public Attributes

uint16_t	panId
int16_t	radioTxPower
uint16_t	radioFreqChannel
uint8_t	nodeType
uint16_t	nodeId
uint16_t	parentId

Public Attribute Documentation

panId

```
uint16_t tokTypeStackNodeData::panId
```

The PanId of the device

Definition at line `249` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/token-stack.h`

radioTxPower

```
int16_t tokTypeStackNodeData::radioTxPower
```

The TX power configured for the device in deci-dBm

Definition at line `250` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/token-stack.h`

radioFreqChannel

```
uint16_t tokTypeStackNodeData::radioFreqChannel
```

The radio channel configured for the device

Definition at line `251` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/token-stack.h`

nodeType

```
uint8_t tokTypeStackNodeData::nodeType
```

The EmberNodeType configured for the device

Definition at line `252` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/token-stack.h`

nodeId

```
uint16_t tokTypeStackNodeData::nodeId
```

The NodeId (short address) of the device

Definition at line `253` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/token-stack.h`

parentId

```
uint16_t tokTypeStackNodeData::parentId
```

The NodeId of the device's parent, if any

Definition at line `254` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/token-stack.h`

tokTypeStackChildTableEntry

Type of an element of TOKEN_STACK_CHILD_TABLE (indexed token). Keeps children information of a device, which has parent support enabled.

Public Attributes

EmberEUI64	longId
EmberNodeId	shortId
uint8_t	flags

Public Attribute Documentation

longId

```
EmberEUI64 tokTypeStackChildTableEntry::longId
```

The Long Id of the child

Definition at line `263` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/token-stack.h`

shortId

```
EmberNodeId tokTypeStackChildTableEntry::shortId
```

The NodeId of the child

Definition at line `264` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/token-stack.h`

flags

```
uint8_t tokTypeStackChildTableEntry::flags
```

Flags for the child required by the stack

Definition at line `265` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/config/token-stack.h`

Event Scheduling

Event Scheduling

Scheduling events for future execution.

See Event Scheduling for documentation. These macros implement an event abstraction that allows the application to schedule code to run after a specified time interval. An event consists of a procedure to be called at some point in the future and a control object that determines which procedure should be called. Events are also useful when an ISR needs to initiate an action that should run outside the 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 via some kind of **emberEventControlSetDelayXX()** call or by deactivating via a call to emberEventControlSetInactive().

When 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. A tick equals 1 ms on every platform supported by Connect. Note, however, that the accuracy of the base tick depends on the timer source, which by default is the LF RC oscillator on EFR32 platforms.

Furthermore, 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. These APIs are therefore doesn't actually mean a quarter of second or a minute on platforms supported by Connect.

However, in the future, Connect support might become available on platforms where one tick is not exactly 1 ms. For example, on the EM357 SoC, 1 s is 1024 ticks, so each tick is 1000 / 1024 = ~0.98 milliseconds. If you need platform independent accurate delays, use the macros MILLISECOND_TICKS_PER_SECOND and MILLISECOND_TICKS_PER_MINUTE. For example, calling emberEventControlSetDelayMS(someEvent, 3 * MILLISECOND_TICKS_PER_MINUTE) will delay for 3 minutes on any platform.

The following are 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 an action has to occur 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 (this is usually generated)
EmberEventData events[] =
 {
   { &delayEvent,    delayEventHandler },
   { &signalEvent,   signalEentHandler },
   { &periodicEvent, periodicEventHandler },
   { NULL, NULL }                            // terminator
 };

void main(void)
{
  // Cause the periodic event to occur once a second.
  emberEventControlSetDelayQS(periodicEvent, 4);

  while (true) {
    emberRunEvents(events);
  }
}
```

Time Manipulation Macros

void	sli_event_control_set_active(EmberEventControl *event) Set EmberEventControl to run at the next available opportunity.
void	emEventControlSetDelayMS(EmberEventControl *event, uint32_t delay) Set EmberEventControl to run some milliseconds in the future.
uint32_t	emEventControlGetRemainingMS(EmberEventControl *event) Check when the event is scheduled to run.

void	emberRunEvents(EmberEventData *events)	Start an event handler if anything is scheduled when this function is called.
void	emberRunTask(EmberTaskId taskid)	Start an event handler if there is anything scheduled at the moment this function is called.
uint32_t	emberMsToNextEvent(EmberEventData *events, uint32_t maxMs)	Check when the next event is scheduled to run.
uint32_t	emberMsToNextEventExtended(EmberEventData *events, uint32_t maxMs, uint8_t *returnIndex)	Check when the next event is scheduled to run.
uint32_t	emberMsToNextStackEvent(void)	Check when the next stack event is scheduled to run.
EmberTaskId	emberTaskInit(EmberEventData *events)	Initialize a task for managing events and processor idling state.
bool	emberMarkTaskIdle(EmberTaskId taskid)	Try to idle the CPU, unless any events in any tasks are pending.
void	emTaskEnableIdling(bool allow)	Enable or disable idling.
void	emMarkTaskActive(EmberTaskId taskid)	Calling it indicates that a task has something to do, so it should prevent the CPU from idling until emberMarkTaskIdle is next called on this task.
#define	elapsedTimeInt8u (oldTime, newTime)	Returns the elapsed time between two 8 bit values. Result may not be valid if the time samples differ by more than 127.
#define	elapsedTimeInt16u (oldTime, newTime)	Returns the elapsed time between two 16 bit values. Result may not be valid if the time samples differ by more than 32767.
#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.
#define	MAX_INT8U_VALUE (0xFF)	Returns true if t1 is greater than t2. Can only account for 1 wrap around of the variable before it is wrong.
#define	HALF_MAX_INT8U_VALUE (0×80)	Returns the elapsed time between two 8 bit values. Result may not be valid if the time samples differ by more than 127.
#define	timeGTorEqualInt8u (t1, t2)	Returns the elapsed time between two 8 bit values. Result may not be valid if the time samples differ by more than 127.
#define	MAX_INT16U_VALUE (0xFFFF)	Returns true if t1 is greater than t2. Can only account for 1 wrap around of the variable before it is wrong.
#define	HALF_MAX_INT16U_VALUE (0×8000)	Returns the elapsed time between two 8 bit values. Result may not be valid if the time samples differ by more than 127.
#define	timeGTorEqualInt16u (t1, t2)	Returns the elapsed time between two 8 bit values. Result may not be valid if the time samples differ by more than 127.
#define	MAX_INT32U_VALUE (0xFFFFFFFFUL)	Returns true if t1 is greater than t2. Can only account for 1 wrap around of the variable before it is wrong.
#define	HALF_MAX_INT32U_VALUE (0×80000000UL)	Returns the elapsed time between two 8 bit values. Result may not be valid if the time samples differ by more than 127.

#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.

#define MILLISECOND_TICKS_PER_SECOND 1000UL
Returns the elapsed time between two 8 bit values. Result may not be valid if the time samples differ by more than 127.

#define MILLISECOND_TICKS_PER_DECISECOND (MILLISECOND_TICKS_PER_SECOND / 10)
Returns the elapsed time between two 8 bit values. Result may not be valid if the time samples differ by more than 127.

#define MILLISECOND_TICKS_PER_QUARTERSECOND (MILLISECOND_TICKS_PER_SECOND >> 2)
Returns the elapsed time between two 8 bit values. Result may not be valid if the time samples differ by more than 127.

#define MILLISECOND_TICKS_PER_MINUTE (60UL * MILLISECOND_TICKS_PER_SECOND)
Returns the elapsed time between two 8 bit values. Result may not be valid if the time samples differ by more than 127.

#define MILLISECOND_TICKS_PER_HOUR (60UL * MILLISECOND_TICKS_PER_MINUTE)
Returns the elapsed time between two 8 bit values. Result may not be valid if the time samples differ by more than 127.

#define MILLISECOND_TICKS_PER_DAY (24UL * MILLISECOND_TICKS_PER_HOUR)
Returns the elapsed time between two 8 bit values. Result may not be valid if the time samples differ by more than 127.

#define EMBER_TASK_COUNT (3)
The number of event tasks that can be used to schedule and run events. Connect stack requires one, while another is used for Application Framework events.

#define emberEventControlSetInactive (control)
Set EmberEventControl as inactive (no pending event).

#define emberEventControlGetActive (control)
Check whether EmberEventControl is currently active. An event is considered active if it is set to run some time in the future (activated by emberEventControlSetActive(), emberEventControlSetDelayMS() or any other emberEventControlSetDelay* functions)

#define emberEventControlSetActive (control)
Set EmberEventControl to run at the next available opportunity.

#define EMBER_MAX_EVENT_CONTROL_DELAY_MS (HALF_MAX_INT32U_VALUE - 1)
The maximum delay that may be passed to emberEventControlSetDelayMS().

#define emberEventControlSetDelayMS (control, delay)
Set EmberEventControl to run some milliseconds in the future.

#define EMBER_MAX_EVENT_CONTROL_DELAY_QS (EMBER_MAX_EVENT_CONTROL_DELAY_MS >> 8)
The maximum delay that may be passed to emberEventControlSetDelayQS().

#define emberEventControlSetDelayQS (control, delay)
Set EmberEventControl to run some quarter seconds in the future.

#define EMBER_MAX_EVENT_CONTROL_DELAY_MINUTES (EMBER_MAX_EVENT_CONTROL_DELAY_MS >> 16)
The maximum delay that may be passed to emberEventControlSetDelayMinutes().

#define emberEventControlSetDelayMinutes (control, delay)
Set EmberEventControl to run some minutes in the future.

#define emberEventControlGetRemainingMS (control)
Check when the event is scheduled to run.

#define emberTaskEnableIdling (allow)
Enable or disable idling.

#define emberMarkTaskActive (taskid)
Calling it indicates that a task has something to do, so it should prevent the CPU from idling until emberMarkTaskIdle is next called on this task.

Macros

#define	__EVENT_H__ undefined

Time Manipulation Macros Documentation

sli_event_control_set_active

```
void sli_event_control_set_active (EmberEventControl *event)
```

Set EmberEventControl to run at the next available opportunity.

Parameters

[in]	event	Pointer to the control of the event to set active

Warnings

- Applications should use emberEventControlSetActive() instead.

Definition at line `299` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h`

emEventControlSetDelayMS

```
void emEventControlSetDelayMS (EmberEventControl *event, uint32_t delay)
```

Set EmberEventControl to run some milliseconds in the future.

Parameters

[in]	event	Pointer to the control of the event to run.
[in]	delay	The delay in milliseconds. Must be less than EMBER_MAX_EVENT_CONTROL_DELAY_MS

Warnings

- Applications should use emberEventControlSetDelayMS() instead.

Definition at line `325` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h`

emEventControlGetRemainingMS

```
uint32_t emEventControlGetRemainingMS (EmberEventControl *event)
```

Check when the event is scheduled to run.

Parameters

[in]	event	Pointer to the control of the event in question.

Returns

- Return the amount of milliseconds remaining before the event is scheduled to run. If the event is inactive, MAX_INT32U_VALUE is returned.

Warnings

- Applications should use emberEventControlGetRemainingMS() instead.

Definition at line `378` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h`

emberRunEvents

```
void emberRunEvents (EmberEventData *events)
```

Start an event handler if anything is scheduled when this function is called.

Parameters

[in]	events	Pointer to the array of events.

An application typically creates an array of events along with their handlers. This function should be called in the main loop to run those events. **Warnings**

- This is normally handled by emberRunTask() in the main plugin.

Definition at line `392` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h`

emberRunTask

```
void emberRunTask (EmberTaskId taskid)
```

Start an event handler if there is anything scheduled at the moment this function is called.

Parameters

N/A	taskid	

If an application has initialized a task via emberTaskInit(), to run the events associated with that task, it should call emberRunTask() instead of emberRunEvents().

Warnings

- This is normally handled by the main plugin.

Definition at line `405` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h`

emberMsToNextEvent

```
uint32_t emberMsToNextEvent (EmberEventData *events, uint32_t maxMs)
```

Check when the next event is scheduled to run.

Parameters

[in]	events	An array of events to check.
[in]	maxMs	If no event is scheduled before maxMs, maxMs will be returned

Returns

- 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, to guarantee the accuracy of this API, it must be called with interrupts disabled.

See Also

- emberMsToNextEventExtended()

Definition at line `420` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h`

emberMsToNextEventExtended

```
uint32_t emberMsToNextEventExtended (EmberEventData *events, uint32_t maxMs, uint8_t *returnIndex)
```

Check when the next event is scheduled to run.

Parameters

[in]	events	An array of events to check.
[in]	maxMs	If no event is scheduled before maxMs, maxMs will be returned
[out]	returnIndex	If not NULL pointer was passed, the index of the next event will be returned here, or 0xFF if no event is scheduled before maxMs.

Returns

- 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, to guarantee the accuracy of this API, it must be called with interrupts disabled.

See Also

- emberMsToNextEvent()

Definition at line `438` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h`

emberMsToNextStackEvent

```
uint32_t emberMsToNextStackEvent (void)
```

Check when the next stack event is scheduled to run.

Parameters

N/A		

Returns

- Returns the number of milliseconds before the next stack event is scheduled to run.

Definition at line `446` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h`

emberTaskInit

```
EmberTaskId emberTaskInit (EmberEventData *events)
```

Initialize a task for managing events and processor idling state.

Parameters

[in]	events	Pointer to the array of events to manage

Returns

- Returns the EmberTaskId which represents the newly created task.

Note

- After the task is created emberRunTask() should be called periodically.

Definition at line 456 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h

emberMarkTaskIdle

```
bool emberMarkTaskIdle (EmberTaskId taskid)
```

Try to idle the CPU, unless any events in any tasks are pending.

Parameters

[in]	taskid	the task which should handle the idling.

Returns

- Returns **true** if the processor was idled **false** if idling wasn't permitted because a task has something to do.

Note

- This API should always be called with interrupts disabled. It will forcibly re-enable interrupts before returning.

Definition at line 468 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h

emTaskEnableIdling

```
void emTaskEnableIdling (bool allow)
```

Enable or disable idling.

Parameters

[in]	allow	Setting it to **true** will enable, while setting it to **false** will disable idling.

Warnings

- Applications should use emberTaskEnableIdling() instead.

Definition at line 484 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h

emMarkTaskActive

```
void emMarkTaskActive (EmberTaskId taskid)
```

Calling it indicates that a task has something to do, so it should prevent the CPU from idling until emberMarkTaskIdle is next called on this task.

Parameters

[in]	taskid	The task to mark active.

Warnings

- Applications should use emberMarkTaskActive() instead.

Definition at line 499 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h

elapsedTimeInt8u

```
#define elapsedTimeInt8u
```

Value:

```
(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 190 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h

elapsedTimeInt16u

```
#define elapsedTimeInt16u
```

Value:

```
(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 197 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h

elapsedTimeInt32u

```
#define elapsedTimeInt32u
```

Value:

```
(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 204 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h

MAX_INT8U_VALUE

```
#define MAX_INT8U_VALUE
```

Value:

```
(0xFF)
```

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 211 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h

HALF_MAX_INT8U_VALUE

```
#define HALF_MAX_INT8U_VALUE
```

Value:

```
(0x80)
```

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 212 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h

timeGTorEqualInt8u

```
#define timeGTorEqualInt8u
```

Value:

```
(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 213 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h

MAX_INT16U_VALUE

```
#define MAX_INT16U_VALUE
```

Value:

```
(0xFFFF)
```

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 220 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h

HALF_MAX_INT16U_VALUE

```
#define HALF_MAX_INT16U_VALUE
```

Value:

```
(0x8000)
```

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 221 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h

timeGTorEqualInt16u

```
#define timeGTorEqualInt16u
```

Value:

```
(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 `222` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h`

MAX_INT32U_VALUE

```
#define MAX_INT32U_VALUE
```

Value:

```
(0xFFFFFFFFUL)
```

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 `229` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h`

HALF_MAX_INT32U_VALUE

```
#define HALF_MAX_INT32U_VALUE
```

Value:

```
(0×80000000UL)
```

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 `230` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h`

timeGTorEqualInt32u

```
#define timeGTorEqualInt32u
```

Value:

```
(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 `231` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h`

MILLISECOND_TICKS_PER_SECOND

```
#define MILLISECOND_TICKS_PER_SECOND
```

Value:

```
1000UL
```

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 234 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h

MILLISECOND_TICKS_PER_DECISECOND

```
#define MILLISECOND_TICKS_PER_DECISECOND
```

Value:

```
(MILLISECOND_TICKS_PER_SECOND / 10)
```

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 237 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h

MILLISECOND_TICKS_PER_QUARTERSECOND

```
#define MILLISECOND_TICKS_PER_QUARTERSECOND
```

Value:

```
(MILLISECOND_TICKS_PER_SECOND >> 2)
```

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 241 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h

MILLISECOND_TICKS_PER_MINUTE

```
#define MILLISECOND_TICKS_PER_MINUTE
```

Value:

```
(60UL * MILLISECOND_TICKS_PER_SECOND)
```

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 245 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h

MILLISECOND_TICKS_PER_HOUR

```
#define MILLISECOND_TICKS_PER_HOUR
```

Value:

```
(60UL * MILLISECOND_TICKS_PER_MINUTE)
```

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 249 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h

MILLISECOND_TICKS_PER_DAY

```
#define MILLISECOND_TICKS_PER_DAY
```

Value:

```
(24UL * MILLISECOND_TICKS_PER_HOUR)
```

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 `253` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h`

EMBER_TASK_COUNT

```
#define EMBER_TASK_COUNT
```

Value:

```
(3)
```

The number of event tasks that can be used to schedule and run events. Connect stack requires one, while another is used for Application Framework events.

Definition at line `261` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h`

emberEventControlSetInactive

```
#define emberEventControlSetInactive
```

Value:

```
(control)
```

Set EmberEventControl as inactive (no pending event).

Definition at line `268` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h`

emberEventControlGetActive

```
#define emberEventControlGetActive
```

Value:

```
(control)
```

Check whether EmberEventControl is currently active. An event is considered active if it is set to run some time in the future (activated by emberEventControlSetActive(), emberEventControlSetDelayMS() or any other emberEventControlSetDelay* functions)

Returns

- Returns **true** if the event is active **false** otherwise

Definition at line `282` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h`

emberEventControlSetActive

```
#define emberEventControlSetActive
```

Value:

```
(control)
```

Set EmberEventControl to run at the next available opportunity.

Definition at line `291` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h`

EMBER_MAX_EVENT_CONTROL_DELAY_MS

```
#define EMBER_MAX_EVENT_CONTROL_DELAY_MS
```

Value:

```
(HALF_MAX_INT32U_VALUE - 1)
```

The maximum delay that may be passed to emberEventControlSetDelayMS().

Definition at line `305` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h`

emberEventControlSetDelayMS

```
#define emberEventControlSetDelayMS
```

Value:

```
(control, delay)
```

Set EmberEventControl to run some milliseconds in the future.

Definition at line `314` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h`

EMBER_MAX_EVENT_CONTROL_DELAY_QS

```
#define EMBER_MAX_EVENT_CONTROL_DELAY_QS
```

Value:

```
(EMBER_MAX_EVENT_CONTROL_DELAY_MS >> 8)
```

The maximum delay that may be passed to emberEventControlSetDelayQS().

Definition at line `330` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h`

emberEventControlSetDelayQS

```
#define emberEventControlSetDelayQS
```

Value:

```
(control, delay)
```

Set EmberEventControl to run some quarter seconds in the future.

Warnings

- Applications should use emberEventControlSetDelayQS() instead.

Definition at line `341` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h`

EMBER_MAX_EVENT_CONTROL_DELAY_MINUTES

```
#define EMBER_MAX_EVENT_CONTROL_DELAY_MINUTES
```

Value:

```
(EMBER_MAX_EVENT_CONTROL_DELAY_MS >> 16)
```

The maximum delay that may be passed to emberEventControlSetDelayMinutes().

Definition at line `347` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h`

emberEventControlSetDelayMinutes

```
#define emberEventControlSetDelayMinutes
```

Value:

```
(control, delay)
```

Set EmberEventControl to run some minutes in the future.

Definition at line `356` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h`

emberEventControlGetRemainingMS

```
#define emberEventControlGetRemainingMS
```

Value:

```
(control)
```

Check when the event is scheduled to run.

Returns

- 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 `366` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h`

emberTaskEnableIdling

```
#define emberTaskEnableIdling
```

Value:

```
(allow)
```

Enable or disable idling.

Definition at line `477` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h`

emberMarkTaskActive

```
#define emberMarkTaskActive
```

Value:

```
(taskid)
```

Calling it indicates that a task has something to do, so it should prevent the CPU from idling until emberMarkTaskIdle is next called on this task.

Definition at line `492` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h`

Macro Definition Documentation

__EVENT_H__

```
#define __EVENT_H__
```

Definition at line `179` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/event.h`

Memory Buffer

Memory Buffer

Ember Connect API dynamically allocates and frees memory.

Generally, dynamic memory allocation in embedded code is not recommended. However, in some cases, the drawbacks of avoiding them is even bigger. Using C standard library dynamic memory is still not recommended due because it could cause fragmented memory.

For these reasons, Connect allocates some (configurable in the stack common plugin) memory as HEAP at compile-time. Memory allocation from this heap is possible through the API below.

See memory-buffer.h for source code.

APIs

This handler is invoked by the memory buffers system garbage collector and allows the application to properly mark the application-defined EmberBuffer variables with emberMarkBuffer().Implement associated callback emberAfMarkApplicationBuffersCallback() to use. See Handlers for additional information.

EmberBuffer	emberAllocateBuffer(uint16_t dataSizeInBytes) Dynamically allocates memory.
void	emberMarkBuffer(EmberBuffer *buffer) Prevent the garbage collector from reclaiming the memory associated with the passed EmberBuffer. The application should call this API within the ::emberMarkApplicationBuffersHandler() stack handler for each EmberBuffer object.
uint8_t *	emberGetBufferPointer(EmberBuffer buffer) Return a pointer to the memory segment corresponding to the passed EmberBuffer buffer. Notice that the garbage collector can move memory segments to defragment the available memory. As result, the application should always use this API to obtain an updated pointer prior to accessing the memory.
uint16_t	emberGetBufferLength(EmberBuffer buffer) Return the length in bytes of the passed EmberBuffer buffer.
uint16_t	emberGetAvailableBufferMemory(void) Return the available memory at the buffer manager in bytes.

Macros

#define	EMBER_NULL_BUFFER 0x0000u A special EmberBuffer ID indicating that no memory is currently allocated.

APIs Documentation

emberAllocateBuffer

```
EmberBuffer emberAllocateBuffer (uint16_t dataSizeInBytes)
```

Dynamically allocates memory.

Parameters

[in]	dataSizeInBytes	The size in bytes of the memory to be allocated.

Returns

- An EmberBuffer value of EMBER_NULL_BUFFER if the memory management system could not allocate the requested memory, or any other EmberBuffer value indicating that the requested memory was successfully allocated. The allocated memory can easily be freed by assigning an EmberBuffer variable to EMBER_NULL_BUFFER. The memory will be freed by the garbage collector during the next emberTick() call.

Definition at line 92 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/memory-buffer.h

emberMarkBuffer

```
void emberMarkBuffer (EmberBuffer *buffer)
```

Prevent the garbage collector from reclaiming the memory associated with the passed EmberBuffer. The application should call this API within the ::emberMarkApplicationBuffersHandler() stack handler for each EmberBuffer object.

Parameters

[in]	buffer	A pointer to the EmberBuffer buffer to be marked.

Definition at line 101 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/memory-buffer.h

emberGetBufferPointer

```
uint8_t * emberGetBufferPointer (EmberBuffer buffer)
```

Return a pointer to the memory segment corresponding to the passed EmberBuffer buffer. Notice that the garbage collector can move memory segments to defragment the available memory. As result, the application should always use this API to obtain an updated pointer prior to accessing the memory.

Parameters

[in]	buffer	A pointer to the EmberBuffer buffer for which the corresponding memory pointer should be returned.

Returns

- A NULL pointer if the passed EmberBuffer value is EMBER_NULL_BUFFER. Otherwise, a pointer to the corresponding memory segment.

Definition at line 116 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/memory-buffer.h

emberGetBufferLength

```
uint16_t emberGetBufferLength (EmberBuffer buffer)
```

Return the length in bytes of the passed EmberBuffer buffer.

Parameters

[in]	buffer	A pointer to the EmberBuffer buffer for which the corresponding length in bytes should be returned.

Returns

- The length in bytes of a memory segment corresponding to the passed EmberBuffer buffer.

Definition at line `127` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/memory-buffer.h`

emberGetAvailableBufferMemory

```
uint16_t emberGetAvailableBufferMemory (void)
```

Return the available memory at the buffer manager in bytes.

Parameters

N/A		

Returns

- The number of available bytes.

Definition at line `134` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/memory-buffer.h`

Macro Definition Documentation

EMBER_NULL_BUFFER

```
#define EMBER_NULL_BUFFER
```

Value:

```
0x0000u
```

A special EmberBuffer ID indicating that no memory is currently allocated.

Definition at line `50` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/memory-buffer.h`

Messaging

Messaging

Connect APIs and handlers for sending and receiving messages.

t_stack

Note that MAC mode and Extended star/direct mode use different APIs for messaging.

See message.h for source code.

Handlers

The Application Framework implements all handlers, directly calling their associated callbacks. By default, Connect projects declare such callbacks as stubs in flex-callbacks-stubs.c. Hence, to use an enabled Connect feature, applications should replace the stub with their own implementation of the associated callback (typically in flex-callbacks.c). See UG235.04 for more info.

void	emberMessageSentHandler(EmberStatus status, EmberOutgoingMessage *message) This handler is invoked when the stack has completed sending a message.
void	emberMacMessageSentHandler(EmberStatus status, EmberOutgoingMacMessage *message) This handler is invoked when a node of EMBER_MAC_MODE_DEVICE type or EMBER_MAC_MODE_SLEEPY_DEVICE type has completed sending a MAC frame.
void	emberIncomingMessageHandler(EmberIncomingMessage *message) This handler is invoked when a packet has been received from a node type other than EMBER_MAC_MODE_DEVICE or EMBER_MAC_MODE_SLEEPY_DEVICE.
void	emberIncomingMacMessageHandler(EmberIncomingMacMessage *message) This handler is invoked when a node of EMBER_MAC_MODE_DEVICE type or EMBER_MAC_MODE_SLEEPY_DEVICE has received a MAC frame.
EmberStatus	emberMessageSend(EmberNodeId destination, uint8_t endpoint, uint8_t messageTag, EmberMessageLength messageLength, uint8_t *message, EmberMessageOptions options) Send a message to the passed destination short ID.
EmberStatus	emberMacMessageSend(EmberMacFrame *macFrame, uint8_t messageTag, EmberMessageLength messageLength, uint8_t *message, EmberMessageOptions options) Create a MAC level frame and sends it to the passed destination. This API can only be used for nodes of EMBER_MAC_MODE_DEVICE node type or EMBER_MAC_MODE_SLEEPY_DEVICE node type.
EmberStatus	emberPollForData(void) Send a data request command to the parent node. Note that if the node short ID is a value of EMBER_USE_LONG_ADDRESS, the node shall use its long ID as source address.
EmberStatus	emberSetPollDestinationAddress(EmberMacAddress *destination) Set data polls destination address for nodes of EMBER_MAC_MODE_DEVICE node type or EMBER_MAC_MODE_SLEEPY_DEVICE node type.
uint16_t	emberGetMaximumPayloadLength(EmberMacAddressMode srcAddressMode, EmberMacAddressMode dstAddressMode, bool interpan, bool secured) Return the maximum payload according to the passed source and destination addressing modes, the passed secured flag, and the current configuration of the node.

bool	emberUsingLongMessages(void) Indicates if the stack is currently using long messages or not.
EmberStatus	emberNcpSetLongMessagesUse(bool useLongMessages) Set the current message length that the stack uses.
EmberStatus	emberPurgeIndirectMessages(void) Purge all indirect transmissions from the indirect message queue.
EmberStatus	emberSetIndirectQueueTimeout(uint32_t timeoutMs) Set indirect queue timeout value. The indirect queue timeout is set by default to EMBER_INDIRECT_TRANSMISSION_TIMEOUT_MS.

Macros

#define	EMBER_MAX_UNSECURED_APPLICATION_PAYLOAD_LENGTH 111
#define	EMBER_MAX_SECURED_APPLICATION_PAYLOAD_LENGTH 102
#define	EMBER_MAX_ENDPOINT 0xF The maximum allowed endpoint value.

Handlers Documentation

emberMessageSentHandler

```
void emberMessageSentHandler (EmberStatus status, EmberOutgoingMessage *message)
```

This handler is invoked when the stack has completed sending a message.

Parameters

[in]	status	An EmberStatus value of: • EMBER_SUCCESS if an ACK was received from the destination or no ACK was requested. • EMBER_MAC_NO_ACK_RECEIVED if an ACK was requested and no ACK was received. • EMBER_MAC_INDIRECT_TIMEOUT if the destination is a sleepy node and the packet timed-out before the sleepy node sent a data request. • EMBER_MAC_INDIRECT_MESSAGE_PURGED if the destination is a sleepy node and it was removed from the child table while the packet was stored in the indirect queue. • EMBER_PHY_TX_CCA_FAIL if the node failed all the clear channel assessment attempts. • EMBER_PHY_TX_INCOMPLETE if the transmission was not completed correctly.
[in]	message	An EmberOutgoingMessage describing the outgoing packet.

Warnings

- Implement associated callback emberAfMessageSentCallback() to use. See Handlers for additional information.

Definition at line `95` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/message.h`

emberMacMessageSentHandler

```
void emberMacMessageSentHandler (EmberStatus status, EmberOutgoingMacMessage *message)
```

This handler is invoked when a node of EMBER_MAC_MODE_DEVICE type or EMBER_MAC_MODE_SLEEPY_DEVICE type has completed sending a MAC frame.

Parameters

[in]	status	An EmberStatus value of: • EMBER_SUCCESS if an ACK was received from the destination or no ACK was requested. • EMBER_MAC_NO_ACK_RECEIVED if an ACK was requested and no ACK was received. • EMBER_MAC_INDIRECT_TIMEOUT if the MAC frame was sent out via the indirect queue and the it timed-out before a data request was received. • EMBER_MAC_INDIRECT_MESSAGE_PURGED if the MAC frame was sent out via the indirect queue and it was removed prior to a data request being received. See emberPurgeIndirectMessages(). • EMBER_MAC_SECURITY_FAILED if the stack failed to encrypt the message. This typically occurs when a node is sending a message using short source addressing with an address other than the node's short address and the no mapping to a corresponding address was found in the short-to-long address mapping table. The application should use the emberMacAddShortToLongAddressMapping to populate such table. • EMBER_PHY_TX_CCA_FAIL if the node failed all the clear channel assessment attempts. • EMBER_PHY_TX_INCOMPLETE if the transmission was not completed correctly.
[in]	message	An EmberOutgoingMacMessage describing the outgoing MAC frame.

Warnings

- Implement associated callback emberAfMacMessageSentCallback() to use. See Handlers for additional information.

Definition at line 127 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/message.h

emberIncomingMessageHandler

```
void emberIncomingMessageHandler (EmberIncomingMessage *message)
```

This handler is invoked when a packet has been received from a node type other than EMBER_MAC_MODE_DEVICE or EMBER_MAC_MODE_SLEEPY_DEVICE.

Parameters

[in]	message	An EmberIncomingMessage describing the incoming packet.

Warnings

- Implement associated callback emberAfIncomingMessageCallback() to use. See Handlers for additional information.

Definition at line 139 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/message.h

emberIncomingMacMessageHandler

```
void emberIncomingMacMessageHandler (EmberIncomingMacMessage *message)
```

This handler is invoked when a node of EMBER_MAC_MODE_DEVICE type or EMBER_MAC_MODE_SLEEPY_DEVICE has received a MAC frame.

Parameters

[in]	message	An EmberIncomingMacMessage describing the incoming packet.

Warnings

- Implement associated callback emberAfIncomingMacMessageCallback() to use. See Handlers for additional information.

Definition at line 152 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/message.h

emberMessageSend

```
EmberStatus emberMessageSend (EmberNodeId destination, uint8_t endpoint, uint8_t messageTag, EmberMessageLength
messageLength, uint8_t *message, EmberMessageOptions options)
```

Send a message to the passed destination short ID.

Parameters

[in]	destination	The destination node short ID.
[in]	endpoint	The destination endpoint of the outgoing message. This value can't exceed EMBER_MAX_ENDPOINT.
[in]	messageTag	A value chosen by the application. This value will be passed in the corresponding emberMessageSentHandler() call.
[in]	messageLength	The size of the message payload in bytes. Use the emberGetMaximumPayloadLength() API to determine the maximum message length allowed.
[in]	message	A pointer to an array of bytes containing the message payload.
[in]	options	Specifies the EmberMessageOptions for the outgoing message.

Returns

- an EmberStatus value of:
  - EMBER_SUCCESS if the message was accepted by the stack. If a success status is returned, the emberMessageSentHandler() callback is invoked by the stack to indicate whether the message was successfully delivered or the reason for failure.
  - EMBER_INVALID_CALL if the node is not joined to a network or the node is of EMBER_MAC_MODE_DEVICE device type or EMBER_MAC_MODE_SLEEPY_DEVICE (use emberMacMessageSend instead).
  - EMBER_BAD_ARGUMENT if the packet length is 0, the passed TX options indicates some feature that is not supported, the passed endpoint exceeds EMBER_MAX_ENDPOINT
  - EMBER_MESSAGE_TOO_LONG if the message does not fit in a single frame.
  - EMBER_PHY_TX_BUSY if the message cannot be sent since the node does not support MAC queuing and the radio is currently busy.
  - EMBER_MAC_TRANSMIT_QUEUE_FULL if the outgoing MAC queue is currently full.
  - EMBER_NO_BUFFERS if the stack could not allocate enough RAM to store the submitted message.
  - EMBER_MAC_UNKNOWN_DESTINATION if the node is part of a star network and the destination node does not appear in the node's routing table.
  - EMBER_MAC_SECURITY_NOT_SUPPORTED if the message was requested to be sent out secured and either the local node does not support security or the destination node is known to not support security.
  - EMBER_MAC_BUSY if the message was not accepted because the MAC is currently performing some critical operation.

Definition at line `203` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/message.h`

emberMacMessageSend

```
EmberStatus emberMacMessageSend (EmberMacFrame *macFrame, uint8_t messageTag, EmberMessageLength
messageLength, uint8_t *message, EmberMessageOptions options)
```

Create a MAC level frame and sends it to the passed destination. This API can only be used for nodes of EMBER_MAC_MODE_DEVICE node type or EMBER_MAC_MODE_SLEEPY_DEVICE node type.

Parameters

[in]	macFrame	A pointer to an EmberMacFrame struct that specifies the source and destination addresses and the source and destination PAN IDs for the message to be sent. Note that if the source/destination PAN ID is not specified, it defaults to the node's PAN ID. Also, the destination address mode must be either EMBER_MAC_ADDRESS_MODE_SHORT or EMBER_MAC_ADDRESS_MODE_LONG.

[in]	messageTag	A value chosen by the application. This value will be passed in the corresponding emberMacMessageSentHandler() call.
[in]	messageLength	The size in bytes of the message payload. The application can use the emberGetMaximumPayloadLength() API to determine the maximum allowable payload, given a permutation of source and destination addressing and other TX options.
[in]	message	A pointer to an array of bytes containing the message payload.
[in]	options	Specifies the EmberMessageOptions for the outgoing message.

Returns

- an EmberStatus value of:
  - EMBER_SUCCESS if the message was accepted by the stack. If a success status is returned, the emberMacMessageSentHandler() callback will be invoked by the stack to indicate whether the message was successfully delivered or the reason for failure.
  - EMBER_INVALID_CALL if the node is of a node type other than EMBER_MAC_MODE_DEVICE or EMBER_MAC_MODE_SLEEPY_DEVICE.
  - EMBER_BAD_ARGUMENT if the packet length is 0, the passed TX options indicates some feature that is not supported or the destination address mode is set to EMBER_MAC_ADDRESS_MODE_NONE.
  - EMBER_MESSAGE_TOO_LONG if the message does not fit in a single frame.
  - EMBER_PHY_TX_BUSY if the message cannot be sent since the node does not support MAC queuing and the radio is currently busy.
  - EMBER_MAC_TRANSMIT_QUEUE_FULL if the outgoing MAC queue is currently full.
  - EMBER_NO_BUFFERS if the stack could not allocate enough RAM to store the submitted message.
  - EMBER_MAC_SECURITY_NOT_SUPPORTED if the message was requested to be sent out with a security but no security plugin was enabled.
  - EMBER_MAC_BUSY if the message was not accepted because the MAC is currently performing a critical operation.

Definition at line `257` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/message.h`

emberPollForData

```
EmberStatus emberPollForData (void)
```

Send a data request command to the parent node. Note that if the node short ID is a value of EMBER_USE_LONG_ADDRESS, the node shall use its long ID as source address.

Parameters

N/A		

Returns

- and EmberStatus value of:
  - EMBER_SUCCESS if the data poll was accepted by the MAC layer.
  - EMBER_INVALID_CALL if the node is not joined to a network, the node is not an end device, an EMBER_MAC_MODE_DEVICE or an EMBER_MAC_MODE_SLEEPY_DEVICE, or the node is of EMBER_MAC_MODE_DEVICE or EMBER_MAC_MODE_SLEEPY_DEVICE node type, is not joined to a coordinator and the poll destination was not correctly set via the emberSetPollDestinationAddress() API.
  - EMBER_MAC_BUSY if the MAC is currently performing a critical operation.

Definition at line `278` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/message.h`

emberSetPollDestinationAddress

```
EmberStatus emberSetPollDestinationAddress (EmberMacAddress *destination)
```

Set data polls destination address for nodes of EMBER_MAC_MODE_DEVICE node type or EMBER_MAC_MODE_SLEEPY_DEVICE node type.

Parameters

N/A	destination	

Returns

- and EmberStatus value of EMBER_SUCCESS if the data poll destination was correctly set, or another EmberStatus value indicating the reason of failure.

Definition at line `288` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/message.h`

emberGetMaximumPayloadLength

```
uint16_t emberGetMaximumPayloadLength (EmberMacAddressMode srcAddressMode, EmberMacAddressMode
dstAddressMode, bool interpan, bool secured)
```

Return the maximum payload according to the passed source and destination addressing modes, the passed secured flag, and the current configuration of the node.

Parameters

[in]	srcAddressMode	An EmberMacAddressMode value indicating the mode of the source address. Note, this parameter is only meaningful if the node was started as EMBER_MAC_MODE_DEVICE or EMBER_MAC_MODE_SLEEPY_DEVICE.
[in]	dstAddressMode	An EmberMacAddressMode value indicating the mode of the destination address. Note, this parameter is only meaningful if the node was started as EMBER_MAC_MODE_DEVICE or EMBER_MAC_MODE_SLEEPY_DEVICE.
[in]	interpan	Indicates whether the frame is an interpan frame or not. Note, this parameter is only meaningful if the node was started as EMBER_MAC_MODE_DEVICE or EMBER_MAC_MODE_SLEEPY_DEVICE.
[in]	secured	Indicates whether the frame should be secured or not.

Returns

- The maximum payload length in bytes achievable according to the passed parameters or **0xFF** if the node is currently active on a network or any of the passed parameters are invalid.

Definition at line `339` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/message.h`

emberUsingLongMessages

```
bool emberUsingLongMessages (void)
```

Indicates if the stack is currently using long messages or not.

Parameters

N/A		

Returns

- True if the stack currently uses long messages (length stored in a uint16_t) or false if it is not the case (length stored in a uint8_t).

Definition at line `351` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/message.h`

emberNcpSetLongMessagesUse

```
EmberStatus emberNcpSetLongMessagesUse (bool useLongMessages)
```

Set the current message length that the stack uses.

Parameters

[in]	useLongMessages	True to use long messages (length stored in a uint16_t), false to use short messages (length stored in a uint8_t).

Note

- This API is here to assure retro compatibility with old NCP Host lib versions. In NCP Host lib versions that do not support OFDM features (v1.1 and older), only short messaging is supported. For the NCP, short messages are used by default. The Host lib needs to call this API with useLongMessages set to true if it supports OFDM.

Warnings

- This API changes the behavior of the Connect Serialization Protocol. It only has effect when using a RTOS or the NCP. Changing it may result in packets being incorrectly transfered through CSP when using a SUN-OFDM or SUN-FSK PHY.

Definition at line `370` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/message.h`

emberPurgeIndirectMessages

```
EmberStatus emberPurgeIndirectMessages (void)
```

Purge all indirect transmissions from the indirect message queue.

Parameters

N/A		

Returns

- an EmberStatus value of EMBER_SUCCESS if all indirect messages were purged, or another EmberStatus value indicating the reason of failure.

Definition at line `299` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/message.h`

emberSetIndirectQueueTimeout

```
EmberStatus emberSetIndirectQueueTimeout (uint32_t timeoutMs)
```

Set indirect queue timeout value. The indirect queue timeout is set by default to EMBER_INDIRECT_TRANSMISSION_TIMEOUT_MS.

Parameters

N/A	timeoutMs	The timeout in milliseconds to be set.

Returns

- an EmberStatus value of EMBER_SUCCESS if the passed timeout was successfully set, or a value of EMBER_BAD_ARGUMENT if the passed value is invalid.

Definition at line `313` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/message.h`

Macro Definition Documentation

EMBER_MAX_UNSECURED_APPLICATION_PAYLOAD_LENGTH

```
#define EMBER_MAX_UNSECURED_APPLICATION_PAYLOAD_LENGTH
```

Value:

```
111
```

DeprecatedThe maximum length in bytes of the application payload for an unsecured message. This define has been deprecated, you should use the emberGetMaximumPayloadLength API instead.

Definition at line `48` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/message.h`

EMBER_MAX_SECURED_APPLICATION_PAYLOAD_LENGTH

```
#define EMBER_MAX_SECURED_APPLICATION_PAYLOAD_LENGTH
```

Value:

```
102
```

DeprecatedThe maximum length in bytes of the application payload for a secured message. This define has been deprecated, you should use the emberGetMaximumPayloadLength API instead.

Definition at line `54` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/message.h`

EMBER_MAX_ENDPOINT

```
#define EMBER_MAX_ENDPOINT
```

Value:

```
0xF
```

The maximum allowed endpoint value.

Definition at line `58` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/stack/include/message.h`

Connect Application Framework API Reference

Connect Application Framework API Reference

Application Framework includes plugins that are built on top of the Connect stack.

Modules

Application Framework Common

Command Interpreter Plugin

Debug Print Plugin

Mailbox Client Plugin

Mailbox Server Plugin

Mailbox Common

Ota Unicast Bootloader Client Plugin

Ota Unicast Bootloader Server Plugin

Ota Unicast Bootloader Common

Ota Broadcast Bootloader Client Plugin

Ota Broadcast Bootloader Server Plugin

Ota Broadcast Bootloader Common

Poll Plugin

WSTK Sensors Plugin

Application Framework Common

Application Framework Common

Application framework common.

Declare all required application framework globals, initialize the Connect stack, and dispatch stack callbacks calls as needed to the application components.

Callbacks

void	emberAfInitCallback(void) Application Framework Initialization Callback.
void	emberAfTickCallback(void) Application Framework Tick Callback.
void	emberAfStackStatusCallback(EmberStatus status) Application framework equivalent of emberStackStatusHandler.
void	emberAfIncomingMessageCallback(EmberIncomingMessage *message) Application framework equivalent of emberIncomingMessageHandler.
void	emberAfIncomingMacMessageCallback(EmberIncomingMacMessage *message) Application framework equivalent of emberIncomingMacMessageHandler.
void	emberAfMessageSentCallback(EmberStatus status, EmberOutgoingMessage *message) Application framework equivalent of emberMessageSentHandler.
void	emberAfMacMessageSentCallback(EmberStatus status, EmberOutgoingMacMessage *message) Application framework equivalent of emberMacMessageSentHandler.
void	emberAfChildJoinCallback(EmberNodeType nodeType, EmberNodeId nodeId) Application framework equivalent of emberChildJoinHandler.
void	emberAfActiveScanCompleteCallback(void) Application framework equivalent of emberActiveScanCompleteHandler.
void	emberAfEnergyScanCompleteCallback(int8_t mean, int8_t min, int8_t max, uint16_t variance) Application framework equivalent of emberEnergyScanCompleteHandler.
void	emberAfMarkApplicationBuffersCallback(void) Application framework equivalent of ::emberMarkApplicationBuffersHandler.
void	emberAfIncomingBeaconCallback(EmberPanId panId, EmberMacAddress *source, int8_t rssi, bool permitJoining, uint8_t beaconFieldsLength, uint8_t *beaconFields, uint8_t beaconPayloadLength, uint8_t *beaconPayload) Application framework equivalent of emberIncomingBeaconHandler.
void	emberAfFrequencyHoppingStartClientCompleteCallback(EmberStatus status) Application framework equivalent of emberFrequencyHoppingStartClientCompleteHandler.
void	emberAfRadioNeedsCalibratingCallback(void) Application framework equivalent of emberRadioNeedsCalibratingHandler.

bool	emberAfStackIdleCallback(uint32_t *idleTimeMs) Application framework equivalent of emberStackIdleHandler.
bool	emberAfCommonOkToEnterLowPowerCallback(bool enter_em2, uint32_t duration_ms) Application framework Low Power notification Callback.

Functions

uint32_t	emberAfGetResetCause(void) Get the last reset cause mask.
EmberStatus	emberAfAllocateEvent(EmberEventControl **control, void(*handler)(void)) Allocate a new event to the app event table.

Callbacks Documentation

emberAfInitCallback

```
void emberAfInitCallback (void)
```

Application Framework Initialization Callback.

Parameters

N/A		

A callback invoked once during the initialization. It is called after the stack and plugins initialization.

Definition at line `53` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/app-framework-common/app_framework_callback.h`

emberAfTickCallback

```
void emberAfTickCallback (void)
```

Application Framework Tick Callback.

Parameters

N/A		

A callback invoked in each iteration of the application super loop and can be used to perform periodic functions. The frequency with which this function is called depends on how quickly the main loop runs. If the application blocks at any time during the main loop, this function will not be called until execution resumes.

Definition at line `64` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/app-framework-common/app_framework_callback.h`

emberAfStackStatusCallback

```
void emberAfStackStatusCallback (EmberStatus status)
```

Application framework equivalent of emberStackStatusHandler.

Parameters

N/A	status	

Definition at line `68` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/app-framework-common/app_framework_callback.h`

emberAfIncomingMessageCallback

```
void emberAfIncomingMessageCallback (EmberIncomingMessage *message)
```

Application framework equivalent of emberIncomingMessageHandler.

Parameters

N/A	message	

Definition at line `72` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/app-framework-common/app_framework_callback.h`

emberAfIncomingMacMessageCallback

```
void emberAfIncomingMacMessageCallback (EmberIncomingMacMessage *message)
```

Application framework equivalent of emberIncomingMacMessageHandler.

Parameters

N/A	message	

Definition at line `76` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/app-framework-common/app_framework_callback.h`

emberAfMessageSentCallback

```
void emberAfMessageSentCallback (EmberStatus status, EmberOutgoingMessage *message)
```

Application framework equivalent of emberMessageSentHandler.

Parameters

N/A	status	
N/A	message	

Definition at line `80` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/app-framework-common/app_framework_callback.h`

emberAfMacMessageSentCallback

```
void emberAfMacMessageSentCallback (EmberStatus status, EmberOutgoingMacMessage *message)
```

Application framework equivalent of emberMacMessageSentHandler.

Parameters

N/A	status	
N/A	message	

Definition at line `85` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/app-framework-common/app_framework_callback.h`

emberAfChildJoinCallback

```
void emberAfChildJoinCallback (EmberNodeType nodeType, EmberNodeId nodeId)
```

Application framework equivalent of emberChildJoinHandler.

Parameters

N/A	nodeType	
N/A	nodeId	

Warnings

- Requires the parent support plugin installed.

Definition at line `91` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/app-framework-common/app_framework_callback.h`

emberAfActiveScanCompleteCallback

```
void emberAfActiveScanCompleteCallback (void)
```

Application framework equivalent of emberActiveScanCompleteHandler.

Parameters

N/A		

Definition at line `96` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/app-framework-common/app_framework_callback.h`

emberAfEnergyScanCompleteCallback

```
void emberAfEnergyScanCompleteCallback (int8_t mean, int8_t min, int8_t max, uint16_t variance)
```

Application framework equivalent of emberEnergyScanCompleteHandler.

Parameters

N/A	mean	
N/A	min	
N/A	max	
N/A	variance	

Definition at line `100` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/app-framework-common/app_framework_callback.h`

emberAfMarkApplicationBuffersCallback

```
void emberAfMarkApplicationBuffersCallback (void)
```

Application framework equivalent of ::emberMarkApplicationBuffersHandler.

Parameters

N/A		

Definition at line `107` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/app-framework-common/app_framework_callback.h`

emberAfIncomingBeaconCallback

```
void emberAfIncomingBeaconCallback (EmberPanId panId, EmberMacAddress *source, int8_t rssi, bool permitJoining,
uint8_t beaconFieldsLength, uint8_t *beaconFields, uint8_t beaconPayloadLength, uint8_t *beaconPayload)
```

Application framework equivalent of emberIncomingBeaconHandler.

Parameters

N/A	panId	
N/A	source	
N/A	rssi	
N/A	permitJoining	
N/A	beaconFieldsLength	
N/A	beaconFields	
N/A	beaconPayloadLength	
N/A	beaconPayload	

Definition at line `111` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/app-framework-common/app_framework_callback.h`

emberAfFrequencyHoppingStartClientCompleteCallback

```
void emberAfFrequencyHoppingStartClientCompleteCallback (EmberStatus status)
```

Application framework equivalent of emberFrequencyHoppingStartClientCompleteHandler.

Parameters

N/A	status	

Definition at line `122` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/app-framework-common/app_framework_callback.h`

emberAfRadioNeedsCalibratingCallback

```
void emberAfRadioNeedsCalibratingCallback (void)
```

Application framework equivalent of emberRadioNeedsCalibratingHandler.

Parameters

N/A		

Definition at line `126` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/app-framework-common/app_framework_callback.h`

emberAfStackIdleCallback

```
bool emberAfStackIdleCallback (uint32_t *idleTimeMs)
```

Application framework equivalent of emberStackIdleHandler.

Parameters

N/A	idleTimeMs	

Definition at line `130` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/app-framework-common/app_framework_callback.h`

emberAfCommonOkToEnterLowPowerCallback

```
bool emberAfCommonOkToEnterLowPowerCallback (bool enter_em2, uint32_t duration_ms)
```

Application framework Low Power notification Callback.

Parameters

[in]	enter_em2	**true** if the system is about to sleep or **false** to idle.
[in]	duration_ms	Duration of the low power period. Time to the next event.

A callback invoked when the system is about to go sleeping.

Returns

- **true** if the application allows the system to go to sleep.

Definition at line `142` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/app-framework-common/app_framework_callback.h`

Function Documentation

emberAfGetResetCause

```
uint32_t emberAfGetResetCause (void)
```

Get the last reset cause mask.

Parameters

N/A		

Returns

- A reset cause mask.

Note

- This API replaces halGetResetInfo() or halGetExtendedResetInfo. emberAfGetResetCause() is a RMU_ResetCauseGet() overhaul. See the reference manual of the EMLIB RMU for a description of the returned reset cause mask.

Definition at line `62` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/app-framework-common/app_framework_common.h`

emberAfAllocateEvent

```
EmberStatus emberAfAllocateEvent (EmberEventControl **control, void(*handler)(void))
```

Allocate a new event to the app event table.

Parameters

[out]	control	The EmberEventControl to allocate
[in]	handler	Pointer to the handler function associated to the event

Returns

- An EmberStatus value of:
  - EMBER_SUCCESS if the event was successfully allocated.
  - EMBER_TABLE_FULL if no more event could be allocated.

See Also

- emberAfAllocateEvent()

Definition at line `77` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/app-framework-common/app_framework_common.h`

Command Interpreter Plugin

Command Interpreter Plugin

Debug Print Plugin

Debug Print Plugin

Mailbox Client Plugin

Mailbox Client Plugin

APIs for mailbox client.

Mailbox protocol is designed for devices that can't be online on the network all the time. The most common example for this is a sleepy end device.

Mailbox clients and the server can submit messages into the mailbox, which is stored in RAM on the mailbox server. Clients can then query the mailbox server for available messages.

The mailbox server will notify clients who submit messages when a message was delivered or when it couldn't be delivered due to an error.

Mailbox uses a plugin-configurable protocol endpoint, which is 15 by default.

The server can also configure the size of the mailbox (in number of packets, 25 by default) and the packet timeout, after which the server drops the message and notifies the source of the error.

The mailbox protocol uses standard data messages, so in case of sleepy end devices, it will use the indirect queue. This means that if a sleepy end device sends a request to a mailbox server, the end device should poll for the response.

Note

- Mailbox is not available in MAC mode due to the lack of endpoints.

See mailbox-client.h and mailbox-client.c for source code.

Callbacks

void	emberAfPluginMailboxClientMessageSubmitCallback(EmberAfMailboxStatus status, EmberNodeId mailboxServer, EmberNodeId messageDestination, uint8_t tag) Mailbox Client Message Submit Callback.
void	emberAfPluginMailboxClientMessageDeliveredCallback(EmberAfMailboxStatus status, EmberNodeId mailboxServer, EmberNodeId messageDestination, uint8_t tag) Mailbox Client Message Delivered Callback.
void	emberAfPluginMailboxClientCheckInboxCallback(EmberAfMailboxStatus status, EmberNodeId mailboxServer, EmberNodeId messageSource, uint8_t *message, EmberMessageLength messageLength, uint8_t tag, bool moreMessages) Mailbox Client Check Inbox Callback.

Functions

EmberAfMailboxStatus	emberAfPluginMailboxClientMessageSubmit(EmberNodeId mailboxServer, EmberNodeId messageDestination, uint8_t *message, EmberMessageLength messageLength, uint8_t tag, bool useSecurity) Submit a data message to a mailbox server. If this API returns an EmberAfMailboxStatus value of EMBER_MAILBOX_STATUS_SUCCESS, the corresponding asynchronous callback emberAfPluginMailboxClientMessageSubmitCallback() will be invoked to indicate whether the message was successfully submitted to the mailbox server or to inform the application of the reason of failure.
EmberAfMailboxStatus	emberAfPluginMailboxClientCheckInbox(EmberNodeId mailboxServer, bool useSecurity) Query a mailbox server for pending messages. If this API returns an EmberAfMailboxStatus value of EMBER_MAILBOX_STATUS_SUCCESS, the corresponding asynchronous callback

emberAfPluginMailboxClientCheckInboxCallback() will be invoked either to provide the retrieved message or to indicate the reason for failure.

Callbacks Documentation

emberAfPluginMailboxClientMessageSubmitCallback

```
void emberAfPluginMailboxClientMessageSubmitCallback (EmberAfMailboxStatus status, EmberNodeId mailboxServer,
EmberNodeId messageDestination, uint8_t tag)
```

Mailbox Client Message Submit Callback.

Parameters

[in]	status	An EmberAfMailboxStatus value of: • EMBER_MAILBOX_STATUS_SUCCESS if the data message was accepted by the mailbox server. • EMBER_MAILBOX_STATUS_STACK_ERROR if the message couldn't be delivered to the mailbox server. • EMBER_MAILBOX_STATUS_MESSAGE_NO_RESPONSE if the client timed-out waiting for a response from the server. • EMBER_MAILBOX_STATUS_MESSAGE_TABLE_FULL if the mailbox server table is currently full. • EMBER_MAILBOX_STATUS_MESSAGE_NO_BUFFERS if the server can't allocate enough memory to store the message.
[in]	mailboxServer	The node ID of the mailbox server.
[in]	messageDestination	The node ID of the destination.
N/A	tag	The tag value passed in the emberAfPluginMailboxClientMessageSubmit() API.

A callback invoked when a message arrived to the mailbox server after a call of emberAfPluginMailboxClientMessageSubmit().

Note

- Receiving this callback requires the reception of a mailbox command message, which is only possible by polling if the message was submitted on a EMBER_STAR_SLEEPY_END_DEVICE.

Definition at line `168` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/mailbox/mailbox-client/mailbox-client.h`

emberAfPluginMailboxClientMessageDeliveredCallback

```
void emberAfPluginMailboxClientMessageDeliveredCallback (EmberAfMailboxStatus status, EmberNodeId mailboxServer,
EmberNodeId messageDestination, uint8_t tag)
```

Mailbox Client Message Delivered Callback.

Parameters

[in]	status	An EmberAfMailboxStatus value of: • EMBER_MAILBOX_STATUS_SUCCESS indicates that the message was successfully delivered to the final destination. • EMBER_MAILBOX_STATUS_MESSAGE_TIMED_OUT indicates that the message timed-out and was removed from the server queue.
[in]	mailboxServer	The node ID of the mailbox server where the message was submitted to.

[in]	messageDestination	The node ID of the destination.
[in]	tag	The tag value passed in the emberAfPluginMailboxClientMessageSubmit() API.

A callback that may be invoked on the submitter of the message either if the message that was submitted to a mailbox server reached its final destination or it timed-out. Note that the callback is not always called. If the status message from the server is lost, the callback won't be called.

Note

- Receiving this callback requires the reception of a mailbox command message, which is only possible by polling if the message was submitted on a EMBER_STAR_SLEEPY_END_DEVICE.

Definition at line `199` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/mailbox/mailbox-client/mailbox-client.h`

emberAfPluginMailboxClientCheckInboxCallback

```
void emberAfPluginMailboxClientCheckInboxCallback (EmberAfMailboxStatus status, EmberNodeId mailboxServer,
EmberNodeId messageSource, uint8_t *message, EmberMessageLength messageLength, uint8_t tag, bool
moreMessages)
```

Mailbox Client Check Inbox Callback.

Parameters

[in]	status	An EmberAfMailboxStatus value of: • EMBER_MAILBOX_STATUS_SUCCESS if a message was retrieved from the mailbox server. • EMBER_MAILBOX_STATUS_MESSAGE_NO_DATA if the server has currently no message for this mailbox client. • EMBER_MAILBOX_STATUS_MESSAGE_NO_RESPONSE if the client timed-out waiting for a query response from the mailbox server. • EMBER_MAILBOX_STATUS_STACK_ERROR if the stack failed to deliver the query message to the mailbox server.
[in]	mailboxServer	The node id of the mailbox server responding.
[in]	messageSource	The source node ID of the retrieved message. Note that this parameter is meaningful only if the status parameter has an EmberAfMailboxStatus value of EMBER_MAILBOX_STATUS_SUCCESS.
[in]	message	A pointer to the retrieved message payload. Note that this parameter is meaningful only if the status parameter has an EmberAfMailboxStatus value of EMBER_MAILBOX_STATUS_SUCCESS.
[in]	messageLength	The length in bytes of the retrieved message payload. Note that this parameter is meaningful only if the status parameter has an EmberAfMailboxStatus value of EMBER_MAILBOX_STATUS_SUCCESS.
[in]	tag	The tag value passed in the emberAfPluginMailboxClientMessageSubmit() API. Note that this parameter is meaningful only if the status parameter has an EmberAfMailboxStatus value of EMBER_MAILBOX_STATUS_SUCCESS.
[in]	moreMessages	This flag is **true** if the mailbox server has more pending messages for this mailbox client. Note that this parameter is meaningful only if the status parameter has an EmberAfMailboxStatus value of EMBER_MAILBOX_STATUS_SUCCESS.

This callback is invoked after a successful call to the emberAfPluginMailboxClientCheckInbox() API. If a message was retrieved from the mailbox server, this callback passes it to the application. Otherwise, it indicates the reason for failure to the application.

Note

- Receiving this callback requires the reception of a mailbox command message, which is only possible by polling if the message was submitted on a EMBER_STAR_SLEEPY_END_DEVICE.

Definition at line `250` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/mailbox/mailbox-client/mailbox-client.h`

Function Documentation

emberAfPluginMailboxClientMessageSubmit

```
EmberAfMailboxStatus emberAfPluginMailboxClientMessageSubmit (EmberNodeId mailboxServer, EmberNodeId
messageDestination, uint8_t *message, EmberMessageLength messageLength, uint8_t tag, bool useSecurity)
```

Submit a data message to a mailbox server. If this API returns an EmberAfMailboxStatus value of EMBER_MAILBOX_STATUS_SUCCESS, the corresponding asynchronous callback emberAfPluginMailboxClientMessageSubmitCallback() will be invoked to indicate whether the message was successfully submitted to the mailbox server or to inform the application of the reason of failure.

Parameters

[in]	mailboxServer	The node ID of the mailbox server.
[in]	messageDestination	The node ID of the destination for this data message.
[in]	message	A pointer to the message to be sent.
[in]	messageLength	The length in bytes of the message to be sent.
[in]	tag	A tag value which will be returned in all of the corresponding callbacks: emberAfPluginMailboxClientMessageSubmitCallback(), emberAfPluginMailboxClientMessageDeliveredCallback() and emberAfPluginMailboxClientCheckInboxCallback(). The application can use it to match the callbacks with the call.
[in]	useSecurity	Set it **true** if the data message should be sent to the server using security.

Returns

- An EmberAfMailboxStatus value of:
  - EMBER_MAILBOX_STATUS_SUCCESS if the message was successfully passed to the network layer to be transmitted to the mailbox server.
  - EMBER_MAILBOX_STATUS_INVALID_CALL if the passed data message is invalid.
  - EMBER_MAILBOX_STATUS_INVALID_ADDRESS if the server ID or the destination ID is an invalid address.
  - EMBER_MAILBOX_STATUS_MESSAGE_TOO_LONG if the passed message does not fit in a single mailbox data message.
  - EMBER_MAILBOX_STATUS_BUSY if the client is still performing a submit message or a query for message action.
  - EMBER_MAILBOX_STATUS_STACK_ERROR if the network layer refused the message (the outgoing queue is currently full).

Note

- Receiving the emberAfPluginMailboxClientMessageSubmitCallback() requires the reception of a mailbox command message, which is only possible by polling if the message was submitted on a EMBER_STAR_SLEEPY_END_DEVICE.

See Also

- emberAfPluginMailboxServerAddMessage()

Definition at line `92` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/mailbox/mailbox-client/mailbox-client.h`

emberAfPluginMailboxClientCheckInbox

```
EmberAfMailboxStatus emberAfPluginMailboxClientCheckInbox (EmberNodeId mailboxServer, bool useSecurity)
```

Query a mailbox server for pending messages. If this API returns an EmberAfMailboxStatus value of EMBER_MAILBOX_STATUS_SUCCESS, the corresponding asynchronous callback emberAfPluginMailboxClientCheckInboxCallback() will be invoked either to provide the retrieved message or to indicate the reason for failure.

Parameters

[in]	mailboxServer	The node ID of the mailbox server.
[in]	useSecurity	Set it **true** if the request command and the responses to it should be sent secured. If a pending message was sent to a server securely, it will be always retrieved securely. This option only affects the request command and the pending messages that were sent without security to the server.

Returns

- An EmberAfMailboxStatus value of:
  - EMBER_MAILBOX_STATUS_SUCCESS if the query command was successfully passed to the network layer to be transmitted to the mailbox server.
  - EMBER_MAILBOX_STATUS_INVALID_ADDRESS if the passed mailbox server short ID is an invalid address.
  - EMBER_MAILBOX_STATUS_BUSY if the client is still performing a submit message or a query for message action.
  - EMBER_MAILBOX_STATUS_STACK_ERROR if the network layer refused the command (the outgoing queue is currently full).

Note

- Receiving the emberAfPluginMailboxClientCheckInboxCallback() requires the reception of a mailbox command message, which is only possible by polling if the message was submitted on a EMBER_STAR_SLEEPY_END_DEVICE.

Definition at line `128` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/mailbox/mailbox-client/mailbox-client.h`

Mailbox Server Plugin

Mailbox Server Plugin

APIs for mailbox server.

Mailbox protocol is designed for devices that can't be online on the network all the time. The most common example for this is a sleepy end device.

Mailbox clients and the server can submit messages into the mailbox, which is stored in RAM on the mailbox server. Clients can then query the mailbox server for available messages.

The mailbox server will notify clients who submit messages when a message was delivered or when it couldn't be delivered due to an error.

Mailbox uses a plugin-configurable protocol endpoint, which is 15 by default.

The server can also configure the size of the mailbox (in number of packets, 25 by default) and the packet timeout, after which the server drops the message and notifies the source of the error.

The mailbox protocol uses standard data messages, so in case of sleepy end devices, it will use the indirect queue. This means that if a sleepy end device sends a request to a mailbox server, the end device should poll for the response.

Note

- Mailbox is not available in MAC mode due to the lack of endpoints.

See mailbox-server.h and mailbox-server.c for source code.

Callbacks

void	emberAfPluginMailboxServerMessageDeliveredCallback(EmberAfMailboxStatus status, EmberNodeId messageDestination, uint8_t tag) Mailbox Server Message Delivered Callback.

Functions

EmberAfMailboxStatus	emberAfPluginMailboxServerAddMessage(EmberNodeId destination, uint8_t *message, EmberMessageLength messageLength, uint8_t tag, bool useSecurity) Add a message to the mailbox server queue. The message is stored in the internal queue until the destination node queries the mailbox server node for messages or upon timeout.

Callbacks Documentation

emberAfPluginMailboxServerMessageDeliveredCallback

```
void emberAfPluginMailboxServerMessageDeliveredCallback (EmberAfMailboxStatus status, EmberNodeId
messageDestination, uint8_t tag)
```

Mailbox Server Message Delivered Callback.

Parameters

[in]	status	An EmberAfMailboxStatus value of: • EMBER_MAILBOX_STATUS_SUCCESS indicates that the message was successfully delivered to the final destination. • EMBER_MAILBOX_STATUS_MESSAGE_TIMED_OUT indicates that the message timed-out and was removed from the server queue.
[in]	messageDestination	The node ID of the destination.
[in]	tag	The tag value passed in the emberAfPluginMailboxServerAddMessage() API.

This callback is invoked at the server when a message submitted locally by the server was successfully delivered or when it timed-out.

See Also

- emberAfPluginMailboxClientMessageDeliveredCallback()

Definition at line `110` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/mailbox/mailbox-server/mailbox-server.h`

Function Documentation

emberAfPluginMailboxServerAddMessage

```
EmberAfMailboxStatus emberAfPluginMailboxServerAddMessage (EmberNodeId destination, uint8_t *message,
EmberMessageLength messageLength, uint8_t tag, bool useSecurity)
```

Add a message to the mailbox server queue. The message is stored in the internal queue until the destination node queries the mailbox server node for messages or upon timeout.

Parameters

[in]	destination	The node ID of the destination for this data message.
[in]	message	A pointer to the message to be enqueued.
[in]	messageLength	The length in bytes of the message to be enqueued.
[in]	tag	A tag value which will be returned in the corresponding emberAfPluginMailboxServerMessageDeliveredCallback() callback. The application can use to match the callbacks with the call.
[in]	useSecurity	Set it **true** if the data message should be sent to the server using security.

Returns

- An EmberAfMailboxStatus value of:
  - EMBER_MAILBOX_STATUS_SUCCESS if the message was successfully added to the packet queue.
  - EMBER_MAILBOX_STATUS_INVALID_CALL if the passed message is invalid.
  - EMBER_MAILBOX_STATUS_INVALID_ADDRESS if the passed destination address is invalid.
  - EMBER_MAILBOX_STATUS_MESSAGE_TOO_LONG if the payload size of the passed message exceeds the maximum allowable payload for the passed transmission options.
  - EMBER_MAILBOX_STATUS_MESSAGE_TABLE_FULL if the packet table is already full.
  - EMBER_MAILBOX_STATUS_MESSAGE_NO_BUFFERS if not enough memory buffers are available for storing the message content.

See Also

- emberAfPluginMailboxClientMessageSubmit()

Definition at line `79` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/mailbox/mailbox-server/mailbox-server.h`

Mailbox Common

Mailbox Common

Types defined for mailbox.

Mailbox protocol is designed for devices that can't be online on the network all the time. The most common example for this is a sleepy end device.

Mailbox clients and the server can submit messages into the mailbox, which is stored in RAM on the mailbox server. Clients can then query the mailbox server for available messages.

The mailbox server will notify clients who submit messages when a message was delivered or when it couldn't be delivered due to an error.

Mailbox uses a plugin-configurable protocol endpoint, which is 15 by default.

The server can also configure the size of the mailbox (in number of packets, 25 by default) and the packet timeout, after which the server drops the message and notifies the source of the error.

The mailbox protocol uses standard data messages, so in case of sleepy end devices, it will use the indirect queue. This means that if a sleepy end device sends a request to a mailbox server, the end device should poll for the response.

Note

- Mailbox is not available in MAC mode due to the lack of endpoints.

Enumerations

```
enum    EmberAfMailboxStatus {
          EMBER_MAILBOX_STATUS_SUCCESS = 0×00
          EMBER_MAILBOX_STATUS_INVALID_CALL = 0×01
          EMBER_MAILBOX_STATUS_BUSY = 0×02
          EMBER_MAILBOX_STATUS_STACK_ERROR = 0×03
          EMBER_MAILBOX_STATUS_INVALID_ADDRESS = 0×04
          EMBER_MAILBOX_STATUS_MESSAGE_TOO_LONG = 0×05
          EMBER_MAILBOX_STATUS_MESSAGE_TABLE_FULL = 0×06
          EMBER_MAILBOX_STATUS_MESSAGE_NO_BUFFERS = 0×07
          EMBER_MAILBOX_STATUS_MESSAGE_NO_RESPONSE = 0×08
          EMBER_MAILBOX_STATUS_MESSAGE_TIMED_OUT = 0×09
          EMBER_MAILBOX_STATUS_MESSAGE_NO_DATA = 0×0A
        }
        Mailbox return status codes.
```

Enumeration Documentation

EmberAfMailboxStatus

EmberAfMailboxStatus

Mailbox return status codes.

Enumerator	
EMBER_MAILBOX_STATUS_SUCCESS	

EMBER_MAILBOX_STATUS_INVALID_CALL	
EMBER_MAILBOX_STATUS_BUSY	
EMBER_MAILBOX_STATUS_STACK_ERROR	
EMBER_MAILBOX_STATUS_INVALID_ADDRESS	
EMBER_MAILBOX_STATUS_MESSAGE_TOO_LONG	
EMBER_MAILBOX_STATUS_MESSAGE_TABLE_FULL	
EMBER_MAILBOX_STATUS_MESSAGE_NO_BUFFERS	
EMBER_MAILBOX_STATUS_MESSAGE_NO_RESPONSE	
EMBER_MAILBOX_STATUS_MESSAGE_TIMED_OUT	
EMBER_MAILBOX_STATUS_MESSAGE_NO_DATA	

Definition at line `68` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/mailbox/mailbox-types.h`

Ota Unicast Bootloader Client Plugin

Ota Unicast Bootloader Client Plugin

APIs/callbacks for ota-unicast-bootloader clients.

OTA bootloading plugins are usable to send firmware images Over The Air when the application is running. When the firmware is downloaded to a device, a bootloader can be started to replace the application in the flash to the one just downloaded.

All Connect bootloader related code relies on the Gecko Bootloader for bootloading and it must be installed on the device for these plugins to work. For details on the Gecko Bootloader, see UG266.

The Unicast OTA plugins implement the OTA download operation in a unicast, addressed way, so only a single client can be addressed from a server in an OTA session, and downloading images to multiple devices will require the server to send the image multiple times. Communication relies on standard unicast data messages, which also means that the routing provided by the Connect stack is availble.

Although bootloading sleepy end devices is theoretically possible with polling, it is not very effective, and it's probably simpler to reconnect as a normal end device while the OTA is active.

Unicast OTA uses a plugin configurable endpoint, which is 13 by default.

Security can be also enabled as plugin configuration on the server, as well as the interval of the messages. The client has a timeout plugin configuration after which it stops the OTA session with an error.

See UG235.06 for further details.

Note

- OTA Unicast Bootloading plugins are not available in MAC mode due to the lack of endpoints.

See ota-unicast-bootloader-client.h and ota-unicast-bootloader-client.c for source code.

Callbacks

bool	emberAfPluginOtaUnicastBootloaderClientNewIncomingImageCallback(EmberNodeId serverId, uint8_t imageTag, uint32_t imageSize, uint32_t *startIndex) A callback invoked when the client starts receiving a new image. The application can choose to start receiving the image or ignore it. If the application chooses to receive the image, other images sent out by other servers shall be ignored until the client completes the download.
void	emberAfPluginOtaUnicastBootloaderClientIncomingImageSegmentCallback(EmberNodeId serverId, uint32_t startIndex, uint32_t endIndex, uint8_t imageTag, uint8_t *imageSegment) A callback invoked when an image segment that is part of an image the application chose to download was received.
void	emberAfPluginOtaUnicastBootloaderClientImageDownloadCompleteCallback(EmberAfOtaUnicastBootloaderStatus status, uint8_t imageTag, uint32_t imageSize) A callback invoked to indicate that an image download has completed.
bool	emberAfPluginOtaUnicastBootloaderClientIncomingRequestBootloadCallback(EmberNodeId serverId, uint8_t imageTag, uint32_t bootloadDelayMs) A callback invoked to indicate that a server has requested to perform a bootload operation at a certain point in time in the future.

Functions

EmberAfOtaUnicastBootloaderStatus	emberAfPluginOtaUnicastBootloaderClientAbortImageDownload(uint8_t imageTag) An API for aborting an ongoing image download process.

Callbacks Documentation

emberAfPluginOtaUnicastBootloaderClientNewIncomingImageCallback

```
bool emberAfPluginOtaUnicastBootloaderClientNewIncomingImageCallback (EmberNodeId serverId, uint8_t imageTag,
uint32_t imageSize, uint32_t *startIndex)
```

A callback invoked when the client starts receiving a new image. The application can choose to start receiving the image or ignore it. If the application chooses to receive the image, other images sent out by other servers shall be ignored until the client completes the download.

Parameters

[in]	serverId	The node ID of the server that initiated the new image distribution process.
[in]	imageTag	A 1-byte tag that identifies the incoming image.
[in]	imageSize	The size in bytes of the new image.
[out]	startIndex	The index of the first byte at which the image download shall be started/resumed. The client can use this argument to resume a partially downloaded image. If this value is not set, it defaults to 0 (that is, the download starts at the beginning of the image). Note, this is ignored in case the server does not support download resume.

Returns

- Return **true** to accept the image or **false** to ignore it.

Definition at line `88` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-unicast-bootloader/ota-unicast-bootloader-client/ota-unicast-bootloader-client.h`

emberAfPluginOtaUnicastBootloaderClientIncomingImageSegmentCallback

```
void emberAfPluginOtaUnicastBootloaderClientIncomingImageSegmentCallback (EmberNodeId serverId, uint32_t
startIndex, uint32_t endIndex, uint8_t imageTag, uint8_t *imageSegment)
```

A callback invoked when an image segment that is part of an image the application chose to download was received.

Parameters

[in]	serverId	The node ID of the server that initiated the image distribution process.
[in]	startIndex	The index of the first byte of the passed segment.
[in]	endIndex	The index of the last byte of the passed segment.
[in]	imageTag	A 1-byte tag of the image the passed segment belongs to.
[in]	imageSegment	An array containing the image segment.

Definition at line `110` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-unicast-bootloader/ota-unicast-bootloader-client/ota-unicast-bootloader-client.h`

emberAfPluginOtaUnicastBootloaderClientImageDownloadCompleteCallback

```
void emberAfPluginOtaUnicastBootloaderClientImageDownloadCompleteCallback (EmberAfOtaUnicastBootloaderStatus
status, uint8_t imageTag, uint32_t imageSize)
```

A callback invoked to indicate that an image download has completed.

Parameters

[in]	status	An EmberAfOtaUnicastBootloaderStatus value of: • EMBER_OTA_UNICAST_BOOTLOADER_STATUS_SUCCESS indicating that the full image corresponding to the passed tag has been received. If this is the case, the client previously handed all the image segments to the application using the emberAfPluginOtaUnicastBootloaderClientIncomingImageSegmentCallback() callback. • EMBER_OTA_UNICAST_BOOTLOADER_STATUS_FAILED indicating that the client failed to fully download the image and the download process was terminated. • EMBER_OTA_UNICAST_BOOTLOADER_STATUS_TIMEOUT indicating that the client timed out waiting for a message from the server. • EMBER_OTA_UNICAST_BOOTLOADER_STATUS_ABORTED indicating that the application aborted the ongoing image download process as result of calling the API emberAfPluginOtaUnicastBootloaderClientAbortImageDownload().
[in]	imageTag	A 1-byte tag of the image this callback refers to.
[in]	imageSize	The total size of the downloaded image in bytes. This parameter is meaningful only in case the status parameter is set to EMBER_OTA_UNICAST_BOOTLOADER_STATUS_SUCCESS.

Definition at line `140` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-unicast-bootloader/ota-unicast-bootloader-client/ota-unicast-bootloader-client.h`

emberAfPluginOtaUnicastBootloaderClientIncomingRequestBootloadCallback

```
bool emberAfPluginOtaUnicastBootloaderClientIncomingRequestBootloadCallback (EmberNodeId serverId, uint8_t
imageTag, uint32_t bootloadDelayMs)
```

A callback invoked to indicate that a server has requested to perform a bootload operation at a certain point in time in the future.

Parameters

[in]	serverId	The ID of the server the request came from.
[in]	imageTag	A 1-byte tag of the image this callback refers to.
[in]	bootloadDelayMs	The delay in milliseconds after which the client has been requested to perform a bootload operation.

Returns

- Return **true** if the application accepted the request of bootloading the specified image at the requested time, **false** otherwise.

Definition at line `159` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-unicast-bootloader/ota-unicast-bootloader-client/ota-unicast-bootloader-client.h`

Function Documentation

emberAfPluginOtaUnicastBootloaderClientAbortImageDownload

```
EmberAfOtaUnicastBootloaderStatus emberAfPluginOtaUnicastBootloaderClientAbortImageDownload (uint8_t imageTag)
```

An API for aborting an ongoing image download process.

Parameters

[in]	imageTag	A 1-byte tag that identifies the image the client should no longer download.

Returns

- An EmberAfOtaUnicastBootloaderStatus value of:
  - EMBER_OTA_UNICAST_BOOTLOADER_STATUS_SUCCESS If the ongoing image download process was successfully aborted.
  - EMBER_OTA_UNICAST_BOOTLOADER_STATUS_INVALID_CALL If the client was not currently involved in an image download process or it was currently downloading an image with a different tag.

Definition at line `59` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-unicast-bootloader/ota-unicast-bootloader-client/ota-unicast-bootloader-client.h`

Ota Unicast Bootloader Server Plugin

Ota Unicast Bootloader Server Plugin

Macros and APIs for ota-unicast-bootloader server.

OTA bootloading plugins are usable to send firmware images Over The Air when the application is running. When the firmware is downloaded to a device, a bootloader can be started to replace the application in the flash to the one just downloaded.

All Connect bootloader related code relies on the Gecko Bootloader for bootloading and it must be installed on the device for these plugins to work. For details on the Gecko Bootloader, see UG266.

The Unicast OTA plugins implement the OTA download operation in a unicast, addressed way, so only a single client can be addressed from a server in an OTA session, and downloading images to multiple devices will require the server to send the image multiple times. Communication relies on standard unicast data messages, which also means that the routing provided by the Connect stack is availble.

Although bootloading sleepy end devices is theoretically possible with polling, it is not very effective, and it's probably simpler to reconnect as a normal end device while the OTA is active.

Unicast OTA uses a plugin configurable endpoint, which is 13 by default.

Security can be also enabled as plugin configuration on the server, as well as the interval of the messages. The client has a timeout plugin configuration after which it stops the OTA session with an error.

See UG235.06 for further details.

Note

- OTA Unicast Bootloading plugins are not available in MAC mode due to the lack of endpoints.

See ota-unicast-bootloader-server.h and ota-unicast-bootloader-server.c for source code.

Callbacks

bool	emberAfPluginOtaUnicastBootloaderServerGetImageSegmentCallback(uint32_t startIndex, uint32_t endIndex, uint8_t imageTag, uint8_t *imageSegment) A callback invoked during an image distribution process to retrieve a contiguous segment of the image being distributed.
void	emberAfPluginOtaUnicastBootloaderServerImageDistributionCompleteCallback(EmberAfOtaUnicastBootloaderStatus status) A callback invoked when the image distribution process is terminated.
void	emberAfPluginOtaUnicastBootloaderServerRequestTargetBootloadCompleteCallback(EmberAfOtaUnicastBootloaderStatu status) A callback invoked when a bootload request process has completed.

Functions

EmberAfOtaUnicastBootloaderStatus	emberAfPluginOtaUnicastBootloaderServerInitiateImageDistribution(EmberNodeId targetId, uint32_t imageSize, uint8_t imageTag) Initiate the image distribution process.
EmberAfOtaUnicastBootloaderStat-	

us	emberAfPluginUnicastBootloaderServerInitiateRequestTargetBootload(uint32_t bootloadDelayMs, uint8_t imageTag, EmberNodeId targetId) Request a target device to initiate the bootload of a received image at some point in the future.
EmberAfOtaUnicastBootloaderStatus	emberAfPluginOtaUnicastBootloaderServerAbortCurrentProcess(void) Abort the ongoing process, such as image distribution or bootload request.

Macros

#define	EMBER_AF_PLUGIN_OTA_UNICAST_BOOTLOADER_SERVER_MAX_STACK_ERRORS 8 The number of consecutive stack message submission errors or stack related errors such as CSMA failures after which the plugin gives up.
#define	EMBER_AF_PLUGIN_OTA_UNICAST_BOOTLOADER_SERVER_MAX_UNICAST_ERRORS 4 The number of consecutive unicast attempts after which a target is declared unreachable. Legal values for this are in the [0,7] range.
#define	EMBER_AF_PLUGIN_OTA_UNICAST_BOOTLOADER_SERVER_RESPONSE_TIMEOUT_MS 250 The time in milliseconds after which the server gives up waiting for a response from a client.

Callbacks Documentation

emberAfPluginOtaUnicastBootloaderServerGetImageSegmentCallback

```
bool emberAfPluginOtaUnicastBootloaderServerGetImageSegmentCallback (uint32_t startIndex, uint32_t endIndex, uint8_t
imageTag, uint8_t *imageSegment)
```

A callback invoked during an image distribution process to retrieve a contiguous segment of the image being distributed.

Parameters

[in]	startIndex	The index of the first byte the application should copy into the passed array.
[in]	endIndex	The index of the last byte the application should copy into the passed array.
[in]	imageTag	A 1-byte tag of the image for which a segment is requested.
[out]	imageSegment	An array of (endIndex - startIndex + 1) length to which the application should copy the requested image segment.

Returns

- A boolean indicating whether the application successfully copied the requested bytes into the passed array. If the application returns **false**, the server will abort the ongoing distribution process.

Definition at line `159` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-unicast-bootloader/ota-unicast-bootloader-server/ota-unicast-bootloader-server.h`

emberAfPluginOtaUnicastBootloaderServerImageDistributionCompleteCallback

```
void emberAfPluginOtaUnicastBootloaderServerImageDistributionCompleteCallback (EmberAfOtaUnicastBootloaderStatus
status)
```

A callback invoked when the image distribution process is terminated.

Parameters

[in]	status	An EmberAfOtaUnicastBootloaderStatus value of: • EMBER_OTA_UNICAST_BOOTLOADER_STATUS_SUCCESS if the target confirms that the full image is received. • EMBER_OTA_UNICAST_BOOTLOADER_STATUS_DATA_UNDERFLOW if the application failed to supply the requested image segments. • EMBER_OTA_UNICAST_BOOTLOADER_STATUS_STACK_ERROR if the server encountered multiple consecutive transmission errors. The Server gives up the image distribution process if EMBER_AF_PLUGIN_OTA_UNICAST_BOOTLOADER_SERVER_MAX_STACK_ERRORS consecutive transmission errors are encountered. • EMBER_OTA_UNICAST_BOOTLOADER_STATUS_FAILED if the distribution process terminated prematurely because the target can't be reached. • EMBER_OTA_UNICAST_BOOTLOADER_STATUS_UNREACHABLE if the server can not establish communication with the target client. • EMBER_OTA_UNICAST_BOOTLOADER_STATUS_ABORTED if the application aborted the current image distribution process. • EMBER_OTA_UNICAST_BOOTLOADER_STATUS_REFUSED if the client refused the image.

Definition at line 188 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-unicast-bootloader/ota-unicast-bootloader-server/ota-unicast-bootloader-server.h

emberAfPluginOtaUnicastBootloaderServerRequestTargetBootloadCompleteCallback

```
void emberAfPluginOtaUnicastBootloaderServerRequestTargetBootloadCompleteCallback
(EmberAfOtaUnicastBootloaderStatus status)
```

A callback invoked when a bootload request process has completed.

Parameters

[in]	status	An EmberAfOtaUnicastBootloaderStatus value of: • EMBER_OTA_UNICAST_BOOTLOADER_STATUS_SUCCESS if the target has been requested to perform a bootload. • EMBER_OTA_UNICAST_BOOTLOADER_STATUS_STACK_ERROR if the server encountered multiple consecutive transmission errors. The Server gives up the bootload request process if EMBER_AF_PLUGIN_OTA_UNICAST_BOOTLOADER_SERVER_MAX_STACK_ERRORS consecutive transmission errors are encountered. • EMBER_OTA_UNICAST_BOOTLOADER_STATUS_ABORTED if the application aborted the current bootload request process. • EMBER_OTA_UNICAST_BOOTLOADER_STATUS_UNREACHABLE if the server can not establish communication with the target client.

Definition at line 207 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-unicast-bootloader/ota-unicast-bootloader-server/ota-unicast-bootloader-server.h

Function Documentation

emberAfPluginOtaUnicastBootloaderServerInitiateImageDistribution

```
EmberAfOtaUnicastBootloaderStatus emberAfPluginOtaUnicastBootloaderServerInitiateImageDistribution (EmberNodeId
targetId, uint32_t imageSize, uint8_t imageTag)
```

Initiate the image distribution process.

Parameters

[in]	targetId	The node ID of the target.
[in]	imageSize	The image size in bytes to be distributed.
[in]	imageTag	A 1-byte tag that will be embedded in the server-to-client over-the-air messages. The application can use the image tag for versioning purposes and/or for distinguishing between different image types.

Returns

- An EmberAfOtaUnicastBootloaderStatus value of:
  - EMBER_OTA_UNICAST_BOOTLOADER_STATUS_BUSY if an image distribution is already in progress
  - EMBER_OTA_UNICAST_BOOTLOADER_STATUS_INVALID_CALL if the given target or the image size is invalid
  - EMBER_OTA_UNICAST_BOOTLOADER_STATUS_SUCCESS if the image distribution was successfully initiated.

Definition at line `87` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-unicast-bootloader/ota-unicast-bootloader-server/ota-unicast-bootloader-server.h`

emberAfPluginUnicastBootloaderServerInitiateRequestTargetBootload

```
EmberAfOtaUnicastBootloaderStatus emberAfPluginUnicastBootloaderServerInitiateRequestTargetBootload (uint32_t
bootloadDelayMs, uint8_t imageTag, EmberNodeId targetId)
```

Request a target device to initiate the bootload of a received image at some point in the future.

Parameters

[in]	bootloadDelayMs	The delay in milliseconds after which the target should perform an image bootload.
[in]	imageTag	A 1-byte tag that identifies the image to be bootloaded at the target device.
[in]	targetId	The node ID of the target.

Returns

- An EmberAfOtaUnicastBootloaderStatus value of:
  - EMBER_OTA_UNICAST_BOOTLOADER_STATUS_SUCCESS if the plugin successfully started the process to request a target and initiate a bootload. If this is the case, the corresponding callback emberAfPluginOtaUnicastBootloaderServerRequestTargetBootloadCompleteCallback() is invoked when the request process is completed.
  - EMBER_OTA_UNICAST_BOOTLOADER_STATUS_INVALID_CALL if some of the passed parameters are invalid.
  - EMBER_OTA_UNICAST_BOOTLOADER_STATUS_BUSY if the server is currently involved in another over-the-air process.

Definition at line `116` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-unicast-bootloader/ota-unicast-bootloader-server/ota-unicast-bootloader-server.h`

emberAfPluginOtaUnicastBootloaderServerAbortCurrentProcess

```
EmberAfOtaUnicastBootloaderStatus emberAfPluginOtaUnicastBootloaderServerAbortCurrentProcess (void)
```

Abort the ongoing process, such as image distribution or bootload request.

Parameters

N/A		

Returns

- An EmberAfOtaUnicastBootloaderStatus value of:
  - EMBER_OTA_UNICAST_BOOTLOADER_STATUS_SUCCESS if the current ongoing process was successfully aborted.
  - EMBER_OTA_UNICAST_BOOTLOADER_STATUS_INVALID_CALL if the server is not currently involved in any process.

Definition at line `131` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-unicast-bootloader/ota-unicast-bootloader-server/ota-unicast-bootloader-server.h`

Macro Definition Documentation

EMBER_AF_PLUGIN_OTA_UNICAST_BOOTLOADER_SERVER_MAX_STACK_ERRORS

```
#define EMBER_AF_PLUGIN_OTA_UNICAST_BOOTLOADER_SERVER_MAX_STACK_ERRORS
```

Value:

```
8
```

The number of consecutive stack message submission errors or stack related errors such as CSMA failures after which the plugin gives up.

Definition at line `51` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-unicast-bootloader/ota-unicast-bootloader-server/ota-unicast-bootloader-server.h`

EMBER_AF_PLUGIN_OTA_UNICAST_BOOTLOADER_SERVER_MAX_UNICAST_ERRORS

```
#define EMBER_AF_PLUGIN_OTA_UNICAST_BOOTLOADER_SERVER_MAX_UNICAST_ERRORS
```

Value:

```
4
```

The number of consecutive unicast attempts after which a target is declared unreachable. Legal values for this are in the [0,7] range.

Definition at line `57` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-unicast-bootloader/ota-unicast-bootloader-server/ota-unicast-bootloader-server.h`

EMBER_AF_PLUGIN_OTA_UNICAST_BOOTLOADER_SERVER_RESPONSE_TIMEOUT_MS

```
#define EMBER_AF_PLUGIN_OTA_UNICAST_BOOTLOADER_SERVER_RESPONSE_TIMEOUT_MS
```

Value:

```
250
```

The time in milliseconds after which the server gives up waiting for a response from a client.

Definition at line `63` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-unicast-bootloader/ota-unicast-bootloader-server/ota-unicast-bootloader-server.h`

Ota Unicast Bootloader Common

Ota Unicast Bootloader Common

Macros and types defined for ota-unicast-bootloaders.

OTA bootloading plugins are usable to send firmware images Over The Air when the application is running. When the firmware is downloaded to a device, a bootloader can be started to replace the application in the flash to the one just downloaded.

All Connect bootloader related code relies on the Gecko Bootloader for bootloading and it must be installed on the device for these plugins to work. For details on the Gecko Bootloader, see UG266.

The Unicast OTA plugins implement the OTA download operation in a unicast, addressed way, so only a single client can be addressed from a server in an OTA session, and downloading images to multiple devices will require the server to send the image multiple times. Communication relies on standard unicast data messages, which also means that the routing provided by the Connect stack is availble.

Although bootloading sleepy end devices is theoretically possible with polling, it is not very effective, and it's probably simpler to reconnect as a normal end device while the OTA is active.

Unicast OTA uses a plugin configurable endpoint, which is 13 by default.

Security can be also enabled as plugin configuration on the server, as well as the interval of the messages. The client has a timeout plugin configuration after which it stops the OTA session with an error.

See UG235.06 for further details.

Note

- OTA Unicast Bootloading plugins are not available in MAC mode due to the lack of endpoints.

Enumerations

```
enum    EmberAfOtaUnicastBootloaderStatus {
          EMBER_OTA_UNICAST_BOOTLOADER_STATUS_SUCCESS = 0x00
          EMBER_OTA_UNICAST_BOOTLOADER_STATUS_INVALID_CALL = 0x01
          EMBER_OTA_UNICAST_BOOTLOADER_STATUS_BUSY = 0x02
          EMBER_OTA_UNICAST_BOOTLOADER_STATUS_DATA_UNDERFLOW = 0x03
          EMBER_OTA_UNICAST_BOOTLOADER_STATUS_STACK_ERROR = 0x04
          EMBER_OTA_UNICAST_BOOTLOADER_STATUS_TIMEOUT = 0x05
          EMBER_OTA_UNICAST_BOOTLOADER_STATUS_FAILED = 0x06
          EMBER_OTA_UNICAST_BOOTLOADER_STATUS_ABORTED = 0x07
          EMBER_OTA_UNICAST_BOOTLOADER_STATUS_REFUSED = 0x08
          EMBER_OTA_UNICAST_BOOTLOADER_STATUS_UNREACHABLE = 0x09
        }
```

OTA Unicast Bootloader return status codes.

Enumeration Documentation

EmberAfOtaUnicastBootloaderStatus

```
EmberAfOtaUnicastBootloaderStatus
```

OTA Unicast Bootloader return status codes.

Enumerator	
EMBER_OTA_UNICAST_BOOTLOADER_STATUS_SUCCESS	
EMBER_OTA_UNICAST_BOOTLOADER_STATUS_INVALID_CALL	
EMBER_OTA_UNICAST_BOOTLOADER_STATUS_BUSY	
EMBER_OTA_UNICAST_BOOTLOADER_STATUS_DATA_UNDERFLOW	
EMBER_OTA_UNICAST_BOOTLOADER_STATUS_STACK_ERROR	
EMBER_OTA_UNICAST_BOOTLOADER_STATUS_TIMEOUT	
EMBER_OTA_UNICAST_BOOTLOADER_STATUS_FAILED	
EMBER_OTA_UNICAST_BOOTLOADER_STATUS_ABORTED	
EMBER_OTA_UNICAST_BOOTLOADER_STATUS_REFUSED	
EMBER_OTA_UNICAST_BOOTLOADER_STATUS_UNREACHABLE	

Definition at line `75` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-unicast-bootloader/ota-unicast-bootloader-types.h`

Ota Broadcast Bootloader Client Plugin

Ota Broadcast Bootloader Client Plugin

Set of APIs for ota-broadcast-bootloader-client.

OTA bootloading plugins are usable to send firmware images Over The Air when the application is running. When the firmware is downloaded to a device, a bootloader can be started to replace the application in the flash to the one just downloaded.

All Connect bootloader-related code relies on the Gecko Bootloader for bootloading and it must be installed on the device for these plugins to work. For details on the Gecko Bootloader, see UG266.

The Broadcast OTA plugins implement the OTA download operation in broadcast, so the same image can be sent to many devices at the same time. The server however requires to know the clients downloading the image, because it implements error handling by querying all clients for missing segments, and then the server will re-broadcast those segments.

Communication relies on standard broadcast data messages, which means routing is not available, and only clients that are in the range of the server can download. However, the same device can be both client and server, i.e. after downloading the image, a client can configure itself to be a server, and provide the image to another part of the network.

Sleepy end devices cannot be addressed in broadcast, but a sleepy end device can reconnect as a normal end device while the OTA is active.

Broadcast OTA uses a plugin configurable endpoint, which is 14 by default.

Security can be also enabled as plugin configuration on the server, as well as the interval of the messages. The client has a timeout plugin configuration after which it stops the OTA session with an error.

See UG235.06 for further details.

Note

- OTA Broadcast Bootloading plugins are not available in MAC mode due to the lack of endpoints.

See ota-broadcast-bootloader-client.h and ota-broadcast-bootloader-client.c for source code.

Callbacks

bool	emberAfPluginOtaBootloaderClientNewIncomingImageCallback(EmberNodeId serverId, EmberNodeId *alternateServerId, uint8_t imageTag) A callback invoked when the OTA Bootloader Client starts receiving a new image. The application can choose to start receiving the image or it can ignore it. If the application chooses to receive the image, other images sent out by other servers are ignored until the client completes this download.
void	emberAfPluginOtaBootloaderClientIncomingImageSegmentCallback(EmberNodeId serverId, uint32_t startIndex, uint32_t endIndex, uint8_t imageTag, uint8_t *imageSegment) A callback invoked when an image segment, that is part of an image that the application elected to download, was received on the OTA Bootloader Client.
void	emberAfPluginOtaBootloaderClientImageDownloadCompleteCallback(EmberAfOtaBootloaderStatus status, uint8_t imageTag, uint32_t imageSize) A callback invoked on an OTA Bootloader Client to indicate that an image downlaod is completed.

void	emberAfPluginOtaBootloaderClientIncomingRequestStatusCallback(EmberNodeId serverId, uint8_t applicationServerStatus, uint8_t *applicationStatus) A callback invoked on the OTA Bootloader Client to indicate that an OTA Bootloader Server has requested the status of the client device.
bool	emberAfPluginOtaBootloaderClientIncomingRequestBootloadCallback(EmberNodeId serverId, uint8_t imageTag, uint32_t bootloadDelayMs, uint8_t *applicationStatus) A callback invoked by the OTA Bootloader Client plugin to indicate that an OTA Bootloader Server has requested to perform a bootload operation at a certain point in time in the future.

Functions

EmberAfOtaBootloaderStatus	emberAfPluginOtaBootloaderClientAbortImageDownload(uint8_t imageTag, uint8_t applicationErrorStatus) Abort an ongoing image download process.

Callbacks Documentation

emberAfPluginOtaBootloaderClientNewIncomingImageCallback

```
bool emberAfPluginOtaBootloaderClientNewIncomingImageCallback (EmberNodeId serverId, EmberNodeId
*alternateServerId, uint8_t imageTag)
```

A callback invoked when the OTA Bootloader Client starts receiving a new image. The application can choose to start receiving the image or it can ignore it. If the application chooses to receive the image, other images sent out by other servers are ignored until the client completes this download.

Parameters

[in]	serverId	The node ID of the server that initiated the new image distribution process.
[out]	alternateServerId	This node ID can be set by the application to include a well-known alternate server. If this is set to a valid address, the client allows segments also from this alternate server. If this is set to EMBER_BROADCAST_ADDRESS, the client accepts segments with the same image tag from any server.
[in]	imageTag	A 1-byte tag that identifies the incoming image.

Returns

- Return **true** to accept the image or **false** to ignore it.

Definition at line `94` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-broadcast-bootloader/ota-broadcast-bootloader-client/ota-broadcast-bootloader-client.h`

emberAfPluginOtaBootloaderClientIncomingImageSegmentCallback

```
void emberAfPluginOtaBootloaderClientIncomingImageSegmentCallback (EmberNodeId serverId, uint32_t startIndex,
uint32_t endIndex, uint8_t imageTag, uint8_t *imageSegment)
```

A callback invoked when an image segment, that is part of an image that the application elected to download, was received on the OTA Bootloader Client.

Parameters

[in]	serverId	The node ID of the server that initiated the image distribution process.
[in]	startIndex	The index of the first byte of the passed segment.
[in]	endIndex	The index of the last byte of the passed segment.
[in]	imageTag	A 1-byte tag of the image the passed segment belongs to.

[in]	imageSegment	An array containing the image segment.

Definition at line 114 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-broadcast-bootloader/ota-broadcast-bootloader-client/ota-broadcast-bootloader-client.h

emberAfPluginOtaBootloaderClientImageDownloadCompleteCallback

> void emberAfPluginOtaBootloaderClientImageDownloadCompleteCallback (EmberAfOtaBootloaderStatus status, uint8_t imageTag, uint32_t imageSize)

A callback invoked on an OTA Bootloader Client to indicate that an image downlaod is completed.

Parameters

[in]	status	An EmberAfOtaBootloaderStatus value of: • EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_SUCCESS indicating that the full image corresponding to the passed tag has been received. If this is the case, the client previously handed all the image segments to the application using the emberAfPluginOtaBootloaderClientIncomingImageSegmentCallback() callback. • EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_FAILED indicating that the client failed to fully download the image and the download process was terminated. • EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_TIMEOUT indicating that the client timed out waiting for a message from the server. • EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_ABORTED indicating that the application aborted the ongoing image download process as result of calling the API emberAfPluginOtaBootloaderClientAbortImageDownload().
[in]	imageTag	A 1-byte tag of the image this callback refers to.
[in]	imageSize	The total size of the downloaded image in bytes. This parameter is meaningful only in case the status parameter is set to EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_SUCCESS.

Definition at line 143 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-broadcast-bootloader/ota-broadcast-bootloader-client/ota-broadcast-bootloader-client.h

emberAfPluginOtaBootloaderClientIncomingRequestStatusCallback

> void emberAfPluginOtaBootloaderClientIncomingRequestStatusCallback (EmberNodeId serverId, uint8_t applicationServerStatus, uint8_t *applicationStatus)

A callback invoked on the OTA Bootloader Client to indicate that an OTA Bootloader Server has requested the status of the client device.

Parameters

[in]	serverId	The ID of the server the request came from.
[in]	applicationServerStatus	The server application status, which was set by emberAfPluginBootloaderServerInitiateRequestTargetsStatus()
[out]	applicationStatus	A 1-byte status set by the client application that is reported to the server.

Definition at line 158 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-broadcast-bootloader/ota-broadcast-bootloader-client/ota-broadcast-bootloader-client.h

emberAfPluginOtaBootloaderClientIncomingRequestBootloadCallback

```
bool emberAfPluginOtaBootloaderClientIncomingRequestBootloadCallback (EmberNodeId serverId, uint8_t imageTag,
uint32_t bootloadDelayMs, uint8_t *applicationStatus)
```

A callback invoked by the OTA Bootloader Client plugin to indicate that an OTA Bootloader Server has requested to perform a bootload operation at a certain point in time in the future.

Parameters

[in]	serverId	The ID of the server the request came from.
[in]	imageTag	A 1-byte tag of the image this callback refers to.
[in]	bootloadDelayMs	The delay in milliseconds after which the client has been requested to perform a bootload operation.
[out]	applicationStatus	A 1-byte status set by the client application that is reported to the server.

Returns

- Return **true** if the application accepted the request of bootloading the specified image at the requested time, **false** otherwise.

Definition at line `179` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-broadcast-bootloader/ota-broadcast-bootloader-client/ota-broadcast-bootloader-client.h`

Function Documentation

emberAfPluginOtaBootloaderClientAbortImageDownload

```
EmberAfOtaBootloaderStatus emberAfPluginOtaBootloaderClientAbortImageDownload (uint8_t imageTag, uint8_t
applicationErrorStatus)
```

Abort an ongoing image download process.

Parameters

[in]	imageTag	A 1-byte tag that identifies the image the client should no longer download.
[in]	applicationErrorStatus	A 1-byte error code reported to the server.

Returns

- An EmberAfOtaBootloaderStatus value of:
  - EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_SUCCESS If the ongoing image download process was successfully aborted.
  - EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_INVALID_CALL If the client was not currently involved in an image download process or it was currently downloading an image with a different tag.

Definition at line `62` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-broadcast-bootloader/ota-broadcast-bootloader-client/ota-broadcast-bootloader-client.h`

Ota Broadcast Bootloader Server Plugin

Ota Broadcast Bootloader Server Plugin

Set of APIs for ota-broadcast-bootloader-server.

OTA bootloading plugins are usable to send firmware images Over The Air when the application is running. When the firmware is downloaded to a device, a bootloader can be started to replace the application in the flash to the one just downloaded.

All Connect bootloader-related code relies on the Gecko Bootloader for bootloading and it must be installed on the device for these plugins to work. For details on the Gecko Bootloader, see UG266.

The Broadcast OTA plugins implement the OTA download operation in broadcast, so the same image can be sent to many devices at the same time. The server however requires to know the clients downloading the image, because it implements error handling by querying all clients for missing segments, and then the server will re-broadcast those segments.

Communication relies on standard broadcast data messages, which means routing is not available, and only clients that are in the range of the server can download. However, the same device can be both client and server, i.e. after downloading the image, a client can configure itself to be a server, and provide the image to another part of the network.

Sleepy end devices cannot be addressed in broadcast, but a sleepy end device can reconnect as a normal end device while the OTA is active.

Broadcast OTA uses a plugin configurable endpoint, which is 14 by default.

Security can be also enabled as plugin configuration on the server, as well as the interval of the messages. The client has a timeout plugin configuration after which it stops the OTA session with an error.

See UG235.06 for further details.

Note

- OTA Broadcast Bootloading plugins are not available in MAC mode due to the lack of endpoints.

See ota-broadcast-bootloader-server.h and ota-broadcast-bootloader-server.c for source code.

Callbacks

bool	emberAfPluginOtaBootloaderServerGetImageSegmentCallback(uint32_t startIndex, uint32_t endIndex, uint8_t imageTag, uint8_t *imageSegment) A callback invoked on the OTA Bootloader Server during an image distribution process to retrieve a contiguous segment of the image being distributed.
void	emberAfPluginOtaBootloaderServerImageDistributionCompleteCallback(EmberAfOtaBootloaderStatus status) A callback invoked on the OTA Bootloader Server when the image distribution process is terminated. The application can use the emberAfPluginBootloaderServerGetTargetStatus() API to retrieve the status reported by each target device.
void	emberAfPluginBootloaderServerRequestTargetsStatusCompleteCallback(EmberAfOtaBootloaderStatus status) A callback invoked on the OTA Bootloader Server when bootload request process has completed. Within this callback, the application should use the emberAfPluginBootloaderServerGetTargetStatus() API to retrieve the status and application status reported by each target.

void	emberAfPluginBootloaderServerRequestTargetsBootloadCompleteCallback(EmberAfOtaBootloaderStatus status) A callback invoked on the OTA Bootloader Server when a bootload request process has completed. Within this callback, the application should use the emberAfPluginBootloaderServerGetTargetStatus() API to retrieve the status and the application status reported by each target.

Functions

EmberAfOtaBootloaderStatus	emberAfPluginOtaBootloaderServerInitiateImageDistribution(uint32_t imageSize, uint8_t imageTag, EmberNodeId *targetList, uint16_t targetListLength) Initiate the image distribution process. The application can use the emberAfPluginBootloaderServerGetTargetStatus() API at any time during the image distribution process to check the status of each target.
EmberAfOtaBootloaderStatus	emberAfPluginBootloaderServerInitiateRequestTargetsStatus(EmberNodeId *targetList, uint16_t targetListLength, uint8_t applicationServerStatus) Initiate the process to request the status of a set of target devices. The application can use the emberAfPluginBootloaderServerGetTargetStatus() API at any time during the target status request process to check the status of each target.
EmberAfOtaBootloaderStatus	emberAfPluginBootloaderServerInitiateRequestTargetsBootload(uint32_t bootloadDelayMs, uint8_t imageTag, EmberNodeId *targetList, uint16_t targetListLength) Start the process where a server requests a set of target devices to initiate the bootload of a received image at some point in the future. The application can use the emberAfPluginBootloaderServerGetTargetStatus() API at any time during the bootload request process to check the status of each target.
EmberAfOtaBootloaderTargetStatus	emberAfPluginBootloaderServerGetTargetStatus(EmberNodeId targetId, uint8_t *applicationTargetStatus) Retrieve the locally stored status of an individual target in the distribution list. The locally stored status can be updated by calling emberAfPluginBootloaderServerInitiateRequestTargetsStatus().
EmberAfOtaBootloaderStatus	emberAfPluginOtaBootloaderServerAbortCurrentProcess(void) Abort the ongoing process (image distribution, status request or bootload request). Note that aborting a bootload request process likely results in some targets performing the bootload while some others do not.

Macros

#define	EMBER_AF_PLUGIN_OTA_BROADCAST_BOOTLOADER_SERVER_MAX_STACK_ERRORS 8 The number of consecutive stack message submission errors or stack-related errors, such as CSMA failures, after which the plugin gives up.
#define	EMBER_AF_PLUGIN_OTA_BROADCAST_BOOTLOADER_SERVER_MAX_UNICAST_ERRORS 4 The number of consecutive unicast attempts after which a target is declared unreachable. Legal values for this are in the [0,7] range.
#define	EMBER_AF_PLUGIN_OTA_BROADCAST_BOOTLOADER_SERVER_RESPONSE_TIMEOUT_MS 250 The time in milliseconds after which the server gives up waiting for a response from a client.
#define	EMBER_AF_PLUGIN_OTA_BROADCAST_BOOTLOADER_SERVER_MAX_BROADCAST_ROUNDS 5 The maximum number of image broadcast rounds the server performs before declaring an image distribution process failed.

Callbacks Documentation

emberAfPluginOtaBootloaderServerGetImageSegmentCallback

```
bool emberAfPluginOtaBootloaderServerGetImageSegmentCallback (uint32_t startIndex, uint32_t endIndex, uint8_t
imageTag, uint8_t *imageSegment)
```

A callback invoked on the OTA Bootloader Server during an image distribution process to retrieve a contiguous segment of the image being distributed.

Parameters

[in]	startIndex	The index of the first byte the application should copy into the passed array.
[in]	endIndex	The index of the last byte the application should copy into the passed array.
[in]	imageTag	A 1-byte tag of the image for which a segment is being requested.
[out]	imageSegment	An array of (endIndex - startIndex + 1) length to which the application should copy the requested image segment.

Returns

- A boolean indicating whether the application successfully copied the requested bytes into the passed array. If the application returns **false**, the OTA Server plugin aborts the ongoing distribution process.

Definition at line `267` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-broadcast-bootloader/ota-broadcast-bootloader-server/ota-broadcast-bootloader-server.h`

emberAfPluginOtaBootloaderServerImageDistributionCompleteCallback

```
void emberAfPluginOtaBootloaderServerImageDistributionCompleteCallback (EmberAfOtaBootloaderStatus status)
```

A callback invoked on the OTA Bootloader Server when the image distribution process is terminated. The application can use the emberAfPluginBootloaderServerGetTargetStatus() API to retrieve the status reported by each target device.

Parameters

[in]	status	An EmberAfOtaBootloaderStatus value of: <ul><li>EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_SUCCESS if all targets have confirmed that the full image was received except for those that have been declared "unreachable".</li><li>EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_DATA_UNDERFLOW if the application failed to supply the requested image segments.</li><li>EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_STACK_ERROR if the server encountered multiple consecutive transmission errors. The Server gives up the image distribution process if EMBER_AF_PLUGIN_OTA_BROADCAST_BOOTLOADER_SERVER_MAX_STACK_ERRORS consecutive transmission errors are encountered.</li><li>EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_FAILED if the distribution process terminated prematurely because all targets have been declared unreachable.</li><li>EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_TIMEOUT if the server performed all the allowable broadcast rounds and there are still missing segments at one or more targets. The maximum allowable rounds are defined by EMBER_AF_PLUGIN_OTA_BROADCAST_BOOTLOADER_SERVER_MAX_BROADCAST_ROUNDS.</li><li>EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_ABORTED if the application aborted the current image distribution process.</li></ul>

Definition at line `298` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-broadcast-bootloader/ota-broadcast-bootloader-server/ota-broadcast-bootloader-server.h`

emberAfPluginBootloaderServerRequestTargetsStatusCompleteCallback

```
void emberAfPluginBootloaderServerRequestTargetsStatusCompleteCallback (EmberAfOtaBootloaderStatus status)
```

A callback invoked on the OTA Bootloader Server when bootload request process has completed. Within this callback, the application should use the emberAfPluginBootloaderServerGetTargetStatus() API to retrieve the status and application status reported by each target.

Parameters

[in]	status	An EmberAfOtaBootloaderStatus value of: • EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_SUCCESS if all the targets have been queried for their status. Notice that some targets might have been declared unreachable. • EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_STACK_ERROR if the server encountered multiple consecutive transmission errors. The Server gives up the targets status request process if EMBER_AF_PLUGIN_OTA_BROADCAST_BOOTLOADER_SERVER_MAX_STACK_ERRORS consecutive transmission errors are encountered. • EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_ABORTED if the application aborted the current targets status request process.

Definition at line `317` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-broadcast-bootloader/ota-broadcast-bootloader-server/ota-broadcast-bootloader-server.h`

emberAfPluginBootloaderServerRequestTargetsBootloadCompleteCallback

```
void emberAfPluginBootloaderServerRequestTargetsBootloadCompleteCallback (EmberAfOtaBootloaderStatus status)
```

A callback invoked on the OTA Bootloader Server when a bootload request process has completed. Within this callback, the application should use the emberAfPluginBootloaderServerGetTargetStatus() API to retrieve the status and the application status reported by each target.

Parameters

[in]	status	An EmberAfOtaBootloaderStatus value of: • EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_SUCCESS if all targets have been requested to perform a bootload. Notice that some targets might have been declared unreachable. • EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_STACK_ERROR if the server encountered multiple consecutive transmission errors. The Server gives up the bootload request process if EMBER_AF_PLUGIN_OTA_BROADCAST_BOOTLOADER_SERVER_MAX_STACK_ERRORS consecutive transmission errors are encountered. • EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_ABORTED if the application aborted the current bootload request process.

Definition at line `336` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-broadcast-bootloader/ota-broadcast-bootloader-server/ota-broadcast-bootloader-server.h`

Function Documentation

emberAfPluginOtaBootloaderServerInitiateImageDistribution

```
EmberAfOtaBootloaderStatus emberAfPluginOtaBootloaderServerInitiateImageDistribution (uint32_t imageSize, uint8_t
imageTag, EmberNodeId *targetList, uint16_t targetListLength)
```

Initiate the image distribution process. The application can use the emberAfPluginBootloaderServerGetTargetStatus() API at any time during the image distribution process to check the status of each target.

Parameters

[in]	imageSize	The image size in bytes to be distributed.
[in]	imageTag	A 1-byte tag that will be embedded in the server-to-client over-the-air messages. The application can use the image tag for versioning purposes and/or for distinguishing between different image types.

[out]	targetList	An array of EmberNodeId indicating the node IDs of the target devices.
[in]	targetListLength	The length of the passed `targetList`

Returns

- An EmberAfOtaBootloaderStatus value of:
  - EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_SUCCESS if the image distribution is successfully initiated. If this is the case, the emberAfPluginOtaBootloaderServerImageDistributionCompleteCallback() callback is invoked when the distribution process terminates.
  - EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_INVALID_CALL if some of the passed parameters are invalid.
  - EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_BUSY if the server is already performing another image distribution or some other over-the-air process.
  - EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_NO_BUFFERS if the server can't allocate memory from the heap to store the passed target list. (See Memory Buffer for details).

Definition at line `104` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-broadcast-bootloader/ota-broadcast-bootloader-server/ota-broadcast-bootloader-server.h`

emberAfPluginBootloaderServerInitiateRequestTargetsStatus

> EmberAfOtaBootloaderStatus emberAfPluginBootloaderServerInitiateRequestTargetsStatus (EmberNodeId *targetList, uint16_t targetListLength, uint8_t applicationServerStatus)

Initiate the process to request the status of a set of target devices. The application can use the emberAfPluginBootloaderServerGetTargetStatus() API at any time during the target status request process to check the status of each target.

Parameters

[in]	targetList	An array of EmberNodeId indicating the node IDs of the target devices that are queried for their status.
[in]	targetListLength	The length of the passed `targetlist` .
[in]	applicationServerStatus	The application can set a status here which will be sent to the clients in emberAfPluginOtaBootloaderClientIncomingRequestStatusCallback()

Returns

- An EmberAfOtaBootloaderStatus value of:
  - EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_SUCCESS if the plugin successfully initiated the process of requesting the status of a set of targets. If this is the case, the corresponding callback emberAfPluginBootloaderServerRequestTargetsStatusCompleteCallback() is invoked when the request process to all targets completes.
  - EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_INVALID_CALL if some of the passed parameters are invalid.
  - EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_BUSY if the server is currently involved in another over-the-air process.

Definition at line `135` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-broadcast-bootloader/ota-broadcast-bootloader-server/ota-broadcast-bootloader-server.h`

emberAfPluginBootloaderServerInitiateRequestTargetsBootload

> EmberAfOtaBootloaderStatus emberAfPluginBootloaderServerInitiateRequestTargetsBootload (uint32_t bootloadDelayMs, uint8_t imageTag, EmberNodeId *targetList, uint16_t targetListLength)

Start the process where a server requests a set of target devices to initiate the bootload of a received image at some point in the future. The application can use the emberAfPluginBootloaderServerGetTargetStatus() API at any time during the bootload request process to check the status of each target.

Parameters

[in]	bootloadDelayMs	The delay in milliseconds after which all the targets should perform an image bootload.
[in]	imageTag	A 1-byte tag that identifies the image to be bootloaded at the target devices.
[in]	targetList	An array of EmberNodeId indicating the node IDs of the target devices that is requested to bootload an image.
[in]	targetListLength	The length of the passed `targetlist` .

Returns

- An EmberAfOtaBootloaderStatus value of:
  - EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_SUCCESS if the plugin successfully initiated the process of requesting a set of targets to initiate a bootload. If this is the case, the corresponding callback emberAfPluginBootloaderServerRequestTargetsBootloadCompleteCallback() shall be invoked when the request process to all targets has completed.
  - EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_INVALID_CALL if some of the passed parameters are invalid.
  - EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_BUSY if the server is currently involved in another over-the-air process.

Definition at line `167` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-broadcast-bootloader/ota-broadcast-bootloader-server/ota-broadcast-bootloader-server.h`

emberAfPluginBootloaderServerGetTargetStatus

> EmberAfOtaBootloaderTargetStatus emberAfPluginBootloaderServerGetTargetStatus (EmberNodeId targetId, uint8_t *applicationTargetStatus)

Retrieve the locally stored status of an individual target in the distribution list. The locally stored status can be updated by calling emberAfPluginBootloaderServerInitiateRequestTargetsStatus().

Parameters

[in]	targetId	The node ID of the target device whose status is being requested.
[out]	applicationTargetStatus	The application status reported by the client side application. This parameter is valid only for certain return status codes (see return status documentation).

Returns

- An EmberAfOtaBootloaderTargetStatus value of:
  - EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_INVALID if the passed node ID does not appear in the current server target list of the current ongoing process or if there is no current ongoing process.
  - EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_UNREACHABLE if the target has not responded to any of the server's unicast messages.
  - EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_DISTRIBUTION_COMPLETED if the server is currently performing an image distribution process and the target confirmed that it received the full image.
  - EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_DISTRIBUTION_ONGOING if the server is currently performing an image distribution process and the target has partially received the image and distribution is continuing.
  - EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_DISTRIBUTION_REFUSED if the target has refused the current image.
  - EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_DISTRIBUTION_FAILED if the server is currently performing an image distribution process and the target reported that an error was encountered.
  - EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_DISTRIBUTION_ABORTED if the server is currently performing an image distribution process and the target decided to abort the image download process. In this case, the client also reports an application status. Therefore, the applicationTargetStatus parameter is valid.
  - EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_STATUS_REQUEST_COMPLETED if the server is currently performing a target status request process and the target has responded to the server's inquiry. In this case, the client also reports an application status. Therefore, the applicationTargetStatus parameter is valid.
  - EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_STATUS_REQUEST_ONGOING if the server is currently performing a target status request process and the target is not yet queried by the server.

- EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_BOOTLOAD_REQUEST_ACCEPTED if the server is currently performing a bootload request process and the target has accepted to perform the requested image bootload.
- EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_BOOTLOAD_REQUEST_REFUSED if the server is currently performing a bootload request process and the target has refused to perform the requested image bootload.
- EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_BOOTLOAD_REQUEST_ONGOING if the server is currently performing a bootload request process and the target is not yet reached by the server.

Definition at line `223` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-broadcast-bootloader/ota-broadcast-bootloader-server/ota-broadcast-bootloader-server.h`

emberAfPluginOtaBootloaderServerAbortCurrentProcess

```
EmberAfOtaBootloaderStatus emberAfPluginOtaBootloaderServerAbortCurrentProcess (void)
```

Abort the ongoing process (image distribution, status request or bootload request). Note that aborting a bootload request process likely results in some targets performing the bootload while some others do not.

Parameters

N/A		

Returns

- An EmberAfOtaBootloaderStatus value of:
  - EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_SUCCESS if the current ongoing process was successfully aborted.
  - EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_INVALID_CALL if the server is not currently involved in any process.

Definition at line `237` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-broadcast-bootloader/ota-broadcast-bootloader-server/ota-broadcast-bootloader-server.h`

Macro Definition Documentation

EMBER_AF_PLUGIN_OTA_BROADCAST_BOOTLOADER_SERVER_MAX_STACK_ERRORS

```
#define EMBER_AF_PLUGIN_OTA_BROADCAST_BOOTLOADER_SERVER_MAX_STACK_ERRORS
```

Value:

```
8
```

The number of consecutive stack message submission errors or stack-related errors, such as CSMA failures, after which the plugin gives up.

Definition at line `51` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-broadcast-bootloader/ota-broadcast-bootloader-server/ota-broadcast-bootloader-server.h`

EMBER_AF_PLUGIN_OTA_BROADCAST_BOOTLOADER_SERVER_MAX_UNICAST_ERRORS

```
#define EMBER_AF_PLUGIN_OTA_BROADCAST_BOOTLOADER_SERVER_MAX_UNICAST_ERRORS
```

Value:

```
4
```

The number of consecutive unicast attempts after which a target is declared unreachable. Legal values for this are in the [0,7] range.

Definition at line `57` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-broadcast-bootloader/ota-broadcast-bootloader-server/ota-broadcast-bootloader-server.h`

EMBER_AF_PLUGIN_OTA_BROADCAST_BOOTLOADER_SERVER_RESPONSE_TIMEOUT_MS

```
#define EMBER_AF_PLUGIN_OTA_BROADCAST_BOOTLOADER_SERVER_RESPONSE_TIMEOUT_MS
```

Value:

```
250
```

The time in milliseconds after which the server gives up waiting for a response from a client.

Definition at line `63` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-broadcast-bootloader/ota-broadcast-bootloader-server/ota-broadcast-bootloader-server.h`

EMBER_AF_PLUGIN_OTA_BROADCAST_BOOTLOADER_SERVER_MAX_BROADCAST_ROUNDS

```
#define EMBER_AF_PLUGIN_OTA_BROADCAST_BOOTLOADER_SERVER_MAX_BROADCAST_ROUNDS
```

Value:

```
5
```

The maximum number of image broadcast rounds the server performs before declaring an image distribution process failed.

Definition at line `69` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-broadcast-bootloader/ota-broadcast-bootloader-server/ota-broadcast-bootloader-server.h`

Ota Broadcast Bootloader Common

Ota Broadcast Bootloader Common

Set of types defined for ota-broadcast-bootloader.

OTA bootloading plugins are usable to send firmware images Over The Air when the application is running. When the firmware is downloaded to a device, a bootloader can be started to replace the application in the flash to the one just downloaded.

All Connect bootloader-related code relies on the Gecko Bootloader for bootloading and it must be installed on the device for these plugins to work. For details on the Gecko Bootloader, see UG266.

The Broadcast OTA plugins implement the OTA download operation in broadcast, so the same image can be sent to many devices at the same time. The server however requires to know the clients downloading the image, because it implements error handling by querying all clients for missing segments, and then the server will re-broadcast those segments.

Communication relies on standard broadcast data messages, which means routing is not available, and only clients that are in the range of the server can download. However, the same device can be both client and server, i.e. after downloading the image, a client can configure itself to be a server, and provide the image to another part of the network.

Sleepy end devices cannot be addressed in broadcast, but a sleepy end device can reconnect as a normal end device while the OTA is active.

Broadcast OTA uses a plugin configurable endpoint, which is 14 by default.

Security can be also enabled as plugin configuration on the server, as well as the interval of the messages. The client has a timeout plugin configuration after which it stops the OTA session with an error.

See UG235.06 for further details.

Note

- OTA Broadcast Bootloading plugins are not available in MAC mode due to the lack of endpoints.

Enumerations

```
enum    EmberAfOtaBootloaderStatus {
          EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_SUCCESS = 0x00
          EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_INVALID_CALL = 0x01
          EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_BUSY = 0x02
          EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_NO_BUFFERS = 0x03
          EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_DATA_UNDERFLOW = 0x04
          EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_STACK_ERROR = 0x05
          EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_TIMEOUT = 0x06
          EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_FAILED = 0x07
          EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_ABORTED = 0x08
        }
```

OTA Broadcast Bootloader return status codes.

enum EmberAfOtaBootloaderTargetStatus {

```
EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_INVALID = 0x00
EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_UNREACHABLE = 0x01
EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_DISTRIBUTION_COMPLETED = 0x02
EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_DISTRIBUTION_ONGOING = 0x03
EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_DISTRIBUTION_FAILED = 0x04
EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_DISTRIBUTION_REFUSED = 0x05
EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_DISTRIBUTION_ABORTED = 0x06
EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_STATUS_REQUEST_COMPLETED = 0x07
EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_STATUS_REQUEST_ONGOING = 0x08
EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_BOOTLOAD_REQUEST_ACCEPTED = 0x09
EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_BOOTLOAD_REQUEST_ONGOING = 0x0A
EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_BOOTLOAD_REQUEST_REFUSED = 0x0B
```

}
OTA Broadcast Bootloader target status codes, returned by emberAfPluginBootloaderServerGetTargetStatus().

Macros

#define EMBER_AF_PLUGIN_OTA_BROADCAST_BOOTLOADER_INVALID_APPLICATION_TARGET_STATUS 0xFF
A value indicating that client application did not set the application level target status in any of the client callbacks.

Enumeration Documentation

EmberAfOtaBootloaderStatus

```
EmberAfOtaBootloaderStatus
```

OTA Broadcast Bootloader return status codes.

Enumerator	
EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_SUCCESS	
EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_INVALID_CALL	
EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_BUSY	
EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_NO_BUFFERS	
EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_DATA_UNDERFLOW	
EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_STACK_ERROR	
EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_TIMEOUT	
EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_FAILED	
EMBER_OTA_BROADCAST_BOOTLOADER_STATUS_ABORTED	

Definition at line `79` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-broadcast-bootloader/ota-broadcast-bootloader-types.h`

EmberAfOtaBootloaderTargetStatus

```
EmberAfOtaBootloaderTargetStatus
```

OTA Broadcast Bootloader target status codes, returned by emberAfPluginBootloaderServerGetTargetStatus().

Enumerator	
EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_INVALID	
EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_UNREACHABLE	
EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_DISTRIBUTION_COMPLETED	
EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_DISTRIBUTION_ONGOING	

EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_DISTRIBUTION_FAILED	
EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_DISTRIBUTION_REFUSED	
EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_DISTRIBUTION_ABORTED	
EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_STATUS_REQUEST_COMPLETED	
EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_STATUS_REQUEST_ONGOING	
EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_BOOTLOAD_REQUEST_ACCEPTED	
EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_BOOTLOAD_REQUEST_ONGOING	
EMBER_OTA_BROADCAST_BOOTLOADER_TARGET_STATUS_BOOTLOAD_REQUEST_REFUSED	

Definition at line `117` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-broadcast-bootloader/ota-broadcast-bootloader-types.h`

Macro Definition Documentation

EMBER_AF_PLUGIN_OTA_BROADCAST_BOOTLOADER_INVALID_APPLICATION_TARGET_STATUS

```
#define EMBER_AF_PLUGIN_OTA_BROADCAST_BOOTLOADER_INVALID_APPLICATION_TARGET_STATUS
```

Value:

```
0xFF
```

A value indicating that client application did not set the application level target status in any of the client callbacks.

Definition at line `189` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/ota-broadcast-bootloader/ota-broadcast-bootloader-types.h`

Poll Plugin

Poll Plugin

APIs for the poll plugin.

The Connect stack supports polling which enables (sleepy) end devices to retrieve pending messages from the parent node (coordinator or range extender).

To use this feature, the Poll plugin must be enabled on the end devices. If polling is enabled, the end device sends a data request to the parent node, which notifies the device whether a message is pending or not using the acknowledge with the pending bit cleared or set. If a message is not pending, the communication ends with the acknowledge. If a message is pending, the parent node sends a data packet containing the pending message which will be acknowledged by the end device.

For convenience, Connect supports two polling intervals, long and short, which behave the same only the polling period differs. For long polling, the period is specified in seconds while for short polling, the period is in quarter seconds. The API provides a function to easily switch between the two. The purpose of long polling is maintaining the connection between the end device and the parent.

The application will receive the polled message via the emberAfIncomingMessageCallback() function.

The poll plugin uses emberPollForData() to retrieve the pending message. If the poll plugin is enabled, using emberPollForData() is strongly not recommended.

See poll.h for source code.

Functions

void	emberAfPluginPollSetShortPollInterval(uint8_t intervalQS) Set the short poll interval.
void	emberAfPluginPollSetLongPollInterval(uint16_t intervalS) Set the long poll interval.
void	emberAfPluginPollEnableShortPolling(bool enable) Enable/disable short polling.

Function Documentation

emberAfPluginPollSetShortPollInterval

```
void emberAfPluginPollSetShortPollInterval (uint8_t intervalQS)
```

Set the short poll interval.

Parameters

[in]	intervalQS	The short poll interval in quarter seconds.

Definition at line `68` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/poll/poll.h`

emberAfPluginPollSetLongPollInterval

```
void emberAfPluginPollSetLongPollInterval (uint16_t intervalS)
```

Set the long poll interval.

Parameters

[in]	intervalS	The long poll interval in seconds.

Definition at line `74` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/poll/poll.h`

emberAfPluginPollEnableShortPolling

```
void emberAfPluginPollEnableShortPolling (bool enable)
```

Enable/disable short polling.

Parameters

[in]	enable	If this parameter is true, short polling is enabled. Otherwise, the node switches back to long polling.

Definition at line `81` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/protocol/flex/poll/poll.h`

WSTK Sensors Plugin

WSTK Sensors Plugin

Hardware Abstraction Layer (HAL) API Reference

Hardware Abstraction Layer (HAL) API Reference

HAL function names have the following prefix conventions:

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

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

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

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

See also hal.h.

Modules

Hardware Abstraction Layer (HAL)

Common Microcontroller Functions

Token Access

Sample APIs for Peripheral Access

System Timer Control

Symbol Timer Control

HAL Configuration

HAL Utilities

Hardware Abstraction Layer (HAL)

Hardware Abstraction Layer (HAL)

The Hardware Abstraction Layer (HAL) is program code between a system's hardware and its software that provides a consistent interface for applications that can run on several different hardware platforms. To take advantage of this capability, applications should access hardware through the API provided by the HAL, rather than directly. Then, when you move to new hardware, you only need to update the HAL. In some cases, due to extreme differences in hardware, the HAL API may also change slightly to accommodate the new hardware. In these cases, the limited scope of the update makes moving the application easier with the HAL than without.

HAL function names have the following prefix conventions:

- **halCommon:** API that is used by the EmberZNet stack and can also be called from an application. This API must be implemented. Custom applications can change the implementation of the API but its functionality must remain the same.
- **hal:** API that is used by sample applications. Custom applications can remove this API or change its implementation as they see fit.
- **halStack:** API used only by the EmberZNet stack. This API must be implemented and should not be directly called from any application. Custom applications can change the implementation of the API, but its functionality must remain the same.
- **halInternal:** API that is internal to the HAL. The EmberZNet stack and applications must never call this API directly. Custom applications can change this API as they see fit. However, be careful not to impact the functionalty of any halStack or halCommon APIs.

See also hal.h.

Common Microcontroller Functions

Common Microcontroller Functions

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.

Modules

RTCCRamData

Enumerations

enum SleepModes {

```
SLEEPMODE_RUNNING = 0U
SLEEPMODE_IDLE = 1U
SLEEPMODE_WAKETIMER = 2U
SLEEPMODE_MAINTAINTIMER = 3U
SLEEPMODE_NOTIMER = 4U
SLEEPMODE_HIBERNATE = 5U
SLEEPMODE_RESERVED = 6U
SLEEPMODE_POWERDOWN = 7U
SLEEPMODE_POWERSAVE = 8U
```

}

Enumerations for the possible microcontroller sleep modes.

Typedefs

typedef uint32_t	WakeEvents
typedef uint32_t	WakeMask

Variables

volatile int8_t halCommonVreg1v8EnableCount

Helper variable to track the state of 1.8V regulator.

Functions

void halStackProcessBootCount(void)

Called from emberInit and provides a means for the HAL to increment a boot counter, most commonly in non-volatile memory.

uint8_t halGetResetInfo(void)

Gets information about what caused the microcontroller to reset.

PGM_P	halGetResetString(void) Calls halGetResetInfo() and supplies a string describing it.
void	halInit(void) Initializes microcontroller-specific peripherals.
void	halReboot(void) Restarts the microcontroller and therefore everything else.
void	halPowerUp(void) Powers up microcontroller peripherals and board peripherals.
void	halPowerDown(void) Powers down microcontroller peripherals and board peripherals.
void	halResume(void) Resumes microcontroller peripherals and board peripherals.
void	halSuspend(void) Suspends microcontroller peripherals and board peripherals.
void	halInternalEnableWatchDog(void) Enables the watchdog timer.
void	halInternalDisableWatchDog(uint8_t magicKey) Disables the watchdog timer.
bool	halInternalWatchDogEnabled(void) Determines whether the watchdog has been enabled or disabled.
void	halSleep(SleepModes sleepMode) Puts the microcontroller to sleep in a specified mode.
void	halSleepPreserveInts(SleepModes sleepMode) Same as halSleep() except it preserves the current interrupt state rather than always enabling interrupts prior to returning.
void	halCommonDelayMicroseconds(uint16_t us) Blocks the current thread of execution for the specified amount of time, in microseconds.
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 exteranl signals include analog sources ADC0 thru ADC5 and VDD_PADS/4.
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 exteranl signals include analog sources ADC0 thru ADC5 and VDD_PADS/4. The state of 1v8 survives deep sleep.
void	halBeforeEM4(uint32_t duration, RTCCRamData input)
RTCCRamData	halAfterEM4(void)

Macros

#define	halGetEm2xxResetInfo () Calls ::halGetExtendedResetInfo() and translates the EM35x reset code to the corresponding value used by the EM2XX HAL. Any reset codes not present in the EM2XX are returned after being OR'ed with 0x80.

#define MICRO_DISABLE_WATCH_DOG_KEY 0xA5U
The value that must be passed as the single parameter to halInternalDisableWatchDog() in order to successfully disable the watchdog timer.

#define GPIO_MASK_SIZE 24

#define GPIO_MASK 0xFFFFFF

#define WAKE_GPIO_MASK GPIO_MASK

#define WAKE_GPIO_SIZE GPIO_MASK_SIZE

#define WAKE_MASK_INVALID (-1)

#define WAKE_EVENT_SIZE WakeMask

#define DEBUG_TOGGLE (n)

Enumeration Documentation

SleepModes

```
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 `106` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro-common.h`

Typedef Documentation

WakeEvents

```
typedef uint32_t WakeEvents
```

Definition at line `141` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro-common.h`

WakeMask

```
typedef uint32_t WakeMask
```

Definition at line `142` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro-common.h`

Variable Documentation

halCommonVreg1v8EnableCount

```
volatile int8_t halCommonVreg1v8EnableCount
```

Helper variable to track the state of 1.8V regulator.

Note

- : Only used when DISABLE_INTERNAL_1V8_REGULATOR is defined.

Definition at line `195` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro-common.h`

Function Documentation

halStackProcessBootCount

```
void halStackProcessBootCount (void)
```

Called from emberInit and provides a means for the HAL to increment a boot counter, most commonly in non-volatile memory.

Parameters

N/A		

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:\n 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.

Definition at line `68` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro.h`

halGetResetInfo

```
uint8_t halGetResetInfo (void)
```

Gets information about what caused the microcontroller to reset.

Parameters

N/A		

Returns

- A code identifying the cause of the reset.

Definition at line `74` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro.h`

halGetResetString

```
PGM_P halGetResetString (void)
```

Calls halGetResetInfo() and supplies a string describing it.

Parameters

N/A		

- Application Usage:\n Useful for diagnostic printing of text just after program

  initialization.

Returns

- A pointer to a program space string.

Definition at line `83` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro.h`

halInit

```
void halInit (void)
```

Initializes microcontroller-specific peripherals.

Parameters

N/A		

Definition at line `30` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro-common.h`

halReboot

```
void halReboot (void)
```

Restarts the microcontroller and therefore everything else.

Parameters

N/A		

Definition at line `34` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro-common.h`

halPowerUp

```
void halPowerUp (void)
```

Powers up microcontroller peripherals and board peripherals.

Parameters

N/A		

Definition at line `38` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro-common.h`

halPowerDown

```
void halPowerDown (void)
```

Powers down microcontroller peripherals and board peripherals.

Parameters

N/A		

Definition at line `42` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro-common.h`

halResume

```
void halResume (void)
```

Resumes microcontroller peripherals and board peripherals.

Parameters

N/A		

Definition at line `46` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro-common.h`

halSuspend

```
void halSuspend (void)
```

Suspends microcontroller peripherals and board peripherals.

Parameters

N/A		

Definition at line `50` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro-common.h`

halInternalEnableWatchDog

```
void halInternalEnableWatchDog (void)
```

Enables the watchdog timer.

Parameters

N/A		

Definition at line 60 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro-common.h

halInternalDisableWatchDog

```
void halInternalDisableWatchDog (uint8_t magicKey)
```

Disables the watchdog timer.

Parameters

N/A	magicKey	A value (MICRO_DISABLE_WATCH_DOG_KEY) that enables the function.

Note

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

Definition at line 69 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro-common.h

halInternalWatchDogEnabled

```
bool halInternalWatchDogEnabled (void)
```

Determines whether the watchdog has been enabled or disabled.

Parameters

N/A		

Returns

- A bool value indicating if the watchdog is current enabled.

Definition at line 75 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro-common.h

halSleep

```
void halSleep (SleepModes sleepMode)
```

Puts the microcontroller to sleep in a specified mode.

Parameters

N/A	sleepMode	A microcontroller sleep mode

Note

- This routine always enables interrupts.

See Also

- SleepModes

Definition at line 160 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro-common.h

halSleepPreserveInts

```
void halSleepPreserveInts (SleepModes sleepMode)
```

Same as halSleep() except it preserves the current interrupt state rather than always enabling interrupts prior to returning.

Parameters

N/A	sleepMode	A microcontroller sleep mode

See Also

- SleepModes

Definition at line `169` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro-common.h`

halCommonDelayMicroseconds

```
void halCommonDelayMicroseconds (uint16_t us)
```

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

Parameters

N/A	us	The specified time, in microseconds. Values should be between 1 and 65535 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.

Definition at line `186` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro-common.h`

halCommonDisableVreg1v8

```
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 exteranl signals include analog sources ADC0 thru ADC5 and VDD_PADS/4.

Parameters

N/A		

Note

- : Only used when DISABLE_INTERNAL_1V8_REGULATOR is defined.

Definition at line `206` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro-common.h`

halCommonEnableVreg1v8

```
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 exteranl signals include analog sources ADC0 thru ADC5 and VDD_PADS/4. The state of 1v8 survives deep sleep.

Parameters

N/A		

Note

- : Only used when DISABLE_INTERNAL_1V8_REGULATOR is defined.

Definition at line `217` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro-common.h`

halBeforeEM4

```
void halBeforeEM4 (uint32_t duration, RTCCRamData input)
```

Parameters

N/A	duration	
N/A	input	

Definition at line `228` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro-common.h`

halAfterEM4

```
RTCCRamData halAfterEM4 (void)
```

Parameters

N/A		

Definition at line `229` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro-common.h`

Macro Definition Documentation

halGetEm2xxResetInfo

```
#define halGetEm2xxResetInfo
```

Value:

```
()
```

Calls ::halGetExtendedResetInfo() and translates the EM35x reset code to the corresponding value used by the EM2XX HAL. Any reset codes not present in the EM2XX are returned after being OR'ed with 0x80.

- Application Usage:\n 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 98 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro.h

MICRO_DISABLE_WATCH_DOG_KEY

```
#define MICRO_DISABLE_WATCH_DOG_KEY
```

Value:

```
0xA5U
```

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

Definition at line 56 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro-common.h

GPIO_MASK_SIZE

```
#define GPIO_MASK_SIZE
```

Value:

```
24
```

Definition at line 137 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro-common.h

GPIO_MASK

```
#define GPIO_MASK
```

Value:

```
0xFFFFFF
```

Definition at line 138 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro-common.h

WAKE_GPIO_MASK

```
#define WAKE_GPIO_MASK
```

Value:

```
GPIO_MASK
```

Definition at line 139 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro-common.h

WAKE_GPIO_SIZE

```
#define WAKE_GPIO_SIZE
```

Value:

```
GPIO_MASK_SIZE
```

Definition at line 140 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro-common.h

```
#define WAKE_MASK_INVALID
```

Value:

```
(-1)
```

Definition at line `145` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro-common.h`

WAKE_EVENT_SIZE

```
#define WAKE_EVENT_SIZE
```

Value:

```
WakeMask
```

Note

- The preprocessor symbol WAKE_EVENT_SIZE has been deprecated. Please use WakeMask instead.

Definition at line `150` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro-common.h`

DEBUG_TOGGLE

```
#define DEBUG_TOGGLE
```

Definition at line `188` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro-common.h`

RTCCRamData

Public Attributes

uint32_t	outgoingNwkFrameCounter
uint32_t	incomingParentNwkFrameCounter
uint32_t	outgoingLinkKeyFrameCounter
uint32_t	incomingLinkKeyFrameCounter

Public Attribute Documentation

outgoingNwkFrameCounter

```
uint32_t RTCCRamData::outgoingNwkFrameCounter
```

Definition at line `223` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro-common.h`

incomingParentNwkFrameCounter

```
uint32_t RTCCRamData::incomingParentNwkFrameCounter
```

Definition at line `224` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro-common.h`

outgoingLinkKeyFrameCounter

```
uint32_t RTCCRamData::outgoingLinkKeyFrameCounter
```

Definition at line `225` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro-common.h`

incomingLinkKeyFrameCounter

```
uint32_t RTCCRamData::incomingLinkKeyFrameCounter
```

Definition at line `226` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/micro-common.h`

Token Access

Token Access

The token system stores such non-volatile information as the manufacturing ID, channel number, transmit power, and various pieces of information that the application needs to be persistent between device power cycles. The token system is design to abstract implementation details and simplify interacting with differing non-volatile systems. The majority of tokens are stored in Simulated EEPROM or NVM3 (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 simeeprom2 module for a detailed description of the necessary support functions for Simulated EEPROM, version 2. Refer to the nvm3 module for a detailed description of the necessary support functions for NVM3. Refer to token-stack.h for stack token definitions. Refer to token-manufacturing.h for manufaturing token definitions.**Note**

- Simulated EEPROM, version 2 is only supported on EM335x chips.
- NVM3 is currently only supported on EFx32 chips.

Modules

Tokens

Simulated EEPROM

Tokens

Tokens

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
#define NVM3KEY_name 20bit_value
#ifdef DEFINETYPES
  typedef data_type type
#endif
#ifdef DEFINETOKENS
  DEFINE_*_TOKEN(name, type, ... ,defaults)
#endif
```

The defined CREATOR is used as a distinct identifier tag for the token when using Simulated EEPROM or with manufacturing tokens. 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 inorder to make the codes more memorable. The 'name' part of the `#define CREATOR_name` must match the 'name' provided in the `DEFINE_*_TOKEN` because the token system uses this name to automatically link the two.

The defined NVM3KEY is used to map the token to an NVM3 key and is needed using NVM3 as the underlying storage mechanism. This key can also be used as an identifier for a token's NVM3 object when using the native NVM3 API. The NVM3 keys must be unique for one instance of the NVM3 backing storage. All tokens share the same NVM3 instance and hence all NVM3KEYS for tokens must be unique. The 'name' part of the `#define NVM3KEY_name` must match the 'name' provided in the `DEFINE_*_TOKEN` because the token system uses this name to automatically link the two. For indexed tokens, the 127 NVM3KEY values following the defined NVM3KEY for a token should also be reserved. This is done as one NVM3KEY is used for each index in an indexed token and hence these NVM3KEYS should not collide with the eys of other tokens.

As NVM3 is shared among several stacks and application code, the NVM3KEY values chosen must be defined in the correct region to avoid collisions.

The following NVM3KEY regions are defined: 0x0xxxx : User objects 0x1xxxx : zigbee stack objects 0x2xxxx : Thread stack objects 0x3xxxx : Connect stack objects 0x4xxxx : Bluetooth stack objects

The typedef provides a convenient and efficient abstraction of the token data. Since some tokens are structs with multiple pieces of data inside of them, type defining the token type allows more efficient and readable local copies of the tokens throughout the code.

The typedef is wrapped with an `#ifdef DEFINETYPES` because the typdefs and token defs live in the same file, and DEFINETYPES is used to select only the typedefs when the file is included. Similarly, the `DEFINE_*_TOKEN` is wrapped with an `#ifdef DEFINETOKENS` as a method for selecting only the token definitions when the file is included.

The abstract definition, `DEFINE_*_TOKEN(name, type, ... ,defaults)` , has seven possible complete definitions: `DEFINE_BASIC_TOKEN(name, type, ...) DEFINE_INDEXED_TOKEN(name, type, arraysize, ...) DEFINE_COUNTER_TOKEN(name, type, ...) DEFINE_MFG_TOKEN(name, type, address, ...)` The three fields common to all `DEFINE_*_TOKEN` are: name - The name of the token, which all information is tied to. type - Type of the token which is the same as the typedef mentioned before. ... - The default value to which the token is set upon initialization.

Note

- The old DEFINE_FIXED* token definitions are no longer used. They remain defined for backwards compatibility. In current systems, the Simulated EEPROM or NVM3 is used for storing non-manufacturing tokens and the Simulated EEPROM or NVM3 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]
```

<br<blockquote> 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. Note that when assigning an NVM3KEY for an indexed token, the 126 higher numbered NVM3KEYs following the NVM3KEY that you define are reserved for that token and no other tokens should be defined with NVM3KEYs in this region.

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 or NVM3 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. The definition is compatible with both Simulated EEPROM and NVM3 as both CREATOR and NVM3KEY defines are included.

```
#define CREATOR_SENSOR_NAME       0x5354
#define CREATOR_SENSOR_PARAMETERS  0x5350
#define NVM3KEY_SENSOR_NAME       0x0AB54
#define NVM3KEY_SENSOR_PARAMETERS 0x00150
#ifdef DEFINETYPES
 typedef uint8_t tokTypeSensorName[10];
 typedef struct {
  uint8_t initValues[5];
  uint8_t reportInterval;
  uint16_t calibrationValue;
 } tokTypeSensorParameters;
#endif
#ifdef DEFINETOKENS
 DEFINE_BASIC_TOKEN(SENSOR_NAME,
                   tokTypeSensorName,
                   {'U','N','A','M','E','D',' ',' ',' ',' '})
 DEFINE_BASIC_TOKEN(SENSOR_PARAMETERS,
                   tokTypeSensorParameters,
                   {{0x01,0x02,0x03,0x04,0x05},5,0x0000})
#endif
```

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 and NVM3.**

The Simulated EEPROM or NVM3 will initialize tokens to their default values if the token does not yet exist, the token's creator code is changed, or the token's size changes.

Changing the number indexes in an INDEXED token will not alter existing entries. If the number of indexes is reduced, the entires that still fit in the token will retain their data and the entries that no longer fit will be erased. If the number of indexes is increased, the existing entries retain their data and the new entries are initialized to the token's defaults.

Further details on exact implementation can be found in code comments in token-stack.h file, the platform specific token-manufacturing.h file, the platform specific token.h file, and the platform specific token.c file.

Some functions in this file return an EmberStatus value. See error-def.h for definitions of all EmberStatus return values.

See hal/micro/token.h for source code.

Functions

EmberStatus	halStackInitTokens(void) Initializes and enables the token system. Checks if the manufacturing and stack non-volatile data versions are correct.

Macros

#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.
#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.
#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.
#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.
#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.
#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.

Function Documentation

halStackInitTokens

```
EmberStatus halStackInitTokens (void)
```

Initializes and enables the token system. Checks if the manufacturing and stack non-volatile data versions are correct.

Parameters

N/A		

Returns

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

Definition at line `294` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/token.h`

Macro Definition Documentation

halCommonGetToken

```
#define halCommonGetToken
```

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.

Definition at line `318` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/token.h`

halCommonGetMfgToken

```
#define halCommonGetMfgToken
```

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.

Definition at line `333` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/token.h`

halCommonGetIndexedToken

```
#define halCommonGetIndexedToken
```

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.

Definition at line `349` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/token.h`

halCommonSetToken

```
#define halCommonSetToken
```

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. For EFR32 devices this function must not be called in IRQ context as it can cause data corruption.

Definition at line `365` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/token.h`

halCommonSetIndexedToken

```
#define halCommonSetIndexedToken
```

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.

Definition at line `382` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/token.h`

halCommonIncrementCounterToken

```
#define halCommonIncrementCounterToken
```

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.

Definition at line `395` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/token.h`

Simulated EEPROM

Simulated EEPROM

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 threhold, 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/plugin/sim-eeprom/sim-eeprom.h for source code.

Functions

void	halSimEepromCallback(EmberStatus status) The Simulated EEPROM callback function, implemented by the application.
uint8_t	halSimEepromErasePage(void) Erases a hardware flash page, if needed.
uint8_t	halSimEepromPagesRemainingToBeErased(void) Get count of pages to be erased.
void	halSimEepromStatus(uint16_t *freeWordsUntilFull, uint16_t *totalPageUseCount) Provides two basic statistics.

Function Documentation

halSimEepromCallback

```
void halSimEepromCallback (EmberStatus status)
```

The Simulated EEPROM callback function, implemented by the application.

Parameters

N/A	status	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 occuring. 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.

Definition at line `130` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/plugin/sim-eeprom/sim-eeprom.h`

halSimEepromErasePage

```
uint8_t halSimEepromErasePage (void)
```

Erases a hardware flash page, if needed.

Parameters

N/A		

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 hardware pages are left to be erased. This return value allows for calling code to easily loop over this function until the function returns 0.

Definition at line 158 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/plugin/sim-eeprom/sim-eeprom.h

halSimEepromPagesRemainingToBeErased

```
uint8_t halSimEepromPagesRemainingToBeErased (void)
```

Get count of pages to be erased.

Parameters

N/A		

This function returns the same value halSimEepromErasePage() would return, but without modifying/erasing any flash.

Returns

- A count of how many hardware pages are left to be erased. This code assist with loops wanting to know how much is left to erase.

Definition at line 168 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/plugin/sim-eeprom/sim-eeprom.h

halSimEepromStatus

```
void halSimEepromStatus (uint16_t *freeWordsUntilFull, uint16_t *totalPageUseCount)
```

Provides two basic statistics.

Parameters

N/A	freeWordsUntilFull	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.
N/A	totalPageUseCount	The value of the highest page counter indicating how many times the Simulated EEPROM has rotated physical flash pages (and approximate write cycles).

- 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).

Definition at line 190 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/plugin/sim-eeprom/sim-eeprom.h

Sample APIs for Peripheral Access

Sample APIs for Peripheral Access

These are sample API for accessing peripherals and can be modified as needed for your applications.

Modules

Serial UART Communication

Serial UART Communication

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 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.

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) Initializes the UART to the given settings (same parameters as ::emberSerialInit()).
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.
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.
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 progess.
void	halInternalStopUartTx(uint8_t port) Called by serial code to stop any interrupt-driven serial transmission currently in progress.
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.
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.
void	halInternalWaitUartTxComplete(uint8_t port) Blocks until the UART has finished transmitting any data in its hardware registers.
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.)

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 is the host is able to send.
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.
bool	halInternalUartTxIsIdle(uint8_t port) Returns true if the uart transmitter is idle, including the transmit shift register.
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.
#define	halInternalUartFlowControl (port) This function is used in FIFO mode when flow control is enabled. It is called from emberSerialReadByte(), and based on the number of bytes used in the uart receive queue, decides when to tell the host it may resume transmission.
#define	halInternalUartRxPump (port) This function exists only in software UART (SOFTUART) mode on the EM3xx. This function is called by ::emberSerialReadByte(). It is responsible for maintaining synchronization between the emSerialRxQueue and the UART DMA.
#define	halInternalUart1FlowControlRxIsEnabled () 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.
#define	halInternalUart1XonRefreshDone () 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.
#define	halInternalUart1TxIsIdle () 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.

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) 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.

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 0 A numerical definition for a possible serial mode the code can test for.
#define	EMBER_SERIAL_FIFO 1 A numerical definition for a possible serial mode the code can test for.
#define	EMBER_SERIAL_LOWLEVEL 2 A numerical definition for a possible serial mode the code can test for.

FIFO Utility Macros

These macros manipulate the FIFO queue data structures to add and remove data.

#define	FIFO_ENQUEUE (queue, data, size) Macro that enqueues a byte of data in a FIFO queue.
#define	FIFO_DEQUEUE (queue, size) Macro that de-queues a byte of data from a FIFO queue.

Enumerations

enum SerialBaudRate {

```
  DEFINE_BAUD =(300) = 0
  DEFINE_BAUD =(300) = 0
  DEFINE_BAUD =(300) = 0
  DEFINE_BAUD =(300) = 0
  DEFINE_BAUD =(300) = 0
  DEFINE_BAUD =(300) = 0
  DEFINE_BAUD =(300) = 0
  DEFINE_BAUD =(300) = 0
  DEFINE_BAUD =(300) = 0
  DEFINE_BAUD =(300) = 0
  DEFINE_BAUD =(300) = 0
  DEFINE_BAUD =(300) = 0
  DEFINE_BAUD =(300) = 0
  DEFINE_BAUD =(300) = 0
  DEFINE_BAUD =(300) = 0
  DEFINE_BAUD =(300) = 0
  DEFINE_BAUD =(300) = 0
  DEFINE_BAUD =(300) = 0
  DEFINE_BAUD =(300) = 0
```

}
Assign numerical values for variables that hold Baud Rate parameters.

enum SerialParity {

```
  DEFINE_PARITY =(NONE) = 0U
  DEFINE_PARITY =(NONE) = 0U
  DEFINE_PARITY =(NONE) = 0U
```

}
CORTEXM3_EFM32_MICRO.

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 HAL APIs Documentation

halInternalUartInit

```
EmberStatus halInternalUartInit (uint8_t port, SerialBaudRate rate, SerialParity parity, uint8_t stopBits)
```

Initializes the UART to the given settings (same parameters as ::emberSerialInit()).

Parameters

N/A	port	Serial port number (0 or 1).
N/A	rate	Baud rate (see SerialBaudRate).
N/A	parity	Parity value (see SerialParity).
N/A	stopBits	Number of stop bits.

Returns

- An error code if initialization failed (such as invalid baud rate), otherise EMBER_SUCCESS.

Definition at line `410` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/serial.h`

halInternalPowerDownUart

```
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.

Parameters

N/A		

Definition at line `419` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/serial.h`

halInternalPowerUpUart

```
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.

Parameters

N/A		

Definition at line `425` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/serial.h`

halInternalStartUartTx

```
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 progess.

Parameters

N/A	port	Serial port number (0 or 1).

Definition at line `433` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/serial.h`

halInternalStopUartTx

```
void halInternalStopUartTx (uint8_t port)
```

Called by serial code to stop any interrupt-driven serial transmission currently in progress.

Parameters

N/A	port	Serial port number (0 or 1).

Definition at line `440` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/serial.h`

halInternalForceWriteUartData

```
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

N/A	port	Serial port number (0 or 1).
N/A	data	Pointer to the data to be transmitted.
N/A	length	The length of data to be transmitted

Definition at line `453` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/serial.h`

halInternalForceReadUartByte

```
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

N/A	port	Serial port number (0 or 1).
N/A	dataByte	The byte to receive data into.

Definition at line `463` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/serial.h`

halInternalWaitUartTxComplete

```
void halInternalWaitUartTxComplete (uint8_t port)
```

Blocks until the UART has finished transmitting any data in its hardware registers.

Parameters

N/A	port	Serial port number (0 or 1).

Definition at line `470` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/serial.h`

halInternalRestartUart

```
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.)

Parameters

N/A		

Definition at line `506` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/serial.h`

halInternalUartFlowControlRxIsEnabled

```
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 is the host is able to send.

Parameters

N/A	port	

Definition at line `513` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/serial.h`

halInternalUartXonRefreshDone

```
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.

Parameters

N/A	port	

Definition at line `525` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/serial.h`

halInternalUartTxIsIdle

```
bool halInternalUartTxIsIdle (uint8_t port)
```

Returns true if the uart transmitter is idle, including the transmit shift register.

Parameters

N/A	port	

Definition at line `537` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/serial.h`

serialDropPacket

```
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.

Parameters

N/A		

Definition at line 547 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/serial.h

halInternalUartFlowControl

```
#define halInternalUartFlowControl
```

Value:

```
(port)
```

This function is used in FIFO mode when flow control is enabled. It is called from emberSerialReadByte(), and based on the number of bytes used in the uart receive queue, decides when to tell the host it may resume transmission.

Definition at line 484 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/serial.h

halInternalUartRxPump

```
#define halInternalUartRxPump
```

Value:

```
(port)
```

This function exists only in software UART (SOFTUART) mode on the EM3xx. This function is called by ::emberSerialReadByte(). It is responsible for maintaining synchronization between the emSerialRxQueue and the UART DMA.

Definition at line 498 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/serial.h

halInternalUart1FlowControlRxIsEnabled

```
#define halInternalUart1FlowControlRxIsEnabled
```

Value:

```
()
```

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.

Definition at line 516 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/serial.h

halInternalUart1XonRefreshDone

```
#define halInternalUart1XonRefreshDone
```

Value:

```
()
```

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.

Definition at line `528` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/serial.h`

halInternalUart1TxIsIdle

```
#define halInternalUart1TxIsIdle
```

Value:

```
()
```

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.

Definition at line `541` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/serial.h`

Virtual UART API Documentation

halStackReceiveVuartMessage

```
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

N/A	data	Pointer to the the data received
N/A	length	Length of the data received

Definition at line `574` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/serial.h`

Serial Mode Definitions Documentation

EMBER_SERIAL_UNUSED

```
#define EMBER_SERIAL_UNUSED
```

Value:

```
0
```

A numerical definition for a possible serial mode the code can test for.

Definition at line `91` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/serial.h`

EMBER_SERIAL_FIFO

```
#define EMBER_SERIAL_FIFO
```

Value:

```
1
```

A numerical definition for a possible serial mode the code can test for.

Definition at line `92` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/serial.h`

EMBER_SERIAL_LOWLEVEL

```
#define EMBER_SERIAL_LOWLEVEL
```

Value:

```
2
```

A numerical definition for a possible serial mode the code can test for.

Definition at line `93` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/serial.h`

FIFO Utility Macros Documentation

FIFO_ENQUEUE

```
#define FIFO_ENQUEUE
```

Value:

```
do {                                    \
  (queue)->fifo[(queue)->head] = (data);        \
  (queue)->head = (((queue)->head + 1) % (size)); \
  (queue)->used++;                              \
} while (0)
```

Macro that enqueues a byte of data in a FIFO queue.

Definition at line `272` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/serial.h`

FIFO_DEQUEUE

```
#define FIFO_DEQUEUE
```

Value:

```
(queue)->fifo[(queue)->tail];                 \
(queue)->tail = (((queue)->tail + 1) % (size)); \
(queue)->used--
```

Macro that de-queues a byte of data from a FIFO queue.

Definition at line `287` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/serial.h`

Enumeration Documentation

SerialBaudRate

```
SerialBaudRate
```

Assign numerical values for variables that hold Baud Rate parameters.

Enumerator	
DEFINE_BAUD	
DEFINE_BAUD	
DEFINE_BAUD	
DEFINE_BAUD	
DEFINE_BAUD	
DEFINE_BAUD	
DEFINE_BAUD	
DEFINE_BAUD	
DEFINE_BAUD	
DEFINE_BAUD	
DEFINE_BAUD	
DEFINE_BAUD	
DEFINE_BAUD	
DEFINE_BAUD	
DEFINE_BAUD	
DEFINE_BAUD	
DEFINE_BAUD	
DEFINE_BAUD	
DEFINE_BAUD	

Definition at line `300` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/serial.h`

SerialParity

> SerialParity

CORTEXM3_EFM32_MICRO.

Assign numerical values for the types of parity. Use for variables that hold Parity parameters.

Enumerator	
DEFINE_PARITY	
DEFINE_PARITY	
DEFINE_PARITY	

Definition at line `370` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/serial.h`

Function Documentation

halHostFlushBuffers

> void halHostFlushBuffers (void)

Parameters

N/A		

Definition at line `585` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/serial.h`

halHostEnqueueTx

```
uint16_t halHostEnqueueTx (const uint8_t *data, uint16_t length)
```

Parameters

N/A	data	
N/A	length	

Definition at line `586` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/serial.h`

halHostFlushTx

```
void halHostFlushTx (void)
```

Parameters

N/A		

Definition at line `587` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/serial.h`

serialCopyFromRx

```
uint16_t serialCopyFromRx (const uint8_t *data, uint16_t length)
```

Parameters

N/A	data	
N/A	length	

Definition at line `590` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/serial.h`

emLoadSerialTx

```
void emLoadSerialTx (void)
```

Parameters

N/A		

Definition at line `593` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/serial.h`

Button Control

Button Control

Sample API functions for using push-buttons.

See button.h for source code.

Button State Definitions

A set of numerical definitions for use with the button APIs indicating the state of a button.

#define	BUTTON_PRESSED 1 Button state is pressed.
#define	BUTTON_RELEASED 0 Button state is released.

Functions

void	halInternalInitButton(void) Initializes the buttons. This function is automatically called by halInit().
uint8_t	halButtonState(uint8_t button) Returns the current state (pressed or released) of a button.
uint8_t	halButtonPinState(uint8_t button) Returns the current state (pressed or released) of the pin associated with a button.
void	halButtonIsr(uint8_t button, uint8_t state) A callback called in interrupt context whenever a button changes its state.

Button State Definitions Documentation

BUTTON_PRESSED

```
#define BUTTON_PRESSED
```

Value:

```
1
```

Button state is pressed.

Definition at line `32` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/button.h`

BUTTON_RELEASED

```
#define BUTTON_RELEASED
```

Value:

```
0
```

Button state is released.

Definition at line `36` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/button.h`

Function Documentation

halInternalInitButton

```
void halInternalInitButton (void)
```

Initializes the buttons. This function is automatically called by halInit().

Parameters

N/A		

Definition at line `43` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/button.h`

halButtonState

```
uint8_t halButtonState (uint8_t button)
```

Returns the current state (pressed or released) of a button.

Parameters

N/A	button	The button being queried, either BUTTON0 or BUTTON1 as defined in the appropriate BOARD_HEADER.

Note

- This function is correlated with halButtonIsr() and so returns the shadow state rather than reading the actual state of the pin.

Returns

- BUTTON_PRESSED if the button is pressed or BUTTON_RELEASED if the button is not pressed.

Definition at line `56` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/button.h`

halButtonPinState

```
uint8_t halButtonPinState (uint8_t button)
```

Returns the current state (pressed or released) of the pin associated with a button.

Parameters

N/A	button	The button being queried, either BUTTON0 or BUTTON1 as defined in the appropriate BOARD_HEADER.

This reads the actual state of the pin and can be used on startup to determine the initial position of the buttons.

Returns

- BUTTON_PRESSED if the button is pressed or BUTTON_RELEASED if the button is not pressed.

Definition at line `70` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/button.h`

halButtonIsr

```
void halButtonIsr (uint8_t button, uint8_t state)
```

A callback called in interrupt context whenever a button changes its state.

Parameters

N/A	button	The button which has changed state, either BUTTON0 or BUTTON1 as defined in the appropriate BOARD_HEADER.
N/A	state	The new state of the button referenced by the button parameter, either BUTTON_PRESSED if the button has been pressed or BUTTON_RELEASED if the button has been released.

- # Application Usage:\n 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.

Definition at line `86` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/button.h`

Buzzer Control

Buzzer Control

LED Control

LED Control

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.

Typedefs

typedef enum HalBoardLedPins	HalBoardLed Ensures that the definitions from the BOARD_HEADER are always used as parameters to the LED functions.

Functions

void	halInternalInitLed(void) Configures GPIOs pertaining to the control of LEDs.
void	halToggleLed(HalBoardLed led) Atomically wraps an XOR or similar operation for a single GPIO pin attached to an LED.
void	halSetLed(HalBoardLed led) Turns on (sets) a GPIO pin connected to an LED so that the LED turns on.
void	halClearLed(HalBoardLed led) Turns off (clears) a GPIO pin connected to an LED, which turns off the LED.
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.

Typedef Documentation

HalBoardLed

```
typedef enum HalBoardLedPins HalBoardLed
```

Ensures that the definitions from the BOARD_HEADER are always used as parameters to the LED functions.

Definition at line `78` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/led.h`

Function Documentation

halInternalInitLed

```
void halInternalInitLed (void)
```

Configures GPIOs pertaining to the control of LEDs.

Parameters

N/A		

Definition at line `70` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/led.h`

halToggleLed

```
void halToggleLed (HalBoardLed led)
```

Atomically wraps an XOR or similar operation for a single GPIO pin attached to an LED.

Parameters

N/A	led	Identifier (from BOARD_HEADER) for the LED to be toggled.

Definition at line `90` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/led.h`

halSetLed

```
void halSetLed (HalBoardLed led)
```

Turns on (sets) a GPIO pin connected to an LED so that the LED turns on.

Parameters

N/A	led	Identifier (from BOARD_HEADER) for the LED to turn on.

Definition at line `97` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/led.h`

halClearLed

```
void halClearLed (HalBoardLed led)
```

Turns off (clears) a GPIO pin connected to an LED, which turns off the LED.

Parameters

N/A	led	Identifier (from BOARD_HEADER) for the LED to turn off.

Definition at line `104` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/led.h`

halStackIndicateActivity

```
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.

Parameters

N/A	turnOn	See Usage.

Typically does something interesting, such as change the state of an LED.

Definition at line `115` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/led.h`

Flash Memory Control

Flash Memory Control

Definition and description of public flash manipulation routines.

Note

- During an erase or a write the flash is not available, which means code will not be executable from flash. These routines still execute from flash, though, since the bus architecture can support doing so. **Additonally, this also means all interrupts will be disabled.**

Hardware documentation indicates 40us for a write and 21ms for an erase.

See flash.h for source code.

Functions

bool	halFlashEraseIsActive(void) Tells the calling code if a Flash Erase operation is active.

Function Documentation

halFlashEraseIsActive

```
bool halFlashEraseIsActive (void)
```

Tells the calling code if a Flash Erase operation is active.

Parameters

N/A		

This state is import to know because Flash Erasing is ATOMIC for 21ms and could disrupt interrupt latency. But if an ISR can know that it wasn't serviced immediately due to Flash Erasing, then the ISR has the opportunity to correct in whatever manner it needs to.

Returns

- A bool flag: true if Flash Erase is active, false otherwise.

Definition at line `46` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/flash.h`

System Timer Control

System Timer Control

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.

Functions

uint16_t	halInternalStartSystemTimer(void) Initializes the system tick.
uint16_t	halCommonGetInt16uMillisecondTick(void) Returns the current system time in system ticks, as a 16-bit value.
uint32_t	halCommonGetInt32uMillisecondTick(void) Returns the current system time in system ticks, as a 32-bit value.
uint64_t	halCommonGetInt64uMillisecondTick(void) Returns the current system time in system ticks, as a 64-bit value.
uint16_t	halCommonGetInt16uQuarterSecondTick(void) Returns the current system time in quarter second ticks, as a 16-bit value.
EmberStatus	halSleepForQuarterSeconds(uint32_t *duration) Uses the system timer to enter SLEEPMODE_WAKETIMER for approximately the specified amount of time (provided in quarter seconds).
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.
EmberStatus	halCommonIdleForMilliseconds(uint32_t *duration) Uses the system timer to enter SLEEPMODE_IDLE for approximately the specified amount of time (provided in milliseconds).

Macros

#define	halIdleForMilliseconds (duration)

Function Documentation

halInternalStartSystemTimer

```
uint16_t halInternalStartSystemTimer (void)
```

Initializes the system tick.

Parameters

N/A		

Returns

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

Definition at line `57` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/system-timer.h`

halCommonGetInt16uMillisecondTick

```
uint16_t halCommonGetInt16uMillisecondTick (void)
```

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

Parameters

N/A		

Returns

- The least significant 16 bits of the current system time, in system ticks.

Definition at line `66` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/system-timer.h`

halCommonGetInt32uMillisecondTick

```
uint32_t halCommonGetInt32uMillisecondTick (void)
```

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

Parameters

N/A		

- EmberStack Usage:\n Unused, implementation optional.

Returns

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

Definition at line `77` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/system-timer.h`

halCommonGetInt64uMillisecondTick

```
uint64_t halCommonGetInt64uMillisecondTick (void)
```

Returns the current system time in system ticks, as a 64-bit value.

Parameters

N/A		

- EmberStack Usage:\n Unused, implementation optional.

Returns

- 64 bits containing the current system time, in system ticks.

Definition at line `87` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/system-timer.h`

halCommonGetInt16uQuarterSecondTick

```
uint16_t halCommonGetInt16uQuarterSecondTick (void)
```

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

Parameters

N/A		

- EmberStack Usage:\n Unused, implementation optional.

Returns

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

Definition at line `98` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/system-timer.h`

halSleepForQuarterSeconds

```
EmberStatus halSleepForQuarterSeconds (uint32_t *duration)
```

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

Parameters

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

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:\n Unused, implementation optional.

Returns

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

Definition at line `141` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/system-timer.h`

halSleepForMilliseconds

```
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.

Parameters

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

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:\n Unused, implementation optional.

Returns

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

Definition at line `184` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/system-timer.h`

halCommonIdleForMilliseconds

```
EmberStatus halCommonIdleForMilliseconds (uint32_t *duration)
```

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

Parameters

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

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:\n Unused, implementation optional.

Returns

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

Definition at line `208` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/system-timer.h`

Macro Definition Documentation

halIdleForMilliseconds

```
#define halIdleForMilliseconds
```

Value:

```
(duration)
```

Definition at line `211` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/system-timer.h`

Symbol Timer Control

Symbol Timer Control

HAL Configuration

HAL Configuration

Modules

Sample Breakout Board Configuration

Sample Breakout Board Configuration

IAR PLATFORM_HEADER Configuration

IAR PLATFORM_HEADER Configuration

Compiler and Platform specific definitions and typedefs for the IAR ARM C compiler.

Note

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

See iar.h and platform-common.h for source code.

Master Variable Types

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

typedef bool	boolean A typedef to make the size of the variable explicitly known.
typedef unsigned char	int8u Denotes that this platform supports 64-bit data-types.
typedef signed char	int8s Denotes that this platform supports 64-bit data-types.
typedef unsigned short	int16u Denotes that this platform supports 64-bit data-types.
typedef signed short	int16s Denotes that this platform supports 64-bit data-types.
typedef unsigned int	int32u Denotes that this platform supports 64-bit data-types.
typedef signed int	int32s Denotes that this platform supports 64-bit data-types.
typedef unsigned long long	int64u Denotes that this platform supports 64-bit data-types.
typedef signed long long	int64s Denotes that this platform supports 64-bit data-types.
typedef unsigned int	PointerType Denotes that this platform supports 64-bit data-types.
#define	HAL_HAS_INT64 undefined Denotes that this platform supports 64-bit data-types.
#define	_HAL_USE_COMMON_PGM_ undefined Use the Master Program Memory Declarations from platform-common.h.

Miscellaneous Macros

void	hallnternalAssertFailed(const char *filename, int linenumber) A prototype definition for use by the assert macro. (see hal/micro/micro.h)
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.
#define	BIGENDIAN_CPU false A convenient method for code to know what endiannes processor it is running on. For the Cortex-M3, we are little endian.
#define	NTOHS (val) Define intrinsics for NTOHL and NTOHS to save code space by making endian.c compile to nothing.
#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.
#define	NO_STRIPPING __root A friendlier name for the compiler's intrinsic for not stripping.
#define	EEPROM errorerror A friendlier name for the compiler's intrinsic for eeprom reference.
#define	__SOURCEFILE__ __FILE__ 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.
#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)
#define	halResetWatchdog () A convenient method for code to know what endiannes processor it is running on. For the Cortex-M3, we are little endian.
#define	UNUSED undefined Declare a variable as unused to avoid a warning. Has no effect in IAR builds.
#define	SIGNED_ENUM undefined Some platforms need to cast enum values that have the high bit set.
#define	STACK_FILL_VALUE 0xCDCDCDCDU Define the magic value that is interpreted by IAR C-SPY's Stack View.
#define	RAMFUNC __ramfunc Define a generic RAM function identifier to a compiler specific one.
#define	NO_OPERATION () Define a generic no operation identifier to a compiler specific one.
#define	SET_REG_FIELD (reg, field, value) A convenience macro that makes it easy to change the field of a register to any unsigned value.
#define	SET_CMSIS_REG (reg, mask, value) A convenience macro that makes it easy to change a register using the provided mask(s) and value(s). Example: SET_CMSIS_REG(GPIO->P[1].CFGH, (_GPIO_P_CFGH_Px5_MASK | _GPIO_P_CFGH_Px6_MASK), (GPIO_P_CFGH_Px5_OUT | GPIO_P_CFGH_Px6_OUT));.
#define	SET_CMSIS_REG_FIELD (reg, field, value) A convenience macro that makes it easy to change the field of a register, as defined in CMSIS Device headers, to any unsigned value. Example using EM35xx: SET_CMSIS_REG_FIELD(GPIO->P[0].CFGL, GPIO_P_CFGL_Px0, _GPIO_P_CFGL_Px0_OUT);.

#define	simulatedTimePasses () Stub for code not running in simulation.
#define	simulatedTimePassesMs (x) Stub for code not running in simulation.
#define	simulatedSerialTimePasses () Stub for code not running in simulation.
#define	_HAL_USE_COMMON_DIVMOD_ undefined Use the Divide and Modulus Operations from platform-common.h.
#define	VAR_AT_SEGMENT (__variableDeclaration, __segmentName) Provide a portable way to specify the segment where a variable lives.
#define	STRINGIZE (X) Convinience macro for turning a token into a string.
#define	ALIGNMENT (X) Provide a portable way to align data.
#define	WEAK (__symbol) Provide a portable way to specify a symbol as weak.
#define	NO_INIT (__symbol) Provide a portable way to specify a non initialized symbol.
#define	STATIC_ASSERT (__condition, __errorstr) Provide a portable way to specify a compile time assert.

External Declarations

If the line below is uncommented we will use Ember memory APIs, otherwise, we will use the C Standard library (memset,memcpy,memmove) APIs.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) 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.
#define	PLATCOMMONOKTOINCLUDE undefined Include platform-common.h last to pick up defaults and common definitions.
#define	MAIN_FUNCTION_PARAMETERS void The kind of arguments the main function takes.
#define	MAIN_FUNCTION_ARGUMENTS undefined Include platform-common.h last to pick up defaults and common definitions.

Portable segment names

#define	__NO_INIT__ ".noinitlegacy" Portable segment names.
#define	__DEBUG_CHANNEL__ "DEBUG_CHANNEL" Portable segment names.
#define	__INTVEC__ ".intvec" Portable segment names.

#define __CSTACK__ "CSTACK"
Portable segment names.

#define __RESETINFO__ "RESETINFO"
Portable segment names.

#define __DATA_INIT__ ".data_init"
Portable segment names.

#define __DATA__ ".data"
Portable segment names.

#define __BSS__ ".bss"
Portable segment names.

#define __CONST__ ".rodata"
Portable segment names.

#define __TEXT__ ".text"
Portable segment names.

#define __TEXTRW_INIT__ ".textrw_init"
Portable segment names.

#define __TEXTRW__ ".textrw"
Portable segment names.

#define __AAT__ "AAT"
Portable segment names.

#define __BAT__ "BAT"
Portable segment names.

#define __BAT_INIT__ "BAT"
Portable segment names.

#define __FAT__ "FAT"
Portable segment names.

#define __RAT__ "RAT"
Portable segment names.

#define __SIMEE__ "SIMEE"
Portable segment names.

#define __PSSTORE__ "PSSTORE"
Portable segment names.

#define __LONGTOKEN__ "LONGTOKEN"
Portable segment names.

#define __EMHEAP__ "EMHEAP"
Portable segment names.

#define __GUARD_REGION__ "GUARD_REGION"
Portable segment names.

#define __DLIB_PERTHREAD_INIT__ "__DLIB_PERTHREAD_init"
Portable segment names.

#define __DLIB_PERTHREAD_INITIALIZED_DATA__ "DLIB_PERTHREAD_INITIALIZED_DATA"
Portable segment names.

#define __DLIB_PERTHREAD_ZERO_DATA__ "DLIB_PERTHREAD_ZERO_DATA"
Portable segment names.

#define __INTERNAL_STORAGE__ "INTERNAL_STORAGE"
Portable segment names.

#define __LOCKBITS_IN_MAINFLASH__ "LOCKBITS_IN_MAINFLASH"
Portable segment names.

#define __UNRETAINED_RAM__ "UNRETAINED_RAM"
Portable segment names.

#define _NO_INIT_SEGMENT_BEGIN __segment_begin(__NO_INIT__)
Portable segment names.

#define _DEBUG_CHANNEL_SEGMENT_BEGIN __segment_begin(__DEBUG_CHANNEL__)
Portable segment names.

#define _INTVEC_SEGMENT_BEGIN __segment_begin(__INTVEC__)
Portable segment names.

#define _CSTACK_SEGMENT_BEGIN __segment_begin(__CSTACK__)
Portable segment names.

#define _RESETINFO_SEGMENT_BEGIN __segment_begin(__RESETINFO__)
Portable segment names.

#define _DATA_INIT_SEGMENT_BEGIN __segment_begin(__DATA_INIT__)
Portable segment names.

#define _DATA_SEGMENT_BEGIN __segment_begin(__DATA__)
Portable segment names.

#define _BSS_SEGMENT_BEGIN __segment_begin(__BSS__)
Portable segment names.

#define _CONST_SEGMENT_BEGIN __segment_begin(__CONST__)
Portable segment names.

#define _TEXT_SEGMENT_BEGIN __segment_begin(__TEXT__)
Portable segment names.

#define _TEXTRW_INIT_SEGMENT_BEGIN __segment_begin(__TEXTRW_INIT__)
Portable segment names.

#define _TEXTRW_SEGMENT_BEGIN __segment_begin(__TEXTRW__)
Portable segment names.

#define _AAT_SEGMENT_BEGIN __segment_begin(__AAT__)
Portable segment names.

#define _BAT_SEGMENT_BEGIN __segment_begin(__BAT__)
Portable segment names.

#define _BAT_INIT_SEGMENT_BEGIN __segment_begin(__BAT_INIT__)
Portable segment names.

#define _FAT_SEGMENT_BEGIN __segment_begin(__FAT__)
Portable segment names.

#define _RAT_SEGMENT_BEGIN __segment_begin(__RAT__)
Portable segment names.

#define _SIMEE_SEGMENT_BEGIN __segment_begin(__SIMEE__)
Portable segment names.

#define _PSSTORE_SEGMENT_BEGIN __segment_begin(__PSSTORE__)
Portable segment names.

#define _LONGTOKEN_SEGMENT_BEGIN __segment_begin(__LONGTOKEN__)
Portable segment names.

#define _EMHEAP_SEGMENT_BEGIN __segment_begin(__EMHEAP__)
Portable segment names.

#define _GUARD_REGION_SEGMENT_BEGIN __segment_begin(__GUARD_REGION__)
Portable segment names.

#define _DLIB_PERTHREAD_INIT_SEGMENT_BEGIN __segment_begin(__DLIB_PERTHREAD_INIT__)
Portable segment names.

#define _DLIB_PERTHREAD_INITIALIZED_DATA_SEGMENT_BEGIN __segment_begin(__DLIB_PERTHREAD_INITIALIZED_DATA__)
Portable segment names.

#define _DLIB_PERTHREAD_ZERO_DATA_SEGMENT_BEGIN __segment_begin(__DLIB_PERTHREAD_ZERO_DATA__)
Portable segment names.

#define _INTERNAL_STORAGE_SEGMENT_BEGIN __segment_begin(__INTERNAL_STORAGE__)
Portable segment names.

#define _LOCKBITS_IN_MAINFLASH_SEGMENT_BEGIN __segment_begin(__LOCKBITS_IN_MAINFLASH__)
Portable segment names.

#define _UNRETAINED_RAM_SEGMENT_BEGIN __segment_begin(__UNRETAINED_RAM__)
Portable segment names.

#define _NO_INIT_SEGMENT_END __segment_end(__NO_INIT__)
Portable segment names.

#define _DEBUG_CHANNEL_SEGMENT_END __segment_end(__DEBUG_CHANNEL__)
Portable segment names.

#define _INTVEC_SEGMENT_END __segment_end(__INTVEC__)
Portable segment names.

#define _CSTACK_SEGMENT_END __segment_end(__CSTACK__)
Portable segment names.

#define _RESETINFO_SEGMENT_END __segment_end(__RESETINFO__)
Portable segment names.

#define _DATA_INIT_SEGMENT_END __segment_end(__DATA_INIT__)
Portable segment names.

#define _DATA_SEGMENT_END __segment_end(__DATA__)
Portable segment names.

#define _BSS_SEGMENT_END __segment_end(__BSS__)
Portable segment names.

#define _CONST_SEGMENT_END __segment_end(__CONST__)
Portable segment names.

#define _TEXT_SEGMENT_END __segment_end(__TEXT__)
Portable segment names.

#define _TEXTRW_INIT_SEGMENT_END __segment_end(__TEXTRW_INIT__)
Portable segment names.

#define _TEXTRW_SEGMENT_END __segment_end(__TEXTRW__)
Portable segment names.

#define _AAT_SEGMENT_END __segment_end(__AAT__)
Portable segment names.

#define _BAT_SEGMENT_END __segment_end(__BAT__)
Portable segment names.

#define _BAT_INIT_SEGMENT_END __segment_end(__BAT_INIT__)
Portable segment names.

#define _FAT_SEGMENT_END __segment_end(__FAT__)
Portable segment names.

#define _RAT_SEGMENT_END __segment_end(__RAT__)
Portable segment names.

#define _SIMEE_SEGMENT_END __segment_end(__SIMEE__)
Portable segment names.

#define _PSSTORE_SEGMENT_END __segment_end(__PSSTORE__)
Portable segment names.

#define _LONGTOKEN_SEGMENT_END __segment_end(__LONGTOKEN__)
Portable segment names.

#define _EMHEAP_SEGMENT_END __segment_end(__EMHEAP__)
Portable segment names.

#define _GUARD_REGION_SEGMENT_END __segment_end(__GUARD_REGION__)
Portable segment names.

#define _DLIB_PERTHREAD_INIT_SEGMENT_END __segment_end(__DLIB_PERTHREAD_INIT__)
Portable segment names.

#define _DLIB_PERTHREAD_INITIALIZED_DATA_SEGMENT_END __segment_end(__DLIB_PERTHREAD_INITIALIZED_DATA__)
Portable segment names.

#define _DLIB_PERTHREAD_ZERO_DATA_SEGMENT_END __segment_end(__DLIB_PERTHREAD_ZERO_DATA__)
Portable segment names.

#define _INTERNAL_STORAGE_SEGMENT_END __segment_end(__INTERNAL_STORAGE__)
Portable segment names.

#define _LOCKBITS_IN_MAINFLASH_SEGMENT_END __segment_end(__LOCKBITS_IN_MAINFLASH__)
Portable segment names.

#define _UNRETAINED_RAM_SEGMENT_END __segment_end(__UNRETAINED_RAM__)
Portable segment names.

#define _NO_INIT_SEGMENT_SIZE __segment_size(__NO_INIT__)
Portable segment names.

#define _DEBUG_CHANNEL_SEGMENT_SIZE __segment_size(__DEBUG_CHANNEL__)
Portable segment names.

#define _INTVEC_SEGMENT_SIZE __segment_size(__INTVEC__)
Portable segment names.

#define _CSTACK_SEGMENT_SIZE __segment_size(__CSTACK__)
Portable segment names.

#define _RESETINFO_SEGMENT_SIZE __segment_size(__RESETINFO__)
Portable segment names.

#define _DATA_INIT_SEGMENT_SIZE __segment_size(__DATA_INIT__)
Portable segment names.

#define _DATA_SEGMENT_SIZE __segment_size(__DATA__)
Portable segment names.

#define _BSS_SEGMENT_SIZE __segment_size(__BSS__)
Portable segment names.

#define _CONST_SEGMENT_SIZE __segment_size(__CONST__)
Portable segment names.

#define _TEXT_SEGMENT_SIZE __segment_size(__TEXT__)
Portable segment names.

#define _TEXTRW_INIT_SEGMENT_SIZE __segment_size(__TEXTRW_INIT__)
Portable segment names.

#define _TEXTRW_SEGMENT_SIZE __segment_size(__TEXTRW__)
Portable segment names.

#define _AAT_SEGMENT_SIZE __segment_size(__AAT__)
Portable segment names.

#define _BAT_SEGMENT_SIZE __segment_size(__BAT__)
Portable segment names.

#define _BAT_INIT_SEGMENT_SIZE __segment_size(__BAT_INIT__)
Portable segment names.

#define _FAT_SEGMENT_SIZE __segment_size(__FAT__)
Portable segment names.

#define _RAT_SEGMENT_SIZE __segment_size(__RAT__)
Portable segment names.

#define _SIMEE_SEGMENT_SIZE __segment_size(__SIMEE__)
Portable segment names.

#define _PSSTORE_SEGMENT_SIZE __segment_size(__PSSTORE__)
Portable segment names.

#define _LONGTOKEN_SEGMENT_SIZE __segment_size(__LONGTOKEN__)
Portable segment names.

#define _EMHEAP_SEGMENT_SIZE __segment_size(__EMHEAP__)
Portable segment names.

#define _GUARD_REGION_SEGMENT_SIZE __segment_size(__GUARD_REGION__)
Portable segment names.

#define _DLIB_PERTHREAD_INIT_SEGMENT_SIZE __segment_size(__DLIB_PERTHREAD_INIT__)
Portable segment names.

#define _DLIB_PERTHREAD_INITIALIZED_DATA_SEGMENT_SIZE __segment_size(__DLIB_PERTHREAD_INITIALIZED_DATA__)
Portable segment names.

#define _DLIB_PERTHREAD_ZERO_DATA_SEGMENT_SIZE __segment_size(__DLIB_PERTHREAD_ZERO_DATA__)
Portable segment names.

#define _INTERNAL_STORAGE_SEGMENT_SIZE __segment_size(__INTERNAL_STORAGE__)
Portable segment names.

#define _LOCKBITS_IN_MAINFLASH_SEGMENT_SIZE __segment_size(__LOCKBITS_IN_MAINFLASH__)
Portable segment names.

#define _UNRETAINED_RAM_SEGMENT_SIZE __segment_size(__UNRETAINED_RAM__)
Portable segment names.

Functions

void _executeBarrierInstructions(void)

Master Variable Types Documentation

boolean

```
typedef bool boolean
```

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

Definition at line `82` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

int8u

```
typedef unsigned char int8u
```

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

Definition at line `83` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

int8s

```
typedef signed char int8s
```

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

Definition at line `84` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

int16u

```
typedef unsigned short int16u
```

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

Definition at line `85` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

int16s

```
typedef signed short int16s
```

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

Definition at line `86` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

int32u

```
typedef unsigned int int32u
```

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

Definition at line `87` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

int32s

```
typedef signed int int32s
```

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

Definition at line `88` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

int64u

```
typedef unsigned long long int64u
```

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

Definition at line `89` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

int64s

```
typedef signed long long int64s
```

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

Definition at line `90` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

PointerType

```
typedef unsigned int PointerType
```

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

Definition at line `91` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

HAL_HAS_INT64

```
#define HAL_HAS_INT64
```

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

Definition at line `97` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_HAL_USE_COMMON_PGM_

```
#define _HAL_USE_COMMON_PGM_
```

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

Definition at line `102` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

Miscellaneous Macros Documentation

halInternalAssertFailed

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

A prototype definition for use by the assert macro. (see hal/micro/micro.h)

Parameters

N/A	filename	
N/A	linenumber	

Definition at line `152` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

halInternalResetWatchDog

```
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.

Parameters

N/A		

Definition at line `184` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

BIGENDIAN_CPU

```
#define BIGENDIAN_CPU
```

Value:

```
false
```

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 `115` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

NTOHS

```
#define NTOHS
```

Value:

```
(val)
```

Define intrinsics for NTOHL and NTOHS to save code space by making endian.c compile to nothing.

Definition at line `121` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

NTOHL

```
#define NTOHL
```

Value:

```
(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 `122` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

NO_STRIPPING

```
#define NO_STRIPPING
```

Value:

```
__root
```

A friendlier name for the compiler's intrinsic for not stripping.

Definition at line `128` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

EEPROM

```
#define EEPROM
```

Value:

```
errorerror
```

A friendlier name for the compiler's intrinsic for eeprom reference.

Definition at line `134` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

__SOURCEFILE__

```
#define __SOURCEFILE__
```

Value:

```
__FILE__
```

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 `143` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

assert

```
#define assert
```

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 `160` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

halResetWatchdog

```
#define halResetWatchdog
```

Value:

```
()
```

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 `191` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

UNUSED

```
#define UNUSED
```

Declare a variable as unused to avoid a warning. Has no effect in IAR builds.

Definition at line `198` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

SIGNED_ENUM

```
#define SIGNED_ENUM
```

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

Definition at line `203` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

STACK_FILL_VALUE

```
#define STACK_FILL_VALUE
```

Value:

```
0xCDCDCDCDU
```

Define the magic value that is interpreted by IAR C-SPY's Stack View.

Definition at line `208` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

RAMFUNC

```
#define RAMFUNC
```

Value:

```
__ramfunc
```

Define a generic RAM function identifier to a compiler specific one.

Definition at line `218` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

NO_OPERATION

```
#define NO_OPERATION
```

Value:

```
()
```

Define a generic no operation identifier to a compiler specific one.

Definition at line `224` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

SET_REG_FIELD

```
#define SET_REG_FIELD
```

Value:

```
do {                                    \
  reg = ((reg & (~field##_MASK))             \
        | ((((uint32_t) value) << field##_BIT) \
          & (field##_MASK)));              \
} while (0)
```

A convenience macro that makes it easy to change the field of a register to any unsigned value.

Definition at line `230` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

SET_CMSIS_REG

```
#define SET_CMSIS_REG
```

Value:

```
do {                               \
  reg = (((reg) & (~mask)) | (value)); \
} while (0)
```

A convenience macro that makes it easy to change a register using the provided mask(s) and value(s). Example: SET_CMSIS_REG(GPIO->P[1].CFGH, (_GPIO_P_CFGH_Px5_MASK | _GPIO_P_CFGH_Px6_MASK), (GPIO_P_CFGH_Px5_OUT | GPIO_P_CFGH_Px6_OUT));.

Definition at line `247` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

SET_CMSIS_REG_FIELD

```
#define SET_CMSIS_REG_FIELD
```

Value:

```
do {                                    \
  reg = ((reg & (~_##field##_MASK))       \
         | ((value << _##field##_SHIFT)     \
            & (_##field##_MASK)));         \
} while (0)
```

A convenience macro that makes it easy to change the field of a register, as defined in CMSIS Device headers, to any unsigned value. Example using EM35xx: SET_CMSIS_REG_FIELD(GPIO->P[0].CFGL, GPIO_P_CFGL_Px0, _GPIO_P_CFGL_Px0_OUT);.

Definition at line `258` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

simulatedTimePasses

```
#define simulatedTimePasses
```

Stub for code not running in simulation.

Definition at line `268` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

simulatedTimePassesMs

```
#define simulatedTimePassesMs
```

Stub for code not running in simulation.

Definition at line `273` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

simulatedSerialTimePasses

```
#define simulatedSerialTimePasses
```

Stub for code not running in simulation.

Definition at line `278` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_HAL_USE_COMMON_DIVMOD_

```
#define _HAL_USE_COMMON_DIVMOD_
```

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

Definition at line `283` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

VAR_AT_SEGMENT

```
#define VAR_AT_SEGMENT
```

Value:

```
(__variableDeclaration, __segmentName)
```

Provide a portable way to specify the segment where a variable lives.

Definition at line `289` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

STRINGIZE

```
#define STRINGIZE
```

Value:

```
(X)
```

Convinience macro for turning a token into a string.

Definition at line `295` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

ALIGNMENT

```
#define ALIGNMENT
```

Value:

```
(X)
```

Provide a portable way to align data.

Definition at line `300` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

WEAK

```
#define WEAK
```

Value:

```
(__symbol)
```

Provide a portable way to specify a symbol as weak.

Definition at line `306` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

NO_INIT

```
#define NO_INIT
```

Value:

```
(__symbol)
```

Provide a portable way to specify a non initialized symbol.

Definition at line `312` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

STATIC_ASSERT

```
#define STATIC_ASSERT
```

Value:

```
(__condition, __errorstr)
```

Provide a portable way to specify a compile time assert.

Definition at line `318` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

External Declarations Documentation

abs

```
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

N/A	I	An integer.

Returns

- A nonnegative integer.

Definition at line `527` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

PLATCOMMONOKTOINCLUDE

```
#define PLATCOMMONOKTOINCLUDE
```

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

Definition at line `536` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

MAIN_FUNCTION_PARAMETERS

```
#define MAIN_FUNCTION_PARAMETERS
```

Value:

```
void
```

The kind of arguments the main function takes.

Definition at line `543` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

MAIN_FUNCTION_ARGUMENTS

```
#define MAIN_FUNCTION_ARGUMENTS
```

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

Definition at line `544` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

Portable segment names Documentation

__NO_INIT__

```
#define __NO_INIT__
```

Value:

```
".noinitlegacy"
```

Portable segment names.

Definition at line `341` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

__DEBUG_CHANNEL__

```
#define __DEBUG_CHANNEL__
```

Value:

```
"DEBUG_CHANNEL"
```

Portable segment names.

Definition at line `344` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

__INTVEC__

```
#define __INTVEC__
```

Value:

```
".intvec"
```

Portable segment names.

Definition at line `345` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

__CSTACK__

```
#define __CSTACK__
```

Value:

```
"CSTACK"
```

Portable segment names.

Definition at line 346 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h

__RESETINFO__

```
#define __RESETINFO__
```

Value:

```
"RESETINFO"
```

Portable segment names.

Definition at line 347 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h

__DATA_INIT__

```
#define __DATA_INIT__
```

Value:

```
".data_init"
```

Portable segment names.

Definition at line 348 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h

__DATA__

```
#define __DATA__
```

Value:

```
".data"
```

Portable segment names.

Definition at line 349 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h

__BSS__

```
#define __BSS__
```

Value:

```
".bss"
```

Portable segment names.

Definition at line `350` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

__CONST__

```
#define __CONST__
```

Value:

```
".rodata"
```

Portable segment names.

Definition at line `351` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

__TEXT__

```
#define __TEXT__
```

Value:

```
".text"
```

Portable segment names.

Definition at line `352` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

__TEXTRW_INIT__

```
#define __TEXTRW_INIT__
```

Value:

```
".textrw_init"
```

Portable segment names.

Definition at line `353` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

__TEXTRW__

```
#define __TEXTRW__
```

Value:

```
".textrw"
```

Portable segment names.

Definition at line `354` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

__AAT__

```
#define __AAT__
```

Value:

```
"AAT"
```

Portable segment names.

Definition at line `355` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

__BAT__

```
#define __BAT__
```

Value:

```
"BAT"
```

Portable segment names.

Definition at line `356` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

__BAT_INIT__

```
#define __BAT_INIT__
```

Value:

```
"BAT"
```

Portable segment names.

Definition at line `357` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

__FAT__

```
#define __FAT__
```

Value:

```
"FAT"
```

Portable segment names.

Definition at line `358` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

__RAT__

```
#define __RAT__
```

Value:

```
"RAT"
```

Portable segment names.

Definition at line `359` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

__SIMEE__

```
#define __SIMEE__
```

Value:

```
"SIMEE"
```

Portable segment names.

Definition at line `360` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

__PSSTORE__

```
#define __PSSTORE__
```

Value:

```
"PSSTORE"
```

Portable segment names.

Definition at line `361` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

__LONGTOKEN__

```
#define __LONGTOKEN__
```

Value:

```
"LONGTOKEN"
```

Portable segment names.

Definition at line `362` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

__EMHEAP__

```
#define __EMHEAP__
```

Value:

```
"EMHEAP"
```

Portable segment names.

Definition at line `363` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

__GUARD_REGION__

```
#define __GUARD_REGION__
```

Value:

```
"GUARD_REGION"
```

Portable segment names.

Definition at line `364` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

__DLIB_PERTHREAD_INIT__

```
#define __DLIB_PERTHREAD_INIT__
```

Value:

```
"__DLIB_PERTHREAD_init"
```

Portable segment names.

Definition at line `365` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

__DLIB_PERTHREAD_INITIALIZED_DATA__

```
#define __DLIB_PERTHREAD_INITIALIZED_DATA__
```

Value:

```
"DLIB_PERTHREAD_INITIALIZED_DATA"
```

Portable segment names.

Definition at line `366` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

__DLIB_PERTHREAD_ZERO_DATA__

```
#define __DLIB_PERTHREAD_ZERO_DATA__
```

Value:

```
"DLIB_PERTHREAD_ZERO_DATA"
```

Portable segment names.

Definition at line `367` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

__INTERNAL_STORAGE__

```
#define __INTERNAL_STORAGE__
```

Value:

```
"INTERNAL_STORAGE"
```

Portable segment names.

Definition at line `368` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

__LOCKBITS_IN_MAINFLASH__

```
#define __LOCKBITS_IN_MAINFLASH__
```

Value:

```
"LOCKBITS_IN_MAINFLASH"
```

Portable segment names.

Definition at line `369` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

__UNRETAINED_RAM__

```
#define __UNRETAINED_RAM__
```

Value:

```
"UNRETAINED_RAM"
```

Portable segment names.

Definition at line `370` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_NO_INIT_SEGMENT_BEGIN

```
#define _NO_INIT_SEGMENT_BEGIN
```

Value:

```
__segment_begin(__NO_INIT__)
```

Portable segment names.

Definition at line `408` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_DEBUG_CHANNEL_SEGMENT_BEGIN

```
#define _DEBUG_CHANNEL_SEGMENT_BEGIN
```

Value:

```
__segment_begin(__DEBUG_CHANNEL__)
```

Portable segment names.

Definition at line `409` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_INTVEC_SEGMENT_BEGIN

```
#define _INTVEC_SEGMENT_BEGIN
```

Value:

```
__segment_begin(__INTVEC__)
```

Portable segment names.

Definition at line `410` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_CSTACK_SEGMENT_BEGIN

```
#define _CSTACK_SEGMENT_BEGIN
```

Value:

```
__segment_begin(__CSTACK__)
```

Portable segment names.

Definition at line `411` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_RESETINFO_SEGMENT_BEGIN

```
#define _RESETINFO_SEGMENT_BEGIN
```

Value:

```
__segment_begin(__RESETINFO__)
```

Portable segment names.

Definition at line `412` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_DATA_INIT_SEGMENT_BEGIN

```
#define _DATA_INIT_SEGMENT_BEGIN
```

Value:

```
__segment_begin(__DATA_INIT__)
```

Portable segment names.

Definition at line `413` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_DATA_SEGMENT_BEGIN

```
#define _DATA_SEGMENT_BEGIN
```

Value:

```
__segment_begin(__DATA__)
```

Portable segment names.

Definition at line `414` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_BSS_SEGMENT_BEGIN

```
#define _BSS_SEGMENT_BEGIN
```

Value:

```
__segment_begin(__BSS__)
```

Portable segment names.

Definition at line `415` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_CONST_SEGMENT_BEGIN

```
#define _CONST_SEGMENT_BEGIN
```

Value:

```
__segment_begin(__CONST__)
```

Portable segment names.

Definition at line `416` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_TEXT_SEGMENT_BEGIN

```
#define _TEXT_SEGMENT_BEGIN
```

Value:

```
__segment_begin(__TEXT__)
```

Portable segment names.

Definition at line `417` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_TEXTRW_INIT_SEGMENT_BEGIN

```
#define _TEXTRW_INIT_SEGMENT_BEGIN
```

Value:

```
__segment_begin(__TEXTRW_INIT__)
```

Portable segment names.

Definition at line `418` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_TEXTRW_SEGMENT_BEGIN

```
#define _TEXTRW_SEGMENT_BEGIN
```

Value:

```
__segment_begin(__TEXTRW__)
```

Portable segment names.

Definition at line `419` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_AAT_SEGMENT_BEGIN

```
#define _AAT_SEGMENT_BEGIN
```

Value:

```
__segment_begin(__AAT__)
```

Portable segment names.

Definition at line `420` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_BAT_SEGMENT_BEGIN

```
#define _BAT_SEGMENT_BEGIN
```

Value:

```
__segment_begin(__BAT__)
```

Portable segment names.

Definition at line `421` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_BAT_INIT_SEGMENT_BEGIN

```
#define _BAT_INIT_SEGMENT_BEGIN
```

Value:

```
__segment_begin(__BAT_INIT__)
```

Portable segment names.

Definition at line `422` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_FAT_SEGMENT_BEGIN

```
#define _FAT_SEGMENT_BEGIN
```

Value:

```
__segment_begin(__FAT__)
```

Portable segment names.

Definition at line `423` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_RAT_SEGMENT_BEGIN

```
#define _RAT_SEGMENT_BEGIN
```

Value:

```
__segment_begin(__RAT__)
```

Portable segment names.

Definition at line `424` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_SIMEE_SEGMENT_BEGIN

```
#define _SIMEE_SEGMENT_BEGIN
```

Value:

```
__segment_begin(__SIMEE__)
```

Portable segment names.

Definition at line `425` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_PSSTORE_SEGMENT_BEGIN

```
#define _PSSTORE_SEGMENT_BEGIN
```

Value:

```
__segment_begin(__PSSTORE__)
```

Portable segment names.

Definition at line `426` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_LONGTOKEN_SEGMENT_BEGIN

```
#define _LONGTOKEN_SEGMENT_BEGIN
```

Value:

```
__segment_begin(__LONGTOKEN__)
```

Portable segment names.

Definition at line `427` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_EMHEAP_SEGMENT_BEGIN

```
#define _EMHEAP_SEGMENT_BEGIN
```

Value:

```
__segment_begin(__EMHEAP__)
```

Portable segment names.

Definition at line `428` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_GUARD_REGION_SEGMENT_BEGIN

```
#define _GUARD_REGION_SEGMENT_BEGIN
```

Value:

```
__segment_begin(__GUARD_REGION__)
```

Portable segment names.

Definition at line `429` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_DLIB_PERTHREAD_INIT_SEGMENT_BEGIN

```
#define _DLIB_PERTHREAD_INIT_SEGMENT_BEGIN
```

Value:

```
__segment_begin(__DLIB_PERTHREAD_INIT__)
```

Portable segment names.

Definition at line `430` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_DLIB_PERTHREAD_INITIALIZED_DATA_SEGMENT_BEGIN

```
#define _DLIB_PERTHREAD_INITIALIZED_DATA_SEGMENT_BEGIN
```

Value:

```
__segment_begin(__DLIB_PERTHREAD_INITIALIZED_DATA__)
```

Portable segment names.

Definition at line `431` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_DLIB_PERTHREAD_ZERO_DATA_SEGMENT_BEGIN

```
#define _DLIB_PERTHREAD_ZERO_DATA_SEGMENT_BEGIN
```

Value:

```
__segment_begin(__DLIB_PERTHREAD_ZERO_DATA__)
```

Portable segment names.

Definition at line `432` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_INTERNAL_STORAGE_SEGMENT_BEGIN

```
#define _INTERNAL_STORAGE_SEGMENT_BEGIN
```

Value:

```
__segment_begin(__INTERNAL_STORAGE__)
```

Portable segment names.

Definition at line `433` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_LOCKBITS_IN_MAINFLASH_SEGMENT_BEGIN

```
#define _LOCKBITS_IN_MAINFLASH_SEGMENT_BEGIN
```

Value:

```
__segment_begin(__LOCKBITS_IN_MAINFLASH__)
```

Portable segment names.

Definition at line `434` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_UNRETAINED_RAM_SEGMENT_BEGIN

```
#define _UNRETAINED_RAM_SEGMENT_BEGIN
```

Value:

```
__segment_begin(__UNRETAINED_RAM__)
```

Portable segment names.

Definition at line `435` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_NO_INIT_SEGMENT_END

```
#define _NO_INIT_SEGMENT_END
```

Value:

```
__segment_end(__NO_INIT__)
```

Portable segment names.

Definition at line `437` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_DEBUG_CHANNEL_SEGMENT_END

```
#define _DEBUG_CHANNEL_SEGMENT_END
```

Value:

```
__segment_end(__DEBUG_CHANNEL__)
```

Portable segment names.

Definition at line `438` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_INTVEC_SEGMENT_END

```
#define _INTVEC_SEGMENT_END
```

Value:

```
__segment_end(__INTVEC__)
```

Portable segment names.

Definition at line `439` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_CSTACK_SEGMENT_END

```
#define _CSTACK_SEGMENT_END
```

Value:

```
__segment_end(__CSTACK__)
```

Portable segment names.

Definition at line `440` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_RESETINFO_SEGMENT_END

```
#define _RESETINFO_SEGMENT_END
```

Value:

```
__segment_end(__RESETINFO__)
```

Portable segment names.

Definition at line `441` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_DATA_INIT_SEGMENT_END

```
#define _DATA_INIT_SEGMENT_END
```

Value:

```
__segment_end(__DATA_INIT__)
```

Portable segment names.

Definition at line `442` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_DATA_SEGMENT_END

```
#define _DATA_SEGMENT_END
```

Value:

```
__segment_end(__DATA__)
```

Portable segment names.

Definition at line `443` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_BSS_SEGMENT_END

```
#define _BSS_SEGMENT_END
```

Value:

```
__segment_end(__BSS__)
```

Portable segment names.

Definition at line `444` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_CONST_SEGMENT_END

```
#define _CONST_SEGMENT_END
```

Value:

```
__segment_end(__CONST__)
```

Portable segment names.

Definition at line `445` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_TEXT_SEGMENT_END

```
#define _TEXT_SEGMENT_END
```

Value:

```
__segment_end(__TEXT__)
```

Portable segment names.

Definition at line `446` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_TEXTRW_INIT_SEGMENT_END

```
#define _TEXTRW_INIT_SEGMENT_END
```

Value:

```
__segment_end(__TEXTRW_INIT__)
```

Portable segment names.

Definition at line `447` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_TEXTRW_SEGMENT_END

```
#define _TEXTRW_SEGMENT_END
```

Value:

```
__segment_end(__TEXTRW__)
```

Portable segment names.

Definition at line `448` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_AAT_SEGMENT_END

```
#define _AAT_SEGMENT_END
```

Value:

```
__segment_end(__AAT__)
```

Portable segment names.

Definition at line `449` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_BAT_SEGMENT_END

```
#define _BAT_SEGMENT_END
```

Value:

```
__segment_end(__BAT__)
```

Portable segment names.

Definition at line `450` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_BAT_INIT_SEGMENT_END

```
#define _BAT_INIT_SEGMENT_END
```

Value:

```
__segment_end(__BAT_INIT__)
```

Portable segment names.

Definition at line `451` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_FAT_SEGMENT_END

```
#define _FAT_SEGMENT_END
```

Value:

```
__segment_end(__FAT__)
```

Portable segment names.

Definition at line `452` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_RAT_SEGMENT_END

```
#define _RAT_SEGMENT_END
```

Value:

```
__segment_end(__RAT__)
```

Portable segment names.

Definition at line `453` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_SIMEE_SEGMENT_END

```
#define _SIMEE_SEGMENT_END
```

Value:

```
__segment_end(__SIMEE__)
```

Portable segment names.

Definition at line `454` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_PSSTORE_SEGMENT_END

```
#define _PSSTORE_SEGMENT_END
```

Value:

```
__segment_end(__PSSTORE__)
```

Portable segment names.

Definition at line `455` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_LONGTOKEN_SEGMENT_END

```
#define _LONGTOKEN_SEGMENT_END
```

Value:

```
__segment_end(__LONGTOKEN__)
```

Portable segment names.

Definition at line `456` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_EMHEAP_SEGMENT_END

```
#define _EMHEAP_SEGMENT_END
```

Value:

```
__segment_end(__EMHEAP__)
```

Portable segment names.

Definition at line `457` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_GUARD_REGION_SEGMENT_END

```
#define _GUARD_REGION_SEGMENT_END
```

Value:

```
__segment_end(__GUARD_REGION__)
```

Portable segment names.

Definition at line `458` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_DLIB_PERTHREAD_INIT_SEGMENT_END

```
#define _DLIB_PERTHREAD_INIT_SEGMENT_END
```

Value:

```
__segment_end(__DLIB_PERTHREAD_INIT__)
```

Portable segment names.

Definition at line `459` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_DLIB_PERTHREAD_INITIALIZED_DATA_SEGMENT_END

```
#define _DLIB_PERTHREAD_INITIALIZED_DATA_SEGMENT_END
```

Value:

```
__segment_end(__DLIB_PERTHREAD_INITIALIZED_DATA__)
```

Portable segment names.

Definition at line `460` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_DLIB_PERTHREAD_ZERO_DATA_SEGMENT_END

```
#define _DLIB_PERTHREAD_ZERO_DATA_SEGMENT_END
```

Value:

```
__segment_end(__DLIB_PERTHREAD_ZERO_DATA__)
```

Portable segment names.

Definition at line `461` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_INTERNAL_STORAGE_SEGMENT_END

```
#define _INTERNAL_STORAGE_SEGMENT_END
```

Value:

```
__segment_end(__INTERNAL_STORAGE__)
```

Portable segment names.

Definition at line `462` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_LOCKBITS_IN_MAINFLASH_SEGMENT_END

```
#define _LOCKBITS_IN_MAINFLASH_SEGMENT_END
```

Value:

```
__segment_end(__LOCKBITS_IN_MAINFLASH__)
```

Portable segment names.

Definition at line `463` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_UNRETAINED_RAM_SEGMENT_END

```
#define _UNRETAINED_RAM_SEGMENT_END
```

Value:

```
__segment_end(__UNRETAINED_RAM__)
```

Portable segment names.

Definition at line `464` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_NO_INIT_SEGMENT_SIZE

```
#define _NO_INIT_SEGMENT_SIZE
```

Value:

```
__segment_size(__NO_INIT__)
```

Portable segment names.

Definition at line `466` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_DEBUG_CHANNEL_SEGMENT_SIZE

```
#define _DEBUG_CHANNEL_SEGMENT_SIZE
```

Value:

```
__segment_size(__DEBUG_CHANNEL__)
```

Portable segment names.

Definition at line `467` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_INTVEC_SEGMENT_SIZE

```
#define _INTVEC_SEGMENT_SIZE
```

Value:

```
__segment_size(__INTVEC__)
```

Portable segment names.

Definition at line `468` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_CSTACK_SEGMENT_SIZE

```
#define _CSTACK_SEGMENT_SIZE
```

Value:

```
__segment_size(__CSTACK__)
```

Portable segment names.

Definition at line `469` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_RESETINFO_SEGMENT_SIZE

```
#define _RESETINFO_SEGMENT_SIZE
```

Value:

```
__segment_size(__RESETINFO__)
```

Portable segment names.

Definition at line `470` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_DATA_INIT_SEGMENT_SIZE

```
#define _DATA_INIT_SEGMENT_SIZE
```

Value:

```
__segment_size(__DATA_INIT__)
```

Portable segment names.

Definition at line `471` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_DATA_SEGMENT_SIZE

```
#define _DATA_SEGMENT_SIZE
```

Value:

```
__segment_size(__DATA__)
```

Portable segment names.

Definition at line `472` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_BSS_SEGMENT_SIZE

```
#define _BSS_SEGMENT_SIZE
```

Value:

```
__segment_size(__BSS__)
```

Portable segment names.

Definition at line `473` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_CONST_SEGMENT_SIZE

```
#define _CONST_SEGMENT_SIZE
```

Value:

```
__segment_size(__CONST__)
```

Portable segment names.

Definition at line `474` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_TEXT_SEGMENT_SIZE

```
#define _TEXT_SEGMENT_SIZE
```

Value:

```
__segment_size(__TEXT__)
```

Portable segment names.

Definition at line `475` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_TEXTRW_INIT_SEGMENT_SIZE

```
#define _TEXTRW_INIT_SEGMENT_SIZE
```

Value:

```
__segment_size(__TEXTRW_INIT__)
```

Portable segment names.

Definition at line `476` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_TEXTRW_SEGMENT_SIZE

```
#define _TEXTRW_SEGMENT_SIZE
```

Value:

```
__segment_size(__TEXTRW__)
```

Portable segment names.

Definition at line `477` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_AAT_SEGMENT_SIZE

```
#define _AAT_SEGMENT_SIZE
```

Value:

```
__segment_size(__AAT__)
```

Portable segment names.

Definition at line `478` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_BAT_SEGMENT_SIZE

```
#define _BAT_SEGMENT_SIZE
```

Value:

```
__segment_size(__BAT__)
```

Portable segment names.

Definition at line `479` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_BAT_INIT_SEGMENT_SIZE

```
#define _BAT_INIT_SEGMENT_SIZE
```

Value:

```
__segment_size(__BAT_INIT__)
```

Portable segment names.

Definition at line `480` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_FAT_SEGMENT_SIZE

```
#define _FAT_SEGMENT_SIZE
```

Value:

```
__segment_size(__FAT__)
```

Portable segment names.

Definition at line `481` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_RAT_SEGMENT_SIZE

```
#define _RAT_SEGMENT_SIZE
```

Value:

```
__segment_size(__RAT__)
```

Portable segment names.

Definition at line `482` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_SIMEE_SEGMENT_SIZE

```
#define _SIMEE_SEGMENT_SIZE
```

Value:

```
__segment_size(__SIMEE__)
```

Portable segment names.

Definition at line `483` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_PSSTORE_SEGMENT_SIZE

```
#define _PSSTORE_SEGMENT_SIZE
```

Value:

```
__segment_size(__PSSTORE__)
```

Portable segment names.

Definition at line 484 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h

_LONGTOKEN_SEGMENT_SIZE

```
#define _LONGTOKEN_SEGMENT_SIZE
```

Value:

```
__segment_size(__LONGTOKEN__)
```

Portable segment names.

Definition at line 485 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h

_EMHEAP_SEGMENT_SIZE

```
#define _EMHEAP_SEGMENT_SIZE
```

Value:

```
__segment_size(__EMHEAP__)
```

Portable segment names.

Definition at line 486 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h

_GUARD_REGION_SEGMENT_SIZE

```
#define _GUARD_REGION_SEGMENT_SIZE
```

Value:

```
__segment_size(__GUARD_REGION__)
```

Portable segment names.

Definition at line 487 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h

_DLIB_PERTHREAD_INIT_SEGMENT_SIZE

```
#define _DLIB_PERTHREAD_INIT_SEGMENT_SIZE
```

Value:

```
__segment_size(__DLIB_PERTHREAD_INIT__)
```

Portable segment names.

Definition at line `488` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_DLIB_PERTHREAD_INITIALIZED_DATA_SEGMENT_SIZE

```
#define _DLIB_PERTHREAD_INITIALIZED_DATA_SEGMENT_SIZE
```

Value:

```
__segment_size(__DLIB_PERTHREAD_INITIALIZED_DATA__)
```

Portable segment names.

Definition at line `489` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_DLIB_PERTHREAD_ZERO_DATA_SEGMENT_SIZE

```
#define _DLIB_PERTHREAD_ZERO_DATA_SEGMENT_SIZE
```

Value:

```
__segment_size(__DLIB_PERTHREAD_ZERO_DATA__)
```

Portable segment names.

Definition at line `490` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_INTERNAL_STORAGE_SEGMENT_SIZE

```
#define _INTERNAL_STORAGE_SEGMENT_SIZE
```

Value:

```
__segment_size(__INTERNAL_STORAGE__)
```

Portable segment names.

Definition at line `491` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_LOCKBITS_IN_MAINFLASH_SEGMENT_SIZE

```
#define _LOCKBITS_IN_MAINFLASH_SEGMENT_SIZE
```

Value:

```
__segment_size(__LOCKBITS_IN_MAINFLASH__)
```

Portable segment names.

Definition at line `492` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

_UNRETAINED_RAM_SEGMENT_SIZE

```
#define _UNRETAINED_RAM_SEGMENT_SIZE
```

Value:

```
__segment_size(__UNRETAINED_RAM__)
```

Portable segment names.

Definition at line `493` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

Function Documentation

_executeBarrierInstructions

```
void _executeBarrierInstructions (void)
```

Parameters

N/A		

Definition at line `501` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/cortexm3/compiler/iar.h`

Common PLATFORM_HEADER Configuration

Common PLATFORM_HEADER Configuration

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.

Generic Types

#define	TRUE 1 An alias for one, used for clarity.
#define	FALSE 0 An alias for zero, used for clarity.
#define	NULL ((void *)0) The null pointer.

Bit Manipulation Macros

#define	BIT (x) Useful to reference a single bit of a byte.
#define	BIT32 (x) Useful to reference a single bit of an uint32_t type.
#define	SETBIT (reg, bit) Sets `bit` in the `reg` register or byte.
#define	SETBITS (reg, bits) Sets the bits in the `reg` register or the byte as specified in the bitmask `bits`.
#define	CLEARBIT (reg, bit) Clears a bit in the `reg` register or byte.
#define	CLEARBITS (reg, bits) Clears the bits in the `reg` register or byte as specified in the bitmask `bits`.
#define	READBIT (reg, bit) Returns the value of `bit` within the register or byte `reg`.
#define	READBITS (reg, bits) Returns the value of the bitmask `bits` within the register or byte `reg`.

Byte Manipulation Macros

#define	LOW_BYTE (n) Returns the low byte of the 16-bit value `n` as an `uint8_t`.

#define	HIGH_BYTE (n) Returns the high byte of the 16-bit value `n` as an `uint8_t`.
#define	HIGH_LOW_TO_INT (high, low) Returns the value built from the two `uint8_t` values `high` and `low`.
#define	INT8U_TO_INT32U (byte3, byte2, byte1, byte0) Returns the value built from the four `uint8_t` as an `uint32_t`.
#define	BYTE_0 (n) Returns the low byte of the 32-bit value `n` as an `uint8_t`.
#define	BYTE_1 (n) Returns the second byte of the 32-bit value `n` as an `uint8_t`.
#define	BYTE_2 (n) Returns the third byte of the 32-bit value `n` as an `uint8_t`.
#define	BYTE_3 (n) Returns the high byte of the 32-bit value `n` as an `uint8_t`.
#define	BYTE_4 (n) Returns the fifth byte of the 64-bit value `n` as an `uint8_t`.
#define	BYTE_5 (n) Returns the sixth byte of the 64-bit value `n` as an `uint8_t`.
#define	BYTE_6 (n) Returns the seventh byte of the 64-bit value `n` as an `uint8_t`.
#define	BYTE_7 (n) Returns the high byte of the 64-bit value `n` as an `uint8_t`.
#define	COUNTOF (a) Returns the number of entries in an array.

Time Manipulation Macros

#define	elapsedTimeInt8u (oldTime, newTime) Returns the elapsed time between two 8 bit values. Result may not be valid if the time samples differ by more than 127.
#define	elapsedTimeInt16u (oldTime, newTime) Returns the elapsed time between two 16 bit values. Result may not be valid if the time samples differ by more than 32767.
#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.
#define	MAX_INT8U_VALUE (0xFF) Returns true if t1 is greater than t2. Can only account for 1 wrap around of the variable before it is wrong.
#define	HALF_MAX_INT8U_VALUE (0×80) Returns the elapsed time between two 8 bit values. Result may not be valid if the time samples differ by more than 127.
#define	timeGTorEqualInt8u (t1, t2) Returns the elapsed time between two 8 bit values. Result may not be valid if the time samples differ by more than 127.
#define	MAX_INT16U_VALUE (0xFFFF) Returns true if t1 is greater than t2. Can only account for 1 wrap around of the variable before it is wrong.

#define HALF_MAX_INT16U_VALUE (0x8000)
Returns the elapsed time between two 8 bit values. Result may not be valid if the time samples differ by more than 127.

#define timeGTorEqualInt16u (t1, t2)
Returns the elapsed time between two 8 bit values. Result may not be valid if the time samples differ by more than 127.

#define MAX_INT32U_VALUE (0xFFFFFFFFUL)
Returns true if t1 is greater than t2. Can only account for 1 wrap around of the variable before it is wrong.

#define HALF_MAX_INT32U_VALUE (0x80000000UL)
Returns the elapsed time between two 8 bit values. Result may not be valid if the time samples differ by more than 127.

#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.

Miscellaneous Macros

#define UNUSED_VAR (x)

#define DEBUG_LEVEL BASIC_DEBUG
Set debug level based on whether DEBUG or DEBUG_STRIPPED are defined.

#define STATIC_ASSERT (__condition, __errorstr)
Disable static assertions on compilers that don't support them.

Macros

#define MEMSET (d, v, l)
Friendly convenience macro pointing to the C Stdlib functions.

#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 Documentation

TRUE

```
#define TRUE
```

Value:

```
1
```

An alias for one, used for clarity.

Definition at line 210 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h

FALSE

```
#define FALSE
```

Value:

```
0
```

An alias for zero, used for clarity.

Definition at line 215 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h

NULL

```
#define NULL
```

Value:

```
((void *)0)
```

The null pointer.

Definition at line 222 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h

Bit Manipulation Macros Documentation

BIT

```
#define BIT
```

Value:

```
(x)
```

Useful to reference a single bit of a byte.

Definition at line 235 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h

BIT32

```
#define BIT32
```

Value:

```
(x)
```

Useful to reference a single bit of an uint32_t type.

Definition at line 240 of file /mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h

SETBIT

```
#define SETBIT
```

Value:

```
(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 `247` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

SETBITS

```
#define SETBITS
```

Value:

```
(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 `254` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

CLEARBIT

```
#define CLEARBIT
```

Value:

```
(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 `261` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

CLEARBITS

```
#define CLEARBITS
```

Value:

```
(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 `268` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

READBIT

```
#define READBIT
```

Value:

```
(reg, bit)
```

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

Definition at line `273` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

READBITS

```
#define READBITS
```

Value:

```
(reg, bits)
```

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

Definition at line `279` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

Byte Manipulation Macros Documentation

LOW_BYTE

```
#define LOW_BYTE
```

Value:

```
(n)
```

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

Definition at line `293` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

HIGH_BYTE

```
#define HIGH_BYTE
```

Value:

```
(n)
```

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

Definition at line `298` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

HIGH_LOW_TO_INT

```
#define HIGH_LOW_TO_INT
```

Value:

```
(       \
((uint16_t) (((uint16_t) (high)) << 8)) \
+ ((uint16_t) ((low) & 0xFFu))          \
)
```

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

Definition at line `304` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

INT8U_TO_INT32U

```
#define INT8U_TO_INT32U
```

Value:

```
( \
(((uint32_t) (byte3)) << 24)                  \
+ (((uint32_t) (byte2)) << 16)                \
+ (((uint32_t) (byte1)) << 8)                 \
+ ((uint32_t) ((byte0) & 0xFFu))              \
)
```

Returns the value built from the four `uint8_t` as an `uint32_t`.

Definition at line `312` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

BYTE_0

```
#define BYTE_0
```

Value:

```
(n)
```

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

Definition at line `322` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

BYTE_1

```
#define BYTE_1
```

Value:

```
(n)
```

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

Definition at line `327` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

BYTE_2

```
#define BYTE_2
```

Value:

```
(n)
```

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

Definition at line `332` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

BYTE_3

```
#define BYTE_3
```

Value:

```
(n)
```

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

Definition at line `337` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

BYTE_4

```
#define BYTE_4
```

Value:

```
(n)
```

Returns the fifth byte of the 64-bit value `n` as an `uint8_t`.

Definition at line `342` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

BYTE_5

```
#define BYTE_5
```

Value:

```
(n)
```

Returns the sixth byte of the 64-bit value `n` as an `uint8_t`.

Definition at line `347` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

BYTE_6

```
#define BYTE_6
```

Value:

```
(n)
```

Returns the seventh byte of the 64-bit value `n` as an `uint8_t`.

Definition at line `352` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

BYTE_7

```
#define BYTE_7
```

Value:

```
(n)
```

Returns the high byte of the 64-bit value `n` as an `uint8_t`.

Definition at line `357` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

COUNTOF

```
#define COUNTOF
```

Value:

```
(a)
```

Returns the number of entries in an array.

Definition at line `362` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

Time Manipulation Macros Documentation

elapsedTimeInt8u

```
#define elapsedTimeInt8u
```

Value:

```
(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 `377` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

elapsedTimeInt16u

```
#define elapsedTimeInt16u
```

Value:

```
(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 `384` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

elapsedTimeInt32u

```
#define elapsedTimeInt32u
```

Value:

```
(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 `391` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

MAX_INT8U_VALUE

```
#define MAX_INT8U_VALUE
```

Value:

```
(0xFF)
```

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 `398` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

HALF_MAX_INT8U_VALUE

```
#define HALF_MAX_INT8U_VALUE
```

Value:

```
(0×80)
```

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 `399` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

timeGTorEqualInt8u

```
#define timeGTorEqualInt8u
```

Value:

```
(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 `400` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

MAX_INT16U_VALUE

```
#define MAX_INT16U_VALUE
```

Value:

```
(0xFFFF)
```

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 `407` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

HALF_MAX_INT16U_VALUE

```
#define HALF_MAX_INT16U_VALUE
```

Value:

```
(0x8000)
```

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 `408` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

timeGTorEqualInt16u

```
#define timeGTorEqualInt16u
```

Value:

```
(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 `409` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

MAX_INT32U_VALUE

```
#define MAX_INT32U_VALUE
```

Value:

```
(0xFFFFFFFFUL)
```

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 `416` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

HALF_MAX_INT32U_VALUE

```
#define HALF_MAX_INT32U_VALUE
```

Value:

```
(0x80000000UL)
```

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 `417` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

timeGTorEqualInt32u

```
#define timeGTorEqualInt32u
```

Value:

```
(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 `418` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

Miscellaneous Macros Documentation

UNUSED_VAR

```
#define UNUSED_VAR
```

Value:

```
(x)
```

- Description:\n Useful macro for avoiding compiler warnings related to unused

  function arguments or unused variables.

Definition at line `436` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

DEBUG_LEVEL

```
#define DEBUG_LEVEL
```

Value:

```
BASIC_DEBUG
```

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

Definition at line `458` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

STATIC_ASSERT

```
#define STATIC_ASSERT
```

Disable static assertions on compilers that don't support them.

Definition at line `466` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

Macro Definition Documentation

MEMSET

```
#define MEMSET
```

Value:

```
(d, v, l)
```

Friendly convenience macro pointing to the C Stdlib functions.

Definition at line `188` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

MEMCOPY

```
#define MEMCOPY
```

Value:

```
(d, s, l)
```

Definition at line `189` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

MEMMOVE

```
#define MEMMOVE
```

Value:

```
(d, s, l)
```

Definition at line `190` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

MEMPGMCOPY

```
#define MEMPGMCOPY
```

Value:

```
(d, s, l)
```

Definition at line `191` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

MEMCOMPARE

```
#define MEMCOMPARE
```

Value:

```
(s0, s1, l)
```

Definition at line `192` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

MEMPGMCOMPARE

```
#define MEMPGMCOMPARE
```

Value:

```
(s0, s1, l)
```

Definition at line `193` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/generic/compiler/platform-common.h`

NVIC Configuration

NVIC Configuration

Reset Cause Type Definitions

Reset Cause Type Definitions

HAL Utilities

HAL Utilities

Modules

Crash and Watchdog Diagnostics

Cyclic Redundancy Code (CRC)

Random Number Generation

Network to Host Byte Order Conversion

Crash and Watchdog Diagnostics

Crash and Watchdog Diagnostics

Cyclic Redundancy Code (CRC)

Cyclic Redundancy Code (CRC)

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

Functions

uint16_t	halCommonCrc16(uint8_t newByte, uint16_t prevResult) Calculates 16-bit cyclic redundancy code (CITT CRC 16).
uint32_t	halCommonCrc32(uint8_t newByte, uint32_t prevResult) Calculates 32-bit cyclic redundancy code.

Macros

#define	INITIAL_CRC 0xFFFFFFFFL
#define	CRC32_START INITIAL_CRC
#define	CRC32_END 0xDEBB20E3L

Function Documentation

halCommonCrc16

```
uint16_t halCommonCrc16 (uint8_t newByte, uint16_t prevResult)
```

Calculates 16-bit cyclic redundancy code (CITT CRC 16).

Parameters

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

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.

Returns

- The new CRC result.

Definition at line `38` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/crc.h`

halCommonCrc32

```
uint32_t halCommonCrc32 (uint8_t newByte, uint32_t prevResult)
```

Calculates 32-bit cyclic redundancy code.

Parameters

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

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.

Returns

- The new CRC result.

Definition at line `55` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/crc.h`

Macro Definition Documentation

INITIAL_CRC

```
#define INITIAL_CRC
```

Value:

```
0xFFFFFFFFL
```

Definition at line `58` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/crc.h`

CRC32_START

```
#define CRC32_START
```

Value:

```
INITIAL_CRC
```

Definition at line `59` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/crc.h`

CRC32_END

```
#define CRC32_END
```

Value:

```
0xDEBB20E3L
```

Definition at line `60` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/crc.h`

Random Number Generation

Random Number Generation

Functions that provide access to random numbers.

These functions may be hardware accelerated, though often are not.

See random.h for source code.

Functions

void	halStackSeedRandom(uint32_t seed) Seeds the halCommonGetRandom() pseudorandom number generator.
uint16_t	halCommonGetRandom(void) Runs a standard LFSR to generate pseudorandom numbers.

Function Documentation

halStackSeedRandom

```
void halStackSeedRandom (uint32_t seed)
```

Seeds the halCommonGetRandom() pseudorandom number generator.

Parameters

N/A	seed	A seed for the pseudorandom number generator.

Called by the stack during initialization with a seed from the radio.

Definition at line `36` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/random.h`

halCommonGetRandom

```
uint16_t halCommonGetRandom (void)
```

Runs a standard LFSR to generate pseudorandom numbers.

Parameters

N/A		

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.

Definition at line `51` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/random.h`

Network to Host Byte Order Conversion

Network to Host Byte Order Conversion

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.

Functions

uint16_t	NTOHS(uint16_t val) Converts a short (16-bit) value from network to host byte order.
uint32_t	NTOHL(uint32_t val) Converts a long (32-bit) value from network to host byte order.
uint32_t	SwapEndiannessInt32u(uint32_t val)

Macros

#define	HTONL NTOHL
#define	HTONS NTOHS

Function Documentation

NTOHS

```
uint16_t NTOHS (uint16_t val)
```

Converts a short (16-bit) value from network to host byte order.

Parameters

N/A	val	

Definition at line `45` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/endian.h`

NTOHL

```
uint32_t NTOHL (uint32_t val)
```

Converts a long (32-bit) value from network to host byte order.

Parameters

N/A	val	

Definition at line `53` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/endian.h`

SwapEndiannessInt32u

```
uint32_t SwapEndiannessInt32u (uint32_t val)
```

Parameters

N/A	val	

Definition at line `79` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/endian.h`

Macro Definition Documentation

HTONL

```
#define HTONL
```

Value:

```
NTOHL
```

Definition at line `71` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/endian.h`

HTONS

```
#define HTONS
```

Value:

```
NTOHS
```

Definition at line `74` of file `/mnt/raid/workspaces/ws.Q8qnkBLX2/overlay/gsdk/platform/base/hal/micro/endian.h`

Deprecated List

Deprecated List

- Global EMBER_MAX_SECURED_APPLICATION_PAYLOAD_LENGTH
  The maximum length in bytes of the application payload for a secured message. This define has been deprecated, you should use the emberGetMaximumPayloadLength API instead.
- Global EMBER_MAX_UNSECURED_APPLICATION_PAYLOAD_LENGTH
  The maximum length in bytes of the application payload for an unsecured message. This define has been deprecated, you should use the emberGetMaximumPayloadLength API instead.

Training

Training

This series of tutorials demonstrates "the essentials" of building applications based on Silicon Labs Connect. The collection presents an incremental review of key techniques and features that help developers access the powerful convenience available thru Connect. These tutorials supplement the Developer's Guide and the API reference. Note: The API reference is the authoritative Connect resource, and represents the most current documentation at all times. Grant it a priority in any conflicts you may encounter in the following (or any other) Connect guidance.

Prerequisites

We strongly recommend familiarity with the resources above (especially the Developer's Guide) before beginning Connect-based application development, as they provide insight on crucial decisions that impact the design phase of your project.

That said, each tutorial in this series walks you through important Connect concepts using accessible demonstrations and discussions that illuminate how you can make Connect work for you. To extract the most value from this tutorial series, programming experience in embedded C and (at least) a baseline understanding of wireless networking theory is recommended. See Fundamentals.

The Tutorials

This group of tutorials leads you from "square one" (tutorial 1) to performing packet analysis on traffic captured from the firmware you've developed along the way (tutorial 7). A Direct mode Connect-based application serves as the demonstration vehicle. Though the coverage spans a broad range of topics, the tutorials are designed to be completed sequentially. However, Tutorial 6 is largely independent, and is immediately accessible as a strong subject matter reference.

1. Getting Started with Application Development
2. Communication Basics: Send and Receive
3. Command Line Interface
4. Communication Features: Acknowledge and Message Queue
5. Communication Features: Security
6. IEEE 802.15.4 Addressing
7. Traffic Analysis: Addressing, Acknowledgement, and Security
8. Network Management