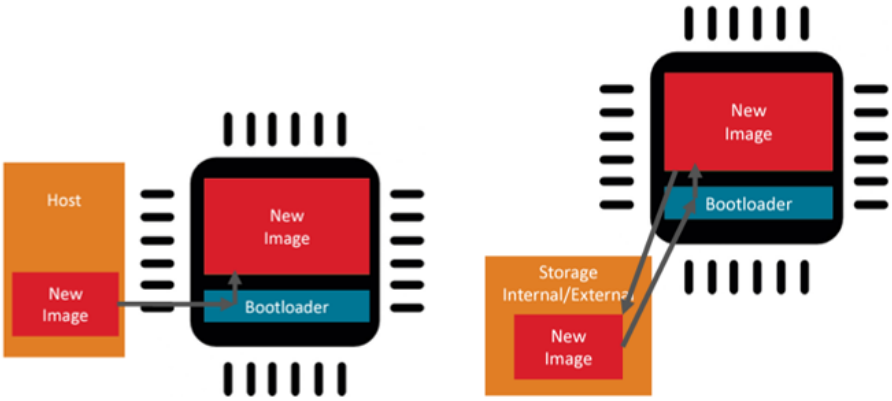
Gecko Bootloader

- Communication
  - UART XMODEM
  - Bluetooth Apploader OTA DFU
  - BGAPI UART DFU
  - EZSP-SPI
  - Utils
    - XMODEM Parser
      - XmodemPacket_t
        - header
        - packetNumber
        - packetNumberC
        - data
        - crcH
        - crcL
      - Commands
        - XMODEM_CMD_SOH
        - XMODEM_CMD_EOT
        - XMODEM_CMD_ACK
        - XMODEM_CMD_NAK
        - XMODEM_CMD_CAN
        - XMODEM_CMD_CTRL_C
        - XMODEM_CMD_C
      - xmodem_reset
      - xmodem_parsePacket
      - xmodem_getLastPacketNumber
      - XMODEM_DATA_SIZE
  - communication_init
  - communication_start
  - communication_main
  - communication_shutdown
  - bootloader_xmodem_communication_init
  - bootloader_xmodem_communication_start
  - bootloader_xmodem_communication_main
  - bootloader_apploader_communication_init
  - bootloader_apploader_communication_start
  - bootloader_apploader_communication_main
  - bootloader_apploader_get_custom_device_address
  - bootloader_bgapi_communication_init
  - bootloader_bgapi_communication_start
  - bootloader_bgapi_communication_main
  - bootloader_ezsp_communication_init
  - bootloader_ezsp_communication_start
  - bootloader_ezsp_communication_main

About the Gecko Bootloader

Gecko Bootloader

The Gecko Bootloader is a program stored in reserved flash memory that can initialize a device, update firmware images, and may also perform integrity checks. Devices are typically programmed with firmware during the product manufacturing process. A bootloader allows for reprogramming devices after production is complete, to add features or fix bugs.



The content on these pages is intended for those who want to experiment with or are already developing a bootloader for use in your devices.

For details about this release: Gecko Bootloader release notes may be found in the combined Gecko Platform release notes, available on the silabs.com Gecko SDK page.

For information about Silicon Labs' hardware or software products: See the developers pages on silabs.com.

For background about bootloading: Bootloader Fundamentals (PDF) is a good place to start.

To get started working with Gecko Bootloader example applications in Simplicity Studio: See the Getting Started page.

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

Bootloaders are usually developed along with the firmware image development. **For information about bootloading with individual protocols,** see the Protocol-Specific Information pages. An extensive body of content is available for each protocol and can be accessed through the docs.silabs.com homepage.

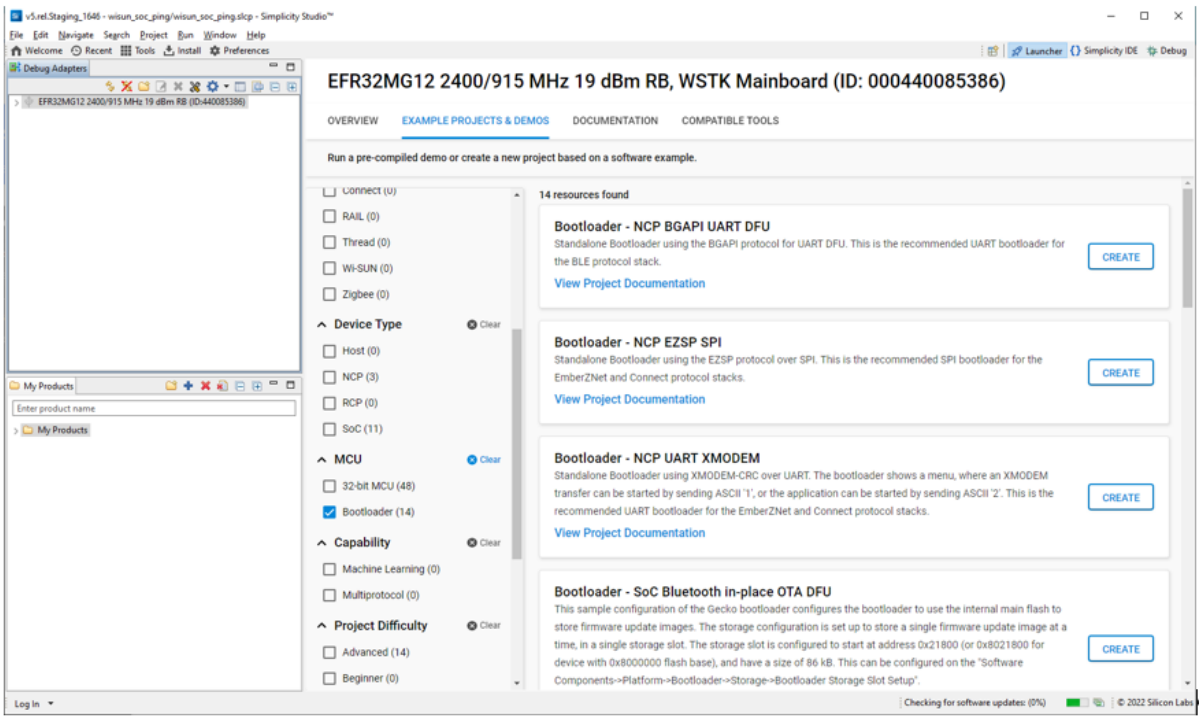
Getting Started

Getting Started with the Gecko Bootloader

The recommended environment for working with the Gecko Bootloader along with your protocol of choice is Simplicity Studio. For information about downloading Simplicity Studio and the Gecko SDK Suite (GSDK), see the Simplicity Studio v5 User's Guide. The application can be downloaded from the Simplicity Studio webpage.

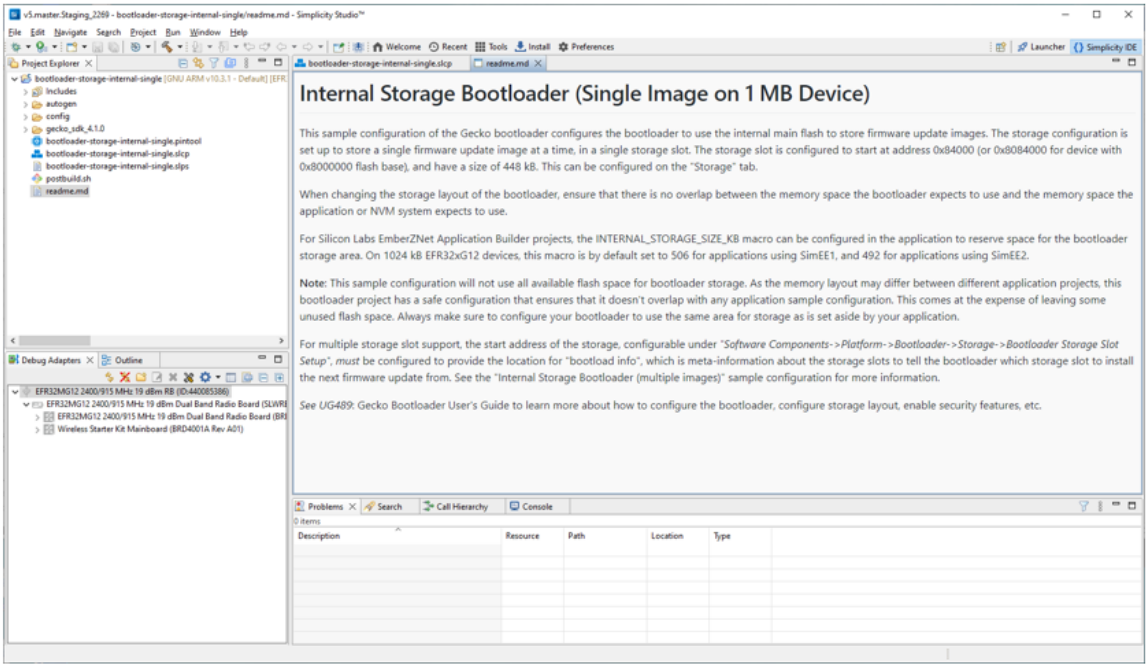
Once you have installed and familiarized yourself with Simplicity Studio and the GSDK, you can get started working with bootloader applications.

1. Create a project based on the Gecko Bootloader example of your choice.

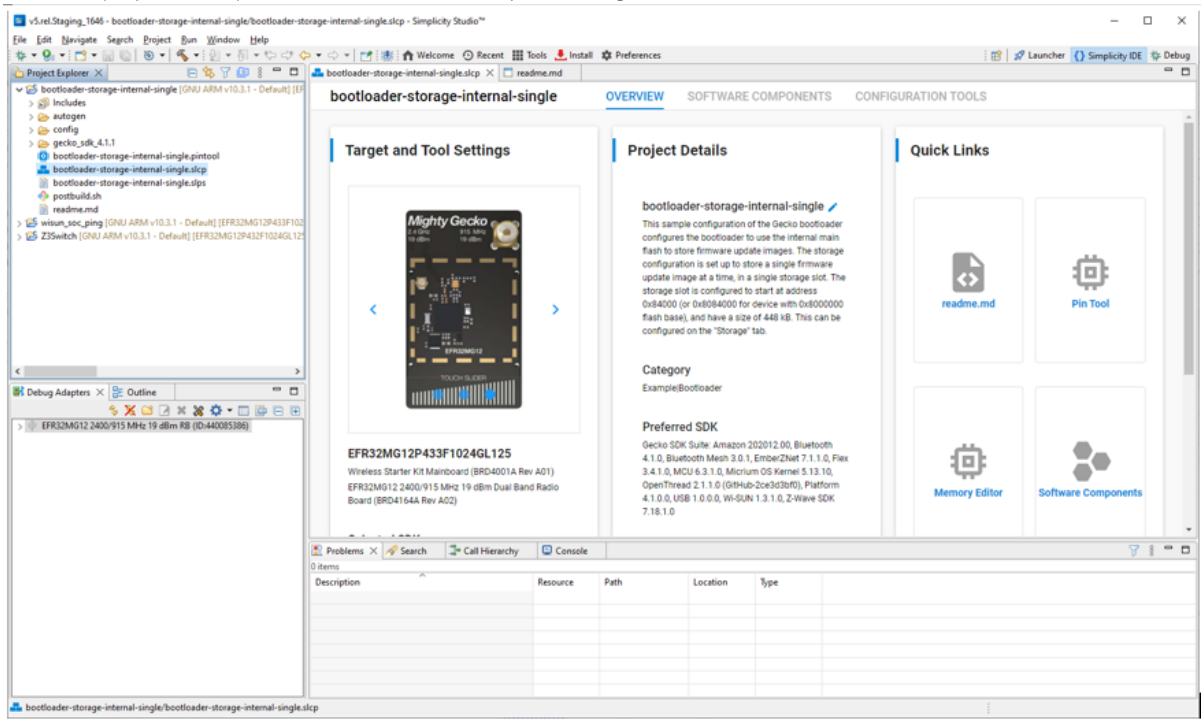


The project

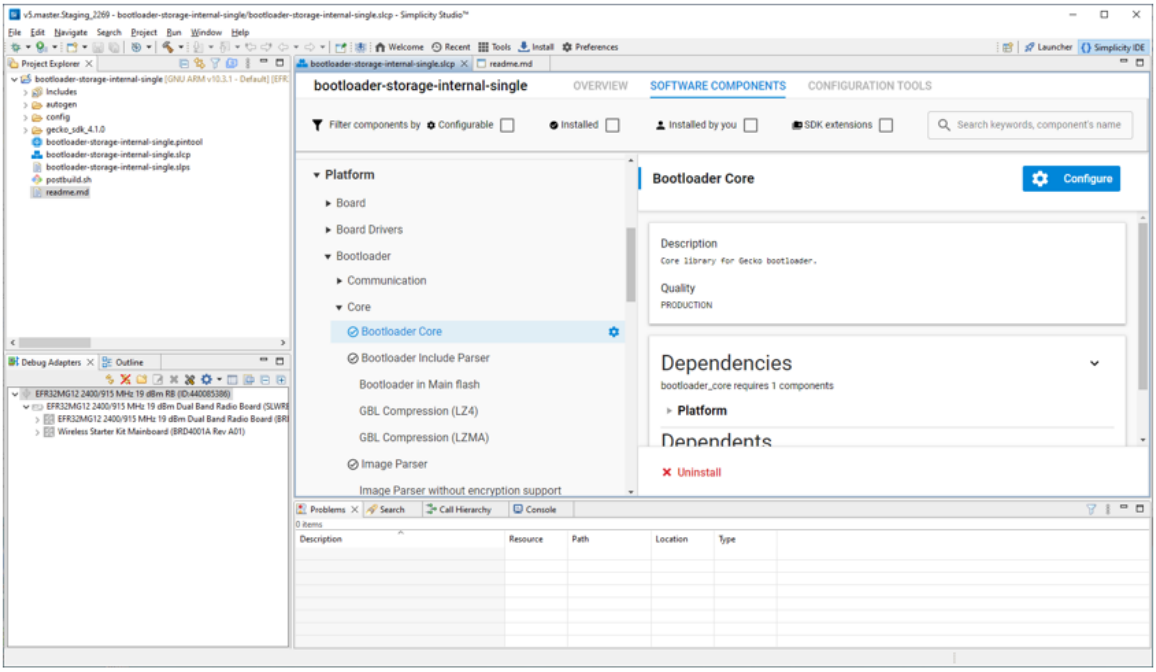
opens with a tab describing the example.



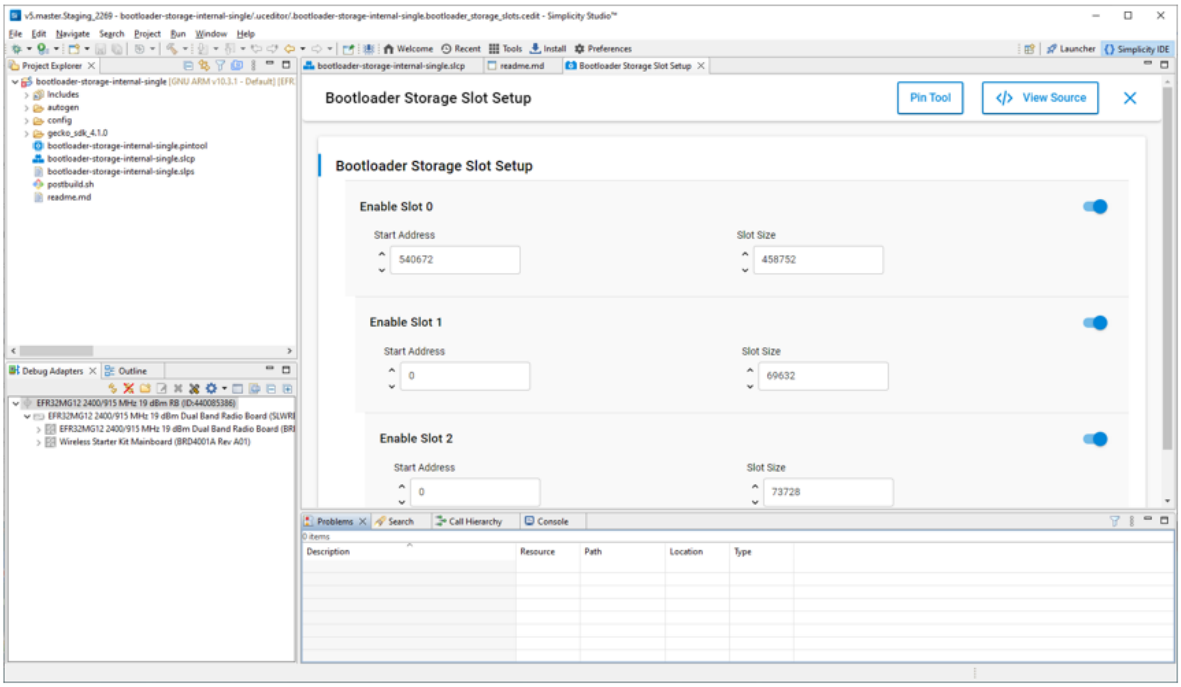
2. Click the project (*.slcp) tab to move to the Project Configurator interface.



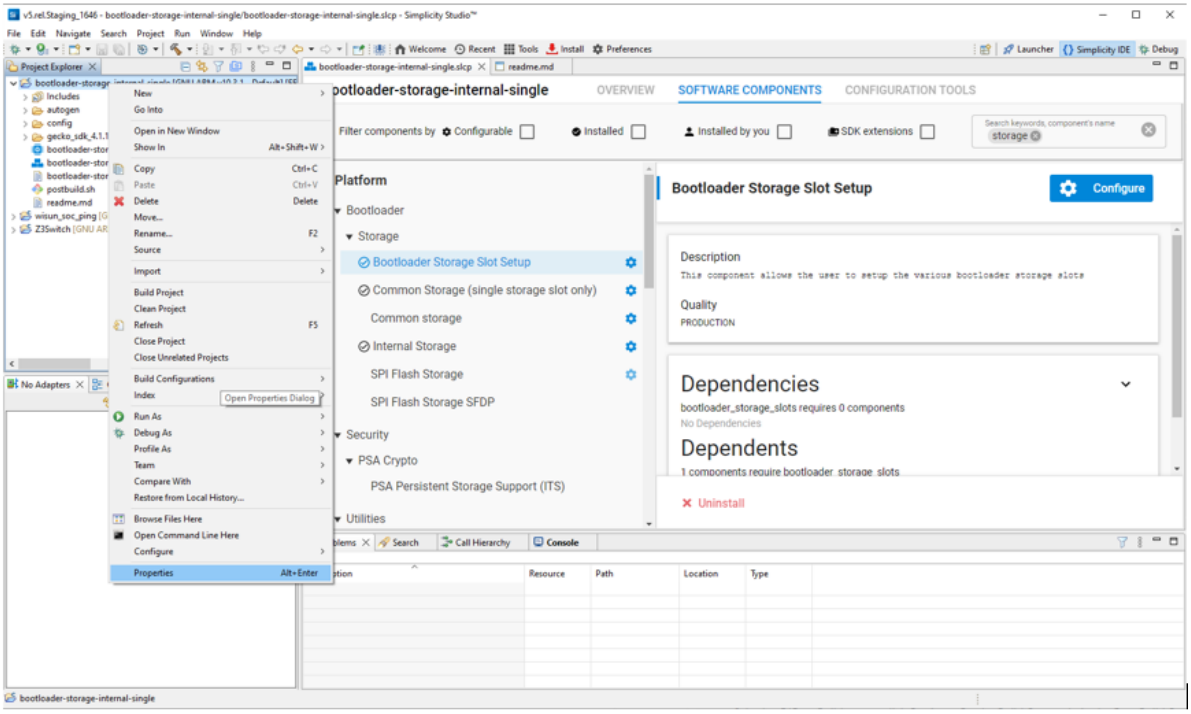
3. The Software Components tab shows the list of available components that can be installed in the project.



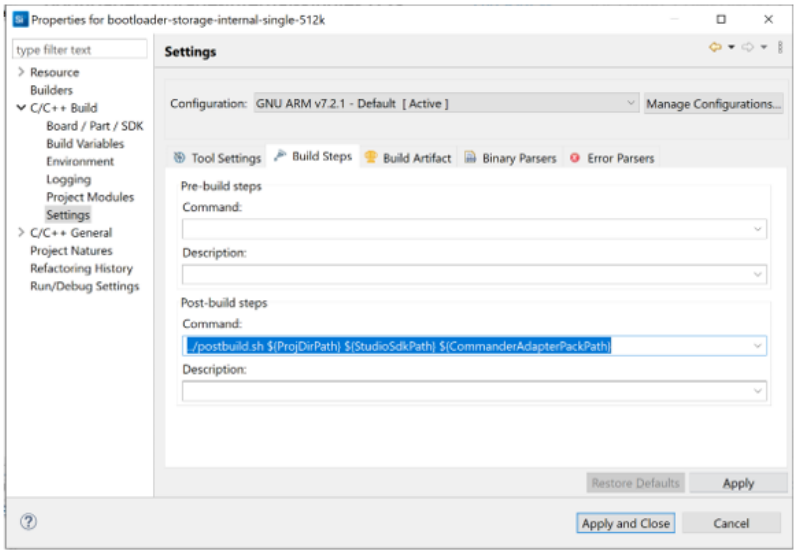
4. The Storage Slot Setup component allows you to configure storage slots to be used if a storage component is also installed. The default configuration matches the target part and bootloader type. This component supports a maximum of three storage slots.



5. Right-click the project name in the Project Explorer view and select **Properties**.



6. In the C/C++ Build group, click **Settings**. On the Build Steps tab, in the Post Build Steps Command field enter `../postbuild.sh "${ProjDirPath}" "${StudioSdkPath}" "${CommanderAdapterPackPath}"` . Click **Apply** and **Close**.



7. Click the Build (hammer) icon.

On Series 1 devices, three bootloader images are generated into the build directory: a main bootloader, a main bootloader with CRC32 checksum, and a combined first stage and main bootloader with CRC32 checksum. The main bootloader image is called <project-name>.s37, the main bootloader with CRC32 checksum is called <projectname>-crc.s37, while the combined first stage image + main bootloader image with a CRC32 checksum is called <projectname>-combined.s37. The first time a device is programmed, whether during development or manufacturing, the combined image needs to be programmed. For subsequent programming, when a first stage bootloader is already present on the device, it is okay to download an image containing just the main bootloader. In this case, the main bootloader with CRC32 should be used.

The requirement is that any main bootloader image that is programmed via serial wire must contain the CRC32 in the image. Files downloaded via serial wire are "s37" files. Most often, the <projectname>-combined.s37 file is the one downloaded during production programming. However, it is possible to download only the main bootloader over serial wire, in which case <projectname>-crc.s37 should be used.

Any main bootloader that is upgraded with the OTA or host method should already contain CRC32 because bootloader-initiated upgrades use GBL files (not "s37" files) and Simplicity Commander adds the CRC32 when it constructs the GBL file. The input files to Simplicity Commander can (and should) use the non-CRC "s37" file.

On Series 2 devices, the combined image is not present, since the first stage bootloader does not exist. The image containing only a main bootloader is the image that must be used to create a GBL file for bootloader upgrade.

Overview

Gecko Bootloader Developer's Guide

The Developer's Guide content is organized in three groups:

- Developing and Debugging: Includes a detailed Gecko Bootloader User's Guide and information about secure bootloading.
- Security: Brings together information about security considerations when bootloading.
- Protocol-Specific Information: Contains information about using the Gecko Bootloader with various protocols.

Overview

Developing and Debugging a Gecko Bootloader

These pages provide detailed information for Gecko Bootloader Developers.

- Silicon Labs Gecko Bootloader User's Guide for GSDK 4.0 and Higher (PDF): Describes the high-level implementation of the Silicon Labs Gecko Bootloader for EFM32 and EFR32 Series 1 and Series 2 microcontrollers, SoCs (System on Chips) and NCPs (Network Co-Processors), and provides information on different aspects of configuring the Gecko Bootloader.
- 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 describes the differences between the versions, including how to configure the new Gecko Bootloader in Simplicity Studio 5.
- Simplicity Commander Reference Guide (PDF): Describes how and when to use the Command-Line Interface (CLI) of Simplicity Commander. Simplicity Commander supports all EFR32 Wireless SoCs, EFR32 Wireless SoC modules (such as the MGM111 or MGM12P), EFM32 MCU families, and EM3xx Wireless SOCs.

Overview

Security

The following documents provide additional information about Gecko Bootloader security.

- 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.
- Series 2 TrustZone (PDF): Provides background on ARMv8-M TrustZone, a technology that provides a foundation for improved system security in embedded applications, and discusses its implementation on Series 2 devices.

Overview

Protocol-Specific Information

These pages provide information about using a Gecko Bootloader in the context of a specific protocol. Additional protocol-specific content is available through the docs.silabs.com homepage.

- Silicon Labs Bluetooth (PDF): Includes detailed information on using the Gecko Bootloader with Silicon Labs Bluetooth applications.
- Silicon Labs Connect (PDF): Provides information on using the Silicon Labs Gecko Bootloader with Connect.
- Zigbee EmberZNet (PDF): Includes detailed information on using the Silicon Labs Gecko Bootloader with EmberZNet.

Silicon Labs Gecko Bootloader

Silicon Labs Gecko Bootloader

Introduction

The Silicon Labs Gecko Bootloader is a common bootloader for all newer Silicon Labs MCUs and wireless MCUs (WMCUs). Key features of the bootloader are:

- Universal bootloader across MCU families (MCU and WMCU)
- Supports image verification and encryption for
  - Integrity
  - Authenticity
  - Confidentiality
- In-field upgradeable
- Configurable

This document describes the requirements and the usage of the Gecko Bootloader.

Device Compatibility

The Silicon Labs Gecko Bootloader is a customizable bootloader compatible with the following devices:

- EFM32 and EFR32 Series 1 Devices
- EFM32 and EFR32 Series 2 Devices

The Gecko bootloader has a two-stage design. The first stage enables upgrading the main bootloader, which enables in-field bootloader upgrades.

Requirements

Flash Size

See UG103.6 Bootloader Fundamentals, section '3. Memory Space For Bootloading' for bootloader flash size requirements.

Capabilities

ECDSA signature verification of the application SHA-256 digest is supported for integrity and security reasons. As a result, the bootloader can only run on devices that support hardware acceleration of these operations. The bootloader is therefore only compatible with the EFM32 and EFR32 Series 1 devices and EFM32 and EFR32 Series 2 devices.

Application Interface

The bootloader includes an application interface, which is exposed through a function table in the bootloader. The application interface provides APIs to store and retrieve upgrade images, verify the integrity of the images, and so on.

To use the application interface, include the `api/btl_interface.h` header in your application and add the following files to the build:

- `api/btl_interface.c`
- `api/btl_interface_storage.c`

See the Application Interface documentation for more information.

Bootloader Components

The bootloader itself consists of the following parts:

Core

The bootloader core contains the main function of both bootloader stages. It also contains functionality to write to the internal main Flash, to perform a bootloader upgrade, and to reset into the application flagging applicable reset reasons. See the Core Documentation for more information.

Driver

Different bootloading applications require different hardware drivers for use by the other components of the bootloader. See the Driver Documentation for more information.

Components

All parts of the bootloader that are either optional or swappable for different implementations are implemented as components. Each component has a generic header file and one or more implementations. The current release contains components for functionality, such as UART and SPI communication protocols, SPI Flash storage, internal Flash storage, and different cryptographic operations. For more information about different components, see the Components Documentation.

Application Interface

Application Interface

Application interface to the bootloader.

The application interface consists of functions that can be included in the customer application that and will communicate with the bootloader through the MainBootloaderTable_t. This table contains function pointers to the bootloader. The 10th word of the bootloader contains a pointer to this struct, allowing any application to easily locate it. To access the bootloader table, use wrapper functions. Avoid accessing the bootloader table directly.

Modules

Application Parser Interface

Application Properties

Application Storage Interface

Common Application Interface

Application Parser Interface

Application Parser Interface

Application interface for the bootloader image parser.

The Parser Interface can be used to parse upgrade images from the context of the application.

Modules

BootloaderParserCallbacks_t

Typedefs

typedef void(* BootloaderParserCallback_t)(uint32_t address, uint8_t *data, size_t length, void *context)
Bootloader parser callback.

Functions

int32_t bootloader_initParser(BootloaderParserContext_t *context, size_t contextSize)
Initialize the image parser.

int32_t bootloader_parseBuffer(BootloaderParserContext_t *context, BootloaderParserCallbacks_t *callbacks, uint8_t data[], size_t numBytes)
Parse a buffer.

int32_t bootloader_parseImageInfo(BootloaderParserContext_t *context, uint8_t data[], size_t numBytes, ApplicationData_t *appInfo, uint32_t *bootloaderVersion)
Parse a buffer and get application and bootloader upgrade metadata from the buffer.

uint32_t bootloader_parserContextSize(void)
Find the size of the context struct BootloaderParserContext used by the bootloader image parser to store parser state.

Typedef Documentation

BootloaderParserCallback_t

```
typedef void(* BootloaderParserCallback_t) (uint32_t address, uint8_t *data, size_t length, void *context) )(uint32_t address, uint8_t *data, size_t length, void *context)
```

Bootloader parser callback.

Parameters

N/A	address	Address of the data
N/A	data	Raw data
N/A	length	Size in bytes of raw data.
N/A	context	A context variable defined by the implementation that is implementing this callback.

Definition at line 48 of file platform/bootloader/api/btl_interface_parser.h

Function Documentation

bootloader_initParser

```
int32_t bootloader_initParser (BootloaderParserContext_t *context, size_t contextSize)
```

Initialize the image parser.

Parameters

[in]	context	Pointer to the parser context struct.
[in]	contextSize	Size of the context struct.

Returns

- BOOTLOADER_OK if success, BOOTLOADER_ERROR_PARSE_CONTEXT if context struct is too small.

Definition at line `77` of file `platform/bootloader/api/btl_interface_parser.h`

bootloader_parseBuffer

```
int32_t bootloader_parseBuffer (BootloaderParserContext_t *context, BootloaderParserCallbacks_t *callbacks, uint8_t data[], size_t numBytes)
```

Parse a buffer.

Parameters

[in]	context	Pointer to the parser context struct.
[in]	callbacks	Callbacks to be called by the parser.
[in]	data	Data to be parsed.
[in]	numBytes	Size of the data buffer.

Returns

- BOOTLOADER_ERROR_PARSE_CONTINUE if the chunk was parsed correctly, and a new chunk is expected. BOOTLOADER_ERROR_PARSE_ERROR if something went wrong while parsing. BOOTLOADER_ERROR_PARSE_SUCCESS if the entire file was successfully parsed.

Definition at line `93` of file `platform/bootloader/api/btl_interface_parser.h`

bootloader_parseImageInfo

```
int32_t bootloader_parseImageInfo (BootloaderParserContext_t *context, uint8_t data[], size_t numBytes, ApplicationData_t *appInfo, uint32_t *bootloaderVersion)
```

Parse a buffer and get application and bootloader upgrade metadata from the buffer.

Parameters

[in]	context	Pointer to the parser context struct.
[in]	data	Data to be parsed.
[in]	numBytes	Size of the data buffer.
[out]	appInfo	Pointer to ApplicationData_t struct.

[out]	bootloaderVersion	Pointer to an integer representing bootloader version.

Note

- `appInfo` and `bootloaderVersion` will default to zeros.

Returns

- BOOTLOADER_OK if metadata was filled successfully.

Definition at line `115` of file `platform/bootloader/api/btl_interface_parser.h`

bootloader_parserContextSize

```
uint32_t bootloader_parserContextSize (void)
```

Find the size of the context struct BootloaderParserContext used by the bootloader image parser to store parser state.

Parameters

N/A		

Returns

- size of BootloaderParserContext, returns 0 if something went wrong.

Definition at line `133` of file `platform/bootloader/api/btl_interface_parser.h`

BootloaderParserCallbacks_t

Function pointers to parser callbacks.

Public Attributes

void *	context Opaque pointer passed to the callback functions.
BootloaderParserCallback_t	applicationCallback Callback function pointer for application image data.
BootloaderParserCallback_t	metadataCallback Callback function pointer for image metadata.
BootloaderParserCallback_t	bootloaderCallback Callback function pointer for bootloader upgrade image data.

Public Attribute Documentation

context

```
void* BootloaderParserCallbacks_t::context
```

Opaque pointer passed to the callback functions.

Definition at line 59 of file platform/bootloader/api/btl_interface_parser.h

applicationCallback

```
BootloaderParserCallback_t BootloaderParserCallbacks_t::applicationCallback
```

Callback function pointer for application image data.

Definition at line 61 of file platform/bootloader/api/btl_interface_parser.h

metadataCallback

```
BootloaderParserCallback_t BootloaderParserCallbacks_t::metadataCallback
```

Callback function pointer for image metadata.

Definition at line 63 of file platform/bootloader/api/btl_interface_parser.h

bootloaderCallback

BootloaderParserCallback_t BootloaderParserCallbacks_t::bootloaderCallback

Callback function pointer for bootloader upgrade image data.

Definition at line 65 of file platform/bootloader/api/btl_interface_parser.h

Application Properties

Application Properties

Properties of the application that can be accessed by the bootloader.

Applications must contain an ApplicationProperties_t struct declaring the application version and capabilities, and so on. The metadata contained in this struct will be extracted from the application by the Simplicity Commander tool and placed in the GBL upgrade file. If this struct is not in the application image, it will be added to the GBL file by the Simplicity Commander.

The struct is also used to declare whether the application image is signed and what type of signature is used. If no ApplicationProperties_t struct is present, the bootloader will assume that the application image is signed using APPLICATION_SIGNATURE_ECDSA_P256.

To ensure that the bootloader can easily locate the ApplicationProperties_t struct, if not already done by the linker, Simplicity Commander will modify word 13 of the application to insert a pointer to the ApplicationProperties_t struct.

Modules

ApplicationData_t

ApplicationCertificate_t

ApplicationProperties_t

Macros

#define APPLICATION_PROPERTIES_MAGIC undefined
Magic value declaring the existence of an ApplicationProperties_t struct.

#define APPLICATION_PROPERTIES_REVERSED undefined
Byte-reversed version of APPLICATION_PROPERTIES_MAGIC.

#define APPLICATION_PROPERTIES_VERSION_MAJOR (1UL)
Major version number of the AppliationProperties_t struct.

#define APPLICATION_PROPERTIES_VERSION_MINOR (2UL)
Minor version number of the AppliationProperties_t struct.

#define APPLICATION_CERTIFICATE_VERSION (1UL)
Version number of the ApplicationCertificate_t struct.

#define APPLICATION_SIGNATURE_NONE (0UL)
The application is not signed.

#define APPLICATION_SIGNATURE_ECDSA_P256 (1UL << 0UL)
The SHA-256 digest of the application is signed using ECDSA with the NIST P-256 curve.

#define APPLICATION_SIGNATURE_CRC32 (1UL << 1UL)
The application is not signed, but has a CRC-32 checksum.

#define APPLICATION_TYPE_ZIGBEE (1UL << 0UL)
The application contains a Zigbee wireless stack.

#define APPLICATION_TYPE_THREAD (1UL << 1UL)
The application contains a Thread wireless stack.

#define APPLICATION_TYPE_FLEX (1UL << 2UL)
The application contains a Flex wireless stack.

#define APPLICATION_TYPE_BLUETOOTH (1UL << 3UL)
The application contains a Bluetooth wireless stack.

#define APPLICATION_TYPE_MCU (1UL << 4UL)
The application is an MCU application.

#define APPLICATION_TYPE_BLUETOOTH_APP (1UL << 5UL)
The application contains a Bluetooth application.

#define APPLICATION_TYPE_BOOTLOADER (1UL << 6UL)
The application contains a bootloader.

#define APPLICATION_TYPE_ZWAVE (1UL << 7UL)
The application contains a Zwave wireless stack.

Macro Definition Documentation

APPLICATION_PROPERTIES_MAGIC

```
#define APPLICATION_PROPERTIES_MAGIC
```

Value:

```
0    { \
0    0x13, 0xb7, 0x79, 0xfa,          \
0    0xc9, 0x25, 0xdd, 0xb7,          \
0    0xad, 0xf3, 0xcf, 0xe0,          \
0    0xf1, 0xb6, 0x14, 0xb8           \
0  }
```

Magic value declaring the existence of an ApplicationProperties_t struct.

Definition at line 48 of file platform/bootloader/api/application_properties.h

APPLICATION_PROPERTIES_REVERSED

```
#define APPLICATION_PROPERTIES_REVERSED
```

Value:

```
0    { \
0    0xb8, 0x14, 0xb6, 0xf1,          \
0    0xe0, 0xcf, 0xf3, 0xad,          \
0    0xb7, 0xdd, 0x25, 0xc9,          \
0    0xfa, 0x79, 0xb7, 0x13           \
0  }
```

Byte-reversed version of APPLICATION_PROPERTIES_MAGIC.

Definition at line 56 of file platform/bootloader/api/application_properties.h

APPLICATION_PROPERTIES_VERSION_MAJOR

```
#define APPLICATION_PROPERTIES_VERSION_MAJOR
```

Value:

```
(1UL)
```

Major version number of the AppliationProperties_t struct.

Definition at line 64 of file platform/bootloader/api/application_properties.h

APPLICATION_PROPERTIES_VERSION_MINOR

```
#define APPLICATION_PROPERTIES_VERSION_MINOR
```

Value:

```
(2UL)
```

Minor version number of the AppliationProperties_t struct.

Definition at line 66 of file platform/bootloader/api/application_properties.h

APPLICATION_CERTIFICATE_VERSION

```
#define APPLICATION_CERTIFICATE_VERSION
```

Value:

```
(1UL)
```

Version number of the ApplicationCertificate_t struct.

Definition at line 68 of file platform/bootloader/api/application_properties.h

APPLICATION_SIGNATURE_NONE

```
#define APPLICATION_SIGNATURE_NONE
```

Value:

```
(0UL)
```

The application is not signed.

Definition at line 70 of file platform/bootloader/api/application_properties.h

APPLICATION_SIGNATURE_ECDSA_P256

```
#define APPLICATION_SIGNATURE_ECDSA_P256
```

Value:

```
(1UL << 0UL)
```

The SHA-256 digest of the application is signed using ECDSA with the NIST P-256 curve.

Definition at line `73` of file `platform/bootloader/api/application_properties.h`

APPLICATION_SIGNATURE_CRC32

```
#define APPLICATION_SIGNATURE_CRC32
```

Value:

```
(1UL << 1UL)
```

The application is not signed, but has a CRC-32 checksum.

Definition at line `75` of file `platform/bootloader/api/application_properties.h`

APPLICATION_TYPE_ZIGBEE

```
#define APPLICATION_TYPE_ZIGBEE
```

Value:

```
(1UL << 0UL)
```

The application contains a Zigbee wireless stack.

Definition at line `78` of file `platform/bootloader/api/application_properties.h`

APPLICATION_TYPE_THREAD

```
#define APPLICATION_TYPE_THREAD
```

Value:

```
(1UL << 1UL)
```

The application contains a Thread wireless stack.

Definition at line `80` of file `platform/bootloader/api/application_properties.h`

APPLICATION_TYPE_FLEX

```
#define APPLICATION_TYPE_FLEX
```

Value:

```
(1UL << 2UL)
```

The application contains a Flex wireless stack.

Definition at line 82 of file platform/bootloader/api/application_properties.h

APPLICATION_TYPE_BLUETOOTH

```
#define APPLICATION_TYPE_BLUETOOTH
```

Value:

```
(1UL << 3UL)
```

The application contains a Bluetooth wireless stack.

Definition at line 84 of file platform/bootloader/api/application_properties.h

APPLICATION_TYPE_MCU

```
#define APPLICATION_TYPE_MCU
```

Value:

```
(1UL << 4UL)
```

The application is an MCU application.

Definition at line 86 of file platform/bootloader/api/application_properties.h

APPLICATION_TYPE_BLUETOOTH_APP

```
#define APPLICATION_TYPE_BLUETOOTH_APP
```

Value:

```
(1UL << 5UL)
```

The application contains a Bluetooth application.

Definition at line 88 of file platform/bootloader/api/application_properties.h

APPLICATION_TYPE_BOOTLOADER

```
#define APPLICATION_TYPE_BOOTLOADER
```

Value:

```
(1UL << 6UL)
```

The application contains a bootloader.

Definition at line 90 of file platform/bootloader/api/application_properties.h

APPLICATION_TYPE_ZWAVE

```
#define APPLICATION_TYPE_ZWAVE
```

Value:

```
(1UL << 7UL)
```

The application contains a Zwave wireless stack.

Definition at line 92 of file platform/bootloader/api/application_properties.h

ApplicationData_t

Application Data.

Public Attributes

uint32_t	type Bitfield representing type of application, e.g., APPLICATION_TYPE_ZIGBEE.
uint32_t	version Version number for this application.
uint32_t	capabilities Capabilities of this application.
uint8_t	productId Unique ID (UUID or GUID) for the product this application is built for.

Public Attribute Documentation

type

```
uint32_t ApplicationData_t::type
```

Bitfield representing type of application, e.g., APPLICATION_TYPE_ZIGBEE.

Definition at line 98 of file platform/bootloader/api/application_properties.h

version

```
uint32_t ApplicationData_t::version
```

Version number for this application.

Definition at line 100 of file platform/bootloader/api/application_properties.h

capabilities

```
uint32_t ApplicationData_t::capabilities
```

Capabilities of this application.

Definition at line 102 of file platform/bootloader/api/application_properties.h

productId

> uint8_t ApplicationData_t::productId[16]

Unique ID (UUID or GUID) for the product this application is built for.

Definition at line 104 of file platform/bootloader/api/application_properties.h

ApplicationCertificate_t

Application Certificate.

Public Attributes

uint8_t	structVersion Version of the certificate structure.
uint8_t	flags Reserved flags.
uint8_t	key Public key.
uint32_t	version The version number of this certificate.
uint8_t	signature Signature of the certificate.

Public Attribute Documentation

structVersion

```
uint8_t ApplicationCertificate_t::structVersion
```

Version of the certificate structure.

Definition at line 110 of file platform/bootloader/api/application_properties.h

flags

```
uint8_t ApplicationCertificate_t::flags[3]
```

Reserved flags.

Definition at line 112 of file platform/bootloader/api/application_properties.h

key

```
uint8_t ApplicationCertificate_t::key[64]
```

Public key.

Definition at line 114 of file platform/bootloader/api/application_properties.h

version

> uint32_t ApplicationCertificate_t::version

The version number of this certificate.

Definition at line 116 of file platform/bootloader/api/application_properties.h

signature

> uint8_t ApplicationCertificate_t::signature[64]

Signature of the certificate.

Definition at line 118 of file platform/bootloader/api/application_properties.h

ApplicationProperties_t

Application Properties struct.

Public Attributes

uint8_t	magic Magic value indicating this is an ApplicationProperties_t struct.
uint32_t	structVersion Version number of this struct.
uint32_t	signatureType Type of signature this application is signed with.
uint32_t	signatureLocation Location of the signature. Typically points to the end of the application.
ApplicationData_t	app Information about the application.
ApplicationCertificate_t *	cert Pointer to information about the certificate.
uint8_t *	longTokenSectionAddress Pointer to Long Token Data Section.
const uint8_t	decryptKey Parser Decryption Key.

Public Attribute Documentation

magic

```
uint8_t ApplicationProperties_t::magic[16]
```

Magic value indicating this is an ApplicationProperties_t struct.

Must equal APPLICATION_PROPERTIES_MAGIC

Definition at line 125 of file platform/bootloader/api/application_properties.h

structVersion

```
uint32_t ApplicationProperties_t::structVersion
```

Version number of this struct.

Definition at line 127 of file platform/bootloader/api/application_properties.h

signatureType

uint32_t ApplicationProperties_t::signatureType

Type of signature this application is signed with.

Definition at line 129 of file platform/bootloader/api/application_properties.h

signatureLocation

uint32_t ApplicationProperties_t::signatureLocation

Location of the signature. Typically points to the end of the application.

Definition at line 131 of file platform/bootloader/api/application_properties.h

app

ApplicationData_t ApplicationProperties_t::app

Information about the application.

Definition at line 133 of file platform/bootloader/api/application_properties.h

cert

ApplicationCertificate_t* ApplicationProperties_t::cert

Pointer to information about the certificate.

Definition at line 135 of file platform/bootloader/api/application_properties.h

longTokenSectionAddress

uint8_t* ApplicationProperties_t::longTokenSectionAddress

Pointer to Long Token Data Section.

Definition at line 137 of file platform/bootloader/api/application_properties.h

decryptKey

const uint8_t ApplicationProperties_t::decryptKey[16]

Parser Decryption Key.

Definition at line 139 of file platform/bootloader/api/application_properties.h

Application Storage Interface

Application Storage Interface

Application interface for interfacing with the bootloader storage.

Note

- These Bootloader APIs are not reentrant and should be wrapped in critical section where needed.

The Storage Interface is only available on bootloaders that declare they support BOOTLOADER_CAPABILITY_STORAGE.

- Example

Example

Snippet for the OTA use case:

```
* ///OTA Example
* /// Assuming the user has an upgrade image downloaded which will be used to upgrade the application
*
* //Initialize the bootloader interface
* bootloader_init();
*
* //Erase the storage slot in internal/SPI flash memory
* bootloader_eraseStorageSlot(0);
*
* //Write the upgrade image (GBL file data) to the slot. blinkGbl ▒ uint8 array holding the GBL data in memory
* bootloader_writeStorage(0, 0, blinkGbl, sizeof(blinkGbl));
*
* //Reboot into the bootloader to install the new image
* bootloader_rebootAndInstall();
*
* The general flow for bootloader interface APIs from the application is:
*
* ///General flow
*
* //Initialize the bootloader interface
* bootloader_init();
*
* //Interface API accesses
* ..............
* ..............
* ..............
*
* //De-initialize the bootloader interface
* bootloader_deinit();
*
```

Modules

BootloaderStorageSlot_t

BootloaderStorageImplementationInformation_t

BootloaderStorageInformation_t

BootloaderEraseStatus_t

BootloaderStorageFunctions_t

Enumerations

enum BootloaderStorageType_t {

SPIFLASH
INTERNAL_FLASH
CUSTOM_STORAGE

}

Possible storage types.

Functions

void bootloader_getStorageInfo(BootloaderStorageInformation_t *info)
Get information about the storage component.

int32_t bootloader_getStorageSlotInfo(uint32_t slotId, BootloaderStorageSlot_t *slot)
Get information about a storage slot.

int32_t bootloader_readStorage(uint32_t slotId, uint32_t offset, uint8_t *buffer, size_t length)
Read data from a storage slot.

int32_t bootloader_writeStorage(uint32_t slotId, uint32_t offset, uint8_t *buffer, size_t length)
Write data to a storage slot.

int32_t bootloader_eraseWriteStorage(uint32_t slotId, uint32_t offset, uint8_t *buffer, size_t length)
Erase and write data to a storage slot.

int32_t bootloader_eraseStorageSlot(uint32_t slotId)
Erase all contents of a storage slot.

int32_t bootloader_initChunkedEraseStorageSlot(uint32_t slotId, BootloaderEraseStatus_t *eraseStat)
Initialize chunked erase of a storage slot.

int32_t bootloader_chunkedEraseStorageSlot(BootloaderEraseStatus_t *eraseStat)
Erase one page from a storage slot according to the struct BootloaderEraseStatus_t.

int32_t bootloader_setImagesToBootload(int32_t *slotIds, size_t length)
Set a prioritized list of images to attempt to bootload.

int32_t bootloader_getImagesToBootload(int32_t *slotIds, size_t length)
Get the prioritized list of images the bootloader will attempt to bootload.

int32_t bootloader_appendImageToBootloadList(int32_t slotId)
Append a single image to the list of images to attempt to bootload.

int32_t bootloader_setImageToBootload(int32_t slotId)
Set a single image to attempt to bootload.

int32_t bootloader_initVerifyImage(uint32_t slotId, void *context, size_t contextSize)
Initialize image verification.

int32_t bootloader_continueVerifyImage(void *context, BootloaderParserCallback_t metadataCallback)
Continue image verification.

int32_t bootloader_verifyImage(uint32_t slotId, BootloaderParserCallback_t metadataCallback)
Verify that the image in the given storage slot is valid.

int32_t bootloader_getImageInfo(uint32_t slotId, ApplicationData_t *appInfo, uint32_t *bootloaderVersion)
Get application and bootloader upgrade metadata from the storage slot.

bool bootloader_storageIsBusy(void)
Check whether the bootloader storage is busy.

int32_t bootloader_readRawStorage(uint32_t address, uint8_t *buffer, size_t length)
Read raw data from storage.

int32_t bootloader_writeRawStorage(uint32_t address, uint8_t *buffer, size_t length)
Write data to storage.

int32_t bootloader_eraseRawStorage(uint32_t address, size_t length)
Erase data from storage.

int32_t bootloader_getAllocatedDMAChannel(void)
Get allocated DMA channel for MSC write.

Macros

#define BOOTLOADER_STORAGE_VERIFICATION_CONTEXT_SIZE (384)
Context size for bootloader verification context.

#define BOOTLOADER_STORAGE_INFO_VERSION (0x30000U)
Current version of the BootloaderStorageInformation_t struct.

#define BOOTLOADER_STORAGE_IMPL_INFO_VERSION (0x0210U)
Current version of the BootloaderStorageImplementationInformation_t struct.

#define BOOTLOADER_STORAGE_IMPL_INFO_VERSION_MAJOR (0x0200U)
Major version of the BootloaderStorageImplementationInformation_t struct.

#define BOOTLOADER_STORAGE_IMPL_INFO_VERSION_MAJOR_MASK (0xFF00U)
Major version mask for BOOTLOADER_STORAGE_IMPL_INFO_VERSION.

#define BOOTLOADER_STORAGE_IMPL_CAPABILITY_ERASE_SUPPORTED (1 << 0)
Spiflash capability indicating that it supports erase.

#define BOOTLOADER_STORAGE_IMPL_CAPABILITY_PAGE_ERASE_REQUIRED (1 << 1)
Spiflash capability indicating it requires full page erases before new data can be written.

#define BOOTLOADER_STORAGE_IMPL_CAPABILITY_BLOCKING_WRITE (1 << 2)
Spiflash capability indicating that the write function is blocking.

#define BOOTLOADER_STORAGE_IMPL_CAPABILITY_BLOCKING_ERASE (1 << 3)
Spiflash capability indicating that the erase function is blocking.

#define BOOTLOADER_STORAGE_ISSI_IS25LQ040B (1U << 0)
ISSI IS25LQ040B SPI Flash.

#define BOOTLOADER_STORAGE_ISSI_IS25LQ020B (1U << 1)
ISSI IS25LQ020B SPI Flash.

#define BOOTLOADER_STORAGE_ISSI_IS25LQ010B (1U << 2)
ISSI IS25LQ010B SPI Flash.

#define BOOTLOADER_STORAGE_ISSI_IS25LQ512B (1U << 3)
ISSI IS25LQ512B SPI Flash.

#define BOOTLOADER_STORAGE_ISSI_IS25LQ025B (1U << 4)
ISSI IS25LQ025B SPI Flash.

#define BOOTLOADER_STORAGE_NUMONYX_M25P16 (1U << 5)
Numonyx M25P16 SPI Flash.

#define BOOTLOADER_STORAGE_NUMONYX_M25P80 (1U << 6)
Numonyx M25P80 SPI Flash.

#define BOOTLOADER_STORAGE_NUMONYX_M25P40 (1U << 7)
Numonyx M25P40 SPI Flash.

#define BOOTLOADER_STORAGE_NUMONYX_M25P20 (1U << 8)
Numonyx M25P20 SPI Flash.

#define BOOTLOADER_STORAGE_ADESTO_AT25SF041 (1U << 9)
Adesto AT25SF041 SPI Flash.

#define BOOTLOADER_STORAGE_ATMEL_AT25DF081A (1U << 10)
Atmel AT25DF081A SPI Flash.

#define BOOTLOADER_STORAGE_ATMEL_AT25DF041A (1U << 11)
Atmel AT25DF041A SPI Flash.

#define BOOTLOADER_STORAGE_MACRONIX_MX25R6435F (1U << 12)
Macronix MX25R6435F SPI Flash.

#define BOOTLOADER_STORAGE_MACRONIX_MX25R3235F (1U << 13)
Macronix MX25R6435F SPI Flash.

#define BOOTLOADER_STORAGE_MACRONIX_MX25U1635E (1U << 14)
Macronix MX25U1635E SPI Flash.

#define BOOTLOADER_STORAGE_MACRONIX_MX25L1606E (1U << 15)
Macronix MX25L1606E SPI Flash.

#define BOOTLOADER_STORAGE_MACRONIX_MX25R8035F (1U << 16)
Macronix MX25R8035F SPI Flash.

#define BOOTLOADER_STORAGE_MACRONIX_MX25L8006E (1U << 17)
Macronix MX25L8006E SPI Flash.

#define BOOTLOADER_STORAGE_MACRONIX_MX25L4006E (1U << 18)
Macronix MX25L4006E SPI Flash.

#define BOOTLOADER_STORAGE_MACRONIX_MX25L2006E (1U << 19)
Macronix MX25L2006E SPI Flash.

#define BOOTLOADER_STORAGE_WINBOND_W25Q80BV (1U << 20)
Winbond W25Q80BV SPI Flash.

#define BOOTLOADER_STORAGE_WINBOND_W25X20BV (1U << 21)
Winbond W25X20BV SPI Flash.

#define BOOTLOADER_STORAGE_SPANSION_S25FL208K (1U << 22)
Spansion S25L208K SPI Flash.

#define BOOTLOADER_STORAGE_INTERNAL_STORAGE (1U << 30)
Internal storage.

#define BOOTLOADER_STORAGE_JEDEC (1U << 31)
JEDEC Supported SPI Flash.

Enumeration Documentation

BootloaderStorageType_t

> BootloaderStorageType_t

Possible storage types.

Enumerator	
SPIFLASH	Storage backend is a SPI flash.
INTERNAL_FLASH	Storage backend is internal flash.
CUSTOM_STORAGE	Storage backend is custom.

Definition at line 81 of file platform/bootloader/api/btl_interface_storage.h

Function Documentation

bootloader_getStorageInfo

> void bootloader_getStorageInfo (BootloaderStorageInformation_t *info)

Get information about the storage component.

Parameters

[out]	info	Information about the storage component.

Definition at line 297 of file platform/bootloader/api/btl_interface_storage.h

bootloader_getStorageSlotInfo

> int32_t bootloader_getStorageSlotInfo (uint32_t slotId, BootloaderStorageSlot_t *slot)

Get information about a storage slot.

Parameters

[in]	slotId	ID of the slot to get information about
[out]	slot	Information about the storage slot

Returns

- BOOTLOADER_OK on success, else error code in BOOTLOADER_ERROR_STORAGE_BASE range

Definition at line 308 of file platform/bootloader/api/btl_interface_storage.h

bootloader_readStorage

> int32_t bootloader_readStorage (uint32_t slotId, uint32_t offset, uint8_t *buffer, size_t length)

Read data from a storage slot.

Parameters

[in]	slotId	ID of the slot
[in]	offset	Offset into the slot to start reading from

[out]	buffer	Buffer to store the data
[in]	length	Amount of data to read

Returns

- BOOTLOADER_OK on success, else error code in BOOTLOADER_ERROR_STORAGE_BASE range

Definition at line 322 of file platform/bootloader/api/btl_interface_storage.h

bootloader_writeStorage

> int32_t bootloader_writeStorage (uint32_t slotId, uint32_t offset, uint8_t *buffer, size_t length)

Write data to a storage slot.

Parameters

[in]	slotId	ID of the slot
[in]	offset	Offset into the slot to start writing to
[in]	buffer	Buffer to read data to write from
[in]	length	Amount of data to write. Must be a multiple of 4.

Note

- If DMA-based MSC write is enabled on the bootloader, writing data from flash to flash is not supported on Series-1 devices. DMA-based MSC write is enabled, both offset and buffer should be word aligned. In case the buffer is not aligned, the normal write procedure is used instead of DMA.

Returns

- BOOTLOADER_OK on success, else error code in BOOTLOADER_ERROR_STORAGE_BASE range

Definition at line 344 of file platform/bootloader/api/btl_interface_storage.h

bootloader_eraseWriteStorage

> int32_t bootloader_eraseWriteStorage (uint32_t slotId, uint32_t offset, uint8_t *buffer, size_t length)

Erase and write data to a storage slot.

Parameters

[in]	slotId	ID of the slot
[in]	offset	Offset into the slot to start writing to
[in]	buffer	Buffer to read data to write from
[in]	length	Amount of data to write. Must be a multiple of 4.

Note

- This function automatically erases the following Flash page whenever the written data crosses a page boundary. In other words, the function can't be used to perform multiple sequential writes to the same address range unless the range starts at a page boundary. For a sequential write, the first call to this function should have a start address at a page boundary. Otherwise, the corresponding page of the starting address needs to be erased explicitly. If DMA-based MSC write is enabled on the bootloader, writing data from flash to flash is not supported on Series-1 devices.

Returns

-

BOOTLOADER_OK on success, else error code in BOOTLOADER_ERROR_STORAGE_BASE range

Definition at line 370 of file platform/bootloader/api/btl_interface_storage.h

bootloader_eraseStorageSlot

```
int32_t bootloader_eraseStorageSlot (uint32_t slotId)
```

Erase all contents of a storage slot.

Parameters

[in]	slotId	ID of the slot

Returns

- BOOTLOADER_OK on success, else error code in BOOTLOADER_ERROR_STORAGE_BASE range

Definition at line 383 of file platform/bootloader/api/btl_interface_storage.h

bootloader_initChunkedEraseStorageSlot

```
int32_t bootloader_initChunkedEraseStorageSlot (uint32_t slotId, BootloaderEraseStatus_t *eraseStat)
```

Initialize chunked erase of a storage slot.

Parameters

[in]	slotId	ID of the slot
[in]	eraseStat	Erase status struct

Note

- This function must be called before calling bootloader_chunkedEraseStorageSlot in a loop.

Returns

- BOOTLOADER_OK on success, else error code in BOOTLOADER_ERROR_STORAGE_BASE range

Definition at line 397 of file platform/bootloader/api/btl_interface_storage.h

bootloader_chunkedEraseStorageSlot

```
int32_t bootloader_chunkedEraseStorageSlot (BootloaderEraseStatus_t *eraseStat)
```

Erase one page from a storage slot according to the struct BootloaderEraseStatus_t.

Parameters

[in]	eraseStat	Erase status struct

Note

- bootloader_initChunkedEraseStorageSlot must be called before calling this function, in order to prepare BootloaderEraseStatus_t.
- This can be called sequentially to, for example, erase all contents of a storage slot.

Returns

BOOTLOADER_ERROR_STORAGE_CONTINUE if erasing a page was successful. Erase can be continued by calling this function again. BOOTLOADER_OK if the entire slot has been erased, else error code in BOOTLOADER_ERROR_STORAGE_BASE range

Definition at line 418 of file platform/bootloader/api/btl_interface_storage.h

bootloader_setImagesToBootload

```
int32_t bootloader_setImagesToBootload (int32_t *slotIds, size_t length)
```

Set a prioritized list of images to attempt to bootload.

Parameters

[in]	slotIds	Prioritized list of slot IDs to attempt to bootload. The first image to pass verification will be bootloaded.
[in]	length	Length of the slotIds array

The last call to this function determines which slot will be installed when bootloader_rebootAndInstall is called.

Returns

- BOOTLOADER_OK on success, else error code in BOOTLOADER_ERROR_BOOTLOAD_BASE range

Definition at line 432 of file platform/bootloader/api/btl_interface_storage.h

bootloader_getImagesToBootload

```
int32_t bootloader_getImagesToBootload (int32_t *slotIds, size_t length)
```

Get the prioritized list of images the bootloader will attempt to bootload.

Parameters

[out]	slotIds	Prioritized list of slot IDs to attempt to bootload. The first image to pass verification will be bootloaded.
[in]	length	Length of the slotIds array

Returns

- BOOTLOADER_OK on success, else error code in BOOTLOADER_ERROR_BOOTLOAD_BASE range

Definition at line 444 of file platform/bootloader/api/btl_interface_storage.h

bootloader_appendImageToBootloadList

```
int32_t bootloader_appendImageToBootloadList (int32_t slotId)
```

Append a single image to the list of images to attempt to bootload.

Parameters

[in]	slotId	ID of the slot to append

Returns

- BOOTLOADER_OK on success, else error code in BOOTLOADER_ERROR_BOOTLOAD_BASE range

Definition at line 454 of file platform/bootloader/api/btl_interface_storage.h

bootloader_setImageToBootload

```
int32_t bootloader_setImageToBootload (int32_t slotId)
```

Set a single image to attempt to bootload.

Parameters

[in]	slotId	ID of the slot

Returns

- BOOTLOADER_OK on success, else error code in BOOTLOADER_ERROR_BOOTLOAD_BASE range

Definition at line 464 of file platform/bootloader/api/btl_interface_storage.h

bootloader_initVerifyImage

```
int32_t bootloader_initVerifyImage (uint32_t slotId, void *context, size_t contextSize)
```

Initialize image verification.

Parameters

[in]	slotId	ID of the slot to check.
N/A	context	Pointer to memory used to hold the parser context.
[in]	contextSize	Size of the context. An error is returned if the supplied context is too small.

Initialize verification of an upgrade image stored in a bootloader storage slot.

Note

- This function must be called before calling bootloader_continueVerifyImage in a loop.
- The context pointer must point to memory allocated by the caller. The caller must ensure that the context pointer is 32 bit aligned. The required size of this context may depend on the version of the bootloader. The required size for the bootloader associated with this version of the application interface is given by the define BOOTLOADER_STORAGE_VERIFICATION_CONTEXT_SIZE.
- Instead of calling bootloader_initVerifyImage followed by bootloader_continueVerifyImage, call bootloader_verifyImage if no time-critical tasks are needed and sufficient stack space is available for the automatically allocated context. The purpose of the init-and-continue functions is to allow the caller to service system needs during verification.

Returns

- BOOTLOADER_OK if the image parser was initialized, else error code.

Definition at line 500 of file platform/bootloader/api/btl_interface_storage.h

bootloader_continueVerifyImage

```
int32_t bootloader_continueVerifyImage (void *context, BootloaderParserCallback_t metadataCallback)
```

Continue image verification.

Parameters

N/A	context	Pointer to a context structure that has been initialized by calling bootloader_initVerifyImage()

[in]	metadataCallback	Function pointer, which is called when the binary metadata in the image is currently verified. Set to NULL if not required.

Continue verification of an upgrade image stored in a bootloader storage slot.

Note

- This function must be called in a loop until anything other than BOOTLOADER_ERROR_PARSE_CONTINUE is returned.
- bootloader_initVerifyImage must be called before calling this function to reset the parser.
- Instead of calling bootloader_initVerifyImage followed by bootloader_continueVerifyImage, call bootloader_verifyImage if no time-critical tasks are needed. The purpose of the init-and-continue functions is to allow the caller to service system needs during verification.

Returns

- BOOTLOADER_ERROR_PARSE_CONTINUE if parsing was successful, and the parser expects more data. BOOTLOADER_ERROR_PARSE_SUCCESS if the parser has successfully parsed the image and it passes verification. Else error code.

Definition at line 540 of file platform/bootloader/api/btl_interface_storage.h

bootloader_verifyImage

```
int32_t bootloader_verifyImage (uint32_t slotId, BootloaderParserCallback_t metadataCallback)
```

Verify that the image in the given storage slot is valid.

Parameters

[in]	slotId	ID of the slot to check
[in]	metadataCallback	Function pointer, which is called when binary metadata is present in the storage slot. Set to NULL if not required.

Note

- This function allocates a context structure of the size given by BOOTLOADER_STORAGE_VERIFICATION_CONTEXT_SIZE on the caller's stack. To manage memory and allocate the context elsewhere (on the heap, as a global variable, and so on), use bootloader_initVerifyImage and bootloader_continueVerifyImage functions instead.

Returns

- BOOTLOADER_OK if the image is valid, else error code.

Definition at line 564 of file platform/bootloader/api/btl_interface_storage.h

bootloader_getImageInfo

```
int32_t bootloader_getImageInfo (uint32_t slotId, ApplicationData_t *appInfo, uint32_t *bootloaderVersion)
```

Get application and bootloader upgrade metadata from the storage slot.

Parameters

[in]	slotId	ID of the slot to check
[out]	appInfo	Pointer to ApplicationData_t struct
[out]	bootloaderVersion	Pointer to an integer representing bootloader version

Returns

BOOTLOADER_OK if metadata was filled successfully

Definition at line 580 of file platform/bootloader/api/btl_interface_storage.h

bootloader_storageIsBusy

```
bool bootloader_storageIsBusy (void)
```

Check whether the bootloader storage is busy.

Parameters

N/A		

Returns

- True if the storage is busy

Definition at line 589 of file platform/bootloader/api/btl_interface_storage.h

bootloader_readRawStorage

```
int32_t bootloader_readRawStorage (uint32_t address, uint8_t *buffer, size_t length)
```

Read raw data from storage.

Parameters

[in]	address	Address to start reading from
[out]	buffer	Buffer to store the data
[in]	length	Amount of data to read

Returns

- BOOTLOADER_OK on success, else error code in BOOTLOADER_ERROR_STORAGE_BASE range

Definition at line 602 of file platform/bootloader/api/btl_interface_storage.h

bootloader_writeRawStorage

```
int32_t bootloader_writeRawStorage (uint32_t address, uint8_t *buffer, size_t length)
```

Write data to storage.

Parameters

[in]	address	Address to start writing to
[in]	buffer	Buffer to read data to write from
[in]	length	Amount of data to write. Must be a multiple of 4.

Note

- If DMA-based MSC write is enabled on the bootloader, writing data from flash to flash is not supported on Series-1 devices.

Returns

- BOOTLOADER_OK on success, else error code in BOOTLOADER_ERROR_STORAGE_BASE range

Definition at line 620 of file platform/bootloader/api/btl_interface_storage.h

bootloader_eraseRawStorage

```
int32_t bootloader_eraseRawStorage (uint32_t address, size_t length)
```

Erase data from storage.

Parameters

[in]	address	Address to start erasing from
[in]	length	Length of data to erase

Note

- Erasing storage must adhere to the limitations of the underlying storage medium, such as requiring full page erases. Use bootloader_getStorageInfo to learn about the limitations of the configured storage medium.

Returns

- BOOTLOADER_OK on success, else error code in BOOTLOADER_ERROR_STORAGE_BASE range

Definition at line 638 of file platform/bootloader/api/btl_interface_storage.h

bootloader_getAllocatedDMAChannel

```
int32_t bootloader_getAllocatedDMAChannel (void)
```

Get allocated DMA channel for MSC write.

Parameters

N/A		

Returns

- A positive number channel. -1 if DMA-based MSC write is not enabled. Otherwise, the error code BOOTLOADER_ERROR_INIT_STORAGE.

Definition at line 648 of file platform/bootloader/api/btl_interface_storage.h

Macro Definition Documentation

BOOTLOADER_STORAGE_VERIFICATION_CONTEXT_SIZE

```
#define BOOTLOADER_STORAGE_VERIFICATION_CONTEXT_SIZE
```

Value:

```
(384)
```

Context size for bootloader verification context.

Definition at line 216 of file platform/bootloader/api/btl_interface_storage.h

BOOTLOADER_STORAGE_INFO_VERSION

```
#define BOOTLOADER_STORAGE_INFO_VERSION
```

Value:

```
(0x30000U)
```

Current version of the BootloaderStorageInformation_t struct.

Definition at line 220 of file platform/bootloader/api/btl_interface_storage.h

BOOTLOADER_STORAGE_IMPL_INFO_VERSION

```
#define BOOTLOADER_STORAGE_IMPL_INFO_VERSION
```

Value:

```
(0x0210U)
```

Current version of the BootloaderStorageImplementationInformation_t struct.

Definition at line 222 of file platform/bootloader/api/btl_interface_storage.h

BOOTLOADER_STORAGE_IMPL_INFO_VERSION_MAJOR

```
#define BOOTLOADER_STORAGE_IMPL_INFO_VERSION_MAJOR
```

Value:

```
(0x0200U)
```

Major version of the BootloaderStorageImplementationInformation_t struct.

Definition at line 224 of file platform/bootloader/api/btl_interface_storage.h

BOOTLOADER_STORAGE_IMPL_INFO_VERSION_MAJOR_MASK

```
#define BOOTLOADER_STORAGE_IMPL_INFO_VERSION_MAJOR_MASK
```

Value:

```
(0xFF00U)
```

Major version mask for BOOTLOADER_STORAGE_IMPL_INFO_VERSION.

Definition at line 226 of file platform/bootloader/api/btl_interface_storage.h

BOOTLOADER_STORAGE_IMPL_CAPABILITY_ERASE_SUPPORTED

```
#define BOOTLOADER_STORAGE_IMPL_CAPABILITY_ERASE_SUPPORTED
```

Value:

```
(1 << 0)
```

Spiflash capability indicating that it supports erase.

Definition at line 229 of file platform/bootloader/api/btl_interface_storage.h

BOOTLOADER_STORAGE_IMPL_CAPABILITY_PAGE_ERASE_REQUIRED

```
#define BOOTLOADER_STORAGE_IMPL_CAPABILITY_PAGE_ERASE_REQUIRED
```

Value:

```
(1 << 1)
```

Spiflash capability indicating it requires full page erases before new data can be written.

Definition at line 232 of file platform/bootloader/api/btl_interface_storage.h

BOOTLOADER_STORAGE_IMPL_CAPABILITY_BLOCKING_WRITE

```
#define BOOTLOADER_STORAGE_IMPL_CAPABILITY_BLOCKING_WRITE
```

Value:

```
(1 << 2)
```

Spiflash capability indicating that the write function is blocking.

Definition at line 234 of file platform/bootloader/api/btl_interface_storage.h

BOOTLOADER_STORAGE_IMPL_CAPABILITY_BLOCKING_ERASE

```
#define BOOTLOADER_STORAGE_IMPL_CAPABILITY_BLOCKING_ERASE
```

Value:

```
(1 << 3)
```

Spiflash capability indicating that the erase function is blocking.

Definition at line 236 of file platform/bootloader/api/btl_interface_storage.h

BOOTLOADER_STORAGE_ISSI_IS25LQ040B

```
#define BOOTLOADER_STORAGE_ISSI_IS25LQ040B
```

Value:

```
(1U << 0)
```

ISSI IS25LQ040B SPI Flash.

Definition at line 239 of file platform/bootloader/api/btl_interface_storage.h

BOOTLOADER_STORAGE_ISSI_IS25LQ020B

```
#define BOOTLOADER_STORAGE_ISSI_IS25LQ020B
```

Value:

```
(1U << 1)
```

ISSI IS25LQ020B SPI Flash.

Definition at line 241 of file platform/bootloader/api/btl_interface_storage.h

BOOTLOADER_STORAGE_ISSI_IS25LQ010B

```
#define BOOTLOADER_STORAGE_ISSI_IS25LQ010B
```

Value:

```
(1U << 2)
```

ISSI IS25LQ010B SPI Flash.

Definition at line 243 of file platform/bootloader/api/btl_interface_storage.h

BOOTLOADER_STORAGE_ISSI_IS25LQ512B

```
#define BOOTLOADER_STORAGE_ISSI_IS25LQ512B
```

Value:

```
(1U << 3)
```

ISSI IS25LQ512B SPI Flash.

Definition at line 245 of file platform/bootloader/api/btl_interface_storage.h

BOOTLOADER_STORAGE_ISSI_IS25LQ025B

```
#define BOOTLOADER_STORAGE_ISSI_IS25LQ025B
```

Value:

```
(1U << 4)
```

ISSI IS25LQ025B SPI Flash.

Definition at line 247 of file platform/bootloader/api/btl_interface_storage.h

BOOTLOADER_STORAGE_NUMONYX_M25P16

```
#define BOOTLOADER_STORAGE_NUMONYX_M25P16
```

Value:

```
(1U << 5)
```

Numonyx M25P16 SPI Flash.

Definition at line 249 of file platform/bootloader/api/btl_interface_storage.h

BOOTLOADER_STORAGE_NUMONYX_M25P80

```
#define BOOTLOADER_STORAGE_NUMONYX_M25P80
```

Value:

```
(1U << 6)
```

Numonyx M25P80 SPI Flash.

Definition at line 251 of file platform/bootloader/api/btl_interface_storage.h

BOOTLOADER_STORAGE_NUMONYX_M25P40

```
#define BOOTLOADER_STORAGE_NUMONYX_M25P40
```

Value:

```
(1U << 7)
```

Numonyx M25P40 SPI Flash.

Definition at line 253 of file platform/bootloader/api/btl_interface_storage.h

BOOTLOADER_STORAGE_NUMONYX_M25P20

```
#define BOOTLOADER_STORAGE_NUMONYX_M25P20
```

Value:

```
(1U << 8)
```

Numonyx M25P20 SPI Flash.

Definition at line 255 of file platform/bootloader/api/btl_interface_storage.h

BOOTLOADER_STORAGE_ADESTO_AT25SF041

```
#define BOOTLOADER_STORAGE_ADESTO_AT25SF041
```

Value:

```
(1U << 9)
```

Adesto AT25SF041 SPI Flash.

Definition at line 257 of file platform/bootloader/api/btl_interface_storage.h

BOOTLOADER_STORAGE_ATMEL_AT25DF081A

```
#define BOOTLOADER_STORAGE_ATMEL_AT25DF081A
```

Value:

```
(1U << 10)
```

Atmel AT25DF081A SPI Flash.

Definition at line 259 of file platform/bootloader/api/btl_interface_storage.h

BOOTLOADER_STORAGE_ATMEL_AT25DF041A

```
#define BOOTLOADER_STORAGE_ATMEL_AT25DF041A
```

Value:

```
(1U << 11)
```

Atmel AT25DF041A SPI Flash.

Definition at line 261 of file platform/bootloader/api/btl_interface_storage.h

BOOTLOADER_STORAGE_MACRONIX_MX25R6435F

```
#define BOOTLOADER_STORAGE_MACRONIX_MX25R6435F
```

Value:

```
(1U << 12)
```

Macronix MX25R6435F SPI Flash.

Definition at line 263 of file platform/bootloader/api/btl_interface_storage.h

BOOTLOADER_STORAGE_MACRONIX_MX25R3235F

```
#define BOOTLOADER_STORAGE_MACRONIX_MX25R3235F
```

Value:

```
(1U << 13)
```

Macronix MX25R6435F SPI Flash.

Definition at line 265 of file platform/bootloader/api/btl_interface_storage.h

BOOTLOADER_STORAGE_MACRONIX_MX25U1635E

```
#define BOOTLOADER_STORAGE_MACRONIX_MX25U1635E
```

Value:

```
(1U << 14)
```

Macronix MX25U1635E SPI Flash.

Definition at line 267 of file platform/bootloader/api/btl_interface_storage.h

BOOTLOADER_STORAGE_MACRONIX_MX25L1606E

```
#define BOOTLOADER_STORAGE_MACRONIX_MX25L1606E
```

Value:

```
(1U << 15)
```

Macronix MX25L1606E SPI Flash.

Definition at line 269 of file platform/bootloader/api/btl_interface_storage.h

BOOTLOADER_STORAGE_MACRONIX_MX25R8035F

```
#define BOOTLOADER_STORAGE_MACRONIX_MX25R8035F
```

Value:

```
(1U << 16)
```

Macronix MX25R8035F SPI Flash.

Definition at line 271 of file platform/bootloader/api/btl_interface_storage.h

BOOTLOADER_STORAGE_MACRONIX_MX25L8006E

```
#define BOOTLOADER_STORAGE_MACRONIX_MX25L8006E
```

Value:

```
(1U << 17)
```

Macronix MX25L8006E SPI Flash.

Definition at line 273 of file platform/bootloader/api/btl_interface_storage.h

BOOTLOADER_STORAGE_MACRONIX_MX25L4006E

```
#define BOOTLOADER_STORAGE_MACRONIX_MX25L4006E
```

Value:

```
(1U << 18)
```

Macronix MX25L4006E SPI Flash.

Definition at line 275 of file platform/bootloader/api/btl_interface_storage.h

BOOTLOADER_STORAGE_MACRONIX_MX25L2006E

```
#define BOOTLOADER_STORAGE_MACRONIX_MX25L2006E
```

Value:

```
(1U << 19)
```

Macronix MX25L2006E SPI Flash.

Definition at line 277 of file platform/bootloader/api/btl_interface_storage.h

BOOTLOADER_STORAGE_WINBOND_W25Q80BV

```
#define BOOTLOADER_STORAGE_WINBOND_W25Q80BV
```

Value:

```
(1U << 20)
```

Winbond W25Q80BV SPI Flash.

Definition at line 279 of file platform/bootloader/api/btl_interface_storage.h

BOOTLOADER_STORAGE_WINBOND_W25X20BV

```
#define BOOTLOADER_STORAGE_WINBOND_W25X20BV
```

Value:

```
(1U << 21)
```

Winbond W25X20BV SPI Flash.

Definition at line 281 of file platform/bootloader/api/btl_interface_storage.h

BOOTLOADER_STORAGE_SPANSION_S25FL208K

```
#define BOOTLOADER_STORAGE_SPANSION_S25FL208K
```

Value:

```
(1U << 22)
```

Spansion S25L208K SPI Flash.

Definition at line 283 of file platform/bootloader/api/btl_interface_storage.h

BOOTLOADER_STORAGE_INTERNAL_STORAGE

```
#define BOOTLOADER_STORAGE_INTERNAL_STORAGE
```

Value:

```
(1U << 30)
```

Internal storage.

Definition at line 285 of file platform/bootloader/api/btl_interface_storage.h

BOOTLOADER_STORAGE_JEDEC

```
#define BOOTLOADER_STORAGE_JEDEC
```

Value:

```
(1U << 31)
```

JEDEC Supported SPI Flash.

Definition at line 287 of file platform/bootloader/api/btl_interface_storage.h

BootloaderStorageSlot_t

Information about a storage slot.

Public Attributes

uint32_t	address Address of the slot.
uint32_t	length Size of the slot.

Public Attribute Documentation

address

```
uint32_t BootloaderStorageSlot_t::address
```

Address of the slot.

Definition at line 93 of file platform/bootloader/api/btl_interface_storage.h

length

```
uint32_t BootloaderStorageSlot_t::length
```

Size of the slot.

Definition at line 95 of file platform/bootloader/api/btl_interface_storage.h

BootloaderStorageImplementationInformation_t

Information about the bootloader storage implementation **Note:** From Gecko Bootloader version >= 2.1, the pointer **partType** will only contain a zero value.

The **partType** variable can be used to find information about the attached storage.

Public Attributes

uint16_t	version The version of this data structure.
uint16_t	capabilitiesMask A bitmask describing the capabilities of this particular storage.
uint32_t	pageEraseMs Maximum time it takes to erase a page. (in milliseconds)
uint32_t	partEraseMs Maximum time it takes to erase the entire part. (in milliseconds)
uint32_t	pageSize The size of a single erasable page in bytes.
uint32_t	partSize The total size of the storage in bytes.
char *	partDescription Pointer to a string describing the attached storage.
uint8_t	wordSizeBytes The number of bytes in a word for the storage.
uint32_t	partType Value representing the attached storage.

Public Attribute Documentation

version

```
uint16_t BootloaderStorageImplementationInformation_t::version
```

The version of this data structure.

Definition at line `105` of file `platform/bootloader/api/btl_interface_storage.h`

capabilitiesMask

```
uint16_t BootloaderStorageImplementationInformation_t::capabilitiesMask
```

A bitmask describing the capabilities of this particular storage.

Definition at line 107 of file platform/bootloader/api/btl_interface_storage.h

pageEraseMs

```
uint32_t BootloaderStorageImplementationInformation_t::pageEraseMs
```

Maximum time it takes to erase a page. (in milliseconds)

Definition at line 109 of file platform/bootloader/api/btl_interface_storage.h

partEraseMs

```
uint32_t BootloaderStorageImplementationInformation_t::partEraseMs
```

Maximum time it takes to erase the entire part. (in milliseconds)

Definition at line 111 of file platform/bootloader/api/btl_interface_storage.h

pageSize

```
uint32_t BootloaderStorageImplementationInformation_t::pageSize
```

The size of a single erasable page in bytes.

Definition at line 113 of file platform/bootloader/api/btl_interface_storage.h

partSize

```
uint32_t BootloaderStorageImplementationInformation_t::partSize
```

The total size of the storage in bytes.

Definition at line 115 of file platform/bootloader/api/btl_interface_storage.h

partDescription

```
char* BootloaderStorageImplementationInformation_t::partDescription
```

Pointer to a string describing the attached storage.

Definition at line 117 of file platform/bootloader/api/btl_interface_storage.h

wordSizeBytes

```
uint8_t BootloaderStorageImplementationInformation_t::wordSizeBytes
```

The number of bytes in a word for the storage.

Definition at line 119 of file platform/bootloader/api/btl_interface_storage.h

partType

```
uint32_t BootloaderStorageImplementationInformation_t::partType
```

Value representing the attached storage.

Definition at line 121 of file platform/bootloader/api/btl_interface_storage.h

BootloaderStorageInformation_t

Information about the bootloader storage **Note:** The **flashInfo** variable is only usable with Gecko Bootloader version >= 2.0.

All previous versions of the Gecko Bootloader do not support the **flashInfo** data field.

Public Attributes

uint32_t	version The version of this data structure.
uint32_t	capabilities The capabilities of the storage component.
BootloaderStorageType_t	storageType Type of storage.
uint32_t	numStorageSlots Number of storage slots.
BootloaderStorageImplementationInformation_t *	info A pointer to detailed information about the attached storage.
BootloaderStorageImplementationInformation_t	flashInfo Detailed information about the attached storage(**available for use only with Gecko Bootloader version >= 2.0**)

Public Attribute Documentation

version

```
uint32_t BootloaderStorageInformation_t::version
```

The version of this data structure.

Definition at line `130` of file `platform/bootloader/api/btl_interface_storage.h`

capabilities

```
uint32_t BootloaderStorageInformation_t::capabilities
```

The capabilities of the storage component.

Definition at line `132` of file `platform/bootloader/api/btl_interface_storage.h`

storageType

```
BootloaderStorageType_t BootloaderStorageInformation_t::storageType
```

Type of storage.

Definition at line 134 of file platform/bootloader/api/btl_interface_storage.h

numStorageSlots

```
uint32_t BootloaderStorageInformation_t::numStorageSlots
```

Number of storage slots.

Definition at line 136 of file platform/bootloader/api/btl_interface_storage.h

info

```
BootloaderStorageImplementationInformation_t* BootloaderStorageInformation_t::info
```

A pointer to detailed information about the attached storage.

Definition at line 138 of file platform/bootloader/api/btl_interface_storage.h

flashInfo

```
BootloaderStorageImplementationInformation_t BootloaderStorageInformation_t::flashInfo
```

Detailed information about the attached storage(**available for use only with Gecko Bootloader version >= 2.0**)

Definition at line 140 of file platform/bootloader/api/btl_interface_storage.h

BootloaderEraseStatus_t

Erase status struct.

Public Attributes

uint32_t	currentPageAddr Address of the current page to be erased.
uint32_t	pageSize The size of a single erasable page in bytes.
BootloaderStorageSlot_t	storageSlotInfo Information about a storage slot.

Public Attribute Documentation

currentPageAddr

```
uint32_t BootloaderEraseStatus_t::currentPageAddr
```

Address of the current page to be erased.

Definition at line 146 of file platform/bootloader/api/btl_interface_storage.h

pageSize

```
uint32_t BootloaderEraseStatus_t::pageSize
```

The size of a single erasable page in bytes.

Definition at line 148 of file platform/bootloader/api/btl_interface_storage.h

storageSlotInfo

```
BootloaderStorageSlot_t BootloaderEraseStatus_t::storageSlotInfo
```

Information about a storage slot.

Definition at line 150 of file platform/bootloader/api/btl_interface_storage.h

BootloaderStorageFunctions_t

Storage API accessible from the application.

Public Attributes

uint32_t	version Version of this struct.
void(*	getInfo Get information about the storage – capabilities, layout, configuration.
int32_t(*	getSlotInfo Get information about storage slot – size, location.
int32_t(*	read Read bytes from slot into buffer.
int32_t(*	write Write bytes from buffer into slot.
int32_t(*	erase Erase an entire slot.
int32_t(*	setImagesToBootload Mark a list of slots for bootload.
int32_t(*	getImagesToBootload Mark a list of slots for bootload.
int32_t(*	appendImageToBootloadList Append a slot to bootload list.
int32_t(*	initParseImage Start image parsing.
int32_t(*	verifyImage Continue image verification.
int32_t(*	getImageInfo Get app and bootloader upgrade information from storage slot.
bool(*	isBusy Check whether the bootloader storage is busy.
int32_t(*	readRaw Read raw bytes from storage.
int32_t(*	writeRaw Write bytes to raw storage.
int32_t(*	eraseRaw Erase storage.
int32_t(*	getDMAchannel Get configured DMA channel.

Public Attribute Documentation

version

```
uint32_t BootloaderStorageFunctions_t::version
```

Version of this struct.

Definition at line 156 of file platform/bootloader/api/btl_interface_storage.h

getInfo

```
void(* BootloaderStorageFunctions_t::getInfo) (BootloaderStorageInformation_t *info)
```

Get information about the storage – capabilities, layout, configuration.

Definition at line 158 of file platform/bootloader/api/btl_interface_storage.h

getSlotInfo

```
int32_t(* BootloaderStorageFunctions_t::getSlotInfo) (uint32_t slotId, BootloaderStorageSlot_t *slot)
```

Get information about storage slot – size, location.

Definition at line 160 of file platform/bootloader/api/btl_interface_storage.h

read

```
int32_t(* BootloaderStorageFunctions_t::read) (uint32_t slotId, uint32_t offset, uint8_t *buffer, size_t length)
```

Read bytes from slot into buffer.

Definition at line 162 of file platform/bootloader/api/btl_interface_storage.h

write

```
int32_t(* BootloaderStorageFunctions_t::write) (uint32_t slotId, uint32_t offset, uint8_t *buffer, size_t length)
```

Write bytes from buffer into slot.

Definition at line 167 of file platform/bootloader/api/btl_interface_storage.h

erase

```
int32_t(* BootloaderStorageFunctions_t::erase) (uint32_t slotId)
```

Erase an entire slot.

Definition at line 172 of file platform/bootloader/api/btl_interface_storage.h

setImagesToBootload

```
int32_t(* BootloaderStorageFunctions_t::setImagesToBootload) (int32_t *slotIds, size_t length)
```

Mark a list of slots for bootload.

Definition at line 175 of file platform/bootloader/api/btl_interface_storage.h

getImagesToBootload

```
int32_t(* BootloaderStorageFunctions_t::getImagesToBootload) (int32_t *slotIds, size_t length)
```

Mark a list of slots for bootload.

Definition at line 177 of file platform/bootloader/api/btl_interface_storage.h

appendImageToBootloadList

```
int32_t(* BootloaderStorageFunctions_t::appendImageToBootloadList) (int32_t slotId)
```

Append a slot to bootload list.

Definition at line 179 of file platform/bootloader/api/btl_interface_storage.h

initParseImage

```
int32_t(* BootloaderStorageFunctions_t::initParseImage) (uint32_t slotId, BootloaderParserContext_t *context, size_t contextSize)
```

Start image parsing.

Definition at line 182 of file platform/bootloader/api/btl_interface_storage.h

verifyImage

```
int32_t(* BootloaderStorageFunctions_t::verifyImage) (BootloaderParserContext_t *context, BootloaderParserCallback_t metadataCallback)
```

Continue image verification.

Definition at line 186 of file platform/bootloader/api/btl_interface_storage.h

getImageInfo

```
int32_t(* BootloaderStorageFunctions_t::getImageInfo) (BootloaderParserContext_t *context, ApplicationData_t *appInfo, uint32_t *bootloaderVersion)
```

Get app and bootloader upgrade information from storage slot.

Definition at line `189` of file `platform/bootloader/api/btl_interface_storage.h`

isBusy

```
bool(* BootloaderStorageFunctions_t::isBusy) (void)
```

Check whether the bootloader storage is busy.

Definition at line `193` of file `platform/bootloader/api/btl_interface_storage.h`

readRaw

```
int32_t(* BootloaderStorageFunctions_t::readRaw) (uint32_t address, uint8_t *buffer, size_t length)
```

Read raw bytes from storage.

Definition at line `195` of file `platform/bootloader/api/btl_interface_storage.h`

writeRaw

```
int32_t(* BootloaderStorageFunctions_t::writeRaw) (uint32_t address, uint8_t *buffer, size_t length)
```

Write bytes to raw storage.

Definition at line `197` of file `platform/bootloader/api/btl_interface_storage.h`

eraseRaw

```
int32_t(* BootloaderStorageFunctions_t::eraseRaw) (uint32_t address, size_t length)
```

Erase storage.

Definition at line `199` of file `platform/bootloader/api/btl_interface_storage.h`

getDMAchannel

```
int32_t(* BootloaderStorageFunctions_t::getDMAchannel) (void)
```

Get configured DMA channel.

Definition at line `201` of file `platform/bootloader/api/btl_interface_storage.h`

Common Application Interface

Common Application Interface

Generic application interface available on all versions of the bootloader, regardless of the available components.

Note

- These Bootloader APIs are not reentrant and should be wrapped in critical section where needed.

Modules

BareBootTable_t

BootloaderInformation_t

BootloaderHeader_t

FirstBootloaderTable_t

MainBootloaderTable_t

Reset Information

Enumerations

enum BootloaderType_t {

NO_BOOTLOADER = 0
SL_BOOTLOADER = 1

}

Bootloader interface APIs are trust zone aware.

Functions

void bootloader_getInfo(BootloaderInformation_t *info)
Get information about the bootloader on this device.

int32_t bootloader_init(void)
Initialize components of the bootloader so the app can use the interface.

int32_t bootloader_deinit(void)
De-initialize components of the bootloader that were previously initialized.

void bootloader_rebootAndInstall(void)
Reboot into the bootloader to perform an install.

bool bootloader_verifyApplication(uint32_t startAddress)
Verify the application image stored in the Flash memory starting at the address startAddress.

bool bootloader_secureBootEnforced(void)
Check whether signature verification on the application image in internal flash is enforced before every boot.

bool bootloader_getUpgradeLocation(uint32_t *location)
Get base address of the bootloader upgrade image.

uint32_t bootloader_remainingApplicationUpgrades(void)
Count the total remaining number of application upgrades.

BootloaderResetCause_t bootloader_getResetReason(void)
Get reset cause of the bootloader.

bool bootloader_pointerValid(const void *ptr)
Check if a pointer is valid and if it points to the bootloader main stage.

Macros

#define BOOTLOADER_VERSION_MAJOR_SHIFT (24U)
Bootloader version major version shift value.

#define BOOTLOADER_VERSION_MINOR_SHIFT (16U)
Bootloader version minor version shift value.

#define BOOTLOADER_VERSION_MAJOR_MASK (0xFF000000U)
Bootloader version major version mask.

#define BOOTLOADER_VERSION_MINOR_MASK (0×00FF0000U)
Bootloader version minor version mask.

#define BOOTLOADER_CAPABILITY_ENFORCE_UPGRADE_SIGNATURE (1 << 0)
Bootloader enforces signed application upgrade images.

#define BOOTLOADER_CAPABILITY_ENFORCE_UPGRADE_ENCRYPTION (1 << 1)
Bootloader enforces encrypted application upgrade images.

#define BOOTLOADER_CAPABILITY_ENFORCE_SECURE_BOOT (1 << 2)
Bootloader enforces signature verification of the application image before every boot.

#define BOOTLOADER_CAPABILITY_BOOTLOADER_UPGRADE (1 << 4)
Bootloader has the capability of being upgraded.

#define BOOTLOADER_CAPABILITY_GBL (1 << 5)
Bootloader has the capability of parsing GBL files.

#define BOOTLOADER_CAPABILITY_GBL_SIGNATURE (1 << 6)
Bootloader has the capability of parsing signed GBL files.

#define BOOTLOADER_CAPABILITY_GBL_ENCRYPTION (1 << 7)
Bootloader has the capability of parsing encrypted GBL files.

#define BOOTLOADER_CAPABILITY_ENFORCE_CERTIFICATE_SECURE_BOOT (1 << 8)
Bootloader enforces signature verification of the application image before every boot using certificate.

#define BOOTLOADER_CAPABILITY_ROLLBACK_PROTECTION (1 << 9)
Bootloader has the capability of application rollback protection.

#define BOOTLOADER_CAPABILITY_PERIPHERAL_LIST (1 << 10)
Bootloader has the capability to check the peripherals in use.

#define BOOTLOADER_CAPABILITY_STORAGE (1 << 16)
Bootloader has the capability of storing data in an internal or external storage medium.

#define BOOTLOADER_CAPABILITY_COMMUNICATION (1 << 20)
Bootloader has the capability of communicating with host processors using a communication interface.

#define BOOTLOADER_MAGIC_FIRST_STAGE (0xB00710ADUL)
Magic word indicating first stage bootloader table.

#define BOOTLOADER_MAGIC_MAIN (0x5ECDB007UL)
Magic word indicating main bootloader table.

#define mainBootloaderTable undefined
Pointer to main bootloader table.

Enumeration Documentation

BootloaderType_t

> BootloaderType_t

Bootloader interface APIs are trust zone aware.

Type of bootloader

Enumerator	
NO_BOOTLOADER	No bootloader present.
SL_BOOTLOADER	Bootloader is a Silicon Labs bootloader.

Definition at line `107` of file `platform/bootloader/api/btl_interface.h`

Function Documentation

bootloader_getInfo

> void bootloader_getInfo (BootloaderInformation_t *info)

Get information about the bootloader on this device.

Parameters

[out]	info	Pointer to the bootloader information struct.

The returned information is fetched from the main bootloader information table.

Definition at line `487` of file `platform/bootloader/api/btl_interface.h`

bootloader_init

> int32_t bootloader_init (void)

Initialize components of the bootloader so the app can use the interface.

Parameters

N/A		

This typically includes initializing serial peripherals for communication with external SPI flashes, and so on.

Returns

- Error code. BOOTLOADER_OK on success, else error code in BOOTLOADER_ERROR_INIT_BASE range.

Definition at line 497 of file platform/bootloader/api/btl_interface.h

bootloader_deinit

```
int32_t bootloader_deinit (void)
```

De-initialize components of the bootloader that were previously initialized.

Parameters

N/A		

This typically includes powering down external SPI flashes and de-initializing the serial peripheral used for communication with the external flash.

Returns

- Error code. BOOTLOADER_OK on success, else error code in BOOTLOADER_ERROR_INIT_BASE range.

Definition at line 508 of file platform/bootloader/api/btl_interface.h

bootloader_rebootAndInstall

```
void bootloader_rebootAndInstall (void)
```

Reboot into the bootloader to perform an install.

Parameters

N/A		

If there is a storage component and a slot is marked for bootload, install the image in that slot after verifying it.

If a communication component is present, open the communication channel and receive an image to be installed.

Definition at line 519 of file platform/bootloader/api/btl_interface.h

bootloader_verifyApplication

```
bool bootloader_verifyApplication (uint32_t startAddress)
```

Verify the application image stored in the Flash memory starting at the address startAddress.

Parameters

[in]	startAddress	Starting address of the application.

If the secure boot is enforced, the function will only return true if the cryptographic signature of the application is valid. Otherwise, the application is verified according to the signature type defined in the ApplicationProperties_t structure embedded in the application. Silicon Labs wireless stacks include a declaration of this structure. However, applications not using a full wireless stack may need to instantiate the structure.

Examples of results when the secure boot is not enforced:

- App has no signature: Valid if initial stack pointer and program counter have reasonable values.
- App has CRC checksum: Valid if checksum is valid.
- App has ECDSA signature: Valid if ECDSA signature is valid.

When secure boot is enforced, only ECDSA signed applications with a valid signature are considered valid.

Returns

- True if the application is valid, else false.

Definition at line 547 of file platform/bootloader/api/btl_interface.h

bootloader_secureBootEnforced

```
bool bootloader_secureBootEnforced (void)
```

Check whether signature verification on the application image in internal flash is enforced before every boot.

Parameters

N/A		

Returns

- True if signature verification is enforced, else false.

Definition at line 555 of file platform/bootloader/api/btl_interface.h

bootloader_getUpgradeLocation

```
bool bootloader_getUpgradeLocation (uint32_t *location)
```

Get base address of the bootloader upgrade image.

Parameters

[out]	location	the base address of bootloader upgrade image.

Returns

- Returns true if the location was found.

Definition at line 564 of file platform/bootloader/api/btl_interface.h

bootloader_remainingApplicationUpgrades

```
uint32_t bootloader_remainingApplicationUpgrades (void)
```

Count the total remaining number of application upgrades.

Parameters

N/A		

Returns

- remaining number of application upgrades.

Definition at line 629 of file platform/bootloader/api/btl_interface.h

bootloader_getResetReason

```
BootloaderResetCause_t bootloader_getResetReason (void)
```

Get reset cause of the bootloader.

Parameters

N/A		

Returns

- Reset cause of the bootloader.

Definition at line 648 of file platform/bootloader/api/btl_interface.h

bootloader_pointerValid

```
bool bootloader_pointerValid (const void *ptr)
```

Check if a pointer is valid and if it points to the bootloader main stage.

Parameters

[in]	ptr	The pointer to check

This function checks pointers to bootloader jump tables.

Returns

- True if the pointer points into the bootloader main stage, false if not.

Definition at line 701 of file platform/bootloader/api/btl_interface.h

Macro Definition Documentation

BOOTLOADER_VERSION_MAJOR_SHIFT

```
#define BOOTLOADER_VERSION_MAJOR_SHIFT
```

Value:

```
(24U)
```

Bootloader version major version shift value.

Definition at line 86 of file platform/bootloader/api/btl_interface.h

BOOTLOADER_VERSION_MINOR_SHIFT

```
#define BOOTLOADER_VERSION_MINOR_SHIFT
```

Value:

```
(16U)
```

Bootloader version minor version shift value.

Definition at line 88 of file platform/bootloader/api/btl_interface.h

BOOTLOADER_VERSION_MAJOR_MASK

```
#define BOOTLOADER_VERSION_MAJOR_MASK
```

Value:

```
(0xFF000000U)
```

Bootloader version major version mask.

Definition at line 90 of file platform/bootloader/api/btl_interface.h

BOOTLOADER_VERSION_MINOR_MASK

```
#define BOOTLOADER_VERSION_MINOR_MASK
```

Value:

```
(0x00FF0000U)
```

Bootloader version minor version mask.

Definition at line 92 of file platform/bootloader/api/btl_interface.h

BOOTLOADER_CAPABILITY_ENFORCE_UPGRADE_SIGNATURE

```
#define BOOTLOADER_CAPABILITY_ENFORCE_UPGRADE_SIGNATURE
```

Value:

```
(1 << 0)
```

Bootloader enforces signed application upgrade images.

Definition at line 225 of file platform/bootloader/api/btl_interface.h

BOOTLOADER_CAPABILITY_ENFORCE_UPGRADE_ENCRYPTION

```
#define BOOTLOADER_CAPABILITY_ENFORCE_UPGRADE_ENCRYPTION
```

Value:

```
(1 << 1)
```

Bootloader enforces encrypted application upgrade images.

Definition at line 227 of file platform/bootloader/api/btl_interface.h

BOOTLOADER_CAPABILITY_ENFORCE_SECURE_BOOT

```
#define BOOTLOADER_CAPABILITY_ENFORCE_SECURE_BOOT
```

Value:

```
(1 << 2)
```

Bootloader enforces signature verification of the application image before every boot.

Definition at line 230 of file platform/bootloader/api/btl_interface.h

BOOTLOADER_CAPABILITY_BOOTLOADER_UPGRADE

```
#define BOOTLOADER_CAPABILITY_BOOTLOADER_UPGRADE
```

Value:

```
(1 << 4)
```

Bootloader has the capability of being upgraded.

Definition at line 233 of file platform/bootloader/api/btl_interface.h

BOOTLOADER_CAPABILITY_GBL

```
#define BOOTLOADER_CAPABILITY_GBL
```

Value:

```
(1 << 5)
```

Bootloader has the capability of parsing GBL files.

Definition at line 236 of file platform/bootloader/api/btl_interface.h

BOOTLOADER_CAPABILITY_GBL_SIGNATURE

```
#define BOOTLOADER_CAPABILITY_GBL_SIGNATURE
```

Value:

```
(1 << 6)
```

Bootloader has the capability of parsing signed GBL files.

Definition at line 238 of file platform/bootloader/api/btl_interface.h

BOOTLOADER_CAPABILITY_GBL_ENCRYPTION

```
#define BOOTLOADER_CAPABILITY_GBL_ENCRYPTION
```

Value:

```
(1 << 7)
```

Bootloader has the capability of parsing encrypted GBL files.

Definition at line 240 of file platform/bootloader/api/btl_interface.h

BOOTLOADER_CAPABILITY_ENFORCE_CERTIFICATE_SECURE_BOOT

```
#define BOOTLOADER_CAPABILITY_ENFORCE_CERTIFICATE_SECURE_BOOT
```

Value:

```
(1 << 8)
```

Bootloader enforces signature verification of the application image before every boot using certificate.

Definition at line 243 of file platform/bootloader/api/btl_interface.h

BOOTLOADER_CAPABILITY_ROLLBACK_PROTECTION

```
#define BOOTLOADER_CAPABILITY_ROLLBACK_PROTECTION
```

Value:

```
(1 << 9)
```

Bootloader has the capability of application rollback protection.

Definition at line 245 of file platform/bootloader/api/btl_interface.h

BOOTLOADER_CAPABILITY_PERIPHERAL_LIST

```
#define BOOTLOADER_CAPABILITY_PERIPHERAL_LIST
```

Value:

```
(1 << 10)
```

Bootloader has the capability to check the peripherals in use.

Definition at line 247 of file platform/bootloader/api/btl_interface.h

BOOTLOADER_CAPABILITY_STORAGE

```
#define BOOTLOADER_CAPABILITY_STORAGE
```

Value:

```
(1 << 16)
```

Bootloader has the capability of storing data in an internal or external storage medium.

Definition at line 251 of file platform/bootloader/api/btl_interface.h

BOOTLOADER_CAPABILITY_COMMUNICATION

```
#define BOOTLOADER_CAPABILITY_COMMUNICATION
```

Value:

```
(1 << 20)
```

Bootloader has the capability of communicating with host processors using a communication interface.

Definition at line 254 of file platform/bootloader/api/btl_interface.h

BOOTLOADER_MAGIC_FIRST_STAGE

```
#define BOOTLOADER_MAGIC_FIRST_STAGE
```

Value:

```
(0xB00710ADUL)
```

Magic word indicating first stage bootloader table.

Definition at line 260 of file platform/bootloader/api/btl_interface.h

BOOTLOADER_MAGIC_MAIN

```
#define BOOTLOADER_MAGIC_MAIN
```

Value:

```
(0x5ECDB007UL)
```

Magic word indicating main bootloader table.

Definition at line 262 of file platform/bootloader/api/btl_interface.h

mainBootloaderTable

```
#define mainBootloaderTable
```

Value:

```
0                    (*(MainBootloaderTable_t **) \
0                    (BTL_MAIN_BOOTLOADER_TABLE_BASE))
```

Pointer to main bootloader table.

Definition at line 473 of file platform/bootloader/api/btl_interface.h

BareBootTable_t

Bare boot table. Can be mapped on top of vector table to access contents.

Public Attributes

uint32_t *	stackTop Pointer to top of stack.
void(*	resetVector Pointer to reset vector.
uint32_t	reserved0 Reserved pointers to fault handlers.
uint32_t	reserved1 Reserved pointers to RESERVED fields.
void *	table Pointer to bootloader table.
uint32_t	reserved2 Reserved pointers to SVC and DebugMon interrupts.
void *	signature Pointer to application signature.

Public Attribute Documentation

stackTop

```
uint32_t* BareBootTable_t::stackTop
```

Pointer to top of stack.

Definition at line 67 of file platform/bootloader/api/btl_interface.h

resetVector

```
void(* BareBootTable_t::resetVector) (void)
```

Pointer to reset vector.

Definition at line 69 of file platform/bootloader/api/btl_interface.h

reserved0

```
uint32_t BareBootTable_t::reserved0[5]
```

Reserved pointers to fault handlers.

Definition at line 71 of file platform/bootloader/api/btl_interface.h

reserved1

```
uint32_t BareBootTable_t::reserved1[3]
```

Reserved pointers to RESERVED fields.

Definition at line 73 of file platform/bootloader/api/btl_interface.h

table

```
void* BareBootTable_t::table
```

Pointer to bootloader table.

Definition at line 75 of file platform/bootloader/api/btl_interface.h

reserved2

```
uint32_t BareBootTable_t::reserved2[2]
```

Reserved pointers to SVC and DebugMon interrupts.

Definition at line 77 of file platform/bootloader/api/btl_interface.h

signature

```
void* BareBootTable_t::signature
```

Pointer to application signature.

Definition at line 79 of file platform/bootloader/api/btl_interface.h

BootloaderInformation_t

Information about the current bootloader.

Public Attributes

uint16_t	version Version, either of the bootloader for standalone bootloaders or of the storage info struct for application bootloaders.
BootloaderType_t	type The type of bootloader.
uint32_t	version Version number of the bootloader.
uint32_t	capabilities Capability mask for the bootloader.

Public Attribute Documentation

version

> uint16_t BootloaderInformation_t::version

Version, either of the bootloader for standalone bootloaders or of the storage info struct for application bootloaders.

Definition at line 122 of file platform/bootloader/api/ember_btl_interface.h

type

> BootloaderType_t BootloaderInformation_t::type

The type of bootloader.

Definition at line 117 of file platform/bootloader/api/btl_interface.h

version

> uint32_t BootloaderInformation_t::version

Version number of the bootloader.

Definition at line 119 of file platform/bootloader/api/btl_interface.h

capabilities

uint32_t BootloaderInformation_t::capabilities

Capability mask for the bootloader.

Definition at line 121 of file platform/bootloader/api/btl_interface.h

BootloaderHeader_t

Common header for bootloader tables.

Public Attributes

uint32_t	type Type of image.
uint32_t	layout Version number of the bootloader/application table.
uint32_t	version Version number of the image.

Public Attribute Documentation

type

```
uint32_t BootloaderHeader_t::type
```

Type of image.

Definition at line 127 of file platform/bootloader/api/btl_interface.h

layout

```
uint32_t BootloaderHeader_t::layout
```

Version number of the bootloader/application table.

Definition at line 129 of file platform/bootloader/api/btl_interface.h

version

```
uint32_t BootloaderHeader_t::version
```

Version number of the image.

Definition at line 131 of file platform/bootloader/api/btl_interface.h

FirstBootloaderTable_t

Address table for the First Stage Bootloader.

Public Attributes

BootloaderHeader_t	header Header of the First Stage Bootloader table.
BareBootTable_t *	mainBootloader Start address of the Main Bootloader.
BareBootTable_t *	upgradeLocation Location of the Main Bootloader upgrade image.

Public Attribute Documentation

header

```
BootloaderHeader_t FirstBootloaderTable_t::header
```

Header of the First Stage Bootloader table.

Definition at line 137 of file platform/bootloader/api/btl_interface.h

mainBootloader

```
BareBootTable_t* FirstBootloaderTable_t::mainBootloader
```

Start address of the Main Bootloader.

Definition at line 139 of file platform/bootloader/api/btl_interface.h

upgradeLocation

```
BareBootTable_t* FirstBootloaderTable_t::upgradeLocation
```

Location of the Main Bootloader upgrade image.

Definition at line 141 of file platform/bootloader/api/btl_interface.h

MainBootloaderTable_t

Address table for the Main Bootloader.

Public Attributes

BootloaderHeader_t	header Header of the Main Bootloader table.
uint32_t	size Size of the Main Bootloader.
BareBootTable_t *	startOfAppSpace Start address of the application.
uint32_t *	endOfAppSpace End address of the allocated application space.
uint32_t	capabilities Capabilities of the bootloader.
int32_t(*	init Initialize bootloader for use from application.
int32_t(*	deinit Deinitialize bootloader after use from application.
bool(*	verifyApplication Verify application.
int32_t(*	initParser Initialize parser.
int32_t(*	parseBuffer Parse a buffer.
const BootloaderStorageFunctions_t *	storage Function table for storage component.
int32_t(*	parseImageInfo Parse a buffer and get application and bootloader upgrade metadata from the buffer.
uint32_t(*	parserContextSize Size of context buffer used by bootloader image parser to store parser state.
uint32_t(*	remainingApplicationUpgrades Remaining number of application upgrades.
void(*	getPeripheralList Get the list of the peripheral that is used by the bootloader.
uint32_t(*	getUpgradeLocation Get base address of bootloader upgrade image.

Public Attribute Documentation

header

```
BootloaderHeader_t MainBootloaderTable_t::header
```

Header of the Main Bootloader table.

Definition at line 147 of file platform/bootloader/api/btl_interface.h

size

```
uint32_t MainBootloaderTable_t::size
```

Size of the Main Bootloader.

Definition at line 149 of file platform/bootloader/api/btl_interface.h

startOfAppSpace

```
BareBootTable_t* MainBootloaderTable_t::startOfAppSpace
```

Start address of the application.

Definition at line 151 of file platform/bootloader/api/btl_interface.h

endOfAppSpace

```
uint32_t* MainBootloaderTable_t::endOfAppSpace
```

End address of the allocated application space.

Definition at line 153 of file platform/bootloader/api/btl_interface.h

capabilities

```
uint32_t MainBootloaderTable_t::capabilities
```

Capabilities of the bootloader.

Definition at line 155 of file platform/bootloader/api/btl_interface.h

init

```
int32_t(* MainBootloaderTable_t::init) (void)
```

Initialize bootloader for use from application.

Definition at line 158 of file platform/bootloader/api/btl_interface.h

deinit

```
int32_t(* MainBootloaderTable_t::deinit) (void)
```

Deinitialize bootloader after use from application.

Definition at line 160 of file platform/bootloader/api/btl_interface.h

verifyApplication

```
bool(* MainBootloaderTable_t::verifyApplication) (uint32_t startAddress)
```

Verify application.

Definition at line 163 of file platform/bootloader/api/btl_interface.h

initParser

```
int32_t(* MainBootloaderTable_t::initParser) (BootloaderParserContext_t *context, size_t contextSize)
```

Initialize parser.

Definition at line 166 of file platform/bootloader/api/btl_interface.h

parseBuffer

```
int32_t(* MainBootloaderTable_t::parseBuffer) (BootloaderParserContext_t *context, const BootloaderParserCallbacks_t *callbacks, uint8_t data[], size_t numBytes)
```

Parse a buffer.

Definition at line 168 of file platform/bootloader/api/btl_interface.h

storage

```
const BootloaderStorageFunctions_t* MainBootloaderTable_t::storage
```

Function table for storage component.

Definition at line 174 of file platform/bootloader/api/btl_interface.h

parseImageInfo

```
int32_t(* MainBootloaderTable_t::parseImageInfo) (BootloaderParserContext_t *context, uint8_t data[], size_t numBytes, ApplicationData_t *appInfo, uint32_t *bootloaderVersion)
```

Parse a buffer and get application and bootloader upgrade metadata from the buffer.

Definition at line 177 of file platform/bootloader/api/btl_interface.h

parserContextSize

```
uint32_t(* MainBootloaderTable_t::parserContextSize) (void)
```

Size of context buffer used by bootloader image parser to store parser state.

Definition at line 184 of file platform/bootloader/api/btl_interface.h

remainingApplicationUpgrades

```
uint32_t(* MainBootloaderTable_t::remainingApplicationUpgrades) (void)
```

Remaining number of application upgrades.

Definition at line 187 of file platform/bootloader/api/btl_interface.h

getPeripheralList

```
void(* MainBootloaderTable_t::getPeripheralList) (uint32_t *ppusatd0, uint32_t *ppusatd1)
```

Get the list of the peripheral that is used by the bootloader.

Definition at line 190 of file platform/bootloader/api/btl_interface.h

getUpgradeLocation

```
uint32_t(* MainBootloaderTable_t::getUpgradeLocation) (void)
```

Get base address of bootloader upgrade image.

Definition at line 193 of file platform/bootloader/api/btl_interface.h

Reset Information

Reset Information

Passing information when resetting into and out of the bootloader.

To signal the bootloader to run, the application needs to write the BootloaderResetCause_t structure to the first address of RAM, setting BootloaderResetCause_t::reason to BOOTLOADER_RESET_REASON_BOOTLOAD.

The reset cause is only valid if BootloaderResetCause_t::signature is set to BOOTLOADER_RESET_SIGNATURE_VALID.

```
BootloaderResetCause_t resetCause = {
  .reason = BOOTLOADER_RESET_REASON_BOOTLOAD,
  .signature = BOOTLOADER_RESET_SIGNATURE_VALID
}
```

When the bootloader reboots back into the app, it sets the reset reason to BOOTLOADER_RESET_REASON_GO if the bootload succeeded, or BOOTLOADER_RESET_REASON_BADIMAGE if the bootload failed due to errors when parsing the upgrade image.

Note

- The reset information is automatically filled out before reset if the bootloader_rebootAndInstall function is called.

Modules

BootloaderResetCause_t

Macros

#define BOOTLOADER_RESET_REASON_UNKNOWN 0x0200u
Unknown bootloader cause (should never occur)

#define BOOTLOADER_RESET_REASON_GO 0x0201u
Bootloader caused reset telling app to run.

#define BOOTLOADER_RESET_REASON_BOOTLOAD 0x0202u
Application requested that bootloader runs.

#define BOOTLOADER_RESET_REASON_BADIMAGE 0x0203u
Bootloader detected bad external upgrade image.

#define BOOTLOADER_RESET_REASON_FATAL 0x0204u
Fatal Error or assert in bootloader.

#define BOOTLOADER_RESET_REASON_FORCE 0x0205u
Forced bootloader activation.

#define BOOTLOADER_RESET_REASON_OTAVALID 0x0206u
OTA Bootloader mode activation.

#define BOOTLOADER_RESET_REASON_DEEPSLEEP 0x0207u
Bootloader initiated deep sleep.

#define BOOTLOADER_RESET_REASON_BADAPP 0x0208u
Application verification failed.

#define BOOTLOADER_RESET_REASON_UPGRADE 0x0209u
Bootloader requested that first stage upgrades main bootloader.

#define BOOTLOADER_RESET_REASON_TIMEOUT 0x020Au
Bootloader timed out waiting for upgrade image.

#define BOOTLOADER_RESET_REASON_FAULT 0x020Bu
Soft-reset was forced to handle a fault.

#define BOOTLOADER_RESET_REASON_TZ_FAULT 0x020Cu
Soft-reset was forced to handle a security fault.

#define BOOTLOADER_RESET_SIGNATURE_VALID 0xF00Fu
Reset signature is valid.

#define BOOTLOADER_RESET_SIGNATURE_INVALID 0xC33Cu
Reset signature is invalid.

Macro Definition Documentation

BOOTLOADER_RESET_REASON_UNKNOWN

```
#define BOOTLOADER_RESET_REASON_UNKNOWN
```

Value:

```
0x0200u
```

Unknown bootloader cause (should never occur)

Definition at line 62 of file platform/bootloader/api/btl_reset_info.h

BOOTLOADER_RESET_REASON_GO

```
#define BOOTLOADER_RESET_REASON_GO
```

Value:

```
0x0201u
```

Bootloader caused reset telling app to run.

Definition at line 64 of file platform/bootloader/api/btl_reset_info.h

BOOTLOADER_RESET_REASON_BOOTLOAD

```
#define BOOTLOADER_RESET_REASON_BOOTLOAD
```

Value:

```
0x0202u
```

Application requested that bootloader runs.

Definition at line 66 of file platform/bootloader/api/btl_reset_info.h

BOOTLOADER_RESET_REASON_BADIMAGE

```
#define BOOTLOADER_RESET_REASON_BADIMAGE
```

Value:

```
0x0203u
```

Bootloader detected bad external upgrade image.

Definition at line 68 of file platform/bootloader/api/btl_reset_info.h

BOOTLOADER_RESET_REASON_FATAL

```
#define BOOTLOADER_RESET_REASON_FATAL
```

Value:

```
0x0204u
```

Fatal Error or assert in bootloader.

Definition at line 70 of file platform/bootloader/api/btl_reset_info.h

BOOTLOADER_RESET_REASON_FORCE

```
#define BOOTLOADER_RESET_REASON_FORCE
```

Value:

```
0x0205u
```

Forced bootloader activation.

Definition at line 72 of file platform/bootloader/api/btl_reset_info.h

BOOTLOADER_RESET_REASON_OTAVALID

```
#define BOOTLOADER_RESET_REASON_OTAVALID
```

Value:

```
0x0206u
```

OTA Bootloader mode activation.

Definition at line 74 of file platform/bootloader/api/btl_reset_info.h

BOOTLOADER_RESET_REASON_DEEPSLEEP

```
#define BOOTLOADER_RESET_REASON_DEEPSLEEP
```

Value:

```
0x0207u
```

Bootloader initiated deep sleep.

Definition at line `76` of file `platform/bootloader/api/btl_reset_info.h`

BOOTLOADER_RESET_REASON_BADAPP

```
#define BOOTLOADER_RESET_REASON_BADAPP
```

Value:

```
0x0208u
```

Application verification failed.

Definition at line `78` of file `platform/bootloader/api/btl_reset_info.h`

BOOTLOADER_RESET_REASON_UPGRADE

```
#define BOOTLOADER_RESET_REASON_UPGRADE
```

Value:

```
0x0209u
```

Bootloader requested that first stage upgrades main bootloader.

Definition at line `80` of file `platform/bootloader/api/btl_reset_info.h`

BOOTLOADER_RESET_REASON_TIMEOUT

```
#define BOOTLOADER_RESET_REASON_TIMEOUT
```

Value:

```
0x020Au
```

Bootloader timed out waiting for upgrade image.

Definition at line `82` of file `platform/bootloader/api/btl_reset_info.h`

BOOTLOADER_RESET_REASON_FAULT

```
#define BOOTLOADER_RESET_REASON_FAULT
```

Value:

```
0x020Bu
```

Soft-reset was forced to handle a fault.

Definition at line 84 of file platform/bootloader/api/btl_reset_info.h

BOOTLOADER_RESET_REASON_TZ_FAULT

```
#define BOOTLOADER_RESET_REASON_TZ_FAULT
```

Value:

```
0x020Cu
```

Soft-reset was forced to handle a security fault.

Definition at line 86 of file platform/bootloader/api/btl_reset_info.h

BOOTLOADER_RESET_SIGNATURE_VALID

```
#define BOOTLOADER_RESET_SIGNATURE_VALID
```

Value:

```
0xF00Fu
```

Reset signature is valid.

Definition at line 89 of file platform/bootloader/api/btl_reset_info.h

BOOTLOADER_RESET_SIGNATURE_INVALID

```
#define BOOTLOADER_RESET_SIGNATURE_INVALID
```

Value:

```
0xC33Cu
```

Reset signature is invalid.

Definition at line 91 of file platform/bootloader/api/btl_reset_info.h

BootloaderResetCause_t

Reset cause of the bootloader.

Public Attributes

uint16_t	reason Reset reason as defined in the reset information.
uint16_t	signature Signature indicating whether the reset reason is valid.

Public Attribute Documentation

reason

```
uint16_t BootloaderResetCause_t::reason
```

Reset reason as defined in the reset information.

Definition at line 158 of file platform/bootloader/api/ember_btl_interface.h

signature

```
uint16_t BootloaderResetCause_t::signature
```

Signature indicating whether the reset reason is valid.

Definition at line 159 of file platform/bootloader/api/ember_btl_interface.h

Bootloader Core

Bootloader Core

Core bootloader functionality.

Core functionality for the Silicon Labs Bootloader.

Modules

Bootload

Flash

Reset

TrustZone

Upgrade

Functions

int32_t btl_init(void)
Initialize bootloader.

int32_t btl_deinit(void)
Deinitialize bootloader.

Function Documentation

btl_init

```
int32_t btl_init (void)
```

Initialize bootloader.

Parameters

N/A		

Returns

- Error code. BOOTLOADER_OK on success, else error code in BOOTLOADER_ERROR_INIT_BASE range.

Definition at line `36` of file `platform/bootloader/core/btl_core.h`

btl_deinit

```
int32_t btl_deinit (void)
```

Deinitialize bootloader.

Parameters

N/A		

Returns

- Error code. BOOTLOADER_OK on success, else error code in BOOTLOADER_ERROR_INIT_BASE range.

Definition at line 44 of file platform/bootloader/core/btl_core.h

Bootload

Bootload

Methods to verify and bootload application images.

Functions

uint32_t	bootload_getUpgradeLocation(void) Get base address of the bootloader upgrade image.
uint32_t	bootload_getBootloaderVersion(void) Get the version of the bootloader.
bool	bootload_getApplicationVersion(uint32_t *version) Get the version of the application.
bool	bootload_checkApplicationPropertiesMagic(void *appProperties) Check application properties magic.
bool	bootload_checkApplicationPropertiesVersion(void *appProperties) Check application properties struct version.
bool	bootload_verifyApplication(uint32_t startAddress) Verify the application image stored in the Flash memory starting at the address startAddress.
void	bootload_bootloaderCallback(uint32_t offset, uint8_t data[], size_t length, void *context) Bootloader upgrade callback implementation.
void	bootload_applicationCallback(uint32_t address, uint8_t data[], size_t length, void *context) Image data callback implementation.
bool	bootload_commitBootloaderUpgrade(uint32_t upgradeAddress, uint32_t size) Perform a bootloader upgrade using the upgrade image present at upgradeAddress with length size.
bool	bootload_verifyApplicationVersion(uint32_t appVersion, bool checkRemainingAppUpgrades) Verify the application version for rollback protection.
bool	bootload_storeApplicationVersion(uint32_t startAddress) Store the application version.
uint32_t	bootload_remainingApplicationUpgrades(void) Count the total remaining number of application upgrades.
void	bootload_storeApplicationVersionResetMagic(void) Store application version reset magic.
void	bootload_removeStoredApplicationVersions(void) Clean the application versions seen.
uint32_t	bootload_getApplicationVersionStorageCapacity(void) Get the application version storage capacity.
uint32_t *	bootload_getApplicationVersionStoragePtr(uint32_t index) Get the address of the application version storage buffer.

bool bootload_gotCertificate(void *appProp)
Check whether the application contains a certificate.

bool bootload_verifyCertificate(void *cert)
Verify a certificate with a bootloader certificate.

bool bootload_verifyApplicationCertificate(void *appProp, void *gotCert)
Verify the application certificate.

Function Documentation

bootload_getUpgradeLocation

```
uint32_t bootload_getUpgradeLocation (void)
```

Get base address of the bootloader upgrade image.

Parameters

N/A		

Returns

- Returns the base address of bootloader upgrade image.

Definition at line 40 of file platform/bootloader/core/btl_bootload.h

bootload_getBootloaderVersion

```
uint32_t bootload_getBootloaderVersion (void)
```

Get the version of the bootloader.

Parameters

N/A		

Returns

- Returns the version of the bootloader.

Definition at line 47 of file platform/bootloader/core/btl_bootload.h

bootload_getApplicationVersion

```
bool bootload_getApplicationVersion (uint32_t *version)
```

Get the version of the application.

Parameters

[out]	version	The retrieved application version

Returns

- Returns true if the version was retrieved successfully

Definition at line 56 of file platform/bootloader/core/btl_bootload.h

bootload_checkApplicationPropertiesMagic

```
bool bootload_checkApplicationPropertiesMagic (void *appProperties)
```

Check application properties magic.

Parameters

N/A	appProperties	Pointer to ApplicationProperties_t

Returns

- True if the application properties magic is valid.

Definition at line `76` of file `platform/bootloader/core/btl_bootload.h`

bootload_checkApplicationPropertiesVersion

```
bool bootload_checkApplicationPropertiesVersion (void *appProperties)
```

Check application properties struct version.

Parameters

N/A	appProperties	Pointer to ApplicationProperties_t

Returns

- True if the application properties struct version is compatible with the bootloader.

Definition at line `86` of file `platform/bootloader/core/btl_bootload.h`

bootload_verifyApplication

```
bool bootload_verifyApplication (uint32_t startAddress)
```

Verify the application image stored in the Flash memory starting at the address startAddress.

Parameters

[in]	startAddress	Starting address of the application

If secure boot is enforced, the function will only return true if the cryptographic signature of the application is valid. Else, the application is verified according to the signature type defined in the ApplicationProperties_t structure embedded in the application. Silicon Labs wireless stacks declare this structure. Applications, which are not using a full wireless stack may need to instantiate the structure.

Examples of results when secure boot is not enforced:

- App has no signature: Valid if initial stack pointer and program counter have reasonable values
- App has CRC checksum: Valid if checksum is valid
- App has ECDSA signature: Valid if ECDSA signature is valid.

When secure boot is enforced, only ECDSA-signed applications with a valid signature are considered valid.

Returns

- True if the image is deemed valid

Definition at line 112 of file platform/bootloader/core/btl_bootload.h

bootload_bootloaderCallback

```
void bootload_bootloaderCallback (uint32_t offset, uint8_t data[], size_t length, void *context)
```

Bootloader upgrade callback implementation.

Parameters

N/A	offset	Offset of bootloader data (byte counter incrementing from 0)
N/A	data	Raw bootloader data
N/A	length	Size in bytes of raw bootloader data.
N/A	context	A context variable defined by the implementation that is implementing this callback.

Definition at line 124 of file platform/bootloader/core/btl_bootload.h

bootload_applicationCallback

```
void bootload_applicationCallback (uint32_t address, uint8_t data[], size_t length, void *context)
```

Image data callback implementation.

Parameters

N/A	address	Address (inside the raw image) the data starts at
N/A	data	Raw image data
N/A	length	Size in bytes of raw image data. Always constrained to a multiple of four.
N/A	context	A context variable defined by the implementation that is implementing this callback.

Definition at line 139 of file platform/bootloader/core/btl_bootload.h

bootload_commitBootloaderUpgrade

```
bool bootload_commitBootloaderUpgrade (uint32_t upgradeAddress, uint32_t size)
```

Perform a bootloader upgrade using the upgrade image present at upgradeAddress with length size.

Parameters

[in]	upgradeAddress	The starting address of the upgrade image
[in]	size	The length of the upgrade image in bytes

If the bootloader upgrade process starts successfully, this function does not return and execution will resume from the reset handler of the upgraded bootloader.

Returns

- False if the bootloader upgrade process didn't start

Definition at line 157 of file platform/bootloader/core/btl_bootload.h

bootload_verifyApplicationVersion

```
bool bootload_verifyApplicationVersion (uint32_t appVersion, bool checkRemainingAppUpgrades)
```

Verify the application version for rollback protection.

Parameters

[in]	appVersion	Application version to be checked.
[in]	checkRemainingAppUpgrades	Check remaining application upgrades.

Returns

- True if the application version is higher or equal than the application versions seen. False if the application version is lower than the application versions seen. False if no remaining application upgrades are left when `checkRemainingAppUpgrades` is true.

Definition at line `172` of file `platform/bootloader/core/btl_bootload.h`

bootload_storeApplicationVersion

```
bool bootload_storeApplicationVersion (uint32_t startAddress)
```

Store the application version.

Parameters

N/A	startAddress	Start address of application.

Note

- Only the version of the verified application should be stored.

Returns

- True if application version is successfully stored.

Definition at line `185` of file `platform/bootloader/core/btl_bootload.h`

bootload_remainingApplicationUpgrades

```
uint32_t bootload_remainingApplicationUpgrades (void)
```

Count the total remaining number of application upgrades.

Parameters

N/A		

Returns

- remaining number of application upgrades.

Definition at line `192` of file `platform/bootloader/core/btl_bootload.h`

bootload_storeApplicationVersionResetMagic

```
void bootload_storeApplicationVersionResetMagic (void)
```

Store application version reset magic.

Parameters

N/A		

Note

- Store application version reset magic to ensure that application versions are cleaned after a bootloader upgrade.

Definition at line `200` of file `platform/bootloader/core/btl_bootload.h`

bootload_removeStoredApplicationVersions

```
void bootload_removeStoredApplicationVersions (void)
```

Clean the application versions seen.

Parameters

N/A		

Note

- The application versions are cleaned only if this is requested with a magic and the application version storage is not already empty.

Definition at line `208` of file `platform/bootloader/core/btl_bootload.h`

bootload_getApplicationVersionStorageCapacity

```
uint32_t bootload_getApplicationVersionStorageCapacity (void)
```

Get the application version storage capacity.

Parameters

N/A		

Returns

- Application version storage capacity.

Definition at line `215` of file `platform/bootloader/core/btl_bootload.h`

bootload_getApplicationVersionStoragePtr

```
uint32_t * bootload_getApplicationVersionStoragePtr (uint32_t index)
```

Get the address of the application version storage buffer.

Parameters

N/A	index	Index of the application version storage buffer.

Returns

- Address of the application version storage buffer with the given index.

Definition at line `225` of file `platform/bootloader/core/btl_bootload.h`

bootload_gotCertificate

```
bool bootload_gotCertificate (void *appProp)
```

Check whether the application contains a certificate.

Parameters

N/A	appProp	Pointer to ApplicationProperties_t of application.

Returns

- True if application contains a certificate.

Definition at line `234` of file `platform/bootloader/core/btl_bootload.h`

bootload_verifyCertificate

```
bool bootload_verifyCertificate (void *cert)
```

Verify a certificate with a bootloader certificate.

Parameters

N/A	cert	Pointer to ApplicationCertificate_t.

Returns

- True if certificate is verified.

Definition at line `243` of file `platform/bootloader/core/btl_bootload.h`

bootload_verifyApplicationCertificate

```
bool bootload_verifyApplicationCertificate (void *appProp, void *gotCert)
```

Verify the application certificate.

Parameters

N/A	appProp	Pointer to ApplicationProperties_t of application.
N/A	gotCert	Boolean to store application certificate presence.

Note

- This function will always return true if certificate support is not enabled. Also true if `appProp` does not contain any certificate and direct signed applications can be accepted.

Returns

- True if application certificate is verified.

Definition at line `258` of file `platform/bootloader/core/btl_bootload.h`

Flash

Flash

Interface to internal flash.

Used for writing application images to the main flash.

Functions

bool flash_erasePage(uint32_t address)
Erase a flash page.

bool flash_writeBuffer_dma(uint32_t address, void *data, size_t length, int ch)
Write buffer to internal flash.

bool flash_writeBuffer(uint32_t address, void *data, size_t length)
Write buffer to internal flash.

Macros

#define SL_GBL_MSC_LDMA_CHANNEL 2
DMA Channel for MSC write.

Function Documentation

flash_erasePage

```
bool flash_erasePage (uint32_t address)
```

Erase a flash page.

Parameters

[in]	address	Start address of the flash page to erase.

Returns

- True if operation was successful

Definition at line 53 of file platform/bootloader/core/flash/btl_internal_flash.h

flash_writeBuffer_dma

```
bool flash_writeBuffer_dma (uint32_t address, void *data, size_t length, int ch)
```

Write buffer to internal flash.

Parameters

N/A	address	Starting address to write data to. Must be half-word aligned.

N/A	data	Data buffer to write to internal flash
N/A	length	Amount of bytes in the data buffer to write
N/A	ch	DMA channel to use

Returns

- True if operation was successful

Definition at line 64 of file platform/bootloader/core/flash/btl_internal_flash.h

flash_writeBuffer

```
bool flash_writeBuffer (uint32_t address, void *data, size_t length)
```

Write buffer to internal flash.

Parameters

N/A	address	Starting address to write data to. Must be half-word aligned.
N/A	data	Data buffer to write to internal flash
N/A	length	Amount of bytes in the data buffer to write

Returns

- True if operation was successful

Definition at line 77 of file platform/bootloader/core/flash/btl_internal_flash.h

Macro Definition Documentation

SL_GBL_MSC_LDMA_CHANNEL

```
#define SL_GBL_MSC_LDMA_CHANNEL
```

Value:

```
2
```

DMA Channel for MSC write.

Definition at line 42 of file platform/bootloader/core/flash/btl_internal_flash.h

Reset

Reset

Methods to reset from the bootloader to the app.

Functions

void	reset_resetWithReason(uint16_t resetReason) Reset from the bootloader with a reset cause.
void	reset_setResetReason(uint16_t resetReason) Set a reset reason.
void	reset_enableResetCounter(void) Use the lower 4 bits of the reset cause signature to store a reset counter able to count up to 15 before wrapping around.
void	reset_disableResetCounter(void) Clear and disable the reset counter.
bool	reset_resetCounterEnabled(void) Check whether the reset counter is enabled.
void	reset_incrementResetCounter(void) Increment the reset counter by one.
uint8_t	reset_getResetCounter(void) Get the reset counter value.
uint16_t	reset_getResetReason(void) Get the reset reason without verifying it.
void	reset_invalidateResetReason(void) Invalidate the reset reason.
uint16_t	reset_classifyReset(void) Classify reset and get the reset reason.

Function Documentation

reset_resetWithReason

> void reset_resetWithReason (uint16_t resetReason)

Reset from the bootloader with a reset cause.

Parameters

N/A	resetReason	A reset reason as defined in the bootloader interface

Note

- This function does not return.

Definition at line 43 of file platform/bootloader/core/btl_reset.h

reset_setResetReason

```
void reset_setResetReason (uint16_t resetReason)
```

Set a reset reason.

Parameters

N/A	resetReason	A reset reason as defined in the bootloader interface

Definition at line 51 of file platform/bootloader/core/btl_reset.h

reset_enableResetCounter

```
void reset_enableResetCounter (void)
```

Use the lower 4 bits of the reset cause signature to store a reset counter able to count up to 15 before wrapping around.

Parameters

N/A		

The value of the counter will be preserved across resets.

Definition at line 58 of file platform/bootloader/core/btl_reset.h

reset_disableResetCounter

```
void reset_disableResetCounter (void)
```

Clear and disable the reset counter.

Parameters

N/A		

Definition at line 63 of file platform/bootloader/core/btl_reset.h

reset_resetCounterEnabled

```
bool reset_resetCounterEnabled (void)
```

Check whether the reset counter is enabled.

Parameters

N/A		

Returns

- True if reset counter is enabled, else false.

Definition at line 70 of file platform/bootloader/core/btl_reset.h

reset_incrementResetCounter

```
void reset_incrementResetCounter (void)
```

Increment the reset counter by one.

Parameters

N/A		

Wraps around to 0 if the current counter value is equal to 15.

Definition at line `77` of file `platform/bootloader/core/btl_reset.h`

reset_getResetCounter

```
uint8_t reset_getResetCounter (void)
```

Get the reset counter value.

Parameters

N/A		

Note

- The reset counter has to be enabled for this value to be valid.

Returns

- The reset counter value

Definition at line `86` of file `platform/bootloader/core/btl_reset.h`

reset_getResetReason

```
uint16_t reset_getResetReason (void)
```

Get the reset reason without verifying it.

Parameters

N/A		

Returns

- The reset reason

Definition at line `93` of file `platform/bootloader/core/btl_reset.h`

reset_invalidateResetReason

```
void reset_invalidateResetReason (void)
```

Invalidate the reset reason.

Parameters

N/A		

Note

- This will also disable the reset counter if it is in use.

Definition at line `100` of file `platform/bootloader/core/btl_reset.h`

reset_classifyReset

```
uint16_t reset_classifyReset (void)
```

Classify reset and get the reset reason.

Parameters

N/A		

Returns

- Reset cause or BOOTLOADER_RESET_REASON_UNKNOWN

Definition at line `107` of file `platform/bootloader/core/btl_reset.h`

TrustZone

TrustZone

TrustZone utilities.

Functions

void	bl_fatal_assert_action(void) TrustZone fatal error handler.
bool	bl_verify_ns_memory_access(const void *p, size_t s) Validate the non-secure pointers.

Function Documentation

bl_fatal_assert_action

```
void bl_fatal_assert_action (void)
```

TrustZone fatal error handler.

Parameters

N/A		

This function triggers a soft-reset with the reset reason set to BOOTLOADER_RESET_REASON_TZ_FAULT.

Definition at line `42` of file `platform/bootloader/core/btl_tz_utils.h`

bl_verify_ns_memory_access

```
bool bl_verify_ns_memory_access (const void *p, size_t s)
```

Validate the non-secure pointers.

Parameters

N/A	p	Pointer to the memory.
N/A	s	Size of the memory to be validated.

Validate if the address of the memory is actually non-secure as expected.

Returns

- True if the memory being accessed is from NS

Definition at line `54` of file `platform/bootloader/core/btl_tz_utils.h`

Upgrade

Upgrade

Methods to verify and upgrade the main bootloader.

Functions

bool btl_checkForUpgrade(void)
Check whether a bootloader upgrade is available.

bool btl_applyUpgrade(void)
Apply a bootloader upgrade.

Function Documentation

btl_checkForUpgrade

```
bool btl_checkForUpgrade (void)
```

Check whether a bootloader upgrade is available.

Parameters

N/A		

Returns

- True if an upgrade image is in the upgrade location

Definition at line 36 of file platform/bootloader/core/btl_upgrade.h

btl_applyUpgrade

```
bool btl_applyUpgrade (void)
```

Apply a bootloader upgrade.

Parameters

N/A		

Returns

- True if the bootloader upgrade was applied successfully

Definition at line 43 of file platform/bootloader/core/btl_upgrade.h

Components

Components

Modules

- Communication
- Debug
- Decompressor
- GPIO Activation
- Image Parser
- Security
- Storage

Communication

Communication

Host communication interface.

The Communication component provides an interface for implementing communication with a host device, such as a computer or a microcontroller.

Communication Protocol Implementations

Several components implement the communication interface, using different transports and protocols.

Modules

UART XMODEM

Bluetooth Apploader OTA DFU

BGAPI UART DFU

EZSP-SPI

Utils

Functions

void communication_init(void)
Initialize hardware for the communication component.

int32_t communication_start(void)
Initialize communication between the bootloader and external host.

int32_t communication_main(void)
Not supposed to return until either the host signals the end of the current session or a new image has been flashed and verified.

void communication_shutdown(void)
Stop communication between the bootloader and external host.

void bootloader_xmodem_communication_init(void)
Initialize hardware for the UART XMODEM Bootloader communication.

int32_t bootloader_xmodem_communication_start(void)
Initialize communication between the UART XMODEM bootloader and external host.

int32_t bootloader_xmodem_communication_main(ImageProperties_t *imageProps, const BootloaderParserCallbacks_t *parseCb)
Communication main for the UART XMODEM bootloader.

void bootloader_apploader_communication_init(void)
Initialize hardware for the BLE Apploader OTA DFU Bootloader communication.

int32_t bootloader_apploader_communication_start(void)
Initialize communication between the BLE Apploader OTA DFU bootloader and external host.

int32_t bootloader_apploader_communication_main(ImageProperties_t *imageProps, void *parserContext, void *decryptContext, void *authContext, const BootloaderParserCallbacks_t *parseCb)
Communication main for the BLE Apploader OTA DFU bootloader.

void bootloader_apploader_get_custom_device_address(sl_apploader_address_t *btAddress)
Get custom device address.

void bootloader_bgapi_communication_init(void)
Initialize hardware for the BGAPI UART DFU Bootloader communication.

int32_t bootloader_bgapi_communication_start(void)
Initialize communication between the BGAPI UART DFU bootloader and external host.

int32_t bootloader_bgapi_communication_main(ImageProperties_t *imageProps, ParserContext_t *parserContext, const BootloaderParserCallbacks_t *parseCb)
Communication main for the BGAPI UART DFU bootloader.

void bootloader_ezsp_communication_init(void)
Initialize hardware for the EZSP SPI Bootloader communication.

int32_t bootloader_ezsp_communication_start(void)
Initialize communication between the EZSP SPI bootloader and external host.

int32_t bootloader_ezsp_communication_main(ImageProperties_t *imageProps, ParserContext_t *parserContext, const BootloaderParserCallbacks_t *parseCb)
Communication main for the EZSP SPI bootloader.

void bootloader_ezsp_communication_shutdown(void)
Stop communication between the bootloader and external host.

Function Documentation

communication_init

```
void communication_init (void)
```

Initialize hardware for the communication component.

Parameters

N/A		

Definition at line `41` of file `platform/bootloader/communication/btl_communication.h`

communication_start

```
int32_t communication_start (void)
```

Initialize communication between the bootloader and external host.

Parameters

N/A		

For example, indicate that all is well to the external host.

Returns

- Error code indicating success or failure

Definition at line `51` of file `platform/bootloader/communication/btl_communication.h`

communication_main

```
int32_t communication_main (void)
```

Not supposed to return until either the host signals the end of the current session or a new image has been flashed and verified.

Parameters

N/A		

Returns

- Error code indicating success or failure

Definition at line `64` of file `platform/bootloader/communication/btl_communication.h`

communication_shutdown

```
void communication_shutdown (void)
```

Stop communication between the bootloader and external host.

Parameters

N/A		

Definition at line `69` of file `platform/bootloader/communication/btl_communication.h`

bootloader_xmodem_communication_init

```
void bootloader_xmodem_communication_init (void)
```

Initialize hardware for the UART XMODEM Bootloader communication.

Parameters

N/A		

Definition at line `72` of file `platform/bootloader/communication/xmodem-uart/btl_comm_xmodem.h`

bootloader_xmodem_communication_start

```
int32_t bootloader_xmodem_communication_start (void)
```

Initialize communication between the UART XMODEM bootloader and external host.

Parameters

N/A		

Returns

- Error code indicating success or failure.

Definition at line `80` of file `platform/bootloader/communication/xmodem-uart/btl_comm_xmodem.h`

bootloader_xmodem_communication_main

```
int32_t bootloader_xmodem_communication_main (ImageProperties_t *imageProps, const BootloaderParserCallbacks_t *parseCb)
```

Communication main for the UART XMODEM bootloader.

Parameters

N/A	imageProps	The image file processed
N/A	parseCb	Bootloader parser callbacks

Returns

- Error code indicating success or failure.

Definition at line `90` of file `platform/bootloader/communication/xmodem-uart/btl_comm_xmodem.h`

bootloader_apploader_communication_init

```
void bootloader_apploader_communication_init (void)
```

Initialize hardware for the BLE Apploader OTA DFU Bootloader communication.

Parameters

N/A		

Definition at line `50` of file `platform/bootloader/communication/apploader/btl_apploader.h`

bootloader_apploader_communication_start

```
int32_t bootloader_apploader_communication_start (void)
```

Initialize communication between the BLE Apploader OTA DFU bootloader and external host.

Parameters

N/A		

Returns

- Error code indicating success or failure.

Definition at line `58` of file `platform/bootloader/communication/apploader/btl_apploader.h`

bootloader_apploader_communication_main

```
int32_t bootloader_apploader_communication_main (ImageProperties_t *imageProps, void *parserContext, void *decryptContext, void *authContext, const BootloaderParserCallbacks_t *parseCb)
```

Communication main for the BLE Apploader OTA DFU bootloader.

Parameters

N/A	imageProps	The image file processed

N/A	parserContext	Image parser context
N/A	decryptContext	Image decryption context
N/A	authContext	Image authentication context
N/A	parseCb	Bootloader parser callbacks

Returns

- Error code indicating success or failure.

Definition at line 71 of file platform/bootloader/communication/apploader/btl_apploader.h

bootloader_apploader_get_custom_device_address

```
void bootloader_apploader_get_custom_device_address (sl_apploader_address_t *btAddress)
```

Get custom device address.

Parameters

N/A	btAddress	Device address

The address must be set in little endian format Default implementation of this function reads the address from MFG_CUSTOM_EUI_64 manufacturing token stored in user data page. The function can be overridden if custom implementation is wanted.

Definition at line 85 of file platform/bootloader/communication/apploader/btl_apploader.h

bootloader_bgapi_communication_init

```
void bootloader_bgapi_communication_init (void)
```

Initialize hardware for the BGAPI UART DFU Bootloader communication.

Parameters

N/A		

Definition at line 159 of file platform/bootloader/communication/bgapi-uart-dfu/btl_comm_bgapi.h

bootloader_bgapi_communication_start

```
int32_t bootloader_bgapi_communication_start (void)
```

Initialize communication between the BGAPI UART DFU bootloader and external host.

Parameters

N/A		

Returns

- Error code indicating success or failure.

Definition at line 167 of file platform/bootloader/communication/bgapi-uart-dfu/btl_comm_bgapi.h

bootloader_bgapi_communication_main

int32_t bootloader_bgapi_communication_main (ImageProperties_t *imageProps, ParserContext_t *parserContext, const BootloaderParserCallbacks_t *parseCb)

Communication main for the BGAPI UART DFU bootloader.

Parameters

N/A	imageProps	The image file processed
N/A	parserContext	Image parser context
N/A	parseCb	Bootloader parser callbacks

Returns

- Error code indicating success or failure.

Definition at line `178` of file `platform/bootloader/communication/bgapi-uart-dfu/btl_comm_bgapi.h`

bootloader_ezsp_communication_init

void bootloader_ezsp_communication_init (void)

Initialize hardware for the EZSP SPI Bootloader communication.

Parameters

N/A		

Definition at line `197` of file `platform/bootloader/communication/ezsp-spi/btl_ezsp_spi.h`

bootloader_ezsp_communication_start

int32_t bootloader_ezsp_communication_start (void)

Initialize communication between the EZSP SPI bootloader and external host.

Parameters

N/A		

Returns

- Error code indicating success or failure.

Definition at line `205` of file `platform/bootloader/communication/ezsp-spi/btl_ezsp_spi.h`

bootloader_ezsp_communication_main

int32_t bootloader_ezsp_communication_main (ImageProperties_t *imageProps, ParserContext_t *parserContext, const BootloaderParserCallbacks_t *parseCb)

Communication main for the EZSP SPI bootloader.

Parameters

N/A	imageProps	The image file processed
N/A	parserContext	Image parser context

N/A	parseCb	Bootloader parser callbacks

Returns

- Error code indicating success or failure.

Definition at line 216 of file platform/bootloader/communication/ezsp-spi/btl_ezsp_spi.h

bootloader_ezsp_communication_shutdown

> void bootloader_ezsp_communication_shutdown (void)

Stop communication between the bootloader and external host.

Parameters

N/A		

Definition at line 223 of file platform/bootloader/communication/ezsp-spi/btl_ezsp_spi.h

UART XMODEM

UART XMODEM

By enabling the UART XMODEM communication component, the bootloader communication interface implements the XMODEM-CRC protocol over UART.

This component makes the bootloader compatible with the legacy `serial-uart-bootloader` that was previously released with the EmberZnet and SL-Thread wireless stacks.

Bluetooth Apploader OTA DFU

Bluetooth Apploader OTA DFU

By enabling the Apploader communication component, the bootloader communication interface implements Bluetooth Apploader over-the-air (OTA) device firmware upgrade (DFU) protocol.

BGAPI UART DFU

BGAPI UART DFU

By enabling the BGAPI communication component, the bootloader communication interface implements the UART DFU protocol using BGAPI commands.

This component makes the bootloader compatible with the legacy UART bootloader that was previously released with the Silicon Labs Bluetooth stack.

EZSP-SPI

EZSP-SPI

By enabling the EZSP-SPI communication component, the bootloader communication interface implements the EZSP protocol over SPI.

This component makes the bootloader compatible with the legacy `ezsp-spi-bootloader` that was previously released with the EmberZnet and SL-Thread wireless stacks.

Utils

Utils

Modules

XMODEM Parser

XMODEM Parser

Parser for XMODEM packets.

XMODEM packet parser supporting XMODEM-CRC.

Modules

XmodemPacket_t

Commands

Functions

void	xmodem_reset(void) Reset the XMODEM parser to start a new transfer.
int32_t	xmodem_parsePacket(XmodemPacket_t *packet, uint8_t *response) Parse an XMODEM packet.
uint8_t	xmodem_getLastPacketNumber(void) Return the packet number of the last packet that was successfully parsed.

Macros

#define	XMODEM_DATA_SIZE 128 Size of an XMODEM packet.

Function Documentation

xmodem_reset

```
void xmodem_reset (void)
```

Reset the XMODEM parser to start a new transfer.

Parameters

N/A		

Definition at line `84` of file `platform/bootloader/communication/xmodem-parser/btl_xmodem.h`

xmodem_parsePacket

```
int32_t xmodem_parsePacket (XmodemPacket_t *packet, uint8_t *response)
```

Parse an XMODEM packet.

Parameters

[in]	packet	The XMODEM packet to parse.
[out]	response	The XMODEM response to the parsed frame

Returns

- BOOTLOADER_OK on success, else error code

Definition at line 94 of file platform/bootloader/communication/xmodem-parser/btl_xmodem.h

xmodem_getLastPacketNumber

```
uint8_t xmodem_getLastPacketNumber (void)
```

Return the packet number of the last packet that was successfully parsed.

Parameters

N/A		

Returns

- Last packet number received. Defaults to 0 if nothing has been received yet.

Definition at line 102 of file platform/bootloader/communication/xmodem-parser/btl_xmodem.h

Macro Definition Documentation

XMODEM_DATA_SIZE

```
#define XMODEM_DATA_SIZE
```

Value:

```
128
```

Size of an XMODEM packet.

Definition at line 45 of file platform/bootloader/communication/xmodem-parser/btl_xmodem.h

XmodemPacket_t

XMODEM packet.

Public Attributes

uint8_t	header Packet header (XMODEM_CMD_SOH)
uint8_t	packetNumber Packet sequence number.
uint8_t	packetNumberC Complement of packet sequence number.
uint8_t	data Payload.
uint8_t	crcH CRC high byte.
uint8_t	crcL CRC low byte.

Public Attribute Documentation

header

```
uint8_t XmodemPacket_t::header
```

Packet header (XMODEM_CMD_SOH)

Definition at line 72 of file platform/bootloader/communication/xmodem-parser/btl_xmodem.h

packetNumber

```
uint8_t XmodemPacket_t::packetNumber
```

Packet sequence number.

Definition at line 73 of file platform/bootloader/communication/xmodem-parser/btl_xmodem.h

packetNumberC

```
uint8_t XmodemPacket_t::packetNumberC
```

Complement of packet sequence number.

Definition at line 74 of file platform/bootloader/communication/xmodem-parser/btl_xmodem.h

> uint8_t XmodemPacket_t::data[XMODEM_DATA_SIZE]

Payload.

Definition at line 75 of file platform/bootloader/communication/xmodem-parser/btl_xmodem.h

crcH

> uint8_t XmodemPacket_t::crcH

CRC high byte.

Definition at line 76 of file platform/bootloader/communication/xmodem-parser/btl_xmodem.h

crcL

> uint8_t XmodemPacket_t::crcL

CRC low byte.

Definition at line 77 of file platform/bootloader/communication/xmodem-parser/btl_xmodem.h

Commands

Commands

Macros

#define XMODEM_CMD_SOH (0×01)
Start of Header.

#define XMODEM_CMD_EOT (0×04)
End of Transmission.

#define XMODEM_CMD_ACK (0×06)
Acknowledge.

#define XMODEM_CMD_NAK (0×15)
Not Acknowledge.

#define XMODEM_CMD_CAN (0×18)
Cancel.

#define XMODEM_CMD_CTRL_C (0×03)
Ctrl+C.

#define XMODEM_CMD_C (0×43)
ASCII 'C'.

Macro Definition Documentation

XMODEM_CMD_SOH

```
#define XMODEM_CMD_SOH
```

Value:

```
(0×01)
```

Start of Header.

Definition at line 53 of file platform/bootloader/communication/xmodem-parser/btl_xmodem.h

XMODEM_CMD_EOT

```
#define XMODEM_CMD_EOT
```

Value:

```
(0×04)
```

End of Transmission.

Definition at line 55 of file platform/bootloader/communication/xmodem-parser/btl_xmodem.h

XMODEM_CMD_ACK

```
#define XMODEM_CMD_ACK
```

Value:

```
(0x06)
```

Acknowledge.

Definition at line 57 of file platform/bootloader/communication/xmodem-parser/btl_xmodem.h

XMODEM_CMD_NAK

```
#define XMODEM_CMD_NAK
```

Value:

```
(0x15)
```

Not Acknowledge.

Definition at line 59 of file platform/bootloader/communication/xmodem-parser/btl_xmodem.h

XMODEM_CMD_CAN

```
#define XMODEM_CMD_CAN
```

Value:

```
(0x18)
```

Cancel.

Definition at line 61 of file platform/bootloader/communication/xmodem-parser/btl_xmodem.h

XMODEM_CMD_CTRL_C

```
#define XMODEM_CMD_CTRL_C
```

Value:

```
(0x03)
```

Ctrl+C.

Definition at line 63 of file platform/bootloader/communication/xmodem-parser/btl_xmodem.h

XMODEM_CMD_C

```
#define XMODEM_CMD_C
```

Value:

(0x43)

ASCII 'C'.

Definition at line 65 of file platform/bootloader/communication/xmodem-parser/btl_xmodem.h

Debug

Debug

Debug Component.

This Component provides the bootloader with support for debugging functions. The Component implements two types of debugging functionality:

- Defining SL_DEBUG_ASSERT enables assertions on compile-time configurable parameters in the bootloader
- Defining SL_DEBUG_PRINT enables debug prints at strategic points in the code.

Macros

#define BTL_ASSERT (exp)
Assertion in bootloader.

#define BTL_DEBUG_INIT ()
Initialize debug output.

#define BTL_DEBUG_PRINT (str)
Print a string to debug out.

#define BTL_DEBUG_PRINTLN (str)
Print a string followed by a newline to debug out.

#define BTL_DEBUG_PRINTC (chr)
Print a character to debut out.

#define BTL_DEBUG_PRINT_CHAR_HEX (number)
Print a single hex byte.

#define BTL_DEBUG_PRINT_SHORT_HEX (number)
Print two hex bytes.

#define BTL_DEBUG_PRINT_WORD_HEX (number)
Print a hex word.

#define BTL_DEBUG_PRINT_LF ()
Print a newline.

Macro Definition Documentation

BTL_ASSERT

```
#define BTL_ASSERT
```

Value:

```
(exp)
```

Assertion in bootloader.

Definition at line 61 of file platform/bootloader/debug/btl_debug.h

BTL_DEBUG_INIT

```
#define BTL_DEBUG_INIT
```

Value:

```
()
```

Initialize debug output.

Definition at line 97 of file platform/bootloader/debug/btl_debug.h

BTL_DEBUG_PRINT

```
#define BTL_DEBUG_PRINT
```

Value:

```
(str)
```

Print a string to debug out.

Definition at line 99 of file platform/bootloader/debug/btl_debug.h

BTL_DEBUG_PRINTLN

```
#define BTL_DEBUG_PRINTLN
```

Value:

```
(str)
```

Print a string followed by a newline to debug out.

Definition at line 101 of file platform/bootloader/debug/btl_debug.h

BTL_DEBUG_PRINTC

```
#define BTL_DEBUG_PRINTC
```

Value:

```
(chr)
```

Print a character to debut out.

Definition at line 103 of file platform/bootloader/debug/btl_debug.h

BTL_DEBUG_PRINT_CHAR_HEX

```
#define BTL_DEBUG_PRINT_CHAR_HEX
```

Value:

```
(number)
```

Print a single hex byte.

Definition at line 105 of file platform/bootloader/debug/btl_debug.h

BTL_DEBUG_PRINT_SHORT_HEX

```
#define BTL_DEBUG_PRINT_SHORT_HEX
```

Value:

```
(number)
```

Print two hex bytes.

Definition at line 107 of file platform/bootloader/debug/btl_debug.h

BTL_DEBUG_PRINT_WORD_HEX

```
#define BTL_DEBUG_PRINT_WORD_HEX
```

Value:

```
(number)
```

Print a hex word.

Definition at line 109 of file platform/bootloader/debug/btl_debug.h

BTL_DEBUG_PRINT_LF

```
#define BTL_DEBUG_PRINT_LF
```

Value:

```
()
```

Print a newline.

Definition at line 111 of file platform/bootloader/debug/btl_debug.h

Decompressor

Decompressor

Decompressors for Gecko Bootloader.

The decompressor module adds support for decompressing OTA upgrade files that have been compressed using the LZ4 or the LZMA compression scheme.

Modules

LZ4 Decompressor

LZ4 Decompressor

LZ4 decompressor LZ4 is a lossless data compression algorithm that is focused on compression and decompression speed.

It belongs to the LZ77 family of byte-oriented compression schemes.

Modules

Lz4Context_t

Typedefs

typedef int32_t(* Lz4DataWrite_t)(uint8_t *data, size_t length)
Function to output data from LZ4 decompressor.

typedef int32_t(* Lz4DataRead_t)(size_t backtrackOffset, uint8_t *data, size_t length)
Function to read data into LZ4 decompressor.

Functions

int32_t lz4_init(Lz4Context_t *ctx, Lz4DataRead_t readFunction)
Initialize the LZ4 decompressor.

int32_t lz4_decompress(Lz4Context_t *ctx, void *inputData, size_t inputLength, Lz4DataWrite_t writeFunction)
Decompress a chunk of data.

int32_t lz4_finish(Lz4Context_t *ctx)
Finish decompressing data.

Macros

#define LZ4_STATE_TOKEN 00U
LZ4 state machine: Token byte.

#define LZ4_STATE_LITERAL_LENGTH 10U
LZ4 state machine: Literal length byte.

#define LZ4_STATE_LITERAL_VALUE 20U
LZ4 state machine: Literal value byte.

#define LZ4_STATE_OFFSET_LSB 30U
LZ4 state machine: LSB of match offset.

#define LZ4_STATE_OFFSET_MSB 35U
LZ4 state machine: MSB of match offset.

#define LZ4_STATE_MATCH_LENGTH 40U
LZ4 state machine: Match length.

#define LZ4_STATE_BACKTRACKING 50U
LZ4 state machine: Backtracking to get matched data.

Typedef Documentation

Lz4DataWrite_t

```
typedef int32_t(* Lz4DataWrite_t) (uint8_t *data, size_t length) )(uint8_t *data, size_t length)
```

Function to output data from LZ4 decompressor.

Definition at line 59 of file platform/bootloader/parser/compression/btl_decompress_lz4.h

Lz4DataRead_t

```
typedef int32_t(* Lz4DataRead_t) (size_t backtrackOffset, uint8_t *data, size_t length) )(size_t backtrackOffset, uint8_t *data, size_t length)
```

Function to read data into LZ4 decompressor.

Definition at line 61 of file platform/bootloader/parser/compression/btl_decompress_lz4.h

Function Documentation

lz4_init

```
int32_t lz4_init (Lz4Context_t *ctx, Lz4DataRead_t readFunction)
```

Initialize the LZ4 decompressor.

Parameters

N/A	ctx	Decompressor context
N/A	readFunction	Function pointer to read back previously written data

Returns

- Error code

Definition at line 84 of file platform/bootloader/parser/compression/btl_decompress_lz4.h

lz4_decompress

```
int32_t lz4_decompress (Lz4Context_t *ctx, void *inputData, size_t inputLength, Lz4DataWrite_t writeFunction)
```

Decompress a chunk of data.

Parameters

N/A	ctx	Decompressor context
N/A	inputData	Compressed input data
N/A	inputLength	Length of inputData in bytes
N/A	writeFunction	Function pointer that is called with decompressed data

Returns

- Error code

Definition at line 95 of file platform/bootloader/parser/compression/btl_decompress_lz4.h

lz4_finish

```
int32_t lz4_finish (Lz4Context_t *ctx)
```

Finish decompressing data.

Parameters

N/A	ctx	Decompressor context

Returns

- Error code indicating success or failure

Definition at line 109 of file platform/bootloader/parser/compression/btl_decompress_lz4.h

Macro Definition Documentation

LZ4_STATE_TOKEN

```
#define LZ4_STATE_TOKEN
```

Value:

```
00U
```

LZ4 state machine: Token byte.

Definition at line 44 of file platform/bootloader/parser/compression/btl_decompress_lz4.h

LZ4_STATE_LITERAL_LENGTH

```
#define LZ4_STATE_LITERAL_LENGTH
```

Value:

```
10U
```

LZ4 state machine: Literal length byte.

Definition at line 46 of file platform/bootloader/parser/compression/btl_decompress_lz4.h

LZ4_STATE_LITERAL_VALUE

```
#define LZ4_STATE_LITERAL_VALUE
```

Value:

```
20U
```

LZ4 state machine: Literal value byte.

Definition at line 48 of file platform/bootloader/parser/compression/btl_decompress_lz4.h

LZ4_STATE_OFFSET_LSB

```
#define LZ4_STATE_OFFSET_LSB
```

Value:

```
30U
```

LZ4 state machine: LSB of match offset.

Definition at line 50 of file platform/bootloader/parser/compression/btl_decompress_lz4.h

LZ4_STATE_OFFSET_MSB

```
#define LZ4_STATE_OFFSET_MSB
```

Value:

```
35U
```

LZ4 state machine: MSB of match offset.

Definition at line 52 of file platform/bootloader/parser/compression/btl_decompress_lz4.h

LZ4_STATE_MATCH_LENGTH

```
#define LZ4_STATE_MATCH_LENGTH
```

Value:

```
40U
```

LZ4 state machine: Match length.

Definition at line 54 of file platform/bootloader/parser/compression/btl_decompress_lz4.h

LZ4_STATE_BACKTRACKING

```
#define LZ4_STATE_BACKTRACKING
```

Value:

```
50U
```

LZ4 state machine: Backtracking to get matched data.

Definition at line 56 of file platform/bootloader/parser/compression/btl_decompress_lz4.h

Lz4Context_t

LZ4 decompressor context.

Public Attributes

uint32_t	literalLength Length of literals.
uint32_t	matchLength Length of match.
uint16_t	backtrackOffset Offset from current write for match.
uint8_t	state Current decompressor state.
Lz4DataRead_t	readFunction Function to read previously written data into decompressor during backtracking.

Public Attribute Documentation

literalLength

```
uint32_t Lz4Context_t::literalLength
```

Length of literals.

Definition at line 66 of file platform/bootloader/parser/compression/btl_decompress_lz4.h

matchLength

```
uint32_t Lz4Context_t::matchLength
```

Length of match.

Definition at line 68 of file platform/bootloader/parser/compression/btl_decompress_lz4.h

backtrackOffset

```
uint16_t Lz4Context_t::backtrackOffset
```

Offset from current write for match.

Definition at line 70 of file platform/bootloader/parser/compression/btl_decompress_lz4.h

state

> uint8_t Lz4Context_t::state

Current decompressor state.

Definition at line 72 of file platform/bootloader/parser/compression/btl_decompress_lz4.h

readFunction

> Lz4DataRead_t Lz4Context_t::readFunction

Function to read previously written data into decompressor during backtracking.

Definition at line 74 of file platform/bootloader/parser/compression/btl_decompress_lz4.h

GPIO Activation

GPIO Activation

Enter bootloader based on GPIO state.

This component provides functionality to enter firmware upgrade mode automatically after reset if a GPIO pin is active during boot.

The GPIO pin location and polarity are configurable.

Modules

Button GPIO

EZSP GPIO

Button GPIO

Button GPIO

Enter bootloader based on Button GPIO state.

By enabling the GPIO activation component, the firmware upgrade mode can be activated by the GPIO configuration.

Functions

bool gpio_enterBootloader(void)
Enter the bootlader if the GPIO pin is active.

Function Documentation

gpio_enterBootloader

```
bool gpio_enterBootloader (void)
```

Enter the bootlader if the GPIO pin is active.

Parameters

N/A		

Returns

- True if the bootloader should be entered

Definition at line 41 of file platform/bootloader/gpio/gpio-activation/btl_gpio_activation.h

EZSP GPIO

EZSP GPIO

Enter bootloader based on EZSP GPIO state.

The EZSP communication protocol over SPI can be used together with this component.

By enabling the EZSP GPIO component, the firmware upgrade mode can be entered by activating the nWake pin.

Functions

bool ezsp_gpio_enterBootloader(void)
Enter the bootlader if the GPIO pin is active.

Function Documentation

ezsp_gpio_enterBootloader

```
bool ezsp_gpio_enterBootloader (void)
```

Enter the bootlader if the GPIO pin is active.

Parameters

N/A		

Returns

- True if the bootloader should be entered

Definition at line 38 of file platform/bootloader/gpio/ezsp-gpio-activation/btl_ezsp_gpio_activation.h

Image Parser

Image Parser

The image parser parses the data and returns bootloader upgrade data in a callback.

Modules

GBL Parser

GBL Parser

GBL Parser

GBL parser implementation.

Image parser for GBL files. Parses GBL files based on the GBL file format specification. Callbacks are used to present data and metadata contents of the GBL file to the bootloader.

Modules

GblTagParsingInfo_t

ImageProperties_t

ParserContext_t

GblInputBuffer_t

Custom GBL Tags

GBL Format

Enumerations

enum GblParserState_t {

GblParserStateInit
GblParserStateIdle
GblParserStateHeader
GblParserStateBootloader
GblParserStateBootloaderData
GblParserStateApplication
GblParserStateMetadata
GblParserStateMetadataData
GblParserStateProg
GblParserStateProgData
GblParserStateEraseProg
GblParserStateFinalize
GblParserStateDone
GblParserStateEncryptionInit
GblParserStateEncryptionContainer
GblParserStateSignature
GblParserStateError

}

State in the GBL parser state machine.

Functions

const GblTagParsingInfo_t * gbl_getTagParsingInfoFromTagId(uint32_t tagId)

Function for looking up and retrieving the parsing information struct associated with a particular GBL tag type / tag ID.

int32_t parser_init(void *context, void *decryptContext, void *authContext, uint8_t flags)

Initialize the parser's context.

int32_t parser_parse(void *context, ImageProperties_t *imageProperties, uint8_t buffer[], size_t length, const BootloaderParserCallbacks_t *callbacks)
Parse an image file to extract the binary and some metadata.

int32_t parser_verifyCertificate(void *context, void *input, void *blProperties, void *shaState)
Verify the GBL certificate.

int32_t gbl_writeProgData(ParserContext_t *context, uint8_t buffer[], size_t length, const BootloaderParserCallbacks_t *callbacks)
Write application data.

Macros

#define PARSER_FLAG_ENCRYPTED (1U << 0U)
GBL file is encrypted.

#define PARSER_FLAG_PARSE_CUSTOM_TAGS (1U << 5U)
Parse custom tags rather than silently traversing them.

#define PARSER_FLAGS_PUBLIC_MASK PARSER_FLAG_PARSE_CUSTOM_TAGS
Some flags are public, some are internal to the parser.

#define GBL_PARSER_BUFFER_SIZE 64UL
GBL parser buffer size.

#define PARSER_REQUIRE_AUTHENTICITY (false)
Bootloader/parser configurations.

#define PARSER_REQUIRE_CONFIDENTIALITY (false)
Parser requires upgrade images to be encrypted, providing confidentiality, if true.

#define PARSER_REQUIRE_CERTIFICATE_AUTHENTICITY (false)
Parser requires upgrade images to be authenticated by the bootloader certificate, if true.

#define PARSER_REQUIRE_ANTI_ROLLBACK_PROTECTION (false)
Parser requires rollback protection of applications, if true.

#define PARSER_APPLICATION_MINIMUM_VERSION_VALID (0UL)
Defines the minimum application version that can be accepted.

#define BTL_IMAGE_CONTENT_APPLICATION 0x01U
Upgrade image contains application upgrade.

#define BTL_IMAGE_CONTENT_BOOTLOADER 0x02U
Upgrade image contains bootloader upgrade.

#define BTL_IMAGE_CONTENT_SE 0x04U
Upgrade image contains SE upgrade.

#define BTL_IMAGE_INSTRUCTION_APPLICATION 0x01U
Application upgrade should be applied from upgrade image.

#define BTL_IMAGE_INSTRUCTION_BOOTLOADER 0x02U
Bootloader upgrade should be applied from upgrade image.

#define BTL_IMAGE_INSTRUCTION_SE 0x04U
SE upgrade should be applied from upgrade image.

Enumeration Documentation

GblParserState_t

```
GblParserState_t
```

State in the GBL parser state machine.

Enumerator	
GblParserStateInit	Initial state.
GblParserStateIdle	Idle state.
GblParserStateHeader	Parsing header tag.
GblParserStateBootloader	Parsing bootloader tag.
GblParserStateBootloaderData	Parsing bootloader tag data.
GblParserStateApplication	Parsing application tag.
GblParserStateMetadata	Parsing metadata tag.
GblParserStateMetadataData	Parsing metadata tag data.
GblParserStateProg	Parsing flash program tag.
GblParserStateProgData	Parsing flash program tag data.
GblParserStateEraseProg	Parsing flash erase&program tag.
GblParserStateFinalize	Finalizing file.
GblParserStateDone	Parsing complete.
GblParserStateEncryptionInit	Parsing encryption init tag.
GblParserStateEncryptionContainer	Parsing encryption data tag.
GblParserStateSignature	Parsing signature tag.
GblParserStateError	Error state.

Definition at line `107` of file `platform/bootloader/parser/gbl/btl_gbl_parser.h`

Function Documentation

gbl_getTagParsingInfoFromTagId

```
const GblTagParsingInfo_t * gbl_getTagParsingInfoFromTagId (uint32_t tagId)
```

Function for looking up and retrieving the parsing information struct associated with a particular GBL tag type / tag ID.

Parameters

[in]	tagId	The tag ID to be looked up.

Returns

- A pointer to the GblTagParsingInfo_t struct associated with tagId, or a NULL pointer if the provided tagId cannot be found.

Definition at line `300` of file `platform/bootloader/parser/gbl/btl_gbl_parser.h`

parser_init

```
int32_t parser_init (void *context, void *decryptContext, void *authContext, uint8_t flags)
```

Initialize the parser's context.

Parameters

N/A	context	Pointer to context for the parser implementation
N/A	decryptContext	Pointer to context for decryption of parsed file
N/A	authContext	Pointer to context for authentication of parsed file
N/A	flags	Flags for parser support

Returns

- BOOTLOADER_OK if OK, error code otherwise.

Definition at line 312 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

parser_parse

```
int32_t parser_parse (void *context, ImageProperties_t *imageProperties, uint8_t buffer[], size_t length, const BootloaderParserCallbacks_t *callbacks)
```

Parse an image file to extract the binary and some metadata.

Parameters

N/A	context	Pointer to the specific parser's context variable
N/A	imageProperties	Pointer to the image file state variable
N/A	buffer	Pointer to byte array containing data to parse
N/A	length	Size in bytes of the data in buffer
N/A	callbacks	Struct containing function pointers to be called by the parser to pass the extracted binary data back to BTL.

Pushes data into the image file parser to be parsed.

Returns

- BOOTLOADER_OK if OK, error code otherwise.

Definition at line 331 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

parser_verifyCertificate

```
int32_t parser_verifyCertificate (void *context, void *input, void *blProperties, void *shaState)
```

Verify the GBL certificate.

Parameters

[inout]	context	GBL parser context
[in]	input	Input data
[in]	blProperties	Pointer to ApplicationProperties_t of bootloader
[inout]	shaState	Pointer to Sha256Context_t used to store SHA256 of GBL certificate

Note

- The behavior of this function depends on the context state.

Returns

-

BOOTLOADER_ERROR_PARSER_PARSED if done parsing the current input buffer. BOOTLOADER_OK if input data is stored in the internal buffer. BOOTLOADER_OK if the certificate in GBL is accepted. BOOTLOADER_ERROR_PARSER_SIGNATURE if the certificate in GBL is rejected.

Definition at line 372 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

gbl_writeProgData

```
int32_t gbl_writeProgData (ParserContext_t *context, uint8_t buffer[], size_t length, const BootloaderParserCallbacks_t *callbacks)
```

Write application data.

Parameters

N/A	context	GBL parser context
N/A	buffer	Input buffer containing data to be written
N/A	length	Size of input buffer
N/A	callbacks	GBL Parser callbacks for writing data

This function is called when parsing any tag with GblProg_t structured content.

Returns

- Error code

Definition at line 388 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

Macro Definition Documentation

PARSER_FLAG_ENCRYPTED

```
#define PARSER_FLAG_ENCRYPTED
```

Value:

```
(1U << 0U)
```

GBL file is encrypted.

Definition at line 53 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

PARSER_FLAG_PARSE_CUSTOM_TAGS

```
#define PARSER_FLAG_PARSE_CUSTOM_TAGS
```

Value:

```
(1U << 5U)
```

Parse custom tags rather than silently traversing them.

Definition at line 55 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

PARSER_FLAGS_PUBLIC_MASK

#define PARSER_FLAGS_PUBLIC_MASK

Value:

```
PARSER_FLAG_PARSE_CUSTOM_TAGS
```

Some flags are public, some are internal to the parser.

Definition at line 58 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

GBL_PARSER_BUFFER_SIZE

#define GBL_PARSER_BUFFER_SIZE

Value:

```
64UL
```

GBL parser buffer size.

Definition at line 61 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

PARSER_REQUIRE_AUTHENTICITY

#define PARSER_REQUIRE_AUTHENTICITY

Value:

```
(false)
```

Bootloader/parser configurations.

Parser requires upgrade images to be signed, providing authenticity, if true.

Definition at line 70 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

PARSER_REQUIRE_CONFIDENTIALITY

#define PARSER_REQUIRE_CONFIDENTIALITY

Value:

```
(false)
```

Parser requires upgrade images to be encrypted, providing confidentiality, if true.

Definition at line 77 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

PARSER_REQUIRE_CERTIFICATE_AUTHENTICITY

#define PARSER_REQUIRE_CERTIFICATE_AUTHENTICITY

Value:

```
(false)
```

Parser requires upgrade images to be authenticated by the bootloader certificate, if true.

Definition at line 85 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

PARSER_REQUIRE_ANTI_ROLLBACK_PROTECTION

```
#define PARSER_REQUIRE_ANTI_ROLLBACK_PROTECTION
```

Value:

```
(false)
```

Parser requires rollback protection of applications, if true.

Definition at line 93 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

PARSER_APPLICATION_MINIMUM_VERSION_VALID

```
#define PARSER_APPLICATION_MINIMUM_VERSION_VALID
```

Value:

```
(0UL)
```

Defines the minimum application version that can be accepted.

Definition at line 100 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

BTL_IMAGE_CONTENT_APPLICATION

```
#define BTL_IMAGE_CONTENT_APPLICATION
```

Value:

```
0x01U
```

Upgrade image contains application upgrade.

Definition at line 195 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

BTL_IMAGE_CONTENT_BOOTLOADER

```
#define BTL_IMAGE_CONTENT_BOOTLOADER
```

Value:

```
0x02U
```

Upgrade image contains bootloader upgrade.

Definition at line 197 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

BTL_IMAGE_CONTENT_SE

```
#define BTL_IMAGE_CONTENT_SE
```

Value:

```
0x04U
```

Upgrade image contains SE upgrade.

Definition at line 199 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

BTL_IMAGE_INSTRUCTION_APPLICATION

```
#define BTL_IMAGE_INSTRUCTION_APPLICATION
```

Value:

```
0x01U
```

Application upgrade should be applied from upgrade image.

Definition at line 202 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

BTL_IMAGE_INSTRUCTION_BOOTLOADER

```
#define BTL_IMAGE_INSTRUCTION_BOOTLOADER
```

Value:

```
0x02U
```

Bootloader upgrade should be applied from upgrade image.

Definition at line 204 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

BTL_IMAGE_INSTRUCTION_SE

```
#define BTL_IMAGE_INSTRUCTION_SE
```

Value:

```
0x04U
```

SE upgrade should be applied from upgrade image.

Definition at line 206 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

GblTagParsingInfo_t

GBL tag parsing info struct. Contains information about the GBL tag order, etc.

Public Attributes

uint32_t	tagId Tag ID.
GblParserState_t	parserState The parser state associated with the tag.
uint8_t	tagOrder Encodes correct order of occurrence in GBL.
uint8_t	reserved Reserved.
uint16_t	flags Flags defining parser behavior.

Public Attribute Documentation

tagId

```
uint32_t GblTagParsingInfo_t::tagId
```

Tag ID.

Definition at line 145 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

parserState

```
GblParserState_t GblTagParsingInfo_t::parserState
```

The parser state associated with the tag.

Definition at line 146 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

tagOrder

```
uint8_t GblTagParsingInfo_t::tagOrder
```

Encodes correct order of occurrence in GBL.

Definition at line 147 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

reserved

uint8_t GblTagParsingInfo_t::reserved

Reserved.

Definition at line 148 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

flags

uint16_t GblTagParsingInfo_t::flags

Flags defining parser behavior.

Definition at line 149 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

ImageProperties_t

Structure containing state of the image file processed.

Public Attributes

uint8_t	contents Image contents.
uint8_t	instructions Parser instructions.
bool	imageCompleted Flag to indicate parsing has completed.
bool	imageVerified Flag to indicate the image file has been validated.
uint32_t	bootloaderVersion Version number of main bootloader extracted from image file.
ApplicationData_t	application Information about the application.
uint32_t	bootloaderUpgradeSize Size of the bootloader upgrade contained in the image file.

Public Attribute Documentation

contents

```
uint8_t ImageProperties_t::contents
```

Image contents.

Definition at line 155 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

instructions

```
uint8_t ImageProperties_t::instructions
```

Parser instructions.

Definition at line 157 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

imageCompleted

```
bool ImageProperties_t::imageCompleted
```

Flag to indicate parsing has completed.

Definition at line 159 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

imageVerified

```
bool ImageProperties_t::imageVerified
```

Flag to indicate the image file has been validated.

Definition at line 161 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

bootloaderVersion

```
uint32_t ImageProperties_t::bootloaderVersion
```

Version number of main bootloader extracted from image file.

Definition at line 163 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

application

```
ApplicationData_t ImageProperties_t::application
```

Information about the application.

Definition at line 165 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

bootloaderUpgradeSize

```
uint32_t ImageProperties_t::bootloaderUpgradeSize
```

Size of the bootloader upgrade contained in the image file.

Definition at line 167 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

ParserContext_t

Image parser context definition.

Public Attributes

uint8_t	internalBuffer Buffer contents.
uint8_t	bytesInInternalBuffer Amount of bytes present in buffer.
uint8_t	internalBufferOffset Current reading offset into the buffer (circular)
uint8_t	flags Parser flags.
bool	inEncryptedContainer Parser is currently inside an encrypted tag.
bool	gotSignature Parser has received and verified signature.
uint8_t	receivedFlags Parser has received bootloader upgrade tag.
GblParserState_t	internalState State of the GBL parser state machine.
void *	aesContext AES-CCM decryption (= AES-CTR) context.
void *	shaContext SHA256 hashing context.
size_t	lengthOfTag Total length of the tag currently being parsed.
size_t	offsetInTag Current offset into tag being parsed.
size_t	lengthOfEncryptedTag Total length of current encrypted data block.
size_t	offsetInEncryptedTag Offset into current encrypted data block.
uint32_t	programmingAddress Current address the image needs to be written to.
uint32_t	tagAddress Current offset of metadata/bootloader being handled (starts at 0)
uint8_t	withheldApplicationVectors Withheld application data.

uint8_t	withheldUpgradeVectors Withheld bootloader upgrade data during app parsing.
uint8_t	withheldBootloaderVectors Withheld bootloader upgrade data during bootloader parsing.
uint32_t	fileCrc Running CRC-32 over the incoming GBL file.
uint32_t	customTagId Context for custom tag.
uint8_t	currentTagOrder Current tag order.
uint8_t	reservedFlags Reserved flags.

Public Attribute Documentation

internalBuffer

```
uint8_t ParserContext_t::internalBuffer[64]
```

Buffer contents.

Definition at line 211 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

bytesInInternalBuffer

```
uint8_t ParserContext_t::bytesInInternalBuffer
```

Amount of bytes present in buffer.

Definition at line 213 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

internalBufferOffset

```
uint8_t ParserContext_t::internalBufferOffset
```

Current reading offset into the buffer (circular)

Definition at line 215 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

flags

```
uint8_t ParserContext_t::flags
```

Parser flags.

Definition at line 217 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

inEncryptedContainer

```
bool ParserContext_t::inEncryptedContainer
```

Parser is currently inside an encrypted tag.

Definition at line 219 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

gotSignature

```
bool ParserContext_t::gotSignature
```

Parser has received and verified signature.

Definition at line 221 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

receivedFlags

```
uint8_t ParserContext_t::receivedFlags
```

Parser has received bootloader upgrade tag.

Definition at line 223 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

internalState

```
GblParserState_t ParserContext_t::internalState
```

State of the GBL parser state machine.

Definition at line 225 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

aesContext

```
void* ParserContext_t::aesContext
```

AES-CCM decryption (= AES-CTR) context.

Definition at line 227 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

shaContext

```
void* ParserContext_t::shaContext
```

SHA256 hashing context.

Definition at line 229 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

lengthOfTag

> size_t ParserContext_t::lengthOfTag

Total length of the tag currently being parsed.

Definition at line 231 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

offsetInTag

> size_t ParserContext_t::offsetInTag

Current offset into tag being parsed.

Definition at line 233 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

lengthOfEncryptedTag

> size_t ParserContext_t::lengthOfEncryptedTag

Total length of current encrypted data block.

Definition at line 235 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

offsetInEncryptedTag

> size_t ParserContext_t::offsetInEncryptedTag

Offset into current encrypted data block.

Definition at line 237 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

programmingAddress

> uint32_t ParserContext_t::programmingAddress

Current address the image needs to be written to.

Definition at line 239 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

tagAddress

> uint32_t ParserContext_t::tagAddress

Current offset of metadata/bootloader being handled (starts at 0)

Definition at line 241 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

withheldApplicationVectors

```
uint8_t ParserContext_t::withheldApplicationVectors[24]
```

Withheld application data.

Definition at line 243 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

withheldUpgradeVectors

```
uint8_t ParserContext_t::withheldUpgradeVectors[4]
```

Withheld bootloader upgrade data during app parsing.

Definition at line 245 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

withheldBootloaderVectors

```
uint8_t ParserContext_t::withheldBootloaderVectors[4]
```

Withheld bootloader upgrade data during bootloader parsing.

Definition at line 247 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

fileCrc

```
uint32_t ParserContext_t::fileCrc
```

Running CRC-32 over the incoming GBL file.

Definition at line 249 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

customTagId

```
uint32_t ParserContext_t::customTagId
```

Context for custom tag.

Definition at line 251 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

currentTagOrder

```
uint8_t ParserContext_t::currentTagOrder
```

Current tag order.

Definition at line 269 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

reservedFlags

```
uint8_t ParserContext_t::reservedFlags[3]
```

Reserved flags.

Definition at line 271 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

GblInputBuffer_t

GBL parser input buffer.

Public Attributes

const uint8_t *	buffer Pointer to a buffer.
const size_t	length Length of the buffer.
size_t	offset Offset of the buffer.

Public Attribute Documentation

buffer

```
const uint8_t* GblInputBuffer_t::buffer
```

Pointer to a buffer.

Definition at line 277 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

length

```
const size_t GblInputBuffer_t::length
```

Length of the buffer.

Definition at line 279 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

offset

```
size_t GblInputBuffer_t::offset
```

Offset of the buffer.

Definition at line 281 of file platform/bootloader/parser/gbl/btl_gbl_parser.h

Custom GBL Tags

Handle custom GBL tags for added functionality in the GBL parser.

Modules

GblCustomTag_t

LZ4 Programming Tag

LZMA Programming Tag

Functions

bool	gbl_isCustomTag(GblTagHeader_t *tagHeader) Indicate whether the GBL tag described by tagHeader is a custom tag.
const GblCustomTag_t *	gbl_getCustomTagProperties(uint32_t tagId) Get properties for a custom GBL tag.

Function Documentation

gbl_isCustomTag

```
bool gbl_isCustomTag (GblTagHeader_t *tagHeader)
```

Indicate whether the GBL tag described by tagHeader is a custom tag.

Parameters

N/A	tagHeader	Pointer to the GBL tag header

Returns

- True if the tag is a custom tag, else false

Definition at line `71` of file `platform/bootloader/parser/gbl/btl_gbl_custom_tags.h`

gbl_getCustomTagProperties

```
const GblCustomTag_t * gbl_getCustomTagProperties (uint32_t tagId)
```

Get properties for a custom GBL tag.

Parameters

N/A	tagId	GBL Tag ID of the custom tag

Returns

- Pointer to the custom tag descriptor

Definition at line 79 of file platform/bootloader/parser/gbl/btl_gbl_custom_tags.h

GblCustomTag_t

Custom tag descriptor.

Public Attributes

uint32_t	tagId GBL Tag ID of the custom tag.
int32_t(*	enterTag Function to call upon entering the tag.
int32_t(*	parseTag Function to call while parsing the tag.
int32_t(*	exitTag Function to call upon exiting the tag.
size_t(*	numBytesRequired Function returning how many bytes should be collected before calling parseTag the next time.

Public Attribute Documentation

tagId

```
uint32_t GblCustomTag_t::tagId
```

GBL Tag ID of the custom tag.

Definition at line 46 of file platform/bootloader/parser/gbl/btl_gbl_custom_tags.h

enterTag

```
int32_t(* GblCustomTag_t::enterTag) (ParserContext_t *ctx)
```

Function to call upon entering the tag.

Definition at line 48 of file platform/bootloader/parser/gbl/btl_gbl_custom_tags.h

parseTag

```
int32_t(* GblCustomTag_t::parseTag) (ParserContext_t *ctx, void *data, size_t length, const BootloaderParserCallbacks_t *callbacks)
```

Function to call while parsing the tag.

Definition at line 50 of file platform/bootloader/parser/gbl/btl_gbl_custom_tags.h

exitTag

```
int32_t(* GblCustomTag_t::exitTag) (ParserContext_t *ctx, const BootloaderParserCallbacks_t *callbacks)
```

Function to call upon exiting the tag.

Definition at line 55 of file platform/bootloader/parser/gbl/btl_gbl_custom_tags.h

numBytesRequired

```
size_t(* GblCustomTag_t::numBytesRequired) (ParserContext_t *ctx)
```

Function returning how many bytes should be collected before calling parseTag the next time.

Definition at line 59 of file platform/bootloader/parser/gbl/btl_gbl_custom_tags.h

LZ4 Programming Tag

LZ4 Programming Tag

Tag to handle LZ4 compressed programming data.

Modules

Lz4ParserContext_t

Functions

int32_t gbl_lz4EnterProgTag(ParserContext_t *ctx)
Enter an LZ4 compressed programming tag.

int32_t gbl_lz4ParseProgTag(ParserContext_t *ctx, void *data, size_t length, const BootloaderParserCallbacks_t *callbacks)
Parse a chunk of data from an LZ4 compressed programming tag.

int32_t gbl_lz4ExitProgTag(ParserContext_t *ctx, const BootloaderParserCallbacks_t *callbacks)
Exit an LZ4 compressed programming tag.

size_t gbl_lz4NumBytesRequired(ParserContext_t *ctx)
Number of bytes needed for the next stage of parsing.

Function Documentation

gbl_lz4EnterProgTag

```
int32_t gbl_lz4EnterProgTag (ParserContext_t *ctx)
```

Enter an LZ4 compressed programming tag.

Parameters

N/A	ctx	Parser context

Returns

- Error code

Definition at line 152 of file platform/bootloader/parser/compression/btl_decompress_lz4.h

gbl_lz4ParseProgTag

```
int32_t gbl_lz4ParseProgTag (ParserContext_t *ctx, void *data, size_t length, const BootloaderParserCallbacks_t *callbacks)
```

Parse a chunk of data from an LZ4 compressed programming tag.

Parameters

N/A	ctx	Parser context

N/A	data	Input data to parse
N/A	length	Length of data
N/A	callbacks	Callbacks to call with parsed data

Returns

- Error code

Definition at line 163 of file platform/bootloader/parser/compression/btl_decompress_lz4.h

gbl_lz4ExitProgTag

```
int32_t gbl_lz4ExitProgTag (ParserContext_t *ctx, const BootloaderParserCallbacks_t *callbacks)
```

Exit an LZ4 compressed programming tag.

Parameters

N/A	ctx	Parser context
N/A	callbacks	Callbacks to call with parsed data

Returns

- Error code

Definition at line 175 of file platform/bootloader/parser/compression/btl_decompress_lz4.h

gbl_lz4NumBytesRequired

```
size_t gbl_lz4NumBytesRequired (ParserContext_t *ctx)
```

Number of bytes needed for the next stage of parsing.

Parameters

N/A	ctx	Parser context

Returns

- Number of bytes required

Definition at line 184 of file platform/bootloader/parser/compression/btl_decompress_lz4.h

Lz4ParserContext_t

LZ4 Compressed Programming GBL Tag Parser Context.

Public Attributes

uint8_t	outputBuffer Buffer to store unaligned decompressed data.
uint8_t	outputOffset Offset into outputBuffer.
bool	firstCall Whether this is the first call to the parser for this tag.
ParserContext_t *	parserContext Stored pointer to the GBL parser context.
const BootloaderParserCallbacks_t *	parserCallbacks Stored pointer to the GBL parser callbacks.
Lz4Context_t	lz4Context Context of the LZ4 decompressor.

Public Attribute Documentation

outputBuffer

```
uint8_t Lz4ParserContext_t::outputBuffer[4]
```

Buffer to store unaligned decompressed data.

Definition at line 133 of file platform/bootloader/parser/compression/btl_decompress_lz4.h

outputOffset

```
uint8_t Lz4ParserContext_t::outputOffset
```

Offset into outputBuffer.

Definition at line 135 of file platform/bootloader/parser/compression/btl_decompress_lz4.h

firstCall

```
bool Lz4ParserContext_t::firstCall
```

Whether this is the first call to the parser for this tag.

Definition at line 137 of file platform/bootloader/parser/compression/btl_decompress_lz4.h

parserContext

```
ParserContext_t* Lz4ParserContext_t::parserContext
```

Stored pointer to the GBL parser context.

Definition at line 139 of file platform/bootloader/parser/compression/btl_decompress_lz4.h

parserCallbacks

```
const BootloaderParserCallbacks_t* Lz4ParserContext_t::parserCallbacks
```

Stored pointer to the GBL parser callbacks.

Definition at line 141 of file platform/bootloader/parser/compression/btl_decompress_lz4.h

lz4Context

```
Lz4Context_t Lz4ParserContext_t::lz4Context
```

Context of the LZ4 decompressor.

Definition at line 143 of file platform/bootloader/parser/compression/btl_decompress_lz4.h

LZMA Programming Tag

LZMA Programming Tag

Tag to handle LZMA compressed programming data.

Functions

int32_t gbl_lzmaEnterProgTag(ParserContext_t *ctx)
Enter an LZMA compressed programming tag.

int32_t gbl_lzmaParseProgTag(ParserContext_t *ctx, void *data, size_t length, const BootloaderParserCallbacks_t *callbacks)
Parse a chunk of data from an LZMA compressed programming tag.

int32_t gbl_lzmaExitProgTag(ParserContext_t *ctx, const BootloaderParserCallbacks_t *callbacks)
Exit an LZMA compressed programming tag.

size_t gbl_lzmaNumBytesRequired(ParserContext_t *ctx)
Number of bytes needed for next stage of parsing.

Macros

#define LZMA_COUNTER_SIZE_KB (10UL)
The maximum size of the array holding probability model counters.

#define LZMA_DICT_SIZE_KB (8UL)
The maximum size of the dictionary.

Function Documentation

gbl_lzmaEnterProgTag

```
int32_t gbl_lzmaEnterProgTag (ParserContext_t *ctx)
```

Enter an LZMA compressed programming tag.

Parameters

N/A	ctx	Parser context

Returns

- Error code

Definition at line 67 of file platform/bootloader/parser/compression/btl_decompress_lzma.h

gbl_lzmaParseProgTag

```
int32_t gbl_lzmaParseProgTag (ParserContext_t *ctx, void *data, size_t length, const BootloaderParserCallbacks_t *callbacks)
```

Parse a chunk of data from an LZMA compressed programming tag.

Parameters

N/A	ctx	Parser context
N/A	data	Input data to parse
N/A	length	Length of data
N/A	callbacks	Callbacks to call with parsed data

Returns

- Error code

Definition at line `78` of file `platform/bootloader/parser/compression/btl_decompress_lzma.h`

gbl_lzmaExitProgTag

```
int32_t gbl_lzmaExitProgTag (ParserContext_t *ctx, const BootloaderParserCallbacks_t *callbacks)
```

Exit an LZMA compressed programming tag.

Parameters

N/A	ctx	Parser context
N/A	callbacks	Callbacks to call with parsed data

Returns

- Error code

Definition at line `90` of file `platform/bootloader/parser/compression/btl_decompress_lzma.h`

gbl_lzmaNumBytesRequired

```
size_t gbl_lzmaNumBytesRequired (ParserContext_t *ctx)
```

Number of bytes needed for next stage of parsing.

Parameters

N/A	ctx	Parser context

Returns

- Number of bytes required

Definition at line `99` of file `platform/bootloader/parser/compression/btl_decompress_lzma.h`

Macro Definition Documentation

LZMA_COUNTER_SIZE_KB

```
#define LZMA_COUNTER_SIZE_KB
```

Value:

```
(10UL)
```

The maximum size of the array holding probability model counters.

The size given here sets the limit for the size of the LC and LP constants used by the LZMA compressor. The necessary size of the counter array can be found from size = 4 KiB + 1.5 KiB * (1 << (LC + LP)). LZMA payloads with too large LC + LP can't be decompressed.

Definition at line `50` of file `platform/bootloader/parser/compression/btl_decompress_lzma.h`

LZMA_DICT_SIZE_KB

```
#define LZMA_DICT_SIZE_KB
```

Value:

```
(8UL)
```

The maximum size of the dictionary.

The size given here sets the limit for the size of the dictionary used by the LZMA compressor. LZMA payloads with a dictionary that's too large can't be decompressed.

Definition at line `58` of file `platform/bootloader/parser/compression/btl_decompress_lzma.h`

GBL Format

Modules

- GblTagHeader_t
- GblHeader_t
- VersionDependency_t
- GblApplication_t
- GblBootloader_t
- GblSeUpgrade_t
- GblMetadata_t
- GblProg_t
- GblEnd_t
- GblEncryptionInitAesCcm_t
- GblEncryptionData_t
- GblCertificateEcdsaP256_t
- GblSignatureEcdsaP256_t

Macros

#define GBL_IMAGE_MAGIC_WORD 0xE35050E3UL
Magic word indicating GBL image.

#define GBL_COMPATIBILITY_MAJOR_VERSION 0×03000000UL
Major version of the GBL spec.

#define GBL_TAG_ID_HEADER_V3 0×03A617EBUL
Tag ID for the GBL header tag.

#define GBL_TAG_ID_BOOTLOADER 0xF50909F5UL
Tag ID for the GBL bootloader tag.

#define GBL_TAG_ID_APPLICATION 0xF40A0AF4UL
Tag ID for the GBL application info tag.

#define GBL_TAG_ID_METADATA 0xF60808F6UL
Tag ID for the GBL metadata tag.

#define GBL_TAG_ID_PROG 0xFE0101FEUL
Tag ID for the GBL flash program tag.

#define GBL_TAG_ID_PROG_LZ4 0xFD0505FDUL
Tag ID for the GBL flash program tag (LZ4)

#define GBL_TAG_ID_PROG_LZMA 0xFD0707FDUL
Tag ID for the GBL flash program tag (LZMA)

#define GBL_TAG_ID_ERASEPROG 0xFD0303FDUL
Tag ID for the GBL flash erase&program tag.

#define GBL_TAG_ID_END 0xFC0404FCUL
Tag ID for the GBL end tag.

#define GBL_TAG_ID_SE_UPGRADE 0x5EA617EBUL
Tag ID for the SE upgrade tag.

#define GBL_TAG_ID_VERSION_DEPENDENCY 0x76A617EBUL
Tag ID for the version dependency tag.

#define GBL_TAG_ID_ENC_HEADER 0xFB0505FBUL
Tag ID for the GBL encryption header tag.

#define GBL_TAG_ID_ENC_INIT 0xFA0606FAUL
Tag ID for the GBL encryption init tag.

#define GBL_TAG_ID_ENC_GBL_DATA 0xF90707F9UL
Tag ID for the GBL encryption data tag.

#define GBL_TAG_ID_SIGNATURE_ECDSA_P256 0xF70A0AF7UL
Tag ID for the GBL ECDSA secp256r1 signature tag.

#define GBL_TAG_ID_CERTIFICATE_ECDSA_P256 0xF30B0BF3UL
Tag ID for the GBL ECDSA certfificate tag.

#define GBL_TYPE_NONE 0x00000000UL
GBL type: Standard GBL.

#define GBL_TYPE_ENCRYPTION_AESCCM 0x00000001UL
GBL type: AES-CCM encrypted GBL.

#define GBL_TYPE_SIGNATURE_ECDSA 0x00000100UL
GBL type: ECDSA P256-signed GBL.

#define GBL_VERSION_DEPENDENCY_TYPE_APPLICATION 0x01U
Image Type : Application Image.

#define GBL_VERSION_DEPENDENCY_TYPE_BOOTLOADER 0x02U
Image Type : Bootloader Image.

#define GBL_VERSION_DEPENDENCY_TYPE_SE 0x03U
Image Type : Secure Engine Image.

#define GBL_VERSION_DEPENDENCY_OPERATOR_MASK 0x0FU
Operator encoding : Operator Mask.

#define GBL_VERSION_DEPENDENCY_OPERATOR_SHIFT 0x00U
Operator encoding : Operator Shift.

#define GBL_VERSION_DEPENDENCY_OPERATOR_TYPE_MASK 0x0EU
Operator encoding : Operator Type Mask.

#define GBL_VERSION_DEPENDENCY_OPERATOR_NEGATOR_BIT_MASK 0x01U
Operator encoding : Negator Bit mask.

#define GBL_VERSION_DEPENDENCY_OPERATOR_LT 0x00U
GBL Version Dependency Operator LT.

#define GBL_VERSION_DEPENDENCY_OPERATOR_LEQ 0x02U
GBL Version Dependency Operator LEQ.

#define GBL_VERSION_DEPENDENCY_OPERATOR_EQ 0x04U
GBL Version Dependency Operator EQ.

#define GBL_VERSION_DEPENDENCY_OPERATOR_GEQ 0x06U
GBL Version Dependency Operator GEQ.

#define GBL_VERSION_DEPENDENCY_OPERATOR_GT 0x08U
GBL Version Dependency Operator GT.

#define GBL_VERSION_DEPENDENCY_CONNECTIVE_MASK 0xF0U
Connective encoding : Connective Encoding Mask.

#define GBL_VERSION_DEPENDENCY_CONNECTIVE_SHIFT 0x04U
Connective encoding : Encoding Shift.

#define GBL_VERSION_DEPENDENCY_CONNECTIVE_TYPE_MASK 0x0EU
Connective encoding : Encoding Type Mask.

#define GBL_VERSION_DEPENDENCY_CONNECTIVE_NEGATOR_BIT_MASK 0x01U
Connective encoding : Negator Bit Mask.

#define GBL_VERSION_DEPENDENCY_CONNECTIVE_AND 0x00U
Connective AND Mask.

#define GBL_VERSION_DEPENDENCY_CONNECTIVE_OR 0x02U
Connective OR Mask.

#define GBL_VERSION_DEPENDENCY_SE_VERSION_MASK 0x00FFFFFFUL
SE version mask for ignoring the compatibility byte when comparing versions.

#define GBL_TAG_ORDER_INIT 0U
Tag order.

#define GBL_TAG_ORDER_HEADER_V3 1U
GBL Tag Order Header.

#define GBL_TAG_ORDER_VERSION_DEPENDENCY 2U
GBL Tag Order Version Dependency.

#define GBL_TAG_ORDER_ENC_INIT 4U
GBL Tag Order encoding Init.

#define GBL_TAG_ORDER_APPLICATION 5U
GBL Tag Order Application.

#define GBL_TAG_ORDER_SE_UPGRADE 6U
GBL Tag Order SE Upgarde.

#define GBL_TAG_ORDER_BOOTLOADER 7U
GBL Tag Order Bootloader.

#define GBL_TAG_ORDER_PROG_AND_METADATA 8U
The Programming and Metadata tags share the same order.

#define GBL_TAG_ORDER_ENC_GBL_DATA 9U
GBL Tag Order Enc GBL Data.

#define GBL_TAG_ORDER_CERTIFICATE 10U
GBL Tag Order Certificate.

#define GBL_TAG_ORDER_SIGNATURE 11U
GBL Tag Order Signature.

#define GBL_TAG_ORDER_END 12U
GBL_TAG_ORDER_END must always be strictly greater than any other tag order.

#define GBL_TAG_FLAG_SINGLE_OCCURRENCE_ONLY (1UL << 0)
Flags.

#define GBL_TAG_FLAG_ALWAYS_UNENCRYPTED (1UL << 1)
Flags.

Macro Definition Documentation

GBL_IMAGE_MAGIC_WORD

```
#define GBL_IMAGE_MAGIC_WORD
```

Value:

```
0xE35050E3UL
```

Magic word indicating GBL image.

Definition at line 41 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_COMPATIBILITY_MAJOR_VERSION

```
#define GBL_COMPATIBILITY_MAJOR_VERSION
```

Value:

```
0x03000000UL
```

Major version of the GBL spec.

Definition at line 43 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_TAG_ID_HEADER_V3

```
#define GBL_TAG_ID_HEADER_V3
```

Value:

```
0x03A617EBUL
```

Tag ID for the GBL header tag.

Definition at line 49 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_TAG_ID_BOOTLOADER

```
#define GBL_TAG_ID_BOOTLOADER
```

Value:

```
0xF50909F5UL
```

Tag ID for the GBL bootloader tag.

Definition at line 51 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_TAG_ID_APPLICATION

```
#define GBL_TAG_ID_APPLICATION
```

Value:

```
0xF40A0AF4UL
```

Tag ID for the GBL application info tag.

Definition at line 53 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_TAG_ID_METADATA

```
#define GBL_TAG_ID_METADATA
```

Value:

```
0xF60808F6UL
```

Tag ID for the GBL metadata tag.

Definition at line 55 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_TAG_ID_PROG

```
#define GBL_TAG_ID_PROG
```

Value:

```
0xFE0101FEUL
```

Tag ID for the GBL flash program tag.

Definition at line 57 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_TAG_ID_PROG_LZ4

```
#define GBL_TAG_ID_PROG_LZ4
```

Value:

```
0xFD0505FDUL
```

Tag ID for the GBL flash program tag (LZ4)

Definition at line 59 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_TAG_ID_PROG_LZMA

```
#define GBL_TAG_ID_PROG_LZMA
```

Value:

```
0xFD0707FDUL
```

Tag ID for the GBL flash program tag (LZMA)

Definition at line 61 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_TAG_ID_ERASEPROG

```
#define GBL_TAG_ID_ERASEPROG
```

Value:

```
0xFD0303FDUL
```

Tag ID for the GBL flash erase&program tag.

Definition at line 63 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_TAG_ID_END

```
#define GBL_TAG_ID_END
```

Value:

```
0xFC0404FCUL
```

Tag ID for the GBL end tag.

Definition at line 65 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_TAG_ID_SE_UPGRADE

```
#define GBL_TAG_ID_SE_UPGRADE
```

Value:

```
0x5EA617EBUL
```

Tag ID for the SE upgrade tag.

Definition at line 67 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_TAG_ID_VERSION_DEPENDENCY

```
#define GBL_TAG_ID_VERSION_DEPENDENCY
```

Value:

```
0x76A617EBUL
```

Tag ID for the version dependency tag.

Definition at line 69 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_TAG_ID_ENC_HEADER

```
#define GBL_TAG_ID_ENC_HEADER
```

Value:

```
0xFB0505FBUL
```

Tag ID for the GBL encryption header tag.

Definition at line 73 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_TAG_ID_ENC_INIT

```
#define GBL_TAG_ID_ENC_INIT
```

Value:

```
0xFA0606FAUL
```

Tag ID for the GBL encryption init tag.

Definition at line 75 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_TAG_ID_ENC_GBL_DATA

```
#define GBL_TAG_ID_ENC_GBL_DATA
```

Value:

```
0xF90707F9UL
```

Tag ID for the GBL encryption data tag.

Definition at line 77 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_TAG_ID_SIGNATURE_ECDSA_P256

```
#define GBL_TAG_ID_SIGNATURE_ECDSA_P256
```

Value:

```
0xF70A0AF7UL
```

Tag ID for the GBL ECDSA secp256r1 signature tag.

Definition at line 81 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_TAG_ID_CERTIFICATE_ECDSA_P256

```
#define GBL_TAG_ID_CERTIFICATE_ECDSA_P256
```

Value:

```
0xF30B0BF3UL
```

Tag ID for the GBL ECDSA certfificate tag.

Definition at line 83 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_TYPE_NONE

```
#define GBL_TYPE_NONE
```

Value:

```
0x00000000UL
```

GBL type: Standard GBL.

Definition at line 89 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_TYPE_ENCRYPTION_AESCCM

```
#define GBL_TYPE_ENCRYPTION_AESCCM
```

Value:

```
0x00000001UL
```

GBL type: AES-CCM encrypted GBL.

Definition at line 91 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_TYPE_SIGNATURE_ECDSA

```
#define GBL_TYPE_SIGNATURE_ECDSA
```

Value:

```
0x00000100UL
```

GBL type: ECDSA P256-signed GBL.

Definition at line 93 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_VERSION_DEPENDENCY_TYPE_APPLICATION

```
#define GBL_VERSION_DEPENDENCY_TYPE_APPLICATION
```

Value:

```
0x01U
```

Image Type : Application Image.

Definition at line 99 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_VERSION_DEPENDENCY_TYPE_BOOTLOADER

```
#define GBL_VERSION_DEPENDENCY_TYPE_BOOTLOADER
```

Value:

```
0x02U
```

Image Type : Bootloader Image.

Definition at line 101 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_VERSION_DEPENDENCY_TYPE_SE

```
#define GBL_VERSION_DEPENDENCY_TYPE_SE
```

Value:

```
0x03U
```

Image Type : Secure Engine Image.

Definition at line 103 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_VERSION_DEPENDENCY_OPERATOR_MASK

```
#define GBL_VERSION_DEPENDENCY_OPERATOR_MASK
```

Value:

```
0x0FU
```

Operator encoding : Operator Mask.

Definition at line 106 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_VERSION_DEPENDENCY_OPERATOR_SHIFT

```
#define GBL_VERSION_DEPENDENCY_OPERATOR_SHIFT
```

Value:

```
0x00U
```

Operator encoding : Operator Shift.

Definition at line 108 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_VERSION_DEPENDENCY_OPERATOR_TYPE_MASK

```
#define GBL_VERSION_DEPENDENCY_OPERATOR_TYPE_MASK
```

Value:

```
0x0EU
```

Operator encoding : Operator Type Mask.

Definition at line 110 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_VERSION_DEPENDENCY_OPERATOR_NEGATOR_BIT_MASK

```
#define GBL_VERSION_DEPENDENCY_OPERATOR_NEGATOR_BIT_MASK
```

Value:

```
0x01U
```

Operator encoding : Negator Bit mask.

Definition at line 112 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_VERSION_DEPENDENCY_OPERATOR_LT

```
#define GBL_VERSION_DEPENDENCY_OPERATOR_LT
```

Value:

```
0x00U
```

GBL Version Dependency Operator LT.

Definition at line 115 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_VERSION_DEPENDENCY_OPERATOR_LEQ

```
#define GBL_VERSION_DEPENDENCY_OPERATOR_LEQ
```

Value:

```
0x02U
```

GBL Version Dependency Operator LEQ.

Definition at line 117 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_VERSION_DEPENDENCY_OPERATOR_EQ

```
#define GBL_VERSION_DEPENDENCY_OPERATOR_EQ
```

Value:

```
0x04U
```

GBL Version Dependency Operator EQ.

Definition at line 119 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_VERSION_DEPENDENCY_OPERATOR_GEQ

```
#define GBL_VERSION_DEPENDENCY_OPERATOR_GEQ
```

Value:

```
0x06U
```

GBL Version Dependency Operator GEQ.

Definition at line 121 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_VERSION_DEPENDENCY_OPERATOR_GT

```
#define GBL_VERSION_DEPENDENCY_OPERATOR_GT
```

Value:

```
0x08U
```

GBL Version Dependency Operator GT.

Definition at line 123 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_VERSION_DEPENDENCY_CONNECTIVE_MASK

```
#define GBL_VERSION_DEPENDENCY_CONNECTIVE_MASK
```

Value:

```
0xF0U
```

Connective encoding : Connective Encoding Mask.

Definition at line 126 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_VERSION_DEPENDENCY_CONNECTIVE_SHIFT

```
#define GBL_VERSION_DEPENDENCY_CONNECTIVE_SHIFT
```

Value:

```
0x04U
```

Connective encoding : Encoding Shift.

Definition at line 128 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_VERSION_DEPENDENCY_CONNECTIVE_TYPE_MASK

```
#define GBL_VERSION_DEPENDENCY_CONNECTIVE_TYPE_MASK
```

Value:

```
0x0EU
```

Connective encoding : Encoding Type Mask.

Definition at line 130 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_VERSION_DEPENDENCY_CONNECTIVE_NEGATOR_BIT_MASK

```
#define GBL_VERSION_DEPENDENCY_CONNECTIVE_NEGATOR_BIT_MASK
```

Value:

```
0x01U
```

Connective encoding : Negator Bit Mask.

Definition at line 132 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_VERSION_DEPENDENCY_CONNECTIVE_AND

```
#define GBL_VERSION_DEPENDENCY_CONNECTIVE_AND
```

Value:

```
0x00U
```

Connective AND Mask.

Definition at line 135 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_VERSION_DEPENDENCY_CONNECTIVE_OR

```
#define GBL_VERSION_DEPENDENCY_CONNECTIVE_OR
```

Value:

```
0x02U
```

Connective OR Mask.

Definition at line 137 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_VERSION_DEPENDENCY_SE_VERSION_MASK

```
#define GBL_VERSION_DEPENDENCY_SE_VERSION_MASK
```

Value:

```
0x00FFFFFFUL
```

SE version mask for ignoring the compatibility byte when comparing versions.

Definition at line 140 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_TAG_ORDER_INIT

```
#define GBL_TAG_ORDER_INIT
```

Value:

```
0U
```

Tag order.

Definition at line 149 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_TAG_ORDER_HEADER_V3

```
#define GBL_TAG_ORDER_HEADER_V3
```

Value:

```
1U
```

GBL Tag Order Header.

Definition at line 151 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_TAG_ORDER_VERSION_DEPENDENCY

```
#define GBL_TAG_ORDER_VERSION_DEPENDENCY
```

Value:

```
2U
```

GBL Tag Order Version Dependency.

Definition at line 153 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_TAG_ORDER_ENC_INIT

```
#define GBL_TAG_ORDER_ENC_INIT
```

Value:

```
4U
```

GBL Tag Order encoding Init.

Definition at line 155 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_TAG_ORDER_APPLICATION

```
#define GBL_TAG_ORDER_APPLICATION
```

Value:

```
5U
```

GBL Tag Order Application.

Definition at line 157 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_TAG_ORDER_SE_UPGRADE

```
#define GBL_TAG_ORDER_SE_UPGRADE
```

Value:

```
6U
```

GBL Tag Order SE Upgarde.

Definition at line 167 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_TAG_ORDER_BOOTLOADER

```
#define GBL_TAG_ORDER_BOOTLOADER
```

Value:

```
7U
```

GBL Tag Order Bootloader.

Definition at line 171 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_TAG_ORDER_PROG_AND_METADATA

```
#define GBL_TAG_ORDER_PROG_AND_METADATA
```

Value:

```
8U
```

The Programming and Metadata tags share the same order.

Definition at line 173 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_TAG_ORDER_ENC_GBL_DATA

```
#define GBL_TAG_ORDER_ENC_GBL_DATA
```

Value:

```
9U
```

GBL Tag Order Enc GBL Data.

Definition at line 175 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_TAG_ORDER_CERTIFICATE

```
#define GBL_TAG_ORDER_CERTIFICATE
```

Value:

```
10U
```

GBL Tag Order Certificate.

Definition at line 177 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_TAG_ORDER_SIGNATURE

```
#define GBL_TAG_ORDER_SIGNATURE
```

Value:

```
11U
```

GBL Tag Order Signature.

Definition at line 179 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_TAG_ORDER_END

```
#define GBL_TAG_ORDER_END
```

Value:

```
12U
```

GBL_TAG_ORDER_END must always be strictly greater than any other tag order.

Definition at line 181 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_TAG_FLAG_SINGLE_OCCURRENCE_ONLY

```
#define GBL_TAG_FLAG_SINGLE_OCCURRENCE_ONLY
```

Value:

```
(1UL << 0)
```

Flags.

Definition at line 184 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GBL_TAG_FLAG_ALWAYS_UNENCRYPTED

```
#define GBL_TAG_FLAG_ALWAYS_UNENCRYPTED
```

Value:

```
(1UL << 1)
```

Flags.

Definition at line 186 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GblTagHeader_t

GBL tag header. Must be the first element in all GBL tags.

Public Attributes

uint32_t	tagId Tag ID.
uint32_t	length Length (in bytes) of the rest of the tag.

Public Attribute Documentation

tagId

```
uint32_t GblTagHeader_t::tagId
```

Tag ID.

Definition at line 193 of file platform/bootloader/parser/gbl/btl_gbl_format.h

length

```
uint32_t GblTagHeader_t::length
```

Length (in bytes) of the rest of the tag.

Definition at line 194 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GblHeader_t

GBL header tag type.

Public Attributes

GblTagHeader_t	header Tag ID and length.
uint32_t	version Version of the GBL spec used in this file.
uint32_t	type Type of GBL.

Public Attribute Documentation

header

```
GblTagHeader_t GblHeader_t::header
```

Tag ID and length.

Definition at line 199 of file platform/bootloader/parser/gbl/btl_gbl_format.h

version

```
uint32_t GblHeader_t::version
```

Version of the GBL spec used in this file.

Definition at line 200 of file platform/bootloader/parser/gbl/btl_gbl_format.h

type

```
uint32_t GblHeader_t::type
```

Type of GBL.

Definition at line 201 of file platform/bootloader/parser/gbl/btl_gbl_format.h

VersionDependency_t

GBL version dependency tag type.

Public Attributes

uint8_t	imageType Type of image (application, bootloader, SE)
uint8_t	statement Encoded dependency statement (ex. appVersion > (0).1.2.3)
uint16_t	reserved Reserved.
uint32_t	version The version number used in the statement (ex. (0).1.2.3)

Public Attribute Documentation

imageType

```
uint8_t VersionDependency_t::imageType
```

Type of image (application, bootloader, SE)

Definition at line `206` of file `platform/bootloader/parser/gbl/btl_gbl_format.h`

statement

```
uint8_t VersionDependency_t::statement
```

Encoded dependency statement (ex. appVersion > (0).1.2.3)

Definition at line `207` of file `platform/bootloader/parser/gbl/btl_gbl_format.h`

reserved

```
uint16_t VersionDependency_t::reserved
```

Reserved.

Definition at line `208` of file `platform/bootloader/parser/gbl/btl_gbl_format.h`

version

```
uint32_t VersionDependency_t::version
```

The version number used in the statement (ex. (0).1.2.3)

Definition at line `209` of file `platform/bootloader/parser/gbl/btl_gbl_format.h`

GblApplication_t

GBL application tag type.

Public Attributes

GblTagHeader_t	header Tag ID and length.
ApplicationData_t	appInfo Information about the application.

Public Attribute Documentation

header

```
GblTagHeader_t GblApplication_t::header
```

Tag ID and length.

Definition at line 214 of file platform/bootloader/parser/gbl/btl_gbl_format.h

appInfo

```
ApplicationData_t GblApplication_t::appInfo
```

Information about the application.

Definition at line 215 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GblBootloader_t

GBL bootloader tag type.

Public Attributes

GblTagHeader_t	header Tag ID and length.
uint32_t	bootloaderVersion Version number of the bootloader.
uint32_t	address Address of the bootloader.
uint8_t *	data Array of data for bootloader upgrade.

Public Attribute Documentation

header

```
GblTagHeader_t GblBootloader_t::header
```

Tag ID and length.

Definition at line `220` of file `platform/bootloader/parser/gbl/btl_gbl_format.h`

bootloaderVersion

```
uint32_t GblBootloader_t::bootloaderVersion
```

Version number of the bootloader.

Definition at line `221` of file `platform/bootloader/parser/gbl/btl_gbl_format.h`

address

```
uint32_t GblBootloader_t::address
```

Address of the bootloader.

Definition at line `222` of file `platform/bootloader/parser/gbl/btl_gbl_format.h`

data

```
uint8_t* GblBootloader_t::data
```

Array of data for bootloader upgrade.

Definition at line 223 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GblSeUpgrade_t

GBL SE upgrade tag type.

Public Attributes

GblTagHeader_t	header Tag ID and length.
uint32_t	blobSize Size of the SE upgrade blob.
uint32_t	version Version of the SE image.
uint8_t *	data Array of data for SE upgrade.

Public Attribute Documentation

header

```
GblTagHeader_t GblSeUpgrade_t::header
```

Tag ID and length.

Definition at line `228` of file `platform/bootloader/parser/gbl/btl_gbl_format.h`

blobSize

```
uint32_t GblSeUpgrade_t::blobSize
```

Size of the SE upgrade blob.

Definition at line `229` of file `platform/bootloader/parser/gbl/btl_gbl_format.h`

version

```
uint32_t GblSeUpgrade_t::version
```

Version of the SE image.

Definition at line `230` of file `platform/bootloader/parser/gbl/btl_gbl_format.h`

data

```
uint8_t* GblSeUpgrade_t::data
```

Array of data for SE upgrade.

Definition at line 231 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GblMetadata_t

GBL metadata tag type.

Public Attributes

GblTagHeader_t	header Tag ID and length.
uint8_t *	metaData Array of metadata.

Public Attribute Documentation

header

```
GblTagHeader_t GblMetadata_t::header
```

Tag ID and length.

Definition at line 236 of file platform/bootloader/parser/gbl/btl_gbl_format.h

metaData

```
uint8_t* GblMetadata_t::metaData
```

Array of metadata.

Definition at line 237 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GblProg_t

GBL flash program tag type.

Public Attributes

GblTagHeader_t	header Tag ID and length.
uint32_t	flashStartAddress Address to start flashing.
uint8_t *	data Array of data to flash.

Public Attribute Documentation

header

```
GblTagHeader_t GblProg_t::header
```

Tag ID and length.

Definition at line 242 of file platform/bootloader/parser/gbl/btl_gbl_format.h

flashStartAddress

```
uint32_t GblProg_t::flashStartAddress
```

Address to start flashing.

Definition at line 243 of file platform/bootloader/parser/gbl/btl_gbl_format.h

data

```
uint8_t* GblProg_t::data
```

Array of data to flash.

Definition at line 244 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GblEnd_t

GBL end tag type.

Public Attributes

GblTagHeader_t	header Tag ID and length.
uint32_t	gblCrc CRC32 of the entire GBL file.

Public Attribute Documentation

header

```
GblTagHeader_t GblEnd_t::header
```

Tag ID and length.

Definition at line 249 of file platform/bootloader/parser/gbl/btl_gbl_format.h

gblCrc

```
uint32_t GblEnd_t::gblCrc
```

CRC32 of the entire GBL file.

Definition at line 250 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GblEncryptionInitAesCcm_t

GBL encryption init tag type. Used with AES-CCM encryption.

Public Attributes

GblTagHeader_t	header Tag ID and length.
uint32_t	msgLen Length of the cipher text in bytes.
uint8_t	nonce Random nonce used for AES-CTR in this message.

Public Attribute Documentation

header

```
GblTagHeader_t GblEncryptionInitAesCcm_t::header
```

Tag ID and length.

Definition at line 255 of file platform/bootloader/parser/gbl/btl_gbl_format.h

msgLen

```
uint32_t GblEncryptionInitAesCcm_t::msgLen
```

Length of the cipher text in bytes.

Definition at line 256 of file platform/bootloader/parser/gbl/btl_gbl_format.h

nonce

```
uint8_t GblEncryptionInitAesCcm_t::nonce[12]
```

Random nonce used for AES-CTR in this message.

Definition at line 257 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GblEncryptionData_t

GBL encryption data tag type.

Public Attributes

GblTagHeader_t	header Tag ID and length.
uint8_t *	encryptedGblData Encrypted data.

Public Attribute Documentation

header

```
GblTagHeader_t GblEncryptionData_t::header
```

Tag ID and length.

Definition at line 263 of file platform/bootloader/parser/gbl/btl_gbl_format.h

encryptedGblData

```
uint8_t* GblEncryptionData_t::encryptedGblData
```

Encrypted data.

After decryption, this data must represent one or more complete unencrypted GBL tags.

Definition at line 264 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GblCertificateEcdsaP256_t

GBL certificate chain for signing.

Public Attributes

GblTagHeader_t	header Tag ID and length.
ApplicationCertificate_t	certificate Certificate used to verify GBL.

Public Attribute Documentation

header

> GblTagHeader_t GblCertificateEcdsaP256_t::header

Tag ID and length.

Definition at line 272 of file platform/bootloader/parser/gbl/btl_gbl_format.h

certificate

> ApplicationCertificate_t GblCertificateEcdsaP256_t::certificate

Certificate used to verify GBL.

Definition at line 273 of file platform/bootloader/parser/gbl/btl_gbl_format.h

GblSignatureEcdsaP256_t

GBL ECDSA secp256r1 signature tag type.

Public Attributes

GblTagHeader_t	header Tag ID and length.
uint8_t	r R-point of ECDSA secp256r1 signature.
uint8_t	s S-point of ECDSA secp256r1 signature.

Public Attribute Documentation

header

```
GblTagHeader_t GblSignatureEcdsaP256_t::header
```

Tag ID and length.

Definition at line 278 of file platform/bootloader/parser/gbl/btl_gbl_format.h

r

```
uint8_t GblSignatureEcdsaP256_t::r[32]
```

R-point of ECDSA secp256r1 signature.

Definition at line 279 of file platform/bootloader/parser/gbl/btl_gbl_format.h

s

```
uint8_t GblSignatureEcdsaP256_t::s[32]
```

S-point of ECDSA secp256r1 signature.

Definition at line 280 of file platform/bootloader/parser/gbl/btl_gbl_format.h

Security

Security

Collection of security components.

Security components provide implementations of cryptographic operations as well as functionality to compute checksums and to read cryptographic keys from manufacturing tokens.

Modules

AES

AES

AES decryption functionality for bootloader.

Modules

AesContext_t

AesCtrContext_t

Functions

void btl_initAesContext(void *ctx)
Initialize AES context.

void btl_setAesKey(void *ctx, const uint8_t *key, unsigned int keySize, bool encryptNotDecrypt)
Set AES key to use for encryption/decryption.

void btl_processAesBlock(void *ctx, uint8_t *inputBlock, uint8_t *outputBlock, bool encryptNotDecrypt)
Process one block of data using AES-ECB.

void btl_initAesCcm(void *ctx, uint8_t flags, uint8_t *nonce, uint32_t counter, const uint8_t *key, unsigned int keySize)
Set up the AES-CTR context structure in CCM mode.

void btl_processAesCtrData(void *ctx, const uint8_t *input, uint8_t *output, size_t length)
Process data using AES-CTR.

Function Documentation

btl_initAesContext

```
void btl_initAesContext (void *ctx)
```

Initialize AES context.

Parameters

N/A	ctx	Context variable of type AesContext_t

Wipes the AES context struct before use.

Definition at line `45` of file `platform/bootloader/security/btl_security_aes.h`

btl_setAesKey

```
void btl_setAesKey (void *ctx, const uint8_t *key, unsigned int keySize, bool encryptNotDecrypt)
```

Set AES key to use for encryption/decryption.

Parameters

N/A	ctx	Context variable of type AesContext_t
N/A	key	Pointer to the AES key
N/A	keySize	Size of the key in bits. Can be 128 or 256.
N/A	encryptNotDecrypt	True if using this context for encryption, false if using for decryption.

Initializes the AES context struct with the key to use.

Definition at line 58 of file platform/bootloader/security/btl_security_aes.h

btl_processAesBlock

```
void btl_processAesBlock (void *ctx, uint8_t *inputBlock, uint8_t *outputBlock, bool encryptNotDecrypt)
```

Process one block of data using AES-ECB.

Parameters

N/A	ctx	Context variable of type AesContext_t
N/A	inputBlock	128-bit (16 byte) buffer/block of data to be en/decrypted
N/A	outputBlock	128-bit (16 byte) buffer/block of data to put the result of the en/decryption in.
N/A	encryptNotDecrypt	True for encryption, false for decryption

Runs one block of data through the AES algorithm. In-place encryption/ decryption is supported.

Definition at line 76 of file platform/bootloader/security/btl_security_aes.h

btl_initAesCcm

```
void btl_initAesCcm (void *ctx, uint8_t flags, uint8_t *nonce, uint32_t counter, const uint8_t *key, unsigned int keySize)
```

Set up the AES-CTR context structure in CCM mode.

Parameters

N/A	ctx	Context variable of type AesCtrContext_t
N/A	flags	CCM flags
N/A	nonce	12-byte nonce specific to this transmission
N/A	counter	3-byte running block counter
N/A	key	Pointer to the AES key
N/A	keySize	Size of the key in bits. Can be 128 or 256.

Initializes an AES-CTR context struct with parameters used in AES-CCM mode. Data can then be en/decrypted using btl_processAesCtrData.

Definition at line 94 of file platform/bootloader/security/btl_security_aes.h

btl_processAesCtrData

```
void btl_processAesCtrData (void *ctx, const uint8_t *input, uint8_t *output, size_t length)
```

Process data using AES-CTR.

Parameters

N/A	ctx	Context variable of type AesCtrContext_t
N/A	input	Raw data to en/decrypt
N/A	output	Output buffer to put en/decrypted data
N/A	length	Size (in bytes) of the input/output buffers

Runs data for encryption or decryption (which uses the same function) through the AES-CTR algorithm. In-place encryption/decryption is supported.

Definition at line 112 of file platform/bootloader/security/btl_security_aes.h

AesContext_t

Context variable type for AES-ECB.

Public Attributes

mbedtls_aes_context	aesContext mbedTLS AES context

Public Attribute Documentation

aesContext

```
mbedtls_aes_context AesContext_t::aesContext
```

mbedTLS AES context

Definition at line 50 of file platform/bootloader/security/btl_security_types.h

AesCtrContext_t

Context variable type for AES-CTR (and AES-CCM)

Public Attributes

mbedtls_aes_context	aesContext mbedTLS AES context
size_t	offsetInBlock Position in block of last byte en/decrypted.
uint8_t	streamBlock Current CTR encrypted block.
uint8_t	counter Current counter/CCM value.

Public Attribute Documentation

aesContext

```
mbedtls_aes_context AesCtrContext_t::aesContext
```

mbedTLS AES context

Definition at line 59 of file platform/bootloader/security/btl_security_types.h

offsetInBlock

```
size_t AesCtrContext_t::offsetInBlock
```

Position in block of last byte en/decrypted.

Definition at line 61 of file platform/bootloader/security/btl_security_types.h

streamBlock

```
uint8_t AesCtrContext_t::streamBlock[16]
```

Current CTR encrypted block.

Definition at line 66 of file platform/bootloader/security/btl_security_types.h

counter

```
uint8_t AesCtrContext_t::counter[16]
```

Current counter/CCM value.

Definition at line 68 of file platform/bootloader/security/btl_security_types.h

CRC16

CRC16

CRC16 functionality for the bootloader.

Functions

uint16_t btl_crc16(const uint8_t newByte, uint16_t prevResult)
Calculate CRC16 on input.

uint16_t btl_crc16Stream(const uint8_t *buffer, size_t length, uint16_t prevResult)
Calculate CRC16 on input stream.

Macros

#define BTL_CRC16_START 0xFFFFU
CRC16 start value.

Function Documentation

btl_crc16

```
uint16_t btl_crc16 (const uint8_t newByte, uint16_t prevResult)
```

Calculate CRC16 on input.

Parameters

N/A	newByte	Byte to append to CRC16 calculation
N/A	prevResult	Previous output from CRC algorithm. BTL_CRC16_START when starting a new calculation.

Returns

- Result of the CRC16 operation

Definition at line 45 of file platform/bootloader/security/btl_crc16.h

btl_crc16Stream

```
uint16_t btl_crc16Stream (const uint8_t *buffer, size_t length, uint16_t prevResult)
```

Calculate CRC16 on input stream.

Parameters

N/A	buffer	Buffer containing bytes to append to CRC16 calculation
N/A	length	Size of the buffer in bytes
N/A	prevResult	Previous output from CRC algorithm. BTL_CRC16_START when starting a new calculation.

Returns

- Result of the CRC16 operation

Definition at line 56 of file platform/bootloader/security/btl_crc16.h

Macro Definition Documentation

BTL_CRC16_START

```
#define BTL_CRC16_START
```

Value:

```
0xFFFFU
```

CRC16 start value.

Definition at line 35 of file platform/bootloader/security/btl_crc16.h

CRC32

CRC32

CRC32 functionality for the bootloader.

Functions

uint32_t btl_crc32Stream(const uint8_t *buffer, size_t length, uint32_t prevResult)
Calculate CRC32 on input buffer.

Macros

#define BTL_CRC32_START 0xFFFFFFFFUL
CRC32 start value.

#define BTL_CRC32_END 0xDEBB20E3UL
CRC32 end value.

Function Documentation

btl_crc32Stream

```
uint32_t btl_crc32Stream (const uint8_t *buffer, size_t length, uint32_t prevResult)
```

Calculate CRC32 on input buffer.

Parameters

N/A	buffer	Buffer containing bytes to append to CRC32 calculation
N/A	length	Size of the buffer in bytes
N/A	prevResult	Previous output from the CRC algorithm. Polynomial if starting a new calculation

Returns

- Result of the CRC32 operation

Definition at line 48 of file platform/bootloader/security/btl_crc32.h

Macro Definition Documentation

BTL_CRC32_START

```
#define BTL_CRC32_START
```

Value:

```
0xFFFFFFFFUL
```

CRC32 start value.

Definition at line 35 of file platform/bootloader/security/btl_crc32.h

BTL_CRC32_END

```
#define BTL_CRC32_END
```

Value:

```
0xDEBB20E3UL
```

CRC32 end value.

Definition at line 37 of file platform/bootloader/security/btl_crc32.h

Decryption

Decryption

Generic decryption functionality for bootloader.

Modules

DecryptContext_t

AuthContext_t

DecryptContext_t

Generic decryption context.

Public Attributes

AesCtrContext_t	aesCtr Context for AES-CTR-128 decryption.

Public Attribute Documentation

aesCtr

> AesCtrContext_t DecryptContext_t::aesCtr

Context for AES-CTR-128 decryption.

Definition at line 95 of file platform/bootloader/security/btl_security_types.h

AuthContext_t

Generic authentication context.

Public Attributes

Sha256Context_t	sha256 Context for SHA-256 digest.

Public Attribute Documentation

sha256

> Sha256Context_t AuthContext_t::sha256

Context for SHA-256 digest.

Definition at line 100 of file platform/bootloader/security/btl_security_types.h

ECDSA

ECDSA

ECDSA signing functionality for the bootloader.

Modules

ECC Library

Functions

int32_t btl_verifyEcdsaP256r1(const uint8_t *sha256, const uint8_t *signatureR, const uint8_t *signatureS, const uint8_t *keyX, const uint8_t *keyY)
Verify an ECDSA signature of a SHA256-hash using secp256r1.

Macros

#define BTL_SECURITY_ECDSA_SHA256_LENGTH 32
Number of bytes of data to verify the signature against.

#define BTL_SECURITY_ECDSA_POINT_LENGTH 32
Number of bytes in the EC points that the signature consists of.

Function Documentation

btl_verifyEcdsaP256r1

> int32_t btl_verifyEcdsaP256r1 (const uint8_t *sha256, const uint8_t *signatureR, const uint8_t *signatureS, const uint8_t *keyX, const uint8_t *keyY)

Verify an ECDSA signature of a SHA256-hash using secp256r1.

Parameters

N/A	sha256	The hash of the data which is authenticated
N/A	signatureR	Byte array (MSB first) of R-point of the ECDSA signature
N/A	signatureS	Byte array (MSB first) of S-point of the ECDSA signature
N/A	keyX	Pointer to the X coordinate of the ECDSA public key
N/A	keyY	Pointer to the Y coordinate of the ECDSA public key

Verifies the authenticity of data by checking the ECDSA signature of the data's SHA256-hash. This function is only for use with the secp256r1 curve. The public key which the signature is validated against will be retrieved from the respective tokens in the lockbits-page.

Returns

- BOOTLOADER_OK if signature is valid, else error code in BOOTLOADER_ERROR_SECURITY_BASE range.

Definition at line 56 of file platform/bootloader/security/btl_security_ecdsa.h

Macro Definition Documentation

BTL_SECURITY_ECDSA_SHA256_LENGTH

```
#define BTL_SECURITY_ECDSA_SHA256_LENGTH
```

Value:

```
32
```

Number of bytes of data to verify the signature against.

Definition at line 36 of file platform/bootloader/security/btl_security_ecdsa.h

BTL_SECURITY_ECDSA_POINT_LENGTH

```
#define BTL_SECURITY_ECDSA_POINT_LENGTH
```

Value:

```
32
```

Number of bytes in the EC points that the signature consists of.

Definition at line 38 of file platform/bootloader/security/btl_security_ecdsa.h

ECC Library

ECC Library

Elliptic Curve Cryptography Library.

The Elliptic Curve Cryptography (ECC) API includes ECDSA verification on one of the elliptic curves recommended by NIST in csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.pdf :

- secp256r1: NIST/SECG X9.62 curve over a 256 bit prime field

Modules

ECC_Point_t

ECC_EcdsaSignature_t

Typedefs

typedef uint32_t ECC_BigInt_t[ECC_BIGINT_SIZE_IN_32BIT_WORDS]
ECC big integer type.

Functions

int32_t ECC_ECDSA_VerifySignatureP256(CRYPTO_TypeDef *crypto, const uint8_t *msgDigest, int msgDigestLen, const ECC_Point_t *publicKey, ECC_EcdsaSignature_t *signature)
Functions **********.

void ECC_HexToBigInt(ECC_BigInt_t bigint, const char *hex)
Convert a large integer from a hexadecimal string to the ECC_BigInt_t format.

void ECC_BigIntToHex(char *hex, ECC_BigInt_t bigint)
Convert a large integer from the ECC_BigInt_t to a hexadecimal string format.

void ECC_ByteArrayToBigInt(ECC_BigInt_t bigint, const uint8_t *bytearray)
Convert a big integer from a byte array to the ECC_BigInt_t format.

void ECC_BigIntToByteArray(uint8_t *bytearray, ECC_BigInt_t bigint)
Convert a large integer from the ECC_BigInt_t to the byte array format.

void ECC_UnsignedIntToBigInt(ECC_BigInt_t bigint, const uint32_t value)
Convert an integer from uint32_t to ECC_BigInt_t format.

Macros

#define ECC_BIGINT_SIZE_IN_BITS (256)
TYPEDEFS **********.

#define ECC_BIGINT_SIZE_IN_BYTES (ECC_BIGINT_SIZE_IN_BITS / 8)
ECC big integer size in bytes.

#define ECC_BIGINT_SIZE_IN_32BIT_WORDS undefined
ECC big integer size in words.

Typedef Documentation

ECC_BigInt_t

```
typedef uint32_t ECC_BigInt_t[ECC_BIGINT_SIZE_IN_32BIT_WORDS] [ECC_BIGINT_SIZE_IN_32BIT_WORDS]
```

ECC big integer type.

Definition at line 59 of file platform/bootloader/security/ecc/ecc.h

Function Documentation

ECC_ECDSA_VerifySignatureP256

```
int32_t ECC_ECDSA_VerifySignatureP256 (CRYPTO_TypeDef *crypto, const uint8_t *msgDigest, int msgDigestLen, const ECC_Point_t *publicKey, ECC_EcdsaSignature_t *signature)
```

Functions **********.

Parameters

[in]	crypto	Pointer to CRYPTO peripheral instance
[in]	msgDigest	The message digest associated with the signature.
[in]	msgDigestLen	The length of the message digest.
[in]	publicKey	Public key of entity that generated signature.
[out]	signature	The signature to verify.

Verify an ECDSA signature.

TBW

Returns

- Error code.

Definition at line 102 of file platform/bootloader/security/ecc/ecc.h

ECC_HexToBigInt

```
void ECC_HexToBigInt (ECC_BigInt_t bigint, const char *hex)
```

Convert a large integer from a hexadecimal string to the ECC_BigInt_t format.

Parameters

[out]	bigint	Pointer to the location where to store the result.
[in]	hex	The hex representation of the large integer to convert.

Convert a large integer from a hexadecimal string representation to a ECC_BigInt_t representation.

Definition at line 120 of file platform/bootloader/security/ecc/ecc.h

ECC_BigIntToHex

```
void ECC_BigIntToHex (char *hex, ECC_BigInt_t bigint)
```

Convert a large integer from the ECC_BigInt_t to a hexadecimal string format.

Parameters

[out]	hex	Buffer where to store the hexadecimal result.
[in]	bigint	The ECC_BigInt_t representation of the large integer to convert.

Convert a large integer from a ECC_BigInt_t representation to a hexadecimal string representation.

Definition at line `134` of file `platform/bootloader/security/ecc/ecc.h`

ECC_ByteArrayToBigInt

```
void ECC_ByteArrayToBigInt (ECC_BigInt_t bigint, const uint8_t *bytearray)
```

Convert a big integer from a byte array to the ECC_BigInt_t format.

Parameters

[out]	bigint	Pointer to the location where to store the result.
[in]	bytearray	The byte array representation of the large integer to convert.

Convert a large integer from a byte array representation to a ECC_BigInt_t representation.

Definition at line `148` of file `platform/bootloader/security/ecc/ecc.h`

ECC_BigIntToByteArray

```
void ECC_BigIntToByteArray (uint8_t *bytearray, ECC_BigInt_t bigint)
```

Convert a large integer from the ECC_BigInt_t to the byte array format.

Parameters

[out]	bytearray	Buffer to store the resulting byte array.
[in]	bigint	The ECC_BigInt_t representation of the large integer to convert.

Convert a large integer from a ECC_BigInt_t representation to a byte array representation. Caution: byte array must be big enough to contain the result.

Definition at line `163` of file `platform/bootloader/security/ecc/ecc.h`

ECC_UnsignedIntToBigInt

```
void ECC_UnsignedIntToBigInt (ECC_BigInt_t bigint, const uint32_t value)
```

Convert an integer from uint32_t to ECC_BigInt_t format.

Parameters

[out]	bigint	Pointer to the location to store the result.

[in]	value	The value to convert.

Convert a integer from an uint32_t representation to a ECC_BigInt_t representation.

Definition at line 176 of file platform/bootloader/security/ecc/ecc.h

Macro Definition Documentation

ECC_BIGINT_SIZE_IN_BITS

```
#define ECC_BIGINT_SIZE_IN_BITS
```

Value:

```
(256)
```

TYPEDEFS **********.

ECC big integer size in bits.

Definition at line 49 of file platform/bootloader/security/ecc/ecc.h

ECC_BIGINT_SIZE_IN_BYTES

```
#define ECC_BIGINT_SIZE_IN_BYTES
```

Value:

```
(ECC_BIGINT_SIZE_IN_BITS / 8)
```

ECC big integer size in bytes.

Definition at line 52 of file platform/bootloader/security/ecc/ecc.h

ECC_BIGINT_SIZE_IN_32BIT_WORDS

```
#define ECC_BIGINT_SIZE_IN_32BIT_WORDS
```

Value:

```
00                  (ECC_BIGINT_SIZE_IN_BYTES \
00                  / sizeof(uint32_t))
```

ECC big integer size in words.

Definition at line 55 of file platform/bootloader/security/ecc/ecc.h

ECC_Point_t

Elliptic curve point structure.

Public Attributes

ECC_BigInt_t	X x coordinate of point.
ECC_BigInt_t	Y y coordinate of point.

Public Attribute Documentation

X

```
ECC_BigInt_t ECC_Point_t::X
```

x coordinate of point.

Definition at line 63 of file platform/bootloader/security/ecc/ecc.h

Y

```
ECC_BigInt_t ECC_Point_t::Y
```

y coordinate of point.

Definition at line 64 of file platform/bootloader/security/ecc/ecc.h

ECC_EcdsaSignature_t

ECDSA signature as defined in FIPS PUB 186-3, Digital Signature Standard (DSS).

Public Attributes

ECC_BigInt_t	r The r component of the signature.
ECC_BigInt_t	s The s component of the signature.

Public Attribute Documentation

r

```
ECC_BigInt_t ECC_EcdsaSignature_t::r
```

The r component of the signature.

Definition at line 70 of file platform/bootloader/security/ecc/ecc.h

s

```
ECC_BigInt_t ECC_EcdsaSignature_t::s
```

The s component of the signature.

Definition at line 71 of file platform/bootloader/security/ecc/ecc.h

SHA256

SHA256

SHA-256 Cryptography Library.

This file includes an alternative implementation of the standard mbed TLS SHA using hardware accelerator incorporated in MCU devices from Silicon Labs.

Modules

btl_sha256_context

Enumerations

enum SHA_Type_t {
  SHA256
}
Type variable for SHA-256 Cryptography.

Functions

void btl_sha256_init(btl_sha256_context *ctx)
Initialize SHA-256 context.

int btl_sha256_starts_ret(btl_sha256_context *ctx, int is224)
Set up SHA-256 context.

int btl_sha256_update_ret(btl_sha256_context *ctx, const unsigned char *input, size_t ilen)
SHA-256 process buffer.

int btl_sha256_finish_ret(btl_sha256_context *ctx, unsigned char output[32])
SHA-256 final digest.

int sha_x_process(SHA_Type_t algo, uint8_t *state_in, const unsigned char *blockdata, uint8_t *state_out, uint32_t num_blocks)
Process (a) block(s) of data to be hashed.

int sha_x_update(SHA_Type_t algo, const unsigned char *data, size_t data_len, uint8_t *state, unsigned char *buffer, uint32_t *counter)
Process an arbitrary number of bytes to be hashed.

int sha_x_finish(SHA_Type_t algo, uint8_t *state, uint8_t *buffer, uint32_t *counter, uint8_t *output)
Process an arbitrary number of bytes to be hashed.

Enumeration Documentation

SHA_Type_t

> SHA_Type_t

Type variable for SHA-256 Cryptography.

Enumerator	
SHA256	SHA-256 type.

Definition at line 44 of file platform/bootloader/security/sha/btl_sha256.h

Function Documentation

btl_sha256_init

```
void btl_sha256_init (btl_sha256_context *ctx)
```

Initialize SHA-256 context.

Parameters

N/A	ctx	SHA-256 context to be initialized

Definition at line 55 of file platform/bootloader/security/sha/btl_sha256.h

btl_sha256_starts_ret

```
int btl_sha256_starts_ret (btl_sha256_context *ctx, int is224)
```

Set up SHA-256 context.

Parameters

N/A	ctx	context to be initialized
N/A	is224	0 = use SHA256, 1 = use SHA224

Returns

- 0 if successful

Definition at line 66 of file platform/bootloader/security/sha/btl_sha256.h

btl_sha256_update_ret

```
int btl_sha256_update_ret (btl_sha256_context *ctx, const unsigned char *input, size_t ilen)
```

SHA-256 process buffer.

Parameters

N/A	ctx	SHA-256 context
N/A	input	buffer holding the data
N/A	ilen	length of the input data

Returns

- 0 if successful

Definition at line 78 of file platform/bootloader/security/sha/btl_sha256.h

btl_sha256_finish_ret

```
int btl_sha256_finish_ret (btl_sha256_context *ctx, unsigned char output[32])
```

SHA-256 final digest.

Parameters

N/A	ctx	SHA-256 context
N/A	output	SHA-224/256 checksum result

Returns

- `0` if successful

Definition at line `89` of file `platform/bootloader/security/sha/btl_sha256.h`

sha_x_process

```
int sha_x_process (SHA_Type_t algo, uint8_t *state_in, const unsigned char *blockdata, uint8_t *state_out, uint32_t num_blocks)
```

Process (a) block(s) of data to be hashed.

Parameters

N/A	algo	Which hashing algorithm to use
[in]	state_in	Previous state of the hashing algorithm
[in]	blockdata	Pointer to the block(s) of data
[out]	state_out	Pointer to block of memory to store state
N/A	num_blocks	Number of SHA blocks in data block

Note

- Watch the blocksize! Blocks are 64 bytes for SHA-1 through SHA-256, and 128 bytes for SHA-384 through SHA-512.
- Watch the state size! State size is half the block size.

Returns

- Zero on success. Negative error code on failure.

Definition at line `105` of file `platform/bootloader/security/sha/btl_sha256.h`

sha_x_update

```
int sha_x_update (SHA_Type_t algo, const unsigned char *data, size_t data_len, uint8_t *state, unsigned char *buffer, uint32_t *counter)
```

Process an arbitrary number of bytes to be hashed.

Parameters

N/A	algo	Which hashing algorithm to use
[inout]	data	Pointer to the hashing algorithm's state buffer
[in]	data_len	Pointer to the data to add to the hash

[inout]	state	Pointer to a block buffer to retrieve/store a partial block in between calls to this function. Needs to have a size equal to the block size.
[inout]	buffer	Counter variable keeping track of the amount of bytes hashed, to later be used for hash finalization. For first use, initialize with zeroes.
[in]	counter	Length to data to add to hash

Note

- Watch the blocksize! Blocks are 64 bytes for SHA-1 through SHA-256, and 128 bytes for SHA-384 through SHA-512.
- Watch the state size! State size is half the block size.
- Watch the counter size! Counter is 64 bytes for SHA-1 through SHA-256, and 128 bytes for SHA-384 through SHA-512.

Returns

- Zero on success. Negative error code on failure.

Definition at line `132` of file `platform/bootloader/security/sha/btl_sha256.h`

sha_x_finish

```
int sha_x_finish (SHA_Type_t algo, uint8_t *state, uint8_t *buffer, uint32_t *counter, uint8_t *output)
```

Process an arbitrary number of bytes to be hashed.

Parameters

N/A	algo	Which hashing algorithm to use
[in]	state	Pointer to the hashing algorithm's state buffer
[in]	buffer	Pointer to a block buffer to retrieve/store a partial block in between calls to this function. Needs to have a size equal to the block size.
[inout]	counter	Counter variable keeping track of the amount of bytes hashed, to later be used for hash finalization. For first use, initialize with zeroes.
[out]	output	Pointer to the destination of the hash.

Note

- Watch the blocksize! Blocks are 64 bytes for SHA-1 through SHA-256, and 128 bytes for SHA-384 through SHA-512.
- Watch the state size! State size is half the block size.
- Watch the counter size! Counter is 64 bytes for SHA-1 through SHA-256, and 128 bytes for SHA-384 through SHA-512.

Returns

- Zero on success. Negative error code on failure.

Definition at line `159` of file `platform/bootloader/security/sha/btl_sha256.h`

btl_sha256_context

Context Variable type for SHA-256 Cryptography.

Public Attributes

uint32_t	total number of bytes processed
uint32_t	state intermediate digest state
unsigned char	buffer data block being processed

Public Attribute Documentation

total

```
uint32_t btl_sha256_context::total[2]
```

number of bytes processed

Definition at line 37 of file platform/bootloader/security/sha/btl_sha256.h

state

```
uint32_t btl_sha256_context::state[8]
```

intermediate digest state

Definition at line 38 of file platform/bootloader/security/sha/btl_sha256.h

buffer

```
unsigned char btl_sha256_context::buffer[64]
```

data block being processed

Definition at line 39 of file platform/bootloader/security/sha/btl_sha256.h

SHA_256

SHA_256

SHA-256 digest functionality for the bootloader.

Modules

Sha256Context_t

Functions

void	btl_initSha256(void *ctx) Initialize SHA256 context variable.
void	btl_updateSha256(void *ctx, const void *data, size_t length) Run data through the SHA256 hashing function.
void	btl_finalizeSha256(void *ctx) Finalize the SHA256 calculation.
int32_t	btl_verifySha256(void *ctx, const void *sha256) Compare the SHA256 from the context variable to a known value.

Macros

#define	BTL_SECURITY_SHA256_DIGEST_LENGTH 32 Number of bytes in a SHA-256 digest.

Function Documentation

btl_initSha256

```
void btl_initSha256 (void *ctx)
```

Initialize SHA256 context variable.

Parameters

N/A	ctx	Pointer to the SHA256 context variable to be initialized

Wipes out the SHA256 context variable and sets it up for re-use.

Definition at line 45 of file platform/bootloader/security/btl_security_sha256.h

btl_updateSha256

```
void btl_updateSha256 (void *ctx, const void *data, size_t length)
```

Run data through the SHA256 hashing function.

Parameters

N/A	ctx	Pointer to the SHA256 context variable
N/A	data	Pointer to an array of binary data to add to the SHA256 calculation in progress
N/A	length	Length of the byte array with data.

Definition at line 56 of file platform/bootloader/security/btl_security_sha256.h

btl_finalizeSha256

```
void btl_finalizeSha256 (void *ctx)
```

Finalize the SHA256 calculation.

Parameters

N/A	ctx	Pointer to the SHA256 context variable to be initialized

Finalizes the running SHA256 calculation. After finalization, the SHA value in the context variable will be valid and no more data can be added.

Definition at line 66 of file platform/bootloader/security/btl_security_sha256.h

btl_verifySha256

```
int32_t btl_verifySha256 (void *ctx, const void *sha256)
```

Compare the SHA256 from the context variable to a known value.

Parameters

N/A	ctx	Pointer to the SHA256 context variable to be initialized
N/A	sha256	Byte array containing SHA256 value to compare to

Returns

- BOOTLOADER_OK if both hash values are equal, else error code from BOOTLOADER_ERROR_SECURITY_BASE range.

Convenience function to compare a known SHA256 hash against the result of a calculation.

Definition at line 79 of file platform/bootloader/security/btl_security_sha256.h

Macro Definition Documentation

BTL_SECURITY_SHA256_DIGEST_LENGTH

```
#define BTL_SECURITY_SHA256_DIGEST_LENGTH
```

Value:

```
32
```

Number of bytes in a SHA-256 digest.

Definition at line 36 of file platform/bootloader/security/btl_security_sha256.h

Sha256Context_t

Context type for SHA algorithm.

Public Attributes

btl_sha256_context	shaContext mbedTLS SHA256 context struct
uint8_t	sha resulting SHA hash

Public Attribute Documentation

shaContext

```
btl_sha256_context Sha256Context_t::shaContext
```

mbedTLS SHA256 context struct

Definition at line 80 of file platform/bootloader/security/btl_security_types.h

sha

```
uint8_t Sha256Context_t::sha[32]
```

resulting SHA hash

Definition at line 81 of file platform/bootloader/security/btl_security_types.h

Tokens

Tokens

Manufacturing token handling for the bootloader.

Functions

const uint8_t *	btl_getSignedBootloaderKeyXPtr(void) Get the X component of the ECDSA secp256r1 public key.
const uint8_t *	btl_getSignedBootloaderKeyYPtr(void) Get the Y component of the ECDSA secp256r1 public key.
const uint8_t *	btl_getImageFileEncryptionKeyPtr(void) Get the AES-CCM encryption key.

Function Documentation

btl_getSignedBootloaderKeyXPtr

```
const uint8_t * btl_getSignedBootloaderKeyXPtr (void)
```

Get the X component of the ECDSA secp256r1 public key.

Parameters

N/A		

Returns

- Pointer to X component of the public key

Definition at line `68` of file `platform/bootloader/security/btl_security_tokens.h`

btl_getSignedBootloaderKeyYPtr

```
const uint8_t * btl_getSignedBootloaderKeyYPtr (void)
```

Get the Y component of the ECDSA secp256r1 public key.

Parameters

N/A		

Returns

- Pointer to Y component of the public key

Definition at line `75` of file `platform/bootloader/security/btl_security_tokens.h`

btl_getImageFileEncryptionKeyPtr

```
const uint8_t * btl_getImageFileEncryptionKeyPtr (void)
```

Get the AES-CCM encryption key.

Parameters

N/A		

Returns

- Pointer to the AES-CCM key

Definition at line `82` of file `platform/bootloader/security/btl_security_tokens.h`

Storage

Storage

Storage component.

This component provides the bootloader with multiple storage options. All storage implementations have to provide a slot-based API to access image files to be bootloaded.

Some storage implementations also support a raw storage API to access the underlying storage medium. This can be used by applications to store other data in parts of the storage medium that are not used for bootloading.

Storage Implementations

Modules

BootloaderStorageLayout_t

Flash Using JEDEC SFDP Standard

Internal Flash

SPI Flash

Bootload Info

SPI Flash Configurations

SPI Flash Configurations using SFDP

Functions

int32_t storage_init(void)
Initialize the storage component.

int32_t storage_main(void)
Main function for storage component.

int32_t storage_shutdown(void)
Shutdown storage component.

void storage_getInfo(BootloaderStorageInformation_t *info)
Get information about the storage component running on the device.

int32_t storage_getSlotInfo(uint32_t slotId, BootloaderStorageSlot_t *slot)
Get information about the layout of a storage slot.

int32_t storage_getSlotMetadata(BootloaderParserContext_t *context, ApplicationData_t *appInfo, uint32_t *bootloaderVersion)
Get information about the contents of a storage slot.

int32_t storage_initParseSlot(uint32_t slotId, BootloaderParserContext_t *context, size_t contextSize)
Initialize the context variable for checking a slot and trying to parse the image contained in it.

int32_t	storage_verifySlot(BootloaderParserContext_t *context, BootloaderParserCallback_t metadataCallback) Check the given slot for a valid image.
bool	storage_upgradeSeFromSlot(uint32_t slotId) Upgrade SE using image contained in a slot.
bool	storage_bootloadBootloaderFromSlot(uint32_t slotId, uint32_t version) Bootload a bootloader image contained in a slot.
bool	storage_bootloadApplicationFromSlot(uint32_t slotId, uint32_t version) Bootload an image contained in a slot.
int32_t	storage_eraseSlot(uint32_t slotId) Erase the contents of a storage slot including all data and metadata.
int32_t	storage_readSlot(uint32_t slotId, uint32_t offset, uint8_t *buffer, size_t numBytes) Read a number of words from a storage slot.
int32_t	storage_writeSlot(uint32_t slotId, uint32_t offset, uint8_t *data, size_t numBytes) Write a number of words to a storage slot.
int32_t	storage_readRaw(uint32_t address, uint8_t *buffer, size_t numBytes) Read number of words from raw storage.
int32_t	storage_writeRaw(uint32_t address, uint8_t *data, size_t numBytes) Write a number of words to raw storage.
int32_t	storage_getDMAchannel(void) Get allocated DMA channel for MSC write.
int32_t	storage_eraseRaw(uint32_t address, size_t length) Erase the raw storage.
bool	storage_isBusy(void) Poll the storage implementation and check whether it is busy.
uint32_t	storage_getSpiUsartPPUSATD(uint32_t *ppusatdNr) Get PPUSATD word of the (E)USART in use.

Macros

#define	BOOTLOADER_STORAGE_FUNCTIONS_VERSION 0×00000100 Version number for bootloader storage functions struct.

Function Documentation

storage_init

```
int32_t storage_init (void)
```

Initialize the storage component.

Parameters

N/A		

Returns

- Error code indicating success or failure.

Definition at line `68` of file `platform/bootloader/storage/btl_storage.h`

storage_main

```
int32_t storage_main (void)
```

Main function for storage component.

Parameters

N/A		

Returns

- BOOTLOADER_OK on success, else error code

Definition at line `75` of file `platform/bootloader/storage/btl_storage.h`

storage_shutdown

```
int32_t storage_shutdown (void)
```

Shutdown storage component.

Parameters

N/A		

Returns

- Error code indicating success or failure.

Definition at line `84` of file `platform/bootloader/storage/btl_storage.h`

storage_getInfo

```
void storage_getInfo (BootloaderStorageInformation_t *info)
```

Get information about the storage component running on the device.

Parameters

[out]	info	Pointer to BootloaderStorageInformation_t struct

Definition at line `91` of file `platform/bootloader/storage/btl_storage.h`

storage_getSlotInfo

```
int32_t storage_getSlotInfo (uint32_t slotId, BootloaderStorageSlot_t *slot)
```

Get information about the layout of a storage slot.

Parameters

[in]	slotId	Slot ID to get info about
[out]	slot	Pointer to BootloaderStorageSlot_t struct

Returns

BOOTLOADER_OK on success, else error code in BOOTLOADER_ERROR_STORAGE_BASE range

Definition at line 102 of file platform/bootloader/storage/btl_storage.h

storage_getSlotMetadata

```
int32_t storage_getSlotMetadata (BootloaderParserContext_t *context, ApplicationData_t *appInfo, uint32_t *bootloaderVersion)
```

Get information about the contents of a storage slot.

Parameters

[in]	context	Parsing context. Should be allocated by the application and initialized by calling storage_initParseSlot before calling this function.
[out]	appInfo	Pointer to ApplicationData_t struct
[out]	bootloaderVersion	Pointer to an unsigned integer representing the bootloader version number.

Note

- storage_initParseSlot**must** be called before calling this function to initialize the context.
- If the slot does not contain an application or a bootloader, the the corresponding values are set to zero.

Returns

- BOOTLOADER_OK on success

Definition at line 123 of file platform/bootloader/storage/btl_storage.h

storage_initParseSlot

```
int32_t storage_initParseSlot (uint32_t slotId, BootloaderParserContext_t *context, size_t contextSize)
```

Initialize the context variable for checking a slot and trying to parse the image contained in it.

Parameters

[in]	slotId	Slot to check for valid image
[in]	context	Pointer to BootloaderParserContext_t struct
[in]	contextSize	Length of the context struct

Returns

- BOOTLOADER_OK on success, else error code in BOOTLOADER_ERROR_PARSE_BASE range

Definition at line 138 of file platform/bootloader/storage/btl_storage.h

storage_verifySlot

```
int32_t storage_verifySlot (BootloaderParserContext_t *context, BootloaderParserCallback_t metadataCallback)
```

Check the given slot for a valid image.

Parameters

[in]	context	Pointer to BootloaderImageParsingContext_t struct

[in]	metadataCallback	Functionpointer which will be called with any binary metadata that might be contained within the image.

Call this function continuously until it stops returning. BOOTLOADER_ERROR_PARSE_CONTINUE.

The function returns BOOTLOADER_ERROR_PARSE_SUCCESS if the image in the slot was successfully verified. For detailed information on the parsed image, see imageProperties in the context variable.

Returns

- BOOTLOADER_ERROR_PARSE_CONTINUE if the parsing is not complete, BOOTLOADER_ERROR_PARSE_SUCCESS on success.

Definition at line 159 of file platform/bootloader/storage/btl_storage.h

storage_upgradeSeFromSlot

```
bool storage_upgradeSeFromSlot (uint32_t slotId)
```

Upgrade SE using image contained in a slot.

Parameters

N/A	slotId	Slot ID to bootload from

Note

- This function assumes the image located in the slotId has been verified first.

Returns

- True if the operation succeeded

Definition at line 172 of file platform/bootloader/storage/btl_storage.h

storage_bootloadBootloaderFromSlot

```
bool storage_bootloadBootloaderFromSlot (uint32_t slotId, uint32_t version)
```

Bootload a bootloader image contained in a slot.

Parameters

N/A	slotId	Slot ID to bootload from
N/A	version	Version number of new bootloader

Note

- This function assumes the image located in the slotId has been verified first.

Returns

- True if operation succeeded

Definition at line 185 of file platform/bootloader/storage/btl_storage.h

storage_bootloadApplicationFromSlot

```
bool storage_bootloadApplicationFromSlot (uint32_t slotId, uint32_t version)
```

Bootload an image contained in a slot.

Parameters

N/A	slotId	Slot ID to bootload from
N/A	version	Cached version number of the image contained in the slot (used for downgrade prevention)

Note

- This function assumes the image located in slotId has been verified first.

Returns

- True if the operation succeeded

Definition at line 199 of file platform/bootloader/storage/btl_storage.h

storage_eraseSlot

```
int32_t storage_eraseSlot (uint32_t slotId)
```

Erase the contents of a storage slot including all data and metadata.

Parameters

N/A	slotId	ID of the slot.

Returns

- BOOTLOADER_OK on success, else error code in BOOTLOADER_ERROR_STORAGE_BASE range

Definition at line 209 of file platform/bootloader/storage/btl_storage.h

storage_readSlot

```
int32_t storage_readSlot (uint32_t slotId, uint32_t offset, uint8_t *buffer, size_t numBytes)
```

Read a number of words from a storage slot.

Parameters

N/A	slotId	ID of the slot.
N/A	offset	The offset into the slot in bytes.
N/A	buffer	Pointer to buffer to store read data in.
N/A	numBytes	Number of bytes to read.

Returns

- BOOTLOADER_OK on success, else error code in BOOTLOADER_ERROR_STORAGE_BASE range

Definition at line 222 of file platform/bootloader/storage/btl_storage.h

storage_writeSlot

```
int32_t storage_writeSlot (uint32_t slotId, uint32_t offset, uint8_t *data, size_t numBytes)
```

Write a number of words to a storage slot.

Parameters

N/A	slotId	ID of the slot.
N/A	offset	The offset into the slot in bytes.
N/A	data	Pointer to data to write.
N/A	numBytes	Length of data to write. Must be a multiple of 4.

Returns

- BOOTLOADER_OK on success, else error code in BOOTLOADER_ERROR_STORAGE_BASE range

Definition at line 238 of file platform/bootloader/storage/btl_storage.h

storage_readRaw

```
int32_t storage_readRaw (uint32_t address, uint8_t *buffer, size_t numBytes)
```

Read number of words from raw storage.

Parameters

N/A	address	The raw address of the storage.
N/A	buffer	Pointer to the buffer to store read data in.
N/A	numBytes	Number of bytes to read.

Returns

- BOOTLOADER_OK on success, else error code in BOOTLOADER_ERROR_STORAGE_BASE range

Definition at line 253 of file platform/bootloader/storage/btl_storage.h

storage_writeRaw

```
int32_t storage_writeRaw (uint32_t address, uint8_t *data, size_t numBytes)
```

Write a number of words to raw storage.

Parameters

N/A	address	The raw address of the storage.
N/A	data	Pointer to data to write.
N/A	numBytes	Length of data to write. Must be a multiple of 4.

Returns

- BOOTLOADER_OK on success, else error code in BOOTLOADER_ERROR_STORAGE_BASE range

Definition at line 267 of file platform/bootloader/storage/btl_storage.h

storage_getDMAchannel

```
int32_t storage_getDMAchannel (void)
```

Get allocated DMA channel for MSC write.

Parameters

N/A		

Returns

- A positive number channel. -1 if DMA-based MSC write is not enabled.

Definition at line `277` of file `platform/bootloader/storage/btl_storage.h`

storage_eraseRaw

```
int32_t storage_eraseRaw (uint32_t address, size_t length)
```

Erase the raw storage.

Parameters

N/A	address	Start address of the region to erase
N/A	length	Number of bytes to erase

Note

- Some devices, such as Flash-based storages, have restrictions on the alignment and size of erased regions. The details of the limitations of a particular storage can be found by reading the BootloaderStorageInformation_t struct using storage_getInfo.

Returns

- BOOTLOADER_OK on success, else error code in BOOTLOADER_ERROR_STORAGE_BASE range

Definition at line `293` of file `platform/bootloader/storage/btl_storage.h`

storage_isBusy

```
bool storage_isBusy (void)
```

Poll the storage implementation and check whether it is busy.

Parameters

N/A		

Returns

- True if the storage is busy

Definition at line `300` of file `platform/bootloader/storage/btl_storage.h`

storage_getSpiUsartPPUSATD

```
uint32_t storage_getSpiUsartPPUSATD (uint32_t *ppusatdNr)
```

Get PPUSATD word of the (E)USART in use.

Parameters

N/A	ppusatdNr	PPUSATD register number

Returns

- Word representing the PPUSATD bit field of the (E)USART in use. 0 if not found.

Definition at line 310 of file platform/bootloader/storage/btl_storage.h

Macro Definition Documentation

BOOTLOADER_STORAGE_FUNCTIONS_VERSION

```
#define BOOTLOADER_STORAGE_FUNCTIONS_VERSION
```

Value:

```
0x00000100
```

Version number for bootloader storage functions struct.

Definition at line 49 of file platform/bootloader/storage/btl_storage.h

BootloaderStorageLayout_t

Information about the storage backend.

Public Attributes

BootloaderStorageType_t	storageType Type of storage.
uint32_t	numSlots Number of slots.
BootloaderStorageSlot_t	slot Array of slot information.

Public Attribute Documentation

storageType

```
BootloaderStorageType_t BootloaderStorageLayout_t::storageType
```

Type of storage.

Definition at line 54 of file platform/bootloader/storage/btl_storage.h

numSlots

```
uint32_t BootloaderStorageLayout_t::numSlots
```

Number of slots.

Definition at line 56 of file platform/bootloader/storage/btl_storage.h

slot

```
BootloaderStorageSlot_t BootloaderStorageLayout_t::slot[]
```

Array of slot information.

Definition at line 58 of file platform/bootloader/storage/btl_storage.h

Flash Using JEDEC SFDP Standard

Flash Using JEDEC SFDP Standard

The SPI Flash storage implementation using JEDEC supports all SPI Flash parts that support JEDEC SFDP.

The supported devices will be detected and configured at runtime using the SFDP standard. The driver queries the parameter table present within a reserved section of the flash memory to get the device properties. The driver initializes the flash device in accordance with the values present in the parameter table. Including support for SFDP standard in the bootloader will lead to slightly larger bootloader code sizes

The SPI Flash storage implementation does not support any write protection functionality.

Internal Flash

Internal Flash

The Internal Flash storage implementation utilizes the internal Flash of the device for upgrade image storage.

SPI Flash

SPI Flash

The SPI Flash storage implementation supports a variety of SPI Flash parts including the following:

- Spansion S25FL208K (8Mbit)
- Winbond W25X20BV (2Mbit), W25Q80BV (8Mbit)
- Macronix MX25L2006E (2Mbit), MX25L4006E (4Mbit), MX25L8006E (8Mbit), MX25R8035F (8Mbit low power), MX25L1606E (16Mbit), MX25U1635E (16Mbit 2Volt), MX25R3235F (32Mbit ultra low power) MX25R6435F (64Mbit low power)
- Atmel/Adesto AT25DF041A (4Mbit), AT25DF081A (8Mbit)
- Numonyx/Micron M25P20 (2Mbit), M25P40 (4Mbit), M25P80 (8Mbit), M25P16 (16Mbit)
- ISSI IS25LQ025B (256Kbit), IS25LQ512B (512Kbit), IS25LQ010B (1Mbit), IS25LQ020B (2Mbit), IS25LQ040B (4Mbit)

The subset of supported devices can be configured at compile time using the configuration defines given in SPI Flash Configurations. Including support for multiple devices requires more Flash space in the bootloader.

The SPI Flash storage implementation does not support any write protection functionality.

Bootload Info

Bootload Info

Indicates which firmware update image should be bootloaded next.

This component provides the bootloader with support for storing multiple images and attempting to bootload a prioritized list. The Bootload Information struct is placed at a known location in storage, and points to a list of images to attempt to bootload.

While the Bootload Information list of images to attempt to bootload has a compile-time configurable size, the bootloader is capable of handling lists of images with different sizes, e.g., if a bootloader upgrade changes the slot layout or if a storage device that was initialized on a different device is used.

If only one storage slot is available, the functions available in this API will do nothing and return applicable error codes (BOOTLOADER_OK for storage_getBootloadList and storage_setBootloadList, and BOOTLOADER_ERROR_BOOTLOAD_LIST_FULL for storage_appendBootloadList).

Modules

BootloadInfo_t

Functions

int32_t storage_getBootloadList(int32_t slotIds[], size_t length)
Get list of firmware update images to attempt to bootload.

int32_t storage_setBootloadList(int32_t slotIds[], size_t length)
Set list of firmware update images to attempt to bootload.

int32_t storage_appendBootloadList(int32_t slotId)
Append a storage slot to the list of storage slots to try bootloading from.

Macros

#define BTL_STORAGE_BOOTLOADINFO_MAGIC 0xE18F5239UL
Magic word indicating BootloadInfo_t struct.

#define BTL_STORAGE_BOOTLOADINFO_VERSION 0x00000001UL
Version number for the BootloadInfo_t struct.

#define BTL_STORAGE_BOOTLOAD_LIST_MAX_LENGTH 16UL
Maximum number of items in the BootloadInfo_t::bootloadList.

Function Documentation

storage_getBootloadList

```
int32_t storage_getBootloadList (int32_t slotIds[], size_t length)
```

Get list of firmware update images to attempt to bootload.

Parameters

[out]	slotIds	Pointer to array of integers to fill with slot IDs, or -1 if list position does not contain a valid slot ID
[in]	length	Number of slot IDs to get

Returns

- Error code: BOOTLOADER_OK if list of slots was successfully filled else error code in BOOTLOADER_ERROR_STORAGE_BASE range

Definition at line 89 of file platform/bootloader/storage/bootloadinfo/btl_storage_bootloadinfo.h

storage_setBootloadList

```
int32_t storage_setBootloadList (int32_t slotIds[], size_t length)
```

Set list of firmware update images to attempt to bootload.

Parameters

[in]	slotIds	Pointer to array of slot IDs to set
[in]	length	Number of slot IDs to set

Returns

- Error code: BOOTLOADER_OK if list of slots was successfully filled

Definition at line 99 of file platform/bootloader/storage/bootloadinfo/btl_storage_bootloadinfo.h

storage_appendBootloadList

```
int32_t storage_appendBootloadList (int32_t slotId)
```

Append a storage slot to the list of storage slots to try bootloading from.

Parameters

[in]	slotId	ID of the slot

Returns

- Error code: BOOTLOADER_OK if slot was successfully appended

Definition at line 109 of file platform/bootloader/storage/bootloadinfo/btl_storage_bootloadinfo.h

Macro Definition Documentation

BTL_STORAGE_BOOTLOADINFO_MAGIC

```
#define BTL_STORAGE_BOOTLOADINFO_MAGIC
```

Value:

```
0xE18F5239UL
```

Magic word indicating BootloadInfo_t struct.

Definition at line 51 of file platform/bootloader/storage/bootloadinfo/btl_storage_bootloadinfo.h

BTL_STORAGE_BOOTLOADINFO_VERSION

```
#define BTL_STORAGE_BOOTLOADINFO_VERSION
```

Value:

```
0x00000001UL
```

Version number for the BootloadInfo_t struct.

Definition at line 54 of file platform/bootloader/storage/bootloadinfo/btl_storage_bootloadinfo.h

BTL_STORAGE_BOOTLOAD_LIST_MAX_LENGTH

```
#define BTL_STORAGE_BOOTLOAD_LIST_MAX_LENGTH
```

Value:

```
16UL
```

Maximum number of items in the BootloadInfo_t::bootloadList.

Definition at line 57 of file platform/bootloader/storage/bootloadinfo/btl_storage_bootloadinfo.h

BootloadInfo_t

Definition of the Bootload Info struct.

The Bootload Info struct contains a prioritized list of firmware update images that the bootloader should attempt to bootload. The first image to pass verification will be bootloaded.

Public Attributes

uint32_t	magic Magic word indicating that this is a Bootload Info struct (BTL_STORAGE_BOOTLOADINFO_MAGIC)
uint32_t	structVersion Struct version number.
uint32_t	length Size of the BootloadInfo_t struct.
int32_t	bootloadList List of addresses of slots to bootload from.
uint32_t	crc32 Checksum of the struct.

Public Attribute Documentation

magic

```
uint32_t BootloadInfo_t::magic
```

Magic word indicating that this is a Bootload Info struct (BTL_STORAGE_BOOTLOADINFO_MAGIC)

Definition at line 68 of file platform/bootloader/storage/bootloadinfo/btl_storage_bootloadinfo.h

structVersion

```
uint32_t BootloadInfo_t::structVersion
```

Struct version number.

Definition at line 70 of file platform/bootloader/storage/bootloadinfo/btl_storage_bootloadinfo.h

length

```
uint32_t BootloadInfo_t::length
```

Size of the BootloadInfo_t struct.

Definition at line 72 of file platform/bootloader/storage/bootloadinfo/btl_storage_bootloadinfo.h

bootloadList

```
int32_t BootloadInfo_t::bootloadList[BTL_STORAGE_BOOTLOAD_LIST_MAX_LENGTH]
```

List of addresses of slots to bootload from.

Definition at line `74` of file `platform/bootloader/storage/bootloadinfo/btl_storage_bootloadinfo.h`

crc32

```
uint32_t BootloadInfo_t::crc32
```

Checksum of the struct.

Definition at line `76` of file `platform/bootloader/storage/bootloadinfo/btl_storage_bootloadinfo.h`

SPI Flash Configurations

SPI Flash Configurations

Configuration parameters for SPI flashes.

Variables

const BootloaderStorageImplementationInformation_t	spansion8MInfo Information for Spansion S25L208K.
const BootloaderStorageImplementationInformation_t	windbond2MInfo Information for Winbond W25X20BV.
const BootloaderStorageImplementationInformation_t	windbond8MInfo Information for Winbond W25Q80BV.
const BootloaderStorageImplementationInformation_t	macronix2MInfo Information for Macronix MX25L2006E.
const BootloaderStorageImplementationInformation_t	macronix4MInfo Information for Macronix MX25L4006E.
const BootloaderStorageImplementationInformation_t	macronix8MInfo Information for Macronix MX25L8006E.
const BootloaderStorageImplementationInformation_t	macronix8MLPInfo Information for Macronix MX25R8035F.
const BootloaderStorageImplementationInformation_t	macronix16MInfo Information for Macronix MX25L1606E.
const BootloaderStorageImplementationInformation_t	macronix16M2VInfo Information for Macronix MX25U1635E.
const BootloaderStorageImplementationInformation_t	macronix32MLPInfo Information for Macronix MX25R3235F.

const BootloaderStorageImplementationInformation_t	macronix64MLPInfo Information for Macronix MX25R6435F.
const BootloaderStorageImplementationInformation_t	atmel4MInfo Information for Atmel AT25DF041A.
const BootloaderStorageImplementationInformation_t	atmel8MInfo Information for Atmel AT25DF081A.
const BootloaderStorageImplementationInformation_t	adesto4MInfo Information for Adesto AT25SF041.
const BootloaderStorageImplementationInformation_t	numonyx2MInfo Information for Numonyx M25P20.
const BootloaderStorageImplementationInformation_t	numonyx4MInfo Information for Numonyx M25P40.
const BootloaderStorageImplementationInformation_t	numonyx8MInfo Information for Numonyx M25P80.
const BootloaderStorageImplementationInformation_t	numonyx16MInfo Information for Numonyx M25P16.
const BootloaderStorageImplementationInformation_t	issi256KInfo Information for ISSI IS25LQ025B.
const BootloaderStorageImplementationInformation_t	issi512KInfo Information for ISSI IS25LQ512B.
const BootloaderStorageImplementationInformation_t	issi1MInfo Information for ISSI IS25LQ010B.
const BootloaderStorageImplementationInformation_t	issi2MInfo Information for ISSI IS25LQ020B.
const BootloaderStorageImplementationInformation_t	issi4MInfo Information for ISSI IS25LQ040B.

Macros

#define BTL_STORAGE_SPIFLASH_ALL_DEVICES undefined
Support all devices.

#define BTL_STORAGE_SPIFLASH_SPANSION_DEVICES undefined
Support all Spansion devices.

#define BTL_STORAGE_SPIFLASH_SPANSION_S25FL208K 1
Support Spansion S25FL208K.

#define BTL_STORAGE_SPIFLASH_WINBOND_DEVICES undefined
Support all Winbond devices.

#define BTL_STORAGE_SPIFLASH_WINBOND_W25X20BV 1
Support Winbond W25X20BV.

#define BTL_STORAGE_SPIFLASH_WINBOND_W25Q80BV 1
Support Winbond W25Q80BV.

#define BTL_STORAGE_SPIFLASH_MACRONIX_DEVICES undefined
Support all Macronix devices.

#define BTL_STORAGE_SPIFLASH_MACRONIX_MX25L2006E 1
Support Macronix MX25L2006E.

#define BTL_STORAGE_SPIFLASH_MACRONIX_MX25L4006E 1
Support Macronix MX25L4006E.

#define BTL_STORAGE_SPIFLASH_MACRONIX_MX25L8006E 1
Support Macronix MX25L8006E.

#define BTL_STORAGE_SPIFLASH_MACRONIX_MX25R8035F 1
Support Macronix MX25R8035F.

#define BTL_STORAGE_SPIFLASH_MACRONIX_MX25L1606E 1
Support Macronix MX25L1606E.

#define BTL_STORAGE_SPIFLASH_MACRONIX_MX25U1635E 1
Support Macronix MX25U1635E.

#define BTL_STORAGE_SPIFLASH_MACRONIX_MX25R3235F 1
Support Macronix MX25R3235F.

#define BTL_STORAGE_SPIFLASH_MACRONIX_MX25R6435F 1
Support Macronix MX25R6435F.

#define BTL_STORAGE_SPIFLASH_ATMEL_DEVICES undefined
Support all Atmel devices.

#define BTL_STORAGE_SPIFLASH_ATMEL_AT25DF041A 1
Support Atmel AT25DF041A.

#define BTL_STORAGE_SPIFLASH_ATMEL_AT25DF081A 1
Support Atmel AT25DF081A.

#define BTL_STORAGE_SPIFLASH_ADESTO_AT25SF041 1
Support Adesto AT25SF041.

#define BTL_STORAGE_SPIFLASH_NUMONYX_DEVICES undefined
Support all Numonyx devices.

#define BTL_STORAGE_SPIFLASH_NUMONYX_M25P20 1
Support Numonyx M26P20.

#define BTL_STORAGE_SPIFLASH_NUMONYX_M25P40 1
Support Numonyx M26P40.

#define BTL_STORAGE_SPIFLASH_NUMONYX_M25P80 1
Support Numonyx M26P80.

#define BTL_STORAGE_SPIFLASH_NUMONYX_M25P16 1
Support Numonyx M26P16.

#define BTL_STORAGE_SPIFLASH_ISSI_DEVICES undefined
Support all ISSI devices.

#define BTL_STORAGE_SPIFLASH_ISSI_IS25LQ025B 1
Support ISSI IS25LQ025B.

#define BTL_STORAGE_SPIFLASH_ISSI_IS25LQ512B 1
Support ISSI IS25LQ512B.

#define BTL_STORAGE_SPIFLASH_ISSI_IS25LQ010B 1
Support ISSI IS25LQ010B.

#define BTL_STORAGE_SPIFLASH_ISSI_IS25LQ020B 1
Support ISSI IS25LQ020B.

#define BTL_STORAGE_SPIFLASH_ISSI_IS25LQ040B 1
Support ISSI IS25LQ040B.

Variable Documentation

spansion8MInfo

```
const BootloaderStorageImplementationInformation_t spansion8MInfo
```

Information for Spansion S25L208K.

Definition at line 319 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

windbond2MInfo

```
const BootloaderStorageImplementationInformation_t windbond2MInfo
```

Information for Winbond W25X20BV.

Definition at line 335 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

windbond8MInfo

```
const BootloaderStorageImplementationInformation_t windbond8MInfo
```

Information for Winbond W25Q80BV.

Definition at line 351 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

macronix2MInfo

```
const BootloaderStorageImplementationInformation_t macronix2MInfo
```

Information for Macronix MX25L2006E.

Definition at line 367 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

macronix4MInfo

```
const BootloaderStorageImplementationInformation_t macronix4MInfo
```

Information for Macronix MX25L4006E.

Definition at line 383 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

macronix8MInfo

```
const BootloaderStorageImplementationInformation_t macronix8MInfo
```

Information for Macronix MX25L8006E.

Definition at line 399 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

macronix8MLPInfo

```
const BootloaderStorageImplementationInformation_t macronix8MLPInfo
```

Information for Macronix MX25R8035F.

Definition at line 415 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

macronix16MInfo

```
const BootloaderStorageImplementationInformation_t macronix16MInfo
```

Information for Macronix MX25L1606E.

Definition at line 431 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

macronix16M2VInfo

```
const BootloaderStorageImplementationInformation_t macronix16M2VInfo
```

Information for Macronix MX25U1635E.

Definition at line 447 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

macronix32MLPInfo

```
const BootloaderStorageImplementationInformation_t macronix32MLPInfo
```

Information for Macronix MX25R3235F.

Definition at line 463 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

macronix64MLPInfo

```
const BootloaderStorageImplementationInformation_t macronix64MLPInfo
```

Information for Macronix MX25R6435F.

Definition at line 480 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

atmel4MInfo

```
const BootloaderStorageImplementationInformation_t atmel4MInfo
```

Information for Atmel AT25DF041A.

Definition at line 496 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

atmel8MInfo

```
const BootloaderStorageImplementationInformation_t atmel8MInfo
```

Information for Atmel AT25DF081A.

Definition at line 512 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

adesto4MInfo

```
const BootloaderStorageImplementationInformation_t adesto4MInfo
```

Information for Adesto AT25SF041.

Definition at line 528 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

numonyx2MInfo

```
const BootloaderStorageImplementationInformation_t numonyx2MInfo
```

Information for Numonyx M25P20.

Definition at line 544 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

numonyx4MInfo

```
const BootloaderStorageImplementationInformation_t numonyx4MInfo
```

Information for Numonyx M25P40.

Definition at line 560 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

numonyx8MInfo

```
const BootloaderStorageImplementationInformation_t numonyx8MInfo
```

Information for Numonyx M25P80.

Definition at line 576 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

numonyx16MInfo

```
const BootloaderStorageImplementationInformation_t numonyx16MInfo
```

Information for Numonyx M25P16.

Definition at line 592 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

issi256KInfo

```
const BootloaderStorageImplementationInformation_t issi256KInfo
```

Information for ISSI IS25LQ025B.

Definition at line 608 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

issi512KInfo

```
const BootloaderStorageImplementationInformation_t issi512KInfo
```

Information for ISSI IS25LQ512B.

Definition at line 624 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

issi1MInfo

```
const BootloaderStorageImplementationInformation_t issi1MInfo
```

Information for ISSI IS25LQ010B.

Definition at line 640 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

issi2MInfo

```
const BootloaderStorageImplementationInformation_t issi2MInfo
```

Information for ISSI IS25LQ020B.

Definition at line 656 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

issi4MInfo

```
const BootloaderStorageImplementationInformation_t issi4MInfo
```

Information for ISSI IS25LQ040B.

Definition at line 672 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

Macro Definition Documentation

BTL_STORAGE_SPIFLASH_ALL_DEVICES

```
#define BTL_STORAGE_SPIFLASH_ALL_DEVICES
```

Support all devices.

Definition at line 163 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

BTL_STORAGE_SPIFLASH_SPANSION_DEVICES

```
#define BTL_STORAGE_SPIFLASH_SPANSION_DEVICES
```

Support all Spansion devices.

Definition at line 168 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

BTL_STORAGE_SPIFLASH_SPANSION_S25FL208K

```
#define BTL_STORAGE_SPIFLASH_SPANSION_S25FL208K
```

Value:

```
1
```

Support Spansion S25FL208K.

Definition at line 173 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

BTL_STORAGE_SPIFLASH_WINBOND_DEVICES

```
#define BTL_STORAGE_SPIFLASH_WINBOND_DEVICES
```

Support all Winbond devices.

Definition at line 178 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

BTL_STORAGE_SPIFLASH_WINBOND_W25X20BV

```
#define BTL_STORAGE_SPIFLASH_WINBOND_W25X20BV
```

Value:

```
1
```

Support Winbond W25X20BV.

Definition at line 183 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

BTL_STORAGE_SPIFLASH_WINBOND_W25Q80BV

```
#define BTL_STORAGE_SPIFLASH_WINBOND_W25Q80BV
```

Value:

```
1
```

Support Winbond W25Q80BV.

Definition at line 185 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

BTL_STORAGE_SPIFLASH_MACRONIX_DEVICES

```
#define BTL_STORAGE_SPIFLASH_MACRONIX_DEVICES
```

Support all Macronix devices.

Definition at line 190 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

BTL_STORAGE_SPIFLASH_MACRONIX_MX25L2006E

```
#define BTL_STORAGE_SPIFLASH_MACRONIX_MX25L2006E
```

Value:

```
1
```

Support Macronix MX25L2006E.

Definition at line 195 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

BTL_STORAGE_SPIFLASH_MACRONIX_MX25L4006E

```
#define BTL_STORAGE_SPIFLASH_MACRONIX_MX25L4006E
```

Value:

```
1
```

Support Macronix MX25L4006E.

Definition at line 197 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

BTL_STORAGE_SPIFLASH_MACRONIX_MX25L8006E

```
#define BTL_STORAGE_SPIFLASH_MACRONIX_MX25L8006E
```

Value:

```
1
```

Support Macronix MX25L8006E.

Definition at line 199 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

BTL_STORAGE_SPIFLASH_MACRONIX_MX25R8035F

```
#define BTL_STORAGE_SPIFLASH_MACRONIX_MX25R8035F
```

Value:

```
1
```

Support Macronix MX25R8035F.

Definition at line 201 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

BTL_STORAGE_SPIFLASH_MACRONIX_MX25L1606E

```
#define BTL_STORAGE_SPIFLASH_MACRONIX_MX25L1606E
```

Value:

```
1
```

Support Macronix MX25L1606E.

Definition at line 203 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

BTL_STORAGE_SPIFLASH_MACRONIX_MX25U1635E

```
#define BTL_STORAGE_SPIFLASH_MACRONIX_MX25U1635E
```

Value:

```
1
```

Support Macronix MX25U1635E.

Definition at line 205 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

BTL_STORAGE_SPIFLASH_MACRONIX_MX25R3235F

```
#define BTL_STORAGE_SPIFLASH_MACRONIX_MX25R3235F
```

Value:

```
1
```

Support Macronix MX25R3235F.

Definition at line 207 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

BTL_STORAGE_SPIFLASH_MACRONIX_MX25R6435F

```
#define BTL_STORAGE_SPIFLASH_MACRONIX_MX25R6435F
```

Value:

```
1
```

Support Macronix MX25R6435F.

Definition at line 209 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

BTL_STORAGE_SPIFLASH_ATMEL_DEVICES

```
#define BTL_STORAGE_SPIFLASH_ATMEL_DEVICES
```

Support all Atmel devices.

Definition at line 214 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

BTL_STORAGE_SPIFLASH_ATMEL_AT25DF041A

```
#define BTL_STORAGE_SPIFLASH_ATMEL_AT25DF041A
```

Value:

```
1
```

Support Atmel AT25DF041A.

Definition at line 220 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

BTL_STORAGE_SPIFLASH_ATMEL_AT25DF081A

```
#define BTL_STORAGE_SPIFLASH_ATMEL_AT25DF081A
```

Value:

```
1
```

Support Atmel AT25DF081A.

Definition at line 222 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

BTL_STORAGE_SPIFLASH_ADESTO_AT25SF041

```
#define BTL_STORAGE_SPIFLASH_ADESTO_AT25SF041
```

Value:

```
1
```

Support Adesto AT25SF041.

Definition at line 224 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

BTL_STORAGE_SPIFLASH_NUMONYX_DEVICES

```
#define BTL_STORAGE_SPIFLASH_NUMONYX_DEVICES
```

Support all Numonyx devices.

Definition at line 229 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

BTL_STORAGE_SPIFLASH_NUMONYX_M25P20

```
#define BTL_STORAGE_SPIFLASH_NUMONYX_M25P20
```

Value:

```
1
```

Support Numonyx M26P20.

Definition at line 234 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

BTL_STORAGE_SPIFLASH_NUMONYX_M25P40

```
#define BTL_STORAGE_SPIFLASH_NUMONYX_M25P40
```

Value:

```
1
```

Support Numonyx M26P40.

Definition at line 236 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

BTL_STORAGE_SPIFLASH_NUMONYX_M25P80

```
#define BTL_STORAGE_SPIFLASH_NUMONYX_M25P80
```

Value:

```
1
```

Support Numonyx M26P80.

Definition at line 238 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

BTL_STORAGE_SPIFLASH_NUMONYX_M25P16

```
#define BTL_STORAGE_SPIFLASH_NUMONYX_M25P16
```

Value:

```
1
```

Support Numonyx M26P16.

Definition at line 240 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

BTL_STORAGE_SPIFLASH_ISSI_DEVICES

```
#define BTL_STORAGE_SPIFLASH_ISSI_DEVICES
```

Support all ISSI devices.

Definition at line 245 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

BTL_STORAGE_SPIFLASH_ISSI_IS25LQ025B

```
#define BTL_STORAGE_SPIFLASH_ISSI_IS25LQ025B
```

Value:

```
1
```

Support ISSI IS25LQ025B.

Definition at line 250 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

BTL_STORAGE_SPIFLASH_ISSI_IS25LQ512B

```
#define BTL_STORAGE_SPIFLASH_ISSI_IS25LQ512B
```

Value:

```
1
```

Support ISSI IS25LQ512B.

Definition at line 252 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

BTL_STORAGE_SPIFLASH_ISSI_IS25LQ010B

```
#define BTL_STORAGE_SPIFLASH_ISSI_IS25LQ010B
```

Value:

```
1
```

Support ISSI IS25LQ010B.

Definition at line 254 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

BTL_STORAGE_SPIFLASH_ISSI_IS25LQ020B

```
#define BTL_STORAGE_SPIFLASH_ISSI_IS25LQ020B
```

Value:

```
1
```

Support ISSI IS25LQ020B.

Definition at line 256 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

BTL_STORAGE_SPIFLASH_ISSI_IS25LQ040B

```
#define BTL_STORAGE_SPIFLASH_ISSI_IS25LQ040B
```

Value:

```
1
```

Support ISSI IS25LQ040B.

Definition at line 258 of file platform/bootloader/storage/spiflash/btl_storage_spiflash_configs.h

SPI Flash Configurations using SFDP

SPI Flash Configurations using SFDP

Configuration parameters for SPI flashes Using JEDEC SFDP Standard.

Macros

#define BTL_STORAGE_SPIFLASH_ATMEL_DEVICES undefined
Support all Atmel devices.

#define BTL_STORAGE_SPIFLASH_ATMEL_AT25DF041A undefined
Support Atmel AT25DF041A.

#define BTL_STORAGE_SPIFLASH_ATMEL_AT25DF081A undefined
Support Atmel AT25DF081A.

Macro Definition Documentation

BTL_STORAGE_SPIFLASH_ATMEL_DEVICES

```
#define BTL_STORAGE_SPIFLASH_ATMEL_DEVICES
```

Support all Atmel devices.

Definition at line 85 of file platform/bootloader/storage/spiflash_sfdp/btl_storage_spiflash_configs_sfdp.h

BTL_STORAGE_SPIFLASH_ATMEL_AT25DF041A

```
#define BTL_STORAGE_SPIFLASH_ATMEL_AT25DF041A
```

Support Atmel AT25DF041A.

Definition at line 90 of file platform/bootloader/storage/spiflash_sfdp/btl_storage_spiflash_configs_sfdp.h

BTL_STORAGE_SPIFLASH_ATMEL_AT25DF081A

```
#define BTL_STORAGE_SPIFLASH_ATMEL_AT25DF081A
```

Support Atmel AT25DF081A.

Definition at line 92 of file platform/bootloader/storage/spiflash_sfdp/btl_storage_spiflash_configs_sfdp.h

Driver

Driver

Hardware drivers for bootloader.

Modules

Delay

Delay

Basic delay functionality.

Simple delay routines for use with components that require small delays.

Functions

void	delay_microseconds(uint32_t usecs) Delay for a number of microseconds.
void	delay_init(void) Initialize the delay driver's millisecond counter.
void	delay_milliseconds(uint32_t msecs, bool blocking) Delay for a number of milliseconds.
bool	delay_expired(void) Check whether the current delay has expired.

Function Documentation

delay_microseconds

```
void delay_microseconds (uint32_t usecs)
```

Delay for a number of microseconds.

Parameters

N/A	usecs	Number of microseconds to delay

Note

- This function can be used without calling delay_init() first. This is not an accurate microsecond delay and can have a delay greater than expected, of about +50%. This error becomes significant for lower values delay.

Definition at line `44` of file `platform/bootloader/driver/btl_driver_delay.h`

delay_init

```
void delay_init (void)
```

Initialize the delay driver's millisecond counter.

Parameters

N/A		

Definition at line `49` of file `platform/bootloader/driver/btl_driver_delay.h`

delay_milliseconds

> void delay_milliseconds (uint32_t msecs, bool blocking)

Delay for a number of milliseconds.

Parameters

N/A	msecs	Number of milliseconds to delay. The number should stay within a single TIMER0 overflow (approx. 3300 ms).
N/A	blocking	Whether to block until the delay has expired. If false, the delay_expired() function can be called to check whether the delay has expired.

Definition at line `60` of file `platform/bootloader/driver/btl_driver_delay.h`

delay_expired

> bool delay_expired (void)

Check whether the current delay has expired.

Parameters

N/A		

Returns

- True if the delay has expired.

Definition at line `67` of file `platform/bootloader/driver/btl_driver_delay.h`

SPI

SPI

Basic Serial Peripheral Interface Driver.

Simple, blocking SPI controller implementation for communication with external devices.

Functions

void	spi_init(void) Initialize a USART peripheral for SPI.
void	spi_deinit(void) De-initialize a USART peripheral for SPI.
void	spi_writeByte(uint8_t data) Write a single byte discarding the received byte.
void	spi_writeHalfword(uint16_t data) Write two bytes discarding the received bytes.
void	spi_write3Byte(uint32_t data) Write three bytes discarding the received bytes.
uint8_t	spi_readByte(void) Read a byte by sending a 0xFF byte.
uint16_t	spi_readHalfword(void) Read two bytes by sending two 0xFF bytes.
void	spi_setCsActive(void) Assert the peripheral select line.
void	spi_setCsInactive(void) Dessert the peripheral select line.
uint32_t	spi_getUsartPPUSATD(uint32_t *ppusatdNr) Get PPUSATD word of the (E)USART in use.

Function Documentation

spi_init

```
void spi_init (void)
```

Initialize a USART peripheral for SPI.

Parameters

N/A		

Definition at line 34 of file platform/bootloader/driver/btl_driver_spi_controller.h

spi_deinit

```
void spi_deinit (void)
```

De-initialize a USART peripheral for SPI.

Parameters

N/A		

Definition at line `39` of file `platform/bootloader/driver/btl_driver_spi_controller.h`

spi_writeByte

```
void spi_writeByte (uint8_t data)
```

Write a single byte discarding the received byte.

Parameters

[in]	data	The byte to send

Definition at line `46` of file `platform/bootloader/driver/btl_driver_spi_controller.h`

spi_writeHalfword

```
void spi_writeHalfword (uint16_t data)
```

Write two bytes discarding the received bytes.

Parameters

[in]	data	The bytes to send, most significant byte first

Definition at line `53` of file `platform/bootloader/driver/btl_driver_spi_controller.h`

spi_write3Byte

```
void spi_write3Byte (uint32_t data)
```

Write three bytes discarding the received bytes.

Parameters

[in]	data	The bytes to send, most significant byte first

Definition at line `60` of file `platform/bootloader/driver/btl_driver_spi_controller.h`

spi_readByte

```
uint8_t spi_readByte (void)
```

Read a byte by sending a 0xFF byte.

Parameters

N/A		

Returns

- The received byte

Definition at line 67 of file platform/bootloader/driver/btl_driver_spi_controller.h

spi_readHalfword

```
uint16_t spi_readHalfword (void)
```

Read two bytes by sending two 0xFF bytes.

Parameters

N/A		

Returns

- The received bytes, most significant byte first

Definition at line 74 of file platform/bootloader/driver/btl_driver_spi_controller.h

spi_setCsActive

```
void spi_setCsActive (void)
```

Assert the peripheral select line.

Parameters

N/A		

Polarity is configured by spi_init.

Definition at line 79 of file platform/bootloader/driver/btl_driver_spi_controller.h

spi_setCsInactive

```
void spi_setCsInactive (void)
```

Dessert the peripheral select line.

Parameters

N/A		

Polarity is configured by spi_init.

Definition at line 84 of file platform/bootloader/driver/btl_driver_spi_controller.h

spi_getUsartPPUSATD

```
uint32_t spi_getUsartPPUSATD (uint32_t *ppusatdNr)
```

Get PPUSATD word of the (E)USART in use.

Parameters

[out]	ppusatdNr	PPUSATD register number

Returns

- The PPUSATD word of the (E)USART in use

Definition at line 93 of file platform/bootloader/driver/btl_driver_spi_controller.h

SPI Peripheral

SPI Peripheral

SPI Peripheral Interface driver.

Flexible SPI Peripheral driver implementation for use in an NCP scenario.

This driver will support both blocking and non-blocking operation, with LDMA backing the background transfers to support nonblocking.

Functions

void	spi_peripheral_init(void) Initialize the configured USART peripheral for the SPI peripheral operation.
void	spi_peripheral_deinit(void) Disable the configured USART peripheral for the SPI operation.
int32_t	spi_peripheral_sendBuffer(uint8_t *buffer, size_t length) Write a data buffer to the controller next time the controller starts clocking SCLK.
int32_t	spi_peripheral_sendByte(uint8_t byte) Write one byte to the controller in a blocking fashion.
size_t	spi_peripheral_getTxBytesLeft(void) Get the amount of bytes left in the TX data buffer.
void	spi_peripheral_enableTransmitter(bool enable) Enable/disable MISO output.
void	spi_peripheral_enableReceiver(bool enable) Enable/disable receiving bytes from the controller into the internal buffer.
size_t	spi_peripheral_getRxAvailableBytes(void) Get the amount of bytes ready for reading.
int32_t	spi_peripheral_receiveBuffer(uint8_t *buffer, size_t requestedLength, size_t *receivedLength, bool blocking, uint32_t timeout) Read from the RX buffer into a local buffer.
int32_t	spi_peripheral_receiveByte(uint8_t *byte) Get one byte from the SPI peripheral in a blocking fashion.
void	spi_peripheral_flush(bool flushTx, bool flushRx) Flush one or both buffers.

Function Documentation

spi_peripheral_init

```
void spi_peripheral_init (void)
```

Initialize the configured USART peripheral for the SPI peripheral operation.

Parameters

N/A		

Also sets up GPIO settings for MOSI, MISO, SCLK and SS. After initialization, the SPI peripheral will have RX enabled.

Definition at line `45` of file `platform/bootloader/driver/btl_driver_spi_peripheral.h`

spi_peripheral_deinit

```
void spi_peripheral_deinit (void)
```

Disable the configured USART peripheral for the SPI operation.

Parameters

N/A		

Definition at line `50` of file `platform/bootloader/driver/btl_driver_spi_peripheral.h`

spi_peripheral_sendBuffer

```
int32_t spi_peripheral_sendBuffer (uint8_t *buffer, size_t length)
```

Write a data buffer to the controller next time the controller starts clocking SCLK.

Parameters

[in]	buffer	The data buffer to send
[in]	length	Number of bytes in the buffer to send

This transfer will be non-blocking and its progress can be tracked through spi_peripheral_getTxBytesLeft.

Returns

- BOOTLOADER_OK if successful, error code otherwise

Definition at line `62` of file `platform/bootloader/driver/btl_driver_spi_peripheral.h`

spi_peripheral_sendByte

```
int32_t spi_peripheral_sendByte (uint8_t byte)
```

Write one byte to the controller in a blocking fashion.

Parameters

[in]	byte	The byte to send

Warnings

- If the controller goes down, this will block forever.

Returns

- BOOTLOADER_OK if successful, error code otherwise

Definition at line 74 of file platform/bootloader/driver/btl_driver_spi_peripheral.h

spi_peripheral_getTxBytesLeft

```
size_t spi_peripheral_getTxBytesLeft (void)
```

Get the amount of bytes left in the TX data buffer.

Parameters

N/A		

Returns

- Number of bytes in the transmit buffer still needing to go out

Definition at line 81 of file platform/bootloader/driver/btl_driver_spi_peripheral.h

spi_peripheral_enableTransmitter

```
void spi_peripheral_enableTransmitter (bool enable)
```

Enable/disable MISO output.

Parameters

[in]	enable	True to enable the transmitter, false to disable

Definition at line 88 of file platform/bootloader/driver/btl_driver_spi_peripheral.h

spi_peripheral_enableReceiver

```
void spi_peripheral_enableReceiver (bool enable)
```

Enable/disable receiving bytes from the controller into the internal buffer.

Parameters

[in]	enable	True to enable the receiver, false to disable

This function makes it possible to avoid filling up the buffer with 0xFF while a controller is polling for new data.

Definition at line 97 of file platform/bootloader/driver/btl_driver_spi_peripheral.h

spi_peripheral_getRxAvailableBytes

```
size_t spi_peripheral_getRxAvailableBytes (void)
```

Get the amount of bytes ready for reading.

Parameters

N/A		

Returns

- Number of bytes in the receive buffer available for reading with spi_peripheral_receiveBuffer

Definition at line 105 of file platform/bootloader/driver/btl_driver_spi_peripheral.h

spi_peripheral_receiveBuffer

> int32_t spi_peripheral_receiveBuffer (uint8_t *buffer, size_t requestedLength, size_t *receivedLength, bool blocking, uint32_t timeout)

Read from the RX buffer into a local buffer.

Parameters

[out]	buffer	The data buffer to receive into
[in]	requestedLength	Number of bytes we'd like to read
[out]	receivedLength	Number of bytes read
[in]	blocking	Indicate whether we should wait for requestedLength bytes to be available and read before returning, or we can read out whatever is currently in the buffer and return.
[in]	timeout	Number of milliseconds to wait for data in blocking mode

Returns

- BOOTLOADER_OK if successful, error code otherwise

Definition at line 120 of file platform/bootloader/driver/btl_driver_spi_peripheral.h

spi_peripheral_receiveByte

> int32_t spi_peripheral_receiveByte (uint8_t *byte)

Get one byte from the SPI peripheral in a blocking fashion.

Parameters

[out]	byte	Pointer to where to put the received byte

Warnings

- If the controller never clocks in a byte, this function will block forever.

Returns

- BOOTLOADER_OK if successful, error code otherwise

Definition at line 136 of file platform/bootloader/driver/btl_driver_spi_peripheral.h

spi_peripheral_flush

> void spi_peripheral_flush (bool flushTx, bool flushRx)

Flush one or both buffers.

Parameters

[in]	flushTx	Flush the transmit buffer when true
[in]	flushRx	Flush the receive buffer when true

Definition at line 144 of file platform/bootloader/driver/btl_driver_spi_peripheral.h

UART

UART

Serial UART Interface Driver.

Flexible UART driver implementation for communication with external devices.

Functions

void	uart_init(void) Initialize the configured USART peripheral for UART operation.
void	uart_deinit(void) Disable the configured USART peripheral for UART operation.
int32_t	uart_sendBuffer(uint8_t *buffer, size_t length, bool blocking) Write a data buffer to the UART.
int32_t	uart_sendByte(uint8_t byte) Write one byte to the UART in a blocking fashion.
bool	uart_isTxIdle(void) Find out whether the UART can accept more data to send.
size_t	uart_getRxAvailableBytes(void) Get the number of bytes ready for reading.
int32_t	uart_receiveBuffer(uint8_t *buffer, size_t requestedLength, size_t *receivedLength, bool blocking, uint32_t timeout) Read from the UART into a data buffer.
int32_t	uart_receiveByte(uint8_t *byte) Get one byte from UART in a blocking fashion.
int32_t	uart_receiveByteTimeout(uint8_t *byte, uint32_t timeout) Get one byte from UART in a blocking fashion.
int32_t	uart_flush(bool flushTx, bool flushRx) Flush one or both UART buffers.

Function Documentation

uart_init

```
void uart_init (void)
```

Initialize the configured USART peripheral for UART operation.

Parameters

N/A		

Also sets up GPIO settings for TX, RX, and, if configured, flow control.

Definition at line 38 of file platform/bootloader/driver/btl_serial_driver.h

uart_deinit

```
void uart_deinit (void)
```

Disable the configured USART peripheral for UART operation.

Parameters

N/A		

Definition at line 43 of file platform/bootloader/driver/btl_serial_driver.h

uart_sendBuffer

```
int32_t uart_sendBuffer (uint8_t *buffer, size_t length, bool blocking)
```

Write a data buffer to the UART.

Parameters

[in]	buffer	The data buffer to send
[in]	length	Number of bytes in the buffer to send
[in]	blocking	Indicates whether this transfer can be offloaded to LDMA and return, or whether to wait on completion before returning.

Returns

- BOOTLOADER_OK if successful, error code otherwise

Definition at line 56 of file platform/bootloader/driver/btl_serial_driver.h

uart_sendByte

```
int32_t uart_sendByte (uint8_t byte)
```

Write one byte to the UART in a blocking fashion.

Parameters

[in]	byte	The byte to send

Returns

- BOOTLOADER_OK if successful, error code otherwise

Definition at line 65 of file platform/bootloader/driver/btl_serial_driver.h

uart_isTxIdle

```
bool uart_isTxIdle (void)
```

Find out whether the UART can accept more data to send.

Parameters

N/A		

Returns

- true if the UART is not currently transmitting

Definition at line 72 of file platform/bootloader/driver/btl_serial_driver.h

uart_getRxAvailableBytes

```
size_t uart_getRxAvailableBytes (void)
```

Get the number of bytes ready for reading.

Parameters

N/A		

Returns

- Number of bytes in the receive buffer available for reading with uart_receiveBuffer

Definition at line 80 of file platform/bootloader/driver/btl_serial_driver.h

uart_receiveBuffer

```
int32_t uart_receiveBuffer (uint8_t *buffer, size_t requestedLength, size_t *receivedLength, bool blocking, uint32_t timeout)
```

Read from the UART into a data buffer.

Parameters

[out]	buffer	The data buffer to receive into
[in]	requestedLength	Number of bytes to read
[out]	receivedLength	Number of bytes read
[in]	blocking	Indicates whether to wait for requestedLength bytes to be available and read before returning, or whether to read out data currently in the buffer and return.
[in]	timeout	Number of milliseconds to wait for data in blocking mode

Returns

- BOOTLOADER_OK if successful, error code otherwise

Definition at line 95 of file platform/bootloader/driver/btl_serial_driver.h

uart_receiveByte

```
int32_t uart_receiveByte (uint8_t *byte)
```

Get one byte from UART in a blocking fashion.

Parameters

[out]	byte	The byte to send

Returns

- BOOTLOADER_OK if successful, error code otherwise

Definition at line 108 of file platform/bootloader/driver/btl_serial_driver.h

uart_receiveByteTimeout

> int32_t uart_receiveByteTimeout (uint8_t *byte, uint32_t timeout)

Get one byte from UART in a blocking fashion.

Parameters

[out]	byte	The byte to send
[in]	timeout	Maximum timeout before aborting transfer

Returns

- BOOTLOADER_OK if successful, error code otherwise

Definition at line 118 of file platform/bootloader/driver/btl_serial_driver.h

uart_flush

> int32_t uart_flush (bool flushTx, bool flushRx)

Flush one or both UART buffers.

Parameters

[in]	flushTx	Flush the transmit buffer when true
[in]	flushRx	Flush the receive buffer when true

Returns

- BOOTLOADER_OK

Definition at line 128 of file platform/bootloader/driver/btl_serial_driver.h

Error Codes

Error Codes

Bootloader error codes.

Modules

Macros

#define BOOTLOADER_OK 0L
No error, operation OK.

Macro Definition Documentation

BOOTLOADER_OK

```
#define BOOTLOADER_OK
```

Value:

```
0L
```

No error, operation OK.

Definition at line 28 of file platform/bootloader/api/btl_errorcode.h

Bootloading Error Codes

Bootloading Error Codes

Bootloader error codes returned by the bootloading process.

Offset from BOOTLOADER_ERROR_BOOTLOAD_BASE

Macros

#define BOOTLOADER_ERROR_BOOTLOAD_LIST_EMPTY (BOOTLOADER_ERROR_BOOTLOAD_BASE | 0×01L)
No images marked for bootload.

#define BOOTLOADER_ERROR_BOOTLOAD_LIST_FULL (BOOTLOADER_ERROR_BOOTLOAD_BASE | 0×02L)
List of images marked for bootload is full.

#define BOOTLOADER_ERROR_BOOTLOAD_LIST_ENTRY_EXISTS (BOOTLOADER_ERROR_BOOTLOAD_BASE | 0×03L)
Image already marked for bootload.

#define BOOTLOADER_ERROR_BOOTLOAD_LIST_OVERFLOW (BOOTLOADER_ERROR_BOOTLOAD_BASE | 0×04L)
Bootload list overflowed, requested length too large.

#define BOOTLOADER_ERROR_BOOTLOAD_LIST_NO_LIST (BOOTLOADER_ERROR_BOOTLOAD_BASE | 0×05L)
No bootload list found at the base of storage.

#define BOOTLOADER_ERROR_BOOTLOAD_LIST_INVALID (BOOTLOADER_ERROR_BOOTLOAD_BASE | 0×06L)
Bootload list found but with invalid CRC.

Macro Definition Documentation

BOOTLOADER_ERROR_BOOTLOAD_LIST_EMPTY

```
#define BOOTLOADER_ERROR_BOOTLOAD_LIST_EMPTY
```

Value:

```
(BOOTLOADER_ERROR_BOOTLOAD_BASE | 0×01L)
```

No images marked for bootload.

Definition at line 143 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_BOOTLOAD_LIST_FULL

```
#define BOOTLOADER_ERROR_BOOTLOAD_LIST_FULL
```

Value:

```
(BOOTLOADER_ERROR_BOOTLOAD_BASE | 0×02L)
```

List of images marked for bootload is full.

Definition at line 146 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_BOOTLOAD_LIST_ENTRY_EXISTS

```
#define BOOTLOADER_ERROR_BOOTLOAD_LIST_ENTRY_EXISTS
```

Value:

```
(BOOTLOADER_ERROR_BOOTLOAD_BASE | 0x03L)
```

Image already marked for bootload.

Definition at line 149 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_BOOTLOAD_LIST_OVERFLOW

```
#define BOOTLOADER_ERROR_BOOTLOAD_LIST_OVERFLOW
```

Value:

```
(BOOTLOADER_ERROR_BOOTLOAD_BASE | 0x04L)
```

Bootload list overflowed, requested length too large.

Definition at line 152 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_BOOTLOAD_LIST_NO_LIST

```
#define BOOTLOADER_ERROR_BOOTLOAD_LIST_NO_LIST
```

Value:

```
(BOOTLOADER_ERROR_BOOTLOAD_BASE | 0x05L)
```

No bootload list found at the base of storage.

Definition at line 155 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_BOOTLOAD_LIST_INVALID

```
#define BOOTLOADER_ERROR_BOOTLOAD_LIST_INVALID
```

Value:

```
(BOOTLOADER_ERROR_BOOTLOAD_BASE | 0x06L)
```

Bootload list found but with invalid CRC.

Definition at line 158 of file platform/bootloader/api/btl_errorcode.h

Communication Component Error Codes

Communication Component Error Codes

Bootloader error codes returned by communication components.

Offset from BOOTLOADER_ERROR_COMMUNICATION_BASE

Macros

#define BOOTLOADER_ERROR_COMMUNICATION_INIT (BOOTLOADER_ERROR_COMMUNICATION_BASE | 0×01L)
Invalid input parameter to security algorithm Could not initialize hardware resources for communication protocol.

#define BOOTLOADER_ERROR_COMMUNICATION_START (BOOTLOADER_ERROR_COMMUNICATION_BASE | 0×02L)
Could not start communication with host (timeout, sync error, version mismatch, ...)

#define BOOTLOADER_ERROR_COMMUNICATION_DONE (BOOTLOADER_ERROR_COMMUNICATION_BASE | 0×03L)
Host closed communication, no image received.

#define BOOTLOADER_ERROR_COMMUNICATION_ERROR (BOOTLOADER_ERROR_COMMUNICATION_BASE | 0×04L)
Unrecoverable error in host-bootloader communication.

#define BOOTLOADER_ERROR_COMMUNICATION_IMAGE_ERROR (BOOTLOADER_ERROR_COMMUNICATION_BASE | 0×05L)
Host closed communication, no valid image received.

#define BOOTLOADER_ERROR_COMMUNICATION_TIMEOUT (BOOTLOADER_ERROR_COMMUNICATION_BASE | 0×06L)
Communication aborted, no response from host.

Macro Definition Documentation

BOOTLOADER_ERROR_COMMUNICATION_INIT

```
#define BOOTLOADER_ERROR_COMMUNICATION_INIT
```

Value:

```
(BOOTLOADER_ERROR_COMMUNICATION_BASE | 0×01L)
```

Invalid input parameter to security algorithm Could not initialize hardware resources for communication protocol.

Definition at line 194 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_COMMUNICATION_START

```
#define BOOTLOADER_ERROR_COMMUNICATION_START
```

Value:

```
(BOOTLOADER_ERROR_COMMUNICATION_BASE | 0×02L)
```

Could not start communication with host (timeout, sync error, version mismatch, ...)

Definition at line 198 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_COMMUNICATION_DONE

```
#define BOOTLOADER_ERROR_COMMUNICATION_DONE
```

Value:

```
(BOOTLOADER_ERROR_COMMUNICATION_BASE | 0x03L)
```

Host closed communication, no image received.

Definition at line 201 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_COMMUNICATION_ERROR

```
#define BOOTLOADER_ERROR_COMMUNICATION_ERROR
```

Value:

```
(BOOTLOADER_ERROR_COMMUNICATION_BASE | 0x04L)
```

Unrecoverable error in host-bootloader communication.

Definition at line 204 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_COMMUNICATION_IMAGE_ERROR

```
#define BOOTLOADER_ERROR_COMMUNICATION_IMAGE_ERROR
```

Value:

```
(BOOTLOADER_ERROR_COMMUNICATION_BASE | 0x05L)
```

Host closed communication, no valid image received.

Definition at line 207 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_COMMUNICATION_TIMEOUT

```
#define BOOTLOADER_ERROR_COMMUNICATION_TIMEOUT
```

Value:

```
(BOOTLOADER_ERROR_COMMUNICATION_BASE | 0x06L)
```

Communication aborted, no response from host.

Definition at line 210 of file platform/bootloader/api/btl_errorcode.h

Compression Error Codes

Compression Error Codes

Bootloader error codes returned by the decompressor.

Offset from BOOTLOADER_ERROR_COMPRESSION_BASE

Macros

#define BOOTLOADER_ERROR_COMPRESSION_INIT (BOOTLOADER_ERROR_COMPRESSION_BASE | 0×01)
Could not initialize decompressor.

#define BOOTLOADER_ERROR_COMPRESSION_STATE (BOOTLOADER_ERROR_COMPRESSION_BASE | 0×02)
Invalid decompressor state – possible invalid input.

#define BOOTLOADER_ERROR_COMPRESSION_DATA (BOOTLOADER_ERROR_COMPRESSION_BASE | 0×03)
Data error.

#define BOOTLOADER_ERROR_COMPRESSION_DATALEN (BOOTLOADER_ERROR_COMPRESSION_BASE | 0×04)
Data length error.

#define BOOTLOADER_ERROR_COMPRESSION_MEM (BOOTLOADER_ERROR_COMPRESSION_BASE | 0×05)
Memory error.

Macro Definition Documentation

BOOTLOADER_ERROR_COMPRESSION_INIT

```
#define BOOTLOADER_ERROR_COMPRESSION_INIT
```

Value:

```
(BOOTLOADER_ERROR_COMPRESSION_BASE | 0×01)
```

Could not initialize decompressor.

Definition at line 360 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_COMPRESSION_STATE

```
#define BOOTLOADER_ERROR_COMPRESSION_STATE
```

Value:

```
(BOOTLOADER_ERROR_COMPRESSION_BASE | 0×02)
```

Invalid decompressor state – possible invalid input.

Definition at line 363 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_COMPRESSION_DATA

```
#define BOOTLOADER_ERROR_COMPRESSION_DATA
```

Value:

```
(BOOTLOADER_ERROR_COMPRESSION_BASE | 0x03)
```

Data error.

Definition at line 366 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_COMPRESSION_DATALEN

```
#define BOOTLOADER_ERROR_COMPRESSION_DATALEN
```

Value:

```
(BOOTLOADER_ERROR_COMPRESSION_BASE | 0x04)
```

Data length error.

Definition at line 369 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_COMPRESSION_MEM

```
#define BOOTLOADER_ERROR_COMPRESSION_MEM
```

Value:

```
(BOOTLOADER_ERROR_COMPRESSION_BASE | 0x05)
```

Memory error.

Definition at line 372 of file platform/bootloader/api/btl_errorcode.h

Error Code Base Values

Error Code Base Values

Bootloader error code base values, per logical function.

Macros

#define BOOTLOADER_ERROR_INIT_BASE 0×0100L
Initialization errors.

#define BOOTLOADER_ERROR_PARSE_BASE 0×0200L
Image verification errors.

#define BOOTLOADER_ERROR_STORAGE_BASE 0×0400L
Storage errors.

#define BOOTLOADER_ERROR_BOOTLOAD_BASE 0×0500L
Bootload errors.

#define BOOTLOADER_ERROR_SECURITY_BASE 0×0600L
Security errors.

#define BOOTLOADER_ERROR_COMMUNICATION_BASE 0×0700L
Communication component errors.

#define BOOTLOADER_ERROR_XMODEM_BASE 0×0900L
XMODEM parser errors.

#define BOOTLOADER_ERROR_PARSER_BASE 0×1000L
Image file parser errors.

#define BOOTLOADER_ERROR_SPI_PERIPHERAL_BASE 0×1100L
SPI Peripheral driver errors.

#define BOOTLOADER_ERROR_UART_BASE 0×1200L
UART driver errors.

#define BOOTLOADER_ERROR_COMPRESSION_BASE 0×1300L
Compression errors.

Macro Definition Documentation

BOOTLOADER_ERROR_INIT_BASE

```
#define BOOTLOADER_ERROR_INIT_BASE
```

Value:

```
0×0100L
```

Initialization errors.

Definition at line 37 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_PARSE_BASE

```
#define BOOTLOADER_ERROR_PARSE_BASE
```

Value:

```
0x0200L
```

Image verification errors.

Definition at line 39 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_STORAGE_BASE

```
#define BOOTLOADER_ERROR_STORAGE_BASE
```

Value:

```
0x0400L
```

Storage errors.

Definition at line 41 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_BOOTLOAD_BASE

```
#define BOOTLOADER_ERROR_BOOTLOAD_BASE
```

Value:

```
0x0500L
```

Bootload errors.

Definition at line 43 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_SECURITY_BASE

```
#define BOOTLOADER_ERROR_SECURITY_BASE
```

Value:

```
0x0600L
```

Security errors.

Definition at line 45 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_COMMUNICATION_BASE

```
#define BOOTLOADER_ERROR_COMMUNICATION_BASE
```

Value:

```
0x0700L
```

Communication component errors.

Definition at line 47 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_XMODEM_BASE

```
#define BOOTLOADER_ERROR_XMODEM_BASE
```

Value:

```
0x0900L
```

XMODEM parser errors.

Definition at line 49 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_PARSER_BASE

```
#define BOOTLOADER_ERROR_PARSER_BASE
```

Value:

```
0x1000L
```

Image file parser errors.

Definition at line 51 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_SPI_PERIPHERAL_BASE

```
#define BOOTLOADER_ERROR_SPI_PERIPHERAL_BASE
```

Value:

```
0x1100L
```

SPI Peripheral driver errors.

Definition at line 53 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_UART_BASE

```
#define BOOTLOADER_ERROR_UART_BASE
```

Value:

```
0x1200L
```

UART driver errors.

Definition at line 55 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_COMPRESSION_BASE

```
#define BOOTLOADER_ERROR_COMPRESSION_BASE
```

Value:

```
0x1300L
```

Compression errors.

Definition at line 57 of file platform/bootloader/api/btl_errorcode.h

Image Parser Error Codes

Image Parser Error Codes

Bootloader error codes returned by the image file parser.

Offset from BOOTLOADER_ERROR_PARSER_BASE

Macros

#define BOOTLOADER_ERROR_PARSER_UNEXPECTED (BOOTLOADER_ERROR_PARSER_BASE | 0x01L)
Encountered unexpected data/option.

#define BOOTLOADER_ERROR_PARSER_BUFFER (BOOTLOADER_ERROR_PARSER_BASE | 0x02L)
Ran out of internal buffer space.

#define BOOTLOADER_ERROR_PARSER_PARSED (BOOTLOADER_ERROR_PARSER_BASE | 0x03L)
Internal state: done parsing the current input buffer.

#define BOOTLOADER_ERROR_PARSER_KEYERROR (BOOTLOADER_ERROR_PARSER_BASE | 0x04L)
Invalid encryption key or no key not present.

#define BOOTLOADER_ERROR_PARSER_CRC (BOOTLOADER_ERROR_PARSER_BASE | 0x05L)
Invalid checksum.

#define BOOTLOADER_ERROR_PARSER_SIGNATURE (BOOTLOADER_ERROR_PARSER_BASE | 0x06L)
Invalid signature.

#define BOOTLOADER_ERROR_PARSER_EOF (BOOTLOADER_ERROR_PARSER_BASE | 0x07L)
Image parsing is already done (or has previously errored out)

#define BOOTLOADER_ERROR_PARSER_UNKNOWN_TAG (BOOTLOADER_ERROR_PARSER_BASE | 0x08L)
Unknown data type in image file.

#define BOOTLOADER_ERROR_PARSER_VERSION (BOOTLOADER_ERROR_PARSER_BASE | 0x09L)
Image file version doesn't match with parser.

#define BOOTLOADER_ERROR_PARSER_FILETYPE (BOOTLOADER_ERROR_PARSER_BASE | 0x0AL)
Image file type doesn't match with parser.

#define BOOTLOADER_ERROR_PARSER_INIT (BOOTLOADER_ERROR_PARSER_BASE | 0x0BL)
Initialization failed.

#define BOOTLOADER_ERROR_PARSER_REJECTED (BOOTLOADER_ERROR_PARSER_BASE | 0x0CL)
Upgrade file was rejected.

#define BOOTLOADER_ERROR_PARSER_OVERLAP (BOOTLOADER_ERROR_PARSER_BASE | 0x0DL)
Upgrade file overlaps with the upgrade location.

#define BOOTLOADER_ERROR_PARSER_INVALID_TAG_ORDER (BOOTLOADER_ERROR_PARSER_BASE | 0x0EL)
A GBL tag occurred in an order forbidden by the GBL format spec.

Macro Definition Documentation

BOOTLOADER_ERROR_PARSER_UNEXPECTED

```
#define BOOTLOADER_ERROR_PARSER_UNEXPECTED
```

Value:

```
(BOOTLOADER_ERROR_PARSER_BASE | 0x01L)
```

Encountered unexpected data/option.

Definition at line 257 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_PARSER_BUFFER

```
#define BOOTLOADER_ERROR_PARSER_BUFFER
```

Value:

```
(BOOTLOADER_ERROR_PARSER_BASE | 0x02L)
```

Ran out of internal buffer space.

Please increase internal buffer size to match biggest header

Definition at line 261 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_PARSER_PARSED

```
#define BOOTLOADER_ERROR_PARSER_PARSED
```

Value:

```
(BOOTLOADER_ERROR_PARSER_BASE | 0x03L)
```

Internal state: done parsing the current input buffer.

Definition at line 264 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_PARSER_KEYERROR

```
#define BOOTLOADER_ERROR_PARSER_KEYERROR
```

Value:

```
(BOOTLOADER_ERROR_PARSER_BASE | 0x04L)
```

Invalid encryption key or no key not present.

Definition at line 267 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_PARSER_CRC

```
#define BOOTLOADER_ERROR_PARSER_CRC
```

Value:

```
(BOOTLOADER_ERROR_PARSER_BASE | 0x05L)
```

Invalid checksum.

Definition at line 270 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_PARSER_SIGNATURE

```
#define BOOTLOADER_ERROR_PARSER_SIGNATURE
```

Value:

```
(BOOTLOADER_ERROR_PARSER_BASE | 0x06L)
```

Invalid signature.

Definition at line 273 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_PARSER_EOF

```
#define BOOTLOADER_ERROR_PARSER_EOF
```

Value:

```
(BOOTLOADER_ERROR_PARSER_BASE | 0x07L)
```

Image parsing is already done (or has previously errored out)

Definition at line 276 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_PARSER_UNKNOWN_TAG

```
#define BOOTLOADER_ERROR_PARSER_UNKNOWN_TAG
```

Value:

```
(BOOTLOADER_ERROR_PARSER_BASE | 0x08L)
```

Unknown data type in image file.

Definition at line 279 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_PARSER_VERSION

```
#define BOOTLOADER_ERROR_PARSER_VERSION
```

Value:

```
(BOOTLOADER_ERROR_PARSER_BASE|0x09L)
```

Image file version doesn't match with parser.

Definition at line 282 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_PARSER_FILETYPE

```
#define BOOTLOADER_ERROR_PARSER_FILETYPE
```

Value:

```
(BOOTLOADER_ERROR_PARSER_BASE|0x0AL)
```

Image file type doesn't match with parser.

Definition at line 285 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_PARSER_INIT

```
#define BOOTLOADER_ERROR_PARSER_INIT
```

Value:

```
(BOOTLOADER_ERROR_PARSER_BASE|0x0BL)
```

Initialization failed.

Definition at line 288 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_PARSER_REJECTED

```
#define BOOTLOADER_ERROR_PARSER_REJECTED
```

Value:

```
(BOOTLOADER_ERROR_PARSER_BASE|0x0CL)
```

Upgrade file was rejected.

Definition at line 291 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_PARSER_OVERLAP

```
#define BOOTLOADER_ERROR_PARSER_OVERLAP
```

Value:

```
(BOOTLOADER_ERROR_PARSER_BASE|0x0DL)
```

Upgrade file overlaps with the upgrade location.

Definition at line 294 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_PARSER_INVALID_TAG_ORDER

```
#define BOOTLOADER_ERROR_PARSER_INVALID_TAG_ORDER
```

Value:

```
(BOOTLOADER_ERROR_PARSER_BASE | 0x0EL)
```

A GBL tag occurred in an order forbidden by the GBL format spec.

Definition at line 297 of file platform/bootloader/api/btl_errorcode.h

Initialization Error Codes

Initialization Error Codes

Bootloader error codes returned by initialization code.

Offset from BOOTLOADER_ERROR_INIT_BASE

Macros

#define BOOTLOADER_ERROR_INIT_STORAGE (BOOTLOADER_ERROR_INIT_BASE | 0×01L)
Storage initialization error.

#define BOOTLOADER_ERROR_INIT_TABLE (BOOTLOADER_ERROR_INIT_BASE | 0×02L)
Bootloader table invalid.

#define BOOTLOADER_ERROR_INIT_SFDP (BOOTLOADER_ERROR_INIT_BASE | 0×03L)
Bootloader SFDP not supported.

Macro Definition Documentation

BOOTLOADER_ERROR_INIT_STORAGE

```
#define BOOTLOADER_ERROR_INIT_STORAGE
```

Value:

```
(BOOTLOADER_ERROR_INIT_BASE | 0×01L)
```

Storage initialization error.

Definition at line 69 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_INIT_TABLE

```
#define BOOTLOADER_ERROR_INIT_TABLE
```

Value:

```
(BOOTLOADER_ERROR_INIT_BASE | 0×02L)
```

Bootloader table invalid.

Definition at line 72 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_INIT_SFDP

```
#define BOOTLOADER_ERROR_INIT_SFDP
```

Value:

```
(BOOTLOADER_ERROR_INIT_BASE | 0x03L)
```

Bootloader SFDP not supported.

Definition at line 75 of file platform/bootloader/api/btl_errorcode.h

Parse Error Codes

Parse Error Codes

Bootloader error codes returned by image parsing.

Offset from BOOTLOADER_ERROR_PARSE_BASE

Macros

#define BOOTLOADER_ERROR_PARSE_CONTINUE (BOOTLOADER_ERROR_PARSE_BASE | 0×01L)
Parse not complete, continue calling the parsing function.

#define BOOTLOADER_ERROR_PARSE_FAILED (BOOTLOADER_ERROR_PARSE_BASE | 0×02L)
Verification failed.

#define BOOTLOADER_ERROR_PARSE_SUCCESS (BOOTLOADER_ERROR_PARSE_BASE | 0×03L)
Verification successfully completed. Image is valid.

#define BOOTLOADER_ERROR_PARSE_STORAGE (BOOTLOADER_ERROR_PARSE_BASE | 0×04L)
Bootloader has no storage, and cannot parse images.

#define BOOTLOADER_ERROR_PARSE_CONTEXT (BOOTLOADER_ERROR_PARSE_BASE | 0×05L)
Parse context incompatible with parse function.

Macro Definition Documentation

BOOTLOADER_ERROR_PARSE_CONTINUE

```
#define BOOTLOADER_ERROR_PARSE_CONTINUE
```

Value:

```
(BOOTLOADER_ERROR_PARSE_BASE|0×01L)
```

Parse not complete, continue calling the parsing function.

Definition at line 89 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_PARSE_FAILED

```
#define BOOTLOADER_ERROR_PARSE_FAILED
```

Value:

```
(BOOTLOADER_ERROR_PARSE_BASE|0×02L)
```

Verification failed.

Definition at line 91 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_PARSE_SUCCESS

```
#define BOOTLOADER_ERROR_PARSE_SUCCESS
```

Value:

```
(BOOTLOADER_ERROR_PARSE_BASE | 0x03L)
```

Verification successfully completed. Image is valid.

Definition at line 93 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_PARSE_STORAGE

```
#define BOOTLOADER_ERROR_PARSE_STORAGE
```

Value:

```
(BOOTLOADER_ERROR_PARSE_BASE | 0x04L)
```

Bootloader has no storage, and cannot parse images.

Definition at line 95 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_PARSE_CONTEXT

```
#define BOOTLOADER_ERROR_PARSE_CONTEXT
```

Value:

```
(BOOTLOADER_ERROR_PARSE_BASE | 0x05L)
```

Parse context incompatible with parse function.

Definition at line 97 of file platform/bootloader/api/btl_errorcode.h

SPI Peripheral Driver Error Codes

SPI Peripheral Driver Error Codes

Bootloader error codes returned by the SPI Peripheral driver.

Offset from BOOTLOADER_ERROR_SPI_PERIPHERAL_BASE

Macros

#define BOOTLOADER_ERROR_SPI_PERIPHERAL_UNINIT (BOOTLOADER_ERROR_SPI_PERIPHERAL_BASE | 0×01)
Operation not allowed because hardware has not been initialized.

#define BOOTLOADER_ERROR_SPI_PERIPHERAL_INIT (BOOTLOADER_ERROR_SPI_PERIPHERAL_BASE | 0×02)
Hardware fail during initialization.

#define BOOTLOADER_ERROR_SPI_PERIPHERAL_ARGUMENT (BOOTLOADER_ERROR_SPI_PERIPHERAL_BASE | 0×03)
Invalid argument.

#define BOOTLOADER_ERROR_SPI_PERIPHERAL_TIMEOUT (BOOTLOADER_ERROR_SPI_PERIPHERAL_BASE | 0×04)
Timeout.

#define BOOTLOADER_ERROR_SPI_PERIPHERAL_OVERFLOW (BOOTLOADER_ERROR_SPI_PERIPHERAL_BASE | 0×05)
Buffer overflow condition.

#define BOOTLOADER_ERROR_SPI_PERIPHERAL_BUSY (BOOTLOADER_ERROR_SPI_PERIPHERAL_BASE | 0×06)
Busy condition.

Macro Definition Documentation

BOOTLOADER_ERROR_SPI_PERIPHERAL_UNINIT

```
#define BOOTLOADER_ERROR_SPI_PERIPHERAL_UNINIT
```

Value:

```
(BOOTLOADER_ERROR_SPI_PERIPHERAL_BASE | 0×01)
```

Operation not allowed because hardware has not been initialized.

Definition at line 310 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_SPI_PERIPHERAL_INIT

```
#define BOOTLOADER_ERROR_SPI_PERIPHERAL_INIT
```

Value:

```
(BOOTLOADER_ERROR_SPI_PERIPHERAL_BASE | 0×02)
```

Hardware fail during initialization.

Definition at line 313 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_SPI_PERIPHERAL_ARGUMENT

```
#define BOOTLOADER_ERROR_SPI_PERIPHERAL_ARGUMENT
```

Value:

```
(BOOTLOADER_ERROR_SPI_PERIPHERAL_BASE | 0x03)
```

Invalid argument.

Definition at line 316 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_SPI_PERIPHERAL_TIMEOUT

```
#define BOOTLOADER_ERROR_SPI_PERIPHERAL_TIMEOUT
```

Value:

```
(BOOTLOADER_ERROR_SPI_PERIPHERAL_BASE | 0x04)
```

Timeout.

Definition at line 319 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_SPI_PERIPHERAL_OVERFLOW

```
#define BOOTLOADER_ERROR_SPI_PERIPHERAL_OVERFLOW
```

Value:

```
(BOOTLOADER_ERROR_SPI_PERIPHERAL_BASE | 0x05)
```

Buffer overflow condition.

Definition at line 322 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_SPI_PERIPHERAL_BUSY

```
#define BOOTLOADER_ERROR_SPI_PERIPHERAL_BUSY
```

Value:

```
(BOOTLOADER_ERROR_SPI_PERIPHERAL_BASE | 0x06)
```

Busy condition.

Definition at line 325 of file platform/bootloader/api/btl_errorcode.h

Security Error Codes

Security Error Codes

Bootloader error codes returned by security algorithms.

Offset from BOOTLOADER_ERROR_SECURITY_BASE

Macros

#define BOOTLOADER_ERROR_SECURITY_INVALID_PARAM (BOOTLOADER_ERROR_SECURITY_BASE | 0×01L)
Invalid input parameter to security algorithm.

#define BOOTLOADER_ERROR_SECURITY_PARAM_OUT_RANGE (BOOTLOADER_ERROR_SECURITY_BASE | 0×02L)
Input parameter to security algorithm is out of range.

#define BOOTLOADER_ERROR_SECURITY_INVALID_OPTION (BOOTLOADER_ERROR_SECURITY_BASE | 0×03L)
Invalid option for security algorithm.

#define BOOTLOADER_ERROR_SECURITY_REJECTED (BOOTLOADER_ERROR_SECURITY_BASE | 0×04L)
Authentication did not check out.

Macro Definition Documentation

BOOTLOADER_ERROR_SECURITY_INVALID_PARAM

```
#define BOOTLOADER_ERROR_SECURITY_INVALID_PARAM
```

Value:

```
(BOOTLOADER_ERROR_SECURITY_BASE|0×01L)
```

Invalid input parameter to security algorithm.

Definition at line 171 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_SECURITY_PARAM_OUT_RANGE

```
#define BOOTLOADER_ERROR_SECURITY_PARAM_OUT_RANGE
```

Value:

```
(BOOTLOADER_ERROR_SECURITY_BASE|0×02L)
```

Input parameter to security algorithm is out of range.

Definition at line 174 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_SECURITY_INVALID_OPTION

```
#define BOOTLOADER_ERROR_SECURITY_INVALID_OPTION
```

Value:

```
(BOOTLOADER_ERROR_SECURITY_BASE | 0x03L)
```

Invalid option for security algorithm.

Definition at line 177 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_SECURITY_REJECTED

```
#define BOOTLOADER_ERROR_SECURITY_REJECTED
```

Value:

```
(BOOTLOADER_ERROR_SECURITY_BASE | 0x04L)
```

Authentication did not check out.

Definition at line 180 of file platform/bootloader/api/btl_errorcode.h

Storage Driver Error Codes

Storage Driver Error Codes

Bootloader error codes returned by a storage driver.

Offset from BOOTLOADER_ERROR_STORAGE_BASE

Macros

#define BOOTLOADER_ERROR_STORAGE_INVALID_SLOT (BOOTLOADER_ERROR_STORAGE_BASE | 0x01L)
Invalid slot.

#define BOOTLOADER_ERROR_STORAGE_INVALID_ADDRESS (BOOTLOADER_ERROR_STORAGE_BASE | 0x02L)
Invalid address. Address not aligned/out of range.

#define BOOTLOADER_ERROR_STORAGE_NEEDS_ERASE (BOOTLOADER_ERROR_STORAGE_BASE | 0x03L)
The storage area needs to be erased before it can be used.

#define BOOTLOADER_ERROR_STORAGE_NEEDS_ALIGN (BOOTLOADER_ERROR_STORAGE_BASE | 0x04L)
The address or length needs to be aligned.

#define BOOTLOADER_ERROR_STORAGE_BOOTLOAD (BOOTLOADER_ERROR_STORAGE_BASE | 0x05L)
An error occured during bootload from storage.

#define BOOTLOADER_ERROR_STORAGE_NO_IMAGE (BOOTLOADER_ERROR_STORAGE_BASE | 0x06L)
There is no image in this storage slot.

#define BOOTLOADER_ERROR_STORAGE_CONTINUE (BOOTLOADER_ERROR_STORAGE_BASE | 0x07L)
Continue calling function.

#define BOOTLOADER_ERROR_STORAGE_GENERIC (BOOTLOADER_ERROR_STORAGE_BASE | 0x08L)
Generic storage error.

Macro Definition Documentation

BOOTLOADER_ERROR_STORAGE_INVALID_SLOT

```
#define BOOTLOADER_ERROR_STORAGE_INVALID_SLOT
```

Value:

```
(BOOTLOADER_ERROR_STORAGE_BASE | 0x01L)
```

Invalid slot.

Definition at line 109 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_STORAGE_INVALID_ADDRESS

```
#define BOOTLOADER_ERROR_STORAGE_INVALID_ADDRESS
```

Value:

```
(BOOTLOADER_ERROR_STORAGE_BASE|0x02L)
```

Invalid address. Address not aligned/out of range.

Definition at line 112 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_STORAGE_NEEDS_ERASE

```
#define BOOTLOADER_ERROR_STORAGE_NEEDS_ERASE
```

Value:

```
(BOOTLOADER_ERROR_STORAGE_BASE|0x03L)
```

The storage area needs to be erased before it can be used.

Definition at line 115 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_STORAGE_NEEDS_ALIGN

```
#define BOOTLOADER_ERROR_STORAGE_NEEDS_ALIGN
```

Value:

```
(BOOTLOADER_ERROR_STORAGE_BASE|0x04L)
```

The address or length needs to be aligned.

Definition at line 118 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_STORAGE_BOOTLOAD

```
#define BOOTLOADER_ERROR_STORAGE_BOOTLOAD
```

Value:

```
(BOOTLOADER_ERROR_STORAGE_BASE|0x05L)
```

An error occured during bootload from storage.

Definition at line 121 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_STORAGE_NO_IMAGE

```
#define BOOTLOADER_ERROR_STORAGE_NO_IMAGE
```

Value:

```
(BOOTLOADER_ERROR_STORAGE_BASE|0x06L)
```

There is no image in this storage slot.

Definition at line 124 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_STORAGE_CONTINUE

```
#define BOOTLOADER_ERROR_STORAGE_CONTINUE
```

Value:

```
(BOOTLOADER_ERROR_STORAGE_BASE|0x07L)
```

Continue calling function.

Definition at line 127 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_STORAGE_GENERIC

```
#define BOOTLOADER_ERROR_STORAGE_GENERIC
```

Value:

```
(BOOTLOADER_ERROR_STORAGE_BASE|0x08L)
```

Generic storage error.

Definition at line 130 of file platform/bootloader/api/btl_errorcode.h

UART Driver Error Codes

UART Driver Error Codes

Bootloader error codes returned by the UART driver.

Offset from BOOTLOADER_ERROR_UART_BASE

Macros

#define BOOTLOADER_ERROR_UART_UNINIT (BOOTLOADER_ERROR_UART_BASE | 0×01)
Operation not allowed because hardware has not been initialized.

#define BOOTLOADER_ERROR_UART_INIT (BOOTLOADER_ERROR_UART_BASE | 0×02)
Hardware fail during initialization.

#define BOOTLOADER_ERROR_UART_ARGUMENT (BOOTLOADER_ERROR_UART_BASE | 0×03)
Invalid argument.

#define BOOTLOADER_ERROR_UART_TIMEOUT (BOOTLOADER_ERROR_UART_BASE | 0×04)
Operation timed out.

#define BOOTLOADER_ERROR_UART_OVERFLOW (BOOTLOADER_ERROR_UART_BASE | 0×05)
Buffer overflow condition.

#define BOOTLOADER_ERROR_UART_BUSY (BOOTLOADER_ERROR_UART_BASE | 0×06)
Busy condition.

Macro Definition Documentation

BOOTLOADER_ERROR_UART_UNINIT

```
#define BOOTLOADER_ERROR_UART_UNINIT
```

Value:

```
(BOOTLOADER_ERROR_UART_BASE|0×01)
```

Operation not allowed because hardware has not been initialized.

Definition at line 338 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_UART_INIT

```
#define BOOTLOADER_ERROR_UART_INIT
```

Value:

```
(BOOTLOADER_ERROR_UART_BASE|0×02)
```

Hardware fail during initialization.

Definition at line 340 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_UART_ARGUMENT

```
#define BOOTLOADER_ERROR_UART_ARGUMENT
```

Value:

```
(BOOTLOADER_ERROR_UART_BASE | 0x03)
```

Invalid argument.

Definition at line 342 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_UART_TIMEOUT

```
#define BOOTLOADER_ERROR_UART_TIMEOUT
```

Value:

```
(BOOTLOADER_ERROR_UART_BASE | 0x04)
```

Operation timed out.

Definition at line 344 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_UART_OVERFLOW

```
#define BOOTLOADER_ERROR_UART_OVERFLOW
```

Value:

```
(BOOTLOADER_ERROR_UART_BASE | 0x05)
```

Buffer overflow condition.

Definition at line 346 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_UART_BUSY

```
#define BOOTLOADER_ERROR_UART_BUSY
```

Value:

```
(BOOTLOADER_ERROR_UART_BASE | 0x06)
```

Busy condition.

Definition at line 348 of file platform/bootloader/api/btl_errorcode.h

XMODEM Error Codes

XMODEM Error Codes

Bootloader error codes returned by the XMODEM parser.

Offset from BOOTLOADER_ERROR_XMODEM_BASE

Macros

#define BOOTLOADER_ERROR_XMODEM_CRCL (BOOTLOADER_ERROR_XMODEM_BASE | 0x01L)
Could not verify lower CRC byte.

#define BOOTLOADER_ERROR_XMODEM_CRCH (BOOTLOADER_ERROR_XMODEM_BASE | 0x02L)
Could not verify upper CRC byte.

#define BOOTLOADER_ERROR_XMODEM_NO_SOH (BOOTLOADER_ERROR_XMODEM_BASE | 0x03L)
No start of header found.

#define BOOTLOADER_ERROR_XMODEM_PKTNUM (BOOTLOADER_ERROR_XMODEM_BASE | 0x04L)
Packet number doesn't match its inverse.

#define BOOTLOADER_ERROR_XMODEM_PKTSEQ (BOOTLOADER_ERROR_XMODEM_BASE | 0x05L)
Packet number error (unexpected sequence)

#define BOOTLOADER_ERROR_XMODEM_PKTDUP (BOOTLOADER_ERROR_XMODEM_BASE | 0x06L)
Packet number error (duplicate)

#define BOOTLOADER_ERROR_XMODEM_DONE (BOOTLOADER_ERROR_XMODEM_BASE | 0x07L)
Transfer is done (Technically not an error)

#define BOOTLOADER_ERROR_XMODEM_CANCEL (BOOTLOADER_ERROR_XMODEM_BASE | 0x08L)
Transfer is canceled.

Macro Definition Documentation

BOOTLOADER_ERROR_XMODEM_CRCL

```
#define BOOTLOADER_ERROR_XMODEM_CRCL
```

Value:

```
(BOOTLOADER_ERROR_XMODEM_BASE | 0x01L)
```

Could not verify lower CRC byte.

Definition at line 223 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_XMODEM_CRCH

```
#define BOOTLOADER_ERROR_XMODEM_CRCH
```

Value:

```
(BOOTLOADER_ERROR_XMODEM_BASE|0x02L)
```

Could not verify upper CRC byte.

Definition at line 226 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_XMODEM_NO_SOH

```
#define BOOTLOADER_ERROR_XMODEM_NO_SOH
```

Value:

```
(BOOTLOADER_ERROR_XMODEM_BASE|0x03L)
```

No start of header found.

Definition at line 229 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_XMODEM_PKTNUM

```
#define BOOTLOADER_ERROR_XMODEM_PKTNUM
```

Value:

```
(BOOTLOADER_ERROR_XMODEM_BASE|0x04L)
```

Packet number doesn't match its inverse.

Definition at line 232 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_XMODEM_PKTSEQ

```
#define BOOTLOADER_ERROR_XMODEM_PKTSEQ
```

Value:

```
(BOOTLOADER_ERROR_XMODEM_BASE|0x05L)
```

Packet number error (unexpected sequence)

Definition at line 235 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_XMODEM_PKTDUP

```
#define BOOTLOADER_ERROR_XMODEM_PKTDUP
```

Value:

```
(BOOTLOADER_ERROR_XMODEM_BASE|0x06L)
```

Packet number error (duplicate)

Definition at line 238 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_XMODEM_DONE

```
#define BOOTLOADER_ERROR_XMODEM_DONE
```

Value:

```
(BOOTLOADER_ERROR_XMODEM_BASE|0x07L)
```

Transfer is done (Technically not an error)

Definition at line 241 of file platform/bootloader/api/btl_errorcode.h

BOOTLOADER_ERROR_XMODEM_CANCEL

```
#define BOOTLOADER_ERROR_XMODEM_CANCEL
```

Value:

```
(BOOTLOADER_ERROR_XMODEM_BASE|0x08L)
```

Transfer is canceled.

Definition at line 244 of file platform/bootloader/api/btl_errorcode.h