

Secure Bootloader with Rollback Feature

1

Generated by Doxygen 1.9.1

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	5
3.1 App_INFO_t Struct Reference	5
3.1.1 Member Data Documentation	5
3.1.1.1 app_size	5
3.1.1.2 app_validity	5
3.1.1.3 hash	5
3.2 BL_INFO_t Struct Reference	6
3.2.1 Member Data Documentation	6
3.2.1.1 app	6
3.2.1.2 update	6
3.3 CRC_Handle_t Struct Reference	6
3.3.1 Member Data Documentation	7
3.3.1.1 crc_initval	7
3.3.1.2 crc_poly	7
3.3.1.3 crc_reg	7
3.4 Update_INFO_t Struct Reference	7
3.4.1 Member Data Documentation	7
3.4.1.1 hash	8
3.4.1.2 update_size	8
3.4.1.3 updated	8
3.5 Version_ID_t Struct Reference	8
3.5.1 Member Data Documentation	8
3.5.1.1 major_version	8
3.5.1.2 minor_version	9
3.5.1.3 patch_version	9
3.5.1.4 vendor_id	9
4 File Documentation	11
4.1 bootloader/bl_typedef.h File Reference	11
4.1.1 Detailed Description	12
4.1.2 Macro Definition Documentation	12
4.1.2.1 APP_NOT_FOUND	12
4.1.2.2 APP_NOT_VERIFIED	13
4.1.2.3 APP_VERIFIED	13
4.1.2.4 BL_CRC_SIZE	13
4.1.2.5 CRC_VERIFY_FAIL	13
4.1.2.6 CRC_VERIFY_PASS	13

4.1.2.7 NOR_APP_NOT_UPDATE_FOUND	13
4.1.2.8 PAGE0_ADDRESS	13
4.1.2.9 PAGE_SIZE	13
4.1.2.10 STM32F103C8T6_PAGES_NUM	14
4.1.2.11 UPDATE_FOUND	14
4.1.2.12 VERIFICATION_ERROR	14
4.1.3 Typedef Documentation	14
4.1.3.1 BL_CMD	14
4.1.3.2 BL_STATUS	14
4.1.4 Enumeration Type Documentation	14
4.1.4.1 BL_CMD	14
4.1.4.2 BL_STATUS	15
4.2 bootloader/bootloader.c File Reference	15
4.2.1 Detailed Description	16
4.2.2 Function Documentation	16
4.2.2.1 BL_DeInit()	16
4.2.2.2 BL_FetchUARTCommand()	16
4.2.2.3 BL_Init()	17
4.2.2.4 BL_InstallApplication()	17
4.2.2.5 BL_JumpToApplication()	18
4.2.2.6 BL_JumpToApplication_Wrapper()	18
4.2.2.7 BL_SendAck()	18
4.2.2.8 BL_SendMSG()	19
4.2.2.9 BL_VerifyApplication()	19
4.2.2.10 BL_WriteAppsInfoToFlash()	20
4.2.2.11 BL_WriteToFlash()	20
4.2.2.12 Bootloader_MemRead()	21
4.2.2.13 generate_random_number()	21
4.2.2.14 Get_CurrentBL_Info()	21
4.2.2.15 Set_ValidUpdate()	21
4.2.3 Variable Documentation	22
4.2.3.1 bl_info_ptr	22
4.3 bootloader/bootloader.h File Reference	22
4.3.1 Detailed Description	23
4.3.2 Function Documentation	24
4.3.2.1 BL_DeInit()	24
4.3.2.2 BL_FetchUARTCommand()	24
4.3.2.3 BL_Init()	24
4.3.2.4 BL_InstallApplication()	25
4.3.2.5 BL_JumpToApplication()	25
4.3.2.6 BL_JumptoVerifiedAPP()	26
4.3.2.7 BL_Read_APPS_INFO()	26

4.3.2.8 BL_SendAck()	26
4.3.2.9 BL_SendMSG()	26
4.3.2.10 BL_VerifyApplication()	27
4.3.2.11 BL_WriteAppsInfoToFlash()	27
4.3.2.12 BL_WriteToFlash()	28
4.3.2.13 Get_CurrentBL_Info()	28
4.3.3 Variable Documentation	28
4.3.3.1 hcrc	28
4.3.3.2 huart1	29
4.3.3.3 huart2	29
4.4 crc/crc.c File Reference	29
4.4.1 Function Documentation	29
4.4.1.1 CRC32_Calculate()	30
4.4.1.2 CRC32_Init()	30
4.4.1.3 CRC32_ResetCRC()	31
4.5 crc/crc.h File Reference	31
4.5.1 Function Documentation	32
4.5.1.1 CRC32_Calculate()	32
4.5.1.2 CRC32_Init()	33
4.5.1.3 CRC32_ResetCRC()	33
4.6 crc/crc_typedefs.h File Reference	34
4.6.1 Macro Definition Documentation	35
4.6.1.1 DEFAULT_INIT_VAL	35
4.6.1.2 DEFAULT_POLY_VAL	35
4.7 main.c File Reference	35
4.7.1 Detailed Description	36
4.7.2 Function Documentation	36
4.7.2.1 Error_Handler()	36
4.7.2.2 main()	36
4.7.2.3 SystemClock_Config()	37
4.7.3 Variable Documentation	38
4.7.3.1 bl_info_ptr	38
4.7.3.2 hcrc	38
4.7.3.3 huart1	38
4.7.3.4 huart2	38
Index	39

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

App_INFO_t	5
BL_INFO_t	6
CRC_Handle_t	6
Update_INFO_t	7
Version_ID_t	8

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

main.c	: Main program body	35
bootloader/ bl_typedef.h	: Header file for typedefs for the STM32F103C8T6 Bootloader	11
bootloader/ bootloader.c	: Source file for the STM32F103C8T6 Bootloader	15
bootloader/ bootloader.h	: Header file for the STM32F103C8T6 Bootloader	22
crc/ crc.c	29
crc/ crc.h	31
crc/ crc_typedefs.h	34

Chapter 3

Class Documentation

3.1 App_INFO_t Struct Reference

```
#include <bl_typedef.h>
```

Public Attributes

- uint8_t [hash](#) [32]
- uint32_t [app_size](#)
- uint8_t [app_validity](#)

3.1.1 Member Data Documentation

3.1.1.1 app_size

```
uint32_t App_INFO_t::app_size
```

3.1.1.2 app_validity

```
uint8_t App_INFO_t::app_validity
```

3.1.1.3 hash

```
uint8_t App_INFO_t::hash[32]
```

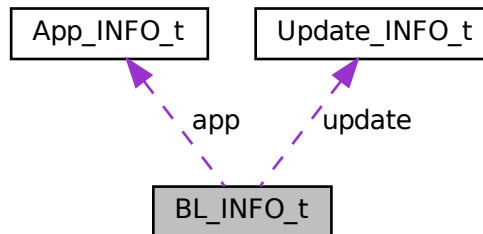
The documentation for this struct was generated from the following file:

- bootloader/[bl_typedef.h](#)

3.2 BL_INFO_t Struct Reference

```
#include <bl_typedef.h>
```

Collaboration diagram for BL_INFO_t:



Public Attributes

- [App_INFO_t app](#)
- [Update_INFO_t update](#)

3.2.1 Member Data Documentation

3.2.1.1 app

```
App\_INFO\_t BL_INFO_t::app
```

3.2.1.2 update

```
Update\_INFO\_t BL_INFO_t::update
```

The documentation for this struct was generated from the following file:

- [bootloader/bl_typedef.h](#)

3.3 CRC_Handle_t Struct Reference

```
#include <crc_typedefs.h>
```

Public Attributes

- uint32_t [crc_poly](#)
- uint32_t [crc_initval](#)
- uint32_t [crc_reg](#)

3.3.1 Member Data Documentation

3.3.1.1 crc_initval

```
uint32_t CRC_Handle_t::crc_initval
```

3.3.1.2 crc_poly

```
uint32_t CRC_Handle_t::crc_poly
```

3.3.1.3 crc_reg

```
uint32_t CRC_Handle_t::crc_reg
```

The documentation for this struct was generated from the following file:

- [crc/crc_typedefs.h](#)

3.4 Update_INFO_t Struct Reference

```
#include <bl_typedef.h>
```

Public Attributes

- uint8_t [hash](#) [32]
- uint32_t [update_size](#)
- uint8_t [updated](#)

3.4.1 Member Data Documentation

3.4.1.1 hash

```
uint8_t Update_INFO_t::hash[32]
```

3.4.1.2 update_size

```
uint32_t Update_INFO_t::update_size
```

3.4.1.3 updated

```
uint8_t Update_INFO_t::updated
```

The documentation for this struct was generated from the following file:

- bootloader/[bl_typedef.h](#)

3.5 Version_ID_t Struct Reference

```
#include <bl_typedef.h>
```

Public Attributes

- uint8_t [vendor_id](#)
- uint8_t [major_version](#)
- uint8_t [minor_version](#)
- uint8_t [patch_version](#)

3.5.1 Member Data Documentation

3.5.1.1 major_version

```
uint8_t Version_ID_t::major_version
```

3.5.1.2 minor_version

```
uint8_t Version_ID_t::minor_version
```

3.5.1.3 patch_version

```
uint8_t Version_ID_t::patch_version
```

3.5.1.4 vendor_id

```
uint8_t Version_ID_t::vendor_id
```

The documentation for this struct was generated from the following file:

- bootloader/[bl_typedef.h](#)

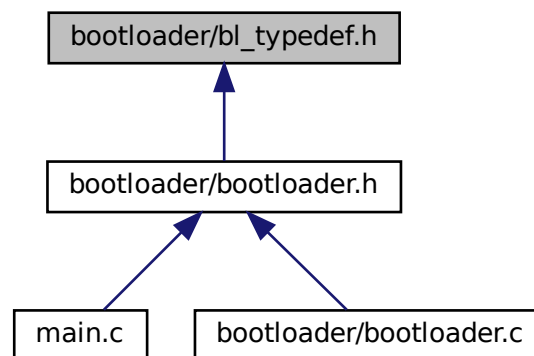
Chapter 4

File Documentation

4.1 bootloader/bl_typedef.h File Reference

: Header file for typedefs for the STM32F103C8T6 Bootloader.

This graph shows which files directly or indirectly include this file:



Classes

- struct [App_INFO_t](#)
- struct [Update_INFO_t](#)
- struct [BL_INFO_t](#)
- struct [Version_ID_t](#)

Macros

- `#define CRC_VERIFY_PASS 0x01`
- `#define CRC_VERIFY_FAIL 0x00`
- `#define APP_VERIFIED 0x00`
- `#define APP_NOT_VERIFIED 0x01`
- `#define APP_NOT_FOUND 0x02`
- `#define UPDATE_FOUND 0x03`
- `#define NOR_APP_NOT_UPDATE_FOUND 0x04`
- `#define VERIFICATION_ERROR 0x05`
- `#define STM32F103C8T6_PAGES_NUM 64`
- `#define PAGE0_ADDRESS 0x08000000`
- `#define PAGE_SIZE (1024u)`
- `#define BL_CRC_SIZE 0x04u`

Typedefs

- `typedef enum BL_CMD BL_CMD`
- `typedef enum BL_STATUS BL_STATUS`

Enumerations

- `enum BL_CMD {
 CBL_GET_VER_CMD = 1 , CBL_GET_CID_CMD , CBL_MEM_READ_CMD , CBL_FLASH_ERASE_CMD ,
 CBL_GET_RDP_STATUS_CMD , CBL_GET_APP_UPDATE , CBL_JUMPTOAPP , CBL_GET_HELP_CMD
}`
- `enum BL_STATUS { BL_ACK = 0xA5 , BL_NACK = 0x5A }`

4.1.1 Detailed Description

: Header file for typedefs for the STM32F103C8T6 Bootloader.

Attention

2024 - Abokhalil. All rights reserved.

4.1.2 Macro Definition Documentation

4.1.2.1 APP_NOT_FOUND

```
#define APP_NOT_FOUND 0x02
```

4.1.2.2 APP_NOT_VERIFIED

```
#define APP_NOT_VERIFIED 0x01
```

4.1.2.3 APP_VERIFIED

```
#define APP_VERIFIED 0x00
```

4.1.2.4 BL_CRC_SIZE

```
#define BL_CRC_SIZE 0x04u
```

4.1.2.5 CRC_VERIFY_FAIL

```
#define CRC_VERIFY_FAIL 0x00
```

4.1.2.6 CRC_VERIFY_PASS

```
#define CRC_VERIFY_PASS 0x01
```

4.1.2.7 NOR_APP_NOT_UPDATE_FOUND

```
#define NOR_APP_NOT_UPDATE_FOUND 0x04
```

4.1.2.8 PAGE0_ADDRESS

```
#define PAGE0_ADDRESS 0x08000000
```

4.1.2.9 PAGE_SIZE

```
#define PAGE_SIZE (1024u)
```

4.1.2.10 STM32F103C8T6_PAGES_NUM

```
#define STM32F103C8T6_PAGES_NUM 64
```

4.1.2.11 UPDATE_FOUND

```
#define UPDATE_FOUND 0x03
```

4.1.2.12 VERIFICATION_ERROR

```
#define VERIFICATION_ERROR 0x05
```

4.1.3 Typedef Documentation

4.1.3.1 BL_CMD

```
typedef enum BL_CMD BL_CMD
```

4.1.3.2 BL_STATUS

```
typedef enum BL_STATUS BL_STATUS
```

4.1.4 Enumeration Type Documentation

4.1.4.1 BL_CMD

```
enum BL_CMD
```

Enumerator

CBL_GET_VER_CMD	
CBL_GET_CID_CMD	
CBL_MEM_READ_CMD	
CBL_FLASH_ERASE_CMD	
CBL_GET_RDP_STATUS_CMD	
CBL_GET_APP_UPDATE	
CBL_JUMPTOAPP	
CBL_GET_HELP_CMD	

4.1.4.2 BL_STATUS

enum [BL_STATUS](#)

Enumerator

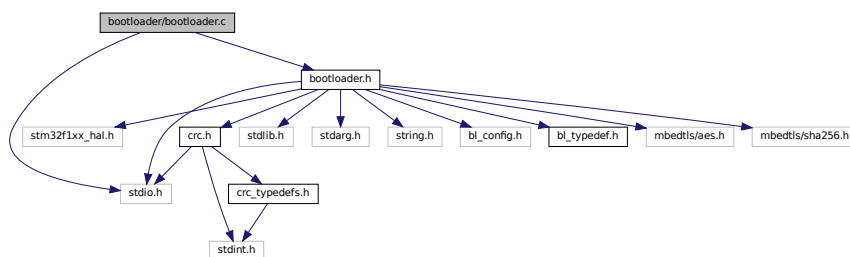
BL_ACK	
BL_NACK	

4.2 bootloader/bootloader.c File Reference

: Source file for the STM32F103C8T6 Bootloader.

```
#include "bootloader.h"
#include <stdio.h>
```

Include dependency graph for bootloader.c:



Functions

- void [BL_JumpToApplication](#) (uint8_t app_starting_page)
- void [BL_JumpToApplication_Wrapper](#) (uint8_t *hostbuffer)
- void [Bootloader_MemRead](#) (uint32_t address, size_t len)
Reads data from the specified memory address after authorization.
- uint8_t [Set_ValidUpdate](#) ([Update_INFO_t](#) *update)
- [BL_STATUS](#) [BL_WriteToFlash](#) (uint8_t page, uint8_t *data, uint32_t size)
- [BL_STATUS](#) [BL_WriteAppsInfoToFlash](#) ([BL_INFO_t](#) *info)
Writes application info structure to flash memory.
- [BL_STATUS](#) [BL_FetchUARTCommand](#) (void)
Fetches the UART command sent to the bootloader.
- uint8_t [BL_VerifyApplication](#) (void)
Verifies if an application is present and valid in memory.
- void [Get_CurrentBL_Info](#) ([BL_INFO_t](#) *dest)
Retrieves the current bootloader information.
- void [BL_DeInit](#) ()

Deinitializes all peripherals used by the bootloader.

- void [BL_SendAck](#) (uint8_t ack)

Sends an acknowledgment byte via UART.

- uint8_t [BL_InstallApplication](#) (void)

Installs the new application from the update location in Flash memory to the application location.

- void [BL_Init](#) ()
- void [BL_SendMSG](#) (UART_HandleTypeDef *huart, char *format,...)
- void [generate_random_number](#) (uint8_t *output)

Generates a random number using the system tick and stores it in the output.

Variables

- [BL_INFO_t](#) * [bl_info_ptr](#) = ([BL_INFO_t](#)*) ([BL_INFO_PAGE](#) * [PAGE_SIZE](#) + [PAGE0_ADDRESS](#))

4.2.1 Detailed Description

: Source file for the STM32F103C8T6 Bootloader.

Attention

2024 - Abokhalil. All rights reserved.

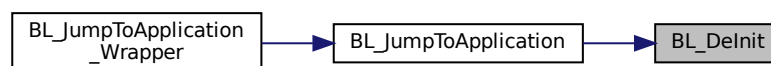
4.2.2 Function Documentation

4.2.2.1 BL_DeInit()

```
void BL_DeInit ( )
```

Deinitializes all peripherals used by the bootloader.

Here is the caller graph for this function:



4.2.2.2 BL_FetchUARTCommand()

```
BL\_STATUS BL_FetchUARTCommand (
    void )
```

Fetches the UART command sent to the bootloader.

Return values

<i>BL_STATUS</i>	Status of the bootloader command (ACK/NACK).
------------------	----------------------------------------------

4.2.2.3 BL_Init()

```
void BL_Init ( )
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.2.2.4 BL_InstallApplication()

```
uint8_t BL_InstallApplication (
    void )
```

Installs the new application from the update location in Flash memory to the application location.

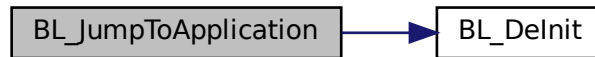
Return values

<i>uint8_t</i>	Returns 0 if installation is successful, or 1 if there is an error.
----------------	---------------------------------------------------------------------

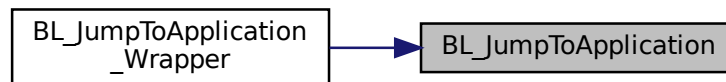
4.2.2.5 BL_JumpToApplication()

```
void BL_JumpToApplication (
    uint8_t app_starting_page )
```

Here is the call graph for this function:



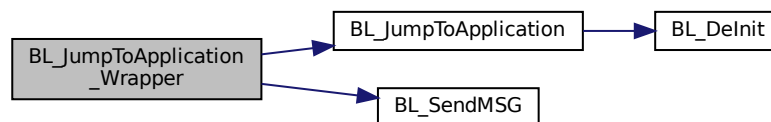
Here is the caller graph for this function:



4.2.2.6 BL_JumpToApplication_Wrapper()

```
void BL_JumpToApplication_Wrapper (
    uint8_t * hostbuffer )
```

Here is the call graph for this function:



4.2.2.7 BL_SendAck()

```
void BL_SendAck (
    uint8_t ack )
```

Sends an acknowledgment byte via UART.

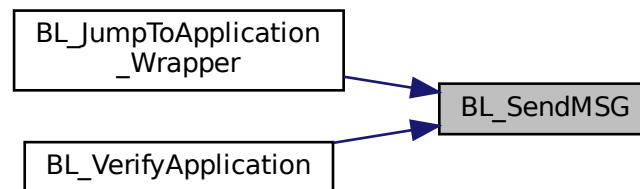
Parameters

<i>ack</i>	The acknowledgment byte to be sent.
------------	-------------------------------------

4.2.2.8 BL_SendMSG()

```
void BL_SendMSG (
    UART_HandleTypeDef * huart,
    char * format,
    ... )
```

Here is the caller graph for this function:



4.2.2.9 BL_VerifyApplication()

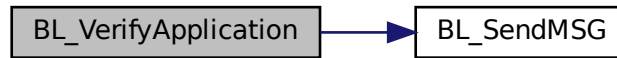
```
uint8_t BL_VerifyApplication (
    void )
```

Verifies if an application is present and valid in memory.

Return values

<i>uint8_t</i>	Status of the verification.
----------------	-----------------------------

Here is the call graph for this function:



4.2.2.10 BL_WriteAppsInfoToFlash()

```
BL_STATUS BL_WriteAppsInfoToFlash (
    BL_INFO_t * info )
```

Writes application info structure to flash memory.

Parameters

<i>info</i>	Pointer to BL_INFO_t structure containing app info to write.
-------------	------------------------------------------------------------------------------

Here is the caller graph for this function:



4.2.2.11 BL_WriteToFlash()

```
BL_STATUS BL_WriteToFlash (
    uint8_t page,
    uint8_t * data,
    uint32_t size )
```

4.2.2.12 Bootloader_MemRead()

```
void Bootloader_MemRead (
    uint32_t address,
    size_t len )
```

Reads data from the specified memory address after authorization.

Parameters

<i>address</i>	Starting address to read from.
<i>len</i>	Number of bytes to read.

4.2.2.13 generate_random_number()

```
void generate_random_number (
    uint8_t * output )
```

Generates a random number using the system tick and stores it in the output.

Parameters

<i>output</i>	Pointer to store the generated random number (16 bytes).
---------------	----------------------------------------------------------

4.2.2.14 Get_CurrentBL_Info()

```
void Get_CurrentBL_Info (
    BL_INFO_t * dest )
```

Retrieves the current bootloader information.

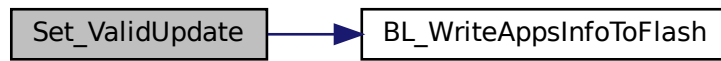
Parameters

<i>dest</i>	Pointer to the destination structure to store bootloader info.
-------------	----------------------------------------------------------------

4.2.2.15 Set_ValidUpdate()

```
uint8_t Set_ValidUpdate (
    Update_INFO_t * update )
```

Here is the call graph for this function:



4.2.3 Variable Documentation

4.2.3.1 bl_info_ptr

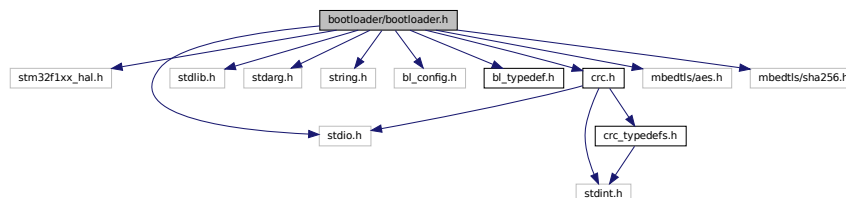
```
BL_INFO_t* bl_info_ptr = (BL_INFO_t*) (BL_INFO_PAGE * PAGE_SIZE + PAGE0_ADDRESS)
```

4.3 bootloader/bootloader.h File Reference

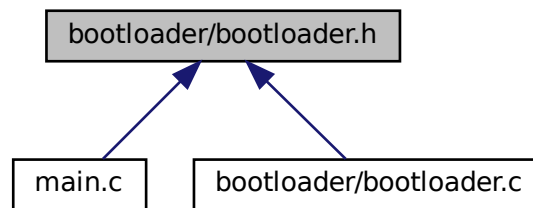
: Header file for the STM32F103C8T6 Bootloader.

```
#include "stm32f1xx_hal.h"
#include "stdio.h"
#include "stdlib.h"
#include "stdarg.h"
#include "string.h"
#include "bl_config.h"
#include "bl_typedef.h"
#include "crc.h"
#include "mbedtls/aes.h"
#include "mbedtls/sha256.h"
```

Include dependency graph for bootloader.h:



This graph shows which files directly or indirectly include this file:



Functions

- [BL_STATUS BL_WriteToFlash](#) (uint8_t page, uint8_t *data, uint32_t size)
- void [BL_Init](#) ()
- void [BL_SendMSG](#) (UART_HandleTypeDef *huart, char *format,...)
- [BL_STATUS BL_FetchUARTCommand](#) ()
Fetches the UART command sent to the bootloader.
- [BL_STATUS BL_Read_APPS_INFO](#) ()
- void [BL_SendAck](#) (uint8_t ack)
Sends an acknowledgment byte via UART.
- uint8_t [BL_VerifyApplication](#) (void)
Verifies if an application is present and valid in memory.
- [BL_STATUS BL_JumptoVerifiedAPP](#) ()
- void [BL_JumpToApplication](#) (uint8_t app_starting_page)
- [BL_STATUS BL_WriteAppsInfoToFlash](#) (BL_INFO_t *info)
Writes application info structure to flash memory.
- void [Get_CurrentBL_Info](#) (BL_INFO_t *dest)
Retrieves the current bootloader information.
- uint8_t [BL_InstallApplication](#) ()
Installs the new application from the update location in Flash memory to the application location.
- void [BL_DeInit](#) ()
Deinitializes all peripherals used by the bootloader.

Variables

- UART_HandleTypeDef [huart2](#)
- UART_HandleTypeDef [huart1](#)
- [CRC_Handle_t hcrc](#)

4.3.1 Detailed Description

: Header file for the STM32F103C8T6 Bootloader.

Attention

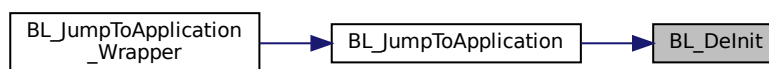
4.3.2 Function Documentation

4.3.2.1 BL_DeInit()

```
void BL_DeInit ( )
```

Deinitializes all peripherals used by the bootloader.

Here is the caller graph for this function:



4.3.2.2 BL_FetchUARTCommand()

```
BL_STATUS BL_FetchUARTCommand (
    void )
```

Fetches the UART command sent to the bootloader.

Return values

<i>BL_STATUS</i>	Status of the bootloader command (ACK/NACK).
------------------	----------------------------------------------

4.3.2.3 BL_Init()

```
void BL_Init ( )
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.3.2.4 BL_InstallApplication()

```
uint8_t BL_InstallApplication (
    void )
```

Installs the new application from the update location in Flash memory to the application location.

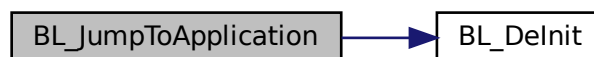
Return values

<i>uint8_t</i>	Returns 0 if installation is successful, or 1 if there is an error.
----------------	---------------------------------------------------------------------

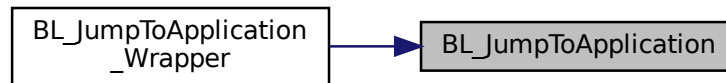
4.3.2.5 BL_JumpToApplication()

```
void BL_JumpToApplication (
    uint8_t app_starting_page )
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.3.2.6 BL_JumptoVerifiedAPP()

```
BL_STATUS BL_JumptoVerifiedAPP ( )
```

4.3.2.7 BL_Read_APPS_INFO()

```
BL_STATUS BL_Read_APPS_INFO ( )
```

4.3.2.8 BL_SendAck()

```
void BL_SendAck (
    uint8_t ack )
```

Sends an acknowledgment byte via UART.

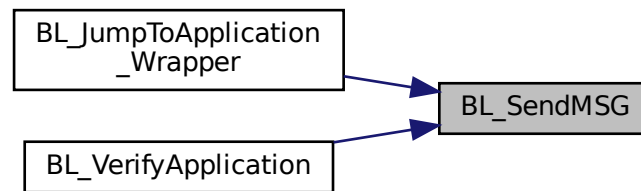
Parameters

<i>ack</i>	The acknowledgment byte to be sent.
------------	-------------------------------------

4.3.2.9 BL_SendMSG()

```
void BL_SendMSG (
    UART_HandleTypeDef * huart,
    char * format,
    ... )
```


Here is the caller graph for this function:



4.3.2.10 BL_VerifyApplication()

```
uint8_t BL_VerifyApplication (
    void )
```

Verifies if an application is present and valid in memory.

Return values

<i>uint8_t</i>	Status of the verification.
----------------	-----------------------------

Here is the call graph for this function:



4.3.2.11 BL_WriteAppsInfoToFlash()

```
BL_STATUS BL_WriteAppsInfoToFlash (
    BL_INFO_t * info )
```

Writes application info structure to flash memory.

Parameters

<i>info</i>	Pointer to BL_INFO_t structure containing app info to write.
-------------	------------------------------------------------------------------------------

Here is the caller graph for this function:



4.3.2.12 BL_WriteToFlash()

```

BL_STATUS BL_WriteToFlash (
    uint8_t page,
    uint8_t * data,
    uint32_t size )
  
```

4.3.2.13 Get_CurrentBL_Info()

```

void Get_CurrentBL_Info (
    BL_INFO_t * dest )
  
```

Retrieves the current bootloader information.

Parameters

<i>dest</i>	Pointer to the destination structure to store bootloader info.
-------------	----------------------------------------------------------------

4.3.3 Variable Documentation

4.3.3.1 hcrc

```

CRC_Handle_t hcrc [extern]
  
```

4.3.3.2 huart1

```
UART_HandleTypeDef huart1 [extern]
```

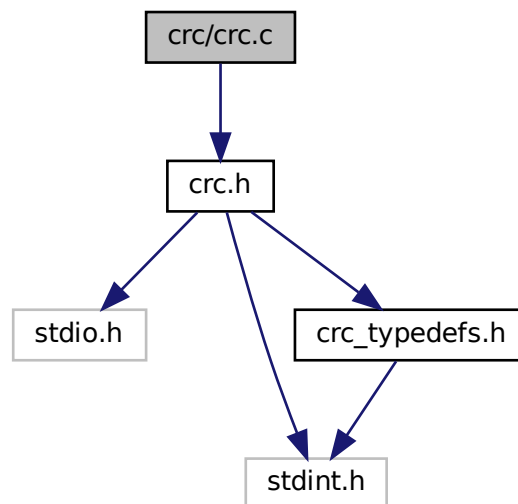
4.3.3.3 huart2

```
UART_HandleTypeDef huart2 [extern]
```

4.4 crc/crc.c File Reference

```
#include "crc.h"
```

Include dependency graph for crc.c:



Functions

- void `CRC32_Init` (`CRC_Handle_t` *hrc)
Initializes the CRC module by setting the CRC register to the initial value.
- uint32_t `CRC32_Calculate` (`CRC_Handle_t` *hrc, uint8_t *data, uint32_t length)
Calculates the CRC32 for the given data buffer.
- void `CRC32_ResetCRC` (`CRC_Handle_t` *hrc)
Resets the CRC register to the initial value.

4.4.1 Function Documentation

4.4.1.1 CRC32_Calculate()

```
uint32_t CRC32_Calculate (
    CRC_Handle_t * hCRC,
    uint8_t * data,
    uint32_t length )
```

Calculates the CRC32 for the given data buffer.

Parameters

<i>hCRC</i>	Pointer to the CRC handle structure, which contains the CRC configuration and state.
<i>data</i>	Pointer to the data buffer for which the CRC is to be calculated.
<i>length</i>	Length of the data buffer.

Return values

<i>The</i>	computed CRC32 value.
------------	-----------------------

4.4.1.2 CRC32_Init()

```
void CRC32_Init (
    CRC_Handle_t * hCRC )
```

Initializes the CRC module by setting the CRC register to the initial value.

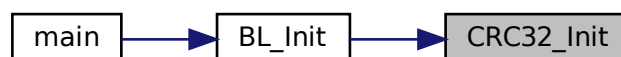
Parameters

<i>hCRC</i>	Pointer to the CRC handle structure, which contains the CRC configuration.
-------------	----------------------------------------------------------------------------

Return values

<i>The</i>	initial CRC value set in the CRC register.
------------	--------------------------------------------

Here is the caller graph for this function:



4.4.1.3 CRC32_ResetCRC()

```
void CRC32_ResetCRC (
    CRC_Handle_t * hCRC )
```

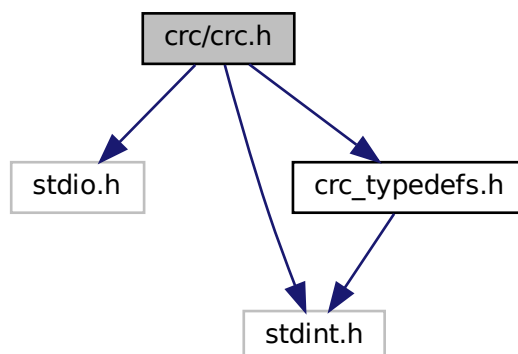
Resets the CRC register to the initial value.

Parameters

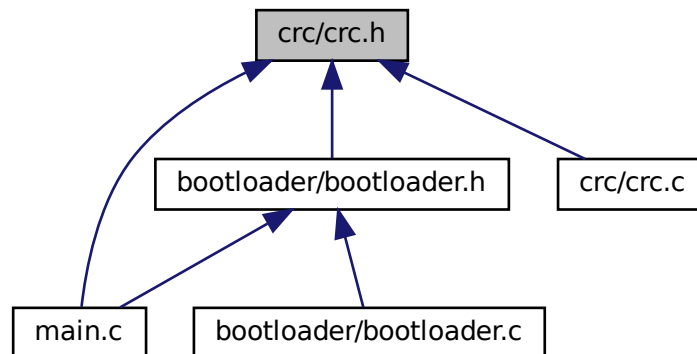
<i>hCRC</i>	Pointer to the CRC handle structure.
-------------	--------------------------------------

4.5 crc/crc.h File Reference

```
#include <stdio.h>
#include <stdint.h>
#include "crc_typedefs.h"
Include dependency graph for crc.h:
```



This graph shows which files directly or indirectly include this file:



Functions

- void [CRC32_Init](#) ([CRC_Handle_t](#) **hcrc*)
Initializes the CRC module by setting the CRC register to the initial value.
- [uint32_t CRC32_Calculate](#) ([CRC_Handle_t](#) **hcrc*, [uint8_t](#) **data*, [uint32_t](#) *length*)
Calculates the CRC32 for the given data buffer.
- void [CRC32_ResetCRC](#) ([CRC_Handle_t](#) **hcrc*)
Resets the CRC register to the initial value.

4.5.1 Function Documentation

4.5.1.1 CRC32_Calculate()

```

uint32_t CRC32_Calculate (
    CRC\_Handle\_t * hcrc,
    uint8\_t * data,
    uint32\_t length )

```

Calculates the CRC32 for the given data buffer.

Parameters

<i>hcrc</i>	Pointer to the CRC handle structure, which contains the CRC configuration and state.
<i>data</i>	Pointer to the data buffer for which the CRC is to be calculated.
<i>length</i>	Length of the data buffer.

Return values

<i>The</i>	computed CRC32 value.
------------	-----------------------

4.5.1.2 CRC32_Init()

```
void CRC32_Init (
    CRC_Handle_t * hcrc )
```

Initializes the CRC module by setting the CRC register to the initial value.

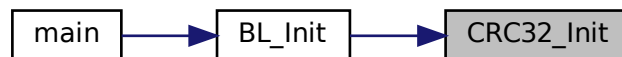
Parameters

<i>hcrc</i>	Pointer to the CRC handle structure, which contains the CRC configuration.
-------------	----------------------------------------------------------------------------

Return values

<i>The</i>	initial CRC value set in the CRC register.
------------	--------------------------------------------

Here is the caller graph for this function:



4.5.1.3 CRC32_ResetCRC()

```
void CRC32_ResetCRC (
    CRC_Handle_t * hcrc )
```

Resets the CRC register to the initial value.

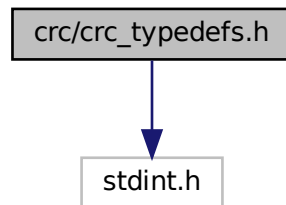
Parameters

<i>hcrc</i>	Pointer to the CRC handle structure.
-------------	--------------------------------------

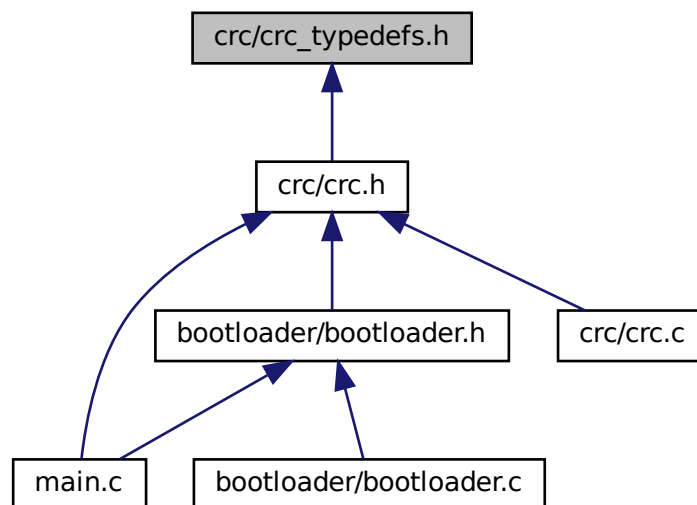
4.6 crc/crc_typedefs.h File Reference

```
#include <stdint.h>
```

Include dependency graph for `crc_typedefs.h`:



This graph shows which files directly or indirectly include this file:



Classes

- struct [CRC_Handle_t](#)

Macros

- #define [DEFAULT_POLY_VAL](#) 0xEDB88320
- #define [DEFAULT_INIT_VAL](#) 0xFFFFFFFF

4.6.1 Macro Definition Documentation

4.6.1.1 DEFAULT_INIT_VAL

```
#define DEFAULT_INIT_VAL 0xFFFFFFFF
```

4.6.1.2 DEFAULT_POLY_VAL

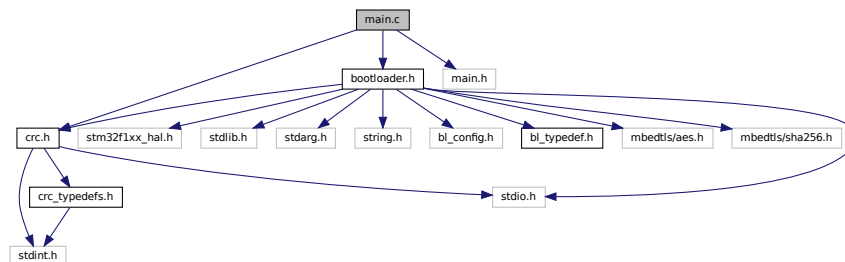
```
#define DEFAULT_POLY_VAL 0xEDB88320
```

4.7 main.c File Reference

: Main program body

```
#include "crc.h"
#include "bootloader.h"
#include "main.h"
```

Include dependency graph for main.c:



Functions

- void [SystemClock_Config](#) (void)
System Clock Configuration.
- int [main](#) (void)
The application entry point.
- void [Error_Handler](#) (void)
This function is executed in case of error occurrence.

Variables

- [CRC_Handle_t](#) [hrcrc](#)
- [UART_HandleTypeDef](#) [huart1](#)
- [UART_HandleTypeDef](#) [huart2](#)
- [BL_INFO_t](#) * [bl_info_ptr](#)

4.7.1 Detailed Description

: Main program body

Attention

Copyright (c) 2024 STMicroelectronics. All rights reserved.

This software is licensed under terms that can be found in the LICENSE file in the root directory of this software component. If no LICENSE file comes with this software, it is provided AS-IS.

4.7.2 Function Documentation

4.7.2.1 Error_Handler()

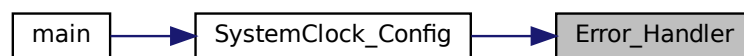
```
void Error_Handler (  
    void )
```

This function is executed in case of error occurrence.

Return values

None	
------	--

Here is the caller graph for this function:



4.7.2.2 main()

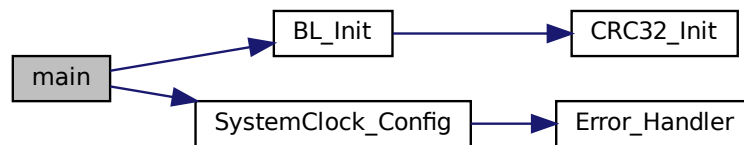
```
int main (  
    void )
```

The application entry point.

Return values

<i>int</i>	
------------	--

Here is the call graph for this function:



4.7.2.3 SystemClock_Config()

```
void SystemClock_Config (  
    void )
```

System Clock Configuration.

Return values

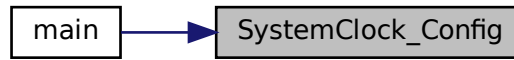
<i>None</i>	
-------------	--

Initializes the RCC Oscillators according to the specified parameters in the `RCC_OscInitTypeDef` structure.

Initializes the CPU, AHB and APB buses clocksHere is the call graph for this function:



Here is the caller graph for this function:



4.7.3 Variable Documentation

4.7.3.1 bl_info_ptr

```
BL_INFO_t* bl_info_ptr [extern]
```

4.7.3.2 hcrc

```
CRC_Handle_t hcrc
```

4.7.3.3 huart1

```
UART_HandleTypeDef huart1
```

4.7.3.4 huart2

```
UART_HandleTypeDef huart2
```

Index

- app
 - BL_INFO_t, [6](#)
- App_INFO_t, [5](#)
 - app_size, [5](#)
 - app_validity, [5](#)
 - hash, [5](#)
- APP_NOT_FOUND
 - bl_typedef.h, [12](#)
- APP_NOT_VERIFIED
 - bl_typedef.h, [12](#)
- app_size
 - App_INFO_t, [5](#)
- app_validity
 - App_INFO_t, [5](#)
- APP_VERIFIED
 - bl_typedef.h, [13](#)
- BL_ACK
 - bl_typedef.h, [15](#)
- BL_CMD
 - bl_typedef.h, [14](#)
- BL_CRC_SIZE
 - bl_typedef.h, [13](#)
- BL_Delnit
 - bootloader.c, [16](#)
 - bootloader.h, [24](#)
- BL_FetchUARTCommand
 - bootloader.c, [16](#)
 - bootloader.h, [24](#)
- bl_info_ptr
 - bootloader.c, [22](#)
 - main.c, [38](#)
- BL_INFO_t, [6](#)
 - app, [6](#)
 - update, [6](#)
- BL_Init
 - bootloader.c, [17](#)
 - bootloader.h, [24](#)
- BL_InstallApplication
 - bootloader.c, [17](#)
 - bootloader.h, [25](#)
- BL_JumpToApplication
 - bootloader.c, [17](#)
 - bootloader.h, [25](#)
- BL_JumpToApplication_Wrapper
 - bootloader.c, [18](#)
- BL_JumptoVerifiedAPP
 - bootloader.h, [26](#)
- BL_NACK
 - bl_typedef.h, [15](#)
- BL_Read_APPS_INFO
 - bootloader.h, [26](#)
- BL_SendAck
 - bootloader.c, [18](#)
 - bootloader.h, [26](#)
- BL_SendMSG
 - bootloader.c, [19](#)
 - bootloader.h, [26](#)
- BL_STATUS
 - bl_typedef.h, [14](#), [15](#)
- bl_typedef.h
 - APP_NOT_FOUND, [12](#)
 - APP_NOT_VERIFIED, [12](#)
 - APP_VERIFIED, [13](#)
 - BL_ACK, [15](#)
 - BL_CMD, [14](#)
 - BL_CRC_SIZE, [13](#)
 - BL_NACK, [15](#)
 - BL_STATUS, [14](#), [15](#)
 - CBL_FLASH_ERASE_CMD, [14](#)
 - CBL_GET_APP_UPDATE, [14](#)
 - CBL_GET_CID_CMD, [14](#)
 - CBL_GET_HELP_CMD, [14](#)
 - CBL_GET_RDP_STATUS_CMD, [14](#)
 - CBL_GET_VER_CMD, [14](#)
 - CBL_JUMPTOAPP, [14](#)
 - CBL_MEM_READ_CMD, [14](#)
 - CRC_VERIFY_FAIL, [13](#)
 - CRC_VERIFY_PASS, [13](#)
 - NOR_APP_NOT_UPDATE_FOUND, [13](#)
 - PAGE0_ADDRESS, [13](#)
 - PAGE_SIZE, [13](#)
 - STM32F103C8T6_PAGES_NUM, [13](#)
 - UPDATE_FOUND, [14](#)
 - VERIFICATION_ERROR, [14](#)
- BL_VerifyApplication
 - bootloader.c, [19](#)
 - bootloader.h, [27](#)
- BL_WriteAppsInfoToFlash
 - bootloader.c, [20](#)
 - bootloader.h, [27](#)
- BL_WriteToFlash
 - bootloader.c, [20](#)
 - bootloader.h, [28](#)
- bootloader.c
 - BL_Delnit, [16](#)
 - BL_FetchUARTCommand, [16](#)
 - bl_info_ptr, [22](#)
 - BL_Init, [17](#)

- BL_InstallApplication, [17](#)
- BL_JumpToApplication, [17](#)
- BL_JumpToApplication_Wrapper, [18](#)
- BL_SendAck, [18](#)
- BL_SendMSG, [19](#)
- BL_VerifyApplication, [19](#)
- BL_WriteAppsInfoToFlash, [20](#)
- BL_WriteToFlash, [20](#)
- Bootloader_MemRead, [20](#)
- generate_random_number, [21](#)
- Get_CurrentBL_Info, [21](#)
- Set_ValidUpdate, [21](#)
- bootloader.h
 - BL_DeInit, [24](#)
 - BL_FetchUARTCommand, [24](#)
 - BL_Init, [24](#)
 - BL_InstallApplication, [25](#)
 - BL_JumpToApplication, [25](#)
 - BL_JumptoVerifiedAPP, [26](#)
 - BL_Read_APPS_INFO, [26](#)
 - BL_SendAck, [26](#)
 - BL_SendMSG, [26](#)
 - BL_VerifyApplication, [27](#)
 - BL_WriteAppsInfoToFlash, [27](#)
 - BL_WriteToFlash, [28](#)
 - Get_CurrentBL_Info, [28](#)
 - hcrc, [28](#)
 - huart1, [28](#)
 - huart2, [29](#)
- bootloader/bl_typedef.h, [11](#)
- bootloader/bootloader.c, [15](#)
- bootloader/bootloader.h, [22](#)
- Bootloader_MemRead
 - bootloader.c, [20](#)
- CBL_FLASH_ERASE_CMD
 - bl_typedef.h, [14](#)
- CBL_GET_APP_UPDATE
 - bl_typedef.h, [14](#)
- CBL_GET_CID_CMD
 - bl_typedef.h, [14](#)
- CBL_GET_HELP_CMD
 - bl_typedef.h, [14](#)
- CBL_GET_RDP_STATUS_CMD
 - bl_typedef.h, [14](#)
- CBL_GET_VER_CMD
 - bl_typedef.h, [14](#)
- CBL_JUMPTOAPP
 - bl_typedef.h, [14](#)
- CBL_MEM_READ_CMD
 - bl_typedef.h, [14](#)
- crc.c
 - CRC32_Calculate, [29](#)
 - CRC32_Init, [30](#)
 - CRC32_ResetCRC, [30](#)
- crc.h
 - CRC32_Calculate, [32](#)
 - CRC32_Init, [33](#)
 - CRC32_ResetCRC, [33](#)
- crc/crc.c, [29](#)
- crc/crc.h, [31](#)
- crc/crc_typedefs.h, [34](#)
- CRC32_Calculate
 - crc.c, [29](#)
 - crc.h, [32](#)
- CRC32_Init
 - crc.c, [30](#)
 - crc.h, [33](#)
- CRC32_ResetCRC
 - crc.c, [30](#)
 - crc.h, [33](#)
- CRC_Handle_t, [6](#)
 - crc_initval, [7](#)
 - crc_poly, [7](#)
 - crc_reg, [7](#)
- crc_initval
 - CRC_Handle_t, [7](#)
- crc_poly
 - CRC_Handle_t, [7](#)
- crc_reg
 - CRC_Handle_t, [7](#)
- crc_typedefs.h
 - DEFAULT_INIT_VAL, [35](#)
 - DEFAULT_POLY_VAL, [35](#)
- CRC_VERIFY_FAIL
 - bl_typedef.h, [13](#)
- CRC_VERIFY_PASS
 - bl_typedef.h, [13](#)
- DEFAULT_INIT_VAL
 - crc_typedefs.h, [35](#)
- DEFAULT_POLY_VAL
 - crc_typedefs.h, [35](#)
- Error_Handler
 - main.c, [36](#)
- generate_random_number
 - bootloader.c, [21](#)
- Get_CurrentBL_Info
 - bootloader.c, [21](#)
 - bootloader.h, [28](#)
- hash
 - App_INFO_t, [5](#)
 - Update_INFO_t, [7](#)
- hcrc
 - bootloader.h, [28](#)
 - main.c, [38](#)
- huart1
 - bootloader.h, [28](#)
 - main.c, [38](#)
- huart2
 - bootloader.h, [29](#)
 - main.c, [38](#)
- main
 - main.c, [36](#)

- main.c, [35](#)
 - bl_info_ptr, [38](#)
 - Error_Handler, [36](#)
 - hcrc, [38](#)
 - huart1, [38](#)
 - huart2, [38](#)
 - main, [36](#)
 - SystemClock_Config, [37](#)
- major_version
 - Version_ID_t, [8](#)
- minor_version
 - Version_ID_t, [8](#)
- NOR_APP_NOT_UPDATE_FOUND
 - bl_typedef.h, [13](#)
- PAGE0_ADDRESS
 - bl_typedef.h, [13](#)
- PAGE_SIZE
 - bl_typedef.h, [13](#)
- patch_version
 - Version_ID_t, [9](#)
- Set_ValidUpdate
 - bootloader.c, [21](#)
- STM32F103C8T6_PAGES_NUM
 - bl_typedef.h, [13](#)
- SystemClock_Config
 - main.c, [37](#)
- update
 - BL_INFO_t, [6](#)
- UPDATE_FOUND
 - bl_typedef.h, [14](#)
- Update_INFO_t, [7](#)
 - hash, [7](#)
 - update_size, [8](#)
 - updated, [8](#)
- update_size
 - Update_INFO_t, [8](#)
- updated
 - Update_INFO_t, [8](#)
- vendor_id
 - Version_ID_t, [9](#)
- VERIFICATION_ERROR
 - bl_typedef.h, [14](#)
- Version_ID_t, [8](#)
 - major_version, [8](#)
 - minor_version, [8](#)
 - patch_version, [9](#)
 - vendor_id, [9](#)