BootCommander - Reference Manual

1.4.1

Generated by Doxygen 1.8.14

Contents

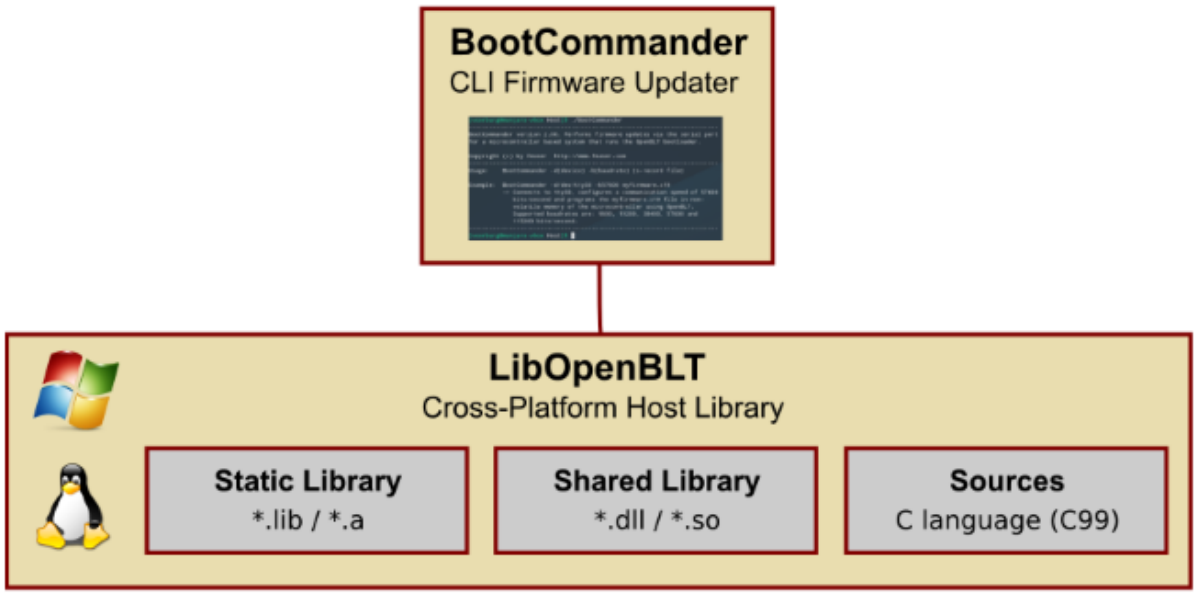
Chapter 1

BootCommander - Command Line Firmware Update Tool

1.1 Introduction

BootCommander is a command line program for performing a firmware update on a connected microcontroller target that runs the OpenBLT bootloader. It is written in the C programming language (C99) and is cross-platform. It has been successfully tested on a Windows PC, Linux PC and even embedded Linux systems such as a Raspberry Pi and a Beagle Board.

BootCommander is built on top of the OpenBLT Host Library (LibOpenBLT), which embeds all functionality needed to communicate with the OpenBLT bootloader on the microcontroller target.



This program is specifically developed for those that prefer a command line program over a program with a graphical user interface, such as MicroBoot. Additionally, it can serve as a reference on how to use LibOpenBLT.

Refer to the OpenBLT website for additional information regarding the usage of the BootCommander program, including a description of the available command line parameters and step-by-step instructions on how to build the BootCommander program from sources: https://www.feaser.com/openblt/doku.↩php?id=manual:bootcommander.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 tProgramSettings Struct Reference

Type for program settings.

4.1.1 Detailed Description

Type for program settings.

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

- main.c

Chapter 5

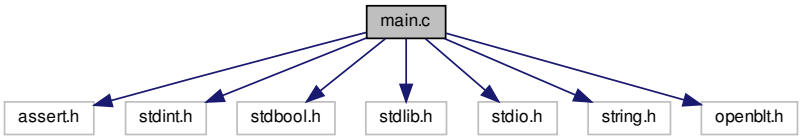
File Documentation

5.1 main.c File Reference

BootCommander program source file.

```
#include <assert.h>
#include <stdint.h>
#include <stdbool.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "openblt.h"
```

Include dependency graph for main.c:



Data Structures

- struct tProgramSettings

  Type for program settings.

Macros

- #define RESULT_OK (0)

  Program return code indicating that the program executed successfully.
- #define RESULT_ERROR_COMMANDLINE (1)

  Program return code indicating that an error was detected when processing the command line parameters/.
- #define RESULT_ERROR_FIRMWARE_LOAD (2)

  Program return code indicating that an error was detected while loading firmware data from the firmware file.
- #define RESULT_ERROR_MEMORY_ERASE (3)

  Program return code indicating that an error was detected during a memory erase operation on the target.
- #define RESULT_ERROR_MEMORY_PROGRAM (4)

  Program return code indicating that an error was detected during a memory progrma operation on the target.

Functions

- static void DisplayProgramInfo (void)

  Outputs information to the user about this program.
- static void DisplayProgramUsage (void)

  Outputs information to the user about how to use this program.
- static void DisplaySessionInfo (uint32_t sessionType, void const *sessionSettings)

  Displays session protocol information on the standard output.
- static void DisplayTransportInfo (uint32_t transportType, void const *transportSettings)

  Displays transport layer information on the standard output.
- static void DisplayFirmwareDataInfo (uint32_t segments, uint32_t base, uint32_t size)

  Displays firmware data information on the standard output.
- static void ExtractProgramSettingsFromCommandLine (int argc, char const *const argv[], tProgramSettings *programSettings)

  Parses the command line to extract the program settings.
- static uint32_t ExtractSessionTypeFromCommandLine (int argc, char const *const argv[])

  Parses the command line to extract the session type. This is the one specified via the -s=[name] parameter.
- static void * ExtractSessionSettingsFromCommandLine (int argc, char const *const argv[], uint32_t sessionType)

  Parses the command line to extract the session settings based on the specified session type. Note that this function allocates the memory necessary to store the settings. It is the caller's responsibility to free this memory after it is done with it.
- static uint32_t ExtractTransportTypeFromCommandLine (int argc, char const *const argv[])

  Parses the command line to extract the transport type. This is the one specified via the -t=[name] parameter.
- static void * ExtractTransportSettingsFromCommandLine (int argc, char const *const argv[], uint32_t transportType)

  Parses the command line to extract the transport settings based on the specified session type. Note that this function allocates the memory necessary to store the settings. It is the caller's responsibility to free this memory after it is done with it.
- static char const *const ExtractFirmwareFileFromCommandLine (int argc, char const *const argv[])

  Parses the command line to extract the firmware file. This is the only parameter that is specified without a '-' character at the start.
- static char const * GetLineTrailerByResult (bool errorDetected)

  Information outputted to the user sometimes has [OK] or [ERROR] appended at the end. This function obtains this trailer based on the value of the parameter.
- static char const * GetLineTrailerByPercentage (uint8_t percentage)

  Information outputted to the user sometimes has [xxx%] appended at the end. This function obtains this trailer based on the value of the parameter.
- static void ErasePercentageTrailer (void)

  Erases a percentage trailer from the standard ouput by means of writing backspace characters.
- int main (int argc, char const *const argv[])

  This is the program entry point.

5.1.1 Detailed Description

BootCommander program source file.

5.1.2 Function Documentation

5.1.2.1 DisplayFirmwareDataInfo()

```
static void DisplayFirmwareDataInfo (
             uint32_t segments,
             uint32_t base,
             uint32_t size )  [static]
```

Displays firmware data information on the standard output.

Parameters

segments	Total number of firmware data segments
base	The base memory address of the firmware data.
size	Total number of firmware data bytes.

Referenced by main().

Here is the caller graph for this function:

5.1.2.2 DisplaySessionInfo()

```
static void DisplaySessionInfo (
             uint32_t sessionType,
             void const * sessionSettings )  [static]
```

Displays session protocol information on the standard output.

Parameters

sessionType	The detected session type.
sessionSettings	The detected session settings.

Referenced by main().

Here is the caller graph for this function:

5.1.2.3 DisplayTransportInfo()

```
static void DisplayTransportInfo (
            uint32_t transportType,
            void const * transportSettings )  [static]
```

Displays transport layer information on the standard output.

Parameters

transportType	The detected transport type.
transportSettings	The detected transport settings.

Referenced by main().

Here is the caller graph for this function:

5.1.2.4 ExtractFirmwareFileFromCommandLine()

```
static char const *const ExtractFirmwareFileFromCommandLine (
            int argc,
            char const *const argv[] )  [static]
```

Parses the command line to extract the firmware file. This is the only parameter that is specified without a '-' character at the start.

Parameters

argc	Number of program arguments.
argv	Array with program parameter strings.

Returns

Pointer to the character array (string) with the firmware file.

Referenced by main().

Here is the caller graph for this function:

5.1.2.5 ExtractProgramSettingsFromCommandLine()

```
static void ExtractProgramSettingsFromCommandLine (
            int argc,
            char const *const argv[],
            tProgramSettings * programSettings )  [static]
```

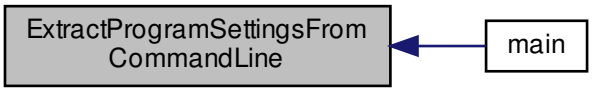
Parses the command line to extract the program settings.

Parameters

argc	Number of program arguments.
argv	Array with program parameter strings.
programSettings	Pointer to the setting structure where the program settings should be written to.

Referenced by main().

Here is the caller graph for this function:



5.1.2.6 ExtractSessionSettingsFromCommandLine()

```
static void * ExtractSessionSettingsFromCommandLine (
            int argc,
            char const *const argv[],
            uint32_t sessionType )  [static]
```

Parses the command line to extract the session settings based on the specified session type. Note that this function allocates the memory necessary to store the settings. It is the caller's responsibility to free this memory after it is done with it.

Parameters

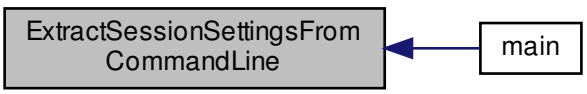
argc	Number of program arguments.
argv	Array with program parameter strings.
sessionType	The session type for which to extract the settings.

Returns

Pointer to the session settings structure as used in LibOpenBLT.

Referenced by main().

Here is the caller graph for this function:



5.1.2.7 ExtractSessionTypeFromCommandLine()

```
static uint32_t ExtractSessionTypeFromCommandLine (
            int argc,
            char const *const argv[] )  [static]
```

Parses the command line to extract the session type. This is the one specified via the -s=[name] parameter.

Parameters

argc	Number of program arguments.
argv	Array with program parameter strings.

Returns

The session type value as used in LibOpenBLT.

Referenced by main().

Here is the caller graph for this function:

5.1.2.8 ExtractTransportSettingsFromCommandLine()

```
static void * ExtractTransportSettingsFromCommandLine (
            int argc,
            char const *const argv[],
            uint32_t transportType )  [static]
```

Parses the command line to extract the transport settings based on the specified session type. Note that this function allocates the memory necessary to store the settings. It is the caller's responsibility to free this memory after it is done with it.

Parameters

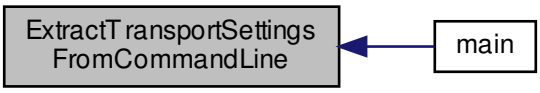
argc	Number of program arguments.
argv	Array with program parameter strings.
transportType	The transport type for which to extract the settings.

Returns

Pointer to the transport settings structure as used in LibOpenBLT.

Referenced by main().

Here is the caller graph for this function:



5.1.2.9 ExtractTransportTypeFromCommandLine()

```
static uint32_t ExtractTransportTypeFromCommandLine (
            int argc,
            char const *const argv[] )  [static]
```

Parses the command line to extract the transport type. This is the one specified via the -t=[name] parameter.

Parameters

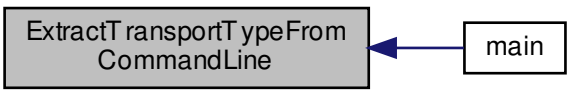
argc	Number of program arguments.
argv	Array with program parameter strings.

Returns

The transport type value as used in LibOpenBLT.

Referenced by main().

Here is the caller graph for this function:



5.1.2.10 GetLineTrailerByPercentage()

```
static char const * GetLineTrailerByPercentage (
            uint8_t percentage )  [static]
```

Information outputted to the user sometimes has [xxx%] appended at the end. This function obtains this trailer based on the value of the parameter.

Parameters

percentage	Percentage value (0..100) to embed in the trailer.

Returns

Pointer to the character array (string) with the trailer.

5.1.2.11 GetLineTrailerByResult()

```
static char const * GetLineTrailerByResult (
            bool errorDetected )  [static]
```

Information outputted to the user sometimes has [OK] or [ERROR] appended at the end. This function obtains this trailer based on the value of the parameter.

Parameters

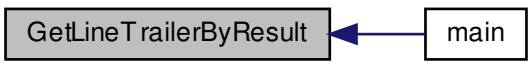
errorDetected	True to obtain an error trailer, false for a success trailer.

Returns

Pointer to the character array (string) with the trailer.

Referenced by main().

Here is the caller graph for this function:



5.1.2.12 main()

```
int main (
            int argc,
            char const *const argv[] )
```

This is the program entry point.

Parameters

argc	Number of program arguments.
argv	Array with program arguments.

Returns

Program return code. 0 for success, error code otherwise.

Index