

## STM32 Blue Pill Drivers

Generated on Sun Jun 18 2023 13:46:18 for STM32 Blue Pill Drivers by Doxygen 1.9.5

Sun Jun 18 2023 13:46:18



<b>1 STM32 Blue Pill Drivers</b>	<b>1</b>
<b>2 Testing applications</b>	<b>3</b>
2.1 ## Applications list . . . . .	3
<b>3 Module Index</b>	<b>5</b>
3.1 Modules . . . . .	5
<b>4 File Index</b>	<b>7</b>
4.1 File List . . . . .	7
<b>5 Module Documentation</b>	<b>9</b>
5.1 Bit Manipulation Math Macros . . . . .	9
5.1.1 Detailed Description . . . . .	9
5.1.2 Macro Definition Documentation . . . . .	9
5.1.2.1 SET_BIT . . . . .	9
5.1.2.2 CLEAR_BIT . . . . .	10
5.1.2.3 TOGGLE_BIT . . . . .	10
5.1.2.4 GET_BIT . . . . .	10
5.2 Compiler standard macros . . . . .	11
5.2.1 Detailed Description . . . . .	11
5.2.2 Macro Definition Documentation . . . . .	11
5.2.2.1 CONST . . . . .	11
5.2.2.2 STATIC . . . . .	12
5.2.2.3 VOLATILE . . . . .	12
5.2.2.4 P2VAR . . . . .	12
5.2.2.5 P2CONST . . . . .	12
5.2.2.6 CONSTP2VAR . . . . .	13
5.2.2.7 CONSTP2CONST . . . . .	13
5.2.2.8 P2FUNC . . . . .	13
5.3 Hardware registers macro-functions . . . . .	13
5.3.1 Detailed Description . . . . .	14
5.3.2 Macro Definition Documentation . . . . .	14
5.3.2.1 REGISTER_ADDRESS . . . . .	14
5.3.2.2 REGISTER . . . . .	15
5.3.2.3 REGISTER_U8 . . . . .	15
5.3.2.4 REGISTER_U16 . . . . .	16
5.3.2.5 REGISTER_U32 . . . . .	16
5.4 Standard Library . . . . .	16
5.4.1 Detailed Description . . . . .	17
5.5 Standard data types . . . . .	17
5.5.1 Detailed Description . . . . .	18
5.5.2 Typedef Documentation . . . . .	18
5.5.2.1 t_bool . . . . .	18

5.5.2.2 t_u8 . . . . .	18
5.5.2.3 t_s8 . . . . .	18
5.5.2.4 t_c8 . . . . .	18
5.5.2.5 t_u16 . . . . .	19
5.5.2.6 t_s16 . . . . .	19
5.5.2.7 t_u32 . . . . .	19
5.5.2.8 t_s32 . . . . .	19
5.5.2.9 t_u64 . . . . .	19
5.5.2.10 t_s64 . . . . .	20
5.5.2.11 t_fl32 . . . . .	20
5.5.2.12 t_fl64 . . . . .	20
5.6 Standard values . . . . .	20
5.6.1 Detailed Description . . . . .	20
5.6.2 Macro Definition Documentation . . . . .	21
5.6.2.1 TRUE . . . . .	21
5.6.2.2 FALSE . . . . .	21
5.6.2.3 NULL . . . . .	21
<b>6 File Documentation</b> . . . . .	<b>23</b>
6.1 APPS_main.c File Reference . . . . .	23
6.1.1 Detailed Description . . . . .	23
6.1.2 Function Documentation . . . . .	23
6.1.2.1 vAPPS_main() . . . . .	24
6.2 APPS_main.h File Reference . . . . .	24
6.2.1 Detailed Description . . . . .	24
6.2.2 Function Documentation . . . . .	24
6.2.2.1 vAPPS_main() . . . . .	25
6.3 github/workspace/README.md File Reference . . . . .	25
6.4 README.md File Reference . . . . .	25
6.5 LIB/LSTD_BITMATH.h File Reference . . . . .	25
6.5.1 Detailed Description . . . . .	25
6.6 LIB/LSTD_COMPILER.h File Reference . . . . .	26
6.6.1 Detailed Description . . . . .	26
6.7 LIB/LSTD_HW_REGS.h File Reference . . . . .	26
6.7.1 Detailed Description . . . . .	27
6.8 LIB/LSTD_TYPES.h File Reference . . . . .	27
6.8.1 Detailed Description . . . . .	28
6.9 LIB/LSTD_VALUES.h File Reference . . . . .	28
6.10 main.c File Reference . . . . .	28
6.10.1 Function Documentation . . . . .	29
6.10.1.1 main() . . . . .	29
<b>Index</b> . . . . .	<b>31</b>

## Chapter 1

# STM32 Blue Pill Drivers

Drivers that could be used to interface and interact with STM32F103C8T6 Microcontroller

View the PDF documentation [here](#)



## Chapter 2

# Testing applications

These testing applications to test most of the drivers' functionalities and make sure they do what they intend to do.

### 2.1 ## Applications list

Below is the applications list and will follow this template of describing each application.

Name: <Application name>  
Description: <Application description>  
Activision Macro: <Application macro name>

Testing application's directory has the following structure:

- APPS\_main.h
- APPS\_main.c
- <Application name>/
  - <Application name>\_main.h
  - <Application name>\_main.c





## Chapter 3

# Module Index

### 3.1 Modules

Here is a list of all modules:

Standard Library . . . . .	16
Bit Manipulation Math Macros . . . . .	9
Compiler standard macros . . . . .	11
Hardware registers macro-functions . . . . .	13
Standard data types . . . . .	17
Standard values . . . . .	20



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

<a href="#">main.c</a> . . . . .	28
<a href="#">APPS_main.c</a> This file contains the implementation of the main function that is responsible for running the applications . . . . .	23
<a href="#">APPS_main.h</a> This file contains the prototypes of the main function that is responsible for running the applications . . . . .	24
<a href="#">LIB/LSTD_BITMATH.h</a> This file contains the bit math manipulation macro-functions . . . . .	25
<a href="#">LIB/LSTD_COMPILER.h</a> This file contains the compiler standard macros . . . . .	26
<a href="#">LIB/LSTD_HW_REGS.h</a> This file contains the hardware registers macro-functions for memory addresses mapping and accessing . . . . .	26
<a href="#">LIB/LSTD_TYPES.h</a> This file contains the standard data types . . . . .	27
<a href="#">LIB/LSTD_VALUES.h</a> This file contains the standard values . . . . .	28



## Chapter 5

# Module Documentation

### 5.1 Bit Manipulation Math Macros

This module contains the bit math manipulation macro-functions.

Collaboration diagram for Bit Manipulation Math Macros:

#### Macros

- `#define SET_BIT(REG, BITNUM) (REG) |= (1 << (BITNUM))`  
*Set a certain bit's value.*
- `#define CLEAR_BIT(REG, BITNUM) (REG) &= ~(1 << (BITNUM))`  
*Clear a certain bit's value to.*
- `#define TOGGLE_BIT(REG, BITNUM) (REG) ^= (1 << (BITNUM))`  
*Toggle a bit to 0 if it's 1, 1 otherwise.*
- `#define GET_BIT(REG, BITNUM) (((REG) >> (BITNUM)) & 1)`  
*Return the value of the bit whether it's 1 or 0*

#### 5.1.1 Detailed Description

This module contains the bit math manipulation macro-functions.

#### 5.1.2 Macro Definition Documentation

##### 5.1.2.1 SET\_BIT

```
#define SET_BIT(  
    REG,  
    BITNUM ) (REG) |= (1 << (BITNUM))
```

```
#include <LIB/LSTD_BITMATH.h>
```

Set a certain bit's value.

**Parameters**

in	<i>REG</i>	The register to set its bit
in	<i>BITNUM</i>	The bit number to set

**5.1.2.2 CLEAR\_BIT**

```
#define CLEAR_BIT(  
    REG,  
    BITNUM ) (REG) &= ~(1 << (BITNUM))  
  
#include <LIB/LSTD_BITMATH.h>
```

Clear a certain bit's value to.

**Parameters**

in	<i>REG</i>	The register to clear its bit
in	<i>BITNUM</i>	The bit number to clear

**5.1.2.3 TOGGLE\_BIT**

```
#define TOGGLE_BIT(  
    REG,  
    BITNUM ) (REG) ^= (1 << (BITNUM))  
  
#include <LIB/LSTD_BITMATH.h>
```

Toggle a bit to 0 if it's 1, 1 otherwise.

**Parameters**

in	<i>REG</i>	The register to toggle its bit
in	<i>BITNUM</i>	The bit number to toggle

**5.1.2.4 GET\_BIT**

```
#define GET_BIT(  
    REG,  
    BITNUM ) (((REG) >> (BITNUM)) & 1)  
  
#include <LIB/LSTD_BITMATH.h>
```

Return the value of the bit whether it's 1 or 0

## Parameters

in	<i>REG</i>	The register to get its bit
in	<i>BITNUM</i>	The bit number to get

## 5.2 Compiler standard macros

This file contains the compiler standard macros.

Collaboration diagram for Compiler standard macros:

### Macros

- #define **CONST** const  
*Declare a standard constant variable with the specified type.*
- #define **STATIC** static  
*Declare a standard static variable.*
- #define **VOLATILE** volatile  
*Declare a standard volatile variable.*
- #define **P2VAR**(ptrtype) ptrtype \*  
*Declare a pointer-to-variable with the specified type.*
- #define **P2CONST**(ptrtype) **CONST** ptrtype \*  
*Declare a constant pointer-to-variable with the specified type.*
- #define **CONSTP2VAR**(ptrtype) ptrtype \***CONST**  
*Declare a pointer-to-variable constant with the specified type.*
- #define **CONSTP2CONST**(ptrtype) **CONST** ptrtype \***CONST**  
*Declare a constant pointer-to-variable constant with the specified type.*
- #define **P2FUNC**(rettype, fctname) rettype(\*fctname)  
*Declare a pointer-to-function with the specified return type.*

### 5.2.1 Detailed Description

This file contains the compiler standard macros.

### 5.2.2 Macro Definition Documentation

#### 5.2.2.1 **CONST**

```
#define CONST const
```

```
#include <LIB/LSTD_COMPILER.h>
```

Declare a standard constant variable with the specified type.

### 5.2.2.2 STATIC

```
#define STATIC static
```

```
#include <LIB/LSTD_COMPILER.h>
```

Declare a standard static variable.

### 5.2.2.3 VOLATILE

```
#define VOLATILE volatile
```

```
#include <LIB/LSTD_COMPILER.h>
```

Declare a standard volatile variable.

### 5.2.2.4 P2VAR

```
#define P2VAR(  
    ptrtype ) ptrtype *
```

```
#include <LIB/LSTD_COMPILER.h>
```

Declare a pointer-to-variable with the specified type.

#### Parameters

in	<i>ptrtype</i>	The type of the pointer
----	----------------	-------------------------

### 5.2.2.5 P2CONST

```
#define P2CONST(  
    ptrtype ) CONST ptrtype *
```

```
#include <LIB/LSTD_COMPILER.h>
```

Declare a constant pointer-to-variable with the specified type.

#### Parameters

in	<i>ptrtype</i>	The type of the pointer
----	----------------	-------------------------



### 5.2.2.6 CONSTP2VAR

```
#define CONSTP2VAR(  
    ptrtype ) ptrtype *CONST  
  
#include <LIB/LSTD_COMPILER.h>
```

Declare a pointer-to-variable constant with the specified type.

#### Parameters

in	<i>ptrtype</i>	The type of the pointer
----	----------------	-------------------------

### 5.2.2.7 CONSTP2CONST

```
#define CONSTP2CONST(  
    ptrtype ) CONST ptrtype *CONST  
  
#include <LIB/LSTD_COMPILER.h>
```

Declare a constant pointer-to-variable constant with the specified type.

#### Parameters

in	<i>ptrtype</i>	The type of the pointer
----	----------------	-------------------------

### 5.2.2.8 P2FUNC

```
#define P2FUNC(  
    rettype,  
    fctname ) rettype(*fctname)  
  
#include <LIB/LSTD_COMPILER.h>
```

Declare a pointer-to-function with the specified return type.

#### Parameters

in	<i>rettype</i>	The return type of the function
in	<i>fctname</i>	The name of the function

## 5.3 Hardware registers macro-functions

These macro-functions help in mapping and accessing the hardware registers.

Collaboration diagram for Hardware registers macro-functions:

## Macros

- `#define REGISTER_ADDRESS(ADDRESS, OFFSET) ((ADDRESS) + (OFFSET))`  
*Placeholder for declaring a register address.*
- `#define REGISTER(REG_TYPE, ADDRESS) (*(VOLATILE P2VAR(REG_TYPE))(ADDRESS))`  
*Map to a certain register by its address in the memory.*
- `#define REGISTER_U8(ADDRESS) REGISTER(t_u8, ADDRESS)`  
*Map to a certain register by its 8-bit address in the memory (used for 8-bit registers)*
- `#define REGISTER_U16(ADDRESS) REGISTER(t_u16, ADDRESS)`  
*Map to a certain register by its 16-bit address in the memory (used for 16-bit registers)*
- `#define REGISTER_U32(ADDRESS) REGISTER(t_u32, ADDRESS)`  
*Map to a certain register by its 32-bit address in the memory (used for 32-bit registers)*

### 5.3.1 Detailed Description

These macro-functions help in mapping and accessing the hardware registers.

### 5.3.2 Macro Definition Documentation

#### 5.3.2.1 REGISTER\_ADDRESS

```
#define REGISTER_ADDRESS(  
    ADDRESS,  
    OFFSET ) ((ADDRESS) + (OFFSET))
```

```
#include <LIB/LSTD_HW_REGS.h>
```

Placeholder for declaring a register address.

#### Parameters

in	<i>ADDRESS</i>	The address of the register
in	<i>OFFSET</i>	The offset of the register

#### Returns

The final address of the register

### 5.3.2.2 REGISTER

```
#define REGISTER(  
    REG_TYPE,  
    ADDRESS ) (*(VOLATILE P2VAR(REG_TYPE)) (ADDRESS))
```

```
#include <LIB/LSTD_HW_REGS.h>
```

Map to a certain register by its address in the memory.

#### Parameters

in	<i>ADDRESS</i>	The address of the register
in	<i>REG_TYPE</i>	The type of the register

#### Note

REG\_TYPE can be a standard type (e.g. t\_u8, t\_u16, t\_u32, ...) or a user-defined type

#### Returns

The value of the register

#### See also

### 5.3.2.3 REGISTER\_U8

```
#define REGISTER_U8(  
    ADDRESS ) REGISTER(t_u8, ADDRESS)
```

```
#include <LIB/LSTD_HW_REGS.h>
```

Map to a certain register by its 8-bit address in the memory (used for 8-bit registers)

#### Parameters

in	<i>ADDRESS</i>	The address of the register
----	----------------	-----------------------------

#### Returns

The value of the register

#### 5.3.2.4 REGISTER\_U16

```
#define REGISTER_U16(  
    ADDRESS ) REGISTER(t_u16, ADDRESS)
```

```
#include <LIB/LSTD_HW_REGS.h>
```

Map to a certain register by its 16-bit address in the memory (used for 16-bit registers)

##### Parameters

in	ADDRESS	The address of the register
----	---------	-----------------------------

##### Returns

The value of the register

#### 5.3.2.5 REGISTER\_U32

```
#define REGISTER_U32(  
    ADDRESS ) REGISTER(t_u32, ADDRESS)
```

```
#include <LIB/LSTD_HW_REGS.h>
```

Map to a certain register by its 32-bit address in the memory (used for 32-bit registers)

##### Parameters

in	ADDRESS	The address of the register
----	---------	-----------------------------

##### Returns

The value of the register

## 5.4 Standard Library

This module contains the standard library-related macros, types, and functions.

Collaboration diagram for Standard Library:

### Modules

- [Bit Manipulation Math Macros](#)

*This module contains the bit math manipulation macro-functions.*

- [Compiler standard macros](#)

*This file contains the compiler standard macros.*

- [Hardware registers macro-functions](#)

*These macro-functions help in mapping and accessing the hardware registers.*

- [Standard data types](#)

*This module contains the standard data types.*

- [Standard values](#)

*This module contains the standard values.*

### 5.4.1 Detailed Description

This module contains the standard library-related macros, types, and functions.

## 5.5 Standard data types

This module contains the standard data types.

Collaboration diagram for Standard data types:

### Typedefs

- typedef unsigned char [t\\_bool](#)

*Type definition for boolean.*

- typedef unsigned char [t\\_u8](#)

*Type definition for 8-bit unsigned INT.*

- typedef signed char [t\\_s8](#)

*Type definition for 8-bit signed INT.*

- typedef unsigned char [t\\_c8](#)

*Type definition for 8-bit char.*

- typedef unsigned short int [t\\_u16](#)

*Type definition for 16-bit unsigned int.*

- typedef signed short int [t\\_s16](#)

*Type definition for 16-bit signed INT.*

- typedef unsigned int [t\\_u32](#)

*Type definition for 32-bit unsigned int.*

- typedef signed int [t\\_s32](#)

*Type definition for 32-bit signed INT.*

- typedef unsigned long int [t\\_u64](#)

*Type definition for 64-bit unsigned int.*

- typedef signed long int [t\\_s64](#)

*Type definition for 64-bit signed int.*

- typedef float [t\\_f32](#)

*Type definition for 32-bit float.*

- typedef double [t\\_f64](#)

*Type definition for 64-bit float.*

### 5.5.1 Detailed Description

This module contains the standard data types.

### 5.5.2 Typedef Documentation

#### 5.5.2.1 t\_bool

t\_bool

```
#include <LIB/LSTD_TYPES.h>
```

Type definition for boolean.

#### 5.5.2.2 t\_u8

t\_u8

```
#include <LIB/LSTD_TYPES.h>
```

Type definition for 8-bit unsigned INT.

#### 5.5.2.3 t\_s8

t\_s8

```
#include <LIB/LSTD_TYPES.h>
```

Type definition for 8-bit signed INT.

#### 5.5.2.4 t\_c8

t\_c8

```
#include <LIB/LSTD_TYPES.h>
```

Type definition for 8-bit char.

#### 5.5.2.5 t\_u16

t\_u16

```
#include <LIB/LSTD_TYPES.h>
```

Type definition for 16-bit unsigned int.

#### 5.5.2.6 t\_s16

t\_s16

```
#include <LIB/LSTD_TYPES.h>
```

Type definition for 16-bit signed INT.

#### 5.5.2.7 t\_u32

t\_u32

```
#include <LIB/LSTD_TYPES.h>
```

Type definition for 32-bit unsigned int.

#### 5.5.2.8 t\_s32

t\_s32

```
#include <LIB/LSTD_TYPES.h>
```

Type definition for 32-bit signed INT.

#### 5.5.2.9 t\_u64

t\_u64

```
#include <LIB/LSTD_TYPES.h>
```

Type definition for 64-bit unsigned int.

#### 5.5.2.10 t\_s64

t\_s64

```
#include <LIB/LSTD_TYPES.h>
```

Type definition for 64-bit signed int.

#### 5.5.2.11 t\_fl32

t\_fl32

```
#include <LIB/LSTD_TYPES.h>
```

Type definition for 32-bit float.

#### 5.5.2.12 t\_fl64

t\_fl64

```
#include <LIB/LSTD_TYPES.h>
```

Type definition for 64-bit float.

## 5.6 Standard values

This module contains the standard values.

Collaboration diagram for Standard values:

### Macros

- #define TRUE ((t\_bool)1)  
*Type definition for TRUE.*
- #define FALSE ((t\_bool)0)  
*Type definition for FALSE.*
- #define NULL ((P2VAR(void))0)  
*Type definition for NULL.*

### 5.6.1 Detailed Description

This module contains the standard values.



## 5.6.2 Macro Definition Documentation

### 5.6.2.1 TRUE

```
#define TRUE ((t_bool)1)
```

```
#include <LIB/LSTD_VALUES.h>
```

Type definition for TRUE.

### 5.6.2.2 FALSE

```
#define FALSE ((t_bool)0)
```

```
#include <LIB/LSTD_VALUES.h>
```

Type definition for FALSE.

### 5.6.2.3 NULL

```
#define NULL ((P2VAR(void))0)
```

```
#include <LIB/LSTD_VALUES.h>
```

Type definition for NULL.



# Chapter 6

## File Documentation

### 6.1 APPS\_main.c File Reference

This file contains the implementation of the main function that is responsible for running the applications.

```
#include "APPS_main.h"  
Include dependency graph for APPS_main.c:
```

#### Functions

- void [vAPPS\\_main](#) (void)  
*Change this to the macro of the desired application to run.*

#### 6.1.1 Detailed Description

This file contains the implementation of the main function that is responsible for running the applications.

##### Author

Mohamed Alaa

##### Version

1.0.0

##### Date

2023-06-16

#### 6.1.2 Function Documentation

### 6.1.2.1 vAPPS\_main()

```
void vAPPS_main (
    void )
```

Change this to the macro of the desired application to run.

```
17 {
18     for (;;)
19     {
20         /* Do nothing */
21     }
22 }
```

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

Here is the caller graph for this function:

## 6.2 APPS\_main.h File Reference

This file contains the prototypes of the main function that is responsible for running the applications.

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

### Functions

- void [vAPPS\\_main](#) (void)  
*Change this to the macro of the desired application to run.*

### 6.2.1 Detailed Description

This file contains the prototypes of the main function that is responsible for running the applications.

#### Author

Mohamed Alaa

#### Version

1.0.0

#### Date

2023-06-16

### 6.2.2 Function Documentation

### 6.2.2.1 vAPPS\_main()

```
void vAPPS_main (
    void )
```

Change this to the macro of the desired application to run.

```
17 {
18     for (;)
19     {
20         /* Do nothing */
21     }
22 }
```

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

Here is the caller graph for this function:

## 6.3 github/workspace/README.md File Reference

## 6.4 README.md File Reference

## 6.5 LIB/LSTD\_BITMATH.h File Reference

This file contains the bit math manipulation macro-functions.

### Macros

- #define [SET\\_BIT](#)(REG, BITNUM) (REG) |= (1 << (BITNUM))  
*Set a certain bit's value.*
- #define [CLEAR\\_BIT](#)(REG, BITNUM) (REG) &= ~(1 << (BITNUM))  
*Clear a certain bit's value to.*
- #define [TOGGLE\\_BIT](#)(REG, BITNUM) (REG) ^= (1 << (BITNUM))  
*Toggle a bit to 0 if it's 1, 1 otherwise.*
- #define [GET\\_BIT](#)(REG, BITNUM) (((REG) >> (BITNUM)) & 1)  
*Return the value of the bit whether it's 1 or 0*

### 6.5.1 Detailed Description

This file contains the bit math manipulation macro-functions.

#### Author

Mohamed alaa

#### Version

1.0.0

#### Date

2023-06-18

## 6.6 LIB/LSTD\_COMPILER.h File Reference

This file contains the compiler standard macros.

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

### Macros

- #define **CONST** const  
*Declare a standard constant variable with the specified type.*
- #define **STATIC** static  
*Declare a standard static variable.*
- #define **VOLATILE** volatile  
*Declare a standard volatile variable.*
- #define **P2VAR**(ptrtype) ptrtype \*  
*Declare a pointer-to-variable with the specified type.*
- #define **P2CONST**(ptrtype) **CONST** ptrtype \*  
*Declare a constant pointer-to-variable with the specified type.*
- #define **CONSTP2VAR**(ptrtype) ptrtype \***CONST**  
*Declare a pointer-to-variable constant with the specified type.*
- #define **CONSTP2CONST**(ptrtype) **CONST** ptrtype \***CONST**  
*Declare a constant pointer-to-variable constant with the specified type.*
- #define **P2FUNC**(rettype, fctname) rettype(\*fctname)  
*Declare a pointer-to-function with the specified return type.*

### 6.6.1 Detailed Description

This file contains the compiler standard macros.

#### Author

Mohamed Alaa

#### Version

1.0.0

#### Date

2023-06-18

## 6.7 LIB/LSTD\_HW\_REGS.h File Reference

This file contains the hardware registers macro-functions for memory addresses mapping and accessing.

```
#include "LSTD_TYPES.h"
#include "LSTD_COMPILER.h"
Include dependency graph for LSTD_HW_REGS.h:
```

## Macros

- #define [REGISTER\\_ADDRESS](#)(ADDRESS, OFFSET) ((ADDRESS) + (OFFSET))  
*Placeholder for declaring a register address.*
- #define [REGISTER](#)(REG\_TYPE, ADDRESS) (\*([VOLATILE P2VAR](#)(REG\_TYPE))(ADDRESS))  
*Map to a certain register by its address in the memory.*
- #define [REGISTER\\_U8](#)(ADDRESS) [REGISTER](#)(t\_u8, ADDRESS)  
*Map to a certain register by its 8-bit address in the memory (used for 8-bit registers)*
- #define [REGISTER\\_U16](#)(ADDRESS) [REGISTER](#)(t\_u16, ADDRESS)  
*Map to a certain register by its 16-bit address in the memory (used for 16-bit registers)*
- #define [REGISTER\\_U32](#)(ADDRESS) [REGISTER](#)(t\_u32, ADDRESS)  
*Map to a certain register by its 32-bit address in the memory (used for 32-bit registers)*

### 6.7.1 Detailed Description

This file contains the hardware registers macro-functions for memory addresses mapping and accessing.

#### Author

Mohamed Alaa

#### Version

1.0.0

#### Date

2023-06-18

## 6.8 LIB/LSTD\_TYPES.h File Reference

This file contains the standard data types.

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

## Typedefs

- typedef unsigned char [t\\_bool](#)  
*Type definition for boolean.*
- typedef unsigned char [t\\_u8](#)  
*Type definition for 8-bit unsigned INT.*
- typedef signed char [t\\_s8](#)  
*Type definition for 8-bit signed INT.*
- typedef unsigned char [t\\_c8](#)  
*Type definition for 8-bit char.*
- typedef unsigned short int [t\\_u16](#)  
*Type definition for 16-bit unsigned int.*
- typedef signed short int [t\\_s16](#)

*Type definition for 16-bit signed INT.*

- typedef unsigned int [t\\_u32](#)

*Type definition for 32-bit unsigned int.*

- typedef signed int [t\\_s32](#)

*Type definition for 32-bit signed INT.*

- typedef unsigned long int [t\\_u64](#)

*Type definition for 64-bit unsigned int.*

- typedef signed long int [t\\_s64](#)

*Type definition for 64-bit signed int.*

- typedef float [t\\_fl32](#)

*Type definition for 32-bit float.*

- typedef double [t\\_fl64](#)

*Type definition for 64-bit float.*

### 6.8.1 Detailed Description

This file contains the standard data types.

#### Author

Mohamed Alaa

#### Version

1.0.0

#### Date

2023-06-18

## 6.9 LIB/LSTD\_VALUES.h File Reference

This file contains the standard values.

```
#include "LSTD_TYPES.h"
```

```
#include "LSTD_COMPILER.h"
```

Include dependency graph for LSTD\_VALUES.h:

## 6.10 main.c File Reference

```
#include "APPS/APPS_main.h"
```

Include dependency graph for main.c:

### Functions

- int [main](#) (void)



## 6.10.1 Function Documentation

### 6.10.1.1 main()

```
int main (
    void )
4 {
5     vAPPS_main();
6
7     for (;;)
8     {
9     }
10 }
```

References [vAPPS\\_main\(\)](#).

Here is the call graph for this function:



# Index

- APPS\_main.c, [23](#)
  - vAPPS\_main, [23](#)
- APPS\_main.h, [24](#)
  - vAPPS\_main, [24](#)
- Bit Manipulation Math Macros, [9](#)
  - CLEAR\_BIT, [10](#)
  - GET\_BIT, [10](#)
  - SET\_BIT, [9](#)
  - TOGGLE\_BIT, [10](#)
- CLEAR\_BIT
  - Bit Manipulation Math Macros, [10](#)
- Compiler standard macros, [11](#)
  - CONST, [11](#)
  - CONSTP2CONST, [13](#)
  - CONSTP2VAR, [12](#)
  - P2CONST, [12](#)
  - P2FUNC, [13](#)
  - P2VAR, [12](#)
  - STATIC, [11](#)
  - VOLATILE, [12](#)
- CONST
  - Compiler standard macros, [11](#)
- CONSTP2CONST
  - Compiler standard macros, [13](#)
- CONSTP2VAR
  - Compiler standard macros, [12](#)
- FALSE
  - Standard values, [21](#)
- GET\_BIT
  - Bit Manipulation Math Macros, [10](#)
- github/workspace/README.md, [25](#)
- Hardware registers macro-functions, [13](#)
  - REGISTER, [14](#)
  - REGISTER\_ADDRESS, [14](#)
  - REGISTER\_U16, [15](#)
  - REGISTER\_U32, [16](#)
  - REGISTER\_U8, [15](#)
- LIB/LSTD\_BITMATH.h, [25](#)
- LIB/LSTD\_COMPILER.h, [26](#)
- LIB/LSTD\_HW\_REGS.h, [26](#)
- LIB/LSTD\_TYPES.h, [27](#)
- LIB/LSTD\_VALUES.h, [28](#)
- main
  - main.c, [29](#)
- main.c, [28](#)
  - main, [29](#)
- NULL
  - Standard values, [21](#)
- P2CONST
  - Compiler standard macros, [12](#)
- P2FUNC
  - Compiler standard macros, [13](#)
- P2VAR
  - Compiler standard macros, [12](#)
- README.md, [25](#)
- REGISTER
  - Hardware registers macro-functions, [14](#)
- REGISTER\_ADDRESS
  - Hardware registers macro-functions, [14](#)
- REGISTER\_U16
  - Hardware registers macro-functions, [15](#)
- REGISTER\_U32
  - Hardware registers macro-functions, [16](#)
- REGISTER\_U8
  - Hardware registers macro-functions, [15](#)
- SET\_BIT
  - Bit Manipulation Math Macros, [9](#)
- Standard data types, [17](#)
  - t\_bool, [18](#)
  - t\_c8, [18](#)
  - t\_fl32, [20](#)
  - t\_fl64, [20](#)
  - t\_s16, [19](#)
  - t\_s32, [19](#)
  - t\_s64, [19](#)
  - t\_s8, [18](#)
  - t\_u16, [18](#)
  - t\_u32, [19](#)
  - t\_u64, [19](#)
  - t\_u8, [18](#)
- Standard Library, [16](#)
- Standard values, [20](#)
  - FALSE, [21](#)
  - NULL, [21](#)
  - TRUE, [21](#)
- STATIC
  - Compiler standard macros, [11](#)
- t\_bool
  - Standard data types, [18](#)
- t\_c8

- Standard data types, [18](#)
- t\_fl32
  - Standard data types, [20](#)
- t\_fl64
  - Standard data types, [20](#)
- t\_s16
  - Standard data types, [19](#)
- t\_s32
  - Standard data types, [19](#)
- t\_s64
  - Standard data types, [19](#)
- t\_s8
  - Standard data types, [18](#)
- t\_u16
  - Standard data types, [18](#)
- t\_u32
  - Standard data types, [19](#)
- t\_u64
  - Standard data types, [19](#)
- t\_u8
  - Standard data types, [18](#)
- TOGGLE\_BIT
  - Bit Manipulation Math Macros, [10](#)
- TRUE
  - Standard values, [21](#)
- vAPPS\_main
  - APPS\_main.c, [23](#)
  - APPS\_main.h, [24](#)
- VOLATILE
  - Compiler standard macros, [12](#)