SWC_EEPROM

Version v1.0
8/25/2023 6:06:00 PM

Table of Contents

Data Structure Index

Data Structures

Here are the data structures with brief descriptions:

File Index

File List

Here is a list of all files with brief descriptions:

Data Structure Documentation

EECR_type Union Reference

: Type define of Union bit field of "EEPROM Control Register"

`#include <EEPROM_priv.h>`
Collaboration diagram for EECR_type:

Data Fields

- u8 u_Reg
- struct {
- __IO u8 m_EERE: 1
- __IO u8 m_EEWE: 1
- __IO u8 m_EEMWE: 1
- __IO u8 m_EERIE: 1
- __IO u8: 4
- } sBits

Detailed Description

: Type define of Union bit field of "EEPROM Control Register"

Type : Union **Unit** : None

Field Documentation

__IO u8 m_EEMWE

EEPROM Master Write Enable

__IO u8 m_EERE

EEPROM Read Enable

__IO u8 m_EERIE

EEPROM Ready Interrupt Enable

__IO u8 m_EEWE

EEPROM Write Enable

struct { ... } sBits

__IO u8

u8 u_Reg

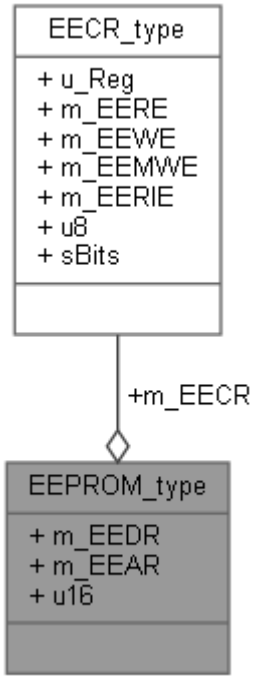
The documentation for this union was generated from the following file:

EEPROM_priv.h

EEPROM_type Struct Reference

: EEPROM Registers

`#include <EEPROM_priv.h>`
Collaboration diagram for EEPROM_type:



Data Fields

- __IO EECR_type m_EECR
- __IO u8 m_EEDR
- __IO u16 m_EEAR: 10
- __IO u16: 6

Detailed Description

: EEPROM Registers

Type : Struct **Unit** : None

Field Documentation

__IO u16 m_EEAR

__IO EECR_type m_EECR

__IO u8 m_EEDR

__IO u16

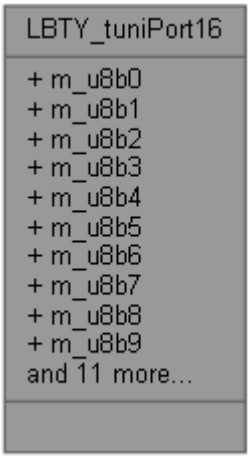
The documentation for this struct was generated from the following file:

EEPROM_priv.h

LBTY_tuniPort16 Union Reference

`#include <LBTY_int.h>`

Collaboration diagram for LBTY_tuniPort16:



Data Fields

- struct {
- u8 m_u8b0:1
- u8 m_u8b1:1
- u8 m_u8b2:1
- u8 m_u8b3:1
- u8 m_u8b4:1
- u8 m_u8b5:1
- u8 m_u8b6:1
- u8 m_u8b7:1
- u8 m_u8b8:1
- u8 m_u8b9:1
- u8 m_u8b10:1
- u8 m_u8b11:1
- u8 m_u8b12:1
- u8 m_u8b13:1
- u8 m_u8b14:1
- u8 m_u8b15:1
- } sBits
- struct {
- u8 m_u8low
- u8 m_u8high
- } sBytes
- u16 u_u16Word

Field Documentation

u8 m_u8b0

u8 m_u8b1

u8 m_u8b10

u8 m_u8b11

u8 m_u8b12

u8 m_u8b13

u8 m_u8b14

u8 m_u8b15

u8 m_u8b2

u8 m_u8b3

u8 m_u8b4

u8 m_u8b5

u8 m_u8b6

u8 m_u8b7

u8 m_u8b8

u8 m_u8b9

u8 m_u8high

u8 m_u8low

struct { ... } sBits

struct { ... } sBytes

u16 u_u16Word

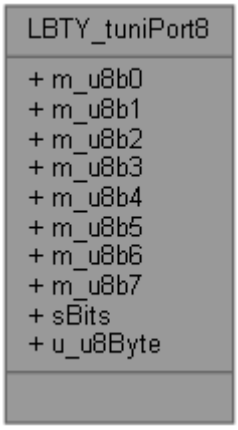
The documentation for this union was generated from the following file:

- H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_BSW/LBTY_int.h

LBTY_tuniPort8 Union Reference

`#include <LBTY_int.h>`

Collaboration diagram for LBTY_tuniPort8:



Data Fields

- struct {
- u8 m_u8b0:1
- u8 m_u8b1:1
- u8 m_u8b2:1
- u8 m_u8b3:1
- u8 m_u8b4:1
- u8 m_u8b5:1
- u8 m_u8b6:1
- u8 m_u8b7:1
- } sBits
- u8 u_u8Byte

Detailed Description

Union Byte bit by bit

Field Documentation

u8 m_u8b0

u8 m_u8b1

u8 m_u8b2

u8 m_u8b3

u8 m_u8b4

u8 m_u8b5

u8 m_u8b6

u8 m_u8b7

struct { ... } sBits

u8 u_u8Byte

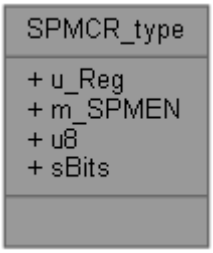
The documentation for this union was generated from the following file:

- H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_BSW/LBTY_int.h

SPMCR_type Union Reference

: Type define of Union bit field of "MCU Control Register"

`#include <EEPROM_priv.h>`
Collaboration diagram for SPMCR_type:



Data Fields

- u8 u_Reg
- struct {
- __IO u8 m_SPMEN: 1
- __I u8: 7
- } sBits

Detailed Description

: Type define of Union bit field of "MCU Control Register"

Type : Union **Unit** : None

Field Documentation

__IO u8 m_SPMEN

Store Program Memory Enable

struct { ... } sBits

__I u8

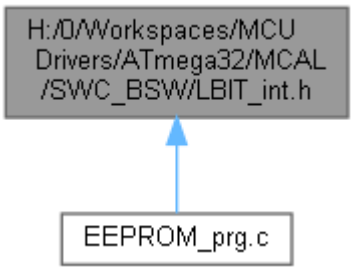
u8 u_Reg

The documentation for this union was generated from the following file:
EEPROM_priv.h

File Documentation

H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_BSW/LBIT_int.h File Reference

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



Macros

- #define _BV(bit) (1u<<(bit))
- #define SET_BIT(REG, bit) ((REG) |= (1u<<(bit)))
- #define CLR_BIT(REG, bit) ((REG) &= ~(1u<<(bit)))
- #define TOG_BIT(REG, bit) ((REG) ^= (1u<<(bit)))
- #define SET_BYTE(REG, bit) ((REG) |= (0xFFu<<(bit)))
- #define CLR_BYTE(REG, bit) ((REG) &= ~(0xFFu<<(bit)))
- #define TOG_BYTE(REG, bit) ((REG) ^= (0xFFu<<(bit)))
- #define SET_MASK(REG, MASK) ((REG) |= (MASK))
- #define CLR_MASK(REG, MASK) ((REG) &= ~(MASK))
- #define TOG_MASK(REG, MASK) ((REG) ^= (MASK))
- #define GET_MASK(REG, MASK) ((REG) & (MASK))
- #define SET_REG(REG) ((REG) = ~(0u))
- #define CLR_REG(REG) ((REG) = (0u))
- #define TOG_REG(REG) ((REG) ^= ~(0u))
- #define GET_BIT(REG, bit) (((REG)>>(bit)) & 0x01u)
- #define GET_NIB(REG, bit) (((REG)>>(bit)) & 0x0Fu)
- #define GET_BYTE(REG, bit) (((REG)>>(bit)) & 0xFFu)
- #define ASSIGN_BIT(REG, bit, value) ((REG) = ((REG) & ~(0x01u<<(bit))) | (((value) & 0x01u)<<(bit)))
- #define ASSIGN_NIB(REG, bit, value) ((REG) = ((REG) & ~(0x0Fu<<(bit))) | (((value) & 0x0Fu)<<(bit)))
- #define ASSIGN_BYTE(REG, bit, value) ((REG) = ((REG) & ~(0xFFu<<(bit))) | (((value) & 0xFFu)<<(bit)))
- #define CON_u8Bits(b7, b6, b5, b4, b3, b2, b1, b0)

  (0b##b7##b6##b5##b4##b3##b2##b1##b0)
- #define CON_u16Bits(b15, b14, b13, b12, b11, b10, b9, b8, b7, b6, b5, b4, b3, b2, b1, b0)

  (0b##b15##b14##b13##b12##b11##b10##b9##b8##b7##b6##b5##b4##b3##b2##b1##b0)

Macro Definition Documentation

#define _BV(bit) (1u<<(bit))

#define ASSIGN_BIT(REG, bit, value) ((REG) = ((REG) & ~(0x01u<<(bit))) | (((value) & 0x01u)<<(bit)))

#define ASSIGN_BYTE(REG, bit, value) ((REG) = ((REG) & ~(0xFFu<<(bit))) | (((value) & 0xFFu)<<(bit)))

#define ASSIGN_NIB(REG, bit, value) ((REG) = ((REG) & ~(0x0Fu<<(bit))) | (((value) & 0x0Fu)<<(bit)))

#define CLR_BIT(REG, bit) ((REG) &= ~(1u<<(bit)))

#define CLR_BYTE(REG, bit) ((REG) &= ~(0xFFu<<(bit)))

#define CLR_MASK(REG, MASK) ((REG) &= ~(MASK))

#define CLR_REG(REG) ((REG) = (0u))

#define CON_u16Bits(b15, b14, b13, b12, b11, b10, b9, b8, b7, b6, b5, b4, b3, b2, b1, b0)

(0b##b15##b14##b13##b12##b11##b10##b9##b8##b7##b6##b5##b4##b3##b2##b1##b0)

#define CON_u8Bits(b7, b6, b5, b4, b3, b2, b1, b0)

(0b##b7##b6##b5##b4##b3##b2##b1##b0)

#define GET_BIT(REG, bit) (((REG)>>(bit)) & 0x01u)

#define GET_BYTE(REG, bit) (((REG)>>(bit)) & 0xFFu)

#define GET_MASK(REG, MASK) ((REG) & (MASK))

#define GET_NIB(REG, bit) (((REG)>>(bit)) & 0x0Fu)

#define SET_BIT(REG, bit) ((REG) |= (1u<<(bit)))

Bitwise Operation

```
#define SET_BYTE( REG,   bit)  ((REG) |=   (0xFFu<<(bit)))

#define SET_MASK( REG,   MASK)  ((REG) |=   (MASK))

#define SET_REG( REG)  ((REG)   = ~(0u))

#define TOG_BIT( REG,   bit)  ((REG) ^=   (1u<<(bit)))

#define TOG_BYTE( REG,   bit)  ((REG) ^=   (0xFFu<<(bit)))

#define TOG_MASK( REG,   MASK)  ((REG) ^=   (MASK))

#define TOG_REG( REG)  ((REG) ^= ~(0u))
```

LBIT_int.h

Go to the documentation of this file.

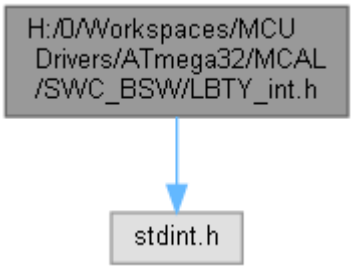
```
1 /*
**************************************************************************** */
2 /* ********************** FILE DEFINITION SECTION ************************** */
3 /* ************************************************************************** */
4 /* File Name    : LBIT_int.h                                                  */
5 /* Author       : MAAM                                                        */
6 /* Version      : v01                                                         */
7 /* date         : Mar 24, 2023                                                */
8 /* description : Bitwise Library                                              */
9 /* ************************************************************************** */
10 /* ************************ HEADER FILES INCLUDES ************************  */
11 /* ************************************************************************** */
12
13 #ifndef LBIT_INT_H_
14 #define LBIT_INT_H_
15
16 /* ************************************************************************** */
17 /* ********************** TYPE_DEF/STRUCT/ENUM SECTION ********************** */
18 /* ************************************************************************** */
19
20 /* ************************************************************************** */
21 /* ************************** MACRO/DEFINE SECTION ************************** */
22 /* ************************************************************************** */
23
24 #define _BV(bit)                                          (1u<<(bit))
25
27 #define SET_BIT(REG, bit)                                 ((REG) |=  (1u<<(bit)))
28 #define CLR_BIT(REG, bit)                                 ((REG) &= ~(1u<<(bit)))
29 #define TOG_BIT(REG, bit)                                 ((REG) ^=  (1u<<(bit)))
30
31 #define SET_BYTE(REG, bit)                                ((REG) |=  (0xFFu<<(bit)))
32 #define CLR_BYTE(REG, bit)                                ((REG) &= ~(0xFFu<<(bit)))
33 #define TOG_BYTE(REG, bit)                                ((REG) ^=  (0xFFu<<(bit)))
34
35 #define SET_MASK(REG, MASK)                               ((REG) |=  (MASK))
36 #define CLR_MASK(REG, MASK)                               ((REG) &= ~(MASK))
37 #define TOG_MASK(REG, MASK)                               ((REG) ^=  (MASK))
38 #define GET_MASK(REG, MASK)                               ((REG) &   (MASK))
39
40 #define SET_REG(REG)                                      ((REG)  = ~(0u))
41 #define CLR_REG(REG)                                      ((REG)  =  (0u))
42 #define TOG_REG(REG)                                      ((REG) ^= ~(0u))
43
44 #define GET_BIT(REG, bit)                                 (((REG)>>(bit)) & 0x01u)
45 #define GET_NIB(REG, bit)                                 (((REG)>>(bit)) & 0x0Fu)
46 #define GET_BYTE(REG, bit)                                (((REG)>>(bit)) & 0xFFu)
47
48 #define ASSIGN_BIT(REG, bit, value)                      ((REG) = ((REG) & ~(0x01u<<(bit)))
| (((value) & 0x01u)<<(bit)))
49 #define ASSIGN_NIB(REG, bit, value)                      ((REG) = ((REG) & ~(0x0Fu<<(bit)))
| (((value) & 0x0Fu)<<(bit)))
50 #define ASSIGN_BYTE(REG, bit, value)                     ((REG) = ((REG) & ~(0xFFu<<(bit)))
| (((value) & 0xFFu)<<(bit)))
51
52 /*
53 #define ASSIGN_BIT(REG,bit,value)                         do{
\
54                                                            REG &= ~(0x01u<<bit);
\
55                                                            REG |=  ((value & 0x01u)<<bit);
\
56                                                           }while(0)
57 */
58
59 /*           bits together in an u8 register           */
60 #define CON_u8Bits(b7, b6, b5, b4, b3, b2, b1, b0)
\
61
(0b##b7##b6##b5##b4##b3##b2##b1##b0)
62
63 /*           bits together in an u16 register          */
64 #define CON_u16Bits(b15, b14, b13, b12, b11, b10, b9, b8, b7, b6, b5, b4, b3, b2, b1,
b0)   \
```

```
65
(0b##b15##b14##b13##b12##b11##b10##b9##b8##b7##b6##b5##b4##b3##b2##b1##b0)
66
67 /* ************************************************************************** */
68 /* ***************************** CONST SECTION ****************************** */
69 /* ************************************************************************** */
70
71 /* ************************************************************************** */
72 /* ***************************** VARIABLE SECTION ***************************** */
73 /* ************************************************************************** */
74
75 /* ************************************************************************** */
76 /* ***************************** FUNCTION SECTION ***************************** */
77 /* ************************************************************************** */
78
79
80 #endif /* LBIT_INT_H_ */
81 /***************************** E N D (LBIT_int.h) ******************************/
```

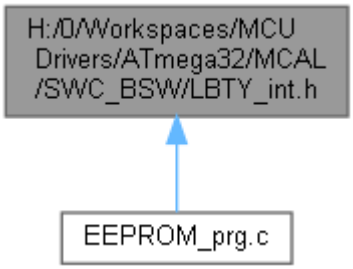
H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_BSW/LBTY_int.h File Reference

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

Include dependency graph for LBTY_int.h:



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



Data Structures

- union LBTY_tuniPort8union LBTY_tuniPort16

Macros

- #define __IO volatile
- #define __O volatile
- #define __I volatile const
- #define LBTY_u8vidNOP()
- #define LBTY_NULL ((void *) 0U)
- #define LBTY_u8ZERO ((u8)0x00U)
- #define LBTY_u8MAX ((u8)0xFFU)
- #define LBTY_s8MAX ((s8)0x7F)
- #define LBTY_s8MIN ((s8)0x80)
- #define LBTY_u16ZERO ((u16)0x0000U)
- #define LBTY_u16MAX ((u16)0xFFFFU)
- #define LBTY_s16MAX ((u16)0x7FFF)
- #define LBTY_s16MIN ((u16)0x8000)
- #define LBTY_u32ZERO ((u32)0x00000000UL)
- #define LBTY_u32MAX ((u32)0xFFFFFFFFUL)
- #define LBTY_s32MAX ((u32)0x7FFFFFFFL)
- #define LBTY_s32MIN ((u32)0x80000000L)
- #define LBTY_u64ZERO ((u64)0x0000000000000000ULL)
- #define LBTY_u64MAX ((u64)0xFFFFFFFFFFFFFFFFULL)
- #define LBTY_s64MAX ((u64)0x7FFFFFFFFFFFFFFFLL)
- #define LBTY_s64MIN ((u64)0x8000000000000000LL)

Typedefs

- typedef uint8_t u8
- typedef uint16_t u16
- typedef uint32_t u32
- typedef uint64_t u64
- typedef int8_t s8
- typedef int16_t s16
- typedef int32_t s32
- typedef int64_t s64
- typedef float f32
- typedef double f64
- typedef u8 * pu8
- typedef u16 * pu16
- typedef u32 * pu32
- typedef u64 * pu64
- typedef s8 * ps8
- typedef s16 * ps16
- typedef s32 * ps32
- typedef s64 * ps64

Enumerations

- enum LBTY_tenuFlagStatus { LBTY_RESET = 0, LBTY_SET = !LBTY_RESET }
- enum LBTY_tenuBoolean { LBTY_TRUE = 0x55, LBTY_FALSE = 0xAA }
- enum LBTY_tenuErrorStatus { LBTY_OK = (u16)0, LBTY_NOK, LBTY_NULL_POINTER, LBTY_INDEX_OUT_OF_RANGE, LBTY_NO_MASTER_CHANNEL, LBTY_READ_ERROR, LBTY_WRITE_ERROR, LBTY_UNDEFINED_ERROR, LBTY_IN_PROGRESS }

Macro Definition Documentation

#define __I volatile const

#define __IO volatile

#define __O volatile

#define LBTY_NULL ((void *) 0U)

#define LBTY_s16MAX ((u16)0x7FFF)

#define LBTY_s16MIN ((u16)0x8000)

#define LBTY_s32MAX ((u32)0x7FFFFFFFL)

#define LBTY_s32MIN ((u32)0x80000000L)

#define LBTY_s64MAX ((u64)0x7FFFFFFFFFFFFFFFLL)

#define LBTY_s64MIN ((u64)0x8000000000000000LL)

#define LBTY_s8MAX ((s8)0x7F)

#define LBTY_s8MIN ((s8)0x80)

#define LBTY_u16MAX ((u16)0xFFFFU)

#define LBTY_u16ZERO ((u16)0x0000U)

#define LBTY_u32MAX ((u32)0xFFFFFFFFUL)

#define LBTY_u32ZERO ((u32)0x00000000UL)

#define LBTY_u64MAX ((u64)0xFFFFFFFFFFFFFFFFULL)

#define LBTY_u64ZERO ((u64)0x0000000000000000ULL)

#define LBTY_u8MAX ((u8)0xFFU)

#define LBTY_u8vidNOP()

#define LBTY_u8ZERO ((u8)0x00U)

Data Types Limitation

Typedef Documentation

typedef float f32

Standard Real Decimal number

typedef double f64

typedef s16* ps16

typedef s32* ps32

typedef s64* ps64

typedef s8* ps8

Standard Pointer to Signed Byte/Word/Long_Word

typedef u16* pu16

typedef u32* pu32

typedef u64* pu64

typedef u8* pu8

Standard Pointer to Unsigned Byte/Word/Long_Word

typedef int16_t s16

typedef int32_t s32

typedef int64_t s64

typedef int8_t s8

Standard Signed Byte/Word/Long_Word

typedef uint16_t u16

typedef uint32_t u32

typedef uint64_t u64

typedef uint8_t u8

Data Types New Definitions Standard Unsigned Byte/Word/Long_Word

---

Enumeration Type Documentation

enum LBTY_tenuBoolean

Boolean type

Enumerator:

LBTY_TRUE	
LBTY_FALSE	

```
96              {
97   LBTY_TRUE = 0x55,
98   LBTY_FALSE = 0xAA
99 } LBTY_tenuBoolean;
```

enum LBTY_tenuErrorStatus

Error Return type

Enumerator:

LBTY_OK	
LBTY_NOK	
LBTY_NULL_POINTER	
LBTY_INDEX_OUT_OF_RANGE	
LBTY_NO_MASTER_CHANNEL	
LBTY_READ_ERROR	
LBTY_WRITE_ERROR	
LBTY_UNDEFINED_ERROR	
LBTY_IN_PROGRESS	

```
102             {
103   LBTY_OK = (u16)0,
104   LBTY_NOK,
105   LBTY_NULL_POINTER,
106   LBTY_INDEX_OUT_OF_RANGE,
107   LBTY_NO_MASTER_CHANNEL,
108   LBTY_READ_ERROR,
109   LBTY_WRITE_ERROR,
110   LBTY_UNDEFINED_ERROR,
111   LBTY_IN_PROGRESS        /* Error is not available, wait for availability */
112 } LBTY_tenuErrorStatus;
```

enum LBTY_tenuFlagStatus

Flag Status type

Enumerator:

LBTY_RESET	
LBTY_SET	

```
90              {
91    LBTY_RESET = 0,
92    LBTY_SET = !LBTY_RESET
93 } LBTY_tenuFlagStatus;
```

LBTY_int.h

Go to the documentation of this file.

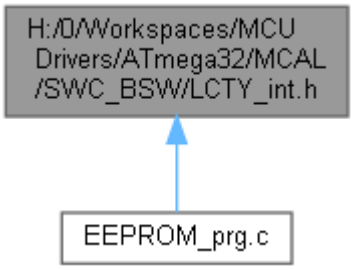
```
1 /*
2 /* ********************** FILE DEFINITION SECTION ************************** */
3 /* ************************************************************************* */
4 /* File Name   : LBTY_int.h                                                  */
5 /* Author      : MAAM                                                        */
6 /* Version     : v01                                                         */
7 /* date        : Mar 23, 2023                                                */
8 /* description : Basic Library                                               */
9 /* ************************************************************************* */
10 /* ************************ HEADER FILES INCLUDES ************************  */
11 /* ************************************************************************* */
12
13 #ifndef _LBTY_INT_H_
14 #define _LBTY_INT_H_
15
16 #include <stdint.h>
17
18 /* ************************************************************************* */
19 /* ****************************** TYPE_DEF SECTION ************************* */
20 /* ************************************************************************* */
21
24 typedef uint8_t     u8 ;
25 typedef uint16_t    u16;
26 typedef uint32_t    u32;
27 typedef uint64_t    u64;
28
30 typedef int8_t      s8 ;
31 typedef int16_t     s16;
32 typedef int32_t     s32;
33 typedef int64_t     s64;
34
36 typedef float       f32;
37 typedef double      f64;
38
40 typedef u8*         pu8 ;
41 typedef u16*        pu16;
42 typedef u32*        pu32;
43 typedef u64*        pu64;
44
46 typedef s8*         ps8 ;
47 typedef s16*        ps16;
48 typedef s32*        ps32;
49 typedef s64*        ps64;
50
51 /* ************************************************************************* */
52 /* ************************** MACRO/DEFINE SECTION ************************* */
53 /* ************************************************************************* */
54
55 /*****************************************************************************/
56 #define __IO    volatile
57 #define __O     volatile
58 #define __I     volatile const
59 /*****************************************************************************/
60
61 #define LBTY_u8vidNOP()
62 #define LBTY_NULL       ((void *) 0U)
63
65 #define LBTY_u8ZERO     ((u8)0x00U)
66 #define LBTY_u8MAX      ((u8)0xFFU)
67 #define LBTY_s8MAX      ((s8)0x7F )
68 #define LBTY_s8MIN      ((s8)0x80 )
69
70 #define LBTY_u16ZERO    ((u16)0x0000U)
71 #define LBTY_u16MAX     ((u16)0xFFFFU)
72 #define LBTY_s16MAX     ((u16)0x7FFF )
73 #define LBTY_s16MIN     ((u16)0x8000 )
74
75 #define LBTY_u32ZERO    ((u32)0x00000000UL)
76 #define LBTY_u32MAX     ((u32)0xFFFFFFFFUL)
77 #define LBTY_s32MAX     ((u32)0x7FFFFFFFL )
78 #define LBTY_s32MIN     ((u32)0x80000000L )
79
```

```
80 #define LBTY_u64ZERO     ((u64)0x0000000000000000ULL)
81 #define LBTY_u64MAX      ((u64)0xFFFFFFFFFFFFFFFFULL)
82 #define LBTY_s64MAX      ((u64)0x7FFFFFFFFFFFFFFFLL )
83 #define LBTY_s64MIN      ((u64)0x8000000000000000LL )
84
85 /* ********************************************************************************* */
86 /* ****************************** ENUM SECTION ************************************* */
87 /* ********************************************************************************* */
88
90 typedef enum {
91     LBTY_RESET = 0,
92     LBTY_SET = !LBTY_RESET
93 } LBTY_tenuFlagStatus;
94
96 typedef enum {
97   LBTY_TRUE = 0x55,
98   LBTY_FALSE = 0xAA
99 } LBTY_tenuBoolean;
100
102 typedef enum {
103   LBTY_OK = (u16)0,
104   LBTY_NOK,
105   LBTY_NULL_POINTER,
106   LBTY_INDEX_OUT_OF_RANGE,
107   LBTY_NO_MASTER_CHANNEL,
108   LBTY_READ_ERROR,
109   LBTY_WRITE_ERROR,
110   LBTY_UNDEFINED_ERROR,
111   LBTY_IN_PROGRESS         /* Error is not available, wait for availability */
112 } LBTY_tenuErrorStatus;
113
114 /* ********************************************************************************* */
115 /* ****************************** STRUCT SECTION ************************************* */
116 /* ********************************************************************************* */
117
119 typedef union {
120   struct {
121     u8 m_u8b0 :1;       // LSB
122     u8 m_u8b1 :1;
123     u8 m_u8b2 :1;
124     u8 m_u8b3 :1;
125     u8 m_u8b4 :1;
126     u8 m_u8b5 :1;
127     u8 m_u8b6 :1;
128     u8 m_u8b7 :1;       // MSB
129   } sBits;
130   u8 u_u8Byte;
131 } LBTY_tuniPort8;
132
133 typedef union {
134   struct  {
135     u8 m_u8b0  :1;      // LSB
136     u8 m_u8b1  :1;
137     u8 m_u8b2  :1;
138     u8 m_u8b3  :1;
139     u8 m_u8b4  :1;
140     u8 m_u8b5  :1;
141     u8 m_u8b6  :1;
142     u8 m_u8b7  :1;
143     u8 m_u8b8  :1;
144     u8 m_u8b9  :1;
145     u8 m_u8b10 :1;
146     u8 m_u8b11 :1;
147     u8 m_u8b12 :1;
148     u8 m_u8b13 :1;
149     u8 m_u8b14 :1;
150     u8 m_u8b15 :1;      // MSB
151   } sBits;
152   struct {
153     u8 m_u8low;
154     u8 m_u8high;
155   } sBytes;
156   u16 u_u16Word;
157 } LBTY_tuniPort16;
158
159 /* ********************************************************************************* */
160 /* ***************************** FUNCTION SECTION ************************************ */
```

```
161 /* ************************************************************************** */
162
163
164 #endif /* _LBTY_INT_H_ */
165 /*************************** E N D (LBTY_int.h) ****************************/
```

H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_BSW/LCTY_int.h File Reference

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



Macros

- #define LCTY_PROGMEM __attribute__((__progmem__))
- #define LCTY_PURE __attribute__((__pure__))
- #define LCTY_INLINE __attribute__((always_inline)) static inline
- #define LCTY_INTERRUPT __attribute__((interrupt))
- #define CTY_PACKED __attribute__((__packed__))
- #define LCTY_CONST __attribute__((__const__))
- #define LCTY_DPAGE __attribute__((dp))
- #define LCTY_NODPAGE __attribute__((nodp))
- #define LCTY_SECTION(section) __attribute__((section(# section)))
- #define LCTY_ASM(cmd) __asm__ __volatile__ (# cmd ::)

Macro Definition Documentation

#define CTY_PACKED __attribute__((__packed__))

#define LCTY_ASM(cmd) __asm__ __volatile__ (# cmd ::)

#define LCTY_CONST __attribute__((__const__))

#define LCTY_DPAGE __attribute__((dp))

#define LCTY_INLINE __attribute__((always_inline)) static inline

#define LCTY_INTERRUPT __attribute__((interrupt))

#define LCTY_NODPAGE __attribute__((nodp))

#define LCTY_PROGMEM __attribute__((__progmem__))

#define LCTY_PURE __attribute__((__pure__))

#define LCTY_SECTION(section) __attribute__((section(# section)))

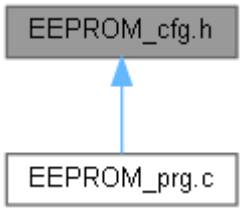
LCTY_int.h

Go to the documentation of this file.

```
1 /*
2 /* ********************** FILE DEFINITION SECTION ************************** */
3 /* ************************************************************************* */
4 /* File Name   : LCTY_int.h                                                  */
5 /* Author      : MAAM                                                        */
6 /* Version     : v00                                                         */
7 /* date        : Apr 26, 2023                                                */
8 /* description : Compiler Library                                            */
9 /* ************************************************************************* */
10 /* ************************ HEADER FILES INCLUDES ************************  */
11 /* ************************************************************************* */
12
13 #ifndef LCTY_INT_H_
14 #define LCTY_INT_H_
15
16 /* ************************************************************************* */
17 /* ********************** TYPE_DEF/STRUCT/ENUM SECTION ********************** */
18 /* ************************************************************************* */
19
20 /* ************************************************************************* */
21 /* ************************** MACRO/DEFINE SECTION ************************** */
22 /* ************************************************************************* */
23
24 /* prog memory attribute */
25 #define LCTY_PROGMEM                __attribute__((__progmem__))
26
27 /* pure attribute */
28 #define LCTY_PURE                   __attribute__((__pure__))
29
30 /* Abstraction for inlining */
31 //#define LCTY_INLINE               static inline
32 #define LCTY_INLINE                 __attribute__((always_inline)) static inline
33
34 /* define function as interrupt handler */
35 #define LCTY_INTERRUPT              __attribute__((interrupt))
36
37 /* Memory packed to pass Memory padding */
38 #define CTY_PACKED                  __attribute__((__packed__))
39
40 /* Const attribute */
41 #define LCTY_CONST                  __attribute__((__const__))
42
43 /* place variable in direct page */
44 #define LCTY_DPAGE                  __attribute__((dp))
45
46 /* do not place variable in direct page */
47 #define LCTY_NODPAGE                __attribute__((nodp))
48
49 /* Sections */
50 #define LCTY_SECTION(section)       __attribute__((section( # section)))
51
52 /* Abstraction for assembly command */
53 # define LCTY_ASM(cmd)              __asm__ __volatile__ ( # cmd ::)
54
55 /* ************************************************************************* */
56 /* ***************************** CONST SECTION ***************************** */
57 /* ************************************************************************* */
58
59 /* ************************************************************************* */
60 /* ***************************** VARIABLE SECTION ************************** */
61 /* ************************************************************************* */
62
63 /* ************************************************************************* */
64 /* ***************************** FUNCTION SECTION ************************** */
65 /* ************************************************************************* */
66
67
68 #endif /* LCTY_INT_H_ */
69 /*************************** E N D (LCTY_int.h) ****************************/
```

EEPROM_cfg.h File Reference

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



Macros

- #define EEPROM_INTERRUPT_INIT_STATE LBTY_RESET

Macro Definition Documentation

#define EEPROM_INTERRUPT_INIT_STATE LBTY_RESET

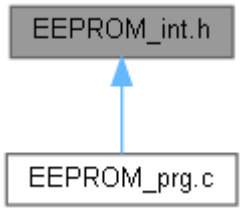
EEPROM_cfg.h

Go to the documentation of this file.

```
1 /*
2 /* ********************** FILE DEFINITION SECTION ************************** */
3 /* ************************************************************************** */
4 /* File Name   : EEPROM_cfg.h                                                 */
5 /* Author      : MAAM                                                         */
6 /* Version     : v01                                                          */
7 /* date        : Apr 30, 2023                                                 */
8 /* ************************************************************************** */
9 /* ************************ HEADER FILES INCLUDES ************************  */
10 /* ************************************************************************** */
11
12 #ifndef EEPROM_CFG_H_
13 #define EEPROM_CFG_H_
14
15 /* ************************************************************************** */
16 /* ********************** TYPE_DEF/STRUCT/ENUM SECTION ********************** */
17 /* ************************************************************************** */
18
19 /* ************************************************************************** */
20 /* ************************** MACRO/DEFINE SECTION ************************** */
21 /* ************************************************************************** */
22
23 #define EEPROM_INTERRUPT_INIT_STATE         LBTY_RESET
24
25 /* ************************************************************************** */
26 /* ***************************** CONST SECTION ****************************** */
27 /* ************************************************************************** */
28
29 /* ************************************************************************** */
30 /* ***************************** VARIABLE SECTION *************************** */
31 /* ************************************************************************** */
32
33 /* ************************************************************************** */
34 /* ***************************** FUNCTION SECTION *************************** */
35 /* ************************************************************************** */
36
37
38 #endif /* EEPROM_CFG_H_ */
39 /***************************** E N D (EEPROM_cfg.h) ******************************/
```

EEPROM_int.h File Reference

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



Functions

- void EEPROM_vidInit (void)
- void EEPROM_vidEnableInterrupt (void)
- void EEPROM_vidDisableInterrupt (void)
- LBTY_tenuErrorStatus EEPROM_u8Erase (u16 u16StartAdd, u16 u16EndAdd)
- LBTY_tenuErrorStatus EEPROM_u8WriteChar (u16 u16Address, u8 u8Data)
- LBTY_tenuErrorStatus EEPROM_u8WriteString (u16 u16Address, u8 *pu8String)
- LBTY_tenuErrorStatus EEPROM_u8ReadChar (u16 u16Address, u8 *pu8Data)
- LBTY_tenuErrorStatus EEPROM_u8ReadString (u16 u16StartAddress, u16 u16EndAddress, u8 *pu8String)
- void EEPROM_vidSetCallBack (void(*CallBack)(void))

Function Documentation

LBTY_tenuErrorStatus EEPROM_u8Erase (u16 *u16StartAdd*, u16 *u16EndAdd*)

```
80                                                                          {
81     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
82     for(u16 i = u16EndAdd - u16StartAdd ; i-- ; ){
83         if(EEPROM_u8WriteChar(u16StartAdd++, LBTY_u8ZERO)){
84             u8RetErrorState = LBTY_NOK;
85             break;
86         }
87     }
88     return u8RetErrorState;
89 }
```

Here is the call graph for this function:

LBTY_tenuErrorStatus EEPROM_u8ReadChar (u16 *u16Address*, u8 * *pu8Data*)

```
141                                                                         {
142     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
143     if(u16Address <= EEPROM_MAX_ADDRESS){
144         while(S_EEPROM->m_EECR.sBits.m_EEWE);
145 
146         S_EEPROM->m_EEAR = u16Address;
147         S_EEPROM->m_EECR.sBits.m_EERE  = LBTY_SET;
148         *pu8Data = S_EEPROM->m_EEDR;
149     }else{
150         u8RetErrorState = LBTY_INDEX_OUT_OF_RANGE;
151     }
152     return u8RetErrorState;
153 }
```

Here is the caller graph for this function:

LBTY_tenuErrorStatus EEPROM_u8ReadString (u16 *u16StartAddress*, u16 *u16EndAddress*, u8 * *pu8String*)

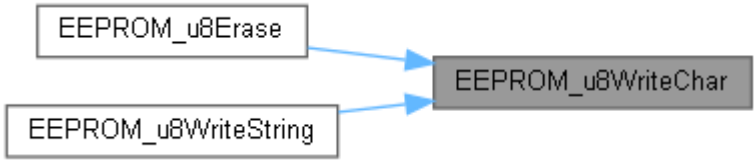
```
161
{
162     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
163     for(int i = u16StartAddress ; i<= u16EndAddress ; i++){
164         if(EEPROM_u8ReadChar(i, pu8String++)){
165             u8RetErrorState = LBTY_READ_ERROR;
166             break;
167         }
168     }
169     return u8RetErrorState;
170 }
```

Here is the call graph for this function:

LBTY_tenuErrorStatus EEPROM_u8WriteChar (u16 *u16Address*, u8 *u8Data*)

```
96                                                                  {
97     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
98     if(u16Address <= EEPROM_MAX_ADDRESS){
99         while(S_EEPROM->m_EECR.sBits.m_EEWE);
100         while(S_SPMCR->sBits.m_SPMEN);
101
102         S_EEPROM->m_EEAR = u16Address;
103         S_EEPROM->m_EEDR = u8Data;
104
105         INTP_vidDisable();
106
107         S_EEPROM->m_EECR.sBits.m_EEMWE = LBTY_SET;
108         S_EEPROM->m_EECR.sBits.m_EEWE  = LBTY_SET;
109
110         INTP_vidEnable();
111
112     }else{
113         u8RetErrorState = LBTY_INDEX_OUT_OF_RANGE;
114     }
115     return u8RetErrorState;
116 }
```

Here is the caller graph for this function:



LBTY_tenuErrorStatus EEPROM_u8WriteString (u16 *u16Address*, u8 * *pu8String*)

```
124                                                                 {
125     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
126     while(*pu8String){
127         if(EEPROM_u8WriteChar(u16Address++, *(pu8String++))){
128             u8RetErrorState = LBTY_WRITE_ERROR;
129             break;
130         }
131     }
132     return u8RetErrorState;
133 }
```

Here is the call graph for this function:

void EEPROM_vidDisableInterrupt (void)

```
71                                      {
72     S_EEPROM->m_EECR.sBits.m_EERIE = LBTY_RESET;
73 }
```

void EEPROM_vidEnableInterrupt (void)

```
62                                          {
63     S_EEPROM->m_EECR.sBits.m_EERIE = LBTY_SET;
64 }
```

void EEPROM_vidInit (void)

```
47                                 {
48     S_EEPROM->m_EEAR = LBTY_u16ZERO;
49     S_EEPROM->m_EEDR = LBTY_u8ZERO;
50
51     S_EEPROM->m_EECR.sBits.m_EERIE = EEPROM_INTERRUPT_INIT_STATE;
52     S_EEPROM->m_EECR.sBits.m_EEMWE = LBTY_RESET;
53     S_EEPROM->m_EECR.sBits.m_EEWE  = LBTY_RESET;
54     S_EEPROM->m_EECR.sBits.m_EERE  = LBTY_RESET;
55 }
```

void EEPROM_vidSetCallBack (void(*)(void) *CallBack*)

```
177                                                    {
178     pfunctionCallBack = CallBack;
179 }
```

EEPROM_int.h

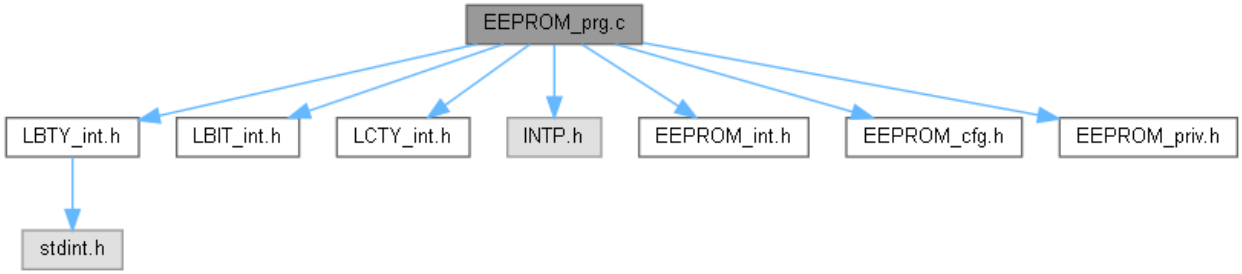
```
Go to the documentation of this file.1 /*
****************************************************************************** */
2 /* ********************** FILE DEFINITION SECTION ************************** */
3 /* ****************************************************************************** */
4 /* File Name    : EEPROM_int.h                                                  */
5 /* Author       : MAAM                                                          */
6 /* Version      : v01                                                           */
7 /* date         : Apr 30, 2023                                                  */
8 /* ****************************************************************************** */
9 /* ************************ HEADER FILES INCLUDES ***************************  */
10 /* ****************************************************************************** */
11
12 #ifndef EEPROM_INT_H_
13 #define EEPROM_INT_H_
14
15 /* ****************************************************************************** */
16 /* ********************** TYPE_DEF/STRUCT/ENUM SECTION ********************** */
17 /* ****************************************************************************** */
18
19 /* ****************************************************************************** */
20 /* ************************** MACRO/DEFINE SECTION ************************** */
21 /* ****************************************************************************** */
22
23 /* ****************************************************************************** */
24 /* ****************************** CONST SECTION ********************************* */
25 /* ****************************************************************************** */
26
27 /* ****************************************************************************** */
28 /* ***************************** VARIABLE SECTION ********************************* */
29 /* ****************************************************************************** */
30
31 /* ****************************************************************************** */
32 /* ***************************** FUNCTION SECTION ********************************* */
33 /* ****************************************************************************** */
34
35 /* ****************************************************************************** */
36 /* Description :    Initialization of the EEPROM                                  */
37 /* Input       :    void                                                          */
38 /* Return      :    void                                                          */
39 /* ****************************************************************************** */
40 void EEPROM_vidInit(void);
41
42 /* ****************************************************************************** */
43 /* Description :    Enable EEPROM Interrupt                                       */
44 /* Input       :    void                                                          */
45 /* Return      :    void                                                          */
46 /* ****************************************************************************** */
47 void EEPROM_vidEnableInterrupt(void);
48
49 /* ****************************************************************************** */
50 /* Description :    Disable EEPROM Interrupt                                        */
51 /* Input       :    void                                                          */
52 /* Return      :    void                                                          */
53 /* ****************************************************************************** */
54 void EEPROM_vidDisableInterrupt(void);
55
56 /* ****************************************************************************** */
57 /* Description :    Erase EEPROM with range                                       */
58 /* Input       :    u16StartAdd, u16EndAdd                                        */
59 /* Return      :    LBTY_tenuErrorStatus                                          */
60 /* ****************************************************************************** */
61 LBTY_tenuErrorStatus EEPROM_u8Erase(u16 u16StartAdd, u16 u16EndAdd);
62
63 /* ****************************************************************************** */
64 /* Description :    Write Char in EEPROM address                                  */
65 /* Input       :    u16Address, u8Data                                            */
66 /* Return      :    LBTY tenuErrorStatus                                          */
67 /* ****************************************************************************** */
68 LBTY tenuErrorStatus EEPROM u8WriteChar(u16 u16Address, u8 u8Data);
69
70 /* ****************************************************************************** */
71 /* Description :    Write String in EEPROM address                                */
72 /* Input       :    u16Address                                                    */
```

```
/* Input/Output:    pu8String                                                 */
/* Return      :    LBTY_tenuErrorStatus                                      */
/* ************************************************************************** */
LBTY_tenuErrorStatus EEPROM_u8WriteString(u16 u16Address, u8* pu8String);

/* ************************************************************************** */
/* Description :    Read Char in EEPROM address                               */
/* Input       :    u16Address                                                */
/* Input/Output:    pu8Data                                                   */
/* Return      :    LBTY_tenuErrorStatus                                      */
/* ************************************************************************** */
LBTY_tenuErrorStatus EEPROM_u8ReadChar(u16 u16Address, u8* pu8Data);

/* ************************************************************************** */
/* Description :    Read String in EEPROM address                             */
/* Input       :    u16StartAddress, u16EndAddress                            */
/* Input/Output:    pu8String                                                 */
/* Return      :    LBTY_tenuErrorStatus                                      */
/* ************************************************************************** */
LBTY_tenuErrorStatus EEPROM_u8ReadString(u16 u16StartAddress, u16 u16EndAddress, u8*
pu8String);

/* ************************************************************************** */
/* Description :    Set EEPROM Interrupt call back                            */
/* Input       :    CallBack                                                  */
/* Return      :    void                                                      */
/* ************************************************************************** */
void EEPROM_vidSetCallBack(void (*CallBack)(void));

#endif /* EEPROM_INT_H_ */
/*************************** E N D (EEPROM_int.h) ****************************/
```

EEPROM_prg.c File Reference

```
#include "LBTY_int.h"
#include "LBIT_int.h"
#include "LCTY_int.h"
#include "INTP.h"
#include "EEPROM_int.h"
#include "EEPROM_cfg.h"
#include "EEPROM_priv.h"
```

Include dependency graph for EEPROM_prg.c:



Functions

- void EEPROM_vidInit (void)
- void EEPROM_vidEnableInterrupt (void)
- void EEPROM_vidDisableInterrupt (void)
- LBTY_tenuErrorStatus EEPROM_u8Erase (u16 u16StartAdd, u16 u16EndAdd)
- LBTY_tenuErrorStatus EEPROM_u8WriteChar (u16 u16Address, u8 u8Data)
- LBTY_tenuErrorStatus EEPROM_u8WriteString (u16 u16Address, u8 *pu8String)
- LBTY_tenuErrorStatus EEPROM_u8ReadChar (u16 u16Address, u8 *pu8Data)
- LBTY_tenuErrorStatus EEPROM_u8ReadString (u16 u16StartAddress, u16 u16EndAddress, u8 *pu8String)
- void EEPROM_vidSetCallBack (void(*CallBack)(void))
- ISR (EE_RDY_vect)

Variables

- void(* pfunctionCallBack)(void)

Function Documentation

LBTY_tenuErrorStatus EEPROM_u8Erase (u16 *u16StartAdd*, u16 *u16EndAdd*)

```
80                                                                          {
81      LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
82      for(u16 i = u16EndAdd - u16StartAdd ; i-- ; ){
83          if(EEPROM_u8WriteChar(u16StartAdd++, LBTY_u8ZERO)){
84              u8RetErrorState = LBTY_NOK;
85              break;
86          }
87      }
88      return u8RetErrorState;
89  }
```

Here is the call graph for this function:

LBTY_tenuErrorStatus EEPROM_u8ReadChar (u16 *u16Address*, u8 * *pu8Data*)

```
141                                                                         {
```

```
142     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
143     if(u16Address <= EEPROM_MAX_ADDRESS){
144         while(S_EEPROM->m_EECR.sBits.m_EEWE);
145 
146         S_EEPROM->m_EEAR = u16Address;
147         S_EEPROM->m_EECR.sBits.m_EERE  = LBTY_SET;
148         *pu8Data = S_EEPROM->m_EEDR;
149     }else{
150         u8RetErrorState = LBTY_INDEX_OUT_OF_RANGE;
151     }
152     return u8RetErrorState;
153 }
```

Here is the caller graph for this function:

LBTY_tenuErrorStatus EEPROM_u8ReadString (u16 *u16StartAddress*, u16 *u16EndAddress*, u8 * *pu8String*)

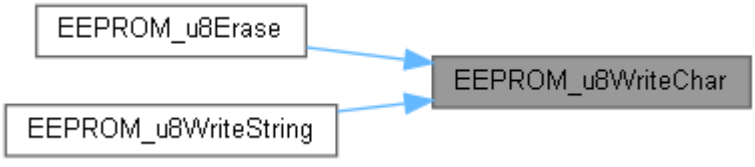
```
161 
{
162     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
163     for(int i = u16StartAddress ; i<= u16EndAddress ; i++){
164         if(EEPROM_u8ReadChar(i, pu8String++)){
165             u8RetErrorState = LBTY_READ_ERROR;
166             break;
167         }
168     }
169     return u8RetErrorState;
170 }
```

Here is the call graph for this function:

LBTY_tenuErrorStatus EEPROM_u8WriteChar (u16 *u16Address*, u8 *u8Data*)

```
96                                                                      {
97     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
98     if(u16Address <= EEPROM_MAX_ADDRESS){
99         while(S_EEPROM->m_EECR.sBits.m_EEWE);
100         while(S_SPMCR->sBits.m_SPMEN);
101 
102         S_EEPROM->m_EEAR = u16Address;
103         S_EEPROM->m_EEDR = u8Data;
104 
105         INTP_vidDisable();
106 
107         S_EEPROM->m_EECR.sBits.m_EEMWE = LBTY_SET;
108         S_EEPROM->m_EECR.sBits.m_EEWE  = LBTY_SET;
109 
110         INTP_vidEnable();
111 
112     }else{
113         u8RetErrorState = LBTY_INDEX_OUT_OF_RANGE;
114     }
115     return u8RetErrorState;
116 }
```

Here is the caller graph for this function:



LBTY_tenuErrorStatus EEPROM_u8WriteString (u16 *u16Address*, u8 * *pu8String*)

```
124                                                                     {
125     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
126     while(*pu8String){
127         if(EEPROM_u8WriteChar(u16Address++, *(pu8String++))){
128             u8RetErrorState = LBTY_WRITE_ERROR;
129             break;
```

```
130         }
131     }
132     return u8RetErrorState;
133 }
```

Here is the call graph for this function:

void EEPROM_vidDisableInterrupt (void)

```
71                                      {
72     S_EEPROM->m_EECR.sBits.m_EERIE = LBTY_RESET;
73 }
```

void EEPROM_vidEnableInterrupt (void)

```
62                                      {
63     S_EEPROM->m_EECR.sBits.m_EERIE = LBTY_SET;
64 }
```

void EEPROM_vidInit (void)

```
47                          {
48     S_EEPROM->m_EEAR = LBTY_u16ZERO;
49     S_EEPROM->m_EEDR = LBTY_u8ZERO;
50
51     S_EEPROM->m_EECR.sBits.m_EERIE = EEPROM_INTERRUPT_INIT_STATE;
52     S_EEPROM->m_EECR.sBits.m_EEMWE = LBTY_RESET;
53     S_EEPROM->m_EECR.sBits.m_EEWE  = LBTY_RESET;
54     S_EEPROM->m_EECR.sBits.m_EERE  = LBTY_RESET;
55 }
```

void EEPROM_vidSetCallBack (void(*)(void) *CallBack*)

```
177                                                 {
178     pfunctionCallBack = CallBack;
179 }
```

ISR (EE_RDY_vect)

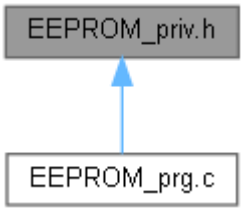
```
181                 {
182     pfunctionCallBack();
183 }
```

Variable Documentation

void(* pfunctionCallBack) (void) (void)

EEPROM_priv.h File Reference

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



Data Structures

union EECR_type: *Type define of Union bit field of "EEPROM Control Register"*

struct EEPROM_type: *EEPROM Registers*

union SPMCR_type: *Type define of Union bit field of "MCU Control Register"*

Macros

- #define S_EEPROM ((EEPROM_type* const)0x3CU)
- #define EECR (*(volatile u8* const)0x3CU)
- #define EEDR (*(volatile u8* const)0x3DU)
- #define EEARL (*(volatile u8* const)0x3EU)
- #define EEARH (*(volatile u8* const)0x3FU)
- #define S_SPMCR ((SPMCR_type* const)0x57U)
- #define SPMCR (*(volatile u8* const)0x57U)
- #define EEPROM_MAX_ADDRESS 1023u

Macro Definition Documentation

#define EEARH (*(volatile u8* const)0x3FU)

#define EEARL (*(volatile u8* const)0x3EU)

#define EECR (*(volatile u8* const)0x3CU)

#define EEDR (*(volatile u8* const)0x3DU)

#define EEPROM_MAX_ADDRESS 1023u

#define S_EEPROM ((EEPROM_type* const)0x3CU)

EEPROM

#define S_SPMCR ((SPMCR_type* const)0x57U)

Store Program Memory Control Register

```
#define SPMCR   (*(volatile u8* const)0x57U)
```

EEPROM_priv.h

```
Go to the documentation of this file.1 /*
****************************************************************************** */
2 /* ********************** FILE DEFINITION SECTION ************************** */
3 /* ****************************************************************************** */
4 /* File Name    : EEPROM_priv.h                                                  */
5 /* Author       : MAAM                                                           */
6 /* Version      : v01                                                            */
7 /* date         : Apr 30, 2023                                                   */
8 /* ****************************************************************************** */
9 /* ************************* HEADER FILES INCLUDES ***************************  */
10 /* ****************************************************************************** */
11
12 #ifndef EEPROM_PRIV_H_
13 #define EEPROM_PRIV_H_
14
15 /* ****************************************************************************** */
16 /* ********************** TYPE_DEF/STRUCT/ENUM SECTION ********************** */
17 /* ****************************************************************************** */
18
21 typedef union{
22     u8 u_Reg;
23     struct {
24         __IO u8 m_EERE : 1;
25         __IO u8 m_EEWE : 1;
26         __IO u8 m_EEMWE: 1;
27         __IO u8 m_EERIE: 1;
28         __IO u8        : 4;
29     }sBits;
30 }EECR_type; //
31
32 /*****************************************************************************/
33
36 typedef struct{
37     __IO EECR_type  m_EECR ;
38     __IO u8         m_EEDR ;
39     __IO u16        m_EEAR : 10;
40     __IO u16               : 6;
41 }EEPROM_type;
42
43 /*****************************************************************************/
44
47 typedef union{
48     u8 u_Reg;
49     struct {
50         __IO u8 m_SPMEN : 1;
51         __I  u8         : 7;
52     }sBits;
53 }SPMCR_type;
54
55 /* ****************************************************************************** */
56 /* ************************* MACRO/DEFINE SECTION ************************** */
57 /* ****************************************************************************** */
58
60 #define S_EEPROM        ((EEPROM_type* const)0x3CU)
61 #define EECR            (*(volatile u8* const)0x3CU)
62 #define EEDR            (*(volatile u8* const)0x3DU)
63 #define EEARL           (*(volatile u8* const)0x3EU)
64 #define EEARH           (*(volatile u8* const)0x3FU)
65
67 #define S_SPMCR         ((SPMCR_type* const)0x57U)
68 #define SPMCR           (*(volatile u8* const)0x57U)
69
70 /* ****************************************************************************** */
71
72 #define EEPROM_MAX_ADDRESS                  1023u
73
74 /* ****************************************************************************** */
75 /* ***************************** CONST SECTION ********************************** */
76 /* ****************************************************************************** */
77
78 /* ****************************************************************************** */
79 /* ***************************** VARIABLE SECTION ********************************* */
80 /* ****************************************************************************** */
```

```
81
82 /* ************************************************************************** */
83 /* ************************** FUNCTION SECTION ********************************** */
84 /* ************************************************************************** */
85
86
87 #endif /* EEPROM_PRIV_H_ */
88 /*************************** E N D (EEPROM_priv.h) ****************************/
```

main.c File Reference