SWC_SPI

Version v1.0
7/27/2023 3:30:00 PM

Table of Contents

Data Structure Index

Data Structures

Here are the data structures with brief descriptions:

File Index

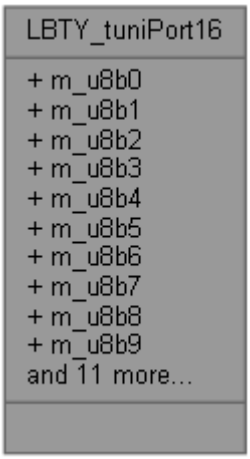
File List

Here is a list of all files with brief descriptions:

Data Structure Documentation

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

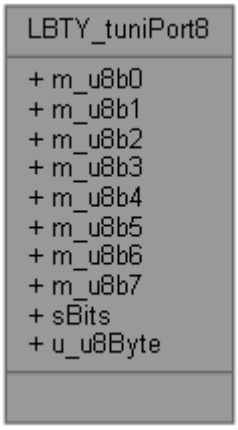
u16 u_u16Word

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

SPCR_type Union Reference

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

```
#include <SPI_priv.h>
```

Collaboration diagram for SPCR_type:

Data Fields

- u8 u_Reg
- struct {
- __IO u8 m_SPR: 2
- __IO u8 m_CPHA: 1
- __IO u8 m_CPOL: 1
- __IO u8 m_MSTR: 1
- __IO u8 m_DORD: 1
- __IO u8 m_SPE: 1
- __IO u8 m_SPIE: 1
- } sBits

Detailed Description

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

Type : Union **Unit** : None

Field Documentation

__IO u8 m_CPHA

Clock Phase

__IO u8 m_CPOL

Clock Polarity

__IO u8 **m_DORD**

Data Order

__IO u8 **m_MSTR**

Master/Slave Select

__IO u8 **m_SPE**

SPI Enable

__IO u8 **m_SPIE**

SPI Interrupt Enable

__IO u8 **m_SPR**

SPI Clock Rate Select 1 and 0

struct { ... } sBits

u8 **u_Reg**

Byte

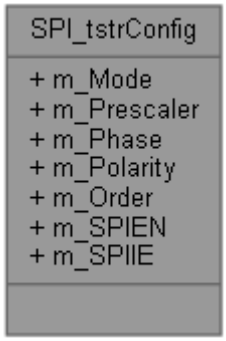
The documentation for this union was generated from the following file:

SPI_priv.h

SPI_tstrConfig Struct Reference

: type define of structure for SPI Configuration

`#include <SPI_int.h>`
Collaboration diagram for SPI_tstrConfig:



Data Fields

- SPI_tenuMode m_Mode
- SPI_tenuClockRate m_Prescaler
- SPI_tenuClockPhase m_Phase
- SPI_tenuClockPolarity m_Polarity
- SPI_tenuDataOrder m_Order
- LBTY_tenuFlagStatus m_SPIEN
- LBTY_tenuFlagStatus m_SPIIE

Detailed Description

: type define of structure for SPI Configuration

Type : struct **Unit** : None

Field Documentation

SPI_tenuMode m_Mode

SPI Operation Mode

SPI_tenuDataOrder m_Order

SPI Data Order

SPI_tenuClockPhase m_Phase

SPI Clock Phase

SPI_tenuClockPolarity m_Polarity

SPI Clock Polarity

SPI_tenuClockRate m_Prescaler

SPI Clock Prescaler

LBTY_tenuFlagStatus m_SPIEN

SPI Enable

LBTY_tenuFlagStatus m_SPIIE

SPI Interrupt Enable

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

SPI_int.h

SPI_tstrSS_Config Struct Reference

: type define of structure for SS Pin Configuration

`#include <SPI_int.h>`
Collaboration diagram for SPI_tstrSS_Config:

Data Fields

- GPIO_tenuPortNum m_Port
- GPIO_tenuPinNum m_Pin

Detailed Description

: type define of structure for SS Pin Configuration

Type : struct **Unit** : None

Field Documentation

GPIO_tenuPinNum m_Pin

Pin Number

GPIO_tenuPortNum m_Port

Port Number

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

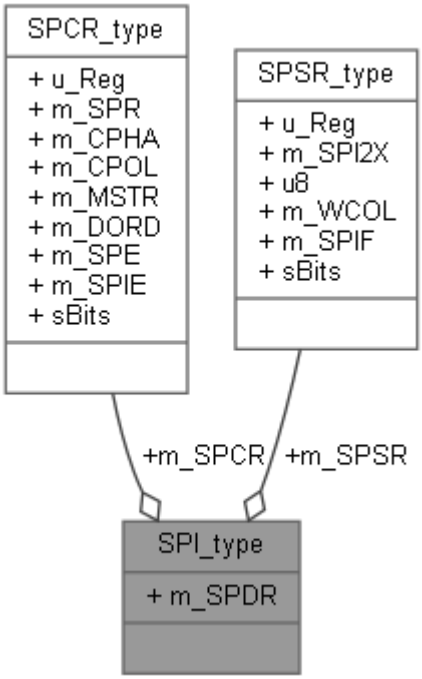
SPI_int.h

SPI_type Struct Reference

: SPI Registers

```
#include <SPI_priv.h>
```

Collaboration diagram for SPI_type:



Data Fields

- __IO SPCR_type m_SPCR
- __IO SPSR_type m_SPSR
- __IO u8 m_SPDR

Detailed Description

: SPI Registers

Type : Struct **Unit** : None

Field Documentation

__IO SPCR_type m_SPCR

SPI Control Reg

__IO u8 m_SPDR

SPI Data Reg

__IO SPSR_type m_SPSR

SPI Status Reg

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

SPI_priv.h

SPSR_type Union Reference

: Type define of Union bit field of "SPI Status Register"

```
#include <SPI_priv.h>
```

Collaboration diagram for SPSR_type:

Data Fields

- u8 u_Reg
- struct {
- __IO u8 m_SPI2X: 1
- __I u8: 5
- __I u8 m_WCOL: 1
- __I u8 m_SPIF: 1
- } sBits

Detailed Description

: Type define of Union bit field of "SPI Status Register"

Type : Union **Unit** : None

Field Documentation

__IO u8 m_SPI2X

Double SPI Speed Bit

__I u8 m_SPIF

SPI Interrupt Flag

__I u8 m_WCOL

Write Collision Flag

struct { ... } sBits

__I u8

Reversed

u8 u_Reg

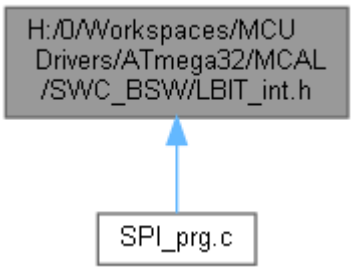
Byte

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

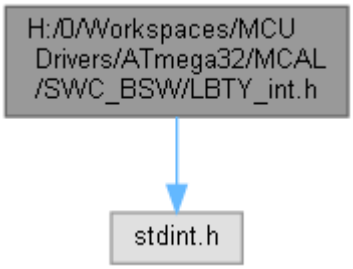
```
1 /*
**************************************************************************** */
2 /* ********************** FILE DEFINITION SECTION ************************** */
3 /* ************************************************************************** */
4 /* File Name    : LBIT_int.h                                                  */
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) ******************************/
```

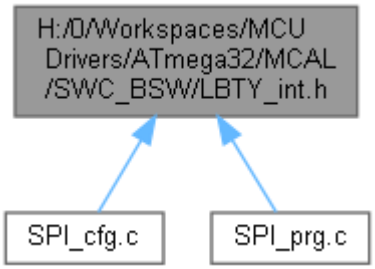
H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_BSW/LBTY_int.h File Reference

```
#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.

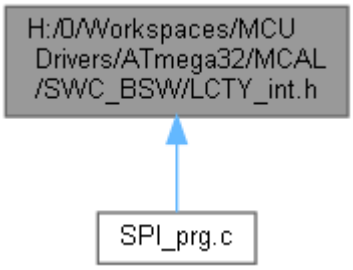
```
1 /*
2 /* ********************** FILE DEFINITION SECTION ************************** */
3 /* ************************************************************************* */
4 /* File Name   : LBTY_int.h                                                  */
5 /* Author      : MAAM                                                        */
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.

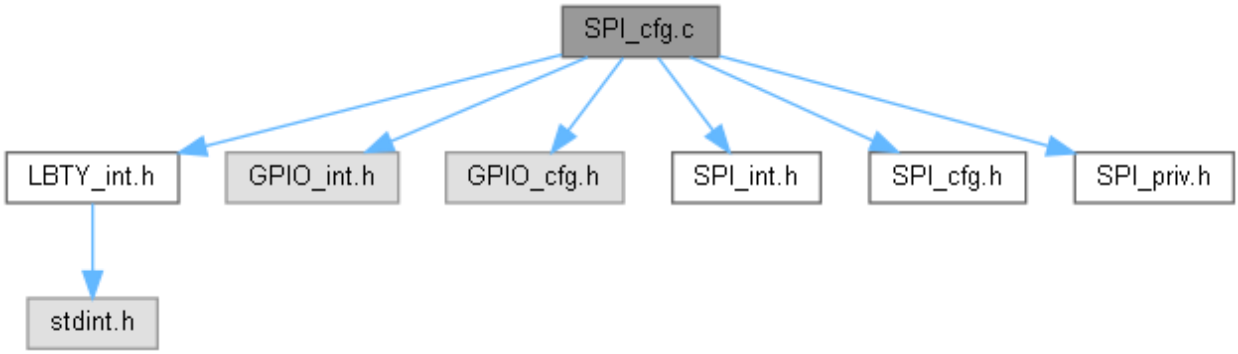
```
1 /*
******************************************************************************* */
2 /* ********************** FILE DEFINITION SECTION ************************** */
3 /* ******************************************************************************* */
4 /* File Name   : LCTY_int.h                                                    */
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) ******************************/
```

main.c File Reference

SPI_cfg.c File Reference

```
#include "LBTY_int.h"
#include "GPIO_int.h"
#include "GPIO_cfg.h"
#include "SPI_int.h"
#include "SPI_cfg.h"
#include "SPI_priv.h"
```

Include dependency graph for SPI_cfg.c:



Variables

- const SPI_tstrSS_Config kastrSSConfiguration_GLB [SPI_SS_NUM]

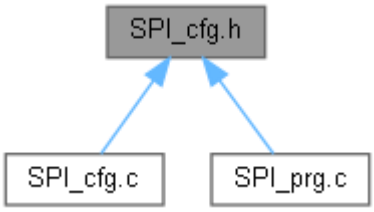
Variable Documentation

const SPI_tstrSS_Config kastrSSConfiguration_GLB[SPI_SS_NUM]

```
Initial value:= {
     {.m_Port = SPI_PORT, .m_Pin = SPI_SS_PIN}









}
```

SPI_cfg.h File Reference

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



Macros

- #define SPI_MODE SPI_Master
- #define SPI_SS_NUM 1u
- #define SPI_CLOCK_PRESCALER SPI_Prescaler_8
- #define SPI_CLOCK_POLARITY SPI_Leading_Rising
- #define SPI_CLOCK_PHASE SPI_Leading_Setup
- #define SPI_DATA_ORDER SPI_LSB_Frist
- #define SPI_INIT LBTY_SET
- #define SPI_INT LBTY_SET

Variables

- const SPI_tstrSS_Config kastrSSConfiguration_GLB [SPI_SS_NUM]

Macro Definition Documentation

#define SPI_CLOCK_PHASE SPI_Leading_Setup

#define SPI_CLOCK_POLARITY SPI_Leading_Rising

#define SPI_CLOCK_PRESCALER SPI_Prescaler_8

#define SPI_DATA_ORDER SPI_LSB_Frist

#define SPI_INIT LBTY_SET

#define SPI_INT LBTY_SET

#define SPI_MODE SPI_Master

#define SPI_SS_NUM 1u

Variable Documentation

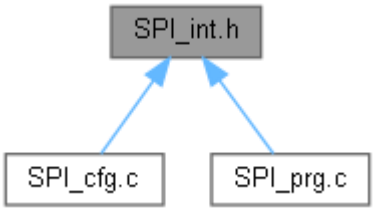
const SPI_tstrSS_Config kastrSSConfiguration_GLB[SPI_SS_NUM] `[extern]`

SPI_cfg.h

```
Go to the documentation of this file.1 /*
****************************************************************************** */
2 /* ********************** FILE DEFINITION SECTION ************************** */
3 /* ****************************************************************************** */
4 /* File Name    : SPI_cfg.h                                                     */
5 /* Author       : MAAM                                                          */
6 /* Version      : v01.2                                                         */
7 /* date         : Apr 12, 2023                                                  */
8 /* ****************************************************************************** */
9 /* ************************ HEADER FILES INCLUDES ***************************  */
10 /* ****************************************************************************** */
11
12 #ifndef SPI_CFG_H_
13 #define SPI_CFG_H_
14
15 /* ****************************************************************************** */
16 /* ********************** TYPE_DEF/STRUCT/ENUM SECTION ********************** */
17 /* ****************************************************************************** */
18
19 /* ****************************************************************************** */
20 /* ************************** MACRO/DEFINE SECTION ************************** */
21 /* ****************************************************************************** */
22
23 #define SPI_MODE                SPI_Master
24 #define SPI_SS_NUM              1u
25
26 #define SPI_CLOCK_PRESCALER     SPI_Prescaler_8
27 #define SPI_CLOCK_POLARITY      SPI_Leading_Rising
28 #define SPI_CLOCK_PHASE         SPI_Leading_Setup
29 #define SPI_DATA_ORDER          SPI_LSB_Frist
30
31 #define SPI_INIT                LBTY_SET
32 #define SPI_INT                 LBTY_SET
33
34 /* ****************************************************************************** */
35 /* ***************************** CONST SECTION ****************************** */
36 /* ****************************************************************************** */
37
38 extern const SPI_tstrSS_Config kastrSSConfiguration_GLB[SPI_SS_NUM];
39
40 /* ****************************************************************************** */
41 /* ***************************** VARIABLE SECTION *************************** */
42 /* ****************************************************************************** */
43
44 /* ****************************************************************************** */
45 /* ***************************** FUNCTION SECTION *************************** */
46 /* ****************************************************************************** */
47
48
49 #endif /* SPI CFG H  */
50 /*************************** E N D (SPI_cfg.h) ****************************/
```

SPI_int.h File Reference

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



Data Structures

struct SPI_tstrConfig*: type define of structure for SPI Configuration*

struct SPI_tstrSS_Config*: type define of structure for SS Pin Configuration*

Enumerations

- enum SPI_tenuMode { SPI_Slave = (u8)0u, SPI_Master }
- enum SPI_tenuClockRate { SPI_Prescaler_4 = (u8)0u, SPI_Prescaler_16, SPI_Prescaler_64, SPI_Prescaler_128, SPI_Prescaler_2, SPI_Prescaler_8, SPI_Prescaler_32 }
- enum SPI_tenuClockPolarity { SPI_Leading_Rising = (u8)0u, SPI_Leading_Falling, SPI_Tralling_Falling = (u8)0u, SPI_Tralling_Rising }
- enum SPI_tenuClockPhase { SPI_Leading_Sample = (u8)0u, SPI_Leading_Setup, SPI_Tralling_Setup = (u8)0u, SPI_Tralling_Sample }
- enum SPI_tenuDataOrder { SPI_MSB_Frist = (u8)0u, SPI_LSB_Frist }

Functions

- void SPI_vidSetConfig (SPI_tstrConfig const *const pstrConfig)
- void SPI_vidSRestConfig (SPI_tstrConfig *const pstrConfig)
- void SPI_vidInit (void)
- void SPI_vidEnable (void)
- void SPI_vidDisable (void)
- void SPI_vidEnableINT (void)
- void SPI_vidDisableINT (void)
- LBTY_tenuErrorStatus SPI_vidSetPrescaler (SPI_tenuClockRate u8Prescaler)
- LBTY_tenuErrorStatus SPI_vidSetClockPhase (SPI_tenuClockPhase u8Phase)
- LBTY_tenuErrorStatus SPI_vidSetClockPolarity (SPI_tenuClockPolarity u8Polarity)
- LBTY_tenuErrorStatus SPI_vidSetDataOrder (SPI_tenuDataOrder u8Order)
- LBTY_tenuErrorStatus SPI_u8SetTransmit (u8 u8Char)
- LBTY_tenuErrorStatus SPI_u8GetTransmit (u8 *pu8Char)
- LBTY_tenuErrorStatus SPI_u8SetChar (u8 u8Char, u8 u8Index)
- LBTY_tenuErrorStatus SPI_u8GetChar (u8 *pu8Char, u8 u8Index)
- void SPI_vidSetStr (u8 *pu8Transmit, u8 u8Index)
- void SPI_vidGetStr (u8 *pu8Receive, u8 u8Index)
- void SPI_vidSetCallBack_OverFlow (void(*pCallBack)(void))

Enumeration Type Documentation

enum SPI_tenuClockPhase

Enumerator:

SPI_Leading_Sample	
SPI_Leading_Setup	
SPI_Tralling_Setup	
SPI_Tralling_Sample	

```
41              {
42      SPI_Leading_Sample = (u8)0u,
43      SPI_Leading_Setup,
44      SPI_Tralling_Setup = (u8)0u,
45      SPI_Tralling_Sample
46  }SPI_tenuClockPhase;
```

enum SPI_tenuClockPolarity

Enumerator:

SPI_Leading_Rising	
SPI_Leading_Falling	
SPI_Tralling_Falling	
SPI_Tralling_Rising	

```
34              {
35      SPI_Leading_Rising = (u8)0u,
36      SPI_Leading_Falling,
37      SPI_Tralling_Falling = (u8)0u,
38      SPI_Tralling_Rising
39  }SPI_tenuClockPolarity;
```

enum SPI_tenuClockRate

Enumerator:

SPI_Prescaler_4	
SPI_Prescaler_16	
SPI_Prescaler_64	
SPI_Prescaler_128	
SPI_Prescaler_2	
SPI_Prescaler_8	
SPI_Prescaler_32	

```
24              {
25      SPI_Prescaler_4 = (u8)0u,
26      SPI_Prescaler_16,
27      SPI_Prescaler_64,
28      SPI_Prescaler_128,
29      SPI_Prescaler_2,
30      SPI_Prescaler_8,
31      SPI_Prescaler_32
32  }SPI_tenuClockRate;
```

enum SPI_tenuDataOrder

Enumerator:

SPI_MSB_Frist	
SPI_LSB_Frist	

```
48              {
```

```
49      SPI_MSB_Frist = (u8)0u,
50      SPI_LSB_Frist
51  }SPI_tenuDataOrder;
```

enum SPI_tenuMode

Enumerator:

SPI_Slave	
SPI_Master	

```
19              {
20      SPI_Slave = (u8)0u,
21      SPI_Master,
22  }SPI_tenuMode;
```

Function Documentation

LBTY_tenuErrorStatus SPI_u8GetChar (u8 * *pu8Char*, u8 *u8Index*)

```
245                                                                 {
246     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
247
248     if(strSPI_Config_GLB.m_Mode == SPI_Master){
249
250         GPIO_u8SetPinValue(kastrSSConfiguration_GLB[u8Index].m_Port,
kastrSSConfiguration_GLB[u8Index].m_Pin,  SPI_SS_ENABLE);
251         u8RetErrorState = SPI_u8GetTransmit(pu8Char);
252         GPIO_u8SetPinValue(kastrSSConfiguration_GLB[u8Index].m_Port,
kastrSSConfiguration_GLB[u8Index].m_Pin, SPI_SS_DISABLE);
253
254     }else if(strSPI_Config_GLB.m_Mode == SPI_Slave){
255
256         u8 u8State = SPI_SS_DISABLE;
257         GPIO_u8GetPinValue(SPI_PORT, SPI_SS_PIN, &u8State);
258
259         if(u8State == SPI_SS_ENABLE){
260             u8RetErrorState = SPI_u8GetTransmit(pu8Char);
261         }
262
263     }else{
264         while(1);
265     }
266
267     return u8RetErrorState;
268 }
```

Here is the call graph for this function:

Here is the caller graph for this function:

LBTY_tenuErrorStatus SPI_u8GetTransmit (u8 * *pu8Char*)

```
207                                                         {
208     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
209
210     if(!(S_SPI->m_SPSR.sBits.m_WCOL)){
211         while((!S_SPI->m_SPSR.sBits.m_SPIF) && (!SPI_u8Flag_GLB));
212         SPI_u8Flag_GLB = LBTY_RESET;
213         *pu8Char = S_SPI->m_SPDR;
214     }else{
215         u8RetErrorState = LBTY_NOK;
216     }
217
218     return u8RetErrorState;
219 }
```

Here is the caller graph for this function:

LBTY_tenuErrorStatus SPI_u8SetChar (u8 *u8Char*, u8 *u8Index*)

```
221                                                                 {
222     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
223
224     if(strSPI_Config_GLB.m_Mode == SPI_Master){
225
226         GPIO_u8SetPinValue(kastrSSConfiguration_GLB[u8Index].m_Port,
kastrSSConfiguration_GLB[u8Index].m_Pin,  SPI_SS_ENABLE);
227         u8RetErrorState = SPI_u8SetTransmit(u8Char);
228         GPIO_u8SetPinValue(kastrSSConfiguration_GLB[u8Index].m_Port,
kastrSSConfiguration_GLB[u8Index].m_Pin, SPI_SS_DISABLE);
229
230     }else if(strSPI_Config_GLB.m_Mode == SPI_Slave){
231
232         u8 u8State = SPI_SS_DISABLE;
233         GPIO_u8GetPinValue(SPI_PORT, SPI_SS_PIN, &u8State);
234
235         if(u8State == SPI_SS_ENABLE){
236             u8RetErrorState = SPI_u8SetTransmit(u8Char);
237         }
238
239     }else{
240         while(1);
241     }
242
243     return u8RetErrorState;
244 }
```

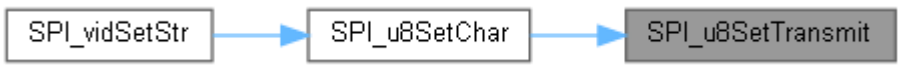
Here is the call graph for this function:

Here is the caller graph for this function:

LBTY_tenuErrorStatus SPI_u8SetTransmit (u8 *u8Char*)

```
194                                                         {
195     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
196
197     if(!(S_SPI->m_SPSR.sBits.m_WCOL)){
198         S_SPI->m_SPDR = u8Char;
199         while((!S_SPI->m_SPSR.sBits.m_SPIF) && (!SPI_u8Flag_GLB));
200         SPI_u8Flag_GLB = LBTY_RESET;
201     }else{
202         u8RetErrorState = LBTY_NOK;
203     }
204
205     return u8RetErrorState;
206 }
```

Here is the caller graph for this function:



void SPI_vidDisable (void)

```
114                             {
115     S_SPI->m_SPCR.sBits.m_SPE  = strSPI_Config_GLB.m_SPIEN = LBTY_RESET;
116 }
```

void SPI_vidDisableINT (void)

```
121                               {
122     S_SPI->m_SPCR.sBits.m_SPIE  = strSPI_Config_GLB.m_SPIIE = LBTY_RESET;
123 }
```

void SPI_vidEnable (void)

```
111                             {
```

```
112     S_SPI->m_SPCR.sBits.m_SPE  = strSPI_Config_GLB.m_SPIEN = LBTY_SET;
113 }
```

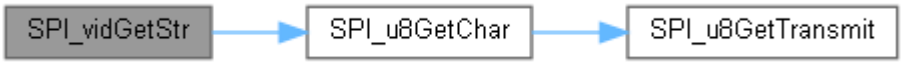
void SPI_vidEnableINT (void)

```
118                             {
119     S_SPI->m_SPCR.sBits.m_SPIE  = strSPI_Config_GLB.m_SPIIE = LBTY_SET;
120 }
```

void SPI_vidGetStr (u8 * *pu8Receive*, u8 *u8Index*)

```
277                                                 {
278     do{
279         if(SPI_u8GetChar(pu8Receive, u8Index)){
280             break;
281         }
282     }while(*(pu8Receive++) != '\0');
283 }
```

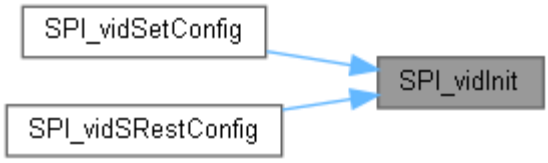
Here is the call graph for this function:



void SPI_vidInit (void)

```
82                          {
83
84     S_SPI->m_SPCR.sBits.m_MSTR = strSPI_Config_GLB.m_Mode;
85     if(strSPI_Config_GLB.m_Mode == SPI_Master){
86         GPIO_u8SetMaskDirection(SPI_PORT, SPI_MODE_MASK, PORT_OUTPUT);
87         GPIO_u8SetPinDirection (SPI_PORT, SPI_MISO_PIN , PIN_INPUT  );
88         for(u8 i = SPI_SS_NUM ; i-- ; ){
89             GPIO_u8SetPinDirection(kastrSSConfiguration_GLB[i].m_Port,
kastrSSConfiguration_GLB[i].m_Pin, PIN_OUTPUT);
90             GPIO_u8SetPinValue    (kastrSSConfiguration_GLB[i].m_Port,
kastrSSConfiguration_GLB[i].m_Pin, SPI_SS_DISABLE);
91         }
92     }else if(strSPI_Config_GLB.m_Mode == SPI_Slave){
93         GPIO_u8SetMaskDirection(SPI_PORT, SPI_MODE_MASK, PORT_INPUT);
94         GPIO_u8SetPinDirection (SPI_PORT, SPI_MISO_PIN , PIN_OUTPUT);
95     }
96
97     //SPI_vidSetPrescaler(strSPI_Config_GLB.m_Prescaler);
98     S_SPI->m_SPSR.sBits.m_SPI2X= GET_BIT(strSPI_Config_GLB.m_Prescaler,
SPI_SPI2X_BIT);
99     S_SPI->m_SPCR.sBits.m_SPR  = GET_MASK(strSPI_Config_GLB.m_Prescaler,
SPI_SPR_MASK);
100     //SPI_vidSetClockPhase(strSPI_Config_GLB.m_Phase);
101     S_SPI->m_SPCR.sBits.m_CPHA = strSPI_Config_GLB.m_Phase;
102     //SPI_vidSetClockPolarity(strSPI_Config_GLB.m_Polarity);
103     S_SPI->m_SPCR.sBits.m_CPOL = strSPI_Config_GLB.m_Polarity;
104     //SPI_vidSetDataOrder(strSPI_Config_GLB.m_Order);
105     S_SPI->m_SPCR.sBits.m_DORD = strSPI_Config_GLB.m_Order;
106
107     S_SPI->m_SPCR.sBits.m_SPIE = strSPI_Config_GLB.m_SPIIE;
108     S_SPI->m_SPCR.sBits.m_SPE  = strSPI_Config_GLB.m_SPIEN;
109 }
```

Here is the caller graph for this function:



void SPI_vidSetCallBack_OverFlow (void(*)(void) *pCallBack*)

```
287                                                 {
288     pFuncCallBack_SPI = pCallBack;
289 }
```

LBTY_tenuErrorStatus SPI_vidSetClockPhase (SPI_tenuClockPhase *u8Phase*)

```
147                                                                 {
148     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
```

```
149
150     switch(u8Phase){
151         case SPI_Leading_Sample:
152         case SPI_Leading_Setup:
153             S_SPI->m_SPCR.sBits.m_CPHA = strSPI_Config_GLB.m_Phase = u8Phase;
154             break;
155         default:
156             u8RetErrorState = LBTY_NOK;
157     }
158
159     return u8RetErrorState;
160 }
```

LBTY_tenuErrorStatus SPI_vidSetClockPolarity (SPI_tenuClockPolarity *u8Polarity*)

```
162
{
163     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
164
165     switch(u8Polarity){
166         case SPI_Leading_Rising:
167         case SPI_Leading_Falling:
168             S_SPI->m_SPCR.sBits.m_CPOL = strSPI_Config_GLB.m_Polarity =
u8Polarity;
169             break;
170         default:
171             u8RetErrorState = LBTY_NOK;
172     }
173
174     return u8RetErrorState;
175 }
```

void SPI_vidSetConfig (SPI_tstrConfig const *const *pstrConfig*)

```
60                                                                  {
61     if(pstrConfig != LBTY_NULL){
62         strSPI_Config_GLB = *pstrConfig;
63     }
64     SPI_vidInit();
65 }
```

Here is the call graph for this function:

LBTY_tenuErrorStatus SPI_vidSetDataOrder (SPI_tenuDataOrder *u8Order*)

```
177                                                                 {
178     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
179
180     switch(u8Order){
181         case SPI_MSB_Frist:
182         case SPI_LSB_Frist:
183             S_SPI->m_SPCR.sBits.m_DORD = strSPI_Config_GLB.m_Order = u8Order;
184             break;
185         default:
186             u8RetErrorState = LBTY_NOK;
187     }
188
189     return u8RetErrorState;
190 }
```

LBTY_tenuErrorStatus SPI_vidSetPrescaler (SPI_tenuClockRate *u8Prescaler*)

```
125                                                                 {
126     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
127
128     switch(u8Prescaler){
129         case SPI_Prescaler_4:
130         case SPI_Prescaler_16:
131         case SPI_Prescaler_64:
132         case SPI_Prescaler_128:
133         case SPI_Prescaler_2:
134         case SPI_Prescaler_8:
135         case SPI_Prescaler_32:
136             S_SPI->m_SPSR.sBits.m_SPI2X= GET_BIT(u8Prescaler, SPI_SPI2X_BIT);
137             S_SPI->m_SPCR.sBits.m_SPR  = GET_MASK(u8Prescaler, SPI_SPR_MASK);
```

```
138             strSPI_Config_GLB.m_Prescaler = u8Prescaler;
139             break;
140         default:
141             u8RetErrorState = LBTY_NOK;
142     }
143
144     return u8RetErrorState;
145 }
```

void SPI_vidSetStr (u8 * *pu8Transmit*, u8 *u8Index*)

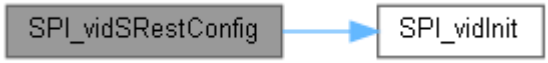
```
270                                                 {
271     while(*pu8Transmit){
272         if(SPI_u8SetChar(*(pu8Transmit++), u8Index)){
273             break;
274         }
275     }
276 }
```

Here is the call graph for this function:

void SPI_vidSRestConfig (SPI_tstrConfig *const *pstrConfig*)

```
67                                                          {
68     strSPI_Config_GLB.m_Mode         = SPI_MODE;
69     strSPI_Config_GLB.m_Prescaler    = SPI_CLOCK_PRESCALER;
70     strSPI_Config_GLB.m_Phase        = SPI_CLOCK_PHASE;
71     strSPI_Config_GLB.m_Polarity     = SPI_CLOCK_POLARITY;
72     strSPI_Config_GLB.m_Order        = SPI_DATA_ORDER;
73     strSPI_Config_GLB.m_SPIEN        = SPI_INIT;
74     strSPI_Config_GLB.m_SPIIE        = SPI_INT;
75
76     if(pstrConfig != LBTY_NULL){
77         *pstrConfig = strSPI_Config_GLB;
78     }
79     SPI_vidInit();
80 }
```

Here is the call graph for this function:



SPI_int.h

Go to the documentation of this file.

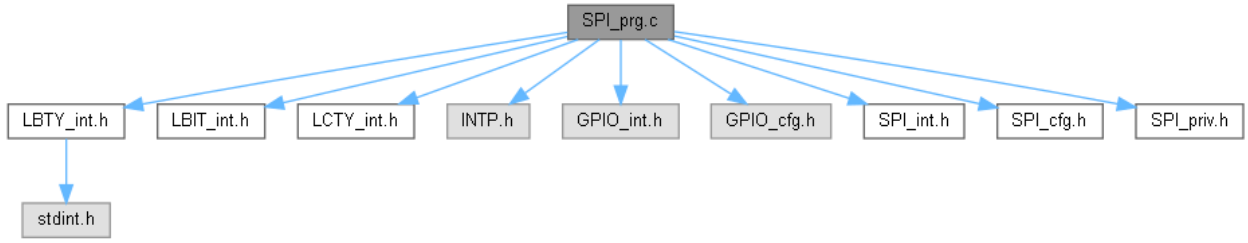
```c
/*
/* ********************************************************************************** */
/* ********************** FILE DEFINITION SECTION ************************************ */
/* ********************************************************************************** */
/* File Name    : SPI_int.h                                                           */
/* Author       : MAAM                                                                */
/* Version      : v01.2                                                               */
/* date         : Apr 12, 2023                                                        */
/* ********************************************************************************** */
/* ************************ HEADER FILES INCLUDES ***************************  */
/* ********************************************************************************** */

#ifndef SPI_INT_H_
#define SPI_INT_H_

/* ********************************************************************************** */
/* ********************** TYPE_DEF/STRUCT/ENUM SECTION ********************************** */
/* ********************************************************************************** */

typedef enum{
    SPI_Slave = (u8)0u,
    SPI_Master,
}SPI_tenuMode;

typedef enum{
    SPI_Prescaler_4 = (u8)0u,
    SPI_Prescaler_16,
    SPI_Prescaler_64,
    SPI_Prescaler_128,
    SPI_Prescaler_2,
    SPI_Prescaler_8,
    SPI_Prescaler_32
}SPI_tenuClockRate;

typedef enum{
    SPI_Leading_Rising = (u8)0u,
    SPI_Leading_Falling,
    SPI_Tralling_Falling = (u8)0u,
    SPI_Tralling_Rising
}SPI_tenuClockPolarity;

typedef enum{
    SPI_Leading_Sample = (u8)0u,
    SPI_Leading_Setup,
    SPI_Tralling_Setup = (u8)0u,
    SPI_Tralling_Sample
}SPI_tenuClockPhase;

typedef enum{
    SPI_MSB_Frist = (u8)0u,
    SPI_LSB_Frist
}SPI_tenuDataOrder;


/*****************************************************************************************
*******************************/

typedef struct{
    SPI_tenuMode            m_Mode;
    SPI_tenuClockRate       m_Prescaler;
    SPI_tenuClockPhase      m_Phase;
    SPI_tenuClockPolarity   m_Polarity;
    SPI_tenuDataOrder       m_Order;
    LBTY_tenuFlagStatus     m_SPIEN;
    LBTY_tenuFlagStatus     m_SPIIE;
}SPI_tstrConfig;

typedef struct{
    GPIO_tenuPortNum        m_Port;
    GPIO_tenuPinNum         m_Pin;
}SPI_tstrSS_Config;

/* ********************************************************************************** */
```

```
76 /* ************************* MACRO/DEFINE SECTION ************************* */
77 /* ********************************************************************** */
78
79 /* ********************************************************************** */
80 /* ***************************** CONST SECTION **************************** */
81 /* ********************************************************************** */
82
83 /* ********************************************************************** */
84 /* ***************************** VARIABLE SECTION ************************* */
85 /* ********************************************************************** */
86
87 /* ********************************************************************** */
88 /* ***************************** FUNCTION SECTION ************************* */
89 /* ********************************************************************** */
90
91 extern void SPI_vidSetConfig(SPI_tstrConfig const* const pstrConfig);
92 extern void SPI_vidSRestConfig(SPI_tstrConfig* const pstrConfig);
93
94 extern void SPI_vidInit(void);
95
96 extern void SPI_vidEnable(void);
97 extern void SPI_vidDisable(void);
98
99 extern void SPI_vidEnableINT(void);
100 extern void SPI_vidDisableINT(void);
101
102 extern LBTY_tenuErrorStatus SPI_vidSetPrescaler(SPI_tenuClockRate u8Prescaler);
103 extern LBTY_tenuErrorStatus SPI_vidSetClockPhase(SPI_tenuClockPhase u8Phase);
104 extern LBTY_tenuErrorStatus SPI_vidSetClockPolarity(SPI_tenuClockPolarity
u8Polarity);
105 extern LBTY_tenuErrorStatus SPI_vidSetDataOrder(SPI_tenuDataOrder u8Order);
106
107
/*******************************************************************************
*******************************/
108
109 extern LBTY_tenuErrorStatus SPI_u8SetTransmit(u8 u8Char);
110 extern LBTY_tenuErrorStatus SPI_u8GetTransmit(u8* pu8Char);
111
112 extern LBTY_tenuErrorStatus SPI_u8SetChar(u8 u8Char, u8 u8Index);
113 extern LBTY_tenuErrorStatus SPI_u8GetChar(u8* pu8Char, u8 u8Index);
114
115 extern void SPI_vidSetStr(u8* pu8Transmit, u8 u8Index);
116 extern void SPI_vidGetStr(u8* pu8Receive, u8 u8Index);
117
118
/*******************************************************************************
*******************************/
119
120 extern void SPI_vidSetCallBack_OverFlow(void (*pCallBack)(void));
121
122 #endif /* SPI_INT_H_ */
123 /************************** E N D (SPI_int.h) ***************************/
```

SPI_prg.c File Reference

```
#include "LBTY_int.h"
#include "LBIT_int.h"
#include "LCTY_int.h"
#include "INTP.h"
#include "GPIO_int.h"
#include "GPIO_cfg.h"
#include "SPI_int.h"
#include "SPI_cfg.h"
#include "SPI_priv.h"
```

Include dependency graph for SPI_prg.c:



Functions

- void SPI_vidSetConfig (SPI_tstrConfig const *const pstrConfig)
- void SPI_vidSRestConfig (SPI_tstrConfig *const pstrConfig)
- void SPI_vidInit (void)
- void SPI_vidEnable (void)
- void SPI_vidDisable (void)
- void SPI_vidEnableINT (void)
- void SPI_vidDisableINT (void)
- LBTY_tenuErrorStatus SPI_vidSetPrescaler (SPI_tenuClockRate u8Prescaler)
- LBTY_tenuErrorStatus SPI_vidSetClockPhase (SPI_tenuClockPhase u8Phase)
- LBTY_tenuErrorStatus SPI_vidSetClockPolarity (SPI_tenuClockPolarity u8Polarity)
- LBTY_tenuErrorStatus SPI_vidSetDataOrder (SPI_tenuDataOrder u8Order)
- LBTY_tenuErrorStatus SPI_u8SetTransmit (u8 u8Char)
- LBTY_tenuErrorStatus SPI_u8GetTransmit (u8 *pu8Char)
- LBTY_tenuErrorStatus SPI_u8SetChar (u8 u8Char, u8 u8Index)
- LBTY_tenuErrorStatus SPI_u8GetChar (u8 *pu8Char, u8 u8Index)
- void SPI_vidSetStr (u8 *pu8Transmit, u8 u8Index)
- void SPI_vidGetStr (u8 *pu8Receive, u8 u8Index)
- void SPI_vidSetCallBack_OverFlow (void(*pCallBack)(void))
- ISR (SPI_STC_vect)

Variables

- static volatile u8 SPI_u8Flag_GLB
- static void(* pFuncCallBack_SPI)(void) = INTP_vidCallBack
- static SPI_tstrConfig strSPI_Config_GLB

Function Documentation

ISR (SPI_STC_vect)

```
291                 {
292     pFuncCallBack_SPI();
293     SPI_u8Flag_GLB = LBTY_SET;
294 }
```

LBTY_tenuErrorStatus SPI_u8GetChar (u8 * *pu8Char*, u8 *u8Index*)

```
245                                                                 {
246     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
247
248     if(strSPI_Config_GLB.m_Mode == SPI_Master){
249
250         GPIO_u8SetPinValue(kastrSSConfiguration_GLB[u8Index].m_Port,
kastrSSConfiguration_GLB[u8Index].m_Pin,  SPI_SS_ENABLE);
251         u8RetErrorState = SPI_u8GetTransmit(pu8Char);
252         GPIO_u8SetPinValue(kastrSSConfiguration_GLB[u8Index].m_Port,
kastrSSConfiguration_GLB[u8Index].m_Pin, SPI_SS_DISABLE);
253
254     }else if(strSPI_Config_GLB.m_Mode == SPI_Slave){
255
256         u8 u8State = SPI_SS_DISABLE;
257         GPIO_u8GetPinValue(SPI_PORT, SPI_SS_PIN, &u8State);
258
259         if(u8State == SPI_SS_ENABLE){
260             u8RetErrorState = SPI_u8GetTransmit(pu8Char);
261         }
262
263     }else{
264         while(1);
265     }
266
267     return u8RetErrorState;
268 }
```

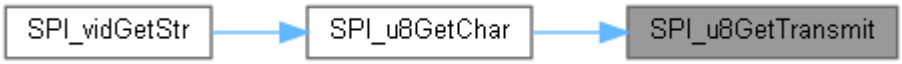
Here is the call graph for this function:

Here is the caller graph for this function:

LBTY_tenuErrorStatus SPI_u8GetTransmit (u8 * *pu8Char*)

```
207                                                     {
208     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
209
210     if(!(S_SPI->m_SPSR.sBits.m_WCOL)){
211         while((!S_SPI->m_SPSR.sBits.m_SPIF) && (!SPI_u8Flag_GLB));
212         SPI_u8Flag_GLB = LBTY_RESET;
213         *pu8Char = S_SPI->m_SPDR;
214     }else{
215         u8RetErrorState = LBTY_NOK;
216     }
217
218     return u8RetErrorState;
219 }
```

Here is the caller graph for this function:



LBTY_tenuErrorStatus SPI_u8SetChar (u8 *u8Char*, u8 *u8Index*)

```
221                                                                 {
222     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
223
224     if(strSPI_Config_GLB.m_Mode == SPI_Master){
225
226         GPIO_u8SetPinValue(kastrSSConfiguration_GLB[u8Index].m_Port,
kastrSSConfiguration_GLB[u8Index].m_Pin,  SPI_SS_ENABLE);
227         u8RetErrorState = SPI_u8SetTransmit(u8Char);
228         GPIO_u8SetPinValue(kastrSSConfiguration_GLB[u8Index].m_Port,
kastrSSConfiguration_GLB[u8Index].m_Pin, SPI_SS_DISABLE);
229
230     }else if(strSPI_Config_GLB.m_Mode == SPI_Slave){
231
232         u8 u8State = SPI_SS_DISABLE;
233         GPIO_u8GetPinValue(SPI_PORT, SPI_SS_PIN, &u8State);
234
235         if(u8State == SPI_SS_ENABLE){
```

```
236             u8RetErrorState = SPI_u8SetTransmit(u8Char);
237         }
238
239     }else{
240         while(1);
241     }
242
243     return u8RetErrorState;
244 }
```

Here is the call graph for this function:

Here is the caller graph for this function:

LBTY_tenuErrorStatus SPI_u8SetTransmit (u8 *u8Char*)

```
194                                                     {
195     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
196
197     if(!(S_SPI->m_SPSR.sBits.m_WCOL)){
198         S_SPI->m_SPDR = u8Char;
199         while((!S_SPI->m_SPSR.sBits.m_SPIF) && (!SPI_u8Flag_GLB));
200         SPI_u8Flag_GLB = LBTY_RESET;
201     }else{
202         u8RetErrorState = LBTY_NOK;
203     }
204
205     return u8RetErrorState;
206 }
```

Here is the caller graph for this function:

void SPI_vidDisable (void)

```
114                             {
115     S_SPI->m_SPCR.sBits.m_SPE  = strSPI_Config_GLB.m_SPIEN = LBTY_RESET;
116 }
```

void SPI_vidDisableINT (void)

```
121                               {
122     S_SPI->m_SPCR.sBits.m_SPIE  = strSPI_Config_GLB.m_SPIIE = LBTY_RESET;
123 }
```

void SPI_vidEnable (void)

```
111                            {
112     S_SPI->m_SPCR.sBits.m_SPE  = strSPI_Config_GLB.m_SPIEN = LBTY_SET;
113 }
```

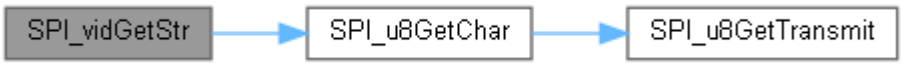
void SPI_vidEnableINT (void)

```
118                              {
119     S_SPI->m_SPCR.sBits.m_SPIE  = strSPI_Config_GLB.m_SPIIE = LBTY_SET;
120 }
```

void SPI_vidGetStr (u8 * *pu8Receive*, u8 *u8Index*)

```
277                                                     {
278     do{
279         if(SPI_u8GetChar(pu8Receive, u8Index)){
280             break;
281         }
282     }while(*(pu8Receive++) != '\0');
283 }
```

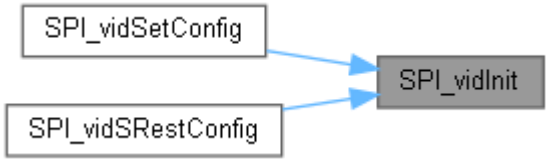
Here is the call graph for this function:



void SPI_vidInit (void)

```
82                          {
83
84      S_SPI->m_SPCR.sBits.m_MSTR = strSPI_Config_GLB.m_Mode;
85      if(strSPI_Config_GLB.m_Mode == SPI_Master){
86          GPIO_u8SetMaskDirection(SPI_PORT, SPI_MODE_MASK, PORT_OUTPUT);
87          GPIO_u8SetPinDirection (SPI_PORT, SPI_MISO_PIN , PIN_INPUT  );
88          for(u8 i = SPI_SS_NUM ; i-- ; ){
89              GPIO_u8SetPinDirection(kastrSSConfiguration_GLB[i].m_Port,
kastrSSConfiguration_GLB[i].m_Pin, PIN_OUTPUT);
90              GPIO_u8SetPinValue    (kastrSSConfiguration_GLB[i].m_Port,
kastrSSConfiguration_GLB[i].m_Pin, SPI_SS_DISABLE);
91          }
92      }else if(strSPI_Config_GLB.m_Mode == SPI_Slave){
93          GPIO_u8SetMaskDirection(SPI_PORT, SPI_MODE_MASK, PORT_INPUT);
94          GPIO_u8SetPinDirection (SPI_PORT, SPI_MISO_PIN , PIN_OUTPUT);
95      }
96
97      //SPI_vidSetPrescaler(strSPI_Config_GLB.m_Prescaler);
98      S_SPI->m_SPSR.sBits.m_SPI2X= GET_BIT(strSPI_Config_GLB.m_Prescaler,
SPI_SPI2X_BIT);
99      S_SPI->m_SPCR.sBits.m_SPR  = GET_MASK(strSPI_Config_GLB.m_Prescaler,
SPI_SPR_MASK);
100     //SPI_vidSetClockPhase(strSPI_Config_GLB.m_Phase);
101     S_SPI->m_SPCR.sBits.m_CPHA = strSPI_Config_GLB.m_Phase;
102     //SPI_vidSetClockPolarity(strSPI_Config_GLB.m_Polarity);
103     S_SPI->m_SPCR.sBits.m_CPOL = strSPI_Config_GLB.m_Polarity;
104     //SPI_vidSetDataOrder(strSPI_Config_GLB.m_Order);
105     S_SPI->m_SPCR.sBits.m_DORD = strSPI_Config_GLB.m_Order;
106
107     S_SPI->m_SPCR.sBits.m_SPIE = strSPI_Config_GLB.m_SPIIE;
108     S_SPI->m_SPCR.sBits.m_SPE  = strSPI_Config_GLB.m_SPIEN;
109 }
```

Here is the caller graph for this function:



void SPI_vidSetCallBack_OverFlow (void(*)(void) *pCallBack*)

```
287                                                         {
288     pFuncCallBack_SPI = pCallBack;
289 }
```

LBTY_tenuErrorStatus SPI_vidSetClockPhase (SPI_tenuClockPhase *u8Phase*)

```
147                                                                 {
148     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
149
150     switch(u8Phase){
151         case SPI_Leading_Sample:
152         case SPI_Leading_Setup:
153             S_SPI->m_SPCR.sBits.m_CPHA = strSPI_Config_GLB.m_Phase = u8Phase;
154             break;
155         default:
156             u8RetErrorState = LBTY_NOK;
157     }
158
159     return u8RetErrorState;
160 }
```

LBTY_tenuErrorStatus SPI_vidSetClockPolarity (SPI_tenuClockPolarity *u8Polarity*)

```
162
{
163     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
164
165     switch(u8Polarity){
166         case SPI_Leading_Rising:
167         case SPI_Leading_Falling:
168             S_SPI->m_SPCR.sBits.m_CPOL = strSPI_Config_GLB.m_Polarity =
u8Polarity;
```

```
169             break;
170         default:
171             u8RetErrorState = LBTY_NOK;
172     }
173
174     return u8RetErrorState;
175 }
```

void SPI_vidSetConfig (SPI_tstrConfig const *const *pstrConfig*)

```
60                                                                  {
61     if(pstrConfig != LBTY_NULL){
62         strSPI_Config_GLB = *pstrConfig;
63     }
64     SPI_vidInit();
65 }
```

Here is the call graph for this function:

LBTY_tenuErrorStatus SPI_vidSetDataOrder (SPI_tenuDataOrder *u8Order*)

```
177                                                                 {
178     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
179
180     switch(u8Order){
181         case SPI_MSB_Frist:
182         case SPI_LSB_Frist:
183             S_SPI->m_SPCR.sBits.m_DORD = strSPI_Config_GLB.m_Order = u8Order;
184             break;
185         default:
186             u8RetErrorState = LBTY_NOK;
187     }
188
189     return u8RetErrorState;
190 }
```

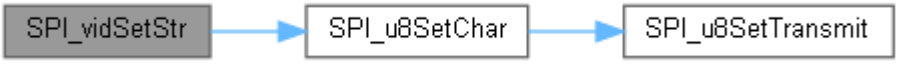
LBTY_tenuErrorStatus SPI_vidSetPrescaler (SPI_tenuClockRate *u8Prescaler*)

```
125                                                                 {
126     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
127
128     switch(u8Prescaler){
129         case SPI_Prescaler_4:
130         case SPI_Prescaler_16:
131         case SPI_Prescaler_64:
132         case SPI_Prescaler_128:
133         case SPI_Prescaler_2:
134         case SPI_Prescaler_8:
135         case SPI_Prescaler_32:
136             S_SPI->m_SPSR.sBits.m_SPI2X= GET_BIT(u8Prescaler, SPI_SPI2X_BIT);
137             S_SPI->m_SPCR.sBits.m_SPR  = GET_MASK(u8Prescaler, SPI_SPR_MASK);
138             strSPI_Config_GLB.m_Prescaler = u8Prescaler;
139             break;
140         default:
141             u8RetErrorState = LBTY_NOK;
142     }
143
144     return u8RetErrorState;
145 }
```

void SPI_vidSetStr (u8 * *pu8Transmit*, u8 *u8Index*)

```
270                                                 {
271     while(*pu8Transmit){
272         if(SPI_u8SetChar(*(pu8Transmit++), u8Index)){
273             break;
274         }
275     }
276 }
```

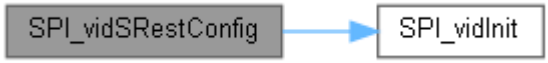
Here is the call graph for this function:



void SPI_vidSRestConfig (SPI_tstrConfig *const *pstrConfig*)

```
67                                                          {
68      strSPI_Config_GLB.m_Mode         = SPI_MODE;
69      strSPI_Config_GLB.m_Prescaler    = SPI_CLOCK_PRESCALER;
70      strSPI_Config_GLB.m_Phase        = SPI_CLOCK_PHASE;
71      strSPI_Config_GLB.m_Polarity     = SPI_CLOCK_POLARITY;
72      strSPI_Config_GLB.m_Order        = SPI_DATA_ORDER;
73      strSPI_Config_GLB.m_SPIEN        = SPI_INIT;
74      strSPI_Config_GLB.m_SPIIE        = SPI_INT;
75
76      if(pstrConfig != LBTY_NULL){
77          *pstrConfig = strSPI_Config_GLB;
78      }
79      SPI_vidInit();
80  }
```

Here is the call graph for this function:



Variable Documentation

void(* pFuncCallBack_SPI) (void) (void) = INTP_vidCallBack `[static]`

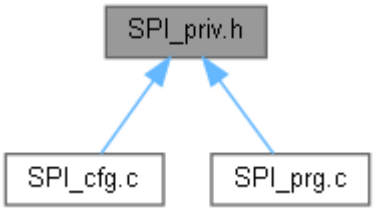
volatile u8 SPI_u8Flag_GLB `[static]`

SPI_tstrConfig strSPI_Config_GLB `[static]`

```
Initial value:= {
        .m_Mode          = SPI_MODE,
        .m_Prescaler     = SPI_CLOCK_PRESCALER,
        .m_Phase         = SPI_CLOCK_PHASE,
        .m_Polarity      = SPI_CLOCK_POLARITY,
        .m_Order         = SPI_DATA_ORDER,

        .m_SPIEN         = SPI_INIT,
        .m_SPIIE         = SPI_INT,
}
```

SPI_priv.h File Reference

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



Data Structures

union SPCR_type: *Type define of Union bit field of "SPI Control Register"*

union SPSR_type: *Type define of Union bit field of "SPI Status Register"*

struct SPI_type: *SPI Registers*

Macros

- #define S_SPI ((SPI_type* const)0x2DU)
- #define SPCR (*(volatile u8* const)0x2DU)
- #define SPSR (*(volatile u8* const)0x2EU)
- #define SPDR (*(volatile u8* const)0x2FU)
- #define SPI_SPR_MASK 3u
- #define SPI_SPI2X_BIT 2u
- #define SPI_PORT B
- #define SPI_MOSI_PIN GPIO_SPI_MOSI
- #define SPI_MISO_PIN GPIO_SPI_MISO
- #define SPI_SCK_PIN GPIO_SPI_SCK
- #define SPI_SS_PIN GPIO_SPI_SS
- #define SPI_MODE_MASK (1<<SPI_SS_PIN) | (1<<SPI_SCK_PIN) | (1<<SPI_MOSI_PIN)
- #define SPI_SS_ENABLE PIN_Low
- #define SPI_SS_DISABLE PIN_High

Macro Definition Documentation

#define S_SPI ((SPI_type* const)0x2DU)

SPI

```
#define SPCR   (*(volatile u8* const)0x2DU)

#define SPDR   (*(volatile u8* const)0x2FU)

#define SPI_MISO_PIN   GPIO_SPI_MISO

#define SPI_MODE_MASK   (1<<SPI_SS_PIN) | (1<<SPI_SCK_PIN) | (1<<SPI_MOSI_PIN)

#define SPI_MOSI_PIN   GPIO_SPI_MOSI

#define SPI_PORT   B

#define SPI_SCK_PIN   GPIO_SPI_SCK

#define SPI_SPI2X_BIT   2u

#define SPI_SPR_MASK   3u

#define SPI_SS_DISABLE   PIN_High

#define SPI_SS_ENABLE   PIN_Low

#define SPI_SS_PIN   GPIO_SPI_SS

#define SPSR   (*(volatile u8* const)0x2EU)
```

SPI_priv.h

```
Go to the documentation of this file.1 /*
****************************************************************************** */
2 /* ********************** FILE DEFINITION SECTION ************************** */
3 /* ****************************************************************************** */
4 /* File Name    : SPI_priv.h                                                    */
5 /* Author       : MAAM                                                          */
6 /* Version      : v01.2                                                         */
7 /* date         : Apr 12, 2023                                                  */
8 /* ****************************************************************************** */
9 /* ************************ HEADER FILES INCLUDES ***************************  */
10 /* ****************************************************************************** */
11
12 #ifndef SPI_PRIV_H_
13 #define SPI_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_SPR : 2;
25         __IO u8 m_CPHA: 1;
26         __IO u8 m_CPOL: 1;
27         __IO u8 m_MSTR: 1;
28         __IO u8 m_DORD: 1;
29         __IO u8 m_SPE : 1;
30         __IO u8 m_SPIE: 1;
31     }sBits;
32 }SPCR_type;
33
34 /*****************************************************************************/
35
38 typedef union{
39     u8 u_Reg;
40     struct {
41         __IO u8 m_SPI2X: 1;
42         __I  u8        : 5;
43         __I  u8 m_WCOL : 1;
44         __I  u8 m_SPIF : 1;
45     }sBits;
46 }SPSR_type;
47
48 /*****************************************************************************/
49
52 typedef struct{
53     __IO SPCR_type m_SPCR;
54     __IO SPSR_type m_SPSR;
55     __IO u8        m_SPDR;
56 }SPI_type;
57
58 /* ****************************************************************************** */
59 /* ************************** MACRO/DEFINE SECTION **************************** */
60 /* ****************************************************************************** */
61
63 #define S_SPI           ((SPI_type* const)0x2DU)
64 #define SPCR            (*(volatile u8* const)0x2DU)
65 #define SPSR            (*(volatile u8* const)0x2EU)
66 #define SPDR            (*(volatile u8* const)0x2FU)
67
68 /* ****************************************************************************** */
69
70 #define SPI_SPR_MASK            3u
71 #define SPI_SPI2X_BIT           2u
72
73 /* ****************************************************************************** */
74
75 #define SPI_PORT                B
76 #define SPI_MOSI_PIN            GPIO_SPI_MOSI
77 #define SPI_MISO_PIN            GPIO_SPI_MISO
78 #define SPI_SCK_PIN             GPIO_SPI_SCK
79 #define SPI_SS_PIN              GPIO_SPI_SS
```

```
80
81 #define SPI_MODE_MASK           (1<<SPI_SS_PIN) | (1<<SPI_SCK_PIN) | (1<<SPI_MOSI_PIN)
82
83 #define SPI_SS_ENABLE           PIN_Low
84 #define SPI_SS_DISABLE          PIN_High
85
86 /* ********************************************************************************** */
87 /* ****************************** CONST SECTION ************************************* */
88 /* ********************************************************************************** */
89
90 /* ********************************************************************************** */
91 /* ***************************** VARIABLE SECTION *********************************** */
92 /* ********************************************************************************** */
93
94 /* ********************************************************************************** */
95 /* ***************************** FUNCTION SECTION *********************************** */
96 /* ********************************************************************************** */
97
98
99 #endif /* SPI_PRIV_H_ */
100 /***************************** E N D (SPI_priv.h) ******************************/
```