SWC_USART

Version v1.0
7/23/2023 7:10:00 PM

Table of Contents

Data Structure Index

Data Structures

Here are the data structures with brief descriptions:

File Index

File List

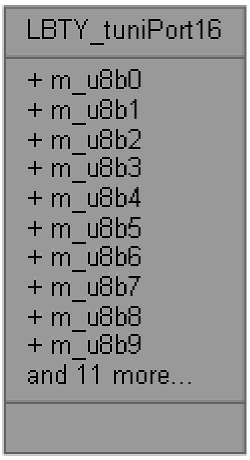
Here is a list of all files with brief descriptions:

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

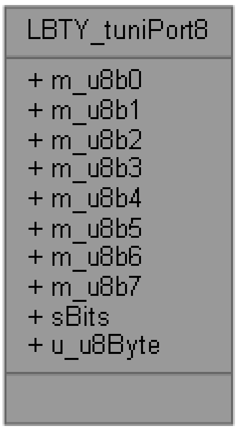
The documentation for this union was generated from the following file:

- 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

UCSRA_type Union Reference

: Type define of Union bit field of "USART Control and Status RegA"

`#include <USART_priv.h>`

Collaboration diagram for UCSRA_type:

Data Fields

- u8 u_Reg
- struct {
- __IO u8 m_MPCM: 1
- __IO u8 m_U2X: 1
- __I u8 m_PE: 1
- __I u8 m_DOR: 1
- __I u8 m_FE: 1
- __I u8 m_UDRE: 1
- __IO u8 m_TXC: 1
- __I u8 m_RXC: 1
- } sBits

Detailed Description

: Type define of Union bit field of "USART Control and Status RegA"

Type : Union **Unit** : None

Field Documentation

__I u8 m_DOR

Data OverRun

__I u8 m_FE

Frame Error

__IO u8 m_MPCM

Multi-processor Communication Mode

__I u8 m_PE

Parity Error

__I u8 m_RXC

USART Receive Complete

__IO u8 m_TXC

USART Transmit Complete

__IO u8 m_U2X

Double the USART Transmission Speed

__I u8 m_UDRE

USART Data Register Empty

struct { ... } sBits

u8 u_Reg

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

USART_priv.h

UCSRB_type Union Reference

: Type define of Union bit field of "USART Control and Status RegB"
`#include <USART_priv.h>`
Collaboration diagram for UCSRB_type:

UCSRB_type
+ u_Reg
+ m_TXB8
+ m_RXB8
+ m_UCSZ2
+ m_TXEN
+ m_RXEN
+ m_UDRIE
+ m_TXCIE
+ m_RXCIE
+ sBits

Data Fields

- u8 u_Reg
- struct {
- __IO u8 m_TXB8: 1
- __I u8 m_RXB8: 1
- __IO u8 m_UCSZ2: 1
- __IO u8 m_TXEN: 1
- __IO u8 m_RXEN: 1
- __IO u8 m_UDRIE: 1
- __IO u8 m_TXCIE: 1
- __IO u8 m_RXCIE: 1
- } sBits

Detailed Description

: Type define of Union bit field of "USART Control and Status RegB"

Type : Union **Unit** : None

Field Documentation

__I u8 m_RXB8

Receive Data Bit 8

__IO u8 m_RXCIE

RX Complete Interrupt Enable

__IO u8 m_RXEN

Receiver Enable

__IO u8 m_TXB8

Transmit Data Bit 8

__IO u8 m_TXCIE

TX Complete Interrupt Enable

__IO u8 m_TXEN

Transmitter Enable

__IO u8 m_UCSZ2

Character Size

__IO u8 m_UDRIE

USART Data Register Empty Interrupt Enable

struct { ... } sBits

u8 u_Reg

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

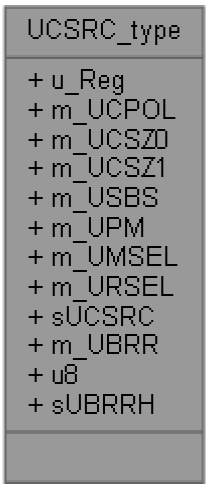
USART_priv.h

UCSRC_type Union Reference

: Type define of Union bit field of "USART Control and Status RegC"

`#include <USART_priv.h>`

Collaboration diagram for UCSRC_type:



Data Fields

- u8 u_Reg
- struct {
- __IO u8 m_UCPOL: 1
- __IO u8 m_UCSZ0: 1
- __IO u8 m_UCSZ1: 1
- __IO u8 m_USBS: 1
- __IO u8 m_UPM: 2
- __IO u8 m_UMSEL: 1
- __IO u8 m_URSEL: 1
- } sUCSRC
- struct {
- __IO u8 m_UBRR: 4
- __IO u8: 3
- __IO u8 m_URSEL: 1
- } sUBRRH

Detailed Description

: Type define of Union bit field of "USART Control and Status RegC"

Type : Union **Unit** : None

Field Documentation

__IO u8 m_UBRR

USART Baud Rate High

__IO u8 m_UCPOL

Clock Polarity

__IO u8 m_UCSZ0

Character Size 0

__IO u8 m_UCSZ1

Character Size 1

__IO u8 m_UMSEL

USART Mode Select

__IO u8 m_UPM

Parity Mode

__IO u8 m_URSEL

Register Select

__IO u8 m_USBS

Stop Bit Select

struct { ... } sUBRRH

struct { ... } sUCSRC

__IO u8

Reversed

u8 u_Reg

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

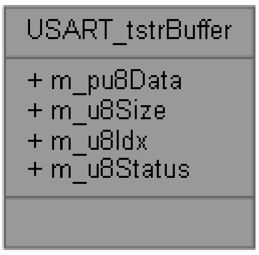
USART_priv.h

USART_tstrBuffer Struct Reference

: UART TX/RX Buffer

`#include <USART_priv.h>`

Collaboration diagram for USART_tstrBuffer:



Data Fields

- pu8 m_pu8Data
- u8 m_u8Size
- u8 m_u8Idx
- u8 m_u8Status

Detailed Description

: UART TX/RX Buffer

Type : Struct **Unit** : None

Field Documentation

pu8 m_pu8Data

Data Pointer

u8 m_u8Idx

Index of Data

u8 m_u8Size

Size of Data Bytes

u8 m_u8Status

Current Status

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

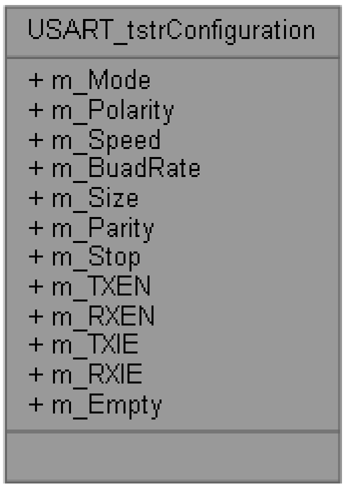
USART_priv.h

USART_tstrConfiguration Struct Reference

: type define of structure for UART/USART Configuration

```
#include <USART_int.h>
```

Collaboration diagram for USART_tstrConfiguration:



Data Fields

- USART_tenumModeSelect m_Mode
- USART_tenumClockPolarity m_Polarity
- USART_tenumSpeed m_Speed
- USART_tenumBuadRate m_BuadRate
- USART_tenumCharSize m_Size
- USART_tenumParityMode m_Parity
- USART_tenumStopBit m_Stop
- LBTY_tenuFlagStatus m_TXEN
- LBTY_tenuFlagStatus m_RXEN
- LBTY_tenuFlagStatus m_TXIE
- LBTY_tenuFlagStatus m_RXIE
- LBTY_tenuFlagStatus m_Empty

Detailed Description

: type define of structure for UART/USART Configuration

Type : struct **Unit** : None

Field Documentation

USART_tenumBuadRate m_BuadRate

BaudRate Register Value

LBTY_tenuFlagStatus m_Empty

EmptyInterrupt Enable Flag

USART_tenumModeSelect m_Mode

Sync or Async Mode

USART_tenumParityMode m_Parity

Parity Bit

USART_tenumClockPolarity m_Polarity

Clock Polarity

LBTY_tenuFlagStatus m_RXEN

RX Enable Flag

LBTY_tenuFlagStatus m_RXIE

RX Interrupt Enable Flag

USART_tenumCharSize m_Size

Data Register Size

USART_tenumSpeed m_Speed

Speed Gain

USART_tenumStopBit m_Stop

Stop Bit

LBTY_tenuFlagStatus m_TXEN

TX Enable Flag

LBTY_tenuFlagStatus m_TXIE

TX Interrupt Enable Flag

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

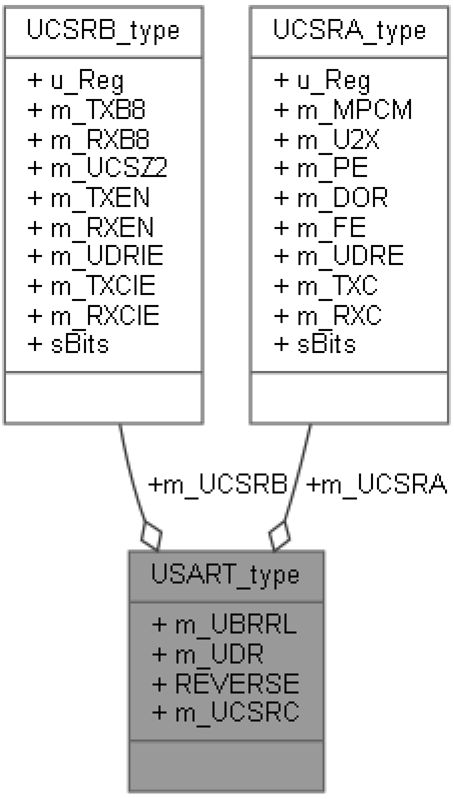
USART_int.h

USART_type Struct Reference

: UART Registers

`#include <USART_priv.h>`

Collaboration diagram for USART_type:



Data Fields

- __IO u8 m_UBRRL
- __IO UCSRB_type m_UCSRB
- __IO UCSRA_type m_UCSRA
- __IO u8 m_UDR
- __I u8 REVERSE [19]
- __IO u8 m_UCSRC

Detailed Description

: UART Registers

Type : Struct **Unit** : None

Field Documentation

__IO u8 m_UBRRL

USART Baud Rate

__IO UCSRA_type m_UCSRA

USART Control and Status Reg A

__IO UCSRB_type m_UCSRB

USART Control and Status Reg B

__IO u8 m_UCSRC

USART Control and Status Reg C

__IO u8 m_UDR

USART Data Reg

__I u8 REVERSE[19]

Reversed

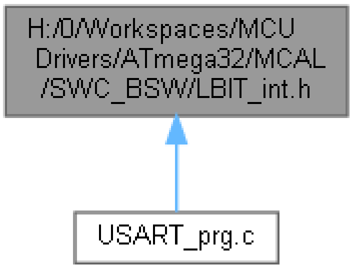
The documentation for this struct was generated from the following file:

USART_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:


H:/0/Workspaces/MCU Drivers/ATmega32/MCAL /SWC_BSW/LBIT_int.h
USART_prg.c

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

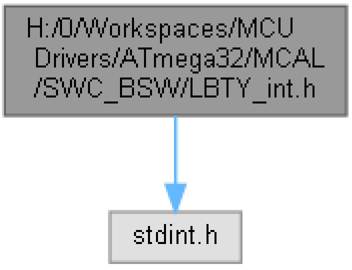
Go to the documentation of this file.

```c
/* ************************************************************************** */
/* ********************** FILE DEFINITION SECTION ************************** */
/* ************************************************************************** */
/* File Name   : LBIT_int.h                                                   */
/* Author      : MAAM                                                         */
/* Version     : v01                                                          */
/* date        : Mar 24, 2023                                                 */
/* description : Bitwise Library                                              */
/* ************************************************************************** */
/* ************************ HEADER FILES INCLUDES **************************  */
/* ************************************************************************** */

#ifndef LBIT_INT_H_
#define LBIT_INT_H_

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

/* ************************************************************************** */
/* ************************** MACRO/DEFINE SECTION ************************** */
/* ************************************************************************** */

#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 ASSIGN_BIT(REG,bit,value)                    do{                                \
                                                      REG &= ~(0x01u<<bit);             \
                                                      REG |=  ((value & 0x01u)<<bit);   \
                                                     }while(0)
*/

/*            bits together in an u8 register          */
#define CON_u8Bits(b7, b6, b5, b4, b3, b2, b1, b0)                                      \
(0b##b7##b6##b5##b4##b3##b2##b1##b0)

/*            bits together in an u16 register         */
#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)

/* ************************************************************************** */
/* ***************************** CONST SECTION ****************************** */
/* ************************************************************************** */

/* ************************************************************************** */
/* ***************************** VARIABLE SECTION *************************** */
/* ************************************************************************** */

/* ************************************************************************** */
/* **************************** FUNCTION SECTION **************************** */
/* ************************************************************************** */


#endif /* LBIT_INT_H_ */
/*************************** E N D (LBIT_int.h) ******************************/
```

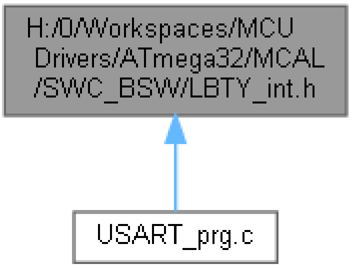
H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_BSW/LBTY_int.h File Reference

`#include <stdint.h>`

Include dependency graph for LBTY_int.h:


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

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


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

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	

```
              {
    LBTY_TRUE = 0x55,
    LBTY_FALSE = 0xAA
} 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	

```
              {
  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        /* Error is not available, wait for availability */
} LBTY_tenuErrorStatus;
```

enum LBTY_tenuFlagStatus

Flag Status type

Enumerator:

LBTY_RESET	
LBTY_SET	

```
              {
    LBTY_RESET = 0,
    LBTY_SET = !LBTY_RESET
} LBTY_tenuFlagStatus;
```

LBTY_int.h

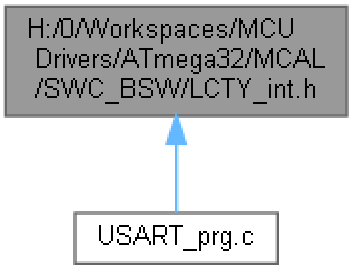
Go to the documentation of this file.

```c
/* ************************************************************************ */
/* ********************** FILE DEFINITION SECTION ************************** */
/* ************************************************************************** */
/* File Name   : LBTY_int.h                                                 */
/* Author      : MAAM                                                       */
/* Version     : v01                                                        */
/* date        : Mar 23, 2023                                               */
/* description : Basic Library                                              */
/* ************************************************************************** */
/* ************************ HEADER FILES INCLUDES *************************  */
/* ************************************************************************** */

#ifndef _LBTY_INT_H_
#define _LBTY_INT_H_

#include <stdint.h>

/* ************************************************************************** */
/* ******************************* TYPE_DEF SECTION ************************ */
/* ************************************************************************** */

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;

/* ************************************************************************** */
/* ************************* MACRO/DEFINE SECTION ************************** */
/* ************************************************************************** */

/************************************************************************/
#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 )

```

```c
#define LBTY_u64ZERO    ((u64)0x0000000000000000ULL)
#define LBTY_u64MAX     ((u64)0xFFFFFFFFFFFFFFFFULL)
#define LBTY_s64MAX     ((u64)0x7FFFFFFFFFFFFFFFLL )
#define LBTY_s64MIN     ((u64)0x8000000000000000LL )

/* *************************************************************************** */
/* ******************************* ENUM SECTION ****************************** */
/* *************************************************************************** */

typedef enum {
    LBTY_RESET = 0,
    LBTY_SET = !LBTY_RESET
} LBTY_tenuFlagStatus;

typedef enum {
  LBTY_TRUE = 0x55,
  LBTY_FALSE = 0xAA
} LBTY_tenuBoolean;

typedef enum {
  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        /* Error is not available, wait for availability */
} LBTY_tenuErrorStatus;

/* *************************************************************************** */
/* ******************************* STRUCT SECTION ***************************** */
/* *************************************************************************** */

typedef union {
  struct {
    u8 m_u8b0 :1;       // LSB
    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;       // MSB
  } sBits;
  u8 u_u8Byte;
} LBTY_tuniPort8;

typedef union {
  struct  {
    u8 m_u8b0  :1;      // LSB
    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;      // MSB
  } sBits;
  struct {
    u8 m_u8low;
    u8 m_u8high;
  } sBytes;
  u16 u_u16Word;
} LBTY_tuniPort16;

/* *************************************************************************** */
/* ***************************** FUNCTION SECTION **************************** */
/* *************************************************************************** */


#endif /* _LBTY_INT_H_ */
/*************************** 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:


H:/0/Workspaces/MCU Drivers/ATmega32/MCAL /SWC_BSW/LCTY_int.h
USART_prg.c

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

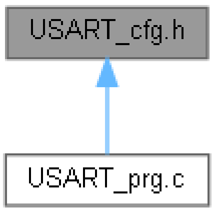
Go to the documentation of this file.

```c
/* ************************************************************************ */
/* ********************** FILE DEFINITION SECTION ************************** */
/* ************************************************************************ */
/* File Name   : LCTY_int.h                                                 */
/* Author      : MAAM                                                       */
/* Version     : v00                                                        */
/* date        : Apr 26, 2023                                               */
/* description : Compiler Library                                           */
/* ************************************************************************ */
/* ************************ HEADER FILES INCLUDES *************************  */
/* ************************************************************************ */

#ifndef LCTY_INT_H_
#define LCTY_INT_H_

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

/* ************************************************************************ */
/* ************************** MACRO/DEFINE SECTION ************************* */
/* ************************************************************************ */

/* prog memory attribute */
#define LCTY_PROGMEM                __attribute__((__progmem__))

/* pure attribute */
#define LCTY_PURE                   __attribute__((__pure__))

/* Abstraction for inlining */
//#define LCTY_INLINE               static inline
#define LCTY_INLINE                 __attribute__((always_inline)) static inline

/* define function as interrupt handler */
#define LCTY_INTERRUPT              __attribute__((interrupt))

/* Memory packed to pass Memory padding */
#define CTY_PACKED                  __attribute__((__packed__))

/* Const attribute */
#define LCTY_CONST                  __attribute__((__const__))

/* place variable in direct page */
#define LCTY_DPAGE                  __attribute__((dp))

/* do not place variable in direct page */
#define LCTY_NODPAGE                __attribute__((nodp))

/* Sections */
#define LCTY_SECTION(section)       __attribute__((section( # section)))

/* Abstraction for assembly command */
# define LCTY_ASM(cmd)              __asm__ __volatile__ ( # cmd ::)

/* ************************************************************************ */
/* ***************************** CONST SECTION ***************************** */
/* ************************************************************************ */

/* ************************************************************************ */
/* ***************************** VARIABLE SECTION ************************** */
/* ************************************************************************ */

/* ************************************************************************ */
/* **************************** FUNCTION SECTION *************************** */
/* ************************************************************************ */


#endif /* LCTY_INT_H_ */
/*************************** E N D (LCTY_int.h) ******************************/
```

main.c File Reference

USART_cfg.h File Reference

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



Macros

- #define USART_OPERATION_MODE USART_Asynchronous
- #define USART_OPERATION_POLARITY USART_Transmit_Rising_Receive_Falling
- #define USART_OPERATION_SPEED USART_Speed_x1
- #define USART_OPERATION_MULTI_PROCESSOR LBTY_RESET
- #define USART_OPERATION_FREQ F_CPU
- #define USART_BUAD_RATE_INIT USART_BuadRate_9600
- #define USART_CHAR_SIZE_INIT USART_8_bit
- #define USART_PARITY_BIT_INIT USART_Parity_Even
- #define USART_STOP_BIT_INIT USART_Stop_1_bit
- #define USART_TRANSMIT_INIT LBTY_SET
- #define USART_RECEIVE_INIT LBTY_SET
- #define USART_TRANSMIT_COMPLETE_INT LBTY_RESET
- #define USART_RECEIVE_COMPLETE_INT LBTY_RESET
- #define USART_DATA_REG_EMPTY_INT LBTY_RESET

Macro Definition Documentation

#define USART_BUAD_RATE_INIT   USART_BuadRate_9600

#define USART_CHAR_SIZE_INIT   USART_8_bit

#define USART_DATA_REG_EMPTY_INT   LBTY_RESET

#define USART_OPERATION_FREQ   F_CPU

#define USART_OPERATION_MODE   USART_Asynchronous

#define USART_OPERATION_MULTI_PROCESSOR   LBTY_RESET

#define USART_OPERATION_POLARITY   USART_Transmit_Rising_Receive_Falling

#define USART_OPERATION_SPEED   USART_Speed_x1

#define USART_PARITY_BIT_INIT   USART_Parity_Even

#define USART_RECEIVE_COMPLETE_INT   LBTY_RESET

#define USART_RECEIVE_INIT   LBTY_SET

#define USART_STOP_BIT_INIT   USART_Stop_1_bit

#define USART_TRANSMIT_COMPLETE_INT   LBTY_RESET

#define USART_TRANSMIT_INIT   LBTY_SET

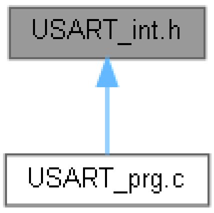
USART_cfg.h

Go to the documentation of this file.

```
/* ************************************************************************** */
/* ********************** FILE DEFINITION SECTION ************************** */
/* ************************************************************************** */
/* File Name   : USART_cfg.h                                                  */
/* Author      : MAAM                                                         */
/* Version     : v01.2                                                        */
/* date        : Apr 10, 2023                                                 */
/* ************************************************************************** */
/* ************************ HEADER FILES INCLUDES ************************* */
/* ************************************************************************** */

#ifndef USART_CFG_H_
#define USART_CFG_H_

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

/* ************************************************************************** */
/* ************************** MACRO/DEFINE SECTION ************************** */
/* ************************************************************************** */

#define USART_OPERATION_MODE                    USART_Asynchronous
#define USART_OPERATION_POLARITY                USART_Transmit_Rising_Receive_Falling
#define USART_OPERATION_SPEED                   USART_Speed_x1
#define USART_OPERATION_MULTI_PROCESSOR         LBTY_RESET
#define USART_OPERATION_FREQ                    F_CPU

#define USART_BUAD_RATE_INIT                    USART_BuadRate_9600
#define USART_CHAR_SIZE_INIT                    USART_8_bit
#define USART_PARITY_BIT_INIT                   USART_Parity_Even
#define USART_STOP_BIT_INIT                     USART_Stop_1_bit

#define USART_TRANSMIT_INIT                     LBTY_SET
#define USART_RECEIVE_INIT                      LBTY_SET

#define USART_TRANSMIT_COMPLETE_INT             LBTY_RESET
#define USART_RECEIVE_COMPLETE_INT              LBTY_RESET
#define USART_DATA_REG_EMPTY_INT                LBTY_RESET

/* ************************************************************************** */
/* ***************************** CONST SECTION ****************************** */
/* ************************************************************************** */

/* ************************************************************************** */
/* *************************** VARIABLE SECTION *************************** */
/* ************************************************************************** */

/* ************************************************************************** */
/* **************************** FUNCTION SECTION *************************** */
/* ************************************************************************** */


#endif /* USART_CFG_H_ */
/*************************** E N D (USART_cfg.h) *****************************/
```

USART_int.h File Reference

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



Data Structures

struct USART_tstrConfiguration*: type define of structure for UART/USART Configuration*

Enumerations

- enum USART_tenumModeSelect { USART_Asynchronous = (u8)0u, USART_Synchronous }
- enum USART_tenumSpeed { USART_Speed_x1 = (u8)0u, USART_Speed_x2 }
- enum USART_tenumClockPolarity { USART_Transmit_Rising_Receive_Falling = (u8)0u, USART_Transmit_Falling_Receive_Rising }
- enum USART_tenumCharSize { USART_5_bit = (u8)0u, USART_6_bit, USART_7_bit, USART_8_bit, USART_9_bit = (u8)7u }
- enum USART_tenumParityMode { USART_Parity_Disable = (u8)0u, USART_Parity_Reserved, USART_Parity_Even, USART_Parity_Odd }
- enum USART_tenumStopBit { USART_Stop_1_bit = (u8)0u, USART_Stop_2_bit }
- enum USART_tenumBuadRate { USART_BuadRate_1200 = 1200u, USART_BuadRate_2400 = 2400u, USART_BuadRate_4800 = 4800u, USART_BuadRate_9600 = 9600u, USART_BuadRate_14400 = 14400u, USART_BuadRate_19200 = 19200u, USART_BuadRate_28800 = 28800u, USART_BuadRate_38400 = 38400u, USART_BuadRate_57600 = 57600u, USART_BuadRate_76600 = 76600u, USART_BuadRate_115200 = 115200u, USART_BuadRate_230400 = 230400u, USART_BuadRate_250000 = 250000u, USART_BuadRate_500000 = 500000u, USART_BuadRate_1000000 = 1000000u }

Functions

- void USART_vidSetConfig (USART_tstrConfiguration const *const pstrConfig)
- void USART_vidResetConfig (USART_tstrConfiguration *const pstrConfig)
- void UART_vidInit (void)
- void USART_vidTransmitterEnable (void)
- void USART_vidTransmitterDisable (void)
- void USART_vidReceiverEnable (void)
- void USART_vidReceiverDisable (void)
- LBTY_tenuErrorStatus USART_u8SetBuadRate (USART_tenumBuadRate u32BuadRate)
- LBTY_tenuErrorStatus USART_u8SetCharSize (USART_tenumCharSize u8CharSize)
- LBTY_tenuErrorStatus USART_u8SetParityMode (USART_tenumParityMode u8Parity)
- LBTY_tenuErrorStatus USART_u8SetStopBit (USART_tenumStopBit u8Stop)
- u8 USART_u8Available (void)
- void USART_vidFlush (void)
- LBTY_tenuErrorStatus USART_u8SetTransmit (void *pvidTransmit)
- LBTY_tenuErrorStatus USART_u8GetTransmit (void *pvidTransmit)
- void USART_vidSetChar (u8 u8Char)
- void USART_vidGetChar (u8 *pu8Char)
- void USART_vidSetStrLine (u8 *pu8Transmit)

- void USART_vidSetStr (u8 *pu8Transmit)
- void USART_vidGetStr (u8 *pu8Receive)
- LBTY_tenuErrorStatus USART_u8SendBuffer (u8 *pu8Data, u8 u8Size)
- LBTY_tenuErrorStatus USART_u8ReceiveBuffer (u8 *pu8Data, u8 u8Size)
- void USART_vidEnableReceiveCompleteINT (void)
- void USART_vidEnableTransmitCompleteINT (void)
- void USART_vidEnableDataRegEmptyINT (void)
- void USART_vidDisableReceiveCompleteINT (void)
- void USART_vidDisableTransmitCompleteINT (void)
- void USART_vidDisableDataRegEmptyINT (void)
- void USART_vidSetCallBack_Empty (void(*pCallBack)(void))
- void USART_vidSetCallBack_TX (void(*pCallBack)(void))
- void USART_vidSetCallBack_RX (void(*pCallBack)(void))

Enumeration Type Documentation

enum USART_tenumBuadRate

Enumerator:

USART_BuadRate_1200	
USART_BuadRate_2400	
USART_BuadRate_4800	
USART_BuadRate_9600	
USART_BuadRate_14400	
USART_BuadRate_19200	
USART_BuadRate_28800	
USART_BuadRate_38400	
USART_BuadRate_57600	
USART_BuadRate_76600	
USART_BuadRate_115200	
USART_BuadRate_230400	
USART_BuadRate_250000	
USART_BuadRate_500000	
USART_BuadRate_1000000	

```
        {
    USART_BuadRate_1200      = 1200u,
    USART_BuadRate_2400      = 2400u,
    USART_BuadRate_4800      = 4800u,
    USART_BuadRate_9600      = 9600u,
    USART_BuadRate_14400     = 14400u,
    USART_BuadRate_19200     = 19200u,
    USART_BuadRate_28800     = 28800u,
    USART_BuadRate_38400     = 38400u,
    USART_BuadRate_57600     = 57600u,
    USART_BuadRate_76600     = 76600u,
    USART_BuadRate_115200    = 115200u,
    USART_BuadRate_230400    = 230400u,
    USART_BuadRate_250000    = 250000u,
    USART_BuadRate_500000    = 500000u,
    USART_BuadRate_1000000   = 1000000u,
}USART_tenumBuadRate;
```

enum USART_tenumCharSize

Enumerator:

USART_5_bit	
USART_6_bit	
USART_7_bit	
USART_8_bit	
USART_9_bit	

```
            {
    USART_5_bit = (u8)0u,
    USART_6_bit,
    USART_7_bit,
    USART_8_bit,
    USART_9_bit = (u8)7u,
}USART_tenumCharSize;
```

enum USART_tenumClockPolarity

Enumerator:

USART_Transmit_Rising_Receive_Falling	
USART_Transmit_Falling_Receive_Rising	

```
            {
    USART_Transmit_Rising_Receive_Falling = (u8)0u,
    USART_Transmit_Falling_Receive_Rising,
}USART_tenumClockPolarity;
```

enum USART_tenumModeSelect

Enumerator:

USART_Asynchronous	
USART_Synchronous	

```
            {
    USART_Asynchronous = (u8)0u,
    USART_Synchronous,
}USART_tenumModeSelect;
```

enum USART_tenumParityMode

Enumerator:

USART_Parity_Disable	
USART_Parity_Reserved	
USART_Parity_Even	

USART_Parity_Odd	

```
            {
    USART_Parity_Disable = (u8)0u,
    USART_Parity_Reserved,
    USART_Parity_Even,
    USART_Parity_Odd,
}USART_tenumParityMode;
```

enum USART_tenumSpeed

Enumerator:

USART_Speed_x1	
USART_Speed_x2	

```
            {
    USART_Speed_x1 = (u8)0u,
    USART_Speed_x2,
}USART_tenumSpeed;
```

enum USART_tenumStopBit

Enumerator:

USART_Stop_1_bit	
USART_Stop_2_bit	

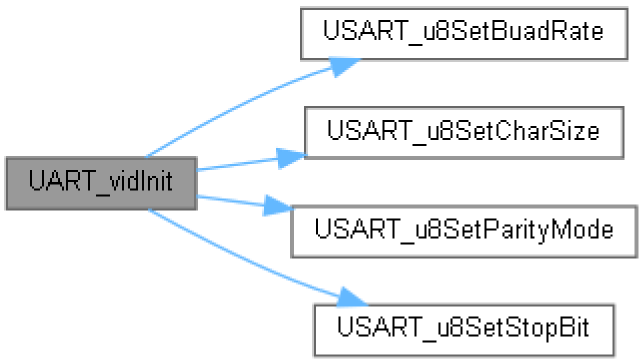
```
            {
    USART_Stop_1_bit = (u8)0u,
    USART_Stop_2_bit,
}USART_tenumStopBit;
```

Function Documentation

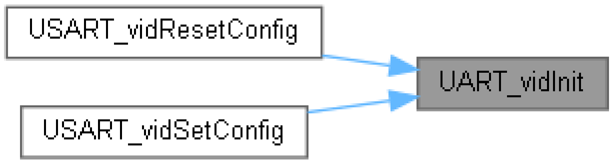
void UART_vidInit (void)

```
                        {

    strUCSRC.sUCSRC.m_URSEL = USART_UCSRC_Reg;
    strUCSRC.sUCSRC.m_UMSEL = strUSART_Config_GLB.m_Mode;
    strUCSRC.sUCSRC.m_UCPOL = strUSART_Config_GLB.m_Polarity;
    S_USART->m_UCSRC = strUCSRC.u_Reg;

    S_USART->m_UCSRA.sBits.m_MPCM = USART_OPERATION_MULTI_PROCESSOR;
    S_USART->m_UCSRA.sBits.m_U2X  = strUSART_Config_GLB.m_Speed;

    USART_u8SetBuadRate   (strUSART_Config_GLB.m_BuadRate);
    USART_u8SetCharSize   (strUSART_Config_GLB.m_Size);
    USART_u8SetParityMode(strUSART_Config_GLB.m_Parity);
    USART_u8SetStopBit    (strUSART_Config_GLB.m_Stop);

    strUCSRC.sUCSRC.m_URSEL = USART_UCSRC_Reg;
    if(strUCSRC.sUCSRC.m_UMSEL == USART_Synchronous){
        GPIO_u8SetPinDirection(USART_XCK_PORT, USART_XCK_PIN, PIN_OUTPUT);
    }
    GPIO_u8SetPinDirection(USART_PORT     , USART_TX_PIN , PIN_OUTPUT);
    GPIO_u8SetPinDirection(USART_XCK_PORT, USART_RX_PIN , PIN_INPUT );

    S_USART->m_UCSRB.sBits.m_UDRIE = strUSART_Config_GLB.m_Empty;
    S_USART->m_UCSRB.sBits.m_TXCIE = strUSART_Config_GLB.m_TXIE;
    S_USART->m_UCSRB.sBits.m_RXCIE = strUSART_Config_GLB.m_RXIE;

    S_USART->m_UCSRB.sBits.m_TXEN  = strUSART_Config_GLB.m_TXEN;
    S_USART->m_UCSRB.sBits.m_RXEN  = strUSART_Config_GLB.m_RXEN;
}
```

Here is the call graph for this function:



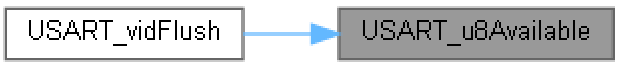
Here is the caller graph for this function:



u8 USART_u8Available (void)

```
                            {
    return S_USART->m_UCSRA.sBits.m_RXC;
}
```

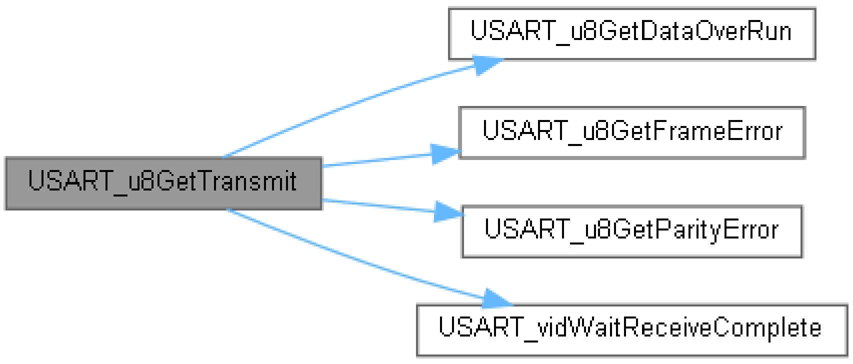
Here is the caller graph for this function:



LBTY_tenuErrorStatus USART_u8GetTransmit (void * *pvidTransmit*)

```
                                                                {
    LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;

    if(pvidTransmit == LBTY_NULL){
        u8RetErrorState = LBTY_NULL_POINTER;
    }else{
        USART_vidWaitReceiveComplete();
        if(USART_u8GetFrameError() || USART_u8GetDataOverRun() ||
USART_u8GetParityError()){
            if(strUSART_Config_GLB.m_Size == USART_9_bit){
                *((u16*)pvidTransmit) = LBTY_u16MAX;
            }else{
                *((u8*)pvidTransmit)  = LBTY_u8MAX;
            }
            u8RetErrorState = LBTY_NOK;
        }else{
            if(strUSART_Config_GLB.m_Size == USART_9_bit){
                *((u16*)pvidTransmit) = (u16)S_USART->m_UDR |
(u16)(S_USART->m_UCSRB.sBits.m_TXB8 << 8) ;
            }else{
                *((u8*)pvidTransmit)  = S_USART->m_UDR;
            }
        }
    }
    return u8RetErrorState;
}
```

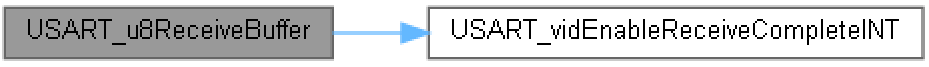
Here is the call graph for this function:



LBTY_tenuErrorStatus USART_u8ReceiveBuffer (u8 * *pu8Data*, u8 *u8Size*)

```
{
    LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;

    if(pu8Data == LBTY_NULL){
        u8RetErrorState = LBTY_NULL_POINTER;
    }else{
        if(strRX_GLB.m_u8Status == RX_IDLE){
            strRX_GLB.m_pu8Data  = pu8Data;
            strRX_GLB.m_u8Size   = u8Size;
            strRX_GLB.m_u8Idx    = LBTY_u8ZERO;
            strRX_GLB.m_u8Status = RX_BUSY;

            USART_vidEnableReceiveCompleteINT();
        }else{
            u8RetErrorState = LBTY_NOK;
        }
    }

    return u8RetErrorState;
}
```

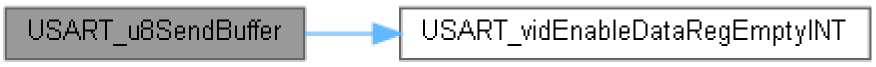
Here is the call graph for this function:



LBTY_tenuErrorStatus USART_u8SendBuffer (u8 * *pu8Data*, u8 *u8Size*)

```
{
    LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;

    if(pu8Data == LBTY_NULL){
        u8RetErrorState = LBTY_NULL_POINTER;
    }else{
        if(strTX_GLB.m_u8Status == TX_IDLE){
            strTX_GLB.m_pu8Data  = pu8Data;
            strTX_GLB.m_u8Size   = u8Size;
            strTX_GLB.m_u8Idx    = LBTY_u8ZERO;
            strTX_GLB.m_u8Status = TX_BUSY;

            if(S_USART->m_UCSRA.sBits.m_UDRE){
                S_USART->m_UDR = strTX_GLB.m_pu8Data[strTX_GLB.m_u8Idx++];
            }
            USART_vidEnableDataRegEmptyINT();
        }else{
            u8RetErrorState = LBTY_NOK;
        }
    }

    return u8RetErrorState;
}
```

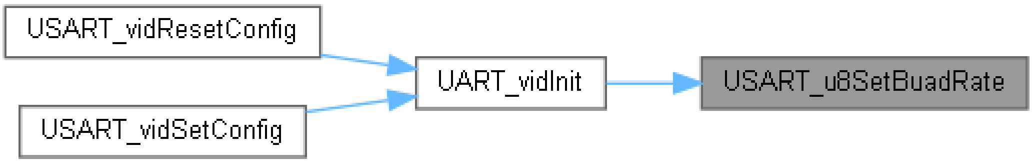
Here is the call graph for this function:



LBTY_tenuErrorStatus USART_u8SetBuadRate (USART_tenumBuadRate *u32BuadRate*)

```
                                                                             {
    LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
    u16 u16UBRR = LBTY_u16ZERO;

    switch(strUCSRC.sUCSRC.m_UMSEL){
        case USART_Asynchronous:
            switch(S_USART->m_UCSRA.sBits.m_U2X){
                case USART_Speed_x1:
                    u16UBRR = (u16)(F_CPU / (16.0f * (u32)u32BuadRate)) - 1;
                    break;
                case USART_Speed_x2:
                    u16UBRR = (u16)(F_CPU / (8.0f  * (u32)u32BuadRate)) - 1;
                    break;
                default:
                    u8RetErrorState = LBTY_NOK;
            }
            break;
        case USART_Synchronous:
            u16UBRR = (u16)(USART_OPERATION_FREQ / (2.0f * (u32)u32BuadRate)) -
1;
            break;
        default:
            u8RetErrorState = LBTY_NOK;
    }
    if(u8RetErrorState == LBTY_OK){
        strUSART_Config_GLB.m_BuadRate = u32BuadRate;

        strUCSRC.sUBRRH.m_URSEL = USART_UBRRH_Reg;
        strUCSRC.sUBRRH.m_UBRR = GET_NIB(u16UBRR, 8);

        S_USART->m_UCSRC = strUCSRC.u_Reg;
        S_USART->m_UBRRL = GET_BYTE(u16UBRR, 0);
    }
    return u8RetErrorState;
}
```

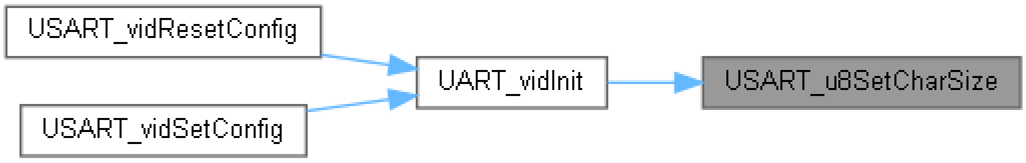
Here is the caller graph for this function:



LBTY_tenuErrorStatus USART_u8SetCharSize (USART_tenumCharSize *u8CharSize*)

```
                                                                            {
    LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;

    switch(u8CharSize){
        case USART_5_bit:
        case USART_6_bit:
        case USART_7_bit:
        case USART_8_bit:
        case USART_9_bit:
            strUCSRC.sUCSRC.m_URSEL = USART_UCSRC_Reg;
            strUCSRC.sUCSRC.m_UCSZ0 = GET_BIT(u8CharSize, USART_UCSZ0_BIT);
            strUCSRC.sUCSRC.m_UCSZ1 = GET_BIT(u8CharSize, USART_UCSZ1_BIT);
            S_USART->m_UCSRB.sBits.m_UCSZ2  = GET_BIT(u8CharSize,
USART_UCSZ2_BIT);

            S_USART->m_UCSRC = strUCSRC.u_Reg;
            strUSART_Config_GLB.m_Size = u8CharSize;
            break;
        default:
            u8RetErrorState = LBTY_NOK;
    }

    return u8RetErrorState;
}
```

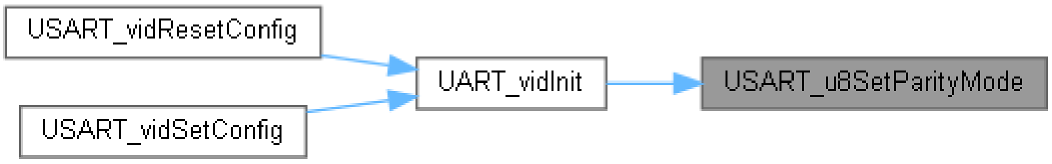
Here is the caller graph for this function:



LBTY_tenuErrorStatus USART_u8SetParityMode (USART_tenumParityMode *u8Parity*)

```
                                                                               {
    LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;

    switch(u8Parity){
        case USART_Parity_Disable:
        case USART_Parity_Even:
        case USART_Parity_Odd:
            strUCSRC.sUCSRC.m_URSEL = USART_UCSRC_Reg;
            strUCSRC.sUCSRC.m_UPM = strUSART_Config_GLB.m_Parity = u8Parity;
            S_USART->m_UCSRC = strUCSRC.u_Reg;
            break;
        default: u8RetErrorState = LBTY_NOK;
    }

    return u8RetErrorState;
}
```

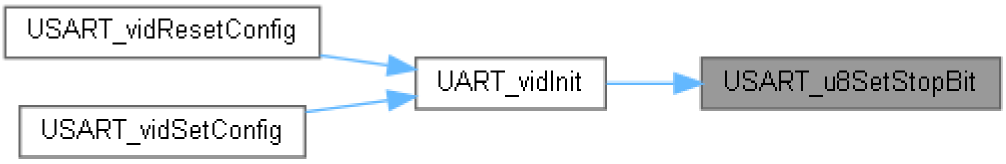
Here is the caller graph for this function:



LBTY_tenuErrorStatus USART_u8SetStopBit (USART_tenumStopBit *u8Stop*)

```
                                                                       {
    LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;

    switch(u8Stop){
        case USART_Stop_1_bit:
        case USART_Stop_2_bit:
            strUCSRC.sUCSRC.m_URSEL = USART_UCSRC_Reg;
            strUCSRC.sUCSRC.m_USBS  = strUSART_Config_GLB.m_Stop = u8Stop;
            S_USART->m_UCSRC = strUCSRC.u_Reg;
            break;
        default: u8RetErrorState = LBTY_NOK;
    }

    return u8RetErrorState;
}
```

Here is the caller graph for this function:

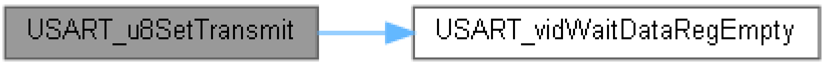


LBTY_tenuErrorStatus USART_u8SetTransmit (void * *pvidTransmit*)

```
                                                                  {
    LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;

    if(pvidTransmit == LBTY_NULL){
        u8RetErrorState = LBTY_NULL_POINTER;
    }else{
        USART_vidWaitDataRegEmpty();
        S_USART->m_UDR = *((u8*)pvidTransmit);
        if(strUSART_Config_GLB.m_Size == USART_9_bit){
            S_USART->m_UCSRB.sBits.m_TXB8 = GET_BIT(*((u16*)pvidTransmit), 8);
        }
    }
    return u8RetErrorState;
```

```
}
```

Here is the call graph for this function:



void USART_vidDisableDataRegEmptyINT (void)

```
{S_USART->m_UCSRB.sBits.m_UDRIE = LBTY_RESET;}
```

void USART_vidDisableReceiveCompleteINT (void)

```
{S_USART->m_UCSRB.sBits.m_RXCIE = LBTY_RESET;}
```

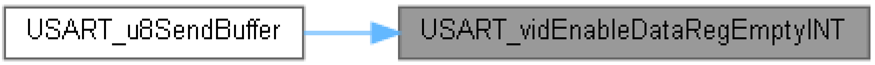
void USART_vidDisableTransmitCompleteINT (void)

```
{S_USART->m_UCSRB.sBits.m_TXCIE = LBTY_RESET;}
```

void USART_vidEnableDataRegEmptyINT (void)

```
{S_USART->m_UCSRB.sBits.m_UDRIE = LBTY_SET;}
```

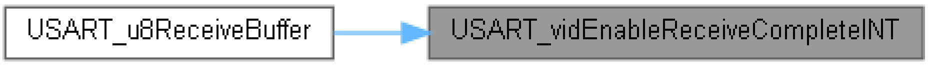
Here is the caller graph for this function:



void USART_vidEnableReceiveCompleteINT (void)

```
{S_USART->m_UCSRB.sBits.m_RXCIE = LBTY_SET;}
```

Here is the caller graph for this function:



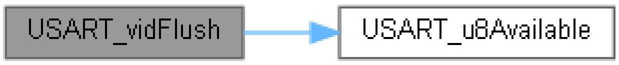
void USART_vidEnableTransmitCompleteINT (void)

```
{S_USART->m_UCSRB.sBits.m_TXCIE = LBTY_SET;}
```

void USART_vidFlush (void)

```
                             {
    u8 dummy;
    while(USART_u8Available()){
        dummy = S_USART->m_UDR;
    }
}
```

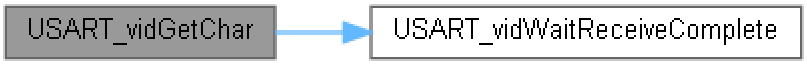
Here is the call graph for this function:



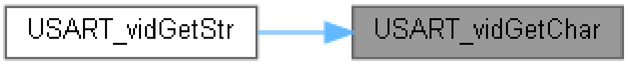
void USART_vidGetChar (u8 * *pu8Char*)

```
                                      {
    USART_vidWaitReceiveComplete();
    *pu8Char = S_USART->m_UDR;
}
```

Here is the call graph for this function:



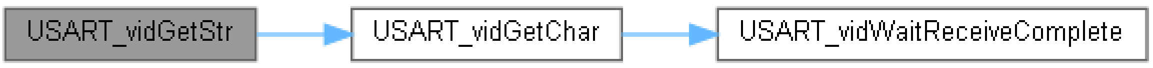
Here is the caller graph for this function:



void USART_vidGetStr (u8 * *pu8Receive*)

```
                                        {
    do{
        USART_vidGetChar(pu8Receive);
    }while(*pu8Receive++ != '\r');
    *pu8Receive = '\0';
}
```

Here is the call graph for this function:



void USART_vidReceiverDisable (void)

```
{
    S_USART->m_UCSRB.sBits.m_RXEN = strUSART_Config_GLB.m_RXEN = LBTY_RESET;
}
```

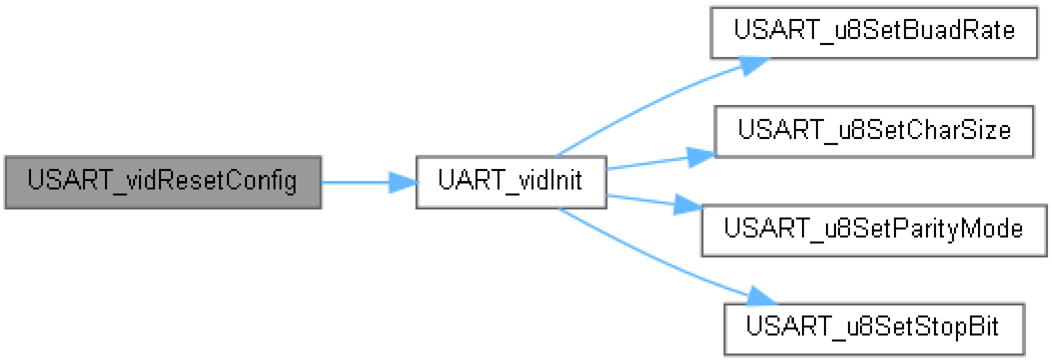
void USART_vidReceiverEnable (void)

```
{
    S_USART->m_UCSRB.sBits.m_RXEN = strUSART_Config_GLB.m_RXEN = LBTY_SET;
}
```

void USART_vidResetConfig (USART_tstrConfiguration *const *pstrConfig*)

```
{
   strUSART_Config_GLB.m_Mode       = USART_OPERATION_MODE;
   strUSART_Config_GLB.m_Polarity   = USART_OPERATION_POLARITY;
   strUSART_Config_GLB.m_Speed      = USART_OPERATION_SPEED;
   strUSART_Config_GLB.m_BuadRate   = USART_BUAD_RATE_INIT;
   strUSART_Config_GLB.m_Size       = USART_CHAR_SIZE_INIT;
   strUSART_Config_GLB.m_Parity     = USART_PARITY_BIT_INIT;
   strUSART_Config_GLB.m_Stop       = USART_STOP_BIT_INIT;
   strUSART_Config_GLB.m_TXEN       = USART_TRANSMIT_INIT;
   strUSART_Config_GLB.m_RXEN       = USART_RECEIVE_INIT;
   strUSART_Config_GLB.m_TXIE       = USART_TRANSMIT_COMPLETE_INT;
   strUSART_Config_GLB.m_RXIE       = USART_RECEIVE_COMPLETE_INT;
   strUSART_Config_GLB.m_Empty      = USART_DATA_REG_EMPTY_INT;

    if(pstrConfig != LBTY_NULL){
        *pstrConfig = strUSART_Config_GLB;
    }
    UART_vidInit();
}
```

Here is the call graph for this function:



void USART_vidSetCallBack_Empty (void(*)(void) *pCallBack*)

```
{
    pvidfunc_Empty_CallBak = pCallBack;
}
```

void USART_vidSetCallBack_RX (void(*)(void) *pCallBack*)

```
{
    pvidfunc_Rx_CallBak = pCallBack;
}
```

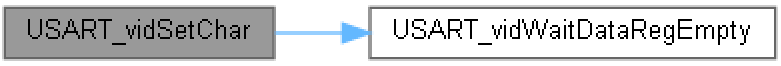
void USART_vidSetCallBack_TX (void(*)(void) *pCallBack*)

```
{
    pvidfunc_Tx_CallBak = pCallBack;
}
```

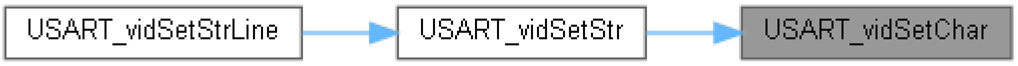
void USART_vidSetChar (u8 *u8Char*)

```
{
    USART_vidWaitDataRegEmpty();
    S_USART->m_UDR = u8Char;
}
```

Here is the call graph for this function:



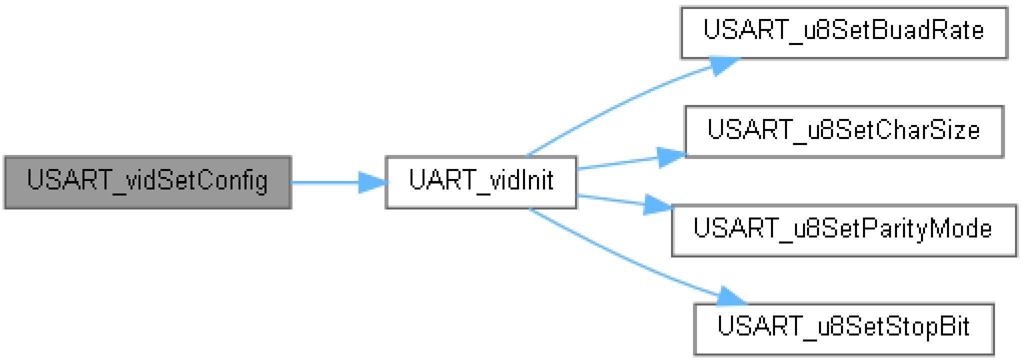
Here is the caller graph for this function:



void USART_vidSetConfig (USART_tstrConfiguration const *const *pstrConfig*)

```
                                                                  {
    if(pstrConfig != LBTY_NULL){
        strUSART_Config_GLB = *pstrConfig;
    }
    UART_vidInit();
}
```

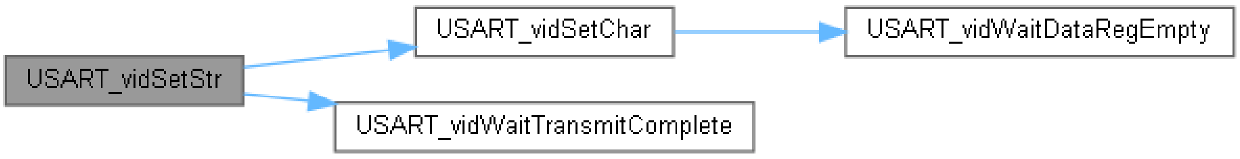
Here is the call graph for this function:



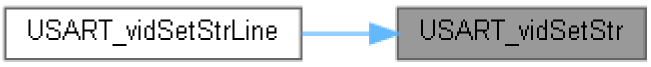
void USART_vidSetStr (u8 * *pu8Transmit*)

```
                                          {
    while(*pu8Transmit){
        USART_vidSetChar(*(pu8Transmit++));
        USART_vidWaitTransmitComplete();
    }
}
```

Here is the call graph for this function:



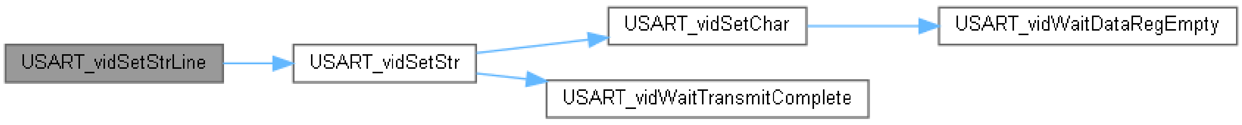
Here is the caller graph for this function:



void USART_vidSetStrLine (u8 * *pu8Transmit*)

```
                                              {
    USART_vidSetStr((u8*)pu8Transmit);
    USART_vidSetStr((u8*)"\r\n");
}
```

Here is the call graph for this function:



void USART_vidTransmitterDisable (void)

```
                                           {
    S_USART->m_UCSRB.sBits.m_TXEN = strUSART_Config_GLB.m_TXEN = LBTY_RESET;
}
```

void USART_vidTransmitterEnable (void)

```
                                        {
    S_USART->m_UCSRB.sBits.m_TXEN = strUSART_Config_GLB.m_TXEN = LBTY_SET;
}
```

USART_int.h

Go to the documentation of this file.

```c
/*
*************************************************************************** */
/* ********************** FILE DEFINITION SECTION ************************** */
/* ************************************************************************* */
/* File Name    : USART_int.h                                                */
/* Author       : MAAM                                                       */
/* Version      : v01.2                                                      */
/* date         : Apr 10, 2023                                               */
/* ************************************************************************* */
/* ************************ HEADER FILES INCLUDES ************************  */
/* ************************************************************************* */

#ifndef USART_INT_H_
#define USART_INT_H_

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

typedef enum{
    USART_Asynchronous = (u8)0u,
    USART_Synchronous,
}USART_tenumModeSelect;

typedef enum{
    USART_Speed_x1 = (u8)0u,
    USART_Speed_x2,
}USART_tenumSpeed;

typedef enum{
    USART_Transmit_Rising_Receive_Falling = (u8)0u,
    USART_Transmit_Falling_Receive_Rising,
}USART_tenumClockPolarity;

typedef enum{
    USART_5_bit = (u8)0u,
    USART_6_bit,
    USART_7_bit,
    USART_8_bit,
    USART_9_bit = (u8)7u,
}USART_tenumCharSize;

typedef enum{
    USART_Parity_Disable = (u8)0u,
    USART_Parity_Reserved,
    USART_Parity_Even,
    USART_Parity_Odd,
}USART_tenumParityMode;

typedef enum{
    USART_Stop_1_bit = (u8)0u,
    USART_Stop_2_bit,
}USART_tenumStopBit;

typedef enum{
    USART_BuadRate_1200     = 1200u,
    USART_BuadRate_2400     = 2400u,
    USART_BuadRate_4800     = 4800u,
    USART_BuadRate_9600     = 9600u,
    USART_BuadRate_14400    = 14400u,
    USART_BuadRate_19200    = 19200u,
    USART_BuadRate_28800    = 28800u,
    USART_BuadRate_38400    = 38400u,
    USART_BuadRate_57600    = 57600u,
    USART_BuadRate_76600    = 76600u,
    USART_BuadRate_115200   = 115200u,
    USART_BuadRate_230400   = 230400u,
    USART_BuadRate_250000   = 250000u,
    USART_BuadRate_500000   = 500000u,
    USART_BuadRate_1000000  = 1000000u,
}USART_tenumBuadRate;

```

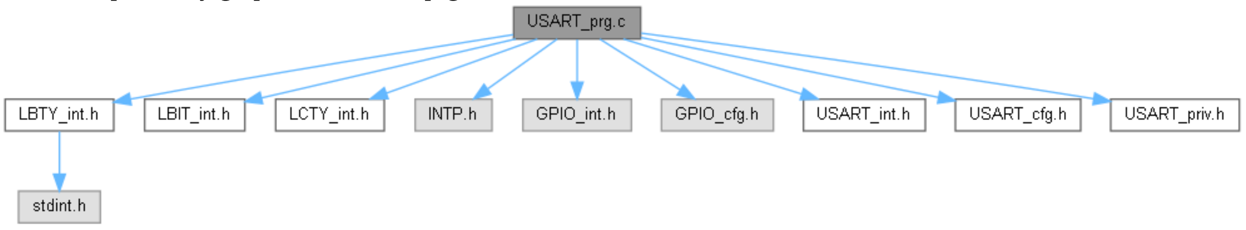
```
/***************************************************************************************
********************************/

typedef struct{
/* HW Config */
    USART_tenumModeSelect    m_Mode;
    USART_tenumClockPolarity m_Polarity;
    USART_tenumSpeed         m_Speed;
/* frame */
    USART_tenumBuadRate      m_BuadRate;
    USART_tenumCharSize      m_Size;
    USART_tenumParityMode    m_Parity;
    USART_tenumStopBit       m_Stop;
/* TX/RX Status */
    LBTY_tenuFlagStatus      m_TXEN;
    LBTY_tenuFlagStatus      m_RXEN;
/* TX/RX Interrupt */
    LBTY_tenuFlagStatus      m_TXIE;
    LBTY_tenuFlagStatus      m_RXIE;
    LBTY_tenuFlagStatus      m_Empty;
}USART_tstrConfiguration;

/* **************************************************************************** */
/* *************************** MACRO/DEFINE SECTION *************************** */
/* **************************************************************************** */

//#define USART_vidSetChar      USART_vidSetTransmit
//#define USART_vidGetChar      USART_vidGetTransmit

/* **************************************************************************** */
/* ****************************** CONST SECTION ******************************* */
/* **************************************************************************** */

/* **************************************************************************** */
/* ***************************** VARIABLE SECTION ***************************** */
/* **************************************************************************** */

/* **************************************************************************** */
/* ***************************** FUNCTION SECTION ***************************** */
/* **************************************************************************** */

extern void USART_vidSetConfig(USART_tstrConfiguration const* const pstrConfig);
extern void USART_vidResetConfig(USART_tstrConfiguration* const pstrConfig);

extern void UART_vidInit(void);

extern void USART_vidTransmitterEnable(void);
extern void USART_vidTransmitterDisable(void);
extern void USART_vidReceiverEnable(void);
extern void USART_vidReceiverDisable(void);

extern LBTY_tenuErrorStatus USART_u8SetBuadRate(USART_tenumBuadRate u32BuadRate);
extern LBTY_tenuErrorStatus USART_u8SetCharSize(USART_tenumCharSize u8CharSize);
extern LBTY_tenuErrorStatus USART_u8SetParityMode(USART_tenumParityMode u8Parity);
extern LBTY_tenuErrorStatus USART_u8SetStopBit(USART_tenumStopBit u8Stop);


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

extern u8 USART_u8Available(void);
extern void USART_vidFlush(void);

extern LBTY_tenuErrorStatus USART_u8SetTransmit(void* pvidTransmit);
extern LBTY_tenuErrorStatus USART_u8GetTransmit(void* pvidTransmit);

extern void USART_vidSetChar(u8 u8Char);
extern void USART_vidGetChar(u8* pu8Char);

extern void USART_vidSetStrLine(u8* pu8Transmit);
extern void USART_vidSetStr(u8* pu8Transmit);
extern void USART_vidGetStr(u8* pu8Receive);

extern LBTY_tenuErrorStatus USART_u8SendBuffer(u8* pu8Data, u8 u8Size);
extern LBTY_tenuErrorStatus USART_u8ReceiveBuffer(u8* pu8Data, u8 u8Size);

```

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

extern void USART_vidEnableReceiveCompleteINT(void);
extern void USART_vidEnableTransmitCompleteINT(void);
extern void USART_vidEnableDataRegEmptyINT(void);

extern void USART_vidDisableReceiveCompleteINT(void);
extern void USART_vidDisableTransmitCompleteINT(void);
extern void USART_vidDisableDataRegEmptyINT(void);

extern void USART_vidSetCallBack_Empty(void (*pCallBack)(void));
extern void USART_vidSetCallBack_TX   (void (*pCallBack)(void));
extern void USART_vidSetCallBack_RX   (void (*pCallBack)(void));

#endif /* USART_INT_H_ */
/*************************** E N D (USART_int.h) ******************************/
```

USART_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 "USART_int.h"
#include "USART_cfg.h"
#include "USART_priv.h"
```

Include dependency graph for USART_prg.c:



Functions

- static void _voidCallBack (void)
- void USART_vidSetConfig (USART_tstrConfiguration const *const pstrConfig)
- void USART_vidResetConfig (USART_tstrConfiguration *const pstrConfig)
- void UART_vidInit (void)
- void USART_vidTransmitterEnable (void)
- void USART_vidTransmitterDisable (void)
- void USART_vidReceiverEnable (void)
- void USART_vidReceiverDisable (void)
- LBTY_tenuErrorStatus USART_u8SetBuadRate (USART_tenumBuadRate u32BuadRate)
- LBTY_tenuErrorStatus USART_u8SetCharSize (USART_tenumCharSize u8CharSize)
- LBTY_tenuErrorStatus USART_u8SetParityMode (USART_tenumParityMode u8Parity)
- LBTY_tenuErrorStatus USART_u8SetStopBit (USART_tenumStopBit u8Stop)
- LCTY_INLINE void USART_vidWaitDataRegEmpty (void)
- LCTY_INLINE void USART_vidWaitTransmitComplete (void)
- LCTY_INLINE void USART_vidWaitReceiveComplete (void)
- LCTY_INLINE u8 USART_u8GetFrameError (void)
- LCTY_INLINE u8 USART_u8GetDataOverRun (void)
- LCTY_INLINE u8 USART_u8GetParityError (void)
- u8 USART_u8Available (void)
- void USART_vidFlush (void)
- LBTY_tenuErrorStatus USART_u8SetTransmit (void *pvidTransmit)
- LBTY_tenuErrorStatus USART_u8GetTransmit (void *pvidTransmit)
- void USART_vidSetChar (u8 u8Char)
- void USART_vidGetChar (u8 *pu8Char)
- void USART_vidSetStrLine (u8 *pu8Transmit)
- void USART_vidSetStr (u8 *pu8Transmit)
- void USART_vidGetStr (u8 *pu8Receive)
- LBTY_tenuErrorStatus USART_u8SendBuffer (u8 *pu8Data, u8 u8Size)
- LBTY_tenuErrorStatus USART_u8ReceiveBuffer (u8 *pu8Data, u8 u8Size)
- void USART_vidEnableReceiveCompleteINT (void)
- void USART_vidEnableTransmitCompleteINT (void)
- void USART_vidEnableDataRegEmptyINT (void)
- void USART_vidDisableReceiveCompleteINT (void)

- void USART_vidDisableTransmitCompleteINT (void)
- void USART_vidDisableDataRegEmptyINT (void)
- void USART_vidSetCallBack_Empty (void(*pCallBack)(void))
- void USART_vidSetCallBack_TX (void(*pCallBack)(void))
- void USART_vidSetCallBack_RX (void(*pCallBack)(void))
- ISR (USART_RXC_vect)
- ISR (USART_UDRE_vect)
- ISR (USART_TXC_vect)

Variables

- static volatile UCSRC_type strUCSRC
- static void(* pvidfunc_Empty_CallBak)(void) = _voidCallBack
- static void(* pvidfunc_Tx_CallBak)(void) = _voidCallBack
- static void(* pvidfunc_Rx_CallBak)(void) = _voidCallBack
- static USART_tstrBuffer strTX_GLB
- static USART_tstrBuffer strRX_GLB
- static volatile USART_tstrConfiguration strUSART_Config_GLB

Function Documentation

static void _voidCallBack (void) [static]

```
{__asm("NOP");}
```

ISR (USART_RXC_vect)

```
                  {
    if((S_USART->m_UCSRA.sBits.m_RXC) && (strRX_GLB.m_u8Idx < strRX_GLB.m_u8Size) && (strRX_GLB.m_u8Status == RX_BUSY)){
        strRX_GLB.m_pu8Data[strRX_GLB.m_u8Idx++] = S_USART->m_UDR;
    }else{
        strRX_GLB.m_u8Status = RX_IDLE;
        //USART_vidDisableReceiveCompleteINT();
        pvidfunc_Rx_CallBak();
    }
}
```

ISR (USART_TXC_vect)

```
                  {
    if(strTX_GLB.m_u8Status == TX_IDLE){
        strTX_GLB.m_u8Status = TX_IDLE;
        //USART_vidDisableTransmitCompleteINT();
        pvidfunc_Tx_CallBak();
    }
}
```

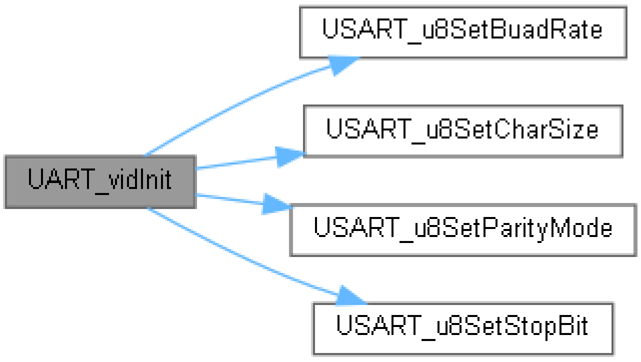
ISR (USART_UDRE_vect)

```
                   {
    if((S_USART->m_UCSRA.sBits.m_UDRE) && (strTX_GLB.m_u8Idx < strTX_GLB.m_u8Size) && (strTX_GLB.m_u8Status == TX_BUSY)){
        S_USART->m_UDR = strTX_GLB.m_pu8Data[strTX_GLB.m_u8Idx++];
    }else{
        strTX_GLB.m_u8Status = TX_IDLE;
        //USART_vidDisableDataRegEmptyINT();
        pvidfunc_Empty_CallBak();
    }
}
```

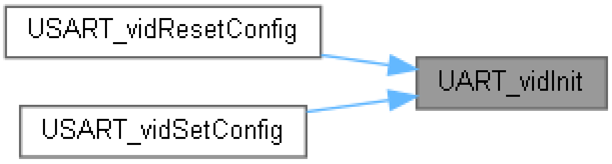
void UART_vidInit (void)

```
                       {

    strUCSRC.sUCSRC.m_URSEL = USART_UCSRC_Reg;
    strUCSRC.sUCSRC.m_UMSEL = strUSART_Config_GLB.m_Mode;
    strUCSRC.sUCSRC.m_UCPOL = strUSART_Config_GLB.m_Polarity;
    S_USART->m_UCSRC = strUCSRC.u_Reg;

    S_USART->m_UCSRA.sBits.m_MPCM = USART_OPERATION_MULTI_PROCESSOR;
    S_USART->m_UCSRA.sBits.m_U2X  = strUSART_Config_GLB.m_Speed;

    USART_u8SetBuadRate   (strUSART_Config_GLB.m_BuadRate);
    USART_u8SetCharSize   (strUSART_Config_GLB.m_Size);
    USART_u8SetParityMode (strUSART_Config_GLB.m_Parity);
    USART_u8SetStopBit    (strUSART_Config_GLB.m_Stop);

    strUCSRC.sUCSRC.m_URSEL = USART_UCSRC_Reg;
    if(strUCSRC.sUCSRC.m_UMSEL == USART_Synchronous){
        GPIO_u8SetPinDirection(USART_XCK_PORT, USART_XCK_PIN, PIN_OUTPUT);
    }
    GPIO_u8SetPinDirection(USART_PORT    , USART_TX_PIN , PIN_OUTPUT);
    GPIO_u8SetPinDirection(USART_XCK_PORT, USART_RX_PIN , PIN_INPUT );

    S_USART->m_UCSRB.sBits.m_UDRIE = strUSART_Config_GLB.m_Empty;
    S_USART->m_UCSRB.sBits.m_TXCIE = strUSART_Config_GLB.m_TXIE;
    S_USART->m_UCSRB.sBits.m_RXCIE = strUSART_Config_GLB.m_RXIE;

    S_USART->m_UCSRB.sBits.m_TXEN  = strUSART_Config_GLB.m_TXEN;
    S_USART->m_UCSRB.sBits.m_RXEN  = strUSART_Config_GLB.m_RXEN;
}
```

Here is the call graph for this function:



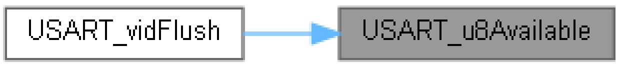
Here is the caller graph for this function:



u8 USART_u8Available (void)

```
                {
    return S_USART->m_UCSRA.sBits.m_RXC;
}
```

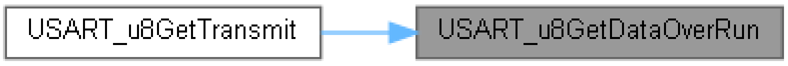
Here is the caller graph for this function:



LCTY_INLINE u8 USART_u8GetDataOverRun (void)

```
{return S_USART->m_UCSRA.sBits.m_DOR;}
```

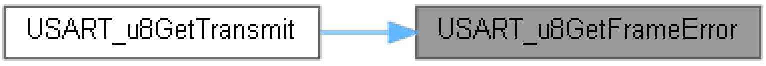
Here is the caller graph for this function:



LCTY_INLINE u8 USART_u8GetFrameError (void)

```
{return S_USART->m_UCSRA.sBits.m_FE;}
```

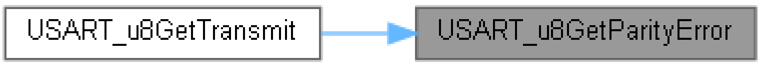
Here is the caller graph for this function:



LCTY_INLINE u8 USART_u8GetParityError (void)

```
{return S_USART->m_UCSRA.sBits.m_PE;}
```

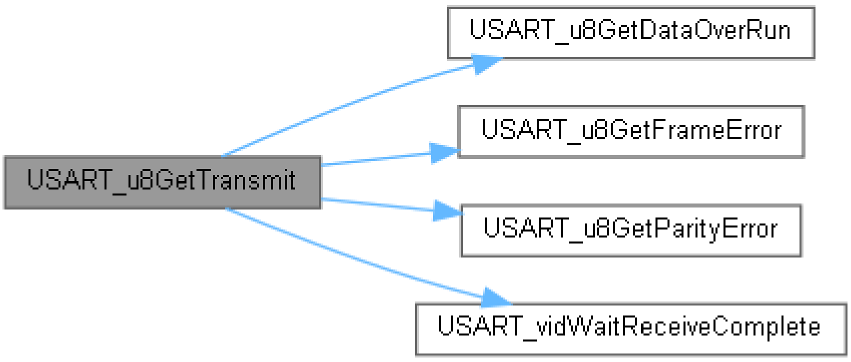
Here is the caller graph for this function:



LBTY_tenuErrorStatus USART_u8GetTransmit (void * *pvidTransmit*)

```
                                                                    {
        LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;

        if(pvidTransmit == LBTY_NULL){
            u8RetErrorState = LBTY_NULL_POINTER;
        }else{
            USART_vidWaitReceiveComplete();
            if(USART_u8GetFrameError() || USART_u8GetDataOverRun() ||
USART_u8GetParityError()){
                if(strUSART_Config_GLB.m_Size == USART_9_bit){
                    *((u16*)pvidTransmit) = LBTY_u16MAX;
                }else{
                    *((u8*)pvidTransmit)  = LBTY_u8MAX;
                }
                u8RetErrorState = LBTY_NOK;
            }else{
                if(strUSART_Config_GLB.m_Size == USART_9_bit){
                    *((u16*)pvidTransmit) = (u16)S_USART->m_UDR |
(u16)(S_USART->m_UCSRB.sBits.m_TXB8 << 8) ;
                }else{
                    *((u8*)pvidTransmit)  = S_USART->m_UDR;
                }
            }
        }
        return u8RetErrorState;
}
```

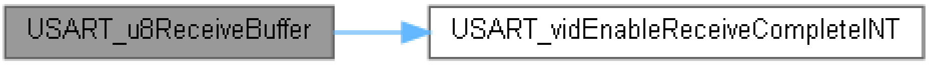
Here is the call graph for this function:



LBTY_tenuErrorStatus USART_u8ReceiveBuffer (u8 * *pu8Data*, u8 *u8Size*)

```
                                                                          {
        LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;

        if(pu8Data == LBTY_NULL){
            u8RetErrorState = LBTY_NULL_POINTER;
        }else{
            if(strRX_GLB.m_u8Status == RX_IDLE){
                strRX_GLB.m_pu8Data   = pu8Data;
                strRX_GLB.m_u8Size    = u8Size;
                strRX_GLB.m_u8Idx     = LBTY_u8ZERO;
                strRX_GLB.m_u8Status = RX_BUSY;

                USART_vidEnableReceiveCompleteINT();
            }else{
                u8RetErrorState = LBTY_NOK;
            }
        }

        return u8RetErrorState;
}
```

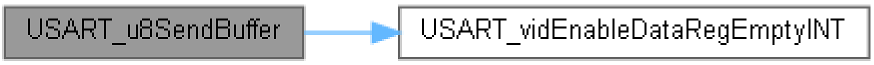
Here is the call graph for this function:



LBTY_tenuErrorStatus USART_u8SendBuffer (u8 *   *pu8Data*, u8   *u8Size*)

```
                                                                {
    LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;

    if(pu8Data == LBTY_NULL){
        u8RetErrorState = LBTY_NULL_POINTER;
    }else{
        if(strTX_GLB.m_u8Status == TX_IDLE){
            strTX_GLB.m_pu8Data  = pu8Data;
            strTX_GLB.m_u8Size   = u8Size;
            strTX_GLB.m_u8Idx    = LBTY_u8ZERO;
            strTX_GLB.m_u8Status = TX_BUSY;

            if(S_USART->m_UCSRA.sBits.m_UDRE){
                S_USART->m_UDR = strTX_GLB.m_pu8Data[strTX_GLB.m_u8Idx++];
            }
            USART_vidEnableDataRegEmptyINT();
        }else{
            u8RetErrorState = LBTY_NOK;
        }
    }

    return u8RetErrorState;
}
```

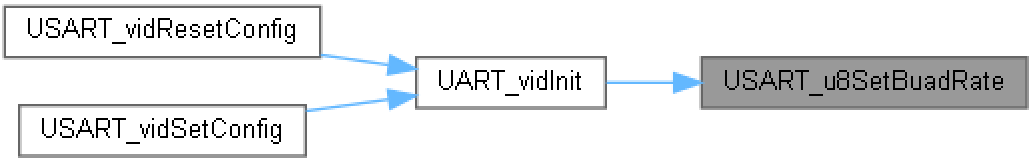
Here is the call graph for this function:



LBTY_tenuErrorStatus USART_u8SetBuadRate (USART_tenumBuadRate *u32BuadRate*)

```
                                                                          {
    LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
    u16 u16UBRR = LBTY_u16ZERO;

    switch(strUCSRC.sUCSRC.m_UMSEL){
        case USART_Asynchronous:
            switch(S_USART->m_UCSRA.sBits.m_U2X){
                case USART_Speed_x1:
                    u16UBRR = (u16)(F_CPU / (16.0f * (u32)u32BuadRate)) - 1;
                    break;
                case USART_Speed_x2:
                    u16UBRR = (u16)(F_CPU / (8.0f  * (u32)u32BuadRate)) - 1;
                    break;
                default:
                    u8RetErrorState = LBTY_NOK;
            }
            break;
        case USART_Synchronous:
            u16UBRR = (u16)(USART_OPERATION_FREQ / (2.0f * (u32)u32BuadRate)) -
1;
            break;
        default:
            u8RetErrorState = LBTY_NOK;
    }
    if(u8RetErrorState == LBTY_OK){
        strUSART_Config_GLB.m_BuadRate = u32BuadRate;

        strUCSRC.sUBRRH.m_URSEL = USART_UBRRH_Reg;
        strUCSRC.sUBRRH.m_UBRR = GET_NIB(u16UBRR, 8);

        S_USART->m_UCSRC = strUCSRC.u_Reg;
        S_USART->m_UBRRL = GET_BYTE(u16UBRR, 0);
    }
    return u8RetErrorState;
}
```

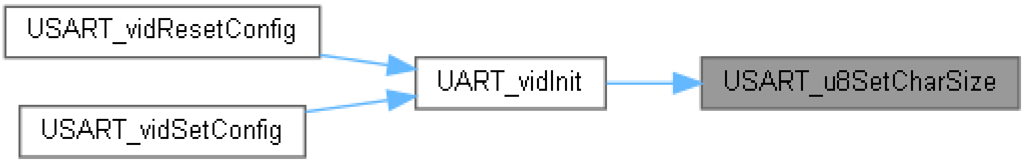
Here is the caller graph for this function:



LBTY_tenuErrorStatus USART_u8SetCharSize (USART_tenumCharSize *u8CharSize*)

```
                                                                               {
        LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;

        switch(u8CharSize){
            case USART_5_bit:
            case USART_6_bit:
            case USART_7_bit:
            case USART_8_bit:
            case USART_9_bit:
                strUCSRC.sUCSRC.m_URSEL = USART_UCSRC_Reg;
                strUCSRC.sUCSRC.m_UCSZ0 = GET_BIT(u8CharSize, USART_UCSZ0_BIT);
                strUCSRC.sUCSRC.m_UCSZ1 = GET_BIT(u8CharSize, USART_UCSZ1_BIT);
                S_USART->m_UCSRB.sBits.m_UCSZ2  = GET_BIT(u8CharSize,
USART_UCSZ2_BIT);

                S_USART->m_UCSRC = strUCSRC.u_Reg;
                strUSART_Config_GLB.m_Size = u8CharSize;
                break;
            default:
                u8RetErrorState = LBTY_NOK;
        }

        return u8RetErrorState;
}
```

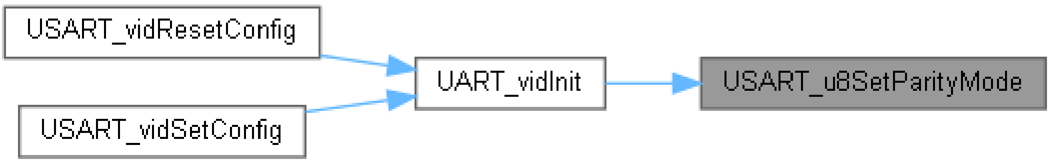
Here is the caller graph for this function:



LBTY_tenuErrorStatus USART_u8SetParityMode (USART_tenumParityMode *u8Parity*)

```
                                                                                 {
        LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;

        switch(u8Parity){
            case USART_Parity_Disable:
            case USART_Parity_Even:
            case USART_Parity_Odd:
                strUCSRC.sUCSRC.m_URSEL = USART_UCSRC_Reg;
                strUCSRC.sUCSRC.m_UPM = strUSART_Config_GLB.m_Parity = u8Parity;
                S_USART->m_UCSRC = strUCSRC.u_Reg;
                break;
            default: u8RetErrorState = LBTY_NOK;
        }

        return u8RetErrorState;
}
```

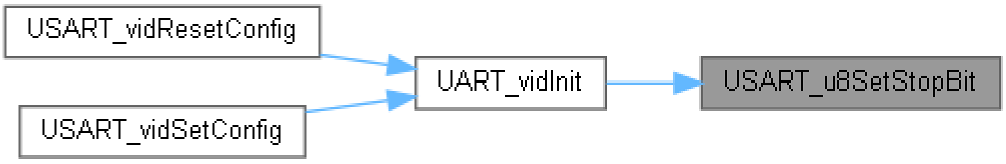
Here is the caller graph for this function:



LBTY_tenuErrorStatus USART_u8SetStopBit (USART_tenumStopBit *u8Stop*)

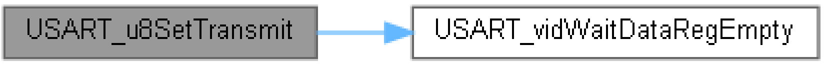
```
                                                                         {
        LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;

        switch(u8Stop){
```

```
        case USART_Stop_1_bit:
        case USART_Stop_2_bit:
            strUCSRC.sUCSRC.m_URSEL = USART_UCSRC_Reg;
            strUCSRC.sUCSRC.m_USBS  = strUSART_Config_GLB.m_Stop = u8Stop;
            S_USART->m_UCSRC = strUCSRC.u_Reg;
            break;
        default: u8RetErrorState = LBTY_NOK;
    }

    return u8RetErrorState;
}
```
Here is the caller graph for this function:



LBTY_tenuErrorStatus USART_u8SetTransmit (void * *pvidTransmit*)

```
                                                                {
    LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;

    if(pvidTransmit == LBTY_NULL){
        u8RetErrorState = LBTY_NULL_POINTER;
    }else{
        USART_vidWaitDataRegEmpty();
        S_USART->m_UDR = *((u8*)pvidTransmit);
        if(strUSART_Config_GLB.m_Size == USART_9_bit){
            S_USART->m_UCSRB.sBits.m_TXB8 = GET_BIT(*((u16*)pvidTransmit), 8);
        }
    }
    return u8RetErrorState;
}
```
Here is the call graph for this function:



void USART_vidDisableDataRegEmptyINT (void)

```
{S_USART->m_UCSRB.sBits.m_UDRIE = LBTY_RESET;}
```

void USART_vidDisableReceiveCompleteINT (void)

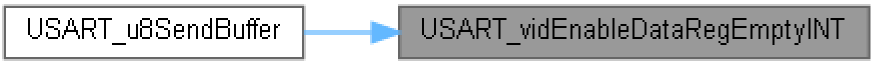
```
{S_USART->m_UCSRB.sBits.m_RXCIE = LBTY_RESET;}
```

void USART_vidDisableTransmitCompleteINT (void)

```
{S_USART->m_UCSRB.sBits.m_TXCIE = LBTY_RESET;}
```

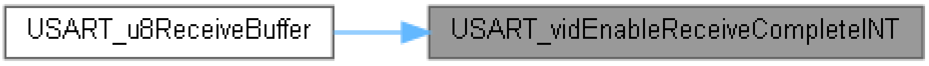
void USART_vidEnableDataRegEmptyINT (void)

```
{S_USART->m_UCSRB.sBits.m_UDRIE = LBTY_SET;}
```
Here is the caller graph for this function:



void USART_vidEnableReceiveCompleteINT (void)

```
{S_USART->m_UCSRB.sBits.m_RXCIE = LBTY_SET;}
```
Here is the caller graph for this function:



void USART_vidEnableTransmitCompleteINT (void)

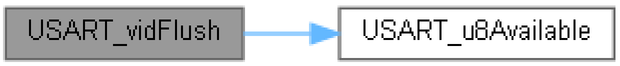
```
{S_USART->m_UCSRB.sBits.m_TXCIE = LBTY_SET;}
```

void USART_vidFlush (void)

```
                     {
```

```
    u8 dummy;
    while(USART_u8Available()){
        dummy = S_USART->m_UDR;
    }
}
```

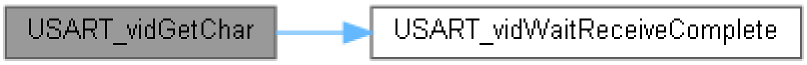
Here is the call graph for this function:



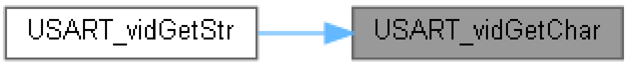
void USART_vidGetChar (u8 * *pu8Char*)

```
                                  {
    USART_vidWaitReceiveComplete();
    *pu8Char = S_USART->m_UDR;
}
```

Here is the call graph for this function:



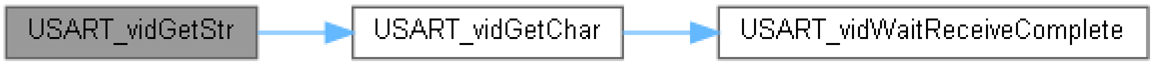
Here is the caller graph for this function:



void USART_vidGetStr (u8 * *pu8Receive*)

```
                                    {
    do{
        USART_vidGetChar(pu8Receive);
    }while(*pu8Receive++ != '\r');
    *pu8Receive = '\0';
}
```

Here is the call graph for this function:



void USART_vidReceiverDisable (void)

```
                                     {
    S_USART->m_UCSRB.sBits.m_RXEN = strUSART_Config_GLB.m_RXEN = LBTY_RESET;
}
```

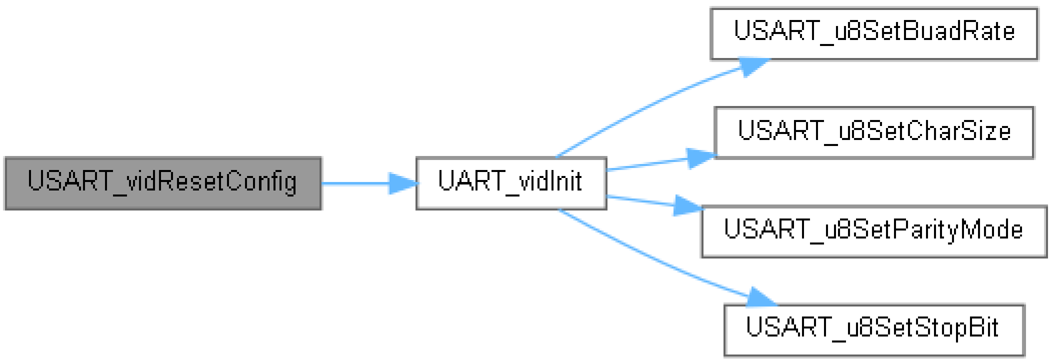
void USART_vidReceiverEnable (void)

```
                                    {
    S_USART->m_UCSRB.sBits.m_RXEN = strUSART_Config_GLB.m_RXEN = LBTY_SET;
}
```

void USART_vidResetConfig (USART_tstrConfiguration *const *pstrConfig*)

```
                                                                   {
   strUSART_Config_GLB.m_Mode       = USART_OPERATION_MODE;
   strUSART_Config_GLB.m_Polarity   = USART_OPERATION_POLARITY;
   strUSART_Config_GLB.m_Speed      = USART_OPERATION_SPEED;
   strUSART_Config_GLB.m_BuadRate   = USART_BUAD_RATE_INIT;
   strUSART_Config_GLB.m_Size       = USART_CHAR_SIZE_INIT;
   strUSART_Config_GLB.m_Parity     = USART_PARITY_BIT_INIT;
   strUSART_Config_GLB.m_Stop       = USART_STOP_BIT_INIT;
   strUSART_Config_GLB.m_TXEN       = USART_TRANSMIT_INIT;
   strUSART_Config_GLB.m_RXEN       = USART_RECEIVE_INIT;
   strUSART_Config_GLB.m_TXIE       = USART_TRANSMIT_COMPLETE_INT;
   strUSART_Config_GLB.m_RXIE       = USART_RECEIVE_COMPLETE_INT;
   strUSART_Config_GLB.m_Empty      = USART_DATA_REG_EMPTY_INT;

    if(pstrConfig != LBTY_NULL){
        *pstrConfig = strUSART_Config_GLB;
    }
    UART_vidInit();
}
```

Here is the call graph for this function:



void USART_vidSetCallBack_Empty (void(*)(void) *pCallBack*)

```
{
    pvidfunc_Empty_CallBak = pCallBack;
}
```

void USART_vidSetCallBack_RX (void(*)(void) *pCallBack*)

```
{
    pvidfunc_Rx_CallBak = pCallBack;
}
```

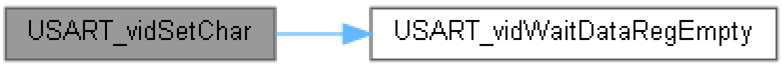
void USART_vidSetCallBack_TX (void(*)(void) *pCallBack*)

```
{
    pvidfunc_Tx_CallBak = pCallBack;
}
```

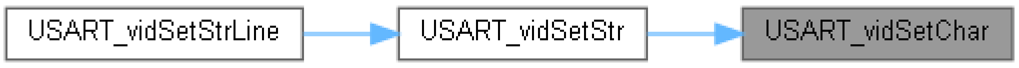
void USART_vidSetChar (u8 *u8Char*)

```
{
    USART_vidWaitDataRegEmpty();
    S_USART->m_UDR = u8Char;
}
```

Here is the call graph for this function:



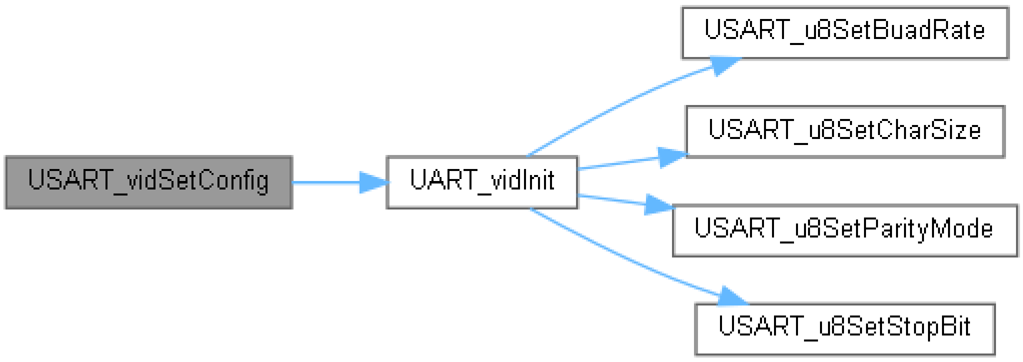
Here is the caller graph for this function:



void USART_vidSetConfig (USART_tstrConfiguration const *const *pstrConfig*)

```
{
    if(pstrConfig != LBTY_NULL){
        strUSART_Config_GLB = *pstrConfig;
    }
    UART_vidInit();
}
```

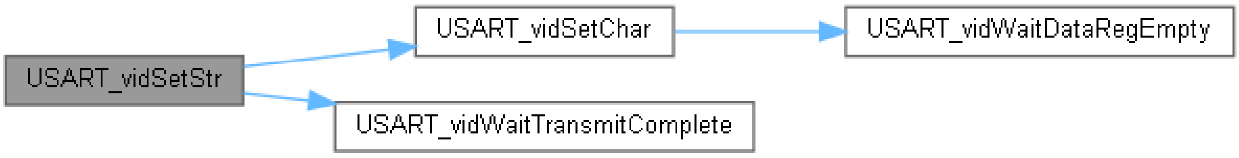
Here is the call graph for this function:



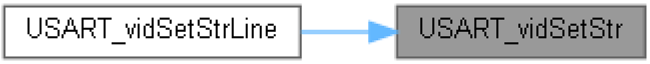
void USART_vidSetStr (u8 * *pu8Transmit*)

```
{
    while(*pu8Transmit){
        USART_vidSetChar(*(pu8Transmit++));
        USART_vidWaitTransmitComplete();
    }
}
```

Here is the call graph for this function:



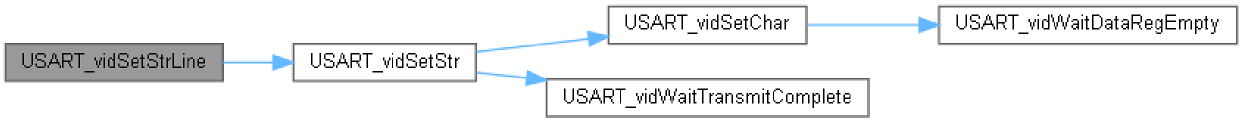
Here is the caller graph for this function:



void USART_vidSetStrLine (u8 * *pu8Transmit*)

```
{
    USART_vidSetStr((u8*)pu8Transmit);
    USART_vidSetStr((u8*)"\r\n");
}
```

Here is the call graph for this function:



void USART_vidTransmitterDisable (void)

```
{
    S_USART->m_UCSRB.sBits.m_TXEN = strUSART_Config_GLB.m_TXEN = LBTY_RESET;
}
```

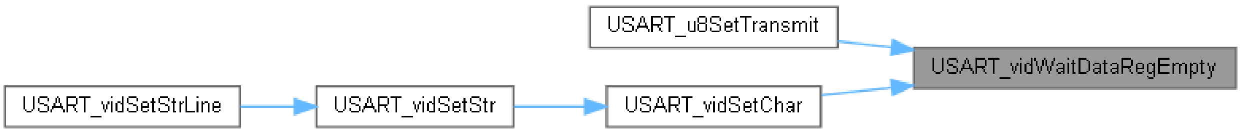
void USART_vidTransmitterEnable (void)

```
{
    S_USART->m_UCSRB.sBits.m_TXEN = strUSART_Config_GLB.m_TXEN = LBTY_SET;
}
```

LCTY_INLINE void USART_vidWaitDataRegEmpty (void)

```
{while(!(S_USART->m_UCSRA.sBits.m_UDRE));}
```

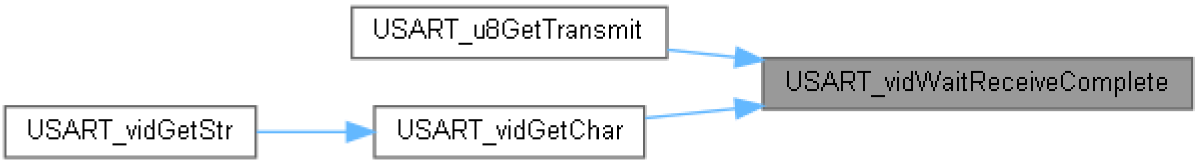
Here is the caller graph for this function:



LCTY_INLINE void USART_vidWaitReceiveComplete (void)

```
{while(!(S_USART->m_UCSRA.sBits.m_RXC));}
```

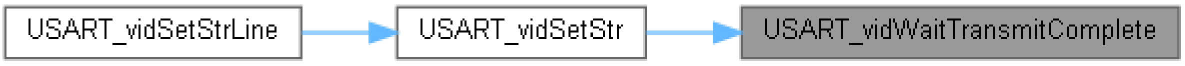
Here is the caller graph for this function:



LCTY_INLINE void USART_vidWaitTransmitComplete (void)

```
{while(!(S_USART->m_UCSRA.sBits.m_TXC));}
```

Here is the caller graph for this function:



Variable Documentation

void(* pvidfunc_Empty_CallBak) (void) (void) = _voidCallBack [static]

void(* pvidfunc_Rx_CallBak) (void) (void) = _voidCallBack [static]

void(* pvidfunc_Tx_CallBak) (void) (void) = _voidCallBack [static]

USART_tstrBuffer strRX_GLB [static]

```
Initial value:= {
    .m_pu8Data   = LBTY_NULL,
    .m_u8Size    = LBTY_u8ZERO,
    .m_u8Idx     = LBTY_u8ZERO,
    .m_u8Status = RX_IDLE,
}
```

USART_tstrBuffer strTX_GLB [static]

```
Initial value:= {
    .m_pu8Data   = LBTY_NULL,
    .m_u8Size    = LBTY_u8ZERO,
    .m_u8Idx     = LBTY_u8ZERO,
    .m_u8Status = TX_IDLE,
}
```

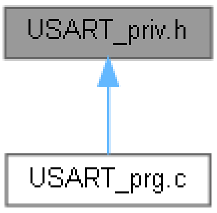
volatile UCSRC_type strUCSRC [static]

volatile USART_tstrConfiguration strUSART_Config_GLB [static]

```
Initial value:= {
    .m_Mode      = USART_OPERATION_MODE,
    .m_Polarity = USART_OPERATION_POLARITY,
    .m_Speed     = USART_OPERATION_SPEED,
    .m_BuadRate = USART_BUAD_RATE_INIT,
    .m_Size      = USART_CHAR_SIZE_INIT,
    .m_Parity    = USART_PARITY_BIT_INIT,
    .m_Stop      = USART_STOP_BIT_INIT,
    .m_TXEN      = USART_TRANSMIT_INIT,
    .m_RXEN      = USART_RECEIVE_INIT,
    .m_TXIE      = USART_TRANSMIT_COMPLETE_INT,
    .m_RXIE      = USART_RECEIVE_COMPLETE_INT,
    .m_Empty     = USART_DATA_REG_EMPTY_INT,
}
```

USART_priv.h File Reference

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



Data Structures

struct USART_tstrBuffer: *UART TX/RX Buffer*

union UCSRC_type: *Type define of Union bit field of "USART Control and Status RegC"*

union UCSRB_type: *Type define of Union bit field of "USART Control and Status RegB"*

union UCSRA_type: *Type define of Union bit field of "USART Control and Status RegA"*

struct USART_type: *UART Registers*

Macros

- #define S_USART ((USART_type* const)0x29U)
- #define UBRRL (*(volatile u8* const)0x29U)
- #define UCSRB (*(volatile u8* const)0x2AU)
- #define UCSRA (*(volatile u8* const)0x2BU)
- #define UDR (*(volatile u8* const)0x2CU)
- #define UCSRC (*(volatile u8* const)0x40U)
- #define UBRRH (*(volatile u8* const)0x40U)
- #define USART_UCSZ0_BIT 0u
- #define USART_UCSZ1_BIT 1u
- #define USART_UCSZ2_BIT 2u
- #define USART_XCK_PORT B
- #define USART_XCK_PIN GPIO_USART_XCK
- #define USART_PORT D
- #define USART_RX_PIN GPIO_UART_RX
- #define USART_TX_PIN GPIO_UART_TX

Enumerations

- enum USART_tstrStatus { TX_IDLE, TX_BUSY, RX_IDLE, RX_BUSY }
  : Type define of TX/RX Status

- enum USART_tenumRegSelect { USART_UBRRH_Reg = (u8)0u, USART_UCSRC_Reg }
  : Type define of UCSRC Register Selection

Macro Definition Documentation

#define S_USART   ((USART_type* const)0x29U)

USART

#define UBRRH   (*(volatile u8* const)0x40U)

#define UBRRL   (*(volatile u8* const)0x29U)

#define UCSRA   (*(volatile u8* const)0x2BU)

#define UCSRB   (*(volatile u8* const)0x2AU)

#define UCSRC   (*(volatile u8* const)0x40U)

#define UDR   (*(volatile u8* const)0x2CU)

#define USART_PORT   D

#define USART_RX_PIN   GPIO_UART_RX

#define USART_TX_PIN   GPIO_UART_TX

#define USART_UCSZ0_BIT   0u

#define USART_UCSZ1_BIT   1u

#define USART_UCSZ2_BIT   2u

#define USART_XCK_PIN   GPIO_USART_XCK

#define USART_XCK_PORT   B

Enumeration Type Documentation

enum USART_tenumRegSelect

: Type define of UCSRC Register Selection

Type   : Enum **Unit**   : None

Enumerator:

USART_UBRRH_Reg	
USART_UCSRC_Reg	

```
        {
    USART_UBRRH_Reg = (u8)0u,
    USART_UCSRC_Reg,
}USART_tenumRegSelect;
```

enum USART_tstrStatus

: Type define of TX/RX Status

Type : Enum **Unit** : None

Enumerator:

TX_IDLE	
TX_BUSY	
RX_IDLE	
RX_BUSY	

```
              {
    TX_IDLE,
    TX_BUSY,
    RX_IDLE,
    RX_BUSY,
}USART_tstrStatus;
```

USART_priv.h

Go to the documentation of this file.

```
/* ************************************************************************ */
/* ********************** FILE DEFINITION SECTION ************************** */
/* ************************************************************************ */
/* File Name   : USART_priv.h                                               */
/* Author      : MAAM                                                       */
/* Version     : v01.2                                                      */
/* date        : Apr 10, 2023                                               */
/* ************************************************************************ */
/* ************************ HEADER FILES INCLUDES ************************* */
/* ************************************************************************ */

#ifndef USART_PRIV_H_
#define USART_PRIV_H_

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

typedef enum{
    TX_IDLE,
    TX_BUSY,
    RX_IDLE,
    RX_BUSY,
}USART_tstrStatus;

typedef struct{
    pu8 m_pu8Data;
    u8  m_u8Size;
    u8  m_u8Idx;
    u8  m_u8Status;
}USART_tstrBuffer;

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

typedef enum{
    USART_UBRRH_Reg = (u8)0u,
    USART_UCSRC_Reg,
}USART_tenumRegSelect;

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

typedef union{
    u8 u_Reg;
    struct {
        __IO u8 m_UCPOL: 1;
        __IO u8 m_UCSZ0: 1;
        __IO u8 m_UCSZ1: 1;
        __IO u8 m_USBS : 1;
        __IO u8 m_UPM  : 2;
        __IO u8 m_UMSEL: 1;
        __IO u8 m_URSEL: 1;
    }sUCSRC;
    struct {
        __IO u8 m_UBRR : 4;
        __IO u8        : 3;
        __IO u8 m_URSEL: 1;
    }sUBRRH;
}UCSRC_type;

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

typedef union{
    u8 u_Reg;
    struct {
        __IO u8 m_TXB8 : 1;
        __I  u8 m_RXB8 : 1;
        __IO u8 m_UCSZ2: 1;
        __IO u8 m_TXEN : 1;
        __IO u8 m_RXEN : 1;
        __IO u8 m_UDRIE: 1;
        __IO u8 m_TXCIE: 1;
        __IO u8 m_RXCIE: 1;
```

```c
    }sBits;
}UCSRB_type;

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

typedef union{
    u8 u_Reg;
    struct {
        __IO u8 m_MPCM: 1;
        __IO u8 m_U2X : 1;
        __I  u8 m_PE  : 1;
        __I  u8 m_DOR : 1;
        __I  u8 m_FE  : 1;
        __I  u8 m_UDRE: 1;
        __IO u8 m_TXC : 1;
        __I  u8 m_RXC : 1;
    }sBits;
}UCSRA_type;

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

typedef struct{
    __IO u8          m_UBRRL;
    __IO UCSRB_type m_UCSRB;
    __IO UCSRA_type m_UCSRA;
    __IO u8          m_UDR;
    __I  u8          REVERSE[19];
    __IO u8          m_UCSRC;
}USART_type;

/* ************************************************************************* */
/* ************************** MACRO/DEFINE SECTION ************************** */
/* ************************************************************************* */

#define S_USART         ((USART_type* const)0x29U)
#define UBRRL           (*(volatile u8* const)0x29U)
#define UCSRB           (*(volatile u8* const)0x2AU)
#define UCSRA           (*(volatile u8* const)0x2BU)
#define UDR             (*(volatile u8* const)0x2CU)

#define UCSRC           (*(volatile u8* const)0x40U)
#define UBRRH           (*(volatile u8* const)0x40U)

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

#define USART_UCSZ0_BIT             0u
#define USART_UCSZ1_BIT             1u
#define USART_UCSZ2_BIT             2u

#define USART_XCK_PORT              B
#define USART_XCK_PIN               GPIO_USART_XCK

#define USART_PORT                  D
#define USART_RX_PIN                GPIO_UART_RX
#define USART_TX_PIN                GPIO_UART_TX

/* ************************************************************************* */
/* ***************************** CONST SECTION ****************************** */
/* ************************************************************************* */

/* ************************************************************************* */
/* ***************************** VARIABLE SECTION *************************** */
/* ************************************************************************* */

/* ************************************************************************* */
/* ***************************** FUNCTION SECTION *************************** */
/* ************************************************************************* */

#endif /* USART_PRIV_H_ */
/*************************** E N D (USART_priv.h) ******************************/
```