

SWC_USART

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

Table of Contents

| | |
|--|-------------------------------------|
| Data Structure Index..... | 2 |
| File Index..... | 3 |
| Data Structure Documentation | 4 |
| LBTY_tuniPort16..... | 4 |
| LBTY_tuniPort8..... | 6 |
| UCSRA_type..... | 8 |
| UCSRB_type..... | 10 |
| UCSRC_type..... | 12 |
| USART_tstrBuffer | 14 |
| USART_tstrConfiguration..... | 15 |
| USART_type..... | 17 |
| File Documentation | 19 |
| H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_BSW/LBIT_int.h | 19 |
| H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_BSW/LBIT_int.h | 22 |
| H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_BSW/LBTY_int.h | 24 |
| H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_BSW/LBTY_int.h | 29 |
| H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_BSW/LCTY_int.h..... | 32 |
| H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_BSW/LCTY_int.h..... | 33 |
| main.c | 34 |
| USART_cfg.h..... | 35 |
| USART_int.h..... | 38 |
| USART_prg.c..... | 53 |
| USART_priv.h | 64 |
| Index..... | Error! Bookmark not defined. |

Data Structure Index

Data Structures

Here are the data structures with brief descriptions:

| | |
|---|----|
| <u>LBTY_tuniPort16</u> | 4 |
| <u>LBTY_tuniPort8</u> | 6 |
| <u>UCSRA_type</u> (: Type define of Union bit field of "USART Control and Status RegA") | 8 |
| <u>UCSRB_type</u> (: Type define of Union bit field of "USART Control and Status RegB") | 10 |
| <u>UCSRC_type</u> (: Type define of Union bit field of "USART Control and Status RegC") | 12 |
| <u>USART_tstrBuffer</u> (: UART TX/RX Buffer) | 14 |
| <u>USART_tstrConfiguration</u> (: type define of structure for UART/USART Configuration) | 15 |
| <u>USART_type</u> (: UART Registers) | 17 |

File Index

File List

Here is a list of all files with brief descriptions:

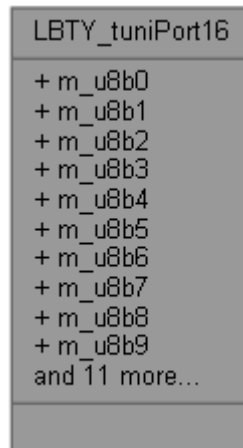
| | |
|---|----|
| H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_BSW/LBIT_int.h | 19 |
| H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_BSW/LBTY_int.h | 24 |
| H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_BSW/LCTY_int.h | 32 |
| main.c | 34 |
| USART_cfg.h | 35 |
| USART_int.h | 38 |
| USART_prg.c | 53 |
| USART_priv.h | 64 |

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

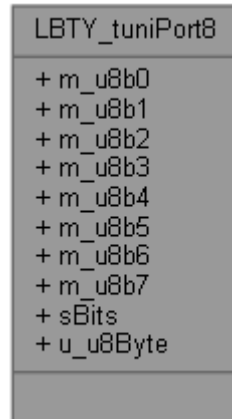
The documentation for this union was generated from the following file:

- 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:

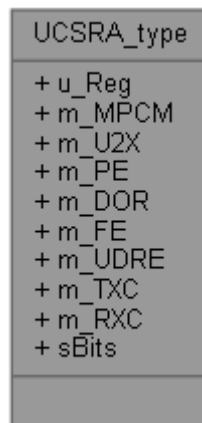
- H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_BSW/[LBTY_int.h](#)

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:

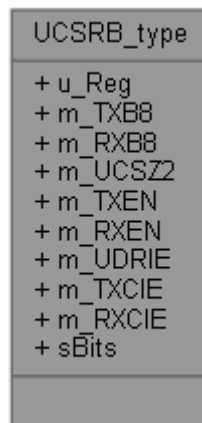
[USART_priv.h](#)

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:



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:

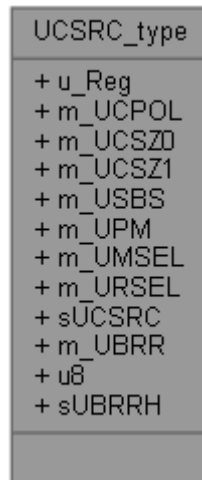
[USART_priv.h](#)

UCSRC_type Union Reference

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

```
#include <USART_priv.h>
```

Collaboration diagram for UCSR_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
- } [s_UCSRC](#)
- struct {
- [__IO u8 m_UBRR](#): 4
- [__IO u8](#): 3
- [__IO u8 m_URSEL](#): 1
- } [s_UBRRH](#)

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:

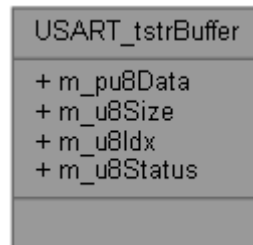
[USART_priv.h](#)

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:

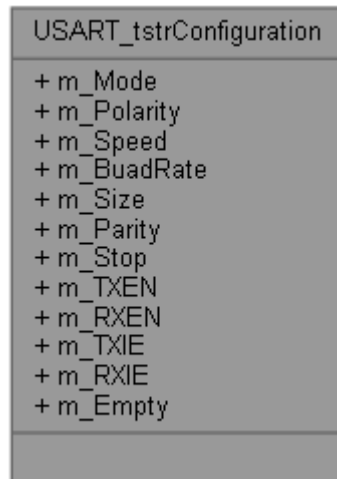
[USART_priv.h](#)

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:

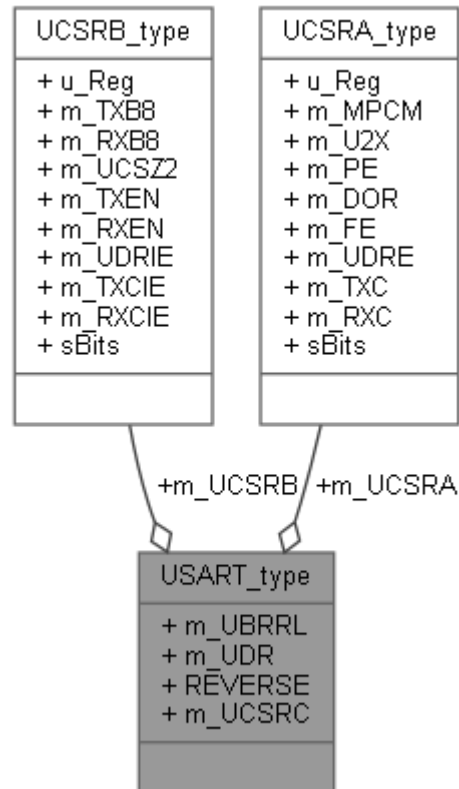
[USART_int.h](#)

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](#)

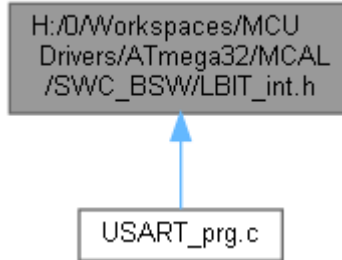
File Documentation

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:



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))
```



```

Go to the documentation of this file.1 /*
***** */
2 /* ***** FILE DEFINITION SECTION ***** */
3 /* ***** */
4 /* File Name      : LBIT_int.h */
5 /* Author         : MAAM */
6 /* Version        : v01 */
7 /* date           : Mar 24, 2023 */
8 /* description    : Bitwise Library */
9 /* ***** */
10 /* ***** HEADER FILES INCLUDES ***** */
11 /* ***** */
12
13 #ifndef LBIT_INT_H_
14 #define LBIT_INT_H_
15
16 /* ***** */
17 /* ***** TYPE_DEF/STRUCT/ENUM SECTION ***** */
18 /* ***** */
19
20 /* ***** */
21 /* ***** MACRO/DEFINE SECTION ***** */
22 /* ***** */
23
24 #define _BV(bit) (1u<<(bit))
25
26 #define SET_BIT(REG, bit) ((REG) |= (1u<<(bit)))
27 #define CLR_BIT(REG, bit) ((REG) &= ~(1u<<(bit)))
28 #define TOG_BIT(REG, bit) ((REG) ^= (1u<<(bit)))
29
30
31 #define SET_BYTE(REG, bit) ((REG) |= (0xFFu<<(bit)))
32 #define CLR_BYTE(REG, bit) ((REG) &= ~(0xFFu<<(bit)))
33 #define TOG_BYTE(REG, bit) ((REG) ^= (0xFFu<<(bit)))
34
35 #define SET_MASK(REG, MASK) ((REG) |= (MASK))
36 #define CLR_MASK(REG, MASK) ((REG) &= ~(MASK))
37 #define TOG_MASK(REG, MASK) ((REG) ^= (MASK))
38 #define GET_MASK(REG, MASK) ((REG) & (MASK))
39
40 #define SET_REG(REG) ((REG) = ~(0u))
41 #define CLR_REG(REG) ((REG) = (0u))
42 #define TOG_REG(REG) ((REG) ^= ~(0u))
43
44 #define GET_BIT(REG, bit) (((REG)>>(bit)) & 0x01u)
45 #define GET_NIB(REG, bit) (((REG)>>(bit)) & 0x0Fu)
46 #define GET_BYTE(REG, bit) (((REG)>>(bit)) & 0xFFu)
47
48 #define ASSIGN_BIT(REG, bit, value) ((REG) = ((REG) & ~(0x01u<<(bit))) | ((value) & 0x01u)<<(bit)))
49 #define ASSIGN_NIB(REG, bit, value) ((REG) = ((REG) & ~(0x0Fu<<(bit))) | ((value) & 0x0Fu)<<(bit)))
50 #define ASSIGN_BYTE(REG, bit, value) ((REG) = ((REG) & ~(0xFFu<<(bit))) | ((value) & 0xFFu)<<(bit)))
51
52 /*
53 #define ASSIGN_BIT(REG,bit,value) do{
54 \
55 \
56 \
57 \
58 \
59 \
60 \
61 \
62 \
63 \
64 \
65 \
66 \
67 \
68 \
69 \
70 \
71 \
72 \
73 \
74 \
75 \
76 \
77 \
78 \
79 \
80 \
81 \
82 \
83 \
84 \
85 \
86 \
87 \
88 \
89 \
90 \
91 \
92 \
93 \
94 \
95 \
96 \
97 \
98 \
99 \
100 \
101 \
102 \
103 \
104 \
105 \
106 \
107 \
108 \
109 \
110 \
111 \
112 \
113 \
114 \
115 \
116 \
117 \
118 \
119 \
120 \
121 \
122 \
123 \
124 \
125 \
126 \
127 \
128 \
129 \
130 \
131 \
132 \
133 \
134 \
135 \
136 \
137 \
138 \
139 \
140 \
141 \
142 \
143 \
144 \
145 \
146 \
147 \
148 \
149 \
150 \
151 \
152 \
153 \
154 \
155 \
156 \
157 \
158 \
159 \
160 \
161 \
162 \
163 \
164 \
165 \
166 \
167 \
168 \
169 \
170 \
171 \
172 \
173 \
174 \
175 \
176 \
177 \
178 \
179 \
180 \
181 \
182 \
183 \
184 \
185 \
186 \
187 \
188 \
189 \
190 \
191 \
192 \
193 \
194 \
195 \
196 \
197 \
198 \
199 \
200 \
201 \
202 \
203 \
204 \
205 \
206 \
207 \
208 \
209 \
210 \
211 \
212 \
213 \
214 \
215 \
216 \
217 \
218 \
219 \
220 \
221 \
222 \
223 \
224 \
225 \
226 \
227 \
228 \
229 \
230 \
231 \
232 \
233 \
234 \
235 \
236 \
237 \
238 \
239 \
240 \
241 \
242 \
243 \
244 \
245 \
246 \
247 \
248 \
249 \
250 \
251 \
252 \
253 \
254 \
255 \
256 \
257 \
258 \
259 \
260 \
261 \
262 \
263 \
264 \
265 \
266 \
267 \
268 \
269 \
270 \
271 \
272 \
273 \
274 \
275 \
276 \
277 \
278 \
279 \
280 \
281 \
282 \
283 \
284 \
285 \
286 \
287 \
288 \
289 \
290 \
291 \
292 \
293 \
294 \
295 \
296 \
297 \
298 \
299 \
300 \
301 \
302 \
303 \
304 \
305 \
306 \
307 \
308 \
309 \
310 \
311 \
312 \
313 \
314 \
315 \
316 \
317 \
318 \
319 \
320 \
321 \
322 \
323 \
324 \
325 \
326 \
327 \
328 \
329 \
330 \
331 \
332 \
333 \
334 \
335 \
336 \
337 \
338 \
339 \
340 \
341 \
342 \
343 \
344 \
345 \
346 \
347 \
348 \
349 \
350 \
351 \
352 \
353 \
354 \
355 \
356 \
357 \
358 \
359 \
360 \
361 \
362 \
363 \
364 \
365 \
366 \
367 \
368 \
369 \
370 \
371 \
372 \
373 \
374 \
375 \
376 \
377 \
378 \
379 \
380 \
381 \
382 \
383 \
384 \
385 \
386 \
387 \
388 \
389 \
390 \
391 \
392 \
393 \
394 \
395 \
396 \
397 \
398 \
399 \
400 \
401 \
402 \
403 \
404 \
405 \
406 \
407 \
408 \
409 \
410 \
411 \
412 \
413 \
414 \
415 \
416 \
417 \
418 \
419 \
420 \
421 \
422 \
423 \
424 \
425 \
426 \
427 \
428 \
429 \
430 \
431 \
432 \
433 \
434 \
435 \
436 \
437 \
438 \
439 \
440 \
441 \
442 \
443 \
444 \
445 \
446 \
447 \
448 \
449 \
450 \
451 \
452 \
453 \
454 \
455 \
456 \
457 \
458 \
459 \
460 \
461 \
462 \
463 \
464 \
465 \
466 \
467 \
468 \
469 \
470 \
471 \
472 \
473 \
474 \
475 \
476 \
477 \
478 \
479 \
480 \
481 \
482 \
483 \
484 \
485 \
486 \
487 \
488 \
489 \
490 \
491 \
492 \
493 \
494 \
495 \
496 \
497 \
498 \
499 \
500 \
501 \
502 \
503 \
504 \
505 \
506 \
507 \
508 \
509 \
510 \
511 \
512 \
513 \
514 \
515 \
516 \
517 \
518 \
519 \
520 \
521 \
522 \
523 \
524 \
525 \
526 \
527 \
528 \
529 \
530 \
531 \
532 \
533 \
534 \
535 \
536 \
537 \
538 \
539 \
540 \
541 \
542 \
543 \
544 \
545 \
546 \
547 \
548 \
549 \
550 \
551 \
552 \
553 \
554 \
555 \
556 \
557 \
558 \
559 \
560 \
561 \
562 \
563 \
564 \
565 \
566 \
567 \
568 \
569 \
570 \
571 \
572 \
573 \
574 \
575 \
576 \
577 \
578 \
579 \
580 \
581 \
582 \
583 \
584 \
585 \
586 \
587 \
588 \
589 \
590 \
591 \
592 \
593 \
594 \
595 \
596 \
597 \
598 \
599 \
600 \
601 \
602 \
603 \
604 \
605 \
606 \
607 \
608 \
609 \
610 \
611 \
612 \
613 \
614 \
615 \
616 \
617 \
618 \
619 \
620 \
621 \
622 \
623 \
624 \
625 \
626 \
627 \
628 \
629 \
630 \
631 \
632 \
633 \
634 \
635 \
636 \
637 \
638 \
639 \
640 \
641 \
642 \
643 \
644 \
645 \
646 \
647 \
648 \
649 \
650 \
651 \
652 \
653 \
654 \
655 \
656 \
657 \
658 \
659 \
660 \
661 \
662 \
663 \
664 \
665 \
666 \
667 \
668 \
669 \
670 \
671 \
672 \
673 \
674 \
675 \
676 \
677 \
678 \
679 \
680 \
681 \
682 \
683 \
684 \
685 \
686 \
687 \
688 \
689 \
690 \
691 \
692 \
693 \
694 \
695 \
696 \
697 \
698 \
699 \
700 \
701 \
702 \
703 \
704 \
705 \
706 \
707 \
708 \
709 \
710 \
711 \
712 \
713 \
714 \
715 \
716 \
717 \
718 \
719 \
720 \
721 \
722 \
723 \
724 \

```

```

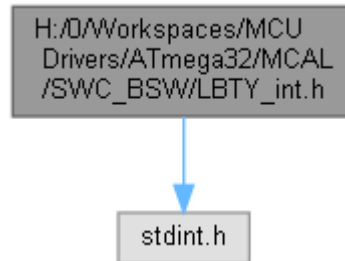
65 (0b##b15##b14##b13##b12##b11##b10##b9##b8##b7##b6##b5##b4##b3##b2##b1##b0)
66
67 /* ***** */
68 /* ***** CONST SECTION ***** */
69 /* ***** */
70
71 /* ***** */
72 /* ***** VARIABLE SECTION ***** */
73 /* ***** */
74
75 /* ***** */
76 /* ***** FUNCTION SECTION ***** */
77 /* ***** */
78
79
80 #endif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

```

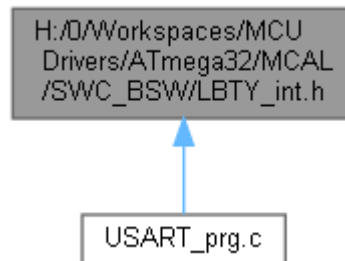
H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_BSW/LBTY_int.h File Reference

#include <stdint.h>

Include dependency graph for LBTY_int.h:



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



Data Structures

- union [LBTY_tuniPort8](#) union [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)0x7FFFFFFFFFFFFFFFL)
- #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)0x7FFFFFFFFFFFFFFFL)`

#define `LBTY_s64MIN` `((u64)0x8000000000000000LL)`

#define `LBTY_s8MAX` `((s8)0x7F)`

#define `LBTY_s8MIN` `((s8)0x80)`

#define `LBTY_u16MAX` `((u16)0xFFFFU)`

#define `LBTY_u16ZERO` `((u16)0x0000U)`

#define `LBTY_u32MAX` `((u32)0xFFFFFFFFUL)`

#define `LBTY_u32ZERO` `((u32)0x00000000UL)`

#define `LBTY_u64MAX` `((u64)0xFFFFFFFFFFFFFFFFULL)`

#define `LBTY_u64ZERO` `((u64)0x0000000000000000ULL)`

#define `LBTY_u8MAX` `((u8)0xFFU)`

#define `LBTY_u8vidNOP()`

#define `LBTY_u8ZERO` `((u8)0x00U)`

Data Types Limitation

Typedef Documentation

typedef `float` [f32](#)

Standard Real Decimal number

typedef double [f64](#)

typedef [s16](#)* [ps16](#)

typedef [s32](#)* [ps32](#)

typedef [s64](#)* [ps64](#)

typedef [s8](#)* [ps8](#)

Standard Pointer to Signed Byte/Word/Long_Word

typedef [u16](#)* [pu16](#)

typedef [u32](#)* [pu32](#)

typedef [u64](#)* [pu64](#)

typedef [u8](#)* [pu8](#)

Standard Pointer to Unsigned Byte/Word/Long_Word

typedef int16_t [s16](#)

typedef int32_t [s32](#)

typedef int64_t [s64](#)

typedef int8_t [s8](#)

Standard Signed Byte/Word/Long_Word

typedef uint16_t [u16](#)

typedef uint32_t [u32](#)

typedef uint64_t [u64](#)

typedef uint8_t [u8](#)

Data Types New Definitions Standard Unsigned Byte/Word/Long_Word

Enumeration Type Documentation

enum [LBTY_tenuBoolean](#)

Boolean type

Enumerator:

| | | |
|--|----------------------------|--|
| | LBTY_TRUE | |
| | LBTY_FALSE | |

```
96 {
97     LBTY\_TRUE = 0x55,
98     LBTY\_FALSE = 0xAA
99 } LBTY\_tenuBoolean;
```

enum [LBTY_tenuErrorStatus](#)

Error Return type

Enumerator:

| | |
|-------------------------|--|
| LBTY_OK | |
| LBTY_NOK | |
| LBTY_NULL_POINTER | |
| LBTY_INDEX_OUT_OF_RANGE | |
| LBTY_NO_MASTER_CHANNEL | |
| LBTY_READ_ERROR | |
| LBTY_WRITE_ERROR | |
| LBTY_UNDEFINED_ERROR | |
| LBTY_IN_PROGRESS | |

```
102     {
103     LBTY\_OK = (u16)0,
104     LBTY\_NOK,
105     LBTY\_NULL\_POINTER,
106     LBTY\_INDEX\_OUT\_OF\_RANGE,
107     LBTY\_NO\_MASTER\_CHANNEL,
108     LBTY\_READ\_ERROR,
109     LBTY\_WRITE\_ERROR,
110     LBTY\_UNDEFINED\_ERROR,
111     LBTY\_IN\_PROGRESS          /* Error is not available, wait for availability */
112 } LBTY\_tenuErrorStatus;
```

enum [LBTY_tenuFlagStatus](#)

Flag Status type

Enumerator:

| | |
|------------|--|
| LBTY_RESET | |
| LBTY_SET | |

```
90     {
91     LBTY\_RESET = 0,
92     LBTY\_SET = !LBTY\_RESET
93 } LBTY\_tenuFlagStatus;
```

LBTY_int.h

```
Go to the documentation of this file.1 /*
*****
2 /* ***** FILE DEFINITION SECTION ***** */
3 /* ***** */
4 /* File Name : LBTY_int.h */
5 /* Author : MAAM */
6 /* Version : v01 */
7 /* date : Mar 23, 2023 */
8 /* description : Basic Library */
9 /* ***** */
10 /* ***** HEADER FILES INCLUDES ***** */
11 /* ***** */
12
13 #ifndef _LBTY_INT_H_
14 #define _LBTY_INT_H_
15
16 #include <stdint.h>
17
18 /* ***** */
19 /* ***** TYPE_DEF SECTION ***** */
20 /* ***** */
21
22 typedef uint8_t u8 ;
23 typedef uint16_t u16;
24 typedef uint32_t u32;
25 typedef uint64_t u64;
26
27
28
29 typedef int8_t s8 ;
30 typedef int16_t s16;
31 typedef int32_t s32;
32 typedef int64_t s64;
33
34
35 typedef float f32;
36 typedef double f64;
37
38
39 typedef u8* pu8 ;
40 typedef u16* pu16;
41 typedef u32* pu32;
42 typedef u64* pu64;
43
44
45 typedef s8* ps8 ;
46 typedef s16* ps16;
47 typedef s32* ps32;
48 typedef s64* ps64;
49
50
51 /* ***** */
52 /* ***** MACRO/DEFINE SECTION ***** */
53 /* ***** */
54
55 /*****
56 #define __IO volatile
57 #define __O volatile
58 #define __I volatile const
59 *****/
60
61 #define LBTY_u8vidNOP()
62 #define LBTY_NULL ((void *) 0U)
63
64 #define LBTY_u8ZERO ((u8)0x00U)
65 #define LBTY_u8MAX ((u8)0xFFU)
66 #define LBTY_s8MAX ((s8)0x7F )
67 #define LBTY_s8MIN ((s8)0x80 )
68
69
70 #define LBTY_u16ZERO ((u16)0x0000U)
71 #define LBTY_u16MAX ((u16)0xFFFFU)
72 #define LBTY_s16MAX ((u16)0x7FFF )
73 #define LBTY_s16MIN ((u16)0x8000 )
74
75 #define LBTY_u32ZERO ((u32)0x00000000UL)
76 #define LBTY_u32MAX ((u32)0xFFFFFFFFUL)
77 #define LBTY_s32MAX ((u32)0x7FFFFFFF )
78 #define LBTY_s32MIN ((u32)0x80000000L )
79

```



```

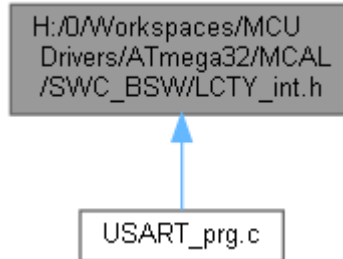
80 #define LBTY_u64ZERO      ((u64)0x0000000000000000ULL)
81 #define LBTY_u64MAX       ((u64)0xFFFFFFFFFFFFFFFFULL)
82 #define LBTY_s64MAX       ((u64)0x7FFFFFFFFFFFFFFFLL )
83 #define LBTY_s64MIN       ((u64)0x8000000000000000LL )
84
85 /* ***** */
86 /* ***** ENUM SECTION ***** */
87 /* ***** */
88
89 typedef enum {
90     LBTY_RESET = 0,
91     LBTY_SET = !LBTY_RESET
92 } LBTY_tenuFlagStatus;
93
94
95 typedef enum {
96     LBTY_TRUE = 0x55,
97     LBTY_FALSE = 0xAA
98 } LBTY_tenuBoolean;
99
100
101 typedef enum {
102     LBTY_OK = (u16)0,
103     LBTY_NOK,
104     LBTY_NULL_POINTER,
105     LBTY_INDEX_OUT_OF_RANGE,
106     LBTY_NO_MASTER_CHANNEL,
107     LBTY_READ_ERROR,
108     LBTY_WRITE_ERROR,
109     LBTY_UNDEFINED_ERROR,
110     LBTY_IN_PROGRESS /* Error is not available, wait for availability */
111 } LBTY_tenuErrorStatus;
112
113
114 /* ***** */
115 /* ***** STRUCT SECTION ***** */
116 /* ***** */
117
118 typedef union {
119     struct {
120         u8 m u8b0 :1; // LSB
121         u8 m u8b1 :1;
122         u8 m u8b2 :1;
123         u8 m u8b3 :1;
124         u8 m u8b4 :1;
125         u8 m u8b5 :1;
126         u8 m u8b6 :1;
127         u8 m u8b7 :1; // MSB
128     } sBits;
129     u8 u u8Byte;
130 } LBTY_tuniPort8;
131
132
133 typedef union {
134     struct {
135         u8 m u8b0 :1; // LSB
136         u8 m u8b1 :1;
137         u8 m u8b2 :1;
138         u8 m u8b3 :1;
139         u8 m u8b4 :1;
140         u8 m u8b5 :1;
141         u8 m u8b6 :1;
142         u8 m u8b7 :1;
143         u8 m u8b8 :1;
144         u8 m u8b9 :1;
145         u8 m u8b10 :1;
146         u8 m u8b11 :1;
147         u8 m u8b12 :1;
148         u8 m u8b13 :1;
149         u8 m u8b14 :1;
150         u8 m u8b15 :1; // MSB
151     } sBits;
152     struct {
153         u8 m u8low;
154         u8 m u8high;
155     } sBytes;
156     u16 u u16Word;
157 } LBTY_tuniPort16;
158
159 /* ***** */
160 /* ***** FUNCTION SECTION ***** */

```

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

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

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



Macros

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

Macro Definition Documentation

#define CTY_PACKED __attribute__((__packed__))

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

#define LCTY_CONST __attribute__((__const__))

#define LCTY_DPAGE __attribute__((dp))

#define LCTY_INLINE __attribute__((always_inline)) static inline

#define LCTY_INTERRUPT __attribute__((interrupt))

#define LCTY_NODPAGE __attribute__((nodp))

#define LCTY_PROGMEM __attribute__((__progmem__))

#define LCTY_PURE __attribute__((__pure__))

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

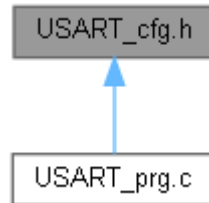
LCTY_int.h

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

main.c File Reference

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_BaudRate_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](#)

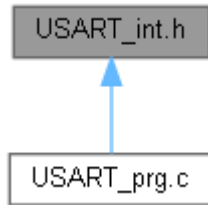
#define USART_TRANSMIT_INIT [LBTY_SET](#)

USART_cfg.h

```
Go to the documentation of this file.1 /*
*****
2 /* ***** FILE DEFINITION SECTION ***** */
3 /* ***** */
4 /* File Name : USART_cfg.h */
5 /* Author : MAAM */
6 /* Version : v01.2 */
7 /* date : Apr 10, 2023 */
8 /* ***** */
9 /* ***** HEADER FILES INCLUDES ***** */
10 /* ***** */
11
12 #ifndef USART_CFG_H_
13 #define USART_CFG_H_
14
15 /* ***** */
16 /* ***** TYPE_DEF/STRUCT/ENUM SECTION ***** */
17 /* ***** */
18
19 /* ***** */
20 /* ***** MACRO/DEFINE SECTION ***** */
21 /* ***** */
22
23 #define USART_OPERATION_MODE USART_Aynchronous
24 #define USART_OPERATION_POLARITY USART_Transmit_Rising_Receive_Falling
25 #define USART_OPERATION_SPEED USART_Speed_x1
26 #define USART_OPERATION_MULTI_PROCESSOR LBTY_RESET
27 #define USART_OPERATION_FREQ F_CPU
28
29 #define USART_BUAD_RATE_INIT USART_BuadRate_9600
30 #define USART_CHAR_SIZE_INIT USART_8_bit
31 #define USART_PARITY_BIT_INIT USART_Parity_Even
32 #define USART_STOP_BIT_INIT USART_Stop_1_bit
33
34 #define USART_TRANSMIT_INIT LBTY_SET
35 #define USART_RECEIVE_INIT LBTY_SET
36
37 #define USART_TRANSMIT_COMPLETE_INT LBTY_RESET
38 #define USART_RECEIVE_COMPLETE_INT LBTY_RESET
39 #define USART_DATA_REG_EMPTY_INT LBTY_RESET
40
41 /* ***** */
42 /* ***** CONST SECTION ***** */
43 /* ***** */
44
45 /* ***** */
46 /* ***** VARIABLE SECTION ***** */
47 /* ***** */
48
49 /* ***** */
50 /* ***** FUNCTION SECTION ***** */
51 /* ***** */
52
53
54 #endif /* USART_CFG_H_ */
55 /***** 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_tenumBaudRate](#) { [USART_BaudRate_1200](#) = 1200u, [USART_BaudRate_2400](#) = 2400u, [USART_BaudRate_4800](#) = 4800u, [USART_BaudRate_9600](#) = 9600u, [USART_BaudRate_14400](#) = 14400u, [USART_BaudRate_19200](#) = 19200u, [USART_BaudRate_28800](#) = 28800u, [USART_BaudRate_38400](#) = 38400u, [USART_BaudRate_57600](#) = 57600u, [USART_BaudRate_76600](#) = 76600u, [USART_BaudRate_115200](#) = 115200u, [USART_BaudRate_230400](#) = 230400u, [USART_BaudRate_250000](#) = 250000u, [USART_BaudRate_500000](#) = 500000u, [USART_BaudRate_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_u8SetBaudRate](#) ([USART_tenumBaudRate](#) u32BaudRate)
- [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 | |

```

54 {
55     USART\_BuadRate\_1200    = 1200u,
56     USART\_BuadRate\_2400    = 2400u,
57     USART\_BuadRate\_4800    = 4800u,
58     USART\_BuadRate\_9600    = 9600u,
59     USART\_BuadRate\_14400   = 14400u,
60     USART\_BuadRate\_19200   = 19200u,
61     USART\_BuadRate\_28800   = 28800u,
62     USART\_BuadRate\_38400   = 38400u,

```

```

63  USART_BaudRate_57600 = 57600u,
64  USART_BaudRate_76600 = 76600u,
65  USART_BaudRate_115200 = 115200u,
66  USART_BaudRate_230400 = 230400u,
67  USART_BaudRate_250000 = 250000u,
68  USART_BaudRate_500000 = 500000u,
69  USART_BaudRate_1000000 = 1000000u,
70 }USART_tenumBaudRate;

```

enum [USART_tenumCharSize](#)

Enumerator:

| | |
|-------------|--|
| USART_5_bit | |
| USART_6_bit | |
| USART_7_bit | |
| USART_8_bit | |
| USART_9_bit | |

```

34  {
35  USART_5_bit = (u8)0u,
36  USART_6_bit,
37  USART_7_bit,
38  USART_8_bit,
39  USART_9_bit = (u8)7u,
40 }USART_tenumCharSize;

```

enum [USART_tenumClockPolarity](#)

Enumerator:

| | |
|---|--|
| USART_Transmit _Rising_Receive_ Falling | |
| USART_Transmit _Falling_Receive_ Rising | |

```

29  {
30  USART_Transmit_Rising_Receive_Falling = (u8)0u,
31  USART_Transmit_Falling_Receive_Rising,
32 }USART_tenumClockPolarity;

```

enum [USART_tenumModeSelect](#)

Enumerator:

| | |
|------------------------|--|
| USART_Asynchro nous | |
| USART_Synchro nous | |

```

19  {
20  USART_Asynchronous = (u8)0u,
21  USART_Synchronous,
22 }USART_tenumModeSelect;

```

enum [USART_tenumParityMode](#)

Enumerator:

| | |
|-----------------------|--|
| USART_Parity_Disable | |
| USART_Parity_Reserved | |
| USART_Parity_Even | |

| | |
|------------------|--|
| USART_Parity_Odd | |
|------------------|--|

```

42 {
43     USART_Parity_Disable = (u8)0u,
44     USART_Parity_Reserved,
45     USART_Parity_Even,
46     USART_Parity_Odd,
47 }USART_tenumParityMode;

```

enum [USART_tenumSpeed](#)

Enumerator:

| | |
|----------------|--|
| USART_Speed_x1 | |
| USART_Speed_x2 | |

```

24 {
25     USART_Speed_x1 = (u8)0u,
26     USART_Speed_x2,
27 }USART_tenumSpeed;

```

enum [USART_tenumStopBit](#)

Enumerator:

| | |
|------------------|--|
| USART_Stop_1_bit | |
| USART_Stop_2_bit | |

```

49 {
50     USART_Stop_1_bit = (u8)0u,
51     USART_Stop_2_bit,
52 }USART_tenumStopBit;

```

Function Documentation

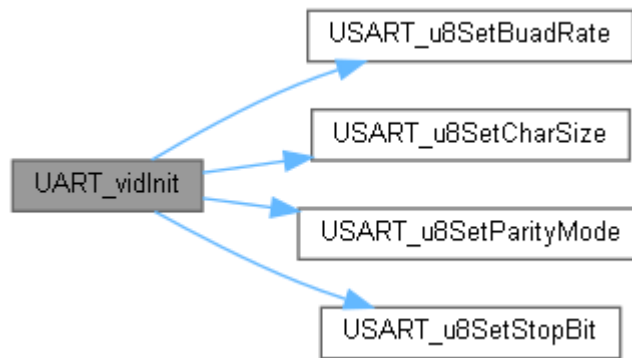
void [UART_vidInit \(void \)](#)

```

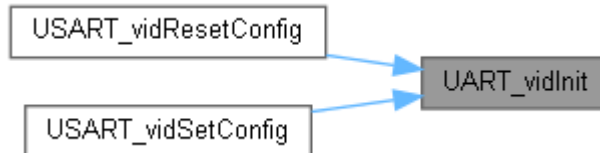
107 {
108
109     strUCSRC.sUCSRC.m URSEL = USART_UCSRC_Reg;
110     strUCSRC.sUCSRC.m UMSEL = strUSART_Config_GLB.m Mode;
111     strUCSRC.sUCSRC.m UCPOL = strUSART_Config_GLB.m Polarity;
112     S_USART->m_UCSRC = strUCSRC.u_Reg;
113
114     S_USART->m_UCSRA.sBits.m MPCM = USART_OPERATION_MULTI_PROCESSOR;
115     S_USART->m_UCSRA.sBits.m U2X = strUSART_Config_GLB.m Speed;
116
117     USART_u8SetBuadRate (strUSART_Config_GLB.m BuadRate);
118     USART_u8SetCharSize (strUSART_Config_GLB.m Size);
119     USART_u8SetParityMode(strUSART_Config_GLB.m Parity);
120     USART_u8SetStopBit (strUSART_Config_GLB.m Stop);
121
122     strUCSRC.sUCSRC.m URSEL = USART_UCSRC_Reg;
123     if(strUCSRC.sUCSRC.m UMSEL == USART_Synchronous){
124         GPIO_u8SetPinDirection(USART_XCK_PORT, USART_XCK_PIN, PIN_OUTPUT);
125     }
126     GPIO_u8SetPinDirection(USART_PORT, USART_TX_PIN, PIN_OUTPUT);
127     GPIO_u8SetPinDirection(USART_XCK_PORT, USART_RX_PIN, PIN_INPUT);
128
129     S_USART->m_UCSRB.sBits.m UDRIE = strUSART_Config_GLB.m Empty;
130     S_USART->m_UCSRB.sBits.m TXCIE = strUSART_Config_GLB.m TXIE;
131     S_USART->m_UCSRB.sBits.m RXCIE = strUSART_Config_GLB.m RXIE;
132
133     S_USART->m_UCSRB.sBits.m TXEN = strUSART_Config_GLB.m TXEN;
134     S_USART->m_UCSRB.sBits.m RXEN = strUSART_Config_GLB.m RXEN;
135 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



u8 USART_u8Available (void)

```

252     {
253     return S_USART->m_UCSRA.sBits.m_RXC;
254 }
  
```

Here is the caller graph for this function:

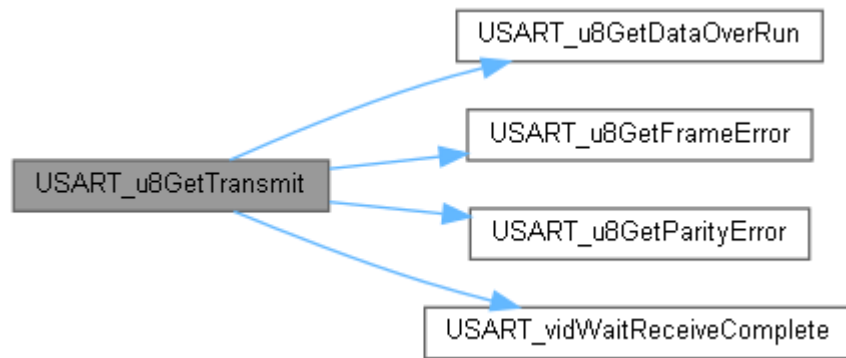


LBTY_tenuErrorStatus USART_u8GetTransmit (void * pvidTransmit)

```

277     {
278     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
279
280     if(pvidTransmit == LBTY_NULL){
281         u8RetErrorState = LBTY_NULL_POINTER;
282     }else{
283         USART_vidWaitReceiveComplete();
284         if(USART_u8GetFrameError() || USART_u8GetDataOverRun() ||
USART_u8GetParityError()){
285             if(strUSART Config GLB.m Size == USART_9_bit){
286                 *((u16*)pvidTransmit) = LBTY_u16MAX;
287             }else{
288                 *((u8*)pvidTransmit) = LBTY_u8MAX;
289             }
290             u8RetErrorState = LBTY_NOK;
291         }else{
292             if(strUSART Config GLB.m Size == USART_9_bit){
293                 *((u16*)pvidTransmit) = (u16)S_USART->m_UDR |
(u16)(S_USART->m_UCSRB.sBits.m_TXB8 << 8);
294             }else{
295                 *((u8*)pvidTransmit) = S_USART->m_UDR;
296             }
297         }
298     }
299     return u8RetErrorState;
300 }
  
```

Here is the call graph for this function:



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

```

351 {
352     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
353
354     if(pu8Data == LBTY_NULL){
355         u8RetErrorState = LBTY_NULL_POINTER;
356     }else{
357         if(strRX_GLB.m_u8Status == RX_IDLE){
358             strRX_GLB.m_pu8Data = pu8Data;
359             strRX_GLB.m_u8Size = u8Size;
360             strRX_GLB.m_u8Idx = LBTY_u8ZERO;
361             strRX_GLB.m_u8Status = RX_BUSY;
362
363             USART_vidEnableReceiveCompleteINT();
364         }else{
365             u8RetErrorState = LBTY_NOK;
366         }
367     }
368
369     return u8RetErrorState;
370 }
  
```

Here is the call graph for this function:

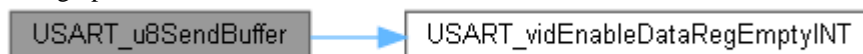


LBTY_tenuErrorStatus USART_u8SendBuffer (u8 * *pu8Data*, u8 *u8Size*)

```

328 {
329     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
330
331     if(pu8Data == LBTY_NULL){
332         u8RetErrorState = LBTY_NULL_POINTER;
333     }else{
334         if(strTX_GLB.m_u8Status == TX_IDLE){
335             strTX_GLB.m_pu8Data = pu8Data;
336             strTX_GLB.m_u8Size = u8Size;
337             strTX_GLB.m_u8Idx = LBTY_u8ZERO;
338             strTX_GLB.m_u8Status = TX_BUSY;
339
340             if(S_USART->m_UCSRA.sBits.m_UDRE){
341                 S_USART->m_UDR = strTX_GLB.m_pu8Data[strTX_GLB.m_u8Idx++];
342             }
343             USART_vidEnableDataRegEmptyINT();
344         }else{
345             u8RetErrorState = LBTY_NOK;
346         }
347     }
348
349     return u8RetErrorState;
350 }
  
```

Here is the call graph for this function:



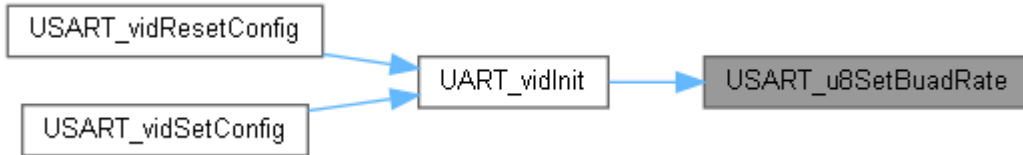
LBTY_tenuErrorStatus USART_u8SetBuadRate (USART_tenumBuadRate u32BuadRate)

```

150
151     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
152     u16 u16UBRR = LBTY_u16ZERO;
153
154     switch(strUCSRC.sUCSRC.m_UMSEL){
155         case USART_Asynchronous:
156             switch(S_USART->m_UCSRA.sBits.m_U2X){
157                 case USART_Speed_x1:
158                     u16UBRR = (u16)(F_CPU / (16.0f * (u32)u32BuadRate)) - 1;
159                     break;
160                 case USART_Speed_x2:
161                     u16UBRR = (u16)(F_CPU / (8.0f * (u32)u32BuadRate)) - 1;
162                     break;
163                 default:
164                     u8RetErrorState = LBTY_NOK;
165             }
166             break;
167         case USART_Synchronous:
168             u16UBRR = (u16)(USART_OPERATION_FREQ / (2.0f * (u32)u32BuadRate)) -
169             1;
170             break;
171         default:
172             u8RetErrorState = LBTY_NOK;
173     }
174     if(u8RetErrorState == LBTY_OK){
175         strUSART_Config_GLB.m_BuadRate = u32BuadRate;
176
177         strUCSRC.sUBRRH.m_URSEL = USART_UBRRH_Reg;
178         strUCSRC.sUBRRH.m_UBRR = GET_NIB(u16UBRR, 8);
179
180         S_USART->m_UCSRC = strUCSRC.u_Reg;
181         S_USART->m_UBRRL = GET_BYTE(u16UBRR, 0);
182     }
183     return u8RetErrorState;
184 }

```

Here is the caller graph for this function:



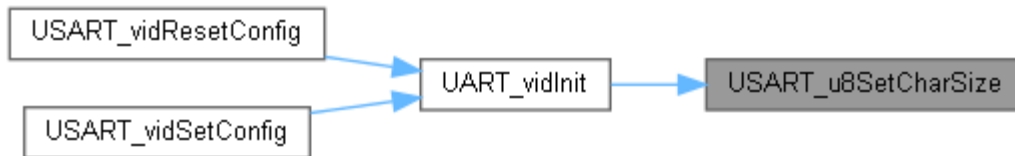
LBTY_tenuErrorStatus USART_u8SetCharSize (USART_tenumCharSize u8CharSize)

```

185
186     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
187
188     switch(u8CharSize){
189         case USART_5_bit:
190         case USART_6_bit:
191         case USART_7_bit:
192         case USART_8_bit:
193         case USART_9_bit:
194             strUCSRC.sUCSRC.m_URSEL = USART_UCSRC_Reg;
195             strUCSRC.sUCSRC.m_UCSZ0 = GET_BIT(u8CharSize, USART_UCSZ0_BIT);
196             strUCSRC.sUCSRC.m_UCSZ1 = GET_BIT(u8CharSize, USART_UCSZ1_BIT);
197             S_USART->m_UCSRB.sBits.m_UCSZ2 = GET_BIT(u8CharSize,
198             USART_UCSZ2_BIT);
199
200             S_USART->m_UCSRC = strUCSRC.u_Reg;
201             strUSART_Config_GLB.m_Size = u8CharSize;
202             break;
203         default:
204             u8RetErrorState = LBTY_NOK;
205     }
206     return u8RetErrorState;
207 }

```

Here is the caller graph for this function:

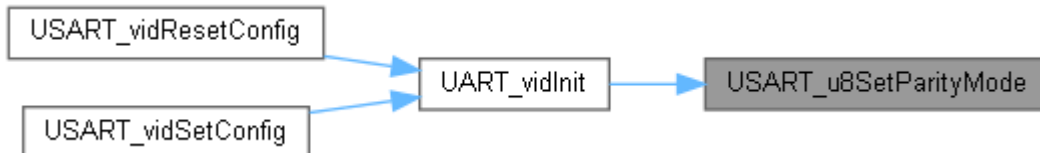


LBTY_tenuErrorStatus USART_u8SetParityMode (USART_tenumParityMode u8Parity)

```

209 {
210     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
211
212     switch(u8Parity){
213         case USART_Parity_Disable:
214         case USART_Parity_Even:
215         case USART_Parity_Odd:
216             strUCSRC.sUCSRC.m URSEL = USART_UCSRC_Reg;
217             strUCSRC.sUCSRC.m UPM = strUSART_Config_GLB.m Parity = u8Parity;
218             S_USART->m_UCSRC = strUCSRC.u_Reg;
219             break;
220         default: u8RetErrorState = LBTY_NOK;
221     }
222     return u8RetErrorState;
223 }
224 
```

Here is the caller graph for this function:



LBTY_tenuErrorStatus USART_u8SetStopBit (USART_tenumStopBit u8Stop)

```

226 {
227     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
228
229     switch(u8Stop){
230         case USART_Stop_1_bit:
231         case USART_Stop_2_bit:
232             strUCSRC.sUCSRC.m URSEL = USART_UCSRC_Reg;
233             strUCSRC.sUCSRC.m USBS = strUSART_Config_GLB.m Stop = u8Stop;
234             S_USART->m_UCSRC = strUCSRC.u_Reg;
235             break;
236         default: u8RetErrorState = LBTY_NOK;
237     }
238     return u8RetErrorState;
239 }
240 
```

Here is the caller graph for this function:



LBTY_tenuErrorStatus USART_u8SetTransmit (void * pvidTransmit)

```

263 {
264     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
265
266     if(pvidTransmit == LBTY_NULL){
267         u8RetErrorState = LBTY_NULL_POINTER;
268     }else{
269         USART_vidWaitDataRegEmpty();
270         S_USART->m_UDR = *((u8*)pvidTransmit);
271         if(strUSART_Config_GLB.m Size == USART_9_bit){
272             S_USART->m_UCSRB.sBits.m TXB8 = GET_BIT(*((u16*)pvidTransmit), 8);
273         }
274     }
275     return u8RetErrorState;
276 }
277 
```



```
276 }
```

Here is the call graph for this function:



void USART_vidDisableDataRegEmptyINT (void)

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

void USART_vidDisableReceiveCompleteINT (void)

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

void USART_vidDisableTransmitCompleteINT (void)

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

void USART_vidEnableDataRegEmptyINT (void)

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

Here is the caller graph for this function:



void USART_vidEnableReceiveCompleteINT (void)

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

Here is the caller graph for this function:



void USART_vidEnableTransmitCompleteINT (void)

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

void USART_vidFlush (void)

```

256 {
257     u8 dummy;
258     while(USART_u8Available()) {
259         dummy = S_USART->m_UDR;
260     }
261 }
```

Here is the call graph for this function:

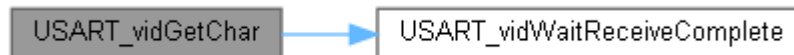


void USART_vidGetChar (u8 * pu8Char)

```

306 {
307     USART_vidWaitReceiveComplete();
308     *pu8Char = S_USART->m_UDR;
309 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

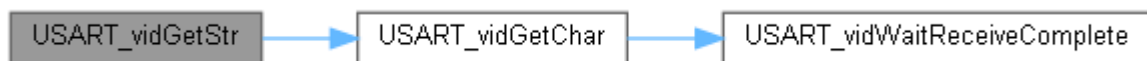


void USART_vidGetStr (u8 * pu8Receive)

```

321 {
322     do{
323         USART_vidGetChar(pu8Receive);
324     }while(*pu8Receive++ != '\r');
325     *pu8Receive = '\0';
326 }
```

Here is the call graph for this function:



void USART_vidReceiverDisable (void)

```

146 {
147     S_USART->m_UCSRB.sBits.m_RXEN = strUSART Config GLB.m_RXEN = LBTY_RESET;
148 }
  
```

void USART_vidReceiverEnable (void)

```

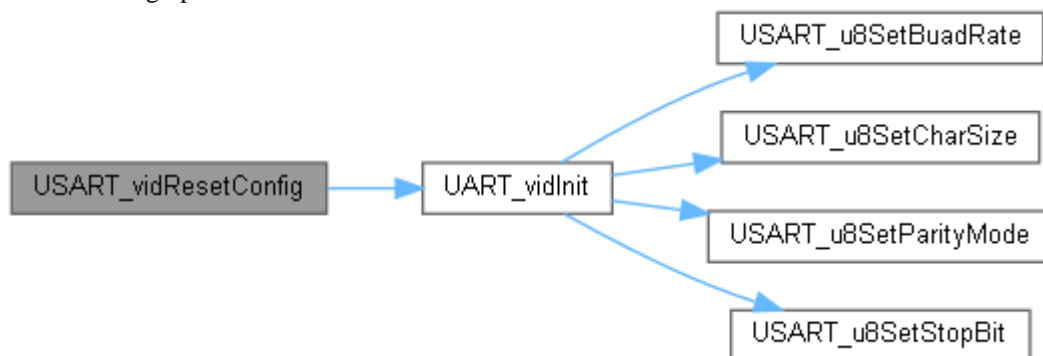
143 {
144     S_USART->m_UCSRB.sBits.m_RXEN = strUSART Config GLB.m_RXEN = LBTY_SET;
145 }
  
```

void USART_vidResetConfig (USART tstrConfiguration *const pstrConfig)

```

87 {
88     strUSART Config GLB.m_Mode = USART OPERATION MODE;
89     strUSART Config GLB.m_Polarity = USART OPERATION POLARITY;
90     strUSART Config GLB.m_Speed = USART OPERATION SPEED;
91     strUSART Config GLB.m_BaudRate = USART BUAD RATE INIT;
92     strUSART Config GLB.m_Size = USART CHAR SIZE INIT;
93     strUSART Config GLB.m_Parity = USART PARITY BIT INIT;
94     strUSART Config GLB.m_Stop = USART STOP BIT INIT;
95     strUSART Config GLB.m_TXEN = USART TRANSMIT INIT;
96     strUSART Config GLB.m_RXEN = USART RECEIVE INIT;
97     strUSART Config GLB.m_TXIE = USART TRANSMIT COMPLETE INT;
98     strUSART Config GLB.m_RXIE = USART RECEIVE COMPLETE INT;
99     strUSART Config GLB.m_Empty = USART DATA REG EMPTY INT;
100
101     if(pstrConfig != LBTY_NULL){
102         *pstrConfig = strUSART Config GLB;
103     }
104     UART_vidInit();
105 }
  
```

Here is the call graph for this function:



void USART_vidSetCallBack_Empty (void*)(void) pCallBack)

```

382 {
383     pvidfunc Empty CallBak = pCallBack;
384 }
  
```

void USART_vidSetCallBack_RX (void*)(void) pCallBack)

```

388 {
389     pvidfunc Rx CallBak = pCallBack;
390 }
  
```

void USART_vidSetCallBack_TX (void*)(void) pCallBack)

```

385 {
386     pvidfunc Tx CallBak = pCallBack;
387 }
  
```

void USART_vidSetChar (u8 u8Char)

```

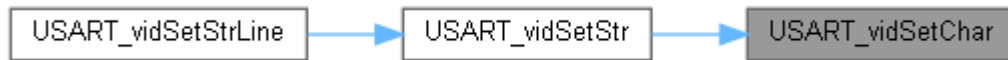
302 {
303     USART_vidWaitDataRegEmpty();
304     S_USART->m_UDR = u8Char;
  
```

```
305 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

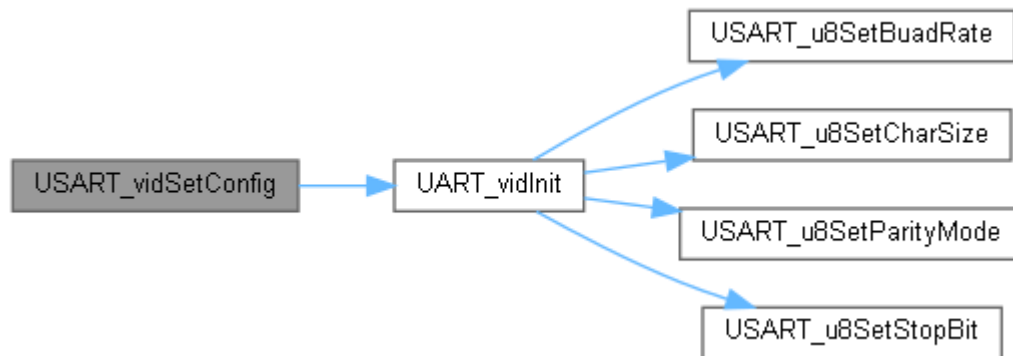


void USART_vidSetConfig (USART_tstrConfiguration const *const pstrConfig)

```

80 {
81     if(pstrConfig != LBTY_NULL){
82         strUSART_Config_GLB = *pstrConfig;
83     }
84     UART_vidInit();
85 }
  
```

Here is the call graph for this function:

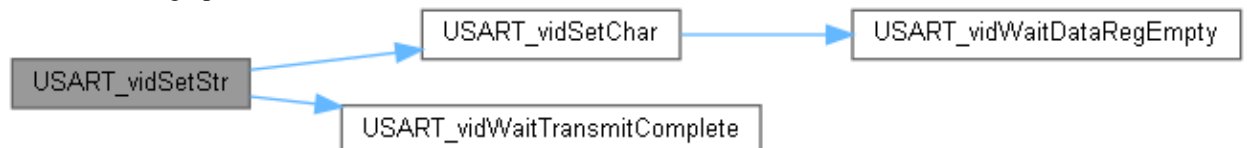


void USART_vidSetStr (u8 * pu8Transmit)

```

315 {
316     while(*pu8Transmit){
317         USART_vidSetChar(*pu8Transmit++);
318         USART_vidWaitTransmitComplete();
319     }
320 }
  
```

Here is the call graph for this function:



Here is the caller graph for this function:

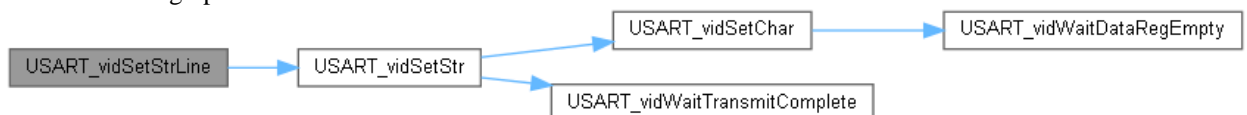


void USART_vidSetStrLine (u8 * pu8Transmit)

```

311 {
312     USART_vidSetStr((u8*)pu8Transmit);
313     USART_vidSetStr((u8*)"\r\n");
314 }
  
```

Here is the call graph for this function:



void USART_vidTransmitterDisable (void)

```

140 {
141     S_USART->m_UCSRB.sBits.m_TXEN = strUSART_Config_GLB.m_TXEN = LBTY_RESET;
142 }
  
```

void USART_vidTransmitterEnable (void)

```
137 {  
138     S USART->m_UCSRB.sBits.m_TXEN = strUSART Config GLB.m TXEN = LBTY SET;  
139 }
```

USART_int.h

```
Go to the documentation of this file.1 /*
*****
2 /* ***** FILE DEFINITION SECTION ***** */
3 /* ***** */
4 /* File Name : USART_int.h */
5 /* Author : MAAM */
6 /* Version : v01.2 */
7 /* date : Apr 10, 2023 */
8 /* ***** */
9 /* ***** HEADER FILES INCLUDES ***** */
10 /* ***** */
11
12 #ifndef USART_INT_H_
13 #define USART_INT_H_
14
15 /* ***** */
16 /* ***** TYPE_DEF/STRUCT/ENUM SECTION ***** */
17 /* ***** */
18
19 typedef enum{
20     USART Asynchronous = (u8)0u,
21     USART Synchronous,
22 }USART tenumModeSelect;
23
24 typedef enum{
25     USART Speed x1 = (u8)0u,
26     USART Speed x2,
27 }USART tenumSpeed;
28
29 typedef enum{
30     USART Transmit Rising Receive Falling = (u8)0u,
31     USART Transmit Falling Receive Rising,
32 }USART tenumClockPolarity;
33
34 typedef enum{
35     USART 5 bit = (u8)0u,
36     USART 6 bit,
37     USART 7 bit,
38     USART 8 bit,
39     USART 9 bit = (u8)7u,
40 }USART tenumCharSize;
41
42 typedef enum{
43     USART Parity Disable = (u8)0u,
44     USART Parity Reserved,
45     USART Parity Even,
46     USART Parity Odd,
47 }USART tenumParityMode;
48
49 typedef enum{
50     USART Stop 1 bit = (u8)0u,
51     USART Stop 2 bit,
52 }USART tenumStopBit;
53
54 typedef enum{
55     USART BaudRate 1200 = 1200u,
56     USART BaudRate 2400 = 2400u,
57     USART BaudRate 4800 = 4800u,
58     USART BaudRate 9600 = 9600u,
59     USART BaudRate 14400 = 14400u,
60     USART BaudRate 19200 = 19200u,
61     USART BaudRate 28800 = 28800u,
62     USART BaudRate 38400 = 38400u,
63     USART BaudRate 57600 = 57600u,
64     USART BaudRate 76600 = 76600u,
65     USART BaudRate 115200 = 115200u,
66     USART BaudRate 230400 = 230400u,
67     USART BaudRate 250000 = 250000u,
68     USART BaudRate 500000 = 500000u,
69     USART BaudRate 1000000 = 1000000u,
70 }USART tenumBaudRate;
71
```

```

72
73
74 /*****
75 *****/
76 typedef struct{
77 /* HW Config */
78     USART_tenumModeSelect    m_Mode;
79     USART_tenumClockPolarity m_Polarity;
80     USART_tenumSpeed         m_Speed;
81 /* frame */
82     USART_tenumBaudRate      m_BaudRate;
83     USART_tenumCharSize      m_Size;
84     USART_tenumParityMode    m_Parity;
85     USART_tenumStopBit       m_Stop;
86 /* TX/RX Status */
87     LBTY_tenuFlagStatus      m_TXEN;
88     LBTY_tenuFlagStatus      m_RXEN;
89 /* TX/RX Interrupt */
90     LBTY_tenuFlagStatus      m_TXIE;
91     LBTY_tenuFlagStatus      m_RXIE;
92     LBTY_tenuFlagStatus      m_Empty;
93 }USART_tstrConfiguration;
94
95 /* *****/
96 /* *****/
97 /* *****/
98
99 // #define USART_vidSetChar      USART_vidSetTransmit
100 // #define USART_vidGetChar      USART_vidGetTransmit
101
102 /* *****/
103 /* *****/
104 /* *****/
105
106 /* *****/
107 /* *****/
108 /* *****/
109
110 /* *****/
111 /* *****/
112 /* *****/
113
114 extern void USART_vidSetConfig(USART_tstrConfiguration const* const pstrConfig);
115 extern void USART_vidResetConfig(USART_tstrConfiguration* const pstrConfig);
116
117 extern void UART_vidInit(void);
118
119 extern void USART_vidTransmitterEnable(void);
120 extern void USART_vidTransmitterDisable(void);
121 extern void USART_vidReceiverEnable(void);
122 extern void USART_vidReceiverDisable(void);
123
124 extern LBTY_tenuErrorStatus USART_u8SetBaudRate(USART_tenumBaudRate u32BaudRate);
125 extern LBTY_tenuErrorStatus USART_u8SetCharSize(USART_tenumCharSize u8CharSize);
126 extern LBTY_tenuErrorStatus USART_u8SetParityMode(USART_tenumParityMode u8Parity);
127 extern LBTY_tenuErrorStatus USART_u8SetStopBit(USART_tenumStopBit u8Stop);
128
129 /*****
130 *****/
131 extern u8 USART_u8Available(void);
132 extern void USART_vidFlush(void);
133
134 extern LBTY_tenuErrorStatus USART_u8SetTransmit(void* pvidTransmit);
135 extern LBTY_tenuErrorStatus USART_u8GetTransmit(void* pvidTransmit);
136
137 extern void USART_vidSetChar(u8 u8Char);
138 extern void USART_vidGetChar(u8* pu8Char);
139
140 extern void USART_vidSetStrLine(u8* pu8Transmit);
141 extern void USART_vidSetStr(u8* pu8Transmit);
142 extern void USART_vidGetStr(u8* pu8Receive);
143
144 extern LBTY_tenuErrorStatus USART_u8SendBuffer(u8* pu8Data, u8 u8Size);
145 extern LBTY_tenuErrorStatus USART_u8ReceiveBuffer(u8* pu8Data, u8 u8Size);
146

```

```

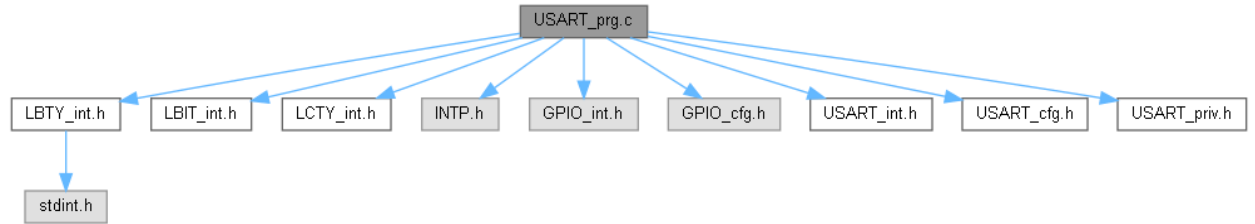
147
/*****
*****
*****/
148
149 extern void USART_vidEnableReceiveCompleteINT(void);
150 extern void USART_vidEnableTransmitCompleteINT(void);
151 extern void USART_vidEnableDataRegEmptyINT(void);
152
153 extern void USART_vidDisableReceiveCompleteINT(void);
154 extern void USART_vidDisableTransmitCompleteINT(void);
155 extern void USART_vidDisableDataRegEmptyINT(void);
156
157 extern void USART_vidSetCallBack_Empty(void (*pCallBack)(void));
158 extern void USART_vidSetCallBack_TX(void (*pCallBack)(void));
159 extern void USART_vidSetCallBack_RX(void (*pCallBack)(void));
160
161 #endif /* USART_INT_H_ */
162 /***** 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]

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

ISR (USART_RXC_vect)

```
392 {
393     if((S_USART->m_UCSRA.sBits.m_RXC) && (strRX_GLB.m_u8Idx <
strRX\_GLB.m\_u8Size) && (strRX_GLB.m_u8Status == RX_BUSY)){
394         strRX_GLB.m_pu8Data[strRX_GLB.m_u8Idx++] = S_USART->m_UDR;
395     }else{
396         strRX_GLB.m_u8Status = RX_IDLE;
397         //USART_vidDisableReceiveCompleteINT();
398         pvidfunc\_Rx\_CallBak();
399     }
400 }
```

ISR (USART_TXC_vect)

```
410 {
411     if(strTX_GLB.m_u8Status == TX_IDLE){
412         strTX_GLB.m_u8Status = TX_IDLE;
413         //USART_vidDisableTransmitCompleteINT();
414         pvidfunc\_Tx\_CallBak();
415     }
416 }
```

ISR (USART_UDRE_vect)

```
401 {
402     if((S_USART->m_UCSRA.sBits.m_UDRE) && (strTX_GLB.m_u8Idx <
strTX\_GLB.m\_u8Size) && (strTX_GLB.m_u8Status == TX_BUSY)){
403         S_USART->m_UDR = strTX_GLB.m_pu8Data[strTX_GLB.m_u8Idx++];
404     }else{
405         strTX_GLB.m_u8Status = TX_IDLE;
406         //USART_vidDisableDataRegEmptyINT();
407         pvidfunc\_Empty\_CallBak();
408     }
409 }
```

void UART_vidInit (void)

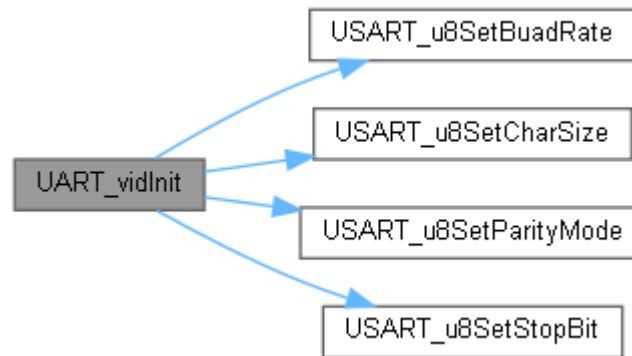
```
107 {
108
109     strUCSRC.sUCSRC.m URSEL = USART_UCSRC_Reg;
110     strUCSRC.sUCSRC.m UMSEL = strUSART_Config_GLB.m_Mode;
111     strUCSRC.sUCSRC.m UCPOL = strUSART_Config_GLB.m_Polarity;
112     S_USART->m_UCSRC = strUCSRC.u_Reg;
```

```

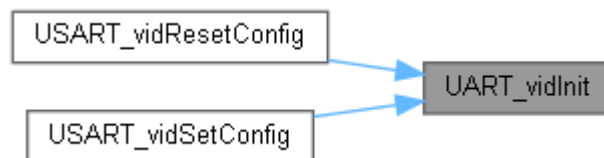
113
114 S USART->m_UCSRA.sBits.m_MPCM = USART OPERATION MULTI PROCESSOR;
115 S USART->m_UCSRA.sBits.m_U2X = strUSART Config GLB.m Speed;
116
117 USART_u8SetBuadRate (strUSART Config GLB.m BuadRate);
118 USART_u8SetCharSize (strUSART Config GLB.m Size);
119 USART_u8SetParityMode(strUSART Config GLB.m Parity);
120 USART_u8SetStopBit (strUSART Config GLB.m Stop);
121
122 strUCSRC.sUCSRC.m URSEL = USART UCSRC Reg;
123 if(strUCSRC.sUCSRC.m UMSEL == USART_Synchronous){
124     GPIO_u8SetPinDirection(USART_XCK_PORT, USART_XCK_PIN, PIN_OUTPUT);
125 }
126 GPIO_u8SetPinDirection(USART_PORT , USART_TX_PIN , PIN_OUTPUT);
127 GPIO_u8SetPinDirection(USART_XCK_PORT, USART_RX_PIN , PIN_INPUT );
128
129 S USART->m_UCSRB.sBits.m_UDRIE = strUSART Config GLB.m Empty;
130 S USART->m_UCSRB.sBits.m_TXCIE = strUSART Config GLB.m TXIE;
131 S USART->m_UCSRB.sBits.m_RXCIE = strUSART Config GLB.m RXIE;
132
133 S USART->m_UCSRB.sBits.m_TXEN = strUSART Config GLB.m TXEN;
134 S USART->m_UCSRB.sBits.m_RXEN = strUSART Config GLB.m RXEN;
135 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



u8 USART_u8Available (void)

```

252 {
253     return S USART->m_UCSRA.sBits.m_RXC;
254 }

```

Here is the caller graph for this function:



LCTY_INLINE u8 USART_u8GetDataOverRun (void)

```

249 {return S USART->m_UCSRA.sBits.m_DOR;}

```

Here is the caller graph for this function:



LCTY_INLINE u8 USART_u8GetFrameError (void)

```

248 {return S USART->m_UCSRA.sBits.m_FE;}

```

Here is the caller graph for this function:



LCTY_INLINE u8 USART_u8GetParityError (void)

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

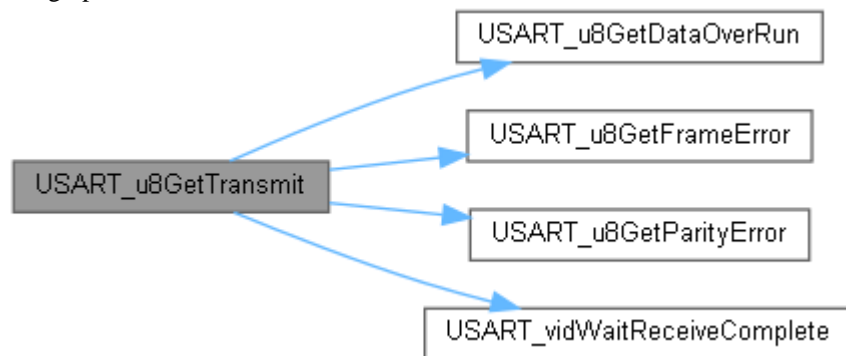
Here is the caller graph for this function:



LBTY_tenuErrorStatus USART_u8GetTransmit (void * pvidTransmit)

```
277 {
278     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
279
280     if(pvidTransmit == LBTY_NULL){
281         u8RetErrorState = LBTY_NULL_POINTER;
282     }else{
283         USART_vidWaitReceiveComplete();
284         if(USART_u8GetFrameError() || USART_u8GetDataOverRun() ||
USART_u8GetParityError()){
285             if(strUSART_Config_GLB.m_Size == USART_9_bit){
286                 *((u16*)pvidTransmit) = LBTY_u16MAX;
287             }else{
288                 *((u8*)pvidTransmit) = LBTY_u8MAX;
289             }
290             u8RetErrorState = LBTY_NOK;
291         }else{
292             if(strUSART_Config_GLB.m_Size == USART_9_bit){
293                 *((u16*)pvidTransmit) = (u16)S_USART->m_UDR |
(u16)(S_USART->m_UCSRB.sBits.m_TXB8 << 8);
294             }else{
295                 *((u8*)pvidTransmit) = S_USART->m_UDR;
296             }
297         }
298     }
299     return u8RetErrorState;
300 }
```

Here is the call graph for this function:



LBTY_tenuErrorStatus USART_u8ReceiveBuffer (u8 * pu8Data, u8 u8Size)

```
351 {
352     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
353
354     if(pu8Data == LBTY_NULL){
355         u8RetErrorState = LBTY_NULL_POINTER;
356     }else{
357         if(strRX_GLB.m_u8Status == RX_IDLE){
358             strRX_GLB.m_pu8Data = pu8Data;
359             strRX_GLB.m_u8Size = u8Size;
360             strRX_GLB.m_u8Idx = LBTY_u8ZERO;
361             strRX_GLB.m_u8Status = RX_BUSY;
362
363             USART_vidEnableReceiveCompleteINT();
364         }else{
365             u8RetErrorState = LBTY_NOK;
366         }
367     }
368
369     return u8RetErrorState;
370 }
```

Here is the call graph for this function:

USART_u8ReceiveBuffer

USART_vidEnableReceiveCompleteINT

LBTY_tenuErrorStatus USART_u8SendBuffer (u8 * pu8Data, u8 u8Size)

```

328
329     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
330
331     if(pu8Data == LBTY_NULL){
332         u8RetErrorState = LBTY_NULL_POINTER;
333     }else{
334         if(strTX_GLB.m_u8Status == TX_IDLE){
335             strTX_GLB.m_pu8Data = pu8Data;
336             strTX_GLB.m_u8Size = u8Size;
337             strTX_GLB.m_u8Idx = LBTY_u8ZERO;
338             strTX_GLB.m_u8Status = TX_BUSY;
339
340             if(S_USART->m_UCSRA.sBits.m_UDRE){
341                 S_USART->m_UDR = strTX_GLB.m_pu8Data[strTX_GLB.m_u8Idx++];
342             }
343             USART_vidEnableDataRegEmptyINT();
344         }else{
345             u8RetErrorState = LBTY_NOK;
346         }
347     }
348
349     return u8RetErrorState;
350 }

```

Here is the call graph for this function:

USART_u8SendBuffer

USART_vidEnableDataRegEmptyINT

LBTY_tenuErrorStatus USART_u8SetBuadRate (USART_tenumBuadRate u32BuadRate)

```

150
151     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
152     u16 u16UBRR = LBTY_u16ZERO;
153
154     switch(strUCSRC.sUCSRC.m_UMSEL){
155         case USART_Asynchrous:
156             switch(S_USART->m_UCSRA.sBits.m_U2X){
157                 case USART_Speed_x1:
158                     u16UBRR = (u16)(F_CPU / (16.0f * (u32)u32BuadRate)) - 1;
159                     break;
160                 case USART_Speed_x2:
161                     u16UBRR = (u16)(F_CPU / (8.0f * (u32)u32BuadRate)) - 1;
162                     break;
163                 default:
164                     u8RetErrorState = LBTY_NOK;
165             }
166             break;
167         case USART_Synchronous:
168             u16UBRR = (u16)(USART_OPERATION_FREQ / (2.0f * (u32)u32BuadRate)) -
169             1;
170             break;
171         default:
172             u8RetErrorState = LBTY_NOK;
173     }
174     if(u8RetErrorState == LBTY_OK){
175         strUSART_Config_GLB.m_BuadRate = u32BuadRate;
176
177         strUCSRC.sUBRRH.m URSEL = USART_UBRRH_Reg;
178         strUCSRC.sUBRRH.m_UBRR = GET_NIB(u16UBRR, 8);
179
180         S_USART->m_UCSRC = strUCSRC.u_Reg;
181         S_USART->m_UBRRL = GET_BYTE(u16UBRR, 0);
182     }
183     return u8RetErrorState;

```

Here is the caller graph for this function:



LBTY_tenuErrorStatus USART_u8SetCharSize (USART_tenumCharSize u8CharSize)

```

185 {
186     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
187
188     switch(u8CharSize){
189         case USART_5_bit:
190         case USART_6_bit:
191         case USART_7_bit:
192         case USART_8_bit:
193         case USART_9_bit:
194             strUCSRC.sUCSRC.m URSEL = USART_UCSRC_Reg;
195             strUCSRC.sUCSRC.m UCSZ0 = GET_BIT(u8CharSize, USART_UCSZ0_BIT);
196             strUCSRC.sUCSRC.m UCSZ1 = GET_BIT(u8CharSize, USART_UCSZ1_BIT);
197             S_USART->m_UCSRB.sBits.m UCSZ2 = GET_BIT(u8CharSize,
198 USART_UCSZ2_BIT);
199             S_USART->m_UCSRC = strUCSRC.u_Reg;
200             strUSART_Config_GLB.m Size = u8CharSize;
201             break;
202         default:
203             u8RetErrorState = LBTY_NOK;
204     }
205
206     return u8RetErrorState;
207 }
  
```

Here is the caller graph for this function:

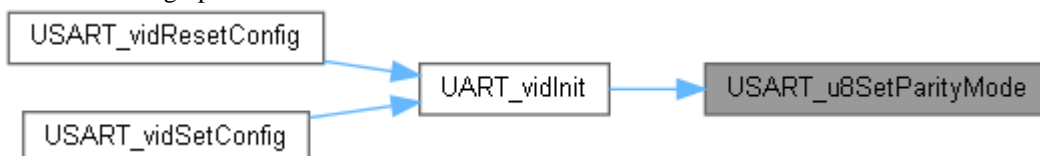


LBTY_tenuErrorStatus USART_u8SetParityMode (USART_tenumParityMode u8Parity)

```

209 {
210     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
211
212     switch(u8Parity){
213         case USART_Parity_Disable:
214         case USART_Parity_Even:
215         case USART_Parity_Odd:
216             strUCSRC.sUCSRC.m URSEL = USART_UCSRC_Reg;
217             strUCSRC.sUCSRC.m UPM = strUSART_Config_GLB.m Parity = u8Parity;
218             S_USART->m_UCSRC = strUCSRC.u_Reg;
219             break;
220         default: u8RetErrorState = LBTY_NOK;
221     }
222
223     return u8RetErrorState;
224 }
  
```

Here is the caller graph for this function:



LBTY_tenuErrorStatus USART_u8SetStopBit (USART_tenumStopBit u8Stop)

```

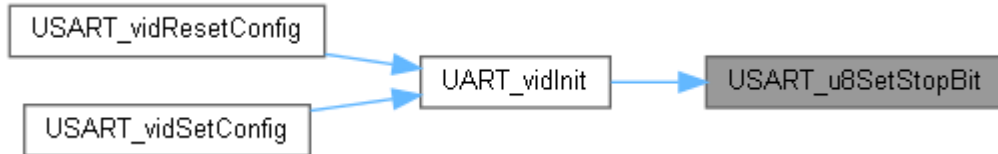
226 {
227     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
228
229     switch(u8Stop){
  
```

```

230     case USART_Stop_1_bit:
231     case USART_Stop_2_bit:
232         strUCSRC.sUCSRC.m URSEL = USART_UCSRC_Reg;
233         strUCSRC.sUCSRC.m USBS = strUSART_Config_GLB.m Stop = u8Stop;
234         S_USART->m_UCSRC = strUCSRC.u_Reg;
235         break;
236     default: u8RetErrorState = LBTY_NOK;
237 }
238
239 return u8RetErrorState;
240 }

```

Here is the caller graph for this function:



LBTY_tenuErrorStatus USART_u8SetTransmit (void * pvidTransmit)

```

263 {
264     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
265
266     if(pvidTransmit == LBTY_NULL){
267         u8RetErrorState = LBTY_NULL_POINTER;
268     }else{
269         USART_vidWaitDataRegEmpty();
270         S_USART->m_UDR = *((u8*)pvidTransmit);
271         if(strUSART_Config_GLB.m Size == USART_9_bit){
272             S_USART->m_UCSRB.sBits.m_TXB8 = GET_BIT(*((u16*)pvidTransmit), 8);
273         }
274     }
275     return u8RetErrorState;
276 }

```

Here is the call graph for this function:



void USART_vidDisableDataRegEmptyINT (void)

```

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

```

void USART_vidDisableReceiveCompleteINT (void)

```

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

```

void USART_vidDisableTransmitCompleteINT (void)

```

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

```

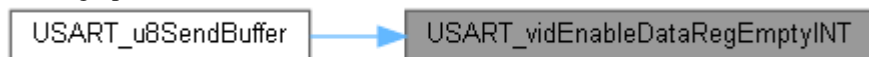
void USART_vidEnableDataRegEmptyINT (void)

```

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

```

Here is the caller graph for this function:



void USART_vidEnableReceiveCompleteINT (void)

```

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

```

Here is the caller graph for this function:



void USART_vidEnableTransmitCompleteINT (void)

```

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

```

void USART_vidFlush (void)

```

256 {

```

```

257     u8 dummy;
258     while(USART_u8Available()) {
259         dummy = S_USART->m_UDR;
260     }
261 }

```

Here is the call graph for this function:



void USART_vidGetChar (u8 * pu8Char)

```

306     {
307         USART_vidWaitReceiveComplete();
308         *pu8Char = S_USART->m_UDR;
309     }

```

Here is the call graph for this function:



Here is the caller graph for this function:



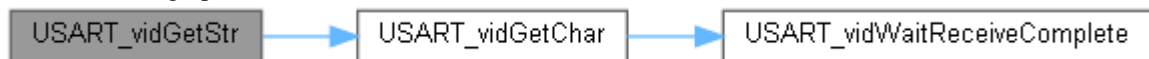
void USART_vidGetStr (u8 * pu8Receive)

```

321     {
322     do{
323         USART_vidGetChar(pu8Receive);
324     }while(*pu8Receive++ != '\r');
325     *pu8Receive = '\0';
326 }

```

Here is the call graph for this function:



void USART_vidReceiverDisable (void)

```

146     {
147         S_USART->m_UCSRB.sBits.m_RXEN = strUSART_Config_GLB.m_RXEN = LBTY_RESET;
148     }

```

void USART_vidReceiverEnable (void)

```

143     {
144         S_USART->m_UCSRB.sBits.m_RXEN = strUSART_Config_GLB.m_RXEN = LBTY_SET;
145     }

```

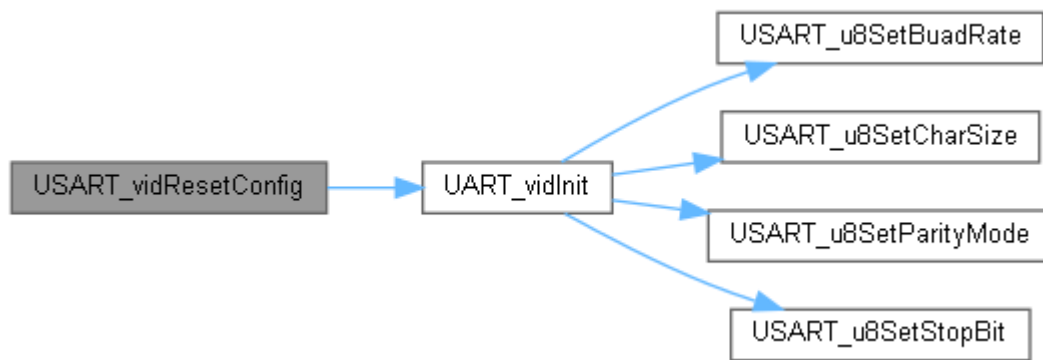
void USART_vidResetConfig (USART_tstrConfiguration *const pstrConfig)

```

87     {
88         strUSART_Config_GLB.m_Mode = USART_OPERATION_MODE;
89         strUSART_Config_GLB.m_Polarity = USART_OPERATION_POLARITY;
90         strUSART_Config_GLB.m_Speed = USART_OPERATION_SPEED;
91         strUSART_Config_GLB.m_BuadRate = USART_BUAD_RATE_INIT;
92         strUSART_Config_GLB.m_Size = USART_CHAR_SIZE_INIT;
93         strUSART_Config_GLB.m_Parity = USART_PARITY_BIT_INIT;
94         strUSART_Config_GLB.m_Stop = USART_STOP_BIT_INIT;
95         strUSART_Config_GLB.m_TXEN = USART_TRANSMIT_INIT;
96         strUSART_Config_GLB.m_RXEN = USART_RECEIVE_INIT;
97         strUSART_Config_GLB.m_TXIE = USART_TRANSMIT_COMPLETE_INT;
98         strUSART_Config_GLB.m_RXIE = USART_RECEIVE_COMPLETE_INT;
99         strUSART_Config_GLB.m_Empty = USART_DATA_REG_EMPTY_INT;
100
101         if(pstrConfig != LBTY_NULL){
102             *pstrConfig = strUSART_Config_GLB;
103         }
104         UART_vidInit();
105     }

```

Here is the call graph for this function:



void USART_vidSetCallBack_Empty (void*)(void) pCallBack)

```

382                                     {
383     pvidfunc Empty CallBak = pCallBack;
384 }
  
```

void USART_vidSetCallBack_RX (void*)(void) pCallBack)

```

388                                     {
389     pvidfunc Rx CallBak = pCallBack;
390 }
  
```

void USART_vidSetCallBack_TX (void*)(void) pCallBack)

```

385                                     {
386     pvidfunc Tx CallBak = pCallBack;
387 }
  
```

void USART_vidSetChar (u8 u8Char)

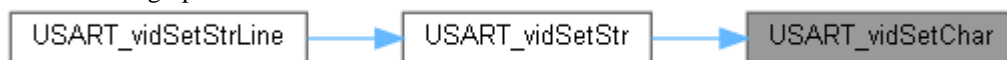
```

302                                     {
303     USART_vidWaitDataRegEmpty();
304     S_USART->m_UDR = u8Char;
305 }
  
```

Here is the call graph for this function:



Here is the caller graph for this function:

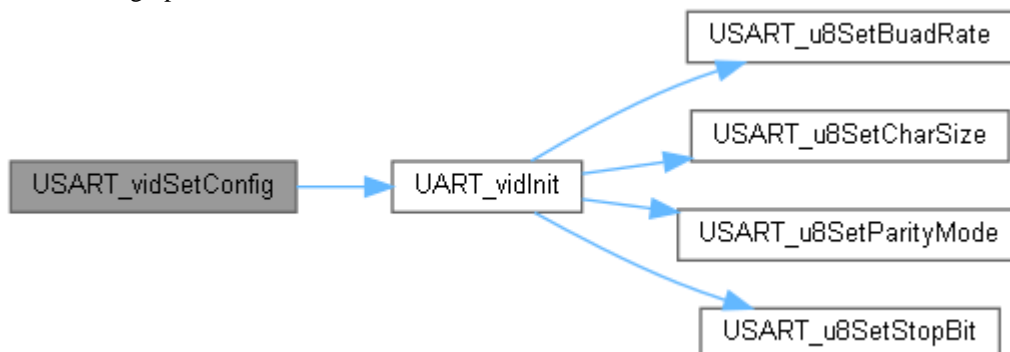


void USART_vidSetConfig (USART_tstrConfiguration const *const pstrConfig)

```

80                                     {
81     if(pstrConfig != LBTY_NULL){
82         strUSART Config GLB = *pstrConfig;
83     }
84     UART_vidInit();
85 }
  
```

Here is the call graph for this function:



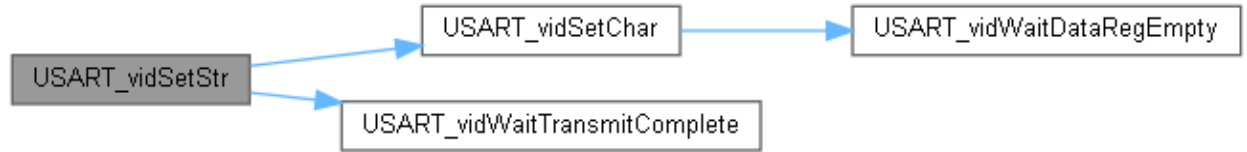
void USART_vidSetStr (u8 * pu8Transmit)

```

315                                     {
316     while(*pu8Transmit){
317         USART_vidSetChar(*pu8Transmit++);
318         USART_vidWaitTransmitComplete();
319     }
320 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



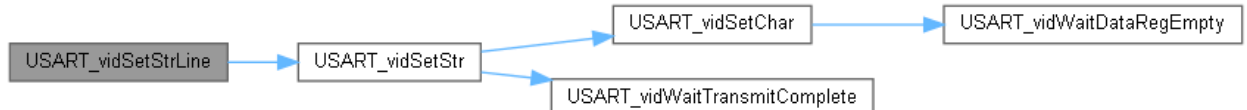
void USART_vidSetStrLine (u8 * pu8Transmit)

```

311                                     {
312     USART_vidSetStr((u8*)pu8Transmit);
313     USART_vidSetStr((u8*)"\r\n");
314 }

```

Here is the call graph for this function:



void USART_vidTransmitterDisable (void)

```

140                                     {
141     S_USART->m_UCSRB.sBits.m_TXEN = strUSART_Config_GLB.m_TXEN = LBTY_RESET;
142 }

```

void USART_vidTransmitterEnable (void)

```

137                                     {
138     S_USART->m_UCSRB.sBits.m_TXEN = strUSART_Config_GLB.m_TXEN = LBTY_SET;
139 }

```

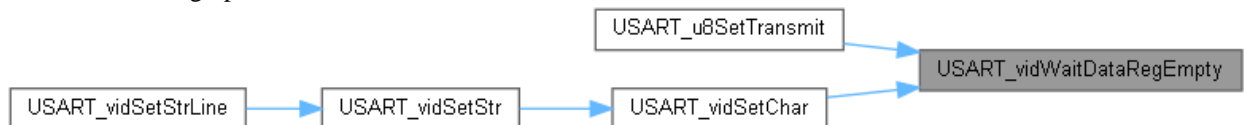
LCTY_INLINE void USART_vidWaitDataRegEmpty (void)

```

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

```

Here is the caller graph for this function:



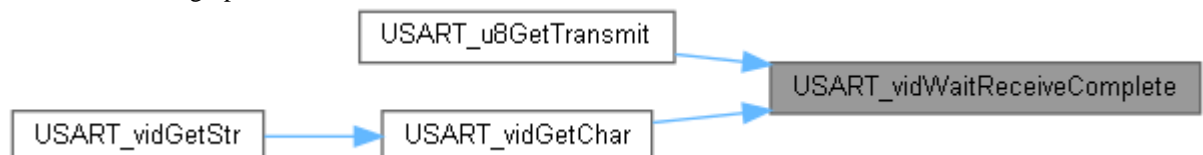
LCTY_INLINE void USART_vidWaitReceiveComplete (void)

```

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

```

Here is the caller graph for this function:



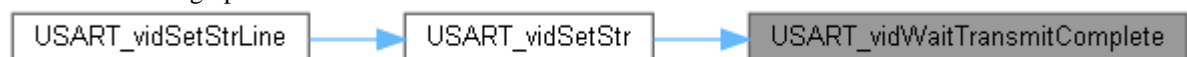
LCTY_INLINE void USART_vidWaitTransmitComplete (void)

```

245 {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,  
}
```

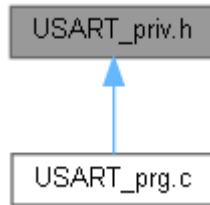
volatile [UCSRC_type](#) strUCSRC[static]

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 [UBRR_L](#) (*(volatile [u8](#)* const)0x29U)
- #define [UCSR_B](#) (*(volatile [u8](#)* const)0x2AU)
- #define [UCSR_A](#) (*(volatile [u8](#)* const)0x2BU)
- #define [UDR](#) (*(volatile [u8](#)* const)0x2CU)
- #define [UCSRC](#) (*(volatile [u8](#)* const)0x40U)
- #define [UBRR_H](#) (*(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 UBRL (*(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 | |

```
41 {
42     USART_UBRRH_Reg = (u8)0u,
43     USART_UCSRC_Reg,
```

```
44 }USART_tenumRegSelect;
```

enum USART_tstrStatus

: Type define of TX/RX Status

Type : Enum **Unit** : None

Enumerator:

| | |
|---------|--|
| TX_IDLE | |
| TX_BUSY | |
| RX_IDLE | |
| RX_BUSY | |

```
21 {  
22     TX_IDLE,  
23     TX_BUSY,  
24     RX_IDLE,  
25     RX_BUSY,  
26 }USART_tstrStatus;
```

USART_priv.h

```
Go to the documentation of this file.1 /*
*****
2 /* ***** FILE DEFINITION SECTION ***** */
3 /* ***** */
4 /* File Name : USART_priv.h */
5 /* Author : MAAM */
6 /* Version : v01.2 */
7 /* date : Apr 10, 2023 */
8 /* ***** */
9 /* ***** HEADER FILES INCLUDES ***** */
10 /* ***** */
11
12 #ifndef USART_PRIV_H
13 #define USART_PRIV_H
14
15 /* ***** */
16 /* ***** TYPE_DEF/STRUCT/ENUM SECTION ***** */
17 /* ***** */
18
19 typedef enum{
20     TX_IDLE,
21     TX_BUSY,
22     RX_IDLE,
23     RX_BUSY,
24 }USART_tstrStatus;
25
26 typedef struct{
27     pu8 m pu8Data;
28     u8 m u8Size;
29     u8 m u8Idx;
30     u8 m u8Status;
31 }USART_tstrBuffer;
32
33 /*****
34
35 typedef enum{
36     USART_UBRRH_Reg = (u8)0u,
37     USART_UCSRC_Reg,
38 }USART_tenumRegSelect;
39
40 /*****
41
42 typedef union{
43     u8 u_Reg;
44     struct {
45         IO u8 m UCPOL: 1;
46         IO u8 m UCSZ0: 1;
47         IO u8 m UCSZ1: 1;
48         IO u8 m USBS: 1;
49         IO u8 m UPM: 2;
50         IO u8 m UMSEL: 1;
51         IO u8 m URSEL: 1;
52     }sUCSRC;
53     struct {
54         IO u8 m_UBRR: 4;
55         IO u8: 3;
56         IO u8 m_URSEL: 1;
57     }sUBRRH;
58 }UCSRC_type;
59
60 /*****
61
62 typedef union{
63     u8 u_Reg;
64     struct {
65         IO u8 m_TXB8: 1;
66         IO u8 m_RXB8: 1;
67         IO u8 m_UCSZ2: 1;
68         IO u8 m_TXEN: 1;
69         IO u8 m_RXEN: 1;
70         IO u8 m_UDRIE: 1;
71         IO u8 m_TXCIE: 1;
72         IO u8 m_RXCIE: 1;
73     }
74 }
```

```

83     }sBits;
84 }UCSRB_type;
85
86 /*****
87
88 typedef union{
89     u8 u_Reg;
90     struct {
91         IO u8 m_MPCM: 1;
92         IO u8 m_U2X : 1;
93         I  u8 m_PE  : 1;
94         I  u8 m_DOR : 1;
95         I  u8 m_FE  : 1;
96         I  u8 m_UDRE: 1;
97         IO u8 m_TXC : 1;
98         I  u8 m_RXC : 1;
99     }sBits;
100 }UCSRA_type;
101
102 /*****
103
104 typedef struct{
105     IO u8 m_UBRRL;
106     IO UCSRB_type m_UCSRB;
107     IO UCSRA_type m_UCSRA;
108     IO u8 m_UDR;
109     I  u8 REVERSE[19];
110     IO u8 m_UCSRC;
111 }USART_type;
112
113 /* *****
114 /* ***** MACRO/DEFINE SECTION *****
115 /* *****
116
117 #define S_USART ((USART_type* const)0x29U)
118 #define UBRRL (* (volatile u8* const)0x29U)
119 #define UCSRB (* (volatile u8* const)0x2AU)
120 #define UCSRA (* (volatile u8* const)0x2BU)
121 #define UDR (* (volatile u8* const)0x2CU)
122
123 #define UCSRC (* (volatile u8* const)0x40U)
124 #define UBRRH (* (volatile u8* const)0x40U)
125
126 /* *****
127
128 #define USART_UCSZ0_BIT 0u
129 #define USART_UCSZ1_BIT 1u
130 #define USART_UCSZ2_BIT 2u
131
132 #define USART_XCK_PORT B
133 #define USART_XCK_PIN GPIO_USART_XCK
134
135 #define USART_PORT D
136 #define USART_RX_PIN GPIO_UART_RX
137 #define USART_TX_PIN GPIO_UART_TX
138
139 /* *****
140 /* ***** CONST SECTION *****
141 /* *****
142
143 /* *****
144 /* ***** VARIABLE SECTION *****
145 /* *****
146
147 /* *****
148 /* ***** FUNCTION SECTION *****
149 /* *****
150
151 #endif /* USART_PRIV_H_ */
152 /***** E N D (USART_priv.h) *****/

```