SWC_MOTOR

Version v1.0
10/16/2023 11:08:00 AM

Table of Contents

Data Structure Index

Data Structures

Here are the data structures with brief descriptions:

File Index

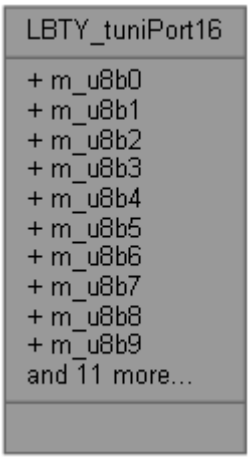
File List

Here is a list of all files with brief descriptions:

Data Structure Documentation

LBTY_tuniPort16 Union Reference

`#include <LBTY_int.h>`
Collaboration diagram for LBTY_tuniPort16:



Data Fields

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

Field Documentation

u8 m_u8b0

u8 m_u8b1

u8 m_u8b10

u8 m_u8b11

u8 m_u8b12

u8 m_u8b13

u8 m_u8b14

u8 m_u8b15

u8 m_u8b2

u8 m_u8b3

u8 m_u8b4

u8 m_u8b5

u8 m_u8b6

u8 m_u8b7

u8 m_u8b8

u8 m_u8b9

u8 m_u8high

u8 m_u8low

struct { ... } sBits

struct { ... } sBytes

u16 u_u16Word

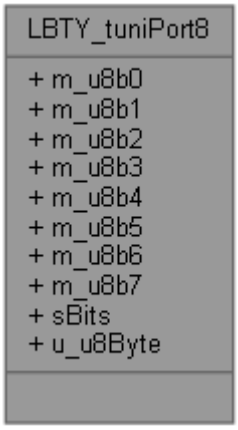
The documentation for this union was generated from the following file:

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

LBTY_tuniPort8 Union Reference

`#include <LBTY_int.h>`

Collaboration diagram for LBTY_tuniPort8:



Data Fields

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

Detailed Description

Union Byte bit by bit

Field Documentation

u8 m_u8b0

u8 m_u8b1

u8 m_u8b2

u8 m_u8b3

u8 m_u8b4

u8 m_u8b5

u8 m_u8b6

u8 m_u8b7

struct { ... } sBits

u8 u_u8Byte

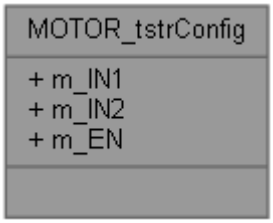
The documentation for this union was generated from the following file:

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

MOTOR_tstrConfig Struct Reference

`#include <MOTOR_int.h>`

Collaboration diagram for MOTOR_tstrConfig:



Data Fields

- GPIO_tstrPinConfig m_IN1
- GPIO_tstrPinConfig m_IN2
- GPIO_tstrPinConfig m_EN

Field Documentation

GPIO_tstrPinConfig m_EN

GPIO_tstrPinConfig m_IN1

GPIO_tstrPinConfig m_IN2

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

MOTOR_int.h

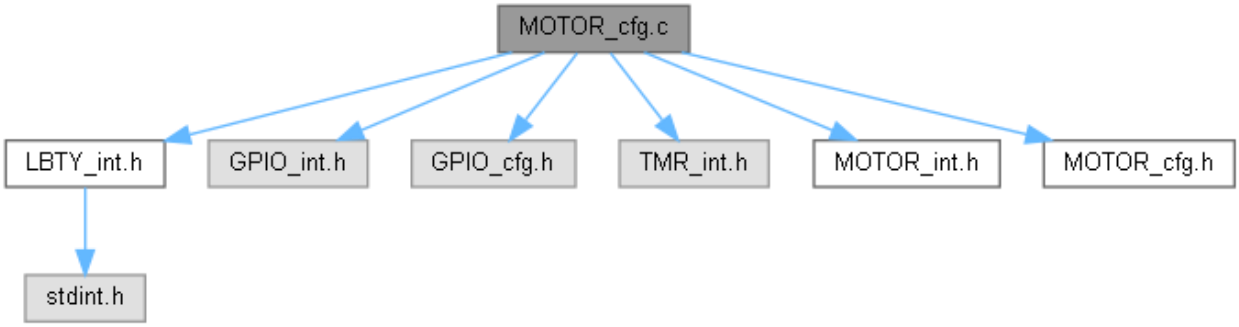
File Documentation

main.c File Reference

MOTOR_cfg.c File Reference

```
#include "LBTY_int.h"
#include "GPIO_int.h"
#include "GPIO_cfg.h"
#include "TMR_int.h"
#include "MOTOR_int.h"
#include "MOTOR_cfg.h"
```

Include dependency graph for MOTOR_cfg.c:



Functions

- void TMR0_ISR (void)
- void TMR1_ISR (void)
- void TMR2_ISR (void)
- void MOTOR_vidPWM0Init (void)
- void MOTOR_vidPWM2Init (void)
- void MOTOR_vidPWM1AInit (void)
- void MOTOR_vidPWM1BInit (void)

Variables

- const MOTOR_tstrConfig kau8MOTORConfiguration_LGB []

Function Documentation

void MOTOR_vidPWM0Init (void)

```
121                                {
122 #if defined(PWM0)
123     TMR0_vidSetConfig((TMR0_tstrConfig* const)&strTMR0_CFG);
124     TMR0_vidSetCallBack_OverFlow(TMR0_ISR);
125     PWM_vidEnable_OC0();
126 #endif
127
128 }
```

Here is the call graph for this function:

Here is the caller graph for this function:

void MOTOR_vidPWM1AInit (void)

```
137                                {
138 #if defined(PWM1)
139     strTMR1_CFG.m_TMR_OutputModeA = TMR1_FastPWM_Clear_on_Match;
```

```
140     TMR1_vidSetConfig((TMR1_tstrConfig* const)&strTMR1_CFG);
141     TMR1_vidSetCallBack_OverFlow(TMR1_ISR);
142     PWM_vidEnable_OC1x();
143 #endif
144
145 }
```

Here is the call graph for this function:

Here is the caller graph for this function:

void MOTOR_vidPWM1BInit (void)

```
146                                 {
147 #if defined(PWM1)
148     strTMR1_CFG.m_TMR_OutputModeB = TMR1_FastPWM_Clear_on_Match;
149     TMR1_vidSetConfig((TMR1_tstrConfig* const)&strTMR1_CFG);
150     TMR1_vidSetCallBack_OverFlow(TMR1_ISR);
151     PWM_vidEnable_OC1x();
152 #endif
153 }
```

Here is the call graph for this function:

Here is the caller graph for this function:

void MOTOR_vidPWM2Init (void)

```
129                                 {
130 #if defined(PWM2)
131     TMR2_vidSetConfig((TMR2_tstrConfig* const)&strTMR2_CFG);
132     TMR2_vidSetCallBack_OverFlow(TMR2_ISR);
133     PWM_vidEnable_OC2();
134 #endif
135
136 }
```

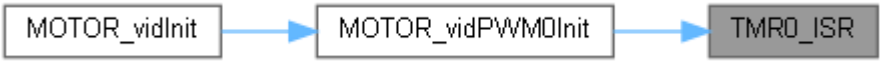
Here is the call graph for this function:

Here is the caller graph for this function:

void TMR0_ISR (void)

```
111                     {
112
113 }
```

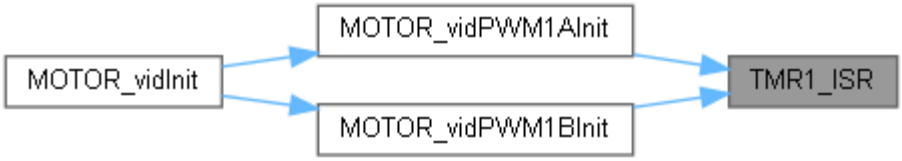
Here is the caller graph for this function:



void TMR1_ISR (void)

```
114                     {
115
116 }
```

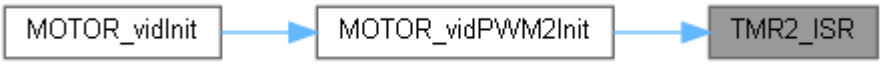
Here is the caller graph for this function:



void TMR2_ISR (void)

```
117                     {
118
119 }
```

Here is the caller graph for this function:



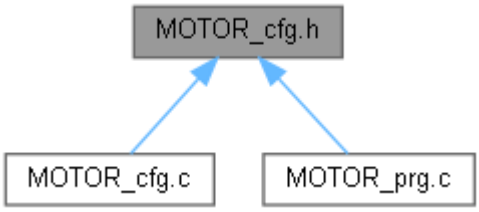
Variable Documentation

const MOTOR_tstrConfig kau8MOTORConfiguration_LGB[]

```
Initial value:= {

    {   .m_EN  = {.m_Port = MOTOR_PORT_ENA , .m_Pin = MOTOR_PIN_ENA , .m_Dir =
PIN_OUTPUT, .m_Value = PIN_Low},
        .m_IN1 = {.m_Port = MOTOR_PORT_INA1, .m_Pin = MOTOR_PIN_INA1, .m_Dir =
PIN_OUTPUT, .m_Value = PIN_Low},
        .m_IN2 = {.m_Port = MOTOR_PORT_INA2, .m_Pin = MOTOR_PIN_INA2, .m_Dir =
PIN_OUTPUT, .m_Value = PIN_Low}
    }


    ,{  .m_EN  = {.m_Port = MOTOR_PORT_ENB , .m_Pin = MOTOR_PIN_ENB , .m_Dir =
PIN_OUTPUT, .m_Value = PIN_Low},
        .m_IN1 = {.m_Port = MOTOR_PORT_INB1, .m_Pin = MOTOR_PIN_INB1, .m_Dir =
PIN_OUTPUT, .m_Value = PIN_Low},
        .m_IN2 = {.m_Port = MOTOR_PORT_INB2, .m_Pin = MOTOR_PIN_INB2, .m_Dir =
PIN_OUTPUT, .m_Value = PIN_Low}
    }


    ,{  .m_EN  = {.m_Port = MOTOR_PORT_ENC , .m_Pin = MOTOR_PIN_ENC , .m_Dir =
PIN_OUTPUT, .m_Value = PIN_Low},
        .m_IN1 = {.m_Port = MOTOR_PORT_INC1, .m_Pin = MOTOR_PIN_INC1, .m_Dir =
PIN_OUTPUT, .m_Value = PIN_Low},
        .m_IN2 = {.m_Port = MOTOR_PORT_INC2, .m_Pin = MOTOR_PIN_INC2, .m_Dir =
PIN_OUTPUT, .m_Value = PIN_Low}
    }


    ,{  .m_EN  = {.m_Port = MOTOR_PORT_END , .m_Pin = MOTOR_PIN_END , .m_Dir =
PIN_OUTPUT, .m_Value = PIN_Low},
        .m_IN1 = {.m_Port = MOTOR_PORT_IND1, .m_Pin = MOTOR_PIN_IND1, .m_Dir =
PIN_OUTPUT, .m_Value = PIN_Low},
        .m_IN2 = {.m_Port = MOTOR_PORT_IND2, .m_Pin = MOTOR_PIN_IND2, .m_Dir =
PIN_OUTPUT, .m_Value = PIN_Low}
    }

}
```

MOTOR_cfg.h File Reference

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



Macros

- #define MOTOR_CHANNELS 4u
- #define MOTOR_EN_CONTROL MOTOR_CTRL_ON_OFF
- #define MOTOR_EN_FREQ 1000u
- #define MOTOR_PORT_ENA B
- #define MOTOR_PIN_ENA 3u
- #define MOTOR_PORT_INA1 A
- #define MOTOR_PIN_INA1 0u
- #define MOTOR_PORT_INA2 A
- #define MOTOR_PIN_INA2 1u
- #define MOTOR_PORT_ENB D
- #define MOTOR_PIN_ENB 7u
- #define MOTOR_PORT_INB1 A
- #define MOTOR_PIN_INB1 2u
- #define MOTOR_PORT_INB2 A
- #define MOTOR_PIN_INB2 3u
- #define MOTOR_PORT_ENC D
- #define MOTOR_PIN_ENC 5u
- #define MOTOR_PORT_INC1 A
- #define MOTOR_PIN_INC1 4u
- #define MOTOR_PORT_INC2 A
- #define MOTOR_PIN_INC2 5u
- #define MOTOR_PORT_END D
- #define MOTOR_PIN_END 4u
- #define MOTOR_PORT_IND1 A
- #define MOTOR_PIN_IND1 6u
- #define MOTOR_PORT_IND2 A
- #define MOTOR_PIN_IND2 7u
- #define MOTOR_DELAY 5u
- #define MOTOR_NUM_DELAY 25u
- #define MOTOR_NUM_RATE 20u

Macro Definition Documentation

#define MOTOR_CHANNELS 4u

#define MOTOR_DELAY 5u

#define MOTOR_EN_CONTROL MOTOR_CTRL_ON_OFF

#define MOTOR_EN_FREQ 1000u

#define MOTOR_NUM_DELAY 25u

#define MOTOR_NUM_RATE 20u

#define MOTOR_PIN_ENA 3u

#define MOTOR_PIN_ENB 7u

#define MOTOR_PIN_ENC 5u

#define MOTOR_PIN_END 4u

#define MOTOR_PIN_INA1 0u

#define MOTOR_PIN_INA2 1u

#define MOTOR_PIN_INB1 2u

#define MOTOR_PIN_INB2 3u

#define MOTOR_PIN_INC1 4u

#define MOTOR_PIN_INC2 5u

#define MOTOR_PIN_IND1 6u

#define MOTOR_PIN_IND2 7u

#define MOTOR_PORT_ENA B

#define MOTOR_PORT_ENB D

#define MOTOR_PORT_ENC D

#define MOTOR_PORT_END D

#define MOTOR_PORT_INA1 A

#define MOTOR_PORT_INA2 A

#define MOTOR_PORT_INB1 A

```
#define MOTOR_PORT_INB2   A

#define MOTOR_PORT_INC1   A

#define MOTOR_PORT_INC2   A

#define MOTOR_PORT_IND1   A

#define MOTOR_PORT_IND2   A
```

MOTOR_cfg.h

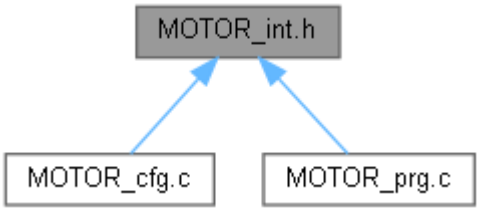
Go to the documentation of this file.

```
1 /*
******************************************************************************* */
2 /* ********************** FILE DEFINITION SECTION ************************** */
3 /* ******************************************************************************* */
4 /* File Name   : MOTOR_cfg.h                                                   */
5 /* Author      : MAAM                                                          */
6 /* Version     : v01.2                                                         */
7 /* date        : Mar 25, 2023                                                  */
8 /* ******************************************************************************* */
9 /* ************************ HEADER FILES INCLUDES ***************************  */
10 /* ******************************************************************************* */
11
12 #ifndef MOTOR_CFG_H_
13 #define MOTOR_CFG_H_
14
15 /* ******************************************************************************* */
16 /* ********************** TYPE_DEF/STRUCT/ENUM SECTION ********************** */
17 /* ******************************************************************************* */
18
19 /* ******************************************************************************* */
20 /* ************************** MACRO/DEFINE SECTION ************************** */
21 /* ******************************************************************************* */
22
23 #define MOTOR_CHANNELS      4u
24 #define MOTOR_EN_CONTROL    MOTOR_CTRL_ON_OFF
25 #define MOTOR_EN_FREQ       1000u
26
27 #if MOTOR_CHANNELS >= 1u
28
29 #define MOTOR_PORT_ENA      B
30 #define MOTOR_PIN_ENA       3u
31
32 #define MOTOR_PORT_INA1     A
33 #define MOTOR_PIN_INA1      0u
34 #define MOTOR_PORT_INA2     A
35 #define MOTOR_PIN_INA2      1u
36
37 #endif
38 #if MOTOR_CHANNELS >= 2u
39
40 #define MOTOR_PORT_ENB      D
41 #define MOTOR_PIN_ENB       7u
42
43 #define MOTOR PORT INB1     A
44 #define MOTOR_PIN_INB1      2u
45 #define MOTOR_PORT_INB2     A
46 #define MOTOR_PIN_INB2      3u
47
48 #endif
49 #if MOTOR CHANNELS >= 3u
50
51 #define MOTOR_PORT_ENC      D
52 #define MOTOR_PIN_ENC       5u
53
54 #define MOTOR PORT INC1     A
55 #define MOTOR_PIN_INC1      4u
56 #define MOTOR_PORT_INC2     A
57 #define MOTOR_PIN_INC2      5u
58
59 #endif
60 #if MOTOR CHANNELS >= 4u
61
62 #define MOTOR_PORT_END      D
63 #define MOTOR_PIN_END       4u
64
65 #define MOTOR_PORT_IND1     A
66 #define MOTOR PIN IND1      6u
67 #define MOTOR_PORT_IND2     A
68 #define MOTOR_PIN_IND2      7u
69
70 #endif
71
72 #define MOTOR_DELAY         5u
```

```
73 #define MOTOR_NUM_DELAY     25u
74 #define MOTOR_NUM_RATE      20u //40u
75
76 /* ********************************************************************************* */
77 /* ***************************** CONST SECTION ***************************** */
78 /* ********************************************************************************* */
79
80 /* ********************************************************************************* */
81 /* ***************************** VARIABLE SECTION ***************************** */
82 /* ********************************************************************************* */
83
84 /* ********************************************************************************* */
85 /* ***************************** FUNCTION SECTION ***************************** */
86 /* ********************************************************************************* */
87
88
89 #endif /* MOTOR_CFG_H_ */
90 /***************************** E N D (MOTOR_cfg.h) ******************************/
```

MOTOR_int.h File Reference

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



Data Structures

struct MOTOR_tstrConfigEnumerations

- enum MOTOR_tenuDirection { MOTOR_Stop = (u8)0u, MOTOR_ClockWith, MOTOR_AntiClockWith }

Functions

- void MOTOR_vidInit (void)
- void MOTOR_vidRun (u8 u8Motor, u16 u16Speed, u8 u8Direction)

Enumeration Type Documentation

enum MOTOR_tenuDirection

Enumerator:

MOTOR_Stop	
MOTOR_ClockWith	
MOTOR_AntiClockWith	

```
23              {
24     MOTOR_Stop = (u8)0u,
25     MOTOR_ClockWith,
26     MOTOR_AntiClockWith
27 }MOTOR_tenuDirection;
```

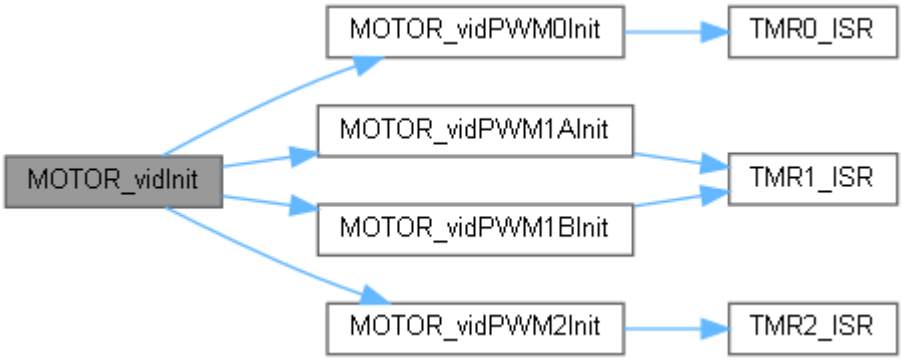
Function Documentation

void MOTOR_vidInit (void)

```
53                          {
54     for(u8 i = MOTOR_NUM ; i-- ; ){
55         GPIO_u8PinInit(kau8MOTORConfiguration_LGB[i].m_IN1);
56         GPIO_u8PinInit(kau8MOTORConfiguration_LGB[i].m_IN2);
57     }
58 #if (MOTOR_EN_CONTROL == MOTOR_CTRL_ON_OFF)
59     for(u8 i = MOTOR_NUM ; i-- ; ){
60         GPIO_u8PinInit(kau8MOTORConfiguration_LGB[i].m_EN);
61     }
62 #elif (MOTOR_EN_CONTROL == MOTOR_CTRL_PWM)
63 #if MOTOR_CHANNELS >= 1u
64     MOTOR_vidPWM0Init();
65 #endif
66 #if MOTOR_CHANNELS >= 2u
67         MOTOR_vidPWM2Init();
68 #endif
```

```
69 #if MOTOR_CHANNELS >= 3u
70         MOTOR_vidPWM1AInit();
71 #endif
72 #if MOTOR_CHANNELS >= 4u
73         MOTOR_vidPWM1BInit();
74 #endif
75 #endif
76 }
```

Here is the call graph for this function:



void MOTOR_vidRun (u8 *u8Motor*, u16 *u16Speed*, u8 *u8Direction*)

```
83                                                                  {
84     if(u8Motor >= MOTOR_NUM)        return;
85     MOTOR_tstrConfig *strTemp =
(MOTOR_tstrConfig*)&kau8MOTORConfiguration_LGB[u8Motor];
86
87 #if (MOTOR_EN_CONTROL == MOTOR_CTRL_ON_OFF)
88     GPIO_u8SetPinValue(strTemp->m_EN.m_Port, strTemp->m_EN.m_Pin, u8Direction !=
MOTOR_Stop);
89 #elif (MOTOR_EN_CONTROL == MOTOR_CTRL_PWM)
90     switch(u8Motor){
91 #if MOTOR_CHANNELS >= 1u
92         case MOTOR_EN0:     PWM_u8SetDuty_OC0(u16Speed);        break;
93 #endif
94 #if MOTOR_CHANNELS >= 2u
95         case MOTOR_EN1:     PWM_u8SetDuty_OC2(u16Speed);        break;
96 #endif
97 #if MOTOR_CHANNELS >= 3u
98         case MOTOR_EN2:     PWM_u8SetDuty_OC1A(u16Speed);       break;
99 #endif
100 #if MOTOR_CHANNELS >= 4u
101          case MOTOR_EN3:     PWM_u8SetDuty_OC1B(u16Speed);       break;
102 #endif
103         default:            break;
104     }
105 #endif
106
107     GPIO_u8SetPinValue(strTemp->m_IN1.m_Port, strTemp->m_IN1.m_Pin,
GET_BIT(u8Direction, 0));
108     GPIO_u8SetPinValue(strTemp->m_IN2.m_Port, strTemp->m_IN2.m_Pin,
GET_BIT(u8Direction, 1));
109
110 }
```

MOTOR_int.h

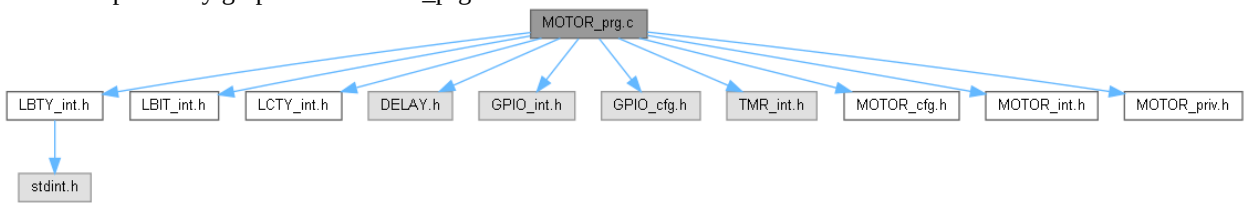
Go to the documentation of this file.

```c
/*
/* ********************************************************************************** */
/* ********************** FILE DEFINITION SECTION ************************************ */
/* ********************************************************************************** */
/* File Name    : MOTOR_int.h                                                         */
/* Author       : MAAM                                                                */
/* Version      : v01.2                                                               */
/* date         : Mar 25, 2023                                                        */
/* ********************************************************************************** */
/* ************************ HEADER FILES INCLUDES ***************************  */
/* ********************************************************************************** */

#ifndef MOTOR_INT_H_
#define MOTOR_INT_H_

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

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

typedef enum{
    MOTOR_Stop = (u8)0u,
    MOTOR_ClockWith,
    MOTOR_AntiClockWith
}MOTOR_tenuDirection;

typedef struct{
    GPIO_tstrPinConfig  m_IN1;
    GPIO_tstrPinConfig  m_IN2;
    GPIO_tstrPinConfig  m_EN;
}MOTOR_tstrConfig;

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

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

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

/* ********************************************************************************** */
/* Description :    Motor initialization                                              */
/* Input       :    void                                                              */
/* Return      :    void                                                              */
/* ********************************************************************************** */
extern void MOTOR_vidInit(void);

/* ********************************************************************************** */
/* Description :    Motor Run with Direction                                          */
/* Input       :    u8Motor, u16Speed, u8Direction                                    */
/* Return      :    void                                                              */
/* ********************************************************************************** */
void MOTOR_vidRun(u8 u8Motor, u16 u16Speed, u8 u8Direction);

#endif /* MOTOR_INT_H_ */
/***************************** E N D (MOTOR_int.h) ******************************/
```

MOTOR_prg.c File Reference

```
#include "LBTY_int.h"
#include "LBIT_int.h"
#include "LCTY_int.h"
#include "DELAY.h"
#include "GPIO_int.h"
#include "GPIO_cfg.h"
#include "TMR_int.h"
#include "MOTOR_cfg.h"
#include "MOTOR_int.h"
#include "MOTOR_priv.h"
```

Include dependency graph for MOTOR_prg.c:



Functions

- void MOTOR_vidInit (void)
- void MOTOR_vidRun (u8 u8Motor, u16 u16Speed, u8 u8Direction)

Variables

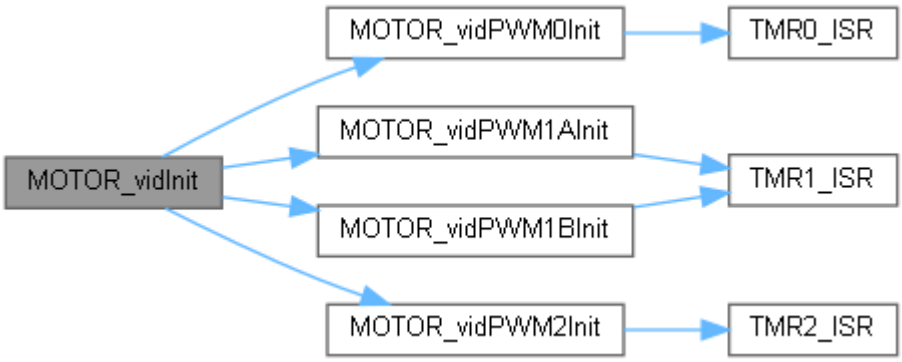
- const MOTOR_tstrConfig kau8MOTORConfiguration_LGB []

Function Documentation

void MOTOR_vidInit (void)

```
53                          {
54     for(u8 i = MOTOR_NUM ; i-- ; ){
55         GPIO_u8PinInit(kau8MOTORConfiguration_LGB[i].m_IN1);
56         GPIO_u8PinInit(kau8MOTORConfiguration_LGB[i].m_IN2);
57     }
58 #if (MOTOR_EN_CONTROL == MOTOR_CTRL_ON_OFF)
59     for(u8 i = MOTOR_NUM ; i-- ; ){
60         GPIO_u8PinInit(kau8MOTORConfiguration_LGB[i].m_EN);
61     }
62 #elif (MOTOR_EN_CONTROL == MOTOR_CTRL_PWM)
63 #if MOTOR_CHANNELS >= 1u
64     MOTOR_vidPWM0Init();
65 #endif
66 #if MOTOR_CHANNELS >= 2u
67         MOTOR_vidPWM2Init();
68 #endif
69 #if MOTOR_CHANNELS >= 3u
70         MOTOR_vidPWM1AInit();
71 #endif
72 #if MOTOR_CHANNELS >= 4u
73         MOTOR_vidPWM1BInit();
74 #endif
75 #endif
76 }
```

Here is the call graph for this function:



void MOTOR_vidRun (u8 *u8Motor*, u16 *u16Speed*, u8 *u8Direction*)

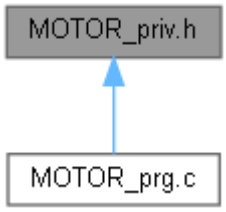
```
83                                                                  {
84     if(u8Motor >= MOTOR_NUM)        return;
85     MOTOR_tstrConfig *strTemp =
(MOTOR_tstrConfig*)&kau8MOTORConfiguration_LGB[u8Motor];
86
87 #if (MOTOR_EN_CONTROL == MOTOR_CTRL_ON_OFF)
88     GPIO_u8SetPinValue(strTemp->m_EN.m_Port, strTemp->m_EN.m_Pin, u8Direction !=
MOTOR_Stop);
89 #elif (MOTOR_EN_CONTROL == MOTOR_CTRL_PWM)
90     switch(u8Motor){
91 #if MOTOR_CHANNELS >= 1u
92         case MOTOR_EN0:     PWM_u8SetDuty_OC0(u16Speed);        break;
93 #endif
94 #if MOTOR_CHANNELS >= 2u
95         case MOTOR_EN1:     PWM_u8SetDuty_OC2(u16Speed);        break;
96 #endif
97 #if MOTOR_CHANNELS >= 3u
98         case MOTOR_EN2:     PWM_u8SetDuty_OC1A(u16Speed);       break;
99 #endif
100 #if MOTOR_CHANNELS >= 4u
101         case MOTOR_EN3:     PWM_u8SetDuty_OC1B(u16Speed);       break;
102 #endif
103         default:            break;
104     }
105 #endif
106
107     GPIO_u8SetPinValue(strTemp->m_IN1.m_Port, strTemp->m_IN1.m_Pin,
GET_BIT(u8Direction, 0));
108     GPIO_u8SetPinValue(strTemp->m_IN2.m_Port, strTemp->m_IN2.m_Pin,
GET_BIT(u8Direction, 1));
109
110 }
```

Variable Documentation

const MOTOR_tstrConfig kau8MOTORConfiguration_LGB[] `[extern]`

MOTOR_priv.h File Reference

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



Macros

- #define MOTOR_CTRL_ON_OFF 0u
- #define MOTOR_CTRL_PWM 1u

Enumerations

- enum MOTOR_tenuEN { MOTOR_NUM }

Macro Definition Documentation

#define MOTOR_CTRL_ON_OFF 0u

#define MOTOR_CTRL_PWM 1u

Enumeration Type Documentation

enum MOTOR_tenuEN

Enumerator:

MOTOR_NUM	

```
19              {
20 #if MOTOR_CHANNELS >= 1u
21     MOTOR_EN0 = (u8)0u,
22 #endif
23 #if MOTOR_CHANNELS >= 2u
24     MOTOR_EN1,
25 #endif
26 #if MOTOR_CHANNELS >= 3u
27     MOTOR_EN2,
28 #endif
29 #if MOTOR_CHANNELS >= 4u
30     MOTOR_EN3,
31 #endif
32     MOTOR_NUM
33 }MOTOR_tenuEN;
```

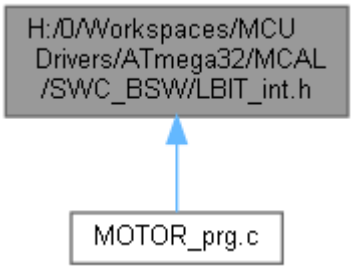
MOTOR_priv.h

Go to the documentation of this file.

```
1 /*
2 /* ********** FILE DEFINITION SECTION ********** */
3 /* ********** FILE DEFINITION SECTION ********** */
4 /* File Name    : MOTOR_priv.h                                          */
5 /* Author       : MAAM                                                  */
6 /* Version      : v01.2                                                 */
7 /* date         : Mar 25, 2023                                          */
8 /* ********** FILE DEFINITION SECTION ********** */
9 /* ********** HEADER FILES INCLUDES ********** */
10 /* ********** FILE DEFINITION SECTION ********** */
11
12 #ifndef MOTOR_PRIV_H_
13 #define MOTOR_PRIV_H_
14
15 /* ********** TYPE_DEF/STRUCT/ENUM SECTION ********** */
16 /* ********** TYPE_DEF/STRUCT/ENUM SECTION ********** */
17 /* ********** TYPE_DEF/STRUCT/ENUM SECTION ********** */
18
19 typedef enum{
20 #if MOTOR_CHANNELS >= 1u
21     MOTOR_EN0 = (u8)0u,
22 #endif
23 #if MOTOR_CHANNELS >= 2u
24     MOTOR_EN1,
25 #endif
26 #if MOTOR_CHANNELS >= 3u
27     MOTOR_EN2,
28 #endif
29 #if MOTOR_CHANNELS >= 4u
30     MOTOR_EN3,
31 #endif
32     MOTOR_NUM
33 }MOTOR_tenuEN;
34
35
36 #define MOTOR_CTRL_ON_OFF   0u
37 #define MOTOR_CTRL_PWM      1u
38
39 /* ********** MACRO/DEFINE SECTION ********** */
40 /* ********** MACRO/DEFINE SECTION ********** */
41 /* ********** MACRO/DEFINE SECTION ********** */
42
43 /* ********** CONST SECTION ********** */
44 /* ********** CONST SECTION ********** */
45 /* ********** CONST SECTION ********** */
46
47 /* ********** VARIABLE SECTION ********** */
48 /* ********** VARIABLE SECTION ********** */
49 /* ********** VARIABLE SECTION ********** */
50
51 /* ********** FUNCTION SECTION ********** */
52 /* ********** FUNCTION SECTION ********** */
53 /* ********** FUNCTION SECTION ********** */
54 #if (MOTOR_EN_CONTROL == MOTOR_CTRL_PWM)
55 void MOTOR_vidPWM0Init(void);
56 void MOTOR_vidPWM2Init(void);
57 void MOTOR_vidPWM1AInit(void);
58 void MOTOR_vidPWM1BInit(void);
59 #endif
60
61 #endif /* MOTOR_PRIV_H_ */
62 /********** E N D (MOTOR_priv.h) **********/
```

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

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



Macros

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

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

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

Macro Definition Documentation

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Bitwise Operation

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

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

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

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

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

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

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

LBIT_int.h

Go to the documentation of this file.

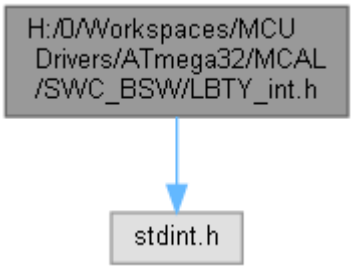
```
1 /*
**************************************************************************** */
2 /* ********************** FILE DEFINITION SECTION ************************** */
3 /* ************************************************************************** */
4 /* File Name    : LBIT_int.h                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

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

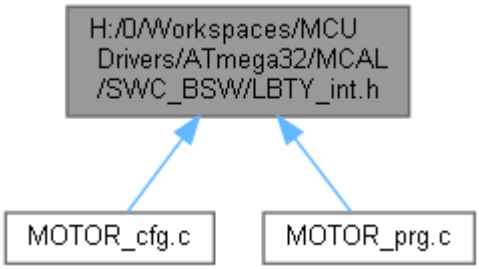
H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_BSW/LBTY_int.h File Reference

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

Include dependency graph for LBTY_int.h:



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



Data Structures

- union LBTY_tuniPort8union LBTY_tuniPort16

Macros

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

Typedefs

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

Enumerations

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

Macro Definition Documentation

#define __I volatile const

#define __IO volatile

#define __O volatile

#define LBTY_NULL ((void *) 0U)

#define LBTY_s16MAX ((u16)0x7FFF)

#define LBTY_s16MIN ((u16)0x8000)

#define LBTY_s32MAX ((u32)0x7FFFFFFFL)

#define LBTY_s32MIN ((u32)0x80000000L)

#define LBTY_s64MAX ((u64)0x7FFFFFFFFFFFFFFFLL)

#define LBTY_s64MIN ((u64)0x8000000000000000LL)

#define LBTY_s8MAX ((s8)0x7F)

#define LBTY_s8MIN ((s8)0x80)

#define LBTY_u16MAX ((u16)0xFFFFU)

#define LBTY_u16ZERO ((u16)0x0000U)

#define LBTY_u32MAX ((u32)0xFFFFFFFFUL)

#define LBTY_u32ZERO ((u32)0x00000000UL)

#define LBTY_u64MAX ((u64)0xFFFFFFFFFFFFFFFFULL)

#define LBTY_u64ZERO ((u64)0x0000000000000000ULL)

#define LBTY_u8MAX ((u8)0xFFU)

#define LBTY_u8vidNOP()

#define LBTY_u8ZERO ((u8)0x00U)

Data Types Limitation

Typedef Documentation

typedef float f32

Standard Real Decimal number

typedef double f64

typedef s16* ps16

typedef s32* ps32

typedef s64* ps64

typedef s8* ps8

Standard Pointer to Signed Byte/Word/Long_Word

typedef u16* pu16

typedef u32* pu32

typedef u64* pu64

typedef u8* pu8

Standard Pointer to Unsigned Byte/Word/Long_Word

typedef int16_t s16

typedef int32_t s32

typedef int64_t s64

typedef int8_t s8

Standard Signed Byte/Word/Long_Word

typedef uint16_t u16

typedef uint32_t u32

typedef uint64_t u64

typedef uint8_t u8

Data Types New Definitions Standard Unsigned Byte/Word/Long_Word

---

Enumeration Type Documentation

enum LBTY_tenuBoolean

Boolean type

Enumerator:

LBTY_TRUE	
LBTY_FALSE	

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

enum LBTY_tenuErrorStatus

Error Return type

Enumerator:

LBTY_OK	
LBTY_NOK	
LBTY_NULL_POINTER	
LBTY_INDEX_OUT_OF_RANGE	
LBTY_NO_MASTER_CHANNEL	
LBTY_READ_ERROR	
LBTY_WRITE_ERROR	
LBTY_UNDEFINED_ERROR	
LBTY_IN_PROGRESS	

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

enum LBTY_tenuFlagStatus

Flag Status type

Enumerator:

LBTY_RESET	
LBTY_SET	

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

LBTY_int.h

Go to the documentation of this file.

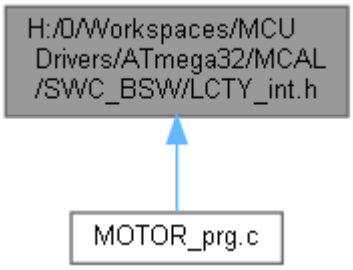
```
1 /*
2 /* ********************** FILE DEFINITION SECTION ************************** */
3 /* ************************************************************************* */
4 /* File Name   : LBTY_int.h                                                  */
5 /* Author      : MAAM                      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

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

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

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

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



Macros

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

Macro Definition Documentation

#define CTY_PACKED __attribute__((__packed__))

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

#define LCTY_CONST __attribute__((__const__))

#define LCTY_DPAGE __attribute__((dp))

#define LCTY_INLINE __attribute__((always_inline)) static inline

#define LCTY_INTERRUPT __attribute__((interrupt))

#define LCTY_NODPAGE __attribute__((nodp))

#define LCTY_PROGMEM __attribute__((__progmem__))

#define LCTY_PURE __attribute__((__pure__))

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

LCTY_int.h

Go to the documentation of this file.

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