SWC_ANA

Version v1.0
7/19/2023 6:56:00 PM

Table of Contents

Data Structure Index

Data Structures

Here are the data structures with brief descriptions:

File Index

File List

Here is a list of all files with brief descriptions:

Data Structure Documentation

ACSR_type Union Reference

: Type define of Union bit field of "Analog Comparator Control and Status"
```
#include <ANA_priv.h>
```
Collaboration diagram for ACSR_type:

```
ACSR_type

+ u_Reg
+ m_ACIS
+ m_ACIC
+ m_ACIE
+ m_ACI
+ m_ACO
+ m_ACBG
+ m_ACD
+ sBits
```
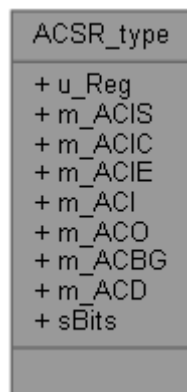
Data Fields

- u8 u_Reg
- struct {
-    __IO u8 m_ACIS: 2
-    __IO u8 m_ACIC: 1
-    __IO u8 m_ACIE: 1
-    __IO u8 m_ACI: 1
-    __I u8 m_ACO: 1
-    __IO u8 m_ACBG: 1
-    __IO u8 m_ACD: 1
- } sBits

Detailed Description

: Type define of Union bit field of "Analog Comparator Control and Status"

Type : Union **Unit** : None

Field Documentation

__IO u8 m_ACBG

Analog Comparator Bandgap Select

__IO u8 m_ACD

Analog Comparator Disable

__IO u8 m_ACI

    Analog Comparator Interrupt Flag

__IO u8 m_ACIC

    Analog Comparator Input Capture Enable

__IO u8 m_ACIE

    Analog Comparator Interrupt Enable

__IO u8 m_ACIS

    Analog Comparator Interrupt Mode Select

__I u8 m_ACO

    Analog Comparator Output

struct { ... } sBits

u8 u_Reg

    Byte

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

ANA_priv.h

ADC_type Struct Reference

: Analog Comparator Registers

```
#include <ANA_priv.h>
```
Collaboration diagram for ADC_type:
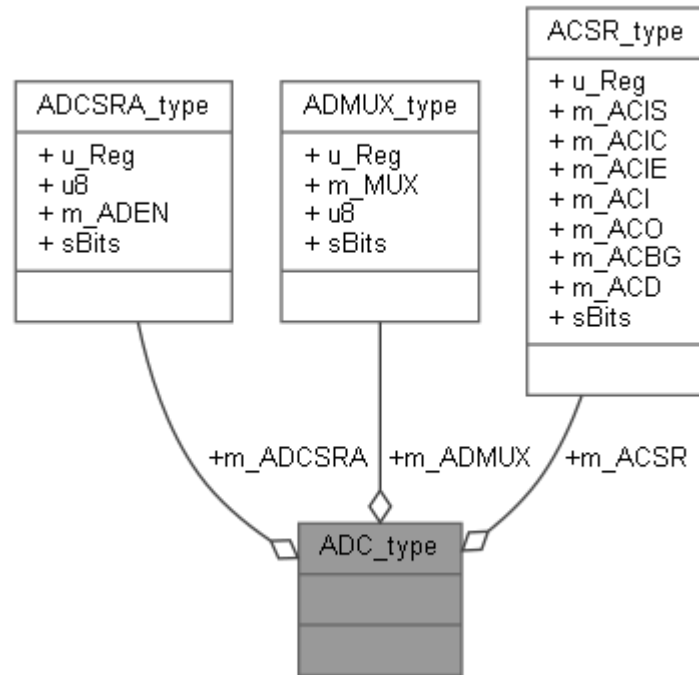
Data Fields

- __IO ADCSRA_type m_ADCSRA
- __IO ADMUX_type m_ADMUX
- __IO ACSR_type m_ACSR

Detailed Description

: Analog Comparator Registers

Type : Struct **Unit** : None

Field Documentation

__IO ACSR_type m_ACSR

Analog Comparator Control and Status Reg

__IO ADCSRA_type m_ADCSRA

ADC Control and Status Reg A

__IO ADMUX_type m_ADMUX

ADC Multiplexer Selection Reg

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

ANA_priv.h

ADCSRA_type Union Reference

: Type define of Union bit field of "ADC Control and Status Reg A"
```
#include <ANA_priv.h>
```
Collaboration diagram for ADCSRA_type:
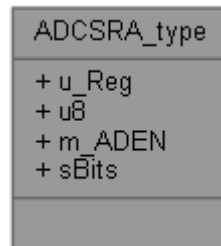
Data Fields

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

Detailed Description

: Type define of Union bit field of "ADC Control and Status Reg A"

Type : Union **Unit** : None

Field Documentation

__IO u8 m_ADEN

ADC Enable

struct { ... } sBits

__I u8

Reversed

u8 u_Reg

Byte

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

ANA_priv.h

ADMUX_type Union Reference

: Type define of Union bit field of "ADC Multiplexer Selection Reg"
```
#include <ANA_priv.h>
```
Collaboration diagram for ADMUX_type:

Data Fields

- u8 u_Reg
- struct {
- __IO u8 m_MUX: 5
- __IO u8: 3
- } sBits

Detailed Description

: Type define of Union bit field of "ADC Multiplexer Selection Reg"

Type : Union **Unit** : None

Field Documentation

__IO u8 m_MUX

Analog Channel and Gain Selection Bits

struct { ... } sBits

__IO u8

Reversed

u8 u_Reg

Byte

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

ANA_priv.h

LBTY_tuniPort16 Union Reference

`#include <LBTY_int.h>`

Collaboration diagram for LBTY_tuniPort16:
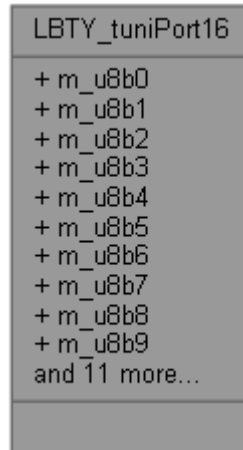
```
LBTY_tuniPort16

+ m_u8b0
+ m_u8b1
+ m_u8b2
+ m_u8b3
+ m_u8b4
+ m_u8b5
+ m_u8b6
+ m_u8b7
+ m_u8b8
+ m_u8b9
and 11 more...
```

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:
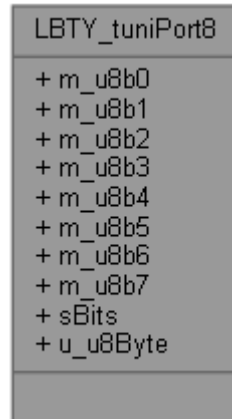
```
LBTY_tuniPort8

+ m_u8b0
+ m_u8b1
+ m_u8b2
+ m_u8b3
+ m_u8b4
+ m_u8b5
+ m_u8b6
+ m_u8b7
+ sBits
+ u_u8Byte
```

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

SFIOR_type Union Reference

: Type define of Union bit field of "Special Function I/O Register"
```
#include <ANA_priv.h>
```
Collaboration diagram for SFIOR_type:
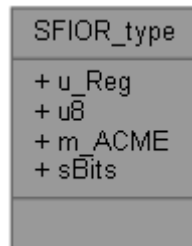
Data Fields

- u8 u_Reg
- struct {
-     __IO u8: 3
-     __IO u8 m_ACME: 1
- } sBits

Detailed Description

: Type define of Union bit field of "Special Function I/O Register"

Type : Union **Unit** : None

Field Documentation

__IO u8 m_ACME

Analog Comparator Multiplexer Enable

struct { ... } sBits

__IO u8

Reversed

u8 u_Reg

Byte

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

ANA_priv.h

File Documentation

ANA_cfg.h File Reference

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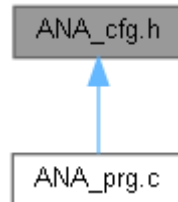
Macros

- #define ANA_INTERRUPT_STATE   LBTY_RESET
- #define ANA_INTERRUPT_MODE   ANA_OutputToggle
- #define ANA_NEG_INPUT   ANA_ADC0
- #define ANA_POS_INPUT   ANA_AIN0

Macro Definition Documentation

#define ANA_INTERRUPT_MODE   ANA_OutputToggle

#define ANA_INTERRUPT_STATE   LBTY_RESET

#define ANA_NEG_INPUT   ANA_ADC0

#define ANA_POS_INPUT   ANA_AIN0

ANA_cfg.h

```c
1 /* **************************************************************************** */
2 /* ********************** FILE DEFINITION SECTION *************************** */
3 /* **************************************************************************** */
4 /* File Name   : ANA_cfg.h                                                   */
5 /* Author      : MAAM                                                        */
6 /* Version     : v01.2                                                       */
7 /* date        : May 3, 2023                                                 */
8 /* **************************************************************************** */
9 /* ********************** HEADER FILES INCLUDES ************************** */
10 /* **************************************************************************** */
11
12 #ifndef ANA_CFG_H_
13 #define ANA_CFG_H_
14
15 /* **************************************************************************** */
16 /* ********************** TYPE_DEF/STRUCT/ENUM SECTION ********************** */
17 /* **************************************************************************** */
18
19 /* **************************************************************************** */
20 /* ************************** MACRO/DEFINE SECTION ************************** */
21 /* **************************************************************************** */
22
23 #define ANA_INTERRUPT_STATE        LBTY_RESET
24 #define ANA_INTERRUPT_MODE         ANA_OutputToggle
25
26 #define ANA_NEG_INPUT              ANA_ADC0
27 #define ANA_POS_INPUT              ANA_AIN0
28
29 /* **************************************************************************** */
30 /* *************************** CONST SECTION **************************** */
31 /* **************************************************************************** */
32
33 /* **************************************************************************** */
34 /* ************************** VARIABLE SECTION ************************** */
35 /* **************************************************************************** */
36
37 /* **************************************************************************** */
38 /* *************************** FUNCTION SECTION ************************** */
39 /* **************************************************************************** */
40
41
42 #endif /* ANA_CFG_H_ */
43 /*********************** E N D (ANA_cfg.h) ***************************/
```
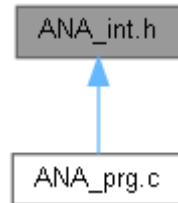
ANA_int.h File Reference

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

Enumerations

- enum ANA_tenuModeSelect { ANA_OutputToggle = (u8)0u, Reversed, ANA_OutputFallingEdge, ANA_OutputRisingEdge }
- enum ANA_tenuNegativeInput { ANA_ADC0 = (u8)0u, ANA_ADC1, ANA_ADC2, ANA_ADC3, ANA_ADC4, ANA_ADC5, ANA_ADC6, ANA_ADC7, ANA_AIN1 }
- enum ANA_tenuPositiveInput { ANA_AIN0 = (u8)0u, ANA_BANDGAP }

Functions

- void ANA_vidInit (void)
- void ANA_vidEnable (void)
- void ANA_vidDisable (void)
- void ANA_vidGetOutput (u8 *pu8Output)
- LBTY_tenuErrorStatus ANA_u8SetNegativeInput (ANA_tenuNegativeInput u8Input)
- LBTY_tenuErrorStatus ANA_u8SetPositiveInput (ANA_tenuPositiveInput u8Input)
- LBTY_tenuErrorStatus ANA_u8SetMode (ANA_tenuModeSelect u8Mode)
- void ANA_vidEnableINT (void)
- void ANA_vidDisableINT (void)
- void ANA_vidClrFlagINT (void)
- void ANA_vidSetCallBack (void(*pvidCallBack)(void))

Enumeration Type Documentation

enum ANA_tenuModeSelect

Enumerator:

ANA_OutputToggle	
Reversed	
ANA_OutputFallingEdge	
ANA_OutputRisingEdge	

```
19          {
20      ANA_OutputToggle = (u8)0u,
21      Reversed,
22      ANA_OutputFallingEdge,
23      ANA_OutputRisingEdge
24 }ANA_tenuModeSelect;
```

enum ANA_tenuNegativeInput

Enumerator:

ANA_ADC0	
ANA_ADC1	
ANA_ADC2	
ANA_ADC3	
ANA_ADC4	
ANA_ADC5	
ANA_ADC6	
ANA_ADC7	
ANA_AIN1	

```
26          {
27      ANA_ADC0 = (u8)0u,
28      ANA_ADC1,
29      ANA_ADC2,
30      ANA_ADC3,
31      ANA_ADC4,
32      ANA_ADC5,
33      ANA_ADC6,
34      ANA_ADC7,
35      ANA_AIN1
36  }ANA_tenuNegativeInput;
```

enum ANA_tenuPositiveInput

Enumerator:

ANA_AIN0	
ANA_BANDGAP	

```
38          {
39      ANA_AIN0 = (u8)0u,
40      ANA_BANDGAP
41  }ANA_tenuPositiveInput;
```

Function Documentation

LBTY_tenuErrorStatus ANA_u8SetMode (ANA_tenuModeSelect *u8Mode*)

```
154                                          {
155      LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
156
157      switch(u8Mode){
158          case ANA_OutputToggle:
159          case ANA_OutputFallingEdge:
160          case ANA_OutputRisingEdge:
161              S_ADC->m_ACSR.sBits.m_ACIS = u8Mode;
162              break;
163          default:   u8RetErrorState = LBTY_NOK;
164      }
165      return u8RetErrorState;
166  }
```

LBTY_tenuErrorStatus ANA_u8SetNegativeInput (ANA_tenuNegativeInput *u8Input*)

```
98                                          {
99      LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
100
101      switch(u8Input){
102          case ANA_ADC0:
103          case ANA_ADC1:
104          case ANA_ADC2:
105          case ANA_ADC3:
106          case ANA_ADC4:
107          case ANA_ADC5:
108          case ANA_ADC6:
109          case ANA_ADC7:
110              S_SFIOR->sBits.m_ACME        = LBTY_SET;
111              S_ADC->m_ADCSRA.sBits.m_ADEN = LBTY_RESET;
112              S_ADC->m_ADMUX.sBits.m_MUX   = u8Input;
```

```
113
114            GPIO_u8SetPinDirection(ADC_PORT, u8Input, PIN_INPUT);
115            break;
116        case ANA_AIN1:
117            S_SFIOR->sBits.m_ACME       = LBTY_SET;
118            S_ADC->m_ADCSRA.sBits.m_ADEN = LBTY_SET;
119
120            GPIO_u8SetPinDirection(ANA_PORT, ANA_PIN_AIN1, PIN_INPUT);
121            break;
122        default:    u8RetErrorState = LBTY_NOK;
123    }
124    return u8RetErrorState;
125 }
```

Here is the caller graph for this function:

LBTY_tenuErrorStatus ANA_u8SetPositiveInput (ANA_tenuPositiveInput   *u8Input*)

```
132                                                                    {
133    LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
134
135    switch(u8Input){
136        case ANA_AIN0:
137            GPIO_u8SetPinDirection(ANA_PORT, ANA_PIN_AIN0, PIN_INPUT);
138            S_ADC->m_ACSR.sBits.m_ACBG = u8Input;
139            break;
140        case ANA_BANDGAP:
141            //GPIO_u8SetPinDirection(ADC_PORT, u8PosChannel, PIN_INPUT);
142            S_ADC->m_ACSR.sBits.m_ACBG = u8Input;
143            break;
144        default:    u8RetErrorState = LBTY_NOK;
145    }
146    return u8RetErrorState;
147 }
```

Here is the caller graph for this function:

void ANA_vidClrFlagINT (void)

```
193                                {
194    S_ADC->m_ACSR.sBits.m_ACI  = LBTY_RESET;
195 }
```

void ANA_vidDisable (void)

Analog Comparator Disable

```
80                              {
81    S_ADC->m_ACSR.sBits.m_ACD  = LBTY_SET;
82 }
```

void ANA_vidDisableINT (void)

```
184                                {
185    S_ADC->m_ACSR.sBits.m_ACIE = LBTY_RESET;
186 }
```

void ANA_vidEnable (void)

Analog Comparator Disable

```
71                              {
72    S_ADC->m_ACSR.sBits.m_ACD  = LBTY_RESET;
73 }
```

void ANA_vidEnableINT (void)

```
175                                {
176    S_ADC->m_ACSR.sBits.m_ACIE  = LBTY_SET;
```

```
177 }
```

void ANA_vidGetOutput (u8 * *pu8Output*)

```
89                                              {
90      *pu8Output = S ADC->m_ACSR.sBits.m_ACO;
91 }
```

void ANA_vidInit (void)

Analog Comparator Disable

```
49                          {
50
51      ANA u8SetPositiveInput(ANA POS INPUT);
52      ANA u8SetNegativeInput(ANA NEG INPUT);
53
54      S ADC->m_ACSR.sBits.m_ACD  = LBTY RESET;
55
56      S ADC->m_ACSR.sBits.m_ACIC = LBTY RESET;
57
58      S ADC->m_ACSR.sBits.m_ACIS = ANA INTERRUPT MODE;
59      S ADC->m_ACSR.sBits.m_ACIE = ANA INTERRUPT STATE;
60      S ADC->m_ACSR.sBits.m_ACI  = LBTY RESET;
61
62 }
```
Here is the call graph for this function:

void ANA_vidSetCallBack (void(*)(void) *pvidCallBack*)

```
202                                                             {
203      pvidfunctionCallBack = pvidCallBack;
204 }
```

ANA_int.h

```c
1  /* *********************************************************************** */
2  /* ******************** FILE DEFINITION SECTION ************************** */
3  /* *********************************************************************** */
4  /* File Name    : ANA_int.h                                                */
5  /* Author       : MAAM                                                     */
6  /* Version      : v01.2                                                    */
7  /* date         : May 3, 2023                                             */
8  /* *********************************************************************** */
9  /* ********************* HEADER FILES INCLUDES *********************** */
10 /* *********************************************************************** */
11
12 #ifndef ANA_INT_H_
13 #define ANA_INT_H_
14
15 /* *********************************************************************** */
16 /* ********************* TYPE_DEF/STRUCT/ENUM SECTION ********************* */
17 /* *********************************************************************** */
18
19 typedef enum{
20     ANA_OutputToggle = (u8)0u,
21     Reversed,
22     ANA_OutputFallingEdge,
23     ANA_OutputRisingEdge
24 }ANA_tenuModeSelect;
25
26 typedef enum{
27     ANA_ADC0 = (u8)0u,
28     ANA_ADC1,
29     ANA_ADC2,
30     ANA_ADC3,
31     ANA_ADC4,
32     ANA_ADC5,
33     ANA_ADC6,
34     ANA_ADC7,
35     ANA_AIN1
36 }ANA_tenuNegativeInput;
37
38 typedef enum{
39     ANA_AIN0 = (u8)0u,
40     ANA_BANDGAP
41 }ANA_tenuPositiveInput;
42
43 /* *********************************************************************** */
44 /* ************************* MACRO/DEFINE SECTION ************************* */
45 /* *********************************************************************** */
46
47 /* *********************************************************************** */
48 /* *************************** CONST SECTION **************************** */
49 /* *********************************************************************** */
50
51 /* *********************************************************************** */
52 /* ************************** VARIABLE SECTION ************************** */
53 /* *********************************************************************** */
54
55 /* *********************************************************************** */
56 /* ************************** FUNCTION SECTION ************************** */
57 /* *********************************************************************** */
58
59 /* *********************************************************************** */
60 /* Description :    Initialization of the ANA                              */
61 /* Input       :    void                                                   */
62 /* Return      :    void                                                   */
63 /* *********************************************************************** */
64 extern void ANA_vidInit(void);
65
66 /* *********************************************************************** */
67 /* Description :    Enable ANA to be ready for conversion                  */
68 /* Input       :    void                                                   */
69 /* Return      :    void                                                   */
70 /* *********************************************************************** */
71 void ANA_vidEnable(void);
72
```

```
73 /* ********************************************************************** */
74 /* Description :   Disable ANA to be wont make further conversions       */
75 /* Input      :    void                                                  */
76 /* Return     :    void                                                  */
77 /* ********************************************************************** */
78 void ANA_vidDisable(void);
79
80 /* ********************************************************************** */
81 /* Description :   Get the Output of Comparator                          */
82 /* Input      :    pu8Output                                             */
83 /* Return     :    void                                                  */
84 /* ********************************************************************** */
85 extern void ANA_vidGetOutput(u8* pu8Output);
86
87 /* ********************************************************************** */
88 /* Description :   Set negative Input                                    */
89 /* Input      :    u8Input                                               */
90 /* Return     :    LBTY_tenuErrorStatus                                  */
91 /* ********************************************************************** */
92 extern LBTY_tenuErrorStatus ANA_u8SetNegativeInput(ANA_tenuNegativeInput u8Input);
93
94 /* ********************************************************************** */
95 /* Description :   Set positive Input                                    */
96 /* Input      :    u8Input                                               */
97 /* Return     :    LBTY_tenuErrorStatus                                  */
98 /* ********************************************************************** */
99 extern LBTY_tenuErrorStatus ANA_u8SetPositiveInput(ANA_tenuPositiveInput u8Input);
100
101 /* ********************************************************************** */
102 /* Description :    Set Interrupt Mode                                   */
103 /* Input      :    u8Mode                                                */
104 /* Return     :    LBTY_tenuErrorStatus                                  */
105 /* ********************************************************************** */
106 extern LBTY_tenuErrorStatus ANA_u8SetMode(ANA_tenuModeSelect u8Mode);
107
108
/*******************************************************************************
*****************************/
109
110 /* ********************************************************************** */
111 /* Description :    Enable ANA Interrupt                                 */
112 /* Input      :    void                                                  */
113 /* Return     :    void                                                  */
114 /* ********************************************************************** */
115 extern void ANA_vidEnableINT(void);
116
117 /* ********************************************************************** */
118 /* Description :    Disable ANA Interrupt                                */
119 /* Input      :    void                                                  */
120 /* Return     :    void                                                  */
121 /* ********************************************************************** */
122 extern void ANA_vidDisableINT(void);
123
124 /* ********************************************************************** */
125 /* Description :    Clear ANA interrupt Flag                             */
126 /* Input      :    void                                                  */
127 /* Return     :    void                                                  */
128 /* ********************************************************************** */
129 extern void ANA_vidClrFlagINT(void);
130
131 /* ********************************************************************** */
132 /* Description :    Pass the CallBack function to ANA ISR to execute     */
133 /* Input      :    void                                                  */
134 /* Return     :    LBTY_tenuErrorStatus                                  */
135 /* ********************************************************************** */
136 extern void ANA_vidSetCallBack(void (*pvidCallBack)(void));
137
138 #endif /* ANA_INT_H_ */
139 /*********************** E N D (ANA_int.h) ***************************/
```
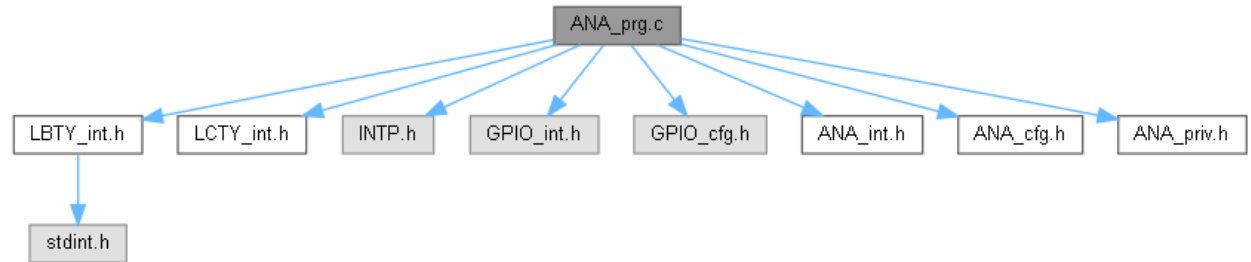
ANA_prg.c File Reference

```
#include "LBTY_int.h"
#include "LCTY_int.h"
#include "INTP.h"
#include "GPIO_int.h"
#include "GPIO_cfg.h"
#include "ANA_int.h"
#include "ANA_cfg.h"
#include "ANA_priv.h"
```

Include dependency graph for ANA_prg.c:

Functions

- void **ANA_vidInit** (void)
- void **ANA_vidEnable** (void)
- void **ANA_vidDisable** (void)
- void **ANA_vidGetOutput** (u8 *pu8Output)
- **LBTY_tenuErrorStatus** **ANA_u8SetNegativeInput** (**ANA_tenuNegativeInput** u8Input)
- **LBTY_tenuErrorStatus** **ANA_u8SetPositiveInput** (**ANA_tenuPositiveInput** u8Input)
- **LBTY_tenuErrorStatus** **ANA_u8SetMode** (**ANA_tenuModeSelect** u8Mode)
- void **ANA_vidEnableINT** (void)
- void **ANA_vidDisableINT** (void)
- void **ANA_vidClrFlagINT** (void)
- void **ANA_vidSetCallBack** (void(*pvidCallBack)(void))
- **ISR** (ANA_COMP_vect)

Variables

- void(* **pvidfunctionCallBack**)(void)

Function Documentation

LBTY_tenuErrorStatus ANA_u8SetMode (ANA_tenuModeSelect *u8Mode*)

```
154                                                  {
155     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
156
157     switch(u8Mode){
158         case ANA_OutputToggle:
159         case ANA_OutputFallingEdge:
160         case ANA_OutputRisingEdge:
161             S_ADC->m_ACSR.sBits.m_ACIS = u8Mode;
162             break;
163         default:    u8RetErrorState = LBTY_NOK;
164     }
165     return u8RetErrorState;
166 }
```

LBTY_tenuErrorStatus ANA_u8SetNegativeInput (ANA_tenuNegativeInput   *u8Input*)

```
98                                                                              {
99      LBTY tenuErrorStatus u8RetErrorState = LBTY OK;
100
101     switch(u8Input){
102         case ANA ADC0:
103         case ANA ADC1:
104         case ANA ADC2:
105         case ANA ADC3:
106         case ANA ADC4:
107         case ANA ADC5:
108         case ANA ADC6:
109         case ANA ADC7:
110             S SFIOR->sBits.m ACME        = LBTY SET;
111             S ADC->m_ADCSRA.sBits.m ADEN = LBTY RESET;
112             S ADC->m_ADMUX.sBits.m MUX   = u8Input;
113
114             GPIO u8SetPinDirection(ADC PORT, u8Input, PIN INPUT);
115             break;
116         case ANA AIN1:
117             S SFIOR->sBits.m ACME        = LBTY SET;
118             S ADC->m_ADCSRA.sBits.m ADEN = LBTY SET;
119
120             GPIO u8SetPinDirection(ANA PORT, ANA PIN AIN1, PIN INPUT);
121             break;
122         default:   u8RetErrorState = LBTY NOK;
123     }
124     return u8RetErrorState;
125 }
```

Here is the caller graph for this function:

LBTY_tenuErrorStatus ANA_u8SetPositiveInput (ANA_tenuPositiveInput   *u8Input*)

```
132                                                                              {
133     LBTY tenuErrorStatus u8RetErrorState = LBTY OK;
134
135     switch(u8Input){
136         case ANA AIN0:
137             GPIO u8SetPinDirection(ANA PORT, ANA PIN AIN0, PIN INPUT);
138             S ADC->m_ACSR.sBits.m ACBG = u8Input;
139             break;
140         case ANA BANDGAP:
141             //GPIO u8SetPinDirection(ADC PORT, u8PosChannel, PIN INPUT);
142             S ADC->m_ACSR.sBits.m ACBG = u8Input;
143             break;
144         default:   u8RetErrorState = LBTY NOK;
145     }
146     return u8RetErrorState;
147 }
```

Here is the caller graph for this function:

void ANA_vidClrFlagINT (void)

```
193                                     {
194     S ADC->m_ACSR.sBits.m ACI  = LBTY RESET;
195 }
```

void ANA_vidDisable (void)

Analog Comparator Disable

```
80                                  {
81     S ADC->m_ACSR.sBits.m ACD  = LBTY SET;
82 }
```

void ANA_vidDisableINT (void)

```
184                                         {
```

```
185      S_ADC->m_ACSR.sBits.m_ACIE = LBTY_RESET;
186 }
```

void ANA_vidEnable (void)

Analog Comparator Disable

```
71                        {
72      S_ADC->m_ACSR.sBits.m_ACD  = LBTY_RESET;
73 }
```

void ANA_vidEnableINT (void)

```
175                       {
176     S_ADC->m_ACSR.sBits.m_ACIE  = LBTY_SET;
177 }
```

void ANA_vidGetOutput (u8 *   pu8Output)

```
89                        {
90      *pu8Output = S_ADC->m_ACSR.sBits.m_ACO;
91 }
```

void ANA_vidInit (void)

Analog Comparator Disable

```
49                        {
50
51      ANA_u8SetPositiveInput(ANA_POS_INPUT);
52      ANA_u8SetNegativeInput(ANA_NEG_INPUT);
53
54      S_ADC->m_ACSR.sBits.m_ACD  = LBTY_RESET;
55
56      S_ADC->m_ACSR.sBits.m_ACIC = LBTY_RESET;
57
58      S_ADC->m_ACSR.sBits.m_ACIS = ANA_INTERRUPT_MODE;
59      S_ADC->m_ACSR.sBits.m_ACIE = ANA_INTERRUPT_STATE;
60      S_ADC->m_ACSR.sBits.m_ACI  = LBTY_RESET;
61
62 }
```

Here is the call graph for this function:

void ANA_vidSetCallBack (void(*)(void)   pvidCallBack)

```
202                                  {
203     pvidfunctionCallBack = pvidCallBack;
204 }
```

ISR (ANA_COMP_vect)

```
211                       {
212     pvidfunctionCallBack();
213 }
```

Variable Documentation

void(* pvidfunctionCallBack) (void) (void)

ANA_priv.h File Reference

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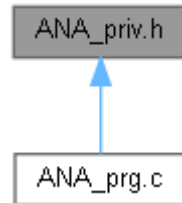
Data Structures

union ADCSRA_type: *Type define of Union bit field of "ADC Control and Status Reg A"*

union ADMUX_type: *Type define of Union bit field of "ADC Multiplexer Selection Reg"*

union ACSR_type: *Type define of Union bit field of "Analog Comparator Control and Status"*

struct ADC_type: *Analog Comparator Registers*

union SFIOR_type: *Type define of Union bit field of "Special Function I/O Register"*

Macros

- #define ADC_PORT   A
- #define ANA_PORT   B
- #define ANA_PIN_AIN0   GPIO_AIN0
- #define ANA_PIN_AIN1   GPIO_AIN1
- #define S_ADC   ((ADC_type* const)0x26U)
- #define ADCSRA   (*(volatile u8* const)0x26U)
- #define ADMUX   (*(volatile u8* const)0x27U)
- #define ACSR   (*(volatile u8* const)0x28U)
- #define S_SFIOR   ((SFIOR_type* const)0x50U)
- #define SFIOR   (*(volatile u8* const)0x50U)

Macro Definition Documentation

#define ACSR   (*(volatile u8* const)0x28U)

#define ADC_PORT   A

#define ADCSRA   (*(volatile u8* const)0x26U)

#define ADMUX   (*(volatile u8* const)0x27U)

#define ANA_PIN_AIN0   GPIO_AIN0

#define ANA_PIN_AIN1   GPIO_AIN1

#define ANA_PORT   B

#define S_ADC   ((ADC_type* const)0x26U)

Analog Digital Converter

#define S_SFIOR   ((SFIOR_type* const)0x50U)

    Special Function I/O Register

#define SFIOR   (*(volatile u8* const)0x50U)

ANA_priv.h

```c
1 /* *************************************************************************** */
2 /* ******************** FILE DEFINITION SECTION ************************** */
3 /* *************************************************************************** */
4 /* File Name    : ANA_priv.h                                               */
5 /* Author       : MAAM                                                     */
6 /* Version      : v01.2                                                    */
7 /* date         : May 3, 2023                                              */
8 /* *************************************************************************** */
9 /* ********************** HEADER FILES INCLUDES ************************** */
10 /* *************************************************************************** */
11
12 #ifndef ANA_PRIV_H_
13 #define ANA_PRIV_H_
14
15 /* *************************************************************************** */
16 /* ********************** TYPE_DEF/STRUCT/ENUM SECTION ****************** */
17 /* *************************************************************************** */
18
21 typedef union{
22     u8 u_Reg;
23     struct {
24         __I  u8         : 7;
25         __IO u8 m_ADEN  : 1;
26     }sBits;
27 }ADCSRA_type;
28
29 /***************************************************************************/
30
33 typedef union{
34     u8 u_Reg;
35     struct {
36         __IO u8 m_MUX   : 5;
37         __IO u8         : 3;
38     }sBits;
39 }ADMUX_type;
40
41 /***************************************************************************/
42
45 typedef union{
46     u8 u_Reg;
47     struct {
48         __IO u8 m_ACIS  : 2;
49         __IO u8 m_ACIC  : 1;
50         __IO u8 m_ACIE  : 1;
51         __IO u8 m_ACI   : 1;
52        __I  u8 m_ACO   : 1;
53         __IO u8 m_ACBG  : 1;
54         __IO u8 m_ACD   : 1;
55     }sBits;
56 }ACSR_type;
57
58 /***************************************************************************/
59
62 typedef struct{
63     __IO ADCSRA_type  m_ADCSRA;
64     __IO ADMUX_type   m_ADMUX;
65     __IO ACSR_type    m_ACSR;
66 }ADC_type;
67
68 /***************************************************************************/
69
72 typedef union{
73     u8 u_Reg;
74     struct {
75         __IO u8         : 3;
76         __IO u8 m_ACME  : 1;
77         __IO u8         : 4;
78     }sBits;
79 }SFIOR_type;
80
81 /* *************************************************************************** */
82 /* ********************** MACRO/DEFINE SECTION ************************** */
```

28

```
83  /* ************************************************************************ */
84
85  #define ADC_PORT                         A
86  #define ANA_PORT                         B
87  #define ANA_PIN_AIN0                     GPIO_AIN0
88  #define ANA_PIN_AIN1                     GPIO_AIN1
89
91  #define S_ADC            ((ADC_type* const)0x26U)
92  #define ADCSRA           (*(volatile u8* const)0x26U)
93  #define ADMUX            (*(volatile u8* const)0x27U)
94  #define ACSR             (*(volatile u8* const)0x28U)
95
97  #define S_SFIOR          ((SFIOR_type* const)0x50U)
98  #define SFIOR            (*(volatile u8* const)0x50U)
99
100 /* ************************************************************************ */
101 /* **************************** CONST SECTION **************************** */
102 /* ************************************************************************ */
103
104 /* ************************************************************************ */
105 /* ************************** VARIABLE SECTION ************************** */
106 /* ************************************************************************ */
107
108 /* ************************************************************************ */
109 /* ************************** FUNCTION SECTION ************************** */
110 /* ************************************************************************ */
111
112
113 #endif /* ANA_PRIV_H_ */
114 /*********************** E N D (ANA_priv.h) ***********************/
```

main.c File Reference

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

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

```c
#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

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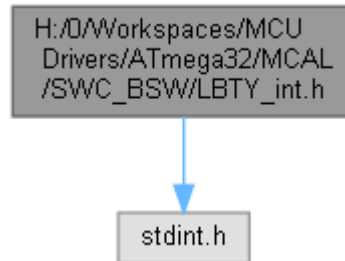
This graph shows which files directly or indirectly include this file:
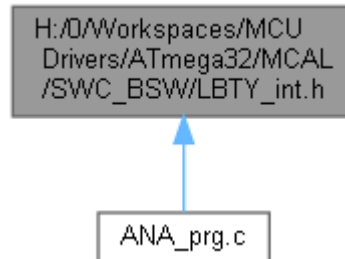
Data Structures

- union **LBTY_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)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_PO INTER	
LBTY_INDEX_O UT_OF_RANGE	
LBTY_NO_MAS TER_CHANNEL	
LBTY_READ_ER ROR	
LBTY_WRITE_E RROR	
LBTY_UNDEFIN ED_ERROR	
LBTY_IN_PROG RESS	

```
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

```
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:
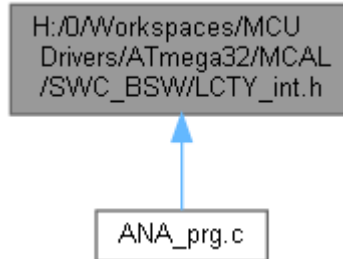
Macros

- #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

```
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) ***************************/
```