SWC_ADC

Version v1.0
7/19/2023 12:07:00 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

ACSR_type Union Reference

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

`#include <ADC_priv.h>`

Collaboration diagram for ACSR_type:

Data Fields

- u8 u_Reg
- struct {
- __IO u8: 7
- __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_ACD

Analog Comparator Disable

struct { ... } sBits

__IO u8

Reversed

u8 u_Reg

Byte

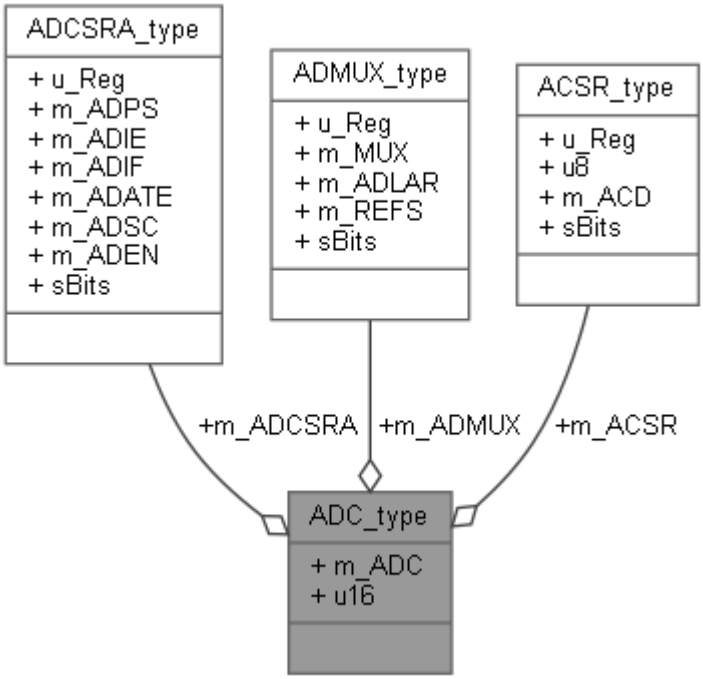
The documentation for this union was generated from the following file:

ADC_priv.h

ADC_type Struct Reference

: Analog to Digital Converter Registers

`#include <ADC_priv.h>`
Collaboration diagram for ADC_type:



Data Fields

- __I u16 m_ADC: 10
- __I u16: 6
- __IO ADCSRA_type m_ADCSRA
- __IO ADMUX_type m_ADMUX
- __IO ACSR_type m_ACSR

Detailed Description

: Analog to Digital Converter Registers

Type : Struct **Unit** : None

Field Documentation

__IO ACSR_type m_ACSR

Analog Comparator Control and Status

__I u16 **m_ADC**

ADC Read Data

__IO ADCSRA_type **m_ADCSRA**

ADC Control and Status Reg A

__IO ADMUX_type **m_ADMUX**

ADC Multiplexer Selection Reg

__I u16

Reversed

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

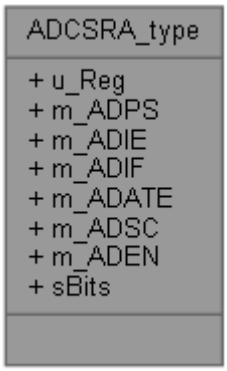
ADC_priv.h

ADCSRA_type Union Reference

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

```
#include <ADC_priv.h>
```

Collaboration diagram for ADCSRA_type:



Data Fields

- u8 u_Reg
- struct {
- __IO u8 m_ADPS: 3
- __IO u8 m_ADIE: 1
- __IO u8 m_ADIF: 1
- __IO u8 m_ADATE: 1
- __IO u8 m_ADSC: 1
- __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_ADATE

ADC Auto Trigger Enable

__IO u8 m_ADEN

ADC Enable

__IO u8 m_ADIE

ADC Interrupt Enable

__IO u8 m_ADIF

ADC Interrupt Flag

__IO u8 m_ADPS

ADC Prescaler Select Bits

__IO u8 m_ADSC

ADC Start Conversion

struct { ... } sBits

u8 u_Reg

Byte

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

ADC_priv.h

ADMUX_type Union Reference

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

```
#include <ADC_priv.h>
```

Collaboration diagram for ADMUX_type:

Data Fields

- u8 u_Reg
- struct {
- __IO u8 m_MUX: 5
- __IO u8 m_ADLAR: 1
- __IO u8 m_REFS: 2
- } sBits

Detailed Description

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

Type : Union **Unit** : None

Field Documentation

__IO u8 m_ADLAR

ADC Left Adjust Result

__IO u8 m_MUX

Analog Channel and Gain Selection Bits

__IO u8 m_REFS

Reference Selection Bits

struct { ... } sBits

u8 u_Reg

Byte

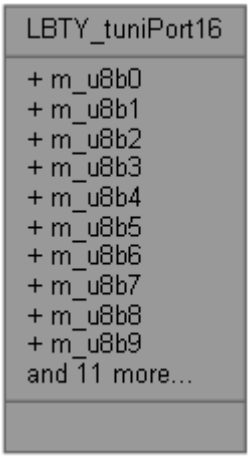
The documentation for this union was generated from the following file:

ADC_priv.h

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

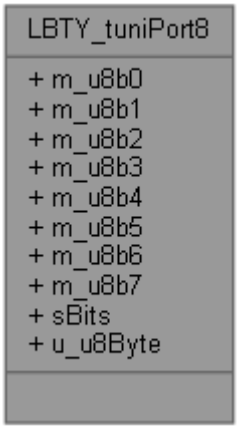
u16 u_u16Word

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:

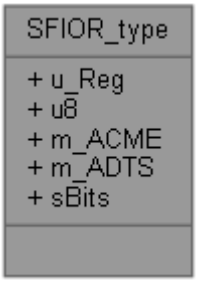
- H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_BSW/LBTY_int.h

SFIOR_type Union Reference

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

```
#include <ADC_priv.h>
```

Collaboration diagram for SFIOR_type:



Data Fields

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

__IO u8 m_ADTS

ADC Auto Trigger Source

struct { ... } sBits

__IO u8

Reversed

u8 u_Reg

Byte

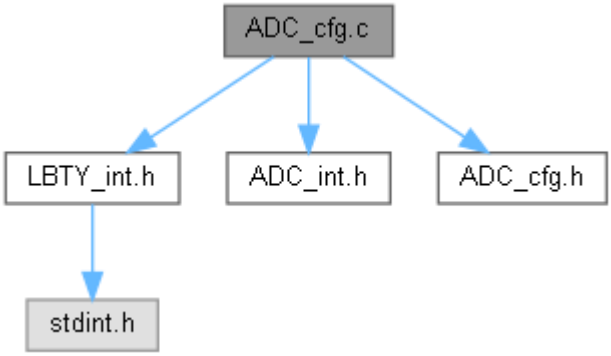
The documentation for this union was generated from the following file:
ADC_priv.h

File Documentation

ADC_cfg.c File Reference

```
#include "LBTY_int.h"
#include "ADC_int.h"
#include "ADC_cfg.h"
```

Include dependency graph for ADC_cfg.c:



Variables

- const u8 kau8ActiveChannel_LGB []

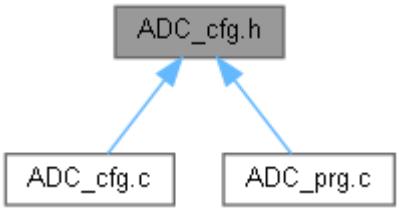
Variable Documentation

const u8 kau8ActiveChannel_LGB[]

```
Initial value:= {

        ADC_CH0

        , ADC_CH1

        , ADC_CH2

        , ADC_CH3

        , ADC_CH4

        , ADC_CH5

        , ADC_CH6

        , ADC_CH7

}
```

ADC_cfg.h File Reference

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



Macros

- #define ADC_CH0 ADC0
- #define ADC_CH1 ADC1
- #define ADC_CH2 ADC2
- #define ADC_CH3 ADC3
- #define ADC_CH4 ADC4
- #define ADC_CH5 ADC5
- #define ADC_CH6 ADC6
- #define ADC_CH7 ADC7
- #define ADC_ADJUSTMENT LBTY_RESET
- #define ADC_TRIG_SRC ADC_Free_Running_Mode
- #define ADC_INIT_STATE LBTY_SET
- #define ADC_AUTO_TRIG LBTY_RESET
- #define ADC_IRQ_STATE LBTY_RESET
- #define ADC_PRESCALER ADC_Division_Factor_2
- #define ADC_V_REF_SRC ADC_AVCC
- #define ADC_V_REF 5u
- #define ADC_READ_DELAY 26u

Variables

- const u8 kau8ActiveChannel_LGB []

Macro Definition Documentation

#define ADC_ADJUSTMENT LBTY_RESET

#define ADC_AUTO_TRIG LBTY_RESET

#define ADC_CH0 ADC0

#define ADC_CH1 ADC1

#define ADC_CH2 ADC2

#define ADC_CH3 ADC3

#define ADC_CH4 ADC4

#define ADC_CH5 ADC5

#define ADC_CH6 ADC6

#define ADC_CH7 ADC7

#define ADC_INIT_STATE LBTY_SET

ADC Control State

#define ADC_IRQ_STATE LBTY_RESET

#define ADC_PRESCALER ADC_Division_Factor_2

#define ADC_READ_DELAY 26u

#define ADC_TRIG_SRC ADC_Free_Running_Mode

SFIOR

#define ADC_V_REF 5u

#define ADC_V_REF_SRC ADC_AVCC

ADC MUX

Variable Documentation

const u8 **kau8ActiveChannel_LGB[]** `[extern]`

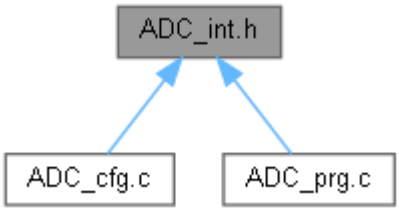
ADC_cfg.h

```
Go to the documentation of this file.1 /*
*************************************************************************** */
2 /* ********************** FILE DEFINITION SECTION ************************** */
3 /* ************************************************************************* */
4 /* File Name    : ADC_cfg.h                                                 */
5 /* Author       : MAAM                                                      */
6 /* Version      : v01.2                                                     */
7 /* date         : Mar 27, 2023                                              */
8 /* ************************************************************************* */
9 /* ************************* HEADER FILES INCLUDES ************************  */
10 /* ************************************************************************* */
11
12 #ifndef ADC_CFG_H_
13 #define ADC_CFG_H_
14
15 /* ************************************************************************* */
16 /* ********************** TYPE_DEF/STRUCT/ENUM SECTION ********************** */
17 /* ************************************************************************* */
18
19 /* ************************************************************************* */
20 /* ************************** MACRO/DEFINE SECTION ************************** */
21 /* ************************************************************************* */
22
23 #if defined(AMIT_KIT)
24
25 #define ADC_CH0                 ADC0
26 #define ADC_CH1                 ADC1
27 #define ADC_CH2                 ADC2
28 #define ADC_CH3                 ADC3
29
30 #elif defined(ETA32_KIT)
31
32 #define ADC_CH0                 ADC0
33 #define ADC_CH1                 ADC1
34
35 #elif defined(ETA32_MINI_KIT)
36
37 #define ADC CH0                 ADC0
38
39 #else
40
41 #define ADC_CH0                 ADC0
42 #define ADC_CH1                 ADC1
43 #define ADC CH2                 ADC2
44 #define ADC_CH3                 ADC3
45 #define ADC_CH4                 ADC4
46 #define ADC_CH5                 ADC5
47 #define ADC_CH6                 ADC6
48 #define ADC_CH7                 ADC7
49
50 #endif
51
52 #define ADC_ADJUSTMENT          LBTY_RESET
53
55 #define ADC TRIG SRC            ADC Free Running Mode
56
58 #define ADC_INIT_STATE          LBTY_SET
59 #define ADC_AUTO_TRIG           LBTY_RESET
60 #define ADC_IRQ_STATE           LBTY_RESET
61 #define ADC_PRESCALER           ADC_Division_Factor_2
62
64 #define ADC_V_REF_SRC           ADC_AVCC
65 #define ADC_V_REF               5u
66
67 #define ADC_READ_DELAY          26u
68
69 /* ************************************************************************* */
70 /* ***************************** CONST SECTION ***************************** */
71 /* ************************************************************************* */
72
73 extern const u8 kau8ActiveChannel LGB[];
74
75 /* ************************************************************************* */
```

```
76 /* ***************************** VARIABLE SECTION ************************** */
77 /* ************************************************************************* */
78
79 /* ************************************************************************* */
80 /* ***************************** FUNCTION SECTION ************************** */
81 /* ************************************************************************* */
82
83
84 #endif /* ADC_CFG_H_ */
85 /***************************** E N D (ADC_cfg.h) ******************************/
```

ADC_int.h File Reference

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



Enumerations

- enum ADC_tenuChannel { ADC0 = (u8)0u, ADC1, ADC2, ADC3, ADC4, ADC5, ADC6, ADC7, ADC0_ADC0_10X, ADC1_ADC0_10X, ADC0_ADC0_200X, ADC1_ADC0_200X, ADC2_ADC2_10X, ADC3_ADC2_10X, ADC2_ADC2_200X, ADC3_ADC2_200X, ADC0_ADC1_1X, ADC1_ADC1_1X, ADC2_ADC1_1X, ADC3_ADC1_1X, ADC4_ADC1_1X, ADC5_ADC1_1X, ADC6_ADC1_1X, ADC7_ADC1_1X, ADC0_ADC2_1X, ADC1_ADC2_1X, ADC2_ADC2_1X, ADC3_ADC2_1X, ADC4_ADC2_1X, ADC5_ADC2_1X, VBG_1V22, GND, ADC_ChannelMux }
- enum ADC_tenuPrescalerSelection { ADC_Division_Factor_2_DEF = (u8)0u, ADC_Division_Factor_2, ADC_Division_Factor_4, ADC_Division_Factor_8, ADC_Division_Factor_16, ADC_Division_Factor_32, ADC_Division_Factor_64, ADC_Division_Factor_128 }
- enum ADC_tenuTriggerSource { ADC_Free_Running_Mode = (u8)0u, ADC_Analog_Comparator, ADC_External_INT0, ADC_TMR0_Compare_MatchA, ADC_TMR0_Overflew, ADC_TMR1_Compare_MatchB, ADC_TMR1_Overflew, ADC_TMR1_Capture_Event }
- enum ADC_tenuRefSelection { ADC_AREF = (u8)0u, ADC_AVCC, RESERVED, ADC_INTERNAL_Vref }

Functions

- void ADC_vidInit (void)
- LBTY_tenuErrorStatus ADC_u8CofigChannel (ADC_tenuChannel u8Channel)
- void ADC_vidCalibrate (void)
- void ADC_vidEnable (void)
- void ADC_vidDisable (void)
- void ADC_vidAutoTriggerEnable (void)
- void ADC_vidAutoTriggerDisable (void)
- LBTY_tenuErrorStatus ADC_u8SetAutoTriggerSource (ADC_tenuTriggerSource u8Source)
- LBTY_tenuErrorStatus ADC_u8SetPrescaler (ADC_tenuPrescalerSelection u8Prescaler)
- LBTY_tenuErrorStatus ADC_u8SetV_REF (ADC_tenuRefSelection u8Vref)
- LBTY_tenuErrorStatus ADC_u8SetChannel (ADC_tenuChannel u8Channel)
- void ADC_vidStartConversion (void)
- void ADC_vidWaitConversion (void)
- u16 ADC_u16GetData (void)
- f32 ADC_f32GetVoltage (void)
- LBTY_tenuErrorStatus ADC_u8StartRead (ADC_tenuChannel u8Channel)
- LBTY_tenuErrorStatus ADC_u8ReadChannel (ADC_tenuChannel u8Channel, u16 *pu16ADC_Value)
- LBTY_tenuErrorStatus ADC_u8ReadConvValue (u8 u8Channel, u16 *pu16ADC_Value)
- LBTY_tenuErrorStatus ADC_u16RefreshADC (void)
- LBTY_tenuErrorStatus ADC_u16GetAll (u16 pu16ADC_Value[])
- void ADC_vidEnableINT (void)
- void ADC_vidDisableINT (void)
- void ADC_vidClrFlagINT (void)

- void ADC_vidSetCallBack (void(*pvidCallBack)(void))

Enumeration Type Documentation

enum ADC_tenuChannel

Enumerator:

ADC0	Analog Channels
ADC1	
ADC2	
ADC3	
ADC4	
ADC5	
ADC6	
ADC7	
ADC0_ADC0_10X	Analog Amplifier
ADC1_ADC0_10X	
ADC0_ADC0_200X	
ADC1_ADC0_200X	
ADC2_ADC2_10X	
ADC3_ADC2_10X	
ADC2_ADC2_200X	
ADC3_ADC2_200X	
ADC0_ADC1_1X	Analog Comparator with ADC1
ADC1_ADC1_1X	
ADC2_ADC1_1X	
ADC3_ADC1_1X	
ADC4_ADC1_1X	
ADC5_ADC1_1X	
ADC6_ADC1_1X	
ADC7_ADC1_1X	
ADC0_ADC2_1X	Analog Comparator with ADC1
ADC1_ADC2_1X	
ADC2_ADC2_1X	
ADC3_ADC2_1X	
ADC4_ADC2_1X	
ADC5_ADC2_1X	
VBG_1V22	Constant Voltage
GND	
ADC_ChannelMux	

```
19              {
21      ADC0 = (u8)0u,
22      ADC1,
23      ADC2,
24      ADC3,
25      ADC4,
26      ADC5,
27      ADC6,
28      ADC7,
30      ADC0_ADC0_10X,
31      ADC1_ADC0_10X,
32      ADC0_ADC0_200X,
33      ADC1_ADC0_200X,
34      ADC2_ADC2_10X,
35      ADC3_ADC2_10X,
36      ADC2_ADC2_200X,
37      ADC3_ADC2_200X,
39      ADC0_ADC1_1X,
40      ADC1_ADC1_1X,
41      ADC2_ADC1_1X,
42      ADC3_ADC1_1X,
43      ADC4_ADC1_1X,
44      ADC5_ADC1_1X,
45      ADC6_ADC1_1X,
46      ADC7_ADC1_1X,
48      ADC0_ADC2_1X,
49      ADC1_ADC2_1X,
50      ADC2_ADC2_1X,
51      ADC3_ADC2_1X,
52      ADC4_ADC2_1X,
53      ADC5_ADC2_1X,
55      VBG_1V22,
56      GND,
57      ADC_ChannelMux
58  }ADC_tenuChannel;       // ADC Channel Selection
```

enum ADC_tenuPrescalerSelection

Enumerator:

ADC_Division_Factor_2_DEF	
ADC_Division_Factor_2	
ADC_Division_Factor_4	
ADC_Division_Factor_8	
ADC_Division_Factor_16	
ADC_Division_Factor_32	
ADC_Division_Factor_64	
ADC_Division_Factor_128	

```
60              {
61      ADC_Division_Factor_2_DEF = (u8)0u,
62      ADC_Division_Factor_2,
63      ADC_Division_Factor_4,
64      ADC_Division_Factor_8,
65      ADC_Division_Factor_16,
66      ADC_Division_Factor_32,
67      ADC_Division_Factor_64,
68      ADC_Division_Factor_128
69  }ADC_tenuPrescalerSelection;       // ADC Prescaler Selections
```

enum ADC_tenuRefSelection

Enumerator:

ADC_AREF	Internal Vref turned off
ADC_AVCC	external capacitor at AREF pin
RESERVED	
ADC_INTERNAL_Vref	2.56v with external capacitor at AREF pin

```
82              {
83      ADC_AREF = (u8)0u,
84      ADC_AVCC,
85      RESERVED,
86      ADC_INTERNAL_Vref
87  }ADC_tenuRefSelection;      // ADC Voltage Reference Selections
```

enum ADC_tenuTriggerSource

Enumerator:

ADC_Free_Running_Mode	
ADC_Analog_Comparator	
ADC_External_INT0	
ADC_TMR0_Compare_MatchA	
ADC_TMR0_Overflew	
ADC_TMR1_Compare_MatchB	
ADC_TMR1_Overflew	
ADC_TMR1_Capture_Event	

```
71              {
72      ADC_Free_Running_Mode = (u8)0u,
73      ADC_Analog_Comparator,
74      ADC_External_INT0,
75      ADC_TMR0_Compare_MatchA,
76      ADC_TMR0_Overflew,
77      ADC_TMR1_Compare_MatchB,
78      ADC_TMR1_Overflew,
79      ADC_TMR1_Capture_Event
80  }ADC_tenuTriggerSource;      // ADC Auto Trigger Source
```

Function Documentation

f32 ADC_f32GetVoltage (void)

```
292                                 {
293      return (f32)ADC_u16GetData() * f32V_REF / ADC_MAX;
294  }
```

Here is the call graph for this function:

LBTY_tenuErrorStatus ADC_u16GetAll (u16 *pu16ADC_Value*[])

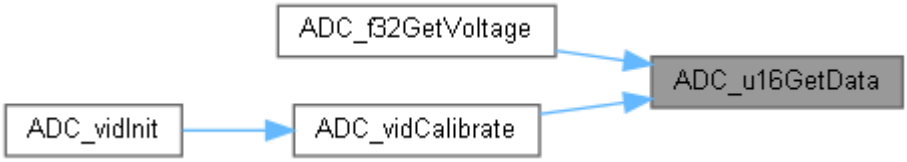
Clear complete flag by writing logic one

```
396                                                                 {
397     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
398
399     if(u8ConvDone_GLB == LBTY_SET){
400         S_ADC->m_ADCSRA.sBits.m_ADIE  = LBTY_RESET;
401         S_ADC->m_ADCSRA.sBits.m_ADIF  = LBTY_SET;
402         for(u8 i = 0 ; i<ADC_Num ; i++){
403             pu16ADC_Value[i] = au8ChannelValue_LGB[i];
404         }
405     }else{
406         u8RetErrorState = LBTY_IN_PROGRESS;
407     }
408     return u8RetErrorState;
409 }
```

u16 ADC_u16GetData (void)

```
283                             {
284     return (u16)S_ADC->m_ADC;
285 }
```

Here is the caller graph for this function:



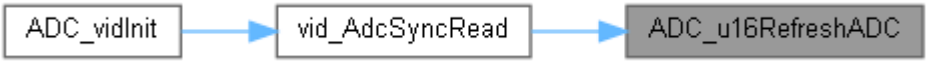
LBTY_tenuErrorStatus ADC_u16RefreshADC (void)

Clear complete flag by writing logic one

```
377                                             {
378     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
379
380     if(u8ConvDone_GLB == LBTY_SET){
381         u8ConvDone_GLB = LBTY_RESET;
382         S_ADC->m_ADCSRA.sBits.m_ADIE  = LBTY_SET;
383         S_ADC->m_ADCSRA.sBits.m_ADIF  = LBTY_SET;
384         u8RetErrorState = ADC_u8StartRead(*kau8ActiveChannel_LGB);
385     }else{
386         u8RetErrorState = LBTY_IN_PROGRESS;
387     }
388     return u8RetErrorState;
389 }
```

Here is the call graph for this function:

Here is the caller graph for this function:



LBTY_tenuErrorStatus ADC_u8CofigChannel (ADC_tenuChannel *u8Channel*)

```
110                                                                 {
111     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
112
113     if(IS_CHANNEL(u8Channel)){
114         u8RetErrorState = GPIO_u8SetPinDirection(ADC_PORT, u8Channel,
PIN_INPUT);
115     }else{
116         u8RetErrorState = LBTY_NOK;
117     }
118     return u8RetErrorState;
119 }
```

Here is the caller graph for this function:

LBTY_tenuErrorStatus ADC_u8ReadChannel (ADC_tenuChannel *u8Channel*, u16 * *pu16ADC_Value*)

```
323
{
324     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
325
326     if(pu16ADC_Value == LBTY_NULL){
327         u8RetErrorState = LBTY_NULL_POINTER;
328     }else{
329         if((u8)u8Channel < ADC_ChannelMux){
330             S_ADC->m_ADMUX.sBits.m_MUX = u8Channel;
331         }else{
332             u8RetErrorState = LBTY_NOK;
333         }
334
335         S_ADC->m_ADCSRA.sBits.m_ADSC = LBTY_SET;
336
337         while(S_ADC->m_ADCSRA.sBits.m_ADSC);
338
339         //vidMyDelay_ms(ADC_READ_DELAY);
340
341         //while(!S_ADC->m_ADCSRA.sBits.m_ADIF);
S_ADC->m_ADCSRA.sBits.m_ADIF = LBTY_RESET;
342
343         *pu16ADC_Value = (u16)S_ADC->m_ADC;
344     }
345     return u8RetErrorState;
346 }
```

LBTY_tenuErrorStatus ADC_u8ReadConvValue (u8 *u8Channel*, u16 * *pu16ADC_Value*)

```
354                                                                                 {
355     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
356
357     if(pu16ADC_Value == LBTY_NULL){
358         u8RetErrorState = LBTY_NULL_POINTER;
359     }else{
360         for(u8 i = 0 ; i<ADC_Num ; i++){
361             if(u8Channel == kau8ActiveChannel_LGB[i]){
362                 *pu16ADC_Value = au8ChannelValue_LGB[i];
363                 break;
364             }else{
365                 u8RetErrorState = LBTY_NOK;
366             }
367         }
368     }
369     return u8RetErrorState;
370 }
```

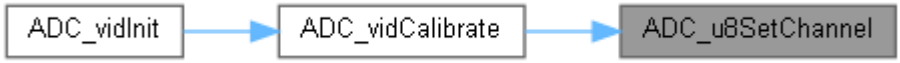
LBTY_tenuErrorStatus ADC_u8SetAutoTriggerSource (ADC_tenuTriggerSource *u8Source*)

```
178
{
179     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
180     switch(u8Source){
181         case ADC_Free_Running_Mode:
182         case ADC_Analog_Comparator:
183         case ADC_External_INT0:
184         case ADC_TMR0_Compare_MatchA:
185         case ADC_TMR0_Overflew:
186         case ADC_TMR1_Compare_MatchB:
187         case ADC_TMR1_Overflew:
188         case ADC_TMR1_Capture_Event:
189             S_SFIOR->sBits.m_ADTS  = u8Source;
190             break;
191         default:
192             u8RetErrorState = LBTY_NOK;
193     }
194
195     return u8RetErrorState;
196 }
```

LBTY_tenuErrorStatus ADC_u8SetChannel (ADC_tenuChannel *u8Channel*)

```
248                                                                 {
249     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
250
251     if((u8)u8Channel < ADC_ChannelMux){
252         S_ADC->m_ADMUX.sBits.m_MUX = u8Channel;
253     }else{
254         u8RetErrorState = LBTY_NOK;
255     }
256
257     return u8RetErrorState;
258 }
```

Here is the caller graph for this function:



LBTY_tenuErrorStatus ADC_u8SetPrescaler (ADC_tenuPrescalerSelection *u8Prescaler*)

```
203
{
204     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
205     switch(u8Prescaler){
206         case ADC_Division_Factor_2_DEF:
207         case ADC_Division_Factor_2:
208         case ADC_Division_Factor_4:
209         case ADC_Division_Factor_8:
210         case ADC_Division_Factor_16:
211         case ADC_Division_Factor_32:
212         case ADC_Division_Factor_64:
213         case ADC_Division_Factor_128:
214             S_ADC->m_ADCSRA.sBits.m_ADPS  = u8Prescaler;
215             break;
216         default:
217             u8RetErrorState = LBTY_NOK;
218     }
219
220     return u8RetErrorState;
221 }
```

LBTY_tenuErrorStatus ADC_u8SetV_REF (ADC_tenuRefSelection *u8Vref*)

```
228                                                                 {
229     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
230     switch(u8Vref){
231         case ADC_AREF:
232         case ADC_AVCC:
233         case ADC_INTERNAL_Vref:
234             S_ADC->m_ADMUX.sBits.m_REFS  = u8Vref;
235             break;
236         default:
237             u8RetErrorState = LBTY_NOK;
238     }
239
240     return u8RetErrorState;
241 }
```

LBTY_tenuErrorStatus ADC_u8StartRead (ADC_tenuChannel *u8Channel*)

```
303                                                                 {
304     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
305
306     if((u8)u8Channel < ADC_ChannelMux){
307         S_ADC->m_ADMUX.sBits.m_MUX = u8Channel;
308     }else{
309         u8RetErrorState = LBTY_NOK;
310     }
311
312     S_ADC->m_ADCSRA.sBits.m_ADSC = LBTY_SET;
313
314     return u8RetErrorState;
315 }
```

Here is the caller graph for this function:

void ADC_vidAutoTriggerDisable (void)

```
169                                     {
170     S_ADC->m_ADCSRA.sBits.m_ADATE = LBTY_RESET;
171 }
```

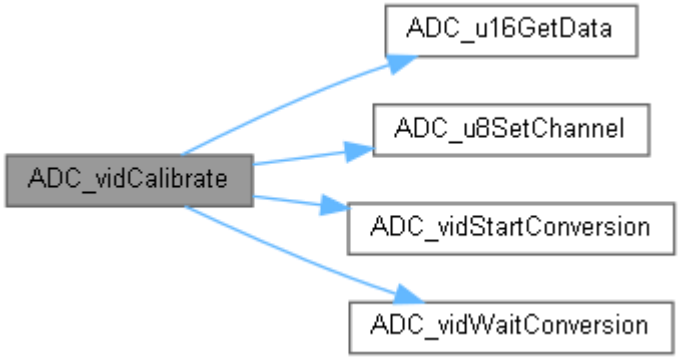
void ADC_vidAutoTriggerEnable (void)

```
160                                     {
161     S_ADC->m_ADCSRA.sBits.m_ADATE = LBTY_RESET;
162 }
```

void ADC_vidCalibrate (void)

```
126                                 {
127     ADC_u8SetChannel(VBG_1V22);
128     ADC_vidStartConversion();
129     ADC_vidWaitConversion();
130 
131     f32V_REF = (f32)(ADC_VBG_1V22 * ADC_MAX) / ADC_u16GetData();
132     ADC_u8SetChannel(ADC0);
133 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

void ADC_vidClrFlagINT (void)

Clear complete flag by writing logic one

```
436                                 {
437     S_ADC->m_ADCSRA.sBits.m_ADIF  = LBTY_SET;
438 }
```

void ADC_vidDisable (void)

```
151                                 {
152     S_ADC->m_ADCSRA.sBits.m_ADEN  = LBTY_RESET;
153 }
```

void ADC_vidDisableINT (void)

```
427                                 {
428     S_ADC->m_ADCSRA.sBits.m_ADIE  = LBTY_RESET;
429 }
```

void ADC_vidEnable (void)

```
142                                 {
143     S_ADC->m_ADCSRA.sBits.m_ADEN  = LBTY_SET;
144 }
```

void ADC_vidEnableINT (void)

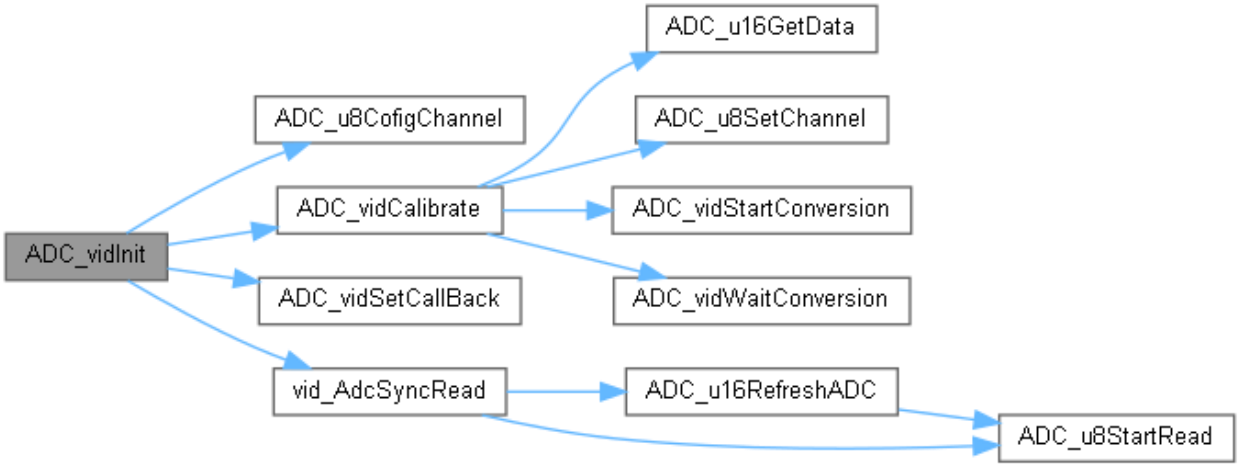
```
418                                 {
419     S_ADC->m_ADCSRA.sBits.m_ADIE  = LBTY_SET;
420 }
```

void ADC_vidInit (void)

Clear complete flag by writing logic one

```
75                          {
76
77     for(u8 i = ADC_Num; i-- ; ){
78         ADC_u8CofigChannel(kau8ActiveChannel_LGB[i]);
79     }
80
81     S_SFIOR->sBits.m_ADTS = ADC_TRIG_SRC;
82
83     S_ADC->m_ADCSRA.sBits.m_ADEN  = LBTY_SET;
84     S_ADC->m_ADCSRA.sBits.m_ADSC  = LBTY_RESET;
85     S_ADC->m_ADCSRA.sBits.m_ADATE = ADC_AUTO_TRIG;
86     S_ADC->m_ADCSRA.sBits.m_ADIE  = ADC_IRQ_STATE;
87     S_ADC->m_ADCSRA.sBits.m_ADIF  = LBTY_SET;
88     S_ADC->m_ADCSRA.sBits.m_ADPS  = ADC_PRESCALER;
89     ADC_vidSetCallBack(vid_AdcSyncRead);
90
91     S_ADC->m_ADMUX.sBits.m_REFS  = ADC_V_REF_SRC;
92     S_ADC->m_ADMUX.sBits.m_ADLAR = ADC_ADJUSTMENT;
93     S_ADC->m_ADMUX.sBits.m_MUX   = ADC0;
94
95     // first conversion will take 25 ADC clock cycles instead of the normal 13.
96     S_ADC->m_ADCSRA.sBits.m_ADSC  = LBTY_SET;
97     while(S_ADC->m_ADCSRA.sBits.m_ADSC);
98
99     ADC_vidCalibrate();
100
101     S_ADC->m_ADCSRA.sBits.m_ADEN  = ADC_INIT_STATE;
102
103 }
```

Here is the call graph for this function:



void ADC_vidSetCallBack (void(*)(void) *pvidCallBack*)

```
445                                                 {
446     pvidFunctionCallBack = pvidCallBack;
447 }
```

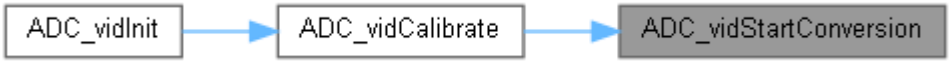
Here is the caller graph for this function:

void ADC_vidStartConversion (void)

```
265                                 {
266     S_ADC->m_ADCSRA.sBits.m_ADSC = LBTY_SET;
```

```
267 }
```

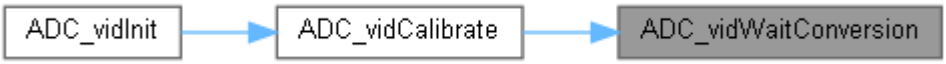
Here is the caller graph for this function:



void ADC_vidWaitConversion (void)

```
274                                 {
275     while(S_ADC->m_ADCSRA.sBits.m_ADSC);
276 }
```

Here is the caller graph for this function:



ADC_int.h

Go to the documentation of this file.

```
1 /*
2 /* ********************** FILE DEFINITION SECTION ************************** */
3 /* ************************************************************************** */
4 /* File Name    : ADC_int.h                                                   */
5 /* Author       : MAAM                                                        */
6 /* Version      : v01.2                                                       */
7 /* date         : Mar 27, 2023                                                */
8 /* ************************************************************************** */
9 /* ************************ HEADER FILES INCLUDES ************************** */
10 /* ************************************************************************** */
11
12 #ifndef ADC_INT_H_
13 #define ADC_INT_H_
14
15 /* ************************************************************************** */
16 /* ********************** TYPE_DEF/STRUCT/ENUM SECTION ********************** */
17 /* ************************************************************************** */
18
19 typedef enum{
21     ADC0 = (u8)0u,
22     ADC1,
23     ADC2,
24     ADC3,
25     ADC4,
26     ADC5,
27     ADC6,
28     ADC7,
30     ADC0_ADC0_10X,
31     ADC1_ADC0_10X,
32     ADC0_ADC0_200X,
33     ADC1_ADC0_200X,
34     ADC2_ADC2_10X,
35     ADC3_ADC2_10X,
36     ADC2_ADC2_200X,
37     ADC3_ADC2_200X,
39     ADC0_ADC1_1X,
40     ADC1_ADC1_1X,
41     ADC2_ADC1_1X,
42     ADC3_ADC1_1X,
43     ADC4_ADC1_1X,
44     ADC5_ADC1_1X,
45     ADC6_ADC1_1X,
46     ADC7_ADC1_1X,
48     ADC0_ADC2_1X,
49     ADC1_ADC2_1X,
50     ADC2_ADC2_1X,
51     ADC3_ADC2_1X,
52     ADC4_ADC2_1X,
53     ADC5_ADC2_1X,
55     VBG_1V22,
56     GND,
57     ADC_ChannelMux
58 }ADC_tenuChannel;      // ADC Channel Selection
59
60 typedef enum{
61     ADC_Division_Factor_2_DEF = (u8)0u,
62     ADC_Division_Factor_2,
63     ADC_Division_Factor_4,
64     ADC_Division_Factor_8,
65     ADC_Division_Factor_16,
66     ADC_Division_Factor_32,
67     ADC_Division_Factor_64,
68     ADC_Division_Factor_128
69 }ADC_tenuPrescalerSelection;      // ADC Prescaler Selections
70
71 typedef enum{
72     ADC_Free_Running_Mode = (u8)0u,
73     ADC_Analog_Comparator,
74     ADC_External_INT0,
75     ADC_TMR0_Compare_MatchA,
76     ADC_TMR0_Overflew,
77     ADC_TMR1_Compare_MatchB,
```

```
78      ADC_TMR1_Overflew,
79      ADC_TMR1_Capture_Event
80  }ADC_tenuTriggerSource;       // ADC Auto Trigger Source
81
82  typedef enum{
83      ADC_AREF = (u8)0u,
84      ADC_AVCC,
85      RESERVED,
86      ADC_INTERNAL_Vref
87  }ADC_tenuRefSelection;       // ADC Voltage Reference Selections
88
89  /* ********************************************************************************** */
90  /* ************************* MACRO/DEFINE SECTION ************************* */
91  /* ********************************************************************************** */
92
93  /* ********************************************************************************** */
94  /* ***************************** CONST SECTION ***************************** */
95  /* ********************************************************************************** */
96
97  /* ********************************************************************************** */
98  /* ***************************** VARIABLE SECTION ************************** */
99  /* ********************************************************************************** */
100
101 /* ********************************************************************************** */
102 /* ***************************** FUNCTION SECTION ************************** */
103 /* ********************************************************************************** */
104
105 /* ********************************************************************************** */
106 /* Description :    Initialization of the ADC                                         */
107 /* Input       :    void                                                              */
108 /* Return      :    void                                                              */
109 /* ********************************************************************************** */
110 extern void ADC_vidInit(void);
111
112 /* ********************************************************************************** */
113 /* Description :    Configuration of the Channel                                      */
114 /* Input       :    u8Channel                                                         */
115 /* Return      :    LBTY_tenuErrorStatus                                              */
116 /* ********************************************************************************** */
117 extern LBTY_tenuErrorStatus ADC_u8CofigChannel(ADC_tenuChannel u8Channel);
118
119 /* ********************************************************************************** */
120 /* Description :    Calibrate ADC Voltage                                             */
121 /* Input       :    void                                                              */
122 /* Return      :    void                                                              */
123 /* ********************************************************************************** */
124 extern void ADC_vidCalibrate(void);
125
126 /* ********************************************************************************** */
127 /* Description :    Enable ADC to be ready for conversion                             */
128 /* Input       :    void                                                              */
129 /* Return      :    void                                                              */
130 /* ********************************************************************************** */
131 extern void ADC_vidEnable(void);
132
133 /* ********************************************************************************** */
134 /* Description :    Disable ADC to be wont make further conversions                   */
135 /* Input       :    void                                                              */
136 /* Return      :    void                                                              */
137 /* ********************************************************************************** */
138 extern void ADC_vidDisable(void);
139
140 /* ********************************************************************************** */
141 /* Description :    Enable ADC Auto Trigger                                           */
142 /* Input       :    void                                                              */
143 /* Return      :    void                                                              */
144 /* ********************************************************************************** */
145 extern void ADC_vidAutoTriggerEnable(void);
146
147 /* ********************************************************************************** */
148 /* Description :    Disable ADC Auto Trigger                                          */
149 /* Input       :    void                                                              */
150 /* Return      :    void                                                              */
151 /* ********************************************************************************** */
152 extern void ADC_vidAutoTriggerDisable(void);
153
154 /* ********************************************************************************** */
```

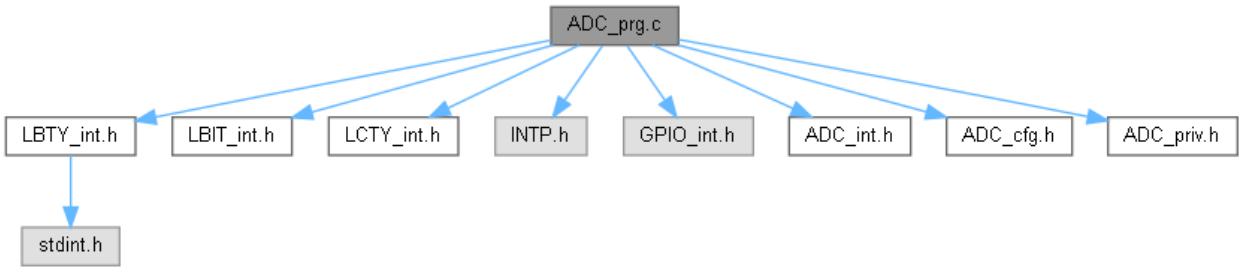
```
155 /* Description :    Set Auto Trigger Source                                   */
156 /* Input       :    u8Source                                                  */
157 /* Return      :    LBTY_tenuErrorStatus                                      */
158 /* ************************************************************************** */
159 extern LBTY_tenuErrorStatus ADC_u8SetAutoTriggerSource(ADC_tenuTriggerSource
u8Source);
160
161 /* ************************************************************************** */
162 /* Description :    Set ADC Prescaler                                         */
163 /* Input       :    u8Prescaler                                               */
164 /* Return      :    LBTY_tenuErrorStatus                                      */
165 /* ************************************************************************** */
166 extern LBTY_tenuErrorStatus ADC_u8SetPrescaler(ADC_tenuPrescalerSelection
u8Prescaler);
167
168 /* ************************************************************************** */
169 /* Description :    Set V_Ref                                                 */
170 /* Input       :    u8Vref                                                    */
171 /* Return      :    LBTY_tenuErrorStatus                                      */
172 /* ************************************************************************** */
173 extern LBTY_tenuErrorStatus ADC_u8SetV_REF(ADC_tenuRefSelection u8Vref);
174
175 /* ************************************************************************** */
176 /* Description :    Set Channel                                               */
177 /* Input       :    u8Channel                                                 */
178 /* Return      :    LBTY_tenuErrorStatus                                      */
179 /* ************************************************************************** */
180 extern LBTY_tenuErrorStatus ADC_u8SetChannel(ADC_tenuChannel u8Channel);
181
182 /* ************************************************************************** */
183 /* Description :    Start conversion                                          */
184 /* Input       :    void                                                      */
185 /* Return      :    void                                                      */
186 /* ************************************************************************** */
187 extern void ADC_vidStartConversion(void);
188
189 /* ************************************************************************** */
190 /* Description :    wait conversion done                                      */
191 /* Input       :    void                                                      */
192 /* Return      :    void                                                      */
193 /* ************************************************************************** */
194 extern void ADC_vidWaitConversion(void);
195
196 /* ************************************************************************** */
197 /* Description :    Get ADC Read                                              */
198 /* Input       :    void                                                      */
199 /* Return      :    u16                                                       */
200 /* ************************************************************************** */
201 extern u16 ADC_u16GetData(void);
202
203 /* ************************************************************************** */
204 /* Description :    Get ADC Read V                                            */
205 /* Input       :    void                                                      */
206 /* Return      :    f32                                                       */
207 /* ************************************************************************** */
208 extern f32 ADC_f32GetVoltage(void);
209
210 /* ************************************************************************** */
211 /* Description :    start ADC Read of the channel                             */
212 /* Input       :    u8Channel                                                 */
213 /* Return      :    LBTY tenuErrorStatus                                      */
214 /* ************************************************************************** */
215 extern LBTY_tenuErrorStatus ADC_u8StartRead(ADC_tenuChannel u8Channel);
216
217 /* ************************************************************************** */
218 /* Description :    Get the ADC Read of the channel                           */
219 /* Input       :    u8Channel                                                 */
220 /* Input/Output:    pu16ADC_Value                                             */
221 /* Return      :    LBTY_tenuErrorStatus                                      */
222 /* ************************************************************************** */
223 extern LBTY_tenuErrorStatus ADC_u8ReadChannel(ADC_tenuChannel u8Channel, u16*
pu16ADC Value);
224
225 /* ************************************************************************** */
226 /* Description :    Get the ADC Read from Conversion Array                    */
227 /* Input       :    u8Channel                                                 */
228 /* Input/Output:    pu16ADC_Value                                             */
```

```
229 /* Return      :    LBTY_tenuErrorStatus                                    */
230 /* ************************************************************************ */
231 extern LBTY_tenuErrorStatus ADC_u8ReadConvValue(u8 u8Channel, u16* pu16ADC_Value);
232
233 /* ************************************************************************ */
234 /* Description :    Start the ADC Interrupt Conversion                      */
235 /* Input       :    void                                                    */
236 /* Return      :    LBTY_tenuErrorStatus                                    */
237 /* ************************************************************************ */
238 extern LBTY_tenuErrorStatus ADC_u16RefreshADC(void);
239
240 /* ************************************************************************ */
241 /* Description :    Get the ADC Interrupt Conversion                        */
242 /* Input       :    pu16ADC_Value                                           */
243 /* Return      :    LBTY_tenuErrorStatus                                    */
244 /* ************************************************************************ */
245 extern LBTY_tenuErrorStatus ADC_u16GetAll(u16 pu16ADC_Value[]);
246
247
/*******************************************************************************
*****************************/
248
249 /* ************************************************************************ */
250 /* Description :    Enable ADC Interrupt                                    */
251 /* Input       :    void                                                    */
252 /* Return      :    void                                                    */
253 /* ************************************************************************ */
254 extern void ADC_vidEnableINT(void);
255
256 /* ************************************************************************ */
257 /* Description :    Disable ADC Interrupt                                   */
258 /* Input       :    void                                                    */
259 /* Return      :    void                                                    */
260 /* ************************************************************************ */
261 extern void ADC_vidDisableINT(void);
262
263 /* ************************************************************************ */
264 /* Description :    Clear ADC interrupt Flag                                */
265 /* Input       :    void                                                    */
266 /* Return      :    void                                                    */
267 /* ************************************************************************ */
268 extern void ADC_vidClrFlagINT(void);
269
270 /* ************************************************************************ */
271 /* Description :    Pass the CallBack function to TMR ISR to execute        */
272 /* Input       :    void                                                    */
273 /* Return      :    void                                                    */
274 /* ************************************************************************ */
275 extern void ADC_vidSetCallBack(void (*pvidCallBack)(void));
276
277
278 #endif /* ADC_INT_H_ */
279 /*************************** E N D (ADC_int.h) ****************************/
```

ADC_prg.c File Reference

```
#include "LBTY_int.h"
#include "LBIT_int.h"
#include "LCTY_int.h"
#include "INTP.h"
#include "GPIO_int.h"
#include "ADC_int.h"
#include "ADC_cfg.h"
#include "ADC_priv.h"
```

Include dependency graph for ADC_prg.c:



Functions

- static void vid_AdcSyncRead (void)
- void ADC_vidInit (void)
- LBTY_tenuErrorStatus ADC_u8CofigChannel (ADC_tenuChannel u8Channel)
- void ADC_vidCalibrate (void)
- void ADC_vidEnable (void)
- void ADC_vidDisable (void)
- void ADC_vidAutoTriggerEnable (void)
- void ADC_vidAutoTriggerDisable (void)
- LBTY_tenuErrorStatus ADC_u8SetAutoTriggerSource (ADC_tenuTriggerSource u8Source)
- LBTY_tenuErrorStatus ADC_u8SetPrescaler (ADC_tenuPrescalerSelection u8Prescaler)
- LBTY_tenuErrorStatus ADC_u8SetV_REF (ADC_tenuRefSelection u8Vref)
- LBTY_tenuErrorStatus ADC_u8SetChannel (ADC_tenuChannel u8Channel)
- void ADC_vidStartConversion (void)
- void ADC_vidWaitConversion (void)
- u16 ADC_u16GetData (void)
- f32 ADC_f32GetVoltage (void)
- LBTY_tenuErrorStatus ADC_u8StartRead (ADC_tenuChannel u8Channel)
- LBTY_tenuErrorStatus ADC_u8ReadChannel (ADC_tenuChannel u8Channel, u16 *pu16ADC_Value)
- LBTY_tenuErrorStatus ADC_u8ReadConvValue (u8 u8Channel, u16 *pu16ADC_Value)
- LBTY_tenuErrorStatus ADC_u16RefreshADC (void)
- LBTY_tenuErrorStatus ADC_u16GetAll (u16 pu16ADC_Value[])
- void ADC_vidEnableINT (void)
- void ADC_vidDisableINT (void)
- void ADC_vidClrFlagINT (void)
- void ADC_vidSetCallBack (void(*pvidCallBack)(void))
- ISR (ADC_vect)

Variables

- static u8 au8ChannelValue_LGB [ADC_Num]
- static u8 u8ConvDone_GLB = LBTY_SET
- static f32 f32V_REF = ADC_V_REF
- static void(* pvidFunctionCallBack)(void)

Function Documentation

f32 ADC_f32GetVoltage (void)

```
292                                 {
293     return (f32)ADC_u16GetData() * f32V_REF / ADC_MAX;
294 }
```

Here is the call graph for this function:

LBTY_tenuErrorStatus ADC_u16GetAll (u16 *pu16ADC_Value*[])

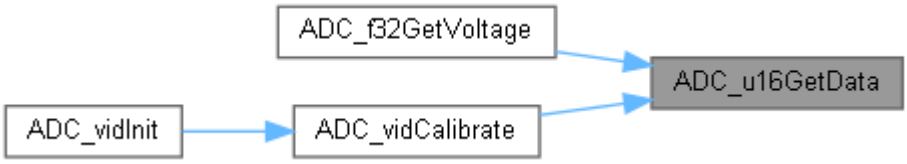
Clear complete flag by writing logic one

```
396                                                                 {
397     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
398 
399     if(u8ConvDone_GLB == LBTY_SET){
400         S_ADC->m_ADCSRA.sBits.m_ADIE  = LBTY_RESET;
401         S_ADC->m_ADCSRA.sBits.m_ADIF  = LBTY_SET;
402         for(u8 i = 0 ; i<ADC_Num ; i++){
403             pu16ADC_Value[i] = au8ChannelValue_LGB[i];
404         }
405     }else{
406         u8RetErrorState = LBTY_IN_PROGRESS;
407     }
408     return u8RetErrorState;
409 }
```

u16 ADC_u16GetData (void)

```
283                                 {
284     return (u16)S_ADC->m_ADC;
285 }
```

Here is the caller graph for this function:



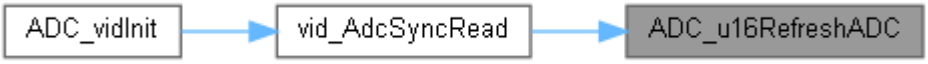
LBTY_tenuErrorStatus ADC_u16RefreshADC (void)

Clear complete flag by writing logic one

```
377                                             {
378     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
379 
380     if(u8ConvDone_GLB == LBTY_SET){
381         u8ConvDone_GLB = LBTY_RESET;
382         S_ADC->m_ADCSRA.sBits.m_ADIE  = LBTY_SET;
383         S_ADC->m_ADCSRA.sBits.m_ADIF  = LBTY_SET;
384         u8RetErrorState = ADC_u8StartRead(*kau8ActiveChannel_LGB);
385     }else{
386         u8RetErrorState = LBTY_IN_PROGRESS;
387     }
388     return u8RetErrorState;
389 }
```

Here is the call graph for this function:

Here is the caller graph for this function:



LBTY_tenuErrorStatus ADC_u8CofigChannel (ADC_tenuChannel *u8Channel*)

```
110                                                                     {
111     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
112
113     if(IS_CHANNEL(u8Channel)){
114         u8RetErrorState = GPIO_u8SetPinDirection(ADC_PORT, u8Channel,
PIN_INPUT);
115     }else{
116         u8RetErrorState = LBTY_NOK;
117     }
118     return u8RetErrorState;
119 }
```

Here is the caller graph for this function:

LBTY_tenuErrorStatus ADC_u8ReadChannel (ADC_tenuChannel *u8Channel*, u16 * *pu16ADC_Value*)

```
323
{
324     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
325
326     if(pu16ADC_Value == LBTY_NULL){
327         u8RetErrorState = LBTY_NULL_POINTER;
328     }else{
329         if((u8)u8Channel < ADC_ChannelMux){
330             S_ADC->m_ADMUX.sBits.m_MUX = u8Channel;
331         }else{
332             u8RetErrorState = LBTY_NOK;
333         }
334
335         S_ADC->m_ADCSRA.sBits.m_ADSC = LBTY_SET;
336
337         while(S_ADC->m_ADCSRA.sBits.m_ADSC);
338
339         //vidMyDelay_ms(ADC_READ_DELAY);
340
341         //while(!S_ADC->m_ADCSRA.sBits.m_ADIF);
S_ADC->m_ADCSRA.sBits.m_ADIF = LBTY_RESET;
342
343         *pu16ADC_Value = (u16)S_ADC->m_ADC;
344     }
345     return u8RetErrorState;
346 }
```

LBTY_tenuErrorStatus ADC_u8ReadConvValue (u8 *u8Channel*, u16 * *pu16ADC_Value*)

```
354                                                                     {
355     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
356
357     if(pu16ADC_Value == LBTY_NULL){
358         u8RetErrorState = LBTY_NULL_POINTER;
359     }else{
360         for(u8 i = 0 ; i<ADC_Num ; i++){
361             if(u8Channel == kau8ActiveChannel_LGB[i]){
362                 *pu16ADC_Value = au8ChannelValue_LGB[i];
363                 break;
364             }else{
365                 u8RetErrorState = LBTY_NOK;
366             }
367         }
368     }
369     return u8RetErrorState;
370 }
```

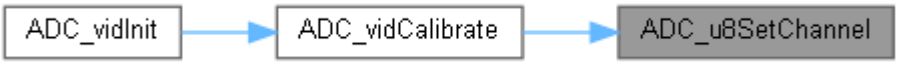
LBTY_tenuErrorStatus ADC_u8SetAutoTriggerSource (ADC_tenuTriggerSource *u8Source*)

```
178
{
179     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
180     switch(u8Source){
181         case ADC_Free_Running_Mode:
182         case ADC_Analog_Comparator:
183         case ADC_External_INT0:
184         case ADC_TMR0_Compare_MatchA:
185         case ADC_TMR0_Overflew:
186         case ADC_TMR1_Compare_MatchB:
187         case ADC_TMR1_Overflew:
188         case ADC_TMR1_Capture_Event:
189             S_SFIOR->sBits.m_ADTS  = u8Source;
190             break;
191         default:
192             u8RetErrorState = LBTY_NOK;
193     }
194
195     return u8RetErrorState;
196 }
```

LBTY_tenuErrorStatus ADC_u8SetChannel (ADC_tenuChannel *u8Channel*)

```
248                                                                 {
249     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
250
251     if((u8)u8Channel < ADC_ChannelMux){
252         S_ADC->m_ADMUX.sBits.m_MUX = u8Channel;
253     }else{
254         u8RetErrorState = LBTY_NOK;
255     }
256
257     return u8RetErrorState;
258 }
```

Here is the caller graph for this function:



LBTY_tenuErrorStatus ADC_u8SetPrescaler (ADC_tenuPrescalerSelection *u8Prescaler*)

```
203
{
204     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
205     switch(u8Prescaler){
206         case ADC_Division_Factor_2_DEF:
207         case ADC_Division_Factor_2:
208         case ADC_Division_Factor_4:
209         case ADC_Division_Factor_8:
210         case ADC_Division_Factor_16:
211         case ADC_Division_Factor_32:
212         case ADC_Division_Factor_64:
213         case ADC_Division_Factor_128:
214             S_ADC->m_ADCSRA.sBits.m_ADPS  = u8Prescaler;
215             break;
216         default:
217             u8RetErrorState = LBTY_NOK;
218     }
219
220     return u8RetErrorState;
221 }
```

LBTY_tenuErrorStatus ADC_u8SetV_REF (ADC_tenuRefSelection *u8Vref*)

```
228                                                                 {
229     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
230     switch(u8Vref){
231         case ADC_AREF:
232         case ADC_AVCC:
233         case ADC_INTERNAL_Vref:
234             S_ADC->m_ADMUX.sBits.m_REFS  = u8Vref;
235             break;
```

```
236         default:
237             u8RetErrorState = LBTY_NOK;
238     }
239 
240     return u8RetErrorState;
241 }
```

LBTY_tenuErrorStatus ADC_u8StartRead (ADC_tenuChannel *u8Channel*)

```
303                                                                 {
304     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
305 
306     if((u8)u8Channel < ADC_ChannelMux){
307         S_ADC->m_ADMUX.sBits.m_MUX = u8Channel;
308     }else{
309         u8RetErrorState = LBTY_NOK;
310     }
311 
312     S_ADC->m_ADCSRA.sBits.m_ADSC = LBTY_SET;
313 
314     return u8RetErrorState;
315 }
```

Here is the caller graph for this function:

void ADC_vidAutoTriggerDisable (void)

```
169                                     {
170     S_ADC->m_ADCSRA.sBits.m_ADATE = LBTY_RESET;
171 }
```

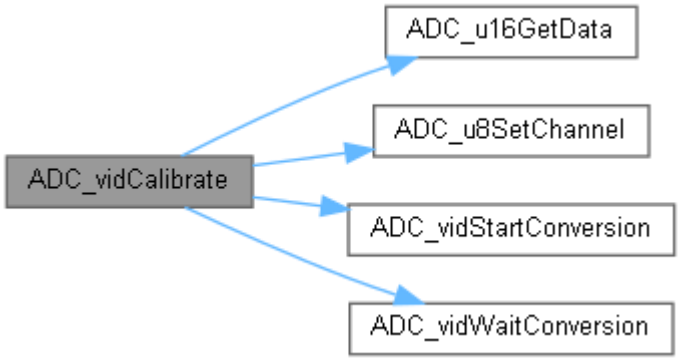
void ADC_vidAutoTriggerEnable (void)

```
160                                    {
161     S_ADC->m_ADCSRA.sBits.m_ADATE = LBTY_RESET;
162 }
```

void ADC_vidCalibrate (void)

```
126                                {
127     ADC_u8SetChannel(VBG_1V22);
128     ADC_vidStartConversion();
129     ADC_vidWaitConversion();
130 
131     f32V_REF = (f32)(ADC_VBG_1V22 * ADC_MAX) / ADC_u16GetData();
132     ADC_u8SetChannel(ADC0);
133 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

void ADC_vidClrFlagINT (void)

Clear complete flag by writing logic one

```
436                                 {
437     S_ADC->m_ADCSRA.sBits.m_ADIF  = LBTY_SET;
438 }
```

void ADC_vidDisable (void)

```
151                             {
152     S_ADC->m_ADCSRA.sBits.m_ADEN  = LBTY_RESET;
153 }
```

void ADC_vidDisableINT (void)

```
427                                 {
428     S_ADC->m_ADCSRA.sBits.m_ADIE  = LBTY_RESET;
429 }
```

void ADC_vidEnable (void)

```
142                             {
143     S_ADC->m ADCSRA.sBits.m ADEN  = LBTY_SET;
144 }
```

void ADC_vidEnableINT (void)

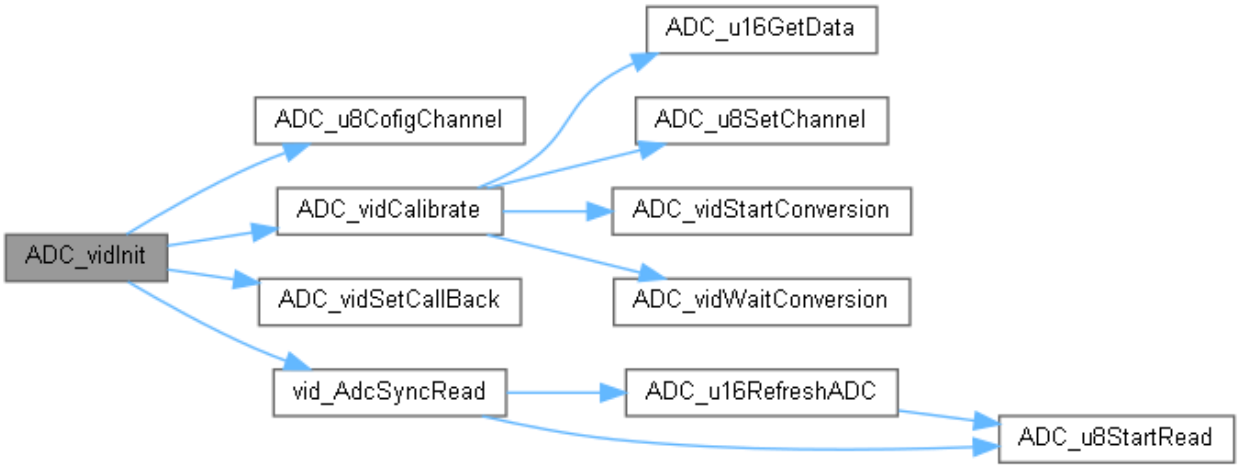
```
418                                 {
419     S_ADC->m_ADCSRA.sBits.m_ADIE  = LBTY_SET;
420 }
```

void ADC_vidInit (void)

Clear complete flag by writing logic one

```
75                          {
76
77     for(u8 i = ADC_Num; i-- ; ){
78         ADC_u8CofigChannel(kau8ActiveChannel_LGB[i]);
79     }
80
81     S_SFIOR->sBits.m_ADTS = ADC_TRIG_SRC;
82
83     S_ADC->m_ADCSRA.sBits.m_ADEN  = LBTY_SET;
84     S_ADC->m_ADCSRA.sBits.m_ADSC  = LBTY_RESET;
85     S_ADC->m_ADCSRA.sBits.m_ADATE = ADC_AUTO_TRIG;
86     S_ADC->m_ADCSRA.sBits.m_ADIE  = ADC_IRQ_STATE;
87     S_ADC->m_ADCSRA.sBits.m_ADIF  = LBTY_SET;
88     S_ADC->m_ADCSRA.sBits.m_ADPS  = ADC_PRESCALER;
89     ADC_vidSetCallBack(vid_AdcSyncRead);
90
91     S_ADC->m_ADMUX.sBits.m_REFS  = ADC_V_REF_SRC;
92     S_ADC->m_ADMUX.sBits.m_ADLAR = ADC_ADJUSTMENT;
93     S_ADC->m_ADMUX.sBits.m_MUX   = ADC0;
94
95     // first conversion will take 25 ADC clock cycles instead of the normal 13.
96     S_ADC->m_ADCSRA.sBits.m_ADSC  = LBTY_SET;
97     while(S_ADC->m_ADCSRA.sBits.m_ADSC);
98
99     ADC_vidCalibrate();
100
101     S_ADC->m_ADCSRA.sBits.m_ADEN  = ADC_INIT_STATE;
102
103 }
```

Here is the call graph for this function:



void ADC_vidSetCallBack (void(*)(void) *pvidCallBack*)

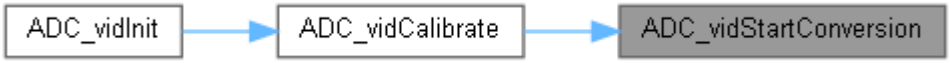
```
445                                                     {
446     pvidFunctionCallBack = pvidCallBack;
447 }
```

Here is the caller graph for this function:

void ADC_vidStartConversion (void)

```
265                                 {
266     S_ADC->m_ADCSRA.sBits.m_ADSC = LBTY_SET;
267 }
```

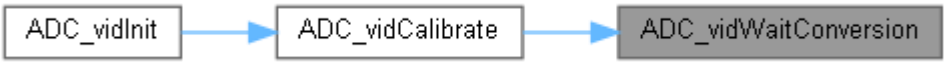
Here is the caller graph for this function:



void ADC_vidWaitConversion (void)

```
274                                 {
275     while(S_ADC->m_ADCSRA.sBits.m_ADSC);
276 }
```

Here is the caller graph for this function:



ISR (ADC_vect)

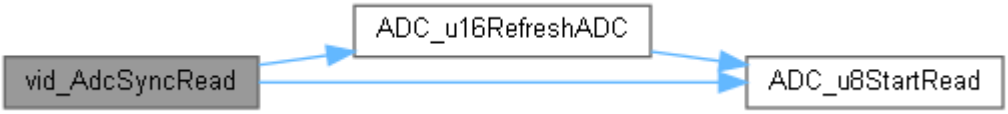
```
454             {
455     pvidFunctionCallBack();     //call back function
456 }
```

static void vid_AdcSyncRead (void) [static]

```
51                                  {
52     static u8 u8Channel = LBTY_u8ZERO;
53
54     if (u8Channel < ADC_Num){
55
56         u8ConvDone_GLB = LBTY_RESET;
57
58         au8ChannelValue_LGB[u8Channel++] = (u16)S_ADC->m_ADC;
59
60         while(ADC_u8StartRead(u8Channel));
61
62     }else{
63
64         ADC_u16RefreshADC();
65         u8Channel = LBTY_u8ZERO;
66         u8ConvDone_GLB = LBTY_SET;
67     }
```

```
68 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

Variable Documentation

u8 au8ChannelValue_LGB[ADC_Num] `[static]`

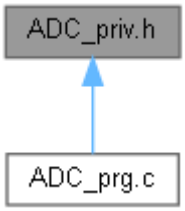
f32 f32V_REF = ADC_V_REF `[static]`

void(* pvidFunctionCallBack) (void) (void) `[static]`

u8 u8ConvDone_GLB = LBTY_SET `[static]`

ADC_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 to Digital Converter Registers*

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

Macros

- #define IS_CHANNEL(channel) ((channel >= ADC0) && (channel <= ADC7))
- #define ADC_PORT A
- #define ADC_MAX 1023u
- #define ADC_VBG_1V22 1.22f
- #define S_ADC ((ADC_type* const)0x24U)
- #define ADCL (*(volatile u8* const)0x24U)
- #define ADCH (*(volatile u8* const)0x25U)
- #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)

Enumerations

- enum ADC_tenuPinNum { ADC_Num }
  : Type define of ADC Pin Number enum

Macro Definition Documentation

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

#define ADC_MAX 1023u

#define ADC_PORT A

#define ADC_VBG_1V22 1.22f

#define ADCH (*(volatile u8* const)0x25U)

#define ADCL (*(volatile u8* const)0x24U)

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

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

#define IS_CHANNEL(channel) ((channel >= ADC0) && (channel <= ADC7))

#define S_ADC ((ADC_type* const)0x24U)

Analog Digital Converter

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

Special Function I/O Register

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

Enumeration Type Documentation

enum ADC_tenuPinNum

: Type define of ADC Pin Number enum

Type : Union **Unit** : None

Enumerator:

ADC_Num	ADC Max pin Number

```
21              {
22 #ifdef ADC_CH0
23         ADC_0 = (u8)0u
24 #endif
25 #ifdef ADC_CH1
26         , ADC_1
27 #endif
28 #ifdef ADC_CH2
29         , ADC_2
30 #endif
31 #ifdef ADC_CH3
32         , ADC_3
```

```
33 #endif
34 #ifdef ADC_CH4
35          , ADC_4
36 #endif
37 #ifdef ADC_CH5
38          , ADC_5
39 #endif
40 #ifdef ADC_CH6
41          , ADC_6
42 #endif
43 #ifdef ADC_CH7
44          , ADC_7
45 #endif
46          , ADC_Num
47 }ADC_tenuPinNum;
```

ADC_priv.h

Go to the documentation of this file.

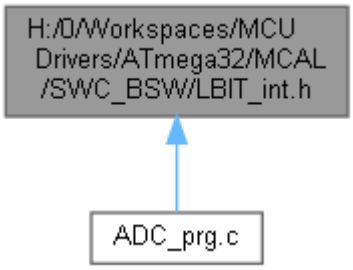
```
1 /*
2 /* ********************** FILE DEFINITION SECTION ************************** */
3 /* ************************************************************************** */
4 /* File Name    : ADC_priv.h                                                  */
5 /* Author       : MAAM                                                        */
6 /* Version      : v01.2                                                       */
7 /* date         : Mar 27, 2023                                                */
8 /* ************************************************************************** */
9 /* ************************ HEADER FILES INCLUDES ************************  */
10 /* ************************************************************************** */
11
12 #ifndef ADC_PRIV_H_
13 #define ADC_PRIV_H_
14
15 /* ************************************************************************** */
16 /* ********************** TYPE_DEF/STRUCT/ENUM SECTION ********************** */
17 /* ************************************************************************** */
18
21 typedef enum {
22 #ifdef ADC_CH0
23         ADC_0 = (u8)0u
24 #endif
25 #ifdef ADC_CH1
26         , ADC_1
27 #endif
28 #ifdef ADC_CH2
29         , ADC_2
30 #endif
31 #ifdef ADC_CH3
32         , ADC_3
33 #endif
34 #ifdef ADC_CH4
35         , ADC_4
36 #endif
37 #ifdef ADC_CH5
38         , ADC_5
39 #endif
40 #ifdef ADC_CH6
41         , ADC_6
42 #endif
43 #ifdef ADC_CH7
44         , ADC_7
45 #endif
46         , ADC_Num
47 }ADC_tenuPinNum;
48
49 /*****************************************************************************/
50
53 typedef union{
54     u8 u_Reg;
55     struct {
56         __IO u8 m_ADPS : 3;
57         __IO u8 m_ADIE : 1;
58         __IO u8 m_ADIF : 1;
59         __IO u8 m_ADATE: 1;
60         __IO u8 m_ADSC : 1;
61         __IO u8 m_ADEN : 1;
62     }sBits;
63 }ADCSRA_type;
64
65 /*****************************************************************************/
66
69 typedef union{
70     u8 u_Reg;
71     struct {
72         __IO u8 m_MUX  : 5;
73         __IO u8 m_ADLAR: 1;
74         __IO u8 m_REFS : 2;
75     }sBits;
76 }ADMUX_type;
77
78 /*****************************************************************************/
```

```
79
82 typedef union{
83     u8 u_Reg;
84     struct {
85         __IO u8        : 7;
86         __IO u8 m_ACD  : 1;
87     }sBits;
88 }ACSR_type;
89
90 /*****************************************************************************/
91
94 typedef struct{
95 #if ADC_ADJUSTMENT
96     __I  u16               : 6;
97     __I  u16          m_ADC : 10;
98 #else
99     __I  u16          m_ADC : 10;
100    __I  u16                : 6;
101 #endif
102     __IO ADCSRA_type  m_ADCSRA;
103     __IO ADMUX_type   m_ADMUX;
104     __IO ACSR_type    m_ACSR;
105 }ADC_type;
106
107 /*****************************************************************************/
108
111 typedef union{
112     u8 u_Reg;
113     struct {
114         __IO u8        : 3;
115         __IO u8 m_ACME : 1;
116         __IO u8        : 1;
117         __IO u8 m_ADTS : 3;
118     }sBits;
119 }SFIOR_type;
120
121 /* ************************************************************************** */
122 /* ************************* MACRO/DEFINE SECTION ************************** */
123 /* ************************************************************************** */
124
125 #define IS_CHANNEL(channel)     ((channel >= ADC0) && (channel <= ADC7))
126
127 #define ADC_PORT                A
128
129 #define ADC_MAX                 1023u
130 #define ADC_VBG_1V22            1.22f
131
133 #define S_ADC           ((ADC_type* const)0x24U)
134 #define ADCL            (*(volatile u8* const)0x24U)
135 #define ADCH            (*(volatile u8* const)0x25U)
136 #define ADCSRA          (*(volatile u8* const)0x26U)
137 #define ADMUX           (*(volatile u8* const)0x27U)
138 #define ACSR            (*(volatile u8* const)0x28U)
139
141 #define S_SFIOR         ((SFIOR_type* const)0x50U)
142 #define SFIOR           (*(volatile u8* const)0x50U)
143
144 /* ************************************************************************** */
145 /* ************************** CONST SECTION ********************************* */
146 /* ************************************************************************** */
147
148 /* ************************************************************************** */
149 /* ************************** VARIABLE SECTION ****************************** */
150 /* ************************************************************************** */
151
152 /* ************************************************************************** */
153 /* ************************** FUNCTION SECTION ****************************** */
154 /* ************************************************************************** */
155
156
157 #endif /* ADC_PRIV_H_ */
158 /************************** E N D (ADC_priv.h) ****************************/
```

main.c File Reference

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.

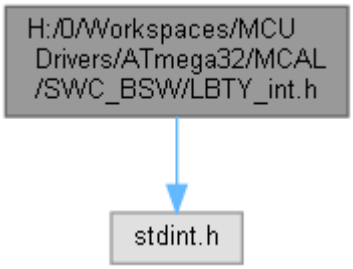
```
1 /*
******************************************************************************* */
2 /* ********************** FILE DEFINITION SECTION ************************** */
3 /* ******************************************************************************* */
4 /* File Name   : LBIT_int.h         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

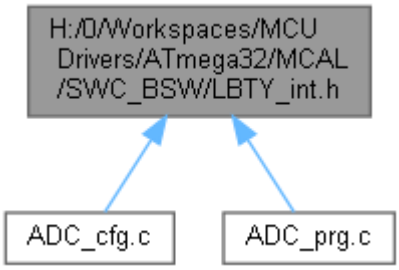
H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_BSW/LBTY_int.h File Reference

```
#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.

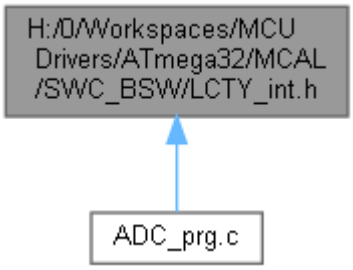
```
1 /*
2 /* ********************** FILE DEFINITION SECTION ************************** */
3 /* ************************************************************************* */
4 /* File Name   : LBTY_int.h                                                  */
5 /* Author      : MAAM                      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