

# **SWC\_ADC\_IF**

Version v1.0

7/19/2023 11:55:00 PM



# Table of Contents

Data Structure Index.....	2
File Index.....	3
Data Structure Documentation .....	4
ACSR_type.....	4
ADC_type.....	5
ADCSRA_type.....	7
ADMUX_type .....	9
LBTY_tuniPort16.....	11
LBTY_tuniPort8.....	13
SENSOR_tstrConfig.....	15
SFIOR_type.....	16
File Documentation .....	18
ADCIF_cfg.c .....	18
ADCIF_cfg.h.....	19
ADCIF_int.h.....	21
ADCIF_prg.c .....	24
ADCIF_priv.h.....	26
main.c .....	28
H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_ADC/main.c .....	29
H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_ADC/ADC_cfg.c .....	30
H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_ADC/ADC_cfg.h .....	31
H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_ADC/ADC_cfg.h .....	33
H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_ADC/ADC_int.h .....	35
H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_ADC/ADC_int.h .....	45
H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_ADC/ADC_prg.c .....	49
H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_ADC/ADC_priv.h.....	57
H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_ADC/ADC_priv.h.....	60
H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_BSW/LBIT_int.h .....	62
H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_BSW/LBIT_int.h .....	65
H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_BSW/LBTY_int.h .....	67
H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_BSW/LBTY_int.h .....	72
H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_BSW/LCTY_int.h .....	75
H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_BSW/LCTY_int.h .....	77
Index.....	<b>Error! Bookmark not defined.</b>



# Data Structure Index

## Data Structures

Here are the data structures with brief descriptions:

<a href="#"><u>ACSR_type</u></a> (: Type define of Union bit field of "Analog Comparator Control and Status" )	.....4
<a href="#"><u>ADC_type</u></a> (: Analog to Digital Converter Registers	
)	.....5
<a href="#"><u>ADCSRA_type</u></a> (: Type define of Union bit field of "ADC Control and Status Reg A" )	.....7
<a href="#"><u>ADMUX_type</u></a> (: Type define of Union bit field of "ADC Multiplexer Selection Reg" )	.....9
<a href="#"><u>LBTY_tuniPort16</u></a>	.....11
<a href="#"><u>LBTY_tuniPort8</u></a>	.....13
<a href="#"><u>SENSOR_tstrConfig</u></a>	.....15
<a href="#"><u>SFIOR_type</u></a> (: Type define of Union bit field of "Special Function I/O Register" )	.....16

# File Index

## File List

Here is a list of all files with brief descriptions:

<a href="#"><u>ADCIF_cfg.c</u></a>	18
<a href="#"><u>ADCIF_cfg.h</u></a>	19
<a href="#"><u>ADCIF_int.h</u></a>	21
<a href="#"><u>ADCIF_prg.c</u></a>	24
<a href="#"><u>ADCIF_priv.h</u></a>	26
<a href="#"><u>main.c</u></a>	28
H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_ADC/ <a href="#"><u>ADC_cfg.c</u></a>	30
H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_ADC/ <a href="#"><u>ADC_cfg.h</u></a>	31
H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_ADC/ <a href="#"><u>ADC_int.h</u></a>	35
H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_ADC/ <a href="#"><u>ADC_prg.c</u></a>	49
H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_ADC/ <a href="#"><u>ADC_priv.h</u></a>	57
H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_ADC/ <a href="#"><u>main.c</u></a>	29
H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_BSW/ <a href="#"><u>LBIT_int.h</u></a>	62
H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_BSW/ <a href="#"><u>LBTY_int.h</u></a>	67
H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_BSW/ <a href="#"><u>LCTY_int.h</u></a>	75

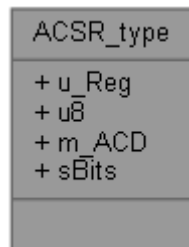
# Data Structure Documentation

## 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:

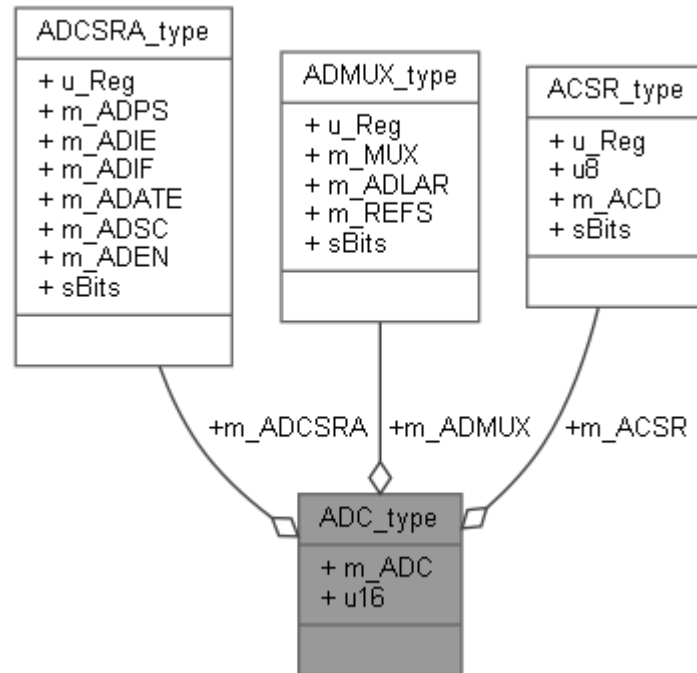
- H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC\_ADC/[ADC\\_priv.h](#)

## 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:**

- H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC\_ADC/[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:**

- H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC\_ADC/[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

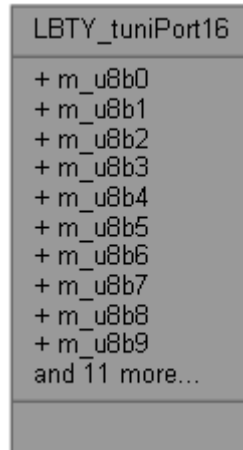
**The documentation for this union was generated from the following file:**

- H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC\_ADC/[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

---

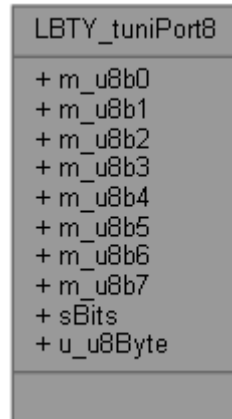
The documentation for this union was generated from the following file:

- 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

---

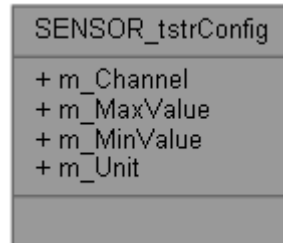
The documentation for this union was generated from the following file:

- H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC\_BSW/[LBTY\\_int.h](#)

## SENSOR\_tstrConfig Struct Reference

```
#include <ADCIF_int.h>
```

Collaboration diagram for SENSOR\_tstrConfig:



### Data Fields

- [ADC\\_tenuChannel](#) `m_Channel`
- [s32](#) `m_MaxValue`
- [s32](#) `m_MinValue`
- [u8](#) `m_Unit`

---

### Field Documentation

[ADC\\_tenuChannel](#) `m_Channel`

[s32](#) `m_MaxValue`

[s32](#) `m_MinValue`

[u8](#) `m_Unit`

---

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

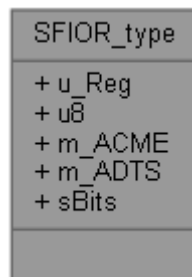
[ADCIF\\_int.h](#)

## 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

---

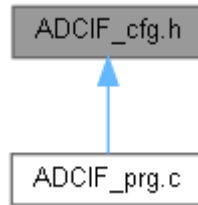
**The documentation for this union was generated from the following file:**

- H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC\_ADC/[ADC\\_priv.h](#)



## ADCIF\_cfg.h File Reference

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



### Variables

- const [SENSOR\\_tstrConfig](#) [katstrSensorConfig\\_GLB](#) []
- const [u8](#) [SENSOR\\_NUM](#)

---

### Variable Documentation

const [SENSOR\\_tstrConfig](#) [katstrSensorConfig\\_GLB](#)[] [extern]

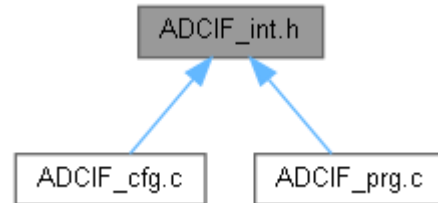
const [u8](#) [SENSOR\\_NUM](#) [extern]

## ADCIF\_cfg.h

```
Go to the documentation of this file.1 /*
*****
2 /* ***** FILE DEFINITION SECTION ***** */
3 /* ***** */
4 /* File Name      : ADCIF_cfg.h */
5 /* Author         : MAAM */
6 /* Version        : v01.2 */
7 /* date           : Jul 19, 2023 */
8 /* ***** */
9 /* ***** HEADER FILES INCLUDES ***** */
10 /* ***** */
11
12 #ifndef ADCIF_CFG_H_
13 #define ADCIF_CFG_H_
14
15 /* ***** */
16 /* ***** TYPE_DEF/STRUCT/ENUM SECTION ***** */
17 /* ***** */
18
19 /* ***** */
20 /* ***** MACRO/DEFINE SECTION ***** */
21 /* ***** */
22
23 /* ***** */
24 /* ***** CONST SECTION ***** */
25 /* ***** */
26 extern const SENSOR\_tstrConfig katstrSensorConfig GLB[];
27 extern const u8 SENSOR\_NUM;
28 /* ***** */
29 /* ***** VARIABLE SECTION ***** */
30 /* ***** */
31
32 /* ***** */
33 /* ***** FUNCTION SECTION ***** */
34 /* ***** */
35
36
37 #endif /* ADCIF_CFG_H */
38 /***** E N D (ADCIF_cfg.h) *****/
```

## ADCIF\_int.h File Reference

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



## Data Structures

### struct [SENSOR\\_tstrConfig](#) Functions

- void [SENSOR\\_vidInit](#) (void)
- [LBTY\\_tenuErrorStatus](#) [SENSOR\\_u8GetData](#) ([f32](#) \*paf32Read)
- [LBTY\\_tenuErrorStatus](#) [SENSOR\\_u8GetChannel](#) ([u8](#) u8Num, [f32](#) \*pf32Read)

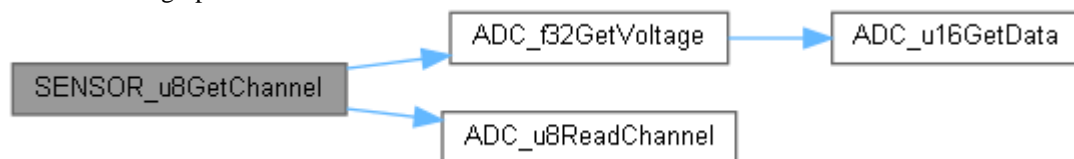
## Function Documentation

### [LBTY\\_tenuErrorStatus](#) [SENSOR\\_u8GetChannel](#) ([u8](#) u8Num, [f32](#) \* pf32Read)

```

68 {
69     LBTY\_tenuErrorStatus u8RetErrorState = LBTY\_OK;
70     u16 u16AdcReadValue = LBTY\_u8ZERO;
71     const SENSOR\_tstrConfig* pstrSensor = (const
SENSOR\_tstrConfig*)&katstrSensorConfig_GLB[u8Num];
72     u8RetErrorState = ADC\_u8ReadChannel(pstrSensor->m_Channel, &u16AdcReadValue);
73     if(u8RetErrorState == LBTY\_OK) {
74         *pf32Read = (f32) (ADC\_f32GetVoltage()
75             * (pstrSensor->m_MaxValue - pstrSensor->m_MinValue) /
76             (f32)ADC\_V\_REF + pstrSensor->m_MinValue);
77     }
78     return u8RetErrorState;
79 }
  
```

Here is the call graph for this function:



Here is the caller graph for this function:

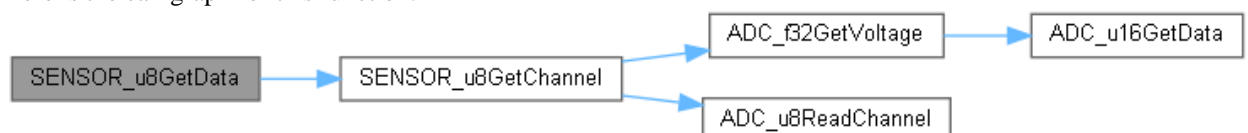


### [LBTY\\_tenuErrorStatus](#) [SENSOR\\_u8GetData](#) ([f32](#) \* paf32Read)

```

55 {
56     LBTY\_tenuErrorStatus u8RetErrorState = LBTY\_OK;
57     for(u8 i = SENSOR\_NUM ; i-- && (u8RetErrorState == LBTY\_OK) ; ){
58         u8RetErrorState = SENSOR\_u8GetChannel(i, paf32Read + i);
59     }
60     return u8RetErrorState;
61 }
  
```

Here is the call graph for this function:

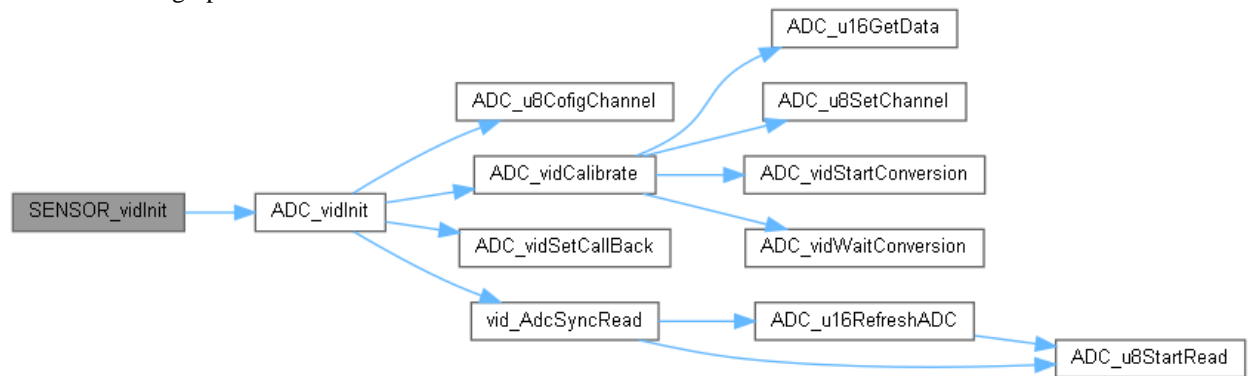




**void SENSOR\_vidInit (void )**

```
46 {  
47     ADC_vidInit ();  
48 }
```

Here is the call graph for this function:



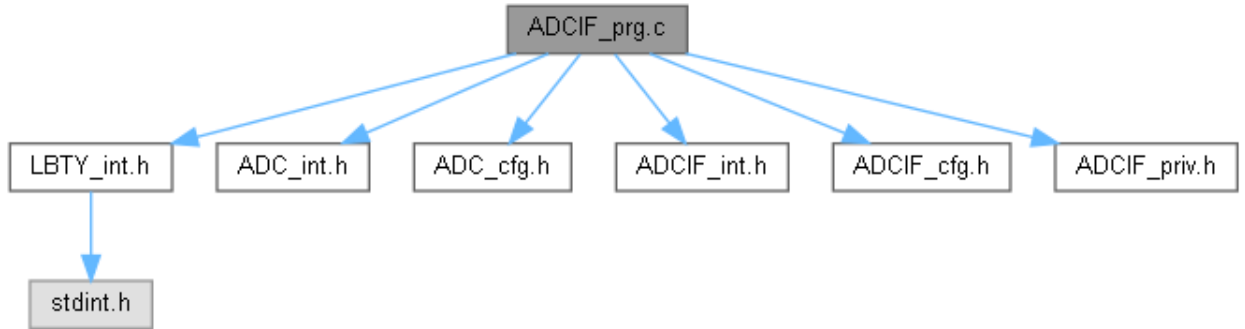
## ADCIF\_int.h

```
Go to the documentation of this file.1 /*
*****
2 /* ***** FILE DEFINITION SECTION ***** */
3 /* ***** */
4 /* File Name : ADCIF_int.h */
5 /* Author : MAAM */
6 /* Version : v01.2 */
7 /* date : Jul 19, 2023 */
8 /* ***** */
9 /* ***** HEADER FILES INCLUDES ***** */
10 /* ***** */
11
12 #ifndef ADCIF_INT_H_
13 #define ADCIF_INT_H_
14
15 /* ***** */
16 /* ***** TYPE_DEF/STRUCT/ENUM SECTION ***** */
17 /* ***** */
18
19 typedef struct{
20     ADC_tenuChannel m_Channel;
21     s32 m_MaxValue;
22     s32 m_MinValue;
23     u8 m_Unit;
24 }SENSOR_tstrConfig;
25
26 /* ***** */
27 /* ***** MACRO/DEFINE SECTION ***** */
28 /* ***** */
29
30 /* ***** */
31 /* ***** CONST SECTION ***** */
32 /* ***** */
33
34 /* ***** */
35 /* ***** VARIABLE SECTION ***** */
36 /* ***** */
37
38 /* ***** */
39 /* ***** FUNCTION SECTION ***** */
40 /* ***** */
41
42 /* ***** */
43 /* Description : Initialization of the Sensors */
44 /* Input : void */
45 /* Return : void */
46 /* ***** */
47 extern void SENSOR_vidInit(void);
48
49 /* ***** */
50 /* Description : Get Sensors Read */
51 /* Input/Output: paf32Read */
52 /* Return : LBTY_tenuErrorStatus */
53 /* ***** */
54 extern LBTY_tenuErrorStatus SENSOR_u8GetData(f32* paf32Read);
55
56 /* ***** */
57 /* Description : Get Sensor Read */
58 /* Input/Output: pf32Read */
59 /* Return : LBTY_tenuErrorStatus */
60 /* ***** */
61 extern LBTY_tenuErrorStatus SENSOR_u8GetChannel(u8 u8Num, f32* pf32Read);
62
63 #endif /* ADCIF_INT_H_ */
64 /****** E N D (ADCIF_int.h) ******/
```

## ADCIF\_prg.c File Reference

```
#include "LBTY_int.h"
#include "ADC_int.h"
#include "ADC_cfg.h"
#include "ADCIF_int.h"
#include "ADCIF_cfg.h"
#include "ADCIF_priv.h"
```

Include dependency graph for ADCIF\_prg.c:



## Functions

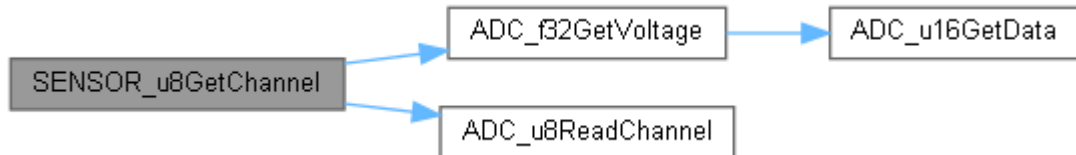
- void [SENSOR\\_vidInit](#) (void)
- [LBTY\\_tenuErrorStatus](#) [SENSOR\\_u8GetData](#) ([f32](#) \*paf32Read)
- [LBTY\\_tenuErrorStatus](#) [SENSOR\\_u8GetChannel](#) ([u8](#) u8Num, [f32](#) \*pf32Read)

## Function Documentation

### [LBTY\\_tenuErrorStatus](#) [SENSOR\\_u8GetChannel](#) ([u8](#) u8Num, [f32](#) \* pf32Read)

```
68 {
69     LBTY\_tenuErrorStatus u8RetErrorState = LBTY\_OK;
70     u16 u16AdcReadValue = LBTY\_u8ZERO;
71     const SENSOR\_tstrConfig* pstrSensor = (const
SENSOR\_tstrConfig*)&katstrSensorConfig GLB[u8Num];
72     u8RetErrorState = ADC\_u8ReadChannel(pstrSensor->m_Channel, &u16AdcReadValue);
73     if(u8RetErrorState == LBTY\_OK) {
74         *pf32Read = (f32) (ADC\_f32GetVoltage()
75             * (pstrSensor->m_MaxValue - pstrSensor->m_MinValue) /
76             (f32)ADC\_V\_REF + pstrSensor->m_MinValue);
77     }
78     return u8RetErrorState;
79 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



### [LBTY\\_tenuErrorStatus](#) [SENSOR\\_u8GetData](#) ([f32](#) \* paf32Read)

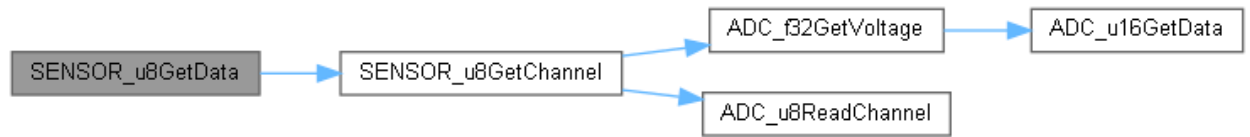
```
55 {
56     LBTY\_tenuErrorStatus u8RetErrorState = LBTY\_OK;
57     for(u8 i = SENSOR\_NUM ; i-- && (u8RetErrorState == LBTY\_OK) ; ){
58         u8RetErrorState = SENSOR\_u8GetChannel(i, paf32Read + i);
```

```

59     }
60     return u8RetErrorState;
61 }

```

Here is the call graph for this function:



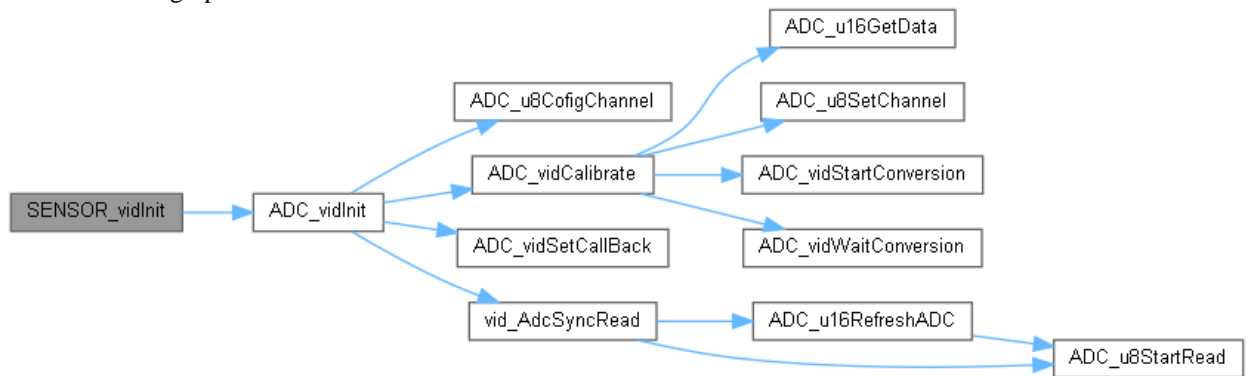
**void SENSOR\_vidInit (void )**

```

46     {
47     ADC_vidInit();
48 }

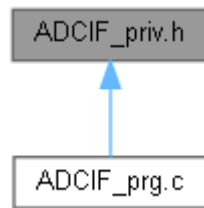
```

Here is the call graph for this function:



## ADCIF\_priv.h File Reference

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



## ADCIF\_priv.h

```
Go to the documentation of this file.1 /*
*****
2 /* ***** FILE DEFINITION SECTION ***** */
3 /* ***** */
4 /* File Name      : ADCIF_priv.h */
5 /* Author         : MAAM */
6 /* Version        : v01.2 */
7 /* date           : Jul 19, 2023 */
8 /* ***** */
9 /* ***** HEADER FILES INCLUDES ***** */
10 /* ***** */
11
12 #ifndef ADCIF_PRIV_H_
13 #define ADCIF_PRIV_H_
14
15 /* ***** */
16 /* ***** TYPE_DEF/STRUCT/ENUM SECTION ***** */
17 /* ***** */
18
19 /* ***** */
20 /* ***** MACRO/DEFINE SECTION ***** */
21 /* ***** */
22
23 /* ***** */
24 /* ***** CONST SECTION ***** */
25 /* ***** */
26
27 /* ***** */
28 /* ***** VARIABLE SECTION ***** */
29 /* ***** */
30
31 /* ***** */
32 /* ***** FUNCTION SECTION ***** */
33 /* ***** */
34
35
36 #endif /* ADCIF_PRIV_H_ */
37 /***** E N D (ADCIF_priv.h) *****/
```

## **main.c File Reference**

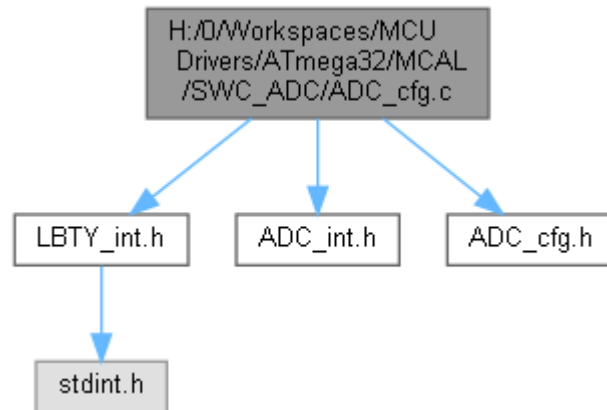
**H:/0/Workspaces/MCU  
Drivers/ATmega32/MCAL/SWC\_ADC/main.c File Reference**



## H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC\_ADC/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](#) []

---

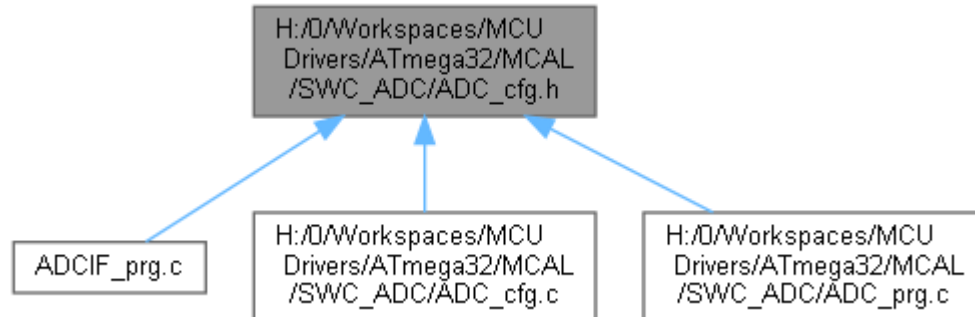
## Variable Documentation

const [u8](#) [kau8ActiveChannel\\_LGB](#) []

```
Initial value:= {  
    ADC\_CH0  
  
    , ADC\_CH1  
  
    , ADC\_CH2  
  
    , ADC\_CH3  
  
    , ADC\_CH4  
  
    , ADC\_CH5  
  
    , ADC\_CH6  
  
    , ADC\_CH7  
}
```

## H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC\_ADC/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
54 #define ADC_TRIG_SRC ADC_Free_Running_Mode
55
56 #define ADC_INIT_STATE LBTY_SET
57 #define ADC_AUTO_TRIG LBTY_RESET
58 #define ADC_IRQ_STATE LBTY_RESET
59 #define ADC_PRESCALER ADC_Division_Factor_2
60
61 #define ADC_V_REF_SRC ADC_AVCC
62 #define ADC_V_REF 5u
63
64 #define ADC_READ_DELAY 26u
65
66
67 /* ***** */
68 /* ***** CONST SECTION ***** */
69 /* ***** */
70
71 extern const u8 kau8ActiveChannel_LGB[];
72
73
74
75 /* ***** */
```

```

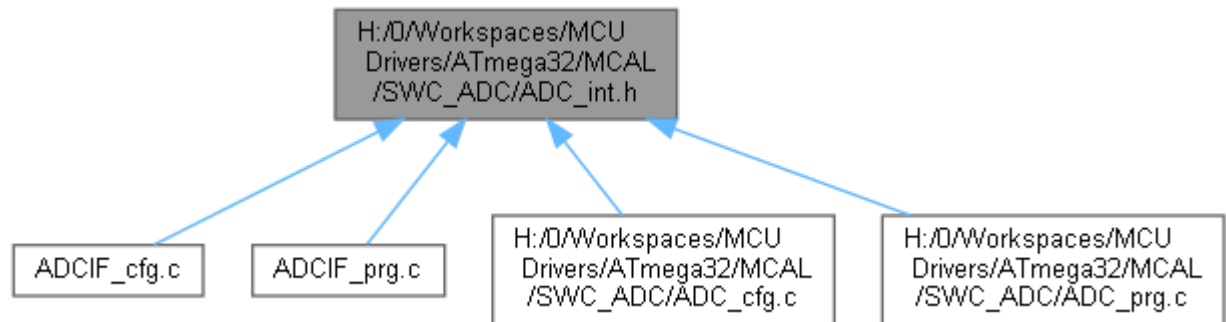
76 /* ***** VARIABLE SECTION ***** */
77 /* ***** */
78
79 /* ***** */
80 /* ***** FUNCTION SECTION ***** */
81 /* ***** */
82
83
84 #endif /* ADC_CFG_H */
85 /***** E N D (ADC_cfg.h) *****/

```

## H:/0/Workspaces/MCU

### Drivers/ATmega32/MCAL/SWC\_ADC/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\\_Overflow](#), [ADC\\_TMR1\\_Compare\\_MatchB](#), [ADC\\_TMR1\\_Overflow](#), [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_10 X	Analog Amplifier
ADC1_ADC0_10 X	
ADC0_ADC0_200 X	
ADC1_ADC0_200 X	
ADC2_ADC2_10 X	
ADC3_ADC2_10 X	
ADC2_ADC2_200 X	
ADC3_ADC2_200 X	
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_ChannelMu x	

```

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_Fa ctor_2_DEF	
ADC_Division_Fa ctor_2	
ADC_Division_Fa ctor_4	
ADC_Division_Fa ctor_8	
ADC_Division_Fa ctor_16	
ADC_Division_Fa ctor_32	
ADC_Division_Fa ctor_64	
ADC_Division_Fa ctor_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_Overflow	
ADC_TMR1_Compare_MatchB	
ADC_TMR1_Overflow	
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_Overflow,
77     ADC_TMR1_Compare_MatchB,
78     ADC_TMR1_Overflow,
79     ADC_TMR1_Capture_Event
80 }ADC_tenuTriggerSource; // ADC Auto Trigger Source

```

## Function Documentation

### [f32](#) [ADC\\_f32GetVoltage \(void \)](#)

```

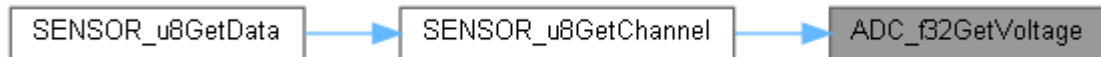
292     {
293     return (f32)ADC_u16GetData() * f32V_REF / ADC_MAX;
294 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



### LBTY\_tenuErrorStatus ADC\_u16GetAll (u16 pu16ADC\_Value[])

Clear complete flag by writing logic one

```

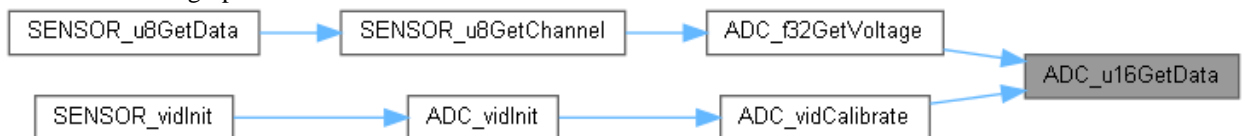
396 {
397     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
398
399     if(u8ConvDone_GLB == LBTY_SET){
400         S_ADC->m_ADCSRA.sBits.m_ADIE = LBTY_RESET;
401         S_ADC->m_ADCSRA.sBits.m_ADIF = LBTY_SET;
402         for(u8 i = 0 ; i<ADC_Num ; i++){
403             pu16ADC_Value[i] = au8ChannelValue_LGB[i];
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:



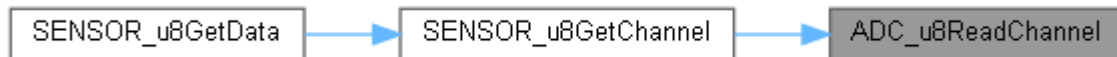
**LBTY\_tenuErrorStatus ADC\_u8ReadChannel (ADC\_tenuChannel u8Channel, u16 \* pu16ADC\_Value)**

```

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_ADSCRA.sBits.m_ADSC = LBTY_SET;
336         while(S_ADC->m_ADSCRA.sBits.m_ADSC);
337
338         //vidMyDelay_ms(ADC_READ_DELAY);
339
340         //while(!S_ADC->m_ADSCRA.sBits.m_ADIF);
341         S_ADC->m_ADSCRA.sBits.m_ADIF = LBTY_RESET;
342
343         *pu16ADC_Value = (u16)S_ADC->m_ADC;
344     }
345     return u8RetErrorState;
346 }

```

Here is the caller graph for this function:



**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_Overflow:
186     case ADC_TMR1_Compare_MatchB:
187     case ADC_TMR1_Overflow:
188     case ADC_TMR1_Capture_Event:
189         S_SFIO->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_ADSCRA.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)**

```

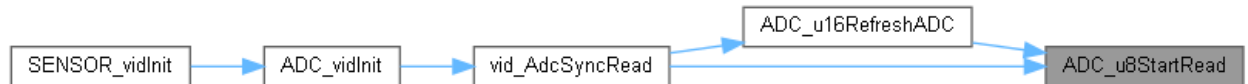
228 {
229     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
230     switch(u8Vref) {
231     case ADC_AREF:
232     case ADC_AVCC:
233     case ADC_INTERNAL_Vref:
234         S_ADC->m_ADMUX.sBits.m_REFS = u8Vref;
235         break;
236     default:
237         u8RetErrorState = LBTY_NOK;
238     }
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_ADSCRA.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_ADSCRA.sBits.m_ADSC = LBTY_RESET;
171 }
```

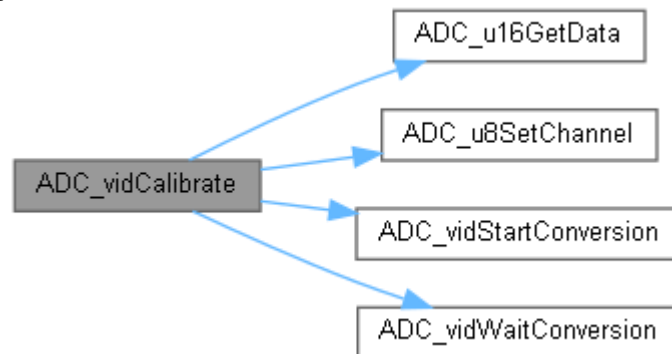
### void ADC\_vidAutoTriggerEnable (void )

```
160 {
161     S_ADC->m_ADSCRA.sBits.m_ADSC = LBTY_RESET;
162 }
```

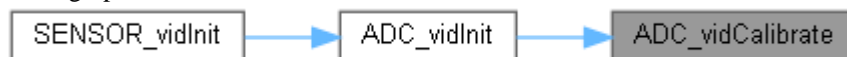
### void ADC\_vidCalibrate (void )

```
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_ADSCRA.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 )

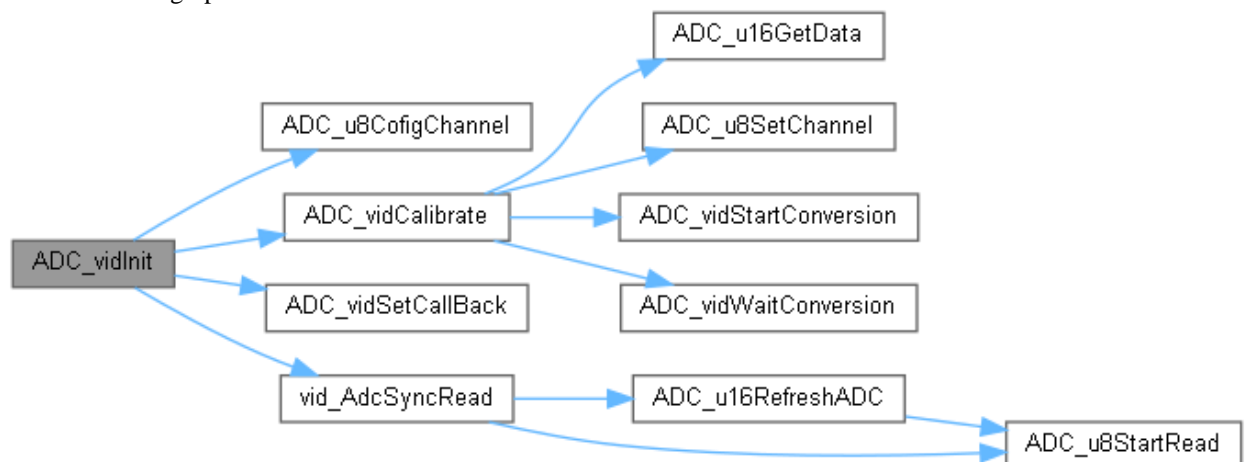
Clear complete flag by writing logic one

```

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

```

Here is the call graph for this function:



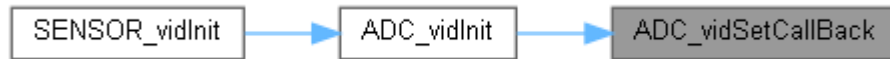
Here is the caller graph for this function:



**void ADC\_vidSetCallBack (void\*)(void) pvidCallBack)**

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

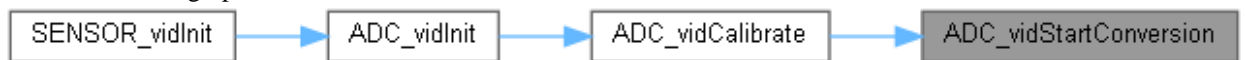
Here is the caller graph for this function:



**void ADC\_vidStartConversion (void )**

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

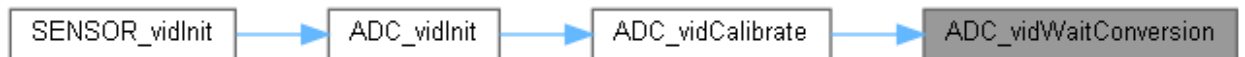
Here is the caller graph for this function:



**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{
20     ADC0 = (u8)0u,
21     ADC1,
22     ADC2,
23     ADC3,
24     ADC4,
25     ADC5,
26     ADC6,
27     ADC7,
28     ADC0 ADC0 10X,
29     ADC1 ADC0 10X,
30     ADC0 ADC0 200X,
31     ADC1 ADC0 200X,
32     ADC2 ADC2 10X,
33     ADC3 ADC2 10X,
34     ADC2 ADC2 200X,
35     ADC3 ADC2 200X,
36     ADC0 ADC1 1X,
37     ADC1 ADC1 1X,
38     ADC2 ADC1 1X,
39     ADC3 ADC1 1X,
40     ADC4 ADC1 1X,
41     ADC5 ADC1 1X,
42     ADC6 ADC1 1X,
43     ADC7 ADC1 1X,
44     ADC0 ADC2 1X,
45     ADC1 ADC2 1X,
46     ADC2 ADC2 1X,
47     ADC3 ADC2 1X,
48     ADC4 ADC2 1X,
49     ADC5 ADC2 1X,
50     VBG 1V22,
51     GND,
52     ADC ChannelMux
53 }ADC_tenuChannel; // ADC Channel Selection
54
55 typedef enum{
56     ADC Division Factor 2 DEF = (u8)0u,
57     ADC Division Factor 2,
58     ADC Division Factor 4,
59     ADC Division Factor 8,
60     ADC Division Factor 16,
61     ADC Division Factor 32,
62     ADC Division Factor 64,
63     ADC Division Factor 128
64 }ADC_tenuPrescalerSelection; // ADC Prescaler Selections
65
66 typedef enum{
67     ADC Free Running Mode = (u8)0u,
68     ADC Analog Comparator,
69     ADC External INT0,
70     ADC TMR0 Compare MatchA,
71     ADC TMR0 Overflow,
72     ADC TMR1 Compare MatchB,
```



```

78     ADC_TMR1_Overflow,
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 */

```

```

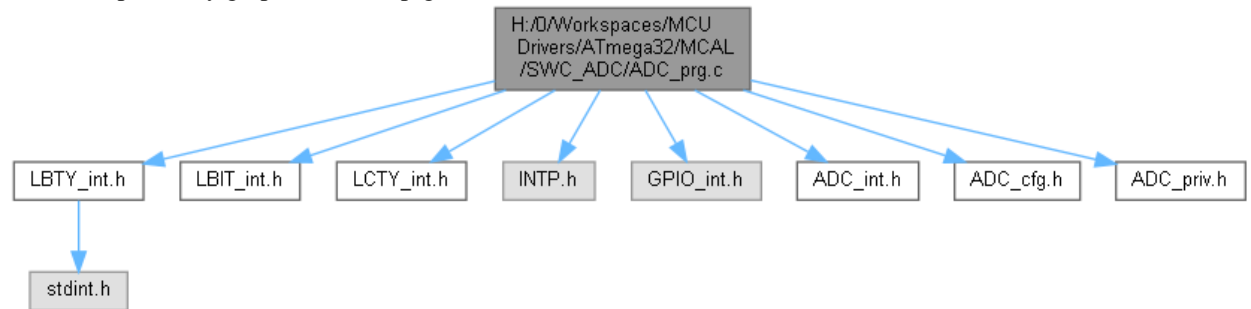
229 /* Return      :    LBTY_tenuErrorStatus */
230 /* ***** */
231 extern LBTY_tenuErrorStatus ADC_u8ReadConvValue(u8 u8Channel, u16* pu16ADC_Value);
232
233 /* ***** */
234 /* Description :    Start the ADC Interrupt Conversion */
235 /* Input      :    void */
236 /* Return     :    LBTY_tenuErrorStatus */
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) *****/

```

## H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC\_ADC/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

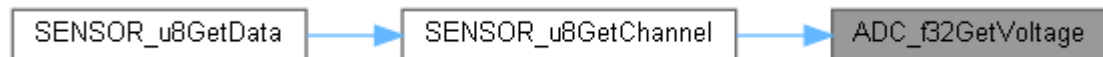
### [f32](#) [ADC\\_f32GetVoltage](#) (void )

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

Here is the call graph for this function:



Here is the caller graph for this function:



### [LBTY\\_tenuErrorStatus](#) [ADC\\_u16GetAll](#) ([u16](#) [pu16ADC\\_Value](#)[])

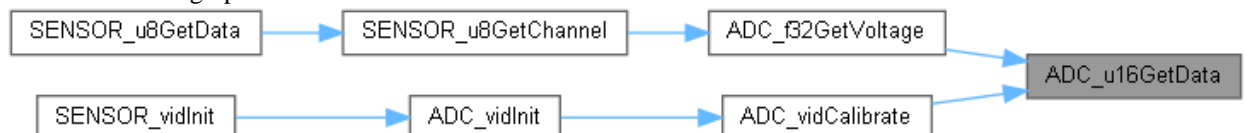
Clear complete flag by writing logic one

```
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:



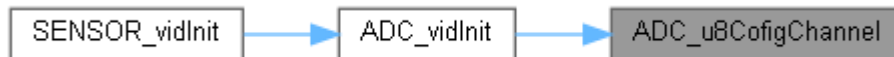
### [LBTY\\_tenuErrorStatus](#) [ADC\\_u8CofigChannel](#) ([ADC\\_tenuChannel](#) [u8Channel](#))

```

110 {
111     LBTY\_tenuErrorStatus u8RetErrorState = LBTY\_OK;
112
113     if (IS\_CHANNEL(u8Channel)) {
114         u8RetErrorState = GPIO\_u8SetPinDirection(ADC\_PORT, u8Channel,
115         PIN\_INPUT);
116     }else{
117         u8RetErrorState = LBTY\_NOK;
118     }
119     return u8RetErrorState;
120 }

```

Here is the caller graph for this function:



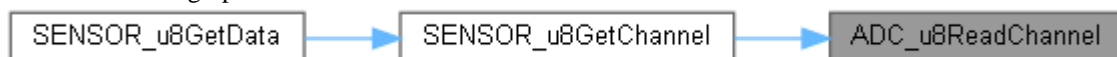
### [LBTY\\_tenuErrorStatus](#) [ADC\\_u8ReadChannel](#) ([ADC\\_tenuChannel](#) [u8Channel](#), [u16](#) \* [pu16ADC\\_Value](#))

```

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_ADSCRA.sBits.m_ADSC = LBTY\_SET;
336
337         while (S\_ADC->m_ADSCRA.sBits.m_ADSC);
338
339         //vidMyDelay_ms(ADC\_READ\_DELAY);
340
341         //while(!S\_ADC->m_ADSCRA.sBits.m_ADIF);
342         S\_ADC->m_ADSCRA.sBits.m_ADIF = LBTY\_RESET;
343
344         *pu16ADC_Value = (u16)S\_ADC->m_ADC;
345     }
346     return u8RetErrorState;
347 }

```

Here is the caller graph for this function:



### [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             *pul6ADC_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 Overflow:
186         case ADC TMR1 Compare MatchB:
187         case ADC TMR1 Overflow:
188         case ADC TMR1 Capture Event:
189             S\_SFIO->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 }

```

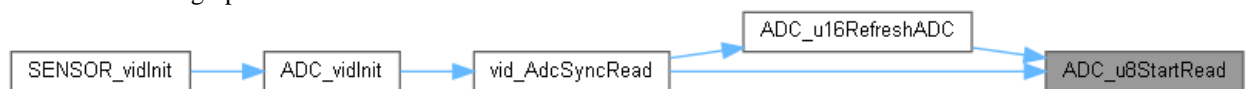
### LBTY\_tenuErrorStatus ADC\_u8StartRead (ADC\_tenuChannel u8Channel)

```

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_ADSCRA.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_ADSCRA.sBits.m_ADATE = LBTY_RESET;
171 }

```

### void ADC\_vidAutoTriggerEnable (void )

```

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

```

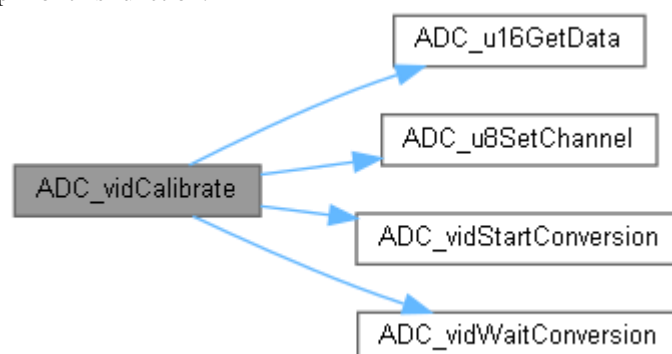
### void ADC\_vidCalibrate (void )

```

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 }

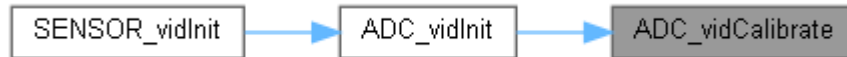
```

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 }
  
```

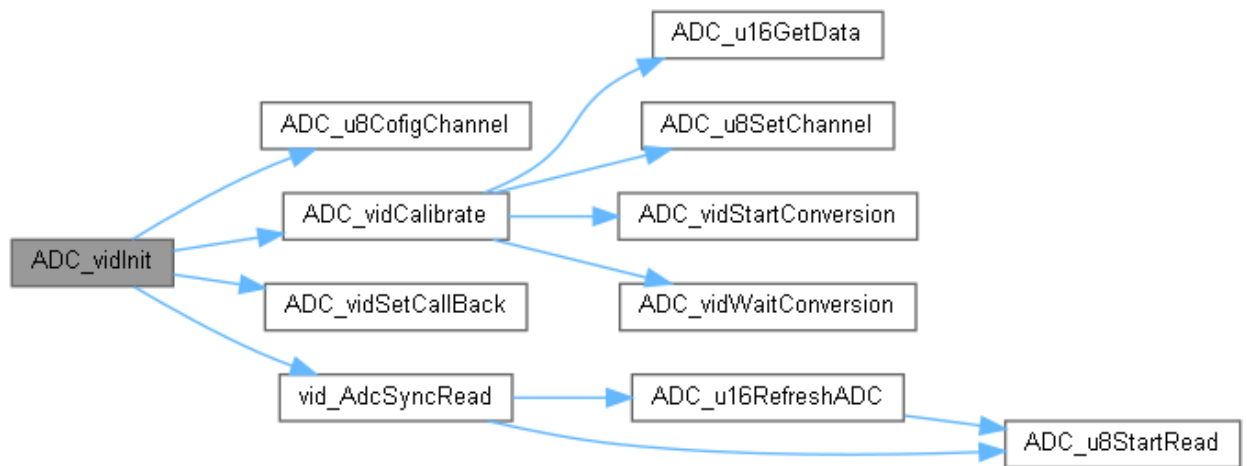
### void ADC\_vidInit (void )

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_SFIO->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:



Here is the caller graph for this function:

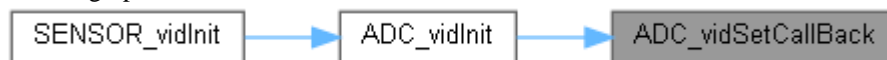


**void ADC\_vidSetCallBack (void\*)(void) pvidCallBack)**

```

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

Here is the caller graph for this function:

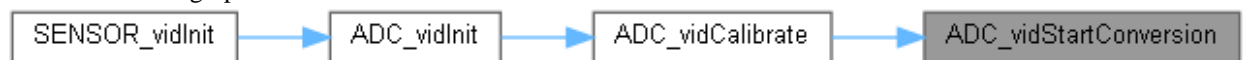


**void ADC\_vidStartConversion (void )**

```

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

Here is the caller graph for this function:

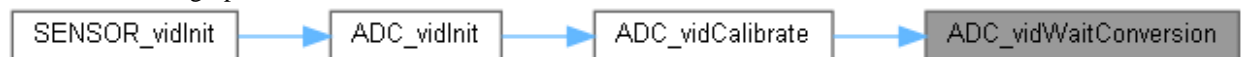


**void ADC\_vidWaitConversion (void )**

```

274 {
275     while(S_ADC->m_ADSCRA.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]**

```

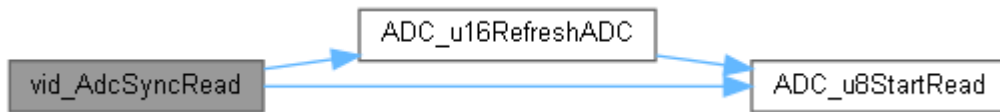
51 {
52     static u8 u8Channel = LBTY_u8ZERO;
53     if (u8Channel < ADC_Num){
54         u8ConvDone GLB = LBTY_RESET;
55         au8ChannelValue_LGB[u8Channel++] = (u16)S_ADC->m_ADC;
56         while(ADC_u8StartRead(u8Channel));
57     }else{
58     }
59 }
  
```

```

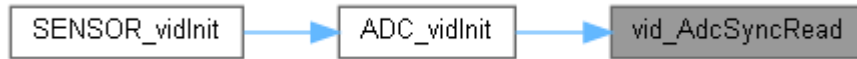
64     ADC_u16RefreshADC();
65     u8Channel = LBTY_u8ZERO;
66     u8ConvDone_GLB = LBTY_SET;
67 }
68 }

```

Here is the call graph for this function:



Here is the caller graph for this function:




---

## Variable Documentation

**u8** au8ChannelValue\_LGB[ADC\_Num][static]

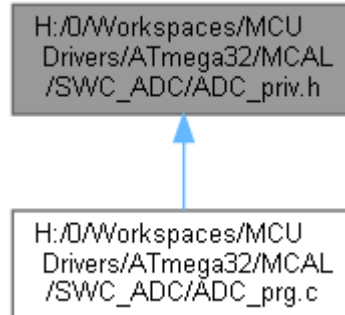
**f32** f32V\_REF = ADC\_V\_REF[static]

**void**(\* pvidFunctionCallback) (void) (void ) [static]

**u8** u8ConvDone\_GLB = LBTY\_SET [static]

## H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC\_ADC/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	<code>#ifndef ADC_CH0</code>
23	<code>ADC_0 = (<a href="#">u8</a>) 0u</code>
24	<code>#endif</code>
25	<code>#ifndef <a href="#">ADC_CH1</a></code>
26	<code>, ADC_1</code>
27	<code>#endif</code>
28	<code>#ifndef <a href="#">ADC_CH2</a></code>
29	<code>, ADC_2</code>
30	<code>#endif</code>
31	<code>#ifndef <a href="#">ADC_CH3</a></code>
32	<code>, ADC_3</code>

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

```

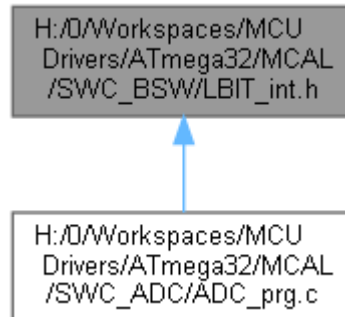
79
82 typedef union{
83     u8 u_Reg;
84     struct {
85         IO u8      : 7;
86         IO u8 m_ACD : 1;
87     }sBits;
88 }ACSR_type;
89
90 /*****
91
94 typedef struct{
95 #if ADC_ADJUSTMENT
96     I u16      : 6;
97     I u16      m_ADC : 10;
98 #else
99     I u16      m_ADC : 10;
100    I u16      : 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 }SFIO_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
132 #define S_ADC              ((ADC_type* const)0x24U)
133 #define ADCL               (*(volatile u8* const)0x24U)
134 #define ADCH               (*(volatile u8* const)0x25U)
135 #define ADCSRA             (*(volatile u8* const)0x26U)
136 #define ADMUX              (*(volatile u8* const)0x27U)
137 #define ACSR               (*(volatile u8* const)0x28U)
138
139
140 #define S_SFIO              ((SFIO_type* const)0x50U)
141 #define SFIO                (*(volatile u8* const)0x50U)
142
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) *****/

```



## H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC\_BSW/LBIT\_int.h File Reference

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



### Macros

- #define [BV](#)(bit) (1u<<(bit))
- #define [SET\\_BIT](#)(REG, bit) ((REG) |= (1u<<(bit)))
- #define [CLR\\_BIT](#)(REG, bit) ((REG) &= ~(1u<<(bit)))
- #define [TOG\\_BIT](#)(REG, bit) ((REG) ^= (1u<<(bit)))
- #define [SET\\_BYTE](#)(REG, bit) ((REG) |= (0xFFu<<(bit)))
- #define [CLR\\_BYTE](#)(REG, bit) ((REG) &= ~(0xFFu<<(bit)))
- #define [TOG\\_BYTE](#)(REG, bit) ((REG) ^= (0xFFu<<(bit)))
- #define [SET\\_MASK](#)(REG, MASK) ((REG) |= (MASK))
- #define [CLR\\_MASK](#)(REG, MASK) ((REG) &= ~(MASK))
- #define [TOG\\_MASK](#)(REG, MASK) ((REG) ^= (MASK))
- #define [GET\\_MASK](#)(REG, MASK) ((REG) & (MASK))
- #define [SET\\_REG](#)(REG) ((REG) = ~(0u))
- #define [CLR\\_REG](#)(REG) ((REG) = (0u))
- #define [TOG\\_REG](#)(REG) ((REG) ^= ~(0u))
- #define [GET\\_BIT](#)(REG, bit) (((REG)>>(bit)) & 0x01u)
- #define [GET\\_NIB](#)(REG, bit) (((REG)>>(bit)) & 0x0Fu)
- #define [GET\\_BYTE](#)(REG, bit) (((REG)>>(bit)) & 0xFFu)
- #define [ASSIGN\\_BIT](#)(REG, bit, value) ((REG) = ((REG) & ~(0x01u<<(bit))) | (((value) & 0x01u)<<(bit)))
- #define [ASSIGN\\_NIB](#)(REG, bit, value) ((REG) = ((REG) & ~(0x0Fu<<(bit))) | (((value) & 0x0Fu)<<(bit)))
- #define [ASSIGN\\_BYTE](#)(REG, bit, value) ((REG) = ((REG) & ~(0xFFu<<(bit))) | (((value) & 0xFFu)<<(bit)))
- #define [CON\\_u8Bits](#)(b7, b6, b5, b4, b3, b2, b1, b0)
 

(0b##b7##b6##b5##b4##b3##b2##b1##b0)
- #define [CON\\_u16Bits](#)(b15, b14, b13, b12, b11, b10, b9, b8, b7, b6, b5, b4, b3, b2, b1, b0)
 

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

## Macro Definition Documentation

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

**#define ASSIGN\_BIT( REG, bit, value) ((REG) = ((REG) & ~(0x01u<<(bit))) |  
(((value) & 0x01u)<<(bit)))**

**#define ASSIGN\_BYTE( REG, bit, value) ((REG) = ((REG) & ~(0xFFu<<(bit))) |  
(((value) & 0xFFu)<<(bit)))**

**#define ASSIGN\_NIB( REG, bit, value) ((REG) = ((REG) & ~(0x0Fu<<(bit))) |  
(((value) & 0x0Fu)<<(bit)))**

**#define CLR\_BIT( REG, bit) ((REG) &= ~(1u<<(bit)))**

**#define CLR\_BYTE( REG, bit) ((REG) &= ~(0xFFu<<(bit)))**

**#define CLR\_MASK( REG, MASK) ((REG) &= ~(MASK))**

**#define CLR\_REG( REG) ((REG) = (0u))**

**#define CON\_u16Bits( b15, b14, b13, b12, b11, b10, b9, b8, b7, b6, b5,  
b4, b3, b2, b1, b0)**  
**(0b##b15##b14##b13##b12##b11##b10##b9##b8##b7##b6##b5##b4##b3##b2##  
b1##b0)**

**#define CON\_u8Bits( b7, b6, b5, b4, b3, b2, b1, b0)**  
**(0b##b7##b6##b5##b4##b3##b2##b1##b0)**

**#define GET\_BIT( REG, bit) (((REG)>>(bit)) & 0x01u)**

**#define GET\_BYTE( REG, bit) (((REG)>>(bit)) & 0xFFu)**

**#define GET\_MASK( REG, MASK) ((REG) & (MASK))**

**#define GET\_NIB( REG, bit) (((REG)>>(bit)) & 0x0Fu)**

**#define SET\_BIT( REG, bit) ((REG) |= (1u<<(bit)))**

Bitwise Operation

```
#define SET_BYTE( REG, bit) ((REG) |= (0xFFu<<(bit)))  
  
#define SET_MASK( REG, MASK) ((REG) |= (MASK))  
  
#define SET_REG( REG) ((REG) = ~(0u))  
  
#define TOG_BIT( REG, bit) ((REG) ^= (1u<<(bit)))  
  
#define TOG_BYTE( REG, bit) ((REG) ^= (0xFFu<<(bit)))  
  
#define TOG_MASK( REG, MASK) ((REG) ^= (MASK))  
  
#define TOG_REG( REG) ((REG) ^= ~(0u))
```

```

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

```

```

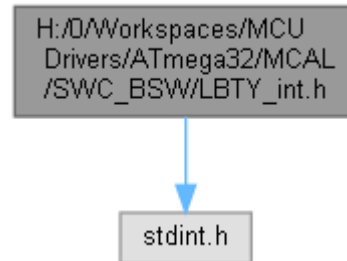
65 (0b##b15##b14##b13##b12##b11##b10##b9##b8##b7##b6##b5##b4##b3##b2##b1##b0)
66
67 /* ***** */
68 /* ***** CONST SECTION ***** */
69 /* ***** */
70
71 /* ***** */
72 /* ***** VARIABLE SECTION ***** */
73 /* ***** */
74
75 /* ***** */
76 /* ***** FUNCTION SECTION ***** */
77 /* ***** */
78
79
80 #endif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

```

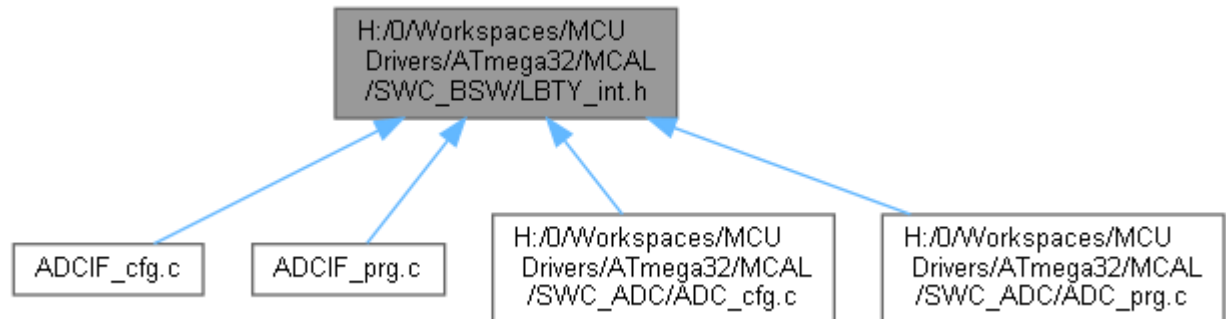
## H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC\_BSW/LBTY\_int.h File Reference

#include <stdint.h>

Include dependency graph for LBTY\_int.h:



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



## Data Structures

- union [LBTY\\_tuniPort8](#) union [LBTY\\_tuniPort16](#)

## Macros

- #define [\\_\\_IO](#) volatile
- #define [\\_\\_O](#) volatile
- #define [\\_\\_I](#) volatile const
- #define [LBTY\\_u8vidNOP\(\)](#)
- #define [LBTY\\_NULL](#) ((void \*) 0U)
- #define [LBTY\\_u8ZERO](#) ((u8)0x00U)
- #define [LBTY\\_u8MAX](#) ((u8)0xFFU)
- #define [LBTY\\_s8MAX](#) ((s8)0x7F )
- #define [LBTY\\_s8MIN](#) ((s8)0x80 )
- #define [LBTY\\_u16ZERO](#) ((u16)0x0000U)
- #define [LBTY\\_u16MAX](#) ((u16)0xFFFFU)
- #define [LBTY\\_s16MAX](#) ((u16)0x7FFF )
- #define [LBTY\\_s16MIN](#) ((u16)0x8000 )
- #define [LBTY\\_u32ZERO](#) ((u32)0x00000000UL)
- #define [LBTY\\_u32MAX](#) ((u32)0xFFFFFFFFFUL)
- #define [LBTY\\_s32MAX](#) ((u32)0x7FFFFFFFL )
- #define [LBTY\\_s32MIN](#) ((u32)0x80000000L )
- #define [LBTY\\_u64ZERO](#) ((u64)0x0000000000000000ULL)
- #define [LBTY\\_u64MAX](#) ((u64)0xFFFFFFFFFFFFFFFFFULL)
- #define [LBTY\\_s64MAX](#) ((u64)0x7FFFFFFFFFFFFFFFL )
- #define [LBTY\\_s64MIN](#) ((u64)0x8000000000000000LL )

## Typedefs

- typedef uint8\_t [u8](#)
- typedef uint16\_t [u16](#)
- typedef uint32\_t [u32](#)
- typedef uint64\_t [u64](#)
- typedef int8\_t [s8](#)
- typedef int16\_t [s16](#)
- typedef int32\_t [s32](#)
- typedef int64\_t [s64](#)
- typedef float [f32](#)
- typedef double [f64](#)
- typedef [u8](#) \* [pu8](#)
- typedef [u16](#) \* [pu16](#)
- typedef [u32](#) \* [pu32](#)
- typedef [u64](#) \* [pu64](#)
- typedef [s8](#) \* [ps8](#)
- typedef [s16](#) \* [ps16](#)
- typedef [s32](#) \* [ps32](#)
- typedef [s64](#) \* [ps64](#)

## Enumerations

- enum [LBTY\\_tenuFlagStatus](#) { [LBTY\\_RESET](#) = 0, [LBTY\\_SET](#) = ![LBTY\\_RESET](#) }
  - enum [LBTY\\_tenuBoolean](#) { [LBTY\\_TRUE](#) = 0x55, [LBTY\\_FALSE](#) = 0xAA }
  - enum [LBTY\\_tenuErrorStatus](#) { [LBTY\\_OK](#) = (u16)0, [LBTY\\_NOK](#), [LBTY\\_NULL\\_POINTER](#), [LBTY\\_INDEX\\_OUT\\_OF\\_RANGE](#), [LBTY\\_NO\\_MASTER\\_CHANNEL](#), [LBTY\\_READ\\_ERROR](#), [LBTY\\_WRITE\\_ERROR](#), [LBTY\\_UNDEFINED\\_ERROR](#), [LBTY\\_IN\\_PROGRESS](#) }
-

## Macro Definition Documentation

**#define** `__I` `volatile const`

**#define** `__IO` `volatile`

**#define** `__O` `volatile`

**#define** `LBTY_NULL` `((void *) 0U)`

**#define** `LBTY_s16MAX` `((u16)0x7FFF )`

**#define** `LBTY_s16MIN` `((u16)0x8000 )`

**#define** `LBTY_s32MAX` `((u32)0x7FFFFFFFL )`

**#define** `LBTY_s32MIN` `((u32)0x80000000L )`

**#define** `LBTY_s64MAX` `((u64)0x7FFFFFFFFFFFFFFFL )`

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

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

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

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

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

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

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

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

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

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

**#define** `LBTY_u8vidNOP()`

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

Data Types Limitation

---

## Typedef Documentation

**typedef** `float` [f32](#)

Standard Real Decimal number



**typedef double [f64](#)**

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

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

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

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

Standard Pointer to Signed Byte/Word/Long\_Word

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

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

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

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

Standard Pointer to Unsigned Byte/Word/Long\_Word

**typedef int16\_t [s16](#)**

**typedef int32\_t [s32](#)**

**typedef int64\_t [s64](#)**

**typedef int8\_t [s8](#)**

Standard Signed Byte/Word/Long\_Word

**typedef uint16\_t [u16](#)**

**typedef uint32\_t [u32](#)**

**typedef uint64\_t [u64](#)**

**typedef uint8\_t [u8](#)**

Data Types New Definitions Standard Unsigned Byte/Word/Long\_Word

---

## Enumeration Type Documentation

**enum [LBTY\\_tenuBoolean](#)**

Boolean type

**Enumerator:**

	<a href="#">LBTY_TRUE</a>	
	<a href="#">LBTY_FALSE</a>	

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

enum [LBTY\\_tenuErrorStatus](#)

Error Return type

Enumerator:

LBTY_OK	
LBTY_NOK	
LBTY_NULL_POINTER	
LBTY_INDEX_OUT_OF_RANGE	
LBTY_NO_MASTER_CHANNEL	
LBTY_READ_ERROR	
LBTY_WRITE_ERROR	
LBTY_UNDEFINED_ERROR	
LBTY_IN_PROGRESS	

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

enum [LBTY\\_tenuFlagStatus](#)

Flag Status type

Enumerator:

LBTY_RESET	
LBTY_SET	

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

## LBTY\_int.h

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

```

```

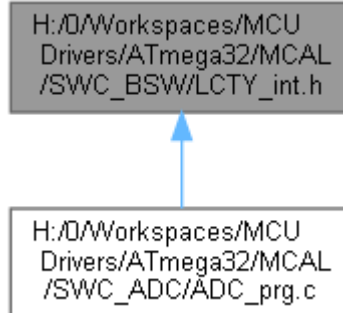
80 #define LBTY_u64ZERO      ((u64)0x0000000000000000ULL)
81 #define LBTY_u64MAX       ((u64)0xFFFFFFFFFFFFFFFFULL)
82 #define LBTY_s64MAX       ((u64)0x7FFFFFFFFFFFFFFFLL )
83 #define LBTY_s64MIN       ((u64)0x8000000000000000LL )
84
85 /* ***** */
86 /* ***** ENUM SECTION ***** */
87 /* ***** */
88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

```

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

## H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC\_BSW/LCTY\_int.h File Reference

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



### Macros

- #define [LCTY\\_PROGMEM](#) \_\_attribute\_\_((\_\_progmem\_\_))
  - #define [LCTY\\_PURE](#) \_\_attribute\_\_((\_\_pure\_\_))
  - #define [LCTY\\_INLINE](#) \_\_attribute\_\_((always\_inline)) static inline
  - #define [LCTY\\_INTERRUPT](#) \_\_attribute\_\_((interrupt))
  - #define [CTY\\_PACKED](#) \_\_attribute\_\_((\_\_packed\_\_))
  - #define [LCTY\\_CONST](#) \_\_attribute\_\_((\_\_const\_\_))
  - #define [LCTY\\_DPAGE](#) \_\_attribute\_\_((dp))
  - #define [LCTY\\_NODPAGE](#) \_\_attribute\_\_((nodp))
  - #define [LCTY\\_SECTION](#)(section) \_\_attribute\_\_((section( # section)))
  - #define [LCTY\\_ASM](#)(cmd) \_\_asm\_\_ \_\_volatile\_\_ ( # cmd ::)
-

## Macro Definition Documentation

**#define CTY\_PACKED \_\_attribute\_\_((packed))**

**#define LCTY\_ASM( cmd) \_\_asm\_\_ \_\_volatile\_\_ ( # cmd ::)**

**#define LCTY\_CONST \_\_attribute\_\_((const))**

**#define LCTY\_DPAGE \_\_attribute\_\_((dp))**

**#define LCTY\_INLINE \_\_attribute\_\_((always\_inline)) static inline**

**#define LCTY\_INTERRUPT \_\_attribute\_\_((interrupt))**

**#define LCTY\_NODPAGE \_\_attribute\_\_((nodp))**

**#define LCTY\_PROGMEM \_\_attribute\_\_((\_\_progmem\_\_))**

**#define LCTY\_PURE \_\_attribute\_\_((\_\_pure\_\_))**

**#define LCTY\_SECTION( section) \_\_attribute\_\_((section( # section)))**

## LCTY\_int.h

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