

ADS1115 Driver

1.0.0

Generated by Doxygen 1.14.0

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	5
3.1 ads1115_config_t Struct Reference	5
3.1.1 Detailed Description	5
3.1.2 Member Data Documentation	5
3.1.2.1 comp_latch	5
3.1.2.2 comp_mode	6
3.1.2.3 comp_pol	6
3.1.2.4 comp_queue	6
3.1.2.5 data_rate	6
3.1.2.6 high_threshold	6
3.1.2.7 low_threshold	6
3.1.2.8 mode	7
3.1.2.9 mux	7
3.1.2.10 range	7
3.2 ads1115_handle_t Struct Reference	7
3.2.1 Detailed Description	7
3.2.2 Member Data Documentation	8
3.2.2.1 config	8
3.2.2.2 delay_ms	8
3.2.2.3 i2c_addr	8
3.2.2.4 i2c_read	8
3.2.2.5 i2c_write	8
3.2.2.6 is_initialized	8
4 File Documentation	9
4.1 interface/ads1115_interface.c File Reference	9
4.1.1 Detailed Description	9
4.1.2 Function Documentation	9
4.1.2.1 ads1115_delay_ms()	9
4.1.2.2 ads1115_i2c_read()	10
4.1.2.3 ads1115_i2c_write()	10
4.2 ads1115_interface.c	11
4.3 interface/ads1115_interface.h File Reference	11
4.3.1 Detailed Description	11
4.3.2 Function Documentation	11
4.3.2.1 ads1115_delay_ms()	11
4.3.2.2 ads1115_i2c_read()	12

4.3.2.3 ads1115_i2c_write()	12
4.4 ads1115_interface.h	13
4.5 src/ads1115.c File Reference	13
4.5.1 Detailed Description	15
4.5.2 Macro Definition Documentation	15
4.5.2.1 ADS1115_COMP_LAT_MASK	15
4.5.2.2 ADS1115_COMP_LAT_SHIFT	15
4.5.2.3 ADS1115_COMP_MODE_MASK	15
4.5.2.4 ADS1115_COMP_MODE_SHIFT	15
4.5.2.5 ADS1115_COMP_POL_MASK	16
4.5.2.6 ADS1115_COMP_POL_SHIFT	16
4.5.2.7 ADS1115_COMP_QUE_MASK	16
4.5.2.8 ADS1115_COMP_QUE_SHIFT	16
4.5.2.9 ADS1115_DR_MASK	16
4.5.2.10 ADS1115_DR_SHIFT	16
4.5.2.11 ADS1115_MAX_VALUE	16
4.5.2.12 ADS1115_MIN_VALUE	16
4.5.2.13 ADS1115_MODE_MASK	17
4.5.2.14 ADS1115_MODE_SHIFT	17
4.5.2.15 ADS1115_MUX_MASK	17
4.5.2.16 ADS1115_MUX_SHIFT	17
4.5.2.17 ADS1115_OS_IDLE	17
4.5.2.18 ADS1115_OS_MASK	17
4.5.2.19 ADS1115_OS_SHIFT	17
4.5.2.20 ADS1115_OS_START_SINGLE	17
4.5.2.21 ADS1115_PGA_MASK	18
4.5.2.22 ADS1115_PGA_SHIFT	18
4.5.2.23 ADS1115_REG_CONFIG	18
4.5.2.24 ADS1115_REG_CONVERSION	18
4.5.2.25 ADS1115_REG_HI_THRESH	18
4.5.2.26 ADS1115_REG_LO_THRESH	18
4.5.3 Function Documentation	18
4.5.3.1 ads1115_continuous_conversion_read()	18
4.5.3.2 ads1115_continuous_conversion_start()	19
4.5.3.3 ads1115_continuous_conversion_stop()	19
4.5.3.4 ads1115_deinit()	19
4.5.3.5 ads1115_get_channel()	20
4.5.3.6 ads1115_get_compare_alert()	20
4.5.3.7 ads1115_get_compare_latch()	20
4.5.3.8 ads1115_get_compare_mode()	21
4.5.3.9 ads1115_get_compare_queue()	21
4.5.3.10 ads1115_get_compare_threshold()	21

4.5.3.11 ads1115_get_data_rate()	22
4.5.3.12 ads1115_get_range()	22
4.5.3.13 ads1115_init()	22
4.5.3.14 ads1115_is_ready()	23
4.5.3.15 ads1115_set_channel()	23
4.5.3.16 ads1115_set_compare_alert()	23
4.5.3.17 ads1115_set_compare_latch()	24
4.5.3.18 ads1115_set_compare_mode()	24
4.5.3.19 ads1115_set_compare_queue()	24
4.5.3.20 ads1115_set_compare_threshold()	25
4.5.3.21 ads1115_set_data_rate()	25
4.5.3.22 ads1115_set_range()	25
4.5.3.23 ads1115_single_read()	26
4.6 ads1115.c	26
4.7 src/ads1115.h File Reference	34
4.7.1 Detailed Description	37
4.7.2 Macro Definition Documentation	37
4.7.2.1 ADS1115_DEFAULT_CONFIGURATION	37
4.7.3 Typedef Documentation	37
4.7.3.1 ads1115_delay_ms_t	37
4.7.3.2 ads1115_i2c_read_t	38
4.7.3.3 ads1115_i2c_write_t	38
4.7.4 Enumeration Type Documentation	38
4.7.4.1 ads1115_comp_latch_t	38
4.7.4.2 ads1115_comp_mode_t	39
4.7.4.3 ads1115_comp_polarity_t	39
4.7.4.4 ads1115_comp_queue_t	39
4.7.4.5 ads1115_data_rate_t	39
4.7.4.6 ads1115_error_t	40
4.7.4.7 ads1115_i2c_addr_t	40
4.7.4.8 ads1115_mode_t	40
4.7.4.9 ads1115_mux_t	41
4.7.4.10 ads1115_range_t	41
4.7.5 Function Documentation	41
4.7.5.1 ads1115_continuous_conversion_read()	41
4.7.5.2 ads1115_continuous_conversion_start()	42
4.7.5.3 ads1115_continuous_conversion_stop()	42
4.7.5.4 ads1115_deinit()	42
4.7.5.5 ads1115_get_channel()	43
4.7.5.6 ads1115_get_compare_alert()	43
4.7.5.7 ads1115_get_compare_latch()	43
4.7.5.8 ads1115_get_compare_mode()	44

4.7.5.9 ads1115_get_compare_queue()	44
4.7.5.10 ads1115_get_compare_threshold()	44
4.7.5.11 ads1115_get_data_rate()	45
4.7.5.12 ads1115_get_range()	45
4.7.5.13 ads1115_init()	45
4.7.5.14 ads1115_is_ready()	46
4.7.5.15 ads1115_set_channel()	46
4.7.5.16 ads1115_set_compare_alert()	46
4.7.5.17 ads1115_set_compare_latch()	47
4.7.5.18 ads1115_set_compare_mode()	47
4.7.5.19 ads1115_set_compare_queue()	47
4.7.5.20 ads1115_set_compare_threshold()	48
4.7.5.21 ads1115_set_data_rate()	48
4.7.5.22 ads1115_set_range()	48
4.7.5.23 ads1115_single_read()	49
4.8 ads1115.h	49

Index	53
--------------	-----------

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

ads1115_config_t	ADS1115 configuration structure	5
ads1115_handle_t	ADS1115 device handle structure	7

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

interface/	ads1115_interface.c	ADS1115 I2C read/write and delay callback function implementations	9
interface/	ads1115_interface.h	ADS1115 I2C read/write and delay callback function declarations	11
src/	ads1115.c	ADS1115 16-Bit ADC Driver Implementation	13
src/	ads1115.h	ADS1115 16-Bit ADC Driver - Platform Independent	34

Chapter 3

Class Documentation

3.1 ads1115_config_t Struct Reference

ADS1115 configuration structure.

```
#include <ads1115.h>
```

Public Attributes

- `ads1115_mux_t mux`
- `ads1115_range_t range`
- `ads1115_mode_t mode`
- `ads1115_data_rate_t data_rate`
- `ads1115_comp_mode_t comp_mode`
- `ads1115_comp_polarity_t comp_pol`
- `ads1115_comp_latch_t comp_latch`
- `ads1115_comp_queue_t comp_queue`
- `int16_t low_threshold`
- `int16_t high_threshold`

3.1.1 Detailed Description

ADS1115 configuration structure.

Definition at line 172 of file `ads1115.h`.

3.1.2 Member Data Documentation

3.1.2.1 comp_latch

```
ads1115_comp_latch_t ads1115_config_t::comp_latch
```

Comparator latching

Definition at line 179 of file `ads1115.h`.

3.1.2.2 **comp_mode**

`ads1115_comp_mode_t ads1115_config_t::comp_mode`

Comparator mode

Definition at line 177 of file [ads1115.h](#).

3.1.2.3 **comp_pol**

`ads1115_comp_polarity_t ads1115_config_t::comp_pol`

Comparator polarity

Definition at line 178 of file [ads1115.h](#).

3.1.2.4 **comp_queue**

`ads1115_comp_queue_t ads1115_config_t::comp_queue`

Comparator queue

Definition at line 180 of file [ads1115.h](#).

3.1.2.5 **data_rate**

`ads1115_data_rate_t ads1115_config_t::data_rate`

Data rate (samples per second)

Definition at line 176 of file [ads1115.h](#).

3.1.2.6 **high_threshold**

`int16_t ads1115_config_t::high_threshold`

High threshold value

Definition at line 182 of file [ads1115.h](#).

3.1.2.7 **low_threshold**

`int16_t ads1115_config_t::low_threshold`

Low threshold value

Definition at line 181 of file [ads1115.h](#).

3.1.2.8 mode

`ads1115_mode_t ads1115_config_t::mode`

Operating mode

Definition at line 175 of file [ads1115.h](#).

3.1.2.9 mux

`ads1115_mux_t ads1115_config_t::mux`

Input multiplexer configuration

Definition at line 173 of file [ads1115.h](#).

3.1.2.10 range

`ads1115_range_t ads1115_config_t::range`

Programmable gain amplifier

Definition at line 174 of file [ads1115.h](#).

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

- src/[ads1115.h](#)

3.2 ads1115_handle_t Struct Reference

ADS1115 device handle structure.

```
#include <ads1115.h>
```

Public Attributes

- `ads1115_i2c_addr_t i2c_addr`
- `ads1115_config_t config`
- `ads1115_i2c_write_t i2c_write`
- `ads1115_i2c_read_t i2c_read`
- `ads1115_delay_ms_t delay_ms`
- bool `is_initialized`

3.2.1 Detailed Description

ADS1115 device handle structure.

Definition at line 188 of file [ads1115.h](#).

3.2.2 Member Data Documentation

3.2.2.1 config

`ads1115_config_t ads1115_handle_t::config`

Current device configuration

Definition at line 190 of file [ads1115.h](#).

3.2.2.2 delay_ms

`ads1115_delay_ms_t ads1115_handle_t::delay_ms`

I2C device address

Definition at line 193 of file [ads1115.h](#).

3.2.2.3 i2c_addr

`ads1115_i2c_addr_t ads1115_handle_t::i2c_addr`

I2C device address

Definition at line 189 of file [ads1115.h](#).

3.2.2.4 i2c_read

`ads1115_i2c_read_t ads1115_handle_t::i2c_read`

I2C read callback

Definition at line 192 of file [ads1115.h](#).

3.2.2.5 i2c_write

`ads1115_i2c_write_t ads1115_handle_t::i2c_write`

I2C write callback

Definition at line 191 of file [ads1115.h](#).

3.2.2.6 is_initialized

`bool ads1115_handle_t::is_initialized`

Initialization status

Definition at line 194 of file [ads1115.h](#).

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

- src/[ads1115.h](#)

Chapter 4

File Documentation

4.1 `interface/ads1115_interface.c` File Reference

ADS1115 I2C read/write and delay callback function implementations.

```
#include "ads1115_interface.h"
```

Functions

- `uint8_t ads1115_i2c_write` (`uint8_t i2c_addr, uint8_t reg_addr, uint8_t *data, uint16_t len)`
ADS1115 I2C write function (`ads1115_i2c_write_t` callback)
- `uint8_t ads1115_i2c_read` (`uint8_t i2c_addr, uint8_t reg_addr, uint8_t *data, uint16_t len)`
ADS1115 I2C read function (`ads1115_i2c_read_t` callback)
- `void ads1115_delay_ms` (`uint32_t milliseconds)`
ADS1115 delay_ms function (`ads1115_delay_ms_t` callback)

4.1.1 Detailed Description

ADS1115 I2C read/write and delay callback function implementations.

Author

Şükrü Can Kılıç

Date

19.01.2026

Definition in file [ads1115_interface.c](#).

4.1.2 Function Documentation

4.1.2.1 `ads1115_delay_ms()`

```
void ads1115_delay_ms (
    uint32_t milliseconds)
```

ADS1115 delay_ms function (`ads1115_delay_ms_t` callback)

Parameters

<i>milliseconds</i>	Delay time in milliseconds
---------------------	----------------------------

Definition at line 24 of file [ads1115_interface.c](#).

4.1.2.2 ads1115_i2c_read()

```
uint8_t ads1115_i2c_read (
    uint8_t i2c_addr,
    uint8_t reg_addr,
    uint8_t * data,
    uint16_t len)
```

ADS1115 I2C read function ([ads1115_i2c_read_t](#) callback)

Parameters

<i>i2c_addr</i>	ADS1115 I2C address
<i>reg_addr</i>	ADS1115 register address
<i>data</i>	Pointer to register value
<i>len</i>	Length of register value

Returns

Error code

Definition at line 17 of file [ads1115_interface.c](#).

4.1.2.3 ads1115_i2c_write()

```
uint8_t ads1115_i2c_write (
    uint8_t i2c_addr,
    uint8_t reg_addr,
    uint8_t * data,
    uint16_t len)
```

ADS1115 I2C write function ([ads1115_i2c_write_t](#) callback)

Parameters

<i>i2c_addr</i>	ADS1115 I2C address
<i>reg_addr</i>	ADS1115 register address
<i>data</i>	Pointer to register value
<i>len</i>	Length of register value

Returns

Error code

Definition at line 10 of file [ads1115_interface.c](#).

4.2 ads1115_interface.c

[Go to the documentation of this file.](#)

```

00001
00007
00008 #include "ads1115_interface.h"
00009
00010 uint8_t ads1115_i2c_write(uint8_t i2c_addr, uint8_t reg_addr, uint8_t *data, uint16_t len)
00011 {
00012     /* Write your platform code for I2C write function (STM32, ESP32, ARDUINO etc.) */
00013
00014     return 0;
00015 }
00016
00017 uint8_t ads1115_i2c_read(uint8_t i2c_addr, uint8_t reg_addr, uint8_t *data, uint16_t len)
00018 {
00019     /* Write your platform code for I2C read function (STM32, ESP32, ARDUINO etc.) */
00020
00021     return 0;
00022 }
00023
00024 void ads1115_delay_ms(uint32_t milliseconds)
00025 {
00026     /* Write your platform code for delay in milliseconds function (STM32, ESP32, ARDUINO etc.) */
00027 }
```

4.3 interface/ads1115_interface.h File Reference

ADS1115 I2C read/write and delay callback function declarations.

```
#include "ads1115.h"
```

Functions

- `uint8_t ads1115_i2c_write (uint8_t i2c_addr, uint8_t reg_addr, uint8_t *data, uint16_t len)`
ADS1115 I2C write function (`ads1115_i2c_write_t` callback)
- `uint8_t ads1115_i2c_read (uint8_t i2c_addr, uint8_t reg_addr, uint8_t *data, uint16_t len)`
ADS1115 I2C read function (`ads1115_i2c_read_t` callback)
- `void ads1115_delay_ms (uint32_t milliseconds)`
ADS1115 delay_ms function (`ads1115_delay_ms_t` callback)

4.3.1 Detailed Description

ADS1115 I2C read/write and delay callback function declarations.

Author

Şükrü Can Kılıç

Date

19.01.2026

Definition in file [ads1115_interface.h](#).

4.3.2 Function Documentation

4.3.2.1 ads1115_delay_ms()

```

void ads1115_delay_ms (
    uint32_t milliseconds)
```

ADS1115 delay_ms function (`ads1115_delay_ms_t` callback)

Parameters

<i>milliseconds</i>	Delay time in milliseconds
---------------------	----------------------------

Definition at line 24 of file [ads1115_interface.c](#).

4.3.2.2 ads1115_i2c_read()

```
uint8_t ads1115_i2c_read (
    uint8_t i2c_addr,
    uint8_t reg_addr,
    uint8_t * data,
    uint16_t len)
```

ADS1115 I2C read function ([ads1115_i2c_read_t](#) callback)

Parameters

<i>i2c_addr</i>	ADS1115 I2C address
<i>reg_addr</i>	ADS1115 register address
<i>data</i>	Pointer to register value
<i>len</i>	Length of register value

Returns

Error code

Definition at line 17 of file [ads1115_interface.c](#).

4.3.2.3 ads1115_i2c_write()

```
uint8_t ads1115_i2c_write (
    uint8_t i2c_addr,
    uint8_t reg_addr,
    uint8_t * data,
    uint16_t len)
```

ADS1115 I2C write function ([ads1115_i2c_write_t](#) callback)

Parameters

<i>i2c_addr</i>	ADS1115 I2C address
<i>reg_addr</i>	ADS1115 register address
<i>data</i>	Pointer to register value
<i>len</i>	Length of register value

Returns

Error code

Definition at line 10 of file [ads1115_interface.c](#).

4.4 ads1115_interface.h

[Go to the documentation of this file.](#)

```
00001
00007
00008 #include "ads1115.h"
00009
00019 uint8_t ads1115_i2c_write(uint8_t i2c_addr, uint8_t reg_addr, uint8_t *data, uint16_t len);
00020
00030 uint8_t ads1115_i2c_read(uint8_t i2c_addr, uint8_t reg_addr, uint8_t *data, uint16_t len);
00031
00037 void ads1115_delay_ms(uint32_t milliseconds);
```

4.5 src/ads1115.c File Reference

ADS1115 16-Bit ADC Driver Implementation.

```
#include "ads1115.h"
#include <stddef.h>
```

Macros

- `#define ADS1115_REG_CONVERSION 0x00`
ADS1115 Register addresses.
- `#define ADS1115_REG_CONFIG 0x01`
- `#define ADS1115_REG_LO_THRESH 0x02`
- `#define ADS1115_REG_HI_THRESH 0x03`
- `#define ADS1115_OS_MASK 0x8000`
- `#define ADS1115_OS_SHIFT 15`
- `#define ADS1115_OS_IDLE 0x8000`
- `#define ADS1115_OS_START_SINGLE 0x8000`
- `#define ADS1115_MUX_MASK 0x7000`
- `#define ADS1115_MUX_SHIFT 12`
- `#define ADS1115_PGA_MASK 0x0E00`
- `#define ADS1115_PGA_SHIFT 9`
- `#define ADS1115_MODE_MASK 0x0100`
- `#define ADS1115_MODE_SHIFT 8`
- `#define ADS1115_DR_MASK 0x00E0`
- `#define ADS1115_DR_SHIFT 5`
- `#define ADS1115_COMP_MODE_MASK 0x0010`
- `#define ADS1115_COMP_MODE_SHIFT 4`
- `#define ADS1115_COMP_POL_MASK 0x0008`
- `#define ADS1115_COMP_POL_SHIFT 3`
- `#define ADS1115_COMP_LAT_MASK 0x0004`
- `#define ADS1115_COMP_LAT_SHIFT 2`
- `#define ADS1115_COMP_QUE_MASK 0x0003`
- `#define ADS1115_COMP_QUE_SHIFT 0`
- `#define ADS1115_MAX_VALUE 32767`
- `#define ADS1115_MIN_VALUE -32768`

Functions

- `ads1115_error_t ads1115_init (ads1115_handle_t *handle)`
Initialize ADS1115 device.
- `ads1115_error_t ads1115_deinit (ads1115_handle_t *handle)`
De-initialize ADS1115 device.
- `ads1115_error_t ads1115_set_range (ads1115_handle_t *handle, ads1115_range_t range)`
Set range of adc.
- `ads1115_error_t ads1115_get_range (ads1115_handle_t *handle, ads1115_range_t *range)`
Get range of adc.
- `ads1115_error_t ads1115_set_data_rate (ads1115_handle_t *handle, ads1115_data_rate_t data_rate)`
Set adc data sample rate.
- `ads1115_error_t ads1115_get_data_rate (ads1115_handle_t *handle, ads1115_data_rate_t *data_rate)`
Get adc data sample rate.
- `ads1115_error_t ads1115_continuous_conversion_start (ads1115_handle_t *handle)`
Start adc continuous conversion mode.
- `ads1115_error_t ads1115_continuous_conversion_stop (ads1115_handle_t *handle)`
Stop adc continuous conversion mode.
- `ads1115_error_t ads1115_continuous_conversion_read (ads1115_handle_t *handle, int16_t *adc_raw, float *voltage)`
Read the result of adc continuously.
- `ads1115_error_t ads1115_single_read (ads1115_handle_t *handle, int16_t *adc_raw, float *voltage)`
Read adc result once.
- `ads1115_error_t ads1115_set_channel (ads1115_handle_t *handle, ads1115_mux_t channel)`
Set adc input channel from mux.
- `ads1115_error_t ads1115_get_channel (ads1115_handle_t *handle, ads1115_mux_t *channel)`
Set adc input channel from mux.
- `ads1115_error_t ads1115_set_compare_mode (ads1115_handle_t *handle, ads1115_comp_mode_t compare)`
Set comparator mode.
- `ads1115_error_t ads1115_get_compare_mode (ads1115_handle_t *handle, ads1115_comp_mode_t *compare)`
Get current comparator mode from device.
- `ads1115_error_t ads1115_set_compare_queue (ads1115_handle_t *handle, ads1115_comp_queue_t comp_queue)`
Set comparatore queue.
- `ads1115_error_t ads1115_get_compare_queue (ads1115_handle_t *handle, ads1115_comp_queue_t *comp_queue)`
Get comparatore queue.
- `ads1115_error_t ads1115_set_compare_latch (ads1115_handle_t *handle, ads1115_comp_latch_t latch)`
Set comparator latching mode.
- `ads1115_error_t ads1115_get_compare_latch (ads1115_handle_t *handle, ads1115_comp_latch_t *latch)`
Get comparator latching mode.
- `ads1115_error_t ads1115_set_compare_alert (ads1115_handle_t *handle, ads1115_comp_polarity_t polarity)`
Set the alert pin polatiry mode (ACTIVE_HIGH or ACTIVE_LOW)
- `ads1115_error_t ads1115_get_compare_alert (ads1115_handle_t *handle, ads1115_comp_polarity_t *polarity)`
Get the alert pin polatiry (ACTIVE_HIGH or ACTIVE_LOW)
- `ads1115_error_t ads1115_set_compare_threshold (ads1115_handle_t *handle, int16_t low_threshold, int16_t high_threshold)`
Set the comparator low and high threshold.

- `ads1115_error_t ads1115_get_compare_threshold (ads1115_handle_t *handle, int16_t *low_threshold, int16_t *high_threshold)`
Get the comparator low and high threshold.
- `ads1115_error_t ads1115_is_ready (ads1115_handle_t *handle, bool *flag)`
Check if conversion is ready.

4.5.1 Detailed Description

ADS1115 16-Bit ADC Driver Implementation.

Author

Şükrü Can Kılıç

Date

19.01.2026

Definition in file [ads1115.c](#).

4.5.2 Macro Definition Documentation

4.5.2.1 ADS1115_COMP_LAT_MASK

```
#define ADS1115_COMP_LAT_MASK 0x0004
```

Definition at line [58](#) of file [ads1115.c](#).

4.5.2.2 ADS1115_COMP_LAT_SHIFT

```
#define ADS1115_COMP_LAT_SHIFT 2
```

Definition at line [59](#) of file [ads1115.c](#).

4.5.2.3 ADS1115_COMP_MODE_MASK

```
#define ADS1115_COMP_MODE_MASK 0x0010
```

Definition at line [50](#) of file [ads1115.c](#).

4.5.2.4 ADS1115_COMP_MODE_SHIFT

```
#define ADS1115_COMP_MODE_SHIFT 4
```

Definition at line [51](#) of file [ads1115.c](#).

4.5.2.5 ADS1115_COMP_POL_MASK

```
#define ADS1115_COMP_POL_MASK 0x0008
```

Definition at line [54](#) of file [ads1115.c](#).

4.5.2.6 ADS1115_COMP_POL_SHIFT

```
#define ADS1115_COMP_POL_SHIFT 3
```

Definition at line [55](#) of file [ads1115.c](#).

4.5.2.7 ADS1115_COMP_QUE_MASK

```
#define ADS1115_COMP_QUE_MASK 0x0003
```

Definition at line [62](#) of file [ads1115.c](#).

4.5.2.8 ADS1115_COMP_QUE_SHIFT

```
#define ADS1115_COMP_QUE_SHIFT 0
```

Definition at line [63](#) of file [ads1115.c](#).

4.5.2.9 ADS1115_DR_MASK

```
#define ADS1115_DR_MASK 0x00E0
```

Definition at line [46](#) of file [ads1115.c](#).

4.5.2.10 ADS1115_DR_SHIFT

```
#define ADS1115_DR_SHIFT 5
```

Definition at line [47](#) of file [ads1115.c](#).

4.5.2.11 ADS1115_MAX_VALUE

```
#define ADS1115_MAX_VALUE 32767
```

Definition at line [69](#) of file [ads1115.c](#).

4.5.2.12 ADS1115_MIN_VALUE

```
#define ADS1115_MIN_VALUE -32768
```

Definition at line [70](#) of file [ads1115.c](#).

4.5.2.13 ADS1115_MODE_MASK

```
#define ADS1115_MODE_MASK 0x0100
```

Definition at line [42](#) of file [ads1115.c](#).

4.5.2.14 ADS1115_MODE_SHIFT

```
#define ADS1115_MODE_SHIFT 8
```

Definition at line [43](#) of file [ads1115.c](#).

4.5.2.15 ADS1115_MUX_MASK

```
#define ADS1115_MUX_MASK 0x7000
```

Definition at line [34](#) of file [ads1115.c](#).

4.5.2.16 ADS1115_MUX_SHIFT

```
#define ADS1115_MUX_SHIFT 12
```

Definition at line [35](#) of file [ads1115.c](#).

4.5.2.17 ADS1115_OS_IDLE

```
#define ADS1115_OS_IDLE 0x8000
```

Definition at line [30](#) of file [ads1115.c](#).

4.5.2.18 ADS1115_OS_MASK

```
#define ADS1115_OS_MASK 0x8000
```

Definition at line [28](#) of file [ads1115.c](#).

4.5.2.19 ADS1115_OS_SHIFT

```
#define ADS1115_OS_SHIFT 15
```

Definition at line [29](#) of file [ads1115.c](#).

4.5.2.20 ADS1115_OS_START_SINGLE

```
#define ADS1115_OS_START_SINGLE 0x8000
```

Definition at line [31](#) of file [ads1115.c](#).

4.5.2.21 ADS1115_PGA_MASK

```
#define ADS1115_PGA_MASK 0x0E00
```

Definition at line 38 of file [ads1115.c](#).

4.5.2.22 ADS1115_PGA_SHIFT

```
#define ADS1115_PGA_SHIFT 9
```

Definition at line 39 of file [ads1115.c](#).

4.5.2.23 ADS1115_REG_CONFIG

```
#define ADS1115_REG_CONFIG 0x01
```

Chip configuration register (read/write)

Definition at line 19 of file [ads1115.c](#).

4.5.2.24 ADS1115_REG_CONVERSION

```
#define ADS1115_REG_CONVERSION 0x00
```

ADS1115 Register addresses.

ADC result register (read-only)

Definition at line 18 of file [ads1115.c](#).

4.5.2.25 ADS1115_REG_HI_THRESH

```
#define ADS1115_REG_HI_THRESH 0x03
```

Interrupt high threshold register (read/write)

Definition at line 21 of file [ads1115.c](#).

4.5.2.26 ADS1115_REG_LO_THRESH

```
#define ADS1115_REG_LO_THRESH 0x02
```

Interrupt low threshold register (read/write)

Definition at line 20 of file [ads1115.c](#).

4.5.3 Function Documentation

4.5.3.1 ads1115_continuous_conversion_read()

```
ads1115_error_t ads1115_continuous_conversion_read (
    ads1115_handle_t * handle,
    int16_t * adc_raw,
    float * voltage)
```

Read the result of adc continuously.

Parameters

<i>handle</i>	Pointer to device handle
<i>adc_raw</i>	Pointer to raw adc result
<i>voltage</i>	Pointer to converted adc result

Returns`ads1115_error_t` Error codeDefinition at line 322 of file [ads1115.c](#).**4.5.3.2 ads1115_continuous_conversion_start()**

```
ads1115_error_t ads1115_continuous_conversion_start (
    ads1115_handle_t * handle)
```

Start adc continuous conversion mode.

Parameters

<i>handle</i>	Pointer to device handle
---------------	--------------------------

Returns`ads1115_error_t` Error codeDefinition at line 300 of file [ads1115.c](#).**4.5.3.3 ads1115_continuous_conversion_stop()**

```
ads1115_error_t ads1115_continuous_conversion_stop (
    ads1115_handle_t * handle)
```

Stop adc continuous conversion mode.

Parameters

<i>handle</i>	Pointer to device handle
---------------	--------------------------

Returns`ads1115_error_t` Error codeDefinition at line 311 of file [ads1115.c](#).**4.5.3.4 ads1115_deinit()**

```
ads1115_error_t ads1115_deinit (
    ads1115_handle_t * handle)
```

De-initialize ADS1115 device.

Parameters

<i>handle</i>	Pointer to device handle
---------------	--------------------------

Returns

[ads1115_error_t](#) Error code

Definition at line 211 of file [ads1115.c](#).

4.5.3.5 ads1115_get_channel()

```
ads1115_error_t ads1115_get_channel (
    ads1115_handle_t * handle,
    ads1115_mux_t * channel)
```

Set adc input channel from mux.

Parameters

<i>handle</i>	Pointer to device handle
<i>channel</i>	Pointer to input adc channel from mux

Returns

[ads1115_error_t](#) Error code

Definition at line 409 of file [ads1115.c](#).

4.5.3.6 ads1115_get_compare_alert()

```
ads1115_error_t ads1115_get_compare_alert (
    ads1115_handle_t * handle,
    ads1115_comp_polarity_t * polarity)
```

Get the alert pin polatiry (ACTIVE_HIGH or ACTIVE_LOW)

Parameters

<i>handle</i>	Pointer to device handle
<i>polarity</i>	Pointer to alert pin polarity

Returns

[ads1115_error_t](#) Error code

Definition at line 565 of file [ads1115.c](#).

4.5.3.7 ads1115_get_compare_latch()

```
ads1115_error_t ads1115_get_compare_latch (
    ads1115_handle_t * handle,
    ads1115_comp_latch_t * latch)
```

Get comparator latching mode.

Parameters

<i>handle</i>	Pointer to device handle
<i>latch</i>	Pointer to comparator latch

Returns

`ads1115_error_t` Error code

Definition at line 526 of file [ads1115.c](#).

4.5.3.8 ads1115_get_compare_mode()

```
ads1115_error_t ads1115_get_compare_mode (
    ads1115_handle_t * handle,
    ads1115_comp_mode_t * compare)
```

Get current comparator mode from device.

Parameters

<i>handle</i>	Pointer to device handle
<i>compare</i>	Pointer to interrupt compare mode

Returns

`ads1115_error_t` Error code

Definition at line 448 of file [ads1115.c](#).

4.5.3.9 ads1115_get_compare_queue()

```
ads1115_error_t ads1115_get_compare_queue (
    ads1115_handle_t * handle,
    ads1115_comp_queue_t * comp_queue)
```

Get comparatore queue.

Parameters

<i>handle</i>	Pointer to device handle
<i>comp_queue</i>	Pointer to comparator queue

Returns

`ads1115_error_t` Error code

Definition at line 487 of file [ads1115.c](#).

4.5.3.10 ads1115_get_compare_threshold()

```
ads1115_error_t ads1115_get_compare_threshold (
    ads1115_handle_t * handle,
    int16_t * low_threshold,
    int16_t * high_threshold)
```

Get the comparator low and high threshold.

Parameters

<i>handle</i>	Pointer to device handle
<i>low_threshold</i>	Pointer to compare low threshold
<i>high_threshold</i>	Pointer to compare high threshold

Returns

`ads1115_error_t` Error code

Definition at line 607 of file [ads1115.c](#).

4.5.3.11 ads1115_get_data_rate()

```
ads1115_error_t ads1115_get_data_rate (
    ads1115_handle_t * handle,
    ads1115_data_rate_t * data_rate)
```

Get adc data sample rate.

Parameters

<i>handle</i>	Pointer to device handle
<i>data_rate</i>	Data rate setting

Returns

`ads1115_error_t` Error code

Definition at line 277 of file [ads1115.c](#).

4.5.3.12 ads1115_get_range()

```
ads1115_error_t ads1115_get_range (
    ads1115_handle_t * handle,
    ads1115_range_t * range)
```

Get range of adc.

Parameters

<i>handle</i>	Pointer to device handle
<i>range</i>	Maximum voltage range

Returns

`ads1115_error_t` Error code

Definition at line 238 of file [ads1115.c](#).

4.5.3.13 ads1115_init()

```
ads1115_error_t ads1115_init (
    ads1115_handle_t * handle)
```

Initialize ADS1115 device.

Parameters

<i>handle</i>	Pointer to device handle
---------------	--------------------------

Returns

`ads1115_error_t` Error code

Definition at line 175 of file [ads1115.c](#).

4.5.3.14 ads1115_is_ready()

```
ads1115_error_t ads1115_is_ready (
    ads1115_handle_t * handle,
    bool * flag)
```

Check if conversion is ready.

Parameters

<i>handle</i>	Pointer to device handle
<i>flag</i>	Pointer to store flag status

Returns

`ads1115_error_t` Error code

Definition at line 639 of file [ads1115.c](#).

4.5.3.15 ads1115_set_channel()

```
ads1115_error_t ads1115_set_channel (
    ads1115_handle_t * handle,
    ads1115_mux_t channel)
```

Set adc input channel from mux.

Parameters

<i>handle</i>	Pointer to device handle
<i>channel</i>	Input adc channel from mux

Returns

`ads1115_error_t` Error code

Definition at line 393 of file [ads1115.c](#).

4.5.3.16 ads1115_set_compare_alert()

```
ads1115_error_t ads1115_set_compare_alert (
    ads1115_handle_t * handle,
    ads1115_comp_polarity_t polarity)
```

Set the alert pin polatiry mode (ACTIVE_HIGH or ACTIVE_LOW)

Parameters

<i>handle</i>	Pointer to device handle
<i>polarity</i>	Alert pin polarity

Returns

`ads1115_error_t` Error code

Definition at line 549 of file `ads1115.c`.

4.5.3.17 ads1115_set_compare_latch()

```
ads1115_error_t ads1115_set_compare_latch (
    ads1115_handle_t * handle,
    ads1115_comp_latch_t latch)
```

Set comparator latching mode.

Parameters

<i>handle</i>	Pointer to device handle
<i>latch</i>	Comparator latch

Returns

`ads1115_error_t` Error code

Definition at line 510 of file `ads1115.c`.

4.5.3.18 ads1115_set_compare_mode()

```
ads1115_error_t ads1115_set_compare_mode (
    ads1115_handle_t * handle,
    ads1115_comp_mode_t compare)
```

Set comparator mode.

Parameters

<i>handle</i>	Pointer to device handle
<i>compare</i>	Interrupt compare mode

Returns

`ads1115_error_t` Error code

Definition at line 432 of file `ads1115.c`.

4.5.3.19 ads1115_set_compare_queue()

```
ads1115_error_t ads1115_set_compare_queue (
    ads1115_handle_t * handle,
    ads1115_comp_queue_t comp_queue)
```

Set comparatore queue.

Parameters

<i>handle</i>	Pointer to device handle
<i>comp_queue</i>	Comparator queue

Returns`ads1115_error_t` Error codeDefinition at line 471 of file [ads1115.c](#).**4.5.3.20 ads1115_set_compare_threshold()**

```
ads1115_error_t ads1115_set_compare_threshold (
    ads1115_handle_t * handle,
    int16_t low_threshold,
    int16_t high_threshold)
```

Set the comparator low and high threshold.

Parameters

<i>handle</i>	Pointer to device handle
<i>low_threshold</i>	Compare low threshold
<i>high_threshold</i>	Compare high threshold

Returns`ads1115_error_t` Error codeDefinition at line 588 of file [ads1115.c](#).**4.5.3.21 ads1115_set_data_rate()**

```
ads1115_error_t ads1115_set_data_rate (
    ads1115_handle_t * handle,
    ads1115_data_rate_t data_rate)
```

Set adc data sample rate.

Parameters

<i>handle</i>	Pointer to device handle
<i>data_rate</i>	Data rate setting

Returns`ads1115_error_t` Error codeDefinition at line 261 of file [ads1115.c](#).**4.5.3.22 ads1115_set_range()**

```
ads1115_error_t ads1115_set_range (
    ads1115_handle_t * handle,
    ads1115_range_t range)
```

Set range of adc.

Parameters

<i>handle</i>	Pointer to device handle
<i>range</i>	Maximum voltage range

Returns`ads1115_error_t` Error codeDefinition at line 222 of file `ads1115.c`.**4.5.3.23 `ads1115_single_read()`**

```
ads1115_error_t ads1115_single_read (
    ads1115_handle_t * handle,
    int16_t * adc_raw,
    float * voltage)
```

Read adc result once.

Parameters

<i>handle</i>	Pointer to device handle
<i>adc_raw</i>	Pointer to raw adc result
<i>voltage</i>	Pointer to converted adc result

Returns`ads1115_error_t` Error codeDefinition at line 347 of file `ads1115.c`.**4.6 ads1115.c**

Go to the documentation of this file.

```
00001
00007
00008 #include "ads1115.h"
00009 #include <stddef.h>
00010
00011 /*=====
00012 /* PRIVATE REGISTER DEFINITION
00013 /*=====
00014
00018 #define ADS1115_REG_CONVERSION 0x00
00019 #define ADS1115_REG_CONFIG 0x01
00020 #define ADS1115_REG_LO_THRESH 0x02
00021 #define ADS1115_REG_HI_THRESH 0x03
00022
00023 /*=====
00024 /* PRIVATE BIT MASKS AND SHIFTS
00025 /*=====
00026
00027 /* Operational Status (Bit 15) */
00028 #define ADS1115_OS_MASK 0x8000
00029 #define ADS1115_OS_SHIFT 15
00030 #define ADS1115_OS_IDLE 0x8000
```

```

00031 #define ADS1115_OS_START_SINGLE 0x8000
00032
00033 /* Input Multiplexer (Bits 14:12) */
00034 #define ADS1115_MUX_MASK 0x7000
00035 #define ADS1115_MUX_SHIFT 12
00036
00037 /* Programmable Gain (Bits 11:9) */
00038 #define ADS1115_PGA_MASK 0x0E00
00039 #define ADS1115_PGA_SHIFT 9
00040
00041 /* Operating Mode (Bit 8) */
00042 #define ADS1115_MODE_MASK 0x0100
00043 #define ADS1115_MODE_SHIFT 8
00044
00045 /* Data Rate (Bits 7:5) */
00046 #define ADS1115_DR_MASK 0x00E0
00047 #define ADS1115_DR_SHIFT 5
00048
00049 /* Comparator Mode (Bit 4) */
00050 #define ADS1115_COMP_MODE_MASK 0x0010
00051 #define ADS1115_COMP_MODE_SHIFT 4
00052
00053 /* Comparator Polarity (Bit 3) */
00054 #define ADS1115_COMP_POL_MASK 0x0008
00055 #define ADS1115_COMP_POL_SHIFT 3
00056
00057 /* Latching Comparator (Bit 2) */
00058 #define ADS1115_COMP_LAT_MASK 0x0004
00059 #define ADS1115_COMP_LAT_SHIFT 2
00060
00061 /* Comparator Queue (Bits 1:0) */
00062 #define ADS1115_COMP_QUE_MASK 0x0003
00063 #define ADS1115_COMP_QUE_SHIFT 0
00064
00065 /***** PRIVATE CONSTANTS *****/
00066 /* PRIVATE FUNCTIONS */
00067 /***** PRIVATE FUNCTIONS *****/
00068
00069 #define ADS1115_MAX_VALUE 32767
00070 #define ADS1115_MIN_VALUE -32768
00071
00072 /* Full Scale Range values in millivolts */
00073 static const uint16_t ADS1115_FSR_VALUES[] = {
00074     6144, /* ADS1115_RANGE_6_144V */
00075     4096, /* ADS1115_RANGE_4_096V */
00076     2048, /* ADS1115_RANGE_2_048V */
00077     1024, /* ADS1115_RANGE_1_024V */
00078     512, /* ADS1115_RANGE_0_512V */
00079     256 /* ADS1115_RANGE_0_256V */
00080 };
00081
00082 /* Conversion time in microseconds */
00083 static const uint32_t ADS1115_CONV_TIME_US[] = {
00084     125000, /* ADS1115_DR_8_SPS */
00085     62500, /* ADS1115_DR_16_SPS */
00086     31250, /* ADS1115_DR_32_SPS */
00087     15625, /* ADS1115_DR_64_SPS */
00088     7813, /* ADS1115_DR_128_SPS */
00089     4000, /* ADS1115_DR_250_SPS */
00090     2106, /* ADS1115_DR_475_SPS */
00091     1163 /* ADS1115_DR_860_SPS */
00092 };
00093
00094 /***** PRIVATE FUNCTIONS *****/
00095 /* PRIVATE FUNCTIONS */
00096 /***** PRIVATE FUNCTIONS *****/
00097
00098 static ads1115_error_t write_register(ads1115_handle_t *handle, uint8_t reg_addr, uint16_t value)
00099 {
00100     uint8_t data[2];
00101
00102     /* ADS1115 uses big-endian (MSB first) */
00103     data[0] = (uint8_t)((value >> 8) & 0xFF);
00104     data[1] = (uint8_t)(value & 0xFF);
00105
00106     if (!handle->i2c_write((uint8_t)handle->i2c_addr, reg_addr, data, 2))
00107     {
00108         return ADS1115_ERROR_I2C_WRITE;
00109     }
00110
00111     return ADS1115_OK;
00112 }
00113
00114 static ads1115_error_t read_register(ads1115_handle_t *handle, uint8_t reg_addr, uint16_t *value)
00115 {
00116     uint8_t data[2];
00117

```

```

00118     if (!handle->i2c_read((uint8_t)handle->i2c_addr, reg_addr, data, 2))
00119     {
00120         return ADS1115_ERROR_I2C_READ;
00121     }
00122
00123     /* ADS1115 uses big-endian (MSB first) */
00124     *value = ((uint16_t)data[0] << 8) | data[1];
00125
00126     return ADS1115_OK;
00127 }
00128
00129 static uint16_t build_config_register(const ads1115_config_t *config)
00130 {
00131     uint16_t reg_value = 0;
00132
00133     reg_value |= ((uint16_t)config->mux << ADS1115_MUX_SHIFT) & ADS1115_MUX_MASK;
00134     reg_value |= ((uint16_t)config->range << ADS1115_PGA_SHIFT) & ADS1115_PGA_MASK;
00135     reg_value |= ((uint16_t)config->mode << ADS1115_MODE_SHIFT) & ADS1115_MODE_MASK;
00136     reg_value |= ((uint16_t)config->data_rate << ADS1115_DR_SHIFT) & ADS1115_DR_MASK;
00137     reg_value |= ((uint16_t)config->comp_mode << ADS1115_COMP_MODE_SHIFT) & ADS1115_COMP_MODE_MASK;
00138     reg_value |= ((uint16_t)config->comp_pol << ADS1115_COMP_POL_SHIFT) & ADS1115_COMP_POL_MASK;
00139     reg_value |= ((uint16_t)config->comp_latch << ADS1115_COMP_LAT_SHIFT) & ADS1115_COMP_LAT_MASK;
00140     reg_value |= ((uint16_t)config->comp_queue << ADS1115_COMP_QUE_SHIFT) & ADS1115_COMP_QUE_MASK;
00141
00142     return reg_value;
00143 }
00144
00145 static ads1115_error_t update_config_register(ads1115_handle_t *handle)
00146 {
00147     uint16_t config_reg = build_config_register(&handle->config);
00148     return write_register(handle, ADS1115_REG_CONFIG, config_reg);
00149 }
00150
00151 static uint32_t get_conversion_time_us(ads1115_data_rate_t data_rate)
00152 {
00153     if (data_rate > ADS1115_DR_860_SPS)
00154     {
00155         return ADS1115_CONV_TIME_US[ADS1115_DR_128_SPS];
00156     }
00157     return ADS1115_CONV_TIME_US[data_rate];
00158 }
00159
00160 static float raw_to_voltage(ads1115_range_t range, int16_t raw_value)
00161 {
00162     if (range > ADS1115_RANGE_0V256)
00163     {
00164         range = ADS1115_RANGE_2V048;
00165     }
00166
00167     float fsr = (float)ADS1115_FSR_VALUES[range];
00168     return (raw_value * fsr) / 32768.0f;
00169 }
00170
00171 /*=====
00172  * PUBLIC API IMPLEMENTATIONS
00173 =====*/
00174
00175 ads1115_error_t ads1115_init(ads1115_handle_t *handle)
00176 {
00177     if (handle == NULL)
00178     {
00179         return ADS1115_ERROR_NULL_POINTER;
00180     }
00181
00182     if ((handle->i2c_read == NULL) || (handle->i2c_write == NULL) || (handle->delay_ms == NULL))
00183     {
00184         return ADS1115_ERROR_INVALID_PARAM;
00185     }
00186
00187     /* Write default configuration */
00188     ads1115_error_t err = update_config_register(handle);
00189     if (err != ADS1115_OK)
00190     {
00191         return err;
00192     }
00193
00194     /* Write default thresholds */
00195     err = write_register(handle, ADS1115_REG_LO_THRESH, (uint16_t)handle->config.low_threshold);
00196     if (err != ADS1115_OK)
00197     {
00198         return err;
00199     }
00200
00201     err = write_register(handle, ADS1115_REG_HI_THRESH, (uint16_t)handle->config.high_threshold);
00202     if (err != ADS1115_OK)
00203     {
00204         return err;
00205     }

```

```
00205     }
00206
00207     handle->is_initialized = true;
00208     return ADS1115_OK;
00209 }
00210
00211 ads1115_error_t ads1115_deinit(ads1115_handle_t *handle)
00212 {
00213     if (handle == NULL)
00214     {
00215         return ADS1115_ERROR_NULL_POINTER;
00216     }
00217
00218     handle->is_initialized = false;
00219     return ADS1115_OK;
00220 }
00221
00222 ads1115_error_t ads1115_set_range(ads1115_handle_t *handle, ads1115_range_t range)
00223 {
00224     if (!handle->is_initialized)
00225     {
00226         return ADS1115_ERROR_NOT_INITIALIZED;
00227     }
00228
00229     if (range > ADS1115_RANGE_0V256)
00230     {
00231         return ADS1115_ERROR_INVALID_PARAM;
00232     }
00233
00234     handle->config.range = range;
00235     return update_config_register(handle);
00236 }
00237
00238 ads1115_error_t ads1115_get_range(ads1115_handle_t *handle, ads1115_range_t *range)
00239 {
00240     if (!handle->is_initialized)
00241     {
00242         return ADS1115_ERROR_NOT_INITIALIZED;
00243     }
00244
00245     if (range == NULL)
00246     {
00247         return ADS1115_ERROR_NULL_POINTER;
00248     }
00249
00250     uint16_t config_reg;
00251     ads1115_error_t err = read_register(handle, ADS1115_REG_CONFIG, &config_reg);
00252     if (err != ADS1115_OK)
00253     {
00254         return err;
00255     }
00256
00257     *range = (ads1115_range_t)((config_reg & ADS1115_PGA_MASK) >> ADS1115_PGA_SHIFT);
00258     return ADS1115_OK;
00259 }
00260
00261 ads1115_error_t ads1115_set_data_rate(ads1115_handle_t *handle, ads1115_data_rate_t data_rate)
00262 {
00263     if (!handle->is_initialized)
00264     {
00265         return ADS1115_ERROR_NOT_INITIALIZED;
00266     }
00267
00268     if (data_rate > ADS1115_DR_860_SPS)
00269     {
00270         return ADS1115_ERROR_INVALID_PARAM;
00271     }
00272
00273     handle->config.data_rate = data_rate;
00274     return update_config_register(handle);
00275 }
00276
00277 ads1115_error_t ads1115_get_data_rate(ads1115_handle_t *handle, ads1115_data_rate_t *data_rate)
00278 {
00279     if (!handle->is_initialized)
00280     {
00281         return ADS1115_ERROR_NOT_INITIALIZED;
00282     }
00283
00284     if (data_rate == NULL)
00285     {
00286         return ADS1115_ERROR_NULL_POINTER;
00287     }
00288
00289     uint16_t config_reg;
00290     ads1115_error_t err = read_register(handle, ADS1115_REG_CONFIG, &config_reg);
00291     if (err != ADS1115_OK)
```

```

00292     {
00293         return err;
00294     }
00295
00296     *data_rate = (ads1115_data_rate_t)((config_reg & ADS1115_DR_MASK) » ADS1115_DR_SHIFT);
00297     return ADS1115_OK;
00298 }
00299
00300 ads1115_error_t ads1115_continuous_conversion_start(ads1115_handle_t *handle)
00301 {
00302     if (!handle->is_initialized)
00303     {
00304         return ADS1115_ERROR_NOT_INITIALIZED;
00305     }
00306
00307     handle->config.mode = ADS1115_MODE_CONTINUOUS;
00308     return update_config_register(handle);
00309 }
00310
00311 ads1115_error_t ads1115_continuous_conversion_stop(ads1115_handle_t *handle)
00312 {
00313     if (!handle->is_initialized)
00314     {
00315         return ADS1115_ERROR_NOT_INITIALIZED;
00316     }
00317
00318     handle->config.mode = ADS1115_MODE_SINGLE_SHOT;
00319     return update_config_register(handle);
00320 }
00321
00322 ads1115_error_t ads1115_continuous_conversion_read(ads1115_handle_t *handle, int16_t *adc_raw, float
*voltage)
00323 {
00324     if (!handle->is_initialized)
00325     {
00326         return ADS1115_ERROR_NOT_INITIALIZED;
00327     }
00328
00329     if ((adc_raw == NULL) || (voltage == NULL))
00330     {
00331         return ADS1115_ERROR_NULL_POINTER;
00332     }
00333
00334     uint16_t raw_data;
00335     ads1115_error_t err = read_register(handle, ADS1115_REG_CONVERSION, &raw_data);
00336     if (err != ADS1115_OK)
00337     {
00338         return err;
00339     }
00340
00341     *adc_raw = (int16_t)raw_data;
00342     *voltage = raw_to_voltage(handle->config.range, *adc_raw);
00343
00344     return ADS1115_OK;
00345 }
00346
00347 ads1115_error_t ads1115_single_read(ads1115_handle_t *handle, int16_t *adc_raw, float *voltage)
00348 {
00349     if (!handle->is_initialized)
00350     {
00351         return ADS1115_ERROR_NOT_INITIALIZED;
00352     }
00353
00354     if ((adc_raw == NULL) || (voltage == NULL))
00355     {
00356         return ADS1115_ERROR_NULL_POINTER;
00357     }
00358
00359     /* Read current config */
00360     uint16_t config_reg;
00361     ads1115_error_t err = read_register(handle, ADS1115_REG_CONFIG, &config_reg);
00362     if (err != ADS1115_OK)
00363     {
00364         return err;
00365     }
00366
00367     /* Start single conversion */
00368     config_reg |= ADS1115_OS_START_SINGLE;
00369     err = write_register(handle, ADS1115_REG_CONFIG, config_reg);
00370     if (err != ADS1115_OK)
00371     {
00372         return err;
00373     }
00374
00375     /* Wait for conversion */
00376     uint32_t conv_time_us = get_conversion_time_us(handle->config.data_rate);
00377     handle->delay_ms((conv_time_us / 1000) + 1);

```

```
00378     /* Read conversion result */
00379     uint16_t raw_data;
00380     err = read_register(handle, ADS1115_REG_CONVERSION, &raw_data);
00381     if (err != ADS1115_OK)
00382     {
00383         return err;
00384     }
00385
00386     *adc_raw = (int16_t)raw_data;
00387     *voltage = raw_to_voltage(handle->config.range, *adc_raw);
00388
00389     return ADS1115_OK;
00390 }
00391
00392
00393 ads1115_error_t ads1115_set_channel(ads1115_handle_t *handle, ads1115_mux_t channel)
00394 {
00395     if (!handle->is_initialized)
00396     {
00397         return ADS1115_ERROR_NOT_INITIALIZED;
00398     }
00399
00400     if (channel > ADS1115_MUX_AIN3_GND)
00401     {
00402         return ADS1115_ERROR_INVALID_PARAM;
00403     }
00404
00405     handle->config.mux = channel;
00406     return update_config_register(handle);
00407 }
00408
00409 ads1115_error_t ads1115_get_channel(ads1115_handle_t *handle, ads1115_mux_t *channel)
00410 {
00411     if (!handle->is_initialized)
00412     {
00413         return ADS1115_ERROR_NOT_INITIALIZED;
00414     }
00415
00416     if (channel == NULL)
00417     {
00418         return ADS1115_ERROR_NULL_POINTER;
00419     }
00420
00421     uint16_t config_reg;
00422     ads1115_error_t err = read_register(handle, ADS1115_REG_CONFIG, &config_reg);
00423     if (err != ADS1115_OK)
00424     {
00425         return err;
00426     }
00427
00428     *channel = (ads1115_mux_t)((config_reg & ADS1115_MUX_MASK) >> ADS1115_MUX_SHIFT);
00429     return ADS1115_OK;
00430 }
00431
00432 ads1115_error_t ads1115_set_compare_mode(ads1115_handle_t *handle, ads1115_comp_mode_t compare)
00433 {
00434     if (!handle->is_initialized)
00435     {
00436         return ADS1115_ERROR_NOT_INITIALIZED;
00437     }
00438
00439     if (compare > ADS1115_COMP_MODE_WINDOW)
00440     {
00441         return ADS1115_ERROR_INVALID_PARAM;
00442     }
00443
00444     handle->config.comp_mode = compare;
00445     return update_config_register(handle);
00446 }
00447
00448 ads1115_error_t ads1115_get_compare_mode(ads1115_handle_t *handle, ads1115_comp_mode_t *compare)
00449 {
00450     if (!handle->is_initialized)
00451     {
00452         return ADS1115_ERROR_NOT_INITIALIZED;
00453     }
00454
00455     if (compare == NULL)
00456     {
00457         return ADS1115_ERROR_NULL_POINTER;
00458     }
00459
00460     uint16_t config_reg;
00461     ads1115_error_t err = read_register(handle, ADS1115_REG_CONFIG, &config_reg);
00462     if (err != ADS1115_OK)
00463     {
00464         return err;
```

```

00465     }
00466
00467     *compare = (ads1115_comp_mode_t)((config_reg & ADS1115_COMP_MODE_MASK) >> ADS1115_COMP_MODE_SHIFT);
00468     return ADS1115_OK;
00469 }
00470
00471 ads1115_error_t ads1115_set_compare_queue(ads1115_handle_t *handle, ads1115_comp_queue_t comp_queue)
00472 {
00473     if (!handle->is_initialized)
00474     {
00475         return ADS1115_ERROR_NOT_INITIALIZED;
00476     }
00477
00478     if (comp_queue > ADS1115_COMP_QUE_DISABLE)
00479     {
00480         return ADS1115_ERROR_INVALID_PARAM;
00481     }
00482
00483     handle->config.comp_queue = comp_queue;
00484     return update_config_register(handle);
00485 }
00486
00487 ads1115_error_t ads1115_get_compare_queue(ads1115_handle_t *handle, ads1115_comp_queue_t *comp_queue)
00488 {
00489     if (!handle->is_initialized)
00490     {
00491         return ADS1115_ERROR_NOT_INITIALIZED;
00492     }
00493
00494     if (comp_queue == NULL)
00495     {
00496         return ADS1115_ERROR_NULL_POINTER;
00497     }
00498
00499     uint16_t config_reg;
00500     ads1115_error_t err = read_register(handle, ADS1115_REG_CONFIG, &config_reg);
00501     if (err != ADS1115_OK)
00502     {
00503         return err;
00504     }
00505
00506     *comp_queue = (ads1115_comp_queue_t)((config_reg & ADS1115_COMP_QUE_MASK) >>
00507     ADS1115_COMP_QUE_SHIFT);
00508     return ADS1115_OK;
00509 }
00510
00511 ads1115_error_t ads1115_set_compare_latch(ads1115_handle_t *handle, ads1115_comp_latch_t latch)
00512 {
00513     if (!handle->is_initialized)
00514     {
00515         return ADS1115_ERROR_NOT_INITIALIZED;
00516     }
00517
00518     if (latch > ADS1115_COMP_LAT_LATCHING)
00519     {
00520         return ADS1115_ERROR_INVALID_PARAM;
00521     }
00522
00523     handle->config.comp_latch = latch;
00524     return update_config_register(handle);
00525 }
00526
00527 ads1115_error_t ads1115_get_compare_latch(ads1115_handle_t *handle, ads1115_comp_latch_t *latch)
00528 {
00529     if (!handle->is_initialized)
00530     {
00531         return ADS1115_ERROR_NOT_INITIALIZED;
00532     }
00533
00534     if (latch == NULL)
00535     {
00536         return ADS1115_ERROR_NULL_POINTER;
00537     }
00538
00539     uint16_t config_reg;
00540     ads1115_error_t err = read_register(handle, ADS1115_REG_CONFIG, &config_reg);
00541     if (err != ADS1115_OK)
00542     {
00543         return err;
00544     }
00545
00546     *latch = (ads1115_comp_latch_t)((config_reg & ADS1115_COMP_LAT_MASK) >> ADS1115_COMP_LAT_SHIFT);
00547     return ADS1115_OK;
00548 }
00549
00550 ads1115_error_t ads1115_set_compare_alert(ads1115_handle_t *handle, ads1115_comp_polarity_t polarity)

```

```
00551     if (!handle->is_initialized)
00552     {
00553         return ADS1115_ERROR_NOT_INITIALIZED;
00554     }
00555
00556     if (polarity > ADS1115_COMP_POL_ACTIVE_HIGH)
00557     {
00558         return ADS1115_ERROR_INVALID_PARAM;
00559     }
00560
00561     handle->config.comp_pol = polarity;
00562     return update_config_register(handle);
00563 }
00564
00565 ads1115_error_t ads1115_get_compare_alert(ads1115_handle_t *handle, ads1115_comp_polarity_t *polarity)
00566 {
00567     if (!handle->is_initialized)
00568     {
00569         return ADS1115_ERROR_NOT_INITIALIZED;
00570     }
00571
00572     if (polarity == NULL)
00573     {
00574         return ADS1115_ERROR_NULL_POINTER;
00575     }
00576
00577     uint16_t config_reg;
00578     ads1115_error_t err = read_register(handle, ADS1115_REG_CONFIG, &config_reg);
00579     if (err != ADS1115_OK)
00580     {
00581         return err;
00582     }
00583
00584     *polarity = (ads1115_comp_polarity_t)((config_reg & ADS1115_COMP_POL_MASK) >>
00585     ADS1115_COMP_POL_SHIFT);
00586     return ADS1115_OK;
00587 }
00588
00589 ads1115_error_t ads1115_set_compare_threshold(ads1115_handle_t *handle, int16_t low_threshold, int16_t
00590 high_threshold)
00591 {
00592     if (!handle->is_initialized)
00593     {
00594         return ADS1115_ERROR_NOT_INITIALIZED;
00595     }
00596
00597     handle->config.low_threshold = low_threshold;
00598     handle->config.high_threshold = high_threshold;
00599
00600     ads1115_error_t err = write_register(handle, ADS1115_REG_LO_THRESH, (uint16_t)low_threshold);
00601     if (err != ADS1115_OK)
00602     {
00603         return err;
00604     }
00605     return write_register(handle, ADS1115_REG_HI_THRESH, (uint16_t)high_threshold);
00606 }
00607
00608 ads1115_error_t ads1115_get_compare_threshold(ads1115_handle_t *handle, int16_t *low_threshold,
00609 int16_t *high_threshold)
00610 {
00611     if (!handle->is_initialized)
00612     {
00613         return ADS1115_ERROR_NOT_INITIALIZED;
00614     }
00615
00616     if ((low_threshold == NULL) || (high_threshold == NULL))
00617     {
00618         return ADS1115_ERROR_NULL_POINTER;
00619     }
00620
00621     uint16_t low_reg, high_reg;
00622
00623     ads1115_error_t err = read_register(handle, ADS1115_REG_LO_THRESH, &low_reg);
00624     if (err != ADS1115_OK)
00625     {
00626         return err;
00627     }
00628
00629     err = read_register(handle, ADS1115_REG_HI_THRESH, &high_reg);
00630     if (err != ADS1115_OK)
00631     {
00632         return err;
00633     }
00634
00635     *low_threshold = (int16_t)low_reg;
00636     *high_threshold = (int16_t)high_reg;
```

```

00635
00636     return ADS1115_OK;
00637 }
00638
00639 ads1115_error_t ads1115_is_ready(ads1115_handle_t *handle, bool *flag)
00640 {
00641     if (!handle->is_initialized)
00642     {
00643         return ADS1115_ERROR_NOT_INITIALIZED;
00644     }
00645
00646     if (flag == NULL)
00647     {
00648         return ADS1115_ERROR_NULL_POINTER;
00649     }
00650
00651     uint16_t config_reg;
00652     ads1115_error_t err = read_register(handle, ADS1115_REG_CONFIG, &config_reg);
00653     if (err != ADS1115_OK)
00654     {
00655         return err;
00656     }
00657
00658     *flag = (config_reg & ADS1115_OS_MASK) ? true : false;
00659     return ADS1115_OK;
00660 }
```

4.7 src/ads1115.h File Reference

ADS1115 16-Bit ADC Driver - Platform Independent.

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

Classes

- struct `ads1115_config_t`
ADS1115 configuration structure.
- struct `ads1115_handle_t`
ADS1115 device handle structure.

Macros

- `#define ADS1115_DEFAULT_CONFIGURATION`
Default configuration values at compile time.

TypeDefs

- `typedef bool(* ads1115_i2c_write_t)` (`uint8_t device_addr, uint8_t reg_addr, const uint8_t *data, uint8_t length`)
I2C write callback function type.
- `typedef bool(* ads1115_i2c_read_t)` (`uint8_t device_addr, uint8_t reg_addr, uint8_t *data, uint8_t length`)
I2C read callback function type.
- `typedef void(* ads1115_delay_ms_t)` (`uint32_t milliseconds`)
Delay callback function type.

Enumerations

- enum `ads1115_error_t` {

`ADS1115_OK = 0, ADS1115_ERROR_INVALID_PARAM, ADS1115_ERROR_I2C_WRITE, ADS1115_ERROR_I2C_READ`

`,`

`ADS1115_ERROR_TIMEOUT, ADS1115_ERROR_NOT_INITIALIZED, ADS1115_ERROR_CONVERSION_BUSY`

`, ADS1115_ERROR_NULL_POINTER }`

Error codes for ADS1115 operations.
- enum `ads1115_i2c_addr_t` { `ADS1115_ADDR_GND = 0x48, ADS1115_ADDR_VDD = 0x49,`

`ADS1115_ADDR_SDA = 0x4A, ADS1115_ADDR_SCL = 0x4B` }

I2C address selection based on ADDR pin connection.
- enum `ads1115_mux_t` {

`ADS1115_MUX_AIN0_AIN1 = 0, ADS1115_MUX_AIN0_AIN3 = 1, ADS1115_MUX_AIN1_AIN3 = 2,`

`ADS1115_MUX_AIN2_AIN3 = 3,`

`ADS1115_MUX_AIN0_GND = 4, ADS1115_MUX_AIN1_GND = 5, ADS1115_MUX_AIN2_GND = 6,`

`ADS1115_MUX_AIN3_GND = 7` }

Input multiplexer configuration (channel selection)
- enum `ads1115_range_t` {

`ADS1115_RANGE_6V144 = 0, ADS1115_RANGE_4V096 = 1, ADS1115_RANGE_2V048 = 2,`

`ADS1115_RANGE_1V024 = 3,`

`ADS1115_RANGE_0V512 = 4, ADS1115_RANGE_0V256 = 5` }

Programmable gain amplifier configuration.
- enum `ads1115_mode_t` { `ADS1115_MODE_CONTINUOUS = 0, ADS1115_MODE_SINGLE_SHOT = 1` }

Device operating mode.
- enum `ads1115_data_rate_t` {

`ADS1115_DR_8_SPS = 0, ADS1115_DR_16_SPS = 1, ADS1115_DR_32_SPS = 2, ADS1115_DR_64_SPS`

`= 3,`

`ADS1115_DR_128_SPS = 4, ADS1115_DR_250_SPS = 5, ADS1115_DR_475_SPS = 6, ADS1115_DR_860_SPS`

`= 7` }

Data rate (samples per second)
- enum `ads1115_comp_mode_t` { `ADS1115_COMP_MODE_TRADITIONAL = 0, ADS1115_COMP_MODE_WINDOW`

`= 1` }

Comperator mode.
- enum `ads1115_comp_polarity_t` { `ADS1115_COMP_POL_ACTIVE_LOW = 0, ADS1115_COMP_POL_ACTIVE_HIGH`

`= 1` }

Comparator polarity.
- enum `ads1115_comp_latch_t` { `ADS1115_COMP_LAT_NON_LATCHING = 0, ADS1115_COMP_LAT_LATCHING`

`= 1` }

Latching comparator.
- enum `ads1115_comp_queue_t` { `ADS1115_COMP_QUE_1_CONV = 0, ADS1115_COMP_QUE_2_CONV`

`= 1, ADS1115_COMP_QUE_4_CONV = 2, ADS1115_COMP_QUE_DISABLE = 3` }

Comparator queue and disable.

Functions

- `ads1115_error_t ads1115_init (ads1115_handle_t *handle)`

Initialize ADS1115 device.
- `ads1115_error_t ads1115_deinit (ads1115_handle_t *handle)`

De-initialize ADS1115 device.
- `ads1115_error_t ads1115_set_range (ads1115_handle_t *handle, ads1115_range_t range)`

Set range of adc.
- `ads1115_error_t ads1115_get_range (ads1115_handle_t *handle, ads1115_range_t *range)`

Get range of adc.

- `ads1115_error_t ads1115_set_data_rate (ads1115_handle_t *handle, ads1115_data_rate_t data_rate)`
Set adc data sample rate.
- `ads1115_error_t ads1115_get_data_rate (ads1115_handle_t *handle, ads1115_data_rate_t *data_rate)`
Get adc data sample rate.
- `ads1115_error_t ads1115_continuous_conversion_start (ads1115_handle_t *handle)`
Start adc continuous conversion mode.
- `ads1115_error_t ads1115_continuous_conversion_stop (ads1115_handle_t *handle)`
Stop adc continuous conversion mode.
- `ads1115_error_t ads1115_continuous_conversion_read (ads1115_handle_t *handle, int16_t *adc_raw, float *voltage)`
Read the result of adc continuously.
- `ads1115_error_t ads1115_single_read (ads1115_handle_t *handle, int16_t *adc_raw, float *voltage)`
Read adc result once.
- `ads1115_error_t ads1115_set_channel (ads1115_handle_t *handle, ads1115_mux_t channel)`
Set adc input channel from mux.
- `ads1115_error_t ads1115_get_channel (ads1115_handle_t *handle, ads1115_mux_t *channel)`
Set adc input channel from mux.
- `ads1115_error_t ads1115_set_compare_mode (ads1115_handle_t *handle, ads1115_comp_mode_t compare)`
Set comparator mode.
- `ads1115_error_t ads1115_get_compare_mode (ads1115_handle_t *handle, ads1115_comp_mode_t *compare)`
Get current comparator mode from device.
- `ads1115_error_t ads1115_set_compare_queue (ads1115_handle_t *handle, ads1115_comp_queue_t comp_queue)`
Set comparatore queue.
- `ads1115_error_t ads1115_get_compare_queue (ads1115_handle_t *handle, ads1115_comp_queue_t *comp_queue)`
Get comparatore queue.
- `ads1115_error_t ads1115_set_compare_alert (ads1115_handle_t *handle, ads1115_comp_polarity_t polarity)`
Set the alert pin polatiry mode (ACTIVE_HIGH or ACTIVE_LOW)
- `ads1115_error_t ads1115_get_compare_alert (ads1115_handle_t *handle, ads1115_comp_polarity_t *polarity)`
Get the alert pin polatiry (ACTIVE_HIGH or ACTIVE_LOW)
- `ads1115_error_t ads1115_set_compare_threshold (ads1115_handle_t *handle, int16_t low_threshold, int16_t high_threshold)`
Set the comparator low and high threshold.
- `ads1115_error_t ads1115_get_compare_threshold (ads1115_handle_t *handle, int16_t *low_threshold, int16_t *high_threshold)`
Get the comparator low and high threshold.
- `ads1115_error_t ads1115_set_compare_latch (ads1115_handle_t *handle, ads1115_comp_latch_t latch)`
Set comparator latching mode.
- `ads1115_error_t ads1115_get_compare_latch (ads1115_handle_t *handle, ads1115_comp_latch_t *latch)`
Get comparator latching mode.
- `ads1115_error_t ads1115_is_ready (ads1115_handle_t *handle, bool *flag)`
Check if conversion is ready.

4.7.1 Detailed Description

ADS1115 16-Bit ADC Driver - Platform Independent.

Author

Şükrü Can Kılıç

Date

19.01.2026

Texas Instruments ADS1115 4-Channel 16-Bit ADC with I2C Interface Datasheet: <https://www.ti.com/lit/ds/symlink/ads1115.pdf>

Definition in file [ads1115.h](#).

4.7.2 Macro Definition Documentation

4.7.2.1 ADS1115_DEFAULT_CONFIGURATION

```
#define ADS1115_DEFAULT_CONFIGURATION
```

Value:

```
{           \
    .mux = ADS1115_MUX_AIN0_AIN1,          \
    .range = ADS1115_RANGE_2V048,          \
    .mode = ADS1115_MODE_SINGLE_SHOT,       \
    .data_rate = ADS1115_DR_128_SPS,        \
    .comp_mode = ADS1115_COMP_MODE_TRADITIONAL, \
    .comp_pol = ADS1115_COMP_POL_ACTIVE_LOW, \
    .comp_latch = ADS1115_COMP_LAT_NON_LATCHING, \
    .comp_queue = ADS1115_COMP_QUE_DISABLE,   \
    .low_threshold = 0x8000,                 \
    .high_threshold = 0x7FFF                \
}
```

Default configuration values at compile time.

These values match the ADS1115 power-up defaults as per datasheet. Users can modify these at runtime using setter functions.

Definition at line 207 of file [ads1115.h](#).

4.7.3 Typedef Documentation

4.7.3.1 ads1115_delay_ms_t

```
typedef void(* ads1115_delay_ms_t) (uint32_t milliseconds)
```

Delay callback function type.

Parameters

<i>milliseconds</i>	Delay duration in milliseconds
---------------------	--------------------------------

Definition at line 163 of file [ads1115.h](#).

4.7.3.2 ads1115_i2c_read_t

```
typedef bool(* ads1115_i2c_read_t) (uint8_t device_addr, uint8_t reg_addr, uint8_t *data,
uint8_t length)
```

I2C read callback function type.

Parameters

<i>device_addr</i>	I2C device address (7-bit)
<i>reg_addr</i>	Register address to read from
<i>data</i>	Pointer to buffer to store read data
<i>length</i>	Number of bytes to read

Returns

true if read successful, false otherwise

Definition at line 156 of file [ads1115.h](#).

4.7.3.3 ads1115_i2c_write_t

```
typedef bool(* ads1115_i2c_write_t) (uint8_t device_addr, uint8_t reg_addr, const uint8_t←
t *data, uint8_t length)
```

I2C write callback function type.

Parameters

<i>device_addr</i>	I2C device address (7-bit)
<i>reg_addr</i>	Register address to write to
<i>data</i>	Pointer to data buffer to write
<i>length</i>	Number of bytes to write

Returns

true if write successful, false otherwise

Definition at line 145 of file [ads1115.h](#).

4.7.4 Enumeration Type Documentation**4.7.4.1 ads1115_comp_latch_t**

```
enum ads1115_comp_latch_t
```

Latching comparator.

Enumerator

ADS1115_COMP_LAT_NON_LATCHING	Nonlatching comparator - Default
ADS1115_COMP_LAT_LATCHING	Latching comparator

Definition at line 117 of file [ads1115.h](#).

4.7.4.2 ads1115_comp_mode_t

```
enum ads1115_comp_mode_t
```

Comperator mode.

Enumerator

ADS1115_COMP_MODE_TRADITIONAL	Traditional comparator
ADS1115_COMP_MODE_WINDOW	Window comparator

Definition at line 101 of file [ads1115.h](#).

4.7.4.3 ads1115_comp_polarity_t

```
enum ads1115_comp_polarity_t
```

Comparator polarity.

Enumerator

ADS1115_COMP_POL_ACTIVE_LOW	Active low - Default
ADS1115_COMP_POL_ACTIVE_HIGH	Active high

Definition at line 109 of file [ads1115.h](#).

4.7.4.4 ads1115_comp_queue_t

```
enum ads1115_comp_queue_t
```

Comparator queue and disable.

Enumerator

ADS1115_COMP_QUE_1_CONV	Assert after one conversion
ADS1115_COMP_QUE_2_CONV	Assert after two conversions
ADS1115_COMP_QUE_4_CONV	Assert after four conversions
ADS1115_COMP_QUE_DISABLE	Disable comparator and set ALERT/RDY pin to high-impedance - Default

Definition at line 125 of file [ads1115.h](#).

4.7.4.5 ads1115_data_rate_t

```
enum ads1115_data_rate_t
```

Data rate (samples per second)

Enumerator

ADS1115_DR_8_SPS	8 samples per second
ADS1115_DR_16_SPS	16 samples per second
ADS1115_DR_32_SPS	32 samples per second
ADS1115_DR_64_SPS	64 samples per second
ADS1115_DR_128_SPS	128 samples per second - Default
ADS1115_DR_250_SPS	250 samples per second
ADS1115_DR_475_SPS	475 samples per second
ADS1115_DR_860_SPS	860 samples per second

Definition at line 87 of file [ads1115.h](#).

4.7.4.6 ads1115_error_t

```
enum ads1115_error_t
```

Error codes for ADS1115 operations.

Enumerator

ADS1115_OK	Operation successful
ADS1115_ERROR_INVALID_PARAM	Invalid parameter passed
ADS1115_ERROR_I2C_WRITE	I2C write operation failed
ADS1115_ERROR_I2C_READ	I2C read operation failed
ADS1115_ERROR_TIMEOUT	Operation timeout
ADS1115_ERROR_NOT_INITIALIZED	Device not initialized
ADS1115_ERROR_CONVERSION_BUSY	Conversion in progress
ADS1115_ERROR_NULL_POINTER	Null pointer passed

Definition at line 29 of file [ads1115.h](#).

4.7.4.7 ads1115_i2c_addr_t

```
enum ads1115_i2c_addr_t
```

I2C address selection based on ADDR pin connection.

Enumerator

ADS1115_ADDR_GND	ADDR pin connected to GND
ADS1115_ADDR_VDD	ADDR pin connected to VDD
ADS1115_ADDR_SDA	ADDR pin connected to SDA
ADS1115_ADDR_SCL	ADDR pin connected to SCL

Definition at line 43 of file [ads1115.h](#).

4.7.4.8 ads1115_mode_t

```
enum ads1115_mode_t
```

Device operating mode.

Enumerator

ADS1115_MODE_CONTINUOUS	Continuous conversion mode
ADS1115_MODE_SINGLE_SHOT	Single-shot conversion mode

Definition at line 79 of file [ads1115.h](#).

4.7.4.9 ads1115_mux_t

```
enum ads1115_mux_t
```

Input multiplexer configuration (channel selection)

Enumerator

ADS1115_MUX_AIN0_AIN1	Differential: AIN0 - AIN1
ADS1115_MUX_AIN0_AIN3	Differential: AIN0 - AIN3
ADS1115_MUX_AIN1_AIN3	Differential: AIN1 - AIN3
ADS1115_MUX_AIN2_AIN3	Differential: AIN2 - AIN3
ADS1115_MUX_AIN0_GND	Single-ended: AIN0
ADS1115_MUX_AIN1_GND	Single-ended: AIN1
ADS1115_MUX_AIN2_GND	Single-ended: AIN2
ADS1115_MUX_AIN3_GND	Single-ended: AIN3

Definition at line 53 of file [ads1115.h](#).

4.7.4.10 ads1115_range_t

```
enum ads1115_range_t
```

Programmable gain amplifier configuration.

Enumerator

ADS1115_RANGE_6V144	+/- 6.144V range (FSR = 6.144V)
ADS1115_RANGE_4V096	+/- 4.096V range (FSR = 4.096V)
ADS1115_RANGE_2V048	+/- 2.048V range (FSR = 2.048V) - Default
ADS1115_RANGE_1V024	+/- 1.024V range (FSR = 1.024V)
ADS1115_RANGE_0V512	+/- 0.512V range (FSR = 0.512V)
ADS1115_RANGE_0V256	+/- 0.256V range (FSR = 0.256V)

Definition at line 67 of file [ads1115.h](#).

4.7.5 Function Documentation

4.7.5.1 ads1115_continuous_conversion_read()

```
ads1115_error_t ads1115_continuous_conversion_read (
    ads1115_handle_t * handle,
    int16_t * adc_raw,
    float * voltage)
```

Read the result of adc continuously.

Parameters

<i>handle</i>	Pointer to device handle
<i>adc_raw</i>	Pointer to raw adc result
<i>voltage</i>	Pointer to converted adc result

Returns

`ads1115_error_t` Error code

Definition at line 322 of file [ads1115.c](#).

4.7.5.2 ads1115_continuous_conversion_start()

```
ads1115_error_t ads1115_continuous_conversion_start (
    ads1115_handle_t * handle)
```

Start adc continuous conversion mode.

Parameters

<i>handle</i>	Pointer to device handle
---------------	--------------------------

Returns

`ads1115_error_t` Error code

Definition at line 300 of file [ads1115.c](#).

4.7.5.3 ads1115_continuous_conversion_stop()

```
ads1115_error_t ads1115_continuous_conversion_stop (
    ads1115_handle_t * handle)
```

Stop adc continuous conversion mode.

Parameters

<i>handle</i>	Pointer to device handle
---------------	--------------------------

Returns

`ads1115_error_t` Error code

Definition at line 311 of file [ads1115.c](#).

4.7.5.4 ads1115_deinit()

```
ads1115_error_t ads1115_deinit (
    ads1115_handle_t * handle)
```

De-initialize ADS1115 device.

Parameters

<i>handle</i>	Pointer to device handle
---------------	--------------------------

Returns

`ads1115_error_t` Error code

Definition at line 211 of file `ads1115.c`.

4.7.5.5 `ads1115_get_channel()`

```
ads1115_error_t ads1115_get_channel (
    ads1115_handle_t * handle,
    ads1115_mux_t * channel)
```

Set adc input channel from mux.

Parameters

<i>handle</i>	Pointer to device handle
<i>channel</i>	Pointer to input adc channel from mux

Returns

`ads1115_error_t` Error code

Definition at line 409 of file `ads1115.c`.

4.7.5.6 `ads1115_get_compare_alert()`

```
ads1115_error_t ads1115_get_compare_alert (
    ads1115_handle_t * handle,
    ads1115_comp_polarity_t * polarity)
```

Get the alert pin polatiry (ACTIVE_HIGH or ACTIVE_LOW)

Parameters

<i>handle</i>	Pointer to device handle
<i>polarity</i>	Pointer to alert pin polarity

Returns

`ads1115_error_t` Error code

Definition at line 565 of file `ads1115.c`.

4.7.5.7 `ads1115_get_compare_latch()`

```
ads1115_error_t ads1115_get_compare_latch (
    ads1115_handle_t * handle,
    ads1115_comp_latch_t * latch)
```

Get comparator latching mode.

Parameters

<i>handle</i>	Pointer to device handle
<i>latch</i>	Pointer to comparator latch

Returns

`ads1115_error_t` Error code

Definition at line 526 of file [ads1115.c](#).

4.7.5.8 ads1115_get_compare_mode()

```
ads1115_error_t ads1115_get_compare_mode (
    ads1115_handle_t * handle,
    ads1115_comp_mode_t * compare)
```

Get current comparator mode from device.

Parameters

<i>handle</i>	Pointer to device handle
<i>compare</i>	Pointer to interrupt compare mode

Returns

`ads1115_error_t` Error code

Definition at line 448 of file [ads1115.c](#).

4.7.5.9 ads1115_get_compare_queue()

```
ads1115_error_t ads1115_get_compare_queue (
    ads1115_handle_t * handle,
    ads1115_comp_queue_t * comp_queue)
```

Get comparatore queue.

Parameters

<i>handle</i>	Pointer to device handle
<i>comp_queue</i>	Pointer to comparator queue

Returns

`ads1115_error_t` Error code

Definition at line 487 of file [ads1115.c](#).

4.7.5.10 ads1115_get_compare_threshold()

```
ads1115_error_t ads1115_get_compare_threshold (
    ads1115_handle_t * handle,
    int16_t * low_threshold,
    int16_t * high_threshold)
```

Get the comparator low and high threshold.

Parameters

<i>handle</i>	Pointer to device handle
<i>low_threshold</i>	Pointer to compare low threshold
<i>high_threshold</i>	Pointer to compare high threshold

Returns`ads1115_error_t` Error codeDefinition at line 607 of file [ads1115.c](#).**4.7.5.11 ads1115_get_data_rate()**

```
ads1115_error_t ads1115_get_data_rate (
    ads1115_handle_t * handle,
    ads1115_data_rate_t * data_rate)
```

Get adc data sample rate.

Parameters

<i>handle</i>	Pointer to device handle
<i>data_rate</i>	Data rate setting

Returns`ads1115_error_t` Error codeDefinition at line 277 of file [ads1115.c](#).**4.7.5.12 ads1115_get_range()**

```
ads1115_error_t ads1115_get_range (
    ads1115_handle_t * handle,
    ads1115_range_t * range)
```

Get range of adc.

Parameters

<i>handle</i>	Pointer to device handle
<i>range</i>	Maximum voltage range

Returns`ads1115_error_t` Error codeDefinition at line 238 of file [ads1115.c](#).**4.7.5.13 ads1115_init()**

```
ads1115_error_t ads1115_init (
    ads1115_handle_t * handle)
```

Initialize ADS1115 device.

Parameters

<i>handle</i>	Pointer to device handle
---------------	--------------------------

Returns

`ads1115_error_t` Error code

Definition at line 175 of file [ads1115.c](#).

4.7.5.14 ads1115_is_ready()

```
ads1115_error_t ads1115_is_ready (
    ads1115_handle_t * handle,
    bool * flag)
```

Check if conversion is ready.

Parameters

<i>handle</i>	Pointer to device handle
<i>flag</i>	Pointer to store flag status

Returns

`ads1115_error_t` Error code

Definition at line 639 of file [ads1115.c](#).

4.7.5.15 ads1115_set_channel()

```
ads1115_error_t ads1115_set_channel (
    ads1115_handle_t * handle,
    ads1115_mux_t channel)
```

Set adc input channel from mux.

Parameters

<i>handle</i>	Pointer to device handle
<i>channel</i>	Input adc channel from mux

Returns

`ads1115_error_t` Error code

Definition at line 393 of file [ads1115.c](#).

4.7.5.16 ads1115_set_compare_alert()

```
ads1115_error_t ads1115_set_compare_alert (
    ads1115_handle_t * handle,
    ads1115_comp_polarity_t polarity)
```

Set the alert pin polatiry mode (ACTIVE_HIGH or ACTIVE_LOW)

Parameters

<i>handle</i>	Pointer to device handle
<i>polarity</i>	Alert pin polarity

Returns

`ads1115_error_t` Error code

Definition at line 549 of file [ads1115.c](#).

4.7.5.17 ads1115_set_compare_latch()

```
ads1115_error_t ads1115_set_compare_latch (
    ads1115_handle_t * handle,
    ads1115_comp_latch_t latch)
```

Set comparator latching mode.

Parameters

<i>handle</i>	Pointer to device handle
<i>latch</i>	Comparator latch

Returns

`ads1115_error_t` Error code

Definition at line 510 of file [ads1115.c](#).

4.7.5.18 ads1115_set_compare_mode()

```
ads1115_error_t ads1115_set_compare_mode (
    ads1115_handle_t * handle,
    ads1115_comp_mode_t compare)
```

Set comparator mode.

Parameters

<i>handle</i>	Pointer to device handle
<i>compare</i>	Interrupt compare mode

Returns

`ads1115_error_t` Error code

Definition at line 432 of file [ads1115.c](#).

4.7.5.19 ads1115_set_compare_queue()

```
ads1115_error_t ads1115_set_compare_queue (
    ads1115_handle_t * handle,
    ads1115_comp_queue_t comp_queue)
```

Set comparatore queue.

Parameters

<i>handle</i>	Pointer to device handle
<i>comp_queue</i>	Comparator queue

Returns`ads1115_error_t` Error codeDefinition at line 471 of file [ads1115.c](#).**4.7.5.20 ads1115_set_compare_threshold()**

```
ads1115_error_t ads1115_set_compare_threshold (
    ads1115_handle_t * handle,
    int16_t low_threshold,
    int16_t high_threshold)
```

Set the comparator low and high threshold.

Parameters

<i>handle</i>	Pointer to device handle
<i>low_threshold</i>	Compare low threshold
<i>high_threshold</i>	Compare high threshold

Returns`ads1115_error_t` Error codeDefinition at line 588 of file [ads1115.c](#).**4.7.5.21 ads1115_set_data_rate()**

```
ads1115_error_t ads1115_set_data_rate (
    ads1115_handle_t * handle,
    ads1115_data_rate_t data_rate)
```

Set adc data sample rate.

Parameters

<i>handle</i>	Pointer to device handle
<i>data_rate</i>	Data rate setting

Returns`ads1115_error_t` Error codeDefinition at line 261 of file [ads1115.c](#).**4.7.5.22 ads1115_set_range()**

```
ads1115_error_t ads1115_set_range (
    ads1115_handle_t * handle,
    ads1115_range_t range)
```

Set range of adc.

Parameters

<i>handle</i>	Pointer to device handle
<i>range</i>	Maximum voltage range

Returns`ads1115_error_t` Error codeDefinition at line 222 of file `ads1115.c`.**4.7.5.23 ads1115_single_read()**

```
ads1115_error_t ads1115_single_read (
    ads1115_handle_t * handle,
    int16_t * adc_raw,
    float * voltage)
```

Read adc result once.

Parameters

<i>handle</i>	Pointer to device handle
<i>adc_raw</i>	Pointer to raw adc result
<i>voltage</i>	Pointer to converted adc result

Returns`ads1115_error_t` Error codeDefinition at line 347 of file `ads1115.c`.**4.8 ads1115.h**

Go to the documentation of this file.

```
00001
00010
00011 #ifndef ADS1115_H
00012 #define ADS1115_H
00013
00014 #ifdef __cplusplus
00015 extern "C"
00016 {
00017 #endif
00018
00019 #include <stdint.h>
00020 #include <stdbool.h>
00021
00022 /*=====
00023 /* TYPE DEFINITIONS
00024 =====*/
00025
00029 typedef enum {
00030     ADS1115_OK = 0,
00031     ADS1115_ERROR_INVALID_PARAM,
00032     ADS1115_ERROR_I2C_WRITE,
00033     ADS1115_ERROR_I2C_READ,
```

```

00034     ADS1115_ERROR_TIMEOUT,
00035     ADS1115_ERROR_NOT_INITIALIZED,
00036     ADS1115_ERROR_CONVERSION_BUSY,
00037     ADS1115_ERROR_NULL_POINTER
00038 } ads1115_error_t;
00039
00043 typedef enum {
00044     ADS1115_ADDR_GND = 0x48,
00045     ADS1115_ADDR_VDD = 0x49,
00046     ADS1115_ADDR_SDA = 0x4A,
00047     ADS1115_ADDR_SCL = 0x4B,
00048 } ads1115_i2c_addr_t;
00049
00053 typedef enum {
00054     ADS1115_MUX_AIN0_AIN1 = 0,
00055     ADS1115_MUX_AIN0_AIN3 = 1,
00056     ADS1115_MUX_AIN1_AIN3 = 2,
00057     ADS1115_MUX_AIN2_AIN3 = 3,
00058     ADS1115_MUX_AIN0_GND = 4,
00059     ADS1115_MUX_AIN1_GND = 5,
00060     ADS1115_MUX_AIN2_GND = 6,
00061     ADS1115_MUX_AIN3_GND = 7,
00062 } ads1115_mux_t;
00063
00067 typedef enum {
00068     ADS1115_RANGE_6V144 = 0,
00069     ADS1115_RANGE_4V096 = 1,
00070     ADS1115_RANGE_2V048 = 2,
00071     ADS1115_RANGE_1V024 = 3,
00072     ADS1115_RANGE_0V512 = 4,
00073     ADS1115_RANGE_0V256 = 5,
00074 } ads1115_range_t;
00075
00079 typedef enum {
00080     ADS1115_MODE_CONTINUOUS = 0,
00081     ADS1115_MODE_SINGLE_SHOT = 1,
00082 } ads1115_mode_t;
00083
00087 typedef enum {
00088     ADS1115_DR_8_SPS = 0,
00089     ADS1115_DR_16_SPS = 1,
00090     ADS1115_DR_32_SPS = 2,
00091     ADS1115_DR_64_SPS = 3,
00092     ADS1115_DR_128_SPS = 4,
00093     ADS1115_DR_250_SPS = 5,
00094     ADS1115_DR_475_SPS = 6,
00095     ADS1115_DR_860_SPS = 7,
00096 } ads1115_data_rate_t;
00097
00101 typedef enum {
00102     ADS1115_COMP_MODE_TRADITIONAL = 0,
00103     ADS1115_COMP_MODE_WINDOW = 1,
00104 } ads1115_comp_mode_t;
00105
00109 typedef enum {
00110     ADS1115_COMP_POL_ACTIVE_LOW = 0,
00111     ADS1115_COMP_POL_ACTIVE_HIGH = 1,
00112 } ads1115_comp_polarity_t;
00113
00117 typedef enum {
00118     ADS1115_COMP_LAT_NON_LATCHING = 0,
00119     ADS1115_COMP_LAT_LATCHING = 1,
00120 } ads1115_comp_latch_t;
00121
00125 typedef enum {
00126     ADS1115_COMP_QUE_1_CONV = 0,
00127     ADS1115_COMP_QUE_2_CONV = 1,
00128     ADS1115_COMP_QUE_4_CONV = 2,
00129     ADS1115_COMP_QUE_DISABLE = 3,
00130 } ads1115_comp_queue_t;
00131
00132 /*=====
00133 /* CALLBACK FUNCTION TYPES
00134 =====*/
00135
00145 typedef bool (*ads1115_i2c_write_t)(uint8_t device_addr, uint8_t reg_addr, const uint8_t *data,
00146                                         uint8_t length);
00147
00156 typedef bool (*ads1115_i2c_read_t)(uint8_t device_addr, uint8_t reg_addr, uint8_t *data, uint8_t
00157                                         length);
00158
00163 typedef void (*ads1115_delay_ms_t)(uint32_t milliseconds);
00164
00165 /*=====
00166 /* CONFIGURATION STRUCTURES
00167 =====*/
00168

```

```

00172 typedef struct {
00173     ads1115_mux_t mux;
00174     ads1115_range_t range;
00175     ads1115_mode_t mode;
00176     ads1115_data_rate_t data_rate;
00177     ads1115_comp_mode_t comp_mode;
00178     ads1115_comp_polarity_t comp_pol;
00179     ads1115_comp_latch_t comp_latch;
00180     ads1115_comp_queue_t comp_queue;
00181     int16_t low_threshold;
00182     int16_t high_threshold;
00183 } ads1115_config_t;
00184
00185 typedef struct {
00186     ads1115_i2c_addr_t i2c_addr;
00187     ads1115_config_t config;
00188     ads1115_i2c_write_t i2c_write;
00189     ads1115_i2c_read_t i2c_read;
00190     ads1115_delay_ms_t delay_ms;
00191     bool is_initialized;
00192 } ads1115_handle_t;
00193
00194 /*=====
00195 * DEFAULT CONFIGURATION
00196 *=====
00197 */
00198 /* ===== */
00199 /* ===== */
00200
00201 #define ADS1115_DEFAULT_CONFIGURATION { \
00202     .mux = ADS1115_MUX_AIN0_AIN1, \
00203     .range = ADS1115_RANGE_2V048, \
00204     .mode = ADS1115_MODE_SINGLE_SHOT, \
00205     .data_rate = ADS1115_DR_128_SPS, \
00206     .comp_mode = ADS1115_COMP_MODE_TRADITIONAL, \
00207     .comp_pol = ADS1115_COMP_POL_ACTIVE_LOW, \
00208     .comp_latch = ADS1115_COMP_LAT_NON_LATCHING, \
00209     .comp_queue = ADS1115_COMP_QUE_DISABLE, \
00210     .low_threshold = 0x8000, \
00211     .high_threshold = 0x7FFF \
00212 }
00213
00214 /*=====
00215 * PUBLIC API FUNCTIONS
00216 *=====
00217 */
00218
00219
00220 /*=====
00221 * PUBLIC API FUNCTIONS
00222 *=====
00223 */
00224
00225 ads1115_error_t ads1115_init(ads1115_handle_t *handle);
00226
00227 ads1115_error_t ads1115_deinit(ads1115_handle_t *handle);
00228
00229 ads1115_error_t ads1115_set_range(ads1115_handle_t *handle, ads1115_range_t range);
00230
00231 ads1115_error_t ads1115_get_range(ads1115_handle_t *handle, ads1115_range_t *range);
00232
00233 ads1115_error_t ads1115_set_data_rate(ads1115_handle_t *handle, ads1115_data_rate_t data_rate);
00234
00235 ads1115_error_t ads1115_get_data_rate(ads1115_handle_t *handle, ads1115_data_rate_t *data_rate);
00236
00237 ads1115_error_t ads1115_continuous_conversion_start(ads1115_handle_t *handle);
00238
00239 ads1115_error_t ads1115_continuous_conversion_stop(ads1115_handle_t *handle);
00240
00241 ads1115_error_t ads1115_continuous_conversion_read(ads1115_handle_t *handle, int16_t *adc_raw, float *voltage);
00242
00243 ads1115_error_t ads1115_single_read(ads1115_handle_t *handle, int16_t *adc_raw, float *voltage);
00244
00245 ads1115_error_t ads1115_set_channel(ads1115_handle_t *handle, ads1115_mux_t channel);
00246
00247 ads1115_error_t ads1115_get_channel(ads1115_handle_t *handle, ads1115_mux_t *channel);
00248
00249 ads1115_error_t ads1115_set_compare_mode(ads1115_handle_t *handle, ads1115_comp_mode_t compare);
00250
00251 ads1115_error_t ads1115_get_compare_mode(ads1115_handle_t *handle, ads1115_comp_mode_t *compare);
00252
00253 ads1115_error_t ads1115_set_compare_queue(ads1115_handle_t *handle, ads1115_comp_queue_t comp_queue);
00254
00255 ads1115_error_t ads1115_get_compare_queue(ads1115_handle_t *handle, ads1115_comp_queue_t *comp_queue);
00256
00257 ads1115_error_t ads1115_set_compare_alert(ads1115_handle_t *handle, ads1115_comp_polarity_t polarity);
00258
00259 ads1115_error_t ads1115_get_compare_alert(ads1115_handle_t *handle, ads1115_comp_polarity_t *polarity);
00260
00261 ads1115_error_t ads1115_set_compare_threshold(ads1115_handle_t *handle, int16_t low_threshold, int16_t high_threshold);
00262
00263 ads1115_error_t ads1115_get_compare_threshold(ads1115_handle_t *handle, int16_t *low_threshold, int16_t *high_threshold);
00264
00265
00266
00267
00268
00269
00270
00271
00272
00273
00274
00275
00276
00277
00278
00279
00280
00281
00282
00283
00284
00285
00286
00287
00288
00289
00290
00291
00292
00293
00294
00295
00296
00297
00298
00299
00300
00301
00302
00303
00304
00305
00306
00307
00308
00309
00310
00311
00312
00313
00314
00315
00316
00317
00318
00319
00320
00321
00322
00323
00324
00325
00326
00327
00328
00329
00330
00331
00332
00333
00334
00335
00336
00337
00338
00339
00340
00341
00342
00343
00344
00345
00346
00347
00348
00349
00350
00351
00352
00353
00354
00355
00356
00357
00358
00359
00360
00361
00362
00363
00364
00365
00366
00367
00368
00369
00370
00371
00372
00373
00374
00375
00376
00377
00378
00379
00380
00381
00382
00383
00384
00385
00386
00387
00388
00389
00390
00391
00392
00393
00394
00395
00396
00397
00398
00399
00400
00401
00402
00403

```

```
00411 ads1115_error_t ads1115_set_compare_latch(ads1115_handle_t *handle, ads1115_comp_latch_t latch);  
00412  
00420 ads1115_error_t ads1115_get_compare_latch(ads1115_handle_t *handle, ads1115_comp_latch_t *latch);  
00421  
00429 ads1115_error_t ads1115_is_ready(ads1115_handle_t *handle, bool *flag);  
00430  
00431 #ifdef __cplusplus  
00432 }  
00433 #endif  
00434  
00435 #endif /* ADS1115_H */
```

Index

ads1115.c
ADS1115_COMP_LAT_MASK, 15
ADS1115_COMP_LAT_SHIFT, 15
ADS1115_COMP_MODE_MASK, 15
ADS1115_COMP_MODE_SHIFT, 15
ADS1115_COMP_POL_MASK, 15
ADS1115_COMP_POL_SHIFT, 16
ADS1115_COMP_QUE_MASK, 16
ADS1115_COMP_QUE_SHIFT, 16
ads1115_continuous_conversion_read, 18
ads1115_continuous_conversion_start, 19
ads1115_continuous_conversion_stop, 19
ads1115_deinit, 19
ADS1115_DR_MASK, 16
ADS1115_DR_SHIFT, 16
ads1115_get_channel, 20
ads1115_get_compare_alert, 20
ads1115_get_compare_latch, 20
ads1115_get_compare_mode, 21
ads1115_get_compare_queue, 21
ads1115_get_compare_threshold, 21
ads1115_get_data_rate, 22
ads1115_get_range, 22
ads1115_init, 22
ads1115_is_ready, 23
ADS1115_MAX_VALUE, 16
ADS1115_MIN_VALUE, 16
ADS1115_MODE_MASK, 16
ADS1115_MODE_SHIFT, 17
ADS1115_MUX_MASK, 17
ADS1115_MUX_SHIFT, 17
ADS1115_OS_IDLE, 17
ADS1115_OS_MASK, 17
ADS1115_OS_SHIFT, 17
ADS1115_OS_START_SINGLE, 17
ADS1115_PGA_MASK, 17
ADS1115_PGA_SHIFT, 18
ADS1115_REG_CONFIG, 18
ADS1115_REG_CONVERSION, 18
ADS1115_REG_HI_THRESH, 18
ADS1115_REG_LO_THRESH, 18
ads1115_set_channel, 23
ads1115_set_compare_alert, 23
ads1115_set_compare_latch, 24
ads1115_set_compare_mode, 24
ads1115_set_compare_queue, 24
ads1115_set_compare_threshold, 25
ads1115_set_data_rate, 25
ads1115_set_range, 25
ads1115_single_read, 26
ads1115.h
ADS1115_ADDR_GND, 40
ADS1115_ADDR_SCL, 40
ADS1115_ADDR_SDA, 40
ADS1115_ADDR_VDD, 40
ADS1115_COMP_LAT_LATCHING, 39
ADS1115_COMP_LAT_NON_LATCHING, 39
ads1115_comp_latch_t, 38
ads1115_comp_mode_t, 39
ADS1115_COMP_MODE_TRADITIONAL, 39
ADS1115_COMP_MODE_WINDOW, 39
ADS1115_COMP_POL_ACTIVE_HIGH, 39
ADS1115_COMP_POL_ACTIVE_LOW, 39
ads1115_comp_polarity_t, 39
ADS1115_COMP_QUE_1_CONV, 39
ADS1115_COMP_QUE_2_CONV, 39
ADS1115_COMP_QUE_4_CONV, 39
ADS1115_COMP_QUE_DISABLE, 39
ads1115_comp_queue_t, 39
ads1115_continuous_conversion_read, 41
ads1115_continuous_conversion_start, 42
ads1115_continuous_conversion_stop, 42
ads1115_data_rate_t, 39
ADS1115_DEFAULT_CONFIGURATION, 37
ads1115_deinit, 42
ads1115_delay_ms_t, 37
ADS1115_DR_128_SPS, 40
ADS1115_DR_16_SPS, 40
ADS1115_DR_250_SPS, 40
ADS1115_DR_32_SPS, 40
ADS1115_DR_475_SPS, 40
ADS1115_DR_64_SPS, 40
ADS1115_DR_860_SPS, 40
ADS1115_DR_8_SPS, 40
ADS1115_ERROR_CONVERSION_BUSY, 40
ADS1115_ERROR_I2C_READ, 40
ADS1115_ERROR_I2C_WRITE, 40
ADS1115_ERROR_INVALID_PARAM, 40
ADS1115_ERROR_NOT_INITIALIZED, 40
ADS1115_ERROR_NULL_POINTER, 40
ads1115_error_t, 40
ADS1115_ERROR_TIMEOUT, 40
ads1115_get_channel, 43
ads1115_get_compare_alert, 43
ads1115_get_compare_latch, 43
ads1115_get_compare_mode, 44
ads1115_get_compare_queue, 44
ads1115_get_compare_threshold, 44

ads1115_get_data_rate, 45
ads1115_get_range, 45
ads1115_i2c_addr_t, 40
ads1115_i2c_read_t, 38
ads1115_i2c_write_t, 38
ads1115_init, 45
ads1115_is_ready, 46
ADS1115_MODE_CONTINUOUS, 41
ADS1115_MODE_SINGLE_SHOT, 41
ads1115_mode_t, 40
ADS1115_MUX_AIN0_AIN1, 41
ADS1115_MUX_AIN0_AIN3, 41
ADS1115_MUX_AIN0_GND, 41
ADS1115_MUX_AIN1_AIN3, 41
ADS1115_MUX_AIN1_GND, 41
ADS1115_MUX_AIN2_AIN3, 41
ADS1115_MUX_AIN2_GND, 41
ADS1115_MUX_AIN3_GND, 41
ads1115_mux_t, 41
ADS1115_OK, 40
ADS1115_RANGE_0V256, 41
ADS1115_RANGE_0V512, 41
ADS1115_RANGE_1V024, 41
ADS1115_RANGE_2V048, 41
ADS1115_RANGE_4V096, 41
ADS1115_RANGE_6V144, 41
ads1115_range_t, 41
ads1115_set_channel, 46
ads1115_set_compare_alert, 46
ads1115_set_compare_latch, 47
ads1115_set_compare_mode, 47
ads1115_set_compare_queue, 47
ads1115_set_compare_threshold, 48
ads1115_set_data_rate, 48
ads1115_set_range, 48
ads1115_single_read, 49
ADS1115_ADDR_GND
 ads1115.h, 40
ADS1115_ADDR_SCL
 ads1115.h, 40
ADS1115_ADDR_SDA
 ads1115.h, 40
ADS1115_ADDR_VDD
 ads1115.h, 40
ADS1115_COMP_LAT_LATCHING
 ads1115.h, 39
ADS1115_COMP_LAT_MASK
 ads1115.c, 15
ADS1115_COMP_LAT_NON_LATCHING
 ads1115.h, 39
ADS1115_COMP_LAT_SHIFT
 ads1115.c, 15
ads1115_comp_latch_t
 ads1115.h, 38
ADS1115_COMP_MODE_MASK
 ads1115.c, 15
ADS1115_COMP_MODE_SHIFT
 ads1115.c, 15
 ads1115_comp_mode_t
 ads1115.h, 39
 ADS1115_COMP_MODE_TRADITIONAL
 ads1115.h, 39
 ADS1115_COMP_MODE_WINDOW
 ads1115.h, 39
 ADS1115_COMP_POL_ACTIVE_HIGH
 ads1115.h, 39
 ADS1115_COMP_POL_ACTIVE_LOW
 ads1115.h, 39
 ADS1115_COMP_POL_MASK
 ads1115.c, 15
 ADS1115_COMP_POL_SHIFT
 ads1115.c, 16
 ads1115_comp_polarity_t
 ads1115.h, 39
 ADS1115_COMP_QUE_1_CONV
 ads1115.h, 39
 ADS1115_COMP_QUE_2_CONV
 ads1115.h, 39
 ADS1115_COMP_QUE_4_CONV
 ads1115.h, 39
 ADS1115_COMP_QUE_DISABLE
 ads1115.h, 39
 ADS1115_COMP_QUE_MASK
 ads1115.c, 16
 ADS1115_COMP_QUE_SHIFT
 ads1115.c, 16
 ads1115_comp_queue_t
 ads1115.h, 39
 ads1115_config_t, 5
 comp_latch, 5
 comp_mode, 5
 comp_pol, 6
 comp_queue, 6
 data_rate, 6
 high_threshold, 6
 low_threshold, 6
 mode, 6
 mux, 7
 range, 7
 ads1115_continuous_conversion_read
 ads1115.c, 18
 ads1115.h, 41
 ads1115_continuous_conversion_start
 ads1115.c, 19
 ads1115.h, 42
 ads1115_continuous_conversion_stop
 ads1115.c, 19
 ads1115.h, 42
 ads1115_data_rate_t
 ads1115.h, 39
 ADS1115_DEFAULT_CONFIGURATION
 ads1115.h, 37
 ads1115_deinit
 ads1115.c, 19
 ads1115.h, 42
 ads1115_delay_ms

ads1115_interface.c, 9
ads1115_interface.h, 11
ads1115_delay_ms_t
 ads1115.h, 37
ADS1115_DR_128_SPS
 ads1115.h, 40
ADS1115_DR_16_SPS
 ads1115.h, 40
ADS1115_DR_250_SPS
 ads1115.h, 40
ADS1115_DR_32_SPS
 ads1115.h, 40
ADS1115_DR_475_SPS
 ads1115.h, 40
ADS1115_DR_64_SPS
 ads1115.h, 40
ADS1115_DR_860_SPS
 ads1115.h, 40
ADS1115_DR_8_SPS
 ads1115.h, 40
ADS1115_DR_MASK
 ads1115.c, 16
ADS1115_DR_SHIFT
 ads1115.c, 16
ADS1115_ERROR_CONVERSION_BUSY
 ads1115.h, 40
ADS1115_ERROR_I2C_READ
 ads1115.h, 40
ADS1115_ERROR_I2C_WRITE
 ads1115.h, 40
ADS1115_ERROR_INVALID_PARAM
 ads1115.h, 40
ADS1115_ERROR_NOT_INITIALIZED
 ads1115.h, 40
ADS1115_ERROR_NULL_POINTER
 ads1115.h, 40
ads1115_error_t
 ads1115.h, 40
ADS1115_ERROR_TIMEOUT
 ads1115.h, 40
ads1115_get_channel
 ads1115.c, 20
 ads1115.h, 43
ads1115_get_compare_alert
 ads1115.c, 20
 ads1115.h, 43
ads1115_get_compare_latch
 ads1115.c, 20
 ads1115.h, 43
ads1115_get_compare_mode
 ads1115.c, 21
 ads1115.h, 44
ads1115_get_compare_queue
 ads1115.c, 21
 ads1115.h, 44
ads1115_get_compare_threshold
 ads1115.c, 21
 ads1115.h, 44
ads1115_get_data_rate
 ads1115.c, 22
 ads1115.h, 45
ads1115_get_range
 ads1115.c, 22
 ads1115.h, 45
ads1115_handle_t, 7
 config, 8
 delay_ms, 8
 i2c_addr, 8
 i2c_read, 8
 i2c_write, 8
 is_initialized, 8
ads1115_i2c_addr_t
 ads1115.h, 40
ads1115_i2c_read
 ads1115_interface.c, 10
 ads1115_interface.h, 12
ads1115_i2c_read_t
 ads1115.h, 38
ads1115_i2c_write
 ads1115_interface.c, 10
 ads1115_interface.h, 12
ads1115_i2c_write_t
 ads1115.h, 38
ads1115_init
 ads1115.c, 22
 ads1115.h, 45
ads1115_interface.c
 ads1115_delay_ms, 9
 ads1115_i2c_read, 10
 ads1115_i2c_write, 10
ads1115_interface.h
 ads1115_delay_ms, 11
 ads1115_i2c_read, 12
 ads1115_i2c_write, 12
ads1115_is_ready
 ads1115.c, 23
 ads1115.h, 46
ADS1115_MAX_VALUE
 ads1115.c, 16
ADS1115_MIN_VALUE
 ads1115.c, 16
ADS1115_MODE_CONTINUOUS
 ads1115.h, 41
ADS1115_MODE_MASK
 ads1115.c, 16
ADS1115_MODE_SHIFT
 ads1115.c, 17
ADS1115_MODE_SINGLE_SHOT
 ads1115.h, 41
ads1115_mode_t
 ads1115.h, 40
ADS1115_MUX_AIN0_AIN1
 ads1115.h, 41
ADS1115_MUX_AIN0_AIN3
 ads1115.h, 41
ADS1115_MUX_AIN0_GND

ads1115.h, 41
 ADS1115_MUX_AIN1_AIN3
 ads1115.h, 41
 ADS1115_MUX_AIN1_GND
 ads1115.h, 41
 ADS1115_MUX_AIN2_AIN3
 ads1115.h, 41
 ADS1115_MUX_AIN2_GND
 ads1115.h, 41
 ADS1115_MUX_AIN3_GND
 ads1115.h, 41
 ADS1115_MUX_MASK
 ads1115.c, 17
 ADS1115_MUX_SHIFT
 ads1115.c, 17
 ads1115_mux_t
 ads1115.h, 41
 ADS1115_OK
 ads1115.h, 40
 ADS1115_OS_IDLE
 ads1115.c, 17
 ADS1115_OS_MASK
 ads1115.c, 17
 ADS1115_OS_SHIFT
 ads1115.c, 17
 ADS1115_OS_START_SINGLE
 ads1115.c, 17
 ADS1115_PGA_MASK
 ads1115.c, 17
 ADS1115_PGA_SHIFT
 ads1115.c, 18
 ADS1115_RANGE_0V256
 ads1115.h, 41
 ADS1115_RANGE_0V512
 ads1115.h, 41
 ADS1115_RANGE_1V024
 ads1115.h, 41
 ADS1115_RANGE_2V048
 ads1115.h, 41
 ADS1115_RANGE_4V096
 ads1115.h, 41
 ADS1115_RANGE_6V144
 ads1115.h, 41
 ads1115_range_t
 ads1115.h, 41
 ADS1115_REG_CONFIG
 ads1115.c, 18
 ADS1115_REG_CONVERSION
 ads1115.c, 18
 ADS1115_REG_HI_THRESH
 ads1115.c, 18
 ADS1115_REG_LO_THRESH
 ads1115.c, 18
 ads1115_set_channel
 ads1115.c, 23
 ads1115.h, 46
 ads1115_set_compare_alert
 ads1115.c, 23

ads1115.h, 46
 ads1115_set_compare_latch
 ads1115.c, 24
 ads1115.h, 47
 ads1115_set_compare_mode
 ads1115.c, 24
 ads1115.h, 47
 ads1115_set_compare_queue
 ads1115.c, 24
 ads1115.h, 47
 ads1115_set_compare_threshold
 ads1115.c, 25
 ads1115.h, 48
 ads1115_set_data_rate
 ads1115.c, 25
 ads1115.h, 48
 ads1115_set_range
 ads1115.c, 25
 ads1115.h, 48
 ads1115_single_read
 ads1115.c, 26
 ads1115.h, 49

comp_latch
 ads1115_config_t, 5
 comp_mode
 ads1115_config_t, 5
 comp_pol
 ads1115_config_t, 6
 comp_queue
 ads1115_config_t, 6
 config
 ads1115_handle_t, 8

data_rate
 ads1115_config_t, 6
 delay_ms
 ads1115_handle_t, 8

high_threshold
 ads1115_config_t, 6

i2c_addr
 ads1115_handle_t, 8
 i2c_read
 ads1115_handle_t, 8
 i2c_write
 ads1115_handle_t, 8

interface/ads1115_interface.c, 9, 11
 interface/ads1115_interface.h, 11, 13
 is_initialized
 ads1115_handle_t, 8

low_threshold
 ads1115_config_t, 6

mode
 ads1115_config_t, 6

mux
 ads1115_config_t, 7

range
 ads1115_config_t, 7

src/ads1115.c, 13, 26
src/ads1115.h, 34, 49