

Data acquisition with the ADS1115 on the raspberry PI

Generated by Doxygen 1.9.8

| | |
|--|-----------|
| 1 rpi_ads1115 | 1 |
| 2 Class Index | 3 |
| 2.1 Class List | 3 |
| 3 File Index | 5 |
| 3.1 File List | 5 |
| 4 Class Documentation | 7 |
| 4.1 ADS1115rpi Class Reference | 7 |
| 4.1.1 Detailed Description | 7 |
| 4.1.2 Member Typedef Documentation | 8 |
| 4.1.2.1 ADSCallbackInterface | 8 |
| 4.1.3 Member Function Documentation | 8 |
| 4.1.3.1 registerCallback() | 8 |
| 4.1.3.2 setChannel() | 8 |
| 4.1.3.3 start() | 8 |
| 4.2 ADS1115settings Struct Reference | 9 |
| 4.2.1 Detailed Description | 10 |
| 4.2.2 Member Data Documentation | 10 |
| 4.2.2.1 DEFAULT_ADS1115_ADDRESS | 10 |
| 5 File Documentation | 11 |
| 5.1 ads1115rpi.h | 11 |
| Index | 15 |

Chapter 1

rpi_ads1115

Raspberry PI C++ library for the ADS1115

github: https://github.com/berndporr/rpi_ads1115

Chapter 2

Class Index

2.1 Class List

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

[ADS1115](#)[rpi](#)

This class reads data from the ADS1115 in the background (separate thread) and calls a callback function whenever data is available

[7](#)

[ADS1115](#)[settings](#)

ADS1115 initial settings when starting the device

[9](#)

Chapter 3

File Index

3.1 File List

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

| | |
|--|----|
| ads1115rpi.h | 11 |
|--|----|

Chapter 4

Class Documentation

4.1 ADS1115rpi Class Reference

This class reads data from the ADS1115 in the background (separate thread) and calls a callback function whenever data is available.

```
#include <ads1115rpi.h>
```

Public Types

- using [ADSCallbackInterface](#) = std::function< void(float)>
Callback function type when a new sample is available.

Public Member Functions

- [~ADS1115rpi](#) ()
Destructor which makes sure the data acquisition stops on exit.
- void [registerCallback](#) ([ADSCallbackInterface](#) ci)
Registers callback with this class.
- void [setChannel](#) ([ADS1115Settings::Input](#) channel)
Selects a different channel at the multiplexer while running.
- void [start](#) ([ADS1115Settings](#) settings=[ADS1115Settings](#)())
Starts the data acquisition in the background and the callback is called with new samples.
- [ADS1115Settings](#) [getADS1115Settings](#) () const
Returns the current settings.
- void [stop](#) ()
Stops the data acquisition.

4.1.1 Detailed Description

This class reads data from the ADS1115 in the background (separate thread) and calls a callback function whenever data is available.

4.1.2 Member Typedef Documentation

4.1.2.1 ADSCallbackInterface

```
using ADS1115rpi::ADSCallbackInterface = std::function<void(float)>
```

Callback function type when a new sample is available.

Callback value is in volt.

4.1.3 Member Function Documentation

4.1.3.1 registerCallback()

```
void ADS1115rpi::registerCallback (
    ADSCallbackInterface ci ) [inline]
```

Registers callback with this class.

Parameters

| | |
|-----------|---|
| <i>ci</i> | Callback method and needs to be of type ADSCallbackInterface. |
|-----------|---|

4.1.3.2 setChannel()

```
void ADS1115rpi::setChannel (
    ADS1115settings::Input channel )
```

Selects a different channel at the multiplexer while running.

Call this in the callback handler hasSample() to cycle through different channels.

Parameters

| | |
|----------------|-------------------------------|
| <i>channel</i> | Sets the channel from A0..A3. |
|----------------|-------------------------------|

4.1.3.3 start()

```
void ADS1115rpi::start (
    ADS1115settings settings = ADS1115settings() )
```

Starts the data acquisition in the background and the callback is called with new samples.

Parameters

| | |
|-----------------|-----------------------------|
| <i>settings</i> | A struct with the settings. |
|-----------------|-----------------------------|

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

- ads1115rpi.h

4.2 ADS1115settings Struct Reference

ADS1115 initial settings when starting the device.

```
#include <ads1115rpi.h>
```

Public Types

- enum [SamplingRates](#) {
FS8HZ = 0 , **FS16HZ** = 1 , **FS32HZ** = 2 , **FS64HZ** = 3 ,
FS128HZ = 4 , **FS250HZ** = 5 , **FS475HZ** = 6 , **FS860HZ** = 7 }
Sampling rates.
- enum [PGA](#) { **FSR2_048** = 2 , **FSR1_024** = 3 , **FSR0_512** = 4 , **FSR0_256** = 5 }
Full scale range: 2.048V, 1.024V, 0.512V or 0.256V.
- enum [Input](#) { **AIN0** = 0 , **AIN1** = 1 , **AIN2** = 2 , **AIN3** = 3 }
Channel indices.

Public Member Functions

- unsigned **getSamplingRate** () const
Get the sampling rate in Hz.

Public Attributes

- int **i2c_bus** = 1
I2C bus used (99% always set to one)
- uint8_t **address** = [DEFAULT_ADS1115_ADDRESS](#)
I2C address of the ads1115.
- [SamplingRates](#) **samplingRate** = FS8HZ
Sampling rate requested.
- [PGA](#) **pgaGain** = FSR2_048
Requested full scale range.
- [Input](#) **channel** = AIN0
Requested input channel (AIN0..AIN3)
- int **drdy_chip** = 0
GPIO Chip number which receives the Data Ready signal.
- int **drdy_gpio** = [DEFAULT_ALERT_RDY_TO_GPIO](#)
GPIO pin connected to ALERT/RDY.

Static Public Attributes

- static constexpr uint8_t [DEFAULT_ADS1115_ADDRESS](#) = 0x48
The default address of the ADS1115.
- static constexpr int **DEFAULT_ALERT_RDY_TO_GPIO** = 17
Default GPIO pin for the ALRT/DRY signal.

4.2.1 Detailed Description

ADS1115 initial settings when starting the device.

4.2.2 Member Data Documentation

4.2.2.1 DEFAULT_ADS1115_ADDRESS

```
constexpr uint8_t ADS1115settings::DEFAULT_ADS1115_ADDRESS = 0x48 [static], [constexpr]
```

The default address of the ADS1115.

48H is the address of the ADS1115 if the ADR pin is pulled to GND and taken here as the default address.

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

- ads1115rpi.h

Chapter 5

File Documentation

5.1 ads1115rpi.h

```
00001 #ifndef __ADS1115RPI_H
00002 #define __ADS1115RPI_H
00003
00004 /*
00005  * ADS1115 class to read data at a given sampling rate
00006  *
00007  * Copyright (c) 2007 MontaVista Software, Inc.
00008  * Copyright (c) 2007 Anton Vorontsov <avorontsov@ru.mvista.com>
00009  * Copyright (c) 2013-2025 Bernd Porr <mail@berndporr.me.uk>
00010  *
00011  * This program is free software; you can redistribute it and/or modify
00012  * it under the terms of the GNU General Public License as published by
00013  * the Free Software Foundation; either version 2 of the License.
00014  *
00015  */
00016 #include <stdint.h>
00017 #include <unistd.h>
00018 #include <stdio.h>
00019 #include <stdlib.h>
00020 #include <assert.h>
00021 #include <linux/i2c-dev.h>
00022 #include <thread>
00023 #include <gpio.h>
00024 #include <functional>
00025
00026 // enable debug messages and error messages to stderr
00027 #ifndef NDEBUG
00028 #define DEBUG
00029 #endif
00030
00031 struct ADS1115Settings
00032 {
00033     int i2c_bus = 1;
00034
00035     static constexpr uint8_t DEFAULT_ADS1115_ADDRESS = 0x48;
00036
00037     uint8_t address = DEFAULT_ADS1115_ADDRESS;
00038
00039     enum SamplingRates
00040     {
00041         FS8HZ = 0,
00042         FS16HZ = 1,
00043         FS32HZ = 2,
00044         FS64HZ = 3,
00045         FS128HZ = 4,
00046         FS250HZ = 5,
00047         FS475HZ = 6,
00048         FS860HZ = 7
00049     };
00050
00051     inline unsigned getSamplingRate() const
00052     {
00053         const unsigned SamplingRateEnum2Value[8] =
00054             {8, 16, 32, 64, 128, 250, 475, 860};
00055         return SamplingRateEnum2Value[samplingRate];
00056     }
00057 }
00058
```

```

00082     SamplingRates samplingRate = FS8HZ;
00083
00084     enum PGA
00085     {
00086         FSR2_048 = 2,
00087         FSR1_024 = 3,
00088         FSR0_512 = 4,
00089         FSR0_256 = 5
00090     };
00091
00092     PGA pgaGain = FSR2_048;
00093
00094     enum Input
00095     {
00096         AIN0 = 0,
00097         AIN1 = 1,
00098         AIN2 = 2,
00099         AIN3 = 3
00100     };
00101
00102     Input channel = AIN0;
00103
00104     int drdy_chip = 0;
00105
00106     static constexpr int DEFAULT_ALERT_RDY_TO_GPIO = 17;
00107
00108     int drdy_gpio = DEFAULT_ALERT_RDY_TO_GPIO;
00109 };
00110
00111 class ADS1115rpi
00112 {
00113 public:
00114     ~ADS1115rpi()
00115     {
00116         stop();
00117     }
00118
00119     using ADSCallbackInterface = std::function<void(float)>;
00120
00121     void registerCallback(ADSCallbackInterface ci)
00122     {
00123         adsCallbackInterface = ci;
00124     }
00125
00126     void setChannel(ADS1115settings::Input channel);
00127
00128     void start(ADS1115settings settings = ADS1115settings());
00129
00130     ADS1115settings getADS1115settings() const
00131     {
00132         return ads1115settings;
00133     }
00134
00135     void stop();
00136 private:
00137     ADS1115settings ads1115settings;
00138
00139     void dataReady();
00140
00141     void worker();
00142
00143     void i2c_writeWord(uint8_t reg, unsigned data);
00144     unsigned i2c_readWord(uint8_t reg);
00145     int i2c_readConversion();
00146
00147     const uint8_t reg_config = 1;
00148     const uint8_t reg_lo_thres = 2;
00149     const uint8_t reg_hi_thres = 3;
00150
00151     float fullScaleVoltage()
00152     {
00153         switch (ads1115settings.pgaGain)
00154         {
00155             case ADS1115settings::FSR2_048:
00156                 return 2.048f;
00157             case ADS1115settings::FSR1_024:
00158                 return 1.024f;
00159             case ADS1115settings::FSR0_512:
00160                 return 0.512f;
00161             case ADS1115settings::FSR0_256:
00162                 return 0.256f;
00163         }
00164         assert(1 == 0);
00165         return 0;
00166     }
00167 }

```



```
00224
00225     std::shared_ptr<gpiod::chip> chip;
00226     std::shared_ptr<gpiod::line_request> request;
00227
00228     std::thread thr;
00229
00230     int fd_i2c = -1;
00231
00232     bool running = false;
00233
00234     ADSCallbackInterface adsCallbackInterface;
00235
00236     // timeout if no DATA READY has been received
00237     static constexpr int64_t ISR_TIMEOUT_MS = 500;
00238 };
00239
00240 #endif
```


Index

- ADS1115rpi, [7](#)
 - ADSCallbackInterface, [8](#)
 - registerCallback, [8](#)
 - setChannel, [8](#)
 - start, [8](#)
- ADS1115settings, [9](#)
 - DEFAULT_ADS1115_ADDRESS, [10](#)
- ADSCallbackInterface
 - ADS1115rpi, [8](#)
- DEFAULT_ADS1115_ADDRESS
 - ADS1115settings, [10](#)
- registerCallback
 - ADS1115rpi, [8](#)
- rpi_ads1115, [1](#)
- setChannel
 - ADS1115rpi, [8](#)
- start
 - ADS1115rpi, [8](#)