

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痈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:

ADS1115rpi	
This class reads data from the ADS1115 in the background (separate thread) and calls a callback function whenever data is available	7
ADS1115settings	
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

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

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

<code>channel</code>	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

<code>settings</code>	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_AMPS1115_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_AMPS1115_ADDRESS** = 0x48

The default address of the ADS1115.
- static constexpr int **DEFAULT_ALERT_RDY_TO_GPIO** = 17

Default GPIO pin for the ALERT/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_AMS1115_ADDRESS

```
constexpr uint8_t ADS1115settings::DEFAULT_AMS1115_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 <gpiod.hpp>
00024 #include <functional>
00025
00026 // enable debug messages and error messages to stderr
00027 #ifndef NDEBUG
00028 #define DEBUG
00029 #endif
00030
00034 struct ADS1115settings
00035 {
00036
00040     int i2c_bus = 1;
00041
00047     static constexpr uint8_t DEFAULT_AMBIENT_ADDRESS = 0x48;
00048
00052     uint8_t address = DEFAULT_AMBIENT_ADDRESS;
00053
00057     enum SamplingRates
00058     {
00059         FS8HZ = 0,
00060         FS16HZ = 1,
00061         FS32HZ = 2,
00062         FS64HZ = 3,
00063         FS128HZ = 4,
00064         FS250HZ = 5,
00065         FS475HZ = 6,
00066         FS860HZ = 7
00067     };
00068
00072     inline unsigned getSamplingRate() const
00073     {
00074         const unsigned SamplingRateEnum2Value[8] =
00075             {8, 16, 32, 64, 128, 250, 475, 860};
00076         return SamplingRateEnum2Value[samplingRate];
00077     }
00078 }
```

```
00082     SamplingRates samplingRate = FS8HZ;
00083
00087     enum PGA
00088     {
00089         FSR2_048 = 2,
00090         FSR1_024 = 3,
00091         FSR0_512 = 4,
00092         FSR0_256 = 5
00093     };
00094
00098     PGA pgaGain = FSR2_048;
00099
00103     enum Input
00104     {
00105         AIN0 = 0,
00106         AIN1 = 1,
00107         AIN2 = 2,
00108         AIN3 = 3
00109     };
00110
00114     Input channel = AIN0;
00115
00119     int drdy_chip = 0;
00120
00124     static constexpr int DEFAULT_ALERT_RDY_TO_GPIO = 17;
00125
00129     int drdy_gpio = DEFAULT_ALERT_RDY_TO_GPIO;
00130 };
00131
00136 class ADS1115rpi
00137 {
00138
00139     public:
00140         ~ADS1115rpi()
00141     {
00142         stop();
00143     }
00144
00145     using ADSCallbackInterface = std::function<void(float)>;
00146
00147     void registerCallback(ADSCallbackInterface ci)
00148     {
00149         adsCallbackInterface = ci;
00150     }
00151
00152     void setChannel(ADS1115settings::Input channel);
00153
00154     void start(ADS1115settings settings = ADS1115settings());
00155
00156     ADS1115settings getADS1115settings() const
00157     {
00158         return ads1115settings;
00159     }
00160
00161     void stop();
00162
00163     private:
00164         ADS1115settings ads1115settings;
00165
00166         void dataReady();
00167
00168         void worker();
00169
00170         void i2c_writeWord(uint8_t reg, unsigned data);
00171         unsigned i2c_readWord(uint8_t reg);
00172         int i2c_readConversion();
00173
00174         const uint8_t reg_config = 1;
00175         const uint8_t reg_lo_thres = 2;
00176         const uint8_t reg_hi_thres = 3;
00177
00178         float fullScaleVoltage()
00179         {
00180             switch (ads1115settings.pgaGain)
00181             {
00182                 case ADS1115settings::FSR2_048:
00183                     return 2.048f;
00184                 case ADS1115settings::FSR1_024:
00185                     return 1.024f;
00186                 case ADS1115settings::FSR0_512:
00187                     return 0.512f;
00188                 case ADS1115settings::FSR0_256:
00189                     return 0.256f;
00190             }
00191             assert(1 == 0);
00192             return 0;
00193         }
```

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


Index

ADS1115rpi, [7](#)
 ADSCallbackInterface, [8](#)
 registerCallback, [8](#)
 setChannel, [8](#)
 start, [8](#)
ADS1115settings, [9](#)
 DEFAULT_AMS1115_ADDRESS, [10](#)
ADSCallbackInterface
 ADS1115rpi, [8](#)

DEFAULT_AMS1115_ADDRESS
 ADS1115settings, [10](#)

registerCallback
 ADS1115rpi, [8](#)
rpi_ads1115, [1](#)

setChannel
 ADS1115rpi, [8](#)
start
 ADS1115rpi, [8](#)