

Data acquisition with the AD7705 on the raspberry PI

Generated by Doxygen 1.8.17

1 Data acquisition with the AD7705 on the raspberry PI!	1
2 Class Index	3
2.1 Class List	3
3 Class Documentation	5
3.1 AD7705callback Class Reference	5
3.1.1 Detailed Description	5
3.1.2 Member Function Documentation	5
3.1.2.1 hasSample()	5
3.2 AD7705Comm Class Reference	6
3.2.1 Detailed Description	6
3.2.2 Constructor & Destructor Documentation	6
3.2.2.1 AD7705Comm()	6
3.2.3 Member Function Documentation	7
3.2.3.1 registerCallback()	7
3.3 AD7705settings Struct Reference	7
3.3.1 Detailed Description	8
Index	9

Chapter 1

Data acquisition with the AD7705 on the raspberry PI!

The AD7705 is a two channel sigma delta converter which has differential inputs, a PGA and programmable data rates. It's perfect for slowly changing inputs such as pressure, temperature, heart rate etc.

This repo offers the class `AD7705Comm` which does all the low level communications with the AD7705. The user just need to register a callback handler which then returns the samples in realtime.

The class uses the DRDY of the AD7705 connected to Port 22 and waits for a falling edge on this port to read the data. This is done via interrupts / poll so that the ADC process sleeps until new data becomes available.

Github: https://github.com/berndporr/rpi_AD7705_daq

Chapter 2

Class Index

2.1 Class List

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

AD7705callback	Callback for new samples which needs to be implemented by the main program	5
AD7705Comm	This class reads data from the AD7705 in the background (separate thread) and calls a callback function whenever data is available	6
AD7705settings	Contains all settings for the data acquisition	7

Chapter 3

Class Documentation

3.1 AD7705callback Class Reference

Callback for new samples which needs to be implemented by the main program.

```
#include <AD7705Comm.h>
```

Public Member Functions

- virtual void [hasSample](#) (float sample)=0
Called after a sample has arrived.

3.1.1 Detailed Description

Callback for new samples which needs to be implemented by the main program.

The function hasSample needs to be overloaded in the main program.

3.1.2 Member Function Documentation

3.1.2.1 hasSample()

```
virtual void AD7705callback::hasSample (  
    float sample ) [pure virtual]
```

Called after a sample has arrived.

Parameters

<i>sample</i>	Contains the ADC reading in Volt.
---------------	-----------------------------------

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

- AD7705Comm.h

3.2 AD7705Comm Class Reference

This class reads data from the AD7705 in the background (separate thread) and calls a callback function whenever data is available.

```
#include <AD7705Comm.h>
```

Public Member Functions

- [AD7705Comm](#) ([AD7705settings](#) settings)
Constructor.
- [~AD7705Comm](#) ()
Destructor which makes sure the data acquisition has stopped.
- void [registerCallback](#) ([AD7705callback](#) *cb)
Registers the callback which is called whenever there is a sample.
- void [unRegisterCallback](#) ()
Unregisters the callback to the callback interface.
- void [start](#) ()
Starts the data acquisition.
- void [stop](#) ()
Stops the data acquisition.

3.2.1 Detailed Description

This class reads data from the AD7705 in the background (separate thread) and calls a callback function whenever data is available.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 AD7705Comm()

```
AD7705Comm::AD7705Comm (
    AD7705settings settings )
```

Constructor.

Opens the SPI device and waits for to start the acquisition.

Parameters

<i>settings</i>	All AD7705 settings.
-----------------	----------------------

3.2.3 Member Function Documentation

3.2.3.1 registerCallback()

```
void AD7705Comm::registerCallback (
    AD7705callback * cb )
```

Registers the callback which is called whenever there is a sample.

Parameters

<i>cb</i>	Pointer to the callback interface.
-----------	------------------------------------

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

- AD7705Comm.h

3.3 AD7705settings Struct Reference

Contains all settings for the data acquisition.

```
#include <AD7705Comm.h>
```

Public Types

- enum [SamplingRates](#) { **FS50HZ** = 0, **FS60HZ** = 1, **FS250HZ** = 2, **FS500HZ** = 3 }
Sampling rates.
- enum [PGAGains](#) {
G1 = 0, **G2** = 1, **G4** = 2, **G8** = 3,
G16 = 4, **G32** = 5, **G64** = 6, **G128** = 7 }
Gains of the PGA.
- enum [AIN](#) { **AIN1** = 0, **AIN2** = 1 }
Channel indices.
- enum [Modes](#) { **Bipolar** = 0, **Unipolar** = 1 }
Unipolar or bipolar mode.

Public Attributes

- `std::string spiDevice = "/dev/spidev0.0"`
The SPI device used.
- `SamplingRates samplingRate = FS50HZ`
Sampling rate requested.
- `PGAGains pgaGain = G1`
Requested gain.
- `AIN channel = AIN1`
Requested input channel (0 or 1)
- `Modes mode = Unipolar`
Unipolar or bipolar.

3.3.1 Detailed Description

Contains all settings for the data acquisition.

They have default settings for all settings which work.

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

- AD7705Comm.h

Index

- AD7705callback, [5](#)
 - hasSample, [5](#)
- AD7705Comm, [6](#)
 - AD7705Comm, [6](#)
 - registerCallback, [7](#)
- AD7705settings, [7](#)
- hasSample
 - AD7705callback, [5](#)
- registerCallback
 - AD7705Comm, [7](#)