Muscle Power Gaming

version 1.3

Generated by Doxygen 1.8.18

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Module Documentation

6.1 ADS1115: ADDRESS PINS

The addresses when tying the I2C address pin.

Macros

- #define ADS1115_ADDRESS_GND (0x48)
  I2C address pin low (GND).
- #define ADS1115_ADDRESS_VDD (0X49)
  I2C address pin high (VDD).
- #define ADS1115_ADDRESS_SDA (0X4A)
  I2C address pin tied to the SDA line.
- #define ADS1115_ADDRESS_SCL (0X4B)
  I2C address pin tied to the SCL line.

6.1.1 Detailed Description

The addresses when tying the I2C address pin.

The configurations for tying the address pin are;

- I2C address pin low
  - Address pin connected to GND (0X48)
- I2C address pin high
  - Address pin connected to VDD (0X49)
- I2C address pin tied to the SDA line (0X4A)
- I2C address pin tied to the SCL line (0X4B)

6.1.2 Macro Definition Documentation

6.1.2.1 ADS1115_ADDRESS_GND

```
#define ADS1115_ADDRESS_GND (0x48)
```

I2C address pin low (GND).

Definition at line 101 of file ads1115.h.

6.1.2.2 ADS1115_ADDRESS_SCL

```
#define ADS1115_ADDRESS_SCL (0X4B)
```

I2C address pin tied to the SCL line.

Definition at line 104 of file ads1115.h.

6.1.2.3 ADS1115_ADDRESS_SDA

```
#define ADS1115_ADDRESS_SDA (0X4A)
```

I2C address pin tied to the SDA line.

Definition at line 103 of file ads1115.h.

6.1.2.4 ADS1115_ADDRESS_VDD

```
#define ADS1115_ADDRESS_VDD (0X49)
```

I2C address pin high (VDD).

Definition at line 102 of file ads1115.h.

6.2 ADS1115: POINTER REGISTER

The addresses of the registers of the ADS1115.

Macros

- #define ADS1115_REG_POINTER_CONVERT (0x00)

  Select the conversion register.
- #define ADS1115_REG_POINTER_CONFIG (0x01)

  Select the config register.
- #define ADS1115_REG_POINTER_LOWTHRESH (0x02)

  Select the Lo_thresh register.
- #define ADS1115_REG_POINTER_HITHRESH (0x03)

  Select the Lo_thresh register.

6.2.1 Detailed Description

The addresses of the registers of the ADS1115.

The ADS1115 has four registers that are accessible via the I2C port. The Conversion register contains the result of the last conversion. The Config register allows the user the change the ADS1115 operating modes and query the status of the devices. The Lo_thresh and Hi_thresh registers set the threshold values used for the comparator function.

- Conversion register
- Config register
- Lo_thresh register
- Hi_thresh register

6.2.2 Macro Definition Documentation

6.2.2.1 ADS1115_REG_POINTER_CONFIG

```
#define ADS1115_REG_POINTER_CONFIG (0x01)
```

Select the config register.

The 16-bit config register can be used to control the ADS1115 operating mode, input selection, data rate, PGA settings, and comparator modes.

Definition at line 136 of file ads1115.h.

6.2.2.2 ADS1115_REG_POINTER_CONVERT

```
#define ADS1115_REG_POINTER_CONVERT (0x00)
```

Select the conversion register.

The 16-bit conversion register contains the result of the last conversion in binary twos complement format. Following reset or power-up, the Conversion register is cleared to '0', and remains '0' until the first conversion is completed.

Definition at line 130 of file ads1115.h.

6.2.2.3 ADS1115_REG_POINTER_HITHRESH

```
#define ADS1115_REG_POINTER_HITHRESH (0x03)
```

Select the Lo_thresh register.

The upper threshold value used by the comparator are stored in this 16-bit register. The register stores values in the same format that the output register displays values; that is, they are stored in twos complement format. Because it is implemented as a digital comparator, special attention should be taken to readjust values whenever PGA settings are changed.

Definition at line 148 of file ads1115.h.

6.2.2.4 ADS1115_REG_POINTER_LOWTHRESH

```
#define ADS1115_REG_POINTER_LOWTHRESH (0x02)
```

Select the Lo_thresh register.

The lower threshold value used by the comparator are stored in this 16-bit register. The register stores values in the same format that the output register displays values; that is, they are stored in twos complement format. Because it is implemented as a digital comparator, special attention should be taken to readjust values whenever PGA settings are changed.

Definition at line 142 of file ads1115.h.

6.3 ADS1115: OPERATIONAL STATUS

ADS1115 Bit[15].

Macros

- #define ADS1115_REG_CONFIG_OS_SINGLE (0x8000)
  Write: Set to start a single-conversion.
- #define ADS1115_REG_CONFIG_OS_BUSY (0x0000)
  Read: 0 when conversion is in progress.
- #define ADS1115_REG_CONFIG_OS_NOTBUSY (0x8000)
  Read: 1 when device is not performing a conversion.

6.3.1 Detailed Description

ADS1115 Bit[15].

This bit determines the operational status of the device. It can only be written when in power-down mode

6.3.2 Macro Definition Documentation

6.3.2.1 ADS1115_REG_CONFIG_OS_BUSY

```
#define ADS1115_REG_CONFIG_OS_BUSY (0x0000)
```

Read: 0 when conversion is in progress.

Definition at line 167 of file ads1115.h.

6.3.2.2 ADS1115_REG_CONFIG_OS_NOTBUSY

```
#define ADS1115_REG_CONFIG_OS_NOTBUSY (0x8000)
```

Read: 1 when device is not performing a conversion.

Definition at line 168 of file ads1115.h.

6.3.2.3 ADS1115_REG_CONFIG_OS_SINGLE

```
#define ADS1115_REG_CONFIG_OS_SINGLE (0x8000)
```

Write: Set to start a single-conversion.

Definition at line 166 of file ads1115.h.

6.4 ADS1115: INPUT MULTIPLEXER CONFIGURATION

ADS1115 Bit[14-12].

Macros

- #define ADS1115_REG_CONFIG_MUX_DIFF_P0_N1 (0x0000)

  Differential input with positive input from AIN0 and negative input from AIN1 (DEFAULT).
- #define ADS1115_REG_CONFIG_MUX_DIFF_P0_N3 (0x1000)

  Differential input with positive input from AIN0 and negative input from AIN3.
- #define ADS1115_REG_CONFIG_MUX_DIFF_P1_N3 (0x2000)

  Differential input with positive input from AIN1 and negative input from AIN3.
- #define ADS1115_REG_CONFIG_MUX_DIFF_P2_N3 (0x3000)

  Differential input with positive input from AIN2 and negative input from AIN3.
- #define ADS1115_REG_CONFIG_MUX_SINGLE_P0_NG (0x4000)

  Single-ended input with positive input from AIN0 with reference to GND.
- #define ADS1115_REG_CONFIG_MUX_SINGLE_P1_NG (0x5000)

  Single-ended input with positive input from AIN1 with reference to GND.
- #define ADS1115_REG_CONFIG_MUX_SINGLE_P2_NG (0x6000)

  Single-ended input with positive input from AIN2 with reference to GND.
- #define ADS1115_REG_CONFIG_MUX_SINGLE_P3_NG (0x7000)

  Single-ended input with positive input from AIN3 with reference to GND.

6.4.1 Detailed Description

ADS1115 Bit[14-12].

These bits configure the input multiplexer.

- Differential
  - Positive 0:1 Negative
  - Positive 0:3 Negative
  - Positive 1:3 Negative
  - Positive 2:3 Negative.
- Single-ended
  - Positive 0:GND Negative
  - Positive 1:GND Negative
  - Positive 2:GND Negative
  - Positive 3:GND Negative

The DEFAULT setting is a differential input;

- Positive 0:1 Negative

6.4.2 Macro Definition Documentation

6.4.2.1 ADS1115_REG_CONFIG_MUX_DIFF_P0_N1

```
#define ADS1115_REG_CONFIG_MUX_DIFF_P0_N1 (0x0000)
```

Differential input with positive input from AIN0 and negative input from AIN1 (DEFAULT).

Definition at line 198 of file ads1115.h.

6.4.2.2 ADS1115_REG_CONFIG_MUX_DIFF_P0_N3

```
#define ADS1115_REG_CONFIG_MUX_DIFF_P0_N3 (0x1000)
```

Differential input with positive input from AIN0 and negative input from AIN3.

Definition at line 199 of file ads1115.h.

6.4.2.3 ADS1115_REG_CONFIG_MUX_DIFF_P1_N3

```
#define ADS1115_REG_CONFIG_MUX_DIFF_P1_N3 (0x2000)
```

Differential input with positive input from AIN1 and negative input from AIN3.

Definition at line 200 of file ads1115.h.

6.4.2.4 ADS1115_REG_CONFIG_MUX_DIFF_P2_N3

```
#define ADS1115_REG_CONFIG_MUX_DIFF_P2_N3 (0x3000)
```

Differential input with positive input from AIN2 and negative input from AIN3.

Definition at line 201 of file ads1115.h.

6.4.2.5 ADS1115_REG_CONFIG_MUX_SINGLE_P0_NG

```
#define ADS1115_REG_CONFIG_MUX_SINGLE_P0_NG (0x4000)
```

Single-ended input with positive input from AIN0 with reference to GND.

Definition at line 202 of file ads1115.h.

6.4.2.6 ADS1115_REG_CONFIG_MUX_SINGLE_P1_NG

```
#define ADS1115_REG_CONFIG_MUX_SINGLE_P1_NG (0x5000)
```

Single-ended input with positive input from AIN1 with reference to GND.

Definition at line 203 of file ads1115.h.

6.4.2.7 ADS1115_REG_CONFIG_MUX_SINGLE_P2_NG

```
#define ADS1115_REG_CONFIG_MUX_SINGLE_P2_NG (0x6000)
```

Single-ended input with positive input from AIN2 with reference to GND.

Definition at line 204 of file ads1115.h.

6.4.2.8 ADS1115_REG_CONFIG_MUX_SINGLE_P3_NG

```
#define ADS1115_REG_CONFIG_MUX_SINGLE_P3_NG (0x7000)
```

Single-ended input with positive input from AIN3 with reference to GND.

Definition at line 205 of file ads1115.h.

6.5 ADS1115: PROGRAMMABLE GAIN AMPLIFIER CONFIGURATION

ADS1115 Bit[11-9].

Macros

- #define ADS1115_REG_CONFIG_PGA_6_144V (0x0000)
  +/-6.144V range = Gain 2/3
- #define ADS1115_REG_CONFIG_PGA_4_096V (0x0200)
  +/-4.096V range = Gain 1
- #define ADS1115_REG_CONFIG_PGA_2_048V (0x0400)
  +/-2.048V range = Gain 2 (DEFAULT)
- #define ADS1115_REG_CONFIG_PGA_1_024V (0x0600)
  +/-1.024V range = Gain 4
- #define ADS1115_REG_CONFIG_PGA_0_512V (0x0800)
  +/-0.512V range = Gain 8
- #define ADS1115_REG_CONFIG_PGA_0_256V (0x0A00)
  +/-0.256V range = Gain 16

6.5.1 Detailed Description

ADS1115 Bit[11-9].

These bits configure the programmable gain amplifier.

- +-6.144V
- +-4.096V
- +-2.048V
- +-1.024V
- +-0.512V
- +-0.256V

The DEFAULT setting is +-2.048V.

6.5.2 Macro Definition Documentation

6.5.2.1 ADS1115_REG_CONFIG_PGA_0_256V

```
#define ADS1115_REG_CONFIG_PGA_0_256V (0x0A00)
```

+/-0.256V range = Gain 16

Definition at line 232 of file ads1115.h.

6.5.2.2 ADS1115_REG_CONFIG_PGA_0_512V

```
#define ADS1115_REG_CONFIG_PGA_0_512V (0x0800)
```

+/-0.512V range = Gain 8

Definition at line 231 of file ads1115.h.

6.5.2.3 ADS1115_REG_CONFIG_PGA_1_024V

```
#define ADS1115_REG_CONFIG_PGA_1_024V (0x0600)
```

+/-1.024V range = Gain 4

Definition at line 230 of file ads1115.h.

6.5.2.4 ADS1115_REG_CONFIG_PGA_2_048V

```
#define ADS1115_REG_CONFIG_PGA_2_048V (0x0400)
```

+/-2.048V range = Gain 2 (DEFAULT)

Definition at line 229 of file ads1115.h.

6.5.2.5 ADS1115_REG_CONFIG_PGA_4_096V

```
#define ADS1115_REG_CONFIG_PGA_4_096V (0x0200)
```

+/-4.096V range = Gain 1

Definition at line 228 of file ads1115.h.

6.5.2.6 ADS1115_REG_CONFIG_PGA_6_144V

```
#define ADS1115_REG_CONFIG_PGA_6_144V (0x0000)
```

+/-6.144V range = Gain 2/3

Definition at line 227 of file ads1115.h.

6.6 ADS1115: DEVICE OPERATING MODE

ADS1115 Bit[8].

Macros

- #define ADS1115_REG_CONFIG_MODE_CONTIN (0x0000)
  Continuous conversion mode.
- #define ADS1115_REG_CONFIG_MODE_ENDCON (0x8583)
  Shut down continuous conversion.
- #define ADS1115_REG_CONFIG_MODE_SINGLE (0x0100)
  Power-down single-shot mode (DEFAULT)

6.6.1 Detailed Description

ADS1115 Bit[8].

This bit controls the operational mode of the ADS1115.

The DEFAULT setting is power-down single-shot mode.

6.6.2 Macro Definition Documentation

6.6.2.1 ADS1115_REG_CONFIG_MODE_CONTIN

```
#define ADS1115_REG_CONFIG_MODE_CONTIN (0x0000)
```

Continuous conversion mode.

Definition at line 248 of file ads1115.h.

6.6.2.2 ADS1115_REG_CONFIG_MODE_ENDCON

```
#define ADS1115_REG_CONFIG_MODE_ENDCON (0x8583)
```

Shut down continuous conversion.

Definition at line 249 of file ads1115.h.

6.6.2.3 ADS1115_REG_CONFIG_MODE_SINGLE

```
#define ADS1115_REG_CONFIG_MODE_SINGLE (0x0100)
```

Power-down single-shot mode (DEFAULT)

Definition at line 250 of file ads1115.h.

6.7 ADS1115: DATA RATE

ADS1115 Bit[7-5].

Macros

- #define ADS1115_RATE_008 (0x0000)
  8 samples per second
- #define ADS1115_RATE_016 (0x0020)
  16 samples per second
- #define ADS1115_RATE_032 (0x0040)
  32 samples per second
- #define ADS1115_RATE_064 (0x0060)
  64 samples per second
- #define ADS1115_RATE_128 (0x0080)
  128 samples per second
- #define ADS1115_RATE_250 (0x00A0)
  250 samples per second (DEFAULT)
- #define ADS1115_RATE_475 (0x00C0)
  475 samples per second
- #define ADS1115_RATE_860 (0x00E0)
  860 samples per second

6.7.1 Detailed Description

ADS1115 Bit[7-5].

These bits control the data rate setting.

- 8 sps
- 16 sps
- 32 sps
- 64 sps
- 128 sps
- 250 sps
- 475 sps
- 860 sps

The DEFAULT data rate is 250 samples per second.

6.7.2 Macro Definition Documentation

6.7.2.1 ADS1115_RATE_008

```
#define ADS1115_RATE_008 (0x0000)
```

8 samples per second

Definition at line 273 of file ads1115.h.

6.7.2.2 ADS1115_RATE_016

```
#define ADS1115_RATE_016 (0x0020)
```

16 samples per second

Definition at line 274 of file ads1115.h.

6.7.2.3 ADS1115_RATE_032

```
#define ADS1115_RATE_032 (0x0040)
```

32 samples per second

Definition at line 275 of file ads1115.h.

6.7.2.4 ADS1115_RATE_064

```
#define ADS1115_RATE_064 (0x0060)
```

64 samples per second

Definition at line 276 of file ads1115.h.

6.7.2.5 ADS1115_RATE_128

```
#define ADS1115_RATE_128 (0x0080)
```

128 samples per second

Definition at line 277 of file ads1115.h.

6.7.2.6 ADS1115_RATE_250

```
#define ADS1115_RATE_250 (0x00A0)
```

250 samples per second (DEFAULT)

Definition at line 278 of file ads1115.h.

6.7.2.7 ADS1115_RATE_475

```
#define ADS1115_RATE_475 (0x00C0)
```

475 samples per second

Definition at line 279 of file ads1115.h.

6.7.2.8 ADS1115_RATE_860

```
#define ADS1115_RATE_860 (0x00E0)
```

860 samples per second

Definition at line 280 of file ads1115.h.

6.8 ADS1115: COMPARATOR MODE

ADS1115 Bit[4].

Macros

- #define ADS1115_COMP_MODE_HYSTERESIS (0x0000)

  Traditional comparator with hysteresis (DEFAULT)
- #define ADS1115_COMP_MODE_WINDOW (0x0010)

  Window comparator.

6.8.1 Detailed Description

ADS1115 Bit[4].

This bit controls the comparator mode of operation. It changes whether the comparator is implemented as a traditional comparator or as a window comparator.

The DEFAULT setting is the window comparator

6.8.2 Macro Definition Documentation

6.8.2.1 ADS1115_COMP_MODE_HYSTERESIS

```
#define ADS1115_COMP_MODE_HYSTERESIS (0x0000)
```

Traditional comparator with hysteresis (DEFAULT)

Definition at line 296 of file ads1115.h.

6.8.2.2 ADS1115_COMP_MODE_WINDOW

```
#define ADS1115_COMP_MODE_WINDOW (0x0010)
```

Window comparator.

Definition at line 297 of file ads1115.h.

6.9 ADS1115: COMPARATOR POLARITY

ADS1115 Bit[3].

Macros

- #define ADS1115_COMP_POL_ACTIVE_LOW (0x0000)
  ALERT/RDY pin is active low (DEFAULT)
- #define ADS1115_COMP_POL_ACTIVE_HIGH (0x0008)
  ALERT/RDY pin is active high.

6.9.1 Detailed Description

ADS1115 Bit[3].

This bit controls the polarity of the ALERT/RDY pin.

The DEFAULT setting holds the ALERT/RDY pin active low

6.9.2 Macro Definition Documentation

6.9.2.1 ADS1115_COMP_POL_ACTIVE_HIGH

```
#define ADS1115_COMP_POL_ACTIVE_HIGH (0x0008)
```

ALERT/RDY pin is active high.

Definition at line 314 of file ads1115.h.

6.9.2.2 ADS1115_COMP_POL_ACTIVE_LOW

```
#define ADS1115_COMP_POL_ACTIVE_LOW (0x0000)
```

ALERT/RDY pin is active low (DEFAULT)

Definition at line 313 of file ads1115.h.

6.10 ADS1115: LATCHING COMPARATOR

ADS1115 Bit[2].

Macros

- #define ADS1115_COMP_LAT_NON_LATCHING (0x0000)

  Non-latching comparator (DEFAULT)
- #define ADS1115_COMP_LAT_LATCHING (0x0004)

  Latching comparator.

6.10.1 Detailed Description

ADS1115 Bit[2].

This bit controls whether the ALERT/RDY pin latches once asserted or clears once conversion are within the margin of the upper and lower threshold values. When COMP_LAT = '0', the ALERT/RDY pin does not latch when asserted. When COMP_LAT = '1', the asserted ALERT/RDY pin remains latched until conversion data are read by the master or an appropriate SMBus alert response is sent by the master, the device responds with its address, and it is the lowest address currently asserting the ALERT/RDY bus line.

The DEFAULT setting holds the ALERT/RDY pin active low

6.10.2 Macro Definition Documentation

6.10.2.1 ADS1115_COMP_LAT_LATCHING

```
#define ADS1115_COMP_LAT_LATCHING (0x0004)
```

Latching comparator.

Definition at line 330 of file ads1115.h.

6.10.2.2 ADS1115_COMP_LAT_NON_LATCHING

```
#define ADS1115_COMP_LAT_NON_LATCHING (0x0000)
```

Non-latching comparator (DEFAULT)

Definition at line 329 of file ads1115.h.

6.11 ADS1115: COMPARATOR QUEUE AND DISABLE

ADS1115 Bit[1-0].

Macros

- #define ADS1115_COMP_QUE_ASSERT1 (0x0000)
  Assert after one conversion.
- #define ADS1115_COMP_QUE_ASSERT2 (0x0001)
  Assert after two conversions.
- #define ADS1115_COMP_QUE_ASSERT4 (0x0002)
  Assert after four conversions.
- #define ADS1115_COMP_QUE_DISABLE (0x0003)
  Disable comparator (DEFAULT)
- #define ADS1115_REG_THRESH_MSB_1 (0x8000)
  Set MSB to 1.
- #define ADS1115_REG_THRESH_MSB_0 (0x7FFF)
  Set MSB to 0.

6.11.1 Detailed Description

ADS1115 Bit[1-0].

Dictates the MSB.

These bits perform two functions. When set to '11', they disable the comparator function and put the ALERT/RDY pin into a high state. When set to any other value, they control the number of successive conversions exceeding the upper or lower thresholds required before asserting the ALERT/RDY pin.

The DEFAULT setting disables the comparator

This bit controls whether the ALERT/RDY pin latches once asserted or clears once conversion are within the margin of the upper and lower threshold values. When COMP_LAT = '0', the ALERT/RDY pin does not latch when asserted. When COMP_LAT = '1', the asserted ALERT/RDY pin remains latched until conversion data are read by the master or an appropriate SMBus alert response is sent by the master, the device responds with its address, and it is the lowest address currently asserting the ALERT/RDY bus line.

The DEFAULT setting holds the ALERT/RDY pin active low

6.11.2 Macro Definition Documentation

6.11.2.1 ADS1115_COMP_QUE_ASSERT1

```
#define ADS1115_COMP_QUE_ASSERT1 (0x0000)
```

Assert after one conversion.

Definition at line 345 of file ads1115.h.

6.11.2.2 ADS1115_COMP_QUE_ASSERT2

```
#define ADS1115_COMP_QUE_ASSERT2 (0x0001)
```

Assert after two conversions.

Definition at line 346 of file ads1115.h.

6.11.2.3 ADS1115_COMP_QUE_ASSERT4

```
#define ADS1115_COMP_QUE_ASSERT4 (0x0002)
```

Assert after four conversions.

Definition at line 347 of file ads1115.h.

6.11.2.4 ADS1115_COMP_QUE_DISABLE

```
#define ADS1115_COMP_QUE_DISABLE (0x0003)
```

Disable comparator (DEFAULT)

Definition at line 348 of file ads1115.h.

6.11.2.5 ADS1115_REG_THRESH_MSB_0

```
#define ADS1115_REG_THRESH_MSB_0 (0x7FFF)
```

Set MSB to 0.

Definition at line 367 of file ads1115.h.

6.11.2.6 ADS1115_REG_THRESH_MSB_1

```
#define ADS1115_REG_THRESH_MSB_1 (0x8000)
```

Set MSB to 1.

Definition at line 366 of file ads1115.h.

Chapter 7

Namespace Documentation

7.1 Ui Namespace Reference

Chapter 8

Class Documentation

8.1 ads1115 Class Reference

Allows control over the ADS1115 ADC.

`#include <ads1115.h>`

Inheritance diagram for ads1115:

Collaboration diagram for ads1115:

Signals

- void readyread (float voltage)

  Conveys the calculated EMG voltage.

Public Member Functions

- ads1115 (uchar addr, QObject *parent=0)

  Configures the ADS1115 into continuous conversion mode.
- void endads ()

  Attempts to exit the continuous mode of the ADS1115.
- float readsig ()

  Performs conversion of input EMG data to a voltage result.

Public Attributes

- int fd

  A reference device identifier obtained using "wiringPiI2CSetup()".
- uchar iicaddr

  A reference to the I2C address pin.

8.1.1 Detailed Description

Allows control over the ADS1115 ADC.

Handles all aspects of controlling the ADS1115 device. Responsible for configuring the device into continuous mode, gathering, handling and relaying the signals from input channels, and self abortion.

Definition at line 378 of file ads1115.h.

8.1.2 Constructor & Destructor Documentation

8.1.2.1 ads1115()

```
ads1115::ads1115 (
            uchar addr,
            QObject * parent = 0 )  [explicit]
```

Configures the ADS1115 into continuous conversion mode.

Parameters

uchar	addr The I2C address of the device (see address pin macros)

Uses the I2C address for device identification. The appropriate configuration options are applied to instantiate continuous conversion and confirms operation through use of the wiringPiI2C library

Definition at line 66 of file ads1115.cpp.

8.1.3 Member Function Documentation

8.1.3.1 endads()

```
void ads1115::endads ( )
```

Attempts to exit the continuous mode of the ADS1115.

Updates the "config" variable to change the CONFIG_MODE setting to END_CON. The "load_config()" function then converts from a hexadecimal to binary representation. The "rcr" variable resets to 0 (success) before being set to the returned value from the "wiringPiI2CWriteReg16() function. A return value of -1 (non-zero) indicates a failure to exit continuous conversion mode.

Parameters

void	

Definition at line 187 of file ads1115.cpp.

8.1.3.2 readsig()

```
float ads1115::readsig ( )
```

Performs conversion of input EMG data to a voltage result.

Selects the conversion register and uses the "WiringPiI2CReadReg16()" function to read the data contained therein. The data is then processed before being emitted through the readyrread() signal.

WiringPi doesn't include stdint so values are int32. Conversions are performed to account for this in producing the voltage output

Parameters

void	

Returns

The read voltage converted from the raw signal input

Definition at line 153 of file ads1115.cpp.

8.1.3.3 readyread

```
void ads1115::readyread (
             float voltage )  [signal]
```

Conveys the calculated EMG voltage.

After conversion from a raw signal to a voltage by function ads1115(), this signal then conveys the result.

Parameters

voltage	The calculated voltage from processed EMG signal

8.1.4 Member Data Documentation

8.1.4.1 fd

```
int ads1115::fd
```

A reference device identifier obtained using "wiringPiI2CSetup()".

"wiringPiI2CSetup(int devID)" initialises the I2C system with the given device identifier (I2C number of the device). The I2C number is one of the address pin macros found in ads1115.h. The return value, stored in fd, is -1 if any error occurs, or the standard Linux filehandle otherwise.

Definition at line 398 of file ads1115.h.

8.1.4.2 iicaddr

```
uchar ads1115::iicaddr
```

A reference to the I2C address pin.

The "wiringPiI2CSetup()" function returns a reference device ID. This address has to then be used when writing and reading to the device registers using the wiringPi library. iicaddr is an explicit reference to the I2C address pin of the device (0x48, 0x49, 0x4A, 0x4B).

Definition at line 406 of file ads1115.h.

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

- /home/finlay/RTEP1/Rpi_end/200322_Rpi_end_together/ads1115.h
- /home/finlay/RTEP1/Rpi_end/200322_Rpi_end_together/ads1115.cpp

8.2 cmdmtx Class Reference

Allows multiple threads to share the same resource.

8.2.1 Detailed Description

Allows multiple threads to share the same resource.

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

- /home/finlay/RTEP1/Rpi_end/200322_Rpi_end_together/GPIOlis.cpp

8.3 GPIOlisten Class Reference

Is responsible for the data communications and threading involved with conversion and processing.

```
#include <GPIOlis.h>
```

Inheritance diagram for GPIOlisten:

Collaboration diagram for GPIOlisten:

Signals

- void ready ()
  Stuff.
- void readyread (float voltage)
  Details.

Public Member Functions

- GPIOlisten (QObject *parent=0)
  Initialises a thread.
- ~GPIOlisten ()
- void interrupt (void)
- void quit () void quit()
  Exit continuous conversion mode and delete QThread.

Public Attributes

- int count
  A simple counter.
- int flag =1
  A marker relating to the state of the interrupt.

Protected Member Functions

- void run ()
  Run.

Private Attributes

- ads1115 * ads1
  Stuff.

8.3.1 Detailed Description

Is responsible for the data communications and threading involved with conversion and processing.

The GPIOlisten class monitors the Alert pin and triggers an interrupt each time an AD conversion is ready. The data is then processed before sleeping until the next interrupt.

Definition at line 28 of file GPIOlis.h.

8.3.2 Constructor & Destructor Documentation

8.3.2.1 GPIOlisten()

```
GPIOlisten::GPIOlisten (
            QObject * parent = 0 )
```

Initialises a thread.

< A new instance of the class ads1115

Definition at line 46 of file GPIOlis.cpp.

8.3.2.2 ~GPIOlisten()

```
GPIOlisten::~GPIOlisten ( )
```

8.3.3 Member Function Documentation

8.3.3.1 interrupt()

```
void GPIOlisten::interrupt (
            void  )
```

8.3.3.2 quit()

```
void GPIOlisten::quit ( )
```

Exit continuous conversion mode and delete QThread.

"endads()" found in "ads1115.h" exits the continuous conversion mode of the ADS1115. The instance of ads1115 is then deleted and the QThread deleted.

Stuff

Details

8.3.3.3 ready

```
void GPIOlisten::ready ( )  [signal]
```

Stuff.

Details

8.3.3.4 readyread

```
void GPIOlisten::readyread (
            float voltage )  [signal]
```

Details.

8.3.3.5 run()

```
void GPIOlisten::run ( )  [protected]
```

Run.

Details

8.3.4 Member Data Documentation

8.3.4.1 ads1

```
ads1115* GPIOlisten::ads1  [private]
```

Stuff.

Details

Definition at line 67 of file GPIOlis.h.

8.3.4.2 count

```
int GPIOlisten::count
```

A simple counter.

Definition at line 40 of file GPIOlis.h.

8.3.4.3 flag

```
int GPIOlisten::flag =1
```

A marker relating to the state of the interrupt.

Definition at line 94 of file GPIOlis.h.

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

- /home/finlay/RTEP1/Rpi_end/200322_Rpi_end_together/GPIOlis.h
- /home/finlay/RTEP1/Rpi_end/200322_Rpi_end_together/GPIOlis.cpp

8.4 load_config Class Reference

Converts "config" from hexadecimal to binary and returns the result.

8.4.1 Detailed Description

Converts "config" from hexadecimal to binary and returns the result.

Takes the hexadecimal format of the ADC configuration described in "config" and converts it to a binary representation.

Parameters

config	The hexadecimal representation of the ADS1115 configuration.

Returns

Overwrites config to produce the binary representation of the ADS1115 configuration.

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

- /home/finlay/RTEP1/Rpi_end/200322_Rpi_end_together/ads1115.cpp

8.5 MainWindow Class Reference

```
#include <mainwindow.h>
```

Inheritance diagram for MainWindow:

Collaboration diagram for MainWindow:

Public Slots

- void refreshScore (int count)

  Calculate the score of the game.
- void Position ()

  Calculate positions of the paddles and ball.
- void receive ()

  Receive UDP signal.

Signals

- void goal (int player)
- void rfsh ()

Public Member Functions

- MainWindow (int scrnwidth, int scrnheight, QWidget *parent=0)

  Unknown.
- ~MainWindow ()

  Deletion of the GUI window.
- qreal CpuP1Motion ()

  Control of Paddle 1 by the CPU.

Public Attributes

- Ui::MainWindow * ui
- int iScore

  A counter variable to monitor game score.
- QGraphicsScene * iScene

  Provides a surface for managing a number of 2D items.
- QGraphicsRectItem * iP2

  Generate the paddle for the Player to control with EMG input.
- QGraphicsRectItem * iP1

  Generate the paddles for the CPU.
- QGraphicsEllipseItem * iBall

  Generate the ball.
- QTimer * iTimer
- QElapsedTimer timer_measure
- QPointF iBallMotion

  Describes the motion of the ball.
- qreal iP2Motion

  Describes the motion of paddle 2 in the X direction.
- qreal iP1Motion

  Describes the motion of paddle 1 in the X direction.
- qreal Xprime

  The X position of the ball.
- qreal Yprime

  The Y position of the ball.
- qreal P2Xprime

  The position of the Player paddle.
- qreal P1Xprime

  The position of the CPU paddle.
- int rfshcount =0

  Refresh the counter.

Private Attributes

- QUdpSocket * rsverSocket
- int wdwidth

  The GUI window width, smaller than screen width/.
- int wdheight

  The GUI window height, smaller than screen width.
- int totalHeight = 350
- int totalWidth = 320

8.5.1 Detailed Description

Definition at line 55 of file mainwindow.h.

8.5.2 Constructor & Destructor Documentation

8.5.2.1 MainWindow()

```
MainWindow::MainWindow (
            int scrnwidth,
            int scrnheight,
            QWidget * parent = 0 )  [explicit]
```

Unknown.

Parameters

scrnwidth	The width of the screen
scrnheight	The height of the screen

Responsible for the GUI and presenting the game to the user.

The `mainwindow` is the code used to a) visualise the pong game, b) communicate with android host via udp

Template Parameters

T	scalar type for real and imaginary components

Parameters

scrnwidth	
scrnheight	

The ball is set to an initial coordinate position (-2,-2)

Definition at line 36 of file mainwindow.cpp.

8.5.2.2 ~MainWindow()

```
MainWindow::~MainWindow ( )
```

Deletion of the GUI window.

Parameters

void	The ~MainWindow() function deletes the paddles (iP1 and iP2), the ball (iBall), the scene (iscene), rserverSocket, iTimer, and UI itself.

Definition at line 112 of file mainwindow.cpp.

8.5.3 Member Function Documentation

8.5.3.1 CpuP1Motion()

```
qreal MainWindow::CpuP1Motion ( )
```

Control of Paddle 1 by the CPU.

Details

Definition at line 203 of file mainwindow.cpp.

8.5.3.2 goal

```
void MainWindow::goal (
            int player )  [signal]
```

8.5.3.3 Position

```
void MainWindow::Position ( )  [slot]
```

Calculate positions of the paddles and ball.

Definition at line 135 of file mainwindow.cpp.

8.5.3.4 receive

```
void MainWindow::receive ( )  [slot]
```

Receive UDP signal.

< Store the number of channels

< Difference between output values to control the position of the user controlled paddle

Definition at line 237 of file mainwindow.cpp.

8.5.3.5 refreshScore

```
void MainWindow::refreshScore (
            int count )  [slot]
```

Calculate the score of the game.

Parameters

int	count A counter which stores the score up to the function call

Upon calling the refreshscore() function, the score at the time of function call is incremented and updated in the member "iscore". The lcdNumber of the UI then dislays this value Renew Score

Definition at line 125 of file mainwindow.cpp.

8.5.3.6 rfsh

```
void MainWindow::rfsh ( )  [signal]
```

8.5.4 Member Data Documentation

8.5.4.1 iBall

```
QGraphicsEllipseItem* MainWindow::iBall
```

Generate the ball.

QGraphicsEllipseItem generates an ellipse item to add to a QGraphicsScene. This acts as the ball for the game.

setBrush() sets the item's brush to 'brush', which is used to fill the item. The brush can then be set using QBrush

moveBy() is inherited from QGraphicsItem, and moves the item by dx points horizontally, and dy points vertically.

Definition at line 120 of file mainwindow.h.

8.5.4.2 iBallMotion

```
QPointF MainWindow::iBallMotion
```

Describes the motion of the ball.

A point is specified by an X coordinate and a Y coordinate which can be accessed using the X() and Y() functions. The coordinates of the point are specified using floating point numbers for accuracy. The isNull() function returns true if both X and Y values are set to 0.0. The coordinates can be set using the setX() and setY() functions.

Definition at line 135 of file mainwindow.h.

8.5.4.3 iP1

```
QGraphicsRectItem* MainWindow::iP1
```

Generate the paddles for the CPU.

iP1 is a member of the QGraphicsRecItem class. QGraphicsRectItem generates a rectangular item to add to a QGraphicsScene. QGraphicsRectItem is constructed with a default ractangle, and the given "width" and "height". This member is the paddle for the CPU.

Definition at line 109 of file mainwindow.h.

8.5.4.4 iP1Motion

```
qreal MainWindow::iP1Motion
```

Describes the motion of paddle 1 in the X direction.

Definition at line 139 of file mainwindow.h.

8.5.4.5 iP2

```
QGraphicsRectItem* MainWindow::iP2
```

Generate the paddle for the Player to control with EMG input.

iP2 is a member of the QGraphicsRecItem class. QGraphicsRectItem generates a rectangular item to add to a QGraphicsScene. This member is the paddle for the Player to control with EMG input.

Definition at line 102 of file mainwindow.h.

8.5.4.6 iP2Motion

```
qreal MainWindow::iP2Motion
```

Describes the motion of paddle 2 in the X direction.

Definition at line 137 of file mainwindow.h.

8.5.4.7 iScene

```
QGraphicsScene* MainWindow::iScene
```

Provides a surface for managing a number of 2D items.

The class serves as a container for QGraphicsItems. It is used together with QGraphicsView for visualising graphical items such as lines and rectangles on a 2D surface.

Items can be added to the QGraphicsScene by calling addItem() and retursn a pointer to the newly added item. The width() and height() functions can be used to declare the position of a QItem. QGraphicsView can then be used to visualize the scene.

The scene rectangle defines the extent of the scene. It is primarily used by QGraphicsView to determine the view's default scrollable area, and by QGraphicsScene to manage item indexing. setSceneRect(qreal x, qreal y, qreal w, qreal h).

Definition at line 95 of file mainwindow.h.

8.5.4.8 iScore

```
int MainWindow::iScore
```

A counter variable to monitor game score.

Definition at line 84 of file mainwindow.h.

8.5.4.9 iTimer

```
QTimer* MainWindow::iTimer
```

Definition at line 124 of file mainwindow.h.

8.5.4.10 P1Xprime

```
qreal MainWindow::P1Xprime
```

The position of the CPU paddle.

Definition at line 147 of file mainwindow.h.

8.5.4.11 P2Xprime

```
qreal MainWindow::P2Xprime
```

The position of the Player paddle.

Definition at line 145 of file mainwindow.h.

8.5.4.12 rfshcount

```
int MainWindow::rfshcount =0
```

Refresh the counter.

Definition at line 152 of file mainwindow.h.

8.5.4.13 rsverSocket

```
QUdpSocket* MainWindow::rsverSocket  [private]
```

Definition at line 178 of file mainwindow.h.

8.5.4.14 timer_measure

```
QElapsedTimer MainWindow::timer_measure
```

Definition at line 125 of file mainwindow.h.

8.5.4.15 totalHeight

```
int MainWindow::totalHeight = 350  [private]
```

Definition at line 181 of file mainwindow.h.

8.5.4.16 totalWidth

```
int MainWindow::totalWidth = 320  [private]
```

Definition at line 182 of file mainwindow.h.

8.5.4.17 ui

```
Ui::MainWindow* MainWindow::ui
```

Definition at line 81 of file mainwindow.h.

8.5.4.18 wdheight

```
int MainWindow::wdheight  [private]
```

The GUI window height, smaller than screen width.

Definition at line 180 of file mainwindow.h.

8.5.4.19 wdwidth

```
int MainWindow::wdwidth  [private]
```

The GUI window width, smaller than screen width/.

Definition at line 179 of file mainwindow.h.

8.5.4.20 Xprime

```
qreal MainWindow::Xprime
```

The X position of the ball.

Definition at line 141 of file mainwindow.h.

8.5.4.21 Yprime

```
qreal MainWindow::Yprime
```

The Y position of the ball.

Definition at line 143 of file mainwindow.h.

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

- /home/finlay/RTEP1/Server_end/0301_Pong_GUI/mainwindow.h
- /home/finlay/RTEP1/Server_end/0301_Pong_GUI/mainwindow.cpp

8.6 Mainwindow Class Reference

Receive UDP signal and move GUI accordingly.

```
#include <mainwindow.h>
```

8.6.1 Detailed Description

Receive UDP signal and move GUI accordingly.

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

- /home/finlay/RTEP1/Server_end/0301_Pong_GUI/mainwindow.h

8.7 plot1 Class Reference

8.7.1 Detailed Description

Definition at line 63 of file window.cpp.

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

- /home/finlay/RTEP1/Rpi_end/200322_Rpi_end_together/window.cpp

8.8 QGraphicsItem Class Reference

Unknown.

```
#include <mainwindow.h>
```

8.8.1 Detailed Description

Unknown.

Details

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

- /home/finlay/RTEP1/Server_end/0301_Pong_GUI/mainwindow.h

8.9 QTimer Class Reference

Unknown.

`#include <window.h>`

8.9.1 Detailed Description

Unknown.

Details

Define an ADS1115 device with an I2C address (0x48, 0x49, 0x4A, 0x4B)

Definition at line 44 of file window.h.

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

- /home/finlay/RTEP1/Rpi_end/200322_Rpi_end_together/window.h

8.10 receive Class Reference

Receives and processes the EMMG signal through the UDP.

8.10.1 Detailed Description

Receives and processes the EMMG signal through the UDP.

Parameters

void	

Returns

void

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

- /home/finlay/RTEP1/Server_end/0301_Pong_GUI/mainwindow.cpp

8.11 Window Class Reference

Unknown.

`#include <window.h>`

Inheritance diagram for Window:

Collaboration diagram for Window:

Public Slots

- void datapros (float)
- void plotrefresh ()

Public Member Functions

- Window (QWidget *parent=0)

  Ddefault constructor - called when a Window is declared without arguments.
- ~Window ()

  Virtual.

Public Attributes

- QUdpSocket * sdersc

  An instance of the QUdpSocket from <QUdpSocket>
- GPIOlisten * gpiolis1
- QTimer * rftimer

Private Attributes

- QwtPlot * plot1

  Qt widget: First plot.
- QwtPlot * plot2

  QT widget: Second plot.
- QwtPlotCurve * curve1

  QT widget: First curve.
- QwtPlotCurve * curve2

  QT widget: Second curve.
- QHBoxLayout * hLayout
- double xData1 [plotDataSize]

  Data array for xData1.
- double yData1 [plotDataSize]

  Data array for yData1.
- double xData2 [plotDataSize]

  Data array for xData2.
- double yData2 [plotDataSize]

  Data array for yData2.
- double xData3 [plotDataSize]
- double yData3 [plotDataSize]

  Data array for yData3.
- double sumpower

Static Private Attributes

- static const int plotDataSize = 5

8.11.1 Detailed Description

Unknown.

The Q_OBJECT macro is needed for the Qt signal/slot framework to operate within the class

Definition at line 66 of file window.h.

8.11.2 Constructor & Destructor Documentation

8.11.2.1 Window()

```
Window::Window (
            QWidget * parent = 0 )  [explicit]
```

Ddefault constructor - called when a Window is declared without arguments.

8.11.2.2 ~Window()

```
Window::~Window ( )
```

Virtual.

Definition at line 119 of file window.cpp.

8.11.3 Member Function Documentation

8.11.3.1 datapros

```
void Window::datapros (
             float inval )  [slot]
```

Definition at line 142 of file window.cpp.

8.11.3.2 plotrefresh

```
void Window::plotrefresh ( )  [slot]
```

Definition at line 194 of file window.cpp.

8.11.4 Member Data Documentation

8.11.4.1 curve1

```
QwtPlotCurve* Window::curve1  [private]
```

QT widget: First curve.

Definition at line 92 of file window.h.

8.11.4.2 curve2

```
QwtPlotCurve* Window::curve2  [private]
```

QT widget: Second curve.

Definition at line 93 of file window.h.

8.11.4.3 gpiolis1

```
GPIOlisten* Window::gpiolis1
```

Definition at line 78 of file window.h.

8.11.4.4 hLayout

```
QHBoxLayout* Window::hLayout  [private]
```

Definition at line 95 of file window.h.

8.11.4.5 plot1

```
QwtPlot* Window::plot1  [private]
```

Qt widget: First plot.

Definition at line 90 of file window.h.

8.11.4.6 plot2

```
QwtPlot* Window::plot2  [private]
```

QT widget: Second plot.

Definition at line 91 of file window.h.

8.11.4.7 plotDataSize

```
const int Window::plotDataSize = 5  [static], [private]
```

Definition at line 96 of file window.h.

8.11.4.8 rftimer

```
QTimer* Window::rftimer
```

Definition at line 80 of file window.h.

8.11.4.9 sdersc

```
QUdpSocket* Window::sdersc
```

An instance of the QUdpSocket from <QUdpSocket>

Definition at line 72 of file window.h.

8.11.4.10 sumpower

```
double Window::sumpower  [private]
```

Definition at line 105 of file window.h.

8.11.4.11 xData1

```
double Window::xData1[plotDataSize]  [private]
```

Data array for xData1.

Definition at line 99 of file window.h.

8.11.4.12 xData2

```
double Window::xData2[plotDataSize]  [private]
```

Data array for xData2.

Definition at line 101 of file window.h.

8.11.4.13 xData3

```
double Window::xData3[plotDataSize]  [private]
```

Definition at line 103 of file window.h.

8.11.4.14 yData1

```
double Window::yData1[plotDataSize]  [private]
```

Data array for yData1.

Definition at line 100 of file window.h.

8.11.4.15 yData2

```
double Window::yData2[plotDataSize]  [private]
```

Data array for yData2.

Definition at line 102 of file window.h.

8.11.4.16 yData3

```
double Window::yData3[plotDataSize]  [private]
```

Data array for yData3.

Definition at line 104 of file window.h.

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

- /home/finlay/RTEP1/Rpi_end/200322_Rpi_end_together/window.h
- /home/finlay/RTEP1/Rpi_end/200322_Rpi_end_together/window.cpp

Chapter 9

File Documentation

9.1 /home/finlay/RTEP1/Rpi_end/200322_Rpi_end_together/ads1115.cpp File Reference

Sets the ADS1115 to the correct configurations and operates accordingly.

```
#include <wiringPi.h>
#include <wiringPiI2C.h>
#include <QElapsedTimer>
#include <QDebug>
#include <QApplication>
#include "ads1115.h"
```

Include dependency graph for ads1115.cpp:

Functions

- int load_config (int config)

Variables

- int low

  A variable used in conversion from hexadecimal to binary forms.
- int high

  A variable used in conversion from hexadecimal to binary forms.
- int config

  Stores the 16-bit config register data for the ADS1115.
- int high_config

  Sets the low_threshold register.
- int low_config

  Sets the high_threshold register.
- float voltage
- int rcr

  A boolean variable to evaluate the success of writing to a register.
- int value

  Stores the contents of the conversion register of the designated device.

9.1.1 Detailed Description

Sets the ADS1115 to the correct configurations and operates accordingly.

The configuration options are outlined in the header file ads1115.h, and the selected macros for this purpose set the ADC to continuous coversion mode, a data rate of 860 SPS, a volatge range of +-4.096V, single-ended input from AIN0, and disables the comparator.

Author

Zonghan Gan

Finlay Nelson

Henry Cowan

9.1.2 Function Documentation

9.1.2.1 load_config()

```
int load_config (
            int config )
```

Definition at line 57 of file ads1115.cpp.

9.1.3 Variable Documentation

9.1.3.1 config

```
int config
```

Stores the 16-bit config register data for the ADS1115.

Definition at line 25 of file ads1115.cpp.

9.1.3.2 high

```
int high
```

A variable used in conversion from hexadecimal to binary forms.

Definition at line 24 of file ads1115.cpp.

9.1.3.3 high_config

```
int high_config
```

Sets the low_threshold register.

Definition at line 27 of file ads1115.cpp.

9.1.3.4 low

```
int low
```

A variable used in conversion from hexadecimal to binary forms.

Definition at line 23 of file ads1115.cpp.

9.1.3.5 low_config

```
int low_config
```

Sets the high_threshold register.

Definition at line 28 of file ads1115.cpp.

9.1.3.6 rcr

```
int rcr
```

A boolean variable to evaluate the success of writing to a register.

"wiringPiI2CWriteReg16(int fd, int reg, int data)" writes a 16-bit data value into the device register indicated.

"rcr" stores the output of the "wiringPiI2CWriteReg16" function. If the result is 0, the set-up procedure has been successful, whilst if the value is non-zero the write has failed.

Definition at line 39 of file ads1115.cpp.

9.1.3.7 value

```
int value
```

Stores the contents of the conversion register of the designated device.

"wiringPiI2CReadReg16" reads the 16-bit coversion register associated with the device address, fd. The register stores the last conversion made, in binary twos complement format.

Definition at line 46 of file ads1115.cpp.

9.1.3.8 voltage

```
float voltage
```

Definition at line 30 of file ads1115.cpp.

9.2 /home/finlay/RTEP1/Rpi_end/200322_Rpi_end_together/ads1115.h File Reference

Defines a set of macros for use in configuring the ADS1115 and a class for controlling the configured device.

```
#include <QObject>
#include <iostream>
#include <string>
```

Include dependency graph for ads1115.h: This graph shows which files directly or indirectly include this file:

Classes

- class ads1115

  Allows control over the ADS1115 ADC.

Macros

- #define ADS1115_ADDRESS_GND (0x48)

  I2C address pin low (GND).
- #define ADS1115_ADDRESS_VDD (0X49)

  I2C address pin high (VDD).
- #define ADS1115_ADDRESS_SDA (0X4A)

  I2C address pin tied to the SDA line.
- #define ADS1115_ADDRESS_SCL (0X4B)

  I2C address pin tied to the SCL line.
- #define ADS1115_REG_POINTER_CONVERT (0x00)

  Select the conversion register.
- #define ADS1115_REG_POINTER_CONFIG (0x01)

  Select the config register.

- #define ADS1115_REG_POINTER_LOWTHRESH (0x02)

  Select the Lo_thresh register.
- #define ADS1115_REG_POINTER_HITHRESH (0x03)

  Select the Lo_thresh register.
- #define ADS1115_REG_CONFIG_OS_SINGLE (0x8000)

  Write: Set to start a single-conversion.
- #define ADS1115_REG_CONFIG_OS_BUSY (0x0000)

  Read: 0 when conversion is in progress.
- #define ADS1115_REG_CONFIG_OS_NOTBUSY (0x8000)

  Read: 1 when device is not performing a conversion.
- #define ADS1115_REG_CONFIG_MUX_DIFF_P0_N1 (0x0000)

  Differential input with positive input from AIN0 and negative input from AIN1 (DEFAULT).
- #define ADS1115_REG_CONFIG_MUX_DIFF_P0_N3 (0x1000)

  Differential input with positive input from AIN0 and negative input from AIN3.
- #define ADS1115_REG_CONFIG_MUX_DIFF_P1_N3 (0x2000)

  Differential input with positive input from AIN1 and negative input from AIN3.
- #define ADS1115_REG_CONFIG_MUX_DIFF_P2_N3 (0x3000)

  Differential input with positive input from AIN2 and negative input from AIN3.
- #define ADS1115_REG_CONFIG_MUX_SINGLE_P0_NG (0x4000)

  Single-ended input with positive input from AIN0 with reference to GND.
- #define ADS1115_REG_CONFIG_MUX_SINGLE_P1_NG (0x5000)

  Single-ended input with positive input from AIN1 with reference to GND.
- #define ADS1115_REG_CONFIG_MUX_SINGLE_P2_NG (0x6000)

  Single-ended input with positive input from AIN2 with reference to GND.
- #define ADS1115_REG_CONFIG_MUX_SINGLE_P3_NG (0x7000)

  Single-ended input with positive input from AIN3 with reference to GND.
- #define ADS1115_REG_CONFIG_PGA_6_144V (0x0000)

  +/-6.144V range = Gain 2/3
- #define ADS1115_REG_CONFIG_PGA_4_096V (0x0200)

  +/-4.096V range = Gain 1
- #define ADS1115_REG_CONFIG_PGA_2_048V (0x0400)

  +/-2.048V range = Gain 2 (DEFAULT)
- #define ADS1115_REG_CONFIG_PGA_1_024V (0x0600)

  +/-1.024V range = Gain 4
- #define ADS1115_REG_CONFIG_PGA_0_512V (0x0800)

  +/-0.512V range = Gain 8
- #define ADS1115_REG_CONFIG_PGA_0_256V (0x0A00)

  +/-0.256V range = Gain 16
- #define ADS1115_REG_CONFIG_MODE_CONTIN (0x0000)

  Continuous conversion mode.
- #define ADS1115_REG_CONFIG_MODE_ENDCON (0x8583)

  Shut down continuous conversion.
- #define ADS1115_REG_CONFIG_MODE_SINGLE (0x0100)

  Power-down single-shot mode (DEFAULT)
- #define ADS1115_RATE_008 (0x0000)

  8 samples per second
- #define ADS1115_RATE_016 (0x0020)

  16 samples per second
- #define ADS1115_RATE_032 (0x0040)

  32 samples per second
- #define ADS1115_RATE_064 (0x0060)

64 samples per second

- #define ADS1115_RATE_128 (0x0080)

  128 samples per second

- #define ADS1115_RATE_250 (0x00A0)

  250 samples per second (DEFAULT)

- #define ADS1115_RATE_475 (0x00C0)

  475 samples per second

- #define ADS1115_RATE_860 (0x00E0)

  860 samples per second

- #define ADS1115_COMP_MODE_HYSTERESIS (0x0000)

  Traditional comparator with hysteresis (DEFAULT)

- #define ADS1115_COMP_MODE_WINDOW (0x0010)

  Window comparator.

- #define ADS1115_COMP_POL_ACTIVE_LOW (0x0000)

  ALERT/RDY pin is active low (DEFAULT)

- #define ADS1115_COMP_POL_ACTIVE_HIGH (0x0008)

  ALERT/RDY pin is active high.

- #define ADS1115_COMP_LAT_NON_LATCHING (0x0000)

  Non-latching comparator (DEFAULT)

- #define ADS1115_COMP_LAT_LATCHING (0x0004)

  Latching comparator.

- #define ADS1115_COMP_QUE_ASSERT1 (0x0000)

  Assert after one conversion.

- #define ADS1115_COMP_QUE_ASSERT2 (0x0001)

  Assert after two conversions.

- #define ADS1115_COMP_QUE_ASSERT4 (0x0002)

  Assert after four conversions.

- #define ADS1115_COMP_QUE_DISABLE (0x0003)

  Disable comparator (DEFAULT)

- #define ADS1115_REG_THRESH_MSB_1 (0x8000)

  Set MSB to 1.

- #define ADS1115_REG_THRESH_MSB_0 (0x7FFF)

  Set MSB to 0.

9.2.1 Detailed Description

Defines a set of macros for use in configuring the ADS1115 and a class for controlling the configured device.

Provides a detailed library of the programmable characteristics of the ADS1115. These characterstics are listed below and the macro associated with each configuration is listed in this header.

- Multiplexer configuration
  - AINp = AIN0 and AINn = AIN1
  - AINp = AIN0 and AINn = AIN3
  - AINp = AIN1 and AINn = AIN3
  - AINp = AIN2 and AINn = AIN3
  - AINp = AIN0 and AINn = GND
  - AINp = AIN1 and AINn = GND

  - AINp = AIN2 and AINn = GND
  - AINp = AIN3 and AINn = GND
- Programmable gain amplifier configuration
  - +-6.144V
  - +-4.096V
  - +-2.048V (DDEFAULT)
  - +-1.024V
  - +-0.512V
  - +-0.256V
- Device operating mode
  - Continuous conversion mode
  - Power-down single-shot mode (DEFAULT)
- Data rate
  - 8SPS
  - 16SPS
  - 32SPS
  - 64SPS
  - 128SPS (DEFAULT)
  - 250SPS
  - 475SPS
  - 860SPS
- Comparator mode
  - Traditional comparator with hysteresis (DEFAULT)
  - "Window" comparator
- Comparator polarity
  - Active low (DEFAULT)
  - Active high
- Latching comparator
  - Non-latching comparator (DEFAULT)
  - Latching comparator
- Comparator queue and disable
  - Assert after one conversion
  - Assert after two conversions
  - Assert after four conversions
  - Disable comparator (DEFAULT)

The addresses associated with tying the address pin to available lines are listed. The available lines are listed below:

- I2C address pin low
- I2C address pin high
- I2C address pin tied to the SDA line
- I2C address pin tied to the SCL line

Author

Zonghan Gan

Finlay Nelson

Henry Cowan

Date

11 April 2020

9.3 /home/finlay/RTEP1/Rpi_end/200322_Rpi_end_together/GPIOlis.cpp File Reference

```
#include <QDebug>
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <stdlib.h>
#include <wiringPi.h>
#include <iostream>
#include <functional>
#include "GPIOlis.h"
```

Include dependency graph for GPIOlis.cpp:

Typedefs

- using gpipinterrupt = 0

  A flag to identify the activation of the threading operation.

Functions

- void interrupt2 (void)

  Called in response to an interrupt, to alter the value of the flag "gpipinterrupt".

Variables

- mutex cmdmtx

9.3.1 Typedef Documentation

9.3.1.1 gpipinterrupt

```
using gpipinterrupt = 0
```

A flag to identify the activation of the threading operation.

Definition at line 17 of file GPIOlis.cpp.

9.3.2 Function Documentation

9.3.2.1 interrupt2()

```
void interrupt2 (
            void  )
```

Called in response to an interrupt, to alter the value of the flag "gpipinterrupt".

< Locks the mutex if the mutex isn't currently locked by any thread

< Unlocks the mutex if the mutex isn't currently locked by any thread

Definition at line 30 of file GPIOlis.cpp.

9.3.3 Variable Documentation

9.3.3.1 cmdmtx

```
mutex cmdmtx
```

Definition at line 24 of file GPIOlis.cpp.

9.4 /home/finlay/RTEP1/Rpi_end/200322_Rpi_end_together/GPIOlis.h File Reference

Threading and interrupts.

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

Include dependency graph for GPIOlis.h: This graph shows which files directly or indirectly include this file:

Classes

- class GPIOlisten

  Is responsible for the data communications and threading involved with conversion and processing.

Variables

- int gpipinterrupt

  Stores a binary value relating to the interrupt state.

9.4.1 Detailed Description

Threading and interrupts.

Listens to the Alert pin of the ADS1115 in continuous mode, interrupts every time an AD conversion is ready, then sends data and triggers processing through the signal-slot mechanism. "run()" listens to the interrupt and calls back the interrupt every time it is detected before sleeping for 1ms. interrupt() reads in data and sends back a readyread signal.

Author

Zonghan Gan

Finlay Nelson

Henry Cowan

Date

11 April 2020

9.4.2 Variable Documentation

9.4.2.1 gpipinterrupt

```
int gpipinterrupt
```

Stores a binary value relating to the interrupt state.

9.5 /home/finlay/RTEP1/Rpi_end/200322_Rpi_end_together/window.cpp File Reference

```
#include <cmath>
#include <QObject>
#include <Iir.h>
#include <stdio.h>
#include <stdlib.h>
#include "window.h"
#include "ads1115.h"
#include "GPIOlis.h"
```

Include dependency graph for window.cpp:

Variables

- const int order = 40

  Order of the IIR filter.
- const float samplingrate = 860

  Data rate for sampling incoming signals.
- int index = 0

  An index counter used in the for-loop within Window::Window.
- const quint16 sderprt = 1117
- const quint16 rscverprt = 1112
- FILE * florigin = NULL
- FILE * flhp1 = NULL
- FILE * flpower =NULL
- Iir::Butterworth::HighPass< order > hp1

  A high-pass Butterworth IIR filter.

9.5.1 Variable Documentation

9.5.1.1 flhp1

```
FILE* flhp1 = NULL
```

Definition at line 35 of file window.cpp.

9.5.1.2 florigin

```
FILE* florigin = NULL
```

Definition at line 34 of file window.cpp.

9.5.1.3 flpower

```
FILE* flpower =NULL
```

Definition at line 36 of file window.cpp.

9.5.1.4 hp1

```
Iir::Butterworth::HighPass<order> hp1
```

A high-pass Butterworth IIR filter.

The order of this instance of the filter is defined by the "order" variable

Definition at line 44 of file window.cpp.

9.5.1.5 index

```
int index = 0
```

An index counter used in the for-loop within Window::Window.

Definition at line 29 of file window.cpp.

9.5.1.6 order

```
const int order = 40
```

Order of the IIR filter.

Definition at line 25 of file window.cpp.

9.5.1.7 rscverprt

```
const quint16 rscverprt = 1112
```

Definition at line 32 of file window.cpp.

9.5.1.8 samplingrate

```
const float samplingrate = 860
```

Data rate for sampling incoming signals.

Definition at line 27 of file window.cpp.

9.5.1.9 sderprt

```
const quint16 sderprt = 1117
```

Definition at line 31 of file window.cpp.

9.6 /home/finlay/RTEP1/Rpi_end/200322_Rpi_end_together/window.h File Reference

```
#include <qwt/qwt_thermo.h>
#include <qwt/qwt_knob.h>
#include <qwt/qwt_plot.h>
#include <qwt/qwt_plot_curve.h>
#include <QBoxLayout>
#include <QUdpSocket>
#include <QTimer>
#include <QWidget>
#include <Iir.h>
#include "ads1115.h"
#include "GPIOlis.h"
```

Include dependency graph for window.h: This graph shows which files directly or indirectly include this file:

Classes

- class QTimer

  Unknown.
- class Window

  Unknown.

9.7 /home/finlay/RTEP1/Server_end/0301_Pong_GUI/main.cpp File Reference

```
#include <QApplication>
#include "mainwindow.h"
#include <QScreen>
```

Include dependency graph for main.cpp:

Functions

- int main (int argc, char ∗argv[ ])

9.7.1 Function Documentation

9.7.1.1 main()

```
int main (
            int argc,
            char * argv[] )
```

The `main` function is used to find and update the mainwindow with the optimum screen and display size for android applications using Qt

Definition at line 10 of file main.cpp.

9.8 /home/finlay/RTEP1/Rpi_end/200322_Rpi_end_together/main.cpp File Reference

```
#include <window.h>
#include <QApplication>
```
Include dependency graph for main.cpp:

Functions

- int main (int argc, char ∗argv[ ])

9.8.1 Function Documentation

9.8.1.1 main()

```
int main (
            int argc,
            char * argv[] )
```

Version

1.3

Date

2020-4-10

Copyright

GNU Public License

/∗ This demo is to test the ads1115 lib self-written to sample continously no such lib for ads1115 in c++ yet. in this test demo the ads sampling is controlled by timer instead of interrupt. This is not because we can't This is to separate the lib and expose if any bug the interrupt thread unit will soon comes out –Zonghan Gan 200318 23-39

Definition at line 19 of file main.cpp.

9.9 /home/finlay/RTEP1/Server_end/0301_Pong_GUI/mainwindow.cpp File Reference

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QGraphicsScene>
#include <QGraphicsRectItem>
#include <QGraphicsEllipseItem>
#include <QPen>
#include <QResizeEvent>
#include <QDebug>
#include <QTimer>
#include <QObject>
```

Include dependency graph for mainwindow.cpp:

Variables

- const quint16 rsPort = 1112

  Not sure what this is for.

9.9.1 Variable Documentation

9.9.1.1 rsPort

```
const quint16 rsPort = 1112
```

Not sure what this is for.

Definition at line 18 of file mainwindow.cpp.

9.10 /home/finlay/RTEP1/Server_end/0301_Pong_GUI/mainwindow.h File Reference

```
#include <QObject>
#include <QGraphicsScene>
#include <QMainWindow>
#include <QUdpSocket>
#include <QElapsedTimer>
```

Include dependency graph for mainwindow.h: This graph shows which files directly or indirectly include this file:

Classes

- class MainWindow

Namespaces

- Ui

Index