i2c_adc_ads7828

v2.0.2

Generated by Doxygen 1.8.12

Contents

1 Arduino library for TI ADS7828 I2C A/D converter.

Version

`2.0.2`

Date

27 Sep 2016

Source Code Repository:

`https://github.com/4-20ma/i2c_adc_ads7828`

Programming Style Guidelines:

`http://geosoft.no/development/cppstyle.html`

Features

The ADS7828 is a single-supply, low-power, 12-bit data acquisition device that features a serial I2C interface and an 8-channel multiplexer. The Analog-to-Digital (A/D) converter features a sample-and-hold amplifier and internal, asynchronous clock. The combination of an I2C serial, 2-wire interface and micropower consumption makes the ADS7828 ideal for applications requiring the A/D converter to be close to the input source in remote locations and for applications requiring isolation. The ADS7828 is available in a TSSOP-16 package.

Schematic

```
                               Arduino
                              .---------------------.
                              |Duemilanove          |
                              |                     |
   TI ADS7828        .-------o|5V                   |
  .--------------.   |        |                     |
-o|1 CH0   VDD 16|o--'   .---o|GND                  |
  |              |       |    |                     |
-o|2 CH1   SDA 15|o------)---o|A4 SDA               |
  |              |       |    |                     |
-o|3 CH2   SCL 14|o------)---o|A5 SCL               |
  |              |       |    '---------------------'
-o|4 CH3     A1 13|o------o
  |              |       |
-o|5 CH4     A0 12|o------o
  |              |       |
-o|6 CH5   COM 11|o-     |
  |              |       |
-o|7 CH6   REF 10|o-     |
  |              |       |
-o|8 CH7   GND  9|o------o
  '--------------'       |
                         |
                        ===
                        GND
```

Caveats

Conforms to Arduino IDE 1.5 Library Specification v2.1 which requires Arduino IDE >= 1.5.

Support

Please `submit an issue` for all questions, bug reports, and feature requests. Email requests will be politely redirected to the issue tracker so others may contribute to the discussion and requestors get a more timely response.

Author

Doc Walker (4-20ma@wvfans.net)

2 Optional Functions List (Troubleshooting)

Member ADS7828::commandByte ()

This function is for testing and troubleshooting.

Member ADS7828::start ()

This function is for testing and troubleshooting and can be used to determine whether a device is available (similar to the TCP/IP `ping command`).

Member ADS7828::start (uint8_t)

This function is for testing and troubleshooting.

Member ADS7828Channel::commandByte ()

This function is for testing and troubleshooting.

Member ADS7828Channel::index ()

This function is for testing and troubleshooting.

Member ADS7828Channel::sample ()

This function is for testing and troubleshooting.

Member ADS7828Channel::start ()

This function is for testing and troubleshooting.

Member ADS7828Channel::total ()

This function is for testing and troubleshooting.

Member ADS7828Channel::update ()

This function is for testing and troubleshooting.

3 Todo List

Member ADS7828Channel::start ()

Determine whether this function is needed.

Member ADS7828Channel::update ()

Determine whether this function is needed.

4 Required Functions List

Member ADS7828::begin ()

Call from within `setup()`to enable I2C communication.

Member ADS7828::update (uint8_t)

Call this or one of the update() / updateAll() functions from within `loop()` in order to read data from device(s).

Member ADS7828::update ()

Call this or one of the update() / updateAll() functions from within `loop()` in order to read data from device(s).

Member ADS7828::updateAll ()

Call this or one of the update() functions from within `loop()` in order to read data from device(s). This is the most commonly-used device update function.

Member ADS7828Channel::value ()

This is the most commonly-used channel function.

5 Class Index

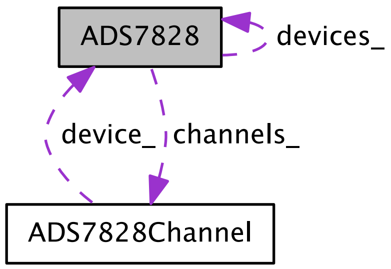
5.1 Class List

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

6 Class Documentation

6.1 ADS7828 Class Reference

Collaboration diagram for ADS7828:



Public Member Functions

- ADS7828 (uint8_t)

  Constructor with the following defaults:
- ADS7828 (uint8_t, uint8_t)
- ADS7828 (uint8_t, uint8_t, uint8_t)
- ADS7828 (uint8_t, uint8_t, uint8_t, uint16_t, uint16_t)
- uint8_t address ()

  Device address as defined by pins A1, A0.
- ADS7828Channel ∗ channel (uint8_t)

  Return pointer to channel object.
- uint8_t commandByte ()

  Return command byte for device object (PD1 PD0 bits only).
- uint8_t start ()

  Initiate communication with device.
- uint8_t start (uint8_t)
- uint8_t update ()

  Update all unmasked channels on device.
- uint8_t update (uint8_t)

Static Public Member Functions

- static void begin ()

  Enable I2C communication.
- static ADS7828 ∗ device (uint8_t)

  Return pointer to device object.
- static uint8_t updateAll ()

  Update all unmasked channels on all registered devices.

Public Attributes

- uint8_t channelMask

  Each bit position containing a 1 represents a channel that is to be read via update() / updateAll().

Private Member Functions

- void init (uint8_t, uint8_t, uint8_t, uint16_t, uint16_t)

  Common code for constructors.
- uint16_t read ()

  Request and receive data from most-recent A/D conversion from device.

Static Private Member Functions

- static uint16_t read (uint8_t)

  Request and receive data from most-recent A/D conversion from device.
- static uint8_t start (uint8_t, uint8_t)

  Initiate communication with device.
- static uint8_t update (ADS7828 ∗)

  Initiate communication with device.
- static uint8_t update (ADS7828 ∗, uint8_t)

  Initiate communication with device.

Private Attributes

- uint8_t address_

  Device address as defined by pins A1, A0.
- ADS7828Channel channels_ [8]

  Array of channel objects.
- uint8_t commandByte_

  Command byte for device object (PD1 PD0 bits only).

Static Private Attributes

- static ADS7828 ∗ devices_ [4] = {}

  Array of pointers to registered device objects.
- static const uint8_t BASE_ADDRESS_ = 0x48

  Factory pre-set slave address.

Related Functions

(Note that these are not member functions.)

- static const uint8_t DIFFERENTIAL = 0 << 7

  Configure channels to use differential inputs (Command byte SD=0).
- static const uint8_t SINGLE_ENDED = 1 << 7

  Configure channels to use single-ended inputs (Command byte SD=1).
- static const uint8_t REFERENCE_OFF = 0 << 3

  Configure channels to turn internal reference OFF between conversions (Command byte PD1=0).
- static const uint8_t REFERENCE_ON = 1 << 3

  Configure channels to turn internal reference ON between conversions (Command byte PD1=1).
- static const uint8_t ADC_OFF = 0 << 2

  Configure channels to turn A/D converter OFF between conversions (Command byte PD0=0).
- static const uint8_t ADC_ON = 1 << 2

  Configure channels to turn A/D converter ON between conversions (Command byte PD0=1).
- static const uint8_t DEFAULT_CHANNEL_MASK = 0xFF

  Default channel mask used in ADS7828 constructor.
- static const uint16_t DEFAULT_MIN_SCALE = 0

  Default scaling minimum value used in ADS7828 constructor.
- static const uint16_t DEFAULT_MAX_SCALE = 0xFFF

  Default scaling maximum value used in ADS7828 constructor.

6.1.1 Detailed Description

Examples:

examples/one_device/one_device.ino, and examples/two_devices/two_devices.ino.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 ADS7828() [1/4]

```
ADS7828::ADS7828 (
            uint8_t address )
```

Constructor with the following defaults:

- differential inputs (SD=0)
- internal reference OFF between conversions (PD1=0)
- A/D converter OFF between conversions (PD0=0)
- min scale=0
- max scale=4095

Parameters

address	device address (0..3)

Usage:

```
...
// construct device with address 2
ADS7828 adc(2);
...
```

See also

ADS7828::address()

```
{
  init(address, (DIFFERENTIAL | REFERENCE_OFF |
  ADC_OFF),
    DEFAULT_CHANNEL_MASK, DEFAULT_MIN_SCALE,
  DEFAULT_MAX_SCALE);
}
```

6.1.2.2 ADS7828() [2/4]

```
ADS7828::ADS7828 (
            uint8_t address,
            uint8_t options )
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

options	command byte bits SD, PD1, PD0

Usage:

```
...
// device address 0, differential inputs, ref/ADC ON between conversions
ADS7828 adc0(0, DIFFERENTIAL | REFERENCE_ON | ADC_ON);

// device address 1, single-ended inputs, ref/ADC OFF between conversions
ADS7828 adc1(1, SINGLE_ENDED | REFERENCE_OFF |
      ADC_OFF);

// device address 2, single-ended inputs, ref/ADC ON between conversions
ADS7828 adc2(2, SINGLE_ENDED | REFERENCE_ON |
      ADC_ON);
...
```

See also

ADS7828Channel::commandByte()

```
{
  init(address, options, DEFAULT_CHANNEL_MASK,
  DEFAULT_MIN_SCALE,
    DEFAULT_MAX_SCALE);
}
```

6.1.2.3 ADS7828() [3/4]

```
ADS7828::ADS7828 (
              uint8_t address,
              uint8_t options,
              uint8_t channelMask )
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

channelMask	bit positions containing a 1 represent channels that are to be read via update() / updateAll()

Usage:

```
...
// device address 0, update all channels via updateAll() (bits 7..0 are set)
ADS7828 adc0(0, 0, 0xFF);

// device address 1, update channels 0..3 via updateAll() (bits 3..0 are set)
ADS7828 adc1(1, 0, 0b00001111);

// device address 2, update channels 0, 1, 2, 7 via updateAll() (bits 7, 2, 1, 0 are set)
ADS7828 adc2(2, 0, 0b10000111);
...
```

See also

ADS7828::channelMask

```
{
  init(address, options, channelMask, DEFAULT_MIN_SCALE,
  DEFAULT_MAX_SCALE);
}
```

6.1.2.4 ADS7828() [4/4]

```
ADS7828::ADS7828 (
              uint8_t address,
              uint8_t options,
              uint8_t channelMask,
              uint16_t min,
              uint16_t max )
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

min	minimum scaling value applied to value()
max	maximum scaling value applied to value()

Usage:

```
...
// device address 2, channel default minScale 0, maxScale 100
ADS7828 adc(2, 0, DEFAULT_CHANNEL_MASK, 0, 100);
...
```

See also

ADS7828Channel::minScale, ADS7828Channel::maxScale

```
{
  init(address, options, channelMask, min, max);
}
```

6.1.3 Member Function Documentation

6.1.3.1 address()

```
uint8_t ADS7828::address ( )
```

Device address as defined by pins A1, A0.

Return values

0x00	A1=0, A0=0
0x01	A1=0, A0=1
0x02	A1=1, A0=0
0x03	A1=1, A0=1

Usage:

```
...
ADS7828 adc(3);
uint8_t deviceAddress = adc.address();
...
```

Examples:

examples/two_devices/two_devices.ino.

```
{
  return address_;
}
```

6.1.3.2 channel()

```
ADS7828Channel * ADS7828::channel (
            uint8_t ch )
```

Return pointer to channel object.

Parameters

ch	channel number (0..7)

Returns

pointer to ADS7828Channel object

Usage:

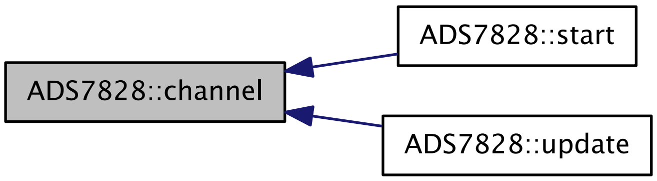
```
...
ADS7828 adc(0);
ADS7828Channel* temperature = adc.channel(0);
...
```

Examples:

examples/one_device/one_device.ino.

```
{
  return &channels_[ch & 0x07];
}
```

Here is the caller graph for this function:



6.1.3.3 commandByte()

```
uint8_t ADS7828::commandByte ( )
```

Return command byte for device object (PD1 PD0 bits only).

Optional Function (Troubleshooting) This function is for testing and troubleshooting.

Return values

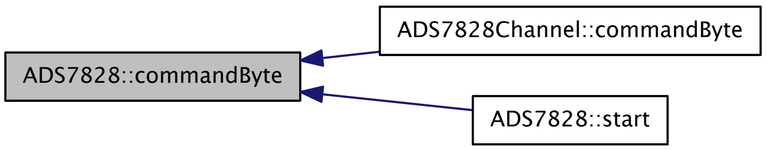
0x00	Power Down Between A/D Converter Conversions
0x04	Internal Reference OFF and A/D Converter ON
0x08	Internal Reference ON and A/D Converter OFF
0x0C	Internal Reference ON and A/D Converter ON

Usage:

```
...
ADS7828 adc(0);
uint8_t command = adc.commandByte();
...
```

```
{
  return commandByte_;
}
```

Here is the caller graph for this function:



6.1.3.4 start() [1/3]

`uint8_t ADS7828::start ( )`

Initiate communication with device.

Optional Function (Troubleshooting) This function is for testing and troubleshooting and can be used to determine whether a device is available (similar to the TCP/IP `ping command`).

Return values

0	success
1	length too long for buffer
2	address send, NACK received **(device not on bus)**
3	data send, NACK received
4	other twi error (lost bus arbitration, bus error, ...)

Usage:

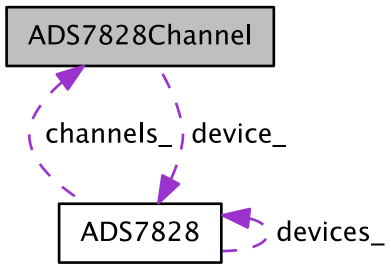
```
...
ADS7828 adc(3);
// test whether device is available
uint8_t status = adc.start();
...
```

```
{
  return start(0);
}
```

Here is the call graph for this function:

6.2 ADS7828Channel Class Reference

Collaboration diagram for ADS7828Channel:



Public Member Functions

- ADS7828Channel (ADS7828 ∗const, uint8_t, uint8_t, uint16_t, uint16_t)
- uint8_t commandByte ()

    Return command byte for channel object.
- ADS7828 ∗ device ()

    Return pointer to parent device object.
- uint8_t id ()

    Return ID number of channel object (+IN connection).
- uint8_t index ()

    Return index position within moving average array.
- void newSample (uint16_t)

    Add (unscaled) sample value to moving average array, update totalizer.
- void reset ()

    Reset moving average array, index, totalizer to zero.
- uint16_t sample ()

Return most-recent (unscaled) sample value from moving average array.

- uint8_t start ()

  Initiate A/D conversion for channel object.
- uint16_t total ()

  Return (unscaled) totalizer value for channel object.
- uint8_t update ()

  Initiate A/D conversion, read data, update moving average for channel object.
- uint16_t value ()

  Return moving average value for channel object.

Public Attributes

- uint16_t maxScale

  Maximum value of moving average (defaults to 0x0FFF).
- uint16_t minScale

  Minimum value of moving average (defaults to 0x0000).

Private Attributes

- uint8_t commandByte_

  Command byte for channel object (SD C2 C1 C0 bits only).
- ADS7828 ∗ device_

  Pointer to parent device object.
- uint8_t index_

  Index position within moving average array.
- uint16_t samples_ [1<< 4]

  Array of (unscaled) sample values.
- uint16_t total_

  (Unscaled) running total of moving average array elements.

Static Private Attributes

- static const uint8_t MOVING_AVERAGE_BITS_ = 4

  Quantity of samples to be averaged = $2^{MOVING\_AVERAGE\_BITS\_}$.

6.2.1 Detailed Description

Examples:

examples/one_device/one_device.ino, and examples/two_devices/two_devices.ino.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 ADS7828Channel()

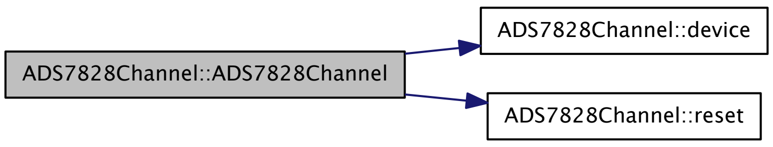
```
ADS7828Channel::ADS7828Channel (
            ADS7828 * const device,
            uint8_t id,
            uint8_t options,
            uint16_t min,
            uint16_t max )
```

Remarks

Invoked by ADS7828 constructor; this function will not normally be called by end user.

```
{
  this->device_ = device;
  this->commandByte_ = (bitRead(options, 7) << 7) | (bitRead(id, 0) << 6) |
    (bitRead(id, 2) << 5) | (bitRead(id, 1) << 4);
  this->minScale = min;
  this->maxScale = max;
  reset();
}
```

Here is the call graph for this function:



6.2.3 Member Function Documentation

6.2.3.1 commandByte()

```
uint8_t ADS7828Channel::commandByte ( )
```

Return command byte for channel object.

Optional Function (Troubleshooting) This function is for testing and troubleshooting.

Returns

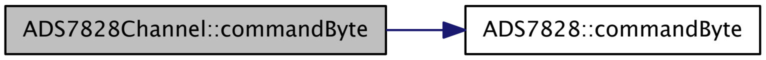
command byte (0x00..0xFC)

Usage:

```
...
ADS7828 adc(0);
ADS7828Channel* temperature = adc.channel(0);
uint8_t command = temperature->commandByte();
...
```

```
{
  return commandByte_ | device_->commandByte();
}
```

Here is the call graph for this function:



Here is the caller graph for this function:

6.2.3.2 device()

`ADS7828 * ADS7828Channel::device ( )`

Return pointer to parent device object.

Returns

pointer to parent ADS7828 object

Usage:

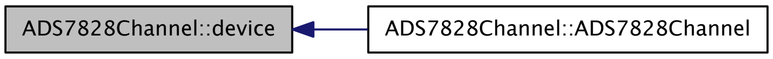
```
...
ADS7828 adc(0);
ADS7828Channel* temperature = adc.channel(0);
ADS7828* parentDevice = temperature->device();
...
```

Examples:

examples/two_devices/two_devices.ino.

```
{
  return device_;
}
```

Here is the caller graph for this function:



6.2.3.3 id()

```
uint8_t ADS7828Channel::id ( )
```

Return ID number of channel object (+IN connection).

Single-ended inputs use COM as -IN; Differential inputs are as follows:

- 0 indicates CH0 as +IN, CH1 as -IN
- 1 indicates CH1 as +IN, CH0 as -IN
- 2 indicates CH2 as +IN, CH3 as -IN
- ...
- 7 indicates CH7 as +IN, CH6 as -IN

Returns

id (0..7)

Return values

0	command byte C2 C1 C0 = 000
1	command byte C2 C1 C0 = 100
2	command byte C2 C1 C0 = 001
3	command byte C2 C1 C0 = 101
4	command byte C2 C1 C0 = 010
5	command byte C2 C1 C0 = 110
6	command byte C2 C1 C0 = 011
7	command byte C2 C1 C0 = 111

Usage:

```
...
ADS7828 adc(0);
ADS7828Channel* temperature = adc.channel(0);
uint8_t channelId = temperature->id();
...
```

Examples:

examples/two_devices/two_devices.ino.

```
{
  return ((bitRead(commandByte_, 5) << 2) | (bitRead(commandByte_, 4) << 1) |
    (bitRead(commandByte_, 6)));
}
```

6.2.3.4 index()

```
uint8_t ADS7828Channel::index ( )
```

Return index position within moving average array.

Optional Function (Troubleshooting) This function is for testing and troubleshooting.

Returns

index (0..$2^{\text{MOVING\_AVERAGE\_BITS\_}}$ - 1)

Usage:

```
...
ADS7828 adc(0);
ADS7828Channel* temperature = adc.channel(0);
uint8_t channelIndex = temperature->index();
...
```

```
{
  return index_;
}
```

6.2.3.5 newSample()

```
void ADS7828Channel::newSample (
              uint16_t sample )
```

Add (unscaled) sample value to moving average array, update totalizer.

Parameters

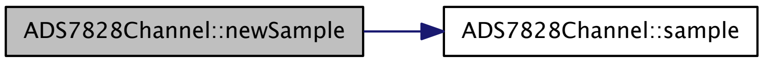
sample	sample value (0x0000..0xFFFF)

Remarks

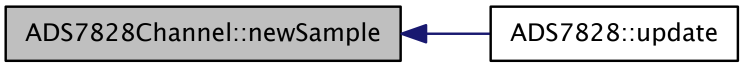
Invoked by ADS7828::update() / ADS7828::updateAll() functions; this function will not normally be called by end user.

```
{
  this->index_++;
  if (index_ >= (1 << MOVING_AVERAGE_BITS_)) this->
  index_ = 0;
  this->total_ -= samples_[index_];
  this->samples_[index_] = sample;
  this->total_ += samples_[index_];
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.2.3.6 reset()

```
void ADS7828Channel::reset ( )
```

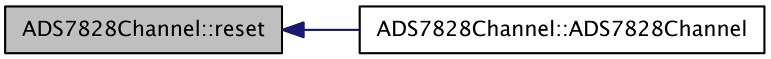
Reset moving average array, index, totalizer to zero.

Usage:

```
...
ADS7828 adc(0);
ADS7828Channel* temperature = adc.channel(0);
temperature->reset();
...
```

```
{
  this->index_ = this->total_ = 0;
  for (uint8_t k = 0; k < (1 << MOVING_AVERAGE_BITS_); k++)
  {
    this->samples_[k] = 0;
  }
}
```

Here is the caller graph for this function:



6.2.3.7 sample()

```
uint16_t ADS7828Channel::sample ( )
```

Return most-recent (unscaled) sample value from moving average array.

Optional Function (Troubleshooting) This function is for testing and troubleshooting.

Returns

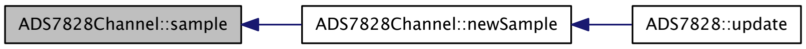
sample value (0x0000..0xFFFF)

Usage:

```
...
ADS7828 adc(0);
ADS7828Channel* temperature = adc.channel(0);
uint16_t sampleValue = temperature->sample();
...
```

```
{
  return samples_[index_];
}
```

Here is the caller graph for this function:



6.2.3.8 start()

```
uint8_t ADS7828Channel::start ( )
```

Initiate A/D conversion for channel object.

Optional Function (Troubleshooting) This function is for testing and troubleshooting.

Todo Determine whether this function is needed.

Return values

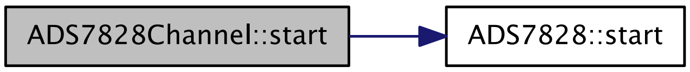
0	success
1	length too long for buffer
2	address send, NACK received **(device not on bus)**
3	data send, NACK received
4	other twi error (lost bus arbitration, bus error, ...)

Usage:

```
...
ADS7828 adc(0);
ADS7828Channel* temperature = adc.channel(0);
uint8_t status = temperature->start();
...
```

```
{
  return device_->start(id());
}
```

Here is the call graph for this function:



Here is the caller graph for this function:

6.2.3.9 total()

```
uint16_t ADS7828Channel::total ( )
```

Return (unscaled) totalizer value for channel object.

Optional Function (Troubleshooting) This function is for testing and troubleshooting.

Returns

totalizer value (0x0000..0xFFFF)

Usage:

```
...
ADS7828 adc(0);
ADS7828Channel* temperature = adc.channel(0);
uint16_t totalValue = temperature->total();
...
```

```
{
  return total_;
}
```

6.2.3.10 update()

```
uint8_t ADS7828Channel::update ( )
```

Initiate A/D conversion, read data, update moving average for channel object.

Optional Function (Troubleshooting) This function is for testing and troubleshooting.

Todo Determine whether this function is needed.

Return values

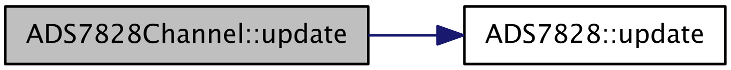
0	success
1	length too long for buffer
2	address send, NACK received **(device not on bus)**
3	data send, NACK received
4	other twi error (lost bus arbitration, bus error, ...)

Usage:

```
...
ADS7828 adc(0);
ADS7828Channel* temperature = adc.channel(0);
uint8_t status = temperature->update();
...
```

```
{
  device_->update(id());
}
```

Here is the call graph for this function:



Here is the caller graph for this function:

6.2.3.11 value()

`uint16_t ADS7828Channel::value ( )`

Return moving average value for channel object.

Required Function This is the most commonly-used channel function.

Returns

scaled value (0x0000..0xFFFF)

Usage:

```
...
ADS7828 adc(0);
ADS7828Channel* temperature = adc.channel(0);
uint16_t ambient = temperature->value();
...
```

Examples:

examples/one_device/one_device.ino, and examples/two_devices/two_devices.ino.

```
{
  uint16_t r = (total_ >> MOVING_AVERAGE_BITS_);
  return map(r, DEFAULT_MIN_SCALE, DEFAULT_MAX_SCALE, minScale, maxScale);
}
```

6.2.4 Member Data Documentation

6.2.4.1 maxScale

```
uint16_t ADS7828Channel::maxScale
```

Maximum value of moving average (defaults to 0x0FFF).

Usage:

```
...
ADS7828 device(0);
ADS7828Channel* temperature = device.channel(0);
uint16_t old = temperature->maxScale; // get current value and/or
temperature->maxScale = 100;          // set new value
...
```

Examples:

examples/one_device/one_device.ino, and examples/two_devices/two_devices.ino.

6.2.4.2 minScale

```
uint16_t ADS7828Channel::minScale
```

Minimum value of moving average (defaults to 0x0000).

Usage:

```
...
ADS7828 device(0);
ADS7828Channel* temperature = device.channel(0);
uint16_t old = temperature->minScale; // get current value and/or
temperature->minScale = 0;            // set new value
...
```

Examples:

examples/one_device/one_device.ino, and examples/two_devices/two_devices.ino.

6.2.4.3 samples_

```
uint16_t ADS7828Channel::samples_[1<< 4]  [private]
```

Array of (unscaled) sample values.

Note

Bit shift must match MOVING_AVERAGE_BITS_.

6.2.4.4 MOVING_AVERAGE_BITS_

```
const uint8_t ADS7828Channel::MOVING_AVERAGE_BITS_ = 4  [static], [private]
```

Quantity of samples to be averaged = $2^{MOVING\_AVERAGE\_BITS\_}$.

Note

MOVING_AVERAGE_BITS_ must match samples_ bit shift.

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

- i2c_adc_ads7828.h
- i2c_adc_ads7828.cpp

7 Example Documentation

7.1 examples/one_device/one_device.ino

```
/*

  one_device.ino - example using i2c_adc_ads7828 library

  Library:: i2c_adc_ads7828
  Author:: Doc Walker <4-20ma@wvfans.net>

  Copyright:: 2009-2016 Doc Walker

  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

      http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.

*/


#include <i2c_adc_ads7828.h>


// device 0
// Address: A1=0, A0=0
// Command: SD=1, PD1=1, PD0=1
ADS7828 device(0, SINGLE_ENDED | REFERENCE_ON |
      ADC_ON, 0x0F);
ADS7828* adc = &device;
ADS7828Channel* ambientTemp = adc->channel(0);
ADS7828Channel* waterTemp = adc->channel(1);
ADS7828Channel* filterPressure = adc->channel(2);
ADS7828Channel* waterLevel = adc->channel(3);


void setup()
{
  // enable serial monitor
  Serial.begin(19200);

  // enable I2C communication
  ADS7828::begin();

  // adjust scaling on an individual channel basis
  ambientTemp->minScale = 0;
  ambientTemp->maxScale = 150;

  waterTemp->minScale = 0;
  waterTemp->maxScale = 100;

  filterPressure->minScale = 0;
  filterPressure->maxScale = 30;

  waterLevel->minScale = 0;
  waterLevel->maxScale = 100;
}


void loop()
{
  // update all registered ADS7828 devices/unmasked channels
  ADS7828::updateAll();

  // output moving average values to console
  Serial.print("\n Ambient: ");
  Serial.print(ambientTemp->value(), DEC);
  Serial.print("\n Water temp: ");
  Serial.print(waterTemp->value(), DEC);
  Serial.print("\n Filter pressure: ");
  Serial.print(filterPressure->value(), DEC);
  Serial.print("\n Water level: ");
  Serial.print(waterLevel->value(), DEC);
  Serial.print("\n- - - - - - - - - - - - - - - - - - - - \n");

  // delay
  delay(1000);
}
```

7.2 examples/two_devices/two_devices.ino

```
/*

  two_devices.ino - example using i2c_adc_ads7828 library

  Library:: i2c_adc_ads7828
  Author:: Doc Walker <4-20ma@wvfans.net>

  Copyright:: 2009-2016 Doc Walker

  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

      http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.

*/


#include <i2c_adc_ads7828.h>


// device 1
// Address: A1=0, A0=1
// Command: SD=1, PD1=1, PD0=1
ADS7828 device1(1, SINGLE_ENDED | REFERENCE_ON |
      ADC_ON, 0xFF);

// device 2
// Address: A1=1, A0=0
// Command: SD=1, PD1=1, PD0=1
// Scaling: min=0, max=1000
ADS7828 device2(2, SINGLE_ENDED | REFERENCE_ON |
      ADC_ON, 0xFF, 0, 1000);

void setup()
{
  // enable serial monitor
  Serial.begin(19200);

  // enable I2C communication
  ADS7828::begin();
}


void loop()
{
  uint8_t a, ch;

  // update all registered ADS7828 devices/unmasked channels
  ADS7828::updateAll();

  // iterate through device 1..2 channels 0..7
  for (a = 1; a <= 2; a++)
  {
    for (ch = 0; ch < 8; ch++)
    {
      serialPrint(ADS7828::device(a)->channel(ch));
    }
  }
  Serial.print("\n");

  // output moving average values to console
  Serial.print("\n- - - - - - - - - - - - - - - - - - - - \n");

  // delay
  delay(1000);
}


void serialPrint(ADS7828Channel* ch)
{
  // device address (0..3)
  Serial.print("\nAD:");
  Serial.print(ch->device()->address(), DEC);

  // channel ID (0..7)
  Serial.print(", CH:");
  Serial.print(ch->id(), DEC);

  // moving average value (scaled)
  Serial.print(", v:");
  Serial.print(ch->value(), DEC);

  // minimum scale applied to moving average value
  Serial.print(", mn:");
  Serial.print(ch->minScale, DEC);

  // maximum scale applied to moving average value
  Serial.print(", mx:");
  Serial.print(ch->maxScale, DEC);
}
```

Index