i2c_adc_ads7828

v1.1.5

Generated by Doxygen 1.8.2

Wed Jan 2 2013 14:33:24

Contents

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

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.

Author

Doc Walker

Version

1.1.5

Date

29 Dec 2012

Copyright

GNU General Public License v3

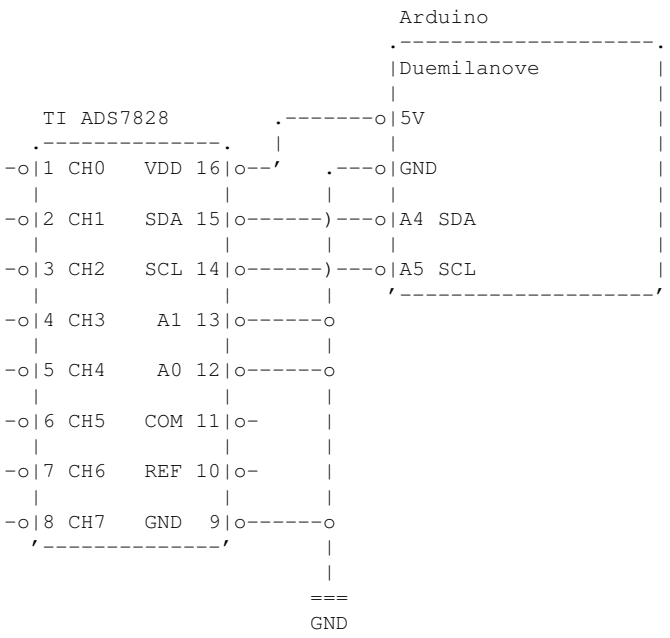
Source Code Repository:

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

Programming Style Guidelines:

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

Schematic:



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

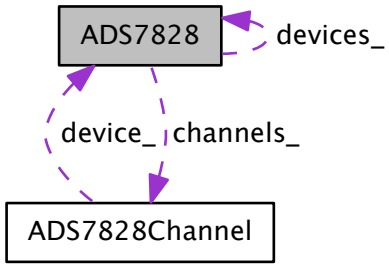
5.1 Class List

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.pde, and examples/two_devices/two_devices.pde.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 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:

```
...
```

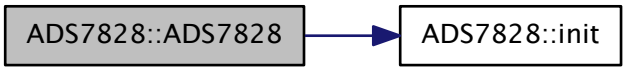
```
// construct device with address 2
ADS7828 adc(2);
...
```

See Also

ADS7828::address()

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

Here is the call graph for this function:



6.1.2.2 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:

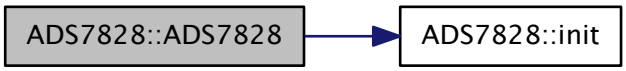
```
...
// device address 0, differential inputs, ref/ADC ON between conversions
ADS7828 adc0(0, DIFFERENTIAL | REFERENCE_ON | ADC_ON
      );

// 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);
}
```

Here is the call graph for this function:



6.1.2.3 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:

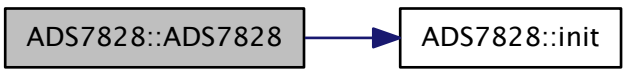
```
...
// device address 0, update all channels via updateAll() (bits 7..0 are set)
ADS7828 adc0(0, 0, 0xFF);

// 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);
}
```

Here is the call graph for this function:



6.1.2.4 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:

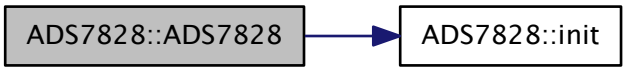
```
...
// device address 2, channel default minScale 0, maxScale 100
ADS7828 adc(2, 0, DEFAULT_CHANNEL_MASK, 0, 100);
...
```

See Also

ADS7828Channel::minScale, ADS7828Channel::maxScale

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

Here is the call graph for this function:



6.1.3 Member Function Documentation

6.1.3.1 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.pde.

```
{
  return address_;
}
```

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

Return pointer to channel object.

Parameters

ch	channel number (0..7)

Returns

pointer to ADS7828Channel object

Usage:

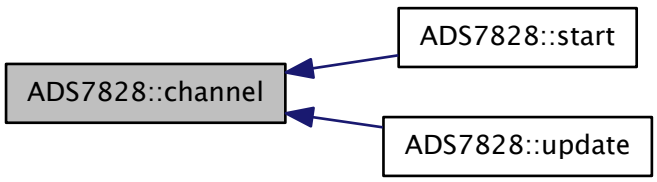
```
...
ADS7828 adc(0);
ADS7828Channel* temperature = adc.channel(0);
...
```

Examples:

examples/one_device/one_device.pde.

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

Here is the caller graph for this function:



6.1.3.3 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

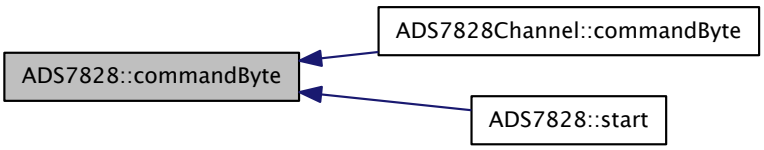
0x00	Power Down Between A/D Converter Conversions
0x04	Internal Reference OFF and A/D Converter ON
0x08	Internal Reference ON and A/D Converter OFF
0x0C	Internal Reference ON and A/D Converter ON

Usage:

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

```
{
  return commandByte_;
}
```

Here is the caller graph for this function:



6.1.3.4 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

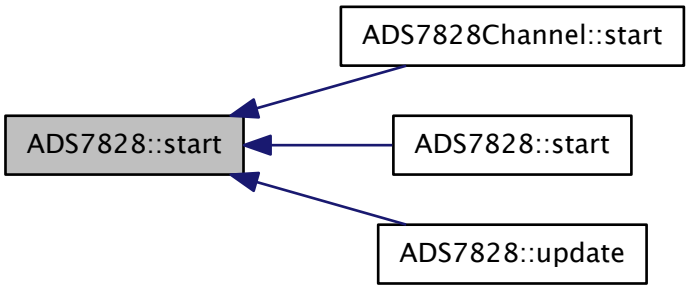
0	success
1	length too long for buffer
2	address send, NACK received **(device not on bus)**
3	data send, NACK received
4	other twi error (lost bus arbitration, bus error, ...)

Usage:

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

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

Here is the caller graph for this function:



6.1.3.5 uint8_t ADS7828::start (uint8_t *ch*)

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

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

Parameters

ch	channel number (0..7)

Return values

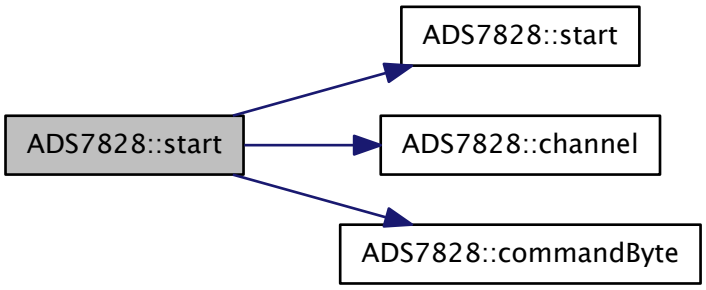
0	success
1	length too long for buffer
2	address send, NACK received **(device not on bus)**
3	data send, NACK received
4	other twi error (lost bus arbitration, bus error, ...)

Usage:

```
    ...
    ADS7828 adc(0);
    // test whether device is available (channel 3 A/D conversion started)
    uint8_t status = adc.start(3);
    ...
```

```
{
  return start(address_, commandByte_ | channel
      (ch)->commandByte());
}
```

Here is the call graph for this function:



6.1.3.6 uint8_t ADS7828::update ( )

Update all unmasked channels on device.

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

Returns

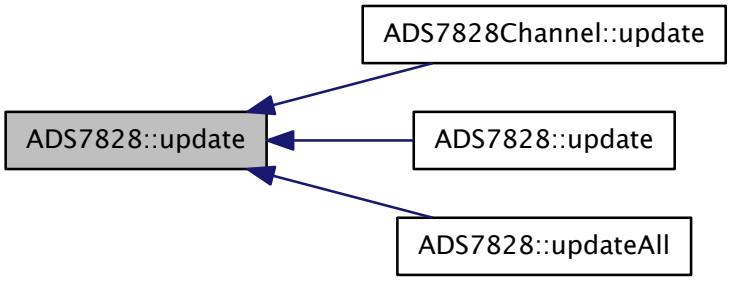
quantity of channels updated (0..8)

Usage:

```
...
ADS7828 adc(0);
...
void loop()
{
  ...
  // update device 0, all unmasked channels
  uint8_t quantity = adc.update();
  ...
}
...
```

```
{
  return update(this);
}
```

Here is the caller graph for this function:



6.1.3.7 uint8_t ADS7828::update (uint8_t *ch*)

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

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

Parameters

ch	channel number (0..7)

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);
...
void loop()
{
  ...
  // update device 0, channel 3
  uint8_t status = adc.update(3);
  ...
}
...
```

```
{
  return update(this, ch);
}
```

Here is the call graph for this function:

6.1.3.8 void ADS7828::begin ( ) [static]

Enable I2C communication.

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

Usage:

```
...
void setup()
{
  // enable I2C communication
  ADS7828::begin();
}
...
```

Examples:

examples/one_device/one_device.pde, and examples/two_devices/two_devices.pde.

```
{
  Wire.begin();
}
```

6.1.3.9 ADS7828 * ADS7828::device (uint8_t *address*) [static]

Return pointer to device object.

Parameters

address	device address (0..3)

Returns

pointer to ADS7828 object

Usage:

```
...
// device 2 pointer
ADS7828* device2 = ADS7828::device(2);
...
```

Examples:

examples/two_devices/two_devices.pde.

```
{
  return devices_[address & 0x03];
}
```

6.1.3.10 uint8_t ADS7828::updateAll ( ) [static]

Update all unmasked channels on all registered devices.

Required Function 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.

Returns

quantity of channels updated (0..32)

Usage:

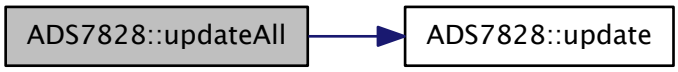
```
...
void loop()
{
  ...
  // update all registered ADS7828 devices/unmasked channels
  uint8_t quantity = ADS7828::updateAll();
  ...
}
...
```

Examples:

examples/one_device/one_device.pde, and examples/two_devices/two_devices.pde.

```
{
  uint8_t a, ch, count = 0;
  for (a = 0; a < 4; a++)
  {
    if (0 != devices_[a]) count += update(devices_[a]);
  }
  return count;
}
```

Here is the call graph for this function:



6.1.3.11 void ADS7828::init (uint8_t *address*, uint8_t *options*, uint8_t *channelMask*, uint16_t *min*, uint16_t *max*) [private]

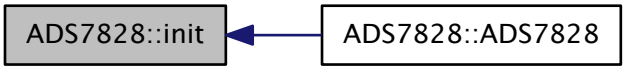
Common code for constructors.

Parameters

address	device address (0..3)
options	command byte bits SD, PD1, PD0
channelMask	bit positions containing a 1 represent channels that are to be read via update() / updateAll()
min	minimum scaling value applied to value()
max	maximum scaling value applied to value()

```
{
  this->address_ = address & 0x03;     // A1 A0 bits
  this->commandByte_ = options & 0x0C; // PD1 PD0 bits
  this->channelMask = channelMask;
  for (uint8_t ch = 0; ch < 8; ch++)
  {
    channels_[ch] = ADS7828Channel(this, ch, options,
      min, max);
  }
  this->devices_[address_] = this;
}
```

Here is the caller graph for this function:



6.1.3.12 uint16_t ADS7828::read ( ) [private]

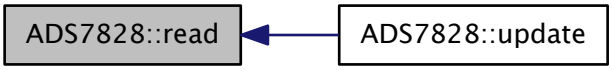
Request and receive data from most-recent A/D conversion from device.

Returns

16-bit zero-padded word (12 data bits D11..D0)

```
{
  return read(address_);
}
```

Here is the caller graph for this function:



6.1.3.13 uint16_t ADS7828::read (uint8_t *address*) [static],[private]

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

Parameters

address	device address (0..3)

Returns

16-bit zero-padded word (12 data bits D11..D0)

```
{
  Wire.requestFrom(BASE_ADDRESS_ | (address & 0x03), 2);
#if defined(ARDUINO) && ARDUINO >= 100
  return word(Wire.read(), Wire.read());
#else
  return word(Wire.receive(), Wire.receive());
#endif
}
```

6.1.3.14 uint8_t ADS7828::start (uint8_t *address*, uint8_t *command*) [static],[private]

Initiate communication with device.

Parameters

address	device address (0..3)
command	command byte (0x00..0xFC)

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, ...)

```
{
  Wire.beginTransmission(BASE_ADDRESS_ | (address & 0x03));
#if defined(ARDUINO) && ARDUINO >= 100
  Wire.write((uint8_t) command);
#else
  Wire.send((uint8_t) command);
#endif
  return Wire.endTransmission();
}
```

6.1.3.15 uint8_t ADS7828::update (ADS7828 * *device*) [static],[private]

Initiate communication with device.

Parameters

device	pointer to device object

Returns

quantity of channels updated (0..8)

```
{
  if (0 == device) device = devices_[0];
  uint8_t ch, count = 0;
  for (ch = 0; ch < 8; ch++)
  {
    if (bitRead(device->channelMask, ch))
    {
      if (0 == update(device, ch)) count++;
    }
  }
  return count;
}
```

Here is the call graph for this function:

6.1.3.16 uint8_t ADS7828::update (ADS7828 * *device*, uint8_t *ch*) [static], [private]

Initiate communication with device.

Parameters

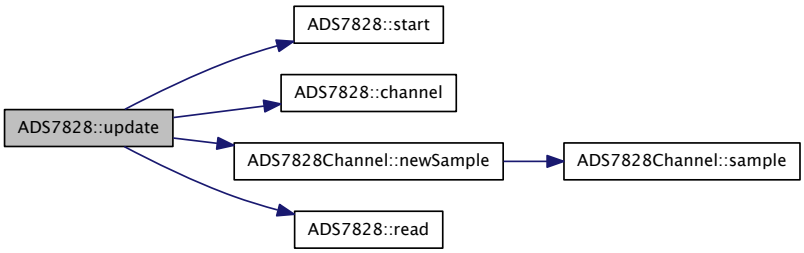
device	pointer to device object
ch	channel number (0..7)

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, ...)

```
{
  if (0 == device) device = devices_[0];
  uint8_t status = device->start(ch);
  if (0 == status) device->channel(ch)->newSample(device->read
      ());
  return status;
}
```

Here is the call graph for this function:



6.1.4 Friends And Related Function Documentation

6.1.4.1 const uint8_t DIFFERENTIAL = 0 << 7 [related]

Configure channels to use differential inputs (Command byte SD=0).

Use either DIFFERENTIAL or SINGLE_ENDED in ADS7828 constructor; default is DIFFERENTIAL.

Usage:

```
...
// address 0, differential inputs, ref/ADC OFF between conversions
ADS7828 adc0(0, DIFFERENTIAL | REFERENCE_OFF | ADC_OFF
      );
...
```

6.1.4.2 const uint8_t SINGLE_ENDED = 1 << 7 [related]

Configure channels to use single-ended inputs (Command byte SD=1).

Use either DIFFERENTIAL or SINGLE_ENDED in ADS7828 constructor; default is DIFFERENTIAL.

Usage:

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

Examples:

examples/one_device/one_device.pde, and examples/two_devices/two_devices.pde.

6.1.4.3 const uint8_t REFERENCE_OFF = 0 << 3 [related]

Configure channels to turn internal reference OFF between conversions (Command byte PD1=0).

Use either REFERENCE_OFF or REFERENCE_ON in ADS7828 constructor; default is REFERENCE_OFF.

Usage:

```
...
```

```
// address 0, differential inputs, ref/ADC OFF between conversions
ADS7828 adc0(0, DIFFERENTIAL | REFERENCE_OFF | ADC_OFF
      );
...
```

6.1.4.4 const uint8_t REFERENCE_ON = 1 << 3 [related]

Configure channels to turn internal reference ON between conversions (Command byte PD1=1).

Use either REFERENCE_OFF or REFERENCE_ON in ADS7828 constructor; default is REFERENCE_OFF.

Usage:

```
...
// address 2, differential inputs, ref ON/ADC OFF between conversions
ADS7828 adc2(2, DIFFERENTIAL | REFERENCE_ON | ADC_OFF
      );
...
```

Examples:

examples/one_device/one_device.pde, and examples/two_devices/two_devices.pde.

6.1.4.5 const uint8_t ADC_OFF = 0 << 2 [related]

Configure channels to turn A/D converter OFF between conversions (Command byte PD0=0).

Use either ADC_OFF or ADC_ON in ADS7828 constructor; default is ADC_OFF.

Usage:

```
...
// address 0, differential inputs, ref/ADC OFF between conversions
ADS7828 adc0(0, DIFFERENTIAL | REFERENCE_OFF | ADC_OFF
      );
...
```

6.1.4.6 const uint8_t ADC_ON = 1 << 2 [related]

Configure channels to turn A/D converter ON between conversions (Command byte PD0=1).

Use either ADC_OFF or ADC_ON in ADS7828 constructor; default is ADC_OFF.

Usage:

```
...
// address 3 , differential inputs, ref OFF/ADC ON between conversions
ADS7828 adc3(3, DIFFERENTIAL | REFERENCE_OFF | ADC_ON
      );
...
```

Examples:

examples/one_device/one_device.pde, and examples/two_devices/two_devices.pde.

6.1.5 Member Data Documentation

6.1.5.1 uint8_t ADS7828::channelMask

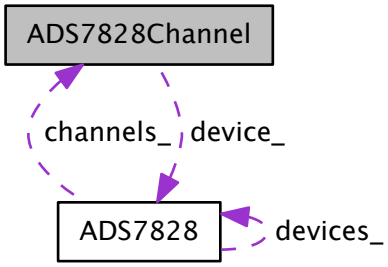
Each bit position containing a 1 represents a channel that is to be read via update() / updateAll().

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

- i2c_adc_ads7828.h
- i2c_adc_ads7828.cpp

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.pde, and examples/two_devices/two_devices.pde.

6.2.2 Constructor & Destructor Documentation

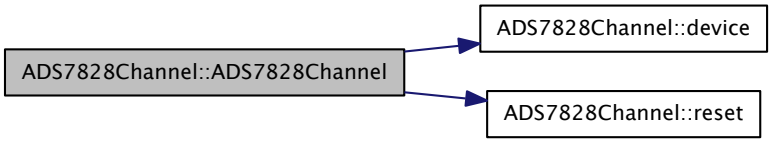
6.2.2.1 ADS7828Channel::ADS7828Channel (ADS7828 * const *device*, uint8_t *id*, uint8_t *options*, uint16_t *min*, uint16_t *max*)

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 uint8_t ADS7828Channel::commandByte ( )

Return command byte for channel object.

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

Returns

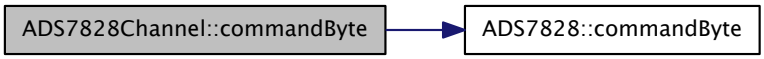
command byte (0x00..0xFC)

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:



6.2.3.2 ADS7828 * ADS7828Channel::device ( )

Return pointer to parent device object.

Returns

pointer to parent ADS7828 object

Usage:

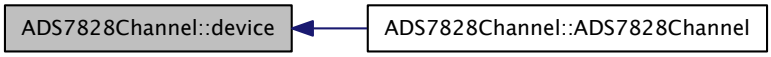
```
...
ADS7828 adc(0);
ADS7828Channel* temperature = adc.channel(0);
ADS7828* parentDevice = temperature->device();
...
```

Examples:

examples/two_devices/two_devices.pde.

```
{
  return device_;
}
```

Here is the caller graph for this function:



6.2.3.3 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.pde.

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

6.2.3.4 uint8_t ADS7828Channel::index ( )

Return index position within moving average array.

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

Returns

index (0..2MOVING_AVERAGE_BITS_ - 1)

Usage:

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

```
{
  return index_;
}
```

6.2.3.5 void ADS7828Channel::newSample (uint16_t *sample*)

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

Parameters

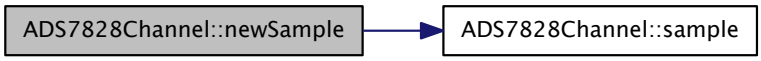
sample	sample value (0x0000..0xFFFF)

Remarks

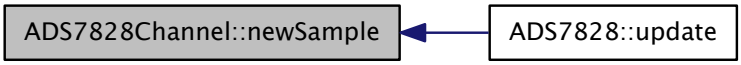
Invoked by ADS7828::update() / ADS7828::updateAll() functions; this function will not normally be called by end user.

```
{
  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 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 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

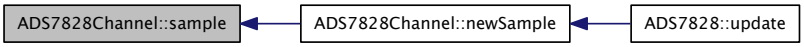
sample value (0x0000..0xFFFF)

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 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

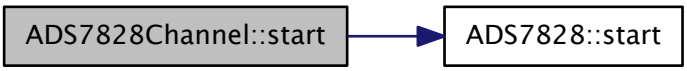
0	success
1	length too long for buffer
2	address send, NACK received **(device not on bus)**
3	data send, NACK received
4	other twi error (lost bus arbitration, bus error, ...)

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:



6.2.3.9 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 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, ...)

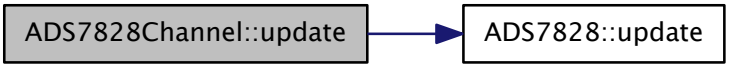
Usage:

```
...
ADS7828 adc(0);
```

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

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

Here is the call graph for this function:



6.2.3.11 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.pde, and examples/two_devices/two_devices.pde.

```
{
  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 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.pde, and examples/two_devices/two_devices.pde.

6.2.4.2 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.pde, and examples/two_devices/two_devices.pde.

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

Array of (unscaled) sample values.

Note

Bit shift must match MOVING_AVERAGE_BITS_.

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

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

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.pde

```
#include <i2c_adc_ads7828.h>
#include <Wire.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.pde

```
#include <i2c_adc_ads7828.h>
#include <Wire.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