Arduino library for the Microchip MCP23017 IO Expander

Keegan Morrow

Version 0.1.2

Tue Apr 18 2017

MCP23017

Arduino library for the Microchip **MCP23017** IO Expander

Hierarchical Index

Class Hierarchy

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

Class Index

Class List

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

File Index

File List

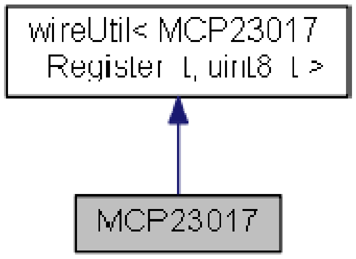
Here is a list of all files with brief descriptions:

Class Documentation

MCP23017 Class Reference

`#include <MCP23017.h>`

Inheritance diagram for MCP23017:



Public Member Functions

- **MCP23017** ()
- void **begin** ()
  Initialize the chip at the default address.
- uint8_t **addressIndex** (uint8_t a)
  Get the hardware address from the logical address of the chip.
- void **pinMode** (uint8_t, uint8_t)
  Set the characteristic of the IO pin.
- void **digitalWrite** (uint8_t, bool)
  Set the state of an output pin.
- bool **digitalRead** (uint8_t)
  Read the state of an input pin.
- void **portMode** (**MCP23017_Port_t**, uint8_t)
  Set the characteristic of a port.
- void **writePort** (**MCP23017_Port_t**, uint8_t)
  Write a byte to an output port.
- uint8_t **readPort** (**MCP23017_Port_t**)
  Read a byte from an input port.
- void **chipMode** (uint8_t)
  Set the characteristic of all pins on the chip.
- void **writeChip** (uint16_t)
  Write a word to the chip.
- uint16_t **readChip** ()
  Read a word from a chip.
- void **setInputPolarity** (bool)
  Sets the input polarity of the chip.
- void **setInputPolarity** (**MCP23017_Port_t**, bool)
  Set the input polarity of a port.
- void **setInputPolarity** (uint8_t, bool)
  Set the input polarity of an individual pin.

- uint8_t **getInterrupt** ()
  Get the pin that caused an interrupt.
- uint16_t **getInterruptCapture** ()
  Get a snapshot of all the input pins at the last interrupt.
- uint8_t **getInterruptCapture** (**MCP23017_Port_t**)
  Get a snapshot of the state of all the pins on a port at the last interrupt.
- void **setInterrupt** (uint8_t, bool)
  Set if an input pin will trigger an interrupt on change.
- void **setInterrupt** (**MCP23017_Port_t**, bool)
  Set interrupt enable on a port.
- void **setInterrupt** (uint16_t)
  Directly set the interrupt mask for the whole chip.
- void **interruptMirror** (bool)
  Set the interrupt pins to mirror each other.
- void **setIntPinMode** (**MCP23017_interruptPinMode_t**)
  Set the electrical characteristic of the interrupt pins.

Additional Inherited Members

Constructor & Destructor Documentation

MCP23017::MCP23017 ()`[inline]`

Member Function Documentation

uint8_t MCP23017::addressIndex (uint8_t *a*)`[inline]`

Get the hardware address from the logical address of the chip.

Parameters:

a	Logical address of the chip

Returns:

Hardware address of the chip

void MCP23017::begin ()`[inline]`, `[virtual]`

Initialize the chip at the default address.

Reimplemented from **wireUtil< MCP23017_Register_t, uint8_t >** (*p.12*).

Here is the call graph for this function:

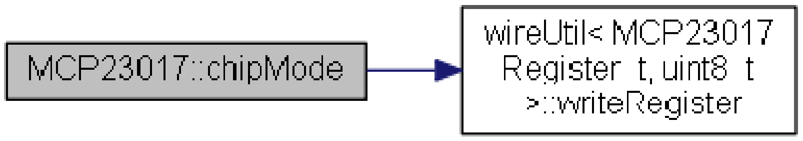
void MCP23017::chipMode (uint8_t *mode*)

Set the characteristic of all pins on the chip.

Parameters:

mode	IO type (INPUT, INPUT_PULLUP, OUTPUT)

Here is the call graph for this function:



bool MCP23017::digitalRead (uint8_t *pin*)

Read the state of an input pin.

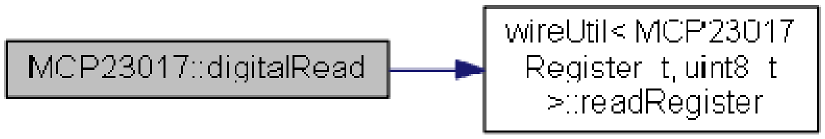
Parameters:

pin	Pin number

Returns:

State of the pin

Here is the call graph for this function:



void MCP23017::digitalWrite (uint8_t *pin*, bool *state*)

Set the state of an output pin.

Parameters:

pin	Pin number
state	State to set the pin (HIGH, LOW)

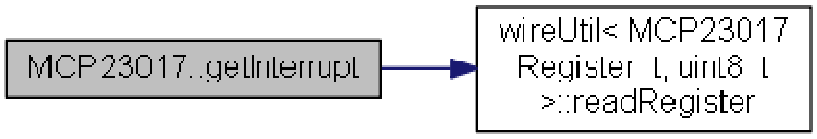
uint8_t MCP23017::getInterrupt ()

Get the pin that caused an interrupt.

Returns:

Pin number

Here is the call graph for this function:



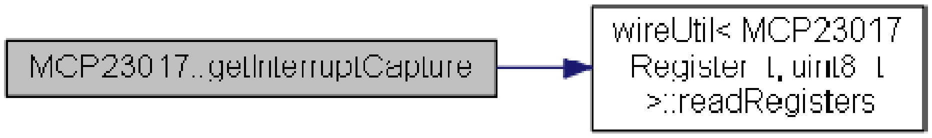
uint16_t MCP23017::getInterruptCapture ()

Get a snapshot of all the input pins at the last interrupt.

Returns:

Snapshot of the input registers

Here is the call graph for this function:


MCP23017::getInterruptCapture
wireUtil< MCP23017 Register_t, uint8_t >::readRegisters

uint8_t MCP23017::getInterruptCapture (MCP23017_Port_t *port*)

Get a snapshot of the state of all the pins on a port at the last interrupt.

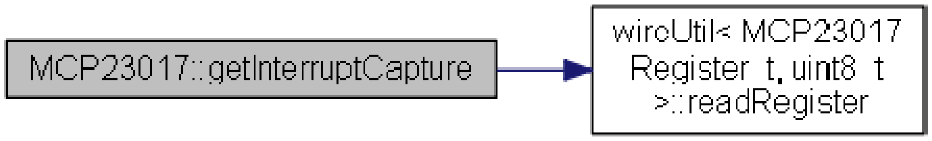
Parameters:

port	Port to get the snapshot from

Returns:

Snapshot of the port

Here is the call graph for this function:


MCP23017::getInterruptCapture
wireUtil< MCP23017 Register_t, uint8_t >::readRegister

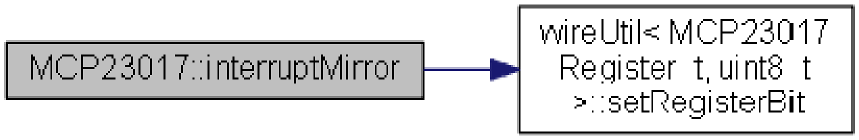
void MCP23017::interruptMirror (bool *state*)

Set the interrupt pins to mirror each other.

Parameters:

state	true = both int pins mirror, false = int by port

Here is the call graph for this function:


MCP23017::interruptMirror
wireUtil< MCP23017 Register_t, uint8_t >::setRegisterBit

void MCP23017::pinMode (uint8_t *pin*, uint8_t *mode*)

Set the characteristic of the IO pin.

Parameters:

pin	Pin number
mode	IO type (INPUT, INPUT_PULLUP, OUTPUT)

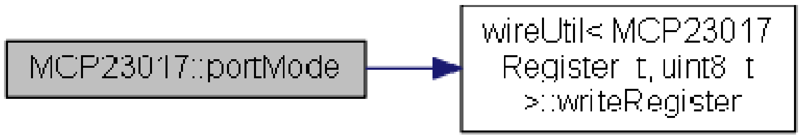
void MCP23017::portMode (MCP23017_Port_t *port*, uint8_t *mode*)

Set the characteristic of a port.

Parameters:

port	Port to set (A, B)
mode	IO type (INPUT, INPUT_PULLUP, OUTPUT)

Here is the call graph for this function:



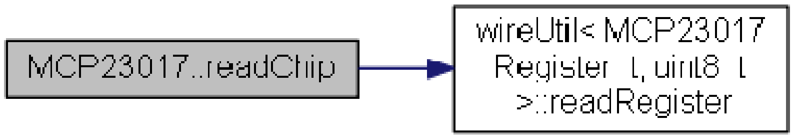
uint16_t MCP23017::readChip ()

Read a word from a chip.

Returns:

A word from the chip

Here is the call graph for this function:



uint8_t MCP23017::readPort (MCP23017_Port_t *port*)

Read a byte from an input port.

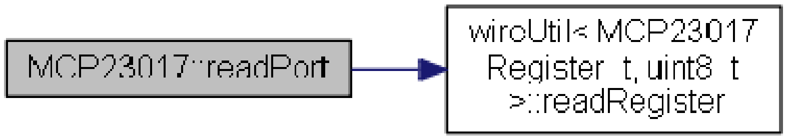
Parameters:

port	Port to read from (A, B)

Returns:

data from the port

Here is the call graph for this function:



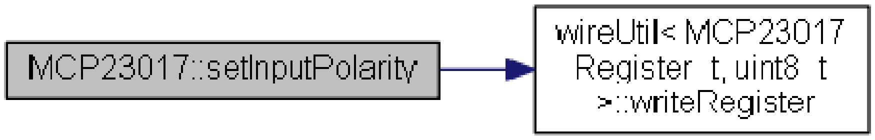
void MCP23017::setInputPolarity (bool *state*)

Sets the input polarity of the chip.

Parameters:

state	true = all input pins inverted

Here is the call graph for this function:


MCP23017::setInputPolarity
wireUtil< MCP23017 Register_t, uint8_t >::writeRegister

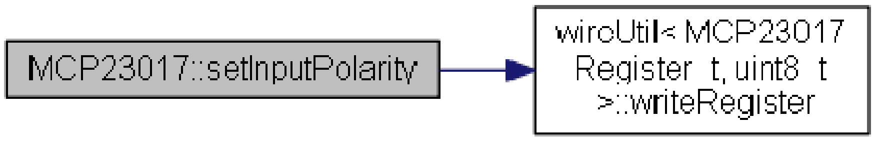
void MCP23017::setInputPolarity (MCP23017_Port_t *port*, bool *state*)

Set the input polarity of a port.

Parameters:

port	Port to set (A, B)
state	true = inverted

Here is the call graph for this function:


MCP23017::setInputPolarity
wireUtil< MCP23017 Register_t, uint8_t >::writeRegister

void MCP23017::setInputPolarity (uint8_t *pin*, bool *state*)

Set the input polarity of an individual pin.

Parameters:

pin	Pin to set
state	true = inverted

void MCP23017::setInterrupt (uint8_t *pin*, bool *state*)

Set if an input pin will trigger an interrupt on change.

Parameters:

pin	Pin to enable or disable interrupt
state	true = enable, false = disable

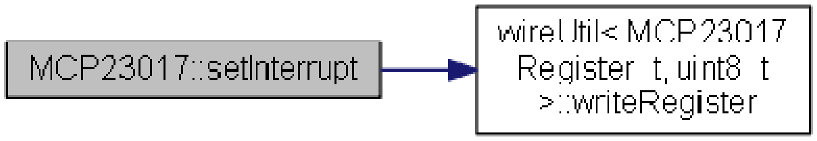
void MCP23017::setInterrupt (MCP23017_Port_t *port*, bool *state*)

Set interrupt enable on a port.

Parameters:

port	Port to set
state	

Here is the call graph for this function:



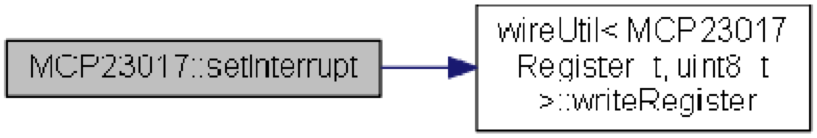
void MCP23017::setInterrupt (uint16_t *mask*)

Directly set the interrupt mask for the whole chip.

Parameters:

mask	16 bit mask 1's = enable

Here is the call graph for this function:



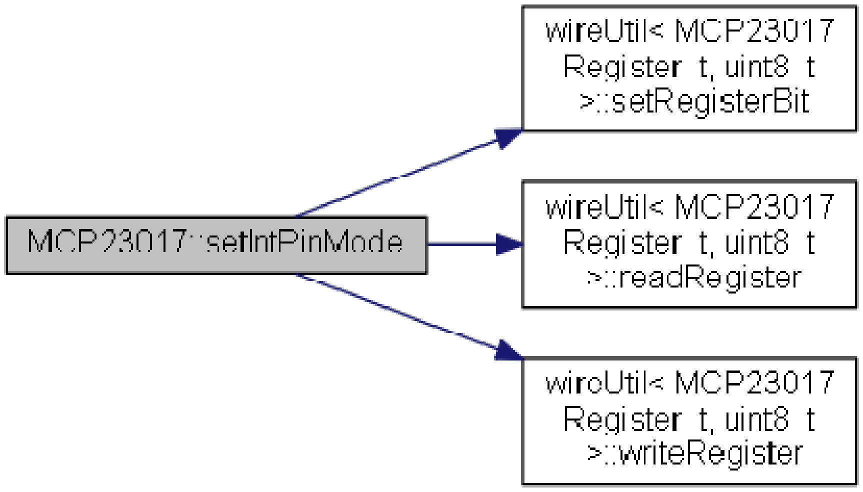
void MCP23017::setIntPinMode (MCP23017_interruptPinMode_t *interruptPinMode*)

Set the electrical characteristic of the interrupt pins.

Parameters:

interruptPinMode	(openDrain, lowOnInt, highOnInt)

Here is the call graph for this function:


MCP23017::setIntPinMode
wireUtil< MCP23017 Register_t, uint8_t >::setRegisterBit
wireUtil< MCP23017 Register_t, uint8_t >::readRegister
wireUtil< MCP23017 Register_t, uint8_t >::writeRegister

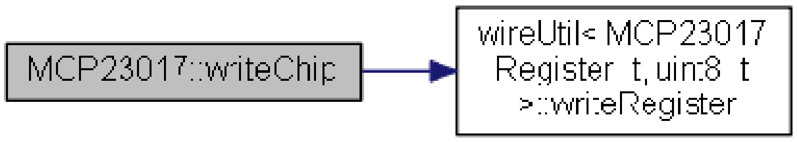
void MCP23017::writeChip (uint16_t *state*)

Write a word to the chip.

Parameters:

state	Word to write

Here is the call graph for this function:



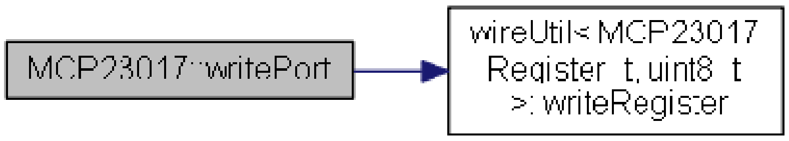
void MCP23017::writePort (MCP23017_Port_t *port*, uint8_t *state*)

Write a byte to an output port.

Parameters:

port	Port to output to (A, B)
state	Byte to write

Here is the call graph for this function:



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

- src/**MCP23017.h**
- src/**MCP23017.cpp**

wireUtil< REGTYPE, DATATYPE > Class Template Reference

Utility base class for reading and writing registers on i2c devices.
`#include <wireUtil.h>`

Public Member Functions

- void **attachTimeoutHandler** (void(*timeOutHandler)(void))
  Attach a function to be called on a read timeout.
- void **attachErrorHandler** (void(*errorHandler)(uint8_t))
  Attach a function to be called on a write NACK.
- bool **getTimeoutFlag** ()
  Safe method to read the state of the timeout flag.
- virtual void **begin** ()
  Initialize the chip at the default address (must be defined later)
- virtual void **begin** (uint8_t)

Initialize the chip at a specific address.

- bool **writeRegister** (REGTYPE, DATATYPE)
  Write a single register on an i2c device.
- bool **writeRegisters** (REGTYPE, DATATYPE *, uint8_t)
  Write to a sequence of registers on an i2c device.
- DATATYPE **readRegister** (REGTYPE)
  Read a single register from an i2c device.
- bool **readRegisters** (REGTYPE, DATATYPE *, uint8_t)
  Read a number of sequential registers from an i2c device.
- bool **setRegisterBit** (REGTYPE, uint8_t, bool)
  Read modify write a bit on a register.

Public Attributes

- unsigned long **timeoutTime**
  Amount of time to wait for a successful read.
- bool **timeoutFlag**
  Set to true if there is a timeout event, reset on the next read.

Protected Attributes

- uint8_t **address**
  Hardware address of the device.

Detailed Description

template<typename REGTYPE, typename DATATYPE = uint8_t>

class wireUtil< REGTYPE, DATATYPE >

Utility base class for reading and writing registers on i2c devices.

Template Parameters:

REGTYPE	An initialized enum type that lists the valid registers for the device
DATATYPE	Data type (register size) supports uint8_t, uint16_t, uint32_t

---

Member Function Documentation

template<typename REGTYPE, typename DATATYPE = uint8_t> void wireUtil< REGTYPE, DATATYPE >::attachErrorHandler (void(*)(uint8_t) *errorHandler*)[inline]

Attach a function to be called on a write NACK.

Parameters:

errorHandler	Pointer to a 'void f(uint8_t)' function. This will be passed the Wire status.

template<typename REGTYPE, typename DATATYPE = uint8_t> void wireUtil< REGTYPE, DATATYPE >::attachTimeoutHandler (void(*)(void) *timeOutHandler*)`[inline]`

Attach a function to be called on a read timeout.

Parameters:

timeOutHandler	Pointer to a 'void f(void)' function

template<typename REGTYPE, typename DATATYPE = uint8_t> virtual void wireUtil< REGTYPE, DATATYPE >::begin ()`[virtual]`

Initialize the chip at the default address (must be defined later)

Reimplemented in **MCP23017** (*p.4*).

template<typename REGTYPE , typename DATATYPE > void wireUtil< REGTYPE, DATATYPE >::begin (uint8_t *address*)`[virtual]`

Initialize the chip at a specific address.

Parameters:

address	Address of the chip

template<typename REGTYPE, typename DATATYPE = uint8_t> bool wireUtil< REGTYPE, DATATYPE >::getTimeoutFlag ()`[inline]`

Safe method to read the state of the timeout flag.

Returns:

State of the timeout flag

template<typename REGTYPE, typename DATATYPE > DATATYPE wireUtil< REGTYPE, DATATYPE >::readRegister (REGTYPE *reg*)

Read a single register from an i2c device.

Parameters:

reg	Register address (from a device specific enum)

Returns:

Data from the device register, 0 if there is a timeout

template<typename REGTYPE, typename DATATYPE> bool wireUtil< REGTYPE, DATATYPE >::readRegisters (REGTYPE *reg*, DATATYPE * *buffer*, uint8_t *len*)

Read a number of sequential registers from an i2c device.

Parameters:

reg	First register address (from a device specific enum)
buffer	Array to contain the data read
len	Number of bytes to read

Returns:

true on success, false on timeout

template<typename REGTYPE, typename DATATYPE > bool wireUtil< REGTYPE, DATATYPE >::setRegisterBit (REGTYPE *reg*, uint8_t *bit*, bool *state*)

Read modify write a bit on a register.

Parameters:

reg	register to modify
bit	index of the bit to set
state	state of the bit to set

Returns:

true on success

template<typename REGTYPE, typename DATATYPE> bool wireUtil< REGTYPE, DATATYPE >::writeRegister (REGTYPE *reg*, DATATYPE *data*)

Write a single register on an i2c device.

Parameters:

reg	Register address (from a device specific enum)
data	Data to be written to the device

Returns:

true on success, false if NACK

template<typename REGTYPE, typename DATATYPE> bool wireUtil< REGTYPE, DATATYPE >::writeRegisters (REGTYPE *reg*, DATATYPE * *buffer*, uint8_t *len*)

Write to a sequence of registers on an i2c device.

Parameters:

reg	First register address (from a device specific enum)
buffer	Array containing the data to be written
len	Number of bytes in the array

Returns:

true on success, false if NACK

Member Data Documentation

template<typename REGTYPE, typename DATATYPE = uint8_t> uint8_t wireUtil< REGTYPE, DATATYPE >::address`[protected]`

Hardware address of the device.

template<typename REGTYPE, typename DATATYPE = uint8_t> bool wireUtil< REGTYPE, DATATYPE >::timeoutFlag

Set to true if there is a timeout event, reset on the next read.

template<typename REGTYPE, typename DATATYPE = uint8_t> unsigned long wireUtil< REGTYPE, DATATYPE >::timeoutTime

Amount of time to wait for a successful read.

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

- src/utility/**wireUtil.h**

File Documentation

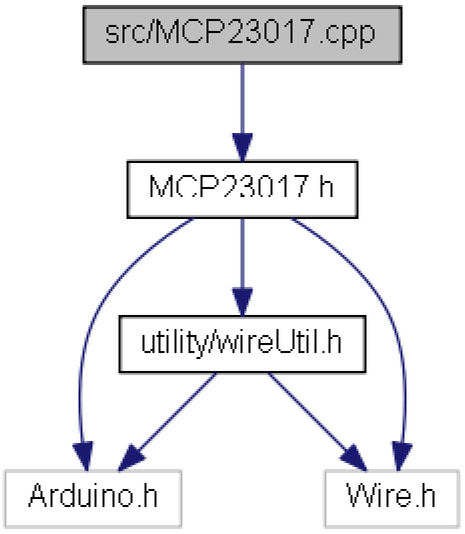
README.md File Reference

---

src/MCP23017.cpp File Reference

`#include "MCP23017.h"`

Include dependency graph for MCP23017.cpp:

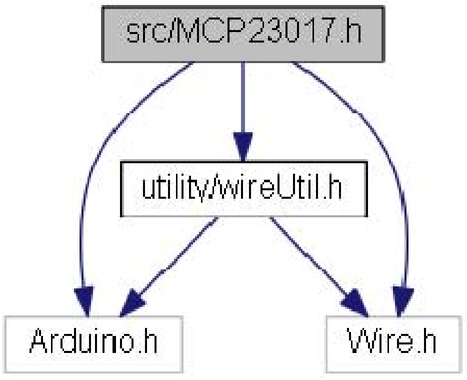


src/MCP23017.h File Reference

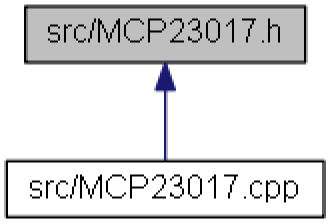
Arduino library for the Microchip **MCP23017** IO Expander.

```
#include <Arduino.h>
#include <Wire.h>
#include "utility/wireUtil.h"
```

Include dependency graph for MCP23017.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **MCP23017**

Enumerations

- enum **MCP23017_Register_t** { **IODIRA_r** = 0x00, **IODIRB_r** = 0x01, **IPOLA_r** = 0x02, **IPOLB_r** = 0x03, **GPINTENA_r** = 0x04, **GPINTENB_r** = 0x05, **DEFVALA_r** = 0x06, **DEFVALB_r** = 0x07, **INTCONA_r** = 0x08, **INTCONB_r** = 0x09, **IOCONA_r** = 0x0A, **IOCONB_r** = 0x0B, **GPPUA_r** = 0x0C, **GPPUB_r** = 0x0D, **INTFA_r** = 0x0E, **INTFB_r** = 0x0F, **INTCAPA_r** = 0x10, **INTCAPB_r** = 0x11, **GPIOA_r** = 0x12, **GPIOB_r** = 0x13, **OLATA_r** = 0x14, **OLATB_r** = 0x15 }
- enum **MCP23017_RegisterGeneric_t** { **IODIR_r** = 0x00, **IPOL_r** = 0x02, **GPINTEN_r** = 0x04, **DEFVAL_r** = 0x06, **INTCON_r** = 0x08, **IOCON_r** = 0x0A, **GPPU_r** = 0x0C, **INTF_r** = 0x0E, **INTCAP_r** = 0x10, **GPIO_r** = 0x12, **OLAT_r** = 0x14 }
- enum **MCP23017_Port_t** { **PORT_A** = 0x00, **PORT_B** = 0x01 }
- enum **MCP23017_interruptPinMode_t** { **openDrain**, **lowOnInt**, **highOnInt** }

Detailed Description

Arduino library for the Microchip **MCP23017** IO Expander.

Author:

Keegan Morrow

Version:

0.1.2

Enumeration Type Documentation

enum MCP23017_interruptPinMode_t

Enumerator:

openDrain	
lowOnInt	
highOnInt	

enum MCP23017_Port_t

Enumerator:

PORT_A	
PORT_B	

enum MCP23017_Register_t

Enumerator:

IODIRA_r	
IODIRB_r	
IPOLA_r	
IPOLB_r	

GPINTENA_r	
GPINTENB_r	
DEFVALA_r	
DEFVALB_r	
INTCONA_r	
INTCONB_r	
IOCONA_r	
IOCONB_r	
GPPUA_r	
GPPUB_r	
INTFA_r	
INTFB_r	
INTCAPA_r	
INTCAPB_r	
GPIOA_r	
GPIOB_r	
OLATA_r	
OLATB_r	

enum MCP23017_RegisterGeneric_t

Enumerator:

IODIR_r	
IPOL_r	
GPINTEN_r	
DEFVAL_r	
INTCON_r	
IOCON_r	
GPPU_r	
INTF_r	
INTCAP_r	
GPIO_r	
OLAT_r	

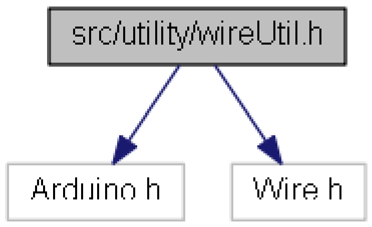
---

src/utility/wireUtil.h File Reference

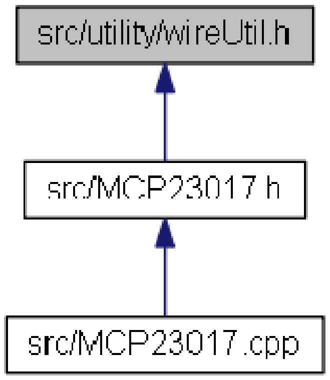
Utility base class for reading and writing registers on i2c devices.

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

Include dependency graph for wireUtil.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **wireUtil< REGTYPE, DATATYPE >**

Utility base class for reading and writing registers on i2c devices.

Detailed Description

Utility base class for reading and writing registers on i2c devices.

Author:

Keegan Morrow

Version:

1.1.2