

MCP23008-I2C

Generated by Doxygen 1.9.1



<b>1 MCP23008-I2C Library</b>	<b>1</b>
1.1 Contents	1
1.2 Library Documentation	1
1.3 Library Usage	1
1.3.1 Controllers	1
1.3.2 Usage the MCP23008-I2C library in the Code	2
1.4 License	2
1.5 Helpful Links	2
<b>2 Namespace Index</b>	<b>3</b>
2.1 Namespace List	3
<b>3 Class Index</b>	<b>5</b>
3.1 Class List	5
<b>4 File Index</b>	<b>7</b>
4.1 File List	7
<b>5 Namespace Documentation</b>	<b>9</b>
5.1 MCP23008_Constants Namespace Reference	9
5.1.1 Detailed Description	10
5.1.2 Variable Documentation	10
5.1.2.1 MCP23008_IOCON_DISSLW	10
5.1.2.2 MCP23008_IOCON_INTPOL	10
5.1.2.3 MCP23008_IOCON_ODR	10
5.1.2.4 MCP23008_IOCON_SEQOP	10
5.2 MCP23008_I2C Namespace Reference	10
5.2.1 Detailed Description	11
<b>6 Class Documentation</b>	<b>13</b>
6.1 MCP23008_I2C::MCP23008 Class Reference	13
6.1.1 Detailed Description	14
6.1.2 Constructor & Destructor Documentation	14
6.1.2.1 MCP23008()	14
6.1.3 Member Function Documentation	15
6.1.3.1 begin()	15
6.1.3.2 disableInterrupt()	15
6.1.3.3 getAddress()	16
6.1.3.4 getInterruptPolarity()	16
6.1.3.5 getPinMode8()	16
6.1.3.6 getPolarity()	17
6.1.3.7 getPolarity8()	17
6.1.3.8 getPullup()	17
6.1.3.9 getPullup8()	18

---

6.1.3.10 isConnected()	18
6.1.3.11 read1()	19
6.1.3.12 read8()	19
6.1.3.13 readInterruptCaptureRegister()	20
6.1.3.14 readInterruptFlagRegister()	20
6.1.3.15 setInterrupt()	20
6.1.3.16 setInterruptPolarity()	21
6.1.3.17 setPinMode1()	22
6.1.3.18 setPinMode8()	22
6.1.3.19 setPolarity()	23
6.1.3.20 setPolarity8()	23
6.1.3.21 setPullup()	24
6.1.3.22 setPullup8()	24
6.1.3.23 write1()	25
6.1.3.24 write8()	25
<b>7 File Documentation</b>	<b>29</b>
7.1 src/MCP23008-Constants.h File Reference	29
7.1.1 Detailed Description	30
7.2 src/MCP23008-I2C.cpp File Reference	30
7.2.1 Detailed Description	30
7.3 src/MCP23008-I2C.h File Reference	31
7.3.1 Detailed Description	31
<b>Index</b>	<b>33</b>

# Chapter 1

## MCP23008-I2C Library

Arduino Library for MCP23008, a 8-port GPIO exander

### 1.1 Contents

- [Library Documentation](#)
- [Library Usage](#)
- [License](#)
- [Helpful Links](#)

### 1.2 Library Documentation

The library documentation is mainly placed in the following pdf document [refman.pdf](#) or located under the following github pages [github.io](#).  
Additionally in combination with the technical datasheet of microchip [MCP23008-Datasheet](#).

### 1.3 Library Usage

#### 1.3.1 Controllers

The library is intended to be used on each microcontroller for Example:

- Arnuino Nano
- Arduino Nano 33 IOT
- ESP8266
- ESP32
- etc ...

### 1.3.2 Usage the MCP23008-I2C library in the Code

Include the library in you project via:

```
#include <MCP23008-I2C.h>
```

Instance an new MCP23008 object by:

```
MCP23008_I2C::MCP23008 mcp{0x20};
```

Now you ca use the object and his members as normal like:

```
mcp.begin();
```

Please refer to the examples and the above mentioned documentation files.

## 1.4 License

This library is licensed under MIT Licence.

[MCP23008-I2C License](#)

## 1.5 Helpful Links

- [ESP8266-01-Adapter](#)

## Chapter 2

# Namespace Index

### 2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

<a href="#">MCP23008_Constants</a>	
Namespace of MCP23008 Constants. Contains mainly the register description . . . . .	9
<a href="#">MCP23008_I2C</a>	
Namespace of <a href="#">MCP23008</a> . includes class declaration, errors, and states . . . . .	10





## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">MCP23008_I2C::MCP23008</a>	
Class <a href="#">MCP23008</a>	13



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all documented files with brief descriptions:

src/ <a href="#">MCP23008-Constants.h</a>	
MCP23008 Constants and Register short Descriptions . . . . .	29
src/ <a href="#">MCP23008-I2C.cpp</a>	
MCP23008 Function and Class Definitions . . . . .	30
src/ <a href="#">MCP23008-I2C.h</a>	
MCP23008 Declarations . . . . .	31



## Chapter 5

# Namespace Documentation

### 5.1 MCP23008\_Constants Namespace Reference

Namespace of MCP23008 Constants. Contains mainly the register description.

#### Variables

- `constexpr uint8_t MCP23008_IODIR_REG {0x00}`  
*I/O Direction Register Address (IODIR)*
- `constexpr uint8_t MCP23008_IPOL_REG {0x01}`  
*Input Polarity Register (IPOL)*
- `constexpr uint8_t MCP23008_GPINTEN_REG {0x02}`  
*Interrupt-On-Change Control Register (GPINTEN)*
- `constexpr uint8_t MCP23008_DEFVAL_REG {0x03}`  
*Default Compare Register for Interrupt-On-Change.*
- `constexpr uint8_t MCP23008_INTCON_REG {0x04}`  
*Interrupt Control Register (INTCON)*
- `constexpr uint8_t MCP23008_IOCON_REG {0x05}`  
*Configuration Register (IOCON)*
- `constexpr uint8_t MCP23008_GPPU_REG {0x06}`  
*Pull-Up Resistor Configuration Register (GPPU)*
- `constexpr uint8_t MCP23008_INTF_REG {0x07}`  
*Interrupt Flag Register (INTF)*
- `constexpr uint8_t MCP23008_INTCAP_REG {0x08}`  
*Interrupt Capture Register (INTCAP)*
- `constexpr uint8_t MCP23008_GPIO_REG {0x09}`  
*Port Register (GPIO)*
- `constexpr uint8_t MCP23008_OLAT_REG {0x0A}`  
*Output Latch Register (OLAT)*
- `constexpr uint8_t MCP23008_IOCON_SEQOP {0x20}`  
*The Sequential Operation (SEQOP) bit.*
- `constexpr uint8_t MCP23008_IOCON_DISSLW {0x10}`  
*Slew Rate control bit for SDA output.*
- `constexpr uint8_t MCP23008_IOCON_ODR {0x04}`  
*The Open-Drain control bit (ODR)*
- `constexpr uint8_t MCP23008_IOCON_INTPOL {0x02}`  
*The Input Polarity Control bit (INTPOL)*

### 5.1.1 Detailed Description

Namespace of MCP23008 Constants. Contains mainly the register description.

### 5.1.2 Variable Documentation

#### 5.1.2.1 MCP23008\_IOCON\_DISSLW

```
constexpr uint8_t MCP23008_Constants::MCP23008_IOCON_DISSLW {0x10} [constexpr]
```

Slew Rate control bit for SDA output.

The Slew Rate (DISSLW) bit controls the slew rate function on the SDA pin. If enabled, the SDA slew rate will be controlled when driving from a high to a low

#### 5.1.2.2 MCP23008\_IOCON\_INTPOL

```
constexpr uint8_t MCP23008_Constants::MCP23008_IOCON_INTPOL {0x02} [constexpr]
```

The Input Polarity Control bit (INTPOL)

The Interrupt Polarity (INTPOL) control bit sets the polarity of the INT pin. This bit is functional only when the ODR bit is cleared, configuring the INT pin as active push-pull.

#### 5.1.2.3 MCP23008\_IOCON\_ODR

```
constexpr uint8_t MCP23008_Constants::MCP23008_IOCON_ODR {0x04} [constexpr]
```

The Open-Drain control bit (ODR)

The Open-Drain (ODR) control bit enables/disables the INT pin for open-drain configuration

#### 5.1.2.4 MCP23008\_IOCON\_SEQOP

```
constexpr uint8_t MCP23008_Constants::MCP23008_IOCON_SEQOP {0x20} [constexpr]
```

The Sequential Operation (SEQOP) bit.

The Sequential Operation (SEQOP) controls the incrementing function of the Address Pointer. If the Address Pointer is disabled, the Address Pointer does not automatically increment after each byte is clocked during a serial transfer. This feature is useful when it is desired to continuously poll (read) or modify (write) a register.

## 5.2 MCP23008\_I2C Namespace Reference

namespace of [MCP23008](#). includes class declaration, errors, and states

## Classes

- class [MCP23008](#)  
*Class [MCP23008](#).*

## Variables

- constexpr const char \* **MCP23008\_LIB\_VERSION** {"1.0.0"}
- constexpr int8\_t [MCP23008\\_STATE\\_OK](#) {0x00}  
*constant which states all ok, no error*
- constexpr int8\_t [MCP23008\\_ERROR\\_PIN](#) {-1}  
*constant which states a wrong pin number was used*
- constexpr int8\_t [MCP23008\\_ERROR\\_I2C](#) {-2}  
*constant which states an error during I2C communication*
- constexpr int8\_t [MCP23008\\_ERROR\\_VALUE](#) {-3}  
*constant which states that there was an error regarding a parameter value*

### 5.2.1 Detailed Description

namespace of [MCP23008](#). includes class declaration, errors, and states





## Chapter 6

# Class Documentation

### 6.1 MCP23008\_I2C::MCP23008 Class Reference

Class [MCP23008](#).

```
#include <MCP23008-I2C.h>
```

#### Public Member Functions

- [MCP23008](#) (uint8\_t address=0x20, TwoWire \*wire=&Wire)  
*Construct a new [MCP23008](#) object.*
- int8\_t [begin](#) (bool inputPullUp=true) const  
*init [MCP23008](#) instance*
- int8\_t [isConnected](#) () const  
*check connection status*
- uint8\_t [getAddress](#) () const  
*Get the address of device.*
- int [setPinMode1](#) (uint8\_t pin, uint8\_t mode) const  
*set pinMode of a single pin (IODIR)*
- int [write1](#) (uint8\_t pin, uint8\_t value) const  
*write value for a single pin to OLAT register (OLAT)*
- int [read1](#) (uint8\_t pin) const  
*read value for a single pin from GPIO register (GPIO)*
- int [setPolarity](#) (uint8\_t pin, bool reversed) const  
*Set the polarity of a single pin in the Input polarity register (IPOL)*
- int [getPolarity](#) (uint8\_t pin) const  
*Get the polarity of a single pin of Input polarity register (IPOL)*
- int [setPullup](#) (uint8\_t pin, bool pullup) const  
*Set the Pull-up register for on pin (GPPU)*
- int [getPullup](#) (uint8\_t pin) const  
*Get the Pull-up register for one pin (GPPU)*
- int8\_t [setPinMode8](#) (uint8\_t mask) const  
*set mask for pinMode in I/O register for all pins at once (INTCON)*
- int [getPinMode8](#) () const  
*Read I/O Direction register (IODIR)*

- `int8_t write8 (uint8_t value) const`  
*write 8-bit value at once to Output Latch register (OLAT)*
- `int read8 () const`  
*read 8 bit at once from GPIO register (GPIO)*
- `int8_t setPolarity8 (uint8_t mask) const`  
*Set the polarity in 8-bit at once in Input polarity register (IPOL) If a bit is set, the corresponding GPIO register bit will reflect the inverted value on the pin.*
- `int getPolarity8 () const`  
*Get the polarity in 8-bit at once in Input polarity register (IPOL) If a bit is set, the corresponding GPIO register bit will reflect the inverted value on the pin.*
- `int8_t setPullup8 (uint8_t mask) const`  
*Set Pull-up for all 8 pins at once (GPPU)*
- `int getPullup8 () const`  
*Get Pull-up for all 8 pins at once (GPPU)*
- `int setInterrupt (uint8_t pin, uint8_t mode) const`  
*Set the Interrupt Control Register for specified pin (INTCON)*
- `int disableInterrupt (uint8_t pin) const`  
*Disable interrupt on specified pin (INTCON)*
- `int readInterruptFlagRegister () const`  
*Read the Interrupt Flag Register (INTF)*
- `int readInterruptCaptureRegister () const`  
*Read the interrupt capture register (INTCAP)*
- `int setInterruptPolarity (uint8_t polarity) const`  
*Set the Interrupt Polarity in IOCON Register.*
- `int getInterruptPolarity () const`  
*Read the Interrupt Polarity.*

### 6.1.1 Detailed Description

Class [MCP23008](#).

### 6.1.2 Constructor & Destructor Documentation

#### 6.1.2.1 MCP23008()

```
MCP23008::MCP23008 (
    uint8_t address = 0x20,
    TwoWire * wire = &Wire )
```

Construct a new [MCP23008](#) object.

#### Parameters

<i>address</i>	optional address of I2C device; default = 0x20;
<i>wire</i>	optional address of Wire instance; default = &Wire;

## 6.1.3 Member Function Documentation

### 6.1.3.1 begin()

```
int8_t MCP23008::begin (
    bool inputPullUp = true ) const
```

init [MCP23008](#) instance

Check connection status and set Pull-up resistors if needed (by default).

#### Parameters

<i>inputPullUp</i>	optional force all inputs with Pull-up; default = true;
--------------------	---

#### Returns

status of begin

#### Return values

0	state OK
<0	error code

### 6.1.3.2 disableInterrupt()

```
int MCP23008::disableInterrupt (
    uint8_t pin ) const
```

Disable interrupt on specified pin (INTCON)

#### Parameters

<i>pin</i>	number of pin to clear the interrupt (0...7)
------------	--

#### Returns

int status

#### Return values

0	state OK
<0	error code

### 6.1.3.3 getAddress()

```
uint8_t MCP23008_I2C::MCP23008::getAddress ( ) const [inline]
```

Get the address of device.

#### Returns

uint8\_t address

### 6.1.3.4 getInterruptPolarity()

```
int MCP23008::getInterruptPolarity ( ) const
```

Read the Interrupt Polarity.

#### Returns

int status

#### Return values

2	Opden-drain (ODR)
1	active-high
0	active-low
<0	error code

### 6.1.3.5 getPinMode8()

```
int MCP23008::getPinMode8 ( ) const
```

Read I/O Direction register (IODIR)

#### Returns

state of I/O direction register

#### Return values

$\geq 0$	register value
<0	error code

### 6.1.3.6 getPolarity()

```
int MCP23008::getPolarity (
    uint8_t pin ) const
```

Get the polarity of a single pin of Input polarity register (IPOL)

If a bit is set, the corresponding GPIO register bit will reflect the inverted value on the pin.

#### Parameters

<i>pin</i>	pin number of pin (0...7)
------------	---------------------------

#### Returns

int status of polatity for pin

#### Return values

0	noninverted => GPIO pin will reflect the same logic state on input pin
1	inverted => GPIO pin will reflect the opposite logic state on input pin
<0	error code

### 6.1.3.7 getPolarity8()

```
int MCP23008::getPolarity8 ( ) const
```

Get the polarity in 8-bit at once in Input polarity register (IPOL) If a bit is set, the corresponding GPIO register bit will reflect the inverted value on the pin.

#### Returns

int status

#### Return values

>=0	register value
<0	error code

### 6.1.3.8 getPullup()

```
int MCP23008::getPullup (
```

```
uint8_t pin ) const
```

Get the Pull-up register for one pin (GPPU)

If a bit is set and the corresponding pin is configured as an input, the corresponding PORT pin is internally pulled up with a 100 kOhm resistor.

#### Parameters

<i>pin</i>	pin number of pin (0...7)
------------	---------------------------

#### Returns

int status of Pull-up for pin

#### Return values

0	Pull-up disabled
1	Pull-up enabled
<0	error code

### 6.1.3.9 getPullup8()

```
int MCP23008::getPullup8 ( ) const
```

Get Pull-up for all 8 pins at once (GPPU)

If a bit is set and the corresponding pin is configured as an input, the corresponding PORT pin is internally pulled up with a 100 kOhm resistor.

#### Returns

int status

#### Return values

>=0	register value
<0	error code

### 6.1.3.10 isConnected()

```
int8_t MCP23008::isConnected ( ) const
```

check connection status

**Returns**

int8\_t status of connection

**Return values**

1	connection OK
<0	error code

**6.1.3.11 read1()**

```
int MCP23008::read1 (
    uint8_t pin ) const
```

read value for a single pin from GPIO register (GPIO)

The GPIO register reflects the value on the port. Reading from this register reads the port.

**Parameters**

<i>pin</i>	pin number of pin 0...7
------------	-------------------------

**Returns**

int status of gpio register for pin

**Return values**

0	pin is in LOW state
1	pin is in HIGH
<0	error code

**6.1.3.12 read8()**

```
int MCP23008::read8 ( ) const
```

read 8 bit at once from GPIO register (GPIO)

The GPIO register reflects the value on the port. Reading from this register reads the port.

**Returns**

int status value of GPIO register

**Return values**

$\geq 0$	register value
$< 0$	error code

**6.1.3.13 readInterruptCaptureRegister()**

```
int MCP23008::readInterruptCaptureRegister ( ) const
```

Read the interrupt capture register (INTCAP)

**Returns**

int read state of interrupt capture register

**Return values**

$\geq 0$	register value
$< 0$	error code

**6.1.3.14 readInterruptFlagRegister()**

```
int MCP23008::readInterruptFlagRegister ( ) const
```

Read the Interrupt Flag Register (INTF)

The INTF register reflects the interrupt condition on the PORT pins of any pin that is enabled for interrupts via the GPINTEN register. A 'set' bit indicates that the associated pin caused the interrupt.

**Returns**

int read state of interrupt flag register

**Return values**

$\geq 0$	register value
$< 0$	error code

**6.1.3.15 setInterrupt()**

```
int MCP23008::setInterrupt (
```



```
uint8_t pin,
uint8_t mode ) const
```

Set the Interrupt Control Register for specified pin (INTCON)

If a bit is set, the corresponding I/O pin is compared against the associated bit in the DEFVAL register. If a bit value is clear, the corresponding I/O pin is compared against the previous value.

#### Parameters

<i>pin</i>	number of pin to set the interrupt (0...7)
<i>mode</i>	mode of interrupt (RISING, FALLING, CHANGE)

#### Returns

int status

#### Return values

0	state OK
<0	error code

#### 6.1.3.16 setInterruptPolarity()

```
int MCP23008::setInterruptPolarity (
    uint8_t polarity ) const
```

Set the Interrupt Polarity in IOCON Register.

The Interrupt Polarity (INTPOL) control bit sets the polarity of the INT pin. This bit is functional only when the ODR bit is cleared, configuring the INT pin as active push-pull.

2 = Open-drain Output (ODR) 1 = active-high 0 = active-low

#### Parameters

<i>polarity</i>	value (2...0)
-----------------	---------------

#### Returns

int status

#### Return values

0	state OK
<0	error code

**6.1.3.17 setPinMode1()**

```
int MCP23008::setPinMode1 (
    uint8_t pin,
    uint8_t mode ) const
```

set pinMode of a single pin (IODIR)

**Parameters**

<i>pin</i>	pin number of pin 0...7
<i>mode</i>	mode of pin (INPUT, INPUT_PULLUP, OUTPUT)

**Returns**

int status of write in IODIR register

**Return values**

0	state OK
<0	error code

**6.1.3.18 setPinMode8()**

```
int8_t MCP23008::setPinMode8 (
    uint8_t mask ) const
```

set mask for pinMode in I/O register for all pins at once (INTCON)

1 = Pin is configured as an input

0 = Pin is configured as an output

Bit pattern to set in hex: 0x10

in binary: 0b00010000

in decimal: 16

**Parameters**

<i>mask</i>	bit mask to set
-------------	-----------------

**Returns**

int status of write to I/O Register

## Return values

0	state OK
<0	error code

**6.1.3.19 setPolarity()**

```
int MCP23008::setPolarity (
    uint8_t pin,
    bool reversed ) const
```

Set the polarity of a single pin in the Input polarity register (IPOL)

If a bit is set, the corresponding GPIO register bit will reflect the inverted value on the pin.

## Parameters

<i>pin</i>	pin number of pin 0...7
<i>reversed</i>	true or false

## Returns

int status

## Return values

0	state OK
<0	error code

**6.1.3.20 setPolarity8()**

```
int8_t MCP23008::setPolarity8 (
    uint8_t mask ) const
```

Set the polarity in 8-bit at once in Input polarity register (IPOL) If a bit is set, the corresponding GPIO register bit will reflect the inverted value on the pin.

## Parameters

<i>mask</i>	to write
-------------	----------

Bit pattern to set in hex: 0x10

in binary: 0b00010000

in decimal: 16

**Returns**

int status

**Return values**

0	state OK
<0	error code

**6.1.3.21 setPullup()**

```
int MCP23008::setPullup (
    uint8_t pin,
    bool pullup ) const
```

Set the Pull-up register for on pin (GPPU)

If a bit is set and the corresponding pin is configured as an input, the corresponding PORT pin is internally pulled up with a 100 kOhm resistor.

**Parameters**

<i>pin</i>	pin number of pin 0...7
<i>pullup</i>	set Pull-up true/false

**Returns**

int status

**Return values**

0	state OK
<0	error code

**6.1.3.22 setPullup8()**

```
int8_t MCP23008::setPullup8 (
    uint8_t mask ) const
```

Set Pull-up for all 8 pins at once (GPPU)

If a bit is set and the corresponding pin is configured as an input, the corresponding PORT pin is internally pulled up with a 100 kOhm resistor.

**Parameters**

<i>mask</i>	mask for Pull-up to set
-------------	-------------------------

**Returns**

int status

**Return values**

0	state OK
<0	error code

**6.1.3.23 write1()**

```
int MCP23008::write1 (
    uint8_t pin,
    uint8_t value ) const
```

write value for a single pin to OLAT register (OLAT)

The OLAT register provides access to the output latches. A write to this register modifies the output latches that modify the pins configured as outputs.

**Parameters**

<i>pin</i>	pin number of pin 0...7
<i>value</i>	to write 0/1

**Returns**

int status of write in olat register

**Return values**

0	state OK
<0	error code

**6.1.3.24 write8()**

```
int8_t MCP23008::write8 (
    uint8_t value ) const
```

write 8-bit value at once to Output Latch register (OLAT)

The OLAT register provides access to the output latches. A read from this register results in a read of the OLAT and not the port itself. A write to this register modifies the output latches that modify the pins configured as outputs.

**Parameters**

<i>value</i>	value to write in hex, bin or decimal
--------------	---------------------------------------

**Returns**

int status of write to Output Latch register

**Return values**

0	state OK
<0	error code

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

- [src/MCP23008-I2C.h](#)
- [src/MCP23008-I2C.cpp](#)





## Chapter 7

# File Documentation

### 7.1 src/MCP23008-Constants.h File Reference

MCP23008 Constants and Register short Descriptions.

#### Namespaces

- [MCP23008\\_Constants](#)

*Namespace of MCP23008 Constants. Contains mainly the register description.*

#### Variables

- `constexpr uint8_t MCP23008_Constants::MCP23008_IODIR_REG {0x00}`  
*I/O Direction Register Address (IODIR)*
- `constexpr uint8_t MCP23008_Constants::MCP23008_IPOL_REG {0x01}`  
*Input Polarity Register (IPOL)*
- `constexpr uint8_t MCP23008_Constants::MCP23008_GPINTEN_REG {0x02}`  
*Interrupt-On-Change Control Register (GPINTEN)*
- `constexpr uint8_t MCP23008_Constants::MCP23008_DEFVAL_REG {0x03}`  
*Default Compare Register for Interrupt-On-Change.*
- `constexpr uint8_t MCP23008_Constants::MCP23008_INTCON_REG {0x04}`  
*Interrupt Control Register (INTCON)*
- `constexpr uint8_t MCP23008_Constants::MCP23008_IOCON_REG {0x05}`  
*Configuration Register (IOCON)*
- `constexpr uint8_t MCP23008_Constants::MCP23008_GPPU_REG {0x06}`  
*Pull-Up Resistor Configuration Register (GPPU)*
- `constexpr uint8_t MCP23008_Constants::MCP23008_INTF_REG {0x07}`  
*Interrupt Flag Register (INTF)*
- `constexpr uint8_t MCP23008_Constants::MCP23008_INTCAP_REG {0x08}`  
*Interrupt Capture Register (INTCAP)*
- `constexpr uint8_t MCP23008_Constants::MCP23008_GPIO_REG {0x09}`  
*Port Register (GPIO)*
- `constexpr uint8_t MCP23008_Constants::MCP23008_OLAT_REG {0x0A}`  
*Output Latch Register (OLAT)*

- constexpr uint8\_t [MCP23008\\_Constants::MCP23008\\_IOCON\\_SEQOP](#) {0x20}  
*The Sequential Operation (SEQOP) bit.*
- constexpr uint8\_t [MCP23008\\_Constants::MCP23008\\_IOCON\\_DISSLW](#) {0x10}  
*Slew Rate control bit for SDA output.*
- constexpr uint8\_t [MCP23008\\_Constants::MCP23008\\_IOCON\\_ODR](#) {0x04}  
*The Open-Drain control bit (ODR)*
- constexpr uint8\_t [MCP23008\\_Constants::MCP23008\\_IOCON\\_INTPOL](#) {0x02}  
*The Input Polarity Control bit (INTPOL)*

### 7.1.1 Detailed Description

MCP23008 Constants and Register short Descriptions.

#### Author

Frank Häfele

#### Date

27.12.2024

#### Version

1.0.0

#### See also

<https://github.com/hasenradball/MCP23008-I2C>

## 7.2 src/MCP23008-I2C.cpp File Reference

MCP23008 Function and Class Definitions.

```
#include "MCP23008-I2C.h"  
#include "MCP23008-Constants.h"
```

### 7.2.1 Detailed Description

MCP23008 Function and Class Definitions.

#### Author

Frank Häfele

#### Date

27.12.2024

#### Version

1.0.0

#### See also

<https://github.com/hasenradball/MCP23008-I2C>

## 7.3 src/MCP23008-I2C.h File Reference

MCP23008 Declarations.

```
#include "Arduino.h"
#include "Wire.h"
```

### Classes

- class [MCP23008\\_I2C::MCP23008](#)  
*Class [MCP23008](#).*

### Namespaces

- [MCP23008\\_I2C](#)  
*namespace of [MCP23008](#). includes class declaration, errors, and states*

### Variables

- constexpr const char \* [MCP23008\\_I2C::MCP23008\\_LIB\\_VERSION](#) {"1.0.0"}
- constexpr int8\_t [MCP23008\\_I2C::MCP23008\\_STATE\\_OK](#) {0x00}  
*constant which states all ok, no error*
- constexpr int8\_t [MCP23008\\_I2C::MCP23008\\_ERROR\\_PIN](#) {-1}  
*constant which states a wrong pin number was used*
- constexpr int8\_t [MCP23008\\_I2C::MCP23008\\_ERROR\\_I2C](#) {-2}  
*constant which states an error during I2C communication*
- constexpr int8\_t [MCP23008\\_I2C::MCP23008\\_ERROR\\_VALUE](#) {-3}  
*constant which states that there was an error regarding a parameter value*

### 7.3.1 Detailed Description

MCP23008 Declarations.

Author

Frank Häfele

Date

27.12.2024

Version

1.0.0

See also

<https://github.com/hasenradball/MCP23008-I2C>



# Index

- begin
  - MCP23008\_I2C::MCP23008, [15](#)
- disableInterrupt
  - MCP23008\_I2C::MCP23008, [15](#)
- getAddress
  - MCP23008\_I2C::MCP23008, [16](#)
- getInterruptPolarity
  - MCP23008\_I2C::MCP23008, [16](#)
- getPinMode8
  - MCP23008\_I2C::MCP23008, [16](#)
- getPolarity
  - MCP23008\_I2C::MCP23008, [17](#)
- getPolarity8
  - MCP23008\_I2C::MCP23008, [17](#)
- getPullup
  - MCP23008\_I2C::MCP23008, [17](#)
- getPullup8
  - MCP23008\_I2C::MCP23008, [18](#)
- isConnected
  - MCP23008\_I2C::MCP23008, [18](#)
- MCP23008
  - MCP23008\_I2C::MCP23008, [14](#)
- MCP23008\_Constants, [9](#)
  - MCP23008\_IOCON\_DISSLW, [10](#)
  - MCP23008\_IOCON\_INTPOL, [10](#)
  - MCP23008\_IOCON\_ODR, [10](#)
  - MCP23008\_IOCON\_SEQOP, [10](#)
- MCP23008\_I2C, [10](#)
- MCP23008\_I2C::MCP23008, [13](#)
  - begin, [15](#)
  - disableInterrupt, [15](#)
  - getAddress, [16](#)
  - getInterruptPolarity, [16](#)
  - getPinMode8, [16](#)
  - getPolarity, [17](#)
  - getPolarity8, [17](#)
  - getPullup, [17](#)
  - getPullup8, [18](#)
  - isConnected, [18](#)
  - MCP23008, [14](#)
  - read1, [19](#)
  - read8, [19](#)
  - readInterruptCaptureRegister, [20](#)
  - readInterruptFlagRegister, [20](#)
  - setInterrupt, [20](#)
  - setInterruptPolarity, [21](#)
  - setPinMode1, [21](#)
  - setPinMode8, [22](#)
  - setPolarity, [23](#)
  - setPolarity8, [23](#)
  - setPullup, [24](#)
  - setPullup8, [24](#)
  - write1, [25](#)
  - write8, [25](#)
- MCP23008\_IOCON\_DISSLW
  - MCP23008\_Constants, [10](#)
- MCP23008\_IOCON\_INTPOL
  - MCP23008\_Constants, [10](#)
- MCP23008\_IOCON\_ODR
  - MCP23008\_Constants, [10](#)
- MCP23008\_IOCON\_SEQOP
  - MCP23008\_Constants, [10](#)
- read1
  - MCP23008\_I2C::MCP23008, [19](#)
- read8
  - MCP23008\_I2C::MCP23008, [19](#)
- readInterruptCaptureRegister
  - MCP23008\_I2C::MCP23008, [20](#)
- readInterruptFlagRegister
  - MCP23008\_I2C::MCP23008, [20](#)
- setInterrupt
  - MCP23008\_I2C::MCP23008, [20](#)
- setInterruptPolarity
  - MCP23008\_I2C::MCP23008, [21](#)
- setPinMode1
  - MCP23008\_I2C::MCP23008, [21](#)
- setPinMode8
  - MCP23008\_I2C::MCP23008, [22](#)
- setPolarity
  - MCP23008\_I2C::MCP23008, [23](#)
- setPolarity8
  - MCP23008\_I2C::MCP23008, [23](#)
- setPullup
  - MCP23008\_I2C::MCP23008, [24](#)
- setPullup8
  - MCP23008\_I2C::MCP23008, [24](#)
- src/MCP23008-Constants.h, [29](#)
- src/MCP23008-I2C.cpp, [30](#)
- src/MCP23008-I2C.h, [31](#)
- write1
  - MCP23008\_I2C::MCP23008, [25](#)
- write8
  - MCP23008\_I2C::MCP23008, [25](#)