PCF8574

Generated by Doxygen 1.9.1

Chapter 1

Change Log PCF8574

All notable changes to this project will be documented in this file.

The format is based on Keep a Changelog and this project adheres to Semantic Versioning.

1.1 [0.4.1] - 2023-09-23

- Update readme with advanced interrupts insights
  - kudos to ddowling for testing.
  - add example
- fix URL examples
- add Wire1 example (ESP32 + RP2040)

1.2 [0.4.0] - 2023-09-23

- refactor API, begin()
- update readme.md
- update examples
- add examples ESP32 RP2040
- minor edits

1.3 [0.3.9] - 2023-09-23

- Add Wire1 support for ESP32
- update readme.md
- minor edits

1.4 [0.3.8] - 2023-02-04

- update readme.md
- update GitHub actions
- update license 2023

1.5 [0.3.7] - 2022-11-16

- fix #40 - add interrupt section to readme.md

1.6 [0.3.6] - 2022-10-19

- fix example PCF8574_rotaryEncoder.ino
- add RP2040 to build-CI
- simplified changelog.md

1.7 [0.3.5] - 2022-06-17

- add select(), selectN(), selectNone() and selectAll() convenience wrappers

1.8 [0.3.4] - 2022-04-11

- add CHANGELOG.md
- fix **begin(int sda, int scl)** int parameters for ESP alike.

1.9 [0.3.3] - 2021-12-23

- update library.json, license, readme, minor edits

1.10 [0.3.2] - 2021-07-04

- fix #25 add setAddress()

1.11 [0.3.1] - 2021-04-23

- Fix for platformIO compatibility

1.12 [0.3.0] - 2021-01-03

- add multiWire support - inspired by mattbue - issue #14

1.13 [0.2.4] - 2020-12-17

- fix #6 tag problem 0.2.3

1.14 [0.2.3] - 2020-12-14

- fix #6 readButton8 ambiguity

1.15 [0.2.2] - 2020-12-07

- add Arduino-ci + start unit test + _wire->h in PCF8574.h

1.16 [0.2.1] - 2020-06-19

- fix library.json

1.17 [0.2.0] - 2020-05-22

- #pragma once; refactor;
- removed pre 1.0 support
- added begin(dsa, scl) for ESP32
- added reverse()

1.18 [0.1.9] - 2017-02-27

- fix warning about return in readButton8()

1.19 [0.1.08] - 2016-05-20'

- Merged work of Septillion
- Fix/refactor ButtonRead8() - see https://github.com/RobTillaart/Arduino/issues/38
- missing begin() => mask parameter

1.20 [0.1.07] - 2016-05-02

- (manually merged) Septillion
- added dataOut so a write() doesn't read first, possibly corrupting a input pin;
- fixed shift comment, should read 1..7;
- added begin() to be sure it's in a known state, states could be different if uC is reset and the PCF8574 isn't;
- added buttonRead() and buttonRead8() which only effect the output while reading

1.21 [0.1.06] - ??

- (intermediate) added defined errors + refactor rotate

1.22 [0.1.05] - 2016-04-30

- refactor, +toggleMask, +rotLeft, +rotRight

1.23 [0.1.04] - 2015-05-09

- removed ambiguity in read8()

1.24 [0.1.03] - 2015-03-02

- address int -> uint8_t

1.25 [0.1.02] - ?

- replaced integers with uint8_t to reduce footprint;
- added default value for shiftLeft() and shiftRight()
- renamed status() to lastError();

1.26 [0.1.01] - ?

- added value(); returns last read 8 bit value (cached); value() does not always reflect the latest state of the pins!

1.27 [0.1.00] - 2013-02-02

- initial version

Chapter 2

README

2.1 PCF8574

Arduino library for PCF8574 - 8 channel I2C IO expander.

2.1.1 Description

Related to the PCF8575 16 channel IO expander library https://github.com/RobTillaart/PCF8575
This library gives easy control over the 8 pins of a PCF8574 and PCF8574A chip. These chips are identical in behaviour although there are two distinct address ranges.

type	address-range	notes
PCF8574	0x20 to 0x27	same range as PCF8575 !

| PCF8574A | 0x38 to 0x3F |
So you can connect up to 16 PCF8574 on one I2C bus, giving access to 16 x 8 = 128 IO lines. To maximize IO lines combine 8 x PCF8575 + 8 x PCF8574A giving 128 + 64 = 192 IO lines. Be sure to have a well dimensioned power supply.
The library allows to read and write both single pins or 8 pins at once. Furthermore some additional functions are implemented that are playful and useful.

2.1.1.0.1 Interrupts intro The PCF8574 has an interrupt output line (INT) to notify an MCU that one of the input lines has changed. This can be used to prevent active polling of the PCF8574, which can be more efficient.
From the datasheet:
*An interrupt is generated by any rising or falling edge of the port inputs in the input mode. After time, (Tiv), INT is valid. Resetting and reactivating the interrupt circuit is achieved when data on the port is **changed to the original setting** or data is **read from**, or **written to**, the port that generated the interrupt. Resetting occurs in the read mode at the acknowledge bit after the rising edge of the SCL signal, or in the write mode at the acknowledge bit after the high-to-low transition of the SCL signal.*
So there are three scenarios how the INT is reset.

1. pins revert to original state (lesser known).
2. read from the device (well known)
3. write to the device (well known)

This implies that polling the PCF8574 can miss an INT in scenario 1. (see #48) In practice if you have faster polling than your signals changes this would not be a problem. E.g. tactile switches and a polling frequency $>$ 100 Hz will work.

2.1.1.0.2 Interrupts library The library cannot handle the PCF8574 interrupts as it has no code for it. The user should catch the interrupt in his own code to set a flag and can use the library to see which line has changed. There are two examples to show how interrupts can be handled:

- **PCF8574_interrupt.ino**
- **PCF8574_rotaryEncoder.ino**

A more advanced interrupt handler would not set a boolean flag in the interrupt routine but increase a counter (uint8_t or larger). Then it would be possible to see that:

1. an interrupt occurred. (counter $>$ 0)
2. if one or more interrupts are not handled (counter $>$ 1)

A minimal example that shows catching missed interrupts:

- **PCF8574_interrupt_advanced.ino**

2.1.1.0.3 0.4.0 Breaking change Version 0.4.0 introduced a breaking change. You cannot set the pins in **begin()** any more. This reduces the dependency of processor dependent Wire implementations. The user has to call **Wire.begin()** and can optionally set the Wire pins before calling **begin()**.

2.1.1.0.4 Related 16 bit port expanders

- https://github.com/RobTillaart/MCP23017_RT
- https://github.com/RobTillaart/MCP23S17
- https://github.com/RobTillaart/PCF8575

8 bit port expanders

- https://github.com/RobTillaart/MCP23008
- https://github.com/RobTillaart/MCP23S08
- https://github.com/RobTillaart/PCF8574

2.1.2 I2C Clock

Tested on UNO with **PCF8574_performance** showed that the PCF8574 still works at 500 KHz and failed at 600 KHz. These values are outside the specs of the datasheet so they are not recommended. However when performance is needed you can try to overclock the chip.

clock speed	Read	Write	Notes
100000	236	240	spec datasheet

| 200000 | 132 | 140 | | 300000 | 104 | 108 | | 400000 | 96 | 96 | max advised speed | | 500000 | 92 | 92 | not recommended | | 600000 | crash | crash |

2.1.3 Interface

```
#include "PCF8574.h"
```

PCF8574_INITIAL_VALUE is a define 0xFF that can be set compile time or before the include of "pcf8574.h" to overrule the default value used with the **begin()** call.

2.1.3.0.1 Constructor

- **PCF8574(uint8_t deviceAddress = 0x20, TwoWire *wire = &Wire)** Constructor with optional address, default 0x20, and the optional Wire interface as parameter.

- **bool begin(uint8_t value = PCF8574_INITIAL_VALUE)** set the initial value (default 0xFF) for the pins and masks.
- **bool isConnected()** checks if the address set in the constructor or by **setAddress()** is visible on the I2C bus.
- **bool setAddress(const uint8_t deviceAddress)** sets the device address after construction. Can be used to switch between PCF8574 modules runtime. Note this corrupts internal buffered values, so one might need to call **read8()** and/or **write8()**. Returns true if address can be found on I2C bus.
- **uint8_t getAddress()** Returns the device address.

2.1.3.0.2 Read and Write

- **uint8_t read8()** reads all 8 pins at once. This one does the actual reading.
- **uint8_t read(uint8_t pin)** reads a single pin; pin = 0..7
- **uint8_t value()** returns the last read inputs again, as this information is buffered in the class this is faster than reread the pins.
- **void write8(const uint8_t value)** writes all 8 pins at once. This one does the actual writing.
- **uint8_t write(const uint8_t pin, const uint8_t value)** writes a single pin; pin = 0..7; value is HIGH(1) or LOW (0)
- **uint8_t valueOut()** returns the last written data.

2.1.3.0.3 Button The ∗∗"button"∗∗ functions are to be used when you mix input and output on one IC. It does not change / affect the pins used for output by masking these. Typical usage is to call **setButtonMask()** once in setup as pins do not (often) change during program execution.

- **void setButtonMask(const uint8_t mask)** sets the (bit) mask which lines are input.
- **uint8_t getButtonMask()** returns the set buttonMask.
- **uint8_t readButton8()** use the mask set by setButtonMask to select specific input pins.
- **uint8_t readButton8(const uint8_t mask)** use a specific mask to select specific input pins. Note this can be a subset of the pins set with **setButtonMask()** if one wants to process not all.
- **uint8_t readButton(const uint8_t pin)** read a single input pin.

Background - https://github.com/RobTillaart/Arduino/issues/38

2.1.3.0.4 Special

- **void toggle(const uint8_t pin)** toggles a single pin
- **void toggleMask(const uint8_t mask = 0xFF)** toggles a selection of pins, if you want to invert all pins use 0xFF (default value).
- **void shiftRight(const uint8_t n = 1)** shifts output channels n pins (default 1) pins right (e.g. LEDs). Fills the higher lines with zero's.
- **void shiftLeft(const uint8_t n = 1)** shifts output channels n pins (default 1) pins left (e.g. LEDs). Fills the lower lines with zero's.
- **void rotateRight(const uint8_t n = 1)** rotates output channels to right, moving lowest line to highest line.
- **void rotateLeft(const uint8_t n = 1)** rotates output channels to left, moving highest line to lowest line.
- **void reverse()** reverse the "bit pattern" of the lines, swapping pin 7 with 0, 6 with 1, 5 with 2 etc.

2.1.3.0.5 Select

Some convenience wrappers.

- **void select(const uint8_t pin)** sets a single pin to HIGH, all others are set to LOW. If pin $>$ 7 all pins are set to LOW. Can be used to select one of n devices.
- **void selectN(const uint8_t pin)** sets pins 0..pin to HIGH, all others are set to LOW. If pin $>$ 7 all pins are set to LOW. This can typical be used to implement a VU meter.
- **void selectNone()** sets all pins to LOW.
- **void selectAll()** sets all pins to HIGH.

2.1.3.0.6 Miscellaneous

- **int lastError()** returns the last error from the lib. (see .h file).

2.1.4 Error codes

name	value	description
PCF8574_OK	0x00	no error
PCF8574_PIN_ERROR	0x81	pin number out of range
PCF8574_I2C_ERROR	0x82	I2C communication error

2.1.5 Operation

See examples.
It is advised to use pull-up or pull-down resistors so the lines have a defined state at startup.

2.1.6 Automated Documatation Creation

By the use of Doxygen it is able to run an automated documentation build.

2.1.6.1 Installation

To be able to run the doxygen build you need to install doxygen
```
sudo apt install doxygen
```
To be able to create PDF Output you need to install LaTex.
```
sudo apt install texlive-full
```

2.1.7 Usage

To build the documentation you have to use the following comand on the commandline
```
doxygen Doxyfile
```
After that you should be able to create the PDF by:
```
cd doxygen/latex
make
```
After that you will find the **refman.pdf** in the `doxygen/latex` folder.
Have fun.

2.1.8 Future

2.1.8.0.1 Must

- keep in sync with PCF8575 (as far as meaningful)

2.1.8.0.2 Should

2.1.8.0.3 Could

- move code to .cpp

2.1.8.0.4 Wont

2.1.9 Support

If you appreciate my libraries, you can support the development and maintenance. Improve the quality of the libraries by providing issues and Pull Requests, or donate through PayPal or GitHub sponsors.
Thank you,

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

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

Chapter 5

Class Documentation

5.1 PCF8574 Class Reference

Public Member Functions

- PCF8574 (const uint8_t deviceAddress=0x20, TwoWire *wire=&Wire)

  Construct a new PCF8574::PCF8574 object.
- bool begin (uint8_t value=PCF8574_INITIAL_VALUE)

  init PCF8574 instance
- bool isConnected ()

  check if device is connected
- bool setAddress (const uint8_t deviceAddress)

  set address of PCF8574 device
- uint8_t getAddress () const

  getter for I2C address of pcf7485 device should be in the range of 0x20...0x27
- uint8_t read8 ()

  read all 8 bit at once from PCF8574
- uint8_t read (const uint8_t pin)

  read one bit from PCF8574 device
- uint8_t value () const

  returns the last read inputs again, as this information is buffered in the class this is faster than re-read the pins.
- void write8 (const uint8_t value)

  write all 8 bit at once to PCF8574 device
- void write (const uint8_t pin, const uint8_t value)

  writes a single pin; pin = 0..7.
- uint8_t valueOut () const

  returns the last written data.
- uint8_t readButton8 ()

  returns the set buttonMask.
- uint8_t readButton8 (const uint8_t mask)

  read Button mask.
- uint8_t readButton (const uint8_t pin)

  read button read a single input pin.
- void setButtonMask (const uint8_t mask)

  sets the bit mask which lines are input.
- uint8_t getButtonMask ()

  returns the internal button mask member
- void toggle (const uint8_t pin)

  toggle one bit on PCF8574 device

- void toggleMask (const uint8_t mask=0xFF)

  toggles a selection of pins, if you want to invert all pins use 0xFF (default value).
- void shiftRight (const uint8_t n=1)

  shifts output channels n pins (default 1) pins right (e.g. LEDs). Fills the higher lines with zero's.
- void shiftLeft (const uint8_t n=1)

  shifts output channels n pins (default 1) pins left (e.g. LEDs). Fills the lower lines with zero's
- void rotateRight (const uint8_t n=1)

  rotates output channels to right, moving lowest line to highest line.
- void rotateLeft (const uint8_t n=1)

  rotates output channels to left, moving highest line to lowest line.
- void reverse ()

  reverse the "bit pattern" of the lines, swapping pin 7 with 0, 6 with 1, 5 with 2 etc.
- void select (const uint8_t pin)

  select sets the specified pin pin to High, all others set to Low. If pin > 7 all pins are set to LOW.
- void selectN (const uint8_t pin)

  select sets the pins from 0...pin to High, all others set to Low. If pin > 7 all pins are set to LOW.
- void selectNone ()

  sets all pins to LOW.
- void selectAll ()

  sets all pins to HIGH.
- int lastError ()

  returns the last error from the lib (see .h file).

5.1.1 Constructor & Destructor Documentation

5.1.1.1 PCF8574()

```
PCF8574::PCF8574 (
            const uint8_t deviceAddress = 0x20,
            TwoWire * wire = &Wire )  [explicit]
```

Construct a new PCF8574::PCF8574 object.

Parameters

deviceAddress	optional address of I2C device; default = 0x20;
wire	optional address of Wire instance; default = &Wire;

5.1.2 Member Function Documentation

5.1.2.1 begin()

```
bool PCF8574::begin (
            uint8_t value = PCF8574_INITIAL_VALUE )
```

init PCF8574 instance

Parameters

value	optional init value to write; default = 0xFF;

Returns

true

false

5.1.2.2 getAddress()

`uint8_t PCF8574::getAddress ( ) const [inline]`

getter for I2C address of pcf7485 device should be in the range of 0x20...0x27

Returns

uint8_t address

5.1.2.3 getButtonMask()

`uint8_t PCF8574::getButtonMask ( ) [inline]`

returns the internal button mask member

Returns

uint8_t the mask for input lines

5.1.2.4 isConnected()

`bool PCF8574::isConnected ( )`

check if device is connected

Returns

true

false

5.1.2.5 lastError()

`int PCF8574::lastError ( )`

returns the last error from the lib (see .h file).

Returns

int error value to return

5.1.2.6 read()

```
uint8_t PCF8574::read (
            const uint8_t pin )
```

read one bit from PCF8574 device

Parameters

pin	pin to read (0...7)

Returns

uint8_t value

5.1.2.7 read8()

```
uint8_t PCF8574::read8 ( )
```

read all 8 bit at once from PCF8574

Returns

uint8_t value read from device

5.1.2.8 readButton()

```
uint8_t PCF8574::readButton (
            const uint8_t pin )
```

read button read a single input pin.

Parameters

pin	pin number to be read

Returns

uint8_t value read from pin

5.1.2.9 readButton8() [1/2]

```
uint8_t PCF8574::readButton8 ( )  [inline]
```

returns the set buttonMask.

Returns

uint8_t the mask for input lines

5.1.2.10 readButton8() [2/2]

```
uint8_t PCF8574::readButton8 (
            const uint8_t mask )
```

read Button mask.

Parameters

mask	use the mask set by setButtonMask to select specific input pins.

Returns

∗ uint8_t

5.1.2.11 rotateLeft()

```
void PCF8574::rotateLeft (
```

```
            const uint8_t n = 1 )
```
rotates output channels to left, moving highest line to lowest line.

Parameters

n	number of moves

5.1.2.12 rotateRight()

```
void PCF8574::rotateRight (
            const uint8_t n = 1 )
```
rotates output channels to right, moving lowest line to highest line.

Parameters

n	number of moves

5.1.2.13 select()

```
void PCF8574::select (
            const uint8_t pin )
```
select sets the specified pin pin to High, all others set to Low. If pin $>$ 7 all pins are set to LOW.

Parameters

pin	pin which should be set to 1

5.1.2.14 selectN()

```
void PCF8574::selectN (
            const uint8_t pin )
```
select sets the pins from 0...pin to High, all others set to Low. If pin $>$ 7 all pins are set to LOW.

Parameters

pin	pin up to which should be set to 1

5.1.2.15 setAddress()

```
bool PCF8574::setAddress (
            const uint8_t deviceAddress )
```
set address of PCF8574 device

Parameters

deviceAddress	I2C address; possible values 0x20...0x27

Returns

true

false

5.1.2.16 setButtonMask()

```
void PCF8574::setButtonMask (
            const uint8_t mask )  [inline]
```

sets the bit mask which lines are input.

Parameters

mask	bit mask of inputs

5.1.2.17 shiftLeft()

```
void PCF8574::shiftLeft (
            const uint8_t n = 1 )
```

shifts output channels n pins (default 1) pins left (e.g. LEDs). Fills the lower lines with zero's

Parameters

n	number of shifts

5.1.2.18 shiftRight()

```
void PCF8574::shiftRight (
            const uint8_t n = 1 )
```

shifts output channels n pins (default 1) pins right (e.g. LEDs). Fills the higher lines with zero's.

Parameters

n	number of shifts

5.1.2.19 toggle()

```
void PCF8574::toggle (
            const uint8_t pin )
```

toggle one bit on PCF8574 device

Parameters

pin	pin number for write (0...7)

5.1.2.20 toggleMask()

```
void PCF8574::toggleMask (
            const uint8_t mask = 0xFF )
```

toggles a selection of pins, if you want to invert all pins use 0xFF (default value).

Parameters

mask	

5.1.2.21 value()

```
uint8_t PCF8574::value ( ) const  [inline]
```

returns the last read inputs again, as this information is buffered in the class this is faster than re-read the pins.

Returns

uint8_t internal member _dataIn

5.1.2.22 valueOut()

```
uint8_t PCF8574::valueOut ( ) const  [inline]
```

returns the last written data.

Returns

uint8_t internal member _dataOut

5.1.2.23 write()

```
void PCF8574::write (
            const uint8_t pin,
            const uint8_t value )
```

writes a single pin; pin = 0..7.

Parameters

pin	pin number for write (0...7)
value	value to write, HIGH(1) or LOW (0)

5.1.2.24 write8()

```
void PCF8574::write8 (
            const uint8_t value )
```

write all 8 bit at once to PCF8574 device

Parameters

value	to write

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

- PCF8574.h
- PCF8574.cpp

Chapter 6

File Documentation

6.1 PCF8574.cpp File Reference

Arduino library for PCF8574 - 8 channel I2C IO expander.
`#include "PCF8574.h"`

6.1.1 Detailed Description

Arduino library for PCF8574 - 8 channel I2C IO expander.

Author

Rob Tillaart

Date

02-febr-2013

Version

0.4.2

See also

https://github.com/RobTillaart/PCF8574
http://forum.arduino.cc/index.php?topic=184800

Index