

stm32-zigbee-sniffer

Generated by Doxygen 1.8.11



# Contents

<b>1</b>	<b>Main Page</b>	<b>1</b>
<b>2</b>	<b>Data Structure Index</b>	<b>3</b>
2.1	Data Structures	3
<b>3</b>	<b>File Index</b>	<b>5</b>
3.1	File List	5
<b>4</b>	<b>Data Structure Documentation</b>	<b>7</b>
4.1	LCD_PCF8574_HandleTypeDef Struct Reference	7
4.1.1	Detailed Description	7
4.1.2	Field Documentation	7
4.1.2.1	lcdbuf	7
4.1.2.2	NUMBER_OF_LINES	8
4.1.2.3	pcf8574	8
4.1.2.4	pins	8
4.1.2.5	state	8
4.1.2.6	type	8
4.2	MAC_AddressUnion Union Reference	8
4.2.1	Detailed Description	8
4.3	MAC_FrameControlFieldDef Struct Reference	9
4.3.1	Detailed Description	9
4.4	MAC_HeaderDef Struct Reference	9
4.4.1	Detailed Description	10
4.5	MRF24J40_HandleDef Struct Reference	10
4.5.1	Detailed Description	11
4.6	PCF8574_HandleTypeDef Struct Reference	11
4.6.1	Detailed Description	11
4.6.2	Field Documentation	11
4.6.2.1	i2c	11
4.6.2.2	PCF_I2C_ADDRESS	11
4.6.2.3	PCF_I2C_TIMEOUT	11

<b>5 File Documentation</b>	<b>13</b>
5.1 include/hd44780.h File Reference	13
5.1.1 Detailed Description	14
5.1.2 Enumeration Type Documentation	14
5.1.2.1 LCD_DIRECTION	14
5.1.2.2 LCD_INTERFACE	14
5.1.2.3 LCD_NUMBER_OF_LINES	15
5.1.2.4 LCD_PIN	15
5.1.2.5 LCD_RESULT	15
5.1.2.6 LCD_TYPE	15
5.1.3 Function Documentation	15
5.1.3.1 LCD_ClearDisplay(LCD_PCF8574_HandleTypeDef *handle)	15
5.1.3.2 LCD_CursorOFF(LCD_PCF8574_HandleTypeDef *handle)	15
5.1.3.3 LCD_CursorON(LCD_PCF8574_HandleTypeDef *handle, uint8_t blink)	16
5.1.3.4 LCD_CustomChar(LCD_PCF8574_HandleTypeDef *handle, uint8_t *pattern, uint8_t address)	16
5.1.3.5 LCD_DeInit(LCD_PCF8574_HandleTypeDef *handle)	16
5.1.3.6 LCD_DisplayOFF(LCD_PCF8574_HandleTypeDef *handle)	17
5.1.3.7 LCD_DisplayON(LCD_PCF8574_HandleTypeDef *handle)	18
5.1.3.8 LCD_EntryModeSet(LCD_PCF8574_HandleTypeDef *handle, LCD_DIRECTION direction, LCD_SHIFT shift)	18
5.1.3.9 LCD_GetBusyFlag(LCD_PCF8574_HandleTypeDef *handle, uint8_t *flag)	18
5.1.3.10 LCD_I2C_WriteOut(LCD_PCF8574_HandleTypeDef *handle)	19
5.1.3.11 LCD_Init(LCD_PCF8574_HandleTypeDef *handle)	19
5.1.3.12 LCD_SetLocation(LCD_PCF8574_HandleTypeDef *handle, uint8_t x, uint8_t y)	19
5.1.3.13 LCD_ShiftCursor(LCD_PCF8574_HandleTypeDef *handle, LCD_DIRECTION direction, uint8_t steps)	19
5.1.3.14 LCD_ShiftDisplay(LCD_PCF8574_HandleTypeDef *handle, uint8_t direction, uint8_t steps)	20
5.1.3.15 LCD_StateLEDControl(LCD_PCF8574_HandleTypeDef *handle, uint8_t on)	20
5.1.3.16 LCD_StateWriteBit(LCD_PCF8574_HandleTypeDef *handle, uint8_t value, LCD_PIN pin)	20
5.1.3.17 LCD_WaitForBusyFlag(LCD_PCF8574_HandleTypeDef *handle)	21

5.1.3.18	LCD_WriteCMD(LCD_PCF8574_HandleTypeDef *handle, uint8_t cmd) . . . . .	21
5.1.3.19	LCD_WriteDATA(LCD_PCF8574_HandleTypeDef *handle, uint8_t data) . . . . .	21
5.1.3.20	LCD_WriteNumber(LCD_PCF8574_HandleTypeDef *handle, unsigned long n, uint8_t base) . . . . .	21
5.1.3.21	LCD_WriteString(LCD_PCF8574_HandleTypeDef *handle, char *s) . . . . .	22
5.1.3.22	LCD_WriteToDataBus(LCD_PCF8574_HandleTypeDef *handle, uint8_t data) . .	22
5.2	include/MAC_Header_Parser.h File Reference . . . . .	22
5.2.1	Detailed Description . . . . .	23
5.2.2	Function Documentation . . . . .	23
5.2.2.1	MAC_Parse_Header(MAC_HeaderTypeDef *mach, uint8_t frame[], uint8_t↵ t frame_length) . . . . .	23
5.3	include/MRF24J40_Driver.h File Reference . . . . .	24
5.3.1	Detailed Description . . . . .	26
5.3.2	Macro Definition Documentation . . . . .	26
5.3.2.1	MRF24J40_RSSI_CONVERT . . . . .	26
5.3.3	Function Documentation . . . . .	26
5.3.3.1	MRF24J40_CreateHandle(MRF24J40_HandleTypeDef *handle, SPI_TypeDef *spi_td) . . . . .	26
5.3.3.2	MRF24J40_InitializeChip(MRF24J40_HandleTypeDef *handle) . . . . .	26
5.3.3.3	MRF24J40_ReadLong(MRF24J40_HandleTypeDef *handle, MRF24J40_Long↵ Addr addr, uint8_t *val) . . . . .	27
5.3.3.4	MRF24J40_ReadShort(MRF24J40_HandleTypeDef *handle, MRF24J40_↵ ShortAddr addr, uint8_t *val) . . . . .	27
5.3.3.5	MRF24J40_ReceiveFrame(MRF24J40_HandleTypeDef *handle) . . . . .	27
5.3.3.6	MRF24J40_SetChannel(MRF24J40_HandleTypeDef *handle, uint8_t channel) .	29
5.3.3.7	MRF24J40_WriteLong(MRF24J40_HandleTypeDef *handle, MRF24J40_Long↵ Addr addr, uint8_t val) . . . . .	29
5.3.3.8	MRF24J40_WriteShort(MRF24J40_HandleTypeDef *handle, MRF24J40_↵ ShortAddr addr, uint8_t val) . . . . .	29
5.4	include/pcf8574.h File Reference . . . . .	30
5.4.1	Detailed Description . . . . .	31
5.4.2	Enumeration Type Documentation . . . . .	31
5.4.2.1	PCF8574_RESULT . . . . .	31
5.4.3	Function Documentation . . . . .	31
5.4.3.1	PCF8574_DeInit(PCF8574_HandleTypeDef *handle) . . . . .	31
5.4.3.2	PCF8574_Init(PCF8574_HandleTypeDef *handle) . . . . .	31
5.4.3.3	PCF8574_Read(PCF8574_HandleTypeDef *handle, uint8_t *val) . . . . .	32
5.4.3.4	PCF8574_Write(PCF8574_HandleTypeDef *handle, uint8_t val) . . . . .	32



## Chapter 1

# Main Page

Sniffer for STM32F4Discovery to work with OLIMEX MRF24J40 radio. Library uses SPI to communicate with radio, I2C to output data to WH2004A LCD display.

MAC Header parser is also introduced in this library.





## Chapter 2

# Data Structure Index

### 2.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">LCD_PCF8574_HandleTypeDef</a> . . . . .	7
<a href="#">MAC_AddressUnion</a> Holds short (16-bit) or long (64-bit) MAC address presented in MAC Header . . . . .	8
<a href="#">MAC_FrameControlFieldDef</a> Parsed MAC Header Frame control field of 802.15.4 packet. For more information about field please refer to 802.15.4. data sheet . . . . .	9
<a href="#">MAC_HeaderDef</a> Parsed data of MAC Header in 802.15.4 packet. Security fields are not included. For more information about this field please refer to 802.15.4 data sheet . . . . .	9
<a href="#">MRF24J40_HandleDef</a> MRF24J40 handle to operate with this radio module. It has HAL SPI handle and various parameters of received frame . . . . .	10
<a href="#">PCF8574_HandleTypeDef</a> . . . . .	11



## Chapter 3

# File Index

### 3.1 File List

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

include/ <a href="#">hd44780.h</a>	Header file for communication with the HD44780 LCD driver. To use it you will have to create a variable of type <a href="#">LCD_PCF8574_HandleTypeDef</a> (e.g. "lcd") and then set the I2C address based on the address pins on your PCF8574 (0-7) (lcd.pcf8574.PCF_I2C_ADDRESS), set the I2C timeout (in milliseconds) (lcd.pcf8574.PCF_I2C_TIMEOUT), set the I2C instance (e.g. I2C1 or I2C2) (lcd.pcf8574.i2c.Instance), set the I2C clock speed (in Hertz) (lcd.pcf8574.i2c.Init.ClockSpeed), set the number of lines (has to be type of LCD_NUMBER_OF_LINES) (lcd.NUMBER_OF_LINES), set the interface type (has to be type of LCD_TYPE) (lcd.type) . . . . .	13
include/ <a href="#">MAC_Header_Parser.h</a>	This module allows to parse MAC header from 802.15.4 frame . . . . .	22
include/ <b>main_page.h</b>	. . . . .	??
include/ <a href="#">MRF24J40_Driver.h</a>	This module allows to operate with MRF24J40 module, connected to MCU via SPI interface. Module configures device to work in promiscuous mode in order to receive all packets with correct CRC (you can gain more info about CRC in 802.15.4 specification) . . . . .	24
include/ <a href="#">pcf8574.h</a>	In order to use this you have to create a <a href="#">PCF8574_HandleTypeDef</a> variable (e.g. "pcf"). Then you will set the the address based on the configuration of your chip (pins A0, A1, A2) ( pcf.PCF_I2C_ADDRESS ) (0 to 7), timeout ( pcf.PCF_I2C_TIMEOUT ) (e.g. 1000 (=1 sec)), I2C instance to use ( pcf.i2c.Instance ) (e.g. I2C1 or I2C2 ...), speed of the communication ( pcf.i2c.Init.ClockSpeed ) (e.g. 100 000 (=100kHz)) . . . . .	30
include/ <b>stm32f4xx_hal_conf.h</b>	. . . . .	??



## Chapter 4

# Data Structure Documentation

### 4.1 LCD\_PCF8574\_HandleTypeDef Struct Reference

```
#include <hd44780.h>
```

#### Data Fields

- [LCD\\_NUMBER\\_OF\\_LINES](#) [NUMBER\\_OF\\_LINES](#)
- `uint8_t` **D**
- `uint8_t` **C**
- `uint8_t` **B**
- `char` [lcdbuf](#) [2][16]
- `int` **x**
- `int` **oldx**
- `int` **y**
- `int` **oldy**
- `uint8_t` [state](#)
- `uint32_t` \* [pins](#)
- [LCD\\_TYPE](#) type
- [PCF8574\\_HandleTypeDef](#) [pcf8574](#)
- `void`(\* **errorCallback** )(LCD\_RESULT)

#### 4.1.1 Detailed Description

Structure that hold all the required variables in order to simplify the communication process

#### 4.1.2 Field Documentation

##### 4.1.2.1 `char lcdbuf[2][16]`

Buffer for the LCD

#### 4.1.2.2 LCD\_NUMBER\_OF\_LINES NUMBER\_OF\_LINES

Number of lines on your LCD

#### 4.1.2.3 PCF8574\_HandleTypeDef pcf8574

[PCF8574\\_HandleTypeDef](#) for communication with PCF8574

#### 4.1.2.4 uint32\_t\* pins

Array of pins based on your hardware (wiring)

#### 4.1.2.5 uint8\_t state

Holds current state of the PCF8574 expander

#### 4.1.2.6 LCD\_TYPE type

Type of hardware you want to use

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

- [include/hd44780.h](#)

## 4.2 MAC\_AddressUnion Union Reference

Holds short (16-bit) or long (64-bit) MAC address presented in MAC Header.

```
#include <MAC_Header_Parser.h>
```

### Data Fields

- `uint64_t` [addr\\_long](#)  
*Long (64-bit) MAC address.*
- `uint16_t` [addr\\_short](#)  
*Short (16-bit) MAC address.*

### 4.2.1 Detailed Description

Holds short (16-bit) or long (64-bit) MAC address presented in MAC Header.

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

- [include/MAC\\_Header\\_Parser.h](#)

## 4.3 MAC\_FrameControlFieldDef Struct Reference

Parsed MAC Header Frame control field of 802.15.4 packet. For more information about field please refer to 802.15.4. data sheet.

```
#include <MAC_Header_Parser.h>
```

### Data Fields

- uint8\_t [frame\\_type](#):2  
*MAC frame type.*
- uint8\_t [security\\_enabled](#):1  
*MAC security enabled flag.*
- uint8\_t [frame\\_pending](#):1  
*MAC frame pending flag.*
- uint8\_t [ack\\_request](#):1  
*MAC ACK request flag.*
- uint8\_t [panID\\_compression](#):1  
*MAC PAN ID comperssion flag.*
- uint8\_t [reserved](#):2 uint8\_t [dst\\_addr\\_mode](#) :2  
*Reserved bits in MAC FCF.*
- uint8\_t [frame\\_ver](#):2  
*MAC Frame version.*
- uint8\_t [src\\_addr\\_mode](#):2  
*Source adresssing mode.*

### 4.3.1 Detailed Description

Parsed MAC Header Frame control field of 802.15.4 packet. For more information about field please refer to 802.15.4. data sheet.

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

- include/[MAC\\_Header\\_Parser.h](#)

## 4.4 MAC\_HeaderDef Struct Reference

Parsed data of MAC Header in 802.15.4 packet. Security fields are not included. For more information about this field please refer to 802.15.4 data sheet.

```
#include <MAC_Header_Parser.h>
```

## Data Fields

- [MAC\\_Address dest\\_address](#)  
*Destination MAC address.*
- [MAC\\_Address src\\_address](#)  
*Source MAC address.*
- [MAC\\_FrameControlField frame\\_control](#)  
*MAC Header frame control field.*
- `uint16_t` [dest\\_panID](#)  
*Destination PAN ID.*
- `uint16_t` [src\\_panID](#)  
*Source PAN ID.*
- `uint8_t` [sequence\\_number](#)  
*MAC Sequence number.*
- `uint8_t` [reserved](#)  
*Reserved byte for alignment.*

### 4.4.1 Detailed Description

Parsed data of MAC Header in 802.15.4 packet. Security fields are not included. For more information about this field please refer to 802.15.4 data sheet.

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

- `include/MAC_Header_Parser.h`

## 4.5 MRF24J40\_HandleDef Struct Reference

MRF24J40 handle to operate with this radio module. It has HAL SPI handle and various parameters of received frame.

```
#include <MRF24J40_Driver.h>
```

## Data Fields

- `SPI_HandleTypeDef` [spi\\_handle](#)  
*HAL SPI handle in order to use SPI easily.*
- `MRF24J40_Callback` [callback](#)  
*Internal callback.*
- `uint8_t` [recieved\\_frame](#) [128]  
*Frame, that was received.*
- `uint8_t` [msg](#) [3]  
*Request that was sent to device.*
- `uint8_t` [frame\\_length](#)  
*Received frame length.*
- `uint8_t` [is\\_receiving](#)  
*Flag, that shows if MCU is busy reading RXFIFO.*
- `uint8_t` [intstat](#)  
*Received interrupt bitmask.*
- `uint8_t` [lqi](#)  
*Link quality indicator of received frame.*
- `uint8_t` [rssi](#)  
*RSSI of received frame.*



### 4.5.1 Detailed Description

MRF24J40 handle to operate with this radio module. It has HAL SPI handle and various parameters of received frame.

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

- [include/MRF24J40\\_Driver.h](#)

## 4.6 PCF8574\_HandleTypeDef Struct Reference

```
#include <pcf8574.h>
```

### Data Fields

- `uint8_t` [PCF\\_I2C\\_ADDRESS](#)
- `uint32_t` [PCF\\_I2C\\_TIMEOUT](#)
- `I2C_HandleTypeDef` [i2c](#)
- `void(* errorCallback )(PCF8574\_RESULT)`

### 4.6.1 Detailed Description

PCF8574 handle structure which wraps all the necessary variables together in order to simplify the communication with the chip

### 4.6.2 Field Documentation

#### 4.6.2.1 `I2C_HandleTypeDef i2c`

`I2C_HandleTypeDef` structure

#### 4.6.2.2 `uint8_t PCF_I2C_ADDRESS`

address of the chip you want to communicate with

#### 4.6.2.3 `uint32_t PCF_I2C_TIMEOUT`

timeout value for the communication in milliseconds

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

- [include/pcf8574.h](#)



## Chapter 5

# File Documentation

### 5.1 include/hd44780.h File Reference

Header file for communication with the HD44780 LCD driver. To use it you will have to create a variable of type [LCD\\_PCF8574\\_HandleTypeDef](#) (e.g. "lcd") and then set the I2C address based on the address pins on your PCF8574 (0-7) (lcd.pcf8574.PCF\_I2C\_ADDRESS), set the I2C timeout (in milliseconds) (lcd.pcf8574.PCF\_I2C\_TIMEOUT), set the I2C instance (e.g. I2C1 or I2C2) (lcd.pcf8574.i2c.Instance), set the I2C clock speed (in Hertz) (lcd.pcf8574.i2c.Init.ClockSpeed), set the number of lines (has to be type of LCD\_NUMBER\_OF\_LINES) (lcd.NUMBER\_OF\_LINES), set the interface type (has to be type of LCD\_TYPE) (lcd.type).

```
#include <stdio.h>
#include <stdint.h>
#include "stm32f4xx_hal.h"
#include "pcf8574.h"
```

#### Data Structures

- struct [LCD\\_PCF8574\\_HandleTypeDef](#)

#### Macros

- #define [LCD\\_INTERFACE\\_SELECTOR\\_PCF8574](#)

#### Enumerations

- enum [LCD\\_INTERFACE](#) { [PCF8574](#), [GPIO](#) }
- enum [LCD\\_RESULT](#) { [LCD\\_OK](#), [LCD\\_ERROR](#) }
- enum [LCD\\_TYPE](#) { [TYPE0](#), [TYPE1](#), [TYPE2](#) }
- enum [LCD\\_NUMBER\\_OF\\_LINES](#) { [NUMBER\\_OF\\_LINES\\_1](#) =0, [NUMBER\\_OF\\_LINES\\_2](#) =1 }
- enum [LCD\\_PIN](#) {  
    [LCD\\_PIN\\_D4](#) =0, [LCD\\_PIN\\_D5](#) =1, [LCD\\_PIN\\_D6](#) =2, [LCD\\_PIN\\_D7](#) =3,  
    [LCD\\_PIN\\_RS](#) =4, [LCD\\_PIN\\_RW](#) =5, [LCD\\_PIN\\_E](#) =6, [LCD\\_PIN\\_LED](#) =7 }
- enum [LCD\\_DIRECTION](#) { [DIRECTION\\_LEFT](#) =0, [DIRECTION\\_RIGHT](#) =1 }
- enum [LCD\\_DIRECTION\\_INC\\_DEC](#) { [DIRECTION\\_INCREMENT](#) =1, [DIRECTION\\_DECREMENT](#) =2 }
- enum [LCD\\_SHIFT](#) { [SHIFT\\_YES](#) =1, [SHIFT\\_NO](#) =0 }

## Functions

- [LCD\\_RESULT LCD\\_Init](#) ([LCD\\_PCF8574\\_HandleTypeDef](#) \*handle)
- [LCD\\_RESULT LCD\\_DeInit](#) ([LCD\\_PCF8574\\_HandleTypeDef](#) \*handle)
- [LCD\\_RESULT LCD\\_WriteCMD](#) ([LCD\\_PCF8574\\_HandleTypeDef](#) \*handle, uint8\_t cmd)
- [LCD\\_RESULT LCD\\_WriteDATA](#) ([LCD\\_PCF8574\\_HandleTypeDef](#) \*handle, uint8\_t data)
- [LCD\\_RESULT LCD\\_GetBusyFlag](#) ([LCD\\_PCF8574\\_HandleTypeDef](#) \*handle, uint8\_t \*flag)
- [LCD\\_RESULT LCD\\_WriteToDataBus](#) ([LCD\\_PCF8574\\_HandleTypeDef](#) \*handle, uint8\_t data)
- [LCD\\_RESULT LCD\\_ClearDisplay](#) ([LCD\\_PCF8574\\_HandleTypeDef](#) \*handle)
- [LCD\\_RESULT LCD\\_WriteString](#) ([LCD\\_PCF8574\\_HandleTypeDef](#) \*handle, char \*s)
- [LCD\\_RESULT LCD\\_SetLocation](#) ([LCD\\_PCF8574\\_HandleTypeDef](#) \*handle, uint8\_t x, uint8\_t y)
- [LCD\\_RESULT LCD\\_DisplayON](#) ([LCD\\_PCF8574\\_HandleTypeDef](#) \*handle)
- [LCD\\_RESULT LCD\\_DisplayOFF](#) ([LCD\\_PCF8574\\_HandleTypeDef](#) \*handle)
- [LCD\\_RESULT LCD\\_CursorON](#) ([LCD\\_PCF8574\\_HandleTypeDef](#) \*handle, uint8\_t blink)
- [LCD\\_RESULT LCD\\_CursorOFF](#) ([LCD\\_PCF8574\\_HandleTypeDef](#) \*handle)
- [LCD\\_RESULT LCD\\_ShiftCursor](#) ([LCD\\_PCF8574\\_HandleTypeDef](#) \*handle, [LCD\\_DIRECTION](#) direction, uint8\_t steps)
- [LCD\\_RESULT LCD\\_ShiftDisplay](#) ([LCD\\_PCF8574\\_HandleTypeDef](#) \*handle, uint8\_t direction, uint8\_t steps)
- [LCD\\_RESULT LCD\\_WriteNumber](#) ([LCD\\_PCF8574\\_HandleTypeDef](#) \*handle, unsigned long n, uint8\_t base)
- [LCD\\_RESULT LCD\\_WriteFloat](#) ([LCD\\_PCF8574\\_HandleTypeDef](#) \*handle, double number, uint8\_t digits)
- [LCD\\_RESULT LCD\\_EntryModeSet](#) ([LCD\\_PCF8574\\_HandleTypeDef](#) \*handle, [LCD\\_DIRECTION\\_INC\\_DEC](#) direction, [LCD\\_SHIFT](#) shift)
- [LCD\\_RESULT LCD\\_CustomChar](#) ([LCD\\_PCF8574\\_HandleTypeDef](#) \*handle, uint8\_t \*pattern, uint8\_t address)
- [LCD\\_RESULT LCD\\_I2C\\_WriteOut](#) ([LCD\\_PCF8574\\_HandleTypeDef](#) \*handle)
- [LCD\\_RESULT LCD\\_StateLEDControl](#) ([LCD\\_PCF8574\\_HandleTypeDef](#) \*handle, uint8\_t on)
- [LCD\\_RESULT LCD\\_StateWriteBit](#) ([LCD\\_PCF8574\\_HandleTypeDef](#) \*handle, uint8\_t value, [LCD\\_PIN](#) pin)
- void [LCD\\_WaitForBusyFlag](#) ([LCD\\_PCF8574\\_HandleTypeDef](#) \*handle)

### 5.1.1 Detailed Description

Header file for communication with the HD44780 LCD driver. To use it you will have to create a variable of type [LCD\\_PCF8574\\_HandleTypeDef](#) (e.g. "lcd") and then set the I2C address based on the address pins on your PCF8574 (0-7) (`lcd.pcf8574.PCF_I2C_ADDRESS`), set the I2C timeout (in milliseconds) (`lcd.pcf8574.PCF_I2C_TIMEOUT`), set the I2C instance (e.g. I2C1 or I2C2) (`lcd.pcf8574.i2c.Instance`), set the I2C clock speed (in Hertz) (`lcd.pcf8574.i2c.Init.ClockSpeed`), set the number of lines (has to be type of [LCD\\_NUMBER\\_OF\\_LINES](#)) (`lcd.NUMBER_OF_LINES`), set the interface type (has to be type of [LCD\\_TYPE](#)) (`lcd.type`).

Example: `example.c` `example_msp.c`

### 5.1.2 Enumeration Type Documentation

#### 5.1.2.1 enum LCD\_DIRECTION

Used to specify the direction in certain LCD operations

#### 5.1.2.2 enum LCD\_INTERFACE

LCD Interface possibilities

Enumerator

**PCF8574** Use PCF8574 I2C IO expander as the interface

**GPIO** Use GPIO pins directly

### 5.1.2.3 enum LCD\_NUMBER\_OF\_LINES

Number of lines on your LCD

### 5.1.2.4 enum LCD\_PIN

Enumeration of the LCD pins

### 5.1.2.5 enum LCD\_RESULT

Possible return values for the functions

Enumerator

**LCD\_OK** Everything went OK

**LCD\_ERROR** An error occurred

### 5.1.2.6 enum LCD\_TYPE

Type of hardware to use

## 5.1.3 Function Documentation

### 5.1.3.1 LCD\_RESULT LCD\_ClearDisplay ( LCD\_PCF8574\_HandleTypeDef \* *handle* )

Clears the LCD

Parameters

<i>handle</i>	- a pointer to the LCD handle
---------------	-------------------------------

Returns

whether the function was successful or not

### 5.1.3.2 LCD\_RESULT LCD\_CursorOFF ( LCD\_PCF8574\_HandleTypeDef \* *handle* )

Turns OFF the cursor

Parameters

<i>handle</i>	- a pointer to the LCD handle
---------------	-------------------------------

**Returns**

whether the function was successful or not

**5.1.3.3 LCD\_RESULT LCD\_CursorON ( LCD\_PCF8574\_HandleTypeDef \* *handle*, uint8\_t *blink* )**

Turns ON the cursor

**Parameters**

<i>handle</i>	- a pointer to the LCD handle
<i>blink</i>	- if you want the cursor to blink set this to 1, else 0

**Returns**

whether the function was successful or not

**5.1.3.4 LCD\_RESULT LCD\_CustomChar ( LCD\_PCF8574\_HandleTypeDef \* *handle*, uint8\_t \* *pattern*, uint8\_t *address* )**

Creates a custom character at the given address

**Parameters**

<i>handle</i>	- a pointer to the LCD handle
<i>pattern</i>	- pointer to the bit pattern of the character
<i>address</i>	- an address to which the character will be written

**Returns**

whether the function was successful or not

**5.1.3.5 LCD\_RESULT LCD\_DeInit ( LCD\_PCF8574\_HandleTypeDef \* *handle* )**

LCD deinitialization function

**Parameters**

<i>handle</i>	- a pointer to the LCD handle
---------------	-------------------------------

**Returns**

whether the function was successful or not

#### 5.1.3.6 LCD\_RESULT LCD\_DisplayOFF ( LCD\_PCF8574\_HandleTypeDef \* *handle* )

Turns OFF the display

**Parameters**

<i>handle</i>	- a pointer to the LCD handle
---------------	-------------------------------

**Returns**

whether the function was successful or not

**5.1.3.7 LCD\_RESULT LCD\_DisplayON ( LCD\_PCF8574\_HandleTypeDef \* *handle* )**

Turns ON the display

**Parameters**

<i>handle</i>	- a pointer to the LCD handle
---------------	-------------------------------

**Returns**

whether the function was successful or not

**5.1.3.8 LCD\_RESULT LCD\_EntryModeSet ( LCD\_PCF8574\_HandleTypeDef \* *handle*, LCD\_DIRECTION\_INC\_DEC *direction*, LCD\_SHIFT *shift* )**

Sets the mode by which data is written to the LCD

**Parameters**

<i>handle</i>	- a pointer to the LCD handle
<i>direction</i>	
<i>shift</i>	

**Returns**

whether the function was successful or not

**5.1.3.9 LCD\_RESULT LCD\_GetBusyFlag ( LCD\_PCF8574\_HandleTypeDef \* *handle*, uint8\_t \* *flag* )**

Gets the state of the busy flag

**Parameters**

<i>handle</i>	- a pointer to the LCD handle
<i>flag</i>	- a pointer to a variable that will contain the state of the flag



**Returns**

whether the function was successful or not

**5.1.3.10 LCD\_RESULT LCD\_I2C\_WriteOut ( LCD\_PCF8574\_HandleTypeDef \* *handle* )**

Writes the current state to the PCF8574 expander

**Parameters**

<i>handle</i>	- a pointer to the LCD handle
---------------	-------------------------------

**Returns**

whether the function was successful or not

**5.1.3.11 LCD\_RESULT LCD\_Init ( LCD\_PCF8574\_HandleTypeDef \* *handle* )**

LCD initialization function

**Parameters**

<i>handle</i>	- a pointer to the LCD handle
---------------	-------------------------------

**Returns**

whether the function was successful or not

**5.1.3.12 LCD\_RESULT LCD\_SetLocation ( LCD\_PCF8574\_HandleTypeDef \* *handle*, uint8\_t *x*, uint8\_t *y* )**

Sets the location of the memory pointer in the controller (used to control other operations (for example where to write a string))

**Parameters**

<i>handle</i>	- a pointer to the LCD handle
<i>x</i>	- x-coordinate of the location
<i>y</i>	- y-coordinate of the location

**Returns**

whether the function was successful or not

**5.1.3.13 LCD\_RESULT LCD\_ShiftCursor ( LCD\_PCF8574\_HandleTypeDef \* *handle*, LCD\_DIRECTION *direction*, uint8\_t *steps* )**

Shifts the cursor in the specified direction certain number of steps

**Parameters**

<i>handle</i>	- a pointer to the LCD handle
<i>direction</i>	- specifies the direction
<i>steps</i>	- specifies how many positions to shift the cursor by

**Returns**

whether the function was successful or not

**5.1.3.14 LCD\_RESULT LCD\_ShiftDisplay ( LCD\_PCF8574\_HandleTypeDef \* *handle*, uint8\_t *direction*, uint8\_t *steps* )**

Shifts the contents of the LCD

**Parameters**

<i>handle</i>	- a pointer to the LCD handle
<i>direction</i>	- directions of the shift
<i>steps</i>	- how many positions to shift the contents by

**Returns**

whether the function was successful or not

**5.1.3.15 LCD\_RESULT LCD\_StateLEDControl ( LCD\_PCF8574\_HandleTypeDef \* *handle*, uint8\_t *on* )**

Controls the state of the LCD backlight

**Parameters**

<i>handle</i>	- a pointer to the LCD handle
<i>on</i>	- set it to 1 if you want to turn the backlight on, else 0

**Returns**

whether the function was successful or not

**5.1.3.16 LCD\_RESULT LCD\_StateWriteBit ( LCD\_PCF8574\_HandleTypeDef \* *handle*, uint8\_t *value*, LCD\_PIN *pin* )**

Rewrites a bit in the state variable with the value specified

**Parameters**

<i>handle</i>	- a pointer to the LCD handle
<i>value</i>	- value of the bit (0 or 1)
<i>pin</i>	- pin which you want to write to

**Returns**

whether the function was successful or not

**5.1.3.17 void LCD\_WaitForBusyFlag ( LCD\_PCF8574\_HandleTypeDef \* *handle* )**

Waits until the busy flag is reset

**Parameters**

<i>handle</i>	- a pointer to the LCD handle
---------------	-------------------------------

**5.1.3.18 LCD\_RESULT LCD\_WriteCMD ( LCD\_PCF8574\_HandleTypeDef \* *handle*, uint8\_t *cmd* )**

Sends a command to the HD44780 controller

**Parameters**

<i>handle</i>	- a pointer to the LCD handle
<i>cmd</i>	- a command you want to send

**Returns**

whether the function was successful or not

**5.1.3.19 LCD\_RESULT LCD\_WriteDATA ( LCD\_PCF8574\_HandleTypeDef \* *handle*, uint8\_t *data* )**

Sends data to the HD44780 controller

**Parameters**

<i>handle</i>	- a pointer to the LCD handle
<i>data</i>	- data you want to send

**Returns**

whether the function was successful or not

**5.1.3.20 LCD\_RESULT LCD\_WriteNumber ( LCD\_PCF8574\_HandleTypeDef \* *handle*, unsigned long *n*, uint8\_t *base* )**

Writes a number to the LCD

**Parameters**

<i>handle</i>	- a pointer to the LCD handle
<i>n</i>	- a number you want to write to the LCD

**Returns**

whether the function was successful or not

**5.1.3.21 LCD\_RESULT LCD\_WriteString ( LCD\_PCF8574\_HandleTypeDef \* handle, char \* s )**

Writes a string to the LCD

**Parameters**

<i>handle</i>	- a pointer to the LCD handle
<i>s</i>	- string you want to write to the LCD

**Returns**

whether the function was successful or not

**5.1.3.22 LCD\_RESULT LCD\_WriteToDataBus ( LCD\_PCF8574\_HandleTypeDef \* handle, uint8\_t data )**

Writes lower 4bits of data to the data bus of the controller

**Parameters**

<i>handle</i>	- a pointer to the LCD handle
<i>data</i>	- data you want to put on the data bus (lower 4bits)

**Returns**

whether the function was successful or not

**5.2 include/MAC\_Header\_Parser.h File Reference**

This module allows to parse MAC header from 802.15.4 frame.

```
#include <stdint.h>
```

**Data Structures**

- struct [MAC\\_FrameControlFieldDef](#)  
*Parsed MAC Header Frame control field of 802.15.4 packet. For more information about field please refer to 802.15.4 data sheet.*
- union [MAC\\_AddressUnion](#)  
*Holds short (16-bit) or long (64-bit) MAC address presented in MAC Header.*
- struct [MAC\\_HeaderDef](#)  
*Parsed data of MAC Header in 802.15.4 packet. Security fields are not included. For more information about this field please refer to 802.15.4 data sheet.*

## Macros

- `#define MAC_HEADER_RES_OK (MAC_Header_Result)(0x00)`  
*Function exited without any errors.*
- `#define MAC_HEADER_RES_ERR (MAC_Header_Result)(0x01)`  
*Function exited with error.*
- `#define MAC_FRAME_TYPE_BEACON (uint8_t)(0b00)`  
*MAC Beacon frame type.*
- `#define MAC_FRAME_TYPE_DATA (uint8_t)(0b01)`  
*MAC Data frame type.*
- `#define MAC_FRAME_TYPE_ACK (uint8_t)(0b10)`  
*MAC ACK frame type.*
- `#define MAC_FRAME_TYPE_CMD (uint8_t)(0b11)`  
*MAC Command frame type.*
- `#define MAC_ADDR_SHORT (uint8_t)(0b10)`  
*MAC 16-bit address mode.*
- `#define MAC_ADDR_LONG (uint8_t)(0b11)`  
*MAC 64-bit address mode.*
- `#define MAC_SRC_ADDR_PRESENTED(FC) ((FC).src_addr_mode)`  
*Determines if source address is presented in MAC header.*
- `#define MAC_DST_ADDR_PRESENTED(FC) ((FC).dst_addr_mode)`  
*Determines if destination address is presented in MAC header.*

## Typedefs

- `typedef uint8_t MAC_Header_Result`  
*Function exit status.*
- `typedef struct MAC_FrameControlFieldDef MAC_FrameControlField`  
*Parsed MAC Header Frame control field of 802.15.4 packet. For more information about field please refer to 802.15.4 data sheet.*
- `typedef union MAC_AddressUnion MAC_Address`  
*Holds short (16-bit) or long (64-bit) MAC address presented in MAC Header.*
- `typedef struct MAC_HeaderDef MAC_HeaderTypeDef`  
*Parsed data of MAC Header in 802.15.4 packet. Security fields are not included. For more information about this field please refer to 802.15.4 data sheet.*

## Functions

- `MAC_Header_Result MAC_Parse_Header (MAC_HeaderTypeDef *mach, uint8_t frame[], uint8_t frame_length)`  
*Parses MAC Header of given frame or packet and stores parsed data in given MAC\_HeaderTypeDef.*

### 5.2.1 Detailed Description

This module allows to parse MAC header from 802.15.4 frame.

### 5.2.2 Function Documentation

#### 5.2.2.1 `MAC_Header_Result MAC_Parse_Header ( MAC_HeaderTypeDef * mach, uint8_t frame[], uint8_t frame_length )`

Parses MAC Header of given frame or packet and stores parsed data in given MAC\_HeaderTypeDef.

## Parameters

<i>mach</i>	- a pointer to MAC Header to store parsed header data
<i>frame</i>	- packet or frame to parse
<i>frame_length</i>	- frame or packet length

## Returns

error code

### 5.3 include/MRF24J40\_Driver.h File Reference

This module allows to operate with MRF24J40 module, connected to MCU via SPI interface. Module configures device to work in promiscuous mode in order to receive all packets with correct CRC (you can gain more info about CRC in 802.15.4 specification).

```
#include "stm32f4xx_hal.h"
```

#### Data Structures

- struct [MRF24J40\\_HandleDef](#)

*MRF24J40 handle to operate with this radio module. It has HAL SPI handle and various parameters of received frame.*

#### Macros

- #define [MRF24J40\\_RESULT\\_OK](#) (uint8\_t)(0x00)  
*Function exited without any errors.*
- #define [MRF24J40\\_RESULT\\_ERR](#) (uint8\_t)(0x01)  
*Function exited with error.*
- #define [MRF24J40\\_RXMCR](#) ([MRF24J40\\_ShortAddr](#))(0x00)  
*MRF24J40 Recieve MAC control register.*
- #define [MRF24J40\\_PACON2](#) ([MRF24J40\\_ShortAddr](#))(0x18)  
*MRF24J40 power amplifier control 2 register.*
- #define [MRF24J40\\_SOFTTRST](#) ([MRF24J40\\_ShortAddr](#))(0x2A)  
*MRF24J40 soft reset register.*
- #define [MRF24J40\\_TXSTBL](#) ([MRF24J40\\_ShortAddr](#))(0x2E)  
*MRF24J40 transmit stabilization register.*
- #define [MRF24J40\\_INTSTAT](#) ([MRF24J40\\_ShortAddr](#))(0x31)  
*MRF24J40 interrupt status register.*
- #define [MRF24J40\\_INTCON](#) ([MRF24J40\\_ShortAddr](#))(0x32)  
*MRF24J40 interrupt configuration register.*
- #define [MRF24J40\\_RFCTL](#) ([MRF24J40\\_ShortAddr](#))(0x36)  
*MRF24J40 RF mode control register.*
- #define [MRF24J40\\_BBREG1](#) ([MRF24J40\\_ShortAddr](#))(0x39)  
*MRF24J40 baseband 1 register.*
- #define [MRF24J40\\_BBREG2](#) ([MRF24J40\\_ShortAddr](#))(0x3A)

- MRF24J40 baseband 2 register.*
- #define [MRF24J40\\_BBREG6](#) ([MRF24J40\\_ShortAddr](#))(0x3E)
- MRF24J40 baseband 6 register.*
- #define [MRF24J40\\_CCAEDTH](#) ([MRF24J40\\_ShortAddr](#))(0x3F)
- MRF24J40 energy detection threshold for CCA register.*
- #define [MRF24J40\\_RFCON](#)(N) ([MRF24J40\\_LongAddr](#))(0x200 + (N))
- MRF24J40 RF control registers.*
- #define [MRF24J40\\_SLPCON1](#) ([MRF24J40\\_LongAddr](#))(0x220)
- MRF24J40 sleep clock control 1 register.*
- #define [MRF24J40\\_RXFIFO](#) ([MRF24J40\\_LongAddr](#))(0x300)
- MRF24J40 RXFIFO start register.*
- #define [MRF24J40\\_RXFIFO\\_DATA](#)(N) ([MRF24J40\\_LongAddr](#))(0x301 + (N))
- Forms address for RXFIFO register with number N.*
- #define [MRF24J40\\_CHANNEL](#)(CH) (uint8\_t)((CH - 11) & 0x0F) << 4)
- Converts channel number from decimal to hex.*
- #define [MRF24J40\\_RSSI\\_CONVERT](#)(VAL) (uint8\_t)(-(VAL)\*0.22 + 90)
- Converts hex RSSI value received from radio to dbm.*

## Typedefs

- typedef uint8\_t [MRF24J40\\_Result](#)  
*Function exit status.*
- typedef uint8\_t [MRF24J40\\_ShortAddr](#)  
*MRF24J40 short address type.*
- typedef uint16\_t [MRF24J40\\_LongAddr](#)  
*MRF24J40 long address type.*
- typedef void(\* [MRF24J40\\_Callback](#)) (void \*)
- typedef struct [MRF24J40\\_HandleDef](#) [MRF24J40\\_HandleTypeDef](#)  
*MRF24J40 handle to operate with this radio module. It has HAL SPI handle and various parameters of received frame.*

## Functions

- [MRF24J40\\_Result](#) [MRF24J40\\_CreateHandle](#) ([MRF24J40\\_HandleTypeDef](#) \*handle, [SPI\\_TypeDef](#) \*spi\_td)  
*Creates MRF24J40 handle and sets up SPI hardware.*
- [MRF24J40\\_Result](#) [MRF24J40\\_InitializeChip](#) ([MRF24J40\\_HandleTypeDef](#) \*handle)  
*Sending initialization sequence to MRF24J40 device as described in data sheet.*
- [MRF24J40\\_Result](#) [MRF24J40\\_WriteShort](#) ([MRF24J40\\_HandleTypeDef](#) \*handle, [MRF24J40\\_ShortAddr](#) addr, uint8\_t val)  
*Sends sequence to specified MRF24J40 device, allowing to write value to device's short address memory (addresses from 0x00 to 0x3F) at given address.*
- [MRF24J40\\_Result](#) [MRF24J40\\_ReadShort](#) ([MRF24J40\\_HandleTypeDef](#) \*handle, [MRF24J40\\_ShortAddr](#) addr, uint8\_t \*val)  
*Sends sequence to specified MRF24J40 device, allowing to read value of device's short address memory (addresses from 0x0 to 0x3F) from given address.*
- [MRF24J40\\_Result](#) [MRF24J40\\_WriteLong](#) ([MRF24J40\\_HandleTypeDef](#) \*handle, [MRF24J40\\_LongAddr](#) addr, uint8\_t val)  
*Sends sequence to specified MRF24J40 device, allowing to write value to device's long address memory (addresses from 0x220 to 0x38F) at given address.*
- [MRF24J40\\_Result](#) [MRF24J40\\_ReadLong](#) ([MRF24J40\\_HandleTypeDef](#) \*handle, [MRF24J40\\_LongAddr](#) addr, uint8\_t \*val)

*Sends sequence to specified MRF24J40 device, allowing to read value of device's long address memory (addresses from 0x220 to 0x38F) from given address.*

- `MRF24J40_Result MRF24J40_SetChannel (MRF24J40_HandleTypeDef *handle, uint8_t channel)`

*Sends sequence to specified MRF24J40 device, allowing to change device's channel.*

- `MRF24J40_Result MRF24J40_ReceiveFrame (MRF24J40_HandleTypeDef *handle)`

*Reads frame data from MRF24J40 RXFIFO register and stores it at given handle's received\_frame field.*

### 5.3.1 Detailed Description

This module allows to operate with MRF24J40 module, connected to MCU via SPI interface. Module configures device to work in promiscuous mode in order to receive all packets with correct CRC (you can gain more info about CRC in 802.15.4 specification).

### 5.3.2 Macro Definition Documentation

5.3.2.1 `#define MRF24J40_RSSI_CONVERT( VAL ) (uint8_t)(-VAL)*0.22 + 90)`

Converts hex RSSI value received from radio to dbm.

In this macro used RSSI function described in datasheet. Instead of making big RSSI tables, RSSI value can be approximated by simple linear function.

### 5.3.3 Function Documentation

5.3.3.1 `MRF24J40_Result MRF24J40_CreateHandle ( MRF24J40_HandleTypeDef * handle, SPI_TypeDef * spi_td )`

Creates MRF24J40 handle and sets up SPI hardware.

#### Parameters

<i>handle</i>	- pointer to the MRF24J40 handle
<i>spi_td</i>	- pointer to the SPI typedef, specified in stm32f407x.h

#### Returns

error code

5.3.3.2 `MRF24J40_Result MRF24J40_InitializeChip ( MRF24J40_HandleTypeDef * handle )`

Sending initialization sequence to MRF24J40 device as described in data sheet.

This function configures MRF24J40 to receive in promiscuous mode (for additional information, please refer to the device data sheet).



## Parameters

<i>handle</i>	- pointer to the MRF24J40 handle
---------------	----------------------------------

## Returns

error code

### 5.3.3.3 MRF24J40\_Result MRF24J40\_ReadLong ( MRF24J40\_HandleTypeDef \* *handle*, MRF24J40\_LongAddr *addr*, uint8\_t \* *val* )

Sends sequence to specified MRF24J40 device, allowing to read value of device's long address memory (addresses from 0x220 to 0x38F) from given address.

This function is non-blocking and requires SPI interrupt function to indicate that reception was completed.

## Parameters

<i>handle</i>	- pointer to the MRF24J40
<i>addr</i>	- address at device's long address memory
<i>val</i>	- pointer to variable, where received value will be stored

## Returns

error code

### 5.3.3.4 MRF24J40\_Result MRF24J40\_ReadShort ( MRF24J40\_HandleTypeDef \* *handle*, MRF24J40\_ShortAddr *addr*, uint8\_t \* *val* )

Sends sequence to specified MRF24J40 device, allowing to read value of device's short address memory (addresses from 0x0 to 0x3F) from given address.

This function is non-blocking and requires SPI interrupt function to indicate that reception was completed.

## Parameters

<i>handle</i>	- pointer to the MRF24J40
<i>addr</i>	- address at device's short address memory
<i>val</i>	- pointer to variable, where received value will be stored

## Returns

error code

### 5.3.3.5 MRF24J40\_Result MRF24J40\_ReceiveFrame ( MRF24J40\_HandleTypeDef \* *handle* )

Reads frame data from MRF24J40 RXFIFO register and stores it at given handle's received\_frame field.

This function is non-blocking and requires properly configured SPI interrupt function to perform successful reception.

## Parameters

<i>handle</i>	- pointer to the MRF24J40 handle
---------------	----------------------------------

## Returns

error code

**5.3.3.6 MRF24J40\_Result MRF24J40\_SetChannel ( MRF24J40\_HandleTypeDef \* *handle*, uint8\_t *channel* )**

Sends sequence to specified MRF24J40 device, allowing to change device's channel.

## Parameters

<i>handle</i>	- pointer to the MRF24J40
<i>channel</i>	- value from 11 to 26, channel to be switched to

## Returns

error code

**5.3.3.7 MRF24J40\_Result MRF24J40\_WriteLong ( MRF24J40\_HandleTypeDef \* *handle*, MRF24J40\_LongAddr *addr*, uint8\_t *val* )**

Sends sequence to specified MRF24J40 device, allowing to write value to device's long address memory (addresses from 0x220 to 0x38F) at given address.

This function is non-blocking and requires SPI interrupt function to indicate that transmission was completed.

## Parameters

<i>handle</i>	- pointer to the MRF24J40
<i>addr</i>	- address at the device's long address memory
<i>val</i>	- value to be written

## Returns

error code

**5.3.3.8 MRF24J40\_Result MRF24J40\_WriteShort ( MRF24J40\_HandleTypeDef \* *handle*, MRF24J40\_ShortAddr *addr*, uint8\_t *val* )**

Sends sequence to specified MRF24J40 device, allowing to write value to device's short address memory (addresses from 0x00 to 0x3F) at given address.

This function is non-blocking and requires SPI interrupt function to indicate that transmission was completed.

**Parameters**

<i>handle</i>	- pointer to the MRF24J40
<i>addr</i>	- address at the device's short address memory
<i>val</i>	- value to be written

**Returns**

error code

**5.4 include/pcf8574.h File Reference**

In order to use this you have to create a [PCF8574\\_HandleTypeDef](#) variable (e.g. "pcf"). Then you will set the the address based on the configuration of your chip (pins A0, A1, A2) ( `pcf.PCF_I2C_ADDRESS` ) (0 to 7), timeout ( `pcf.PCF_I2C_TIMEOUT` ) (e.g. 1000 (=1 sec)), I2C instance to use ( `pcf.i2c.Instance` ) (e.g. I2C1 or I2C2 ...), speed of the communication ( `pcf.i2c.Init.ClockSpeed` ) (e.g. 100 000 (=100kHz)).

```
#include "stm32f4xx_hal.h"
```

**Data Structures**

- struct [PCF8574\\_HandleTypeDef](#)

**Macros**

- `#define PCF8574_I2C_ADDRESS_MASK 0x40`

**Enumerations**

- enum [PCF8574\\_RESULT](#) { [PCF8574\\_OK](#), [PCF8574\\_ERROR](#) }

**Functions**

- [PCF8574\\_RESULT](#) [PCF8574\\_Init](#) ([PCF8574\\_HandleTypeDef](#) \*handle)
- [PCF8574\\_RESULT](#) [PCF8574\\_DeInit](#) ([PCF8574\\_HandleTypeDef](#) \*handle)
- [PCF8574\\_RESULT](#) [PCF8574\\_Write](#) ([PCF8574\\_HandleTypeDef](#) \*handle, uint8\_t val)
- [PCF8574\\_RESULT](#) [PCF8574\\_Read](#) ([PCF8574\\_HandleTypeDef](#) \*handle, uint8\_t \*val)

**Variables**

- uint32\_t [PCF8574\\_Type0Pins](#) []

### 5.4.1 Detailed Description

In order to use this you have to create a [PCF8574\\_HandleTypeDef](#) variable (e.g. "pcf"). Then you will set the the address based on the configuration of your chip (pins A0, A1, A2) ( `pcf.PCF_I2C_ADDRESS` ) (0 to 7), timeout ( `pcf.PCF_I2C_TIMEOUT` ) (e.g. 1000 (=1 sec)), I2C instance to use ( `pcf.i2c.Instance` ) (e.g. I2C1 or I2C2 ...), speed of the communication ( `pcf.i2c.Init.ClockSpeed` ) (e.g. 100 000 (=100kHz)).

Example: `example.c` `example_msp.c`

### 5.4.2 Enumeration Type Documentation

#### 5.4.2.1 enum PCF8574\_RESULT

Provides possible return values for the functions

Enumerator

**PCF8574\_OK** Everything went OK

**PCF8574\_ERROR** An error occurred

### 5.4.3 Function Documentation

#### 5.4.3.1 PCF8574\_RESULT PCF8574\_DeInit ( PCF8574\_HandleTypeDef \* *handle* )

Deinitializes the I2C

Parameters

<i>handle</i>	- a pointer to the PCF8574 handle
---------------	-----------------------------------

Returns

whether the function was successful or not

#### 5.4.3.2 PCF8574\_RESULT PCF8574\_Init ( PCF8574\_HandleTypeDef \* *handle* )

Initializes the I2C for communication

Parameters

<i>handle</i>	- a pointer to the PCF8574 handle
---------------	-----------------------------------

Returns

whether the function was successful or not

#### 5.4.3.3 PCF8574\_RESULT PCF8574\_Read ( PCF8574\_HandleTypeDef \* *handle*, uint8\_t \* *val* )

Reads the current state of the port of PCF8574

##### Parameters

<i>handle</i>	- a pointer to the PCF8574 handle
<i>val</i>	- a pointer to the variable that will be assigned a value from the chip

##### Returns

whether the function was successful or not

#### 5.4.3.4 PCF8574\_RESULT PCF8574\_Write ( PCF8574\_HandleTypeDef \* *handle*, uint8\_t *val* )

Writes a given value to the port of PCF8574

##### Parameters

<i>handle</i>	- a pointer to the PCF8574 handle
<i>val</i>	- a value to be written to the port

##### Returns

whether the function was successful or not

# Index

## GPIO

hd44780.h, [14](#)

## hd44780.h

GPIO, [14](#)  
LCD\_ClearDisplay, [15](#)  
LCD\_CursorOFF, [15](#)  
LCD\_CursorON, [16](#)  
LCD\_CustomChar, [16](#)  
LCD\_DIRECTION, [14](#)  
LCD\_DeInit, [16](#)  
LCD\_DisplayOFF, [16](#)  
LCD\_DisplayON, [18](#)  
LCD\_ERROR, [15](#)  
LCD\_EntryModeSet, [18](#)  
LCD\_GetBusyFlag, [18](#)  
LCD\_I2C\_WriteOut, [19](#)  
LCD\_INTERFACE, [14](#)  
LCD\_Init, [19](#)  
LCD\_NUMBER\_OF\_LINES, [14](#)  
LCD\_OK, [15](#)  
LCD\_PIN, [15](#)  
LCD\_RESULT, [15](#)  
LCD\_SetLocation, [19](#)  
LCD\_ShiftCursor, [19](#)  
LCD\_ShiftDisplay, [20](#)  
LCD\_StateLEDControl, [20](#)  
LCD\_StateWriteBit, [20](#)  
LCD\_TYPE, [15](#)  
LCD\_WaitForBusyFlag, [21](#)  
LCD\_WriteCMD, [21](#)  
LCD\_WriteDATA, [21](#)  
LCD\_WriteNumber, [21](#)  
LCD\_WriteString, [22](#)  
LCD\_WriteToDataBus, [22](#)  
PCF8574, [14](#)

## i2c

PCF8574\_HandleTypeDef, [11](#)

include/MAC\_Header\_Parser.h, [22](#)

include/MRF24J40\_Driver.h, [24](#)

include/hd44780.h, [13](#)

include/pcf8574.h, [30](#)

LCD\_ClearDisplay

hd44780.h, [15](#)

LCD\_CursorOFF

hd44780.h, [15](#)

LCD\_CursorON

hd44780.h, [16](#)

LCD\_CustomChar

hd44780.h, [16](#)

LCD\_DIRECTION

hd44780.h, [14](#)

LCD\_DeInit

hd44780.h, [16](#)

LCD\_DisplayOFF

hd44780.h, [16](#)

LCD\_DisplayON

hd44780.h, [18](#)

LCD\_ERROR

hd44780.h, [15](#)

LCD\_EntryModeSet

hd44780.h, [18](#)

LCD\_GetBusyFlag

hd44780.h, [18](#)

LCD\_I2C\_WriteOut

hd44780.h, [19](#)

LCD\_INTERFACE

hd44780.h, [14](#)

LCD\_Init

hd44780.h, [19](#)

LCD\_NUMBER\_OF\_LINES

hd44780.h, [14](#)

LCD\_OK

hd44780.h, [15](#)

LCD\_PCF8574\_HandleTypeDef, [7](#)

lcdbuf, [7](#)

NUMBER\_OF\_LINES, [7](#)

pcf8574, [8](#)

pins, [8](#)

state, [8](#)

type, [8](#)

LCD\_PIN

hd44780.h, [15](#)

LCD\_RESULT

hd44780.h, [15](#)

LCD\_SetLocation

hd44780.h, [19](#)

LCD\_ShiftCursor

hd44780.h, [19](#)

LCD\_ShiftDisplay

hd44780.h, [20](#)

LCD\_StateLEDControl

hd44780.h, [20](#)

LCD\_StateWriteBit

hd44780.h, [20](#)

LCD\_TYPE

hd44780.h, [15](#)

LCD\_WaitForBusyFlag  
     hd44780.h, [21](#)  
 LCD\_WriteCMD  
     hd44780.h, [21](#)  
 LCD\_WriteDATA  
     hd44780.h, [21](#)  
 LCD\_WriteNumber  
     hd44780.h, [21](#)  
 LCD\_WriteString  
     hd44780.h, [22](#)  
 LCD\_WriteToDataBus  
     hd44780.h, [22](#)  
 lcdbuf  
     LCD\_PCF8574\_HandleTypeDef, [7](#)  
  
 MAC\_AddressUnion, [8](#)  
 MAC\_FrameControlFieldDef, [9](#)  
 MAC\_Header\_Parser.h  
     MAC\_Parse\_Header, [23](#)  
 MAC\_HeaderDef, [9](#)  
 MAC\_Parse\_Header  
     MAC\_Header\_Parser.h, [23](#)  
 MRF24J40\_CreateHandle  
     MRF24J40\_Driver.h, [26](#)  
 MRF24J40\_Driver.h  
     MRF24J40\_CreateHandle, [26](#)  
     MRF24J40\_InitializeChip, [26](#)  
     MRF24J40\_RSSI\_CONVERT, [26](#)  
     MRF24J40\_ReadLong, [27](#)  
     MRF24J40\_ReadShort, [27](#)  
     MRF24J40\_ReceiveFrame, [27](#)  
     MRF24J40\_SetChannel, [29](#)  
     MRF24J40\_WriteLong, [29](#)  
     MRF24J40\_WriteShort, [29](#)  
 MRF24J40\_HandleDef, [10](#)  
 MRF24J40\_InitializeChip  
     MRF24J40\_Driver.h, [26](#)  
 MRF24J40\_RSSI\_CONVERT  
     MRF24J40\_Driver.h, [26](#)  
 MRF24J40\_ReadLong  
     MRF24J40\_Driver.h, [27](#)  
 MRF24J40\_ReadShort  
     MRF24J40\_Driver.h, [27](#)  
 MRF24J40\_ReceiveFrame  
     MRF24J40\_Driver.h, [27](#)  
 MRF24J40\_SetChannel  
     MRF24J40\_Driver.h, [29](#)  
 MRF24J40\_WriteLong  
     MRF24J40\_Driver.h, [29](#)  
 MRF24J40\_WriteShort  
     MRF24J40\_Driver.h, [29](#)  
  
 NUMBER\_OF\_LINES  
     LCD\_PCF8574\_HandleTypeDef, [7](#)  
  
 PCF8574  
     hd44780.h, [14](#)  
 PCF8574\_DeInit  
     pcf8574.h, [31](#)  
  
 PCF8574\_ERROR  
     pcf8574.h, [31](#)  
 PCF8574\_HandleTypeDef, [11](#)  
     i2c, [11](#)  
     PCF\_I2C\_ADDRESS, [11](#)  
     PCF\_I2C\_TIMEOUT, [11](#)  
 PCF8574\_Init  
     pcf8574.h, [31](#)  
 PCF8574\_OK  
     pcf8574.h, [31](#)  
 PCF8574\_RESULT  
     pcf8574.h, [31](#)  
 PCF8574\_Read  
     pcf8574.h, [31](#)  
 PCF8574\_Write  
     pcf8574.h, [32](#)  
 PCF\_I2C\_ADDRESS  
     PCF8574\_HandleTypeDef, [11](#)  
 PCF\_I2C\_TIMEOUT  
     PCF8574\_HandleTypeDef, [11](#)  
 pcf8574  
     LCD\_PCF8574\_HandleTypeDef, [8](#)  
 pcf8574.h  
     PCF8574\_DeInit, [31](#)  
     PCF8574\_ERROR, [31](#)  
     PCF8574\_Init, [31](#)  
     PCF8574\_OK, [31](#)  
     PCF8574\_RESULT, [31](#)  
     PCF8574\_Read, [31](#)  
     PCF8574\_Write, [32](#)  
 pins  
     LCD\_PCF8574\_HandleTypeDef, [8](#)  
  
 state  
     LCD\_PCF8574\_HandleTypeDef, [8](#)  
  
 type  
     LCD\_PCF8574\_HandleTypeDef, [8](#)