stm32-zigbee-sniffer

Generated by Doxygen 1.8.11

Contents

Chapter 1

Main Page

Sniffer for STM32F4Discovery to work with OLIMEX MRF24J40 radio. Library uses SPI to comunicate 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:

Chapter 3

File Index

3.1 File List

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

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 adressing 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.NU↩ MBER_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.NU↩ MBER_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 occured

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

handle	- 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

handle	- 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

handle	- a pointer to the LCD handle
blink	- 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

handle	- a pointer to the LCD handle
pattern	- pointer to the bit pattern of the character
address	- 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

handle	- 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

handle	- 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

handle	- 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

handle	- a pointer to the LCD handle
direction	
shift	

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

handle	- a pointer to the LCD handle
flag	- 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

handle	- 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

handle	- 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

handle	- a pointer to the LCD handle
x	- x-coordinate of the location
y	- 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

handle	- a pointer to the LCD handle
direction	- specifies the direction
steps	- 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

handle	- a pointer to the LCD handle
direction	- directions of the shift
steps	- 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

handle	- a pointer to the LCD handle
on	- 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

handle	- a pointer to the LCD handle
value	- value of the bit (0 or 1)
pin	- 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

handle	- 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

handle	- a pointer to the LCD handle
cmd	- 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

handle	- a pointer to the LCD handle
data	- 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

handle	- a pointer to the LCD handle
n	- 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

handle	- a pointer to the LCD handle
s	- 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

handle	- a pointer to the LCD handle
data	- 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 adress mode.
- #define MAC_ADDR_LONG (uint8_t)(0b11)

  MAC 64-bit adress mode.
- #define MAC_SRC_ADDR_PRESENTED(FC) ((FC).src_addr_mode)

  Determmines if source address is presented in MAC header.
- #define MAC_DST_ADDR_PRESENTED(FC) ((FC).dst_addr_mode)

  Determmines 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

mach	- a pointer to MAC Header to store parsed header data
frame	- packet or frame to parse
frame_length	- 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 Recievie MAC control register.
- #define MRF24J40_PACON2 (MRF24J40_ShortAddr)(0x18)

  MRF24J40 power amplifier control 2 register.
- #define MRF24J40_SOFTRST (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 approximized 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

handle	- pointer to the MRF24J40 handle
spi_td	- 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

handle	- 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

handle	- pointer to the MRF24J40
addr	- address at device's long address memory
val	- 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

handle	- pointer to the MRF24J40
addr	- address at device's short address memory
val	- 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

handle	- 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

handle	- pointer to the MRF24J40
channel	- 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

handle	- pointer to the MRF24J40
addr	- address at the device's long address memory
val	- 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

handle	- pointer to the MRF24J40
addr	- address at the device's short address memory
val	- 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 occured

5.4.3 Function Documentation

5.4.3.1 PCF8574_RESULT PCF8574_DeInit (PCF8574_HandleTypeDef * *handle*)

Deinitializes the I2C

Parameters

handle	- 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

handle	- 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

handle	- a pointer to the PCF8574 handle
val	- 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

handle	- a pointer to the PCF8574 handle
val	- a value to be written to the port

Returns

whether the function was successful or not

Index