Minibee - ZigBee radio Module Reference Manual

1

Generated by Doxygen 1.5.4

Fri Oct 24 21:00:20 2008

Contents

Chapter 1

Zigbee radio MC13192 or MC13202

Author:

Bc. Jiri Kubias , DCE FEL CTU 2008

1.1 Introduction

This code provides same sample how to use MC13xx2 radio in packet mode. The basic libraries for testing are writen mostly platform independent. See the function description. In some function is noted that use some platform depended parts whis should be modified. Originaly this code was developed for MC13192 radio, but now we use MC13202 radio which is almost fully compatible.

MC1319x.c - is mostly platform independent, see notes inside. This library contains basic function which contains funcion for send, recieve, set clko, read ID, set frequency and do energy detect.

MC1319xdef.h - contains register definions and bit masks for MC13xxx radio

spi.h - platform independent. Contains function prototypes and definition SPI_SPEED for setting SPI speed in Hz.

spi_LPC.c - platfor dependent. Contains inicializing function for SPI channel, all other used pins and installs ISR.

Porting to another platform: For another platform you must implement your own SPI function, set pins and install ISR. It must support the prototypes functions which is defined in spi.h. You also must implement your function for uart communication (init and send/put char).

The last tested platform is LpcEurobot which is ARM7 micproporcesor LPC2119. It also use eurobot ebb library for UART communication for ARM7. Older platform was AVR which is no longer supported due to no platform for testing. In this code is added spi_avr.c which is not fully ported.

Chapter 2

Minibee - ZigBee radio Module Data Structure Index

2.1 Minibee - ZigBee radio Module Data Structures

Here are the data structures with brief descriptions:

Chapter 3

Minibee - ZigBee radio Module File Index

3.1 Minibee - ZigBee radio Module File List

Here is a list of all files with brief descriptions:

Chapter 4

Minibee - ZigBee radio Module Data Structure Documentation

4.1 Message Struct Reference

Radio message structure.

`#include <MC1319xdef.h>`

Data Fields

- uint8_t done

  Send / recieve flag.

- uint8_t error

  error flag - ocured during send or recieveng

- uint8_t len

  data length t osend (0 to 125)

- uint8_t data [25]

  data

4.1.1 Detailed Description

Radio message structure.

4.1.2 Field Documentation

4.1.2.1 uint8_t Message::done

Send / recieve flag.

4.1.2.2 uint8_t Message::error

error flag - ocured during send or recieveng

4.1.2.3 uint8_t Message::len

data length t osend (0 to 125)

4.1.2.4 uint8_t Message::data[25]

data

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

- MC1319xdef.h

Chapter 5

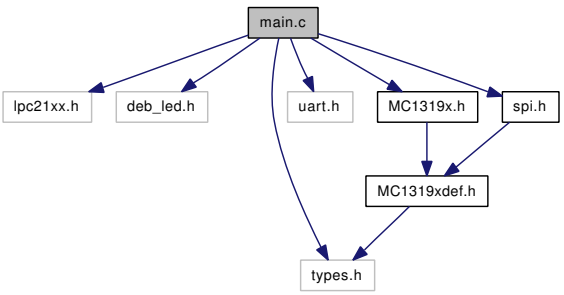
Minibee - ZigBee radio Module File Documentation

5.1 main.c File Reference

Demo application demonstrating use of MC1319x library. This file provides simply how-to use MC1319x library. From main function is called init function of SPI and UART. After sucessfull inicialize it tries read chip ID and recognize connected radio. Whan everythig is sucessfull it runs command processor which allows you to test radio functions.

```
#include <lpc21xx.h>
#include <deb_led.h>
#include <types.h>
#include <uart.h>
#include "MC1319x.h"
#include "spi.h"
```

Include dependency graph for main.c:



Functions

- void send_rs_str (char data[ ])
- void send_rs_int (int val)
- void sendData (struct Message *msg)

- void recieveData (void)
- int main (void)

Variables

- struct Message MsgSnd
  Send buffer.
- struct Message MsgRcv
  Recieve buffer.

5.1.1 Detailed Description

Demo application demonstrating use of MC1319x library. This file provides simply how-to use MC1319x library. From main function is called init function of SPI and UART. After sucessfull inicialize it tries read chip ID and recognize connected radio. Whan everythig is sucessfull it runs command processor which allows you to test radio functions.

Author:

Bc. Jiri Kubias , DCE FEL CTU 2008

5.1.2 Function Documentation

5.1.2.1 int main (void)

At first it inicializes serial and SPI line + other pins + ISR (RXTXEN, IRQ....) then it inicialzes Radio and reads radio ID. Whan everything is sucessfull runs the serial command processor.

5.1.2.2 void recieveData (void)

Reads data from Radio buffer

5.1.2.3 void send_rs_int (int *val*)

Send int value to serial output in ASCII code. Removes unused zeros.

Parameters:

val value to print

5.1.2.4 void send_rs_str (char *data*[ ])

Send string to serial output in ASCII code. .

Parameters:

data[ ] string to print

5.1.2.5 void sendData (struct Message * *msg*)

Send data via radio

Parameters:

∗***msg*** pointer to Message sructure

5.1.3 Variable Documentation

5.1.3.1 struct Message MsgRcv

Recieve buffer.

5.1.3.2 struct Message MsgSnd

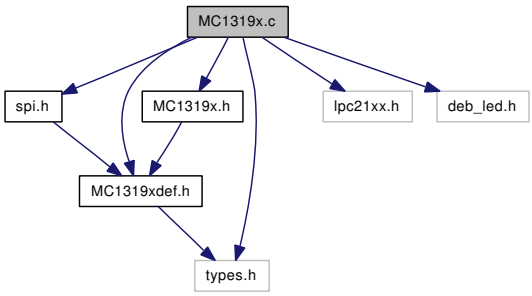
Send buffer.

5.2 MC1319x.c File Reference

Library for MC13xx2 freescale radio.

```
#include "spi.h"
#include "MC1319x.h"
#include "MC1319xdef.h"
#include <lpc21xx.h>
#include <types.h>
#include <deb_led.h>
```

Include dependency graph for MC1319x.c:



Functions

- uint8_t MC_RecievePaket (struct Message *msg)
- void ext_isr (void)

  protype if ISR function, platform dependent

- void dummy_wait ()
- void MC_Woodoo (void)
- uint8_t MC_Reset (void)
- uint8_t MC_ED (void)
- uint8_t MC_SendPaket (struct Message *msg)
- uint8_t MC_ReSendPaket (struct Message *msg)
- uint16_t MC_WhoAmI (void)
- uint8_t MC_SetChannel (uint8_t channel)
- uint8_t MC_SetClko (uint16_t tick, uint8_t enable)
- uint8_t MC_SetPa (uint8_t val)

Variables

- struct Message * rcvBuf

  Pointer to recieve Message buffer.

- struct Message * sndBuf

  Pointer to send Message buffer.

5.2.1 Detailed Description

Library for MC13xx2 freescale radio.

Author:

Bc. Jiri Kubias , DCE FEL CTU 2008

5.2.2 Function Documentation

5.2.2.1 void dummy_wait ()

Delay function.

Note:

Should be removed in future

5.2.2.2 void ext_isr (void)

protype if ISR function, platform dependent

Interupt handling. Here is reading IRQ register and deremine what generates IRQ

Note:

This function is platform depended

5.2.2.3 uint8_t MC_ED (void)

Scan selected radio channel and returns measured energy (lower is better)

Note:

This function is not using ISR, but in future it may do.

5.2.2.4 uint8_t MC_RecievePaket (struct Message * *msg*)

Reads packet stored in radio buffer

Parameters:

msg pointer to Message structure

5.2.2.5 uint8_t MC_ReSendPaket (struct Message * *msg*)

Resend the packed stored in radio transmit buffer

Parameters:

msg pointer to Message structure

Note:

Not sure if it is working

5.2.2.6 uint8_t MC_Reset (void)

Reset the radio to it default state and inicialize it

5.2.2.7 uint8_t MC_SendPaket (struct Message * *msg*)

Sends the packed stored in Message structure

Parameters:

msg ponter to Message structure

5.2.2.8 uint8_t MC_SetChannel (uint8_t *channel*)

Sets radio frequency channel

Parameters:

channel specifies frequency output (1∼16)

Note:

See definions from MC1319xdef.h

5.2.2.9 uint8_t MC_SetClko (uint16_t *tick*, uint8_t *enable*)

Sets CLKO pin to specified clock output

Parameters:

tick specifies frequency output

enable enable or disable output

Note:

Use definions from MC1319xdef.h

5.2.2.10 uint8_t MC_SetPa (uint8_t *val*)

Im not remeber what is this doing.

5.2.2.11 uint16_t MC_WhoAmI (void)

Returns radio identification

5.2.2.12 void MC_Woodoo (void)

Woodoo function inicialize radio. I hove no idea what it is doing, but its VERY inportatnt.

5.2.3 Variable Documentation

5.2.3.1 struct Message* rcvBuf

Pointer to recieve Message buffer.

5.2.3.2 struct Message* sndBuf

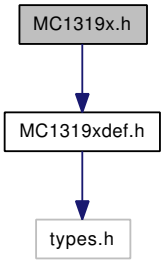
Pointer to send Message buffer.

5.3 MC1319x.h File Reference

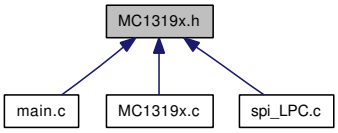
Function prototypes for MC1319x.c.

```
#include "MC1319xdef.h"
```

Include dependency graph for MC1319x.h:



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



Functions

- uint8_t MC_Reset (void)
- uint16_t MC_WhoAmI (void)
- uint8_t MC_ED (void)
- uint8_t MC_SendPaket (struct Message *msg)
- uint8_t MC_ReSend (struct Message *msg)
- uint8_t MC_RecievePaket (struct Message *msg)
- uint8_t MC_SetClko (uint16_t tick, uint8_t enable)
- uint8_t MC_SetPa (uint8_t val)
- void dummy_wait ()
- void ext_isr (void) __attribute__((interrupt))

  protype if ISR function, platform dependent

5.3.1 Detailed Description

Function prototypes for MC1319x.c.

Author:

Bc. Jiri Kubias , DCE FEL CTU 2008

5.3.2 Function Documentation

5.3.2.1 void dummy_wait ()

Delay function.

Note:

Should be removed in future

5.3.2.2 void ext_isr (void)

protype if ISR function, platform dependent

Interupt handling. Here is reading IRQ register and deremine what generates IRQ

Note:

This function is platform depended

5.3.2.3 uint8_t MC_ED (void)

Scan selected radio channel and returns measured energy (lower is better)

Note:

This function is not using ISR, but in future it may do.

5.3.2.4 uint8_t MC_RecievePaket (struct Message * *msg*)

5.3.2.5 uint8_t MC_ReSend (struct Message * *msg*)

5.3.2.6 uint8_t MC_Reset (void)

Reset the radio to it default state and inicialize it

5.3.2.7 uint8_t MC_SendPaket (struct Message * *msg*)

Sends the packed stored in Message structure

Parameters:

msg ponter to Message structure

5.3.2.8 uint8_t MC_SetClko (uint16_t *tick*, uint8_t *enable*)

Sets CLKO pin to specified clock output

Parameters:

tick specifies frequency output

enable enable or disable output

Note:

Use definions from MC1319xdef.h

5.3.2.9 uint8_t MC_SetPa (uint8_t *val*)

Im not remeber what is this doing.

5.3.2.10 uint16_t MC_WhoAmI (void)

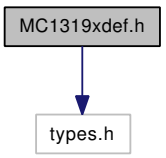
Returns radio identification

5.4 MC1319xdef.h File Reference

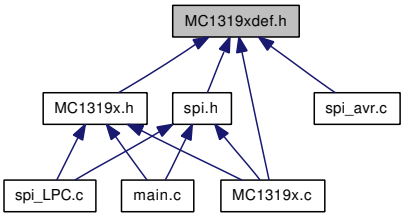
Radio register definition and platform specific definitions.

`#include <types.h>`

Include dependency graph for MC1319xdef.h:



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



Data Structures

- struct Message

  Radio message structure.

Defines

- #define LPC

  Select LPC as master MCU.

- #define IRQ 9

  Inverted , input, ISR request.

- #define CE 7

  slave select , CE on MC radio

- #define ATTN 11

  Inverted , output.

- #define RXTXEN 8

  enable RX or TX operations

- #define RST 12

  Inverted , output.

- #define MOSI 6

  SPI - master output.

- #define MISO 5

  SPI - master input.

- #define SCK 4

  SPI - master clock.

- #define CLR(x) { IO0CLR |= (1<<(x)); }

  clear pin command

- #define SET(x) { IO0SET |= (1<<(x)); }

  set pin command

- #define RD 0x80
- #define WR 0x00
- #define RESET 0x00
- #define RX_PKT_RAM 0x01
- #define TX_PKT_RAM 0x02
- #define TX_PKT_CTL 0x03
- #define TX_PKT_CTL_RAM2_SELm 15
- #define TX_PKT_CTL_PKT_LENGHT(x) (x + 2)
- #define CCA_THRESH 0x04
- #define IRQ_MASK 0x05
- #define IRQ_MASK_ATTNm 0x8000
- #define IRQ_MASK_RAM_ADDRm 0x1000
- #define IRQ_MASK_ARB_BUSYm 0x0800
- #define IRQ_MASK_STRM_DATAm 0x0400
- #define IRQ_MASK_PPL_LOCKm 0x0200
- #define IRQ_MASK_ACOMAm 0x0100
- #define IRQ_MASK_DOZEm 0x0010
- #define IRQ_MASK_TMR4m 0x0008
- #define IRQ_MASK_TMR3m 0x0004
- #define IRQ_MASK_TMR2m 0x0002
- #define IRQ_MASK_TMR1m 0x0001
- #define CONTROL_A 0x06
- #define CONTROL_A_TX_STRMm 0x1000
- #define CONTROL_A_RX_STRMm 0x0800
- #define CONTROL_A_CCAm 0x0400
- #define CONTROL_A_TX_SENTm 0x0200
- #define CONTROL_A_RX_RCVDm 0x0100
- #define CONTROL_A_TMR_TRIG_ENm 0x0080
- #define CONTROL_A_CCA_TYPE1m 0x0020
- #define CONTROL_A_CCA_TYPE0m 0x0010
- #define CONTROL_A_CCA_CCAm CONTROL_A_CCA_TYPE0m
- #define CONTROL_A_CCA_EDm CONTROL_A_CCA_TYPE1m
- #define CONTROL_A_CCA_XCVR_SEG1m 0x0002
- #define CONTROL_A_CCA_XCVR_SEG0m 0x0001

- #define CONTROL_A_CCA_XCVR_IDLEm 0
- #define CONTROL_A_CCA_XCVR_CCAEDm CONTROL_A_CCA_XCVR_SEG0m
- #define CONTROL_A_CCA_XCVR_PMRXm CONTROL_A_CCA_XCVR_SEG1m
- #define CONTROL_A_CCA_XCVR_PMTXm (CONTROL_A_CCA_XCVR_SEG1m | CONTROL_A_CCA_XCVR_SEG0m)
- #define CONTROL_A_CCA_XCVR_CLRm (CONTROL_A_CCA_XCVR_SEG1m | CONTROL_A_CCA_XCVR_SEG0m)
- #define CONTROL_B 0x07
- #define CONTROL_B_TMR_LOADm 0x8000
- #define CONTROL_B_MISO_HIZ_ENm 0x0800
- #define CONTROL_B_CLKO_DOZE_ENm 0x0200
- #define CONTROL_B_TX_DONEm 0x0080
- #define CONTROL_B_RX_DONEm 0x0040
- #define CONTROL_B_USE_STM_MODEm 0x0020
- #define CONTROL_B_HIB_ENm 0x0002
- #define CONTROL_B_DOZE_ENm 0x0001
- #define PA_ENABLE 0x08
- #define PA_ENABLE_PA_ENm 0x8000
- #define CONTROL_C 0x09
- #define CONTROL_C_GPIO_ALT_ENm 0x0080
- #define CONTROL_C_CLKO_ENm 0x0020
- #define CONTROL_C_TMR_PRESCALE2m 0x0004
- #define CONTROL_C_TMR_PRESCALE1m 0x0002
- #define CONTROL_C_TMR_PRESCALE0m 0x0001
- #define CLKO_CTL 0x0A
- #define CLKO_CTL_CLKO_RATE2m 0x0004
- #define CLKO_CTL_CLKO_RATE1m 0x0002
- #define CLKO_CTL_CLKO_RATE0m 0x0001
- #define CLKO_CTL_CLKO_16Mm 0x00
- #define CLKO_CTL_CLKO_8Mm CLKO_CTL_CLKO_RATE0m
- #define CLKO_CTL_CLKO_4Mm CLKO_CTL_CLKO_RATE1m
- #define CLKO_CTL_CLKO_2Mm (CLKO_CTL_CLKO_RATE0m | CLKO_CTL_CLKO_RATE1m)
- #define CLKO_CTL_CLKO_1Mm CLKO_CTL_CLKO_RATE2m
- #define CLKO_CTL_CLKO_62Km (CLKO_CTL_CLKO_RATE0m | CLKO_CTL_CLKO_RATE2m)
- #define CLKO_CTL_CLKO_32Km (CLKO_CTL_CLKO_RATE1m | CLKO_CTL_CLKO_RATE2m)
- #define CLKO_CTL_CLKO_16Km (CLKO_CTL_CLKO_RATE0m | CLKO_CTL_CLKO_RATE1m| CLKO_CTL_CLKO_RATE2m)
- #define CLKO_CTL_CLKO_ONm 0x01
- #define CLKO_CTL_CLKO_OFFm 0x00
- #define CLKO_CTL_CLKO_CLRm (CLKO_CTL_CLKO_RATE0m | CLKO_CTL_CLKO_RATE1m| CLKO_CTL_CLKO_RATE2m)
- #define GPIO_DIR 0x0B
- #define GPIO_DATA_OUT 0x0C
- #define LO1_INT_DIV 0x0F
- #define LO1_INT_DIV_CH1 0x95
- #define LO1_INT_DIV_CH2 0x95
- #define LO1_INT_DIV_CH3 0x95
- #define LO1_INT_DIV_CH4 0x96

- #define LO1_INT_DIV_CH5 0x96
- #define LO1_INT_DIV_CH6 0x96
- #define LO1_INT_DIV_CH7 0x97
- #define LO1_INT_DIV_CH8 0x97
- #define LO1_INT_DIV_CH9 0x97
- #define LO1_INT_DIV_CH10 0x98
- #define LO1_INT_DIV_CH11 0x98
- #define LO1_INT_DIV_CH12 0x98
- #define LO1_INT_DIV_CH13 0x99
- #define LO1_INT_DIV_CH14 0x99
- #define LO1_INT_DIV_CH15 0x99
- #define LO1_INT_DIV_CH16 0x9A
- #define LO1_NUM 0x10
- #define LO1_NUM_CH1 0x5000
- #define LO1_NUM_CH2 0xA000
- #define LO1_NUM_CH3 0xF000
- #define LO1_NUM_CH4 0x4000
- #define LO1_NUM_CH5 0x9000
- #define LO1_NUM_CH6 0xE000
- #define LO1_NUM_CH7 0x3000
- #define LO1_NUM_CH8 0x8000
- #define LO1_NUM_CH9 0xD000
- #define LO1_NUM_CH10 0x2000
- #define LO1_NUM_CH11 0x7000
- #define LO1_NUM_CH12 0xC000
- #define LO1_NUM_CH13 0x1000
- #define LO1_NUM_CH14 0x6000
- #define LO1_NUM_CH15 0xB000
- #define LO1_NUM_CH16 0x0000
- #define ZB_CH1 1
- #define ZB_CH2 2
- #define ZB_CH3 3
- #define ZB_CH4 4
- #define ZB_CH5 5
- #define ZB_CH6 6
- #define ZB_CH7 7
- #define ZB_CH8 8
- #define ZB_CH9 9
- #define ZB_CH10 10
- #define ZB_CH11 11
- #define ZB_CH12 12
- #define ZB_CH13 13
- #define ZB_CH14 14
- #define ZB_CH15 15
- #define ZB_CH16 16
- #define ZB_CH802_11 ZB_CH1
- #define ZB_CH802_12 ZB_CH2
- #define ZB_CH802_13 ZB_CH3
- #define ZB_CH802_14 ZB_CH4
- #define ZB_CH802_15 ZB_CH5

- #define ZB_CH802_16 ZB_CH6
- #define ZB_CH802_17 ZB_CH7
- #define ZB_CH802_18 ZB_CH8
- #define ZB_CH802_19 ZB_CH9
- #define ZB_CH802_20 ZB_CH10
- #define ZB_CH802_21 ZB_CH11
- #define ZB_CH802_22 ZB_CH12
- #define ZB_CH802_23 ZB_CH13
- #define ZB_CH802_24 ZB_CH14
- #define ZB_CH802_25 ZB_CH15
- #define ZB_CH802_26 ZB_CH16
- #define PA_LVL 0x12
- #define TMR_CMP1_A 0x1B
- #define TMR_CMP1_B 0x1C
- #define TMR_CMP2_A 0x1D
- #define TMR_CMP2_B 0x1E
- #define TMR_CMP3_A 0x1F
- #define TMR_CMP3_B 0x20
- #define TMR_CMP4_A 0x21
- #define TMR_CMP4_B 0x22
- #define TC2_PRIME 0x23
- #define IRQ_STATUS 0x24
- #define IRQ_STATUS_PLL_LOCK_IRQm 0x8000
- #define IRQ_STATUS_RAM_ADR_ERRm 0x4000
- #define IRQ_STATUS_ARB_BUSY_ERRm 0x2000
- #define IRQ_STATUS_SRTM_DATA_ERRm 0x1000
- #define IRQ_STATUS_ATTN_IRQm 0x0400
- #define IRQ_STATUS_DOZE_IRQm 0x0200
- #define IRQ_STATUS_TMR1_IRQm 0x0100
- #define IRQ_STATUS_RX_RCVD_IRQm 0x0080
- #define IRQ_STATUS_TX_SENT_IRQm 0x0040
- #define IRQ_STATUS_CCA_IRQm 0x0020
- #define IRQ_STATUS_TMR3_IRQm 0x0010
- #define IRQ_STATUS_TMR4_IRQm 0x0008
- #define IRQ_STATUS_TMR2_IRQm 0x0004
- #define IRQ_STATUS_CCAm 0x0002
- #define IRQ_STATUS_CRC_VALIDm 0x0001
- #define RST_IND 0x25
- #define RST_IND_RESET_INDm 0x0080
- #define CURRENT_TIME_A 0x26
- #define CURRENT_TIME_B 0x27
- #define GPIO_DATA_IN 0x28
- #define CHIP_ID 0x2C
- #define RX_STATUS 0x2D
- #define TIMESTAMP_A 0x2E
- #define TIMESTAMP_B 0x2F
- #define BER_ENABLE 0x30
- #define BER_ENABLE_BER_EN 0x8000

5.4.1 Detailed Description

Radio register definition and platform specific definitions.

Author:

Bc. Jiri Kubias , DCE FEL CTU 2008

5.4.2 Define Documentation

5.4.2.1 #define ATTN 11

Inverted , output.

5.4.2.2 #define BER_ENABLE 0x30

5.4.2.3 #define BER_ENABLE_BER_EN 0x8000

5.4.2.4 #define CCA_THRESH 0x04

5.4.2.5 #define CE 7

slave select , CE on MC radio

5.4.2.6 #define CHIP_ID 0x2C

5.4.2.7 #define CLKO_CTL 0x0A

5.4.2.8 #define CLKO_CTL_CLKO_16Km (CLKO_CTL_CLKO_RATE0m | CLKO_CTL_CLKO_RATE1m| CLKO_CTL_CLKO_RATE2m)

5.4.2.9 #define CLKO_CTL_CLKO_16Mm 0x00

5.4.2.10 #define CLKO_CTL_CLKO_1Mm CLKO_CTL_CLKO_RATE2m

5.4.2.11 #define CLKO_CTL_CLKO_2Mm (CLKO_CTL_CLKO_RATE0m | CLKO_CTL_CLKO_RATE1m)

5.4.2.12 #define CLKO_CTL_CLKO_32Km (CLKO_CTL_CLKO_RATE1m | CLKO_CTL_CLKO_RATE2m)

5.4.2.13 #define CLKO_CTL_CLKO_4Mm CLKO_CTL_CLKO_RATE1m

5.4.2.14 #define CLKO_CTL_CLKO_62Km (CLKO_CTL_CLKO_RATE0m | CLKO_CTL_CLKO_RATE2m)

5.4.2.15 #define CLKO_CTL_CLKO_8Mm CLKO_CTL_CLKO_RATE0m

5.4.2.16 #define CLKO_CTL_CLKO_CLRm (CLKO_CTL_CLKO_RATE0m | CLKO_CTL_CLKO_RATE1m| CLKO_CTL_CLKO_RATE2m)

5.4.2.17 #define CLKO_CTL_CLKO_OFFm 0x00

5.4.2.18 #define CLKO_CTL_CLKO_ONm 0x01

5.4.2.19 #define CLKO_CTL_CLKO_RATE0m 0x0001

5.4.2.20 #define CLKO_CTL_CLKO_RATE1m 0x0002

5.4.2.21 #define CLKO_CTL_CLKO_RATE2m 0x0004

5.4.2.22 #define CLR(x) { IO0CLR |= (1<<(x)); }

clear pin command

5.4.2.23 #define CONTROL_A 0x06

5.4.2.24 #define CONTROL_A_CCA_CCAm CONTROL_A_CCA_TYPE0m

5.4.2.25 #define CONTROL_A_CCA_EDm CONTROL_A_CCA_TYPE1m

5.4.2.26 #define CONTROL_A_CCA_TYPE0m 0x0010

5.4.2.27 #define CONTROL_A_CCA_TYPE1m 0x0020

5.4.2.28 #define CONTROL_A_CCA_XCVR_CCAEDm CONTROL_A_CCA_XCVR_SEG0m

5.4.2.29 #define CONTROL_A_CCA_XCVR_CLRm (CONTROL_A_CCA_XCVR_SEG1m | CONTROL_A_CCA_XCVR_SEG0m)

5.4.2.30 #define CONTROL_A_CCA_XCVR_IDLEm 0

5.4.2.31 #define CONTROL_A_CCA_XCVR_PMRXm CONTROL_A_CCA_XCVR_SEG1m

5.4.2.32 #define CONTROL_A_CCA_XCVR_PMTXm (CONTROL_A_CCA_XCVR_SEG1m | CONTROL_A_CCA_XCVR_SEG0m)

5.4.2.33 #define CONTROL_A_CCA_XCVR_SEG0m 0x0001

5.4.2.34 #define CONTROL_A_CCA_XCVR_SEG1m 0x0002

5.4.2.35 #define CONTROL_A_CCAm 0x0400

5.4.2.36 #define CONTROL_A_RX_RCVDm 0x0100

5.4.2.37 #define CONTROL_A_RX_STRMm 0x0800

5.4.2.38 #define CONTROL_A_TMR_TRIG_ENm 0x0080

5.4.2.39 #define CONTROL_A_TX_SENTm 0x0200

5.4.2.40 #define CONTROL_A_TX_STRMm 0x1000

5.4.2.41 #define CONTROL_B 0x07

5.4.2.42 #define CONTROL_B_CLKO_DOZE_ENm 0x0200

5.4.2.43 #define CONTROL_B_DOZE_ENm 0x0001

5.4.2.44 #define CONTROL_B_HIB_ENm 0x0002

5.4.2.45 #define CONTROL_B_MISO_HIZ_ENm 0x0800

5.4.2.46 #define CONTROL_B_RX_DONEm 0x0040

5.4.2.47 #define CONTROL_B_TMR_LOADm 0x8000

5.4.2.48 #define CONTROL_B_TX_DONEm 0x0080

5.4.2.49 #define CONTROL_B_USE_STM_MODEm 0x0020

5.4.2.50 #define CONTROL_C 0x09

5.4.2.51 #define CONTROL_C_CLKO_ENm 0x0020

5.4.2.52 #define CONTROL_C_GPIO_ALT_ENm 0x0080

5.4.2.62 #define IRQ_MASK 0x05

5.4.2.63 #define IRQ_MASK_ACOMAm 0x0100

5.4.2.64 #define IRQ_MASK_ARB_BUSYm 0x0800

5.4.2.65 #define IRQ_MASK_ATTNm 0x8000

5.4.2.66 #define IRQ_MASK_DOZEm 0x0010

5.4.2.67 #define IRQ_MASK_PPL_LOCKm 0x0200

5.4.2.68 #define IRQ_MASK_RAM_ADDRm 0x1000

5.4.2.69 #define IRQ_MASK_STRM_DATAm 0x0400

5.4.2.70 #define IRQ_MASK_TMR1m 0x0001

5.4.2.71 #define IRQ_MASK_TMR2m 0x0002

5.4.2.72 #define IRQ_MASK_TMR3m 0x0004

5.4.2.73 #define IRQ_MASK_TMR4m 0x0008

5.4.2.74 #define IRQ_STATUS 0x24

5.4.2.75 #define IRQ_STATUS_ARB_BUSY_ERRm 0x2000

5.4.2.76 #define IRQ_STATUS_ATTN_IRQm 0x0400

5.4.2.77 #define IRQ_STATUS_CCA_IRQm 0x0020

5.4.2.78 #define IRQ_STATUS_CCAm 0x0002

5.4.2.79 #define IRQ_STATUS_CRC_VALIDm 0x0001

5.4.2.80 #define IRQ_STATUS_DOZE_IRQm 0x0200

5.4.2.81 #define IRQ_STATUS_PLL_LOCK_IRQm 0x8000

5.4.2.82 #define IRQ_STATUS_RAM_ADR_ERRm 0x4000

5.4.2.83 #define IRQ_STATUS_RX_RCVD_IRQm 0x0080

5.4.2.84 #define IRQ_STATUS_SRTM_DATA_ERRm 0x1000

5.4.2.85 #define IRQ_STATUS_TMR1_IRQm 0x0100

5.4.2.86 #define IRQ_STATUS_TMR2_IRQm 0x0004

5.4.2.87 #define IRQ_STATUS_TMR3_IRQm 0x0010

5.4.2.88 #define IRQ_STATUS_TMR4_IRQm 0x0008

5.4.2.89 #define IRQ_STATUS_TX_SENT_IRQm 0x0040

5.4.2.90 #define LO1_INT_DIV 0x0F

5.4.2.91 #define LO1_INT_DIV_CH1 0x95

5.4.2.92 #define LO1_INT_DIV_CH10 0x98

5.4.2.125 #define MISO 5

SPI - master input.

5.4.2.126 #define MOSI 6

SPI - master output.

5.4.2.127 #define PA_ENABLE 0x08

5.4.2.128 #define PA_ENABLE_PA_ENm 0x8000

5.4.2.129 #define PA_LVL 0x12

5.4.2.130 #define RD 0x80

5.4.2.131 #define RESET 0x00

5.4.2.132 #define RST 12

Inverted , output.

5.4.2.133 #define RST_IND 0x25

5.4.2.134 #define RST_IND_RESET_INDm 0x0080

5.4.2.135 #define RX_PKT_RAM 0x01

5.4.2.136 #define RX_STATUS 0x2D

5.4.2.137 #define RXTXEN 8

enable RX or TX operations

5.4.2.138 #define SCK 4

SPI - master clock.

5.4.2.139 #define SET(x) { IO0SET |= (1<<(x)); }

set pin command

5.4.2.140 #define TC2_PRIME 0x23

5.4.2.141 #define TIMESTAMP_A 0x2E

5.4.2.142 #define TIMESTAMP_B 0x2F

5.4.2.143 #define TMR_CMP1_A 0x1B

5.4.2.144 #define TMR_CMP1_B 0x1C

5.4.2.145 #define TMR_CMP2_A 0x1D

5.4.2.146 #define TMR_CMP2_B 0x1E

5.4.2.147 #define TMR_CMP3_A 0x1F

5.4.2.148 #define TMR_CMP3_B 0x20

5.4.2.149 #define TMR_CMP4_A 0x21

5.4.2.150 #define TMR_CMP4_B 0x22

5.4.2.151 #define TX_PKT_CTL 0x03

5.4.2.152 #define TX_PKT_CTL_PKT_LENGHT(x) (x + 2)

5.4.2.153 #define TX_PKT_CTL_RAM2_SELm 15

5.4.2.154 #define TX_PKT_RAM 0x02

5.4.2.155 #define WR 0x00

5.4.2.156 #define ZB_CH1 1

5.4.2.157 #define ZB_CH10 10

5.4.2.158 #define ZB_CH11 11

5.4.2.159 #define ZB_CH12 12

5.4.2.160 #define ZB_CH13 13

5.4.2.161 #define ZB_CH14 14

5.4.2.162 #define ZB_CH15 15

5.4.2.163 #define ZB_CH16 16

5.4.2.164 #define ZB_CH2 2

5.4.2.165 #define ZB_CH3 3

5.4.2.166 #define ZB_CH4 4

5.4.2.167 #define ZB_CH5 5

5.4.2.168 #define ZB_CH6 6

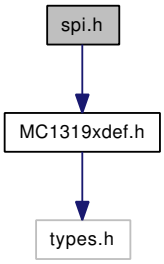
5.4.2.169 #define ZB_CH7 7

5.4.2.170 #define ZB_CH8 8

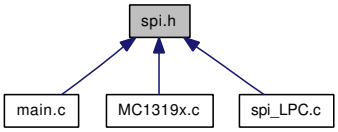
5.5 spi.h File Reference

```
#include "MC1319xdef.h"
```

Include dependency graph for spi.h:



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



Defines

- #define SPI_SPEED 200

  SPI frequency in Hz.

Functions

- void spi_Init (int rx_isr_vect)
- uint16_t spi_Read (uint8_t reg)
- void spi_Write (uint8_t reg, uint16_t val)
- void spi_ReadModifiWrite (uint8_t reg, uint16_t ormask, uint16_t andmask)
- void spi_Write_Buf (struct Message *msg)
- void spi_Read_Buf (struct Message *msg)
- uint8_t MC_SetChannel (uint8_t channel)
- void disable_IRQ_pin (void)
- void enable_IRQ_pin (void)

5.5.1 Define Documentation

5.5.1.1 #define SPI_SPEED 200

SPI frequency in Hz.

5.5.2 Function Documentation

5.5.2.1 void disable_IRQ_pin (void)

Disabling interupt from radio IRQ pin

5.5.2.2 void enable_IRQ_pin (void)

Enabling interupt from radio IRQ pin

5.5.2.3 uint8_t MC_SetChannel (uint8_t *channel*)

Sets radio frequency channel

Parameters:

channel specifies frequency output (1∼16)

Note:

See definions from MC1319xdef.h

5.5.2.4 void spi_Init (int *rx_isr_vect*)

Inits SPI channel to speed given in SPI_SPEED. Also preset other pins (ATTN,IRQ, TXRXEN, ATTN). Initiates ISR for IRQ PIn, but it is not enabled.

5.5.2.5 uint16_t spi_Read (uint8_t *reg*)

Reads one register in radio

Parameters:

reg register to read

Returns:

value in rgister

5.5.2.6 void spi_Read_Buf (struct Message * *msg*)

Read recieved data to Message structure

Parameters:

∗***msg*** pointer to Messge structure

5.5.2.7 void spi_ReadModifiWrite (uint8_t *reg*, uint16_t *ormask*, uint16_t *andmask*)

Read-modify write function for modifiing registers in radio

Parameters:

reg register to modify

ormask OR mask

andmask AND mask

5.5.2.8 void spi_Write (uint8_t *reg*, uint16_t *val*)

Write value to one register in radio

Parameters:

reg register to write

val value to write

5.5.2.9 void spi_Write_Buf (struct Message * *msg*)

Write messgage to internal radio buffer

Parameters:

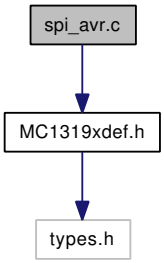
∗***msg*** pointer to Message structure

5.6 spi_avr.c File Reference

SPI, IRQ and pin inicialization, platform dependent! Contains support functions for SPI communiacations. This portage is not fully ported, some functions are missing. This part is no longer supprted.

```
#include "MC1319xdef.h"
```

Include dependency graph for spi_avr.c:



Functions

- void spi_Init (void)
- uint16_t spi_Read (uint8_t reg)
- void spi_Write (uint8_t reg, uint16_t val)
- void spi_Read_Buf (struct Message *msg)
- void spi_Write_Buf (struct Message *msg)
- void spi_ReadModifiWrite (uint8_t reg, uint16_t ormask, uint16_t andmask)

5.6.1 Detailed Description

SPI, IRQ and pin inicialization, platform dependent! Contains support functions for SPI communiacations. This portage is not fully ported, some functions are missing. This part is no longer supprted.

Author:

Bc. Jiri Kubias , DCE FEL CTU 2008

5.6.2 Function Documentation

5.6.2.1 void spi_Init (void)

5.6.2.2 uint16_t spi_Read (uint8_t *reg*)

5.6.2.3 void spi_Read_Buf (struct Message * *msg*)

5.6.2.4 void spi_ReadModifiWrite (uint8_t *reg*, uint16_t *ormask*, uint16_t *andmask*)

5.6.2.5 void spi_Write (uint8_t *reg*, uint16_t *val*)

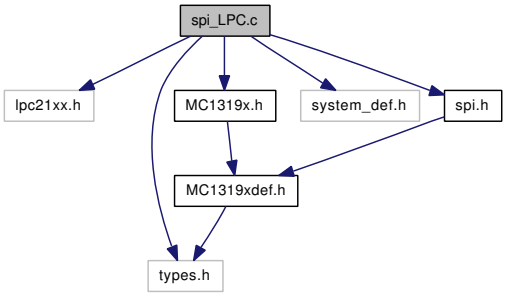
5.6.2.6 void spi_Write_Buf (struct Message * *msg*)

5.7 spi_LPC.c File Reference

SPI, IRQ and pin inicialization function prototypes A new platform must support this functions.

```
#include <lpc21xx.h>
#include <types.h>
#include "MC1319x.h"
#include <system_def.h>
#include "spi.h"
```

Include dependency graph for spi_LPC.c:



Functions

- void disable_IRQ_pin (void)
- void enable_IRQ_pin (void)
- void spi_Init (int rx_isr_vect)
- uint16_t spi_Read (uint8_t reg)
- void spi_Write (uint8_t reg, uint16_t val)
- void spi_Read_Buf (struct Message *msg)
- void spi_Write_Buf (struct Message *msg)
- void spi_ReadModifiWrite (uint8_t reg, uint16_t ormask, uint16_t andmask)

5.7.1 Detailed Description

SPI, IRQ and pin inicialization function prototypes A new platform must support this functions.

SPI, IRQ and pin inicialization, platform dependent! Contains support functions for SPI communiacations.

Author:

Bc. Jiri Kubias , DCE FEL CTU 2008

5.7.2 Function Documentation

5.7.2.1 void disable_IRQ_pin (void)

Disabling interupt from radio IRQ pin

5.7.2.2 void enable_IRQ_pin (void)

Enabling interupt from radio IRQ pin

5.7.2.3 void spi_Init (int *rx_isr_vect*)

Inits SPI channel to speed given in SPI_SPEED. Also preset other pins (ATTN,IRQ, TXRXEN, ATTN). Initiates ISR for IRQ PIn, but it is not enabled.

5.7.2.4 uint16_t spi_Read (uint8_t *reg*)

Reads one register in radio

Parameters:

reg register to read

Returns:

value in rgister

5.7.2.5 void spi_Read_Buf (struct Message * *msg*)

Read recieved data to Message structure

Parameters:

∗***msg*** pointer to Messge structure

5.7.2.6 void spi_ReadModifiWrite (uint8_t *reg*, uint16_t *ormask*, uint16_t *andmask*)

Read-modify write function for modifiing registers in radio

Parameters:

reg register to modify

ormask OR mask

andmask AND mask

5.7.2.7 void spi_Write (uint8_t *reg*, uint16_t *val*)

Write value to one register in radio

Parameters:

reg register to write

val value to write

5.7.2.8 void spi_Write_Buf (struct Message * *msg*)

Write messgage to internal radio buffer

Parameters:

∗***msg*** pointer to Message structure

Index