SOCIAL WIRELESS - STM32F4

Generated by Doxygen 1.8.4

Mon Mar 31 2014 18:57:56

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

Class Documentation

2.1 NRF24L01P Class Reference

Public Member Functions

- NRF24L01P ()
- **NRF24L01P** (const NRF24L01P &orig)
- void writeRegister (int regAddress, int regData)
- int readRegister (int regAddress)
- int readStatusRegister ()
- void powerUp ()
- void powerDown ()
- void setReceiveMode ()
- void setTransmitMode ()
- void setStaticPayloadSize (int size)
- void disableAllAutoAck ()
- void maskIrq (int w)
- void returnStandByI ()
- int receiveDataFromRx ()
- int readRPD ()
- void configureInterrupt ()
- void configureWidthAddress (int n_byte)
- void flushTx ()
- void flushRx ()
- void TrasmitData (char *data_pointer, int dim)
- void resetModule ()
- void showInternal ()
- void setRfChannel (int channel)
- void setAirDataRate (int air_rate)
- void resetRXirq ()
- void resetTXirq ()

2.1.1 Constructor & Destructor Documentation

2.1.1.1 NRF24L01P::NRF24L01P ( )

Constructor of transceiver

2.1.2 Member Function Documentation

2.1.2.1 void NRF24L01P::configureInterrupt ( )

Configuration of the interrupt in the PA1 pin of STM32 board

2.1.2.2 void NRF24L01P::configureWidthAddress (int *number*)

Configure the width of the TX/RX-Address If I want 5 bytes the n_bit is 3 (11 will be in the register) 4 bytes the n_bit is 2 (10) 3 bytes the n_bit is 1 (01) (00 Illegal)

Parameters

number	represent the number of bytes that we want

2.1.2.3 void NRF24L01P::disableAllAutoAck ( )

Disable all auto-ack on all pipe (this function modify the EN_AA register)

2.1.2.4 void NRF24L01P::flushRx ( )

Function that flush the content of RX-BUFFER

2.1.2.5 void NRF24L01P::flushTx ( )

Function that flush the content of TX-BUFFER

2.1.2.6 void NRF24L01P::maskIrq (int *w*)

Function used to mask the wanted irq 000 no masking 010 mask tx_ds 001 mask max_rt 100 mask rx_dr (pointless) ecc...

Parameters

w	is the masking option that we want to activate

2.1.2.7 void NRF24L01P::powerDown ( )

PowerDown the module putting 0 the PWR_UP bit in CONFIG register

2.1.2.8 void NRF24L01P::powerUp ( )

This function put the transceiver in StandBy I mode (see charts) configuring the various register and waiting the right time to power on.

2.1.2.9 int NRF24L01P::readRegister (int *regAddress*)

This function is used to read a general register in the NRF24L01P

Parameters

regAddress,:	the address of the register that you want to read

Returns

: the value in the register (alias its current config)

2.1.2.10 int NRF24L01P::readRPD ( )

This function is used to read the RPD bit that tells if a valid RF signal >-64dbm is detected. This means that a NRF24L01p is near us. Remember that this feature is available only in receive mode

Returns

is the status of RPD bit in RPD register

2.1.2.11 int NRF24L01P::readStatusRegister ( )

Shortcut to read the status register of transceiver the status register is passed via miso after a dummy write on mosi

Returns

: the value of status register of NRF24L01P

2.1.2.12 int NRF24L01P::receiveDataFromRx ( )

used to take data from RX_FIFO_BUFFER through SPI (using the command _NRF24L01P_SPI_CMD_RD_RX_-PAYLOAD)

Returns

the data read

2.1.2.13 void NRF24L01P::resetModule ( )

Reset default value to all registers of module function used by exorcizo binary

2.1.2.14 void NRF24L01P::resetRXirq ( )

This function write '1' to the RX_DR bit to clear it (reset RX_DR for next IRQ)

2.1.2.15 void NRF24L01P::resetTXirq ( )

This write '1' to the TX_DS bit to clear it (reset TX_DS for next IRQ)

2.1.2.16 void NRF24L01P::returnStandByI ( )

Explicit function that put the transceiver in StandByI mode by pull down the CE

2.1.2.17 void NRF24L01P::setAirDataRate (int *air_rate*)

Function that set the air data rate in which the transceiver operates writing in the RF_SETUP register

AirDataRate air_rate 250kbps 38 better receiver sensitivity 1Mbps 6 2Mbps 14 lower power consumption and low collision

Parameters

air_rate	is the value to push in the register according to the table

2.1.2.18 void NRF24L01P::setReceiveMode ( )

Function that put the transceiver in receive mode In order to call this function you have to call power_up, PWR_UP MUST be 1!

2.1.2.19 void NRF24L01P::setRfChannel (int *offset*)

Function that set the RF channel in which the transceiver operates Remember that the RF channel is set according to formula: F0 = 2400 + RF_CH [MHZ] the frequency are from 2,4 GHZ to 2,525GHZ, so the offset is between 0 and 125

Parameters

offset	is the RF_CH to apply at the given formula (value between 0 and 125 are legal)

2.1.2.20 void NRF24L01P::setStaticPayloadSize (int *size*)

If you are going to use static payload this function set the value of static payload is used (RX_PW_P0 register for the pipe0)

Parameters

size	is the value that you are going to use (value between 1 and 32 bytes are legal)

2.1.2.21 void NRF24L01P::setTransmitMode ( )

This function set the registers in the NRF24L01P in order to transmit data in TX-BUFFER. at the end of this function the transceiver starts to transmit data. WARNING: before call this function a packet in TX must be present.

2.1.2.22 void NRF24L01P::showInternal ( )

This function prints through serial the status of (almost) all registers

2.1.2.23 void NRF24L01P::TrasmitData (char * *data,* int *dim*)

Function that push data to TX-BUFFER (using the proper command _NRF24L01P_SPI_CMD_WR_TX_PAYLOAD) and then transmit by * calling setTransmitMode(). Finally it reset the tx irq bit and returns in receive mode.

Parameters

data	is a char pointer to the data to send. Data to send is an integer so the pointer scans byte per byte this integer
dim	is the dimension in bytes of the data to send (in case of our integer is 4)

2.1.2.24 void NRF24L01P::writeRegister (int *regAddress,* int *regData*)

Function that permit to write in NRF24L01P's registers through the SPI driver is the address of the register in which you want to write(in decimal notation) is the data (alias configuration) that you want to push in the register (always in decimal notation)

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

- NRF24L01P.h
- NRF24L01P.cpp

2.2 spi Class Reference

Public Member Functions

- **spi** (const spi &orig)
- void send (int data)
- int receive ()

2.2.1 Member Function Documentation

2.2.1.1 int spi::receive ( )

Shortcut to read data from SPI by a dummy write on SPI2->DR

Returns

the data from spi

2.2.1.2 void spi::send (int *data_byte*)

Send data over spi

When RXNE is set indicates that there are valid received data in the buffer RX. It is reset when SPI_DR register is read.

SPI_SR_RXNE = Receive buffer Not Empty (programmer manual pg 900) if RXNE=0 the buffer is empty.

A write on data register (DR) writes on the TX buffer, A read from data register returns the value in the RX buffer

Parameters

data_byte	data to be send through SPI

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

- spi.h
- spi.cpp

Index