Arduino RFID Driver

Generated by Doxygen 1.8.11

Contents

1 Hierarchical Index **2**

1.1 Class Hierarchy 2

2 Class Index **3**

2.1 Class List 3

3 File Index **5**

3.1 File List 5

4 Class Documentation **5**

4.1 RadioFrequencyIdentificationMFRC522::BIT_FRAMINGbits Union Reference 5

4.1.1 Detailed Description 5

4.1.2 Member Data Documentation 6

4.2 RadioFrequencyIdentificationMFRC522::COLLbits Union Reference 6

4.2.1 Detailed Description 6

4.2.2 Member Data Documentation 7

4.3 RadioFrequencyIdentificationMFRC522::COM_I_ENbits Union Reference 7

4.3.1 Detailed Description 8

4.3.2 Member Data Documentation 8

4.4 RadioFrequencyIdentificationMFRC522::COM_IRQbits Union Reference 9

4.4.1 Detailed Description 9

4.4.2 Member Data Documentation 9

4.5 RadioFrequencyIdentificationMFRC522::COMMANDbits Union Reference 10

4.5.1 Detailed Description 10

4.5.2 Member Data Documentation 10

4.6 RadioFrequencyIdentificationMFRC522::CONTROLbits Union Reference 11

4.6.1 Detailed Description 11

4.6.2 Member Data Documentation 11

4.7 RadioFrequencyIdentificationMFRC522::CW_GS_Pbits Union Reference 12

4.7.1 Detailed Description 12

4.7.2 Member Data Documentation 12
4.8 RadioFrequencyIdentificationMFRC522::DEMODbits Union Reference 13
4.8.1 Detailed Description 13
4.8.2 Member Data Documentation 13
4.9 RadioFrequencyIdentificationMFRC522::DIV_I_ENbits Union Reference 14
4.9.1 Detailed Description 14
4.9.2 Member Data Documentation 14
4.10 RadioFrequencyIdentificationMFRC522::DIV_IRQbits Union Reference 15
4.10.1 Detailed Description 15
4.10.2 Member Data Documentation 15
4.11 RadioFrequencyIdentificationMFRC522::ERRORbits Union Reference 16
4.11.1 Detailed Description 16
4.11.2 Member Data Documentation 17
4.12 RadioFrequencyIdentificationMFRC522::FIFO_LEVELbits Union Reference 17
4.12.1 Detailed Description 18
4.12.2 Member Data Documentation 18
4.13 RadioFrequencyIdentificationMFRC522::GS_Nbits Union Reference 18
4.13.1 Detailed Description 19
4.13.2 Member Data Documentation 19
4.14 RadioFrequencyIdentificationMFRC522::MF_RXbits Union Reference 19
4.14.1 Detailed Description 19
4.14.2 Member Data Documentation 20
4.15 RadioFrequencyIdentificationMFRC522::MF_TXbits Union Reference 20
4.15.1 Detailed Description 20
4.15.2 Member Data Documentation 20
4.16 RadioFrequencyIdentificationMFRC522::MOD_GS_Pbits Union Reference 21
4.16.1 Detailed Description 21
4.16.2 Member Data Documentation 21
4.17 RadioFrequencyIdentificationMFRC522::MODEbits Union Reference 22
4.17.1 Detailed Description 22

4.17.2 Member Data Documentation 22
4.18 RadioFrequencyIdentification Class Reference 23
4.18.1 Detailed Description 23
4.19 RadioFrequencyIdentificationMFRC522 Class Reference 24
4.19.1 Detailed Description 26
4.19.2 Member Enumeration Documentation 27
4.19.3 Constructor & Destructor Documentation 30
4.19.4 Member Function Documentation 30
4.19.5 Member Data Documentation 34
4.20 RadioFrequencyIdentificationMFRC522::RF_CFGbits Union Reference 34
4.20.1 Detailed Description 34
4.20.2 Member Data Documentation 34
4.21 RadioFrequencyIdentificationMFRC522::RX_MODEbits Union Reference 35
4.21.1 Detailed Description 35
4.21.2 Member Data Documentation 35
4.22 RadioFrequencyIdentificationMFRC522::RX_SELbits Union Reference 36
4.22.1 Detailed Description 36
4.22.2 Member Data Documentation 36
4.23 RadioFrequencyIdentificationMFRC522::RX_THRESHOLDbits Union Reference 37
4.23.1 Detailed Description 37
4.23.2 Member Data Documentation 37
4.24 RadioFrequencyIdentificationMFRC522::SERIAL_SPEEDbits Union Reference 38
4.24.1 Detailed Description 38
4.24.2 Member Data Documentation 38
4.25 RadioFrequencyIdentificationMFRC522::STATUS1bits Union Reference 39
4.25.1 Detailed Description 39
4.25.2 Member Data Documentation 39
4.26 RadioFrequencyIdentificationMFRC522::STATUS2bits Union Reference 40
4.26.1 Detailed Description 40
4.26.2 Member Data Documentation 41

4.27 RadioFrequencyIdentificationMFRC522::T_MODEbits Union Reference 41
4.27.1 Detailed Description 42
4.27.2 Member Data Documentation 42
4.28 RadioFrequencyIdentificationMFRC522::TX_ASKbits Union Reference 42
4.28.1 Detailed Description 43
4.28.2 Member Data Documentation 43
4.29 RadioFrequencyIdentificationMFRC522::TX_CONTROLbits Union Reference 43
4.29.1 Detailed Description 44
4.29.2 Member Data Documentation 44
4.30 RadioFrequencyIdentificationMFRC522::TX_MODEbits Union Reference 45
4.30.1 Detailed Description 45
4.30.2 Member Data Documentation 45
4.31 RadioFrequencyIdentificationMFRC522::TX_SELbits Union Reference 46
4.31.1 Detailed Description 46
4.31.2 Member Data Documentation 46
4.32 RadioFrequencyIdentificationMFRC522::Uid Struct Reference 47
4.32.1 Detailed Description 47
4.32.2 Member Data Documentation 47
4.33 RadioFrequencyIdentificationMFRC522::VERSIONbits Union Reference 48
4.33.1 Detailed Description 48
4.33.2 Member Data Documentation 48
4.34 RadioFrequencyIdentificationMFRC522::WATER_LEVELbits Union Reference 48
4.34.1 Detailed Description 49
4.34.2 Member Data Documentation 49

5 File Documentation **49**
5.1 RadioFrequencyIdentification.cpp File Reference 49
5.2 RadioFrequencyIdentification.cpp 50
5.3 RadioFrequencyIdentification.h File Reference 50
5.4 RadioFrequencyIdentification.h 50
5.5 RadioFrequencyIdentificationMFRC522.cpp File Reference 50
5.6 RadioFrequencyIdentificationMFRC522.cpp 51
5.7 RadioFrequencyIdentificationMFRC522.h File Reference 55
5.7.1 Macro Definition Documentation 56
5.7.2 Variable Documentation 57
5.8 RadioFrequencyIdentificationMFRC522.h 57

Index 73

1 Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

RadioFrequencyIdentificationMFRC522::BIT_FRAMINGbits 5
RadioFrequencyIdentificationMFRC522::COLLbits 6
RadioFrequencyIdentificationMFRC522::COM_I_ENbits 7
RadioFrequencyIdentificationMFRC522::COM_IRQbits 9
RadioFrequencyIdentificationMFRC522::COMMANDbits 10
RadioFrequencyIdentificationMFRC522::CONTROLbits 11
RadioFrequencyIdentificationMFRC522::CW_GS_Pbits 12
RadioFrequencyIdentificationMFRC522::DEMODbits 13
RadioFrequencyIdentificationMFRC522::DIV_I_ENbits 14
RadioFrequencyIdentificationMFRC522::DIV_IRQbits 15
RadioFrequencyIdentificationMFRC522::ERRORbits 16
RadioFrequencyIdentificationMFRC522::FIFO_LEVELbits 17
RadioFrequencyIdentificationMFRC522::GS_Nbits 18
RadioFrequencyIdentificationMFRC522::MF_RXbits 19
RadioFrequencyIdentificationMFRC522::MF_TXbits 20
RadioFrequencyIdentificationMFRC522::MOD_GS_Pbits 21
RadioFrequencyIdentificationMFRC522::MODEbits 22
RadioFrequencyIdentification 23
  RadioFrequencyIdentificationMFRC522 24
RegisterBasedDevice
  RadioFrequencyIdentificationMFRC522 24
RadioFrequencyIdentificationMFRC522::RF_CFGbits 34
RadioFrequencyIdentificationMFRC522::RX_MODEbits 35
RadioFrequencyIdentificationMFRC522::RX_SELbits 36
RadioFrequencyIdentificationMFRC522::RX_THRESHOLDbits 37
RadioFrequencyIdentificationMFRC522::SERIAL_SPEEDbits 38

RadioFrequencyIdentificationMFRC522::STATUS1bits 39

RadioFrequencyIdentificationMFRC522::STATUS2bits 40

RadioFrequencyIdentificationMFRC522::T_MODEbits 41

RadioFrequencyIdentificationMFRC522::TX_ASKbits 42

RadioFrequencyIdentificationMFRC522::TX_CONTROLbits 43

RadioFrequencyIdentificationMFRC522::TX_MODEbits 45

RadioFrequencyIdentificationMFRC522::TX_SELbits 46

RadioFrequencyIdentificationMFRC522::Uid 47

RadioFrequencyIdentificationMFRC522::VERSIONbits 48

RadioFrequencyIdentificationMFRC522::WATER_LEVELbits 48

2 Class Index

2.1 Class List

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

RadioFrequencyIdentificationMFRC522::BIT_FRAMINGbits
BIT_FRAMING register 5

RadioFrequencyIdentificationMFRC522::COLLbits
COLL register 6

RadioFrequencyIdentificationMFRC522::COM_I_ENbits
COM_I_EN register 7

RadioFrequencyIdentificationMFRC522::COM_IRQbits
COM_IRQ register 9

RadioFrequencyIdentificationMFRC522::COMMANDbits
COMMAND register (address 01h) Reset value: 20h bit allocation 10

RadioFrequencyIdentificationMFRC522::CONTROLbits
CONTROL register 11

RadioFrequencyIdentificationMFRC522::CW_GS_Pbits
CW_GS_P register 12

RadioFrequencyIdentificationMFRC522::DEMODbits
DEMOD register 13

RadioFrequencyIdentificationMFRC522::DIV_I_ENbits
DIV_I_EN register 14

RadioFrequencyIdentificationMFRC522::DIV_IRQbits
DIV_IRQ register 15

RadioFrequencyIdentificationMFRC522::ERRORbits
ERROR register 16

RadioFrequencyIdentificationMFRC522::FIFO_LEVELbits
FIFO_LEVEL register 17

RadioFrequencyIdentificationMFRC522::GS_Nbits
GS_N register 18

RadioFrequencyIdentificationMFRC522::MF_RXbits
MF_RX register 19

RadioFrequencyIdentificationMFRC522::MF_TXbits
MF_TX register 20

RadioFrequencyIdentificationMFRC522::MOD_GS_Pbits
MOD_GS_P register 21

RadioFrequencyIdentificationMFRC522::MODEbits
MODE register 22

RadioFrequencyIdentification
Arduino - Radio Frequency Identification 23

RadioFrequencyIdentificationMFRC522 24

RadioFrequencyIdentificationMFRC522::RF_CFGbits
RF_CFG register 34

RadioFrequencyIdentificationMFRC522::RX_MODEbits
RX_MODE register 35

RadioFrequencyIdentificationMFRC522::RX_SELbits
RX_SEL register 36

RadioFrequencyIdentificationMFRC522::RX_THRESHOLDbits
RX_THRESHOLD register 37

RadioFrequencyIdentificationMFRC522::SERIAL_SPEEDbits
SERIAL_SPEED register 38

RadioFrequencyIdentificationMFRC522::STATUS1bits
STATUS1 register 39

RadioFrequencyIdentificationMFRC522::STATUS2bits
STATUS2 register 40

RadioFrequencyIdentificationMFRC522::T_MODEbits
T_MODE register 41

RadioFrequencyIdentificationMFRC522::TX_ASKbits
TX_ASK register 42

RadioFrequencyIdentificationMFRC522::TX_CONTROLbits
TX_CONTROL register 43

RadioFrequencyIdentificationMFRC522::TX_MODEbits
TX_MODE register 45

RadioFrequencyIdentificationMFRC522::TX_SELbits
TX_SEL register 46

RadioFrequencyIdentificationMFRC522::Uid 47

RadioFrequencyIdentificationMFRC522::VERSIONbits
VERSION register 48

RadioFrequencyIdentificationMFRC522::WATER_LEVELbits
WATER_LEVEL register 48

3 File Index

3.1 File List

Here is a list of all files with brief descriptions:

RadioFrequencyIdentification.cpp 49

RadioFrequencyIdentification.h 50

RadioFrequencyIdentificationMFRC522.cpp 50

RadioFrequencyIdentificationMFRC522.h 55

4 Class Documentation

4.1 RadioFrequencyIdentificationMFRC522::BIT_FRAMINGbits Union Reference

```
#include <RadioFrequencyIdentificationMFRC522.h>
```

Public Attributes

- struct {
    unsigned char TX_LAST_BITS:3
    unsigned char:1
    unsigned char RX_ALIGN:3
    unsigned char START_SEND:1
  };
- unsigned char value

4.1.1 Detailed Description

BIT_FRAMING register.

Miscellaneous control bits.

Definition at line 685 of file RadioFrequencyIdentificationMFRC522.h.

4.1.2 Member Data Documentation

4.1.2.1 struct { ... }

4.1.2.2 unsigned RadioFrequencyIdentificationMFRC522::BIT_FRAMINGbits::char

Definition at line 694 of file RadioFrequencyIdentificationMFRC522.h.

4.1.2.3 unsigned char RadioFrequencyIdentificationMFRC522::BIT_FRAMINGbits::RX_ALIGN

Definition at line 702 of file RadioFrequencyIdentificationMFRC522.h.

4.1.2.4 unsigned char RadioFrequencyIdentificationMFRC522::BIT_FRAMINGbits::START_SEND

Definition at line 705 of file RadioFrequencyIdentificationMFRC522.h.

4.1.2.5 unsigned char RadioFrequencyIdentificationMFRC522::BIT_FRAMINGbits::TX_LAST_BITS

Definition at line 691 of file RadioFrequencyIdentificationMFRC522.h.

4.1.2.6 unsigned char RadioFrequencyIdentificationMFRC522::BIT_FRAMINGbits::value

Definition at line 707 of file RadioFrequencyIdentificationMFRC522.h.

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

- RadioFrequencyIdentificationMFRC522.h

4.2 RadioFrequencyIdentificationMFRC522::COLLbits Union Reference

```
#include <RadioFrequencyIdentificationMFRC522.h>
```

Public Attributes

- struct {
    unsigned char COLL_POS:5
    unsigned char COLL_POS_NOT_VALID:1
    unsigned char:1
    unsigned char VALUES_AFTER_COLL:1
  };

- unsigned char value

4.2.1 Detailed Description

COLL register.

Miscellaneous control bits.

Definition at line 715 of file RadioFrequencyIdentificationMFRC522.h.

4.2.2 Member Data Documentation

4.2.2.1 struct { ... }

4.2.2.2 unsigned RadioFrequencyIdentificationMFRC522::COLLbits::char

Definition at line 731 of file RadioFrequencyIdentificationMFRC522.h.

4.2.2.3 unsigned char RadioFrequencyIdentificationMFRC522::COLLbits::COLL_POS

Definition at line 725 of file RadioFrequencyIdentificationMFRC522.h.

4.2.2.4 unsigned char RadioFrequencyIdentificationMFRC522::COLLbits::COLL_POS_NOT_VALID

Definition at line 728 of file RadioFrequencyIdentificationMFRC522.h.

4.2.2.5 unsigned char RadioFrequencyIdentificationMFRC522::COLLbits::value

Definition at line 736 of file RadioFrequencyIdentificationMFRC522.h.

4.2.2.6 unsigned char RadioFrequencyIdentificationMFRC522::COLLbits::VALUES_AFTER_COLL

Definition at line 734 of file RadioFrequencyIdentificationMFRC522.h.

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

- RadioFrequencyIdentificationMFRC522.h

4.3 RadioFrequencyIdentificationMFRC522::COM_I_ENbits Union Reference

```
#include <RadioFrequencyIdentificationMFRC522.h>
```

Public Attributes

- struct {
  unsigned char TIMER_I_EN:1
  unsigned char ERR_I_EN:1
  unsigned char LO_ALERT_I_EN:1
  unsigned char HI_ALERT_I_EN:1
  unsigned char IDLE_I_EN:1
  unsigned char RX_I_EN:1
  unsigned char TX_I_EN:1
  unsigned char I_RQ_INV:1
  };
- unsigned char value

4.3.1 Detailed Description

COM_I_EN register.

Control bits to enable and disable the passing of interrupt requests.

Definition at line 339 of file RadioFrequencyIdentificationMFRC522.h.

4.3.2 Member Data Documentation

4.3.2.1 struct { ... }

4.3.2.2 unsigned char RadioFrequencyIdentificationMFRC522::COM_I_ENbits::ERR_I_EN

Definition at line 347 of file RadioFrequencyIdentificationMFRC522.h.

4.3.2.3 unsigned char RadioFrequencyIdentificationMFRC522::COM_I_ENbits::HI_ALERT_I_EN

Definition at line 353 of file RadioFrequencyIdentificationMFRC522.h.

4.3.2.4 unsigned char RadioFrequencyIdentificationMFRC522::COM_I_ENbits::I_RQ_INV

Definition at line 367 of file RadioFrequencyIdentificationMFRC522.h.

4.3.2.5 unsigned char RadioFrequencyIdentificationMFRC522::COM_I_ENbits::IDLE_I_EN

Definition at line 356 of file RadioFrequencyIdentificationMFRC522.h.

4.3.2.6 unsigned char RadioFrequencyIdentificationMFRC522::COM_I_ENbits::LO_ALERT_I_EN

Definition at line 350 of file RadioFrequencyIdentificationMFRC522.h.

4.3.2.7 unsigned char RadioFrequencyIdentificationMFRC522::COM_I_ENbits::RX_I_EN

Definition at line 359 of file RadioFrequencyIdentificationMFRC522.h.

4.3.2.8 unsigned char RadioFrequencyIdentificationMFRC522::COM_I_ENbits::TIMER_I_EN

Definition at line 344 of file RadioFrequencyIdentificationMFRC522.h.

4.3.2.9 unsigned char RadioFrequencyIdentificationMFRC522::COM_I_ENbits::TX_I_EN

Definition at line 362 of file RadioFrequencyIdentificationMFRC522.h.

4.3.2.10 unsigned char RadioFrequencyIdentificationMFRC522::COM_I_ENbits::value

Definition at line 369 of file RadioFrequencyIdentificationMFRC522.h.

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

- RadioFrequencyIdentificationMFRC522.h

4.4 RadioFrequencyIdentificationMFRC522::COM_IRQbits Union Reference

```
#include <RadioFrequencyIdentificationMFRC522.h>
```

Public Attributes

- struct {
  unsigned char TIMER_IRQ:1
  unsigned char ERR_IRQ:1
  unsigned char LO_ALERT_IRQ:1
  unsigned char HI_ALERT_IRQ:1
  unsigned char IDLE_IRQ:1
  unsigned char RX_IRQ:1
  unsigned char TX_IRQ:1
  unsigned char SET1:1
  };
- unsigned char value

4.4.1 Detailed Description

COM_IRQ register.

Interrupt request bits.

Definition at line 408 of file RadioFrequencyIdentificationMFRC522.h.

4.4.2 Member Data Documentation

4.4.2.1 struct { ... }

4.4.2.2 unsigned char RadioFrequencyIdentificationMFRC522::COM_IRQbits::ERR_IRQ

Definition at line 416 of file RadioFrequencyIdentificationMFRC522.h.

4.4.2.3 unsigned char RadioFrequencyIdentificationMFRC522::COM_IRQbits::HI_ALERT_IRQ

Definition at line 426 of file RadioFrequencyIdentificationMFRC522.h.

4.4.2.4 unsigned char RadioFrequencyIdentificationMFRC522::COM_IRQbits::IDLE_IRQ

Definition at line 432 of file RadioFrequencyIdentificationMFRC522.h.

4.4.2.5 unsigned char RadioFrequencyIdentificationMFRC522::COM_IRQbits::LO_ALERT_IRQ

Definition at line 421 of file RadioFrequencyIdentificationMFRC522.h.

4.4.2.6 unsigned char RadioFrequencyIdentificationMFRC522::COM_IRQbits::RX_IRQ

Definition at line 437 of file RadioFrequencyIdentificationMFRC522.h.

4.4.2.7 unsigned char RadioFrequencyIdentificationMFRC522::COM_IRQbits::SET1

Definition at line 444 of file RadioFrequencyIdentificationMFRC522.h.

4.4.2.8 unsigned char RadioFrequencyIdentificationMFRC522::COM_IRQbits::TIMER_IRQ

Definition at line 413 of file RadioFrequencyIdentificationMFRC522.h.

4.4.2.9 unsigned char RadioFrequencyIdentificationMFRC522::COM_IRQbits::TX_IRQ

Definition at line 440 of file RadioFrequencyIdentificationMFRC522.h.

4.4.2.10 unsigned char RadioFrequencyIdentificationMFRC522::COM_IRQbits::value

Definition at line 446 of file RadioFrequencyIdentificationMFRC522.h.

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

- RadioFrequencyIdentificationMFRC522.h

4.5 RadioFrequencyIdentificationMFRC522::COMMANDbits Union Reference

```
#include <RadioFrequencyIdentificationMFRC522.h>
```

Public Attributes

- struct {
  unsigned char COMMAND:4
  unsigned char POWER_DOWN:1
  unsigned char RCV_OFF:1
  unsigned char:2
  };
- unsigned char value

4.5.1 Detailed Description

COMMAND register (address 01h) Reset value: 20h bit allocation.

Definition at line 312 of file RadioFrequencyIdentificationMFRC522.h.

4.5.2 Member Data Documentation

4.5.2.1 struct { ... }

4.5.2.2 unsigned RadioFrequencyIdentificationMFRC522::COMMANDbits::char

Definition at line 329 of file RadioFrequencyIdentificationMFRC522.h.

4.5.2.3 unsigned char RadioFrequencyIdentificationMFRC522::COMMANDbits::COMMAND

Definition at line 318 of file RadioFrequencyIdentificationMFRC522.h.

4.5.2.4 unsigned char RadioFrequencyIdentificationMFRC522::COMMANDbits::POWER_DOWN

Definition at line 323 of file RadioFrequencyIdentificationMFRC522.h.

4.5.2.5 unsigned char RadioFrequencyIdentificationMFRC522::COMMANDbits::RCV_OFF

Definition at line 326 of file RadioFrequencyIdentificationMFRC522.h.

4.5.2.6 unsigned char RadioFrequencyIdentificationMFRC522::COMMANDbits::value

Definition at line 331 of file RadioFrequencyIdentificationMFRC522.h.

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

- RadioFrequencyIdentificationMFRC522.h

4.6 RadioFrequencyIdentificationMFRC522::CONTROLbits Union Reference

```
#include <RadioFrequencyIdentificationMFRC522.h>
```

Public Attributes

- struct {
  unsigned char RX_LAST_BITS:3
  unsigned char:2
  unsigned char T_START_NOW:1
  unsigned char T_STOP_NOW:1
  };
- unsigned char value

4.6.1 Detailed Description

CONTROL register.

Miscellaneous control bits.

Definition at line 659 of file RadioFrequencyIdentificationMFRC522.h.

4.6.2 Member Data Documentation

4.6.2.1 struct { ... }

4.6.2.2 unsigned RadioFrequencyIdentificationMFRC522::CONTROLbits::char

Definition at line 667 of file RadioFrequencyIdentificationMFRC522.h.

4.6.2.3 unsigned char RadioFrequencyIdentificationMFRC522::CONTROLbits::RX_LAST_BITS

Definition at line 664 of file RadioFrequencyIdentificationMFRC522.h.

4.6.2.4 unsigned char RadioFrequencyIdentificationMFRC522::CONTROLbits::T_START_NOW

Definition at line 671 of file RadioFrequencyIdentificationMFRC522.h.

4.6.2.5 unsigned char RadioFrequencyIdentificationMFRC522::CONTROLbits::T_STOP_NOW

Definition at line 675 of file RadioFrequencyIdentificationMFRC522.h.

4.6.2.6 unsigned char RadioFrequencyIdentificationMFRC522::CONTROLbits::value

Definition at line 677 of file RadioFrequencyIdentificationMFRC522.h.

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

- RadioFrequencyIdentificationMFRC522.h

4.7 RadioFrequencyIdentificationMFRC522::CW_GS_Pbits Union Reference

```
#include <RadioFrequencyIdentificationMFRC522.h>
```

Public Attributes

- struct {
      unsigned char CW_GS_P:6
      unsigned char:2
  };
- unsigned char value

4.7.1 Detailed Description

CW_GS_P register.

Defines the conductance of the p-driver output during periods of no modulation.

Definition at line 1157 of file RadioFrequencyIdentificationMFRC522.h.

4.7.2 Member Data Documentation

4.7.2.1 struct { ... }

4.7.2.2 unsigned RadioFrequencyIdentificationMFRC522::CW_GS_Pbits::char

Definition at line 1166 of file RadioFrequencyIdentificationMFRC522.h.

4.7.2.3 unsigned char RadioFrequencyIdentificationMFRC522::CW_GS_Pbits::CW_GS_P

Definition at line 1163 of file RadioFrequencyIdentificationMFRC522.h.

4.7.2.4 unsigned char RadioFrequencyIdentificationMFRC522::CW_GS_Pbits::value

Definition at line 1168 of file RadioFrequencyIdentificationMFRC522.h.

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

- RadioFrequencyIdentificationMFRC522.h

4.8 RadioFrequencyIdentificationMFRC522::DEMODbits Union Reference

```
#include <RadioFrequencyIdentificationMFRC522.h>
```

Public Attributes

- struct {
  unsigned char TAU_SYNC:2
  unsigned char TAU_RCV:2
  unsigned char T_PRESCAL_EVEN:1
  unsigned char FIX_IQ:1
  unsigned char ADD_IQ:2
  };
- unsigned char value

4.8.1 Detailed Description

DEMOD register.

Defines demodulator settings.

Definition at line 1012 of file RadioFrequencyIdentificationMFRC522.h.

4.8.2 Member Data Documentation

4.8.2.1 struct { ... }

4.8.2.2 unsigned char RadioFrequencyIdentificationMFRC522::DEMODbits::ADD_IQ

Definition at line 1037 of file RadioFrequencyIdentificationMFRC522.h.

4.8.2.3 unsigned char RadioFrequencyIdentificationMFRC522::DEMODbits::FIX_IQ

Definition at line 1031 of file RadioFrequencyIdentificationMFRC522.h.

4.8.2.4 unsigned char RadioFrequencyIdentificationMFRC522::DEMODbits::T_PRESCAL_EVEN

Definition at line 1027 of file RadioFrequencyIdentificationMFRC522.h.

4.8.2.5 unsigned char RadioFrequencyIdentificationMFRC522::DEMODbits::TAU_RCV

Definition at line 1021 of file RadioFrequencyIdentificationMFRC522.h.

4.8.2.6 unsigned char RadioFrequencyIdentificationMFRC522::DEMODbits::TAU_SYNC

Definition at line 1017 of file RadioFrequencyIdentificationMFRC522.h.

4.8.2.7 unsigned char RadioFrequencyIdentificationMFRC522::DEMODbits::value

Definition at line 1039 of file RadioFrequencyIdentificationMFRC522.h.

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

- RadioFrequencyIdentificationMFRC522.h

4.9 RadioFrequencyIdentificationMFRC522::DIV_I_ENbits Union Reference

```
#include <RadioFrequencyIdentificationMFRC522.h>
```

Public Attributes

- struct {
    unsigned char:2
    unsigned char CRC_I_EN:1
    unsigned char MFIN_ACT_I_EN:1
    unsigned char IRQ_PUSH_PULL:1
  };
- unsigned char value

4.9.1 Detailed Description

DIV_I_EN register.

Control bits to enable and disable the passing of interrupt requests.

Definition at line 377 of file RadioFrequencyIdentificationMFRC522.h.

4.9.2 Member Data Documentation

4.9.2.1 struct { ... }

4.9.2.2 unsigned RadioFrequencyIdentificationMFRC522::DIV_I_ENbits::char

Definition at line 382 of file RadioFrequencyIdentificationMFRC522.h.

4.9.2.3 unsigned char RadioFrequencyIdentificationMFRC522::DIV_I_ENbits::CRC_I_EN

Definition at line 385 of file RadioFrequencyIdentificationMFRC522.h.

4.9.2.4 unsigned char RadioFrequencyIdentificationMFRC522::DIV_I_ENbits::IRQ_PUSH_PULL

Definition at line 398 of file RadioFrequencyIdentificationMFRC522.h.

4.9.2.5 unsigned char RadioFrequencyIdentificationMFRC522::DIV_I_ENbits::MFIN_ACT_I_EN

Definition at line 391 of file RadioFrequencyIdentificationMFRC522.h.

4.9.2.6 unsigned char RadioFrequencyIdentificationMFRC522::DIV_I_ENbits::value

Definition at line 400 of file RadioFrequencyIdentificationMFRC522.h.

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

- RadioFrequencyIdentificationMFRC522.h

4.10 RadioFrequencyIdentificationMFRC522::DIV_IRQbits Union Reference

```
#include <RadioFrequencyIdentificationMFRC522.h>
```

Public Attributes

- struct {
  unsigned char:2
  unsigned char CRC_IRQ:1
  unsigned char MFIN_ACT_IRQ:1
  unsigned char SET2:1
  };
- unsigned char value

4.10.1 Detailed Description

DIV_IRQ register.

Interrupt request bits.

Definition at line 454 of file RadioFrequencyIdentificationMFRC522.h.

4.10.2 Member Data Documentation

4.10.2.1 struct { ... }

4.10.2.2 unsigned RadioFrequencyIdentificationMFRC522::DIV_IRQbits::char

Definition at line 459 of file RadioFrequencyIdentificationMFRC522.h.

4.10.2.3 unsigned char RadioFrequencyIdentificationMFRC522::DIV_IRQbits::CRC_IRQ

Definition at line 462 of file RadioFrequencyIdentificationMFRC522.h.

4.10.2.4 unsigned char RadioFrequencyIdentificationMFRC522::DIV_IRQbits::MFIN_ACT_IRQ

Definition at line 468 of file RadioFrequencyIdentificationMFRC522.h.

4.10.2.5 unsigned char RadioFrequencyIdentificationMFRC522::DIV_IRQbits::SET2

Definition at line 475 of file RadioFrequencyIdentificationMFRC522.h.

4.10.2.6 unsigned char RadioFrequencyIdentificationMFRC522::DIV_IRQbits::value

Definition at line 477 of file RadioFrequencyIdentificationMFRC522.h.

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

- RadioFrequencyIdentificationMFRC522.h

4.11 RadioFrequencyIdentificationMFRC522::ERRORbits Union Reference

```
#include <RadioFrequencyIdentificationMFRC522.h>
```

Public Attributes

- struct {
    unsigned char PROTOCOL_ERR:1
    unsigned char PARITY_ERR:1
    unsigned char CRC_ERR:1
    unsigned char COLL_ERR:1
    unsigned char BUFFER_OVFL:1
    unsigned char:1
    unsigned char TEMP_ERR:1
    unsigned char WR_ERR:1
  };

- unsigned char value

4.11.1 Detailed Description

ERROR register.

Error bit register showing the error status of the last command executed.

Definition at line 485 of file RadioFrequencyIdentificationMFRC522.h.

4.11.2 Member Data Documentation

4.11.2.1 struct { ... }

4.11.2.2 unsigned char RadioFrequencyIdentificationMFRC522::ERRORbits::BUFFER_OVFL

Definition at line 509 of file RadioFrequencyIdentificationMFRC522.h.

4.11.2.3 unsigned RadioFrequencyIdentificationMFRC522::ERRORbits::char

Definition at line 512 of file RadioFrequencyIdentificationMFRC522.h.

4.11.2.4 unsigned char RadioFrequencyIdentificationMFRC522::ERRORbits::COLL_ERR

Definition at line 505 of file RadioFrequencyIdentificationMFRC522.h.

4.11.2.5 unsigned char RadioFrequencyIdentificationMFRC522::ERRORbits::CRC_ERR

Definition at line 500 of file RadioFrequencyIdentificationMFRC522.h.

4.11.2.6 unsigned char RadioFrequencyIdentificationMFRC522::ERRORbits::PARITY_ERR

Definition at line 496 of file RadioFrequencyIdentificationMFRC522.h.

4.11.2.7 unsigned char RadioFrequencyIdentificationMFRC522::ERRORbits::PROTOCOL_ERR

Definition at line 492 of file RadioFrequencyIdentificationMFRC522.h.

4.11.2.8 unsigned char RadioFrequencyIdentificationMFRC522::ERRORbits::TEMP_ERR

Definition at line 515 of file RadioFrequencyIdentificationMFRC522.h.

4.11.2.9 unsigned char RadioFrequencyIdentificationMFRC522::ERRORbits::value

Definition at line 522 of file RadioFrequencyIdentificationMFRC522.h.

4.11.2.10 unsigned char RadioFrequencyIdentificationMFRC522::ERRORbits::WR_ERR

Definition at line 520 of file RadioFrequencyIdentificationMFRC522.h.

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

- RadioFrequencyIdentificationMFRC522.h

4.12 RadioFrequencyIdentificationMFRC522::FIFO_LEVELbits Union Reference

```
#include <RadioFrequencyIdentificationMFRC522.h>
```

Public Attributes

- struct {
  unsigned char FIFO_LEVEL:7
  unsigned char FLUSH_BUFFER:1
  };
- unsigned char value

4.12.1 Detailed Description

FIFO_LEVEL register.

Indicates the number of bytes stored in the FIFO.

Definition at line 617 of file RadioFrequencyIdentificationMFRC522.h.

4.12.2 Member Data Documentation

4.12.2.1 struct { ... }

4.12.2.2 unsigned char RadioFrequencyIdentificationMFRC522::FIFO_LEVELbits::FIFO_LEVEL

Definition at line 623 of file RadioFrequencyIdentificationMFRC522.h.

4.12.2.3 unsigned char RadioFrequencyIdentificationMFRC522::FIFO_LEVELbits::FLUSH_BUFFER

Definition at line 627 of file RadioFrequencyIdentificationMFRC522.h.

4.12.2.4 unsigned char RadioFrequencyIdentificationMFRC522::FIFO_LEVELbits::value

Definition at line 629 of file RadioFrequencyIdentificationMFRC522.h.

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

- RadioFrequencyIdentificationMFRC522.h

4.13 RadioFrequencyIdentificationMFRC522::GS_Nbits Union Reference

```
#include <RadioFrequencyIdentificationMFRC522.h>
```

Public Attributes

- struct {
  unsigned char MOD_GS_N:4
  unsigned char CW_GS_N:4
  };
- unsigned char value

4.13.1 Detailed Description

GS_N register.

Defines the conductance of the antenna driver pins TX1 and TX2 for the n-driver when the driver is switched on.

Definition at line 1134 of file RadioFrequencyIdentificationMFRC522.h.

4.13.2 Member Data Documentation

4.13.2.1 struct { ... }

4.13.2.2 unsigned char RadioFrequencyIdentificationMFRC522::GS_Nbits::CW_GS_N

Definition at line 1147 of file RadioFrequencyIdentificationMFRC522.h.

4.13.2.3 unsigned char RadioFrequencyIdentificationMFRC522::GS_Nbits::MOD_GS_N

Definition at line 1141 of file RadioFrequencyIdentificationMFRC522.h.

4.13.2.4 unsigned char RadioFrequencyIdentificationMFRC522::GS_Nbits::value

Definition at line 1149 of file RadioFrequencyIdentificationMFRC522.h.

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

- RadioFrequencyIdentificationMFRC522.h

4.14 RadioFrequencyIdentificationMFRC522::MF_RXbits Union Reference

```
#include <RadioFrequencyIdentificationMFRC522.h>
```

Public Attributes

- struct {
    unsigned char:4
    unsigned char PARITY_DISABLE:1
  };
- unsigned char value

4.14.1 Detailed Description

MF_RX register.

Controls some MIFARE communication receive parameters.

Definition at line 1065 of file RadioFrequencyIdentificationMFRC522.h.

4.14.2 Member Data Documentation

4.14.2.1 struct { ... }

4.14.2.2 unsigned RadioFrequencyIdentificationMFRC522::MF_RXbits::char

Definition at line 1070 of file RadioFrequencyIdentificationMFRC522.h.

4.14.2.3 unsigned char RadioFrequencyIdentificationMFRC522::MF_RXbits::PARITY_DISABLE

Definition at line 1074 of file RadioFrequencyIdentificationMFRC522.h.

4.14.2.4 unsigned char RadioFrequencyIdentificationMFRC522::MF_RXbits::value

Definition at line 1079 of file RadioFrequencyIdentificationMFRC522.h.

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

- RadioFrequencyIdentificationMFRC522.h

4.15 RadioFrequencyIdentificationMFRC522::MF_TXbits Union Reference

```
#include <RadioFrequencyIdentificationMFRC522.h>
```

Public Attributes

- struct {
  unsigned char TX_WAIT:2
  unsigned char:6
  };
- unsigned char value

4.15.1 Detailed Description

MF_TX register.

Controls some MIFARE communication transmit parameters.

Definition at line 1047 of file RadioFrequencyIdentificationMFRC522.h.

4.15.2 Member Data Documentation

4.15.2.1 struct { ... }

4.15.2.2 unsigned RadioFrequencyIdentificationMFRC522::MF_TXbits::char

Definition at line 1055 of file RadioFrequencyIdentificationMFRC522.h.

4.15.2.3 unsigned char RadioFrequencyIdentificationMFRC522::MF_TXbits::TX_WAIT

Definition at line 1052 of file RadioFrequencyIdentificationMFRC522.h.

4.15.2.4 unsigned char RadioFrequencyIdentificationMFRC522::MF_TXbits::value

Definition at line 1057 of file RadioFrequencyIdentificationMFRC522.h.

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

- RadioFrequencyIdentificationMFRC522.h

4.16 RadioFrequencyIdentificationMFRC522::MOD_GS_Pbits Union Reference

```
#include <RadioFrequencyIdentificationMFRC522.h>
```

Public Attributes

- struct {
  unsigned char MOD_GS_P:6
  unsigned char:2
  };
- unsigned char value

4.16.1 Detailed Description

MOD_GS_P register.

Defines the conductance of the p-driver output during modulation.

Definition at line 1176 of file RadioFrequencyIdentificationMFRC522.h.

4.16.2 Member Data Documentation

4.16.2.1 struct { ... }

4.16.2.2 unsigned RadioFrequencyIdentificationMFRC522::MOD_GS_Pbits::char

Definition at line 1186 of file RadioFrequencyIdentificationMFRC522.h.

4.16.2.3 unsigned char RadioFrequencyIdentificationMFRC522::MOD_GS_Pbits::MOD_GS_P

Definition at line 1183 of file RadioFrequencyIdentificationMFRC522.h.

4.16.2.4 unsigned char RadioFrequencyIdentificationMFRC522::MOD_GS_Pbits::value

Definition at line 1188 of file RadioFrequencyIdentificationMFRC522.h.

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

- RadioFrequencyIdentificationMFRC522.h

4.17 RadioFrequencyIdentificationMFRC522::MODEbits Union Reference

```
#include <RadioFrequencyIdentificationMFRC522.h>
```

Public Attributes

- struct {
  unsigned char CRC_PRESET:2
  unsigned char:1
  unsigned char POL_M_FIN:1
  unsigned char TX_WAIT_RF:1
  unsigned char MSB_FIRST:1
  };
- unsigned char value

4.17.1 Detailed Description

MODE register.

Defines general mode settings for transmitting and receiving.

Definition at line 744 of file RadioFrequencyIdentificationMFRC522.h.

4.17.2 Member Data Documentation

4.17.2.1 struct { ... }

4.17.2.2 unsigned RadioFrequencyIdentificationMFRC522::MODEbits::char

Definition at line 758 of file RadioFrequencyIdentificationMFRC522.h.

4.17.2.3 unsigned char RadioFrequencyIdentificationMFRC522::MODEbits::CRC_PRESET

Definition at line 755 of file RadioFrequencyIdentificationMFRC522.h.

4.17.2.4 unsigned char RadioFrequencyIdentificationMFRC522::MODEbits::MSB_FIRST

Definition at line 778 of file RadioFrequencyIdentificationMFRC522.h.

4.17.2.5 unsigned char RadioFrequencyIdentificationMFRC522::MODEbits::POL_M_FIN

Definition at line 764 of file RadioFrequencyIdentificationMFRC522.h.

4.17.2.6 unsigned char RadioFrequencyIdentificationMFRC522::MODEbits::TX_WAIT_RF

Definition at line 770 of file RadioFrequencyIdentificationMFRC522.h.

4.17.2.7 unsigned char RadioFrequencyIdentificationMFRC522::MODEbits::value

Definition at line 780 of file RadioFrequencyIdentificationMFRC522.h.

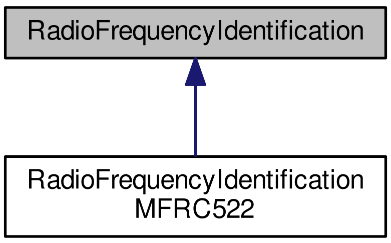
The documentation for this union was generated from the following file:

- RadioFrequencyIdentificationMFRC522.h

4.18 RadioFrequencyIdentification Class Reference

```
#include <RadioFrequencyIdentification.h>
```

Inheritance diagram for RadioFrequencyIdentification:


RadioFrequencyIdentification
RadioFrequencyIdentification MFRC522

4.18.1 Detailed Description

Arduino - Radio Frequency Identification.

Author

Dalmir da Silva `dalmirdasilva@gmail.com`

Definition at line 10 of file RadioFrequencyIdentification.h.

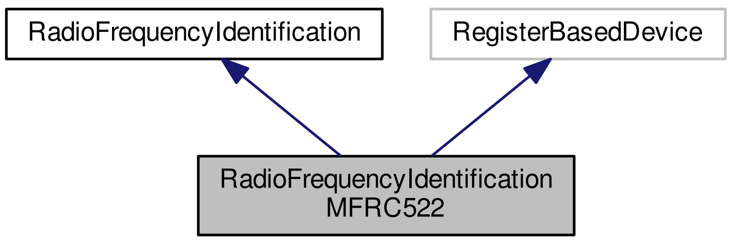
The documentation for this class was generated from the following file:

- RadioFrequencyIdentification.h

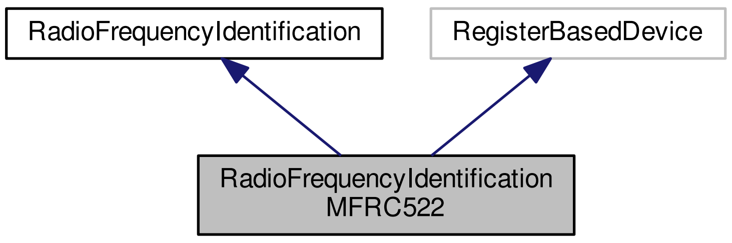
4.19 RadioFrequencyIdentificationMFRC522 Class Reference

`#include <RadioFrequencyIdentificationMFRC522.h>`

Inheritance diagram for RadioFrequencyIdentificationMFRC522:


RadioFrequencyIdentification
RegisterBasedDevice
RadioFrequencyIdentification MFRC522

Collaboration diagram for RadioFrequencyIdentificationMFRC522:


RadioFrequencyIdentification
RegisterBasedDevice
RadioFrequencyIdentification MFRC522

Classes

- union BIT_FRAMINGbits
- union COLLbits
- union COM_I_ENbits
- union COM_IRQbits
- union COMMANDbits
- union CONTROLbits
- union CW_GS_Pbits
- union DEMODbits
- union DIV_I_ENbits
- union DIV_IRQbits
- union ERRORbits
- union FIFO_LEVELbits
- union GS_Nbits

- union MF_RXbits
- union MF_TXbits
- union MOD_GS_Pbits
- union MODEbits
- union RF_CFGbits
- union RX_MODEbits
- union RX_SELbits
- union RX_THRESHOLDbits
- union SERIAL_SPEEDbits
- union STATUS1bits
- union STATUS2bits
- union T_MODEbits
- union TX_ASKbits
- union TX_CONTROLbits
- union TX_MODEbits
- union TX_SELbits
- struct Uid
- union VERSIONbits
- union WATER_LEVELbits

Public Types

- enum Register {
  COMMAND = 0x01, COM_I_EN = 0x02, DIV_I_EN = 0x03, COM_IRQ = 0x04,
  DIV_IRQ = 0X05, ERROR = 0X06, STATUS1 = 0x07, STATUS2 = 0x08,
  FIFO_DATA = 0X09, FIFO_LEVEL = 0x0a, WATER_LEVEL = 0x0b, CONTROL = 0x0c,
  BIT_FRAMING = 0x0d, COLL = 0x0e, MODE = 0x11, TX_MODE = 0x12,
  RX_MODE = 0x13, TX_CONTROL = 0x14, TX_ASK = 0x15, TX_SEL = 0x16,
  RX_SEL = 0x17, RX_THRESHOLD = 0x18, DEMOD = 0x19, MF_TX = 0x1c,
  MF_RX = 0x1d, SERIAL_SPEED = 0x1f, CRC_RESULT_HIDH = 0x21, CRC_RESULT_LOW = 0x22,
  MOD_WIDTH = 0x24, RFC_FG = 0x26, GS_N = 0x27, CW_GS_P = 0x28,
  MOD_GS_P = 0x29, T_MODE = 0x2a, T_PRESCALER_LOW = 0x2b, T_RELOAD_HIGH = 0x2c,
  T_RELOAD_LOW = 0x2d, T_COUNTER_VAL_HIGH = 0x2e, T_COUNTER_VAL_LOW = 0x2f, TEST_SEL1 = 0x31,
  TEST_SEL2 = 0x32, TEST_PIN_EN = 0x33, TEST_PIN_VALUE = 0x34, TEST_BUS = 0x35,
  AUTO_TEST = 0x36, VERSION = 0x37, ANALOG_TEST = 0x38, TEST_DAC1 = 0x39,
  TEST_DAC2 = 0x3a, TEST_ADC = 0x3b }
- enum Command {
  IDLE = 0x00, MEM = 0x01, GENERATE_RANDOM_ID = 0x02, CALC_CRC = 0x03,
  TRANSMIT = 0x04, NO_CMD_CHANGE = 0x07, RECEIVE = 0x08, TRANSCEIVE = 0x0c,
  MF_AUTHENT = 0x0e, SOFT_RESET = 0x0F }
- enum Error {
  NO_ERROR = 0x00, GENERAL_ERROR = 0x01, TIMEOUT_ERROR = 0x02, COMMUNICATION_ERROR = 0x03,
  CRC_ERROR = 0x04 }
- enum Mask {
  TX_CONTROL_TX1_RF_EN = 0x01, TX_CONTROL_TX2_RF_EN = 0x02, TX_CONTROL_TX_RF_EN = TX_CONTROL_TX1_RF_EN | TX_CONTROL_TX2_RF_EN, CONTROL_T_STOP_NOW = 0x80,
  CONTROL_T_START_NOW = 0x40, COM_I_EN_INTERRUPT_EN = 0x7f, COM_IRQ_TIMER_IRQ = 0x01, COM_IRQ_ERR_IRQ = 0x02,
  COM_IRQ_LO_ALERT_IRQ = 0x04, COM_IRQ_HI_ALERT_IRQ = 0x08, COM_IRQ_IDLE_IRQ = 0x10, C↩
  OM_IRQ_RX_IRQ = 0x20,
  COM_IRQ_TX_IRQ = 0x40, COM_IRQ_ALL_IRQ = 0x7f, COM_IRQ_SET1 = 0x80, DIV_I_EN_CRC_I_EN =

0x04,
DIV_I_EN_MFIN_ACT_I_EN = 0x10, DIV_I_EN_INTERRUPT_EN = DIV_I_EN_CRC_I_EN | DIV_I_EN_↩
MFIN_ACT_I_EN, DIV_IRQ_CRC_IRQ = 0x04, DIV_IRQ_MFIN_ACT_IRQ = 0x10,
DIV_IRQ_ALL_IRQ = DIV_IRQ_CRC_IRQ | DIV_IRQ_MFIN_ACT_IRQ, DIV_IRQ_SET2 = 0x80, FIFO_L↩
EVEL_FLUSH_BUFFER = 0x80, FIFO_LEVEL_FIFO_LEVEL = 0x7f,
WATER_LEVEL_WATER_LEVEL = 0x3f, BIT_FRAMING_START_SEND = 0x80, AUTO_TEST_ENABLE =
0x09 }
- enum Interrupt : unsigned int {
NONE_IRQ = 0x0000, COM_TIMER_IRQ = 0x0001, COM_ERR_IRQ = 0x0002, COM_LO_ALERT_IRQ =
0x0004,
COM_HI_ALERT_IRQ = 0x0008, COM_IDLE_IRQ = 0x0010, COM_RX_IRQ = 0x0020, COM_TX_IRQ =
0x0040,
COM_ALL_IRQ = 0x007f, DIV_CRC_IRQ = 0x0400, DIV_MFIN_ACT_IRQ = 0x1000, DIV_ALL_IRQ = DI↩
V_CRC_IRQ | DIV_MFIN_ACT_IRQ }
- enum Version { CLONE = 0x88, V0_0 = 0x90, V1_0 = 0x91, V2_0 = 0x92 }

Public Member Functions

- RadioFrequencyIdentificationMFRC522 (RegisterBasedDevice ∗device, unsigned char resetPin)
- void initialize ()
- void sendCommand (Command command)
- unsigned char getLastError ()
- void clearLastError ()
- void softReset ()
- void setAntennaOn ()
- void setAntennaOff ()
- void configureTimer (unsigned int prescaler, unsigned int reload, bool autoStart, bool autoRestart)
- void startTimer ()
- void stopTimer ()
- void enableInterrupt (Interrupt interrupt)
- void disableInterrupt (Interrupt interrupt)
- void clearInterrupt (Interrupt interrupt)
- void flushQueue ()
- void setWaterLevel (unsigned char level)
- int generateRandomId (unsigned char buf[10])
- int communicate (Command command, unsigned char ∗output, unsigned char ∗input, unsigned char output↩
Len, bool checkCRC)
- unsigned int calculateCRC (unsigned char ∗buff, unsigned char len)
- bool waitForRegisterBits (unsigned char reg, unsigned char mask)
- bool waitForRegisterBits (unsigned char reg, unsigned char mask, unsigned long timeout)
- Version getVersion ()
- bool performSelfTest ()
- int readRegisterBlock (unsigned char reg, unsigned char ∗buf, unsigned char len)
- unsigned char writeRegisterBlock (unsigned char reg, unsigned char ∗buf, unsigned char len)

Private Attributes

- RegisterBasedDevice ∗ device
- unsigned char resetPin
- unsigned char lastError

4.19.1 Detailed Description

Definition at line 57 of file RadioFrequencyIdentificationMFRC522.h.

4.19.2 Member Enumeration Documentation

4.19.2.1 enum RadioFrequencyIdentificationMFRC522::Command

Enumerator

- ***IDLE***
- ***MEM***
- ***GENERATE_RANDOM_ID***
- ***CALC_CRC***
- ***TRANSMIT***
- ***NO_CMD_CHANGE***
- ***RECEIVE***
- ***TRANSCEIVE***
- ***MF_AUTHENT***
- ***SOFT_RESET***

Definition at line 221 of file RadioFrequencyIdentificationMFRC522.h.

4.19.2.2 enum RadioFrequencyIdentificationMFRC522::Error

Enumerator

- ***NO_ERROR***
- ***GENERAL_ERROR***
- ***TIMEOUT_ERROR***
- ***COMMUNICATION_ERROR***
- ***CRC_ERROR***

Definition at line 254 of file RadioFrequencyIdentificationMFRC522.h.

4.19.2.3 enum RadioFrequencyIdentificationMFRC522::Interrupt : unsigned int

Enumerator

- ***NONE_IRQ***
- ***COM_TIMER_IRQ***
- ***COM_ERR_IRQ***
- ***COM_LO_ALERT_IRQ***
- ***COM_HI_ALERT_IRQ***
- ***COM_IDLE_IRQ***
- ***COM_RX_IRQ***
- ***COM_TX_IRQ***
- ***COM_ALL_IRQ***
- ***DIV_CRC_IRQ***
- ***DIV_MFIN_ACT_IRQ***
- ***DIV_ALL_IRQ***

Definition at line 292 of file RadioFrequencyIdentificationMFRC522.h.

4.19.2.4 enum RadioFrequencyIdentificationMFRC522::Mask

Enumerator

- ***TX_CONTROL_TX1_RF_EN***
- ***TX_CONTROL_TX2_RF_EN***
- ***TX_CONTROL_TX_RF_EN***
- ***CONTROL_T_STOP_NOW***
- ***CONTROL_T_START_NOW***
- ***COM_I_EN_INTERRUPT_EN***
- ***COM_IRQ_TIMER_IRQ***
- ***COM_IRQ_ERR_IRQ***
- ***COM_IRQ_LO_ALERT_IRQ***
- ***COM_IRQ_HI_ALERT_IRQ***
- ***COM_IRQ_IDLE_IRQ***
- ***COM_IRQ_RX_IRQ***
- ***COM_IRQ_TX_IRQ***
- ***COM_IRQ_ALL_IRQ***
- ***COM_IRQ_SET1***
- ***DIV_I_EN_CRC_I_EN***
- ***DIV_I_EN_MFIN_ACT_I_EN***
- ***DIV_I_EN_INTERRUPT_EN***
- ***DIV_IRQ_CRC_IRQ***
- ***DIV_IRQ_MFIN_ACT_IRQ***
- ***DIV_IRQ_ALL_IRQ***
- ***DIV_IRQ_SET2***
- ***FIFO_LEVEL_FLUSH_BUFFER***
- ***FIFO_LEVEL_FIFO_LEVEL***
- ***WATER_LEVEL_WATER_LEVEL***
- ***BIT_FRAMING_START_SEND***
- ***AUTO_TEST_ENABLE***

Definition at line 262 of file RadioFrequencyIdentificationMFRC522.h.

4.19.2.5 enum RadioFrequencyIdentificationMFRC522::Register

Enumerator

- ***COMMAND***
- ***COM_I_EN***
- ***DIV_I_EN***
- ***COM_IRQ***
- ***DIV_IRQ***
- ***ERROR***
- ***STATUS1***
- ***STATUS2***
- ***FIFO_DATA***

FIFO_LEVEL

WATER_LEVEL

CONTROL

BIT_FRAMING

COLL

MODE

TX_MODE

RX_MODE

TX_CONTROL

TX_ASK

TX_SEL

RX_SEL

RX_THRESHOLD

DEMOD

MF_TX

MF_RX

SERIAL_SPEED

CRC_RESULT_HIDH

CRC_RESULT_LOW

MOD_WIDTH

RFC_FG

GS_N

CW_GS_P

MOD_GS_P

T_MODE

T_PRESCALER_LOW

T_RELOAD_HIGH

T_RELOAD_LOW

T_COUNTER_VAL_HIGH

T_COUNTER_VAL_LOW

TEST_SEL1

TEST_SEL2

TEST_PIN_EN

TEST_PIN_VALUE

TEST_BUS

AUTO_TEST

VERSION

ANALOG_TEST

TEST_DAC1

TEST_DAC2

TEST_ADC

Definition at line 68 of file RadioFrequencyIdentificationMFRC522.h.

4.19.2.6 enum RadioFrequencyIdentificationMFRC522::Version

Enumerator

CLONE

V0_0

V1_0

V2_0

Definition at line 1261 of file RadioFrequencyIdentificationMFRC522.h.

4.19.3 Constructor & Destructor Documentation

4.19.3.1 RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522 (RegisterBasedDevice ∗ *device*, unsigned char *resetPin*)

Definition at line 4 of file RadioFrequencyIdentificationMFRC522.cpp.

4.19.4 Member Function Documentation

4.19.4.1 unsigned int RadioFrequencyIdentificationMFRC522::calculateCRC (unsigned char ∗ *buff*, unsigned char *len*)

Definition at line 218 of file RadioFrequencyIdentificationMFRC522.cpp.

4.19.4.2 void RadioFrequencyIdentificationMFRC522::clearInterrupt (Interrupt *interrupt*)

Definition at line 98 of file RadioFrequencyIdentificationMFRC522.cpp.

4.19.4.3 void RadioFrequencyIdentificationMFRC522::clearLastError ( ) [inline]

Definition at line 1281 of file RadioFrequencyIdentificationMFRC522.h.

4.19.4.4 int RadioFrequencyIdentificationMFRC522::communicate (Command *command*, unsigned char ∗ *output*, unsigned char ∗ *input*, unsigned char *outputLen*, bool *checkCRC*)

Definition at line 142 of file RadioFrequencyIdentificationMFRC522.cpp.

4.19.4.5 void RadioFrequencyIdentificationMFRC522::configureTimer (unsigned int *prescaler*, unsigned int *reload*, bool *autoStart*, bool *autoRestart*)

The timer unit can be used to measure the time interval between two events or to indicate that a specific event occurred after a specific time.

The timer can be triggered by events explained in the paragraphs below. The timer does not influence any internal events, for example, a time-out during data reception does not automatically influence the reception process. Furthermore, several timer-related bits can be used to generate an interrupt. The timer has an input clock of 13.56 MHz derived from the 27.12 MHz quartz crystal oscillator. The timer consists of two stages: prescaler and counter. The prescaler (TPrescaler) is a 12-bit counter. The reload values (TReloadVal_Hi[7:0] and TReloadVal_Lo[7:0]) for TPrescaler can be set between 0 and 4095 in the TModeReg register's TPrescaler_Hi[3:0] bits and TPrescalerReg register's TPrescaler_Lo[7:0] bits. The reload value for the counter is defined by 16 bits between 0 and 65535 in the TReloadReg register. The current value of the timer is indicated in the TCounterValReg register. When the counter reaches 0, an interrupt is automatically generated, indicated by the ComIrqReg register's TimerIRq bit setting. If enabled, this event can be indicated on pin IRQ. The TimerIRq bit can be set and reset by the host. Depending on the configuration, the timer will stop at 0 or restart with the value set in the TReloadReg register. The timer status is indicated by the Status1Reg register's TRunning bit. The timer can be started manually using the ControlReg register's TStartNow bit and stopped using the ControlReg register's TStopNow bit. The timer can also be activated automatically to meet any dedicated protocol requirements by setting the TModeReg register's TAuto bit to logic 1.

Parameters

prescaler	12 bit prescaler value.
reload	16 bit reload value.
autoStart	1: timer starts automatically at the end of the transmission in all communication modes at all speeds if the RxModeReg register's RxMultiple bit is not set, the timer stops immediately after receiving the 5th bit (1 start bit, 4 data bits) if the RxMultiple bit is set to logic 1 the timer never stops, in which case the timer can be stopped by setting the ControlReg register's TStopNow bit to logic 1 0: indicates that the timer is not influenced by the protocol
autoRestart	1: timer automatically restarts its count-down from the 16-bit timer reload value instead of counting down to zero 0 timer decrements to 0 and the ComIrqReg register's TimerIRq bit is set to logic 1

Definition at line 68 of file RadioFrequencyIdentificationMFRC522.cpp.

4.19.4.6 void RadioFrequencyIdentificationMFRC522::disableInterrupt (Interrupt *interrupt*)

Definition at line 94 of file RadioFrequencyIdentificationMFRC522.cpp.

4.19.4.7 void RadioFrequencyIdentificationMFRC522::enableInterrupt (Interrupt *interrupt*)

Definition at line 90 of file RadioFrequencyIdentificationMFRC522.cpp.

4.19.4.8 void RadioFrequencyIdentificationMFRC522::flushQueue ( )

Definition at line 105 of file RadioFrequencyIdentificationMFRC522.cpp.

4.19.4.9 int RadioFrequencyIdentificationMFRC522::generateRandomId (unsigned char *buf[10]*)

Definition at line 113 of file RadioFrequencyIdentificationMFRC522.cpp.

4.19.4.10 unsigned char RadioFrequencyIdentificationMFRC522::getLastError ( ) `[inline]`

Definition at line 1277 of file RadioFrequencyIdentificationMFRC522.h.

4.19.4.11 RadioFrequencyIdentificationMFRC522::Version RadioFrequencyIdentificationMFRC522::getVersion ( )

Definition at line 297 of file RadioFrequencyIdentificationMFRC522.cpp.

4.19.4.12 void RadioFrequencyIdentificationMFRC522::initialize ( )

Definition at line 11 of file RadioFrequencyIdentificationMFRC522.cpp.

4.19.4.13 bool RadioFrequencyIdentificationMFRC522::performSelfTest ( )

1. Perform a soft reset.
2. Clear the internal buffer by writing 25 bytes of 00h and implement the Config command.
3. Enable the self test by writing 09h to the AutoTestReg register.
4. Write 00h to the FIFO buffer.
5. Start the self test with the CalcCRC command.
6. The self test is initiated.
7. When the self test has completed, the FIFO buffer contains the following 64 bytes:

FIFO buffer byte values for MFRC522 version 1.0: 00h, C6h, 37h, D5h, 32h, B7h, 57h, 5Ch, C2h, D8h, 7Ch, 4Dh, D9h, 70h, C7h, 73h, 10h, E6h, D2h, AAh, 5Eh, A1h, 3Eh, 5Ah, 14h, AFh, 30h, 61h, C9h, 70h, DBh, 2Eh, 64h, 22h, 72h, B5h, BDh, 65h, F4h, ECh, 22h, BCh, D3h, 72h, 35h, CDh, AAh, 41h, 1Fh, A7h, F3h, 53h, 14h, DEh, 7Eh, 02h, D9h, 0Fh, B5h, 5Eh, 25h, 1Dh, 29h, 79h

FIFO buffer byte values for MFRC522 version 2.0: 00h, EBh, 66h, BAh, 57h, BFh, 23h, 95h, D0h, E3h, 0Dh, 3Dh, 27h, 89h, 5Ch, DEh, 9Dh, 3Bh, A7h, 00h, 21h, 5Bh, 89h, 82h, 51h, 3Ah, EBh, 02h, 0Ch, A5h, 00h, 49h, 7Ch, 84h, 4Dh, B3h, CCh, D2h, 1Bh, 81h, 5Dh, 48h, 76h, D5h, 71h, 061h, 21h, A9h, 86h, 96h, 83h, 38h, CFh, 9Dh, 5Bh, 6Dh, DCh, 15h, BAh, 3Eh, 7Dh, 95h, 03Bh, 2Fh

Definition at line 260 of file RadioFrequencyIdentificationMFRC522.cpp.

4.19.4.14 int RadioFrequencyIdentificationMFRC522::readRegisterBlock (unsigned char *reg,* unsigned char ∗ *buf,* unsigned char *len*)

Reads values from the device, starting by the reg register.

Parameters

reg	The register number.
buf	The buffer where to place read bytes. MSB become LSB inside buffer.
len	How many bytes to read.

Returns

If >= 0: How many bytes were read. If < 0: Error code:

- -1: data too long to fit in transmit buffer
- -2: received NACK on transmit of address
- -3: received NACK on transmit of data
- -4: other error
- -5: timeout

Definition at line 54 of file RadioFrequencyIdentificationMFRC522.cpp.

4.19.4.15 void RadioFrequencyIdentificationMFRC522::sendCommand (Command *command*) [inline]

Definition at line 1273 of file RadioFrequencyIdentificationMFRC522.h.

4.19.4.16 void RadioFrequencyIdentificationMFRC522::setAntennaOff ( )

Definition at line 50 of file RadioFrequencyIdentificationMFRC522.cpp.

4.19.4.17 void RadioFrequencyIdentificationMFRC522::setAntennaOn ( )

Definition at line 46 of file RadioFrequencyIdentificationMFRC522.cpp.

4.19.4.18 void RadioFrequencyIdentificationMFRC522::setWaterLevel (unsigned char *level*)

Definition at line 109 of file RadioFrequencyIdentificationMFRC522.cpp.

4.19.4.19 void RadioFrequencyIdentificationMFRC522::softReset ( )

Definition at line 42 of file RadioFrequencyIdentificationMFRC522.cpp.

4.19.4.20 void RadioFrequencyIdentificationMFRC522::startTimer ( )

Definition at line 82 of file RadioFrequencyIdentificationMFRC522.cpp.

4.19.4.21 void RadioFrequencyIdentificationMFRC522::stopTimer ( )

Definition at line 86 of file RadioFrequencyIdentificationMFRC522.cpp.

4.19.4.22 bool RadioFrequencyIdentificationMFRC522::waitForRegisterBits (unsigned char *reg,* unsigned char *mask*) `[inline]`

Definition at line 1358 of file RadioFrequencyIdentificationMFRC522.h.

4.19.4.23 bool RadioFrequencyIdentificationMFRC522::waitForRegisterBits (unsigned char *reg,* unsigned char *mask,* unsigned long *timeout*)

Definition at line 250 of file RadioFrequencyIdentificationMFRC522.cpp.

4.19.4.24 unsigned char RadioFrequencyIdentificationMFRC522::writeRegisterBlock (unsigned char *reg,* unsigned char ∗ *buf,* unsigned char *len*)

Writes a sequence of values to a sequence of registers, starting by the reg address.

Parameters

reg	The register number.
buf	The buffer.
len	Buffer length.

Returns

The result of Wire.endTransmission().

Definition at line 61 of file RadioFrequencyIdentificationMFRC522.cpp.

4.19.5 Member Data Documentation

4.19.5.1 RegisterBasedDevice∗ RadioFrequencyIdentificationMFRC522::device [private]

Definition at line 60 of file RadioFrequencyIdentificationMFRC522.h.

4.19.5.2 unsigned char RadioFrequencyIdentificationMFRC522::lastError [private]

Definition at line 64 of file RadioFrequencyIdentificationMFRC522.h.

4.19.5.3 unsigned char RadioFrequencyIdentificationMFRC522::resetPin [private]

Definition at line 62 of file RadioFrequencyIdentificationMFRC522.h.

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

- RadioFrequencyIdentificationMFRC522.h
- RadioFrequencyIdentificationMFRC522.cpp

4.20 RadioFrequencyIdentificationMFRC522::RF_CFGbits Union Reference

```
#include <RadioFrequencyIdentificationMFRC522.h>
```

Public Attributes

- struct {
  unsigned char:4
  unsigned char RX_GAIN:3
  };
- unsigned char value

4.20.1 Detailed Description

RF_CFG register.

Configures the receiver gain.

Definition at line 1105 of file RadioFrequencyIdentificationMFRC522.h.

4.20.2 Member Data Documentation

4.20.2.1 struct { ... }

4.20.2.2 unsigned RadioFrequencyIdentificationMFRC522::RF_CFGbits::char

Definition at line 1110 of file RadioFrequencyIdentificationMFRC522.h.

4.20.2.3 unsigned char RadioFrequencyIdentificationMFRC522::RF_CFGbits::RX_GAIN

Definition at line 1121 of file RadioFrequencyIdentificationMFRC522.h.

4.20.2.4 unsigned char RadioFrequencyIdentificationMFRC522::RF_CFGbits::value

Definition at line 1126 of file RadioFrequencyIdentificationMFRC522.h.

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

- RadioFrequencyIdentificationMFRC522.h

4.21 RadioFrequencyIdentificationMFRC522::RX_MODEbits Union Reference

```
#include <RadioFrequencyIdentificationMFRC522.h>
```

Public Attributes

- struct {
  unsigned char:2
  unsigned char RX_MULTIPLE:1
  unsigned char RX_NO_ERR:1
  unsigned char RX_SPEED:3
  unsigned char RX_CRC_EN:1
  };
- unsigned char value

4.21.1 Detailed Description

RX_MODE register.

Defines the data rate during reception.

Definition at line 817 of file RadioFrequencyIdentificationMFRC522.h.

4.21.2 Member Data Documentation

4.21.2.1 struct { ... }

4.21.2.2 unsigned RadioFrequencyIdentificationMFRC522::RX_MODEbits::char

Definition at line 822 of file RadioFrequencyIdentificationMFRC522.h.

4.21.2.3 unsigned char RadioFrequencyIdentificationMFRC522::RX_MODEbits::RX_CRC_EN

Definition at line 846 of file RadioFrequencyIdentificationMFRC522.h.

4.21.2.4 unsigned char RadioFrequencyIdentificationMFRC522::RX_MODEbits::RX_MULTIPLE

Definition at line 832 of file RadioFrequencyIdentificationMFRC522.h.

4.21.2.5 unsigned char RadioFrequencyIdentificationMFRC522::RX_MODEbits::RX_NO_ERR

Definition at line 835 of file RadioFrequencyIdentificationMFRC522.h.

4.21.2.6 unsigned char RadioFrequencyIdentificationMFRC522::RX_MODEbits::RX_SPEED

Definition at line 842 of file RadioFrequencyIdentificationMFRC522.h.

4.21.2.7 unsigned char RadioFrequencyIdentificationMFRC522::RX_MODEbits::value

Definition at line 848 of file RadioFrequencyIdentificationMFRC522.h.

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

- RadioFrequencyIdentificationMFRC522.h

4.22 RadioFrequencyIdentificationMFRC522::RX_SELbits Union Reference

```
#include <RadioFrequencyIdentificationMFRC522.h>
```

Public Attributes

- struct {
    unsigned char RX_WAIT:6
    unsigned char UART_SEL:2
  };
- unsigned char value

4.22.1 Detailed Description

RX_SEL register.

Selects internal receiver settings.

Definition at line 966 of file RadioFrequencyIdentificationMFRC522.h.

4.22.2 Member Data Documentation

4.22.2.1 struct { ... }

4.22.2.2 unsigned char RadioFrequencyIdentificationMFRC522::RX_SELbits::RX_WAIT

Definition at line 973 of file RadioFrequencyIdentificationMFRC522.h.

4.22.2.3 unsigned char RadioFrequencyIdentificationMFRC522::RX_SELbits::UART_SEL

Definition at line 980 of file RadioFrequencyIdentificationMFRC522.h.

4.22.2.4 unsigned char RadioFrequencyIdentificationMFRC522::RX_SELbits::value

Definition at line 982 of file RadioFrequencyIdentificationMFRC522.h.

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

- RadioFrequencyIdentificationMFRC522.h

4.23 RadioFrequencyIdentificationMFRC522::RX_THRESHOLDbits Union Reference

```
#include <RadioFrequencyIdentificationMFRC522.h>
```

Public Attributes

- struct {
    unsigned char COLL_LEVEL:3
    unsigned char:1
    unsigned char MIN_LEVEL:4
  };
- unsigned char value

4.23.1 Detailed Description

RX_THRESHOLD register.

Selects thresholds for the bit decoder.

Definition at line 990 of file RadioFrequencyIdentificationMFRC522.h.

4.23.2 Member Data Documentation

4.23.2.1 struct { ... }

4.23.2.2 unsigned RadioFrequencyIdentificationMFRC522::RX_THRESHOLDbits::char

Definition at line 999 of file RadioFrequencyIdentificationMFRC522.h.

4.23.2.3 unsigned char RadioFrequencyIdentificationMFRC522::RX_THRESHOLDbits::COLL_LEVEL

Definition at line 996 of file RadioFrequencyIdentificationMFRC522.h.

4.23.2.4 unsigned char RadioFrequencyIdentificationMFRC522::RX_THRESHOLDbits::MIN_LEVEL

Definition at line 1002 of file RadioFrequencyIdentificationMFRC522.h.

4.23.2.5 unsigned char RadioFrequencyIdentificationMFRC522::RX_THRESHOLDbits::value

Definition at line 1004 of file RadioFrequencyIdentificationMFRC522.h.

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

- RadioFrequencyIdentificationMFRC522.h

4.24 RadioFrequencyIdentificationMFRC522::SERIAL_SPEEDbits Union Reference

```
#include <RadioFrequencyIdentificationMFRC522.h>
```

Public Attributes

- struct {
    unsigned char BR_T1:5
    unsigned char BR_T0:3
  };
- unsigned char value

4.24.1 Detailed Description

SERIAL_SPEED register.

Selects the speed of the serial UART interface.

Definition at line 1087 of file RadioFrequencyIdentificationMFRC522.h.

4.24.2 Member Data Documentation

4.24.2.1 struct { ... }

4.24.2.2 unsigned char RadioFrequencyIdentificationMFRC522::SERIAL_SPEEDbits::BR_T0

Definition at line 1095 of file RadioFrequencyIdentificationMFRC522.h.

4.24.2.3 unsigned char RadioFrequencyIdentificationMFRC522::SERIAL_SPEEDbits::BR_T1

Definition at line 1092 of file RadioFrequencyIdentificationMFRC522.h.

4.24.2.4 unsigned char RadioFrequencyIdentificationMFRC522::SERIAL_SPEEDbits::value

Definition at line 1097 of file RadioFrequencyIdentificationMFRC522.h.

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

- RadioFrequencyIdentificationMFRC522.h

4.25 RadioFrequencyIdentificationMFRC522::STATUS1bits Union Reference

```
#include <RadioFrequencyIdentificationMFRC522.h>
```

Public Attributes

- struct {
    unsigned char LO_ALERT:1
    unsigned char HI_ALERT:1
    unsigned char T_RUNNING:1
    unsigned char IRQ:1
    unsigned char CRC_READY:1
    unsigned char CRC_OK:1
    unsigned char:1
  };
- unsigned char value

4.25.1 Detailed Description

STATUS1 register.

Contains status bits of the CRC, interrupt and FIFO buffer.

Definition at line 530 of file RadioFrequencyIdentificationMFRC522.h.

4.25.2 Member Data Documentation

4.25.2.1 struct { ... }

4.25.2.2 unsigned RadioFrequencyIdentificationMFRC522::STATUS1bits::char

Definition at line 567 of file RadioFrequencyIdentificationMFRC522.h.

4.25.2.3 unsigned char RadioFrequencyIdentificationMFRC522::STATUS1bits::CRC_OK

Definition at line 564 of file RadioFrequencyIdentificationMFRC522.h.

4.25.2.4 unsigned char RadioFrequencyIdentificationMFRC522::STATUS1bits::CRC_READY

Definition at line 558 of file RadioFrequencyIdentificationMFRC522.h.

4.25.2.5 unsigned char RadioFrequencyIdentificationMFRC522::STATUS1bits::HI_ALERT

Definition at line 546 of file RadioFrequencyIdentificationMFRC522.h.

4.25.2.6 unsigned char RadioFrequencyIdentificationMFRC522::STATUS1bits::IRQ

Definition at line 555 of file RadioFrequencyIdentificationMFRC522.h.

4.25.2.7 unsigned char RadioFrequencyIdentificationMFRC522::STATUS1bits::LO_ALERT

Definition at line 539 of file RadioFrequencyIdentificationMFRC522.h.

4.25.2.8 unsigned char RadioFrequencyIdentificationMFRC522::STATUS1bits::T_RUNNING

Definition at line 551 of file RadioFrequencyIdentificationMFRC522.h.

4.25.2.9 unsigned char RadioFrequencyIdentificationMFRC522::STATUS1bits::value

Definition at line 569 of file RadioFrequencyIdentificationMFRC522.h.

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

- RadioFrequencyIdentificationMFRC522.h

4.26 RadioFrequencyIdentificationMFRC522::STATUS2bits Union Reference

```
#include <RadioFrequencyIdentificationMFRC522.h>
```

Public Attributes

- struct {
    unsigned char MODEM_STATE:3
    unsigned char MF_CRYPTO1_ON:1
    unsigned char:2
    unsigned char I2C_FORCE_HS:1
    unsigned char TEMP_SENS_CLEAR:1
  };
- unsigned char value

4.26.1 Detailed Description

STATUS2 register.

Contains status bits of the receiver, transmitter and data mode detector.

Definition at line 577 of file RadioFrequencyIdentificationMFRC522.h.

4.26.2 Member Data Documentation

4.26.2.1 struct { ... }

4.26.2.2 unsigned RadioFrequencyIdentificationMFRC522::STATUS2bits::char

Definition at line 599 of file RadioFrequencyIdentificationMFRC522.h.

4.26.2.3 unsigned char RadioFrequencyIdentificationMFRC522::STATUS2bits::I2C_FORCE_HS

Definition at line 604 of file RadioFrequencyIdentificationMFRC522.h.

4.26.2.4 unsigned char RadioFrequencyIdentificationMFRC522::STATUS2bits::MF_CRYPTO1_ON

Definition at line 596 of file RadioFrequencyIdentificationMFRC522.h.

4.26.2.5 unsigned char RadioFrequencyIdentificationMFRC522::STATUS2bits::MODEM_STATE

Definition at line 591 of file RadioFrequencyIdentificationMFRC522.h.

4.26.2.6 unsigned char RadioFrequencyIdentificationMFRC522::STATUS2bits::TEMP_SENS_CLEAR

Definition at line 607 of file RadioFrequencyIdentificationMFRC522.h.

4.26.2.7 unsigned char RadioFrequencyIdentificationMFRC522::STATUS2bits::value

Definition at line 609 of file RadioFrequencyIdentificationMFRC522.h.

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

- RadioFrequencyIdentificationMFRC522.h

4.27 RadioFrequencyIdentificationMFRC522::T_MODEbits Union Reference

```
#include <RadioFrequencyIdentificationMFRC522.h>
```

Public Attributes

- struct {
    unsigned char T_PRESCALER_HI:4
    unsigned char T_AUTO_RESTART:1
    unsigned char T_GATED:2
    unsigned char T_AUTO:1
  };
- unsigned char value

4.27.1 Detailed Description

T_MODE register.

These registers define the timer settings.

Remark: The TPrescaler setting higher 4 bits are in the TModeReg register and the lower 8 bits are in the T↩
PrescalerReg register.

Definition at line 1198 of file RadioFrequencyIdentificationMFRC522.h.

4.27.2 Member Data Documentation

4.27.2.1 struct { ... }

4.27.2.2 unsigned char RadioFrequencyIdentificationMFRC522::T_MODEbits::T_AUTO

Definition at line 1227 of file RadioFrequencyIdentificationMFRC522.h.

4.27.2.3 unsigned char RadioFrequencyIdentificationMFRC522::T_MODEbits::T_AUTO_RESTART

Definition at line 1212 of file RadioFrequencyIdentificationMFRC522.h.

4.27.2.4 unsigned char RadioFrequencyIdentificationMFRC522::T_MODEbits::T_GATED

Definition at line 1220 of file RadioFrequencyIdentificationMFRC522.h.

4.27.2.5 unsigned char RadioFrequencyIdentificationMFRC522::T_MODEbits::T_PRESCALER_HI

Definition at line 1208 of file RadioFrequencyIdentificationMFRC522.h.

4.27.2.6 unsigned char RadioFrequencyIdentificationMFRC522::T_MODEbits::value

Definition at line 1229 of file RadioFrequencyIdentificationMFRC522.h.

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

- RadioFrequencyIdentificationMFRC522.h

4.28 RadioFrequencyIdentificationMFRC522::TX_ASKbits Union Reference

```
#include <RadioFrequencyIdentificationMFRC522.h>
```

Public Attributes

- struct {
    unsigned char:6
    unsigned char FORCE_100_ASK:1
  };
- unsigned char value

4.28.1 Detailed Description

TX_ASK register.

Controls transmit modulation settings.

Definition at line 911 of file RadioFrequencyIdentificationMFRC522.h.

4.28.2 Member Data Documentation

4.28.2.1 struct { ... }

4.28.2.2 unsigned RadioFrequencyIdentificationMFRC522::TX_ASKbits::char

Definition at line 916 of file RadioFrequencyIdentificationMFRC522.h.

4.28.2.3 unsigned char RadioFrequencyIdentificationMFRC522::TX_ASKbits::FORCE_100_ASK

Definition at line 919 of file RadioFrequencyIdentificationMFRC522.h.

4.28.2.4 unsigned char RadioFrequencyIdentificationMFRC522::TX_ASKbits::value

Definition at line 924 of file RadioFrequencyIdentificationMFRC522.h.

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

- RadioFrequencyIdentificationMFRC522.h

4.29 RadioFrequencyIdentificationMFRC522::TX_CONTROLbits Union Reference

```
#include <RadioFrequencyIdentificationMFRC522.h>
```

Public Attributes

- struct {
  unsigned char TX1_RF_EN:1
  unsigned char TX2_RF_EN:1
  unsigned char:1
  unsigned char TX2_CW:1
  unsigned char INV_TX1_RF_OFF:1
  unsigned char INV_TX2_RF_OFF:1
  unsigned char INV_TX1_RF_ON:1
  unsigned char INV_TX2_RF_ON:1
  };

- struct {
  unsigned char TX_RF_EN:2
  unsigned char:2
  unsigned char INV_TX_RF_OFF:2
  unsigned char INV_TX_RF_ON:2
  };

- unsigned char value

4.29.1 Detailed Description

TX_CONTROL register.

Controls the logical behavior of the antenna driver pins TX1 and TX2.

Definition at line 856 of file RadioFrequencyIdentificationMFRC522.h.

4.29.2 Member Data Documentation

4.29.2.1 struct { ... }

4.29.2.2 struct { ... }

4.29.2.3 unsigned RadioFrequencyIdentificationMFRC522::TX_CONTROLbits::char

Definition at line 867 of file RadioFrequencyIdentificationMFRC522.h.

4.29.2.4 unsigned char RadioFrequencyIdentificationMFRC522::TX_CONTROLbits::INV_TX1_RF_OFF

Definition at line 874 of file RadioFrequencyIdentificationMFRC522.h.

4.29.2.5 unsigned char RadioFrequencyIdentificationMFRC522::TX_CONTROLbits::INV_TX1_RF_ON

Definition at line 880 of file RadioFrequencyIdentificationMFRC522.h.

4.29.2.6 unsigned char RadioFrequencyIdentificationMFRC522::TX_CONTROLbits::INV_TX2_RF_OFF

Definition at line 877 of file RadioFrequencyIdentificationMFRC522.h.

4.29.2.7 unsigned char RadioFrequencyIdentificationMFRC522::TX_CONTROLbits::INV_TX2_RF_ON

Definition at line 883 of file RadioFrequencyIdentificationMFRC522.h.

4.29.2.8 unsigned char RadioFrequencyIdentificationMFRC522::TX_CONTROLbits::INV_TX_RF_OFF

Definition at line 897 of file RadioFrequencyIdentificationMFRC522.h.

4.29.2.9 unsigned char RadioFrequencyIdentificationMFRC522::TX_CONTROLbits::INV_TX_RF_ON

Definition at line 901 of file RadioFrequencyIdentificationMFRC522.h.

4.29.2.10 unsigned char RadioFrequencyIdentificationMFRC522::TX_CONTROLbits::TX1_RF_EN

Definition at line 861 of file RadioFrequencyIdentificationMFRC522.h.

4.29.2.11 unsigned char RadioFrequencyIdentificationMFRC522::TX_CONTROLbits::TX2_CW

Definition at line 871 of file RadioFrequencyIdentificationMFRC522.h.

4.29.2.12 unsigned char RadioFrequencyIdentificationMFRC522::TX_CONTROLbits::TX2_RF_EN

Definition at line 864 of file RadioFrequencyIdentificationMFRC522.h.

4.29.2.13 unsigned char RadioFrequencyIdentificationMFRC522::TX_CONTROLbits::TX_RF_EN

Definition at line 890 of file RadioFrequencyIdentificationMFRC522.h.

4.29.2.14 unsigned char RadioFrequencyIdentificationMFRC522::TX_CONTROLbits::value

Definition at line 903 of file RadioFrequencyIdentificationMFRC522.h.

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

- RadioFrequencyIdentificationMFRC522.h

4.30 RadioFrequencyIdentificationMFRC522::TX_MODEbits Union Reference

```
#include <RadioFrequencyIdentificationMFRC522.h>
```

Public Attributes

- struct {
  unsigned char:3
  unsigned char INV_MOD:1
  unsigned char TX_SPEED:3
  unsigned char TX_CRC_EN:1
  };
- unsigned char value

4.30.1 Detailed Description

TX_MODE register.

Defines the data rate during transmission.

Definition at line 788 of file RadioFrequencyIdentificationMFRC522.h.

4.30.2 Member Data Documentation

4.30.2.1 struct { ... }

4.30.2.2 unsigned RadioFrequencyIdentificationMFRC522::TX_MODEbits::char

Definition at line 793 of file RadioFrequencyIdentificationMFRC522.h.

4.30.2.3 unsigned char RadioFrequencyIdentificationMFRC522::TX_MODEbits::INV_MOD

Definition at line 796 of file RadioFrequencyIdentificationMFRC522.h.

4.30.2.4 unsigned char RadioFrequencyIdentificationMFRC522::TX_MODEbits::TX_CRC_EN

Definition at line 807 of file RadioFrequencyIdentificationMFRC522.h.

4.30.2.5 unsigned char RadioFrequencyIdentificationMFRC522::TX_MODEbits::TX_SPEED

Definition at line 803 of file RadioFrequencyIdentificationMFRC522.h.

4.30.2.6 unsigned char RadioFrequencyIdentificationMFRC522::TX_MODEbits::value

Definition at line 809 of file RadioFrequencyIdentificationMFRC522.h.

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

- RadioFrequencyIdentificationMFRC522.h

4.31 RadioFrequencyIdentificationMFRC522::TX_SELbits Union Reference

```
#include <RadioFrequencyIdentificationMFRC522.h>
```

Public Attributes

- struct {
  unsigned char MF_OUT_SEL:4
  unsigned char:2
  };
- unsigned char value

4.31.1 Detailed Description

TX_SEL register.

Selects the internal sources for the analog module.

Definition at line 932 of file RadioFrequencyIdentificationMFRC522.h.

4.31.2 Member Data Documentation

4.31.2.1 struct { ... }

4.31.2.2 unsigned RadioFrequencyIdentificationMFRC522::TX_SELbits::char

Definition at line 953 of file RadioFrequencyIdentificationMFRC522.h.

4.31.2.3 unsigned char RadioFrequencyIdentificationMFRC522::TX_SELbits::MF_OUT_SEL

Definition at line 946 of file RadioFrequencyIdentificationMFRC522.h.

4.31.2.4 unsigned char RadioFrequencyIdentificationMFRC522::TX_SELbits::value

Definition at line 958 of file RadioFrequencyIdentificationMFRC522.h.

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

- RadioFrequencyIdentificationMFRC522.h

4.32 RadioFrequencyIdentificationMFRC522::Uid Struct Reference

```
#include <RadioFrequencyIdentificationMFRC522.h>
```

Public Attributes

- unsigned char size
- unsigned char uid [10]
- unsigned char sak

4.32.1 Detailed Description

Definition at line 1250 of file RadioFrequencyIdentificationMFRC522.h.

4.32.2 Member Data Documentation

4.32.2.1 unsigned char RadioFrequencyIdentificationMFRC522::Uid::sak

Definition at line 1258 of file RadioFrequencyIdentificationMFRC522.h.

4.32.2.2 unsigned char RadioFrequencyIdentificationMFRC522::Uid::size

Definition at line 1253 of file RadioFrequencyIdentificationMFRC522.h.

4.32.2.3 unsigned char RadioFrequencyIdentificationMFRC522::Uid::uid[10]

Definition at line 1255 of file RadioFrequencyIdentificationMFRC522.h.

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

- RadioFrequencyIdentificationMFRC522.h

4.33 RadioFrequencyIdentificationMFRC522::VERSIONbits Union Reference

```
#include <RadioFrequencyIdentificationMFRC522.h>
```

Public Attributes

- struct {
    unsigned char VERSION:4
    unsigned char CHIPTYPE:4
  };
- unsigned char value

4.33.1 Detailed Description

VERSION register.

Shows the MFRC522 software version.

Definition at line 1237 of file RadioFrequencyIdentificationMFRC522.h.

4.33.2 Member Data Documentation

4.33.2.1 struct { ... }

4.33.2.2 unsigned char RadioFrequencyIdentificationMFRC522::VERSIONbits::CHIPTYPE

Definition at line 1245 of file RadioFrequencyIdentificationMFRC522.h.

4.33.2.3 unsigned char RadioFrequencyIdentificationMFRC522::VERSIONbits::value

Definition at line 1247 of file RadioFrequencyIdentificationMFRC522.h.

4.33.2.4 unsigned char RadioFrequencyIdentificationMFRC522::VERSIONbits::VERSION

Definition at line 1242 of file RadioFrequencyIdentificationMFRC522.h.

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

- RadioFrequencyIdentificationMFRC522.h

4.34 RadioFrequencyIdentificationMFRC522::WATER_LEVELbits Union Reference

```
#include <RadioFrequencyIdentificationMFRC522.h>
```

Public Attributes

- struct {
    unsigned char WATER_LEVEL:7
    unsigned char:1
  };
- unsigned char value

4.34.1 Detailed Description

WATER_LEVEL register.

Defines the level for FIFO under- and overflow warning.

Definition at line 637 of file RadioFrequencyIdentificationMFRC522.h.

4.34.2 Member Data Documentation

4.34.2.1 struct { ... }

4.34.2.2 unsigned RadioFrequencyIdentificationMFRC522::WATER_LEVELbits::char

Definition at line 649 of file RadioFrequencyIdentificationMFRC522.h.

4.34.2.3 unsigned char RadioFrequencyIdentificationMFRC522::WATER_LEVELbits::value

Definition at line 651 of file RadioFrequencyIdentificationMFRC522.h.

4.34.2.4 unsigned char RadioFrequencyIdentificationMFRC522::WATER_LEVELbits::WATER_LEVEL

Definition at line 646 of file RadioFrequencyIdentificationMFRC522.h.

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

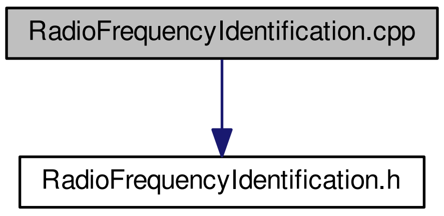
- RadioFrequencyIdentificationMFRC522.h

5 File Documentation

5.1 RadioFrequencyIdentification.cpp File Reference

```
#include "RadioFrequencyIdentification.h"
```

Include dependency graph for RadioFrequencyIdentification.cpp:

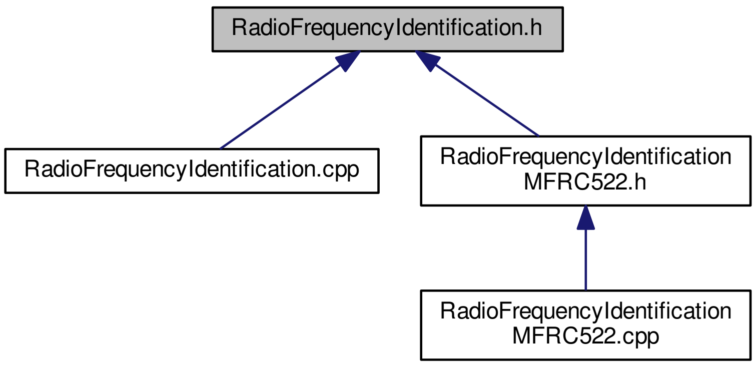

RadioFrequencyIdentification.cpp
RadioFrequencyIdentification.h

5.2 RadioFrequencyIdentification.cpp

```
#include "RadioFrequencyIdentification.h"
```

5.3 RadioFrequencyIdentification.h File Reference

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


RadioFrequencyIdentification.h
RadioFrequencyIdentification.cpp
RadioFrequencyIdentification MFRC522.h
RadioFrequencyIdentification MFRC522.cpp

Classes

- class RadioFrequencyIdentification

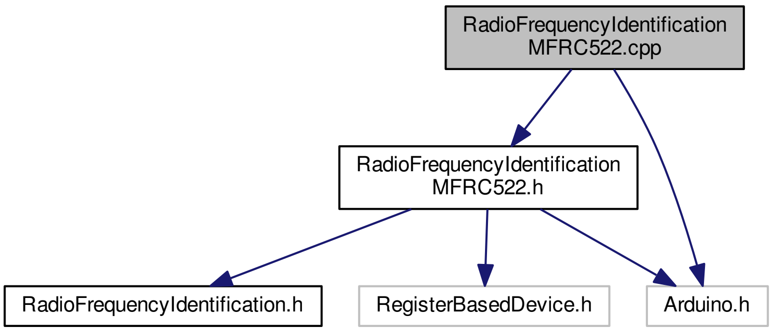
5.4 RadioFrequencyIdentification.h

```

#ifndef __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_H__
#define __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_H__ 1

class RadioFrequencyIdentification {

};

#endif // __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_H__
```

5.5 RadioFrequencyIdentificationMFRC522.cpp File Reference

```
#include "RadioFrequencyIdentificationMFRC522.h"
#include <Arduino.h>
```

Include dependency graph for RadioFrequencyIdentificationMFRC522.cpp:


RadioFrequencyIdentification MFRC522.cpp
RadioFrequencyIdentification MFRC522.h
RadioFrequencyIdentification.h
RegisterBasedDevice.h
Arduino.h

5.6 RadioFrequencyIdentificationMFRC522.cpp

```
#include "RadioFrequencyIdentificationMFRC522.h"
#include <Arduino.h>

RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522
(
        RegisterBasedDevice *device, unsigned char resetPin)
        : RadioFrequencyIdentification(), device(device), resetPin(resetPin) {
    pinMode(resetPin, OUTPUT);
    digitalWrite(resetPin, LOW);
}

void RadioFrequencyIdentificationMFRC522::initialize() {
    if (digitalRead(resetPin) == LOW) {
        digitalWrite(resetPin, HIGH);
        delay(50);
    } else {
        softReset();
    }
    clearRegisterBits(AUTO_TEST, AUTO_TEST_ENABLE);

//    // TAuto=1; f(Timer) = 6.78MHz/TPreScaler
//    device->writeRegister(T_MODE, 0x8d);
//
//    // TModeReg[3..0] + TPrescalerReg
//    device->writeRegister(T_PRESCALER_LOW, 0x3e);
//    device->writeRegister(T_RELOAD_LOW, 0x1e);
//    device->writeRegister(T_RELOAD_HIGH, 0);

// 100% ASK
    writeRegister(TX_ASK, 0x40);

    // CRC Initial value 0x6363  ???
    writeRegister(MODE, 0x3d);

    // ClearBitMask(STATUS2, 0x08);       //MFCrypto1On=0
    // Write_AddicoreRFID(RxSelReg, 0x86);       //RxWait = RxSelReg[5..0]
    // Write_AddicoreRFID(RFCfgReg, 0x7f);           //RxGain = 48dB

    // Open the antenna
    setAntennaOn();
}

void RadioFrequencyIdentificationMFRC522::softReset() {
    writeRegister(COMMAND, SOFT_RESET);
}

void RadioFrequencyIdentificationMFRC522::setAntennaOn() {
    setRegisterBits(TX_CONTROL, TX_CONTROL_TX_RF_EN);
}

void RadioFrequencyIdentificationMFRC522::setAntennaOff()
 {
    clearRegisterBits(TX_CONTROL, TX_CONTROL_TX_RF_EN);
```

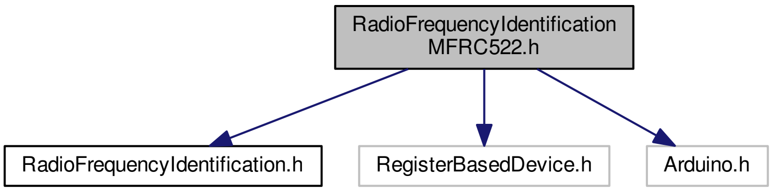
```
}

int RadioFrequencyIdentificationMFRC522::readRegisterBlock(unsigned char reg,
        unsigned char *buf, unsigned char len) {

    // MSB == 1 is for reading. LSB is not used in address.
    return device->readRegisterBlock(((reg << 1) & 0x7e) | 0x80, buf, len);
}

unsigned char RadioFrequencyIdentificationMFRC522::writeRegisterBlock(unsigned char reg,
        unsigned char *buf, unsigned char len) {

    // MSB == 0 is for writing. LSB is not used in address.
    return device->writeRegisterBlock((reg << 1) & 0x7e, buf, len);
}

void RadioFrequencyIdentificationMFRC522::configureTimer(unsigned int prescaler,
        unsigned int reload, bool autoStart, bool autoRestart) {
    T_MODEbits timerMode;
    timerMode.value = readRegister(T_MODE);
    timerMode.T_PRESCALER_HI = (prescaler >> 8) & 0x0f;
    timerMode.T_AUTO = autoStart;
    timerMode.T_GATED = (unsigned char) 0;
    timerMode.T_AUTO_RESTART = autoRestart;
    writeRegister(T_MODE, timerMode.value);
    writeRegister(T_PRESCALER_LOW, prescaler & 0xff);
    writeRegister(T_RELOAD_HIGH, (reload >> 8) & 0xff);
    writeRegister(T_RELOAD_LOW, reload & 0xff);
}

void RadioFrequencyIdentificationMFRC522::startTimer() {
    setRegisterBits(CONTROL, CONTROL_T_START_NOW);
}

void RadioFrequencyIdentificationMFRC522::stopTimer() {
    setRegisterBits(CONTROL, CONTROL_T_STOP_NOW);
}

void RadioFrequencyIdentificationMFRC522::enableInterrupt(Interrupt interrupt) {
    setRegisterBits(MFR522_INT_TO_EN_REG(interrupt), MFR522_INT_TO_EN_MASK(interrupt));
}

void RadioFrequencyIdentificationMFRC522::disableInterrupt(Interrupt interrupt) {
    clearRegisterBits(MFR522_INT_TO_EN_REG(interrupt), MFR522_INT_TO_EN_MASK(interrupt));
}

void RadioFrequencyIdentificationMFRC522::clearInterrupt(Interrupt interrupt) {

    // 0x7f: first bit 0 indicates that the marked bits in the register are cleared
    configureRegisterBits(MFR522_INT_TO_IRQ_REG(interrupt),
            (MFR522_INT_TO_IRQ_MASK(interrupt)) | 0x80, 0x7f);
}

void RadioFrequencyIdentificationMFRC522::flushQueue() {
    setRegisterBits(FIFO_LEVEL, FIFO_LEVEL_FLUSH_BUFFER);
}

void RadioFrequencyIdentificationMFRC522::setWaterLevel(unsigned char level) {
    writeRegister(WATER_LEVEL, WATER_LEVEL_WATER_LEVEL & level);
}

int RadioFrequencyIdentificationMFRC522::generateRandomId(unsigned char buf[10]) {

    // Stop any active command.
    sendCommand(IDLE);

    // Clear all seven interrupt request bits
    clearInterrupt(COM_ALL_IRQ);

    // FlushBuffer = 1, FIFO initialization
    flushQueue();

    // Send command
    sendCommand(GENERATE_RANDOM_ID);

    // Wait for command to complete.
    waitForRegisterBits(COM_IRQ, COM_IRQ_IDLE_IRQ);

    // FlushBuffer = 1, FIFO initialization
    flushQueue();

    // Transfers 25 bytes from the internal buffer to the FIFO buffer.
    sendCommand(MEM);

    // Wait for command to complete.
    waitForRegisterBits(COM_IRQ, COM_IRQ_IDLE_IRQ);
    sendCommand(IDLE);
    return readRegisterBlock(FIFO_DATA, buf, 10);
}

int RadioFrequencyIdentificationMFRC522::communicate(
Command command, unsigned char *output,
        unsigned char *input, unsigned char outputLen, bool checkCRC) {

    int len = 0;
    COM_IRQbits irq;

    // Stop any active command.
    sendCommand(IDLE);

    // Clear all seven interrupt request bits
    clearInterrupt(COM_ALL_IRQ);

    // FlushBuffer = 1, FIFO initialization
    flushQueue();

    // Write sendData to the FIFO
    writeRegisterBlock(FIFO_DATA, output, outputLen);

    // Execute the command
    sendCommand(command);

    if (command == TRANSCEIVE) {

        // StartSend=1, transmission of data starts
        setRegisterBits(BIT_FRAMING, BIT_FRAMING_START_SEND);
    }

    // Wait for the command to complete.
    // If timer was configured and T_AUTO flag is active in T_MODE register,
    // timer will start automatically after all data is transmitted.
    // See: configureTimer method
    do {
        irq.value = readRegister(COM_IRQ);

        // Timer interrupt - nothing received
        if (irq.TIMER_IRQ) {
            lastError = TIMEOUT_ERROR;
            return -1;
        }
    } while (!irq.IDLE_IRQ && !irq.RX_IRQ);

    // Stop now if any errors except collisions were detected.
    // ErrorReg[7..0] bits are: WrErr TempErr reserved BufferOvfl CollErr CRCErr ParityErr ProtocolErr
    ERRORbits error;
    error.value = readRegister(ERROR);
    if (error.BUFFER_OVFL || error.PARITY_ERR || error.
PROTOCOL_ERR || error.COLL_ERR) {
        lastError = COMMUNICATION_ERROR;
        return -1;
    }

    // If the caller wants data back, get it from the MFRC522.
    if (input != NULL) {

        // Number of bytes in the FIFO
        int inpulLen = readRegister(FIFO_LEVEL);

        // Get received data from FIFO
        len = readRegisterBlock(FIFO_DATA, input, inpulLen);

        // Perform CRC_A validation if requested.
        if (len > 0 && checkCRC) {

            // Verify CRC_A - do our own calculation and store the control in controlBuffer.
            unsigned int expectedCRC = calculateCRC(input, inpulLen - 2);
            unsigned int returnedCRC = input[inpulLen - 1];
            returnedCRC <<= 8;
            returnedCRC |= input[inpulLen - 2] & 0xff;
            if (expectedCRC != returnedCRC) {
                lastError = CRC_ERROR;
                return -1;
            }
        }
    }
    return len;
}

unsigned int RadioFrequencyIdentificationMFRC522::calculateCRC
 (unsigned char *buff,
        unsigned char len) {

    unsigned int crc = 0;

    // Stop any active command.
    sendCommand(IDLE);

    // Clear all seven interrupt request bits
    clearInterrupt(DIV_ALL_IRQ);

    // FlushBuffer = 1, FIFO initialization
    flushQueue();

    // Write sendData to the FIFO
    writeRegisterBlock(FIFO_DATA, buff, len);

    // Start the calculation
    sendCommand(CALC_CRC);

    // Wait for the CRC calculation to complete.
    waitForRegisterBits(DIV_IRQ, DIV_IRQ_CRC_IRQ);

    // Stop calculating CRC for new content in the FIFO.
    sendCommand(IDLE);

    crc = readRegister(CRC_RESULT_HIDH);
    crc <<= 8;
    crc |= readRegister(CRC_RESULT_LOW) & 0xff;
    return crc;
}

bool RadioFrequencyIdentificationMFRC522::waitForRegisterBits
 (unsigned char reg,
        unsigned char mask, unsigned long timeout) {
    unsigned char v;
    unsigned long start = millis();
    do {
        v = readRegister(reg);
    } while (!(v & mask) && start + timeout > millis());
    return (v & mask) > 0;
}

bool RadioFrequencyIdentificationMFRC522::performSelfTest
 () {
    unsigned char *firmwareReference;
    unsigned char buffer[64] = { 0 };
    writeRegister(AUTO_TEST, 0x00);
    softReset();
    flushQueue();
    writeRegisterBlock(FIFO_DATA, buffer, 25);
    sendCommand(MEM);
    writeRegister(AUTO_TEST, AUTO_TEST_ENABLE);
    writeRegister(FIFO_DATA, 0x00);
    sendCommand(CALC_CRC);
    waitForRegisterBits(DIV_IRQ, DIV_IRQ_CRC_IRQ, 100);
    readRegisterBlock(FIFO_DATA, buffer, 64);
    switch (getVersion()) {
    case CLONE:
        firmwareReference = (unsigned char *) FM17522_FIRMWARE_REFERENCE;
        break;
    case V0_0:
        firmwareReference = (unsigned char *) MFRC522_FIRMWARE_REFERENCE_V0_0;
        break;
    case V1_0:
        firmwareReference = (unsigned char *) MFRC522_FIRMWARE_REFERENCE_V1_0;
        break;
    case V2_0:
        firmwareReference = (unsigned char *) MFRC522_FIRMWARE_REFERENCE_V2_0;
        break;
    default:
        return false;
    }
    for (unsigned char i = 0; i < 64; i++) {
        if (buffer[i] != pgm_read_byte(&(firmwareReference[i]))) {
            return false;
        }
    }
    return true;
}

RadioFrequencyIdentificationMFRC522::Version
```

```
RadioFrequencyIdentificationMFRC522::getVersion() {
    return (Version) readRegister(VERSION);
}
```

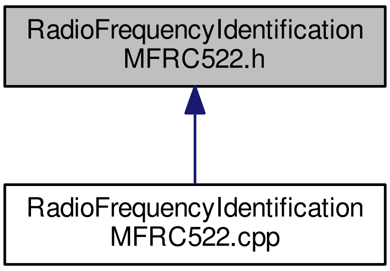
5.7 RadioFrequencyIdentificationMFRC522.h File Reference

```
#include <RadioFrequencyIdentification.h>
#include <RegisterBasedDevice.h>
#include <Arduino.h>
```

Include dependency graph for RadioFrequencyIdentificationMFRC522.h:


RadioFrequencyIdentification MFRC522.h
RadioFrequencyIdentification.h
RegisterBasedDevice.h
Arduino.h

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


RadioFrequencyIdentification MFRC522.h
RadioFrequencyIdentification MFRC522.cpp

Classes

- class RadioFrequencyIdentificationMFRC522
- union RadioFrequencyIdentificationMFRC522::COMMANDbits
- union RadioFrequencyIdentificationMFRC522::COM_I_ENbits
- union RadioFrequencyIdentificationMFRC522::DIV_I_ENbits
- union RadioFrequencyIdentificationMFRC522::COM_IRQbits
- union RadioFrequencyIdentificationMFRC522::DIV_IRQbits
- union RadioFrequencyIdentificationMFRC522::ERRORbits
- union RadioFrequencyIdentificationMFRC522::STATUS1bits
- union RadioFrequencyIdentificationMFRC522::STATUS2bits

- union RadioFrequencyIdentificationMFRC522::FIFO_LEVELbits
- union RadioFrequencyIdentificationMFRC522::WATER_LEVELbits
- union RadioFrequencyIdentificationMFRC522::CONTROLbits
- union RadioFrequencyIdentificationMFRC522::BIT_FRAMINGbits
- union RadioFrequencyIdentificationMFRC522::COLLbits
- union RadioFrequencyIdentificationMFRC522::MODEbits
- union RadioFrequencyIdentificationMFRC522::TX_MODEbits
- union RadioFrequencyIdentificationMFRC522::RX_MODEbits
- union RadioFrequencyIdentificationMFRC522::TX_CONTROLbits
- union RadioFrequencyIdentificationMFRC522::TX_ASKbits
- union RadioFrequencyIdentificationMFRC522::TX_SELbits
- union RadioFrequencyIdentificationMFRC522::RX_SELbits
- union RadioFrequencyIdentificationMFRC522::RX_THRESHOLDbits
- union RadioFrequencyIdentificationMFRC522::DEMODbits
- union RadioFrequencyIdentificationMFRC522::MF_TXbits
- union RadioFrequencyIdentificationMFRC522::MF_RXbits
- union RadioFrequencyIdentificationMFRC522::SERIAL_SPEEDbits
- union RadioFrequencyIdentificationMFRC522::RF_CFGbits
- union RadioFrequencyIdentificationMFRC522::GS_Nbits
- union RadioFrequencyIdentificationMFRC522::CW_GS_Pbits
- union RadioFrequencyIdentificationMFRC522::MOD_GS_Pbits
- union RadioFrequencyIdentificationMFRC522::T_MODEbits
- union RadioFrequencyIdentificationMFRC522::VERSIONbits
- struct RadioFrequencyIdentificationMFRC522::Uid

Macros

- #define MFRC522_DEFAULT_TIMEOUT 100
- #define MFR522_INT_TO_EN_REG(i) (i > COM_ALL_IRQ) ? DIV_I_EN : COM_I_EN
- #define MFR522_INT_TO_EN_MASK(i) (i > COM_ALL_IRQ) ? (i >> 8) & DIV_I_EN_INTERRUPT_EN : i & COM_I_EN_INTERRUPT_EN
- #define MFR522_INT_TO_IRQ_REG(i) (i > COM_ALL_IRQ) ? DIV_IRQ : COM_IRQ
- #define MFR522_INT_TO_IRQ_MASK(i) (i > COM_ALL_IRQ) ? (i >> 8) & DIV_IRQ_ALL_IRQ : i & COM↩
_IRQ_ALL_IRQ

Variables

- const unsigned char MFRC522_FIRMWARE_REFERENCE_V0_0[ ] PROGMEM

5.7.1 Macro Definition Documentation

5.7.1.1 #define MFR522_INT_TO_EN_MASK(*i*) (i > COM_ALL_IRQ) ? (i >> 8) & DIV_I_EN_INTERRUPT_EN : i & COM_I_EN_INTERRUPT_EN

Definition at line 16 of file RadioFrequencyIdentificationMFRC522.h.

5.7.1.2 #define MFR522_INT_TO_EN_REG(*i*) (i > COM_ALL_IRQ) ? DIV_I_EN : COM_I_EN

Definition at line 15 of file RadioFrequencyIdentificationMFRC522.h.

5.7.1.3 #define MFR522_INT_TO_IRQ_MASK(*i*) (i $>$ COM_ALL_IRQ) ? (i $>>$ 8) & DIV_IRQ_ALL_IRQ : i & COM_IRQ_ALL_IRQ

Definition at line 19 of file RadioFrequencyIdentificationMFRC522.h.

5.7.1.4 #define MFR522_INT_TO_IRQ_REG(*i*) (i $>$ COM_ALL_IRQ) ? DIV_IRQ : COM_IRQ

Definition at line 18 of file RadioFrequencyIdentificationMFRC522.h.

5.7.1.5 #define MFRC522_DEFAULT_TIMEOUT 100

Arduino - Radio Frequency Identification MFRC522.

Author

Dalmir da Silva `dalmirdasilva@gmail.com`

Definition at line 13 of file RadioFrequencyIdentificationMFRC522.h.

5.7.2 Variable Documentation

5.7.2.1 const byte FM17522_FIRMWARE_REFERENCE [ ] PROGMEM

Initial value:

```
= { 0x00, 0x87, 0x98, 0x0f,
        0x49, 0xff, 0x07, 0x19, 0xbf, 0x22, 0x30, 0x49, 0x59, 0x63, 0xad, 0xca, 0x7f, 0xe3,
        0x4e, 0x03, 0x5c, 0x4e, 0x49, 0x50, 0x47, 0x9a, 0x37, 0x61, 0xe7, 0xe2, 0xc6, 0x2e,
        0x75, 0x5a, 0xed, 0x04, 0x3d, 0x02, 0x4b, 0x78, 0x32, 0xff, 0x58, 0x3b, 0x7c, 0xe9,
        0x00, 0x94, 0xb4, 0x4a, 0x59, 0x5b, 0xfd, 0xc9, 0x29, 0xdf, 0x35, 0x96, 0x98, 0x9e,
        0x4f, 0x30, 0x32, 0x8d }
```

Definition at line 23 of file RadioFrequencyIdentificationMFRC522.h.

5.8 RadioFrequencyIdentificationMFRC522.h

```
00001
00006 #ifndef __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_MFRC522_H__
00007 #define __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_MFRC522_H__ 1
00008
00009 #include <RadioFrequencyIdentification.h>
00010 #include <RegisterBasedDevice.h>
00011 #include <Arduino.h>
00012
00013 #define MFRC522_DEFAULT_TIMEOUT 100
00014
00015 #define MFR522_INT_TO_EN_REG(i)      (i > COM_ALL_IRQ) ? DIV_I_EN : COM_I_EN
00016 #define MFR522_INT_TO_EN_MASK(i)     (i > COM_ALL_IRQ) ? (i >> 8) & DIV_I_EN_INTERRUPT_EN : i &
       COM_I_EN_INTERRUPT_EN
00017
00018 #define MFR522_INT_TO_IRQ_REG(i)     (i > COM_ALL_IRQ) ? DIV_IRQ : COM_IRQ
00019 #define MFR522_INT_TO_IRQ_MASK(i)    (i > COM_ALL_IRQ) ? (i >> 8) & DIV_IRQ_ALL_IRQ : i & COM_IRQ_ALL_IRQ
00020
00021 // Version 0.0 (0x90)
00022 // Philips Semiconductors; Preliminary Specification Revision 2.0 - 01 August 2005; 16.1 self-test
00023 const unsigned char MFRC522_FIRMWARE_REFERENCE_V0_0[] PROGMEM = { 0x00, 0x87, 0x98, 0x0f,
00024         0x49, 0xff, 0x07, 0x19, 0xbf, 0x22, 0x30, 0x49, 0x59, 0x63, 0xad, 0xca, 0x7f, 0xe3,
00025         0x4e, 0x03, 0x5c, 0x4e, 0x49, 0x50, 0x47, 0x9a, 0x37, 0x61, 0xe7, 0xe2, 0xc6, 0x2e,
00026         0x75, 0x5a, 0xed, 0x04, 0x3d, 0x02, 0x4b, 0x78, 0x32, 0xff, 0x58, 0x3b, 0x7c, 0xe9,
00027         0x00, 0x94, 0xb4, 0x4a, 0x59, 0x5b, 0xfd, 0xc9, 0x29, 0xdf, 0x35, 0x96, 0x98, 0x9e,
00028         0x4f, 0x30, 0x32, 0x8d };
00029
```

```
// Version 1.0 (0x91)
// NXP Semiconductors; Rev. 3.8 - 17 September 2014; 16.1.1 self-test
const byte MFRC522_FIRMWARE_REFERENCE_V1_0[] PROGMEM = { 0x00, 0xc6, 0x37, 0xd5, 0x32, 0xb7,
        0x57, 0x5c, 0xc2, 0xd8, 0x7c, 0x4d, 0xd9, 0x70, 0xc7, 0x73, 0x10, 0xe6, 0xd2, 0xaa,
        0x5e, 0xa1, 0x3e, 0x5a, 0x14, 0xaf, 0x30, 0x61, 0xc9, 0x70, 0xdb, 0x2e, 0x64, 0x22,
        0x72, 0xb5, 0xbd, 0x65, 0xf4, 0xec, 0x22, 0xbc, 0xd3, 0x72, 0x35, 0xcd, 0xaa, 0x41,
        0x1f, 0xa7, 0xf3, 0x53, 0x14, 0xde, 0x7e, 0x02, 0xd9, 0x0f, 0xb5, 0x5e, 0x25, 0x1d,
        0x29, 0x79 };

// Version 2.0 (0x92)
// NXP Semiconductors; Rev. 3.8 - 17 September 2014; 16.1.1 self-test
const byte MFRC522_FIRMWARE_REFERENCE_V2_0[] PROGMEM = { 0x00, 0xeb, 0x66, 0xba, 0x57, 0xbf,
        0x23, 0x95, 0xd0, 0xe3, 0x0d, 0x3d, 0x27, 0x89, 0x5c, 0xde, 0x9d, 0x3b, 0xa7, 0x00,
        0x21, 0x5b, 0x89, 0x82, 0x51, 0x3a, 0xeb, 0x02, 0x0c, 0xa5, 0x00, 0x49, 0x7c, 0x84,
        0x4d, 0xb3, 0xcc, 0xd2, 0x1b, 0x81, 0x5d, 0x48, 0x76, 0xd5, 0x71, 0x61, 0x21, 0xa9,
        0x86, 0x96, 0x83, 0x38, 0xcf, 0x9d, 0x5b, 0x6d, 0xdc, 0x15, 0xba, 0x3e, 0x7d, 0x95,
        0x3b, 0x2f };

// Clone
// Fudan Semiconductor FM17522 (0x88)
const byte FM17522_FIRMWARE_REFERENCE[] PROGMEM = { 0x00, 0xd6, 0x78, 0x8c, 0xe2, 0xaa,
        0x0c, 0x18, 0x2a, 0xb8, 0x7a, 0x7f, 0xd3, 0x6a, 0xcf, 0x0b, 0xb1, 0x37, 0x63, 0x4b,
        0x69, 0xae, 0x91, 0xc7, 0xc3, 0x97, 0xae, 0x77, 0xf4, 0x37, 0xd7, 0x9b, 0x7c, 0xf5,
        0x3c, 0x11, 0x8f, 0x15, 0xc3, 0xd7, 0xc1, 0x5b, 0x00, 0x2a, 0xd0, 0x75, 0xde, 0x9e,
        0x51, 0x64, 0xab, 0x3e, 0xe9, 0x15, 0xb5, 0xab, 0x56, 0x9a, 0x98, 0x82, 0x26, 0xea,
        0x2a, 0x62 };

class RadioFrequencyIdentificationMFRC522: public
RadioFrequencyIdentification,
        public RegisterBasedDevice {

    RegisterBasedDevice *device;

    unsigned char resetPin;

    unsigned char lastError;

public:

    enum Register {

        // Starts and stops command execution
        COMMAND = 0x01,

        // Enable and disable interrupt request control bits
        COM_I_EN = 0x02,

        // Enable and disable interrupt request control bits
        DIV_I_EN = 0x03,

        // Interrupt request bits
        COM_IRQ = 0x04,

        // Interrupt request bits Table 31 on page 40
        DIV_IRQ = 0X05,

        // Error bits showing the error status of the last command
        ERROR = 0X06,

        // Communication status bits
        STATUS1 = 0x07,

        // Receiver and transmitter status bits
        STATUS2 = 0x08,

        // Input and output of 64 byte FIFO buffer
        FIFO_DATA = 0X09,

        // Number of bytes stored in the FIFO buffer
        FIFO_LEVEL = 0x0a,

        // Level for FIFO underflow and overflow warning
        WATER_LEVEL = 0x0b,

        // Miscellaneous control registers
        CONTROL = 0x0c,

        // Adjustments for bit-oriented frames
        BIT_FRAMING = 0x0d,

        // Bit position of the first bit-collision detected on the RF
        COLL = 0x0e,

        // Defines general modes for transmitting and receiving
        MODE = 0x11,

        // Defines transmission data rate and framing
        TX_MODE = 0x12,

        // Defines reception data rate and framing
        RX_MODE = 0x13,

        // Controls the logical behavior of the antenna driver pins TX1 and TX2
        TX_CONTROL = 0x14,

        // Controls the setting of the transmission modulation
        TX_ASK = 0x15,

        // Selects the internal sources for the antenna driver
        TX_SEL = 0x16,

        // Selects internal receiver settings
        RX_SEL = 0x17,

        // Selects thresholds for the bit decoder
        RX_THRESHOLD = 0x18,

        // Defines demodulator settings
        DEMOD = 0x19,

        // Some MIFARE communication transmit parameters
        MF_TX = 0x1c,

        // Controls some MIFARE communication receive parameters
        MF_RX = 0x1d,

        // Selects the speed of the serial UART interface
        SERIAL_SPEED = 0x1f,

        // Shows the MSB and LSB values of the CRC calculation (HIGH)
        CRC_RESULT_HIDH = 0x21,

        // Shows the MSB and LSB values of the CRC calculation (LOW)
        CRC_RESULT_LOW = 0x22,

        // Controls the ModWidth setting
        MOD_WIDTH = 0x24,

        // Configures the receiver gain
        RFC_FG = 0x26,

        // Selects the conductance of the antenna driver pins TX1 and TX2 for modulation
        GS_N = 0x27,

        // The conductance of the p-driver output during periods of no modulation
        CW_GS_P = 0x28,

        // Defines the conductance of the p-driver output during periods of modulation
        MOD_GS_P = 0x29,

        // Defines settings for the internal timer
        T_MODE = 0x2a,

        // Defines settings for the internal timer
        T_PRESCALER_LOW = 0x2b,

        // Defines the 16-bit timer reload value (HIGH)
        T_RELOAD_HIGH = 0x2c,

        // Defines the 16-bit timer reload value (LOW)
        T_RELOAD_LOW = 0x2d,

        // Shows the 16-bit timer value (HIGH)
        T_COUNTER_VAL_HIGH = 0x2e,

        // Shows the 16-bit timer value (LOW)
        T_COUNTER_VAL_LOW = 0x2f,

        // Test signal configuration
        TEST_SEL1 = 0x31,

        // Test signal configuration and PRBS control
        TEST_SEL2 = 0x32,

        // Enables pin output driver on pins D1 to D7
        TEST_PIN_EN = 0x33,

        // Defines the values for D1 to D7 when it is used as an I/O bus
        TEST_PIN_VALUE = 0x34,

        // Shows the status of the internal test bus
        TEST_BUS = 0x35,

        // Controls the digital self test
        AUTO_TEST = 0x36,

        // Shows the software version
        VERSION = 0x37,

        // Controls the pins AUX1 and AUX2
        ANALOG_TEST = 0x38,

        // Defines the test value for TestDAC1
        TEST_DAC1 = 0x39,

        // Defines the test value for TestDAC2
        TEST_DAC2 = 0x3a,

        // Shows the value of ADC I and Q channels
        TEST_ADC = 0x3b
    };

    enum Command {

        // no action, cancels current command execution
        IDLE = 0x00,

        // stores 25 bytes into the internal buffer
        MEM = 0x01,

        // generates a 10-byte random ID number
        GENERATE_RANDOM_ID = 0x02,

        // activates the CRC calculation or performs a self test
        CALC_CRC = 0x03,

        // Transmit data
        TRANSMIT = 0x04,

        // no command change, can be used to modify the CommandReg register bits without affecting the
 command, for example, the PowerDown bit
        NO_CMD_CHANGE = 0x07,

        // activates the receiver circuits (receive data)
        RECEIVE = 0x08,

        // transmits data from FIFO buffer to antenna and automatically activates the receiver after
 transmission (transmit and receive data)
        TRANSCEIVE = 0x0c,

        // performs the MIFARE standard authentication as a reader (authentication)
        MF_AUTHENT = 0x0e,

        // resets the MFRC522
        SOFT_RESET = 0x0F
    };

    enum Error {
        NO_ERROR = 0x00,
        GENERAL_ERROR = 0x01,
        TIMEOUT_ERROR = 0x02,
        COMMUNICATION_ERROR = 0x03,
        CRC_ERROR = 0x04
    };

    enum Mask {
        TX_CONTROL_TX1_RF_EN = 0x01,
        TX_CONTROL_TX2_RF_EN = 0x02,
        TX_CONTROL_TX_RF_EN = TX_CONTROL_TX1_RF_EN |
TX_CONTROL_TX2_RF_EN,
        CONTROL_T_STOP_NOW = 0x80,
        CONTROL_T_START_NOW = 0x40,
        COM_I_EN_INTERRUPT_EN = 0x7f,
        COM_IRQ_TIMER_IRQ = 0x01,
        COM_IRQ_ERR_IRQ = 0x02,
        COM_IRQ_LO_ALERT_IRQ = 0x04,
        COM_IRQ_HI_ALERT_IRQ = 0x08,
        COM_IRQ_IDLE_IRQ = 0x10,
        COM_IRQ_RX_IRQ = 0x20,
        COM_IRQ_TX_IRQ = 0x40,
        COM_IRQ_ALL_IRQ = 0x7f,
        COM_IRQ_SET1 = 0x80,
        DIV_I_EN_CRC_I_EN = 0x04,
        DIV_I_EN_MFIN_ACT_I_EN = 0x10,
        DIV_I_EN_INTERRUPT_EN = DIV_I_EN_CRC_I_EN |
DIV_I_EN_MFIN_ACT_I_EN,
        DIV_IRQ_CRC_IRQ = 0x04,
        DIV_IRQ_MFIN_ACT_IRQ = 0x10,
        DIV_IRQ_ALL_IRQ = DIV_IRQ_CRC_IRQ |
DIV_IRQ_MFIN_ACT_IRQ,
        DIV_IRQ_SET2 = 0x80,
        FIFO_LEVEL_FLUSH_BUFFER = 0x80,
        FIFO_LEVEL_FIFO_LEVEL = 0x7f,
        WATER_LEVEL_WATER_LEVEL = 0x3f,
        BIT_FRAMING_START_SEND = 0x80,
        AUTO_TEST_ENABLE = 0x09
    };

    enum Interrupt
        : unsigned int {
            NONE_IRQ = 0x0000,
        COM_TIMER_IRQ = 0x0001,
        COM_ERR_IRQ = 0x0002,
        COM_LO_ALERT_IRQ = 0x0004,
        COM_HI_ALERT_IRQ = 0x0008,
        COM_IDLE_IRQ = 0x0010,
        COM_RX_IRQ = 0x0020,
        COM_TX_IRQ = 0x0040,
        COM_ALL_IRQ = 0x007f,
        DIV_CRC_IRQ = 0x0400,
        DIV_MFIN_ACT_IRQ = 0x1000,
        DIV_ALL_IRQ = DIV_CRC_IRQ | DIV_MFIN_ACT_IRQ
    };

    union COMMANDbits {

        struct {

            // Activates a command based on the Command value;
            // reading this register shows which command is executed
            unsigned char COMMAND :4;

            // 1: Soft power-down mode entered
            // 0: when the MFRC522 is ready
            // Remark: The PowerDown bit cannot be set when the SoftReset command is activated
            unsigned char POWER_DOWN :1;

            // Analog part of the receiver is switched off
            unsigned char RCV_OFF :1;

            // Reserved
            unsigned char :2;
        };
        unsigned char value;
    };

    union COM_I_ENbits {

        struct {

            // Allows the timer interrupt request (TimerIRq bit) to be propagated to pin IRQ
            unsigned char TIMER_I_EN :1;

            // Allows the error interrupt request (ErrIRq bit) to be propagated to pin IRQ
            unsigned char ERR_I_EN :1;

            // Allows the low alert interrupt request (LoAlertIRq bit) to be propagated to pin IRQ
            unsigned char LO_ALERT_I_EN :1;

            // Allows the high alert interrupt request (HiAlertIRq bit) to be propagated to pin IRQ
            unsigned char HI_ALERT_I_EN :1;

            // Allows the idle interrupt request (IdleIRq bit) to be propagated to pin IRQ
            unsigned char IDLE_I_EN :1;

            // Allows the receiver interrupt request (RxIRq bit) to be propagated to pin IRQ
            unsigned char RX_I_EN :1;

            // Allows the transmitter interrupt request (TxIRq bit) to be propagated to pin IRQ
            unsigned char TX_I_EN :1;

            // 1: Signal on pin IRQ is inverted with respect to the Status1Reg register's IRq bit
            // 0: signal on pin IRQ is equal to the IRq bit; in combination with the DivIEnReg register's
            // IRqPushPull bit, the default value of logic 1 ensures that the output level on pin IRQ is
3-state
            unsigned char I_RQ_INV :1;
        };
        unsigned char value;
    };

    union DIV_I_ENbits {

        struct {

            // Reserved
            unsigned char :2;

            // Allows the CRC interrupt request, indicated by the DivIrqReg register's CRCIRq bit, to be
propagated to pin IRQ
            unsigned char CRC_I_EN :1;

            // Reserved
            unsigned char :1;

            // Allows the MFIN active interrupt request to be propagated to pin IRQ
            unsigned char MFIN_ACT_I_EN :1;

            // Reserved
            unsigned char :2;

            // 1: pin IRQ is a standard CMOS output pin
            // 0: pin IRQ is an open-drain output pin
            unsigned char IRQ_PUSH_PULL :1;
        };
        unsigned char value;
    };

    union COM_IRQbits {

        struct {

            // The timer decrements the timer value in register TCounterValReg to zero
            unsigned char TIMER_IRQ :1;

            // Any error bit in the ErrorReg register is set
            unsigned char ERR_IRQ :1;

            // Status1Reg register's LoAlert bit is set in opposition to the LoAlert bit,
            // the LoAlertIRq bit stores this event and can only be reset as indicated by
            // the Set1 bit in this register
            unsigned char LO_ALERT_IRQ :1;

            // Status1Reg register's HiAlert bit is set in opposition to the HiAlert bit,
            // the HiAlertIRq bit stores this event and can only be reset as indicated by
            // the Set1 bit in this register
            unsigned char HI_ALERT_IRQ :1;

            // If a command terminates, for example, when the CommandReg changes
            // its value from any command to the Idle command if an unknown command is started,
            // the CommandReg register Command[3:0] value changes to the idle state and the IdleIRq bit is
set
            // The microcontroller starting the Idle command does not set the IdleIRq bit
            unsigned char IDLE_IRQ :1;

            // Receiver has detected the end of a valid data stream
            // if the RxModeReg register's RxNoErr bit is set to logic 1, the RxIRq bit is
            // only set to logic 1 when data bytes are available in the FIFO
            unsigned char RX_IRQ :1;

            // Set immediately after the last bit of the transmitted data was sent out
            unsigned char TX_IRQ :1;

            // 1: indicates that the marked bits in the ComIrqReg register are set
            // 0: indicates that the marked bits in the ComIrqReg register are cleared
            unsigned char SET1 :1;
        };
        unsigned char value;
    };

    union DIV_IRQbits {

        struct {

            // Reserved
            unsigned char :2;

            // The CalcCRC command is active and all data is processed
            unsigned char CRC_IRQ :1;

            // Reserved
            unsigned char :1;

            // MFIN is active this interrupt is set when either a rising or falling signal edge is
detected.
            unsigned char MFIN_ACT_IRQ :1;

            // Reserved
            unsigned char :2;

            // 1: indicates that the marked bits in the DivIrqReg register are set
            // 0: indicates that the marked bits in the DivIrqReg register are cleared
            unsigned char SET2 :1;
        };
        unsigned char value;
    };

    union ERRORbits {

        struct {

            // Set to logic 1 if the SOF is incorrect automatically cleared during receiver start-up phase
            // bit is only valid for 106 kBd during the MFAuthent command, the ProtocolErr bit is set to
            // logic 1 if the number of bytes received in one data stream is incorrect
            unsigned char PROTOCOL_ERR :1;

            // Parity check failed. Automatically cleared during receiver start-up phase
            // only valid for ISO/IEC 14443 A/MIFARE communication at 106 kBd
            unsigned char PARITY_ERR :1;

            // The RxModeReg register's RxCRCEn bit is set and the CRC calculation fails
            // automatically cleared to logic 0 during receiver start-up phase
            unsigned char CRC_ERR :1;

            // A bit-collision is detected cleared automatically at receiver start-up phase
            // only valid during the bitwise anticollision at 106 kBd always set to logic 0 during communication
            // protocols at 212 kBd, 424 kBd and 848 kBd
            unsigned char COLL_ERR :1;

            // The host or a MFRC522's internal state machine (e.g. receiver) tries to
            // write data to the FIFO buffer even though it is already full
            unsigned char BUFFER_OVFL :1;

            // Reserved
            unsigned char :1;

            // Internal temperature sensor detects overheating, in which case the antenna drivers are automatically switched off
            unsigned char TEMP_ERR :1;

            // Data is written into the FIFO buffer by the host during the MFAuthent command or if data is written
            // into the FIFO buffer by the host during the time between sending the last bit on the RF interface and
            // receiving the last bit on the RF interface
            unsigned char WR_ERR :1;
        };
        unsigned char value;
    };

    union STATUS1bits {

        struct {

            // The number of bytes stored in the FIFO buffer corresponds to equation:
            // HiAlert = FIFOLength <= WaterLevel
            // example:
            // FIFO length = 4, WaterLevel = 4 > LoAlert = 1
            // FIFO length = 5, WaterLevel = 4 > LoAlert = 0
            unsigned char LO_ALERT :1;

            // The number of bytes stored in the FIFO buffer corresponds to equation:
            // HiAlert = (64 - FIFOLength) <= WaterLevel
            // example:
            // FIFO length = 60, WaterLevel = 4 > HiAlert = 1
            // FIFO length = 59, WaterLevel = 4 > HiAlert = 0
            unsigned char HI_ALERT :1;

            // MFRC522's timer unit is running, i.e. the timer will decrement the TCounterValReg register with the next timer clock
            // Remark: in gated mode, the TRunning bit is set to logic 1 when the timer is enabled by TModeReg register's TGated[1:0] bits;
            // this bit is not influenced by the gated signal
            unsigned char T_RUNNING :1;

            // Indicates if any interrupt source requests attention with respect to the setting of the interrupt enable bits:
            // see the ComIEnReg and DivIEnReg registers
            unsigned char IRQ :1;

            // The CRC calculation has finished only valid for the CRC coprocessor calculation using the CalcCRC command
            unsigned char CRC_READY :1;

            // The CRC result is zero
            // for data transmission and reception, the CRCOk bit is undefined: use the
            // ErrorReg register's CRCErr bit indicates the status of the CRC coprocessor, during calculation the value
            // changes to logic 0, when the calculation is done correctly the value changes to logic 1
            unsigned char CRC_OK :1;

            // Reserved
            unsigned char :1;
        };
        unsigned char value;
    };

    union STATUS2bits {

        struct {

            // Shows the state of the transmitter and receiver state machines:
            //  000: idle
            //  001: wait for the BitFramingReg register’s StartSend bit
            //  010: TxWait: wait until RF field is present if the TModeReg register’s
            // TxWaitRF bit is set to logic 1 the minimum time for TxWait is defined by the TxWaitReg register
            //  011: transmitting
            //  100: RxWait: wait until RF field is present if the TModeReg register’s TxWaitRF bit is set to logic 1
            // the minimum time for RxWait is defined by the RxWaitReg register
            //  101: wait for data
            //  110: receiving
            unsigned char MODEM_STATE :3;

            // Indicates that the MIFARE Crypto1 unit is switched on and therefore all data communication with the card is encrypted
            // can only be set to logic 1 by a successful execution of the MFAuthent command only valid in Read/Write mode for
            // MIFARE standard cards this bit is cleared by software
            unsigned char MF_CRYPTO1_ON :1;

            // Reserved
            unsigned char :2;

            // I2C-bus input filter settings:
            // 1: the I2C-bus input filter is set to the High-speed mode independent of the I2C-bus protocol
            // 0: the I2C-bus input filter is set to the I2C-bus protocol used
            unsigned char I2C_FORCE_HS :1;

            // Clears the temperature error if the temperature is below the alarm limit of 125C
            unsigned char TEMP_SENS_CLEAR :1;
        };
        unsigned char value;
    };

    union FIFO_LEVELbits {

        struct {

            // Indicates the number of bytes stored in the FIFO buffer writing to the FIFODataReg
            // register increments and reading decrements the FIFOLevel value
            unsigned char FIFO_LEVEL :7;

            // Immediately clears the internal FIFO buffer’s read and write pointer and ErrorReg
            // register’s BufferOvfl bit reading this bit always returns 0
            unsigned char FLUSH_BUFFER :1;
        };
        unsigned char value;
    };

    union WATER_LEVELbits {

        struct {

            // Defines a warning level to indicate a FIFO buffer overflow or underflow:
            // Status1Reg register’s HiAlert bit is set to logic 1 if the remaining
            // number of bytes in the FIFO buffer space is equal to, or less than the defined number of WaterLevel bytes
            // Status1Reg register’s LoAlert bit is set to logic 1 if equal to, or less than the WaterLevel bytes in the FIFO buffer
            // Remark: to calculate values for HiAlert and LoAlert see Section 9.3.1.8 on page 42.
            unsigned char WATER_LEVEL :7;

            // Reserved
            unsigned char :1;
        };
        unsigned char value;
    };

    union CONTROLbits {

        struct {

            // Indicates the number of valid bits in the last received byte if this value is 000b, the whole byte is valid
            unsigned char RX_LAST_BITS :3;

            // Reserved
            unsigned char :2;

            // Timer starts immediately
            // reading this bit always returns it to logic 0
            unsigned char T_START_NOW :1;

            // Timer stops immediately
            // reading this bit always returns it to logic0
            unsigned char T_STOP_NOW :1;
        };
        unsigned char value;
    };

    union BIT_FRAMINGbits {

        struct {

            // Used for transmission of bit oriented frames: defines the number of bits of the last byte that will be transmitted
            // 000b indicates that all bits of the last byte will be transmitted
            unsigned char TX_LAST_BITS :3;

            // Reserved
            unsigned char :1;

            // used for reception of bit-oriented frames: defines the bit position for the first bit received to be stored in the FIFO buffer
            // example:
            // 0: LSB of the received bit is stored at bit position 0, the second received bit is stored at bit position 1
            // 1: LSB of the received bit is stored at bit position 1, the second received bit is stored at bit position 2
            // 7: LSB of the received bit is stored at bit position 7, the second received bit is stored in the next byte that follows at bit position 0
            // These bits are only to be used for bitwise anticollision at 106 kBd, for all other modes they are set to 0
            unsigned char RX_ALIGN :3;

            // Starts the transmission of data only valid in combination with the Transceive command
            unsigned char START_SEND :1;
        };
        unsigned char value;
    };

    union COLLbits {

        struct {

            // Shows the bit position of the first detected collision in a received frame only data bits are interpreted
            // example:
            //  00h: indicates a bit-collision in the 32nd bit
            //  01h: indicates a bit-collision in the 1st bit
            //  08h: indicates a bit-collision in the 8th bit
            // These bits will only be interpreted if the CollPosNotValid bit is set to logic 0
            unsigned char COLL_POS :5;

            // No collision detected or the position of the collision is out of the range of CollPos[4:0]
            unsigned char COLL_POS_NOT_VALID :1;

            // Reserved
            unsigned char :1;

            // All received bits will be cleared after a collision only used during bitwise anticollision at 106 kBd, otherwise it is set to logic 1
            unsigned char VALUES_AFTER_COLL :1;
        };
        unsigned char value;
    };

    union MODEbits {

        struct {

            // defines the preset value for the CRC coprocessor for the CalcCRC command
            // Remark: during any communication, the preset values are selected automatically according to
            // the definition of bits in the RxModeReg and TxModeReg registers
            //  00: 0000h
            //  01: 6363h
            //  10: A671h
            //  11: FFFFh
            unsigned char CRC_PRESET :2;

            // Reserved
            unsigned char :1;

            // Defines the polarity of pin MFIN
            // Remark: the internal envelope signal is encoded active LOW, changing this bit generates a MFinActIRq event
            // 1: polarity of pin MFIN is active HIGH
            // 0: polarity of pin MFIN is active LOW
            unsigned char POL_M_FIN :1;

            // Reserved
            unsigned char :1;

            // Transmitter can only be started if an RF field is generated
            unsigned char TX_WAIT_RF :1;

            // Reserved
            unsigned char :1;

            // CRC coprocessor calculates the CRC with MSB first in the CRCResultReg register the values for the
            // CRCResultMSB[7:0] bits and the CRCResultLSB[7:0] bits are bit reversed
            // Remark: during RF communication this bit is ignored
            unsigned char MSB_FIRST :1;
        };
        unsigned char value;
    };

    union TX_MODEbits {

        struct {

            // Reserved
            unsigned char :3;

            // Modulation of transmitted data is inverted
            unsigned char INV_MOD :1;

            // Defines the bit rate during data transmission the MFRC522 handles transfer speeds up to 848 kBd
            //  000: 106 kBd
            //  001: 212 kBd
            //  010: 424 kBd
            //  011: 848 kBd
            unsigned char TX_SPEED :3;

            // Enables CRC generation during data transmission
            // Remark: can only be set to logic 0 at 106 kBd
            unsigned char TX_CRC_EN :1;
        };
        unsigned char value;
    };

    union RX_MODEbits {

        struct {

            // Reserved
            unsigned char :2;

            // 0: receiver is deactivated after receiving a data frame
            // 1: able to receive more than one data frame only valid for data rates above 106 kBd in order to handle
            // the polling command after setting this bit the Receive and Transceive commands will not terminate automatically.
            // Multiple reception can only be deactivated by writing any command (except the Receive command) to the CommandReg
            // register, or by the host clearing the bit if set to logic 1, an error byte is added to the FIFO buffer at the
            // end of a received data stream which is a copy of the ErrorReg register value. For the MFRC522 version 2.0 the CRC status is
            // reflected in the signal CRCOk, which indicates the actual status of the CRC coprocessor. For the MFRC522 version 1.0 the CRC
            // status is reflected in the signal CRCErr.
            unsigned char RX_MULTIPLE :1;

            // An invalid received data stream (less than 4 bits received) will be ignored and the receiver remains active
            unsigned char RX_NO_ERR :1;

            // Defines the bit rate while receiving data the MFRC522 handles transfer speeds up to 848 kBd
            //  000: 106 kBd
            //  001: 212 kBd
            //  010: 424 kBd
            //  011: 848 kBd
            unsigned char RX_SPEED :3;

            // Enables the CRC calculation during reception
            // Remark: can only be set to logic 0 at 106 kBd
            unsigned char RX_CRC_EN :1;
        };
        unsigned char value;
    };

    union TX_CONTROLbits {

        struct {

            // Output signal on pin TX1 delivers the 13.56 MHz energy carrier modulated by the transmission
data
            unsigned char TX1_RF_EN :1;

            // Output signal on pin TX2 delivers the 13.56 MHz energy carrier modulated by the transmission
data
            unsigned char TX2_RF_EN :1;

            // Reserved
            unsigned char :1;

            // 1: output signal on pin TX2 continuously delivers the unmodulated 13.56 MHz energy carrier
            // 0: Tx2CW bit is enabled to modulate the 13.56 MHz energy carrier
            unsigned char TX2_CW :1;

            // Output signal on pin TX1 inverted when driver TX1 is disabled
            unsigned char INV_TX1_RF_OFF :1;

            // Output signal on pin TX2 inverted when driver TX2 is disabled
            unsigned char INV_TX2_RF_OFF :1;

            // Output signal on pin TX1 inverted when driver TX1 is enabled
            unsigned char INV_TX1_RF_ON :1;

            // Output signal on pin TX2 inverted when driver TX2 is enabled
            unsigned char INV_TX2_RF_ON :1;
        };

        struct {

            // Output signal on pin TX1 delivers the 13.56 MHz energy carrier modulated by the transmission
data
            // Output signal on pin TX2 delivers the 13.56 MHz energy carrier modulated by the transmission
data
            unsigned char TX_RF_EN :2;

            // Reserved
            unsigned char :2;

            // Output signal on pin TX1 inverted when driver TX1 is disabled
            // Output signal on pin TX2 inverted when driver TX2 is disabled
            unsigned char INV_TX_RF_OFF :2;

            // Output signal on pin TX1 inverted when driver TX1 is enabled
            // Output signal on pin TX2 inverted when driver TX2 is enabled
            unsigned char INV_TX_RF_ON :2;
        };
        unsigned char value;
    };

    union TX_ASKbits {

        struct {

            // Reserved
            unsigned char :6;

            // Forces a 100% ASK modulation independent of the ModGsPReg register setting
            unsigned char FORCE_100_ASK :1;

            // Reserved
            unsigned char :1;
        };
        unsigned char value;
    };

    union TX_SELbits {

        struct {

            // Selects the input for pin MFOUT
            //  0000: 3-state
            //  0001: LOW
            //  0010: HIGH
            //  0011: test bus signal as defined by the TestSel1Reg register's TstBusBitSel[2:0] value
            //  0100: modulation signal (envelope) from the internal encoder, Miller pulse encoded
            //  0101: serial data stream to be transmitted, data stream before Miller encoder
            //  0110: reserved
            //  0111: serial data stream received, data stream after Manchester decoder
            //  1000: to 1111 reserved
            unsigned char MF_OUT_SEL :4;

            // Selects the input of drivers TX1 and TX2
            //  00: 3-state; in soft power-down the drivers are only in 3-state mode if the DriverSel[1:0] value is set to 3-state mode
            //  01: modulation signal (envelope) from the internal encoder, Miller pulse encoded
            //  10: modulation signal (envelope) from pin MFIN
            //  11: HIGH; the HIGH level depends on the setting of bits InvTx1RFOn/InvTx1RFOff and InvTx2RFOn/InvTx2RFOff
            unsigned char :2;

            // Reserved
            unsigned char :2;
        };
        unsigned char value;
    };

    union RX_SELbits {

        struct {

            // After data transmission the activation of the receiver is delayed for RxWait bit-clocks, during this 'frame guard time
            // any signal on pin RX is ignored this parameter is ignored by the Receive command all other commands, such as Transceive,
            // MFAuthent use this parameter the counter starts immediately after the external RF field is switched on
            unsigned char RX_WAIT :6;

            // Selects the input of the contactless UART
            //  00: constant LOW
            //  01: Manchester with subcarrier from pin MFIN
            //  10: modulated signal from the internal analog module, default
            //  11: NRZ coding without subcarrier from pin MFIN which is only valid for transfer speeds above 106 kBd
            unsigned char UART_SEL :2;
        };
        unsigned char value;
    };

    union RX_THRESHOLDbits {

        struct {

            // defines the minimum signal strength at the decoder input that must be reached by the weaker half-bit of the
            // Manchester encoded signal to generate a bit-collision relative to the amplitude of the stronger half-bit
            unsigned char COLL_LEVEL :3;

            // Reserved
            unsigned char :1;

            // Defines the minimum signal strength at the decoder input that will be accepted if the signal strength is below this level it is not evaluated
            unsigned char MIN_LEVEL :4;
        };
        unsigned char value;
    };

    union DEMODbits {

        struct {

            // Changes the time-constant of the internal PLL during burst
            unsigned char TAU_SYNC :2;

            // Changes the time-constant of the internal PLL during data reception
            // Remark: if set to 00b the PLL is frozen during data reception
            unsigned char TAU_RCV :2;

            // If set to logic 0 the following formula is used to calculate the timer frequency of the prescaler:
            // F_timer = 13.56 MHz / (2*TPreScaler+1).
            // If set to logic 1 the following formula is used to calculate the timer frequency of the prescaler:
            // F_timer = 13.56 MHz / (2*TPreScaler+2)
            unsigned char T_PRESCAL_EVEN :1;

            // If AddIQ[1:0] are set to X0b, the reception is fixed to I channel
            // If AddIQ[1:0] are set to X1b, the reception is fixed to Q channel
            unsigned char FIX_IQ :1;

            // Defines the use of I and Q channel during reception
            // Remark: the FixIQ bit must be set to logic 0 to enable the following settings:
            //  00: selects the stronger channel
            //  01: selects the stronger channel and freezes the selected channel during communication
            unsigned char ADD_IQ :2;
        };
        unsigned char value;
    };

    union MF_TXbits {

        struct {

            // Defines the additional response time 7 bits are added to the value of the register bit by
default
            unsigned char TX_WAIT :2;

            // Reserved
            unsigned char :6;
        };
        unsigned char value;
    };

    union MF_RXbits {

        struct {

            // Reserved
            unsigned char :4;

            // Generation of the parity bit for transmission and the parity check for receiving is switched
off
            // the received parity bit is handled like a data bit
            unsigned char PARITY_DISABLE :1;

            // Reserved
            unsigned char :3;
        };
        unsigned char value;
    };

    union SERIAL_SPEEDbits {

        struct {

            // Factor BR_T1 adjusts the transfer speed
            unsigned char BR_T1 :5;

            // Factor BR_T0 adjusts the transfer speed
            unsigned char BR_T0 :3;
        };
        unsigned char value;
    };

    union RF_CFGbits {

        struct {

            // Reserved
            unsigned char :4;

            // Defines the receiver's signal voltage gain factor:
            //  000: 18 dB
            //  001: 23 dB
            //  010: 18 dB
            //  011: 23 dB
            //  100: 33 dB
            //  101: 38 dB
            //  110: 43 dB
            //  111: 48 dB
            unsigned char RX_GAIN :3;

            // Reserved
            unsigned char :1;
        };
        unsigned char value;
    };

    union GS_Nbits {

        struct {

            // Defines the conductance of the output n-driver during periods without modulation which can
be used to regulate the modulation index
            // Remark: the conductance value is binary weighted during soft Power-down mode the highest bit
is forced to logic 1
            // value is only used if driver TX1 or TX2 is switched on
            unsigned char MOD_GS_N :4;

            // defines the conductance of the output n-driver during periods without modulation which can
```

```
be used to regulate the output power and
            // subsequently current consumption and operating distance
            // Remark: the conductance value is binary-weighted during soft Power-down mode the highest bit
is forced to logic 1
            // value is only used if driver TX1 or TX2 is switched on
            unsigned char CW_GS_N :4;
        };
        unsigned char value;
    };

    union CW_GS_Pbits {

        struct {

            // defines the conductance of the p-driver output which can be used to regulate the output
power and subsequently current consumption and operating distance
            // Remark: the conductance value is binary weighted during soft Power-down mode the highest bit
is forced to logic 1
            unsigned char CW_GS_P :6;

            // Reserved
            unsigned char :2;
        };
        unsigned char value;
    };

    union MOD_GS_Pbits {

        struct {

            // Defines the conductance of the p-driver output during modulation which can be used to
regulate the modulation index
            // Remark: the conductance value is binary weighted during soft Power-down mode the highest bit
is forced to logic 1
            // if the TxASKReg register's Force100ASK bit is set to logic 1 the value of ModGsP has no
effect
            unsigned char MOD_GS_P :6;

            // Reserved
            unsigned char :2;
        };
        unsigned char value;
    };

    union T_MODEbits {

        struct {

            // Defines the higher 4 bits of the TPrescaler value
            // The following formula is used to calculate the timer frequency if the DemodReg register's
TPrescalEven bit in Demot Regis set to logic 0:
            // F_timer = 13.56 MHz / (2*TPreScaler+1)
            // Where TPreScaler = [TPrescaler_Hi:TPrescaler_Lo] (TPrescaler value on 12 bits) (Default
TPrescalEven bit is logic 0)
            // The following formula is used to calculate the timer frequency if the DemodReg register's
TPrescalEven bit is set to logic 1:
            // F_timer = 13.56 MHz / (2*TPreScaler+2).
            unsigned char T_PRESCALER_HI :4;

            // 1: timer automatically restarts its count-down from the 16-bit timer reload value instead of
counting down to zero
            // 0: timer decrements to 0 and the ComIrqReg register's TimerIRq bit is set to logic 1
            unsigned char T_AUTO_RESTART :1;

            // Internal timer is running in gated mode
            // Remark: in gated mode, the Status1Reg register's TRunning bit is logic 1 when the timer is
enabled by the
            // TModeReg register's TGated[1:0] bits this bit does not influence the gating signal
            //  00: non-gated mode
            //  01: gated by pin MFIN
            //  10: gated by pin AUX1
            unsigned char T_GATED :2;

            // 1: timer starts automatically at the end of the transmission in all communication modes at
all speeds
            // if the RxModeReg register's RxMultiple bit is not set, the timer stops immediately after
receiving the 5th bit (1 start bit, 4 data bits)
            // if the RxMultiple bit is set to logic 1 the timer never stops, in which case the timer can
be stopped by setting the ControlReg register's
            // TStopNow bit to logic 1
            // 0: indicates that the timer is not influenced by the protocol
            unsigned char T_AUTO :1;
        };
        unsigned char value;
    };

    union VERSIONbits {
        struct {

            // '1' stands for MFRC522 version 1.0 and '2' stands for MFRC522 version 2.0.
            unsigned char VERSION :4;

            // '9' stands for MFRC522
            unsigned char CHIPTYPE :4;
        };
        unsigned char value;
    };

    struct Uid {

        // Number of bytes in the UID. 4, 7 or 10.
        unsigned char size;

        unsigned char uid[10];

        // The SAK (Select acknowledge) byte returned from the PICC after successful selection.
        unsigned char sak;
    };

    enum Version {
        CLONE = 0x88,
        V0_0 = 0x90,
        V1_0 = 0x91,
        V2_0 = 0x92
    };

    RadioFrequencyIdentificationMFRC522(RegisterBasedDevice *device,
            unsigned char resetPin);

    void initialize();

    void sendCommand(Command command) {
        writeRegister(COMMAND, (unsigned char) command);
    }

    unsigned char getLastError() {
        return lastError;
    }

    void clearLastError() {
        lastError = NO_ERROR;
    }

    void softReset();

    void setAntennaOn();

    void setAntennaOff();

    void configureTimer(unsigned int prescaler, unsigned int reload, bool autoStart,
            bool autoRestart);

    void startTimer();

    void stopTimer();

    void enableInterrupt(Interrupt interrupt);

    void disableInterrupt(Interrupt interrupt);

    void clearInterrupt(Interrupt interrupt);

    void flushQueue();

    void setWaterLevel(unsigned char level);

    int generateRandomId(unsigned char buf[10]);

    int communicate(Command command, unsigned char *output, unsigned char *input,
            unsigned char outputLen, bool checkCRC);

    unsigned int calculateCRC(unsigned char *buff, unsigned char len);

    bool waitForRegisterBits(unsigned char reg, unsigned char mask) {
        return waitForRegisterBits(reg, mask,
MFRC522_DEFAULT_TIMEOUT);
    }

    bool waitForRegisterBits(unsigned char reg, unsigned char mask, unsigned long
timeout);

    Version getVersion();

    bool performSelfTest();

    int readRegisterBlock(unsigned char reg, unsigned char *buf, unsigned char len);

    unsigned char writeRegisterBlock(unsigned char reg, unsigned char *buf,
            unsigned char len);
};

#endif // __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_MFRC522_H__
```

Index

ADD_IQ
RadioFrequencyIdentificationMFRC522::DEMO↩
Dbits, 13
ANALOG_TEST
RadioFrequencyIdentificationMFRC522, 29
AUTO_TEST_ENABLE
RadioFrequencyIdentificationMFRC522, 28
AUTO_TEST
RadioFrequencyIdentificationMFRC522, 29

BIT_FRAMING_START_SEND
RadioFrequencyIdentificationMFRC522, 28
BIT_FRAMING
RadioFrequencyIdentificationMFRC522, 29
BR_T0
RadioFrequencyIdentificationMFRC522::SERIA↩
L_SPEEDbits, 38
BR_T1
RadioFrequencyIdentificationMFRC522::SERIA↩
L_SPEEDbits, 38
BUFFER_OVFL
RadioFrequencyIdentificationMFRC522::ERRO↩
Rbits, 17

CALC_CRC
RadioFrequencyIdentificationMFRC522, 27
CHIPTYPE
RadioFrequencyIdentificationMFRC522::VERSI↩
ONbits, 48
CLONE
RadioFrequencyIdentificationMFRC522, 30
COLL_ERR
RadioFrequencyIdentificationMFRC522::ERRO↩
Rbits, 17
COLL_LEVEL
RadioFrequencyIdentificationMFRC522::RX_TH↩
RESHOLDbits, 37
COLL_POS_NOT_VALID
RadioFrequencyIdentificationMFRC522::COLLbits,
7
COLL_POS
RadioFrequencyIdentificationMFRC522::COLLbits,
7
COLL
RadioFrequencyIdentificationMFRC522, 29
COM_ALL_IRQ
RadioFrequencyIdentificationMFRC522, 27
COM_ERR_IRQ
RadioFrequencyIdentificationMFRC522, 27
COM_HI_ALERT_IRQ
RadioFrequencyIdentificationMFRC522, 27
COM_I_EN_INTERRUPT_EN
RadioFrequencyIdentificationMFRC522, 28
COM_I_EN
RadioFrequencyIdentificationMFRC522, 28
COM_IDLE_IRQ
RadioFrequencyIdentificationMFRC522, 27
COM_IRQ_ALL_IRQ
RadioFrequencyIdentificationMFRC522, 28
COM_IRQ_ERR_IRQ
RadioFrequencyIdentificationMFRC522, 28
COM_IRQ_HI_ALERT_IRQ
RadioFrequencyIdentificationMFRC522, 28
COM_IRQ_IDLE_IRQ
RadioFrequencyIdentificationMFRC522, 28
COM_IRQ_LO_ALERT_IRQ
RadioFrequencyIdentificationMFRC522, 28
COM_IRQ_RX_IRQ
RadioFrequencyIdentificationMFRC522, 28
COM_IRQ_SET1
RadioFrequencyIdentificationMFRC522, 28
COM_IRQ_TIMER_IRQ
RadioFrequencyIdentificationMFRC522, 28
COM_IRQ_TX_IRQ
RadioFrequencyIdentificationMFRC522, 28
COM_IRQ
RadioFrequencyIdentificationMFRC522, 28
COM_LO_ALERT_IRQ
RadioFrequencyIdentificationMFRC522, 27
COM_RX_IRQ
RadioFrequencyIdentificationMFRC522, 27
COM_TIMER_IRQ
RadioFrequencyIdentificationMFRC522, 27
COM_TX_IRQ
RadioFrequencyIdentificationMFRC522, 27
COMMAND
RadioFrequencyIdentificationMFRC522, 28
RadioFrequencyIdentificationMFRC522::COMM↩
ANDbits, 10
COMMUNICATION_ERROR
RadioFrequencyIdentificationMFRC522, 27
CONTROL_T_START_NOW
RadioFrequencyIdentificationMFRC522, 28
CONTROL_T_STOP_NOW
RadioFrequencyIdentificationMFRC522, 28
CONTROL
RadioFrequencyIdentificationMFRC522, 29
CRC_ERROR
RadioFrequencyIdentificationMFRC522, 27
CRC_ERR
RadioFrequencyIdentificationMFRC522::ERRO↩
Rbits, 17
CRC_I_EN
RadioFrequencyIdentificationMFRC522::DIV_I_↩
ENbits, 14
CRC_IRQ
RadioFrequencyIdentificationMFRC522::DIV_IR↩
Qbits, 15
CRC_OK
RadioFrequencyIdentificationMFRC522::STATU↩
S1bits, 39

CRC_PRESET
RadioFrequencyIdentificationMFRC522::MOD↩Ebits, 22
CRC_READY
RadioFrequencyIdentificationMFRC522::STATU↩S1bits, 39
CRC_RESULT_HIDH
RadioFrequencyIdentificationMFRC522, 29
CRC_RESULT_LOW
RadioFrequencyIdentificationMFRC522, 29
CW_GS_N
RadioFrequencyIdentificationMFRC522::GS_↩Nbits, 19
CW_GS_P
RadioFrequencyIdentificationMFRC522, 29
RadioFrequencyIdentificationMFRC522::CW_G↩S_Pbits, 12
calculateCRC
RadioFrequencyIdentificationMFRC522, 30
char
RadioFrequencyIdentificationMFRC522::BIT_FR↩AMINGbits, 6
RadioFrequencyIdentificationMFRC522::COLLbits, 7
RadioFrequencyIdentificationMFRC522::COMM↩ANDbits, 10
RadioFrequencyIdentificationMFRC522::CONT↩ROLbits, 11
RadioFrequencyIdentificationMFRC522::CW_G↩S_Pbits, 12
RadioFrequencyIdentificationMFRC522::DIV_I_↩ENbits, 14
RadioFrequencyIdentificationMFRC522::DIV_IR↩Qbits, 15
RadioFrequencyIdentificationMFRC522::ERRO↩Rbits, 17
RadioFrequencyIdentificationMFRC522::MF_R↩Xbits, 20
RadioFrequencyIdentificationMFRC522::MF_T↩Xbits, 20
RadioFrequencyIdentificationMFRC522::MOD_↩GS_Pbits, 21
RadioFrequencyIdentificationMFRC522::MOD↩Ebits, 22
RadioFrequencyIdentificationMFRC522::RF_CF↩Gbits, 34
RadioFrequencyIdentificationMFRC522::RX_M↩ODEbits, 35
RadioFrequencyIdentificationMFRC522::RX_TH↩RESHOLDbits, 37
RadioFrequencyIdentificationMFRC522::STATU↩S1bits, 39
RadioFrequencyIdentificationMFRC522::STATU↩S2bits, 41
RadioFrequencyIdentificationMFRC522::TX_AS↩Kbits, 43
RadioFrequencyIdentificationMFRC522::TX_CO↩NTROLbits, 44
RadioFrequencyIdentificationMFRC522::TX_MO↩DEbits, 45
RadioFrequencyIdentificationMFRC522::TX_SE↩Lbits, 46
RadioFrequencyIdentificationMFRC522::WATE↩R_LEVELbits, 49
clearInterrupt
RadioFrequencyIdentificationMFRC522, 30
clearLastError
RadioFrequencyIdentificationMFRC522, 30
Command
RadioFrequencyIdentificationMFRC522, 27
communicate
RadioFrequencyIdentificationMFRC522, 30
configureTimer
RadioFrequencyIdentificationMFRC522, 30

DEMOD
RadioFrequencyIdentificationMFRC522, 29
DIV_ALL_IRQ
RadioFrequencyIdentificationMFRC522, 27
DIV_CRC_IRQ
RadioFrequencyIdentificationMFRC522, 27
DIV_I_EN_CRC_I_EN
RadioFrequencyIdentificationMFRC522, 28
DIV_I_EN_INTERRUPT_EN
RadioFrequencyIdentificationMFRC522, 28
DIV_I_EN_MFIN_ACT_I_EN
RadioFrequencyIdentificationMFRC522, 28
DIV_I_EN
RadioFrequencyIdentificationMFRC522, 28
DIV_IRQ_ALL_IRQ
RadioFrequencyIdentificationMFRC522, 28
DIV_IRQ_CRC_IRQ
RadioFrequencyIdentificationMFRC522, 28
DIV_IRQ_MFIN_ACT_IRQ
RadioFrequencyIdentificationMFRC522, 28
DIV_IRQ_SET2
RadioFrequencyIdentificationMFRC522, 28
DIV_IRQ
RadioFrequencyIdentificationMFRC522, 28
DIV_MFIN_ACT_IRQ
RadioFrequencyIdentificationMFRC522, 27
device
RadioFrequencyIdentificationMFRC522, 34
disableInterrupt
RadioFrequencyIdentificationMFRC522, 31

ERR_I_EN
RadioFrequencyIdentificationMFRC522::COM_I↩_ENbits, 8
ERR_IRQ
RadioFrequencyIdentificationMFRC522::COM_I↩RQbits, 9
ERROR
RadioFrequencyIdentificationMFRC522, 28
enableInterrupt
RadioFrequencyIdentificationMFRC522, 31
Error

RadioFrequencyIdentificationMFRC522, 27

FIFO_DATA
RadioFrequencyIdentificationMFRC522, 28
FIFO_LEVEL_FIFO_LEVEL
RadioFrequencyIdentificationMFRC522, 28
FIFO_LEVEL_FLUSH_BUFFER
RadioFrequencyIdentificationMFRC522, 28
FIFO_LEVEL
RadioFrequencyIdentificationMFRC522, 28
RadioFrequencyIdentificationMFRC522::FIFO_L↩EVELbits, 18
FIX_IQ
RadioFrequencyIdentificationMFRC522::DEMO↩Dbits, 13
FLUSH_BUFFER
RadioFrequencyIdentificationMFRC522::FIFO_L↩EVELbits, 18
FORCE_100_ASK
RadioFrequencyIdentificationMFRC522::TX_AS↩Kbits, 43
flushQueue
RadioFrequencyIdentificationMFRC522, 31

GENERAL_ERROR
RadioFrequencyIdentificationMFRC522, 27
GENERATE_RANDOM_ID
RadioFrequencyIdentificationMFRC522, 27
GS_N
RadioFrequencyIdentificationMFRC522, 29
generateRandomId
RadioFrequencyIdentificationMFRC522, 31
getLastError
RadioFrequencyIdentificationMFRC522, 31
getVersion
RadioFrequencyIdentificationMFRC522, 31

HI_ALERT_I_EN
RadioFrequencyIdentificationMFRC522::COM_I↩_ENbits, 8
HI_ALERT_IRQ
RadioFrequencyIdentificationMFRC522::COM_I↩RQbits, 9
HI_ALERT
RadioFrequencyIdentificationMFRC522::STATU↩S1bits, 39

I2C_FORCE_HS
RadioFrequencyIdentificationMFRC522::STATU↩S2bits, 41
I_RQ_INV
RadioFrequencyIdentificationMFRC522::COM_I↩_ENbits, 8
IDLE_I_EN
RadioFrequencyIdentificationMFRC522::COM_I↩_ENbits, 8
IDLE_IRQ
RadioFrequencyIdentificationMFRC522::COM_I↩RQbits, 9
IDLE
RadioFrequencyIdentificationMFRC522, 27
INV_MOD
RadioFrequencyIdentificationMFRC522::TX_MO↩DEbits, 45
INV_TX1_RF_OFF
RadioFrequencyIdentificationMFRC522::TX_CO↩NTROLbits, 44
INV_TX1_RF_ON
RadioFrequencyIdentificationMFRC522::TX_CO↩NTROLbits, 44
INV_TX2_RF_OFF
RadioFrequencyIdentificationMFRC522::TX_CO↩NTROLbits, 44
INV_TX2_RF_ON
RadioFrequencyIdentificationMFRC522::TX_CO↩NTROLbits, 44
INV_TX_RF_OFF
RadioFrequencyIdentificationMFRC522::TX_CO↩NTROLbits, 44
INV_TX_RF_ON
RadioFrequencyIdentificationMFRC522::TX_CO↩NTROLbits, 44
IRQ_PUSH_PULL
RadioFrequencyIdentificationMFRC522::DIV_I_↩ENbits, 15
IRQ
RadioFrequencyIdentificationMFRC522::STATU↩S1bits, 40
initialize
RadioFrequencyIdentificationMFRC522, 31
Interrupt
RadioFrequencyIdentificationMFRC522, 27

LO_ALERT_I_EN
RadioFrequencyIdentificationMFRC522::COM_I↩_ENbits, 8
LO_ALERT_IRQ
RadioFrequencyIdentificationMFRC522::COM_I↩RQbits, 9
LO_ALERT
RadioFrequencyIdentificationMFRC522::STATU↩S1bits, 40
lastError
RadioFrequencyIdentificationMFRC522, 34

MEM
RadioFrequencyIdentificationMFRC522, 27
MF_AUTHENT
RadioFrequencyIdentificationMFRC522, 27
MF_CRYPTO1_ON
RadioFrequencyIdentificationMFRC522::STATU↩S2bits, 41
MF_OUT_SEL
RadioFrequencyIdentificationMFRC522::TX_SE↩Lbits, 46
MF_RX
RadioFrequencyIdentificationMFRC522, 29
MF_TX

RadioFrequencyIdentificationMFRC522, 29
MFIN_ACT_I_EN
RadioFrequencyIdentificationMFRC522::DIV_I_←ENbits, 15
MFIN_ACT_IRQ
RadioFrequencyIdentificationMFRC522::DIV_IR←Qbits, 16
MFR522_INT_TO_EN_MASK
RadioFrequencyIdentificationMFRC522.h, 56
MFR522_INT_TO_EN_REG
RadioFrequencyIdentificationMFRC522.h, 56
MFR522_INT_TO_IRQ_MASK
RadioFrequencyIdentificationMFRC522.h, 56
MFR522_INT_TO_IRQ_REG
RadioFrequencyIdentificationMFRC522.h, 57
MFRC522_DEFAULT_TIMEOUT
RadioFrequencyIdentificationMFRC522.h, 57
MIN_LEVEL
RadioFrequencyIdentificationMFRC522::RX_TH←RESHOLDbits, 37
MOD_GS_N
RadioFrequencyIdentificationMFRC522::GS_←Nbits, 19
MOD_GS_P
RadioFrequencyIdentificationMFRC522, 29
RadioFrequencyIdentificationMFRC522::MOD_←GS_Pbits, 21
MOD_WIDTH
RadioFrequencyIdentificationMFRC522, 29
MODEM_STATE
RadioFrequencyIdentificationMFRC522::STATU←S2bits, 41
MODE
RadioFrequencyIdentificationMFRC522, 29
MSB_FIRST
RadioFrequencyIdentificationMFRC522::MOD←Ebits, 22
Mask
RadioFrequencyIdentificationMFRC522, 27

NO_CMD_CHANGE
RadioFrequencyIdentificationMFRC522, 27
NO_ERROR
RadioFrequencyIdentificationMFRC522, 27
NONE_IRQ
RadioFrequencyIdentificationMFRC522, 27

PARITY_DISABLE
RadioFrequencyIdentificationMFRC522::MF_R←Xbits, 20
PARITY_ERR
RadioFrequencyIdentificationMFRC522::ERRO←Rbits, 17
POL_M_FIN
RadioFrequencyIdentificationMFRC522::MOD←Ebits, 22
POWER_DOWN
RadioFrequencyIdentificationMFRC522::COMM←ANDbits, 11
PROGMEM
RadioFrequencyIdentificationMFRC522.h, 57
PROTOCOL_ERR
RadioFrequencyIdentificationMFRC522::ERRO←Rbits, 17
performSelfTest
RadioFrequencyIdentificationMFRC522, 31

RCV_OFF
RadioFrequencyIdentificationMFRC522::COMM←ANDbits, 11
RECEIVE
RadioFrequencyIdentificationMFRC522, 27
RFC_FG
RadioFrequencyIdentificationMFRC522, 29
RX_ALIGN
RadioFrequencyIdentificationMFRC522::BIT_FR←AMINGbits, 6
RX_CRC_EN
RadioFrequencyIdentificationMFRC522::RX_M←ODEbits, 35
RX_GAIN
RadioFrequencyIdentificationMFRC522::RF_CF←Gbits, 34
RX_I_EN
RadioFrequencyIdentificationMFRC522::COM_I←_ENbits, 8
RX_IRQ
RadioFrequencyIdentificationMFRC522::COM_I←RQbits, 9
RX_LAST_BITS
RadioFrequencyIdentificationMFRC522::CONT←ROLbits, 11
RX_MODE
RadioFrequencyIdentificationMFRC522, 29
RX_MULTIPLE
RadioFrequencyIdentificationMFRC522::RX_M←ODEbits, 35
RX_NO_ERR
RadioFrequencyIdentificationMFRC522::RX_M←ODEbits, 36
RX_SEL
RadioFrequencyIdentificationMFRC522, 29
RX_SPEED
RadioFrequencyIdentificationMFRC522::RX_M←ODEbits, 36
RX_THRESHOLD
RadioFrequencyIdentificationMFRC522, 29
RX_WAIT
RadioFrequencyIdentificationMFRC522::RX_SE←Lbits, 36
RadioFrequencyIdentification, 23
RadioFrequencyIdentification.cpp, 49, 50
RadioFrequencyIdentification.h, 50
RadioFrequencyIdentificationMFRC522, 24
ANALOG_TEST, 29
AUTO_TEST_ENABLE, 28
AUTO_TEST, 29
BIT_FRAMING_START_SEND, 28

BIT_FRAMING, 29
CALC_CRC, 27
CLONE, 30
COLL, 29
COM_ALL_IRQ, 27
COM_ERR_IRQ, 27
COM_HI_ALERT_IRQ, 27
COM_I_EN_INTERRUPT_EN, 28
COM_I_EN, 28
COM_IDLE_IRQ, 27
COM_IRQ_ALL_IRQ, 28
COM_IRQ_ERR_IRQ, 28
COM_IRQ_HI_ALERT_IRQ, 28
COM_IRQ_IDLE_IRQ, 28
COM_IRQ_LO_ALERT_IRQ, 28
COM_IRQ_RX_IRQ, 28
COM_IRQ_SET1, 28
COM_IRQ_TIMER_IRQ, 28
COM_IRQ_TX_IRQ, 28
COM_IRQ, 28
COM_LO_ALERT_IRQ, 27
COM_RX_IRQ, 27
COM_TIMER_IRQ, 27
COM_TX_IRQ, 27
COMMAND, 28
COMMUNICATION_ERROR, 27
CONTROL_T_START_NOW, 28
CONTROL_T_STOP_NOW, 28
CONTROL, 29
CRC_ERROR, 27
CRC_RESULT_HIDH, 29
CRC_RESULT_LOW, 29
CW_GS_P, 29
calculateCRC, 30
clearInterrupt, 30
clearLastError, 30
Command, 27
communicate, 30
configureTimer, 30
DEMOD, 29
DIV_ALL_IRQ, 27
DIV_CRC_IRQ, 27
DIV_I_EN_CRC_I_EN, 28
DIV_I_EN_INTERRUPT_EN, 28
DIV_I_EN_MFIN_ACT_I_EN, 28
DIV_I_EN, 28
DIV_IRQ_ALL_IRQ, 28
DIV_IRQ_CRC_IRQ, 28
DIV_IRQ_MFIN_ACT_IRQ, 28
DIV_IRQ_SET2, 28
DIV_IRQ, 28
DIV_MFIN_ACT_IRQ, 27
device, 34
disableInterrupt, 31
ERROR, 28
enableInterrupt, 31
Error, 27
FIFO_DATA, 28
FIFO_LEVEL_FIFO_LEVEL, 28
FIFO_LEVEL_FLUSH_BUFFER, 28
FIFO_LEVEL, 28
flushQueue, 31
GENERAL_ERROR, 27
GENERATE_RANDOM_ID, 27
GS_N, 29
generateRandomId, 31
getLastError, 31
getVersion, 31
IDLE, 27
initialize, 31
Interrupt, 27
lastError, 34
MEM, 27
MF_AUTHENT, 27
MF_RX, 29
MF_TX, 29
MOD_GS_P, 29
MOD_WIDTH, 29
MODE, 29
Mask, 27
NO_CMD_CHANGE, 27
NO_ERROR, 27
NONE_IRQ, 27
performSelfTest, 31
RECEIVE, 27
RFC_FG, 29
RX_MODE, 29
RX_SEL, 29
RX_THRESHOLD, 29
RadioFrequencyIdentificationMFRC522, 30
readRegisterBlock, 32
Register, 28
resetPin, 34
SERIAL_SPEED, 29
SOFT_RESET, 27
STATUS1, 28
STATUS2, 28
sendCommand, 32
setAntennaOff, 32
setAntennaOn, 33
setWaterLevel, 33
softReset, 33
startTimer, 33
stopTimer, 33
T_COUNTER_VAL_HIGH, 29
T_COUNTER_VAL_LOW, 29
T_MODE, 29
T_PRESCALER_LOW, 29
T_RELOAD_HIGH, 29
T_RELOAD_LOW, 29
TEST_ADC, 29
TEST_BUS, 29
TEST_DAC1, 29
TEST_DAC2, 29
TEST_PIN_EN, 29
TEST_PIN_VALUE, 29

TEST_SEL1, 29
TEST_SEL2, 29
TIMEOUT_ERROR, 27
TRANSCEIVE, 27
TRANSMIT, 27
TX_ASK, 29
TX_CONTROL_TX1_RF_EN, 28
TX_CONTROL_TX2_RF_EN, 28
TX_CONTROL_TX_RF_EN, 28
TX_CONTROL, 29
TX_MODE, 29
TX_SEL, 29
V0_0, 30
V1_0, 30
V2_0, 30
VERSION, 29
Version, 29
WATER_LEVEL_WATER_LEVEL, 28
WATER_LEVEL, 29
waitForRegisterBits, 33
writeRegisterBlock, 33
RadioFrequencyIdentificationMFRC522.cpp, 50, 51
RadioFrequencyIdentificationMFRC522.h, 55, 57
MFR522_INT_TO_EN_MASK, 56
MFR522_INT_TO_EN_REG, 56
MFR522_INT_TO_IRQ_MASK, 56
MFR522_INT_TO_IRQ_REG, 57
MFRC522_DEFAULT_TIMEOUT, 57
PROGMEM, 57
RadioFrequencyIdentificationMFRC522::BIT_FRAMI↩NGbits, 5
char, 6
RX_ALIGN, 6
START_SEND, 6
TX_LAST_BITS, 6
value, 6
RadioFrequencyIdentificationMFRC522::COLLbits, 6
COLL_POS_NOT_VALID, 7
COLL_POS, 7
char, 7
VALUES_AFTER_COLL, 7
value, 7
RadioFrequencyIdentificationMFRC522::COM_I_E↩Nbits, 7
ERR_I_EN, 8
HI_ALERT_I_EN, 8
I_RQ_INV, 8
IDLE_I_EN, 8
LO_ALERT_I_EN, 8
RX_I_EN, 8
TIMER_I_EN, 8
TX_I_EN, 8
value, 8
RadioFrequencyIdentificationMFRC522::COM_IRQbits, 9
ERR_IRQ, 9
HI_ALERT_IRQ, 9
IDLE_IRQ, 9
LO_ALERT_IRQ, 9
RX_IRQ, 9
SET1, 9
TIMER_IRQ, 10
TX_IRQ, 10
value, 10
RadioFrequencyIdentificationMFRC522::COMMAN↩Dbits, 10
COMMAND, 10
char, 10
POWER_DOWN, 11
RCV_OFF, 11
value, 11
RadioFrequencyIdentificationMFRC522::CONTRO↩Lbits, 11
char, 11
RX_LAST_BITS, 11
T_START_NOW, 12
T_STOP_NOW, 12
value, 12
RadioFrequencyIdentificationMFRC522::CW_GS_↩Pbits, 12
CW_GS_P, 12
char, 12
value, 13
RadioFrequencyIdentificationMFRC522::DEMODbits, 13
ADD_IQ, 13
FIX_IQ, 13
T_PRESCAL_EVEN, 13
TAU_RCV, 14
TAU_SYNC, 14
value, 14
RadioFrequencyIdentificationMFRC522::DIV_I_ENbits, 14
CRC_I_EN, 14
char, 14
IRQ_PUSH_PULL, 15
MFIN_ACT_I_EN, 15
value, 15
RadioFrequencyIdentificationMFRC522::DIV_IRQbits, 15
CRC_IRQ, 15
char, 15
MFIN_ACT_IRQ, 16
SET2, 16
value, 16
RadioFrequencyIdentificationMFRC522::ERRORbits, 16
BUFFER_OVFL, 17
COLL_ERR, 17
CRC_ERR, 17
char, 17
PARITY_ERR, 17
PROTOCOL_ERR, 17
TEMP_ERR, 17
value, 17
WR_ERR, 17

RadioFrequencyIdentificationMFRC522::FIFO_LEVE↩ Lbits, 17
    FIFO_LEVEL, 18
    FLUSH_BUFFER, 18
    value, 18
RadioFrequencyIdentificationMFRC522::GS_Nbits, 18
    CW_GS_N, 19
    MOD_GS_N, 19
    value, 19
RadioFrequencyIdentificationMFRC522::MF_RXbits, 19
    char, 20
    PARITY_DISABLE, 20
    value, 20
RadioFrequencyIdentificationMFRC522::MF_TXbits, 20
    char, 20
    TX_WAIT, 20
    value, 21
RadioFrequencyIdentificationMFRC522::MOD_GS_↩ Pbits, 21
    char, 21
    MOD_GS_P, 21
    value, 21
RadioFrequencyIdentificationMFRC522::MODEbits, 22
    CRC_PRESET, 22
    char, 22
    MSB_FIRST, 22
    POL_M_FIN, 22
    TX_WAIT_RF, 23
    value, 23
RadioFrequencyIdentificationMFRC522::RF_CFGbits, 34
    char, 34
    RX_GAIN, 34
    value, 35
RadioFrequencyIdentificationMFRC522::RX_MOD↩ Ebits, 35
    char, 35
    RX_CRC_EN, 35
    RX_MULTIPLE, 35
    RX_NO_ERR, 36
    RX_SPEED, 36
    value, 36
RadioFrequencyIdentificationMFRC522::RX_SELbits, 36
    RX_WAIT, 36
    UART_SEL, 36
    value, 37
RadioFrequencyIdentificationMFRC522::RX_THRES↩ HOLDbits, 37
    COLL_LEVEL, 37
    char, 37
    MIN_LEVEL, 37
    value, 38
RadioFrequencyIdentificationMFRC522::SERIAL_SP↩ EEDbits, 38
    BR_T0, 38
    BR_T1, 38
    value, 38
RadioFrequencyIdentificationMFRC522::STATUS1bits, 39
    CRC_OK, 39
    CRC_READY, 39
    char, 39
    HI_ALERT, 39
    IRQ, 40
    LO_ALERT, 40
    T_RUNNING, 40
    value, 40
RadioFrequencyIdentificationMFRC522::STATUS2bits, 40
    char, 41
    I2C_FORCE_HS, 41
    MF_CRYPTO1_ON, 41
    MODEM_STATE, 41
    TEMP_SENS_CLEAR, 41
    value, 41
RadioFrequencyIdentificationMFRC522::T_MODEbits, 41
    T_AUTO_RESTART, 42
    T_AUTO, 42
    T_GATED, 42
    T_PRESCALER_HI, 42
    value, 42
RadioFrequencyIdentificationMFRC522::TX_ASKbits, 42
    char, 43
    FORCE_100_ASK, 43
    value, 43
RadioFrequencyIdentificationMFRC522::TX_CONTR↩ OLbits, 43
    char, 44
    INV_TX1_RF_OFF, 44
    INV_TX1_RF_ON, 44
    INV_TX2_RF_OFF, 44
    INV_TX2_RF_ON, 44
    INV_TX_RF_OFF, 44
    INV_TX_RF_ON, 44
    TX1_RF_EN, 44
    TX2_CW, 44
    TX2_RF_EN, 44
    TX_RF_EN, 45
    value, 45
RadioFrequencyIdentificationMFRC522::TX_MODEbits, 45
    char, 45
    INV_MOD, 45
    TX_CRC_EN, 46
    TX_SPEED, 46
    value, 46
RadioFrequencyIdentificationMFRC522::TX_SELbits, 46
    char, 46
    MF_OUT_SEL, 46
    value, 47
RadioFrequencyIdentificationMFRC522::Uid, 47
    sak, 47

size, 47
uid, 47
RadioFrequencyIdentificationMFRC522::VERSIONbits, 48
CHIPTYPE, 48
VERSION, 48
value, 48
RadioFrequencyIdentificationMFRC522::WATER_LE↩VELbits, 48
char, 49
value, 49
WATER_LEVEL, 49
readRegisterBlock
RadioFrequencyIdentificationMFRC522, 32
Register
RadioFrequencyIdentificationMFRC522, 28
resetPin
RadioFrequencyIdentificationMFRC522, 34

SERIAL_SPEED
RadioFrequencyIdentificationMFRC522, 29
SET1
RadioFrequencyIdentificationMFRC522::COM_I↩RQbits, 9
SET2
RadioFrequencyIdentificationMFRC522::DIV_IR↩Qbits, 16
SOFT_RESET
RadioFrequencyIdentificationMFRC522, 27
START_SEND
RadioFrequencyIdentificationMFRC522::BIT_FR↩AMINGbits, 6
STATUS1
RadioFrequencyIdentificationMFRC522, 28
STATUS2
RadioFrequencyIdentificationMFRC522, 28
sak
RadioFrequencyIdentificationMFRC522::Uid, 47
sendCommand
RadioFrequencyIdentificationMFRC522, 32
setAntennaOff
RadioFrequencyIdentificationMFRC522, 32
setAntennaOn
RadioFrequencyIdentificationMFRC522, 33
setWaterLevel
RadioFrequencyIdentificationMFRC522, 33
size
RadioFrequencyIdentificationMFRC522::Uid, 47
softReset
RadioFrequencyIdentificationMFRC522, 33
startTimer
RadioFrequencyIdentificationMFRC522, 33
stopTimer
RadioFrequencyIdentificationMFRC522, 33

T_AUTO_RESTART
RadioFrequencyIdentificationMFRC522::T_MO↩DEbits, 42
T_AUTO
RadioFrequencyIdentificationMFRC522::T_MO↩DEbits, 42
T_COUNTER_VAL_HIGH
RadioFrequencyIdentificationMFRC522, 29
T_COUNTER_VAL_LOW
RadioFrequencyIdentificationMFRC522, 29
T_GATED
RadioFrequencyIdentificationMFRC522::T_MO↩DEbits, 42
T_MODE
RadioFrequencyIdentificationMFRC522, 29
T_PRESCAL_EVEN
RadioFrequencyIdentificationMFRC522::DEMO↩Dbits, 13
T_PRESCALER_HI
RadioFrequencyIdentificationMFRC522::T_MO↩DEbits, 42
T_PRESCALER_LOW
RadioFrequencyIdentificationMFRC522, 29
T_RELOAD_HIGH
RadioFrequencyIdentificationMFRC522, 29
T_RELOAD_LOW
RadioFrequencyIdentificationMFRC522, 29
T_RUNNING
RadioFrequencyIdentificationMFRC522::STATU↩S1bits, 40
T_START_NOW
RadioFrequencyIdentificationMFRC522::CONT↩ROLbits, 12
T_STOP_NOW
RadioFrequencyIdentificationMFRC522::CONT↩ROLbits, 12
TAU_RCV
RadioFrequencyIdentificationMFRC522::DEMO↩Dbits, 14
TAU_SYNC
RadioFrequencyIdentificationMFRC522::DEMO↩Dbits, 14
TEMP_ERR
RadioFrequencyIdentificationMFRC522::ERRO↩Rbits, 17
TEMP_SENS_CLEAR
RadioFrequencyIdentificationMFRC522::STATU↩S2bits, 41
TEST_ADC
RadioFrequencyIdentificationMFRC522, 29
TEST_BUS
RadioFrequencyIdentificationMFRC522, 29
TEST_DAC1
RadioFrequencyIdentificationMFRC522, 29
TEST_DAC2
RadioFrequencyIdentificationMFRC522, 29
TEST_PIN_EN
RadioFrequencyIdentificationMFRC522, 29
TEST_PIN_VALUE
RadioFrequencyIdentificationMFRC522, 29
TEST_SEL1
RadioFrequencyIdentificationMFRC522, 29

TEST_SEL2
RadioFrequencyIdentificationMFRC522, 29
TIMEOUT_ERROR
RadioFrequencyIdentificationMFRC522, 27
TIMER_I_EN
RadioFrequencyIdentificationMFRC522::COM_I↩_ENbits, 8
TIMER_IRQ
RadioFrequencyIdentificationMFRC522::COM_I↩RQbits, 10
TRANSCEIVE
RadioFrequencyIdentificationMFRC522, 27
TRANSMIT
RadioFrequencyIdentificationMFRC522, 27
TX1_RF_EN
RadioFrequencyIdentificationMFRC522::TX_CO↩NTROLbits, 44
TX2_CW
RadioFrequencyIdentificationMFRC522::TX_CO↩NTROLbits, 44
TX2_RF_EN
RadioFrequencyIdentificationMFRC522::TX_CO↩NTROLbits, 44
TX_ASK
RadioFrequencyIdentificationMFRC522, 29
TX_CONTROL_TX1_RF_EN
RadioFrequencyIdentificationMFRC522, 28
TX_CONTROL_TX2_RF_EN
RadioFrequencyIdentificationMFRC522, 28
TX_CONTROL_TX_RF_EN
RadioFrequencyIdentificationMFRC522, 28
TX_CONTROL
RadioFrequencyIdentificationMFRC522, 29
TX_CRC_EN
RadioFrequencyIdentificationMFRC522::TX_MO↩DEbits, 46
TX_I_EN
RadioFrequencyIdentificationMFRC522::COM_I↩_ENbits, 8
TX_IRQ
RadioFrequencyIdentificationMFRC522::COM_I↩RQbits, 10
TX_LAST_BITS
RadioFrequencyIdentificationMFRC522::BIT_FR↩AMINGbits, 6
TX_MODE
RadioFrequencyIdentificationMFRC522, 29
TX_RF_EN
RadioFrequencyIdentificationMFRC522::TX_CO↩NTROLbits, 45
TX_SEL
RadioFrequencyIdentificationMFRC522, 29
TX_SPEED
RadioFrequencyIdentificationMFRC522::TX_MO↩DEbits, 46
TX_WAIT_RF
RadioFrequencyIdentificationMFRC522::MOD↩Ebits, 23
TX_WAIT
RadioFrequencyIdentificationMFRC522::MF_T↩Xbits, 20

UART_SEL
RadioFrequencyIdentificationMFRC522::RX_SE↩Lbits, 36
uid
RadioFrequencyIdentificationMFRC522::Uid, 47

V0_0
RadioFrequencyIdentificationMFRC522, 30
V1_0
RadioFrequencyIdentificationMFRC522, 30
V2_0
RadioFrequencyIdentificationMFRC522, 30
VALUES_AFTER_COLL
RadioFrequencyIdentificationMFRC522::COLLbits, 7
VERSION
RadioFrequencyIdentificationMFRC522, 29
RadioFrequencyIdentificationMFRC522::VERSI↩ONbits, 48
value
RadioFrequencyIdentificationMFRC522::BIT_FR↩AMINGbits, 6
RadioFrequencyIdentificationMFRC522::COLLbits, 7
RadioFrequencyIdentificationMFRC522::COM_I↩_ENbits, 8
RadioFrequencyIdentificationMFRC522::COM_I↩RQbits, 10
RadioFrequencyIdentificationMFRC522::COMM↩ANDbits, 11
RadioFrequencyIdentificationMFRC522::CONT↩ROLbits, 12
RadioFrequencyIdentificationMFRC522::CW_G↩S_Pbits, 13
RadioFrequencyIdentificationMFRC522::DEMO↩Dbits, 14
RadioFrequencyIdentificationMFRC522::DIV_I_↩ENbits, 15
RadioFrequencyIdentificationMFRC522::DIV_IR↩Qbits, 16
RadioFrequencyIdentificationMFRC522::ERRO↩Rbits, 17
RadioFrequencyIdentificationMFRC522::FIFO_L↩EVELbits, 18
RadioFrequencyIdentificationMFRC522::GS_↩Nbits, 19
RadioFrequencyIdentificationMFRC522::MF_R↩Xbits, 20
RadioFrequencyIdentificationMFRC522::MF_T↩Xbits, 21
RadioFrequencyIdentificationMFRC522::MOD_↩GS_Pbits, 21
RadioFrequencyIdentificationMFRC522::MOD↩Ebits, 23

RadioFrequencyIdentificationMFRC522::RF_CF↩
Gbits, 35
RadioFrequencyIdentificationMFRC522::RX_M↩
ODEbits, 36
RadioFrequencyIdentificationMFRC522::RX_SE↩
Lbits, 37
RadioFrequencyIdentificationMFRC522::RX_TH↩
RESHOLDbits, 38
RadioFrequencyIdentificationMFRC522::SERIA↩
L_SPEEDbits, 38
RadioFrequencyIdentificationMFRC522::STATU↩
S1bits, 40
RadioFrequencyIdentificationMFRC522::STATU↩
S2bits, 41
RadioFrequencyIdentificationMFRC522::T_MO↩
DEbits, 42
RadioFrequencyIdentificationMFRC522::TX_AS↩
Kbits, 43
RadioFrequencyIdentificationMFRC522::TX_CO↩
NTROLbits, 45
RadioFrequencyIdentificationMFRC522::TX_MO↩
DEbits, 46
RadioFrequencyIdentificationMFRC522::TX_SE↩
Lbits, 47
RadioFrequencyIdentificationMFRC522::VERSI↩
ONbits, 48
RadioFrequencyIdentificationMFRC522::WATE↩
R_LEVELbits, 49

Version
RadioFrequencyIdentificationMFRC522, 29

WATER_LEVEL_WATER_LEVEL
RadioFrequencyIdentificationMFRC522, 28

WATER_LEVEL
RadioFrequencyIdentificationMFRC522, 29
RadioFrequencyIdentificationMFRC522::WATE↩
R_LEVELbits, 49

WR_ERR
RadioFrequencyIdentificationMFRC522::ERRO↩
Rbits, 17

waitForRegisterBits
RadioFrequencyIdentificationMFRC522, 33

writeRegisterBlock
RadioFrequencyIdentificationMFRC522, 33